Reference Manual

Yahoo! Widgets
Version 4.5

Release History

First Release		February 10, 2003
Second Release		February 12, 2003
Third Release		February 15, 2003
Fourth Release		February 19, 2003
Fifth Release	Version 1.5	July 23, 2003
Sixth Release	Version 1.5.1	September 26, 2003
Seventh Release		October 8, 2003
Eighth Release	Version 1.6.2	June 6, 2004
Ninth Release	Version 1.8	November 8, 2004
Tenth Release	Version 1.8.1	November 24, 2004
Eleventh Release	Version 1.8.3	January 18, 2005
Twelfth Release	Version 2.1	July 23, 2005
Thirteenth Release	Version 2.1.1	August 3, 2005
Fourteenth Release	Version 3.0	December 7, 2005
Fifteenth Release	Version 3.0.x	January 6, 2006
Sixteenth Release	Version 3.1	March 30, 2006
Seventeenth Release	Version 3.1.1	April 14, 2006
Eighteenth Release	Version 4.0	March 22, 2007
Nineteenth Release	Version 4.5	November 27, 2007

Thanks to all who have submitted comments and corrections.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit.

http://www.openssl.org/

Contents

The Basics

The Yahoo! Widgets Konfabulator (also called "Widget Engine" or "engine" in this document) uses XML to define Widgets and the objects that make them up. XML provides a clear hierarchy for what each object is and the order it's drawn in, and associates the correct attributes with each object. A very simple Widget might look like this:

```
<widget minimumVersion="4.0">
    <window id="main_window"
        title="Sample Yahoo! Widget"
        width="500" height="500">
        <image id="sun1"
            src="Resources/Sun.png"
            hOffset="250" vOffset="250"
            hAlign="center"/>
        <text id="text1"
            style="font-size:36px; font-weight: bold"
            hOffset="250" vOffset="100"
            hAlign="center"
            onMouseUp="this.changeOpacity();">
            Click Here/>
    </window>
    <script>
        function changeOpacity() {
            this.opacity = (this.opacity/100)*90;
        }
    </script>
</widget>
```

This Widget reduces the opacity of an image by 10% every time the user clicks the text that says "Click Here." Obviously this isn't terribly useful, but we'll use this simplified example to illustrate a few points. This sample depends on one external file, `Resources/Sun.png`. If you run it without that file, it displays a "missing image" placeholder.

First, note the structure of the Widget. XML is a symmetrical language in that each object specifier (e.g., `<window>`) has a corresponding terminator (`</window>`). Within these pairs, the attributes of the objects are defined as screen positions, alignments, and so on. Also note that objects defined in XML (like `sun1`) can be manipulated in JavaScript (see the `onMouseDown` handler in the `text1` object). IDs of objects must begin with a letter, and only letters, numbers, and underscores are allowed. The XML for a Widget is stored in a file with the extension `.kon` (see "Widget File Structure" and "Widget Packaging" for a discussion of the bundle this file lives in).

Real Widgets can have hundreds of images and text objects, multiple JavaScript sections (often in external files), and usually create new objects at runtime using JavaScript to implement complex functionality.

By far the best and easiest way to get started creating Yahoo! Widgets is to take an existing Widget and start making changes to it. The Widget Engine comes with a selection of Widgets that perform a variety of tasks, any of which would be a good place to start—just remember that although the XML and JavaScript in these Widgets is freely available for reuse, the art assets are not and they must not be redistributed. Consult any license files inside the Widget for specific licensing requirements.

Widget Consolidation

The Widget Engine has a feature that copies any Widget you run into your Widgets folder (My Widgets on Windows). This feature can cause problems when you are developing Widgets, since they will disappear from the folder where you are editing them when you run them. To disable this feature, right-click on the gear icon in the system tray (on Mac, click on the gear icon in the menu bar), choose "Preferences" and select the "General" tab, then confirm that "Consolidate Widgets" is not checked.

XML Syntax

The preferred syntax to use in our XML is attribute-based. This means that you should specify properties of objects as XML attributes, and children as elements. For example, an image should be specified as:

<image src="images/image1.png" hOffset="10" vOffset="20" onClick="clickMe(event)"/>

You will see some Widgets that use elements to define things like `h0ffset`, etc. This is accepted by the parser, but considered bad practice, as it slows parsing down. In general, if you use attributes, and use the CSS styling for objects, there should really never be a need to specify a property as an element.

One other thing to avoid is using the `window` attribute on images, text, etc. You should instead just be sure to enclose the objects inside a window element, as such:

```
<window width="200" height="200">
  <image src="Sun.png" hOffset="10" vOffset="10"/>
  <text vOffset="40" hOffset="10" style="font-size:14px; font-weight: bold">This is some text</text>
</window>
```

We also fully support creating a text object outside of a window element without attaching it to any window. But if you want an object to be bound to a specific window, put it inside the element for best practice and improved parsing.

Please note that while you are currently only *encouraged* to follow this format, it may be required in future versions of the engine.

JavaScript

Because the XML engine looks for the < and > symbols to mark blocks of XML data, our JavaScript engine needs to have these symbols replaced with `<` and `>` respectively. For example:

```
<script>
  if (x &lt; 5)
     displayResults();
</script>
```

Alternatively, you can put the JavaScript code inside CDATA sections to prevent the XML parser from looking at them. This is a better solution for larger blocks of code.

```
<script>
<![CDATA[
  if (x < 5)
    displayResults();
]]>
</script>
```

(CDATA sections were introduced in version 2.1 of the engine.)

You can also avoid the XML parser looking at your JavaScript entirely by putting it in a separate `.js` file and importing it, either via the `src` attribute (note that the charset attribute is also mandatory):

```
<script src="myscripts.js" charset="utf-8" />
```

Or via the `include()` statement:

```
<script>
    include('myscripts.js');
    // additional code here...
</script>
```

File Paths

File paths in the engine are always relative to the location of the XML file. That means a file reference without a directory (e.g., `main.js`) is looked for in the same directory as the XML file while one with a directory (e.g., `javascript/main.js`) is looked for in the specified subdirectory of the directory the XML file resides in. It is not advised to use absolute paths (ones that begin with a /) since the disk layout of people's computers can differ quite markedly.

Widget Metadata

To deal with the increased amount of metadata we've been accumulating over time, including the new Widget dock icon, version 4.0 introduced the concept of a separate file to encapsulate it all: widget.xml. This file should be placed next to your .kon file. The widget.xml file includes information such as the Widget's name, author, etc., and it overrides any such information that might be specified in your .kon file's widget attributes. If you provide a widget.xml file and you still have older attributes in your .kon file, the debug window issues warnings about this to inform you that this is the case.

While the `widget.xml` file is the preferred way to specify your Widget's metadata in version 4.0 and later, you should still leave a `minimumVersion` attribute in your Widget's `.kon` file. This ensures that if you run your Widget on a version prior to 4.0, it will fail gracefully, informing the user that your Widget requires 4.0 or later.

Along with your Widget's metadata, the `widget.xml` file also specifies security settings for your Widget. In versions prior to 4.0, you would put this information inside your `.kon` file. The format of the security element is unchanged, only its location is different. Currently, the security element only controls your Widget's access to Yahoo! web APIs. In the future, it will help control access to the local machine (files, networking, etc.).

Here's an example `widget.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<metadata>
      <name>Vitality Test</name>
      <version>1.0</version>

<identifier>com.yahoo.widget.vitalitytest</identifier>
      <image usage="dock" src="images/SmallSun.png"/>
      <image usage="security"
src="images/SecuritySun.png"/>
      <author name="Edward Voas" organization="Yahoo! Inc."

href="http://www.yahoo.com"/>
      <copyright>(c) 2006-2007 Yahoo! Inc.</copyright>
      <description>This is a really neat
Widget!</description>
      <platform minVersion="4.0" os="macintosh"/>
</metadata>
```

CSS Styles

The engine supports a small subset of CSS for certain objects. The specific attributes supported by the engine are listed later in the section "CSS Reference".

Styles are only accessible when your Widget's minimum platform version is set to 4.0 or newer. This enables styles and adds a style attribute to most objects, and also switches off the 2.0-style behavior of the style attribute on text and textarea objects.

Currently, we do not support style sheets, but that is inevitable as we move forward. We do support inheritance of styles (but not the explicit "inherit" keyword). This means you can set the text style of a frame and have it affect all text objects within it automatically. You can also set the text style of a window and affect all objects in that window.

Styles can be set in two main ways. First, directly in the XML:

```
<text data="Hello, world"
    id="text1"
    style="font-weight: bold; font-family: 'Arial',sans-serif;">
```

Note the single-quotes around font names; unlike browsers, the engine currently requires these quotes. However, "sans-serif" is a keyword and does not require quotes. The second way is in JavaScript:

```
var myTextObject = widget.getElementById("text1");
myTextObject.style.fontSize = "30px";
```

Note that the JavaScript names of styles vary from their CSS versions, as they do in web browsers: hyphens are removed, and letters previously preceded by hyphens are capitalized.

Event Handlers

Event handlers are the lifeblood of Widgets. They allow you to define how a Widget behaves, e.g., when the user interacts with the Widget.

With a minimum version of 4.5 or later:

Event handlers must be functions:

```
myObj.onMouseDown = function( event )
   {
    print( event.x, event.y );
   }
```

If you set an event handler as anything other than a function, the behavior is currently undefined.

Handlers specified in XML are compiled into functions. For example:

```
<image onMouseDown="this.opacity=128;"/>
```

This gets turned into:

```
<function( event )
   {
This.opacity = 128;
   }
```


Versions prior to 4.5:

In versions prior to 4.5, no event parameter is passed to handler functions. Instead, you must use the system.event global to access event data. This is error prone in certain cases though, so if you are on 4.5 or later, you should definitely use the parameter that is passed to you.

In versions prior to 3.0, the only way to specify an action was to set the action to some JavaScript text.

Security Note: Although older versions of the engine allowed you to set strings as event handlers, you should avoid the practice entirely because it's a vehicle for code injection in certain cases. Always use functions.

Object Names

In the XML description, you can set a `name` attribute. This defines the global JavaScript object that will be created and bound to the object the name is a part of. For example:

```
<window name="mainWindow" .../>
```

This creates a JS variable at the global scope with the name `mainWindow`. All names must be unique.

In version 4.0 and newer, you can use a new id attribute instead to identify your objects. An object's ID should be unique within your entire Widget document. You can assign an id attribute at runtime or change an object's ID at any time. You can always find the object by using widget.getDocumentById(). In general, IDs are preferred over names for two reasons:

1. They are more flexible. You can change them on the fly.
2. They don't create global variables. Global variables can be bad if you ever need to make things go away in JavaScript to save memory. For example, if you have a window with 10 items in it, if you create names for those items, you can set the window reference to null but the window won't be destroyed because you have globals pointing at the 10 items (which in turn have pointers to the window). This means you have to track everything and make sure to null out all 10 references, as well as see the object get garbage collected and the memory get reclaimed.

That said, using a name for something that is permanent and global is still a valid use case. Just be aware of the trade-offs.

Widget Preferences

A Widget can provide a number of preference objects to allow itself to save out settings. These settings are saved out in per-user preference storage.

- On the Mac, this is in `~/Library/Preferences/Konfabulator`.
- On the PC, this is in `HKEY_CURRENT_USER\Software\Yahoo\WidgetEngine`.

You can define preferences in the XML:

```
<preference name="windowLevel" hidden="true" value="1"/>
<preference name="windowOpacity" hidden="true" value="0.7"/>
```

And get and set preference objects in JavaScript:

```
preferences.windowOpacity = "0.9";
print(preferences.windowOpacity);
```

If your Widget will be saving large amounts of data, consider using the SQLite embedded database available within the engine.

Frames and Subviews

A `Frame` object allows you to group a set of objects and set properties on that group as a whole. This group is also sometimes called a subview. Objects inside a frame take `hOffset` and `vOffset` values relative to that frame. If you set the opacity of a frame, the opacity of all the objects within that frame are set relative to that value, e.g., an object with opacity 0.5 in a frame with opacity 0.5 will have an effective opacity of 0.25. The visibility of a frame likewise affects the visibility of everything within it.

You can add and remove objects from a Window or a Frame using the DOM-style appendChild() and removeChild() methods.

Even objects not explicitly inside a Frame object are inside a root-level view, and can be accessed via the Window object that contains them. A window does not, however, have all the properties of a full frame, such as scrolling.

You can scroll the contents of frames, to create things like scrolling lists of search results. The engine provides a standard ScrollBar object that you can attach to a frame to scroll its contents. Binding a ScrollBar to a Frame automatically enables mouse wheel support. A ScrollBar can have its appearance customized, either simply by colorizing the standard controls, or by supplying your own images for the track and thumbs. See ScrollBar for more information.

Support for frames was added in version 3.0 of the engine. DOM support began in version 4.0.

Working with the Widget Dock

Version 4.0 and newer include a Widget management UI in the form of a Widget dock. This dock can be used to show either all installed Widgets, or just the Widgets that are running. As a Widget author, you can control what is displayed in your dock item. You can supply an icon, which is used for the times your Widget is not running, as well as more dynamic information that might get shown while the Widget is running.

The dock info area is 75x70 pixels. Your icon should be designed to fit in this area. You do not need to treat the edges of your icon to fit in with the look of the Widget dock. The dock automatically stylizes your image to match. See below for more information.

Optionally, you can specify text and graphics to be displayed in that area instead of or in conjunction with your icon while your Widget is running. For example, if you were writing a mail checker Widget, you might want to show the number of mail messages waiting. A weather Widget might want to show the current temperature and other weather statistics.

You specify this information by calling widget.setDockItem(). You pass a well-formed XML document to that function. XML is a subset of the Widget language. You are allowed to use `Frame`, `Text`, and `Image` objects only.

Due to performance concerns, always use as large an interval as your Widget can tolerate. The general rule of thumb is to not call more than every 10-15 seconds. Some Widgets might need more frequent updates (e.g., a CPU meter), but these should be kept to a minimum. Never call setDockItem() more than once per second. Due to the amount of work and the number of Widgets that could be running, even once per second could have a negative impact on overall system performance.

To specify an icon for a Widget, you must include a `widget.xml` file and the appropriate icon element (see "Widget Metadata" for more info).

To specify dock info, the easiest thing to do is create a small XML file that lives next to your `.kon` file. At runtime, just read that file into a DOM. You can then manipulate the info in the DOM to change the values of what you want to show. Then pass the DOM document into setDockItem().

Here's an example of a small XML file representing a dock item's live state (also called vitality):

```
<?xml version="1.0" encoding="utf-8"?>
<dock-item version="1.0">
   <image id="img" src="images/background.png"/>
   <text id="text"
      hAlign="center"
      vOffset="20"
      style="font-family:'Trubuchet MS';
         font-size:18; font-weight:bold; color:white"/>
</dock-item>
```

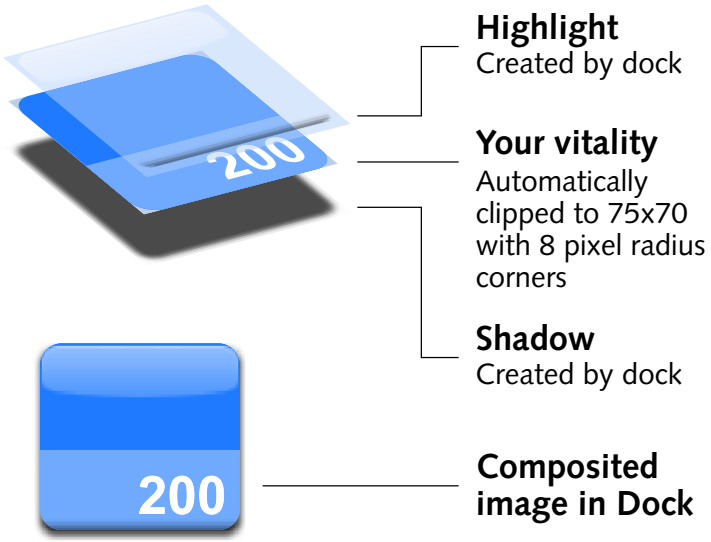
Here's a small sample Widget that demonstrates how to use this file to periodically set the dock info:

```
<widget debug="on">
    <script>
       var words = ["hi", "there", "you", "guys"];
       var i = 0;

       var doc = XMLDOM.parse( filesystem.readFile( "vitality.xml" ) );
       t = doc.getElementById( "text" );

       function setMyDockInfo()
       {
          t.setAttribute( "data", words[i++] );
          if ( i > (words.length-1) )
             i = 0;

          widget.setDockItem( doc );
       }
    </script>

    <timer interval="10.0" ticking="true">
       setMyDockInfo();
    </timer>
    <action trigger="onLoad">
       setMyDockInfo();
    </action>
</widget>
```

See Appendix Afor the DTD of the dock item XML.

As mentioned above, when you specify your icon, the dock stylizes it automatically. This is also true for your vitality. You can turn off the stylization for vitality by setting the transparent attribute of the dock-item element to be true.

In general, you should always let the dock stylize your image for you. When creating your icon or dock info, you should always treat the area as a 75x70 pixel rectangular area. While the first version of the dock has rounded corners, other releases might not. The rounded corner effect is part of the stylization. In essence, the image that you provide to us is masked and then sandwiched between a shadow and a highlight layer. Below is a picture illustrating how stylization works.



In general, corners might be rounded, or they might not. You should plan your content so that no important information is jammed into the corners as it might get partially cut off depending on the current method of stylization.

Publishing Your Widget

Once you've created your working Widget, how do you get it out to the world? Upload it to the Gallery! Your Widget will be reviewed and approved by a real person, and appear on widgets.yahoo.com, usually within a week. The Gallery Upload page is here:

`http://widgets.yahoo.com/gallery/upload.php`

But first, it's a good idea to make sure your Widget is formatted properly for quick approval.

Widget File Structure

Your Widget files should be organized in a folder structure that looks like this:

```
myWidget.widget
    Contents
        widget.xml
        myWidget.kon
        Resources
            <any files used by the Widget>
```

It is important that the folder that contains your Widget have ".widget" in its name, and that it contain a folder called Contents in which your actual .kon file should be placed. (The Contents folder looks a bit superfluous, and is a legacy of the way Widget files used to be organized on the Mac. Just accept this and move on.) You can put any other files or folders anywhere within your Contents folder, but we recommend organizing them into a Resources folder, as shown.

The widget.xml file is not necessary to run your Widget, but is very strongly recommended. It specifies information about your Widget such as the name of the Widget, the author, organization, copyright information, and relative paths to images used by the Widget dock and any security dialogs. See "Widget Metadata" for more information.

Widget Packaging

Widgets are contained in `.widget` files, which are simply your `.widget` folder compressed into a single file, either as a Zip file or a custom flat-file format created using the Widget Converter tool. The flat-file format is newer and is the recommended way to create your Widget, as the Zip file solution suffers from a number of performance problems.

To create a flat-file Widget, download the Widget Converter Widget from the Workshop (http://widgets.yahoo.com/workshop) and simply drag your .widget folder onto it. It will automatically output a .widget flat file.

To create a Zip-formatted Widget, simply add your .widget folder to a Zip file using your favorite compression tool, such as StuffIt on Mac OS X or the "Send to… Compressed folder" right-click option on Windows XP. The Zip file must have the .widget folder as the top-level item within it, not the Contents directory.

On Mac OS X, you can also create a .widget file by packaging your files into a bundle, which is a directory that is treated as a single unit by the operating system. You can control-click one of the default Widgets and choose the Show Package Contents option to see this structure in use. However, Widgets created in this way will not work on Windows, so it is no longer recommended.

Important note about using the flat file format
Because your Widget is a flat file, you cannot use items such as DLLs that you might have packaged with your Widget unless you use a new API (widget.extractFile()) to extract the file out of your flat-file Widget into a location in the filesystem. However, sound files played through the play() function will work without any changes.

Giving Your Widget an Identifier

Since version 4.0 of Konfabulator, Widgets should have a widget.xml file (see "Widget Metadata" for more details). widget.xml includes an <identifier> field. You can use any value for this identifier as long as it is guaranteed to always be unique (i.e., no two Widgets should have the same value for Identifier). To be safe, we recommend using a UUID, e.g.,

```
<identifier>425b0c3e-fd99-11db-8314-0800200c9a66</identifier>
```

UUIDs can be generated for free by a number of tools, including this one:

```
http://www.famkruithof.net/uuid/uuidgen
```

Using a UUID means we will always be able to identify your Widget, even if you change the name or other metadata. It also means your Widget will automatically update itself! See "Updating Your Widget" for more information.

Updating Your Widget

After you publish your Widget, you will nearly always spot some bugs that you'd like to fix, or invent some new features you'd like to include. Great! You can use the Gallery Upload tool to do this without creating a new Widget every time.

If you use a UUID to identify your Widget (see "Giving Your Widget an Identifier"), users of your Widget will be notified when you update it by a small icon that appears on your Widget's dock tile. Clicking the icon will produce a dialog asking the user if they want to update to the new version. If they agree, the new version is automatically downloaded and installed (neat!).

Digital Signatures

Starting in version 3.1, the engine supports the digital signing of Widget files. This is done with the converter command-line tool (see "The Converter Tool" for more info) or the Widget Converter utility, which can be found on the Yahoo! Widgets Workshop page, or in the Widget Gallery at http://widgets.yahoo.com/gallery.

At a minimum, signing your Widget ensures that the integrity of the Widget is intact from the time it was signed. The engine will display a security dialog whenever it encounters a Widget for the first time and displays its signature information (see "Security").

If you sign your Widget with a code-signing certificate issued by VeriSign, we can also verify the authenticity of the certificate itself. We intend to support more certificate authorities in future releases.

Security Windows

There are two similar types of security windows that might appear in the Widget Engine. The first is a first-run/modification window. On first run of a Widget that the Widget Engine is not familiar with, a window appears telling the user they are about to open a new Widget and asking them to OK this. This is to protect against Widgets that might run without the user's knowledge. Also, if the user allows a Widget

to run and later on that Widget is modified, the next time the Widget is launched, another window appears to note this change. Again, the user can confirm or deny the request to launch depending on whether or not the modification was expected.

If you are actively debugging a Widget, you can turn debug mode on (which is probably a good idea anyway). This suppresses the first-run/modification windows, so you are not notified every time you tweak your code and reload the Widget.

The second type of window is a "sandbox" window. Currently, the only sandboxed action is logging in to a user's Yahoo! account (more actions will be sandboxed in future releases). The first time a Widget attempts to log in to a user's Yahoo! account, a window appears to alert the user and ask whether the Widget should be granted permission to use their Yahoo! data. Sandbox windows cannot be disabled.

Security

Version 4.5 introduces a much higher level of security specification to protect users by requiring authors to narrowly define access.

In this release the security block will contain the following definable elements:

- `filesystem`
- `com/AppleScript`
- `command`
- `http`
- `hotkey`

One thing to note is that the API element is no longer necessary when defining the minimum version of the widget to 4.5 (see below for more details).

When the widget is first loaded, just like in the last version, a dialog will be displayed explaining to the user what access levels the widget needs. Anything not expressly allowed will be denied. So for example, if a Widget doesn't announce it will use the filesystem, then tries to read a file, it will fail.

Security Definition Details

The block must be defined in the widget.xml metadata file:

`http`	• Permitted values: domains (i.e. digg.com), all, none • Purpose: To restrict HTTP requests • Default: none (No Access to any domain)
`filesystem`	• Permitted values: none, sandbox, user, full none - No filesystem access sandbox - Widget data directory only user - User home directory (Platform defined) full - Anywhere on the system (Platform limited) • Purpose: To restrict filesystem access • Note: SQLite access is considered filesystem acess. To use SQLite, filesystem must be set to sandbox or greater. • Default: none)
`com`	• Permitted values: duid of the COM object • Optional values: name of the COM object • Purpose: To restrict COM requests • Default: No COM plugins allowed

applescript	• Permitted values: true, false • Purpose: To restrict Applescript execution • Default: No Applescript execution
command	• Permitted values: true, false • Purpose: To restrict system commands • Default: No runCommand or runCommandInBg execution
hotkey	• Permitted values: yes, no, true, false • Purpose: To restrict global hotkey setting • Default: No hotkeys can be set

Example:

```
<security>
     <http name="Digg">digg.com</http>
     <filesystem>sandbox</filesystem>
     <com name="Custom Date">77F81807-FF09-4080-9726-3944E8682CD3</com>
     <applescript>true</applescript>
     <command>true</command>
    <hotkey>false</hotkey>
</security>
```

What happened to api?

Because we have expanded our security block to include the http element in version 4.5, the api definition is no longer necessary. However, if you set the minimumVersion attribute of your widget to less than 4.5, you are still required to provide the api definition for backwards compatibility.

What about older widgets?

For backwards compatibility, only widgets set to a minimumVersion of 4.5 will use the new security block. This means that if the minimumVersion is set to less than 4.5, the widget will only be restricted by the api definition. This also means that the security dialog presented to the user will request a much higher level of access, which may cause unnecessary concern for the user. It is **highly** recommended that the minimumVersion be set to 4.5 for this reason.

Security Violation

If a Widget attempts to perform an action that violates the security permissions granted, the Widget displays a dialog to the user informing them of the nature of the action (http, filesystem, com, applescript, command). The dialog has only one button, close widget that terminates the process immediately. The widget is **not** directed through the normal close routine, but is immediately killed.

If the widget debug mode is turned on, we don't force the widget to shut down. Instead, the debug window will show the security exception that caused the failure so you can correct the situation.

Advanced

Migrating From Previous Versions

The minimum platform version of a Widget should be specified by the `minimumVersion` attribute in the `<widget>` tag as well as in the `widget.xml` file so that it can be read by all versions of the engine. Setting the minimum version allows the engine to inform the user that they need to upgrade their engine if they attempt to run your Widget in an older version of the engine. More importantly, it specifies the way the engine should treat certain attributes whose behavior has changed since the earlier versions of the engine. To get the most correct behavior out of the engine, you should always specify the current `minimumVersion` for your Widget.

Version 3.0

If you set your minimum version to 3.0 (which you should if you are taking advantage of features in 3.0), the following behaviors come into play:

1. No views are auto-bound to the default window. This used to be the case, but with the advent of hierarchical views in 3.0, this became problematic. As a result, you must specify the window that an object belongs to, or use `frame.addSubview()` to embed an object into a frame. If your interface is mostly constructed with XML, the simplest thing to do is enclose your `image/text/frame/scrollbar` objects inside your window object:

   ```
   <window ...>
        <image src=.../>
        <text data=.../>
        <frame .../>
             <image src=.../>
        </frame>
   </window>
   ```

   As you migrate your Widget to 3.0, the most common error you would probably encounter is to see some of your views not appearing. This is due to this behavioral change. Simply double-check that all your views are bound to some window or parent frame.
2. JavaScript lifetime changes. In prior releases, calling delete on an object or setting it to null would make the object disappear from the window. In version 3.0, if you wish for an object to be removed, you must call `<object>.removeFromSuperview()`. This change makes it easier to code a Widget. In the past you had to maintain lists of all your objects to ensure they didn't disappear with the window. With the advent of subviews, the number of objects can become unmanageable quickly. Now you no longer need to care if you have a reference to an object if you've added it to the window. This means items that would never change during the course of your Widget never need to be tracked by you. This makes your code more obvious in many ways so you can concentrate on doing what you do best.
3. We no longer blindly replace XML entities in your `.kon` or `.js` files when files are loaded. If you want to ensure that JavaScript code that has < or > in it doesn't trip up the parser, you should use CDATA sections, as mentioned in "Entities."
4. Rotation changes. If you center an image using `hAlign` and `vAlign` and then rotate it, it will rotate around the center of the image using `hOffset` and `vOffset`.
5. JavaScript code in an XML element is read simply as JavaScript code. Previously, the engine would try to see if it was a file by trying to read a file with the given path. We now only try to read a file if your action has the `file` attribute. If you want to include a file in the element, use `include()`. This should improve loading performance as we won't hit the filesystem for every chunk of JavaScript code in your Widget.

Version 3.1

On Windows if you set your minimum version to 3.1 or newer, the `window.shadow` attribute is respected.

Version 4.0

When your Widget's minimum platform version is set to 4.0 or newer:

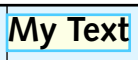
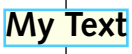
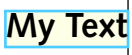
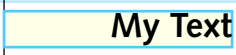
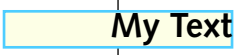
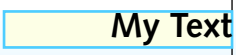
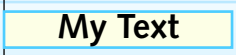
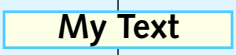
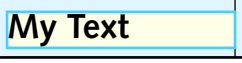
1. The zOrder attribute on objects is treated differently. In prior releases, the zOrder of an object was an ever-increasing number. That is, as new objects were created, each object was given a higher zOrder. This caused issues when a Widget created many objects during its lifetime if it wanted to try to use zOrder to forcibly order an object above all other objects. Eventually, new objects would end up getting a higher zOrder index than the object in question.

   To avoid this, we no longer increase the zOrder for each object. All objects now get a default zOrder of zero. To adjust zOrder among sibling views, use the new functions orderAbove() and orderBelow(). You can still use zOrder to move views above other views, but they act more like layers now. For example, all `zOrder 1` objects are above all `zOrder 0` objects.
2. Putting JavaScript code in comments is not supported. Use CDATA as needed.
3. Rudimentary CSS styling is enabled. As a result, the style attribute of text and objects is repurposed to be a CSS-style declaration list. Values of bold and italic are no longer supported.
4. XMLHttpRequest auto-forwards by default (i.e., it handles redirects), and a per-session cookie jar is enabled for your Widget. You no longer need to handle cookie headers yourself. Redirection can be disabled using the autoRedirect attribute on the XMLHttpRequest object. Automatic cookie handling cannot be disabled.
5. Animations always call the `done` function if one is given, whether the animation was run synchronously or asynchronously. Previously, we would only run the `done` function for asynchronous animations.
6. The `onMouseMove` action now occurs whenever the mouse is moving across an object. The new `onMouseDrag` action duplicates the old behavior of `onMouseMove`, and triggers only when the mouse button is held down. Now you can properly differentiate between a real drag action and just a move.
7. The behavior of XMLHttpRequest is more in line with the W3C Working Draft published on June 19, 2006 (http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060619/). In general this shouldn't affect you unless you are trying to set headers, e.g., after calling `open()`. We now throw DOMExceptions in that case, as per the draft.

Version 4.5

When your Widget's minimum platform version is set to 4.5 or newer:

- All event handlers are required to be functions.
- The `hAlign` property of a text object with the `wrap` attribute set to `false` no longer affects the text alignment. You must use the CSS `text-align` attribute to change justification inside its box.

text-align vs. hAlign	width	hAlign	CSS text-align	hOffset
With no width specified, the Text object's width is the bounds of the text it contains and CSS alignment has no visible effect.	(n/a)	left	right	
	(n/a)	center	left	
	(n/a)	right	center	
When a width is specified that's wider than the enclosed text, CSS alignment positions the text within that width.	85	left	right	
	85	center	right	
	85	right	right	
Depending on the Text object's width, hAlign and CSS text-align values, the actual text position may not be what you expected.	85	left	center	
	85	center	center	
	85	right	left	

Entities

In XML, some characters such as "<" and ">" are important when parsing the syntax, and so are considered "reserved" and should not be used anywhere else in the document. In order to represent these characters in the XML, you need to refer to them by codes. This is known as entity escaping, and Konfabulator uses a standard set of entities:

```
&amp;       &
&quot;      "
&apos;      '
&lt;        <
&gt;        >
```

In addition to the basic entities, you can refer to any character using its Unicode reference, either as decimal or hexadecimal, for example:

```
&#32;          <space character, decimal>
&#x20;         <space character, hex>
&#x176;        <degree symbol, decimal>
&#x00B0;       <degree symbol, hex>
```

However, the XML parser does not handle JavaScript. Therefore, if you wish to represent special characters within JavaScript strings, you need to use the JavaScript syntax to refer to them, for example:

```
\u0020      <space character>
\u00B0      <degree symbol>
```

Make sure you use the right type of escaping at the right time:

```
<widget minimumVersion="4.0">
    <window
        style="background-color: blue;"
        width="140"
        height="40">
        <text
            style="background-color: white;"
            hOffset="20"
```

```
            vOffset="20"
            width="100"
            height="20"
            data="w&#x00F8;&#x00F8;"
            id="text"
            onclick="getCode();">
        </text>
    </window>
    <script>
    var getCode = function() {
        var text = widget.getElementById('text');
        text.data = "y\u00E5y";
    }
    </script>
</widget>
```

There are many tools for finding the Unicode number of a character, such as this one:

`http://people.w3.org/rishida/scripts/uniview/descn`

Widget Runtime

This section discusses how Widgets are run and some issues to keep in mind.

When a Widget is opened, it is run as a separate process. This is done to prevent one broken Widget from bringing down all the others.

Zipped Widgets are unzipped into a special location (/tmp on the Mac, and C:\Documents and Settings\<user>\Local Settings\Application Data on the PC). A Widget that is not zipped is run right from where it is located. For this reason, the location of your Widget when it is run is not guaranteed.

The working directory of a Widget is the directory in which the .kon file is located (usually Contents). This allows relative paths to resources to work. Your Widget would reference an image as Resources/Image1.png, for example, if its images were inside a Resources folder inside Contents.

When your Widget is opened, the .kon file is parsed and all the objects defined within it are created. The onLoad handler is then called. You should be careful that your onLoad functions do not take overly long to run, as many Widgets set their visibility to false initially and then make themselves visible at the end of their onLoad handler.

Each time your Widget is run, the Widget is unzipped again. For this reason you should not store information in your Widget's working directory. Instead, you should use system.widgetDataFolder.

Any Widget running when the engine is closed will be re-opened the next time the engine is run.

Debugging

To help Widget developers, Konfabulator has a Debug Mode. In this mode, a debug output window opens when your Widget is started. While developing your Widget, you should always turn debug to on so you can keep track of errors. For example, if you misspell an attribute, or make a syntax error in your JavaScript, the output window tells you this, along with where the problem is in your code. Calls to `log()` or `print()` in your JavaScript code appear in this window. Any errors encountered inside the Widget Engine are also reported in this window.

The debug window has a command line. You can enter any valid JavaScript statements to be evaluated into this (this is often useful for testing the values of variables). You can also enter special debug commands to trace variables and functions and perform other useful tasks. To see a complete list of these commands, type `"/help"` into a debug window.

To activate debug mode, hold down Control and Shift and right-click on the Gear icon in the system tray (on Windows) or the menu bar (on Mac). An additional option to activate debug mode will appear in the menu. You will need to close and re-open any Widgets that are already running if you want to put them into debug mode. You can also activate debug mode on a per-Widget basis by putting a `<debug>` tag with a value of `"on"` inside your `<settings>` block (see "Settings" for more information). Debugging is the same no matter how it is activated, and it is usually easier to use engine-wide debugging.

Exceptions

When things go awry, the engine will sometimes give you a chance to recover from the error before informing the user, behavior known as *throwing an exception*. This is useful in the COM interfaces for Windows, and the filesystem object. Here is an example of how to use exceptions, from the Day Planner Widget:

```
try
{
    outlook = COM.createObject("Outlook.Application");
    // work with Outlook object here...
}
catch( e )
{
    print( "Unable to connect to Outlook: " + e );
}
```

Exceptions are real JS objects, but when printed, they'll automatically display a string representation in version 4.5 and newer. Prior to version 4.5, exceptions were merely strings. Handling exceptions is essential when dealing with COM interfaces of any kind.

XML Services

In version 3.0 and newer, we provide services to allow you to work with XML more easily. In 3.0 we now have a built-in XML parser that is significantly faster than using the external JavaScript-based `xmldom.js` file. This XML parser always operates in "strict" mode.

The output of the parser is a Level 1 W3C DOM and we follow the specification for that DOM. There are a few omissions (entities, for one), but the important core is there. You can also create and edit these DOM trees to make your own XML documents and output them.

The DOM API is nice, but in general it's not convenient to traverse an XML tree to find the important bits. So we've also added XPath 1.0 support (minus namespace-specific functions). This makes it much easier to pull out pieces of a XML tree instead of using the DOM API.

To aid in moving code into Yahoo! Widgets, we also support the `XMLHttpRequest` object. For POSTing files, we still recommend you use the `URL` object.

Yahoo! Login Support

Version 3.0 and newer allow you to use APIs that require a Yahoo! login. The engine itself takes care of the details of logging in and storing credentials. Your Widget only has to check the current login state or request to log in. When logged in, when sending the API request to the server, the engine automatically adds the user's credentials for you.

Important: In version 3.1 and newer, you must specify the specific Yahoo! APIs your Widget wants to connect to in a `<security>` block (see the section on the security block in the XML reference for more info). This list of APIs will be confirmed by the user before your Widget is allowed to access them with Yahoo! credentials, and only those APIs will receive the Yahoo! credentials. If you were using `yahooLogin()` before 3.1, your Widget will no longer be able to access those APIs until it is modified to included the security block.

To behave like a good citizen, you should first check to see whether you are logged in by calling `yahooCheckLogin()`. If this returns true, you are all set to access the Yahoo! API your Widget would call. If it returns false, you should display a placard or some other indication that your Widget cannot display its information because the user is not currently logged in and give them a button/link/something to click to enable them to log in from your Widget.

In your `onLoad` handler, for example:

```
if ( yahooCheckLogin() )
   loggedIn(); // display your UI in the logged in state.
else
   loggedOut(); // display your UI in the logged out state.
```

It is considered bad form to blindly call `yahooLogin()` in your `onLoad` handler.

When the user clicks your button to log in, call `yahooLogin()`. If this function returns true, you are already logged in. If it returns false, the user must authenticate. When `yahooLogin()` returns false, your Widget should wait for an `onYahooLoginChanged` event to come to your Widget (i.e., the function behaves asynchronously). This means your code for a logged-in state should always be in a function triggered by an `onYahooLoginChanged` event. This pattern is useful, since the user logging in or out via the Gear menu can also trigger the `onYahooLoginChanged` event.

When your `onYahooLoginChanged` handler is called, you must call `yahooCheckLogin()` to see what your new state is (this call also loads information such as necessary cookies). Based on the state returned, you would either behave logged in or out, just as shown above for `onLoad`.

Be warned that even if `yahooCheckLogin()` returns true, your request to the API server might fail due to expired credentials (this can happen after a prolonged period of inactivity). In this case, it is good Widget behavior to call `yahooLogout()` so that other Widgets are aware of the situation.

```
<action trigger="onYahooLoginChanged">
        <![CDATA[
           if( yahooCheckLogin( ) ) {
              // your logged-in code here...
           } else {
              yahooLogout( );
              // your logged-out code here...
           }
        ]]>
      </action>
```

Localized Widgets

If your Widget is going to be used across the world, you can create a single Widget that can be viewed in multiple languages. To do this, replace all the text strings in your Widget with calls to the localized string function, for example:

```
widget.getLocalizedString("save_as_button");
```


The engine will look up the "translation" for this string in the most specific language file it can find. Language files should be named Localizable.strings, and are written in a keyed string file format. For example:

```
"save_as_button" = "Save As";
```

The key value (the one you send to getLocalizedString) is on the left, and the translated value is on the right. Both sides must be in quotes, and each mapping must end with a semicolon.

The engine expects to find these files in a series of directories within the Resources folder of your Widget. The directory you put these files in must be named using the ISO 639-1 language code and optionally the ISO 3166 locale code. For example:

English language, all locales:

```
Resources/en/Localizable.strings
```

English language, U.S. locale:

```
Resources/en_US/Localizable.strings
```

To find the correct strings file, the engine searches for <lang>_<locale> first, then simply <lang>. If it can't find either, it loads "en" by default. The language/locale we use is defined by the Language setting in the engine's Preferences dialog. This language setting affects only Widgets that are run after the setting has been made. One of the settings is the system default, i.e., whatever language your OS is running.

See also locale.

Keep in mind that your Widget needs to deal with strings that might be longer than what the Widget was designed for. For example, German strings are typically much longer than their English equivalents.

A list of ISO 639-1 (and 2) codes can be found here:

```
http://www.loc.gov/standards/iso639-2/php/code_list.php
```

A list of ISO 3166 codes can be found here:

```
http://www.iso.org/iso/country_codes/iso-3166-code-lists/
     english_country_names_and_code_elements.htm
```

Core DOM Reference

The following sections describe the objects and attributes that make up Widgets. Objects are organized into a hierarchy as follows:

```
<widget>
<about-box/>
<action/>
<hotkey/>
<preference/>
<security/>
<script/>
<window>
  <canvas/>
  <image/>
  <text/>
  <frame/>
  <textarea/>
  <scrollbar/>
  <script/>
</window>
</widget>
```

Other blocks are read as subblocks:

```
<menuItem/>
<shadow/>
```

Objects must be inside a window object (this was not the case in previous versions). This means you can put objects like images, text, and text areas into the block for the window:

```
<window>
...
<image name="foo"/>
<text .../>
</window>
```

Using this format, you do not need to include the `window` attribute for any of the nested images since the window is known to be the containing window specified in the XML. If you do specify a window, you will get an error in the debug window warning you of this fact.

Important: In version 3.0 and newer, it is a requirement to put your objects inside the window declaration, otherwise they are not visible.

The DOM

the Document Object Model

In version 4.5 and later, there is a full W3C DOM in place. Whereas objects such as `Window` and `Image` existed in the XML, there was no way to access those at runtime unless you provided name attributes for them. The DOM allows you to get at any aspect of your original XML document from Javascript, with a few minor exceptions.

Like any HTML document, you can access anything in your Widget via the document object. Typically, you'd use document.getElementById() to look up an element and return a reference to it. Since there's a real DOM behind everything, you can also use Konfabulator's XPath facility to do batch lookups. For

example, you might want all image tags from a particular window. You can use an XPath expression such as window.evaluate('image') to fetch them all. You can also use the simpler getElementsByTagName()API to do simple fetches of descendants with specific names.

Having the DOM allows you to do things you never could before. For example, you could change your about-box object at runtime before it's displayed. Or you can now change context menu items of an object just by using the standard DOM APIs. Another highly useful thing you can do with the DOM is to adjust your preferences on the fly. For example, you can dynamically build your option children of a preferences item before preferences are shown.

XML DOM API

This section lists the various objects and methods/attributes currently supported by the Widget Engine's Level 1 W3C DOM implementation. We currently provide a large subset of the full API. The current parser does not yet deal with DTDs, so it does not do things such as fill in attributes with default values automatically.

The following is a brief overview of the attributes and functions we support. For more information, we suggest you visit the w3c.org web site.

DOMException

standard exception class for the DOM

When an exceptional situation arises, a `DOMException` is thrown as a JavaScript exception. You can inspect the object's `code` attribute to see what happened. Level 1 exception codes are:

```
INDEX_SIZE_ERR                  1
DOMSTRING_SIZE_ERR              2
HIERARCHY_REQUEST_ERR           3
WRONG_DOCUMENT_ERR              4
INVALID_CHARACTER_ERR           5
NO_DATA_ALLOWED_ERR             6
NO_MODIFICATION_ALLOWED_ERR     7
NOT_FOUND_ERR                   8
NOT_SUPPORTED_ERR               9
IN_USE_ATTRIBUTE_ERR            10
```

DOMDocument

represents an entire XML document

Attributes

Attribute	Description
`doctype`	The document type definition for the document.
`documentElement`	The root element of the document.

Functions

`DOMElement createElement(string tagName);`

Creates a new element node for the document with the given tag name. You must attach it to the document as appropriate using `appendChild`.

`DOMText createTextNode(string data);`

Creates a new text node for the document with the given content.

```
DOMComment createComment(string data);
```

Creates a new comment node with the given content.

```
DOMCDATASection createCDATASection(string data);
```

Creates a new CDATA section with the given data.

```
DOMProcessingInstruction
createProcessingInstruction(string target, string data);
```

Creates a new processing instruction with the given target and data.

```
DOMAttribute createAttribute(string name);
```

Creates a new attribute node with the given name.

```
DOMNodeList getElementsByName(string name);
```

Returns a list of all elements in the document with the specified name.

```
DOMNode importNode(DOMNode node, boolean deep);
```

Imports a node from some other document into this one. Available in version 3.1 or newer.

DOMNode

base class for items in an XML tree

`DOMNode` is the base class for pretty much everything you'll deal with in the DOM API. You'll never encounter a `DOMNode` in everyday life, but its interface is something that is common to all node types (text, element, CDATA, etc.) and as such is documented once here rather than separately for each subclass

Attributes

Attribute	Name
`nodeName`	The name of this node.
`nodeType`	The node type, expressed as an integer.
`parentNode`	The parent of this node (can be null).
`childNodes`	A `DOMNodeList` of children.
`firstChild`	The first child node of this node.
`lastChild`	The last child node of this node.
`previousSibling`	The previous sibling node of the current node.
`nextSibling`	The next sibling node of the current node.
`attributes`	A `DOMNamedNodeMap` of this node's attributes (only valid for `Element` nodes, null otherwise).
`ownerDocument`	The `DOMDocument` that this node belongs to.

Functions

```
DOMNode insertBefore(DOMNode newChild, DOMNode refChild);
```

Inserts `newChild` before `refChild` in this node's children.

```
DOMNode replaceChild(DOMNode newChild, DOMNode oldChild);
```

Replaces `oldChild` with `newChild`.

```
DOMNode removeChild(DOMNode oldChild);
```

Removes `oldChild` from this node's children and returns it.

`DOMNode appendChild(DOMNode newChild);`

Adds the given child node (if this node type allows children).

`boolean hasChildNodes();`

Returns true if this node has child nodes.

`DOMNode cloneNode(boolean deep);`

Clones this node. If deep is true, clones all descendents as well.

`DOMNode importNode(DOMNode inNode, boolean deep);`

Clones the given node from its world and adds it into our document. If deep is true, clones all descendents as well. We currently do not do any sort of validation that the node type is valid per any DTD.

`<various> evaluate(string xpath-expression);`

This is an extension defined by the Widget Engine that lets you interface with the engine's XPath support. Using the current node as the context for the XPath expression, you can execute almost any XPath 1.0 expression that you can dream up (except for some namespace-specific functions). The result could be a string, number, or a set of nodes. The Widget Engine returns node sets as `DOMNodeLists`. See the section on XPath Support for more information.

`string toXML();`

Widget Engine `DOMNode` extension. Converts the subtree starting at this node into XML for output for either writing to a file, or possibly for debugging purposes.

DOMNodeList

simple list of nodes

In keeping with W3C ways, any list of nodes as expressed through the DOM API is represented as a `DOMNodeList`, not as a JavaScript array.

Attributes

Attribute	Description
`length`	The number of items in the list.

Functions

`DOMNode item(n)`

Returns the nth item in the list. `DOMNodeLists` are zero-based.

DOMNamedNodeMap

map of nodes that is accessible by name or index

When attribute nodes are returned through the `attributes` attribute of the `DOMElement` node, they are returned in a named node map. This map is primarily accessible by name, but you can also traverse it through an index like a `DOMNodeList`. The order of the attributes is not guaranteed and should never be relied upon.

Attributes

Attribute	Description
`length`	The number of items in the list.

Functions

`DOMNode getNamedItem(string name);`

Returns the item with the given name, or null if the item is not found.

`DOMNode setNamedItem(string node);`

Adds the given node to the map. If a node with the given name exists, it is replaced and the old node is returned. If a node with the given name does not exist, null is returned.

`DOMNode removeNamedItem(string name);`

Removes the item with the given name, if it exists.

`DOMNode item(int n);`

Returns the nth item in the list. `DOMNodeLists` are zero-based.

DOMCharacterData

base class for text and comment nodes

This class, like `DOMNode`, is something that you'll never encounter in real life, but its interface is available for both `DOMText` and `DOMComment` nodes. As with `DOMNode`, the interface is shown here once and not repeated in both of those classes.

Attributes

Attribute	Description
`data`	The actual character data.
`length`	The length of the character data.

Functions

`string substringData(int offset, int count);`

Returns a substring of the data as a string. It returns `count` characters of the data starting at `offset`.

`void appendData(string data);`

Appends the given text to the node's data.

`void insertData(int offset, string data);`

Inserts the given string at the specified `offset`.

`void deleteData(int offset, int count);`

Erases `count` characters of data starting at `offset`.

`void replaceData(int offset, int count, string data);`

Replaces the sequence of `count` characters starting at `offset` with `string`.

DOMAttribute

attribute node for an element

Attributes

Attribute	Description
`name`	The name of the attribute.
`value`	The value of the attribute. Character and entity references are resolved before returning this value.

DOMElement

element node

Attributes

Attribute	Description
`tagName`	The tag name of the element.

Functions

```
string getAttribute(string name);
```

Returns the value of the attribute specified, or an empty string if that attribute does not exist.

```
setAttribute(string name, string value);
```

Adds the given attribute and its value to the element, replacing any attribute of the same name that might already exist.

```
removeAttribute(string name);
```

Removes the attribute with the specified name, if present.

```
DOMAttribute getAttributeNode(string name);
```

Returns the attribute node corresponding to the name passed in, or null if the attribute does not exist.

```
DOMAttribute setAttributeNode(DOMAttributes attr);
```

Adds the given attribute to the element, replacing any attribute that might exist with the same name. If the node replaces an existing node, the old node is returned as the result, else null is returned.

```
DOMAttribute removeAttributeNode(DOMAttribute attr);
```

Removes the node specified from the element's attributes and returns it.

```
DOMNodeList getElementsByTagName(string name);
```

Returns a list of all elements with the specified tag name that are a descendant of this node.

