gs.content.js.loader Documentation

Release 1.0.6

GroupServer.org

January 25, 2016

Contents

Author Michael JasonSmith

Contact Michael JasonSmith <mpj17@onlinegroups.net>

Date 2016-01-25

Organization GroupServer.org

Contents:

CHAPTER 1

gs.content.js.loader API

In GroupServer a page may be made up of multiple components that are quite separate from each other. A common pattern is for some a component to add some HTML to the page, and then to inject some JavaScript code into the bottom of the page. Each component is usually unaware of what modules have been loaded by the other components. To get around this global tracking of loaded modules prevents modules from being loaded and parsed unnecessarily. While originally written for GroupServer, there is nothing specific to GroupServer in this product.

gsJsLoader
The global object (singleton) that keeps track of what is loaded.

The *gsJsLoader.with_module()* method of the gsJsLoader object loads one or more resources and runs a function afterwards. Introspection is provided by the *gsJsLoader.loaded()*, *gsJsLoader.loading()* and *gsJsLoader.exists()* methods.

gsJsLoader.**with_module**(*url* | *[url1, url2... urlN], fn*)

Arguments

- **url** (*string*) – The URL of the module you wish to load. If a *list* of URLs is supplied each module is loaded *in sequence*.
- **fn** (*function*) – The function to execute after the module has loaded. (Usually this is the code that depends on the module.) If a *list* of URLs is supplied as the first argument then the function is executed after all the modules have been loaded.

Returns Nothing.

With some *modules* run a *function.*

- If the module has not been loaded and not been requested:
  - Inserts <script> elements into the <head> of the document, to cause the module or modules specified in url to load.
  - Connects the function to the load (or onload for older versions of Internet Explorer) callback of the <script>.
- If the module has been requested, but has not been loaded:
  - Connects the function to the load callback of the <script>.
- If the module has been loaded:
  - Calls the function.

gsJsLoader.**loaded**(*url*)

Arguments

- **url** (*string*) – The URL of the module to test.

Returns `true` if the module has been loaded; `false` if the module is *being* loaded or has yet to be requested.

Return type boolean

This function tests if a module has *been* loaded (past-tense).

`gsJsLoader.`**`loading`**(*url*)

Arguments

- **url** (*string*) – The URL of the module to test.

Returns `true` if the module is being loaded; `false` if the module has *been* loaded or has not been requested.

Return type boolean

Test if a module is being *loaded* (present continuous tense).

`gsJsLoader.`**`exists`**(*url*)

Arguments

- **url** (*string*) – The URL of the module to test.

Returns `true` if the module has being loaded or has been requested; `false` otherwise.

Return type boolean

Test if a module is known (it has either been loaded, or is being loaded).

1.1 Examples

Run the function `init_topic_search` with the base search code:

```
gsJsLoader.with_module('/++resource++gs-search-base-js-20121217.js',
                       init_topic_search);
```

Load two modules, jQuery and then Twitter Bootstrap. Execute the function `my_code` after both modules have been loaded.

```
gsJsLoader.with_module(['/++resource++jquery-1.8.3.js',
                        '/++resource++bootstrap-2.2.2/js/bootstrap.js'],
                       my_code);
```

Wait for the window to load, then initialise the post searching code. Detect and support loading if we are using a version of Microsoft Internet Explorer that does not support the standard `addEventListener` method:

```
if (window.addEventListener) {
    window.addEventListener('load', function () {
        gsJsLoader.with_module(
            '/++resource++gs-search-base-js-20121217.js',
            init_post_search);
    }, false);

} else {
    window.attachEvent('onload', function () {
        gsJsLoader.with_module(
            '/++resource++gs-search-base-js-20121217.js',
```

```
            init_post_search);
    });
}
```

The same call as above, but using jQuery to attach to the `load` event:

```
jQuery(window).load(function () {
          gsJsLoader.with_module(
              '/++resource++gs-search-base-js-20121217.js',
              init_post_search);
 });
```

(This module is devoid of jQuery code, so it can be used to *load* jQuery.)

CHAPTER 2

Resource

This product provides a JavaScript module as a Zope browser resource. Any Zope or Plone project should be able to use this product as-is by placing the following line in a page template:

```
<script type="text/javascript"
        src="/++resource++gs-content-js-loader-20160125.js"> </script>
```

A minified version of the module is also provided:

```
<script type="text/javascript"
        src="/++resource++gs-content-js-loader-20160125.js"> </script>
```

Users of other systems are invited to copy the file `gs/content/js/loader/browser/javascript/loader.js` out of this product.

CHAPTER 3

Changelog

3.1 1.0.6 (2016-01-25)

- Cleaning up the JavaScript, thanks to the Google JavaScript linter
- Moving the documentation to Read the Docs
- Naming the reStructuredText files as such
- Using GitHub as the canonical repository

3.2 1.0.5 (2014-03-14)

- Fixing an issue with Microsoft Internet Explorer 8
- Switching the JavaScript to `strict` mode

3.3 1.0.4 (2013-11-06)

- Setting the `zip_safe` flag, in the product setup, to `False`

3.4 1.0.3 (2013-08-30)

- Fixing the product metadata

3.5 1.0.2 (2013-03-18)

- Fixing the product metadata

3.6 1.0.1 (2013-02-19)

- Fixing the `load_modules` call

3.7 1.0.0 (2013-02-14)

Initial version. Prior to the creation of this product GroupServer lacked a standard way to asynchronously load JavaScript.

This module contains a dynamic JavaScript module loader as a Zope Resource. It allows modules to be loaded *from JavaScript*, rather than having the dependencies written in HTML `script` elements. It is based on the ideas of others (see *Acknowledgements* below), but it mainly differs from prior work in that it keeps track of a **global** list of loaded scripts. The reason for this is the odd way that GroupServer is written.

The getScript function from jQuery performs a similar role to the loader presented here. Most people should use `getScript` as I am sure that it is better written and better maintained.

CHAPTER 4

Acknowledgements

The Loader code was based on two jQuery loaders, from CSS Tricks and the blog by Joel Varty. The code to load multiple modules was based on the Async Script Loader with Callback from CSS Tricks.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

CHAPTER 6

Resources

- Documentation: http://groupserver.readthedocs.org/projects/gscontentjsloader
- Code repository: https://github.com/groupserver/gs.content.js.loader/
- Questions and comments to http://groupserver.org/groups/development/
- Report bugs at https://redmine.iopen.net/projects/groupserver/

G