JavaScript

Overview

Joomla! 1.5 comes with the MooTools 1.11 Javascript Framework. MooTools is not only a visual effects library–it also support Ajax request and JSON notation, table sort, drag & drop operations and much more. All current HUBzero JavaScripts are built on this framework.

Directory & Files

The MooTools framework can be found within the /media/system/js directory. Joomla! includes both a compressed version used for production and an uncompressed version used for debug mode and developer reference.

```
/hubzero
  /media
    /system
      /js
        mootools-uncompressed.js
        mootools.js
```

Most HUBzero templates will include some scripts of their own for basic setup, visual effects, etc. These are generally stored in (but not limited to) a sub-directory, named /js, of the template's main directory.

```
/hubzero
  /templates
    /{TemplateName}
      /js
        globals.js
        hub.js
        modal.js
        tooltips.js
        ...
```

Of the scripts commonly found in a HUBzero template, hub.js and globals.js are perhaps the most important and it is strongly encouraged that developers include these files in their template.

hub.js

If a template includes hub.js, it will be auto-loaded by the system (thus, no reason to specifically declare it in your layout file). When loaded successfully, it will check for the inclusion of the MooTools framework and its version. Should everything pass, the script will then load any other scripts you declare in a comma-separated string.

```
var HUBzero = {
  Version: '1.1',
  require: function(libraryName) {
    // inserting via DOM fails in Safari 2.0, so brute force approach
    document.write('<script type="text/javascript" src="'+libraryName+
'"></script>');
  },
  load: function() {
    if((typeof MooTools=='undefined') ||
      parseFloat(MooTools.version)
      throw("This HUB requires the MooTools JavaScript framework >= 1.
1.0");

    $A(document.getElementsByTagName("script")).each( function(s) {
      if (s.src && s.src.match(/hub.js(?.*)?$/)) {
   var path = s.src.replace(/hub.js(?.*)?$/,'');
      var includes = s.src.match(/?.*load=([a-z,]*)/);
      (includes ? includes[1] : 'globals,tooltips').split(',').each(
       function(include) { HUBzero.require(path+include+'.js') });
   }
    });
  }
}

HUBzero.load();
```

globals.js and the HUB Namespace

Most HUBzero templates will include a global.js file that first establishes a HUB namespace and then proceeds through some basic setup routines. All HUBzero built components, modules, and templates that employ JavaScript place scripts within this HUB namespace. This helps prevent any naming collisions with third-party libraries. While it is recommended that any scripts you may add to your code is also placed within the HUB namespace, it is not required.

Some additional sub-spaces for further organization are available within the HUB namespace.

Separate spaces for Modules, Components, and Plugins are created. Once again, this further helps avoid possible naming/script collisions. Additionally, one more Base space is created for basic setup and utilities that may be used in other scripts.

```
//  Create our namespace
if (!HUB) {
 var HUB = {};

 // Establish a space for setup/init and utilities
 HUB.Base = {};

 // Establish sub-spaces for the various extensions
 HUB.Components = {};
 HUB.Modules = {};
 HUB.Plugins = {};
}
```

To demonstrate adding code to the namespace, below is code from a script in a component named com_example.

```
//  Create our namespace
if (!HUB) {
 var HUB = {};

 // sub-space for components
 HUB.Components = {};
}

// The Example namespace and init method
HUB.Components.Example = {
 init: function() {
  // do something
 }
}

// Initialize the code
window.addEvent('domready', HUB.Components.Example.init);
```

Loading From An Extension

Components

Occasionally a component will have scripts of its own. Pushing JavaScript to the template from a component is quite easy and involves only a few lines of code.

```
// Get the document object
$document =& JFactory::getDocument();
// Check if the file actually exist
if (is_file(JPATH_ROOT.DS.'components'.DS.'com_example'.DS.'example.js
')) {
 // Add the file to the list of scripts to be outputted in the templat
e
 $document->addScript('components'.DS.'com_example'.DS.'example.js');
}
```

First, we load the document object. Next we check for the existence of the JavaScript file we wish to load. If found, we add it to the array of scripts that will be outputted in the <head> of the site template.

Modules

Loading JavaScript from a module is the same as loading from a component save one minor difference: the path to the JavaScript file is obviously different.

```
// Get the document object
$document =& JFactory::getDocument();
// Check if the file actually exist
if (is_file(JPATH_ROOT.DS.'modules'.DS.'mod_example'.DS.'mod_example.j
s')) {
 // Add the file to the list of scripts to be outputted in the templat
e
 $document->addScript('modules'.DS.'mod_example'.DS.'mod_example.js');
}
```

Plugins

Loading JavaScript from a plugin is the same as loading from a component or module save one minor difference: the path to the JavaScript file is obviously different.

```
// Get the document object
$document =& JFactory::getDocument();
// Check if the file actually exist
```

JAVASCRIPT

```
if (is_file(JPATH_ROOT.DS.'plugins'.DS.'examples'.DS.'test.js')) {
 // Add the file to the list of scripts to be outputted in the templat
e
 $document->addScript('plugins'.DS.'examples'.DS.'test.js');
}
```