```
void normalize();
```

If there are contiguous `DOMText` nodes in the subtree starting with the current element, this function combines them into a single element.

DOMText

text element

Functions

```
DOMText splitText(int offset);
```


Splits the given node into two and adds the new node as its new sibling following it in the tree. This node contains the text up until `offset`. The following node contains the remainder of the text. The new text node is returned.

DOMComment

comment node

This node has the attributes and functions of the `DOMCharacterData` interface.

DOMCDATASection

CDATA section

This node has the attributes and functions of the `DOMCharacterData` interface.

DOMDocumentType

document type node

Currently, this node only defines the `name` attribute. Entities and notations are not supported by the current version of the Widget Engine.

Attributes

Attribute	Description
`name`	The name of the document's root object. For a Widget, this would be "widget." For HTML it would be "html."

DOMNotation

notation node

Currently unsupported.

DOMEntity

node representing an entity

Currently unsupported.

DOMEntityReference

node representing an entity reference

Currently unsupported.

DOMProcessingInstruction

node representing a processing instruction

Attributes

Attribute	Description
`target`	The target of the processing instruction.
`data`	The content of the processing instruction. This is from the first non-whitespace character after the target to the character immediately preceding the "?>".

Common Attributes and Functions

things that are common to many different objects

This section lists the attributes and functions that are common to many of the objects that follow. For example, several different objects can have `onMouseUp` handlers, context menus, etc.

Attributes

- contextMenuItems
- hAlign
- height
- hOffset
- id
- firstChild
- lastChild
- name
- nextSibling
- previousSibling
- onClick
- onContextMenu
- onDragDrop
- onDragEnter
- onDragExit
- onMouseDown
- onMouseDrag
- onMouseEnter
- onMouseExit
- onMouseMove
- onMouseUp
- onMouseWheel
- onMultiClick
- opacity
- parentNode
- style
- subviews
- superview
- tooltip
- tracking
- vAlign
- visible
- vOffset
- width
- window
- zOrder

Functions

- addSubview()
- appendChild()
- convertPointFromParent()
- convertPointFromWindow()
- convertPointToParent()
- convertPointToWindow()
- getElementById()
- orderAbove()
- orderBelow()
- removeChild()
- removeFromParentNode()

removeFromSuperview()
saveImageToFile()

contextMenuItems

array of context menu items

Usage

JavaScript, XML

Description

You can add items to the standard context menu that appears when the user right-clicks the mouse button on your frame. You can also dynamically build your context items by responding to the `contextmenu` event (see onContextMenu for more information).

You specify your items by including an array of `menuItem` objects. See the MenuItem section for more information about menu items.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `Web`, and `Window` objects.

Example

```
<image>
  ...
  <contextMenuItems>
     <menuItem title="Test" onSelect="beep();"/>
     <menuItem title="Another Test"
  onSelect="alert( 'hello' );"/>
  </contextMenuItems>
</image>
```

See the onContextMenu section for an example of building a context menu in JavaScript.

Availability

Available in version 2.0 or newer.

hAlign

horizontal alignment of an object

Usage

JavaScript, XML

Values

`left`
`right`
`center`

Description

The `hAlign` attribute of an object defines the initial horizontal alignment with respect to its `hOffset` attribute. For example, an object with right alignment is drawn so that its right edge appears at the `hOffset`. The default alignment is `left`.

Applies To

Canvas, Frame, Image, Text, TextArea, and ScrollBar objects.

Example

```
<frame hAlign="right"/>

myFrame.hAlign = "left";
```

Availability

Available in version 3.0 or newer.

height

height of the object

Usage

JavaScript, XML

Description

The height attribute controls the vertical dimension of an object. If no height is specified for a frame, its height is determined automatically by the extent of its subviews.

Applies To

Canvas, Flash, Frame, Image, Text, TextArea, ScrollBar, Web, and Window objects.

Example

```
<frame height="300"/>

myFrame.height = 300;
```

Availability

Available in version 1.0 or newer.

hOffset

horizontal offset of an object

Usage

JavaScript, XML

Description

The hOffset attribute of an object defines the horizontal (left to right) offset for the image based on 0,0 being the upper-left corner of the its parent view (superview). The greater the value assigned, the farther to the right the object appears.

Applies To

Canvas, Flash, Frame, Image, Text, TextArea, ScrollBar, Web, and Window objects.

Example

```
<frame hOffset="30"/>

frame.hOffset = 30;
```


Availability

Available in version 1.0 or newer.

id

unique identifier of an object

Usage

JavaScript, XML

Description

The `id` attribute of an object allows you to identify an object uniquely across the entire Widget. This can be used to later find the element using `getElementById`. In the future, it will tie into CSS styling as well.

Applies To

`Canvas`, `Flash`, `Frame`, `HotKey`, `Image`, `Preference`, `Text`, `TextArea`, `Timer`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
<frame id="wicked"/>

frame.id = "wicked";

f = widget.getElementById( "wicked" );
```

Availability

Available in version 4.0 or newer.

firstChild

first child object

Usage

JavaScript (read-only)

Description

The `firstChild` attribute returns the first subobject of a given object, or null if the object has no children. This is useful for walking the DOM hierarchy of a window.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
child = window.firstChild;

while ( child != null )
{
   processChild( child );
   child = child.nextSibling;
}
```

Availability

Available in version 4.0 or newer.

lastChild

last child object

Usage

JavaScript (read-only)

Description

The `lastChild` attribute returns the last subobject of a given object, or null if the object has no children. This is useful for walking the DOM hierarchy of a window.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
// walk backwards...
child = window.lastChild;

while ( child != null )
{
   processChild( child );
   child = child.previousSibling;
}
```

Availability

Available in version 4.0 or newer.

name

JavaScript name for an object

Usage

JavaScript (read-only), XML

Description

The name attribute is a read-only attribute specified in XML. If the name attribute of an object is specified, a JavaScript global variable with that name is created, allowing you to access that object.

Its use in JavaScript is deprecated in favor of using the `id` attribute.

Applies To

`Canvas`, `Flash`, `Frame`, `HotKey`, `Image`, `Preference`, `Text`, `TextArea`, `Timer`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
<frame name="wicked"/>

wicked.lineSize = 10;
```


Availability

Available in version 4.0 or newer.

nextSibling

object immediately after a selected one in the DOM tree

Usage

JavaScript (read-only)

The `nextSibling` attribute returns the next object after a selected object in the DOM tree, or null if the object has no sibling following it. This is useful for walking the DOM hierarchy of a window.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
child = window.firstChild;

while ( child != null )
{
   processChild( child );
   child = child.nextSibling;
}
```

Availability

Available in version 4.0 or newer.

previousSibling

object immediately before a selected one in the DOM tree

Usage

JavaScript (read-only)

Description

The `previous` attribute returns the object immediately preceding a selected object in the DOM tree, or null if the object has no sibling before it. This is useful for walking the DOM hierarchy of a window.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
// walk backwards...
child = window.lastChild;

while ( child != null )
{
   processChild( child );
   child = child.previousSibling;
}
```

Availability

Available in version 4.0 or newer.

onClick

installs handler for click event

For More Information

See click.

onContextMenu

installs handler for contextmenu event

For More Information

See contextmenu.

onDragDrop

installs handler for dragdrop event

For More Information

See dragdrop.

onDragEnter

installs handler for dragenter event

For More Information

See dragenter.

onDragExit

installs handler for dragexit event

For More Information

See dragexit.

onMouseDown

installs handler for mousedown event

For More Information

See mousedown.

onMouseDrag

installs handler for mousedrag event

For More Information

See mousedrag.

onMouseEnter

installs handler for mouseenter event

For More Information

See mouseenter.

onMouseExit

installs handler for mouseexit event

For More Information

See mouseexit.

onMouseMove

installs handler for mousemove event

For More Information

See mousemove.

onMouseUp

installs handler for mouseup event

For More Information

See mouseup.

onMouseWheel

installs handler for mousewheel event

For More Information

See mousewheel.

onMultiClick

installs handler for multiclick event

For More Information

See multiclick.

onTextInput

installs handler for textinput event

For More Information

See textinput.

opacity

opacity of an object

Usage

JavaScript, XML

Description

The `opacity` attribute allows you to specify a value from 0 to 255 that controls the alpha value with which the object is rendered. An opacity of 0 is completely transparent (invisible) and has such side effects as preventing the object from reacting to mouse events. A value of 255 renders the image at its natural opacity. So if your image is already semi-transparent, leave transparency at 255 and it will appear correctly.

You can also use the CSS `opacity` attribute via the `style` attribute. Note that CSS `opacity` takes a value from 0 to 1 instead.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
<image opacity=128"/>

myImage.opacity = 128;
```

Availability

Available in version 1.0 or newer.

parentNode

parent node of an object in the DOM tree

Usage

JavaScript (read-only)

Description

The `parentNode` attribute returns the parent of any node in the DOM tree, or null if the object has no parent. Objects at the root level of a Window return the Window object, for example. Note that this is different than the result of the value of the `superview` attribute (see superview).

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
parent = object.parentNode;
```

Availability

Available in version 4.0 or newer.

rotation

degrees clockwise to rotate object

Usage

JavaScript, XML

Description

The rotation attribute of the object defines by what degree, or fraction of a degree, the object is rotated.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `ScrollBar`, `Web`, and `Web` objects

Example

```
<frame name="frame1" rotation="180"/>

frame1.rotation = 180;
```

style

CSS style information for an object in the DOM tree

Usage

JavaScript, XML

Description

The `style` attribute allows you to alter the CSS style attributes for an object. See the *CSS Reference* for the various attributes you can control through using the `style` attribute.

When using XML, you can supply multiple attributes using a semicolon-separated list of declarations, just like in HTML.

See the "CSS Reference" for the various properties you can control using this attribute.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

Example

```
<frame name="myFrame" style="background-image:url(Sun.png)"/>

myFrame.style.backgroundImage="url(Sun.png)";
```

Availability

Available in version 4.0 or newer.

subviews

child views of an object

Deprecation Notice

Instead of this attribute, you should use the `firstChild` and `nextSibling` attributes to walk the children of an object. It is far more efficient, and more DOM-centric.

Usage

JavaScript (deprecated, read-only)

Description

The `subviews` attribute returns an array of all subviews. In version 4.0 and newer, it is better to walk child objects with `firstChild` and so on. Also, be warned that since this array is built only when needed, if you change the children of a frame and then access this array while in a loop, the performance can be very seriously compromised (N squared, for the geeks out there).

Applies To

`Frame` objects.

Example

```
subs = myFrame.subviews;
```

Availability

Available in version 3.0 or newer.

superview

parent views of an object

Deprecation Notice

Instead of this attribute, you should use the `parentNode` attribute to walk upwards in the DOM tree. It is more DOM-centric.

Usage

JavaScript (deprecated, read-only)

Description

The `subviews` attribute returns the parent view of an object. The ultimate parent of all views is the root view of a window, accessed through `window.root`.

You should instead use `parentNode` as of 4.0 and newer.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Web` objects.

Example

```
dad = myFrame.superview;
```

Availability

Available in version 3.0 or newer.

tooltip

tooltip for an object

Usage

JavaScript, XML

Description

The `tooltip` attribute defines the text displayed in a pop-up tooltip window when the mouse cursor rests over an object.

Applies To

`Canvas`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Window` objects.

JavaScript

`object``.tooltip`

Example

```
<image name="myImage" src="Example.png"
      tooltip="Example tooltip"/>

myImage.tooltip = "Example tooltip";
```

Availability

Available in version 3.0 or newer.

tracking

controls the hit clickable area of an object

Usage

JavaScript, XML

Description

The tracking attribute specifies whether an object's opacity should be used to determine the clickable portions of the image rather than the bounding rectangle. By default transparent parts of an image are not clickable. This can be changed by setting the tracking attribute to rectangle, which makes the entire object's bounds respond to mouse clicks.

Applies To

`Canvas` and `Image` objects.

Example

```
<image name="img1" tracking="rectangle"/>

img1.tracking="rectangle";
```

See Also

Settings

vAlign

vertical alignment of an object

Usage

JavaScript, XML

Values

`top`

```
bottom
center
```

Description

The `vAlign` attribute of an object defines how it is positioned vertically relative to its `vOffset`. For example, an image with a bottom alignment is drawn so that its bottom edge appears at the `vOffset`. If this tag is not specified, the default value is `top`.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `TextArea`, `ScrollBar`, and `Web` objects.

JavaScript

`myObjectName``.vAlign`

Example

```
<frame name = “myFrame” vAlign = “bottom”/>

myFrame.vAlign = "bottom";
```

Availability

Available in version 3.0 or newer.

visible

visibility of an object

Usage

JavaScript, XML

Description

You can set the `visible` attribute of an object to show or hide it by setting it to true or false, respectively. This allows you to hide objects without affecting their opacity, or having to track the current opacity to restore it later. The default visibility for any object if not specified is true.

Setting visibility on a window object will cause the window to fade in or out smoothly. Other objects will simply appear or disappear. To fade in and out other objects, use `FadeAnimation`.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

JavaScript

`myObjectName``.visible`

Example

```
<frame name="myFrame" visible="false"/>

myFrame.visible = true;
```

Availability

Available in version 3.0 or newer.

vOffset

vertical offset of an object

Usage

JavaScript, XML

Description

The `vOffset` attribute defines the vertical (top to bottom) offset for the object based on 0, 0 being the upper-left corner of the object's parent view (superview). The greater the value assigned, the farther down the object appears.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

JavaScript

`object``.vOffset`

Example

```
<frame name="myFrame" vOffset="20"/>

myFrame.vOffset=20;
```

Availability

Available in version 1.0 or newer.

width

width of an object

Usage

JavaScript, XML

Description

The `width` attribute controls the horizontal size of an object. If no width is specified, the width is the ideal width of the object (e.g., the original size of an image, the full width of a piece of text). If the object is a frame and no width is specified, the horizontal extent of its subviews determines its size.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, `Web`, and `Window` objects.

JavaScript

`myObjectName``.width`

Example

```
<image name="myImage" width="300"/>

myImage.width = 300;
```

Availability

Available in version 1.0 or newer.

window

window to which this object belongs

Deprecation Notice

This attribute is deprecated as a setter and you should instead use APIs such as `appendChild` to add an object to a window. In XML, you should simply enclose your object in an appropriate window element.

Usage

JavaScript, XML

Description

You can specify the window an object belongs to by specifying its name in the XML or its variable in JavaScript. If you do not specify a window, the object will remain invisible until you do. In XML, you do not have to specify the window if your object's tag is already inside a window tag.

This is useful for creating customized buttons that respond visually to a user's click.

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Web` objects.

JavaScript

`myObjectName``.window`

Example

```
<window name="fred" width="100" height="100"/>
<frame window="fred"/>

// Or in code
var myWind = new Window();
myFrame.window = myWind;

// You can also specify it in the constructor

var myFrame = new Frame( myWind );
```

Availability

Available in version 2.0 or newer.

zOrder

stacking order of an object

Usage

JavaScript, XML (deprecated)

Description

The `zOrder` attribute defines the stacking order of an object. Objects with a higher `zOrder` are drawn on top of those with a lesser `zOrder`. Normally the `zOrder` is determined by the order in which objects are defined in the XML file with earlier objects being drawn under later ones, but it can also be manipulated using JavaScript at runtime.

Applies To

Canvas, Flash, Frame, Image, Text, TextArea, ScrollBar, and Web objects.

JavaScript

myObjectName.zOrder

Example

```
<frame name="myFrame" zOrder="10"/>

myFrame.zOrder = customZOrder++;
```

Notes

This attribute behaves differently if your Widget's minimumVersion attribute is set to 4.0 or higher. To adjust z-order in that situation, use orderAbove() and orderBelow(). Also, see the section regarding minimumVersion for more information on how z-ordering now behaves.

Availability

Available in version 1.0 or newer.

addSubview()

adds a view to a frame as a subview

Synopsis

```
void Frame.addSubview( object );
```

Applies To

Frame objects.

Description

This function adds an object to a frame. Currently Canvas, Image, Text, TextArea, Frame, and ScrollBar objects can be added to a frame object as a child.

In 4.0 or newer, you should use the more generic appendChild() function.

Example

```
myFrame.addSubview( myImage );
```

Availability

Available in version 3.0 or newer.

appendChild()

adds a child object

Synopsis

```
DOMNode object.appendChild( DOMNode );
```

Applies To

Canvas, Flash, Frame, Image, MenuItem, Text, TextArea, ScrollBar, Web, and Window objects.

Description

This function adds an object to another object as a child. This acts exactly like the DOMNode API documented in "XML Services" and on the W3C web site.

While this is a generic DOM node API, not all objects allow children. If an error occurs, a DOMException is thrown. The node added is returned as the function result.

In 4.0 or newer, you should use this in place of the `addSubview()` function.

Example

```
myFrame.appendChild( myImage );
```

Availability

Available in version 4.0 or newer.

convertPointFromParent()

converts a point from parent coordinates into view coordinates

Synopsis

```
Point object.convertPointFromParent( x, y );
Point object.convertPointFromParent( point );
```

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `ScrollBar`, `Text`, `TextArea`, and `Web` objects.

Description

This function converts a point from the coordinate system of the parent (the containing object) into the coordinate system of the view. There are two variants of this function, one which takes an x and y value and another which takes a point object. Both return a point object with the transformed coordinate. This function is extremely useful when a view is transformed in some way (rotated, scrolled, etc.).

Example

```
var p = myView.convertPointFromParent( 30, 20 );
print( p.x, p.y );
```

Availability

Available in version 4.0 or newer.

See Also

`convertPointFromWindow()`, `convertPointToWindow()`, `convertPointToParent()`

convertPointFromWindow()

converts a point from window coordinates into view coordinates

Synopsis

```
Point object.convertPointFromWindow( x, y );
Point object.convertPointFromWindow( point );
```

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `ScrollBar`, `Text`, `TextArea`, and `Web` objects.

Description

This function converts a point from the coordinate system of the containing window into the coordinate system of the view. There are two variants of this function, one which takes an x and y value and another which takes a point object. Both return a point object with the transformed coordinate. This function is extremely useful when a view is transformed in some way (rotated, scrolled, etc.).

Example

```
var p = myView.convertPointFromWindow( event.hOffset,
                    event.vOfffset );
print( p.x, p.y );
```

Availability

Available in version 4.0 or newer.

See Also

convertPointFromParent(), convertPointToParent(), convertPointToWindow()

convertPointToParent()

converts a point from view coordinates into parent coordinates

Synopsis

```
Point object.convertPointToParent( x, y );
Point object.convertPointToParent( point );
```

Applies To

Canvas, Flash, Frame, Image, ScrollBar, Text, TextArea, and Web objects.

Description

This function converts a point from the coordinate system of the given view into that of its parent (e.g., its containing frame). There are two variants of this function, one which takes an x and y value and another which takes a point object. Both return a point object with the transformed coordinate. This function is extremely useful when a view is transformed in some way (rotated, scrolled, etc.).

Example

```
var p = myView.convertPointToParent( event.x, event.y );
print( p.x, p.y );
```

Availability

Available in version 4.0 or newer.

See Also

convertPointFromParent(), convertPointFromWindow(), convertPointToWindow()

convertPointToWindow()

converts a point from view coordinates into window coordinates

Synopsis

```
Point object.convertPointToWindow( x, y );
Point object.convertPointToWindow( point );
```

Applies To

Canvas, Flash, Frame, Image, ScrollBar, Text, TextArea, and Web objects.

Description

This function converts a point from the coordinate system of the view to the coordinate system of the containing window. There are two variants of this function, one which takes an x and y value and another which takes a point object. Both return a point object with the transformed coordinate. This function is extremely useful when a view is transformed in some way (rotated, scrolled, etc.).

One example of using this is with the popupMenu() function, which takes window coordinates. You can use view coordinates you receive in a mouse event and convert those into window coordinates easily with this function.

Example

```
var p = myView.convertPointToWindow( event.x, event.y );
print( p.x, p.y );
```

Availability

Available in version 4.0 or newer.

See Also

convertPointFromWindow(), convertPointFromParent(), convertPointToParent()

getElementById()

finds an element in the document by ID

Synopsis

```
DOMNode object.getElementById( id );
```

Applies To

Frame, Window, and Widget objects.

Description

This function allows you to locate any item in the document by its ID. The ID of an object should be unique throughout the entire Widget document. At present, you can only call this on a Frame, Window, or the global Widget object.

Example

```
img = widget.getElementById( "fiery" );
```

Availability

Available in version 4.0 or newer.

orderAbove()

moves an object above another in the z-order

Synopsis

```
void object.orderAbove( object | null );
```


Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Web` objects.

Description

This function allows you to change an object's z-order and bring it above another view, or to the top of all its sibling views. Passing a sibling view moves this object above the given view. Passing null brings the view to the top of all of its siblings.

It's important to understand that z-order shifting is only relative to sibling views, i.e., you cannot move a view in front of one of its parent's siblings.

This function only operates within a `zOrder` level. So you cannot order a view with `zOrder` 0 above a view with `zOrder` 1. Likewise, you cannot order a view with `zOrder` 1 behind a view with `zOrder` 0.

Example

```
myFrame.orderAbove( myImage );
myOtherFrame.orderAbove( null );
```

Availability

Available in version 4.0 or newer.

orderBelow()

move an object below another in the z-order

Synopsis

```
void object.orderBelow( object | null );
```

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Web` objects.

Description

This function allows you to change an object's z-order and move it below another view, or below of all its sibling views. Passing a sibling view moves this object below the given view. Passing null brings the view to the bottom of all of its siblings (i.e., "send to back").

It's important to understand that z-order shifting is only ever relative to sibling views, i.e., you cannot move a view behind of one of its parent's siblings.

This function only operates within a `zOrder` level. So you cannot order a view with `zOrder` zero above a view with `zOrder` 1. Likewise, you cannot order a view with `zOrder` 1 behind a view with `zOrder` 0.

Example

```
myFrame.orderBelow( myImage );
myOtherFrame.orderBelow( null );
```

Availability

Available in version 4.0 or newer.

removeChild()

removes a child object

Synopsis

```
DOMNode object.removeChild( DOMNode );
```

Applies To

Canvas, Flash, Frame, Image, MenuItem, Text, TextArea, ScrollBar, Web and Window objects.

Description

This function removes a child from a given parent node. This acts exactly like the DOMNode API as documented later in this reference and on the W3C web site.

While this is a generic DOM node API, not all objects allow children. If an error occurs, a DOMException is thrown. The node removed is returned as the function result.

In 4.0 or newer, you can use this in place of the `removeFromSuperview()` function.

Example

```
myFrame.removeChild( notWanted );
```

Availability

Available in version 4.0 or newer.

removeFromParentNode()

detaches an object from its parent

Synopsis

```
void object.removeFromParentNode()
```

Applies To

All objects.

Description

This function behaves like removeFromSuperview, but is more generic because it applies to any DOM node. It allows you to easily detach a node from its parent node if it has one.

Example

```
myObject.removeFromParentNode();
```

Availability

Available in version 4.5 or newer.

removeFromSuperview()

detaches an object from its parent view

Synopsis

```
void object.removeFromSuperview()
```

Applies To

Canvas, Flash, Frame, Image, Text, TextArea, ScrollBar, and Web objects.

Description

This method removes an object from a window. You might do this because you are done with the object and are reloading new information.

When your Widget's `minimumVersion` is set to 3.0, you must call this to remove an object from a window. Deleting the reference will not work.

This has been deprecated in favor of `removeChild()`.

Example

```
myObject.removeFromSuperview();
```

Availability

Available in version 3.0 or newer.

saveImageToFile()

writes a bitmap of an object to a file

Synopsis

```
void object.saveImageToFile( path, type )
```

Applies To

`Canvas`, `Flash`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Web` objects.

Description

Use this method to write out an image file with the contents of an object as it appears on screen. For example, if you took an image object, made its size larger and told it to tile, this function would write an image of that tiled image to disk, or you could use this to resize an image an write a thumbnail to disk.

You can write the image as a JPEG or PNG file. PNG files will get whatever transparency your object has. You must pass jpeg or png as the second parameter.

Example

```
myImage.saveImageToFile( system.widgetDataFolder + "/myImage.png", "png" );
```

Availability

Available in version 4.0 or newer.

About Box

block to define images for an about box

XML Name

`<about-box>`

JavaScript Name

Not available.

Attributes

`about-image`
`about-text`
`about-version`

Description

If used, the `about-box` XML block must contain one or more references to a path to an image contained in an image block.

The about box option is not currently accessible through JavaScript.

about-image

block containing a path to an image

Usage

XML

Description

The `image` attribute of the `about-box` block must contain a valid path to an image.

If more than one `image` attribute is used, the images are shown sequentially to the user. When they are the same size, they simply replace each other; when they are different sizes, the first fades out and the next fades in.

Example

```
<about-box>
      <about-image>Resources/About.png</about-image>
      <about-image>Resources/Thanks.png</about-image>
</about-box>
```

Availability

Available in version 2.1 or newer.

about-text

text to display

Usage

XML

Description

You can specify any number of text objects to be displayed in your about box. These text items at present only appear on the first page of your about box.

Attributes

`color`
`data`
`hOffset`
`font`
`size`
`style`
`shadow`
`url`
`vOffset`

Except for `shadow` and `url`, these are all the same attributes as can be used on a full-fledged text object. See the section on Text objects for information on how these attributes are used. See the section on Shadow objects for information about how those objects are structured. The `url` attribute turns the text object into a clickable link that opens a browser targeted at the URL you specify.

Availability

Available in version 2.1 or newer. The `url` attribute is available in 3.0 or newer.

about-version

element to describe where and how the version should be placed

Usage

XML

Description

This is a special case of the text element, described above. It has all the same attributes, and can only be placed on the first page of an about box. The only difference is that this tag represents where the Widget's version number will appear. The version number is taken from the Widget definition's `version` attribute.

Availability

Available in version 2.1 or newer.

Action

code block not associated with an object

XML Name

`<action>`

JavaScript Name

Not directly available (see below).

Attributes

`file`
`interval`
`trigger`

Description

The `action` XML block defines when and how a Widget executes code that is triggered automatically rather than by a user.

Actions are accessible in JavaScript as properties of the Widget object. Instead of producing an array of actions, in version 4.0 and newer you can reference actions by getting and setting them on the `widget` global object. For example, you can set your Widget's `onUnload` handler during runtime simply by using the trigger name:

```
widget.onUnload = myShutdownFunction;
```

See the Widget object for more information on these triggers.

file

path to an external JavaScript file

Description

Embedding JavaScript code into an XML file might present unique problems for some developers. Your preferred text editor might not gracefully support syntax highlighting for both XML and JavaScript at the same time, your JavaScript code might be large and complex and need better management, or you might just be frustrated by the impositions of having to escape common characters that would confuse the XML portion of the Widget. To alleviate any or all of these, we allow you to reference an external file.

You can reference files by specifying the `file` attribute for the `<action>` block. Alternatively, you can simply use `include()`.

Example

```
<action trigger="onLoad" file="main.js"/>
<action trigger="onLoad">
    include( "main.js" );
</action>
```

interval

time in seconds to wait between triggers

Description

The `interval` attribute for the action block is to be used with the `onTimer` trigger attribute. It defines how many seconds, or fractions of a second, to wait between `onTimer` code executions.

If no interval is defined for an on timer trigger, it defaults to one minute.

Example

```
<!-- This will cause the Widget to beep every
     two minutes -->
<action trigger="onTimer" interval="120">
  beep();
</action>
<!-- This will cause the counter to increase ten
     times a second -->
<action trigger="onTimer" interval="0.1">
  counter ++;
</action>
```

Starting in version 2.0, this mechanism is deprecated in favor of the new `Timer` objects (see the section on Timer objects later in this manual).

trigger

event that triggers the enclosed code

Values

```
onGainFocus
onKeyDown
onKeyUp
onKonsposeActivated
onKonsposeDeactivated
onLoad
onLoseFocus
```


```
onMouseDown
onMouseEnter
onMouseExit
onMouseUp
onPreferencesCancelled
onPreferencesChanged
onRunCommandInBgComplete
onScreenChanged
onTellWidget
onTimer
onUnload
onWakeFromSleep
onWillChangePreferences
onYahooLoginChanged
```

Description

The `trigger` attribute for the action block defines what triggers the contained block of code.

In version 4.0 and newer, you can change what gets executed by these triggers using the `widget` global object. That object now has a set of attributes with the same names as these triggers. See the `widget` object for more information.

`onGainFocus` triggers when the Widget is activated by the user. This is useful if you want your Widget to have an active and inactive state. This action is typically triggered when the Widget first starts running. In version 2.0 and newer, you should generally use the `onGainFocus` handler on each window and reserve the Widget `onGainFocus` handler for truly Widget-wide activation handling.

`onKonsposeActivated` and `onKonsposeDeactivated` execute when the user invokes and dismisses Heads-up Display (formerly Konsposé) mode. This gives the Widget the opportunity to change display modes or take other actions (for example, some Widgets display their "focused" mode as if `onGainFocus` had been received when Heads-up Display is active).

`onLoad` executes when the Widget is first loaded and is used to define and store functions that might be used elsewhere in the Widget.

`onLoseFocus` triggers when the Widget is deactivated by the user. This is useful if you want your Widget to have an active and inactive state. In version 2.0 and newer, you should generally use the `onLoseFocus` handler on each window and reserve the Widget `onLoseFocus` handler for truly Widget-wide activation handling.

`onPreferencesCancelled` is executed when the user cancels out of the preferences dialog.

`onPreferencesChanged` is executed when the user saves the preferences. Note that nothing is executed if the user cancels out of the preferences dialog as they didn't change the preferences in that case.

`onRunCommandInBgComplete` is executed when a command started with `runCommandInBg()` completes (see onRunCommandInBgComplete).

`onScreenChanged` fires if any screen size, arrangement, or color depth changes are made using the Displays System Preference panel (note that the screen the Widget itself is on might have been affected).

`onTellWidget` is called when another Widget or application calls the `tellWidget` interface to send your Widget a message. You should be careful about what you decide to do with the message you receive. See tellWidget() for more detail. This trigger is available in Widget Engine 2.0 or newer.

`onTimer` has been deprecated in favor of Timer objects. See "Timer" for more information.

`onUnload` executes when the Widget is closed. This is useful for doing any last minute manual preference saving (preferences set in the Widget Preferences dialog are saved automatically when they are changed by the user), as well as making sure any external applications you are talking to are closed up and aware of your departure. Note that you should not perform any lengthy operations in this trigger as Widgets are encouraged to shut down quickly (an example of a lengthy operation would be retrieving something from the network).

`onWakeFromSleep` executes when the computer wakes from a state of sleep. It should be noted that some desktops have a several second lag between waking up and reconnecting to the network, so you might want to add a `sleep()` call to your code if your Widget wants to connect to the Internet. In version 3.0 or newer, timers are stopped when the computer goes to sleep and are not restarted until `onWakeFromSleep` is called.

`onWillChangePreferences` executes when the user asks to edit the Widget's preferences (or when the `showWidgetPreferences()` JavaScript call is made).

`onYahooLoginChanged` executes when the user logs in or out of their Yahoo! account. When called, you can check the current state of the user login by calling `yahooCheckLogin()`.

The remaining triggers `onKeyDown`, `onKeyUp`, `onMouseDown`, `onMouseUp`, `onMouseEnter`, and `onMouseExit` execute when the corresponding user action is detected within the main window of the active Widget, and there is no other object to receive them. Note that using the global scope mouse actions causes your Widget to no longer be draggable without having to hold down the command key.

Example

```
<!– Redraw the clock when we wake from sleep -->
<action trigger="onWakeFromSleep">
   updateClockFace();
</action>
<!– Update our info when the user changes the preferences -->
<action trigger="onPreferencesChanged">
   refreshTickerSymbols();
</action>
```

Canvas

free-form drawing into a bitmap

XML Name

`<canvas>`

JavaScript Name

`Canvas`

Attributes

`getContext()`

Description

The `Canvas` object allows free-form vector drawing into a bitmap you control. Currently, only 2D graphics are possible. `Canvas` follows the same API as that provided by Safari and Firefox. Code written for those environments should also work in a Widget. In general, we will match Firefox's rendering since we use Cairo to draw in a `Canvas` object.

Our implementation of `Canvas` does not feature the `toDataURL` method available in Firefox. If you wish to save the image you have created to a file, you can use the general `saveImageToFile` method of the engine.

Example

```
// in XML
<canvas id="test" width="200" height="200" vOffset="10"
        hOffset="10"/>

// Then later, in Javascript
var c = widget.getElementById( "test" );

// or create right in Javascript...
var c = new Canvas();
c.width = c.height = 200;
c.hOffset = c.vOffset = 10;
```

Attributes

getContext()

gets a drawing context for a canvas

Synopsis

```
context = canvas.getContext( type );
```

Description

This function returns a drawing context that you use to draw things in a `Canvas` object's bitmap. Currently, the only type of context that can be used is a 2D context. This means that at present, the only parameter that is valid is `2d`.

See the section on CanvasRenderingContext2D for the different drawing APIs that are available.

Example

```
c = widget.getElementById( "myCanvas" );

ctx = c.getContext( "2d" );
```

Availability

Available in version 4.0 or newer.

CanvasRenderingContext2D

object used to draw into a canvas object

XML Name

Not available in XML.

JavaScript Name

`CanvasRenderingContext2D`

Attributes

`fillStyle`
`globalAlpha`
`globalCompositeOperation`
`lineCap`
`lineJoin`
`lineWidth`

miterLimit
strokeStyle

Functions

addColorStop()
arc()
beginPath()
bezierCurveTo()
clearRect()
clip()
closePath()
createLinearGradient()
createPattern()
createRadialGradient()
drawImage()
fill()
fillRect()
lineTo()
moveTo()
quadraticCurveTo()
rect()
restore()
rotate()
save()
scale()
stroke()
strokeRect()
translate()

Description

This object can only be created through a call to `canvas.getContext( "2d" )`.

You use this object to do the actual drawing into the context.

Example

```
// in XML
<canvas id="test" width="200" height="200" vOffset="10"
        hOffset="10"/>

// Then later, in Javascript
var c = widget.getElementById( "test" );

var ctx = c.getContext( "2d" );
ctx.clearRect( 0, 0, 200, 200 );
```

fillStyle

color or pattern to use to fill an area of the canvas

Description

You can set a color or pattern to use when filling paths in the canvas drawing context. You can use a simple color expressed as #FFFFFF (or the shorter #FFF), as well as a gradient or pattern object. If you wish to draw something semi-transparent, you can specify transparency using `rgba()` colors. The last parameter of `rgba` is a floating-point number from 0 to 1.0 indicating the amount of transparency, where 1.0 is fully opaque.

Example

```
// standard color
ctx.fillStyle = "#FFF"; // or "#FFFFFF"

// with alpha
ctx.fillStyle = "rgba( 255, 255, 255, 0.5)";

// gradient
grad = ctx.createLinearGradient( 0, 50, 0, 95 );
lingrad2.addColorStop( 0.5, "#000" );
lingrad2.addColorStop( 1, "rgba( 0,0,0,0 )" );
ctx.fillStyle = grad;
```

Availability

Available in version 4.0 or newer.

globalAlpha

controls the overall alpha to draw with

Description

You can set the `globalAlpha` attribute to affect all drawing calls that follow it. This global alpha setting is multiplied with any alpha you might be drawing with. For example, if you set the global alpha to 0.5 and then fill a path with a color that has an alpha of 0.5, the effective alpha is 0.25. In other words, it's as if you adjusted the opacity of the entire `Canvas` object to 0.5.

Example

```
// half transparent
ctx.globalAlpha = 0.5;
```

Availability

Available in version 4.0 or newer.

globalCompositeOperation

compositing mode of the canvas

Description

You can control how images are composited onto the canvas by setting the `globalCompositeOperation` attribute. The default compositing mode is `source-over`. This means that as new items are drawn, they are blended over the existing contents. There are many different compositing modes:

```
copy
darker
destination-atop
destination-in
destination-out
destination-over
lighter
source-atop
source-out
source-over
xor
```

Example

```
// draw behind what's there.
ctx.globalCompositeOperation = "destination-over";
```

Availability

Available in version 4.0 or newer.

lineCap

style of line cap to use

Values

`butt`
`round`
`square`

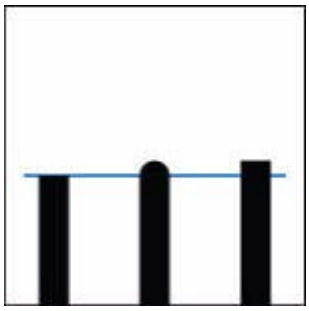
Description

You can control how the end caps of lines are displayed by setting the `lineCap` attribute. The default value is `butt`.

Example

```
ctx.lineCap = "round";
```

The following image shows butt, round, and square ends, respectively.



Availability

Available in version 4.0 or newer.

lineJoin

style to use when joining lines together

Values

`round`
`bevel`
`miter`

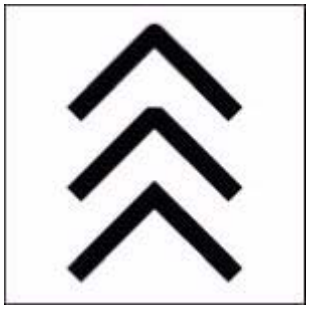
Description

You can control how lines are joined together by setting the `lineJoin` attribute. The default value is `miter`.

Example

```
ctx.lineJoin = "round";
```


The following image shows round, bevel, and miter joins, respectively.



Availability

Available in version 4.0 or newer.

lineWidth

width to use when drawing lines

Description

This attribute allows you to set the width of lines as they are drawn.

Example

```
ctx.lineWidth = 2.0;
```

Availability

Available in version 4.0 or newer.

miterLimit

controls how the miter line join deals with angles

Description

This attribute allows you to control the point at which what would normally be a miter join turns into a bevel join. This is used for angles that are acute, where the intersection of the lines would form a very long miter otherwise. The drawing system divides the length of the miter by the line width. If the result is greater than the miter limit, the style is converted to a bevel. The default value for this attribute is 10.0.

Example

```
ctx.miterLimit = 2.0;
```

Availability

Available in version 4.0 or newer.

strokeStyle

color or pattern to use to stroke a path

Description

You can set a color or pattern to use when stroking paths in the canvas. You can use a simple color expressed as #FFFFFF (or the shorter #FFF), as well as a gradient or pattern object. If you wish to use a color with alpha, you can use `rgba()` notation as show below. The last parameter of `rgba` is a floating point number from 0 to 1.0 indicating the amount of transparency, where 1.0 is fully opaque.

Example

```
// standard color
ctx.strokeStyle = "#FFF"; // or "#FFFFFF"

// with alpha
ctx.strokeStyle = "rgba( 255, 255, 255, 0.5 )";

// gradient
grad = ctx.createLinearGradient( 0, 50, 0, 95 );
lingrad2.addColorStop( 0.5, "#000" );
lingrad2.addColorStop( 1, "rgba( 0,0,0,0 )" );
ctx.strokeStyle = grad;
```

Availability

Available in version 4.0 or newer.

addColorStop()

adds a color stop to a linear or radial gradient

Synopsis

```
gradient.addColorStop( offset, color );
```

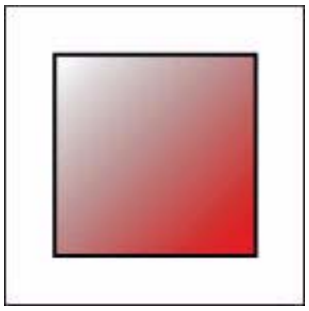
Description

This function is used on linear and radial gradient objects created with `createLinearGradient()` and `createRadialGradient()`. It controls the color ramp of the gradient—what each color is along the way and where they start and end. The offset parameter is a float between 0.0 and 1.0.

The simplest ramp would be to have two stops at offsets 0 and 1, with the color at 0 being one color and the color at another color.

Example

```
// go from transparent black to opaque red
grad = ctx.createLinearGradient( 25, 25, 125, 125 );
grad.addColorStop( 0, “rgba( 0, 0, 0, 0 )” );
grad.addColorStop( 1, “rgba( 255, 0, 0, 1 )” );

ctx.fillStyle=grad;
ctx.fillRect( 25, 25, 100, 100 );
ctx.strokeRect( 25, 25, 100, 100 );
```



Availability

Available in version 4.0 or newer.

arc()

adds an arc to the current path

Synopsis

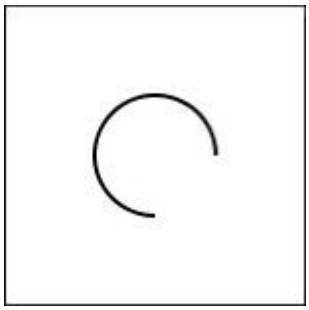
```
context.arc(x, y, radius, startAngle, endAngle, anticlockwise);
```

Description

This method adds an arc of a circle to the current path. x and y are the coordintes of the arc's center point. The arc's starting point is the distance of the radius from the arc's center point along the `startAngle`. If the current path already has a subpath, a straight line is drawn from the end point of that segment to the starting point of the arc. The clockwise parameter indicates the direction the arc should be drawn. Angles are expressed in radians, and 0 starts on the X axis.

Example

```
// arcing 90 degrees, but counter-clockwise
ctx.beginPath();
ctx.arc( 75, 75, 30, 0, Math.PI / 2, true );
ctx.stroke();
```



Availability

Available in version 4.0 or newer.

beginPath()

starts a new path

Synopsis

```
context.beginPath();
```

Description

To draw arbitrary shapes, you first create a path using `beginPath()`. Then you add segments (subpaths) to it with APIs such as `arc()` or `rect()`. When you are done with a path, you can close it with `closePath()`, or stroke or fill the current path with `stroke()` and `fill()`, respectively.

Example

```
ctx.beginPath();
ctx.rect( 0, 0, 100, 100 );
ctx.stroke();
```

Availability

Available in version 4.0 or newer.

bezierCurveTo()

adds a bezier curve to the current path

Synopsis

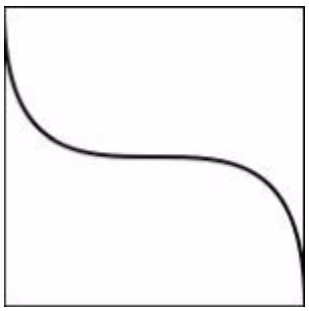
```
context.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y );
```

Description

This method adds a bezier curve from the current point in the current subpath to x, y. The curve is affected by the two control points given.

Example

```
// draw a large, diagonal S-curve
ctx.beginPath();
ctx.moveTo( 0, 0 );
ctx.bezierCurveTo( 0, 150, 150, 0, 150, 150 );
ctx.stroke();
```



Availability

Available in version 4.0 or newer.

clearRect()

clears an area of the canvas

Synopsis

```
context.clearRect( x, y, width, height );
```

Description

This method clears the given rectangle by filling the rectangle with transparent pixels. This can be used to completely start over and redraw a portion of the canvas. It is equivalent to setting the composite mode to "copy" and then filling a rectangle with a color of `rgba(0,0,0,0)`.

Example

```
ctx.clearRect( 10, 10, 75, 75 );
```

Availability

Available in version 4.0 or newer.

clip()

clips the current path

Synopsis

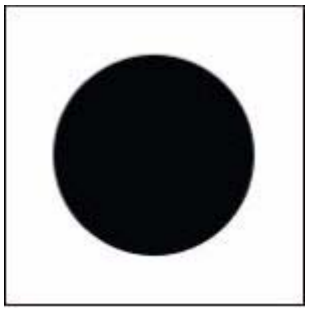
```
context.clip();
```

Description

This function clips the drawing in the context to the current path, using the nonzero winding-number rule. Clipping a context always intersects with the current clip. If you wish to restore the clip to a previous state, you must save and restore the graphics state as needed.

Example

```
// fill a circle the hard way, clip to a
// circular arc, then fill the entire canvas.
ctx.beginPath();
ctx.arc( 75, 75, 50, 0, Math.PI*2, true );
ctx.clip();
ctx.fillRect( 0, 0, 150, 150 );
```



Availability

Available in version 4.0 or newer.

closePath()

closes the current path

Synopsis

```
context.closePath();
```

Description

This method closes any open subpath by drawing a straight line from the current path back to the starting point of the path. Open paths are implicitly closed if you call `fill()` or `clip()`. You can call `stroke()` without closing the path, however.

Example

```
ctx.closePath();
```

Availability

Available in version 4.0 or newer.

createLinearGradient()

creates a linear gradient

Synopsis

```
context.createLinearGradient( x0, y0, x1, y1 );
```

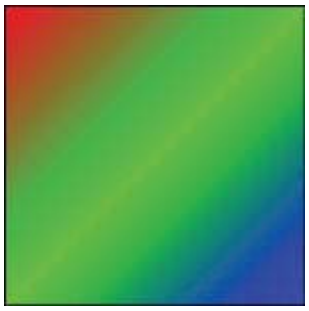
Description

This method creates a new linear gradient that gradates between the two points given. It is important to realize that the gradient will continue beyond those points as well when you use it to fill or stroke an area of the canvas.

After creating a gradient, you need to assign color stops to tell it what colors it will be blending along the way. See `addColorStop()` for more information.

Example

```
grad = ctx.createLinearGradient( 0, 0, 150, 150 );
grad.addColorStop( 0, “#F00” );
grad.addColorStop( 0.5, “#0F0” );
grad.addColorStop( 1, “#00F” );
ctx.fillStyle = grad;
ctx.fillRect( 0, 0, 150, 150 );
```



Availability

Available in version 4.0 or newer.

createPattern()

creates a pattern object

Synopsis

```
context.createPattern( image, repeat );
```

Values

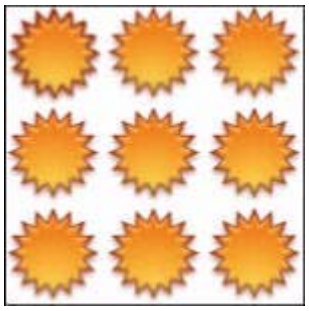
```
repeat
repeat-x
repeat-y
no-repeat
```

Description

This method creates a new pattern based on the image you provide. You can control how the pattern repeats with the `repeat` parameter. The values are the same as those used by CSS background-repeat.

Example

```
patt = ctx.createPattern( myImage, "repeat" );
ctx.fillStyle = patt;
ctx.fillRect( 0, 0, 150, 150 );
```



Availability

Available in version 4.0 or newer.

createRadialGradient()

creates a radial gradient

Synopsis

```
context.createRadialGradient( x0, y0, r0, x1, y1, r1 );
```

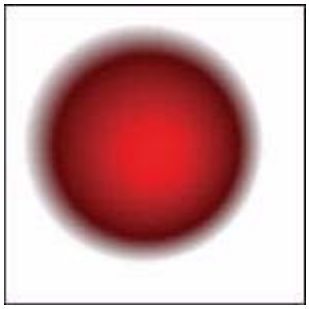
Description

This method creates a new radial gradient that gradates between the two given points. It is important to realize that the gradient will continue beyond those points as well when you use it to fill or stroke an area of the canvas.

After creating a gradient, you need to assign color stops to tell it what colors it will be blending along the way. See addColorStop() for more information.

Example

```
radgrad = ctx.createRadialGradient( 75, 75, 10, 70, 70, 40 );
radgrad.addColorStop( 0, "#F00" );
radgrad.addColorStop( 0.7, "#600" );
radgrad.addColorStop( 1, "rgba( 33, 0, 0, 0 )" );
ctx.fillStyle = radgrad;
ctx.fillRect( 0, 0, 150, 150 );
```



Availability

Available in version 4.0 or newer.

drawImage()

draws an image

Synopsis

```
context.drawImage( image, x, y );
context.drawImage( image, x, y, width, height );
context.drawImage( image, srcX, srcY, srcWidth, srcHeight,
     dstX, dstY, dstWidth, dstHeight );
```

Description

This function allows you to draw an image into a canvas. There are three variants, each with a different level of control. The first variant draws the image at its full size at the location specified. The second variant allows you to draw the image at a specific location and size. The third variant allows you to render just a specific portion of an image at a specified location and size.

In version 4.5 or later, this function also accepts a `Canvas` object as well as an `Image`.

Example

```
ctx.drawImage( myImage, 0, 0, 300, 300 );
```

Availability

Available in version 4.0 or newer.

fill()

fills the current path

Synopsis

```
context.fill();
```

Description

This method fills the current path with the current `fillStyle` setting. It closes any currently open subpath by connecting the current point to the starting point of the subpath with a straight line.

Example

```
ctx.beginPath();
ctx.fillStyle = "#00B";
ctx.rect( 10, 10, 75, 75 );
ctx.fill();
```


Availability

Available in version 4.0 or newer.

fillRect()

fills a rectangle with the current fill style

Synopsis

```
context.fillRect( x, y, width, height );
```

Description

This method fills the given rectangle with the current `fillStyle`. Any current path that might be open is closed and ignored.

Example

```
ctx.fillStyle = "#00B";
ctx.fillrect( 10, 10, 75, 75 );
```

Availability

Available in version 4.0 or newer.

lineTo()

adds a line segment to the current path

Synopsis

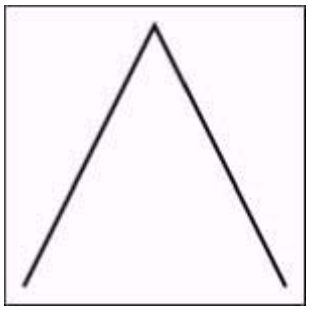
```
context.lineTo( x, y );
```

Description

This method adds a line segment from the current point to the point specified. This point becomes the current point. If no current point has been specified yet in the current path, this function sets the current point to the point specified.

Example

```
ctx.beginPath();
ctx.moveTo( 10, 140 );
ctx.lineTo( 75, 10 );
ctx.lineTo( 140, 140 );
ctx.stroke();
```



Availability

Available in version 4.0 or newer.

moveTo()

begins a new subpath in the current path

Synopsis

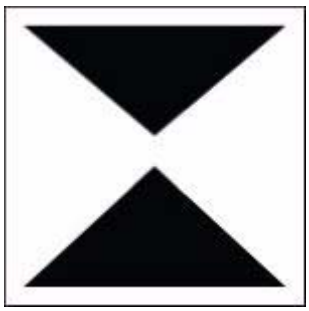
```
context.moveTo( x, y );
```

Description

This method closes any previous subpath and starts a new one. You can use this to create what appear to be multiple shapes that are part of the same path.

Example

```
// create two triangles and fill them
ctx.beginPath();
ctx.moveTo( 10, 10 );
ctx.lineTo( 75, 65 );
ctx.lineTo( 140, 10 );
ctx.moveTo( 10, 140 );
ctx.lineTo( 75, 80 );
ctx.lineTo( 140, 140 );
ctx.fill();
```



Availability

Available in version 4.0 or newer.

quadraticCurveTo()

adds a quadratic curve to the current path

Synopsis

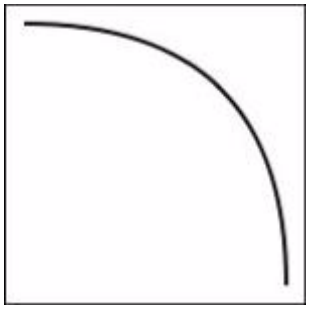
```
context.quadraticCurveTo( cpx, cpy, x, y );
```

Description

This method adds a quadratic curve that travels from the current point to the x and y positions specified. The control point specified by `cpx` and `cpy` adjusts the curve.

Example

```
ctx.beginPath();
ctx.moveTo( 10, 10 );
ctx.quadraticCurveTo( 140, 10, 140, 140 );
ctx.stroke();
```

Availability

Available in version 4.0 or newer.

rect()

adds a rectangle to the current path

Synopsis

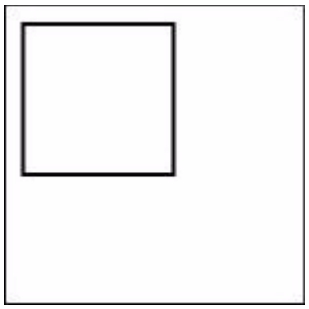
```
context.rect( x, y, width, height );
```

Description

This method adds a rectangle to the current path as a complete subpath.

Example

```
ctx.beginPath();
ctx.lineWidth = 2;
ctx.rect( 10, 10, 75, 75 );
ctx.stroke();
```



Availability

Available in version 4.0 or newer.

restore()

restores a previously saved graphics state

Synopsis

```
context.restore();
```

Description

This method restores the most recent graphics state saved by `save()`. See the save() parameter for more information.

Example

```
ctx.restore();
```

Availability

Available in version 4.0 or newer.

rotate()

applies a rotation transformation

Synopsis

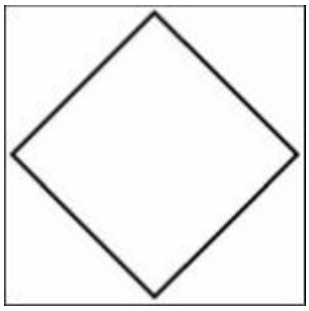
```
context.rotate( angleInRadians );
```

Description

This method applies a rotation to the current transformation matrix (CTM). The angle is specified in radians. All drawing is affected by this transformation. To restore the transformation to normal, it is necessary to save the graphics state before rotating and then restore it when you no longer need it.

Example

```
rad = 45 * .01745329252; // 45 degrees
ctx.save();
ctx.translate( 75, 75 );
ctx.rotate( rad );
ctx.strokeRect( -50, -50, 100, 100 );
ctx.restore();
```



Availability

Available in version 4.0 or newer.

save()

saves the current graphics state

Synopsis

```
context.save();
```


Description

This method saves the current graphics state onto a stack. This allows you to make changes to various settings in the drawing context and then restore the settings to what you had previously. Calls to save must be balanced by calls to `restore()`.

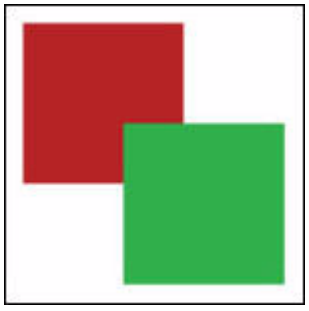
The settings that are saved as part of the graphics state are:

```
translation matrix (CTM)
clip
line cap
line width
line join
miter limit
fill and stroke styles
global alpha
```

Note: The current path is not saved.

Example

```
ctx.fillStyle = "#0B0";
ctx.save();
ctx.fillStyle = "#B00";
ctx.fillRect( 10, 10, 80, 80 );
ctx.restore();
// now the fill style is green again.
ctx.fillRect( 60, 60, 80, 80 );
```



Availability

Available in version 4.0 or newer.

scale()

scales the current graphics state

Synopsis

```
context.scale( sx, sy );
```

Description

This method scales the current transformation matrix of the context. In the example below, we scale the drawing to five times normal.

As with rotation or translation, if you wish to restore the state, you should use `save()` and `restore()` as appropriate.

Example

```
ctx.scale( 5, 5 );

grad = ctx.createLinearGradient( 5, 5, 100, 100 );
grad.addColorStop( 0, “rgba( 255, 0, 0, 0 )” );
grad.addColorStop( 1, “rgba( 255, 0, 0, 1 )” );

ctx.fillStyle = grad;
ctx.fillRect( 5, 5, 100, 100 );
ctx.strokeRect( 5, 5, 100, 100 );
```

Availability

Available in version 4.0 or newer.

stroke()

strokes the current path

Synopsis

```
context.stroke();
```

Description

This method strokes the current path with the current `strokeStyle`, `lineWidth`, `lineCap`, `lineJoin`, and `miterLimit` settings. It does not close any currently open subpath.

Example

```
ctx.beginPath();
ctx.lineWidth = 2;
ctx.rect( 10, 10, 75, 75 );
ctx.stroke();
```

Availability

Available in version 4.0 or newer.

strokeRect()

strokes a rectangle with the current stroke style

Synopsis

```
context.strokeRect( x, y, width, height );
```


Description

This method strokes the given rectangle with the current `strokeStyle`, `lineWidth`, `lineCap`, `lineJoin`, and `miterLimit` settings. Any current path that might be open when this is called is closed and ignored.

Example

```
ctx.lineWidth = 2;
ctx.strokeRect( 10, 10, 75, 75 );
```

Availability

Available in version 4.0 or newer.

translate()

translates the transformation matrix

Synopsis

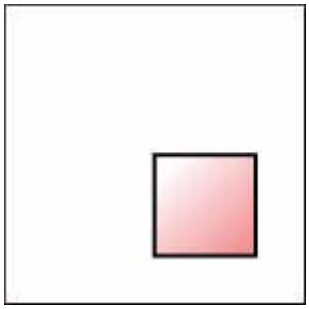
```
context.translate( dx, dy );
```

Description

This method translates the current transformation matrix by the distances given. This effectively moves the origin of the coordinate system you are drawing with.

Example

```
ctx.translate( 75, 75 );

grad = ctx.createLinearGradient( 5, 5, 100, 100 );
grad.addColorStop( 0, “rgba( 255, 0, 0, 0 )” );
grad.addColorStop( 1, “rgba( 255, 0, 0, 1 )” );

ctx.lineWidth = 2;
ctx.fillStyle=grad;
ctx.fillRect( 0, 0, 50, 50 );
ctx.strokeRect( 0, 0, 50, 50 );
```



Availability

Available in version 4.0 or newer.

Flash

block to define a flash object

Common Attributes

hAlign
height
hRegistrationPoint
name
rotation
vAlign
vRegistrationPoint
width

Attributes

allowNetworking
base
bgColor
deviceFont
frameNumber
useFlashContextMenu
flashVars
loop
minVersion
onFsCommand
onFsReadyState
quality
sAlign
scale
src
wMode

Functions

back()
forward()
getVariable()
gotoFrame()
isPlaying()
loadMovie()
pan()
percentLoaded()
play()
reload()
rewind()
setVariable()
setZoomRect()
stop()
stopPlay()
tCallFrame()
tCallLabel()
tCurrentFrame()
tCurrentLabel()
tGotoFrame()
tGotoLabel()
totalFrames()
tGetProperty()

tGetPropertyAsNumber()
tGetPropertyNum()
tSetProperty()
tSetPropertyNum()
tStopPlay()
tPlay()
version()
zoom()

Synopsis

flash

Description

The functions/attributes/events that are in **bold** and *italics* are those that are Flash OCX, and are features of Macromedia flash that are exposed to assist the Widget Developer in doing what they normally can do from a webpage.

Example

```
<widget minimumVersion="4.5">
     <window name="mainWindow" width="500" height="500"
                                        style="background-color:#ffeeaa">
          <flash name="flashView" height="450" width="450" hOffset="25"
                                        vOffset="25">
               <src>http://l.yimg.com/cosmos.bcst.yahoo.com/ver/242/player-2007-08-
                                        28-1213/swf/FLVVideo.swf</src>
          </flash>
     </window>
</widget>
```

Availability

The flash object is available in version 4.5 or newer.

allowNetworking

enables network functionality

Values

all
internal
none

Description

Use this attribute to enable an swf file to access network functionality. Setting to all allows use of any networking APIs in the swf file. Setting to internal allows the swf file to call networking APIs that *do not* perform browser navigation or browser interaction functions. Setting to none prevents the swf file from calling any browser navigation APIs, browser interaction APIs, or SWF-to-SWF communication APIs.

Example

```
<flash ... allowNetworking = “all” ... />

myFlash.allowNetworking = "all";
```

Availability

Available in version 4.5 or newer.

base

specifies base URL

Description

Specifies a base directory or URL used to resolve all relative path statements in a Flash Player movie. This attribute is helpful when your Flash Player movies are kept in a different directory from your other files.

Example

```
<flash ... base= “http://www.yourdomain.com” ... />

myFlash.base = "http://www.yourdomain.com";
```

Availability

Available in version 4.5 or newer.

bgColor

sets the background color of a movie

Description

Overrides the background color of a movie, or uses the default. Specify an integer of the form red*65536+green*256+blue to override the default. Specify -1 to use the default. Can also accept standard HTML hex (e.g., #ffffff = white). For Macintosh, you need to use a reload() call to change bgColor.

Example

```
<flash ... bgColor = “#AAFFFF” ... />

myFlash.bgColor = "#AAFFFF";
```

Availability

Available in version 4.5 or newer.

deviceFont

substitutes a font

Values

`true`
`false`

Description

If set to true and the font is installed on the system that is viewing the Flash movie, the font information installed on the system is substituted. Text is aliased (rough) regardless of the value of the quality attribute.

If the font is not installed on the system, the text comes out anti-aliased (smooth) if the quality of the movie is set to high.

Default is `false`.

Example

```
<flash ... deviceFont = “true” ... />
```


```
myFlash.deviceFont = true;
```

Availability

Available in version 4.5 or newer.

frameNumber

sets the currently displayed frame of the movie

Description

To advance or rewind the movie, use this attribute to set the current frame number. The change takes place immediately, so no reload is necessary.

Example

```
<flash ... frameNumber = “5” ... />

myFlash.frameNumber=5;
```

Availability

Available in version 4.5 or newer.

useFlashContextMenu

sets the menu to use for right click

Values

```
true
false
```

Description

Use this attribute to control which menu displays when the user right-clicks on the Flash view. If true, the Flash view uses the flash context menu. If false, the Flash view uses the widget context menu. Default is false.

Example

```
<flash ... useFlashContextMenu = “true” ... />

myFlash.useFlashContextMenu = true;
```

Availability

Available in version 4.5 or newer.

flashVars

allows variable information to pass to the flash file

Description

Use this attribute to allow variable information to be passed to the flash file. All data is passed as key/value pairs separated by &. For Macintosh, you need to use a `reload()` call to change `flashVars`.

Example

```
<flash ... flashVars="pig=oink&cow=moo&dog=bark" ... />

myFlash.flashVars = "pig=oink&cow=moo&dog=bark";
```

Availability

Available in version 4.5 or newer.

loop

enables looping

Values

```
true
false
```

Description

Use this attribute to enable the animation to loop. Setting to true enables looping; setting to false plays once and stops. Default is true.

Example

```
<flash ... loop = “true” ... />

myFlash.loop = true;
```

Platform Notes

For Macintosh, you need to use a `reload()` call to change the loop value.

Availability

Available in version 4.5 or newer.

minVersion

sets the minimum Flash version required

Description

Use this attribute to specify the minimum Flash version that must be available for this widget to work properly. It is specified as multiples of 65536. For e.g. version 9.0 is 9 * 65536 = 589824.

Example

```
<flash ... minversion = “589824” ... />

myFlash.minversion = 589824;
```

Platform Notes

For Macintosh the required minimum version is 9.0, but you can set a higher minimum version.

Availability

Available in version 4.5 or newer.

onFsCommand

installs handler for fscommand event

For More Information

See fscommand.

onFsReadyState

installs handler for fsreadystate event

For More Information

See fsreadystate.

quality

sets the rendering quality

Values

`Low`
`AutoLow`
`Medium`
`AutoHigh`
`High`
`Best`

Description

Use this attribute to set the rendering quality. Default is `High`.

`Low` prioritizes speed over appearance and never uses anti-aliasing.

`AutoLow` prioritizes speed at first, but improves appearance whenever possible. Playback begins with anti-aliasing turned off, then turns on if Flash Player detects that the processor can handle it. Note: `SWF` files authored using ActionScript 3.0 do not recognize `AutoLow`.

`Medium` applies some anti-aliasing and bitmaps are not smoothed. Quality is better than `Low`; worse than `High`.

`AutoHigh` prioritizes playback speed and appearance equally at first, but compromises appearance for speed when necessary. Playback begins with anti-aliasing turned on, then turns off if frame rate drops below the specified threshold.

`High` prioritizes appearance over speed and always applies anti-aliasing. If the `SWF` file does not contain animation, bitmaps are smoothed; if it contains animation, bitmaps are not smoothed.

`Best` provides the best display quality and does not consider playback speed. All output is anti-aliased and all bitmaps are smoothed.

Examples

```
// to Set
<flash ... quality = “Low” ... />

myFlash.quality = “Low”;
```

Availability

Available in version 4.5 or newer.

sAlign

sets the alignment mode

Values

`"t"`
`"r"`

```
"l"
"b"
"tr"
"tl"
"br"
"bl"
```

Description

Use this attribute to set the alignment mode, which consists of bit flags. t=top, r=right, l=left, b=bottom, tr=top-right, tl=top-left, br=bottom-right, bl=bottom-left.

Example

```
<flash ... sAlign = “lt” ... />

myFlash.sAlign = "lt";
```

Availability

Available in version 4.5 or newer.

scale

sets the scale mode

Values

```
showAll
noBorder
exactFit
noScale
```

Description

Use this attribute to specify the scale mode. Default is showAll.

Example

```
<flash ... scale = “exactFit” ... />

myFlash.scale = "exactfit";
```

Availability

Available in version 4.5 or newer.

src

specifies the Flash file to load

Description

Use this attribute to specify the Flash file to load. Returns a string.

Example

```
<flash ... src = “http://www.yourdomain.com/flash.swf” ... />

myFlash.src = "http://www.yourdomain.com/flash.swf";
```

Platform Notes

For Macintosh, changing this value requires a `reload()` for the new value to take affect.

Note

Special security settings are dependent on the value of `src`. If src is given a remote URL, `loadMovie` is restricted to only remote URL loading and `fsCommand` is disabled.

Availability

Available in version 4.5 or newer.

wMode

gets Window drawing mode

Description

Use this attribute to get the Window drawing mode, which is always "transparent." Read only; cannot be changed.

Example

```
var currentMode = myFlash.wMode;
```

Availability

Available in version 4.5 or newer.

back()

returns to the previous frame

Synopsis

```
back();
```

Description

This function returns to the previous frame, making that the current frame.

Example

```
myFlash.back();
```

Availability

Available in version 4.5 or newer.

forward()

advances to the next frame

Synopsis

```
forward();
```

Description

This function advances to the next frame, making that the current frame.

Example

```
myFlash.forward();
```

Availability

Available in version 4.5 or newer.

getVariable()

returns the value of the variable

Synopsis

```
getVariable( string varName );
```

Description

This function returns the value of the Flash variable varName. Returns null if the variable does not exist.

Example

```
myFlash.getVariable("/:message");
```

Availability

Available in version 4.5 or newer.

gotoFrame()

goes to the specified frame

Synopsis

```
gotoFrame( int frameNumber );
```

Description

This function activates frameNumber as the current frame. If the data for the frame is not yet available, the player goes to the last available frame and stops, causing unexpected results during playback. Use percentLoaded() to determine whether enough of the movie is available before executing gotoFrame().

Example

```
myFlash.gotoFrame(5);
```

Availability

Available in version 4.5 or newer.

isPlaying()

returns true if the movie is playing

Synopsis

```
isPlaying();
```

Description

This function returns true if the movie is currently playing; false if not.

Example

```
if(myFlash.isPlaying()) {
         alert("Playing");
}else {
      alert("Not Playing");
}
```

Availability

Available in version 4.5 or newer.

loadMovie()

loads the movie

Synopsis

```
loadMovie( int layerNumber, string url);
```

Description

This function loads the movie specified by url to layerNumber.

Example

```
myFlash.loadMovie(0, "http://www.yourdomain.com/flash.swf");
```

Note

Special security settings are dependent on the value of `src`. If src is given a remote URL, `loadMovie` is restricted to only remote URL loading.

Availability

Available in version 4.5 or newer.

pan()

pans the visible portion of the movie

Synopsis

```
pan( int x, int y, int mode);
```

Description

This function pans the visible portion of a zoomed-in movie to the specified x and y coordinates. Modes: 0=pixels, 1=% of window.

Example

```
myFlash.pan(10, 10, 1);
```

Availability

Available in version 4.5 or newer.

percentLoaded()

returns the percent already loaded

Synopsis

```
percentLoaded();
```

Description

This function returns the percent (as an integer) of the movie that has streamed into the browser at the time the function is called.

Example

```
myText.data = myFlash.percentLoaded() + "%";
```

Availability

Available in version 4.5 or newer.

play()

starts playing the movie

Synopsis

```
play();
```

Description

This function starts to play the movie.

Example

```
myFlash.play();
```

Availability

Available in version 4.5 or newer.

reload()

reloads the swf file

Synopsis

```
reload();
```

Description

This function causes the swf file to reload.

Example

```
myFlash.reload();
```

Availability

Available in version 4.5 or newer.

rewind()

returns to the first frame

Synopsis

```
rewind();
```

Description

This function returns to the first frame, making that the current frame.

Example

```
myFlash.rewind();
```

Availability

Available in version 4.5 or newer.

setVariable()

sets the value of the variable

Synopsis

```
setVariable( string varName, string varValue );
```


Description

This function sets the value of the Flash variable varName to varValue.

Example

```
myFlash.setVariable("/:message", "Message to Flash");
```

Availability

Available in version 4.5 or newer.

setZoomRect()

zooms in on the specified rectangle

Synopsis

```
setZoomRect( int left, int top, int right, int bottom );
```

Description

This function zooms in on a rectangular area of the movie, specified by the supplied coordinates in TWIPS (1440 units per inch). To calculate a rectangle, set ruler units to Points and multiply coordinates by 20 to get TWIPS.

Example

```
myFlash.setZoomRect(left*20, top*20, right*20, bottom*20);
```

Availability

Available in version 4.5 or newer.

stop()

stops playing the movie

Synopsis

```
stop();
```

Description

This function stops playing the movie.

Example

```
myFlash.stop();
```

Availability

Available in version 4.5 or newer.

stopPlay()

stops playing the movie

Synopsis

```
stopPlay();
```

Description

This function stops playing the movie.

Example

```
myFlash.stopPlay();
```

Availability

Available in version 4.5 or newer.

tCallFrame()

executes action in specified target and frameNumber

Synopsis

```
tCallFrame( string target, int frameNumber );
```

Description

In the timeline specified by target, this function executes the action in the specified frameNumber.

Example

```
myFlash.tCallFrame("/:action", 9);
```

Availability

Available in version 4.5 or newer.

tCallLabel()

executes action in specified target and label

Synopsis

```
tCallLabel( string target, string label);
```

Description

In the timeline specified by target, this function executes the action in the specified label.

Example

```
myFlash.tCallLabel("/:action", "actionLabel");
```

Availability

Available in version 4.5 or newer.

tCurrentFrame()

returns number of the current frame

Synopsis

```
tCurrentFrame( string target );
```

Description

Returns the number of the current frame for the target timeline. Returns 0 for frame 1 of the movie, 1 for frame 2, and so on.

Example

```
alert(myFlash.tCurrentFrame("/:Movie1"));
```


Availability

Available in version 4.5 or newer.

tCurrentLabel()

returns label of the current frame

Synopsis

```
tCurrentLabel( string target );
```

Description

Returns the label of the current frame for the target timeline. If there is no current frame label, returns an empty string.

Example

```
alert(myFlash.tCurrentLabel("/:Movie1"));
```

Availability

Available in version 4.5 or newer.

tGotoFrame()

goes to the specified frame

Synopsis

```
tGotoFrame( string target, int frameNumber );
```

Description

In the timeline specified by target, this function activates frameNumber as the current frame. If the data for the frame is not yet available, the player goes to the last available frame and stops, causing unexpected results during playback. Use percentLoaded() to determine whether enough of the movie is available before executing tGotoFrame().

Example

```
myFlash.tGotoFrame("/:Movie1", 35);
```

Availability

Available in version 4.5 or newer.

tGotoLabel()

goes to the specified label

Synopsis

```
tGotoLabel( string target, string label );
```

Description

In the timeline specified by target, this function goes to the specified frame label.

Example

```
myFlash.tGotoLabel("/:Movie1", "Frame20");
```

Availability

Available in version 4.5 or newer.

totalFrames()

returns the total number of frames for the movie

Synopsis

```
totalFrames();
```

Description

This function returns the total number of frames for the movie.

Example

```
alert(myFlash.totalFrames());
```

Availability

Available in version 4.5 or newer.

tGetProperty()

returns the property value as a string

Synopsis

```
tGetProperty( string target, int property );
```

Description

In the timeline specified by target, this function returns a string indicating the value of property. For property, enter the integer corresponding to the desired property. See Properties and Property Numbers for a list of properties and their corresponding integers.

Example

```
myFlash.tGetProperty("mov", 3);
```

Availability

Available in version 4.5 or newer.

tGetPropertyAsNumber()

returns the property value as a number

Synopsis

```
tGetPropertyAsNumber( string target, int property );
```

Description

In the timeline specified by target, this function returns a number indicating the value of property. For property, enter the integer corresponding to the desired property. See Properties and Property Numbers for a list of properties and their corresponding integers.

Example

```
myFlash.tGetPropertyAsNumber("mov", 3);
```


Availability

Available in version 4.5 or newer.

tGetPropertyNum()

returns the property value as a string

Synopsis

```
tGetPropertyNum( string target, int property );
```

Description

In the timeline specified by target, this function returns a string indicating the value of property. For property, enter the integer corresponding to the desired property. See Properties and Property Numbers for a list of properties and their corresponding integers.

Example

```
myFlash.tGetPropertyNum("mov", 3);
```

Availability

Available in version 4.5 or newer.

tPlay()

plays the target timeline

Synopsis

```
getVariable( string target );
```

Description

Plays the timeline specified by target.

Example

```
myFlash.tPlay("/:Movie1");
```

Availability

Available in version 4.5 or newer.

tSetProperty()

sets the property value

Synopsis

```
tSetProperty( string target, int property, string value);
```

Description

In the timeline specified by target, this function sets property to value, which can be a string or a number. For property, enter the integer corresponding to the desired property. See Properties and Property Numbers for a list of properties and their corresponding integers.

Example

```
myFlash.tSetProperty("mov", 13, "spiffy");
```

Availability

Available in version 4.5 or newer.

tSetPropertyNum()

returns the number of the property value

Synopsis

```
tGetProperty( string target, int property, double number );
```

Description

In the timeline specified by target, this function sets the property value to number. For property, enter the integer corresponding to the desired property. See Properties and Property Numbers for a list of properties and their corresponding integers.

Example

```
myFlash.tSetPropertyNum("/:mov", 18, 5);
```

Availability

Available in version 4.5 or newer.

tStopPlay()

stops the target timeline

Synopsis

```
tStopPlay( string target);
```

Description

Stops the timeline specified by target.

Example

```
myFlash.tStopPlay("/:Movie1");
```

Availability

Available in version 4.5 or newer.

version()

returns the Macromedia Flash version number

Synopsis

```
version();
```

Description

This function returns the version of the Macromedia Flash control. To get the major version number, divide by *65536*.

Example

```
if ( (myFlash.version()/65536) == 9 ) {
     alert("Version is Flash 9");
}
```


Availability

Available in version 4.5 or newer.

zoom()

zooms the view

Synopsis

```
zoom( int percent );
```

Description

This function zooms the view by a relative scale factor. For example, zoom(50) doubles the size of the objects in the view, and zoom(200) reduces the size of objects in the view by half.

Example

```
myFlash.zoom(90);
```

Availability

Available in version 4.5 or newer.

Properties and Property Numbers

The functions tSetProperty(), tGetProperty(), tGetPropertyNum() and tGetPropertyAsNumber() refer to properties by their property number. The following tables list the available properties and their corresponding property numbers.

Property	Property Number
X POSITION (_x)	0
Y POSITION (_y)	1
X SCALE	2
Y SCALE	3
CURRENTFRAME	4
TOTALFRAMES	5
ALPHA	6
VISIBILITY	7
WIDTH	8
HEIGHT	9
ROTATION	10
TARGET	11
FRAMESLOADED	12
NAME	13
DROPTARGET	14
URL (_url)	15

Global Property	Property Number
HIGHQUALITY	16

Global Property	Property Number
FOCUSRECT	17
SOUNDBUFTIME	18

Frame

scrollable container object

Frame objects act as containers for other objects (similar to a div in HTML). As such, you can nest other view objects inside them in the XML, as well as use JavaScript to place other objects inside them. When moved, all subviews of a frame move. Similarly, when the opacity of a subview changes, so does the effective opacity of everything in it. Unlike a div in HTML, frames automatically clip their contents to the bounds of the frame.

Frames also allow scrolling. You can do so manually by adjusting the scrollX and scrollY attributes, but you can also simply attach a scrollbar to a frame and have everything work automatically.

XML Name

<frame>

JavaScript Name

Frame

Attributes

hLineSize
hScrollBar
scrollX
scrollY
vLineSize
vScrollBar

Functions

end()
home()
lineDown()
lineLeft()
lineRight()
lineUp()
pageDown()
pageLeft()
pageRight()
pageUp()
updateScrollBars()

hLineSize

size of a line of data for use when scrolling

Description

The hLineSize attribute specifies how far a frame should scroll (in pixels) if the lineLeft() or lineRight() functions are called. It is also factored in when the frame reacts to the mouse wheel (if a scrollbar is attached). The default line size is 10 pixels.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" hLineSize="5"/>

myFrame.hLineSize = 5;
```

Availability

Available in version 3.0 or newer.

hScrollBar

horizontal scrollbar for this frame

Description

The `hScrollBar` attribute of a frame defines what scrollbar object should control the horizontal scrolling for this frame. When expressed in XML, you specify the name of a `<scrollbar>` object you wish to bind to the frame for its `hScrollBar`. If the scrollbar object does not exist, an error appears in the Widget's debug window.

Attaching a scrollbar automatically sets up communication between the frame and the scrollbar.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" hScrollBar="my_scrollbar"/>
<scrollbar name="my_scrollbar" ... />

// in Javascript:
myFrame.hScrollBar = my_scrollbar;
```

Availability

Available in version 3.0 or newer.

scrollX

horizontal scrolling offset

Description

The `scrollX` attribute allows you to specify the horizontal scrolling offset. For example, setting this attribute to -10 scrolls a frame's contents to the left 10 pixels. Normally you don't need to modify this attribute directly. Simply attaching a scrollbar to a frame causes this attribute to get updated as necessary to scroll the contents.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" scrollX="-10"/>

myFrame.scrollX = -10;
```

Availability

Available in version 3.0 or newer.

scrollY

vertical scrolling offset

Description

The `scrollY` attribute allows you to specify the vertical scrolling offset. For example, setting this attribute to -10 scrolls a frame's contents upward 10 pixels. Normally you don't need to modify this attribute directly. Simply attaching a scrollbar to a frame causes this attribute to get updated as necessary to scroll the contents.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" scrollY="-10"/>

myFrame.scrollY = -10;
```

Availability

Available in version 3.0 or newer.

vLineSize

size of a line of data for use when scrolling

Description

The `vLineSize` attribute specifies how far a frame should scroll (in pixels) if the `lineUp()` or `lineDown()` functions are called. It is also factored in when the frame reacts to the mouse wheel (if a scrollbar is attached). The default line size is 10 pixels.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" vLineSize="5"/>

myFrame.vLineSize = 5;
```

Availability

Available in version 3.0 or newer.

vScrollBar

vertical scrollbar for this frame

Description

The `vScrollBar` attribute of a frame defines what scrollbar object should control the vertical scrolling for this frame. When expressed in XML, you specify the name of the `<scrollbar>` object you wish to bind to the frame for its `vScrollBar`. If the scrollbar object does not exist, an error will appear in the Widget's debug window.

Attaching a scrollbar automatically sets up communication between the frame and the scrollbar.

Usage

JavaScript, XML

Example

```
<frame name="myFrame" vScrollBar="my_scrollbar"/>
<scrollbar name="my_scrollbar" ... />

// in Javascript:
myFrame.vScrollBar = my_scrollbar;
```

Availability

Available in version 3.0 or newer.

end()

scrolls a frame to the bottom left

Synopsis

```
void Frame.end();
```

Description

This function scrolls a frame to the bottom of its contents.

Example

```
myFrame.end();
```

Availability

Available in version 3.0 or newer.

home()

scrolls a frame to the upper left

Synopsis

```
void Frame.home();
```

Description

This function sets the `scrollX` and `scrollY` attributes to 0, 0.

Example

```
myFrame.home();
```

Availability

Available in version 3.0 or newer.

lineDown()

scrolls a frame one line down

Synopsis

```
void Frame.lineDown();
```

Description

This function scrolls a frame one line down by the amount specified by the frame's `vLineSize` attribute.

Example

```
myFrame.lineDown();
```

Availability

Available in version 3.0 or newer.

lineLeft()

scrolls a frame one line left

Synopsis

```
void Frame.lineLeft();
```

Description

This function scrolls a frame one line left by the amount specified by the frame's `hLineSize` attribute.

Example

```
myFrame.lineLeft();
```

Availability

Available in version 3.0 or newer.

lineRight()

scrolls a frame one line right

Synopsis

```
void Frame.lineRight();
```

Description

This function scrolls a frame one line right by the amount specified by the frame's `hLineSize` attribute.

Example

```
myFrame.lineRight();
```

Availability

Available in version 3.0 or newer.

lineUp()

scrolls a frame one line up

Synopsis

```
void Frame.lineUp();
```

Description

This function scrolls a frame one line up by the amount specified by the frame's `vLineSize` attribute.

Example

```
myFrame.lineUp();
```


Availability

Available in version 3.0 or newer.

pageDown()

scrolls a frame one page down

Synopsis

```
void Frame.pageDown();
```

Description

This function scrolls a frame one page down by the height of the frame minus one line height as specified by `vLineSize`.

Example

```
myFrame.pageDown();
```

Availability

Available in version 3.0 or newer.

pageLeft()

scrolls a frame one page left

Synopsis

```
void Frame.pageLeft();
```

Description

This function scrolls a frame one page left by the width of the frame minus one line height as specified by `hLineSize`.

Example

```
myFrame.pageLeft();
```

Availability

Available in version 3.0 or newer.

pageRight()

scrolls a frame one page right

Synopsis

```
void Frame.pageRight();
```

Description

This function scrolls a frame one page right by the width of the frame minus one line height as specified by `hLineSize`.

Example

```
myFrame.pageRight();
```

Availability

Available in version 3.0 or newer.

pageUp()

scrolls a frame one page up

Synopsis

```
void Frame.pageUp();
```

Description

This function scrolls a frame one page up by the height of the frame minus one line height as specified by `vLineSize`.

Example

```
myFrame.pageUp();
```

Availability

Available in version 3.0 or newer.

updateScrollBars()

updates the scroll bars for a frame immediately

Synopsis

```
void Frame.updateScrollBars();
```

Description

This function allows you to ensure the scroll bars for a frame are up-to-date. You normally don't need to call this function, as a frame's scroll bars are updated automatically. However, this is done lazily, i.e. they aren't updated until absolutely necessary, typically right before the frame is drawn. But there are times when it is necessary to update them immediately right after adding contents to the frame. For example, you might need to know the new max value of one of the scroll bars.

This should not be called too frequently, as it does need to calculate the bounds of all of a frame's children to do its work, which can be expensive.

Example

```
myFrame.updateScrollBar();
m = myFrame.hScrollBar.max; // now max is up-to-date
```

Availability

Available in version 4.0 or newer.

HotKey

block defining a hot key and associated default attributes

XML Name

```
<hotkey>
```

JavaScript Name

```
HotKey
```

Attributes

key
modifier
onKeyDown

onKeyUp

Description

The HotKey object allows your Widget to intercept a hot key action by the user. Hot keys are system-level key triggers that allow Widgets to be accessed through the keyboard. For example, a search Widget could be coded to come to the foreground with a sequence like **Control+Shift+F2**.

HotKey objects can also be created and destroyed dynamically using the JavaScript engine. This can be useful if you allow the user to customize your Widget's hot keys.

Note: Some key combinations are reserved by the system (e.g., **Alt+Tab** on Windows OS or **Command+Tab** on Mac OS X). On Mac OS X, if more than one Widget or application uses the same hot key, then all receive a notification when the user presses those keys. On Windows, only the first to try gets the hot key.

Example

```
<hotkey name="hkey1" key="F4" modifier="control+shift"
      onKeyDown="focusWidget()"/>

hkey1 = new HotKey();
hkey1.key = "F4";
hkey1.modifier = "control+shift";
```

key

the name of the function key

Description

On Mac OS X, hot keys can be defined for any of the following keys:

```
Delete, End, Escape, ForwardDelete, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11,
F12, F13, F14, F15, F16, Help, Home, PageDown, PageUp, Space, Tab
```

On Windows OS the following keys can be used:

```
UpArrow, DownArrow, LeftArrow, RightArrow, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10,
F11, F13, F14, F15, F16, Insert, ForwardDelete, Home, End, PageUp, PageDown, Help,
Clear, PrintScreen, ScrollLock, Pause, Enter, Return, Backspace, Delete, Space,
Tab, Escape
```

At least one modifier is required, which is **Command** on Mac OS X and **Control** on Windows OS by default.

Hot keys can also be defined for any letter or punctuation key but **two** modifiers must be specified in this case (to avoid confusing users by having familiar key combinations have unexpected effects).

JavaScript

myObjectName.key

Example

```
<hotkey name="hkey1" key="F2"/>

hkey1.key = "F2";
```

modifier

the modifier keys for the hot key

Description

The modifier attribute can be any combination of:

On Mac OS X: `command, control, option, shift`
On Windows OS: `control, alt, shift`

A modifier is always used and is Command on Mac OS X or Control on Windows OS by default.

Example

```
<hotkey name="hkey1" key="Home" modifier="control+shift"/>

hkey1.key = "Home";
hkey1.modifier = "control+shift";
```

onKeyDown

the code that is activated when a hot key is pressed

Description

The code to be run when the hot key is pressed is specified with the `onKeyDown` attribute.

Example

```
<hotkey name="hkey1" key="F10" modifier="control"
  onKeyDown=
    "print('Hotkey ' + event.keyString +
          ' pressed')"/>

hkey1.onKeyDown="print('Hotkey ' + event.keyString +
          ' pressed')";
```

Platform Notes

Note, on Mac OS X it is generally best to attach key code to the `onKeyUp` action as that is what users expect. However, note that only `onKeyDown` fires on Windows OS.

onKeyUp

the code that is activated when a hot key is released

Description

The code to be run when the hot key is released is specified with the `onKeyUp` attribute.

A common action to perform when a Widget's hot key is pressed is `focusWidget()`.

JavaScript

```
newObjectName = new HotKey()
```

Example

```
<hotkey name="hkey1" key="F10" modifier="control"
  onKeyUp="focusWidget()"/>

hkey1.onKeyUp="focusWidget()";
```


Platform Notes

This trigger is not available on Windows OS.

Image

image and associated default attributes

XML Name

<image>

JavaScript Name

Image

Attributes

clipRect
colorize
fillMode
hRegistrationPoint
hslAdjustment
hslTinting
loadingSrc
missingSrc
remoteAsync
rotation
src
srcHeight
srcWidth
tileOrigin
useFileIcon
vRegistrationPoint

Functions

fade()
moveTo()
reload()
slide()

Description

Image objects can be specified in XML and can be created and destroyed dynamically through the JavaScript engine. This latter method can be useful if you're creating a Widget that lists an indeterminate number of items.

Example

```
<image src="Images/Sun.png" name="sun1"
  hOffset="250" vOffset="250"
      height="20" width="30"
  hAlign="center"/>
```

clipRect

controls what part of an image is visible

Usage

JavaScript, XML

Description

You can limit what part of an image is drawn by applying a clip rectangle to it. Coordinates are given in X, Y, width, height order.

It's simpler and easier to clip by embedding an image into a frame, particularly if you are scrolling to show different parts of the image.

Note: Clipping does not reduce the memory requirements of your Widget. The entire image is in memory, not just the part that is visible.

JavaScript

```
myObjectName.clipRect
```

Example

If you have a 100x100 pixel image and only want to show the area starting at 20, 20 and extending to 50, 50, you would add this tag to your image:

```
<image ... clipRect="20, 20, 30, 30"/>
```

You can set or clear it at any time in JavaScript:

```
myImage.clipRect = "20, 20, 30, 30";
myImage.clipRect = null;
```

You can clear by setting to an empty string as well as null.

Availability

Available in version 2.0 or newer.

colorize

controls the overall colorization of an image

Usage

JavaScript, XML

Description

Colorizing turns an image into grayscale and, given a color, maps the color onto the gray ramp. The image turns all shades of whatever color you specify. You can do a sort of sepia tone effect with this, as well as various other interesting and surprisingly useful things.

In version 4.0 or newer, the CSS type colors such as `rgb( r, g, b )` or `rgba( r, g, b, a )` can be used. You can also use the 17 standard CSS colors such as black, white, etc.

Example

```
<image ... colorize="#993333"/>
```

To clear any colorization, just set it to null or an empty string in your code:

```
myImage.colorize = "";
```

Platform Notes

On Windows OS, some 8-bit image formats (GIF) might not work well with colorization.

Availability

Available in version 2.0 or newer.

fillMode

controls how an image fills its area

Usage

JavaScript, XML

Description

Normally, an image always stretches to fill the area it should occupy if you specify a width and height for the image. This tag allows you to override this and instead either stretch or tile the image by specifying either `tile` or `stretch`. If you were to use tiling, you also might need to use the tileOrigin attribute (described later).

If this tag is not specified, the default fill mode is `stretch`.

Example

```
<image name="img1" fillMode="tile" tileOrigin="bottomLeft"/>

img1.fillMode = "tile";
img1.tileOrigin="bottomLeft";
```

Availability

Available in version 2.0 or newer.

hRegistrationPoint

horizontal offset for defining a registration point

Usage

JavaScript, XML

Description

The `hRegistrationPoint` attribute of the image block defines the horizontal offset to use for placing and rotating the image. For example, if you have an 8x8 pixel image, and you set the `hRegistrationPoint` to be 4, the image would draw centered based on the `hOffset` you gave it.

Example

```
<image name"hourhand" src="hourHand.png"
      hRegistrationPoint="4"/>

hourhand.hRegistrationPoint = 4;
```

Notes

This attribute does not work correctly with `hAlign` and `vAlign`. Use these tags if you are trying to align something to an edge or to center it and reserve this tag for rotation purposes.

hslAdjustment

adjusts an image by adjusting it by Hue-Saturation-Lightness (HSL)

Usage

JavaScript, XML

Description

HSL adjustment works as in Photoshop's Adjust Hue/Saturation dialog when "Colorize" is not checked. You can shift the hue as well as increase color saturation and lightness. Hue can be adjusted from -180 to +180, saturation can be adjusted from -100 to +100. And lightness can be adjusted from -100 to +100. Adjusting the lightness upward can affect the saturation as well, so keep that in mind. You might use this for a throbbing effect where you need to shift all pixel hues evenly. This is also highly useful for doing things like changing an image to look selected by darkening it (decrease the lightness by about -55).

Example

```
<image name="img1"
  hslAdjustment="-20, 5, 0"/>

img1.hslAdjustment = "-20, 5, 0";
```

To clear any adjustment, just set it to null or an empty string in your code:

```
myImage.hslAdjustment = "";
```

Platform Notes

On Windows OS, some 8-bit image formats (GIF) might not work well with this feature.

Availability

Available in version 2.0 or newer.

hslTinting

colorizes an image using HSL tweaking

Usage

JavaScript, XML

Description

HSL tinting works as in Photoshop's Adjust Hue/Saturation dialog when "Colorize" is checked. You can set the hue and color saturation while adjusting lightness. Hue can be set from 0 to 360, saturation can be adjusted from 0 to +100. And lightness can be adjusted from -100 to +100. Adjusting the lightness upward might affect the saturation as well.

Example

```
<image name="img1" hslTinting="-20, 5, 0"/>

img1.hslTinting = "-20, 5, 0";
```

To clear any tinting, just set it to null or an empty string in your code:

```
myImage.hslTinting = "";
```

Platform Notes

On Windows OS, some 8-bit image formats (GIF) might not work well with this feature.

Availability

Available in version 2.0 or newer.

loadingSrc

path to an image to display while an image loads asynchronously

Usage

JavaScript, XML

Description

If a remote image is loaded asynchronously (by setting `remoteAsync` to true), you can display an alternate image in its place while the image is fetched from the server by setting this attribute. When the image is finally loaded, it replaces the `loadingSrc` automatically. If you wish to be informed when this happens, specify an action to happen using the `onImageLoaded` attribute.

Example

```
<image name="img1" src="http://www.imadethisup.com/remote.jpg"
      loadingSrc="images/loading.png" remoteAsync="true"/>

img1.loadingSrc = "images/loading.png";
img1.remoteAsync = true;
```

Availability

Available in version 3.0 or newer.

missingSrc

path to an image to display if src cannot be loaded

Usage

JavaScript, XML

Description

This attribute is used to customize the image that is displayed when an image's `src` attribute cannot be loaded. The Widget Engine has a default "missing" image for this situation, but it might not be adequate for all situations. Typically you use this when loading a remote source that might not exist or be accessible.

Example

```
<image name="img1" src="http://www.imadethisup.com/notthere.jpg"
      missingSrc="images/missing.png"/>

img1.missingSrc = "images/missing.png";
```

Availability

Available in version 3.0 or newer.

remoteAsync

sets fetch of remote images to be asynchronous

Usage

JavaScript, XML

Description

When set to true, `remoteAsync` tells the image object to load the image source in the background, allowing your Widget to do other things in the meantime. If you wish to specify an image to display while the image is being fetched, you can set the `loadingSrc` attribute. If you wish to be informed when the image is finally loaded, set the `onImageLoaded` attribute with some appropriate JavaScript.

Example

```
<image name="img1" src="http://www.a.remote.server.com/image.png"
       loadingSrc="images/loading.png"
       remoteAsync="true"/>

myImage.loadingSrc = "images/loading.png";
myImage.remoteAsync = true;
```

Availability

Available in version 3.0 or newer.

src

path to the image being displayed

Usage

JavaScript, XML

Description

The `src` attribute for the image block defines the source of the image. It takes a path to the file on your hard drive relative to the XML file of the Widget it's referenced from.

You can specify a path or URL as the source for the image.

Example

```
<image name="img1" src="Resources/Buttons/button.png"/>

img1.src = "Resources/Buttons/button.png";
```

srcHeight

original height of the source image

Usage

JavaScript

Description

The `srcHeight` attribute gives the original height of the image as it was when it was read from the disk before any resizing was done. This attribute is read-only, so setting it has no effect.

Example

```
origHeight = myButton.srcHeight;
```


srcWidth

original width of the source image

Usage

JavaScript

Description

The `srcWidth` attribute gives the original width of the image as it was when it was read from the disk before any resizing was done. This attribute is read-only, so setting it has no effect.

Example

```
origWidth = myButton.srcWidth;
```

tileOrigin

controls how image is tiled

Usage

JavaScript, XML

Values

```
topLeft
topRight
bottomLeft
bottomRight
```

Description

The `tileOrigin` attribute is used with the `fillMode` tag, described above. If `fillMode` is set to `tile`, an image is tiled into its width and height (assuming they are larger than the natural size of the image). This attribute controls what corner of an image the tiling starts from. If this attribute is not specified, the default is `topLeft`.

Example

```
<image name="img1" fillMode="tile"
  tileOrigin="bottomLeft"/>

img1.fillMode = "tile";
img1.tileOrigin = "bottomLeft";
```

Availability

Available in version 2.0 or newer.

useFileIcon

retrieves the icon for the file

Usage

JavaScript, XML

Description

If your image uses the `useFileIcon` attribute, the engine will look inside the file specified by your `src` attribute for an icon. In this case, your `src` attribute can be an executable or any other type of file.

Example

```
<image name="img1" useFileIcon="true"
        src="/Applications/iChat.app"/>

img1.src = "/Applications/iChat.app";
img1.useFileIcon = "true";
```

vRegistrationPoint

vertical offset for defining a registration point

Description

The `vRegistrationPoint` attribute of the image block defines the vertical offset to use for placing and rotating the image. For example, if you have an 8x8 pixel image, and you set the `vRegistrationPoint` to be 4, the image would draw centered on the `vOffset` you gave it.

Example

```
<image name="img1" src="hourHand.png"
      vRegistrationPoint="36"/>

img1.vRegistrationPoint = 36;
```

Notes

This attribute does not work correctly with `hAlign` and `vAlign`. Use these tags if you are trying to align something to an edge or to center it and reserve this tag for rotation purposes.

fade()

fades in or fades out an image

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. See Animation for more information.

Synopsis

```
Image.fade(start, end, duration)
```

Description

The `fade()` command will cause an image to fade from a starting opacity to a finishing opacity. `duration` specifies the time (in tenths of a second) you want the animation to last for.

Example

```
newOpacity = 0;
myImage.fade(myImage.opacity, newOpacity, 1);
```

moveTo()

moves an image from point a to point b using animation

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. See Animation for more information.

Synopsis

```
Image.moveTo(newX, newY, duration)
```

Description

The image's origin (`hOffset`, `vOffset`) is moved to the new coordinates specified by `newX` and `newY`. `duration` specifies the time (in tenths of a second) you want the animation to last for. The move of the object is animated (which is what makes this different from just changing `hOffset` and `vOffset`).

Example

```
myImage.moveTo(50, 50, 3);
```

reload()

reloads an image

Synopsis

```
Image.reload()
```

Description

Use this method to reload an image. This is especially useful if your Widget makes use of a graphic that is being constantly updated by an external process as it defeats the normal caching behavior of the `Image` object.

Be aware that reloading an image that is remote requires an HTTP fetch. You should reload sparingly in these cases for better performance.

Example

```
myImage.reload();
```

slide()

slides an image in a specified direction and duration

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. To accomplish this effect, it is best to put the image into a frame and then move the image within it. The frame clips it out as appropriate. See Animation for more information

Synopsis

```
Image.slide(direction, amountOfImageToConceal, duration)
```

Description

The `slide()` command is an animation effect used to hide and reveal parts of the user interface. It is used when you want to slide an image in a particular direction, and have it disappear into itself rather than just move. `duration` specifies the time (in tenths of a second) you want the animation to last for.

Example

```
myImage.slide( "up,left", 50, 3 );
```

MenuItem

menu item for use in a context menu or pop-up

XML Name

<menuitem>

JavaScript Name

MenuItem

Attributes

checked
enabled
onSelect
title

Description

Menu items are used by the context menu arrays and handlers to provide extra items for the standard Widget context menu. They are also used in conjunction with the popupMenu global function.

Version 4.0 and newer include the ability to have submenus by adding child menu items using appendChild(). You can also remove submenu items using removeChild().

Examples

```
a = new Array();

x = new MenuItem();
x.title = "Item 1";
x.onSelect = "print( 'item 1' )";

a.push( x );

x = new MenuItem();
x.title = "Submenu!";

y = new MenuItem();
y.title = "SubItem 1";
y.onSelect = "print( 'sub item 1' )";

x.appendChild( y );

a.push( x );
popupMenu( a, 10, 10 );
```

Availability

Available in version 2.0 or newer. Submenus available in version 4.0 or newer.

checked

specifies item is checked

Description

This attribute specifies that the item should have a checkmark next to it when displayed in the menu. If this attribute is not specified, the default is false.

Example

```
<menuItem name="myItem"
   title="Widgetz R0x0r!!!11" checked="true"/>

// or in JavaScript
myItem.checked = true;
```

Availability

Available in version 2.0 or newer.

enabled

specifies item is enabled

Description

This attribute specifies that the item should be enabled in the menu. If set to false, the item appears grayed out and cannot be chosen by the user. If this attribute is not specified, the default is true.

Example

```
<menuItem title="Recent Locations" enabled="false"/>
// or in JavaScript
myItem.enabled = false;
```

Availability

Available in version 2.0 or newer.

onSelect

specifies the JavaScript to run when item is chosen

Description

This attribute provides the action to carry out when an item is chosen from the context menu.

Example

```
<menuItem title="Recent Locations" enabled="false"
   onSelect="beep();"/>

// or in JavaScript
myItem.onSelect = "beep();";
```

Availability

Available in version 2.0 or newer.

title

specifies the text of a menu item

Description

This attribute provides the text to display for a menu item.

Example

```
<menuItem title="I am the title"/>

// or in JavaScript
```

```
myItem.title = "Choose me!";
```

Availability

Available in version 2.0 or newer.

Point

represents a point on a two-dimensional plane

XML Name

n/a

JavaScript Name

Point

Attributes

x
y

Description

Point objects are used to identify a point in a window or view. Currently, Point objects are only used with the convertPointFromWindow, convertPointToWindow, convertPointToParent, and convertPointFromParent functions that are available on many of the DOM objects in a window.

They are a simple object, only containing an x and a y attribute. The attributes are not explained further below due to their simplicity.

Example

```
var p = new Point( 10, 100 );
print( p.x, p.y );
```

Availability

Available in version 4.0 or newer.

Preference

block defining a preference setting and associated attributes

Attributes

defaultValue
description
directory
extension
group
hidden
kind
maxLength
minLength
notSaved
option
optionValue
secure
style
ticks
tickLabel

```
title
type
value
```

Description

Preferences allow a Widget to save and restore either state or user settings in between when a Widget is quit and restarted. Preferences are defined in XML only at present, but can be altered through JavaScript at runtime. All preferences are held in the preferences global variable, where each preference is an attribute of that variable. For example, a preference with the name foo can be accessed in JavaScript with `preferences.foo`.

There are two preferences that are provided automatically:

`konfabulatorWindowLevel`

The level the Widget window displays at on the user's screen: `topMost`, `normal`, `below`, or `desktop`.

`konfabulatorWindowOpacity`

The master opacity of the Widget. All windows multiply this opacity with their own.

These preferences allow the user to control how the Widget displays on their desktop. If you want to provide this functionality yourself, all you have to do is call your preferences the same names, `windowLevel` and `windowOpacity`. If you want to disable this feature, define two preferences as follows in your Widget:

```
<preference name="windowLevel">
  <hidden>true</hidden>
</preference>

<preference name="windowOpacity">
  <hidden>true</hidden>
</preference>
```

defaultValue

default value of the preference

Description

The `defaultValue` attribute of the preference block specifies what the value should be by default. This makes it possible to prepopulate your preferences as well as have placeholders until the user enters proper data. This is the value your JavaScript code sees if it accesses the preferences before the user has customized them.

Example

```
<preference name="colorPref">
    <defaultValue>red</defaultValue>
</preference>

colorPref.defaultValue = "red";
```

description

descriptive text displayed in the preference panel

Description

The `description` attribute of the preference block defines the descriptive text that goes underneath a preference when being displayed in the preference panel's user interface.

It's optional, but highly recommended, to explain the preference and its usage to your users.

Example

```
<preference name="colorPref">
    <description>
        Enter the desired color
    </description>
</preference>
```

directory

default starting directory for a preference of type selector

Description

Preferences of type `selector` can have their starting directory set using this attribute.

Example

```
<preference>
    <type>selector</type>
    <style>open</style>
    <directory>~/Documents</directory>
</preference>
```

extension

file type for a preference of type selector

Description

Preferences of type `selector` displaying an open system dialog can be limited to returning only files with certain extensions using this attribute.

Example

```
<preference>
    <type>selector</type>
    <style>open</style>
    <extension>.jpg</extension>
    <extension>.gif</extension>
    <extension>.png</extension>
</preference>
```


group

group this preference belongs to

Description

Preferences can be divided into groups, and displayed in a multipane dialog. This attribute tells the Widget Engine which preference group this particular preference belongs to. If this attribute is not specified, the preference is rolled into a General group automatically.

Example

```
<preference>
    <type>selector</type>
    <style>save</style>
    <group>my_group</group>
</preference>
```

The above example assumes that you've defined an appropriate preference group called `my_group` somewhere in your XML. See the section on PreferenceGroup for more information.

Availability

Available in version 2.0 or newer.

hidden

hides preference from user

Description

If a preference has the `hidden` attribute, the ability to edit or see that preference is not offered to the end user. The preference can still be manipulated in JavaScript but it isn't displayed in the Widget Preferences dialog. If a Widget has only hidden preferences, the user is not offered the Widget Preferences option on the context menu. Hidden preferences are often used to implement settings the user makes using controls on the Widget rather than by opening the Widget Preferences dialog.

Example

```
<preference name="colorPref">
    <hidden>true</hidden>
    <type>text</type>
    <defaultValue>red</defaultValue>
</preference>
```

kind

kind of item for a preference of type selector

Description

Preferences of type `selector` displaying an open system dialog can be limited to files, folders, or both using the `kind` attribute.

Example

```
<preference>
    <type>selector</type>
    <style>open</style>
    <kind>folders</kind>
</preference>
```

maxLength

maximum value of a slider preference

Description

Currently used only for slider preferences, this is the maximum value the slider can represent.

Example

```
<preference>
    <maxLength>200</maxLength>
</preference>
```

minLength

minimum value of a slider preference

Description

Currently used only for slider preferences, this is the minimum value the slider can represent.

Example

```
<preference>
    <minLength>1</minLength>
</preference>
```

notSaved

prevents a preference value from being automatically saved

Description

The `notSaved` attribute sets the preference so that it is not automatically saved in the user's preference file for the Widget. This can be useful if you want to display a control on the preferences panel but handle the value returned in code. In a way, this attribute is the opposite of `hidden`.

Example

```
<preference>
    <notSaved>true</notSaved>
</preference>
```

option

choices for a preference of type popup

Description

Preferences of type popup are displayed as a pop-up (or in Windows, a drop-down) menu in the Widget Preferences dialog. Several option attributes can be used to provide a set of choices for a pop-up menu.

Specifying the string `(-` for an option causes a separator to be displayed at that point in the pop-up (which cannot be selected by the user).

Example

```
<preference name="colorPref">
    <type>popup</type>
    <option>Red</option>
    <option>White</option>
    <option>-</option>
```


```
    <option>Blue</option>
</preference>
```

optionValue

values corresponding to the choices for a preference of type popup

Description

When an `option` (see above) is chosen, if you want the value returned to be different, specify an `optionValue` for each option. There should be an `optionValue` for every option (note that if you use a separator option, see above, you need to give it a corresponding dummy `optionValue` even though this value can never be returned).

Example

```
    <optionValue>#0000FF</optionValue>
</preference><preference name="colorPref">
    <type>popup</type>
    <option>Red</option>
    <optionValue>#FF0000</optionValue>
    <option>White</option>
    <optionValue>#FFFFFF</optionValue>
    <option>-</option>
    <optionValue>none</optionValue>
    <option>Blue</option>
```

secure

specifies that a preference value should be obfuscated

Description

Any type of preference can be secure, which causes its data to be saved in a manner that cannot easily be read. This is useful for saving items such as passwords. Text preferences additionally display a "password" style user interface (bullets appear instead of typed characters).

If the code that reads a previously secure preference is changed to be nonsecure, the value of the preference is reset to the `defaultValue`.

Example

```
<preference>
    <type>text</type>
    <secure>yes</secure>
</preference>
```

style

dialog style for a preference of type selector

Description

Preferences of type `selector` can display either a standard system open or save dialog. The former allows the user to choose existing files, the latter a place to save or create a new file.

Example

```
<preference>
    <type>selector</type>
    <style>open</style>
```

```
</preference>
```

ticks

number of tick marks to display on a slider preference

Description

To make the slider display tick marks, use the `ticks` attribute. A side effect of this is that the slider only returns values corresponding to the ticks.

For sliders, the `minLength` and `maxLength` attributes define the minimum and maximum values that can be set. The first tick corresponds to `minLength` and the last to `maxLength`.

Example

```
<preference>
    <type>slider</type>
    <ticks>10</ticks>
    <minLength>0</minLength>
    <maxLength>100</maxLength>
</preference>
```

tickLabel

labels for slider preferences

Description

To make the slider display labels under the track, specify one or many `tickLabel` attributes. The labels are evenly distributed along the length of the slider.

Example

```
<preference>
    <type>slider</type>
    <tickLabel>One</tickLabel>
    <tickLabel>Volume</tickLabel>
    <tickLabel>Eleven</tickLabel>
</preference>
```

title

label displayed in the preference panel

Description

The `title` attribute of the preference block defines the label title that is displayed to the user through the built in preference interface.

Example

```
<preference>
    <title>Color:</title>
</preference>
```


type

type of data and control to display

Description

The `type` attribute of the preference block defines what type of user interface object is used to display the data choices. Type can be one of the following:

`checkbox`	Displays a checkbox to gather yes/no input. The value returned to the Widget is either `"0"` or `"1"`.
`color`	Displays a color swatch and allow colors to be picked. The value returned to the Widget is a standard color specifier like `"#123456"`.
`font`	Displays a font name and allow a font to be picked from those available on the system. The `font` attribute of a `Text` object can be set to the value returned.
`hotkey`	Displays a hot key and its modifier and allow alternative key combinations to be chosen. The value returned can be used to set the modifier and key for a `HotKey` object.
`popup`	Displays a choice and allow alternatives to be chosen from a Widget specified list. A string is returned (either one of the `options` or, if specified, one of the `optionValues`).
`selector`	Display a file name and allow other file names to be chosen. The value returned to the Widget is the fully qualified path of the file (a web style path with / separators).
`slider`	Display a slider and allow numeric values to be input. A numeric value is returned.
`text`	A standard text field in which the user can type text. A string is returned (see the `secure` attribute for information on displaying password style text fields).

Example

```
<preference>
    <type>checkbox</type>
</preference>
```

value

current value of the preference

Description

The `value` attribute of the preference contains the current value assigned to the preference. This might have just been entered by the user or might have been read from the Widget's preference file at startup.

Notes

The `value` attribute is always treated as a string even if it contains a number. If you want to use a preference value as a number, use the appropriate conversion routine when accessing it. For instance:

```
numberOfItems = Number(preferences.numItems.value) + 1;
```

PreferenceGroup

group to organize preferences

Attributes

`name`

```
icon
order
title
```

Description

Preference groups allow you to organize your preferences when displayed in the Preferences dialog. The dialog is displayed as a multipane dialog in version 2.0 and newer. You define your groups using `preferenceGroups` and then set the group attribute of each preference you want in a particular group.

Availability

Available in version 2.0 or newer.

name

name of the preference group

Description

This attribute defines the group name. This name is an identifier and should be unique among all preference groups. When defining a preference item that belongs to a group, it is this name you use to identify the group to which it belongs. The `name` attribute should not be confused with the `title` attribute, which is the user-visible name that is shown in the preferences window toolbar.

Example

```
<preferenceGroup name="colors" title="Colors"/>
```

Availability

Available in version 2.0 or newer.

icon

image to display for the group

Description

Use the `icon` attribute to specify the image that is displayed in the dialog to represent your group. This image must be 32x32 pixels maximum at present. If you do not specify an icon, a default one is provided for your group automatically.

Example

```
<preferenceGroup icon="Resources/myPrefIcon.png"/>
```

Availability

Available in version 2.0 or newer.

order

defines order for your groups

Description

Use this attribute to help control the order in which your preference groups appear in the dialog. The numbering is completely up to you, but the lowest number is displayed in the leftmost position.

Example

```
<preferenceGroup
```


```
    title="First Group"
    order="0" />
<preferenceGroup>
    title="Second Group"
    order="1"/>
```

Availability

Available in version 2.0 or newer.

title

title of your preference group

Description

This attribute defines what text should appear below the icon of your preference group in the dialog. These titles should generally be short and one or two words long.

Example

```
<preferenceGroup>
    title="General"
    order="0"/>
<preferenceGroup>
    title="Special"
    order="1"/>
```

Availability

Available in version 2.0 or newer.

Rectangle

specifies a rectangle object

XML Name

n/a

JavaScript Name

Rectangle

Description

Rectangles are typically used to define a rectangular region in a window or view.

Attributes

x
y
width
height

Functions

offset(dX, dY)
inset(dX, dY)
getMinX()
getMinY()
getMaxX()
getMaxY()

```
getMidX()
getMidY()
setEmpty()
isEmpty()
makeIntegral()
containsPoint(Point | x,y)
unionWith(Rectangle | x, y, width, height)
intersectWith(Rectangle | x, y, width, height)
```

x

the horizontal origin of the rectangle

Usage

JavaScript, XML

Description

This attribute specifies the horizontal origin of the rectangle, usually the left side.

Example

```
var leftEdge = myRectangle.x;
```

Availability

Available in version 4.5 or newer.

y

the vertical origin of the rectangle

Usage

JavaScript, XML

Description

This attribute specifies the vertical origin of the rectangle, usually the top.

Example

```
var topEdge = myRectangle.y;
```

Availability

Available in version 4.5 or newer.

width

the width of the rectangle

Usage

JavaScript, XML

Description

This attribute specifies the width of the rectangle.

Example

```
var w = myRectangle.width;
```

Availability

Available in version 4.5 or newer.

height

the height of the rectangle

Usage

JavaScript, XML

Description

This attribute specifies the height of the rectangle.

Example

```
var h = myRectangle.height;
```

Availability

Available in version 4.5 or newer.

offset(dX, dY)

offsets the rectangle

Synopsis

```
Rectangle.offset(dX, dY)
```

Description

Use this function to offset the rectangle by dX horizontally and dY vertically. The deltas can be negative.

Example

```
myRectangle.offset(0, 100);
```

Availability

Available in version 4.5 or newer.

inset(dX, dY)

insets the rectangle

Synopsis

```
Rectangle.inset(dX, dY)
```

Description

Use this function to inset the rectangle by dX horizontally and dY vertically. The deltas can be negative, resulting in outsetting the rectangle.

Example

```
myRectangle.inset(0, 100);
```

Availability

Available in version 4.5 or newer.

getMinX()

returns the minimum X value of the rectangle

Synopsis

```
Rectangle.getMinX()
```

Description

Use this method to get the minimum x value of the rectangle (i.e., the left). This is the same as the x attribute, but is here for completeness.

Example

```
myRectangle.getMinX();
```

Availability

Available in version 4.5 or newer.

getMinY()

returns the minimum Y value of the rectangle

Synopsis

```
Rectangle.getMinY()
```

Description

Use this function to get the minimum y value of the rectangle (i.e., the top). This is the same as the y attribute, but is here for completeness.

Example

```
myRectangle.getMinY();
```

Availability

Available in version 4.5 or newer.

getMaxX()

returns the maximum X value of the rectangle

Synopsis

```
Rectangle.getMaxX()
```

Description

Use this function to get the maximum x value of the rectangle (i.e., the right). This is essentially `x + width`.

Example

```
myRectangle.getMaxX();
```

Availability

Available in version 4.5 or newer.

getMaxY()

returns the maximum Y value of the rectangle

Synopsis

```
Rectangle.getMaxY()
```

Description

Use this function to get the maximum y value of the rectangle (i.e., the bottom). This is essentially `y + height`.

Example

```
myRectangle.getMaxY();
```

Availability

Available in version 4.5 or newer.

getMidX()

returns the midpoint X value of the rectangle

Synopsis

```
Rectangle.getMidX()
```

Description

Use this function to get the midpoint x value of the rectangle. This is essentially `x + width/2`.

Example

```
myRectangle.getMidX();
```

Availability

Available in version 4.5 or newer.

getMidY()

returns the midpoint Y value of the rectangle

Synopsis

```
Rectangle.getMidY()
```

Description

Use this method to get the midpoint y value of the rectangle. This is essentially `y + height/2`.

Example

```
myRectangle.getMidY();
```

Availability

Available in version 4.5 or newer.

setEmpty()

sets all rectangle attributes to 0

Synopsis

```
Rectangle.setEmpty()
```

Description

Use this function to set the rectangle to `0, 0, 0, 0`.

Example

```
myRectangle.setEmpty();
```

Availability

Available in version 4.5 or newer.

isEmpty()

determines if rectangle has height or width

Synopsis

```
Rectangle.isEmpty()
```

Description

Returns `TRUE` if the rectangle has a height and width of `0`. The origin is not considered.

Example

```
myRectangle.isEmpty();
```

Availability

Available in version 4.5 or newer.

makeIntegral()

rounds rectangle dimensions to integers

Synopsis

```
Rectangle.makeIntegral()
```

Description

When centering, etc., rectangles can result with fractional positions and sizes. Use this function to ensure that the resulting rectangle has valid coordinates and is of the correct size. For example, a rectangle of `10.1, 10, 100.7, 100` is returned by this function as `10, 10, 101, 100`.

Example

```
myRectangle.makeIntegral();
```

Availability

Available in version 4.5 or newer.

containsPoint(Point | x,y)

determines if a point is inside the rectangle

Synopsis

```
Rectangle.containsPoint(Point | x,y)
```

Description

Returns TRUE if the point is located inside the rectangle's area.

Example

```
myRectangle.containsPoint(40,75);
```

Availability

Available in version 4.5 or newer.

unionWith(Rectangle | x, y, width, height)

joins two rectangles

Synopsis

```
Rectangle.unionWith(Rectangle | x, y, width, height)
```

Description

Joins the rectangle with another `Rectangle` or set of rectangle specifications `x, y, width, height`. Using `unionWith` on an empty rectangle has no effect.

Example

```
myRectangle.unionWith(Rectangle | 1, 10, 40,75);
```

Availability

Available in version 4.5 or newer.

intersectWith(Rectangle | x, y, width, height)

intersects two rectangles

Synopsis

```
Rectangle.intersectWith(Rectangle | x, y, width, height)
```

Description

Intersects the rectangle with another `Rectangle` or set of rectangle specifications `x, y, width, height`. Intersecting with an empty rectangle yields an empty rectangle. Also returns an empty rectangle if the two rectangles do not intersect.

Example

```
myRectangle.intersectWith(Rectangle | 1, 10, 40,75);
```

Availability

Available in version 4.5 or newer.

Script

specifies JavaScript to execute

Usage

XML

Attributes

`src`
`charset`

Description

The `script` object, introduced in version 4.0, allows you to put script in more places in your `.kon` file than just the `onLoad` handler and object event handlers. It is very much like the `<script>` element in HTML. When a `<script>` element is encountered during parsing, its contents are fed to the JavaScript engine and executed.

You can either specify the code as the contents of the element, or point to another file to load. You can also specify the character set of the file you are loading.

Example

```
<script>
      print( "hello!" );
</script>
<script src="myexternalfile.js" charset="utf-8"/>
```

ScrollBar

specifies a scrollbar object

Description

Scrollbars are typically used in conjunction with frames to allow the user to scroll the frame's contents. You can also use them apart from frames, for example, as sliders.

Common Attributes

`hAlign`
`height`
`hOffset`
`firstChild`
`lastChild`
`nextSibling`
`previousSibling`
`onClick`
`onDragDrop`
`onDragEnter`
`onDragExit`
`onMouseDown`
`onMouseEnter`
`onMouseExit`
`onMouseMove`
`onMouseUp`
`onMultiClick`
`opacity`
`parentNode`
`subviews`

tooltip
vAlign
visible
vOffset
width
window
zOrder

Attributes

autoHide
max
min
onValueChanged
orientation
pageSize
thumbColor
value

Functions

setRange()
setThumbInfo()
setTrackInfo()

autoHide

hides scrollbar when there's nothing to scroll

Usage

JavaScript, XML

Description

This attribute hides the scrollbar when there is nothing to scroll. The default of this attribute is false. If the scrollbar is not set to auto-hide, it instead becomes 50% transparent when there is nothing to scroll.

Example

```
<scrollbar>
      <autoHide>true</autoHide>
</scrollbar>

myScrollbar.autoHide = true;
```

Availability

Available in version 3.0 or newer.

max

maximum value of a scrollbar

Usage

JavaScript, XML

Description

The `max` attribute defines the maximum value of a scrollbar. Together with `min`, it defines the range of values the scrollbar can have. Values are pinned between `min` and `max`.

If you attach a scrollbar to a frame, you rarely need to deal with this attribute as it is set up automatically.

In versions prior to 4.0, you must use the `setRange()` function to affect this attribute in JavaScript.

JavaScript

myObjectName.max

Example

```
<scrollbar name="sb" min="0" max="100"/>

sb.min = 0;
sb.max = 100;
```

Availability

Available in version 3.0 or newer.

min

minimum value of a scrollbar

Usage

JavaScript, XML

Description

The `min` attribute defines the minimum value of a scrollbar. Together with `max`, it defines the range of values the scrollbar can have. Values are pinned between `min` and `max`.

If you attach a scrollbar to a frame, you rarely need to deal with this attribute as it is set up automatically.

In versions prior to 4.0, you must use the `setRange()` function to affect this attribute in JavaScript.

JavaScript

myObjectName.min

Example

```
<scrollbar name="sb" min="0" max="100"/>

sb.min = 0;
sb.max = 100;
```

Availability

Available in version 3.0 or newer.

onValueChanged

installs handler for valuechanged event

For More Information

See valuechanged.

orientation

orientation of a scrollbar

Usage

JavaScript, XML

Description

The `orientation` attribute allows you to specify the orientation of a scrollbar. Its possible values are `vertical` and `horizontal`. The default is `vertical`.

Example

```
<scrollbar>
    <orientation>vertical</orientation>
</scrollbar>

myScrollbar.orientation = "horizontal";
```

Availability

Available in version 3.0 or newer.

pageSize

page size of a scrollbar

Usage

JavaScript, XML

Description

The `pageSize` attribute is used to help determine the size of the thumb for a proportional scrollbar. Typically, this is the height of the view being scrolled (assuming a vertical scrollbar).

If you have attached a scrollbar to a frame for scrolling, you do not need to deal with this attribute directly as it is automatically set up and handled.

Example

```
<scrollbar>
    <pageSize>140</pageSize>
</scrollbar>

myScrollbar.pageSize = 100;
```

Availability

Available in version 3.0 or newer.

thumbColor

thumb color of a scrollbar

Usage

JavaScript, XML

Description

The `thumbColor` attribute is used to control the tint of the thumb. The default thumb in the standard scrollbar is a medium gray. The color you specify is applied using colorization.

To clear the current color completely, you can set it to null in JavaScript.

Example

```
<scrollbar>
    <thumbColor>#333366</thumbColor>
</scrollbar>
myScrollbar.thumbColor = "#333366";
myScrollbar.thumbColor = null;
```

Availability

Available in version 3.0 or newer.

value

current value of the scrollbar

Usage

JavaScript, XML

Description

The `value` attribute contains the current value of the scrollbar. You can also use it to set the value to some value between the scrollbar's minimum and maximum values. If you specify a value less than the minimum or greater than the maximum, the value is pinned to those values.

JavaScript

myObjectName.value

Example

```
<scrollbar>
    <min>-100</min>
    <max>100</max>
    <value>0</value>
</scrollbar>

myScrollbar.value = 10;
```

Availability

Available in version 3.0 or newer.

setRange()

sets the min and max of a scrollbar

Synopsis

```
ScrollBar.setRange(int min, int max)
```


Description

Use this method to define the range of values that can be expressed by a scrollbar. In versions prior to 4.0, you cannot modify the `min` and `max` attributes directly, so you must use this function to set those attributes. The value is pinned to this range if the value falls outside of the new range when this function is called.

You rarely need to deal with this function if you are attaching a scrollbar to a frame object as the range is set automatically.

Example

```
myScrollbar.setRange(0, 100);
```

Availability

Available in version 3.0 or newer.

setThumbInfo()

replace the images that comprise the thumb of a scrollbar

Synopsis

```
ScrollBar.setThumbInfo(int offset, string|array images )
```

Description

While the standard `ScrollBar` object allows you to customize the thumb color, this might not meet every Widget's need. To facilitate more customization, you can pass either one or three image paths to this function as a string or array, respectively, and the scrollbar uses those images to make the thumb. If you pass one image, this implies a fixed size thumb and your scrollbar is not proportional. This might be appropriate if you are trying to make a slider object instead. If you pass three, the first and third image paths specify the caps to use for top/bottom. The second image is a stretchable center. Therefore, three-part scrollbar thumbs are always proportional.

The offset parameter controls how far into the scrollbar the thumb should be positioned. For example if you specify 3, your scrollbar images appear 3 pixels to the right of the left edge of the scrollbar view. For horizontal scrollbars, this is the number down to nudge the thumb.

Notes

At present, horizontal scrollbar images have to be created with the same orientation as vertical ones. So you need to design your scrollbars horizontally, then rotate them 90 degrees clockwise before chopping them up and saving out the pieces.

Example

```
myScrollbar.setThumbInfo( 1, new Array( "images/topCap.png",
        "images/middle.png", "images/bottomCap.png" ) );
myScrollbar.setThumbInfo( 0, "images/fixedthumb.png" ) );
```

Availability

Available in version 3.0 or newer.

setTrackInfo()

replace the images that comprise the track of a scrollbar

Synopsis

```
ScrollBar.setTrackInfo(int offset, int topLimit,
```

```
    int bottomLimit, string|array images )
```

Description

To customize the standard `ScrollBar` object's track, pass either one or three image paths to this function (as a string or array, respectively), and the track uses those images to draw itself. If you pass three, the second image is a stretchable center.

The `offset` parameter controls how far into the scrollbar the track should be positioned (perhaps you have a one-line track for a slider look). For example, if you specify 3, your images appear 3 pixels in from the left edge of the scrollbar view. For horizontal scrollbars, this is the number down to nudge the track images.

The `topLimit` and `bottomLimit` parameters are used to set the limits of how far the thumb can travel. They are "bumper" limits. If you want to ensure the thumb can not travel anywhere less than 5 pixels from the top of your scrollbar, specify 5 for the `topLimit`.

Note: At present, horizontal scrollbar images have to be created with the same orientation as vertical ones. So you need to design your scrollbars horizontally, then rotate them 90 degrees clockwise before chopping them up and saving out the pieces.

Example

```
myScrollbar.setTrackInfo( 0, 2, 2, new Array( "images/topCap.png",
        "images/middle.png", "images/bottomCap.png" ) );
```

Availability

Available in version 3.0 or newer.

Security

specifies security attributes for an object

XML Name

`<security>`

JavaScript Name

Not available.

Attributes

`api`

Description

> The security block has been moved to the `widget.xml` file in version 4.0 or later. If your Widget requires a minimum version of 4.0 or later, the security block inside the `.kon` file will be ignored. A notice to this effect will be printed in the debug window.

The `security` block tells the engine what the Widget can and cannot do. It is used to enforce behaviors to protect the user from a Widget from stepping outside its bounds at times.

Availability

Available in version 3.1 or newer.

api

Yahoo! API this Widget wants to connect to

Usage

XML

Description

> The security block has been moved to the `widget.xml` file in version 4.0 or later. If your Widget requires a minimum version of 4.0 or later, the security block inside the `.kon` file will be ignored. A notice to this effect will be printed in the debug window.

The `api` element identifies to the engine the APIs a Widget wants to access. This list is presented to the user the first time the Widget tries to log in to their Yahoo! account. The user's Yahoo! credentials are only passed to the APIs listed in these <api> items.

Most of the APIs are known by the engine and so they automatically get a human-readable name in the security dialog that's presented. If there's a new API added that the engine doesn't know about, you can specify a `name` attribute that tells us what to display in the dialog. If we use the name you specify, we also put the host you specify in the dialog so the user can see exactly what's being accessed. If no name is specified, the host is displayed.

Example

```
<security>
   <!-- the notepad API the engine knows -->
   <api>api.notepad.yahoo.com</api>
   <!-- some future API below -->
   <api name="Yahoo! Foozball">foozball.yahoo.com</api>
</security>
```

Availability

Available in version 3.1 or newer.

Settings

specifies various settings for a Widget

XML Name

<settings>

JavaScript Name

Not available.

Description

The settings element contains one or more <setting> elements. Each setting has two attributes, name and value. The valid settings we currently support are:

Name	Legal Values	Default
debug	on, off, verbose	off
defaultTracking	rectangle, opacity	opacity
allowCustomObjectAttributes	true, false	false

Name	Legal Values	Default
dontRememberWindowPosition	true, false	false
defaultAntiAlias (Windows only)	system, smooth	smooth

The defaultAntiAlias setting is generally for use in Asian Widgets where our default setting of smooth can render Asian fonts illegible. By setting it to system, we apply whatever smoothing settings are specified in the user's Display Properties. This is only valid on Windows. On Mac OS X the system settings are always used.

The above settings replace the old <debug>, <defaultTracking>, and <option> elements. Those elements are considered deprecated in version 4.0 or later.

If you set your Widget's minimum version to 4.0, you will get deprecation notices in the debug window for the old deprecated elements, but they will still function. Be warned though that the engine will obey the last setting it sees. If you have a settings block followed by a <debug> element, the <debug> element will override any setting in the <settings> block.

Example

```
<settings>
    <setting name="defaultTracking" value="rectangle"/>
    <setting name="debug" value="on"/>
</settings>
```

Availability

Available in version 4.0 or newer.

Shadow

specifies shadow parameters for an object

XML Name

<shadow>

JavaScript Name

Shadow

Attributes

color
hOffset
opacity
vOffset

Description

The shadow element is currently only used in text and about-text items. It allows you to set a solid shadow on an item with a certain color and opacity. The hOffset and vOffset you specify are offsets from the object you are shadowing (currently, text).

Version Notes

In version 4.0 or newer, it is recommended to use the special CSS styles for shadows when used on text items instead of this object.

Availability

Available in version 2.1 or newer.

color

color of the shadow

Usage

JavaScript, XML

Description

Specifies the color of the shadow to cast.

Example

```
<shadow color="#333333"/>
```

Availability

Available in version 2.1 or newer.

hOffset

horizontal offset of the shadow

Usage

JavaScript, XML

Description

Specifies the horizontal offset from the original object to cast the shadow. A value of 1 would mean the shadow was offset 1 pixel to the right of the object.

Example

```
<shadow hOffset="1"/>
```

Availability

Available in version 2.1 or newer.

opacity

opacity of the shadow

Usage

JavaScript, XML

Description

Specifies the opacity of the shadow from 0 to 255, where 0 is completely transparent and 255 is completely opaque.

Example

```
<shadow opacity="255"/>
```

Availability

Available in version 2.1 or newer.

vOffset

vertical offset of the shadow

Usage

JavaScript, XML

Description

Specifies the vertical offset from the original object to cast the shadow. A value of 1 would mean the shadow was offset 1 pixel to below the object.

Example

```
<shadow vOffset="1"/>
```

Availability

Available in version 2.1 or newer.

Text

single-line text object

XML Name

```
<text>
```

JavaScript Name

```
Text
```

Attributes

anchorStyle
bgColor
bgOpacity
color
data
font
scrolling
shadow
size
style
truncation
wrap

Functions

fade()
moveTo()
slide()

Description

Text objects represent a single or multi-line text object in a Widget. They are the equivalent of a "static text" or "label" object in other environments.

anchorStyle

controls where the vOffset is aligned

Usage

JavaScript, XML

Description

This attribute allows you to control how a text object is aligned to its vOffset attribute. In versions prior to 4.0, text objects were always aligned by baseline, so the vOffset of a text object represented where the baseline of the text should be. This is still the default behavior. In version 4.0 and newer however, you can change the anchor to be the top-left of the object (which is the way all other objects are aligned) if that better suits your needs.

Keep in mind that using top-left alignment is not always the best choice, as font differences can affect where the actual baseline lands.

The two values you can use are `topLeft` and `baseline`.

Example

```
<text name="text1" data="Example Text" anchorStyle="topLeft"/>

text1.anchorStyle = "baseline";
```

Availability

Available in version 4.0 or newer.

bgColor

background color of a text object

Usage

JavaScript, XML

Description

Note: It is preferred that you use the CSS style to set the background color. Use `rgb()` or `rgba()` CSS-style colors. Using the `rgba()` style allows you to set the color and opacity simultaneously.

Sets the color of the background of a text object. Colors are specified as browser style hex RGB triplets. For example:

#FF0000

Example

```
<text name="text1" data="Example Text" bgColor="#FFFFFF"
    bgOpacity="150"/>

text1.bgColor = "#FFFFFF";
text1.bgOpacity = 150;

// in 4.0 or later...
text1.bgColor = "rgba( 255, 255, 255, 0.6 )";

// or even better...
text1.style.backgroundColor = "rgba( 255, 255, 255, 0.6 )";
```

Notes

This attribute is closely linked with the `bgOpacity` attribute; both should be set to get a visible result.

bgOpacity

opacity of the background of a text object

Usage

JavaScript, XML

Description

Note: It is preferred that you use the CSS style to set the opacity. Use `rgb()` or `rgba()` CSS-style colors. Using the `rgba()` style allows you to set the color and opacity simultaneously.

Sets the opacity of the background of a text object. Opacities are specified as a number between 0 and 255.

Notes

This attribute is closely linked with the `bgColor` attribute; both should be set to get a visible result.

See Also

Refer to the example above in `bgColor` for how to set the opacity.

color

color that the text object draws in

Usage

JavaScript, XML

Description

Note: It is preferred to use the CSS style to set the background color. Use `rgb()` or `rgba()` CSS-style colors. Using the `rgba()` style allows you to set the color and opacity simultaneously.

Sets the color of the text. Colors are specified as browser style hex RGB triplets. For example:

#00FF00

If you set the color and `bgColor` to the same value you won't be able to see the text.

Example

```
<text name="text1" data="Example Text" color="#F42DA6"/>

text1.color = "#F42DA6";

// in 4.0 or later...
text1.style.color = "#F42DA6";
```

data

text that the text object draws

Usage

JavaScript, XML

Description

The text to be displayed. Note that for a non-wrapping Text object, any new lines or carriage returns in the text are converted to spaces before display.

Example

```
<text data="Example Text"/>
```

font

font in which the text object is drawn

Usage

JavaScript, XML

Description

Note: The preferred method of setting the font is to use CSS styles instead of this attribute.

The name of the font to be used to render the text. If the specified font cannot be found, then the default system font is used. Separate multiple font names with commas to specify *fallbacks* (fonts that used in the event a preceding font isn't found on the user's system).

Example

```
<text data="Example Text" font="Palatino"/>

text1.font = "Monaco, Courier";
```

scrolling

direction and type of animated scrolling

Usage

JavaScript, XML

Description

The `scrolling` attribute can take values of `off` (the default), `left`, `right`, `autoLeft`, or `autoRight`. If set, the text in the object scrolls continuously in the direction specified, reappearing on the opposite edge as it disappears.

The "auto" variants only scroll if the text is too big for the area specified for its display (this is the most common use of scrolling, to make long text visible in a small space).

Example

```
<text data="Example Text" name="myLabel"
      scrolling="autoLeft"/>

myLabel.scrolling = "autoLeft";
```

shadow

sets shadow parameters for a text object

Usage

JavaScript, XML

Description

Note: The preferred method of setting the shadow is to use the special CSS styles.

You can specify a shadow to be displayed underneath a text object using the `shadow` attribute. To clear it, just set the `shadow` attribute to null. The shadow XML is the same as that described in the `<shadow>` section.

Example

```
<text data="Example Text" name="myLabel">
      <shadow hOffset="1" vOffset="1" color="#000000"/>
</text>

var s = new Shadow();
s.hOffset = 1;
s.vOffset = 1;
s.color = "#000000";
myLabel.shadow = s;
myLabel2.shadow = s;

// In version 4.0 or later...
<text data="Example Text" name="myLabel"
   style="-kon-shadow:black 1px 1px"/>

myLabel.style.KONShadowColor = "#333";
myLabel.style.KONShadowOffset = "1px 0px";
myLabel.style.KONShadow = "black 1px 0px";
```

Availability

Available in version 3.0 or newer.

size

font size for the text block

Usage

JavaScript, XML

Description

Note: The preferred method to set the font size is to use the CSS font-size attribute.

The point size for the text object.

Example

```
<text data="Example Text" name="myLabel" size="22"/>
myLabel.size = 22;

// version 4.0 or later...
<text data="Example" style="font-size: 22px"/>

myLabel.style.fontSize = "22px";
```


style

style of the text to display

Usage

JavaScript, XML

Description

The style in which to render the text. Style can be any combination of `italic`, `bold`, `narrow`, `expanded`, `condensed`, `smallcap`, `poster`, `compressed`, and `fixed`.

For example:

```
textObject.style = "bold, italic";
```

requests a bold, italic variation of the font named in the `font` attribute.

Notes

The font must have the requested variation or else the style is ignored. Most fonts support only two or three variations.

Windows OS only supports bold and italic. Mac only supports bold, italic, and condensed.

If your Widget's `minimumVersion` is set to 4.0 or newer, this attribute is replaced by the CSS `style` attribute, as described in Common Attributes and Functions.

Example

```
<text data="Example Text" name="myLabel" style="bold"/>

myLabel.style = 'italic';
```

truncation

specifies whether to truncate text with an ellipsis

Usage

JavaScript, XML

Values

`none`
`center`
`end`

Description

Note: The preferred method for truncation is to use the CSS style `-ywe-text-truncation`.

Normally, a text object draws without any truncation. If there is no room for the entire text object to draw, it gets clipped. This tag allows you to specify that if the width of the text object is too small for the text, truncate it intelligently using an ellipsis.

This tag only takes effect if there is a width specified for the text item, the text item is longer than the width, and no `scrolling` attribute is specified.

In version 4.0 or newer, it is preferred to use the CSS `style -ywe-text-truncation`.

Example

```
<text data="Example Text" name="myLabel" width="50"
```

```
   truncation="end"/>

myLabel.truncation = "none";

// version 4.0 or later
<text data="Example" width="50" style="-ywe-text-truncation:end"/>

myLabel.style.YWETextTruncation="end";
```

Availability

Available in version 2.1 or newer. "Center" truncation is available in version 3.0 or newer only.

wrap

specifies whether text should wrap or be single-line

Usage

JavaScript, XML

Description

You can enable word wrap on a text object using the wrap attribute. By default, this is set to false, and text objects are single-line only,

If you set this, several behaviors change:

The text acts as if the `anchorStyle` was set to `topLeft`.

Automatic text scrolling is disabled.

Whereas single-line text objects use `hAlign` to decide their justification, wrapped text objects only obey the text-align CSS style.

If you combine this with text truncation, the result is undefined due to platform differences.

While you can use a text area to do multiline text objects, it is preferred to use a text object, as they are lightweight and can be rotated and alpha-blended properly on both Windows OS and Mac OS X. They can also have a shadow applied to them, which text areas cannot.

Example

```
<text data="Example Text" name="myLabel" width="50" wrap="true"/>
myLabel.wrap = true;
```

Availability

Available in version 4.0 or newer.

fade()

fades in or fades out a text object

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. See Animation for more information.

Synopsis

`Text.fade(`*`start`*`, `*`end`*`, `*`duration`*`)`

Description

The `fade()` command causes a text object to fade from a starting to a finishing opacity. `duration` specifies the time (in tenths of a second) you want the animation to last for.

Example

```
newOpacity = 0;
myText.fade(myText.opacity, newOpacity, 6);
```

moveTo()

moves text from point a to point b using animation

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. See Animation for more information.

Synopsis

`Text.moveTo(`*`newX`*`, `*`newY`*`, `*`duration`*`)`

Description

The text's origin (`hOffset`, `vOffset`) is moved to the new coordinates specified by `newX` and `newY`. `duration` specifies the time (in tenths of a second) you want the animation to last for. The move of the object is animated (which is what makes this different from just changing `hOffset` and `vOffset`).

Example

```
myText.moveTo(50, 50, 3);
```

slide()

slides a text object in a specified direction and duration

Deprecation Notice

This function is obsolete with the advent of the animation APIs. It will likely be removed in a future release. To accomplish this effect, it is best to put the object into a frame and then move the object within it. The frame clips it out as appropriate.

Synopsis

`Text.slide(`*`direction`*`, `*`amountOfTextToConceal`*`, `*`duration`*`)`

Description

The `slide()` function is used when you want to slide a text object in a particular direction, and have it disappear into itself rather than just move. `duration` specifies the time (in tenths of a second) you want the animation to last for.

Example

```
myText.slide("up,left", 50, 8);
```

TextArea

block defining a textarea object and associated default attributes

XML Name

```
<textarea>
```

JavaScript Name

TextArea

Attributes

bgColor
bgOpacity
color
columns
data
editable
font
lines
onGainFocus
onKeyDown
onKeyPress
onKeyUp
onLoseFocus
secure
scrollbar
size
spellcheck
style
thumbColor

Functions

focus()
loseFocus()
rejectKeyPress()
replaceSelection()
select()

Description

The `textarea` block in the XML file defines the initial placement and mouse event scripts for an editable text object in a Widget.

`textarea` objects can also be created and destroyed dynamically with the JavaScript engine.

When you create more than one dynamic object with the same name, only the last object created receives attribute changing events with JavaScript. As a result you should make sure that you're calling each dynamic object a unique name so they can be referenced properly (using a JavaScript array is often a good way to achieve this).

You can remove a dynamic object after you create it using the JavaScript `delete` instruction.

JavaScript

```
newObjectName = new TextArea()
delete newObjectName
```

bgColor

background color of a textarea object

Usage

JavaScript, XML

Description

Sets the color of the background of a `textarea` object. Colors are specified as browser style hex RGB triplets. For example:

#FF0000

Note: This attribute is closely linked with the `bgOpacity` attribute; both should be set to get a visible result.

In version 4.0 or newer, you can use `rgb()` or `rgba()` CSS-style colors. Using the `rgba()` style allows you to set the color and opacity simultaneously. However, it is preferred to use the CSS style to set the background color.

Example

```
<textarea data="Example Text" bgColor="#FFFFFF"
     bgOpacity="150"/>

ta.bgColor = "#FFFFFF";
ta.bgOpacity = 150;

// in 4.0 or later...
ta.bgColor = "rgba( 255, 255, 255, 0.6 )";

// or even better...
ta.style.backgroundColor = "rgba( 255, 255, 255, 0.6 )";
```

bgOpacity

opacity of the background of a textarea object

Usage

JavaScript, XML

Description

Sets the opacity of the background of a `textarea` object. Opacity is specified as a number between 0 and 255.

Notes

This attribute is closely linked with the `bgColor` attribute; both should be set to get a visible result.

In version 4.0 or newer, you can set the color and opacity at the same time using CSS styles. See bgColor above.

See the example above in bgColor for how to set the opacity.

color

color that the text draws in

Usage

JavaScript, XML

Description

Sets the color of the text. Colors are specified as browser style hex RGB triplets. For example:

#00FF00

In version 4.0 or newer, you can use `rgb()` or `rgba()` CSS-style colors. Using the `rgba()` style allows you to set the color and opacity simultaneously. However, it is preferred to use the CSS style to set the background color.

Note: If you set the `color` and `bgColor` to the same value, you won't be able to see the text.

Example

```
<textarea data="Example Text" color="#F42DA6"/>

ta.color = "#F42DA6";

// in 4.0 or later...
ta.style.color = "#F42DA6";
```

columns

number of columns wide to make the object

Usage

JavaScript, XML

Description

Instead of giving a width and height for `textarea` objects, their size can be specified in terms of a number of columns and lines of text in the current font.

Example

```
<textarea columns="40" lines="10"/>

ta.columns = 40;
```

Notes

Using a proportional font makes the number of columns approximate.

data

text that the textarea object contains

Usage

JavaScript, XML

Description

The text to be edited. This is optional. If omitted, the user is presented with an empty text entry field.

Example

```
<textarea data="Example Text"/>
```

editable

sets text as editable or display only

Usage

JavaScript, XML

Description

Set `editable` to false to make the `textarea` display only. The default is true.

If your `textarea` will never be editable, it's generally preferable to use a wrapped text object.

Example

```
<textarea data="Example Text" editable="false"/>

ta1.editable = false;
```

font

font that the text area uses

Usage

JavaScript, XML

Description

The name of the font to be used to render the text. If the specified font cannot be found, then the default system font is used. Separate multiple font names with commas to specify *fallbacks* (fonts used in the event that a preceding font isn't found on the user's system).

In 4.0 or newer, it is preferred to use CSS styles to set the font instead of this attribute.

Example

```
<textarea data="Example Text" font="Palatino, Times"/>

ta.font = "Palatino, Times";
```

lines

number of lines high to make the object

Usage

JavaScript, XML

Description

Instead of giving a `width` and `height` for `textarea` objects, their size can be specified in terms of a number of columns and lines of text in the current font.

Specifying a value of 1 for lines changes the behavior of the `textarea` object slightly. Instead of wrapping, the text scrolls sideways when then the edge of the object is reached while typing.

Example

```
<textarea columns="40" lines="10"/>

ta.lines = 10;
```

onGainFocus

installs handler for gainfocus event

For More Information

See gainfocus.

onKeyDown

installs handler for keydown event

For More Information

See keydown.

onKeyPress

installs handler for keypress event

For More Information

See keypress.

onKeyUp

installs handler for keyup event

For More Information

See keyup.

onLoseFocus

installs handler for losefocus event

For More Information

See losefocus.

secure

sets the text area to display bullets instead of text

Usage

JavaScript, XML

Description

This attribute is used to mimic a password field, where the user cannot see the actual text being typed, but rather just a series of bullet characters (small circles). This should generally be used with one row of text.

Example

```
<textarea data="hello!" secure="true"/>

ta.secure = true;
```

Availability

Available in version 2.1 or newer.

scrollbar

controls the display of a scrollbar on the text area

Usage

JavaScript, XML

Description

By default, a text area displays a vertical scrollbar. Use this attribute to turn it off.

Example

```
<textarea data="Example Text" scrollbar="false"/>

ta1.scrollbar = true;
```

size

font size for the text area block

Usage

JavaScript, XML

Description

The point size for the text area object.

In version 4.0 or newer, it is preferred to use the CSS font-size attribute.

Example

```
<textarea data="Example Text" name="ta1" size="22"/>

ta1.size = 33;
```

spellcheck

controls continuous spellchecking in the text area

Usage

JavaScript, XML

Description

By default, a text area highlights spelling errors as the user types. Use this attribute to turn it off.

Example

```
<textarea data="Example Text" spellcheck="false"/>

ta1.spellcheck = true;
```

style

font style for the text area block

Usage

JavaScript, XML

Description

The style in which to render the text. Style can be any combination of `italic`, `bold`, `narrow`, `expanded`, `condensed`, `smallcap`, `poster`, `compressed`, and `fixed`.

For example:

```
textAreaObject.style = "bold,italic";
```

requests a bold, italic variation of the font named in the `font` attribute.

Notes

The font must have the requested variation or else the style is ignored. Most fonts support only two or three variations.

Windows OS only supports bold and italic. Mac only supports bold, italic, and condensed.

If your Widget's `minimumVersion` is set to 4.0 or newer, this attribute is replaced by the CSS `style` attribute, as described in Common Attributes and Functions.

Example

```
<textarea data="Example Text" name="ta1" style="bold"/>

ta1.style = 'italic';
```

thumbColor

thumb color of the scrollbar

Usage

JavaScript, XML

Description

The `thumbColor` attribute is used to control the tint of the scrollbar thumb if your text area has a scrollbar specified. The default thumb in the standard scrollbar is a medium gray. The color you specified is applied using colorization.

To clear the current color completely, you can set it to null in JavaScript.

Example

```
<scrollbar ... thumbColor="#333366"/>

myScrollbar.thumbColor = "#333366";
myScrollbar.thumbColor = null;
```

Availability

Available in version 3.0 or newer. Currently only available on Windows OS.

focus()

makes the current text area the focus of key presses

Synopsis

`TextArea.focus()`

Description

The `focus()` function makes the given text area be the one to which typed keys are sent. It is most useful when there are several text areas on a Widget and you want to move the insertion point from one to another. The text area must be editable for this to be effective.

When focus is acquired, the `onGainFocus` action is called for the text area in version 3.0 or newer.

Example

```
mytextarea.focus();
```


loseFocus()

relinquishes keyboard focus if the text area currently is the focus

Synopsis

```
TextArea.loseFocus()
```

Description

The `loseFocus()` function releases the keyboard focus from the text area if the text area is the current focus (through a call to `focus()`). This function is useful for clearing the focus, e.g., after the user enters a value in a text area used as a search field. There is no need to call this when the text area loses the focus, as it automatically loses focus in that case.

When focus is lost, the `onLoseFocus` action is called for the text area in version 3.0 or newer.

Example

```
mytextarea.loseFocus();
```

Availability

Available in version 3.0 or newer.

rejectKeyPress()

controls whether keys are accepted by a text area

Synopsis

```
TextArea.rejectKeyPress()
```

Description

The `rejectKeyPress()` function is used in the `<onKeyPress>` action to control whether the current key press affects the text area.

Example

```
<onKeyPress>
<<![CDATA[
  // Convert all typed characters to uppercase
  var key = event.key;

  if (key.charCodeAt(0) >= "A".charCodeAt(0) &&
      key.charCodeAt(0) <= "z".charCodeAt(0))
  {
    // Tell the text area to ignore this keyPress as
    // we are replacing it with our own
    ta1.rejectKeyPress();

    // Append an upper case copy of the key pressed
    // (the insertion point is a 0 length selection)
    ta1.replaceSelection(key.toUpperCase());
  }
]]
</onKeyPress>
```

replaceSelection()

replaces the current selection in a text area with a string

Synopsis

```
TextArea.replaceSelection(string)
```

Description

The `replaceSelection()` function replaces the current selection in the text area with the given string. Note that the "cursor" or "insertion point" is a selection of zero length, so if nothing is selected in the text area, using `replaceSelection()` has the effect of inserting the given string at the current cursor position.

Example

```
replacement = "new text";
mytextarea.replaceSelection(replacement);
```

select()

selects text in the text area

Synopsis

```
TextArea.select(start, end)
```

Description

The `select` function changes the selection in the text area. Characters from start to end are selected. As a special case, the position –1 means "the end of the text," thus:

```
mytextarea.select(0, -1);
```

selects all the text.

To set the position of the "cursor" or "insertion point" specify a selection of zero length, for example:

```
mytextarea.select(10, 10);
```

To set the insertion point after any text already in the text area you would use:

```
mytextarea.select(-1, -1);
```

When the insertion point is set, the contents of the text area are scrolled so it is visible to the user.

Example

```
mytextarea.select(5, 15);
```

Timer

a repeating action

XML Name

```
<timer>
```

JavaScript Name

```
Timer
```

Attributes

```
interval
ticking
```


onTimerFired

Functions

reset()

Description

Timer objects allow you to perform a task at a periodic interval (e.g., every 5 seconds) or can be used to simply fire once at a later time. They are a replacement for the older onTimer triggers in actions. They allow you to create multiple timers all running on different frequencies. They can also be started and stopped without having to delete them.

The intervals are not guaranteed to be exact. Timers run "cooperatively," meaning that they fire when the Widget is not busy doing other things. You cannot do high-precision time-based actions in a timer.

Availability

Available in version 2.0 or newer.

interval

frequency of firing for the timer

Usage

JavaScript, XML

Description

This is simply the interval the timer should fire, in seconds. It can be expressed as a decimal, so if you want a timer to fire every half second, specify 0.5 for this attribute. Each time the timer fires, it executes the JavaScript in the onTimerFired attribute.

Example

```
<timer name="myTimer" interval="1.0"/>

myTimer.interval = 1.0;
```

Availability

Available in version 2.0 or newer.

ticking

determines if the timer is running

Usage

JavaScript, XML

Description

This allows you to turn a timer on and off by setting it to true and false, respectively. If you want to disable a timer for a while, just set ticking to false. Later, set it to true, and it starts firing again. Once restarted, the next time it fires will be "now" plus the interval. So if you have a one second timer, it fires one second after you set ticking to true.

Example

```
<timer name="myTimer" ticking="false"/>
```

```
myTimer.ticking = true;
```

Availability

Available in version 2.0 or newer.

onTimerFired

installs handler for timerfired event

For More Information

See timerfired.

Functions

reset()

reset()

restarts countdown of timer

Synopsis

```
Timer.reset()
```

Description

The `reset()` function causes a timer to start its countdown over. For example, if you had a timer that was on a one-minute interval, and 30 seconds after it started to run you called `reset()`, it would start its one minute countdown over again. So instead of firing in 30 seconds, it would start over and fire in one minute.

A good example of when this is useful is if you were trying to implement some sort of idle timer. Let's say you wanted to do something in your Widget if the user hasn't interacted with it in 15 seconds. If the user clicked your Widget, you could start a timer that will fire in 15 seconds. If the user clicks again, you can reset the timer, starting the 15 second countdown over. Eventually, after not clicking for 15 seconds, the timer will fire.

Example

```
myTimer.reset();
```

Availability

Available in version 2.0 or newer.

Web

block defining a web object

XML Name

<web>

JavaScript Name

Web

Attributes

scrollX
scrollY
url

html
bgColor
autoVScrollBar
autoHScrollBar
base
vScrollBar
hScrollBar
title
statusBar

Functions

stopLoading()
reload()

Description

The web block in the XML file defines the initial placement and event handlers for an HTML rendering surface.

web objects can also be created and destroyed dynamically with the JavaScript engine.

When you create more than one dynamic object with the same name only the last object created receives attribute changing events with JavaScript. As a result you should make sure that you're calling each dynamic object a unique name so they can be referenced properly (using a JavaScript array is often a good way to achieve this).

You can remove a dynamic object after you create it using the JavaScript delete instruction.

To pull content from your Widget bundle use the konfabulator:/// scheme to refer to media inside of your bundle.

From within the HTML environment a global variable Konfabulator is made available meant to point back to the konfabulator environment. Under strict security only a small subset of the Konfabulator environment is made available, however when the HTML content all comes from inside your widget bundle no security restrictions are in place and you may freely call between the Konfabulator object (from the HTML) and the base attribute (from the Widget).

Availability

Available in version 4.5 or newer.

scrollX

sets the current scroll vertical position of the web page, if scrollable

Usage

JavaScript, XML

Description

If a vScrollBar and/or autoVScrollBar are set, then the web view has the ability to scroll. When the page is too long to be viewed in a single page, setting scrollX causes the view to scroll.

Positive values flow downwards.

Example

```
webObj.scrollx = 100;
```

Availability

Available in version 4.5 or newer.

scrollY

sets the current scroll horizontal position of the web page, if scrollable

Usage

JavaScript, XML

Description

If a hScrollBar and/or autoHScrollBar are set, then the web view has the ability to scroll. When the page is too wide to be viewed in a single page, setting scrollY causes the view to scroll.

Positive values flow to the right.

Example

```
webObj.scrolly = 100
```

Availability

Available in version 4.5 or newer.

url

sets the base url of the current document

Usage

JavaScript, XML

Description

The url attribute is either the source of the document as fetched, or it is the base url when the html attribute is used. As with all media you can have this refer inside your Widget bundle (for example .html that are shipped with your Widget) by referring to it via konfabulator:///your_file.html.

The value returned may not be the same as set if a redirect occurs. See also webresourcerequested.

Example

```
<web url="http://yahoo.com"/>
```

Availability

Available in version 4.5 or newer.

html

sets the content of the web page directly

Usage

JavaScript, XML

Description

Setting this value removes all previously gathered data and the innerHTML of the document is set accordingly. Script tags are executed here as well.

The value returned may not the same as set because the accessor will return simply the innerHTML as currently viewed by the web DOM.

Example

```
webObj.html = "<body><H1>Hello world</H1></body>";
```

Availability

Available in version 4.5 or newer.

bgColor

background color of a textarea object

Usage

JavaScript, XML

Description

Sets the color of the background of a web object. Colors are specified as browser style hex RGB triplets. For example:

#FF0000

In version 4.0 or newer, you can use rgb() or rgba() CSS-style colors. Using the rgba() style allows you to set the color and opacity simultaneously. However, it is preferred to use the CSS style to set the background color.

Example

```
<web html="<body><H1>Hello world</H1></body>" bgColor="#FFFFFF"/>

webObj.bgColor = "#FFFFFF";
webObj.bgOpacity = 150;
// in 4.0 or later...
webObj.bgColor = "rgba( 255, 255, 255, 0.6 )";
// or even better...
webObj.style.backgroundColor = "rgba( 255, 255, 255, 0.6 )";
```

Availability

Available in version 4.5 or newer.

autoVScrollBar

tells the web view to automanage the vertical scrollbar position

Usage

JavaScript, XML

Description

This value indicates that you want the web view to manage the scrollbar positions. If a custom scrollbar is set (vScrollBar attribute) then that scrollbar is managed. If no vScrollBar is set, then one is created automatically on first need and managed from then on.

This attribute defaults to true.

Example

```
webObj.autoVScrollBar = true;
```

Availability

Available in version 4.5 or newer.

autoHScrollBar

tells the webview to automanage the horizontal scrollbar position

Usage

JavaScript, XML

Description

This value indicates that you want the web view to manage the scrollbar positions. If a custom scrollbar is set (hScrollBar attribute) then that scrollbar is managed. If no hScrollBar is set, then one is created automatically on first need and managed from then on.

This attribute defaults to true.

Example

```
webObj.autoHScrollBar = true;
```

Availability

Available in version 4.5 or newer.

base

allows read/write access to the DOM of the web view

Usage

JavaScript, XML

Description

If your widget requires access to the web view's DOM the entire environment is available from this variable. The base attribute is the globalObject of the web view, and allows finding nodes, calling functions, etc. To cross the other direction (HTML talking to the widget functions) you can use the global object Konfabulator which refers directly to the globalObject inside of the widget.

Example

```
var child = webObj.base.firstChild;
```

Availability

Available in version 4.5 or newer.

vScrollBar

vertical scrollbar for this web view

Usage

JavaScript, XML

Description

The vScrollBar attribute of a web defines what scrollbar object should control the vertical scrolling. When expressed in XML you specify the name of the <scrollbar> object you wish to bind to the web for its vScrollBar. If the scrollbar object does not exist an error appears in the Widget's debug window.

Attaching a scrollbar automatically sets up communication between the frame and the scrollbar. If autoVScrollBar is set, the geometry of the scrollbar is automatically managed as well.

Example

```
webObj.vScrollBar = myScrollbarV;
```

Availability

Available in version 4.5 or newer.

hScrollBar

horizontal scrollbar for this web view

Usage

JavaScript, XML

Description

The hScrollBar attribute of a web defines what scrollbar object should control the horizontal scrolling. When expressed in XML you specify the name of the <scrollbar> object you wish to bind to the web for its hScrollBar. If the scrollbar object does not exist an error appears in the Widget's debug window.

Attaching a scrollbar automatically sets up communication between the frame and the scrollbar. If autoHScrollBar is set, the geometry of the scrollbar is automatically managed as well.

Example

```
webObj.hScrollBar = myScrollbarH;
```

Availability

Available in version 4.5 or newer.

title

read-only access to the title of the current HTML document

Usage

JavaScript, XML

Description

The title attribute allows direct access to the title of the current HTML document (whether from url or html). See also onWebTitleChanged.

Example

```
var webTitle = webObj.title;
```

Availability

Available in version 4.5 or newer.

statusBar

read-only access to the text value of the status bar

Usage

JavaScript, XML

Description

Periodically a web page may request setting the statusBar value. This can be done from javascript code in the web page, or by hovering over a link. See also onWebStatusBarChanged.

Example

```
var statusText = webObj.statusBar;
```

Availability

Available in version 4.5 or newer.

onWebAlert

installs handler for webalert event

For More Information

See webalert.

onWebConfirm

installs handler for webconfirm event

For More Information

See webconfirm.

onWebCreateWindow

installs handler for webcreatewindow event

For More Information

See webcreatewindow.

onWebException

installs handler for webexception event

For More Information

See webexception.

onWebLinkClicked

installs handler for weblinkclicked event

For More Information

See weblinkclicked.

onWebPageLoadComplete

installs handler for webpageloadcomplete event

For More Information

See webpageloadcomplete.

onWebPrompt

installs handler for webprompt event

For More Information

See webprompt.

onWebResourceLoadComplete

installs handler for webresourceloadcomplete event

For More Information

See webresourceloadcomplete.

onWebResourceRequested

installs handler for webresourcerequested event

For More Information

See webresourcerequested.

onWebStatusBarChanged

installs handler for webstatusbarchanged event

For More Information

See webstatusbarchanged.

onWebTitleChanged

installs handler for webtitlechanged event

For More Information

See webtitlechanged.

onWebURLChanged

installs handler for weburlchanged event

For More Information

See weburlchanged.

stopLoading()

instructs the web view to stop loading the current page

Synopsis

```
stopLoading()
```

Description

You can call stopLoading() at any time to cancel the content that is currently loading.

Example

```
mywebView.stopLoading();
```

Availability

Available in version 4.5 or newer.

reload()

instructs the web view to reload its current contents

Synopsis

```
reload()
```

Description

You may call reload() at anytime and the current content is reloaded, whether the current view is of data set in the html attribute or from a document as expressed by the url attribute.

Example

```
myWebView.reload();
```

Availability

Available in version 4.5 or newer.

Widget

block defining the scope of the Widget

Attributes

dockOpen
locale
minimumVersion
onDockClosed
onDockOpened
onGainFocus
onKonsposeActivated
onKonsposeDeactivated
onLoad
onLoseFocus
onPreferencesCancelled
onPreferencesChanged
onRunCommandInBgComplete
onScreenChanged
onTellWidget
onUnload
onWakeFromSleep
onWillChangePreferences
onYahooLoginChanged
option
version
visible

Deprecated Attributes

author
company
copyright
debug
defaultTracking
image

onIdle
onTimer
requiredPlatform

Functions

extractFile()
getLocalizedString()
setDockItem()

Description

The outermost scope in the XML file is defined by the `widget` block. This groups together all the objects that make up the Widget, and is accessible via the global object named, surprisingly, "widget." Currently all attributes are read-only.

Deprecation Notice

The deprecated attributes `author`, `company`, `copyright`, `version`, `image` and `requiredPlatform` are now part of the `widget.xml` metadata file. (see Widget Metadata for more information). The `defaultTracking` attribute is now part of the `settings` block (see Settings for more information). `onTimer` has been replaced by Timer objects, but `onIdle` should no longer be used for performance reasons.

author

specifies the author's name

Usage

JavaScript, XML

Description

You can provide this optional piece of information for your Widget. In version 3.0 and newer, this is used to display in our "first run" security dialog so that people can see who wrote the Widget (along with the `company` attribute). In the future, this information will be used in other areas of the interface, so it's a good idea to supply author, company, and copyright attributes.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). However, if your Widget supports releases before 4.0, you might want to leave this attribute in place so that other versions can find it.

Availability

Available in version 3.0 or newer.

company

specifies the company name

Usage

JavaScript, XML

Description

You can provide this optional piece of information for your Widget. In version 3.0 this is used to display in our "first run" security dialog so that people can see what company or organization is publishing the Widget. In the future, this information will be used in other areas of the interface, so it's a good idea to supply author, company, and copyright attributes.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). However, if your Widget supports releases before 4.0, you might want to leave this attribute in place so that other versions can find it.

Availability

Available in version 3.0 or newer.

copyright

specifies the Widget's copyright string

Usage

JavaScript, XML

Description

You can provide this optional piece of information for your Widget. In the future, this information will be used in other areas of the interface, so it's a good idea to supply author, company, and copyright attributes.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). However, if your Widget supports releases before 4.0, you might want to leave this attribute in place so that other versions can find it.

Availability

Available in version 3.0 or newer.

debug

controls whether the debug console is shown for this Widget

Usage

JavaScript, XML

Description

> This attribute is deprecated in version 4.0 and later. Please use `<settings>` in Widgets that have a minimum version of 4.0 or later.

If you need to enable or suppress debug output, add this block inside the `widget` block.

```
<debug>errors</debug>
```

Turns Widget debug information on if an error occurs while the Widget is running. Errors can be JavaScript or Widget Engine runtime messages. This is the default.

```
<debug>on</debug>
```


Turns Widget debug information on when the Widget is opened (useful when developing a Widget).

```
<debug>off</debug>
```

Keeps Widget debug information off when the Widget is opened even if errors occur. This mode should only be used when a Widget is thoroughly debugged. Note that if a Widget generates 10 errors, the debug console is displayed no matter what the setting of this option is.

```
<debug>verbose</debug>
```

Turns Widget debug information on when the Widget is opened and causes information about object actions and other automatically triggered events to be displayed (useful when developing a Widget).

Notes

In version 2.0.1 or newer, you can hold control-shift while selecting the Gear menu and a "Debug Mode" option is presented. Checking that enables debugging for any Widget launched after that option is turned on. This means you don't necessarily need to set the `debug` attribute in a Widget any longer.

defaultTracking

sets the default tracking style for images

Usage

XML

Description

The `defaultTracking` attribute specifies the default cursor tracking style for images. It can be either `opacity` (the default) or `rectangle` which makes images clickable anywhere inside their bounding rectangle instead of only on their nontransparent parts.

See `Image` for more details about tracking

Example

```
<widget defaultTracking="rectangle">
     ...
</widget>
```

dockOpen

indicates whether the dock is open or closed

Usage

JavaScript

Description

This attribute is a simple Boolean indicating whether the dock is opened or closed. If closed, you should avoid calling `setDockItem`, as it will just cause unneeded CPU usage. You can use this in concert with the `onDockOpened` and `onDockClosed` handlers.

Availability

Available in version 4.0 or newer.

image

specifies the Widget's icon/image path for the security dialog

Usage

JavaScript, XML

Description

You can provide this optional piece of information for your Widget. This attribute contains the relative path to a 150x150 pixel image to display to represent your Widget. Currently, this image is only displayed in our standard Security dialog when an unknown Widget is run for the first time, or has been modified since last run. In the future, it might be used in other areas of the interface, so it's a good idea to supply an image for your Widget along with author/company/copyright information.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). However, if your Widget supports releases before 4.0, you might want to leave this attribute in place so that other versions can find it.

Availability

Available in version 3.0 or newer.

locale

the language/locale the Widget is currently running in

Usage

JavaScript, XML

Description

You can use this attribute to help determine what language and locale your Widget is running in. The format of the returned string is:

```
<language>[_<locale>]
```

Where the `language` parameter is the lowercase ISO language code and the `locale` parameter is the uppercase ISO locale code. For example:

```
en
en_US
ja
```

The locale portion is only present for some languages. For example, there might be multiple variants of English (US, UK, etc.). You can use this information if you care to, or if you want to base any decisions in your presentation based on language alone, you can just look at the first two characters and be done with it. You access this attribute through the global `widget` object.

Example

```
var locale = widget.locale;

if ( locale.substr( 0, 2 ) == "en" )
    print( "It's English" );
```

Availability

Available in version 3.1 or newer.

minimumVersion

minimum version of the Widget Engine that is required to run this Widget

Usage

JavaScript, XML

Description

You must specify the `minimumVersion` attribute of your widget as `"4.0"` in order for it to behave as expected in version 4.0 of the engine. You should also specify it in your `widget.xml` file (see Widget Metadata).

Specifying `minimumVersion` for a Widget causes the Widget Engine to check it against the version of the engine that is currently running. If the current version is less than the version specified, an error message is displayed to the user and the Widget won't run.

If you omit the `minimumVersion` attribute, a value of 2.0 is assumed. Version 2.0 widgets have markedly different behavior to 4.0 widgets.

Example

```
<widget minimumVersion="1.5">
     ...
</widget>
```

Starting in version 3.0 and later, this attribute also tells the Widget Engine that it can enable new behaviors. We use this as a clue that the Widget has been modified to work with the minimum version specified and as such, has been adjusted to behave correctly with that version. With this mechanism, we allow older Widgets to continue to run unmodified, but newer or modified Widgets that advertise their support for 3.0 might require modifications to run in the new environment. This allows us to change the way things work without breaking existing Widgets.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). That said, you should generally continue to put this in your `.kon` file as well so that versions prior to 4.0 will know not to run it.

Availability

Available in version 1.5 or newer. Available in JavaScript in version 3.0 or newer.

onDockClosed

installs handler for dockclosed event

For More Information

See dockclosed.

onDockOpened

installs handler for dockopened event

For More Information

See dockopened.

onGainFocus

installs handler for gainfocus event

For More Information

See gainfocus.

onIdle

installs handler for idle event

For More Information

See idle.

onKonsposeActivated

installs handler for konsposeactivated event

For More Information

See konsposeactivated.

onKonsposeDeactivated

installs handler for konsposedeactivated event

For More Information

See konsposedeactivated.

onLoad

installs handler for load event

For More Information

See load.

onLoseFocus

installs handler for losefocus event

For More Information

See losefocus.

onPreferencesCancelled

installs handler for preferencescancelled event

For More Information

See preferencescancelled.

onPreferencesChanged

installs handler for preferenceschanged event

For More Information

See preferenceschanged.

onRunCommandInBgComplete

installs handler for runcommandinbgcomplete event

For More Information

See runcommandinbgcomplete.

onScreenChanged

installs handler for screenchanged event

For More Information

See screenchanged.

onTellWidget

installs handler for tellwidget event

For More Information

See tellwidget.

onTimer

installs handler for timer event

For More Information

See timer.

onUnload

installs handler for unload event

For More Information

See unload.

onWakeFromSleep

installs handler for wakefromsleep event

For More Information

See wakefromsleep.

onWillChangePreferences

installs handler for willchangepreferences event

For More Information

See willchangepreferences.

onYahooLoginChanged

installs handler for yahoologinchanged event

For More Information

See yahoologinchanged.

option

various Widget options

Usage

JavaScript, XML

Description

> This attribute is deprecated in version 4.0 and later. Please use `<settings>` in Widgets that have a minimum version of 4.0 or later.

These options affect the behavior of the Widget as a whole.

`<option>allowCustomObjectAttributes</option>`

When this Widget attribute is specified, custom object attributes are allowed, and will not trigger debug errors.

`<option>dontRememberWindowPosition</option>`

This tells the Widget Engine to not save the window position of this Widget when it is closed. Normally, the positions of all Widgets are remembered between invocations of the Widget Engine so the user can lay out their desktop just as they like but some kinds of Widgets work better by positioning themselves programmatically each time they run.

Availability

Available in version 1.5 or newer. Available in JavaScript in version 3.0 or newer.

requiredPlatform

specifies if this Widget requires a particular platform to run

Usage

JavaScript, XML

Description

While it is preferred that Widgets run on all platforms that the Widget Engine supports, at times Widgets might make use of highly specific platform features (such as COM on Windows OS). To indicate that your Widget requires a particular platform, you can specify this attribute with the values `macintosh` or `windows`.

Availability

Available in version 1.8 or newer in XML. Available in version 3.0 or newer in JavaScript.

version

version number of the Widget

Usage

JavaScript, XML

Description

You can specify the current version of your Widget in this attribute. If you make use of the `<about-version>` object in your `<about-box>` object, this is the information that gets displayed.

In version 4.0 or newer, if you have a widget.xml file present in your Widget, this attribute will be ignored if it is in your `.kon` file (a warning is printed to the debug window in this case). However, if your Widget supports releases before 4.0, you might want to leave this attribute in place so that other versions can find it.

Availability

Available in version 1.5 or newer in XML. Available in version 3.0 or newer in JavaScript.

visible

determine widget visibility

Usage

JavaScript

Description

The visible attribute allows you to tell whether your Widget is currently hidden. Each Widget can be hidden or shown by the user, either by choosing "Hide Widget" from the context menu, or by clicking an appropriate button in the Widget dock. The user can also specify that a Widget be initially hidden when it launches. In this case, all windows in the Widget are hidden, and the Widget cannot show any new windows unless this attribute is set to true. Therefore, a window's visibility is controlled by its own visibility as well as the Widget's visibility.

Example

```
print( widget.visible );
```

Availability

Available in version 4.0 and newer.

createWindowFromXML()

closes a window

Synopsis

```
Widget.createWindowFromXML( string | DOMDocument )
```

Description

This function creates a window from a separate XML document outside your `.kon` file. The root level element tag should be `<window>`.

Example

```
var d  = XMLDOM.parseFile( "window.xml" );
var w = widget.createWindowFromXML( d );
w.visible = true;
```

Availability

Available in version 4.5 or newer.

extractFile()

copies a file out of a Widget and into the Widget's data folder

Synopsis

```
path = Widget.extractFile( file )
```

Description

This function allows you to copy a file out of a Widget package and into a Widget's "widget data" folder. This is particularly needed in flat-file Widgets in 3.1 or newer in order to use items that must exist as files in the filesystem (e.g., DLLs). Your Widget does not need to be a flat-file Widget to use this function.

You specify the relative path of the file you want to extract and the path to the extracted file is returned. This function is accessed off the global `widget` object.

Example

```
var extPath = widget.extractFile( "Resources/myLibrary.dll" );
```

Availability

Available in version 3.1 or newer.

getLocalizedString()

returns the localized string for a given key

Synopsis

```
localString = Widget.getLocalizedString( key )
```

Description

This function returns the localized string for a given key. You call this with the global `widget` object.

Example

```
var welcome = widget.getLocalizedString( "welcome_msg" );
```

Availability

Available in version 3.1 or newer.

setDockItem()

customizes the look of a running Widget in the Widget dock

Synopsis

```
widget.setDockItem( xml [,transition] )
```


Description

This function allows you to set custom graphics or text to represent your Widget in the Widget dock while it is running. You might use this to show cursory information about the state of your Widget (e.g., temperature, time, or number of unread messages).

The xml parameter can be passed as a raw XML string, or as a valid DOM document.

You can specify the kind of transition that should occur when your info is displayed. The set of valid transitions are none, fade, wipe-left, wipe-right, push-left, and push-right.

Only call this function if the dock is visible. You can check the current visibility of the dock by inspecting the value of widget.dockOpen. If this is true, it is OK to call this function; otherwise, you are wasting CPU resources. You should listen to the `onDockOpened` and `onDockClosed` messages so that you are informed if the dock is opened or closed so you can either start or stop any periodic timer you might be using to set your dock item.

Example

```
var xml = generateMyDockInfo();

widget.setDockItem( xml, "fade" );
```

Availability

Available in version 3.1 or newer.

Window

block that defines the main window of the Widget

XML Name

`<window>`

JavaScript Name

`Window`

Attributes

`level`
`locked`
`onFirstDisplay`
`onGainFocus`
`onLoseFocus`
`root`
`shadow`
`title`

Functions

`focus()`
`getBestDisplay()`
`moveTo()`
`recalcShadow()`

Description

The `window` block describes the size and position of a Widget window. This window is always transparent and only the images and text objects you put in it are visible to the user.

Multiple Windows

Starting in version 2.0, you can have multiple windows in your Widgets. You attach the objects (images, text, etc.) to your window by specifying the variable of a window as the `window` attribute of the object. This way, each object knows where it lives.

If you give your window a name (either in the XML or in a JavaScript constructor), the Widget Engine tracks the windows position and save it to preferences automatically for you. Unnamed windows do not have their preferences saved (they are usually a small transient window such as a bezel so there's no point).

Each window maintains its own window level. However, if the user sets the Widget's level in the Preferences dialog to a value, all windows are set to that value. If you don't wish a window to take on such a value, reset it to what you like in an `onPreferencesChanged` handler.

level

position of window relative to other windows

Usage

JavaScript, XML

Description

This attribute can have one of the following values: `konspose`, `desktop`, `below`, `normal`, `topmost`, or `floating`. It specifies how the window behaves with respect to other windows on the desktop, whether it appears below others or floats above everything (`konspose` means it only appears in Heads-Up Display mode). The default is `normal`.

In versions prior to 4.0, you could only set this value via JavaScript.

Example

```
<window title="My Widget">
     <level>below</level>
</window>

myWindow.level = 'normal';
```

locked

disables dragging of a window

Usage

JavaScript, XML

Description

You can set or inspect a window's `locked` attribute to control whether the user can drag a window or not. This setting is largely controlled by a Widget's window preferences in the Preferences dialog. This attribute is not expressed in the XML interface in versions prior to 4.0.

Example

```
myWindow.locked = true;
```


onFirstDisplay

installs handler for firstdisplay event

For More Information

See firstdisplay.

onGainFocus

installs handler for gainfocus event

For More Information

See gainfocus.

onLoseFocus

installs handler for losefocus event

For More Information

See losefocus.

root

root view of the window

Usage

JavaScript

Description

This attribute contains the root view of the window. This value is never null and contains all the views at the top level of the window. To add a view to a window, you can either call `root.addSubView()` or set an object's `window` attribute. In both cases the view is a child view of the root.

In version 4, use DOM functions. See notes below.

Example

```
var root = myWindow.root;
```

Availability

Available in version 3.0 or newer. The Widget's minimum version must be set to at least version 3.0 for this attribute to exist.

Notes

When running under 4.0 or newer and using the DOM-style APIs to walk the DOM tree, the root is not accessible. The parent of any object in the root is considered to be the window. Likewise, the `firstChild` of the window object is considered to be the `firstChild` of the root view. For all practical purposes, you should consider the root a nonentity with the advent of the DOM APIs.

shadow

determines whether a window casts a shadow

Usage

JavaScript, XML

Description

Controls whether the Widget has a shadow.

Example

```
<window title="My Widget">
     <shadow>false</shadow>
</window>

myWindow.shadow = true;
```

Platform Notes

On Windows OS, window shadows are available in version 3.1 or newer. You must have your Widget's `minimumVersion` set to 3.1 for this attribute to function.

As of Mac OS X 10.2, there's an additional gray border that gets placed around the window when you turn on this feature. Keep this in mind when designing your Widget. Currently, this border is not present on Windows.

title

name of the window for display

Usage

JavaScript, XML

Description

This is the name for the Widget in the context menu.

Example

```
<window>
     <title>My Widget</title>
</window>

myWindow.title = "My New Widget";
```

focus()

brings a window to the front

Synopsis

```
Window.focus()
```

Description

If you need to bring a window forward, use the `focus()` method on a window. On Windows OS, this API might not always bring the window completely forward, particularly if the Widget is not active to begin with. But if you are interacting with the Widget and require an inspector or other secondary window to come forward, this API does just that.

Example

```
myWindow.focus();
```

Availability

Available in version 3.0 or newer.

getBestDisplay()

return the display this window intersects the most

Synopsis

```
Display window.getBestDisplay()
```

Description

This function returns the display the window intersects most. You can use this to decide what the best choice might be to move the whole window on screen, etc.

Availability

Available on version 4.5 or newer.

moveTo()

moves a window around the screen

Deprecation Notice

You should generally use the animation facilities instead of this function, as they are more robust and allow more control.

Synopsis

```
Window.moveTo(newX, newY, duration)
```

Description

The window's origin (`hOffset`, `vOffset`) is moved to the new coordinates specified by `newX` and `newY`. The `duration` option specifies the time (in tenths of a second) you want the animation to last for. The move of the object is animated (which is what makes this different from just changing `hOffset` and `vOffset`).

Example

```
myWindow.moveTo(150, 150, 2);
```

recalcShadow()

recalculates the window's shadow

Synopsis

```
Window.recalcShadow()
```

Description

If a Widget that has a shadow changes its shape (for example, by hiding or showing images) it should call the `recalcShadow()` method of its main window before returning control to the user so that the shadow is correctly displayed. If the Widget's window changes its size, there is no need to call this function as a shadow is generated automatically.

We do not continually recalculate the shadow for performance reasons.

Example

```
if (myWindow.shadow)
     myWindow.recalcShadow();
```

Platform Notes

On Windows OS, window shadows are supported in version 3.1 or newer.

Events

Events are the lifeblood of your Widget. They allow you to listen to input from the user (mouse clicks, etc.) as well as requests from the engine itself, e.g. `onLoad`. Events sent to you by the Konfabulator engine allow you to give life to your Widget and make it do what it should in these situations.

You listen to events by setting a handler property on an object, e.g. `onMouseDown`. When the event is sent, your handler is called and you can react accordingly. Your handler should take one parameter that receives the event.

Event Listeners

Event listeners can either be:

Functions that take one parameter:

```
function mouseFunc( evt )
{
   …
}
```

Or events or objects that implement a method called handleEvent, and take an event parameter:

```
var myObj = function()
{
   this.handleEvent = function( evt )
   {
      …
   }
}
```

Listeners that are set via handlers in XML get compiled into function bodies and are given an event attribute:

```
<image src="Sun.png" onMouseDown="print( event.type )"/>

// is effectively the same as…

myImage.onMouseDown = function( event )
{
   print( event.type );
}
```

Default Actions

Some events have, or can have, a default action associated with them. For example, the HTML object's `LinkClicked` event's default action is to call `openURL()` with the url presented to it. To stop the default action from occurring, you must call `event.preventDefault()`. Typically if you called this, you'd also want to call one of the functions that stop Propagation at the same time.

Older Engine Behavior

If you want to successfully write a Widget that works on versions earlier than 4.5 as well as 4.5 or later, it's important to know these restrictions.

With engines earlier than 4.5, you can only install handlers via the handler property (`onLoad`, etc.) or by the older `<action>` tags, which end up getting translated into on handlers. Also, in those earlier releases, the event is not sent to your handler -- you must look at `system.event`. There is also no event propagation that you can control.

One other major difference is that with earlier engines there really is only one event class that combines all event data. For 4.5 or later, there are several event classes. These classes only contain the data necessary for the event in question. This improves simplicity and performance, because we don't have to gather information for events that don't need it. For example, the `tellwidget` event does not need the current mouse position. Realizing that some pre-4.5 Widgets may be getting their mouse position information through events that weren't designed for that purpose, we've added new APIs to allow you to get the current mouse position directly when you need it. Before compiled functions, the way to get at event information was through `system.event`. This global variable holds the current event. In versions prior to 4.5, there was one large event that contained all types of information: mouse location, keyboard status, etc. In version 4.5 or later, the correct event class is stored in `system.event.` This means that during events such as `onLoad`, you will not get mouse location, etc.

Konfabulator Events

Below is a table listing all events sent by Konfabulator and the types of objects they normally target. The term *views* indicates any of the visual DOM Nodes - `Image`, `Text`, `TextArea`, `Frames`, `ScrollBar`, `Canvas`, and `Window`.

Type	Event Class	Received By	Handler Name
`contextmenu`	`DOMEvent`	views	onContextMenu
`dockopened`	`DOMEvent`	widget	onDockOpened
`dockclosed`	`DOMEvent`	widget	onDockClosed
`firstdisplay`	`DOMEvent`	window	onFirstDisplay
`dragdrop`	`DragDropEvent`	views	onDragDrop
`dragenter`	`DragDropEvent`	views	onDragEnter
`dragexit`	`DragDropEvent`	views	onDragExit
`fscommand`	`FlashEvent`	flash	onFSCommand
`fsreadystate`	`FlashEvent`	flash	onFSReadyState
`gainfocus`	`DOMEvent`	window, textarea	onGainFocus
`losefocus`	`DOMEvent`	window, textarea	onLoseFocus
`imageloaded`	`DOMEvent`	image	onImageLoaded
`konsposeactivated`	`DOMEvent`	widget	onKonsposeActivated
`keydown`	`KeyboardEvent`	hotkey, textarea, window	onKeyDown
`keypress`	`KeyboardEvent`	textarea, window	onKeyPress
`keyup`	`KeyboardEvent`	hotkey,	
`textarea, window`	`onKeyUp`		
`konsposedeactivated`	`DOMEvent`	widget	onKonsposeDeactivated
`load`	`DOMEvent`	widget	onLoad
`mousedown`	`MouseEvent`	views	onMouseDown
`mousedrag`	`MouseEvent`	views	onMouseDrag

Type	Event Class	Received By	Handler Name
mouseenter	MouseEvent	views	onMouseEnter
mouseexit	MouseEvent	views	onMouseExit
mousemove	MouseEvent	views	onMouseMove
mouseup	MouseEvent	views	onMouseUp
mousewheel	MouseEvent	views	onMouseWheel
multiclick	MouseEvent	views	onMultiClick
preferencescancelled	DOMEvent	widget	onPreferencesCancelled
preferenceschanged	DOMEvent	widget	onPreferencesChanged
runcommandinbgcomplete	DOMEvent	widget	onRunCommandInBgComplete
select	DOMEvent	menuitem	onSelect
screenchanged	DOMEvent	widget	onScreenChanged
tellwidget	DataEvent	widget	onTellWidget
textinput	TextEvent	window	onTextInput
unload	DOMEvent	widget	onUnload
valuechanged	DOMEvent	scrollbar	onValueChanged
visibilitychanged	DOMEvent	widget	onVisibilityChanged
wakefromsleep	DOMEvent	widget	onWakeFromSleep
webalert	WebEvent	web view	onWebAlert
webconfirm	WebEvent	web view	onWebConfirm
webcreatewindow	WebEvent	web view	onWebCreateWindow
webexception	WebEvent	web view	onWebException
weblinkclicked	WebEvent	web view	onWebLinkClicked
webpageloadcomplete	WebEvent	web view	onWebPageLoadComplete
webprompt	WebEvent	web view	onWebPrompt
webresourceloadcomplete	WebEvent	web view	onWebResourceLoadComplete
webresourcerequested	WebEvent	web view	onWebResourceRequested
webstatusbarchanged	WebEvent	web view	onWebStatusBarChanged
webtitlechanged	WebEvent	web view	onWebTitleChanged
webURLChanged	WebEvent	web view	onWebURLChanged
willchangepreferences	DOMEvent	widget	onWillChangePreferences
yahoologinchanged	DOMEvent	widget	onYahooLoginChanged

Event Classes

Each event that is sent is based on a certain class of event. For example, `mousedown` events are sent as `MouseEvent` objects. The `DOMEvent` class is the base class for all events. All of the properties and functions in that class are available to any event that is generated by Konfabulator.

DOMEvent

base class for all events

Properties

Name	Type	Read-Only	Description
`type`	`string`	yes	Returns the type of the event, e.g. `mousedown`.
`timestamp`	`date`	yes	Returns the time the event occurred.
`cancelable`	`boolean`	yes	This event can be cancelled (i.e., `preventDefault()` can be used).
`defaultPrevented`	`boolean`	yes	`preventDefault()` was called on this event.

Functions

Signature	Description
`void preventDefault()`	Prevents the default action from taking place.

DataEvent

simple event type with string data

Description

This event class is currently used for `tellwidget` and `runcommandinbgcomplete` events.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
`type`	`string`	yes	Returns the type of the event, e.g. `mousedown`.

TextEvent

text was input

Description

This event currently is only used for the `textinput` event. It contains the string of the text that was input. Currently, this is only generated from keyboard input, but could conceivably be sent from a paste action in the future.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
`data`	`string`	yes	The character or string that was input.

MouseEvent

a mouse action occurred

Description

This event is used for all mouse events. It contains the information necessary to deal with events including the location of the mouse, state of the modifier keys, etc.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
`screenX`	`int`	yes	The x location of the mouse in screen coordinates.
`screenY`	`int`	yes	The y location of the mouse in screen coordinates.
`hOffset`	`int`	yes	The x location of the mouse in window coordinates.
`vOffset`	`int`	yes	The y location of the mouse in window coordinates.
`x`	`int`	yes	The x location of the mouse in view coordinates of the target.
`y`	`int`	yes	The y location of the mouse in view coordinates of the target.
`clickCount`	`int`	yes	For `multiclick` events, the number of clicks that have occurred. Zero otherwise.
`scrollDelta`	`int`	yes	For `mousewheel` events, the signed delta you should scroll. Zero otherwise.
`ctrlKey`	`boolean`	yes	`true` if the `Ctrl` key was down during the mouse event.
`altKey`	`boolean`	yes	`true` if the `Alt` key was down during the click (option key on Mac).
`shiftKey`	`boolean`	yes	`true` if the `Shift` key was down during the mouse event.
`metaKey`	`boolean`	yes	`true` if the command key was down during the mouse event (Mac only).
`alphaLockKey`	`boolean`	yes	`true` if the `Caps Lock` key was down during the mouse event.
`modifiers`	`string`	yes	**Deprecated**: string representation of what modifiers were down during the mouse event.

KeyboardEvent

a keyboard action occurred

Description

This event is used for all keyboard events. It contains the information about what key was pressed, as well as the state of the key modifiers at the time.

Because keys might not generate actual characters, this reports merely what key was pressed, not what text, if any, was generated by it. To get the text, use the `textinput` event.

Some of the properties are deprecated and only exist because some older Widgets might still reference them.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
`keyIdentifier`	`string`	yes	String representation of what key was pressed. Results aren't always clear; use `keyCode` instead.
`keyCode`	`int`	yes	Value representing what key was pressed.
`ctrlKey`	`boolean`	yes	`true` if the `Ctrl` key was down during the keyboard event.
`altKey`	`boolean`	yes	`true` if the `Alt` key was down during the keystroke (option key on Mac).
`shiftKey`	`boolean`	yes	`true` if the `Shift` key was down during the keyboard event.
`metaKey`	`boolean`	yes	`true` if the command key was down during the keyboard event (Mac only).
`alphaLockKey`	`boolean`	yes	`true` if the `Caps Lock` key was down during the keyboard event.
`modifiers`	`string`	yes	**Deprecated**: string representation of what modifiers were down during the keyboard event.
`key`	`string`	yes	**Deprecated**: no longer used.
`keyString`	`string`	yes	**Deprecated**: string representation of what key was pressed. Similar to `keyIdentifier`, but `keyIdentifier` is part of the W3C spec.
`isChar`	`boolean`	yes	**Deprecated**: `true` if the key event generated a character.
`charCode`	`int`	yes	**Deprecated**: the character code generated by this key. No longer used.

DragDropEvent

a drag and drop related action occurred

Description

This event is used for drag and drop events.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
`screenX`	`int`	yes	The x location of the mouse in screen coordinates.
`screenY`	`int`	yes	The y location of the mouse in screen coordinates.
`hOffset`	`int`	yes	The x location of the mouse in window coordinates.
`vOffset`	`int`	yes	The y location of the mouse in window coordinates.
`x`	`int`	yes	The x location of the mouse in view coordinates of the target.
`y`	`int`	yes	The y location of the mouse in view coordinates of the target.
`ctrlKey`	`boolean`	yes	`true` if the `Ctrl` key was down during the mouse event.
`altKey`	`boolean`	yes	`true` if the `Alt` key was down during the click (option key on Mac).
`shiftKey`	`boolean`	yes	`true` if the `Shift` key was down during the mouse event.
`metaKey`	`boolean`	yes	`true` if the command key was down during the mouse event (Mac only).
`alphaLockKey`	`boolean`	yes	`true` if the `Caps Lock` key was down during the mouse event.
`dataType`	`string`	yes	String representing the type of data in the drag: `"text"`, `"URLs"`, or `"filenames"`.
`items`	`array`	yes	Array of strings representing the data. For text drags, `item[0]` contains the value.
`data`	`array`	yes	**Deprecated**: This is the backwards-compatible accessor for drag information. Item 0 contains the type and items 1 through n contain the data.

FlashEvent

events for the flash object

Description

This event class is used for `flashcommand` and `flashreadystate` events.

Inherits From

`DOMEvent`

Constants

Name	Value	Description
READYSTATE_UNITIALIZED	0	
READYSTATE_LOADING	1	Swf file is about to begin loading.
READYSTATE_LOADED	2	Connection / file open started.
READYSTATE_INTERACTIVE	3	Loading. Data from the file/url is being processed/received.
READYSTATE_COMPLETE	4	Done. All data has been loaded and completed.

Properties

Name	Type	Read-Only	Description
readyState	string	yes	The current ready state for the `flashreadystateevent`.
command	string	yes	The command for `fscommand` events.
arguments	string	yes	The arguments for `fscommand` events.

WebEvent

events for the Web object

Description

This event class is used for events. sent from the Web object.

Inherits From

`DOMEvent`

Properties

Name	Type	Read-Only	Description
data	varies	yes	Depending on the event, this could contain a URL or status bar text. See the detail on the different events for more information.
statusCode	int	yes	Indicates the status of a resource load for the `webresourceloadcomplete` event.
browserResult	varies	yes	Used to return a new Web object instance when the `webcreatewindow` event is sent.

Name	Type	Read-Only	Description
promptResult	string	yes	If the Widget handles webprompt events, the result should be stored here.
confirmResult	boolean	yes	If the Widget handles webconfirm events, the result should be stored here.

Event Reference

This section provides details of all events sent by Konfabulator.

Data Events

This section provides details of all data events sent by Konfabulator.

Events

runcommandinbgcomplete
tellwidget

runcommandinbgcomplete

sent when onRunCommandInBg is finished running

Handler Name

onRunCommandInBgComplete

Description

If you have called runCommandInBg(), this event is sent when the task completes.

Sent To

Widget object.

Example

```
widget.onRunCommandInBgComplete = function( event ) {print( 'result: ' +
event.data )};
```

Availability

Available in version 4.0 or newer.

tellwidget

sent in response to a tellWidget() call from another Widget

Handler Name

onTellWidget

Description

The tellwidget event is sent by this event will be called when another Widget uses the tellWidget() function to communicate with your own. For more information, see tellWidget().

Sent To

Widget object.

Example

```
var handleExternalCall = function(event)
{
  if(event.data == "reload")
  {
    reloadWidget();
  }
};
widget.onTellWidget = handleExternalCall;
```

Availability

Available in version 2.0 or newer.

DragDrop Events

This section provides details of all drag-drop events sent by Konfabulator.

Events

dragdrop
dragenter
dragexit

dragdrop

sent when something is dropped on the object

Handler Name

onDragDrop

Description

The dragdrop event is sent when a file, URL, or string is dragged from another application and dropped on the object.

In a handler for this event, objects can access the dataType via event.dataType. The items are passed in an array stored in event.items.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image>
<onDragDrop>
     if (event.dataType == "filenames")
     {
       processDroppedFiles(event.items);
     }
</onDragDrop>
</image>

myImage.onDragDrop = function( event ) {handleDragDrop(); } ;
```

Availability

Available in version 1.0 or newer.

dragenter

sent when an item is dragged into the object

Handler Name

`onDragEnter`

Description

The `dragenter` event is sent when the user has dragged an item from another application into the object. This happens before the item is dropped (indeed it may not be dropped as the user can change their mind).

This is useful for triggering a visual change of the object to indicate to the user that the dragged object will be accepted or rejected if it is dropped. The type of data being dragged is contained in `event.dataType`.

Sent To

`Canvas`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Window` objects.

Example

```
<image name="well"
    onDragEnter= function( event ) {highlightDropTarget(well); } />
well.onDragEnter = function( event ) {highlightDropTarget(well); } ;
```

Availability

Available in version 1.0 or newer.

dragexit

sent when an item is dragged out of the object

Handler Name

`onDragExit`

Description

The `dragexit` event is sent when the user has dragged an item from another application into the object and then out again.

This is useful for undoing things that were done in response to `dragenter`, such as removing a drag highlight effect.

Sent To

`Canvas`, `Frame`, `Image`, `Text`, `TextArea`, `ScrollBar`, and `Window` objects.

Example

```
<image name="well"
     onDragExit="unhighlightDropTarget(well)"/>

well.onDragExit = function( event ) {unhighlightDropTarget(event); } ;
```

Availability

Available in version 1.0 or newer.

Flash Events

This section provides details of all Flash events sent by Konfabulator.

Events

fscommand

sent when an FsCommand action occurs

Handler Name

onFsCommand

Description

The onFsCommand event is sent when an FsCommand action occurs in a movie with a URL starting with FsCommand. This is useful for creating a response to a frame or button action in a Flash movie. The argument type is `string`.

When the event is sent, the command and argument can be retrieved from the event via the `command` and `arguments` properties.

Sent To

`Flash` objects.

Example

```
<flash name="myFlash" src="file">
<onFsCommand>
      if ( event.command == "myCommand" )
      {
      //do what I say.
      myJsFunction ( event.arguments );
      }
</onFsCommand>
</flash>
```

Note

Special security settings are dependent on the value of `src`. If `src` is given a remote URL, `fsCommand` is disabled.

Availability

Available in version 4.5 or newer.

fsreadystate

sent when an FSReadyState event occurs

Handler Name

onFsReadyState

Description

The onFsReadyState event is sent during the src file load, returning one of the following values: 0=Loading, 1=Uninitialized, 2=Loaded, 3=Interactive, 4=Complete. The resulting value can be retrieved from event.readyState. Note that some flash files fire the Complete event continuously. Code accordingly.

Sent To

`Flash` objects.

Example

```
<flash name="myFlash" src="file">
<onFsReadyState>
      if ( event.readyState == 4 )
      {
      //do what I say.
      }
</onFsReadyState>
</flash>
```

Availability

Available in version 4.5 or newer.

Keyboard Events

This section provides details of all Keyboard events sent by Konfabulator.

Events

keydown
keypress
keyup

keydown

sent when a key is pressed

Handler Name

onKeyDown

Description

The keydown event is sent when a key is pressed that is specified with the onKeyDown attribute. A keydown event is a raw keyboard event. It contains the keyCode of the key that was pressed. To get the text that was generated, listen for `textinput` events instead.

Sent To

`TextArea` and `Window` objects.

Example

```
<textarea data="Type Stuff Here" name="typomatic"
     onKeyDown="print ( event.keyCode )"/>
typomatic.onKeyDown = "keypressed = true";
```

keypress

sent after a keydown event

Handler Name

onKeyPress

Description

The keypress event is sent from TextArea objects. It allows you to potentially stop a key from reaching the textarea by calling rejectKeyPress(). The most typical use of this type of behavior is to intercept the enter key in a text area to mean 'execute' or 'search' rather than it just adding a return to the text area.

This is an older event, and in version 4.5 or later, is largely unneeded, as you can instead call preventDefault() on a keydown or textinput event.

This is useful for performing validation of text entry. Normally any key pressed is processed by the system and the appropriate change is made to the text area (adding a character, deleting a word, etc.). You can override this behavior by calling the textarea method rejectKeyPress(), which causes the key press to be ignored. The value of the key pressed is always available in system.event.key.

Sent To

TextArea objects.

Example

```
<textarea name="ta1" onKeyPress="handleKey(event)">
  <script>
     function handleKey(event)
     {
        // Convert input to uppercase
        var key = event.key;
        if (key.charCodeAt(0) &gt;= "A".charCodeAt(0) &&
            key.charCodeAt(0) &lt;= "z".charCodeAt(0))
        {
            // Tell the text area to ignore this keyPress
            ta1.rejectKeyPress();

            // Append an upper case copy of the key pressed
            ta1.replaceSelection(key.toUpperCase());
        }
  </script>
</textarea>
ta1.onKeyPress = handleKey;
```

keyup

sent when a key is released

Handler Name

onKeyUp

Description

The keyup event is sent when a key is pressed that is specified with the onKeyUp attribute.

Sent To

TextArea and Window objects.

Example

```
<textarea data="Type Stuff Here" name="typomatic"
     onKeyUp="print event.keyCode "/>

typomatic.onKeyUp = "keypressed = true";
```

Miscellaneous Events

This section provides details of all miscellaneous events sent by Konfabulator.

Events

contextmenu

sent when a context menu is about to appear

Handler Name

onContextMenu

Description

The simplest way to specify context menu items that get added to the standard context menu for a Widget is to use the contextMenuItems tag in the XML. However, for those Widgets that need to build their items dynamically, the onContextMenu handler is the hook to do so. When the menu is about to be presented, this is called for all elements under the mouse from front to back in the view order until some view responds. The first element to capture the event will halt the propagation of the event, except for the window object, which will always receive this event.

To build the context menu items, create an array of MenuItem objects and set the contextMenuItems attribute to the array. See MenuItem for a description of the items.

Sent To

Canvas, Frame, Image, Text, TextArea, and Window objects.

Example

```
<onContextMenu>
var items = new Array();
items[0] = new MenuItem();
items[0].title = "This is the title";
items[0].enabled = false;
items[0].checked = true;
items[0].onSelect = function( event ) {alert( 'you chose it!' ); } ;

items[1] = new MenuItem();
items[1].title = "This is the second title";
items[1].onSelect = function( event ) {beep(); } ;

myImage.contextMenuItems = items;
</onContextMenu>
```

Availability

Available in version 2.0 or newer.

dockclosed

sent when the dock was just closed by the user

Handler Name

`onDockClosed`

Description

This event is sent whenever the dock is closed by the user. If your Widget uses the setDockItem API, you can use this as an indication that you should no longer call that function, as the dock is no longer around. If you are using a timer to periodically set your dock item, you should stop the timer to avoid using unnecessary CPU time.

Sent To

`Widget` object.

Example

```
widget.onDockClosed= function( event ) {stopDockItemTimer()};
```

Availability

Available in version 4.0 or newer.

dockopened

sent when the dock was just opened by the user

Handler Name

`onDockOpened`

Description

This event is sent whenever the dock is opened by the user. If your Widget uses the setDockItem API, you can use this as an indication that is now OK to start calling that API. Otherwise, if the dock is not open, setDockItem will waste CPU time.

Sent To

`Widget` object.

Example

```
widget.onDockOpened= function( event ) {setMyDockItem()};
```

Availability

Available in version 4.0 or newer.

firstdisplay

sent the very first time a window is displayed

Handler Name

`onFirstDisplay`

Description

The very first time a window is ever shown in a Widget (i.e., we see that it has no saved preferences for position), the `firstdisplay` event is sent to the window. This allows a Widget to decide where it should appear the very first time the Widget is launched.

Sent To

`Window` object.

Example

```
<onFirstDisplay>
    setInitialPosition();
</onFirstDisplay>
```

Remember, this is only sent the first time the window appears. After the prefs are saved for that window, you won't receive this message again.

Availability

Available in version 2.0 or newer.

gainfocus

sent when an element has acquired user focus

Handler Name

`onGainFocus`

Description

This event allows you to detect when a window or text area has gained focus. For a window, this means it's become the active window. For a text area, it means the text area is now the current keyboard focus.

Sent To

`Window, TextArea`

Example

```
widget.onGainFocus= function( event ) {print( 'Focused' )};
```

idle

sent in response to the onIdle trigger

Handler Name

`onIdle`

Description

This attribute allows you to change what gets executed for a Widget by the onIdle trigger. The onIdle handler is called every 0.2 seconds.

onIdle is considered deprecated. You should instead use Timer objects, as they are far more flexible.

Sent To

`Widget` object.

Example

```
widget.onIdle = function( event ) {print( 'onIdle' )};
```

Availability

Available in version 4.0 or newer.

konsposeactivated

sent when heads-up display is entered

Handler Name

`onKonsposeActivated`

Description

This event is sent when "heads-up display" mode is entered. Konsposé was the prior name for the heads-up display feature of the product.

Sent To

`Widget` object.

Example

```
widget.onKonsposeActivated = function( event ) {print( 'heads up display entered'
)};
```

Availability

Available in version 4.0 or newer.

konsposedeactivated

sent when heads-up display is exited

Handler Name

`onKonsposeDeactivated`

Description

This event is sent when "heads-up display" mode is exited. Konsposé was the prior name for the "heads-up display" feature of the product.

Sent To

`Widget` object.

Example

```
widget.onKonsposeDeactivated = function( event ) {print( 'heads up display exited'
)};
```

Availability

Available in version 4.0 or newer.

load

sent when a widget is first loaded

Handler Name

`onLoad`

Description

This event is sent when your widget is first loaded. Since `onLoad` happens early in your Widget's lifetime, if you wish to set this programmatically, you will need to set this attribute in a <script> block. Otherwise, it's usually easiest to set your onLoad handler in an XML <action> block.

Sent To

`Widget` object.

Example

```
<script>
   widget.onLoad = doOnLoad;
</script>
```

Availability

Available in version 4.0 or newer.

losefocus

sent when an element has lost user focus

Handler Name

`onLoseFocus`

Description

This event allows you to detect when a window or text area has lost focus. For a window, this means it's become inactive. For a text area, it means the text area is no longer the current keyboard focus. You can use this to clear any focus adornment you might draw around the text area to indicate focus. Only editable text areas get the keyboard focus, and as such this action is only called for an editable text area.

Sent To

`Window, TextArea`

Example

```
<textarea data="Type Stuff Here" name="typomatic"
     onLoseFocus="print('I lost focus!');"/>
```

```
typomatic.onLoseFocus = function( event ) { print( 'focus lost' ); }
```

preferencescancelled

sent when the preferences dialog is cancelled

Handler Name

`onPreferencesCancelled`

Description

This event is sent when the user presses cancel in the preferences dialog. It is used to balance the `willchangepreferences` event, and allows you to undo an action performed during `willchangepreferences`.

Sent To

`Widget` object.

Example

```
widget.onPreferencesCancelled = function( event ) { print( 'prefs cancelled' ) };
```

Availability

Available in version 4.5 or newer.

preferenceschanged

sent when the preferences dialog is confirmed

Handler Name

`onPreferencesChanged`

Description

This event is sent after the Preferences dialog is closed when the user presses OK. If the user cancels out of the Preferences dialog, the `preferencescancelled` event is sent instead.

Sent To

`Widget` object.

Example

```
widget.onPreferencesChanged = function( event ) { print( 'prefs changed' ) };
```

Availability

Available in version 4.0 or newer.

screenchanged

sent when screen geometry changes

Handler Name

`onScreenChanged`

Description

This event is sent whenever the main screen changes its configuration (resolution, size, etc.).

Sent To

`Widget` object.

Example

```
widget.onScreenChanged = function( event ) {print( 'screen changed' )};
```

Availability

Available in version 4.0 or newer.

timer

sent in response to the onTimer trigger

Handler Name

`onTimer`

Description

This event is sent when the global timer mechanism fires.

There can only be one `onTimer` trigger per Widget. For multiple timers running at different frequencies, use the Timer object. This attribute is considered deprecated and is only here for completeness.

Sent To

`Widget` object.

Example

```
widget.onTimer = function( event ) {print( 'timer fired' )};
```

Availability

Available in version 4.0 or newer.

timerfired

sent when a timer fires

Handler Name

`onTimerFired`

Description

This event is sent when a timer fires.

Sent To

`Timer` objects.

Example

```
<timer name="myTimer"
     onTimerFired="alert( 'hello!' );"/>

myTimer.onTimerFired = "alert( 'fired!' );";
```

Availability

Available in version 2.0 or newer.

unload

sent when the widget is being shut down

Handler Name

`onUnload`

Description

This event is sent right before your Widget is terminated.

You should not perform any lengthy operations in this trigger as Widgets are encouraged to shut down quickly (an example of a lengthy operation would be retrieving something from the network).

Sent To

`Widget` object.

Example

```
widget.onUnload = disconnectMyCOMObjects;
```

Availability

Available in version 4.0 or newer.

valuechanged

sent when a scrollbar's value changes

Handler Name

`onValueChanged`

Description

This event is sent whenever a scrollbar's value changes.

If you attach a scrollbar to a frame, you rarely need to specify anything for this attribute. The frame automatically reacts to scrollbar changes.

Sent To

`Scrollbar`

Example

```
<scrollbar name="sb">
      <onValueChanged>
         print( "Whoa! value is now " + this.value );
      </onValueChanged>
</scrollbar>
```

Availability

Available in version 3.0 or newer.

wakefromsleep

sent when the computer wakes from sleep mode

Handler Name

`onWakeFromSleep`

Description

This event is sent when the computer wakes from sleep mode.

In version 3.0 or newer, timers are stopped when the computer goes to sleep and are not restarted until onWakeFromSleep is called. There is usually a delay of about 15 seconds between when the computer wakes and when this event is sent. This is to give the networking stack time to recover before the Widget timers start to function again.

Sent To

`Widget` object.

Example

```
widget.onWakeFromSleep = handleWake;
```

Availability

Available in version 1.5 or newer.

Version Notes

In version 3.0 or newer, timers are stopped when the computer goes to sleep and are not restarted until `onWakeFromSleep` is called.

willchangepreferences

sent right before the preferences dialog is opened

Handler Name

`onWillChangePreferences`

Description

This event is sent right before the Preferences dialog is opened.

Sent To

`Widget` object.

Example

```
widget.onWillChangePreferences = handleWillChangePrefs;
```

Availability

Available in version 4.0 or newer.

yahoologinchanged

sent when the Yahoo! login status changes

Handler Name

`onYahooLoginChanged`

Description

This event is sent when the login state changes (i.e., either the user has logged in or out of their Yahoo! account).You generally only need to listen to this event if you use the Yahoo! login APIs.

Sent To

`Widget` object.

Example

```
widget.onYahooLoginChanged = handleLoginChanged;
```

Availability

Available in version 4.0 or newer.

Mouse Events

This section provides details of all Mouse events sent by Konfabulator.

Events

- `click`
- `mousedown`
- `mousedrag`
- `mouseenter`
- `mouseexit`
- `mousemove`
- `mouseup`
- `mousewheel`
- `multiclick`

click

sent when a single click occurs

Handler Name

`onClick`

Description

This event is sent when the user clicks in an object. This is the best way to track a true click as it also guarantees the mouse was released in the object this handler is attached to.

Applying an `onClick` handler of any kind will cause the mouse cursor to change to a hand icon. Currently, this cannot be disabled.

Sent To

`Canvas`, `Frame`, `Image`, `Text`, `TextArea`, and `ScrollBar` objects.

Example

```
<image src="..." onClick="print( 'clicked' )"/>

myImage.onClick = function( event ) { handleClick(); } ;
```

Availability

Available in version 4.0 or newer.

mousedown

sent when the mouse button is pressed inside the object

Handler Name

onMouseDown

Description

The mousedown event is sent when the user presses the mouse button down within the object.

This is useful for creating customized buttons that respond visually to a user's click.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image onMouseDown="beep();"/>

myImage.onMouseDown = function( event ) {beep(); } ;
```

Availability

Available in version 1.0 or newer.

mousedrag

sent when the mouse moves inside an object and the mouse button is down

Handler Name

onMouseDrag

Description

The mousedrag event is sent when the user moves the mouse over an object with the mouse button held down.

In version 4.0 and newer, this is the replacement for the older onMouseMove attribute, and onMouseMove is a true mouse move event, i.e. the mouse moves with no buttons pressed. See mousemove for more information.

The mouse location is available in the global system.event object.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image onMouseDrag="moveSliderThumb();"/>

myImage.onMouseDrag = function( event ) {moveSliderThumb(); } ;
```

Availability

Available in version 4.0 or newer.

mouseenter

sent when the mouse enters an object

Handler Name

onMouseEnter

Description

The mouseenter event is sent when the user has moved the cursor within the object.

This is useful for triggering a visual change of the object based on a rollover state.

This is equivalent to the onmouseover event in a browser.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image onMouseEnter="print( "Mouse entered!" );"/>

myImage.onMouseEnter = function( event ) {handleEntered(); } ;
```

Availability

Available in version 1.0 or newer.

mouseexit

sent when the mouse exits an object

Handler Name

onMouseExit

Description

The mouseexit event is sent when the user has moved the cursor from within the object to outside the object.

This is useful for triggering a visual change of the object based on a rollover state.

This is equivalent to the onmouseout event in a browser.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image onMouseExit="handleMouseExit();"/>

myImage.onMouseExit = function( event ) {handleMouseExit(); } ;
```

Availability

Available in version 1.0 or newer.

mousemove

sent when the mouse moves within an object

Handler Name

onMouseMove

Description

The mousemove event is sent when the user drags the mouse cursor within the bounds of an object.

This attribute is called in two different situations depending on the value of your Widget's minimum platform version. With minimum platform versions under 4.0 or no minimum version set, this handler is called when the mouse moves over the object and the mouse button is down. This is called a *drag event*.

With minimum platform version of 4.0 and newer, this handler is called when the mouse moves over your object and the mouse button is up. This is called a *simple move event*.

This change in behavior was instituted in version 4.0 to allow us to have a true mouse move event and to fix the terminology. Handling mouse drags can be handled with the onMouseDrag handler, introduced in that version of the engine.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image name="myImage" onMouseMove=
"print(system.event.x + ", " + system.event.y );"/>

myImage.onMouseMove = function( event ) handleMouseMove(); } ;
```

Availability

Available in version 1.0 or newer.

mouseup

sent when the mouse button is released in an object

Handler Name

onMouseUp

Description

The mouseup event is sent when the user has released the mouse after having it down within the object.

This is useful for triggering a visual change of the object based on a pressed state.

Note: onMouseUp triggers even if the mouse is not inside the object when the mouse is released. To create buttons that have a standard button tracking behavior, you must use all four of onMouseDown, onMouseEnter, onMouseExit and onMouseUp to track the state of the mouse relative to your button. You can see an example of this behavior in the source of the standard Calendar Widget.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<image name="myImage" onMouseUp="handleOnMouseUp();"/>

myFrame.onMouseUp = function( event ) {handleOnMouseUp(); } ;
```

Availability

Available in version 1.0 or newer.

mousewheel

sent when the mouse wheel is moved while over an object

Handler Name

onMouseWheel

Description

The mousewheel event is sent when the user moves the mouse wheel while hovering over the object. The delta can be gotten from system.event.scrollDelta.

You normally don't need to use this hook, as when a scrollbar is attached to a frame, the mouse wheel is handled automatically.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<frame name="myFrame" onMouseWheel=
    "handleOnMouseWheel(event.scrollDelta);"/>

myFrame.onMouseWheel = myScrollWheelHandler ;
```

Availability

Available in version 3.0 or newer.

multiclick

sent when a multiple click occurs

Handler Name

onMultiClick

Description

You can easily trap double-clicks (triple-clicks, etc.) by responding to the multiclick event. When the event is sent, you can inspect event.clickCount to see what the value is. It will always be 2 (for a double-click) or greater.

It is also possible to inspect the event.clickCount in an onMouseUp handler as well in lieu of listening to multiclick events. However, the advantage to using multiclick is that it does not interfere with window dragging the way that mouseup does, i.e., a mouse up handler on an image prevents a window from being dragged if you click that image. If your image only needs to respond to multiclicks, you can respond to multiclick and the Widget can still be dragged as usual.

Sent To

Canvas, Frame, Image, Text, TextArea, ScrollBar, and Window objects.

Example

```
<onMultiClick>
   if ( system.event.clickCount == 2 )
      alert( "Double Click!" );
</onMultiClick>

myImage.onMultiClick= function( event ) {handleMultiClick(); } ;
```

Availability

Available in version 3.0 or newer.

Text Events

This section provides details of all Text events sent by Konfabulator.

Events

textinput

textinput

sent when a keydown or series of keydowns generates textcode activated when a key is pressed

Handler Name

onTextInput

Description

This event is sent when the user types something on the keyboard that generates text. Not all keydown events generate a character, so if you really care about exactly what was typed, you can intercept this event.

Sent To

TextArea and Window objects.

Example

```
widget.onTextInput = function( event ) { print( event.data + " was input!" ) };
```

Availability

Available in version 4.5 or newer.

Web Events

This section provides details of all Web events sent by Konfabulator.

Events

webalert
webconfirm
webcreatewindow
webexception
weblinkclicked
webpageloadcomplete
webprompt

webresourceloadcomplete
webresourcerequested
webstatusbarchanged
webtitlechanged
weburlchanged

webalert

sent when an alert has been raised by a web page

Handler Name

onWebAlert

Description

When a web page requests an alert, this event is sent. The alert message in the alert can be accessed via the data property of the event object.

If the default is not handled, a simple alert dialog is created automatically.

Sent To

Web objects.

Example

```
web.onConfirm = function() { alert( event.data ); }
```

Availability

Available in version 4.5 or newer.

webconfirm

sent when a webpage request confirmation is completed

Handler Name

onWebConfirm

Description

When a web page requests a prompt, this event is sent. The prompt text can be accessed via event.data. The result is placed in the confirmResult property of the event object.

If the default is not handled, a simple confirm dialog is created automatically.

Sent To

Web objects.

Example

```
web.onWebConfirm = function()
  { event.confirmResult = myConfirmResult( event.data ); }
```

Availability

Available in version 4.5 or newer.

webcreatewindow

sent when a new window is requested

Handler Name

`onWebCreateWindow`

Description

When a web page wants to create a new window this event is sent. The function must set a new `Web` instance in the `browserResult` property of the event object. Consequently the `url` is set on that object and will run on its own. You can wrap that `Web` object in another window and in whatever chrome fits your application, and the instance is used there.

The default is to disallow creation of new windows.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webexception

sent when a web page causes an exception to be thrown

Handler Name

`onWebException`

Description

This event is sent when a Javascript exception occurs in a web page. The exception thrown is accessible via the exception property of the event object.

Sent To

`Web` objects.

Example

```
web.onWebException = function() { log("Caught exception: " + event.exception ); }
```

Availability

Available in version 4.5 or newer.

weblinkclicked

sent when a non-JavaScript link is clicked

Handler Name

`onWebLinkClicked`

Description

If an anchor has an `href` that changes the base document, then this event is sent to handle it. The `url` is accessible via the data property of the event.

If your intention is for the web viewer to change base documents, then the proper response is:

```
myWebInstance.url = event.data;
```

If you override this, you should call `event_preventDefault()`. Otherwise, the default action will execute, which is equivalent to:

```
openURL( event.data );
```

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webpageloadcomplete

sent when a base document and its media have been fully loaded

Handler Name

`onWebPageLoadComplete`

Description

This is meant to only advise of the document fully loading. This can be interesting for giving a completely loaded indicator.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webprompt

sent when a web page elicits input from a user

Handler Name

`onWebPrompt`

Description

When a web page requests a prompt, this event is sent. The prompt string can be accessed via `event.data`.

The default value is placed in `event.promptResult`. You may modify this with what the user inputs, otherwise the default value is used as the input value. Finally, you must set `event.confirmResult` to true if the dialog was confirmed; otherwise false.

Sent To

`Web` objects.

Example

```
web.onWebPrompt = function(event)
  { event.confirm.Result = myConfirmFunction( event.data ); }
```


Availability

Available in version 4.5 or newer.

webresourceloadcomplete

sent when a resource has completed loading

Handler Name

`onWebResourceLoadComplete`

Description

Like webResourceRequested, the URL in question can be accessed via event.data. Additionally, you can find out the success of the load via the statusCode property of the event object.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webresourcerequested

sent when a resource is about to be requested

Handler Name

`onWebResourceRequested`

Description

The URL in question can be accessed via the data property of the event object. This is meant to only advise that media has been requested. This can be interesting for displaying a list of all media on the current document.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webstatusbarchanged

sent when the status bar text is changed

Handler Name

`onWebStatusBarChanged`

Description

You can access the current status bar text via the statusBar attribute, or the data property of the event object.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

webtitlechanged

sent when the title of a document changes

Handler Name

`onWebTitleChanged`

Description

You can access the current title via the title attribute, or the data property of the event object.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

weburlchanged

sent whenever the base url of a document changes

Handler Name

`onWebURLChanged`

Description

This can happen either by explicitly setting the url attribute, or via a redirect. This can be used to allow for a history stack.

You can access the current url via the url attribute, or the data property of the event object.

Sent To

`Web` objects.

Availability

Available in version 4.5 or newer.

Event Functions

This section describes the global functions that are related to events.

getGlobalMousePosition()

returns the current mouse position in global coordinates

Synopsis

```
Point getGlobalMousePosition();
```


Description

This function is used to return the current mouse position in global (screen) coordinates. You can use this to find out where the mouse is when you might otherwise not be able to tell, such as during a keyboard event.

If you wish to know the current mouse position relative to a specific window, use `window.getCurrentMousePosition()`

Example

```
var myPt = getGlobalMousePosition();
print( 'mouse is at ', myPt.x, myPt.y );
```

Availability

Available in version 4.5 or newer.

System DOM Reference

These attributes and functions give JavaScript code access to various system settings and hardware information.

COM

functions to call COM interfaces on Windows

Functions

COM.connectObject
COM.createObject
COM.disconnectObject

Description

The COM object is an interface to enable your Widgets to call a COM interface in the system. For example, you could use it to talk to iTunes (if you didn't use our built-in support), MSN Messenger, Outlook, etc.

You can connect to any COM object using `COM.createObject( progID|CLSID )`. The COM interface does need to have sufficient type info supplied. At present we cannot support lazy binding to methods, etc., so all type information needs to be provided up front (i.e., the `ITypeInfo` interface needs to return the appropriate info so we can perform our introspection to obtain info about things such as the number of parameters and their types).

Example

This example prints some info from iTunes:

```
var it = COM.createObject( "iTunes.Application" );

track = it.CurrentTrack;

print( track.Album );
print( track.Artist );
```

Here's another example that prints some info from MSN Messenger:

```
messenger = COM.createObject( "Messenger.UIAutomation" );

contacts = messenger.MyContacts;
num = contacts.Count;
for ( i = 0; i < num; i++ ) {
   contact = contacts.Item( i );
   print( " " + contact.FriendlyName + " " + contact.Status );
}
```

This code works only when logged in to MSN Messenger. Things like the status of a contact can be found out using the Web, MSDN, etc.

It is also possible to hook up an "event sink." This allows you to have an application inform you when things change (e.g., buddy status) as opposed to having to poll for that information. `COM.connectObject` tells the Widget Engine that you want to be informed of object events. `COM.disconnectObject` tells us that you no longer wish to be informed of those events. You should always disconnect a sink when you are done with it.

And finally, a note on object references. Be sure to clear your references out to null whenever you know you are through with an interface. This is because COM requires a certain amount of reference counting to work, and JavaScript's garbage collection mentality can confuse it a bit. For those reasons, it's good to always clear your references to interfaces (COM objects) you've gotten when you are done with them. It's not strictly required, but it prevents potential confusion later.

Availability

COM support is available in version 2.0 and later.

COM.connectObject
COM.createObject
COM.disconnectObject

COM.connectObject

connects an event sink to listen to an object's events

Synopsis

```
COM.connectObject( object, prefix )
```

Description

This function allows you to connect to an object created or otherwise received from using the COM interface to listen to events. Many objects in the COM world have an event interface that you can listen to for events. For example, you can connect to the main iTunes application object and it will tell you when the player starts and stops, as well as when the application is about to quit.

The object you pass in must have been either created through COM.createObject or gotten through a call to a COM object that you created (e.g., if you receive a track from iTunes, you can use that iTunes track, provided it has an event interface).

For the second parameter, you pass a prefix for a function that will be called when an event occurs. For example, the iTunes COM interface sends out `OnPlayerPlayEvent` when the player starts a track, passing it the track that it started to play. The prefix helps us find the function to call. When the event occurs, it tries to call a function called `<prefix>OnPlayerPlayEvent()`. The following example shows us listening to that event.

Example

```
var iTunesObj = COM.createObject( "iTunes.Application" );

COM.connectObject( iTunesObj, "iTunes_" );

function iTunes_OnPlayerPlayEvent( track )
{
   print( "Started to play " + track.Name );
}
```

When you are done with an object and listening to its events, you should take care to call `disconnectObject`.

There can be only one established event sink per object at a time.

Notes

This function throws an exception if it cannot successfully connect. It is recommended to call this inside a `try/catch` handler.

Our function is called `iTunes_OnPlayerPlayEvent`, which is the combination of the prefix we specified, and the name of the COM method which gets invoked. Also note that we were passed a parameter that is another COM object representing the track. From there we are able to reference its `Name` attribute.

Availability

Available in version 2.0 or newer.

COM.createObject

creates a COM object using progID or CLSID

Synopsis

```
COM.createObject( progID | CLSID )
```

Description

This is the main place to start your romp through the COM forest. The trick here is to find out what interfaces you can call. Unfortunately, there's no simple way to figure this out short of looking in a COM browser and using trial and error. Some information is available on the Web (try a quick Yahoo! search for things like "automation," "COM," and your favorite application). You can also use regedit to look for progID. But really, the COM browser provided by Visual Studio is probably the best way (and cheapest if you already have Visual Studio).

Example

```
var iTunesObj = COM.createObject( "iTunes.Application" );
```

Availability

Available in version 2.0 or newer.

COM.disconnectObject

disconnects an event sink previously established with connectObject

Synopsis

```
COM.disconnectObject( object )
```

Description

After you are done with an event sink created with a call to `connectObject`, you should call this function to break the connection. In some cases, it's important that you do this before releasing the main COM object due to the way some applications are written.

Example

```
COM.disconnectObject( iTunesObj );
```

Availability

Available in version 2.0 or newer.

Filesystem

gets information from and interacts with the filesystem

Synopsis

```
file system
```

Description

The `file system` object provides access to the underlying files and directories of the system on which the Widget is running. See below for details of the individual functions and attributes.

Attributes

`filesystem.volumes`

Functions

`filesystem.copy()`
`filesystem.createDirectory()`
`filesystem.emptyRecycleBin()` `filesystem.emptyTrash()`
`filesystem.getDirectoryContents()`
`filesystem.getDisplayName()`
`filesystem.getFileInfo()`
`filesystem.getMD5()`
`filesystem.getRecycleBinInfo()` `filesystem.getTrashInfo()`
`filesystem.isPathAllowed()`
`filesystem.itemExists()`
`filesystem.move()`
`filesystem.moveToRecycleBin()` `filesystem.moveToTrash()`
`filesystem.open()`
`filesystem.openRecycleBin()` `filesystem.openTrash()`
`filesystem.readFile()`
`filesystem.reveal()`
`filesystem.remove()`
`filesystem.unzip()`
`filesystem.writeFile()`
`filesystem.zip()`

Platform Notes

Note that in version 3.1 and later, the version for Windows OS does not allow the `filesystem` object to change anything inside C:\Windows (or whatever your Windows directory is set to). The Mac version gets this behavior for free due to file permissions.

Availability

The `filesystem` object is available in version 1.8 or newer.

filesystem.volumes

array of currently mounted volumes

Description

The `volumes` attribute of the `filesystem` object contains an array of all the currently mounted volumes. Each entry in the array is a small object with the following attributes:

```
path
freeBytes
totalBytes
```

You can use these in concert with functions like `bytesToUIString` or `filesystem.getDisplayName`.

Example

```
vols = filesystem.volumes;
for ( a in vols )
{
```


```
    print( "Volume " +
       filesystem.getDisplayName( vols[a].path ) +
       " (path " + vols[a].path + ") has a capacity of " +
       bytesToUIString( vols[a].totalBytes ) + " and " +
       bytesToUIString( vols[a].freeBytes ) + " free." );
}
```

Availability

Available in version 2.0 or newer.

filesystem.copy()

copies files or directories to a destination

Synopsis

```
filesystem.copy( path, destination );
```

Description

This function allows you to copy one or more files or directories to a specified destination directory. The source can be a single path, or an array of paths.

If copying one file, the destination need not exist. In this case, it is assumed the destination is a new file name for the file. If the destination does exist, it is assumed it specifies a directory in which to copy the new file.

If copying multiple files, the destination must be an existing directory.

This function returns true if successful, false otherwise.

Example

```
// to copy a file to a folder
filesystem.copy( "myfile.txt", "/Users/evoas" );

// to duplicate a file
filesystem.copy( "myfile.txt", "myfile_copy.txt" );
```

Availability

Available in version 2.0 or newer.

filesystem.createDirectory()

creates a new directory

Synopsis

```
filesystem.createDirectory( path );
```

Description

This function creates a new directory. It does not create intermediate directories (i.e., it does not do the equivalent of `mkdir -p`).

Currently, directories can only be created inside your Widget's `WidgetData` folder.

Example

```
// determine path
path = system.widgetDataFolder + "/test";
filesystem.createDirectory( path );
```

Availability

Available in version 4.0 or newer.

filesystem.emptyRecycleBin()
filesystem.emptyTrash()

empties the system trash

Synopsis

```
filesystem.emptyRecycleBin()
filesystem.emptyTrash()
```

Description

This function empties the trash/recycle bin. If the user has their settings set to see the warning dialog, it comes up automatically.

The function has two names to reflect the different terms used on the two platforms, Windows OS and Mac OS X.

Example

```
fileystem.emptyTrash();
```

Availability

Available in version 2.0 or newer.

filesystem.getDirectoryContents()

gets the names of the files in a directory

Synopsis

```
array = filesystem.getDirectoryContents(directory, recurse)
```

Description

Retrieves the names of the files in the specified directory optionally recursing (descending) into each subdirectory.

Example

```
fileList = filesystem.getDirectoryContents(path, false);
```

Platform Notes

As of version 2.0, this function behaves the same on both Mac and Windows OS. It now always returns an array of names that are rooted at the directory you pass and never a full path. Previously, Windows would return full paths if you passed in a full path.

Availability

Available in version 1.8 or newer.

filesystem.getDisplayName()

returns the user-friendly name of a file

Synopsis

```
filesystem.getDisplayName( path )
```


Description

This function returns the display name for a file path. The display name is what you'd see in the Finder or Explorer. For example, if a file's extension is supposed to be hidden, this function removes it. You are guaranteed to print the same name for a file path that you see in the OS.

Example

```
print( filesystem.getDisplayName( "C:\" ) );
```

Availability

Available in version 2.0 or newer.

filesystem.getFileInfo()

returns information about a file or directory

Synopsis

```
filesytem.getFileInfo( path )
```

Description

This function returns a small object that describes the file or directory passed to it. The object has the following attributes:

```
size
isDirectory
isHidden
lastModified
```

When there is an `isDirectory` function, this information comes with that tidbit as well to save the number of filesystem operations necessary to traverse a tree of files.

Example

```
info = filesystem.getFileInfo( "myfile.txt" );

print( "myfile is " + bytesToUIString( info.size ) + " in size" );
```

Availability

Available in version 2.0 or newer.

filesystem.getMD5()

computes the md5 digest for a file

Synopsis

```
string filesystem.getMD5( string )
```

Description

This function computes the md5 digest (or 'hash') for a file. An md5 digest, while not 100% guaranteed to be unique, is typically used to determine if files are identical. The engine actually uses md5 hashes to determine when a Widget changes.

Example

```
var hash = filesystem.getMD5( "myfile.txt" );

if ( hash != lastHash )
```

```
print( "File has changed!" );
```

Availability

Available in Version 4.0 or newer.

filesystem.getRecycleBinInfo()
filesystem.getTrashInfo()

gets information about files that have been deleted but not yet purged

Synopsis

```
filesystem.getRecycleBinInfo()
filesystem.getTrashInfo()
```

Description

Retrieves the number and total size of files that are in the user's trash or recycle bin. An object with two members is returned, `numItems` and `size`.

The function has two names to reflect the different terms used on the two platforms, Windows OS and Mac OS X.

Example

```
mytrash = filesystem.getRecycleBinInfo();
mesg = myTrash.numItems + " items (" + myTrash.size +
               " bytes)";
```

Availability

Available in version 1.8 or newer.

filesystem.isDirectory()

determines if a given path is a directory

Synopsis

`filesystem.isDirectory(`*path*`)`

Description

Returns true if the given `path` is a directory, false otherwise.

Example

```
isDir = filesystem.isDirectory(path);
```

Availability

Available in version 1.8 or newer.

filesystem.isPathAllowed()

tests security permissions for a path

Synopsis

```
bool filesystem.isPathAllowed( string )
```


Description

This function tests the given file path to see if the Widget has permission to access that location based on the security settings in the Widget's metadata file. A common use is to test the result of a selector preference after preferences have been changed.

Example

```
var canAccess = filesystem.isPathAllowed( preferences.selector.value )
```

Availability

Available in version 4.5 or newer.

filesystem.itemExists()

determines if a given path exists

Synopsis

`filesystem.itemExists(`*`path`*`)`

Description

Returns true if the given `path` exists (is a file or a directory), false otherwise.

Example

```
exists = filesystem.itemExists(path);
```

Availability

Available in version 1.8 or newer.

filesystem.move()

moves a file or files to a location

Synopsis

```
filesystem.move( path, destination );
```

Description

This function allows you to move a file or files to a specified destination. The source can be a single path, or an array of paths.

Version Notes

In versions earlier than 4.0, you cannot rename a file by moving it. As such, the destination must be a directory that exists. This function returns true if successful, false otherwise.

In version 4.0 and newer, this restriction is lifted, and the destination can point to a file that does not yet exist (though the parent directory must exist).

Example

```
filesystem.move( "myfile.txt", "/Users/evoas" );
```

Availability

Available in version 2.0 or newer.

filesystem.moveToRecycleBin()

filesystem.moveToTrash()

deletes items by moving them to the trash or recycle bin

Synopsis

```
filesystem.moveToRecycleBin(files)
filesystem.moveToTrash(files)
```

Description

Sends the specified file or files (you should provide an array of strings to delete multiple files at a time).

The function has two names to reflect the different terms used on the two platforms, Windows OS and Mac OS X.

Example

```
filesystem.moveToTrash(myTmpFile);
```

Availability

Available in version 1.8 or newer.

filesystem.open()

opens a file based on its file type/extension

Synopsis

```
filesystem.open( path )
```

Description

You can use this function to open an arbitrary file with the correct application. For example, passing a Widget file path into this function opens the Widget in the Yahoo! Widget Engine (i.e., it runs the Widget, as expected). Passing a folder in opens it in Finder/Explorer.

Example

```
filesystem.open( "PIM Overview.widget" );
```

Availability

Available in version 2.0 or newer.

filesystem.openRecycleBin()
filesystem.openTrash()

opens the folder that contains the items in the trash/recycle bin

Synopsis

```
filesystem.openRecycleBin()
filesystem.openTrash()
```

Description

This function opens a window showing the contents of the Trash/Recycle Bin. It is the equivalent of double-clicking the Trash or Recycle Bin icons.

The function has two names to reflect the different terms used on the two platforms, Windows OS and Mac OS X.

Example

```
filesytem.openTrash();
```

Availability

Available in version 2.0 or newer.

filesystem.readFile()

reads a text file into a string or array

Synopsis

```
filesystem.readFile( path [,asLines [,charset]] )
```

Description

This function is used to read a text file into either a string or array variable. If the optional second parameter is true, the file is read and broken into lines and an array of those lines is returned. Otherwise, just one long string of the contents is returned.

The Widget Engine can read most of the typical text file formats, but works best with either UTF-16 or UTF-8 encodings.

In version 4.5 and later, an optional third attribute exists to allow you to pass the `charset` of the file, if known.

Example

```
var data = filesystem.readFile( "myfile.txt" );

var lines = filesystem.readFile( "myfile.txt", true );
```

Availability

Available in version 2.0 or newer. The `charset` attribute is available in version 4.5 or newer.

filesystem.reveal()

makes the system file browser display an item in context

Synopsis

```
filesystem.reveal(path)
```

Description

This function causes the system file browser (Explorer on Windows OS, the Finder on Mac OS X) to display the directory containing the specified item. This is useful for revealing filesystem items to the user.

Example

```
filesystem.reveal(myPath);
```

Availability

Available in version 1.8 or newer.

filesystem.remove()

permanently deletes a file or directory

Synopsis

```
filesystem.remove(path|array)
```

Description

This function deletes a file, directory, or array of files or directories permanently (it is not moved to the trash/recycle bin).

Currently, this function can only be used to delete files from your Widget's WidgetData directory.

Example

```
filesystem.remove(myPath);
```

Availability

Available in version 4.0 or newer.

filesystem.unzip()

unzips an archive into the specified directory

Synopsis

```
bool filesystem.unzip( path, destination );
```

Description

This function allows you to unzip an archive into a specified destination directory. The destination directory and the path to the archive must already exist. The files inside the archive are placed directly in the destination directory.

This function returns true if successful, false otherwise.

Example

```
// to unzip a zipped Widget bundle
filesystem.unzip( "MyWidget.widget", "Unpacked" );

// the above would result in the following directory
// structure. Note that the top element in the Widget is
// MyWidget.widget. If you had simply zipped two files
// into the archive, it would just place those two files
// into the "Unpacked" directory.

Unpacked/MyWidget.widget
Unpacked/MyWidget.widget/Contents
Unpacked/MyWidget.widget/Contents/MyWidget.kon
Unpacked/MyWidget.widget/Contents/Resources
Unpacked/MyWidget.widget/Contents/Resources/MyImage.png
```

Availability

Available in version 4.0 or newer.

filesystem.writeFile()

writes a string or array to a text file

Synopsis

```
filesystem.writeFile( path, string | array [, append] )
```

Description

This function writes out a file given either a string or an array of strings. If given a string, the data is written out as is. If passed an array, the data is written out separated by return characters "\n." Currently, this function always writes files as UTF-8.

You can pass an optional third parameter that instructs this function to append to an existing file (creating if needed) instead of always creating a new file (the default).

Example

```
filesystem.writeFile( "myfile.txt", myData );
```

Availability

Available in version 2.0 or newer.

filesystem.zip()

creates a Zip archive from the specified files and directories

Synopsis

```
bool filesystem.zip( path | array, destination [, baseDir] );
```

Description

This function allows you to create a Zip archive from a specified set of files or directories. The list of items can be a single path or an array of paths.

The destination parameter can either be a path to a directory that exists, or the ultimate path to the archive file you wish to create. The parent directory of the archive file must already exist. If you supply a path to a directory, the default name Archive.zip is used. If this default name is used and a file of that name is already present in the destination directory, it will find a unique name such as Archive 1.zip so it will not overwrite an existing file. If you specify the archive name, however, an existing file with the same name will be overwritten.

The baseDir parameter is an optional parameter. By default, all items you pass to the function get added to the top of the Zip archive. You can specify a directory to use as the base directory inside the archive using the baseDir parameter. For example, you might use this to gather multiple files from different places on disk and put them into a single directory that is the top level of the Zip archive.

This function returns true if successful, false otherwise.

Example

```
// to zip up a Widget bundle
filesystem.zip( "MyWidget.widget", "Built/MyWidget.widget" );

// 5 files
filesystem.zip( new Array( "file1.png", "file2.png", "other/moon.png",
"other/sun.png", "yetAnother/path/tothis.png" ), "myfiles.zip" );

// The Above yields an archive that looks like:
file1.png
```

```
file2.png
moon.png
sun.png
tothis.png
// specify a base dir
filesystem.zip( new Array( "file1.png", "another/file2.png" ),
        "archive.zip", "Contents/Test" );

// the above yields
Contents/Test/file1.png
Contents/Test/file2.png
```

Availability

Available in version 4.0 or newer.

Screen

information about the display

Attributes

screen.availHeight
screen.availLeft
screen.availTop
screen.availWidth
screen.colorDepth
screen.height
screen.pixelDepth
screen.resolution
screen.width

Synopsis

```
screen
```

Description

The screen object has various attributes which describe the metrics of the current screen (the display the main window of a Widget is mostly on). See below for details of the individual attributes.

Example

```
for (a in screen)
   print("screen." + a + ": " + eval("screen." + a));
```

screen.availHeight

available height of the current screen

Synopsis

```
screen.availHeight
```

Description

The number of pixels available vertically on the screen most of the Widget's window occupies. This value omits space taken by things like the system menubar and the dock.

Example

```
myWindow.vOffset = screen.availHeight - 30;
```


screen.availLeft

leftmost available position on the screen

Synopsis

```
screen.availLeft
```

Description

The first available position at the left of the screen most of the Widget's window occupies that is not occupied by a system feature such as the dock.

Example

```
myWindow.hOffset = screen.availLeft + 30;
```

screen.availTop

topmost available position on the screen

Synopsis

```
screen.availTop
```

Description

The first available position at the top of the screen most of the Widget's window occupies that is not occupied by a system feature such as the menubar.

Example

```
myWindow.vOffset = screen.availTop + 10;
```

screen.availWidth

available width of the current screen

Synopsis

```
screen.availWidth
```

Description

The number of pixels available vertically on the screen most of the Widget's window occupies. This value omits space taken by system features like the dock.

Example

```
myWindow.width = screen.availWidth / 4;
```

screen.colorDepth

color depth of the current screen

Synopsis

```
screen.colorDepth
```

Description

The number of bits per pixel available on the screen most of the Widget's window occupies.

Example

```
alert("Bits per pixel: " + screen.colorDepth);
```

screen.height

height of the current screen

Synopsis

```
screen.height
```

Description

The number of pixels available vertically on the screen most of the Widget's window occupies. Normally `screen.availHeight` provides a more useful measure of the screen's height.

Example

```
myWindow.vOffset = screen.availHeight - 30;
```

screen.pixelDepth

color depth of the current screen

Synopsis

```
screen.pixelDepth
```

Description

The number of bits per pixel available on the screen most of the Widget's window occupies. This is a synonym for `screen.colorDepth` and is provided for compatibility.

Example

```
alert("Bits per pixel: " + screen.pixelDepth);
```

screen.resolution

resolution of the current screen

Synopsis

```
screen.resolution
```

Description

The raster resolution in dots per inch (dpi) of the screen most of the Widget's window occupies.

Example

```
alert("Screen resolution: " + screen.resolution);
```

screen.width

the current screen's width

Synopsis

```
screen.width
```

Description

The number of pixels available horizontally on the screen most of the Widget's window occupies. Normally `screen.availWidth` provides a more useful measure of the screen's width.

Example

```
myWindow.hOffset = screen.width - 80;
```


System

information about the computer or environment

Attributes

system.airport, system.wireless
system.airport.available, system.wireless.available
system.airport.info, system.wireless.info
system.airport.network, system.wireless.network
system.airport.noise, system.wireless.noise
system.airport.powered, system.wireless.powered
system.airport.signal, system.wireless.signal
system.appearance
system.battery
system.battery[n].currentCapacity
system.battery[n].isCharging
system.battery[n].isPresent
system.battery[n].maximumCapacity
system.battery[n].name
system.battery[n].powerSourceState
system.battery[n].timeToEmpty
system.battery[n].timeToFullCharge
system.battery[n].transportType
system.batteryCount
system.clipboard
system.cpu
system.cpu.activity
system.cpu.idle
system.cpu.nice
system.cpu.numProcessors
system.cpu.sys
system.cpu.user
system.event
system.languages
system.memory
system.memory.availPhysical
system.memory.availVirtual
system.memory.load
system.memory.totalPhysical
system.memory.totalVirtual
system.mute
system.platform
system.userDocumentsFolder, system.userDesktopFolder, system.userPicturesFolder,
system.userMoviesFolder, system.userMusicFolder, system.userWidgetsFolder,
system.applicationsFolder, system.temporaryFolder, system.trashFolder
system.volume
system.widgetDataFolder

Description

The system object is your interface to things about your computer or some aspect of the environment. For example, you can get information about the state of the battery or wireless connection if present.

system.airport, system.wireless

built-in support for accessing WiFi/AirPort information

Attributes

Attributes	Description
`available`	True if WiFi/AirPort is installed
`info`	A summary of WiFi/AirPort status
`network`	The name of the current network
`noise`	The connection's noise level
`powered`	True if WiFi/AirPort is powered on
`signal`	The connection's signal level

Description

The settings and status of an installed WiFi/AirPort card are available through the `system.airport` or `system.wireless` object.

The WiFi/AirPort Widget makes extensive use of this object.

Example

```
if (system.airport.available && system.airport.powered)
  alert("Current network is " + system.airport.network);
if(system.wireless.available && system.wireless.powered)
  alert("Current network is "+ system.wireless.network);
```

system.airport.available, system.wireless.available

checks if a WiFi/AirPort (or other compatible wireless card) is installed

Synopsis

```
system.airport.available
system.wireless.available
```

Description

The `available` attribute returns a Boolean true/false value that corresponds to the availability of the wireless device capable of connecting to a network.

Example

```
if (! system.airport.available)
    signal_status.src = "NoCard.png";
else
    signal_status.src = "Signal.png";
if (! system.wireless.available)
    signal_status.src = "NoCard.png";
else
    signal_status.src = "Signal.png";
```

See Also

`system.airport.signal, system.wireless.signal`

system.airport.info, system.wireless.info

WiFi/AirPort status summary

Synopsis

```
system.airport.info
system.wireless.info
```

Description

A brief, human-readable description of WiFi/AirPort status.

Example

```
alert(system.airport.info);
alert(system.wireless.info);
```

system.airport.network, system.wireless.network

name of current WiFi/AirPort network

Synopsis

```
system.airport.network
system.wireless.network
```

Description

This attribute contains the name of the current WiFi/AirPort network, if any.

Example

```
alert("AirPort network " + system.airport.network + " in use");

alert("WiFi network " + system.wireless.network + " in use");
```

system.airport.noise, system.wireless.noise

noise level of the current WiFi/AirPort connection

Synopsis

```
system.airport.noise
system.wireless.noise
```

Description

This attribute contains a numeric value that indicates the level of noise on the current WiFi/AirPort connection.

This value is not generally reliable.

Example

```
if (system.airport.noise > 20)
  status.src = "noisy.png";
if (system.wireless.noise > 20)
  status.src = "noisy.png";
```

Platform Notes

On Windows OS, this attribute is always zero.

system.airport.powered, system.wireless.powered

checks if the WiFi/AirPort card is on or off

Synopsis

```
system.airport.powered
system.wireless.powered
```

Description

This Boolean variable indicates whether the WiFi/AirPort is currently turned on or off. Use this to decide whether to access the other WiFi/AirPort status attributes.

Example

```
if (system.airport.available && system.airport.powered)
  alert("Current network is " + system.airport.network);
if(system.wireless.available && system.wireless.powered)
  alert("Current network is "+ system.wireless.network);
```

system.airport.signal, system.wireless.signal

signal strength of the current WiFi/AirPort connection

Synopsis

```
system.airport.signal
system.wireless.signal
```

Description

The `signal` attribute of the WiFi/AirPort object returns a number value that corresponds to the signal strength of the wireless network the device is connected to.

Example

```
theStrength = system.airport.signal;
if ( theStrength =&lt; 33 )
   signalBars.src = "halfFull.png"
theStrength = system.wireless.signal;
if ( theStrength =&lt; 50 )
   signalBars.src = "halfFull.png"
```

Version Notes

In version 4.0 of the Widget Engine, the range is 0-75 and is not a linear mapping to Apple's signal strength.

system.appearance

appearance of the current system

Synopsis

```
system.appearance
```

Description

The current appearance of the system.

Example

```
if (system.appearance == "Graphite")
```


```
  header.src = "graphiteHeader.png";
else
  header.src = "aquaHeader.png";
```

Version Notes

In version 4.0 of the Widget Engine we do not support notifying Widgets of an appearance change.

Platform Notes

As of Mac OS X 10.3, this will only be `Blue` or `Graphite`.

If your Widget uses images that you would like to be specific to the current Mac OS X appearance, simply use this variable to get the running appearance and adjust your image source file appropriately.

It should be noted that these get returned with initial caps, so make sure you test for the words "Blue" and "Graphite," not "blue" and "graphite."

On Windows OS, the value `Blue` is always returned.

system.battery

built-in support for accessing battery and UPS information

Synopsis

```
system.battery
```

Description

The battery number is an array that's 0 based. Single battery laptops are always `battery[0]`, however when run on a computer with dual batteries, the expected primary bay registers as `battery 1`, and the optional battery bay registers as `battery 0`. The number of batteries installed in the current system is available in `system.batteryCount`.

Version Notes

In version 4.0 of the Widget Engine, only one battery is supported on Windows OS (however, information about it is an aggregate of all batteries in the system).

system.battery[*n*].currentCapacity

current charge percentage of the battery

Synopsis

```
system.battery[batteryNumber].currentCapacity
```

Description

Current charge percentage of the battery.

system.battery[*n*].isCharging

charging state of battery

Synopsis

```
system.battery[batteryNumber].isCharging
```

Description

True if battery is being charged (i.e., it is at less that 100% capacity and the system is plugged into AC power).

system.battery[*n*].isPresent

checks if battery is installed

Synopsis

```
system.battery[batteryNumber].isPresent
```

Description

True if battery is physically present.

system.battery[*n*].maximumCapacity

maximum charge capacity of the battery

Synopsis

```
system.battery[batteryNumber].maximumCapacity
```

Description

Maximum capacity of the battery (since capacity is represented as a percentage, this is always 100).

system.battery[*n*].name

name of the battery

Synopsis

```
system.battery[batteryNumber].name
```

Description

The human-readable name of the battery.

system.battery[*n*].powerSourceState

current power source

Synopsis

```
system.battery[batteryNumber].powerSourceState
```

Description

Returns "AC Power" or "Battery Power" based on whether the system is plugged in or not.

system.battery[*n*].timeToEmpty

minutes of charge remaining

Synopsis

```
system.battery[batteryNumber].timeToEmpty
```

Description

This value is in minutes. A value of -1 means the system is still determining how fast the battery is draining (also known as the "calculating" phase).

system.battery[*n*].timeToFullCharge

minutes until battery is fully charged

Synopsis

```
system.battery[batteryNumber].timeToFullCharge
```

Description

This value is in minutes. A value of `-1` means the system is still determining how fast the battery is charging (also known as the "calculating" phase).

Note: The `system.battery[n].currentCapacity` value is generally a more reliable determination of how charged the battery is.

Example

```
alert(system.battery[0].timeToFullCharge + ' minutes to full charge');
```

system.battery[*n*].transportType

battery communication channel

Synopsis

```
system.battery[batteryNumber].transportType
```

Description

"Internal" or method of UPS communication.

system.batteryCount

number of batteries installed

Synopsis

```
system.batteryCount
```

Description

The number of batteries installed in the current system is available in `system.batteryCount`. Normally this is 1, but some laptops support more so any Widget that intends to work with batteries should take this into account.

Example

```
for (b = 0; b < system.batteryCount; b++)
   totalTime += system.battery[b].timeToEmpty;
```

Platform Notes

For Windows OS, the value of this attribute is always 1 (though the power available from all batteries is reported).

system.clipboard

accesses the current system clipboard

Synopsis

```
system.clipboard
```

Description

system.clipboard contains the text (if any) on the system clipboard. Setting this attribute loads the system clipboard with that data, removing anything there previously.

Example

```
myText = system.clipboard;
myNewText = "--<(" + myText + ")>--";

system.clipboard = myNewText;
```

system.cpu

information about the current CPU load

Synopsis

```
system.cpu
```

Description

system.cpu is an object with several members that summarize the level of activity of the system CPU (members are detailed below).

Notes

The underlying mechanism that gathers this data has a resolution of 1 second so that is as quickly as this information can change. In other words, polling system.cpu more than once per second is not useful.

Example

```
for (a in system.cpu)
   print("system.cpu." + a + ": " + eval("system.cpu." + a));
```

See Also

system.cpu.activity, system.cpu.idle, system.cpu.nice, system.cpu.numProcessors, system.cpu.sys, system.cpu.user

system.cpu.activity

information about the current CPU activity

Synopsis

```
system.cpu.activity
```

Description

system.cpu.activity contains the current percentage load of the CPU. If the computer is very busy, it is near 100. It is the sum of the other system.cpu members user, sys, and nice. It represents the load of the computer as a whole, no matter how many processors it has.

Example

```
load = system.cpu.activity;
```

system.cpu.idle

information about idle CPU cycles

Synopsis

```
system.cpu.idle
```


Description

`system.cpu.idle` provides a measure (in percentage) of how much of the CPU is available for more work.

Example

```
idle_percent = system.cpu.idle;
```

system.cpu.nice

information about raised priority CPU cycles

Synopsis

```
system.cpu.nice
```

Description

`system.cpu.nice` is a measure of how much of the CPU is occupied with tasks whose priority has been raised (normal processes are reported as `system.cpu.user`).

Example

```
priorityTasks = system.cpu.nice;
```

Platform Notes

For Windows OS, the value of this attribute is always zero.

system.cpu.numProcessors

number of processors in the system

Synopsis

```
system.cpu.numProcessors
```

Description

`system.cpu.numProcessors` indicates how many processors there are in the current system.

Example

```
if (system.cpu.numProcessors == 2)
  system = "Dualie";
```

system.cpu.sys

information about system CPU cycles

Synopsis

```
system.cpu.sys
```

Description

`system.cpu.sys` contains the percentage of the CPU occupied with system tasks (as opposed to user tasks).

Example

```
systemTime = system.cpu.sys;
```

system.cpu.user

information about user CPU cycles

Synopsis

```
system.cpu.user
```

Description

`system.cpu.user` is a measure of how much of the CPU is occupied with normal tasks (as opposed to system tasks).

Example

```
userTasks = system.cpu.user;
```

system.event

information about the last event received

Synopsis

```
system.event
```

Description

`system.event` contains a variety of information about the last event the Widget received (typically as the result of a user action such as a mouse click).

In 4.5 and later, the event stored in `system.event` is the event being dispatched. For this reason, the information contained within the event will change depending on the class of event being dispatched. For example, if a mouse event is being sent, that event is stored in `system.event` as well as sent to any event listeners. Since the event is a mouse event, it will not contain `keyCode`, for example. Likewise, a load event will not contain mouse or keyboard information at all. Consult the Events section for information about any specific event that might be dispatched.

system.languages

current set of languages preferred by the user

Synopsis

```
system.languages
```

Description

`system.languages` contains the list of languages the user has specified in the International System Preference panel. Element 0 is their primary language, 1 their second choice, and so forth.

You can only read this setting; it cannot be changed except by using the System Preferences panel.

Example

```
print("system.languages: " + system.languages);

system.languages: en,de,ja,fr,nl,it,es,zh_TW
```


system.memory

information about the physical/virtual memory of a computer

Attributes

```
availPhysical
availVirtual
load
totalPhysical
totalVirtual
```

Description

You can inspect the amount of memory on a computer through this system object. Note that at present, the virtual memory numbers are somewhat suspicious on both platforms.

system.memory.availPhysical

amount of available physical memory

Description

Returns the number of bytes of available physical memory. You can use `bytesToUIString` to turn this into something more user-friendly.

Example

```
print( "available RAM: " +
   bytesToUIString( system.memory.availPhysical ) );
```

Availability

Available in version 2.0 or newer.

system.memory.availVirtual

amount of available virtual memory

Description

Returns the number of bytes of available virtual memory. You can use `bytesToUIString` to turn this into something more user-friendly.

Example

```
print( "avail virtual memory: " +
   bytesToUIString( system.memory.availVirtual ) );
```

Availability

Available in version 2.0 or newer.

Notes

This number is not quite accurate at present, particularly on Windows OS.

system.memory.load

percentage of used memory

Description

Returns a number from 0 to 100 indicating the current amount of physical RAM that is in use.

Example

```
print( "current system load: " +
   system.memory.load + "%" );
```

Availability

Available in version 2.0 or newer.

system.memory.totalPhysical

amount of physical RAM installed

Description

Returns the number of bytes of installed physical memory. You can use `bytesToUIString` to turn this into something more user-friendly.

Example

```
print( "Installed RAM: " +
   bytesToUIString( system.memory.totalPhysical ) );
```

Availability

Available in version 2.0 or newer.

system.memory.totalVirtual

amount of total virtual memory

Description

Returns the number of bytes of total virtual memory. You can use `bytesToUIString` to turn this into something more user-friendly.

Example

```
print( "total virtual memory: " +
   bytesToUIString( system.memory.totalVirtual ) );
```

Availability

Available in version 2.0 or newer.

Platform Notes

This number is not quite accurate at present, particularly on Windows OS.

system.mute

gets or sets the mute state of your system volume

Synopsis

```
system.mute
```

Description

This variable reflects whether the computer's sound is muted. Setting it to true mutes the system sound.

Examples

```
// Find out if the computer is muted or not
if ( system.mute )
```


```
    print("What? I can't hear you!");
else
    print("I can hear sounds from my Mac!");
// Turn off system sound
system.mute = true;
```

system.platform

type of system the Widget is running on

Synopsis

```
system.platform
```

Description

This variable contains the current platform a Widget is executing on.

Example

```
print("platform: " + system.platform);
```

On Mac OS X results in:

```
platform: macintosh
```

On Windows OS results in:

```
platform: windows
```

system.userDocumentsFolder, system.userDesktopFolder, system.userPicturesFolder, system.userMoviesFolder, system.userMusicFolder, system.userWidgetsFolder, system.applicationsFolder, system.temporaryFolder, system.trashFolder

system variables that contain the names of various user folders

Synopsis

```
system.userDocumentsFolder
system.userDesktopFolder
system.userPicturesFolder
system.userMoviesFolder
system.userMusicFolder
system.userWidgetsFolder
system.applicationsFolder
system.temporaryFolder
system.trashFolder
```

Description

These variables contain the paths of various user-centric and system folders. The correct locations can be determined in a platform-independent manner using these variables.

Example

```
print("userMusicFolder: " + system.userMusicFolder);
```

Mac OS X results:

```
userMusicFolder: /Users/joe/Music
```

Windows OS results:

```
userMusicFolder:
    c:/Documents and Settings/joe/My Documents/My Music
```

system.volume

gets or sets the system audio volume

Synopsis

```
system.volume
```

Description

This variable reflects the current audio volume. Setting it to a number between 0 and 16 to changes the system volume level. Setting the volume to 0 is the same as setting `system.mute` to true.

Example

```
// Set the audio volume to 50%
system.volume = 8;
```

system.widgetDataFolder

name of folder where the Widget can safely store data

Synopsis

```
system.widgetDataFolder
```

Description

This variable contains the name of a folder on the user's hard disk where persistent data (or even data that needs to be cached for a short length of time) can safely be saved by the Widget. Historically, Widgets have tried to save data inside their own bundles, but this has various drawbacks chiefly that the location might not be writable. Each Widget gets a separate folder that is created if it does not already exist.

Example

```
saveFileName = system.widgetDataFolder + "/data";
```

Platform Notes

On the Mac, the location of this folder is ~/Library/Application Support/Konfabulator/Widgets.

On the PC, it is located in C:\Documents and Settings\<user>\Local Settings\Application Data\Yahoo\Widget Engine\Widget Data.

Miscellaneous DOM Reference

These attributes and functions give JavaScript code access to certain applications allowing remote control and retrieval of data.

Currently the only supported application is `iTunes` (available from http://www.apple.com/itunes for both Windows OS and Mac OS X).

iTunes

gets information from and interacts with iTunes

Attributes

`iTunes.playerPosition`
`iTunes.playerStatus`
`iTunes.random iTunes.shuffle`
`iTunes.repeatMode`
`iTunes.running`
`iTunes.streamURL`
`iTunes.trackAlbum`
`iTunes.trackArtist`
`iTunes.trackLength`
`iTunes.trackRating`
`iTunes.trackTitle`
`iTunes.trackType`
`iTunes.version`
`iTunes.volume`

Functions

`iTunes.backTrack()`
`iTunes.fastForward()`
`iTunes.nextTrack()`
`iTunes.pause()`
`iTunes.play()`
`iTunes.playPause()`
`iTunes.resume()`
`iTunes.rewind()`
`iTunes.stop()`

Synopsis

```
iTunes
```

Description

The `iTunes` object allows remote control and display of iTunes track and artist information. See below for details of the individual functions and attributes.

Availability

The `iTunes` object is available in version 1.8 or newer.

iTunes.playerPosition

current position within the current track

Description

This attribute returns the current position (in seconds) within the currently playing track. Setting it moves the playback position to the specified number of seconds into the track.

Synopsis

```
iTunes.playerPosition
```

Example

```
position = iTunes.playerPosition;
```

iTunes.playerStatus

string describing the current state of iTunes

Synopsis

```
iTunes.playerStatus
```

Description

This attribute returns one of the following strings: `stopped`, `paused`, `playing`, `fast forwarding`, `rewinding`, or `unknown`.

Example

```
currentState = iTunes.playerStatus;
```

iTunes.random
iTunes.shuffle

shuffle state of iTunes

Synopsis

```
iTunes.random
```

Description

This attribute reflects the shuffle state of iTunes. If the current playlist is set to shuffle, it is true, false otherwise. Setting the attribute changes iTunes' shuffle state.

Example

```
iTunes.random = 1;
shuffleState = iTunes.shuffle;
```

iTunes.repeatMode

current repeat mode of iTunes

Synopsis

```
iTunes.repeatMode
```

Description

This attribute returns one of the following strings: `off`, `one`, or `all` indicating the current repeat mode. The repeat mode can be set by setting the attributes to one of those strings.

Example

```
mode = iTunes.repeatMode;
iTunes.repeatMode = 'off';
```

iTunes.running

checks if iTunes is currently running

Synopsis

```
iTunes.running
```

Description

Use this attribute to determine if iTunes is currently running.

Example

```
iTunes.running;
```

iTunes.streamURL

URL of the currently playing stream

Synopsis

```
iTunes.streamURL
```

Description

If iTunes is currently playing an audio stream, this attribute contains the URL of the stream.

Example

```
url = iTunes.streamURL;
```

iTunes.trackAlbum

name of the current album

Synopsis

```
iTunes.trackAlbum
```

Description

This attribute contains the name of the current album (if known). If a stream is playing, the name of the stream appears here.

Example

```
currAlbum = iTunes.trackAlbum;
```

iTunes.trackArtist

artist of the current track

Synopsis

```
iTunes.trackArtist
```

Description

This attribute contains the name of the artist of the current album (if known). If a stream is playing, this information is not available.

Example

```
artist = iTunes.trackArtist;
```

iTunes.trackLength

length of the current track

Synopsis

```
iTunes.trackLength
```

Description

This attribute contains the length of the currently playing track. If a stream is playing, this information is not available.

Example

```
len = iTunes.trackLength;
```

iTunes.trackRating

rating of the current track

Synopsis

```
iTunes.trackRating
```

Description

This attribute contains the rating of the currently playing track. Setting the attribute changes the current track's rating in iTunes. If a stream is playing, this information is not available.

Example

```
rating = iTunes.trackRating;
```

iTunes.trackTitle

title of the current track

Synopsis

```
iTunes.trackTitle
```

Description

This attribute contains the title of the currently playing track. If a stream is playing, this information might not be available.

Example

```
title = iTunes.trackTitle;
```

iTunes.trackType

type of the current track

Synopsis

```
iTunes.trackType
```


Description

This attribute contains the type of the currently playing track. It can include one of the following: `audio file`, `audio cd track`, `audio stream`, `audio device`, `shared library`, or `unknown`.

Example

```
tt = iTunes.trackType;
```

iTunes.version

version of iTunes

Synopsis

```
iTunes.version
```

Description

This attribute contains the version of the copy of iTunes that is being controlled.

Example

```
log("iTunes Version: " + iTunes.version);
```

iTunes.volume

current iTunes volume

Synopsis

```
iTunes.volume
```

Description

This attribute reflects the volume iTunes is playing at. This can be between 0 and 100. Assign a value to the attribute to change the volume.

Example

```
iTunes.volume = 60;
```

iTunes.backTrack()

tells iTunes to move to the previous track

Synopsis

```
iTunes.backTrack()
```

Description

Tells iTunes to move to the previous track.

Example

```
iTunes.backTrack();
```

See Also

`iTunes.nextTrack()`

iTunes.fastForward()

tells iTunes to fast forward within the current track

Synopsis

```
iTunes.fastForward()
```

Description

Tells iTunes to skip forward within the current track.

Example

```
iTunes.fastForward();
```

See Also

iTunes.rewind()

iTunes.nextTrack()

tells iTunes to move to the next track

Synopsis

```
iTunes.nextTrack()
```

Description

Tells iTunes to move to the next track.

Example

```
iTunes.nextTrack();
```

See Also

iTunes.backTrack()

iTunes.pause()

tells iTunes to pause playback

Synopsis

```
iTunes.pause()
```

Description

Tells iTunes to pause playback.

Example

```
iTunes.pause();
```

See Also

iTunes.resume()

iTunes.play()

tells iTunes to start playing the current track

Synopsis

```
iTunes.play()
```


Description

Tells iTunes to play the current track.

Example

```
iTunes.play();
```

See Also

iTunes.pause()

iTunes.playPause()

tells iTunes to toggle between playing and pause

Synopsis

```
iTunes.playPause()
```

Description

Tells iTunes to play if it's currently paused, or pause if it's currently playing.

Example

```
iTunes.playPause();
```

iTunes.resume()

tells iTunes to resume playback

Synopsis

```
iTunes.resume()
```

Description

Tells iTunes to resume playback after being paused.

Example

```
iTunes.resume();
```

See Also

iTunes.pause()

iTunes.rewind()

tells iTunes to skip backward

Synopsis

```
iTunes.rewind()
```

Description

Tells iTunes to skip backward in the current track.

Example

```
iTunes.rewind();
```

See Also

iTunes.fastForward()

iTunes.stop()

tells iTunes to stop playing

Synopsis

```
iTunes.stop()
```

Description

Tells iTunes to stop playing.

Example

```
iTunes.stop();
```

See Also

iTunes.play()

URL Object

allows you to express a URL and retrieve data from that URL

XML Name

Not available.

JavaScript Name

URL

Description

The URL object represents a URL in its entirety or its separate parsed parts. A URL can then be used to manage a connection to a remote resource. URLs are never defined in the XML section of a Widget.

The Widget Engine also supports XMLHttpRequest (see that section for information), which is a more standard method of doing web connections.

Attributes

URL.autoRedirect
URL.hostname
URL.location
URL.outputFile
URL.password
URL.path
URL.port
URL.postData
URL.queryString
URL.response
URL.responseData
URL.result
URL.scheme
URL.timeout
URL.username

Functions

URL.addPostFile()
URL.cancel()
URL.clear()
URL.fetch()

URL.fetchAsync()
URL.getResponseHeaders()
URL.setRequestHeader()

URL.autoRedirect

controls whether to automatically follow redirects

Synopsis

```
URL.autoRedirect
```

Description

This attribute allows you to control whether a URL object follows redirects automatically. The default is true. Setting this to false allows you to get the 302 redirect response and process it as you wish.

Example

```
var myURL = new URL;
myURL.autoRedirect = false;
myURL.location = "http://mysite.com";
myURL.fetch();
```

Availability

Available in version 2.1 or newer.

URL.hostname

provides access to the requested hostname for the URL

Synopsis

```
URL.hostname
```

Description

This attribute allows you access to the `hostname` part of the URL as parsed. If the URL does not parse correctly or no hostname has been provided, then `hostname` may be empty. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://myserver.com/path/to/file?today=TGIF");
var myVar = myURL.hostname; //assigns "myserver.com" to myVar
```

Availability

Available in version 4.5 or newer.

URL.location

web address of the URL

Synopsis

```
URL.location
```

Description

Specifies the web address that the URL will fetch data from.

Example

```
var url = new URL();
url.location = "http://www.yahoo.com";
contents = url.fetch();
```

URL.outputFile

file to store the fetched data in

Synopsis

```
URL.outputFile
```

Description

Specifies an optional file into which fetched data will be stored. If you are retrieving textual data (e.g., an HTML file) it is usually easier to just use the return value of `URL.fetch()`, but if you are retrieving binary data (e.g., an image file) then the retrieved data must be stored in a file as the process of converting it to a string will render it invalid.

Example

```
var url = new URL();
url.outputFile = system.widgetDataFolder + "/mytempfile";
url.location = "http://www.example.com/graphic.jpg";
url.fetch();
myIng.src = url.outputFile;
```

URL.password

provides access to the requested password for the URL

Synopsis

```
URL.password
```

Description

This attribute allows you access to the `password` part of the URL as parsed. If the URL does not parse correctly or no password has been provided then `password` may be empty. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://sam:l33tspeak@myserver.com/path/to/file?today=TGIF");
var myVar = myURL.password; //assigns "l33tspeak" to myVar
```

Availability

Available in version 4.5 or newer.

URL.path

provides access to the requested path for the URL

Synopsis

```
URL.path
```


Description

This attribute allows you access to the `path` part of the URL as parsed. If the URL does not parse correctly or no path has been provided, then `path` defaults to `"/"`. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://myserver.com/path/to/file?today=TGIF");
var myVar = myURL.path; //assigns "/path/to/file" to myVar
```

Availability

Available in version 4.5 or newer.

URL.port

provides access to the requested port for the URL

Synopsis

```
URL.port
```

Description

This attribute allows you access to the `port` part of the URL as parsed. If the URL does not parse correctly or no port has been provided, then the `port` may be `-1`. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://myserver.com:8080/path/to/file?today=TGIF");
var myVar = myURL.port; //assigns 8080 to myVar
```

Availability

Available in version 4.5 or newer.

URL.postData

data to be POSTed to a web server

Synopsis

```
URL.postData
```

Description

Setting `postData` causes a `URL` object to POST to its location rather than performing a GET operation. To post nothing, set `postData` to an empty string. To make the `URL` object GET again, set `postData` to null.

The format of this data should be URL encoded, i.e., each parameter is passed as `name=value` and parameters are separated by a "&" symbol. Use `encode()` when your data contains spaces, "&," "=," or non-ASCII characters.

Example

```
var url = new URL();
var text = encode( "a lot of &&& bad text" );
url.postData = "x=123&y=456&q=" + text;
contents =
    url.fetch("http://www.example.com/myscript.php");
```

URL.queryString

provides access to the requested queryString for the URL

Synopsis

```
URL.queryString
```

Description

This attribute allows you access to the `queryString` part of the URL as parsed. If the URL does not parse correctly or no queryString has been provided, then `queryString` may be empty. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://myserver.com/path/to/file?today=TGIF");
var myVar = myURL.queryString; //assigns "today=TGIF" to myVar
```

Availability

Available in version 4.5 or newer.

URL.response

HTTP response code for the last fetch

Synopsis

```
URL.response
```

Description

The `response` attribute indicates the HTTP response code received as a result of the last `fetch` call. Codes greater than or equal to 400 indicate that there was a problem completing the request.

Note: A `response` code is only available if a web server was contacted and a request made. If the server is not available or an invalid URL is supplied for the `location`, then the response attribute is 0. A successful web page retrieval is usually indicated by a response code of 200. By default, the Widget Engine does redirection automatically, so you will never see a response code of 302 unless you set the `autoRedirect` attribute to false.

Example

```
var url = new URL();
url.location = "http://www.yahoo.com";
contents = url.fetch();
log("Response was: " + url.response);
```

URL.responseData

result of the last request

Synopsis

```
URL.responseData
```

Description

The `responseData` attribute is used to get the actual text response from the server regardless of the status code sent back. This differs from the `URL.result` attribute in that you always get the real response text back and never any status string. If the connection failed, this attribute is empty.

With this attribute, you can get the actual 404 page that is returned if you get a 404 error from the server.

Example

```
var url = new URL();
url.location = "http://www.mysite.com/a_url_that_doesnt_exist";
url.fetch();

// will print "Could not load URL"
print( url.result );

// will print the actual response from the server
print( url.responseData );
```

Availability

Available in version 2.1.1 or newer.

URL.result

result of the last request, or an error string

Synopsis

```
URL.result
```

Description

The `result` attribute indicates the result received from the last request made using `fetch()` or `fetchAsync()`. This contains the actual text of the result (e.g., a web page), or the error strings "Could not load URL" or "Could not load URL with POST." If you need the actual response even when the status code from the server is not 200, use `responseData` in version 2.1.1 or newer.

Example

```
var url = new URL();
url.location = "http://www.yahoo.com";
url.fetch();
print( url.result );
```

Availability

Available in version 2.1 or newer.

URL.scheme

provides access to the requested scheme for the URL

Synopsis

```
URL.scheme
```

Description

This attribute allows you access to the `scheme` part of the URL as parsed. If the URL does not parse correctly the `scheme` may be empty. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://myserver.com/path/to/file?today=TGIF");
var myVar = myURL.scheme; //assigns "http" to myVar
```

Availability

Available in version 4.5 or newer.

URL.timeout

length of time to wait for a response from the server

Synopsis

```
URL.timeout
```

Description

This attribute sets the time to wait for a server response. The default timeout for a request is 60 seconds. You can only set `timeout` for an integral number of seconds.

Example

```
var url = new URL();
url.location = "http://www.yahoo.com";
url.timeout = 120; // two minutes
url.fetch();
print( url.result );
```

Availability

Available in version 2.1 or newer.

URL.username

provides access to the requested username for the URL

Synopsis

```
URL.username
```

Description

This attribute allows you access to the `username` part of the URL as parsed. If the URL does not parse correctly or no username has been provided, then `username` may be empty. This value can be written to, and in so doing effects the `URL.location` as well as any subsequent web connections.

Example

```
var myURL = new URL("http://sam:l33tspeak@myserver.com/path/to/file?today=TGIF");
var myVar = myURL.username; //assigns "sam" to myVar
```

Availability

Available in version 4.5 or newer.

URL.addPostFile()

adds a file for a multipart POST request

Synopsis

```
URL.addPostFile(path)
```


Description

This function adds a file path to a list of files to be sent along with a POST request. The path is not tested for existence until the POST is sent. When files have been added, your request is automatically set to be a POST request.

Example

```
var myURL = new URL;
myURL.addPostFile( "myfile.png", "/A/Local/File/Path.png" );
myURL.location = "http://mysite.com";
myURL.fetch();
```

Availability

Available in version 2.1 or newer.

URL.cancel()

cancels an asynchronous request sent using fetchAsync()

Synopsis

```
URL.cancel()
```

Description

This function cancels an outstanding request sent using `fetchAsync()`. It has no effect if there is no async request pending. If called, the request is dropped and the function that would normally receive the result of the request is not called.

Example

```
var myURL = new URL;
myURL.location = "http://mysite.com";
myURL.fetchAsync( myCallback );
...
myURL.cancel();
```

Availability

Available in version 2.1 or newer.

URL.clear()

clears the current settings of a URL object

Synopsis

```
URL.clear()
```

Description

After using a URL object, if you wish to reuse it to send another request, you can call the `clear()` method to ensure any prior post data (e.g., files) is gone from the object. If you call this function on a URL object that is currently running an async request, the request is canceled before the object is cleared.

Example

```
myURL = new URL;
myURL.location = "http://widgets.yahoo.com/"
result = myURL.fetch();
// reuse the object
```

```
myURL.clear();
myURL.location = "http://www.yahoo.com/"
result = myURL.fetch();
```

Availability

Available in version 2.1 or newer.

URL.fetch()

returns URL data as a string

Synopsis

```
URL.fetch([location])
```

Description

Retrieves data from the remote location specified or from the web address specified in the URL's `location` attribute. If a location is specified, this also sets the value of the `location` attribute of the URL. The data is either returned as a string (the default) or into a file if the `outputFile` attribute has been set. This is done synchronously so the Widget pauses until the data is retrieved.

If an error occurs and `fetch()` is returning a string, then it returns the string "Could not load URL" (or the string "Could not load URL with POST" if the attribute `postData` is set). The `response` attribute contains the code indicating the type of error.

Note: If you are retrieving an RSS feed (or any web resource) , make sure you do not fetch it too often. Any frequency shorter than 30 minutes should be carefully considered. Your Widget might be used by thousands of people and the web site supplying the data may not appreciate the automated traffic. Also make sure that you do not implement a scheme that causes all instances of a Widget to try and fetch data at the same time (e.g., every hour on the hour) as this can also cause problems for sites (using an `onTimer` action is fine because different people's Widgets start at different times).

Example

```
var url = new URL();
webAddress = "http://www.yahoo.com";
contents = url.fetch(webAddress);
```

Version Notes

In version 2.1 or newer, you can also get the result using the `result` attribute.

URL.fetchAsync()

GETs or POSTs something asynchronously

Synopsis

URL.fetchAsync(*function*)

Description

This works similarly to `fetch()` except that it performs the request asynchronously, leaving your Widget to go about its business while the request completes. When the request is done, it calls the function you pass into the function. Your function receives the `url` object that started the request, which you can query to get the result and the response of the request.

Use of this function greatly improves the responsiveness of your Widget, allowing the user to drag and otherwise interact with it while the request is running.

Example

```
var url = new URL();
url.location = "http://www.yahoo.com";
url.fetchAsync(url_done);

function url_done( url )
{
   print( "fetch complete" );
   print( "response: " + url.response );
   print( "result: " + url.result );
}
```

Availability

Available in version 2.1 or newer.

URL.getResponseHeaders()

returns headers from an HTTP response

Synopsis

```
URL.getRespsonseHeaders( name )
```

Description

This function allows you to get the headers that accompany an HTTP response. The most useful purpose of this is to get "Set-Cookie" headers for use at a later time. Version 2.1 and later disable automatic cookie handling for security reasons, so if your Widget needs to use cookies to work, you need to use this function to get them out of a response. You can then pass the cookies back to the server in a later call to `fetch()` by setting them with `setRequestHeader`.

This function returns an array of the headers that match the name you pass in. In version 3.0 or newer, you can pass "*" as the name and you will receive an array of the complete headers, including the name (passing a name yields the value of the headers only).

Example

```
var url = new URL();
url.location = "http://www.my_site.com";|
url.fetch();
var cookies = URL.getResponseHeaders( "Set-Cookie" );
```

Availability

Available in version 2.1 or newer.

URL.setRequestHeader()

sets a header on an HTTP request

Synopsis

```
URL.setRequestHeader( name, value )
```

Description

This function sets headers to accompany an HTTP request. The most common use of this is to set cookies for a request. Version 2.1 disables automatic cookie support, so this function is necessary in order to continue to use cookies. With the `getResponseHeaders` function, this function can be used to deal with

cookies in your Widget. After receiving cookies in a prior response (see URL.getResponseHeaders()), you can use this function to set the cookie or cookies in a future request. The name parameter is the name of the header, the value parameter is the actual contents of the header.

Example

```
var url = new URL();
url.location = "http://www.my_site.com";
url.setRequestHeader( "Cookie", myCookie );
url.fetch();
```

Availability

Available in version 2.1 or newer.

Animation

objects and functions to aid in doing animations

Version 2.1 and later of the Widget Engine contains animation support that allows you to do animations asynchronously as well as synchronously. It also allows you to do custom animations written in JavaScript. You can fade, move, resize, and rotate objects.

A new object called animator controls the animation. You tell the animator to start an animation and run it asynchronously, allowing your Widget to do other things in the meantime. You can also take an animation (or multiple animations) and run them all synchronously, meaning the call will block until all the animations are complete.

These facilities take all of the hard work out of doing pretty interesting animations. They also provide "ease" functions for you to use to get the standard animation technique of easing, where an object's speed can ramp up or down at the start or end of the animation to give a better feeling of realism to the movement.

Each animation type has a done function that can be called to let you know when the animation is complete. If your Widget's minimum platform version is set to 4.0 or newer, this function is called for both synchronous and asynchronous animations. In prior releases, it was only called when running an animation asynchronously. You can use the done function to chain animations together, starting a new animation when an older one is ending.

The MoveAnimation, FadeAnimation, ResizeAnimation, and RotateAnimation objects all contain an object called owner that is the object the animation is operating on. owner ensures that the object does not get garbage collected while the animation is running, but you can use this attribute to reference the target object.

Animator

the master animation object

The animator object is the core of the animation system in version 2.1 or newer. It is what you use to start animations. You can also call methods on it to help you deal with "ease" transitions.

```
animator.ease()
animator.kEaseIn, animator.kEaseOut, animator.kEaseInOut, animator.kEaseNone
animator.milliseconds
animator.runUntilDone()
animator.start()
```


animator.ease()

blends a number between two numbers for an "ease" effect

Synopsis

```
animator.ease( start, end, percent, easeType )
```

Description

This function helps you create an "ease" effect in your animations. All of the built-in move animations that have been in Widget Engine 2.0 and newer have had this effect. You can make an object speed up as it moves away or slow down as it stops to give it a more realistic feeling of movement.

To use this function, you pass the starting and ending number, along with the percentage complete as a fraction (i.e., if you are half complete, pass 0.5). The ease type is specified with one of the constants attached to the animator object: `kEaseBounce`, `kEaseNone`, `kEaseIn`, `kEaseOut`, `kEaseInOut`. See the animator.kEaseIn, animator.kEaseOut, animator.kEaseInOut, animator.kEaseNone section for what these constants mean.

Example

```
var n = animator.ease( 0, 100, .7, animator.kEaseOut );

// at this point, n is some place between 0 and 100
// depending on the ease out curve. It is not linear.
```

Availability

Available in version 2.1 or newer.

animator.kEaseIn, animator.kEaseOut, animator.kEaseInOut, animator.kEaseNone

constants to dictate the type of easing to use

Description

These constants are used when creating different animation objects as well as using the `animator.ease()` function. If you are familiar with easing, the engine currently uses a sinusoidal ease function.

`EaseIn` means that the object starts to move slowly and then speeds up as it moves.

`EaseOut` means the object starts quickly and slows down as it comes to rest.

`EaseInOut` means the object starts slowly, reaches full speed, then starts to slow down as it approaches the end of its journey.

`EaseNone` means no easing is in effect. The speed is constant from beginning to end.

Availability

Available in version 2.1 or newer.

animator.milliseconds

current animation timebase

Description

For custom animations, it is useful to get the current animation timebase to mark the start time (or just know when "now" is). This attribute of the `animator` object allows you to determine the current time.

Example

```
myAnimation.startTime = animator.milliseconds;
```

Availability

Available in version 2.1 or newer.

animator.runUntilDone()

runs animations to completion

Synopsis

```
animator.runUntilDone( object | array )
```

Description

This function is used to run an animation or animations until they are all complete. This function does not return until all of the animations specified are completely done. For this reason, this function should be used only when you are running short, finite animations. An infinite animation such as a "pulsing button" effect would mean this call would never exit, so care must be taken to ensure this does not occur.

Example

```
// crossfade
var a = new FadeAnimation( myImage1, 0, 350,
                           animator.kEaseOut );
var b = new FadeAnimation( myImage2, 255, 350,
                           animator.kEaseOut );
animator.runUntilDone( new Array( a, b ) );
// at this point both animations are complete.
```

Availability

Available in version 2.1 or newer.

animator.start()

starts asynchronous animations

Synopsis

```
animator.start( object | array )
```

Description

This function is used to run an animation or animations asynchronously. This function returns immediately—it does not wait for the animations to complete. The animations do not even start until your JavaScript code is exited and control returns back to the Widget's main event loop. This means you can start multiple animations and they start at the exact same time. You can call `start` for each one, or pass them all as an array into `start`, it doesn't matter.

Example

```
// crossfade asynchronously
var a = new FadeAnimation( myImage1, 0, 350,
                           animator.kEaseOut );
var b = new FadeAnimation( myImage2, 255, 350,
                           animator.kEaseOut );
animator.start( new Array( a, b ) );
// at this point nothing has started yet. When we leave our Javascript code, the
animations will start up at the exact same time.
```


Availability

Available in version 2.1 or newer.

animation.kill()

base class method to terminate a running animation

Synopsis

```
animation.kill()
```

Description

This function is a "base class" function that can be called on any of the animation objects below. It is used for stopping asynchronous animations that might be running. For example, if you have an animation that rotates an object indefinitely while in a certain mode, you need to stop that animation when you exit the mode. To do that, just use this function.

Example

```
var a = new CustomAnimation( 1, SpinMeRightRoundBaby );
animator.start( a );

// some time later, maybe after the user clicks a button
if ( a != undefined )
  a.kill();
```

Availability

Available in version 2.1 or newer.

CustomAnimation()

custom animation routine written in JavaScript

Synopsis

```
new CustomAnimation( interval, updateFunc [,doneFunc] );
```

Description

This is the most flexible animation object available to you, but you do need to do all the work. In general, you can do fairly interesting things by using combinations of the `fade`, `move`, and `rotate` animation objects provided below.

The first parameter is the interval your animation should start running at, in milliseconds. You can change this interval in your update function. This allows you to have an animation that changes speed, fades in or out, and so on. A good example of this is something along the lines of an animated GIF, in that each frame can have its own duration. When your update function is called the "this" object is the animation itself, so you can alter the interval as such:

```
function MyUpdate()
{
   this.interval = 5000; // switch to 5 seconds
   return true;
}
```

The next parameter is the `update` function. This is where you do the work on the animation. You can do things like move an object, change its opacity, or adjust an image's HSL settings. A custom animation runs until the `update` function returns false. So a perpetual animation always returns true, as did the code snippet above. You can always kill an animation that was perpetual by calling the `kill()` method on the animation:

```
myAnimation.kill();
```

The last, optional parameter is the `done` function. This is called when your animation is done. If you have a finite animation, it is called right after your `update` function returns false. Alternatively, do the work in the `update` function right before you return false.

Along with the interval, your custom animation has another attribute accessible to it, `startTime`. This is set automatically when your animation is added to the queue (in the case of using start) or when `runUntilDone` is called. You can query this value inside your update function to determine how much time has elapsed. The example below shows this in use.

Example

```
var x = new CustomAnimation( 1, UpdateMe );
// some custom attributes for my animation
x.duration = 350;
x.startOpacity = myObject.opacity;
x.endOpacity = 0;

function UpdateMe()
{
   var now = animator.milliseconds;
   var t = limit( now - this.startTime, 0,
                  this.duration );
   var percent = t / this.duration;

   // set the new opacity of our object based on
   // easing.
   myObject.opacity = animator.ease( this.startOpacity,
                  this.endOpacity, percent,
                  animator.kEaseOut );

   // If the duration is up, let's get out of here
   if ( animator.milliseconds >=
        (this.startTime + this.duration )
   {
        // make sure we reached the end
        myObject.opacity = this.endOpacity;
        return false; // we're done
   }
   return true; // keep going
}
```

Availability

Available in version 2.1 or newer.

FadeAnimation()

animation object to adjust the opacity of an object

Synopsis

```
new FadeAnimation( object, toOpacity, duration,
```


```
                    easeType [, doneFunc]);
```

Description

This animation object can be used to adjust the opacity of an `Image`, `Frame`, `Text`, `TextArea`, or `Window` object. This can be used to fade an object in or out. Pass the opacity that you ultimately want to reach in the `toOpacity` parameter. The duration is specified in milliseconds. You can specify the type of easing in the `easeType` parameter.

When you've created this animation object, you can pass it to `animator.start()` or `animator.runUntilDone()`.

If you pass a function for the `doneFunc` parameter, and you started your animation with `animator.start()`, when the animation is complete, the function you passed is called.

Example

```
var a = new FadeAnimation( myObject, 0, 350,
            animator.kEaseOut, FadeDone );
animator.start( a );

function FadeDone()
{
   // the fade above has finished
   print( "fade complete" );
}
```

Availability

Available in version 2.1 or newer.

MoveAnimation()

animation object to adjust the position of an object

Synopsis

```
new MoveAnimation( object, toX, toY, duration,
                   easeType [, doneFunc]);
```

Description

This animation object can be used to adjust the position of an `Image`, `Frame`, `Text`, `TextArea`, or `Window` object. This can be used to move an object on screen. It works by adjusting the `hOffset` and `vOffset` attributes of the object you pass in. Pass the `hOffset` and `vOffset` that you ultimately want the object to be. The duration is specified in milliseconds. You can specify the type of easing in the `easeType` parameter.

When you've created this animation object, you can pass it to `animator.start()` or `animator.runUntilDone()`.

If you pass a function for the `doneFunc` parameter, and you started your animation with `animator.start()`, when the animation is complete, the function you passed is called.

Example

```
var a = new MoveAnimation( myObject, 100, 100, 350,
             animator.kEaseOut, MoveDone );
animator.start( a );

function MoveDone()
{
```

```
    // the move above has finished
    print( "move complete" );
}
```

Availability

Available in version 2.1 or newer.

RotateAnimation()

animation object to adjust the rotation of an object

Synopsis

```
new RotateAnimation( image, toAngle, duration,
                     easeType [, doneFunc]);
```

Description

This animation object can be used to adjust the rotation of an `Image` object. It does not affect `Text`, `TextArea`, or `Window` objects. It works by adjusting the `rotation` attribute of the image you pass in. Pass the angle that you ultimately want the object to be when the animation is finished. The duration is specified in milliseconds. You can specify the type of easing in the `easeType` parameter.

When you've created this animation object, you can pass it to `animator.start()` or `animator.runUntilDone()`.

If you pass a function for the `doneFunc` parameter, and you started your animation with `animator.start()`, when the animation is complete, the function you passed is called.

Example

```
var a = new RotateAnimation( myObject, 180, 350,
             animator.kEaseOut, RotateDone );
animator.start( a );

function RotateDone()
{
    // the rotate above has finished
    print( "rotate complete" );
}
```

Availability

Available in version 2.1 or newer

ResizeAnimation()

animation object to adjust the size of an object

Synopsis

```
new ResizeAnimation( object, toWidth, toHeight, duration,
                     easeType [, doneFunc]);
```

Description

This animation object can be used to adjust the size of an object. It works by adjusting the width and height attributes of the objects you pass in. Pass the size that you ultimately want the object to be when the animation is complete. The duration is specified in milliseconds. You can specify the type of easing in the `easeType` parameter.

When you've created this animation object, you can pass it to `animator.start()` or `animator.runUntilDone()`.

If you pass a function for the `doneFunc` parameter, and you started your animation with `animator.start()`, when the animation is complete, the function you passed is called.

Example

```
var a = new ResizeAnimation( myObject, 200, 200, 350,
             animator.kEaseOut, ResizeDone );
animator.start( a );

function ResizeDone()
{
   // the resize above has finished
   print( "resize complete" );
}
```

Availability

Available in version 4.0 or newer.

JSON

The engine supports JSON encoding and decoding in version 4.5 or later. There are two APIs for use: `JSON.stringify` and `JSON.parse`. These are implemented using a public domain JSON object.

`JSON.stringify`
`JSON.parse`

JSON.stringify

turns a value into a JSON string

Synopsis

```
string JSON.stringify( value[,whitelist] )
```

Description

This function takes any value and turns it into a JSON string. If you pass an array as the second parameter, it should contain the names of the properties you wish to selectively output in the object you passed as the first parameter. It essentially filters the output. Any properties which are either undefined or functions will not be output if they are part of an object, and output as null if they are an array element. If an object has a `toJSON()` method, that method will be called and the result will itself be turned into an appropriate string.

Example

```
var output = JSON.stringify( array );
```

Availability

Available in version 4.5 or newer

JSON.parse

parses a JSON string into an object/array/value

Synopsis

```
value = JSON.parse( string[,filter] )
```

Description

This function parses a JSON string and returns an appropriate value. The optional filter parameter is a function which can modify the results. It is passed the key and value. It can then perform whatever action it wishes on the input. It's output is used as the value to be stored. If it returns undefined, the value will be dropped.

Example

```
var obj = JSON.parse( myJSONstring )
```

Availability

Available in version 4.5 or newer.

Display

The display object represents a display that is attached to the computer. There can be one or more displays in operation at any time. Your Widget can query the display list to help you position windows appropriately. The display object just contains two properties: `rect` and `workRect`. They are considered to be immutable, i.e. setting them will have no effect on the system. The display objects are generally valid until you receive a screenchanged event, at which point they should be considered invalid and you should requery for the latest state of things.

There are two APIs you can call to get display information: `getMainDisplay()` which returns the main display, and `getDisplays()`, which returns the entire display list (item 0 is normally the main display).

To help you position windows, you can also use the window API `window.getBestDisplay()`. This function will give you the display object that the window is intersecting the most.

Display.rect

the rectangle of the display

Description

This property represents the raw rectangle of the display. The main display always has an x, y value of 0. Other displays will have an appropriately transformed origin.

Availability

Available in version 4.5 or newer.

Display.workRect

the usable rectangle of the display

Description

This property represents the usable rectangle of the display. On Mac, this means the area of the display not accounting for the menu bar and dock. On Windows, it means the area of the display minus the area of the task bar, etc.

Availability

Available in version 4.5 or newer.

CSS Reference

Cascading Style Sheets (CSS) is a standard for Web-based display of content. CSS is typically used in conjunction with HTML to render web pages. Yahoo! has adopted CSS for use in our rendering model as of version 4.0 of the Yahoo! Widget Engine, because continuously adding style information was becoming unwieldy. Also, the inheritance attributes of CSS made it an ideal choice to help us bring a little order to things and make it simpler to stylize a Widget. CSS also works towards finally separating the visual aspects of a Widget from its structure.

There are limitations to the CSS implementation in version 4.0 and newer:

1. Style sheets are not yet supported.
2. The inherit keyword is not supported (but attributes that are supposed to be inherited do get inherited by child objects).
3. Sizes and offsets are limited to using pixel units.
4. Font-weight values can only be normal, bold, 400, and 700.

Usage

The CSS style can be set on an object in both JavaScript and XML by accessing an object's `style` attribute. In XML, you use the actual CSS names of the attributes and in JavaScript you use slightly different naming due to JavaScript naming rules. For example:

```
// XML
<text style="font-size: 10px; -kon-text-truncation:end"/>

// Javascript
myText.style.fontSize = "10px";
myText.style.KonTextTruncation = "end";
```

As each style is described below, the CSS and JS names will be listed.

CSS Colors

This section explains the different ways colors can be specified when used in CSS in the Yahoo! Widget Engine. There are four ways colors can be specified:

- Hex number
- `rgb()`
- `rgba()`
- Keywords

The hex method is probably the most familiar to anyone who has done HTML programming. It allows you to specify an RGB triplet in a hex value, preceded by a hash mark, e.g., #FF0000. You can also use a shorter version such as #F00, which is equivalent to #FF0000.

The rgb() and rgba() methods are functions which allow you to specify the red, green, and blue components as numbers ranging from 0 to 255 in decimal. The rgba() function also takes a fourth parameter which is the expression of the amount of alpha the color should have in a range from 0.0 to 1.0.

The last method is using keyword colors such as red and green. The full list of colors the engine supports is aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow.

Common Styles

styles that are common to many different objects

This section lists the styles that are common to many of the objects that support CSS styles. For example, several different objects can have `backgroundColor` styles.

Styles

background

controls the background color and image of an object

Usage

JavaScript: background

CSS/XML: background

Values

```
<background-color> || <background-image> || <background-repeat> ||
      <background-attachment> || <background-position>
```

Initial Value

See individual attributes.

Inherited

No.

Description

This is a shortcut style to allow you to set multiple attributes of an object's background at the same time.

Applies To

`Frame`, `Text`, and `TextArea` objects.

Example

Sets the background to be a color:

```
<frame name="myFrame" style="background:red"/>
```

```
myFrame.style.background = "red";
```

Sets the background to be an image:

```
<frame name="myFrame" style="background:url(http://example.com/marble.png)"/>
```

```
myFrame.style.background = "#FF0000 url(http://example.com/marble.png)";
```

Availability

Available in version 4.0 or newer.

backgroundAttachment

controls whether a background image is fixed or scrolls

Usage

JavaScript: backgroundAttachment

CSS/XML: background-attachment

Values

`scroll | fixed`

Initial Value

`scroll`

Inherited

No.

Description

If a background image is specified, this attribute specifies whether it is fixed in place (fixed) or scrolls along with the contents of the frame (scroll).

Applies To

`Frame` objects.

Example

This sets the background to contain an image that stays in place when the content is scrolled.

```
x = widget.getElementById( "foo" );
x.style.background = "red url(Sun.png)";
x.style.backgroundAttachment = "fixed";
```

Availability

Available in version 4.0 or newer.

backgroundColor

sets the background color of the object

Usage

JavaScript: backgroundColor

CSS/XML: background-color

Values

`<color> | transparent`

Initial Value

`transparent`

Inherited

No.

Description

Sets the background color of an element. You can use either a color value or the keyword `transparent`, which omits use of a background color, leaving the object transparent.

The color can be either a typical hex color (e.g., #313131) or it can use the rgb() or rgba() notation. You can also use several named colors, such as white. See the CSS Colors section for more information.

Applies To

`Frame`, `Text`, and `TextArea` objects.

Example

```
<frame name="myFrame" style="background-color:red"/>

myFrame.style.backgroundColor = "red";
```

Availability

Available in version 4.0 or newer.

See Also

The `bgColor` attribute of the Text object, and the `bgColor` attribute of the TextArea object.

backgroundImage

sets the background image of an object

Usage

JavaScript: backgroundImage

CSS/XML: background-image

Values

`<uri> | none`

Initial Value

none

Inherited

No.

Description

This attribute sets the background image of an element. When you specify an image, it is rendered on top of any background color, and the color is visible in the transparent parts of the image.

Values for this attribute are either `<uri>`, to specify the location of an image, or `none`, to ensure that no image is used.

Applies To

`Frame` objects.

Example

```
<frame style="background-image:url(sun.png)"/>

// or in Javascript:
frame.style.backgroundImage = "url(sun.png)";
```

Availability

Available in version 4.0 or newer.

backgroundPosition

specifies the initial position of a background image

Usage

JavaScript: backgroundPosition

CSS/XML: background-position

Values

```
[ [ <percentage> | <length> | left | center | right ] [ <percentage> | <length> |
      top | center | bottom ]? ] | [ [ left | center | right ] || [ top | center |
      bottom ] ]
```

Initial Value

0% 0%

Inherited

No.

Description

This attribute allows you to control the position of a background image, if one is specified using the backgroundImage attribute. You can express the position in terms of absolute values, such as 20px, or in terms of percentages. A value of 0% for a horizontal position would indicate the left edge of the image starts at the left edge of the object (in this case a frame). A value of 100% would indicate the right edge of the image would align with the right edge of the object.

You can mix the two position styles. For example, you can indicate a position of "25% 10px" to indicate the image should start 25% in from the left and 10 pixels from the top of the object's bounds. You can also use keywords such as center and right. Left is equivalent to 0% and right is equivalent to 100% for a horizontal value. Likewise, top is equivalent to 0% and bottom is equivalent to 100% for vertical values. The keyword center means 50% for both directions.

While you can use combinations of absolute values, percentages, and keywords, If you use keywords and nonkeyword values, the first term must represent the horizontal position and the second must represent the vertical. If you use keywords for both values, you can put them in either order. You may also use negative values as needed.

If you set only one position, it is taken to mean the horizontal position. The vertical position will be set to 50%.

If `backgroundRepeat` is set to repeat images, the position affects the first image and all others are tiled out from that initial image.

Applies To

`Frame` objects.

Example

```
<frame style="background-image:url(Sun.png); background-position: 50% 50%"/>

myFrame.style.backgroundImage = "url(Sun.png)";
myFrame.style.backgroundPosition = "50% 50%";
```

Availability

Available in version 4.0 or newer.

backgroundRepeat

specifies whether a background image is repeated and how

Usage

JavaScript: backgroundRepeat

CSS/XML: background-repeat

Values

`repeat | repeat-x | repeat-y | no-repeat`

Initial Value

`repeat`

Inherited

No.

Description

If a background image is specified, this attribute controls how the image is repeated, if at all. All repeating is based off the first image, which is normally placed in the top, left of the content area, but can be affected by the `backgroundPosition` attribute.

Applies To

`Frame` objects.

Example

```
<frame style="background-repeat:no-repeat"/>

myFrame.style.backgroundRepeat="no-repeat";
```


Availability

Available in version 4.0 or newer.

color

foreground color of the text element

Usage

JavaScript: color

CSS/XML: color

Values

`<color>`

Initial Value

Depends on user agent.

Inherited

Yes.

Description

This attribute describes the foreground color of an element's text content. The color can be specified using any standard CSS color format. See the CSS Colors section for more information.

Since this attribute is inherited, you can set it on a frame or window and have all elements within it that obey the color attribute follow suit. In version 4.0, only Text and TextArea objects obey the color attribute.

Applies To

`Text`, `TextArea`, `Frame`, and `Window` objects.

Example

```
<text style="color:blue"/>

myText.style.color="blue";
```

Availability

Available in version 4.0 or newer.

fontFamily

prioritized list of font family, generic family names

Usage

JavaScript: fontFamily

CSS/XML: font-family

Values

```
[[ <family-name> | <generic-family> ] [, <family-name>| <generic-family>]* ]
```

Initial Value

`Arial`

Inherited

Yes.

Description

The `fontFamily` attribute allows you to specify a comma-separated list of font families to use. You specify the list in priority order. The engine attempts to use a family from the list until it finds a match on the system. For example, if you specified "SuperFont, Helvetica," the engine first tries to use SuperFont and if it was not available, uses Helvetica.

There are two types of font families, a specific, named family, and a generic family, such as sans-serif.

Here is the list of generic families and the fonts they map to in the Yahoo! Widget Engine

```
serif (Times New Roman)
sans-serif (Arial)
cursive (Comic Sans MS)
fantasy (Futura)
monospace (Courier New)
```

Since this is an inherited attribute, you can set this attribute on a frame or a window and all text elements within that window will take on this style (unless a particular element has a style specified for it, which would override the inherited value).

Applies To

`Text`, `TextArea`, `Frame`, and `Window` objects.

Example

```
<text name="text1" data="Example Text" style="font-family:Gill, Helvetica, sans-
serif">

text1.style.fontFamily = "Gill, Helvetica, sans-serif";
```

Availability

Available in version 4.0 or newer.

fontSize

specifies the size of the font

Usage

JavaScript: fontSize

CSS/XML: font-size

Values

`<length>`

Initial Value

`12px`

Inherited

Yes.

Description

This attribute allows you to specify the size of the font to use for an object. At present, you must specify the font size in pixels. Point sizes, ems, etc., are not supported in Version 4.0.

Negative values are not allowed.

Since this is an inherited attribute, you can set this attribute on a frame or a window and all text elements within that window will take on this style (unless a particular element has a style specified for it, which would override the inherited value).

Applies To

`Text` and `TextArea` objects.

Example

```
<text style="font-size:24px"/>

myText.style.fontSize = "24px";
```

Availability

Available in version 4.0 or newer.

fontStretch

controls the compression and expansion of a font

Usage

JavaScript: fontStretch

CSS/XML: font-stretch

Values

`normal | condensed`

Initial Value

`normal`

Inherited

Yes.

Description

The fontStretch attribute selects a normal, condensed, or extended face from a font family. In Version 4.0 we only support normal and condensed. Condensed is only valid in Version 4.0 on Mac OS X. Setting it to condensed will have no effect on Windows OS.

Since this is an inherited attribute, you can set this attribute on a frame or a window and all text elements within that window will take on this style (unless a particular element has a style specified for it, which would override the inherited value).

Applies To

`Text` and `TextArea` objects.

Example

```
<text style="font-stretch:condensed"/>
```

```
myText.style.fontStretch = "condensed";
```

Availability

Available in version 4.0 or newer.

fontStyle

specifies normal, italic, or oblique style

Usage

JavaScript: fontStyle

CSS/XML: font-style

Values

`normal | italic`

Initial Value

`normal`

Inherited

Yes.

Description

The `fontStyle` attribute can be used to specify either to use normal or italic type. If you wish to have bold text, you should use the fontWeight attribute.

Since this is an inherited attribute, you can set the font style of a frame or a window and all text elements within that window takes on that style (unless a particular element has a style specified for it, which overrides the inherited value).

Applies To

`Text`, `TextArea`, `Frame`, and `Window` objects.

Example

In this example, emphasized text within `H1` appears in normal face.

```
<text style = "font-style:italic"/>

myText.style.fontStyle = "normal";
```

Availability

Available in version 4.0 or newer.

fontWeight

selects the weight of the font

Usage

JavaScript: fontWeight

CSS/XML: font-weight

Values

```
normal | bold | 400 | 700
```

Initial Value

```
normal
```

Inherited

Yes.

Description

You can use the fontWeight attribute to specify a bold font. In Version 4.0, the engine only supports normal and bold. 400 and 700 are synonyms for those two keywords.

Since this is an inherited attribute, you can set the font weight of a frame or a window and all text elements within that window will take on that value (unless a particular element has a style specified for it, which would override the inherited value).

Applies To

`Text`, `TextArea`, `Frame`, and `Window` objects.

Example

```
<text style="font-size:12px; font-weight:bold"/>

myText.style.fontSize="12px";
myText.style.fontWeight = "bold";
```

Availability

Available in version 4.0 or newer.

opacity

controls the opacity of the object

Usage

JavaScript: opacity

CSS/XML: opacity

Values

```
<alphavalue> | inherit
```

Initial Value

1

Inherited

No.

Description

The <alphavalue> specifies the opacity of the object. The number can range from 0.0 (fully transparent) to 1.0 (fully opaque).

Applies To

`Frame`, `Text`, `TextArea`, `ScrollBar`, `Image`, and `Canvas` objects.

Example

```
<text style="opacity:0.5"/>

myText.style.opacity = 0.5;
```

Availability

Available in version 4.0 or newer.

textAlign

specifies how text is aligned

Usage

JavaScript: textAlign

CSS/XML: text-align

Values

```
left | right | center | justify | inherit
```

Initial Value

The default is the script direction of the system.

Inherited

Yes.

Description

This attribute specifies how a block of text is aligned.

In Version 4.0, justify only works on the Mac platform.

Applies To

`Frame`, `Text`, and `TextArea` objects.

Example

```
<text style="text-align:center"/>

myText.style.textAlign = "center";
```

Availability

Available in version 4.0 or newer.

textDecoration

decorations added to the text of an object using the object's color

Usage

JavaScript: textDecoration

CSS/XML: text-decoration

Values

```
none | [ underline || line-through ]
```


Initial Value

none

Inherited

No.

Description

This attribute allows you to specify special text effects. In Version 4.0, the only values supported are underline and line-through. The underline will be drawn in whatever color the text is drawn.

Applies To

`Text` and `TextArea` objects.

Example

```
<text name="text1" data="Example Text" style="text-decoration:underline"/>

text1.style.textDecoration = "underline";
```

Availability

Available in version 4.0 or newer.

KonBackgroundFill

controls the background color of an object

Usage

JavaScript: KonBackgroundFill

CSS/XML: -kon-background-fill

Values

`stretch | tile`

Initial Value

`tile`

Inherited

No.

Description

This attribute is an extension to CSS that allows you to specify that the background is stretched instead of tiled (the default). If you set this attribute to stretch, all positioning is ignored (repeat, origin, etc.) and it remains fixed in place (i.e., it will not be scrolled with any content).

Applies To

`Frame` objects.

Example

```
<text style="-kon-background-fill: stretch"/>

myText.style.KonBackgroundFill="stretch";
```

Availability

Available in version 4.0 or newer.

KonShadow

specifies the shadow cast by an object

Usage

JavaScript: KonShadow

CSS/XML: -kon-shadow

Values

`<shadow-offset> || <shadow-color>`

Initial Value

See individual attributes.

Inherited

No.

Description

This attribute is a shortcut which allows you to set the shadow color and offset for text at the same time. See the individual attributes for more information.

Applies To

`Text` objects.

Example

```
<text style="-kon-shadow: 0px 1px #FFFFFF"/>

myText.style.KonShadow="0px 1px white";
```

Availability

Available in version 4.0 or newer.

KonShadowColor

specifies the color of the shadow cast by a text object

Usage

JavaScript: KonShadowColor

CSS/XML: -kon-shadow-color

Values

`<color>`

Initial Value

#000000

Inherited

No.

Description

This attribute controls the color of a shadow applied to text. The default shadow color is black.

Applies To

Text objects.

Example

```
<text style="-kon-shadow-color: #FFFFFF"/>

myText.style.KonShadowColor="white";
```

Availability

Available in version 4.0 or newer.

KonShadowOffset

specifies the offset of the shadow for an object

Usage

JavaScript: KonShadowOffset

CSS/XML: -kon-shadow-offset

Values

<length> <length>?

Initial Value

0px

Inherited

No.

Description

The values for shadow offset can be either one or two values. If one is specified, it applies to both the horizontal and vertical directions. If two are specified, the first value is the horizontal offset and the second is the vertical offset. In version 4.0, the only supported unit of measure is pixels.

Negative values are allowed.

Applies To

Text objects.

Example

```
<text style="-kon-shadow-offset: 2px"/>

myText.style.KonShadowOffset="0px 1px";
```

Availability

Available in version 4.0 or newer.

KonTextTruncation

specifies how text is truncated in an object

Usage

JavaScript: KonTextTruncation

CSS/XML: -kon-text-truncation

Values

`'none' | 'end' | 'middle'`

Initial Value

none

Inherited

No.

Description

This property allows you to control the manner in which text is truncated if the text is too long to fit in its assigned width. Text truncation settings only take effect if there's an imposed width on an object (i.e., the width has been explicitly set) and there is no text scrolling set for the object.

Applies To

`Text` objects.

Example

```
// XML
<text width="40" data="A longer text string than will fit" style="font-size: 10px;
-kon-text-truncation:end"/>

// Javascript
myText.data = "A longer text string than will fit"
myText.width = 40;
myText.style.fontSize = "10px";
myText.style.KonTextTruncation = "end";
```

Availability

Available in version 4.0 or newer.

SQLite Reference

Version 4.0 or newer of the Widget Engine supports the use of SQLite. You can use it as an alternative storage medium to XML or a simple text file. The support in the engine is a subset of the full SQLite functionality, but it should provide Widget authors with sufficient functionality to do most common tasks.

To use the SQLite support, you create `SQLite` objects and then interact with those objects. For example:

```
var db = new SQLite();

db.open( "myfile" );
db.exec( "CREATE TABLE test (id int(11), name varchar(255)); \
       INSERT INTO test (id, name) values( 1, \"Ed\" );" );

print( db.numRowsAffected );

db.close();
```

Along with the `SQLite` object, there is a `SQLiteResult` object that is returned from `query()` calls. You can use methods on it to extract information from the result.

```
var r = db.query( "SELECT * from test" );

while( (x = r.getRow()) != null )
   print( x['id'], x['name'] );
```

In addition to getting the data, you can also get information about the columns that were returned. See the reference below for more details.

SQLite Object

represents a database

Attributes

lastInsertRowID
numRowsAffected

Functions

open()
close()
exec()
query()

lastInsertRowID

return the last row ID from an INSERT statement

Description

This attribute allows you to get the ID of the last inserted row. You typically use this when you have a column set up as an auto-incrementing primary key. Since the database controls what the key is, you can't know what ID was assigned to a row until the row has been inserted. After inserting a row in such a table, reference this attribute to get the ID.

Example

```
db.exec( "INSERT INTO mytable (name) VALUES (\"Ed\")" );
var id = db.lastInsertRowID;
```

Availability

Available in version 4.0 or newer.

numRowsAffected

returns the number of rows affected by the last statement

Description

This attribute allows you to determine how many rows in a table were affected by the last statement executed.

Example

```
db.exec( "DELETE FROM mytable" ); // wipe it out
print( "Deleted " + db.numRowsAffected + " rows" );
```

Availability

Available in version 4.0 or newer.

open()

opens or creates a database

Synopsis

```
db.open( path );
```

Description

This function opens or creates a database file. If the path passed in does not exist, the file is created and opened as a new database. If any error occurs while opening the file, an exception is thrown.

When open, you can then use the `query()` and `exec()` calls. When you are done with the database, call the `close()` function.

If you do not want to create a file, but instead want to use an in-memory database, you can pass the special "`:memory:`" path name.

Calling `open()` on an already open database raises an exception.

Example

```
db = new SQLite();

db.open( ":memory:" ); // in-memory database
```

Availability

Available in version 4.0 or newer.

close()

closes a database

Synopsis

```
db.close();
```


Description

This function closes a database previously opened by `open()`. If the database is already closed, `close()` has no effect. After the database is closed, if you wish to reuse the database, you must call `open()` again.

If you have any `result` objects around that were created from querying the database file, and you call `close()`, it fails with an exception. You must use the `dispose()` call to ensure that any results are properly finalized before calling `close()`.

Example

```
db = new SQLite();

db.open( ":memory:" ); // in-memory database
...
db.close();
```

Availability

Available in version 4.0 or newer.

exec()

executes a statement or series of statements with no result

Synopsis

```
db.exec( statements );
```

Description

This function allows you to execute a series of statements that do not return results. The classic example of this is creating a table or inserting rows. You can have one statement or a series of them separated by semicolons.

Example

```
db.exec( "CREATE TABLE foo (id int(11), name varchar(255);" );
```

Availability

Available in version 4.0 or newer.

query()

executes a query statement and returns results

Synopsis

```
SQLiteResult db.query( statement );
```

Description

This function allows you to execute a single query statement that returns results. For example, you could use this to select rows in a table.

Example

```
r = db.query( "SELECT * FROM mytable" );
```

Availability

Available in version 4.0 or newer.

SQLiteResult

the results of a database query

Attributes

numColumns

Functions

current()
next()
rewind()
getAll()
getRow()
getColumn()
getColumnName()
dispose()

numColumns

number of columns in the current row

Description

This returns the number of columns in the current row. If the current row is empty, zero is returned.

Example

```
r = db.query( "SELECT * from mytable" );
print( r.numColumns );
```

Availability

Available in version 4.0 or newer.

current()

returns the current row result of the statement as an associative array

Synopsis

```
array result.current();
```

Description

This function returns the current row of a `result` object as an associative array. You generally use this function with the `next()` function to iterate over the rows in a `result` object. Alternatively, you can use `getRow()`, which returns the current row and automatically steps to the next row. If you are at the end of the result set, this function returns null.

Example

```
r = db.query( "SELECT * FROM mytable" );

do
{
   print( r.current()['id'] );
}
while( r.next() );
```

Availability

Available in version 4.0 or newer.

next()

skips to the next row in the result set

Synopsis

```
boolean result.next();
```

Description

This function attempts to move to the next row in the result set and returns true. If there are no more rows in the set, false is returned.

Example

```
r = db.query( "SELECT * FROM mytable" );

do
{
   print( r.current()['id'] );
}
while( r.next() );
```

Availability

Available in version 4.0 or newer.

rewind()

returns to the first row of the result set

Synopsis

```
boolean result.rewind();
```

Description

This function attempts to move to the beginning of the result set, point to the first row in the set, and return true. If there are no rows in the set, false is returned.

Upon successful completion of this function, `current()` returns the first row in the set.

Example

```
r = db.query( "SELECT * FROM mytable" );

do
{
   print( r.current()['id'] );
}
while( r.next() );

r.rewind();

print( r.current()['id'] );
```

Availability

Available in version 4.0 or newer.

getAll()

returns all rows in the set as an array of associative arrays

Synopsis

```
array result.getAll();
```

Description

This function returns all the rows in the result set as an array of associative arrays. You can use this if you need to randomly access the rows returned by a query.

If there are no rows in the result, you receive an empty array object.

Example

```
r = db.query( "SELECT * FROM mytable" );

x = r.getAll();
print( "The query returned " + x.length + " rows" );
print( "First key: " + x[0]['id'] );
```

Availability

Available in version 4.0 or newer.

getRow()

returns the current row and advances to the next

Synopsis

```
array result.getRow();
```

Description

This function returns the current row as an associative array and advances to the next row in the set. You can use it instead of `current()` and `next()` as an easy way to iterate results.

This function returns null when there are no more rows.

Example

```
var r = db.query( "SELECT * from test" );

while( (x = r.getRow()) != null )
   print( x['id'], x['name'] );
```

Availability

Available in version 4.0 or newer.

getColumn()

returns a particular column of the current row

Synopsis

```
mixed result.getColumn( index | name );
```

Description

This function returns the data of the given column index or name. Column indexes start at 0.

If a numeric index is out of bounds, or a named column does not exist, an exception is raised.

If the current row is empty because there are no more rows, null is returned.

Example

```
var r = db.query( "SELECT * from test" );

print( r.getColumn( 0 ) );
print( r.getColumn( “id” ) );
```

Availability

Available in version 4.0 or newer.

getColumnName()

returns the name of a column of the current row

Synopsis

```
string result.getColumnName( index );
```

Description

This function returns the name of the given column index. Column indexes start at 0.

If a numeric index is out of bounds, an exception is raised.

If the current row has no data, null is returned.

Example

```
var r = db.query( "SELECT * from test" );

print( r.getColumnName( 0 ) );
```

Availability

Available in version 4.0 or newer.

dispose()

disposes of the current row data

Synopsis

```
result.dispose();
```

Description

You can use this function to make sure the result of a query has been disposed of properly. You generally only need to call this if you are going to explicitly close your database file. In that case, all `result` objects generated by the file must be disposed of before the file can be successfully closed. Since it is not possible to know when a result might be garbage collected, this function allows you to be explicit.

Example

```
var r = db.query( "SELECT * from test" );

print( r.getColumn( 0 ) );
print( r.getColumn( "id" ) );

r.dipsose(); // close will fail without this
```

```
db.close();
```

Availability

Available in version 4.0 or newer.

SQLiteError

contains information about an exception

Attributes

errCode
errMsg

errCode

error code

Description

This attribute contains the SQLite error code.

Example

```
try
{
   db = new SQLite();
   db.open( “test” );
   db.open( “test” ); // this will fail
}
catch( e )
{
    print( e.errCode, e.errMsg );
}
```

Availability

Available in version 4.0 or newer.

errMsg

error message

Description

This attribute contains a textual message explaining the exception.

Example

```
try
{
   db = new SQLite();
   db.open( "test" );
   db.open( "test" ); // this will fail
}
catch( e )
{
   print( e.errCode, e.errMsg );
}
```


Availability

Available in version 4.0 or newer.

Global Functions

This section describes the extensions to JavaScript that are provided by the Widget Engine. If JavaScript is new to you, consider obtaining a guide to the language to help with its syntax and structure. The Yahoo! Widget Engine implements a JavaScript engine (Mozilla SpiderMonkey) that conforms to the JavaScript 1.5 standard (ECMA-262, revision 3).

alert()

displays an alert dialog

Synopsis

```
alert(string, [button one, button two, button three])
```

Attributes

Attribute	Description
`string`	The text contents of the alert that that will be displayed.
`button one`	The text presented on the first (or only) button shown on the alert. This argument is optional.
`button two`	The text presented on the second button of the alert. This argument is optional.
`button three`	The text presented on the third button shown on the alert. This argument is optional.

Returns

When the alert dialog is presented to the user, the dialog returns `1`, `2`, or `3` based on which button was pressed.

Description

Used to give the user an immediate message in a standard alert dialog, or to ask them to pick from up to three options. The return value can be `1`, `2`, or `3` to indicate which of three optional buttons were pressed.

Example

```
alert("The time is now " + Date());

answer = alert("Do you wish to continue?", "Yes", "No");

if (answer == 2)
   closeWidget();
```

appleScript()

executes an AppleScript

Synopsis

```
appleScript(appleScriptCode[, timeout])
```

Parameters

Parameter	Description
`appleScriptCode`	A string that contains a complete AppleScript code snippet that you want to have executed. If the string consists only of a valid file name, then the code is loaded from that file.
`timeout`	The optional number of seconds to wait for the AppleScript to complete. For compatibility reasons, the default timeout is 2 seconds.

Description

Using this function, your Widget can control an element of the system or an application using an AppleScript call.

The AppleScript must be formatted as a nonbreaking line, using new-line characters to indicate a physical break. We suggest preformatting and validating your AppleScript in Apple's Script Editor application before using it in a Widget.

The iTunes Remote Widget makes extensive use of the `appleScript()` call.

Example

```
// Note the embedded new-lines that are required
// in AppleScripts.
appleScript('tell application "Safari"\nopen location"' + newURL + '"\nend
tell\n');
```

beep()

plays the alert sound

Synopsis

```
beep()
```

Description

This function causes the user's Mac to beep. This can be useful if you need to get their attention, would like to notify them of a completed task, or for debugging your Widget's script.

Example

```
if (done)
    beep();
```

bytesToUIString()

turns a number of bytes into a UI-friendly string

Synopsis

```
string = bytesToUIString(integer)
```

Description

This function turns a given number of bytes into a string such as "1K" or "34.2M."

Example

```
print( "There is " + bytesToUIString( numBytes ) + " memory available" );
```

Availability

Available in version 2.0 or newer.

chooseColor()

puts up a standard color picker dialog box and allows the user to choose a color

Synopsis

```
string = chooseColor( [string] );
```

Description

You can use this function to display the standard color picker for the platform and allow the user to select a color. You can optionally pass the initial color that is selected as a parameter. This function returns the color as an RGB hex string (e.g., "#FF0000") or null if the user canceled the dialog.

Example

```
print( chooseColor( "#EEEEEE" ) );
```

Availability

Available in version 2.0 or newer.

chooseFile()

puts up a standard file dialog box and allows the user to choose a file

Synopsis

```
file | array = chooseFile( [string | array] [, allowMultiple] );
```

Description

You can use this function to display the standard open dialog for the platform and allow the user to select a file. You can also optionally pass a single extension or an array of extensions into this function to limit what kinds of files the user can choose. If the dialog is canceled by the user, null is returned.

The `allowMultiple` parameter is set to true, the user can choose multiple items in the dialog. In this mode, the function always returns an array, even if only one item is chosen. If the user cancels the dialog however, the result is null as in the single item case.

Example

```
print( chooseFile() ); // select anything
print( chooseFile( ".png" ) ); // just PNG files
print( chooseFile( new Array( ".png", ".jpg" ) ) )

files = chooseFile( ".png", true ); // allow multiple png files
```

Availability

Available in version 2.0 or newer.

chooseFolder()

puts up a standard file dialog box and allows the user to choose a folder

Synopsis

```
file = chooseFolder();
```

Description

You can use this function to display the standard open dialog for the platform and allow the user to select a folder. If the dialog is canceled by the user, null is returned.

Example

```
print( chooseFolder() );
```

Availability

Available in version 2.0 or newer.

convertPathToHFS()

converts a UNIX-style path to a Mac HFS path

Synopsis

`convertPathToHFS(`*`myPath[, localize]`*`)`

Description

Converts a UNIX-style path (with "/") to a Mac HFS-style path (with a volume name and ":"). If the optional second Boolean parameter is true, then the returned path is localized in the current system language. Note that the file referenced by the path must exist for conversion to succeed if the localized path is requested.

Example

```
convertPathToHFS('/Users/joe/foo.txt');
```

Yields:

```
Macintosh HD:Users:joe:foo.txt
```

On a German system:

```
convertPathToHFS('~/Movies', true)
```

Yields:

```
Macintosh HD:Benutzer:joe:Filme
```

Platform Notes

This function returns an empty string on Windows OS.

convertPathToPlatform()

converts a JavaScript-style path to a platform-specific one

Synopsis

`convertPathToPlatform(`*`myPath[, forDisplay]`*`)`

Description

Converts a JavaScript style file path (`"/foo/bar/baz"`) to a platform style path (e.g., on Windows OS, `"\\foo\\bar\\baz"`). Note that by default the path is escaped (has any backslashes doubled), ready for use with `runCommand()`. If you want a path suitable for display to a user, specify true for the optional second parameter.

On Mac OS X this function does nothing (paths are already in the correct format).

Example

```
convertPathToPlatform('c:/temp/foo.txt');
```

On Windows OS, yields:

```
c:\\temp\\foo.txt
```

and

```
convertPathToPlatform('c:/temp/foo.txt', true);
```

On Windows OS, yields:

```
c:\temp\foo.txt
```

closeWidget()

closes the Widget

Synopsis

```
closeWidget()
```

Description

Shuts down the currently running Widget as if the user had selected **Close Widget** from the context menu.

Example

```
answer = alert("Do you wish to continue?", "Yes", "No");

if (answer == 2)
   closeWidget();
```

escape()

encodes a string to safely be used as a URL

Synopsis

```
escape(string)
```

Attributes

Attribute	Description
String	A string containing text that is intended for use as a URL.

Returns

A string that contains the argument, but with characters unsuitable for URLs converted to their escaped counterparts.

Description

This is useful if you're collecting information from a user preference that you would like to pass with a URL. It saves having to validate the strings yourself before passing them off to the URL handler.

Example

```
// The single quote, spaces and ampersand will be
// replaced with URL escape characters
mySearch = "Konfabulator's FAQ & JavaScript Reference";
openURL("yahoo.com/search?q=" + escape(mySearch) );
```

See Also

unescape()

focusWidget()

brings the Widget to the foreground

Synopsis

```
focusWidget()
```


Description

Brings the Widget to the foreground on the user's desktop. Useful when responding to a hot key.

Note: If a Widget comes to the foreground when not requested, the user might become annoyed and will probably trash the Widget.

Example

```
<hotkey name="hkey1">
      <key>F2</key>
   <modifier>command+control</modifier>
  <onKeyUp>focusWidget();</onKeyUp>
</hotkey>
```

form()

generates a preference-like form for acquiring user input through a dialog

Synopsis

`form(`*`fieldArray, [dialogTitle], [confirmButtonLabel], [cancelButtonLabel]`*`)`

Description

`form()` takes up to four parameters. The first argument is an array of `FormField` objects (which have the same parameters as preference objects which are defined in the XML). This array is used to define a dialog that is displayed to the user. When the user presses the "confirm" button, the `form()` function returns an array of strings representing the values entered in the form (if the "dismiss" button is pressed, null is returned). The remaining parameters are, in order, a title for the dialog, the label for the "confirm" button, and the label for the "dismiss" button. The last three parameters are optional.

Example

```
var formfields = Array();
formfields[0].description = 'This is a description of a text field.';formfields[0] =
new FormField();
formfields[0].name = 'name1';
formfields[0].type = 'text';
formfields[0].title = 'Text Pref Title';
formfields[0].defaultValue = 20;

formfields[1] = new FormField();
formfields[1].title = 'Basic Field';

formfields[3] = new FormField();
formfields[3].name = 'name4';
formfields[3].title = Checkbox Pref Title';
formfields[3].type = 'checkbox';
formfields[3].defaultValue = 1;
formfields[3].description = 'This is a description of a checkbox field.';

formResults = form(formfields, 'my title', 'Save It And Continue');

if (formResults != null) {
    print("formResults = " + formResults);
} else {
    print("form was cancelled");
}
```

getMainDisplay()

returns the main display

Synopsis

```
Display getMainDisplay()
```

Description

This returns the main display attached to the computer. This is normally the display with the menu bar on Mac and the task bar on Windows.

Availability

Available in version 4.5 or newer.

getDisplays()

returns all displays attached to the computer

Synopsis

```
array getDisplays()
```

Description

This returns the list of displays attached to the computer as an array. The main display is usually item 0.

Availability

Available in version 4.5 or newer.

include()

includes the contents of another JavaScript file

Synopsis

`include(`*`string`*`)`

Description

Includes the contents of the specified file at the current point in the script. Using `include()` arranges for any error messages to have correct file names and line numbers.

Example

```
include("onload.js");
```

isApplicationRunning()

returns true if specified application is running

Synopsis

```
isApplicationRunning(string)
```

Description

You can use this function to decide if an application is currently running. This is often useful before you do something like invoke AppleScript on the Mac or COM on Windows OS. Pass the name of the application you are interested in, not a full path.

Example

```
// On Mac
if ( isApplicationRunning( "iTunes" ) )

// On Windows
if ( isApplicationRunning( "itunes.exe" ) )
```

Platform Notes

This is an area where you have to use the exact name for the platform. As shown above you might use `itunes.exe` on Windows, whereas on Macintosh you just use `itunes` for the application name parameter.

Availability

Available in version 2.0 or newer.

konfabulatorVersion()

returns the current version of the Widget Engine as a string

Synopsis

```
konfabulatorVersion()
```

Description

You can use this function for informational purposes, or to control how your code behaves on different versions of the Widget Engine.

Example

```
print( "This version is " + konfabulatorVersion() );
```

log()

displays a string in the debug window with a timestamp

Synopsis

```
log(string)
```

Description

Often used for debugging. Note that you need to specify:

```
<debug>on</debug>
```

in the Widget's XML to see the output.

Example

```
log("idx = " + idx);
```

openURL()

opens the specified URL in the default web browser

Synopsis

```
openURL(validURL)
```

Description

Using this function to launch a URL causes the URL to be launched using the appropriate application set in the user's Internet System Preferences. This function returns true if the argument is a well-formed URL, otherwise false is returned. Note that even a well-formed URL might point to a nonexistent resource so the Widget Engine would return true while your browser might still complain.

To open a URL and retrieve the response within your widget, use the `URL` or `XMLHttpRequest` objects.

Example

```
openURL("http://widgets.yahoo.com");
openURL("ftp://myname:pa55w0rd@ftp.mysite.com");
```

See Also

escape(), unescape(), URL.fetch()

play()

plays a sound file

Synopsis

`play(`*`pathToSound[, truncate]`*`)`

Description

Supported formats are MP3, AIFF, AU, WAV, and SND. The call returns immediately and the sound is played asynchronously. `pathToSound` must point to a valid sound file either somewhere on the user's hard drive, or inside the Widget's bundle. The optional second Boolean parameter specifies whether the new sound should truncate (stop) any currently playing sounds.

Example

```
// without Boolean
play("sounds/sample.mp3");

// with Boolean
play("sounds/bark.aiff", true);
```

popupMenu()

displays a pop-up menu at a specified location

Synopsis

```
popupMenu( menuItems, x, y );
```

Description

This function allows you to display a pop-up menu at a specified location. You pass an array of `menuItem` objects in the first parameter, much like you would for a context menu. The `x` and `y` coordinates are passed in window coordinates.

You should only call this function while handling a mouse-down event.

Example

```
// put up a popup menu where the mouse is
<onMouseDown>
  var items = new Array;
```


```
    items[0] = new MenuItem;
    items[0].title = "This is item 1";
    items[0].enabled = true;
    items[1] = new MenuItem;
    items[1].title = "this is item 2";
    items[1].enabled = true;

    popupMenu( items, system.event.hOffset,
        system.event.vOffset );
</onMouseDown>
```

Availability

Available in version 2.1 or newer.

print()

prints a string in the debug window

Synopsis

`print(`*`string`*`)`

Description

Often used for debugging. Note that you need to specify:

`<debug>on</debug>`

in the Widget's XML to see the output.

Note that you must be in debug mode. See "Debugging" for more information.

Example

```
print("idx = " + idx);
```

prompt()

provides a text entry field for user input

Synopsis

`prompt(`*`<promptText>, [defaultValue], [dialogTitle], [confirmButtonLabel], [cancelButtonLabel]`*`)`

	Description
`promptText`	Prompt to be displayed to the user.
`defaultValue`	Value to populate the text field with (and the value that will be returned if the user does not change anything).
`dialogTitle`	Title that will be used for the dialog.
`confirmButtonLabel`	Label for the button that confirms the user's changes to the dialog.
`cancelButtonLabel`	Label used for the button that cancels the dialog.

Description

Used to get a string of text back from the user. This is a subset of the functionality found in `form()`, and is provided for ease of coding. Note that null is returned if the user cancels this dialog.

Example

```
result = prompt("Name:", "Your Name","Name Dialog", "OK", "Cancel");

if (!result)
    result = "no name";
```

See Also

alert()

random()

returns a random number

Synopsis

random(*[lower_limit, upper_limit]*)

Description

This function generates a random number, optionally within given limits. Note that the lower limit can be included in the returned values while the upper limit cannot.

Example

```
// This will return a random number between 0 and 64K
number = random();
// This will return a random number between 0 and 100
percentage = random(100);
// This will return a random number between 27 and 72
number = random(27,72);
```

reloadWidget()

causes the Widget to reload itself

Synopsis

reloadWidget()

Description

Calling this function restarts the Widget. This is the same result as if the user had held down the **Command** key while choosing the Widget in the dock.

See Also

closeWidget(), focusWidget()

resolvePath()

normalizes a filesystem file path

Synopsis

resolvePath(*pathToFile*)

Description

This function can make the following changes in the provided path:

- Expand an initial tilde expression (e.g., ~/Pictures) to the correct directory (e.g., /Users/joe)

- Reduce empty components and references to the current directory (that is, the sequences "//" and "/./") to single path separators.
- In absolute paths only, resolve references to the parent directory (that is, the component "..") to the real parent directory if possible, which consults the filesystem to resolve each potential symbolic link.
- In relative paths, because symbolic links can't be resolved, references to the parent directory are left in place.
- Remove an initial component of `/private` from the path if the result still indicates an existing file or directory (checked by consulting the filesystem).
- If the path is an HFS+ alias, the file name that is the target of the alias is returned (note that this only works for the final path element, aliases embedded in paths will not be resolved and may have to be handled specially if expected).
- If the given path is "." it is expanded to the fully qualified path of the current directory.

Example

```
realPath = resolvePath(myPath);
```

resumeUpdates()

allows Widgets to visually update dynamically

Synopsis

```
resumeUpdates()
```

Description

JavaScript code can affect the layout of all the objects in the Widgets window. If the Widget is complex, it can be quite inefficient (and possibly unattractive) to have these changes appear individually. By bracketing areas of code that rearrange the visible parts of the Widget with `suppressUpdates()` and `resumeUpdates()`, the Widget author can control what the user sees.

See Also

suppressUpdates(), updateNow()

runCommand()

executes a shell command and returns the result

Synopsis

```
runCommand(string)
```

Description

This function allows any command in the UNIX layer of the operating system to be executed and the results saved in a string variable. Note that only commands that the user has privileges for can be run.

If the last character of the result is a new line, it is removed.

Example

```
str = runCommand("ls -l /");
print(str);
```

runCommandInBg()

executes a shell command in the background

Synopsis

```
runCommandInBg(string, tag)
```

Description

This takes a UNIX command and a *tag*, runs the command in the background (i.e., does not wait for it to complete) and when it does complete, causes a global action called `onRunCommandInBgComplete` to be triggered and sets the value of a variable called `tag` to the results of the command (the value of `system.event.data` is set to the name of the tag). The order in which commands finish might be unrelated to the order which they were started.

Example

```
<action trigger="onLoad">
  var yahooData;
  runCommandInBg("curl www.yahoo.com", "yahooData");
</action>

<action trigger="onRunCommandInBgComplete">
  print("onRunCommandInBgComplete for tag: " +
         system.event.data);
  print("Yahoo's home page is " +
         yahooData.length + " bytes");
</action>
```

Notes

The value of `system.event.data` changes whenever a background command finishes. This can happen in the middle of an action if you have multiple commands in the background at one time. You should save the value at the beginning of the `onRunCommandInBgComplete` action to avoid unexpected results. Also note that the tag specifies the name of the variable that will receive the data, not the variable itself.

saveAs()

displays standard Save As dialog box

Synopsis

```
string = saveAs([string | array])
```

Description

This function allows you to display the standard dialog box to allow the user to choose a destination folder and save the file in that location. The path to the folder is returned. If the user canceled the dialog box, null is returned.

This function takes an optional string or array of strings as a parameter. This parameter sets the possible extensions that can be saved, similar to `chooseFile()`.

To save a file to the selected path, use `filesystem.writeFile`.

Example

```
destination = saveAs();
if ( destination != null )
     saveFileTo( destination );

destination = saveAs( new Array( ".png", ".jpg" ) );
if ( destination != null )
     saveFileTo( destination );
```

Availability

Available in version 2.0 or newer.

savePreferences()

saves the Widget's preferences

Synopsis

```
savePreferences()
```

Description

Normally a Widget's preferences are automatically saved whenever the user edits them using the Widget Preferences panel or when the Widget exits. If a Widget is manipulating preference values in JavaScript, it can ensure they are saved to disk in a timely manner by calling this function.

showWidgetPreferences()

opens the Widget's preference panel

Synopsis

```
showWidgetPreferences()
```

Description

This opens the Widget Preferences panel just as if the user had selected **Widget Preferences** from the context menu. It is often used to provide a preferences button on the face of the Widget or to get initial preferences the first time a Widget runs.

sleep()

suspends script execution

Synopsis

```
sleep(number)
```

Description

Suspends execution of the Widget's code for the specified number of milliseconds (one thousandth of a second).

Example

```
// pause script for one second
sleep(1000);
```

speak()

speaks text

Synopsis

```
speak(string)
```

Description

This function speaks the given text in the default voice of the computer (which can be set using the Speech panel in the System Preferences).

Example

```
speak("Now there's something you don't see everyday.");
speak("Unless you're me.");
```

suppressUpdates()

makes Widgets wait to visually update

Synopsis

```
suppressUpdates()
```

Description

Suppresses screen updating until a corresponding call to `resumeUpdates()`. Alternatively, updates can be performed manually using `updateNow()`. Suppressing updates can improve performance or hide messy interim states from the Widget user.

See Also

resumeUpdates(), updateNow()

tellWidget()

sends a message to another Widget

Synopsis

```
tellWidget(nameOrPath, message);
```

Description

You can use `tellWidget` to do inter-Widget messaging. For this to work successfully, the Widget you are sending the message to must have an `onTellWidget` handler. The message is passed in `system.event.data`. It's completely up to the Widget author to decide what is an acceptable message. In its simplest form, you could send JavaScript over and `eval()` it. That is not very safe however, because you have no idea what the JavaScript in question might do. So Widget authors might want to consider a special set of terms that they support using messaging like this. For example, a webcam might support the "reload" action.

```
<action trigger="onTellWidget">
   if ( system.event.data == "reload" )
      reloadCamPicture();
</action>
```

In our PIM Overview Widget, we settled on the following structure:

```
msg = action ":" params
params = (param) (";" param)*
param = name "=" value
```

`action`, `name`, and `value` are strings. Value could be placed in quotes, perhaps.

This is implemented in AppleScript on Mac OS X and COM on Windows OS. This means you could write scripts in AppleScript on the Mac, or on Windows, you could use JavaScript, VB, etc., to send messages to a Widget.

Availability

Available in version 2.0 or newer.

Notes

You can send a message to either the name of the Widget (as long as it is either running, or lives in the user's Widgets folder), or the path to the Widget. At present, this is a one-way message. Later versions will allow a response to be sent back.

Platform Notes

Currently, Windows OS will launch the Widget if it can find it. Mac will only launch the Widget if it's not running and you give it a full path. In the future there will instead be a Boolean parameter to control this more exactly.

Security Notes

You should always double-check the input message and never `eval()` the message.

unescape()

unencodes a string that contains URL escapes

Synopsis

```
unescape(string)
```

Description

This is the inverse of `escape()`.

Example

```
encURL = escape(url);

url = unescape(encURL);
```

updateNow()

forces a Widget's visual update

Synopsis

```
updateNow()
```

Description

By using `suppressUpdates()` and calling `updateNow()` as needed, the Widget author can completely control how their Widget is displayed. Note that if your code fails to call `updateNow()` when updates are suppressed, the screen might not reflect the true state of the Widget.

Example

```
updateNow();
```

See Also

resumeUpdates(), suppressUpdates()

yahooCheckLogin()

verifies whether a Widget is currently logged in

Synopsis

```
boolean yahooCheckLogin()
```

Description

This function is used to see whether the user is currently logged in to their Yahoo! account. If the function returns true, they are, and if it returns false, well guess what: they're not. You can use this to predicate whether or not you can use Yahoo! APIs, which require a logged in user.

Example

```
var loggedIn = yahooCheckLogin();
```

See Also

yahooLogin(), yahooLogout()

Availability

Available in version 3.0 or newer.

yahooLogin()

ensures a user is logged in, authenticating if necessary

Synopsis

```
boolean yahooLogin()
```

Description

This function is used to log in to a user's Yahoo! account if you are using Web APIs that require a logged in user. If `yahooCheckLogin()` returns false, you would normally call this function. It presents the standard Yahoo! Widget Engine login dialog to prompt the user for their user name and password.

This call generally works asynchronously. If the user is already logged in, `yahooLogin()` simply returns true and you are done. If `yahooLogin()` returns false, then the user needs to authenticate. This happens automatically while your Widget is free to do other things. When the user finally authenticates, you receive notification through the `onYahooLoginChanged` action. In there you can call `yahooCheckLogin()` to see if the user is now logged in.

In general, you should always wait for the `onYahooLoginChanged` event if you are not logged in when your Widget starts up before trying to do anything with APIs that require a logged-in user.

Example

```
var loggedIn = yahooCheckLogin();

if ( !loggedIn )
     yahooLogin();

// go about our business while the user authenticates.

// In your XML:
<action trigger="onYahooLoginChanged">
     if ( yahooCheckLogin() )
          RefreshInformation();
     else
          LoggedOut();
</action>
```

Notes

In our `onYahooLoginChanged` action we also have to deal with the case where we've logged out. When this happens, you might need to clear your display and present a way for the user to log in again. You might get logged out at unexpected times, so you must be prepared to deal with this.

Availability

Available in version 3.0 or newer.

See Also

yahooCheckLogin(), yahooLogout()v

yahooLogout()

logs out of a user's Yahoo! account

Synopsis

```
yahooLogout()
```

Description

This function requests that the Widget Engine log out of a user's Yahoo! account. This returns immediately while the request is pending. On completion, all Widgets receive an `onYahooLoginChanged` action and `yahooCheckLogin` returns false. Widgets must be prepared to deal with the situation where the user has logged out. This might happen if the previously valid credentials have timed out, so always be prepared to deal with a logout. Also, keep in mind this call affects all Widgets that require the user's Yahoo! credentials to use Yahoo! APIs, not just the current Widget.

Example

```
yahooLogout();

// go about our business while the logout occurs.

// In your XML:
<action trigger="onYahooLoginChanged">
     if ( yahooCheckLogin() )
          RefreshInformation();
     else
          LoggedOut();
</action>
```

Availability

Available in version 3.0 or newer.

See Also

yahooCheckLogin(), yahooLogin()

XML Services

About XML Services

The Widget Engine provides several mechanisms for dealing with XML. With these services, you can create, parse, and manipulate XML trees. You can also use the built-in implementation of `XMLHttpRequest`, a pseudo-standard for fetching XML off of web servers.

The parsing and creating of XML documents is done with the global XML object. From there, you can use standard W3C Level 1 DOM APIs to manipulate the XML tree. To make it even easier to extract data from a tree, we provide an XPath 1.0 implementation using the `node.evaluate()` addition.

XMLDOM Object

The `XMLDOM` global object allows you to parse and create XML documents. Note that per the Level 1 DOM API you cannot create `DOMNode` entities using `new`. You must use the `XMLDOM` object to create a document and the document itself to create elements to attach to the document (i.e., the `DOMDocument` is the factory for all elements, text items, comments, etc.).

XMLDOM.createDocument()

creates a new, empty DOMDocument

Synopsis

```
doc = XMLDOM.createDocument();
```

Description

This function allows you to create a new `DOMDocument` element. From there you can use the DOMDocument API to create elements to add to the document, as specified in the W3C Level 1 DOM specification.

Example

```
doc = XMLDOM.createDocument();
root = doc.createElement( "root" );
doc.appendChild( root );
print( doc.toXML() );
```

Availability

Available in version 3.0 or newer.

XMLDOM.parse()

parses XML and yields a DOMDocument

Synopsis

```
doc = XMLDOM.parse(xml);
```

Description

This parse function parses the given XML string (gotten either from a web server or using a call such as `filesystem.readFile()`) and returns a `DOMDocument` node. The document node is a W3C Level 1 DOMDocument and conforms to the API as specified by the W3C (modulo some omissions such as entity objects).

If the XML fails to parse, an exception is thrown containing the error string. You should always call `XML.parse` inside a `try/catch` block to deal with failures.

Example

```
try
{
  doc = XMLDOM.parse( xmlStream );
  root = doc.documentElement;
  ...
}
catch( e )
{
  print( e );
}
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest

fetches an item off of an HTTP server

XML Name

Not available.

JavaScript Name

`XMLHttpRequest`

Description

The `XMLHttpRequest` object is very much like the `URL` object that has existed in the Widget Engine since very early on. `XMLHttpRequest` is, however, a de facto standard for doing XML over HTTP in web browsers, so its addition here is to provide people with an easier migration path when moving AJAX code over to the Widget Engine, as well as simply trying to adhere to standards so developers find it more approachable.

Attributes

`XMLHttpRequest.autoRedirect`
`XMLHttpRequest.onreadystatechange`
`XMLHttpRequest.readyState`
`XMLHttpRequest.responseText`
`XMLHttpRequest.responseXML`
`XMLHttpRequest.status`
`XMLHttpRequest.statusText`
`XMLHttpRequest.timeout`

Functions

`XMLHttpRequest.abort()`
`XMLHttpRequest.getAllResponseHeaders()`
`XMLHttpRequest.getResponseHeader()`
`XMLHttpRequest.open()`
`XMLHttpRequest.send()`
`XMLHttpRequest.setRequestHeader()`

XMLHttpRequest.autoRedirect

controls whether redirection is handled automatically

Synopsis

```
XMLHttpRequest.autoRedirect
```

Description

This attribute allows you to control whether a request handles redirects automatically.

In versions prior to 4.0, redirection was disabled. Widgets were required to handle the 302 status themselves and do a manual redirect. Version 4.0 or newer adds automatic redirection as well as automatic cookie handling to simplify web connections that use redirected authentication. To enable this automatic behavior, you must set your `minimumVersion` to 4.0 in your Widget. This allows older Widgets to operate as they always did. If you need to manually handle a redirect, you can set this attribute to false.

Example

```
var request = new XMLHttpRequest();
request.autoRedirect = false;
request.open( "GET", "http://www.yahoo.com", true );
request.send();
```

Availability

Available in version 4.0 or newer.

XMLHttpRequest.onreadystatechange

function to call as an async request is processed

Synopsis

```
XMLHttpRequest.onreadystatechange
```

Description

If a request is sent asynchronously (see XMLHttpRequest.open() you must specify a function to be called as the status of the request changes. No parameters are passed to this function. When your function is called, "this" refers to the request. Generally, you'll only care when the `readyState` of your request is the value 4 (complete).

Example

```
var request = new XMLHttpRequest();
request.onreadystatechange = myStatusProc;
request.open( "GET", "http://www.yahoo.com", true );
request.send();

// someplace else
function myStatusProc()
{
   if ( this.readyState == 4 ) // complete
   {
     print( this.status );
   }
}
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.readyState

current state of the request

Synopsis

```
XMLHttpRequest.readyState (read-only)
```

Description

This is used to determine the current state of the request. This is typically only used when sending an asynchronous request in your `onreadystatechange` function.

The values for `readyState` are:

0	uninitialized
1	loading
2	loaded
3	interactive
4	complete

The Widget Engine only sets the `readyState` to 0, 1, or 4 in version 3.0.

Example

```
var request = new XMLHttpRequest();
request.onreadystatechange = myStatusProc;
request.open( "GET", "http://www.yahoo.com", true );
request.send();

// someplace else
function myStatusProc()
{
   if ( this.readyState == 4 ) // complete
   {
      print( this.status );
   }
}
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.responseText

text returned by the request

Synopsis

```
XMLHttpRequest.responseText (read-only)
```

Description

This attribute contains the text returned by the web server for the request you sent. Typically this is a web page or XML.

Example

```
var request = new XMLHttpRequest();
request.open( "GET", "http://www.yahoo.com", false );
request.send();
if ( request.status == 200 )
   print( request.responseText );
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.responseXML

XML DOM returned by the request

Synopsis

```
XMLHttpRequest.responseXML (read-only)
```

Description

If the response to the request returned data with a content type of `text/xml`, this attribute contains the `DOMDocument` node representing the XML document (i.e., it is automatically parsed and ready for use). If the document cannot be parsed, or the content type is not `text/xml`, this attribute is set to null.

Example

```
var request = new XMLHttpRequest();
request.open( "GET", "http://www.yahoo.com", false );
request.send();
if ( request.status == 200 )
   print( request.responseXML.toXML() );
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.status

returns the status of the response

Synopsis

```
XMLHttpRequest.status (read-only)
```

Description

This attribute represents the HTTP status code returned by the server, e.g., 200, 404.

Example

```
var request = new XMLHttpRequest();
request.open( "POST", "http://www.yahoo.com", false );
request.setRequestHeader( "Content-type", "text/xml" );
request.send( xml );
if ( request.status == 200 ) // success!
   DoSomethingWonderful();
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.statusText

returns the status text of the response

Synopsis

```
XMLHttpRequest.statusText (read-only)
```

Description

This attribute represents the HTTP status text returned by the server, e.g., "OK," "Not Found." These exactly correspond to the codes returned through status. Normally, you use `status` and not `statusText`.

Example

```
var request = new XMLHttpRequest();
request.open( "POST", "http://www.yahoo.com", false );
request.setRequestHeader( "Content-type", "text/xml" );
request.send( xml );
if ( request.statusText == "OK" ) // success!
   DoSomethingWonderful();
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.timeout

length of time to wait for a response from the server

Synopsis

```
XMLHttpRequest.timeout
```

Description

This attribute sets the time to wait for a server response. The default timeout for a request is 60 seconds. You can only set `timeout` for an integral number of seconds. This attribute can only be set while the request is in the open state. That is, after calling `open()` but before calling `send()`. Otherwise an exception is generated.

Example

```
var req = new XMLHttpRequest();

req.open( "http://www.yahoo.com"; , true )

req.timeout = 120; // two minutes

req.send();
```

Availability

Available in version 4.5 or newer.

XMLHttpRequest.abort()

aborts an async request

Synopsis

```
XMLHttpRequest.abort()
```


Description

If true was passed for the `async` parameter of `open()`, this call can be used to terminate the request if it is still outstanding.

Example

```
request.abort();
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.getAllResponseHeaders()

returns all the headers from a response

Synopsis

```
array XMLHttpRequest.getAllResponseHeaders()
```

Description

After a request is complete, this call can be used to retrieve all the headers returned with the response as an array of strings.

Example

```
var headers = request.getAllResponseHeaders();
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.getResponseHeader()

returns one or more headers from a response by name

Synopsis

```
string|array XMLHttpRequest.getReponseHeader(string)
```

Description

After a request is complete, this call can be used to retrieve one or more headers with the given name. If there is only one header, it returns a single string result. If there are multiple, it returns an array of matches.

Example

```
var cookies = request.getResponseHeader( "Set-Cookie" );
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.open()

sets up a request for sending

Synopsis

```
XMLHttpRequest.open(method,url,async);
```

Description

This sets up a request for sending. You pass the method, the URL, and a flag indicating whether you wish to send this request asynchronously. Note that at present, we do not support the traditional user name and password parameters. They might be supported in a later release.

Valid values for the method parameter are `GET`, `POST`, `HEAD`, `OPTIONS`, `PUT`, and `DELETE`.

Example

```
var request = new XMLHttpRequest();
request.onreadystatechange = myStatusProc;
request.open( "GET", "http://www.yahoo.com", true );
request.send();
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.send()

sends the request to the server

Synopsis

```
XMLHttpRequest.send([body])
```

Description

This function sends the data to the server. You can optionally pass data to be passed as the body of the HTTP request into this function.

Example

```
var request = new XMLHttpRequest();
request.open( "POST", "http://www.yahoo.com", false );
request.send( someXML );
```

Availability

Available in version 3.0 or newer.

XMLHttpRequest.setRequestHeader()

sets a request header

Synopsis

```
XMLHttpRequest.setRequestHeader(name,value)
```

Description

This function adds a header to a request, potentially replacing any existing header with the same name.

Example

```
var request = new XMLHttpRequest();
request.open( "POST", "http://www.yahoo.com", false );
request.setRequestHeader( "Content-type", "text/xml" );
request.send( xml );
```

Availability

Available in version 3.0 or newer.

XPath Support

brief overview of XPath

Starting in version 3.0, the Widget Engine supports the XPath 1.0 language for extracting nodes and node information from an XML tree. XPath is far easier and more straightforward, so it's unlikely that you will use the raw DOM API to extract node information. This section explains how XPath is integrated into the Widget Engine and how you access it. It also demonstrates some examples of how it can be used.

This is only a brief explanation. For full documentation, consult the w3c.org web site on XPath 1.0.

First, let's define an example XML tree:

```
<?xml version="1.0" encoding="utf-8"?>
<my-data>
  <element1 name="fred" size="200">
     This is some text
  </element1>
  <image-list>
     <image size="32"
         src="http://www.yahoo.com/image.png"/>
     <image size="48"
         src="http://www.yahoo.com/image2.png"/>
  </image-list>
</my-data>
```

Now let's try to do some specific things. Normally, XPath returns a list of nodes. The Widget Engine returns those nodes as a `DOMNodeList`. XPath also starts at a "context node," i.e., where the XPath search should be relative to. You can also specify a search to start at the top by starting the path with a "/". Let's say you have the document node in a variable called doc:

```
element = doc.evaluate( "my-data/element1" );
```

The above example fetches all nodes that match the path "my-data/element1" (in this case, one node) and returns them as a node list.

```
images = doc.evaluate( "my-data/image-list/image" );
```

The above statement returns all the nodes that match the given path. This time, we'll get a node list back with two elements (the two child nodes of image-list).

This is XPath at its simplest—just selecting nodes out of a document. Now we'll get a little fancier. Let's say we want to select the image that has a size of 48. We can use a predicate for this. A predicate is a condition applied to the search. It filters the results to those that match the condition.

```
image = doc.evaluate( "my-data/image-list/image[@size='48']" );
```

This says "find me all items that match this path, but only the ones whose `size` attribute has the value `48`." The @ symbol is shorthand for specifying that you are looking for an attribute. The longhand for the predicate would be `[attribute::name='48']`.

Now, if you wanted to get the `src` attribute for that image, do this:

```
src = image.item(0).getAttribute( src );
```

But there's an easier way as we'll see in a second. First, let's extract some text. Let's say we want the text inside `element1`. Here's one way to do this:

```
element = doc.evaluate( "my-data/element1" );
text = element.item(0).firstChild.data;
```

We need to know that `element` is a node list, so we extract item 0 and then ask it for its first child (since the text is a subnode of `element1` as far as the XML tree goes). Then we get its data. Well, that's cute, but it's somewhat complicated. Fortunately, XPath has functions you can call to make life easier. We'll use the `string()` function:

```
text = doc.evaluate( "string(my-data/element1)" );
```

That's it. The `string()` function takes the result of the expression passed to it and turns it into a string. For element nodes, it takes all the text subelements under it, concatenates them, and returns them. In our case, we only had one element and one text node, so we got the exact result we wanted. For attribute nodes, `string()` returns the value of the attribute. Now we can revisit our attempt above to get the `src` attribute of the image with size 48:

```
src = doc.evaluate( "string(my-data/image-list/image[@size='48']/attribute::src)"
);
```

This time, we used the same basic path as before, but added another path segment to extract the `src` attribute from the result and then the string function returned the value of that attribute.

There are various things you can search for in the XML using XPath, and many ways to search. You can find elements with certain parents, you can fetch elements who have a particular subelement, and so on. Consult the full XPath 1.0 specification on the w3c.org web site for more information.

The Converter Tool

Version 3.1 and newer of the engine supports a new flat-file format for Widgets. To generate this format a "Converter" command-line tool was created. This tool is used by the "Widget Converter" utility, but you can also use the tool itself on the command line to do automated Widget builds with no UI.

The tool converts between bundle format and flat-format. It can also be used to digitally sign a Widget. To sign and be fully validated, you must have a Netscape Code Signing certificate from VeriSign. Other root certificate authorities will be supported in the future. If you sign a Widget with another type of certificate, the integrity can be verified, but the engine can't validate the authenticity of the signature (i.e., we won't be able to tell for sure that you are you). This type of information is shown to the user when your Widget is run for the first time (and if the Widget somehow becomes modified).

Following is the syntax for the converter tool on Mac OS X:

```
converter [-v] [-list] [-flat | -unflat] [-sign -key host.key -cert
          certfile [-p foo | -pf passfile]] [-verify] file-or-bundle
          [-o output-dir]
```

Following is the syntax for version 1.1 of the converter tool on Windows OS:

```
converter [-v] [-list]
          [-flat | -unflat]
          [-sign [-cert certfile [-p foo | -pf passfile]] | -certui]
          [-verify]
          file-or-bundle
          [-o output-dir]
```

Prior to version 4.0, both Mac OS and Windows OS used the same syntax, but 4.0 now uses the built-in cryptography of the OS to save on bulk, so the syntax is different. For Windows, you specify the certificate in a .pfx or .p12 file. This file will normally also have the private key inside it. Since the system's crypto system is being used, you can also use the -certui option instead of passing a certificate and you can choose a signing certificate from your personal certificate store using the standard Windows UI.

Operations

`-list`

Lists the contents of a flattened Widget.

`-flat`

Flattens a Widget bundle.

`-unflat`

Unflattens a Widget bundle.

Signing Options

`-sign [sign-params]`

Signs a Widget after flattening it (if `-flat` is specified) or signs an already flattened Widget.

`-verify`

Verifies the signature of a Widget.

Signing Parameters

`-cert`

The certificate file (`.pem` format) to sign with.

`-certui`

Use the standard MS Windows certificate UI to choose a certificate (Windows XP and newer only).

`-key`

The private key to sign with (`.pem` format). It must be the private key that corresponds. Not supported on MS Windows as of version 1.1 of the converter tool. The key should instead be stored in your .p12 or .pfx file.

`-p | -pf [filename]`

You can specify the password to use for a password-protected private key file directly on the command line with `-p`. You can instead use `-pf` if you want to specify the password in a file. This is useful for automating the build process for a Widget without putting the password into source control.

Options

`-v`

Verbose mode. Prints the tool version and status along the way.

`-o [output-dir]`

Specifies the output directory to put the converted/signed file.

Windows OS and Mac OS X Differences

This section points out the differences between the Mac OS X and Windows OS versions of the Widget Engine.

UNIX Commands

Yahoo! Widget Engine's roots are in the first version of Konfabulator for Mac OS X. Because the Widgets were originally written with a BSD environment under them, they could make use of various UNIX system commands. When the product was ported to Windows OS, it was decided to retain a subset of commands to enable the same Widget to function on both Mac and Windows.

Over time, it was discovered that not that many Widgets took advantage of these utilities, or at best only took advantage of a small number of them. Also, the utilities took up 4.7MB of hard disk space. As a result, they are no longer a part of the engine's base install package in version 4.0 and newer. However, they are still available as a separate download.

In general, these utilities are considered to be deprecated and instead you are encouraged to use built-in functions of the engine, or to use the native commands in the OS. Another alternative is to just package one of the utilities in your Widget bundle itself.

Following is a list of the utilities in the package:

basename	bc	bunzip2	bzip2	bzip2recover
cal	cat	cksum	cmp	Comm.
compress	cp	curl	cut	Date
dc	dd	df	diff3	Diff
dirname	du	echo	egrep	Env
expand	expr	fgrep	find	Fmt
fold	fsplit	gawk	grep	Gunzip
gzip	head	id	join	Less
lesskey	ln	logname	ls	m4
md5sum	mkdir	mv	od	Open
paste	patch	pr	printenv	Pwd
rm	rmdir	sdiff	sed	Shar
sleep	sort	split	sum	Sync
tail	tar	tee	touch	Tr
uname	unexpand	uniq	unzip	Uudecode
uuencode	wc	which	whoami	Xargs
yes	zcat	zip		

In addition, `zsh` is also available to run shell scripts.

Command Key

In Windows OS, the **Control** key takes the place of the Mac's **Command** key. For example, to drag a Widget, use **Control**-drag. This also affects hot keys, as described below.

Key Names

The naming of keys comes from the original Mac key naming, and we cannot change it. So, when you press the **Delete** key on a Windows OS keyboard, you receive `ForwardDelete`. If you press **Backspace**, you receive `Delete`. On Mac keyboards, **Return** and **Enter** are two distinct keys. However, on most Windows OS keyboards, there is an **Enter** key but no **Return** key. We currently return `Return` for the **Enter** key on Windows OS. This is an important platform distinction to be aware of.

Hot Keys

If you install a hot key that was cmd-control-<key> it will just be control-<key> on Windows OS (since there's no **Command** key).

When registering a hot key on Windows OS, the key is exclusive, which is different from Mac OS. So multiple Widgets cannot register for the same hot key. The second one to try is denied the hot key. Unfortunately, we don't have any way for the Widget to know this at present.

F1 typically shouldn't be used, as it is the **Help** key. **F12** is reserved on 2000/XP and latest NT variants for the debugger.

There is no notification on release of the key. Only press, so anyone's `onKeyUp` handler never gets fired on Windows OS.

Certain hot key sequences are illegal, such as alt-tab and ctrl-alt-delete.

Paths

The native path system of Mac OS X is UNIX-style, or forward-slashed paths. Windows OS has a drive letter and backslashes. The Widget Engine considers its native path style to be forward-slashed paths. This is because of Mac and JavaScript heritage, which is forward-slashed.

If you use `runCommand` to call UNIX or Windows OS functions, you need to take care to convert paths appropriately. The UNIX commands can accept forward- or backward-slashed paths. Windows OS commands, however, must get backward-slashed paths. To convert between the two, you should use the `convertPathToPlatform` function. However, there is currently no function to convert from Windows OS style to UNIX/JavaScript style.

`convertPathToHFS` returns an empty string on Windows OS.

Perl and PHP

There is no Perl or PHP support in the Widget Engine for Windows OS. The size is prohibitive to be a standard part of our install, so we recommend that those Widgets that demand Perl point users to an appropriate Perl environment for the PC.

Appendix A: DTDs

A Document Type Definition (DTD) defines the formats of XML files. A DTD can be declared in your XML document, or as an external reference. You can use a DTD with an editor such as Oxygen to help validate your XML documents.

Metadata DTD

```
<!--
   Widget Metadata DTD
   Version 4.0
   Copyright (c) 2007 Yahoo! Inc.  All Rights Reserved.

   This DTD module is identified by the PUBLIC and SYSTEM identifiers:
   PUBLIC "-//YAHOO//DTD Widget Metadata 4.0//EN"
   SYSTEM "http://widgets.yahoo.com/engine/DTD/widget-meta-4.0.dtd"
-->
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT metadata
(name|version|identifier|image|author|copyright|description|platform|security)*>

<!-- name -->
<!ELEMENT name (#PCDATA)>

<!-- version -->
<!ELEMENT version (#PCDATA)>

<!-- identifier -->
<!ELEMENT identifier (#PCDATA)>

<!-- image-->
<!ELEMENT image EMPTY>
<!ATTLIST image
    usage (dock|security) #REQUIRED
    src CDATA #REQUIRED
>

<!-- author -->
<!ELEMENT author EMPTY>
<!ATTLIST author
    name CDATA #IMPLIED
    organization CDATA #IMPLIED
    href CDATA #IMPLIED
>

<!-- copyright -->
<!ELEMENT copyright (#PCDATA)>

<!-- description -->
<!ELEMENT description (#PCDATA)>

<!-- platform -->
<!ELEMENT platform (#PCDATA)>
<!ATTLIST platform
    minVersion CDATA #IMPLIED
```

```
    os CDATA #IMPLIED
>

<!-- security -->
<!ELEMENT security (api)*>
<!ELEMENT api (#PCDATA)>
```

Widget Dock Item DTD

```
<!--
   Widget Dock Item DTD
   Version 4.0
   Copyright (c) 2007 Yahoo! Inc.  All Rights Reserved.

   This DTD module is identified by the PUBLIC and SYSTEM identifiers:
   PUBLIC "-//YAHOO//DTD Widget Dock Item 4.0//EN"
   SYSTEM "http://widgets.yahoo.com/engine/DTD/widget-dock-4.0.dtd"
-->

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT dock-item (frame|text|image)*>
<!ATTLIST dock-item
    version CDATA #REQUIRED
    transparent (true|false) #IMPLIED
>

<!-- frame -->
<!ELEMENT frame (text|image|frame)*>
<!ATTLIST frame
    hOffset CDATA #REQUIRED
    vOffset CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    hAlign (left|center|right) #IMPLIED
    vAlign (top|center|bottom) #IMPLIED
    visible CDATA #IMPLIED
    rotation CDATA #IMPLIED
    hRegistrationPoint CDATA #IMPLIED
    vRegistrationPoint CDATA #IMPLIED
 >

<!-- image -->
<!ELEMENT image EMPTY>
<!ATTLIST image
    hOffset CDATA #REQUIRED
    vOffset CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    hAlign (left|center|right) #IMPLIED
    vAlign (top|center|bottom) #IMPLIED
    visible CDATA #IMPLIED
    rotation CDATA #IMPLIED
    hRegistrationPoint CDATA #IMPLIED
    vRegistrationPoint CDATA #IMPLIED
    colorize CDATA #IMPLIED
    hslTinting CDATA #IMPLIED
```


```
    hslAdjustment CDATA #IMPLIED
 >

<!-- text -->
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
    hOffset CDATA #REQUIRED
    vOffset CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    hAlign (left|center|right) #IMPLIED
    vAlign (top|center|bottom) #IMPLIED
    visible CDATA #IMPLIED
    rotation CDATA #IMPLIED
    hRegistrationPoint CDATA #IMPLIED
    vRegistrationPoint CDATA #IMPLIED
    truncation (end|middle|none) #IMPLIED
 >
```

Acknowledgments

This software contains the Cairo graphics software, version 1.2.6. For the various copyright notices and attributions, see

http://cairographics.org.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z