JavaScript

DokuWiki makes uses of JavaScript to enhance the user experience. Like for stylesheets all JavaScript files are delivered through a single dispatcher to minimize HTTP requests, for caching and compression.

This page gives you an overview how JavaScript is loaded from DokuWiki core, plugins and templates. It also gives some info about event handling and coding style when writing JavaScript for use in DokuWiki.

JavaScript Loading

All JavaScript code is collected from all found files and concatenated as one block of code. It is then whitespace compressed (if compress is enabled) and delivered as one file. As a live example you can view the JavaScript file that is in effect on this website lib/exe/js.php. This file will be cached in the DokuWiki cache at `/dokuwiki/data/cache` and DokuWiki also instructs browsers to cache this file. So when you are developing new JavaScript, make sure to refresh those caches (hitting Shift-F5, Shift+CTRL+R or similar), whenever your script was updated.

DokuWiki will load JavaScript from the following places:

- autogenerated JavaScript (language strings, config settings, toolbar)
- lib/scripts/*.js
- lib/plugins/*/script.js
- lib/tpl/<currenttemplate>/script.js
- conf/userscript.js

As you can see you can provide JavaScript with your templates and plugins (through a `script.js` file) and can define your own scripts in `conf/userscript.js` (just create this file if it does not yet exist).

Deferred Loading

All JavaScript is loaded with the defer attribute. Here's more information about deferred loading.

If you load JavaScript through other means than the recommended methods below and that JavaScript has dependencies on any DokuWiki provided code, you need to ensure it is deferred as well. This also means that you cannot use `document.write` that relies on DokuWiki-loaded scripts.

From the 2020 Hogfather version onwards, all javascript must be load in a deferred way. To update your existing templates javascript to work in this version, you can add the `defer` attribute to the `<script>` tag.

Temporary, the feature flag defer js is available, which allows disabling.

Include Syntax

DokuWiki's JavaScript dispatcher allows you to use special JavaScript comments to include other script files. This is useful for cases where usually only a single JavaScript file would be parsed, e.g. in templates or plugins.

❗ Included files are not checked for updates by the cache logic. You need to touch the master file for updating the cache.

❗ Includes are **not** supported inside included files to avoid any circular references.

❗ Includepath may only consist of letter,digit, underscore, «/» and «.».

include

```
/* DOKUWIKI:include somefile.js */
```

This syntax will include the given file where the comment is placed. The filename is relative to the file containing the include markup unless it starts with a slash which indicates an absolute URL path.

include_once

```
/* DOKUWIKI:include_once common_library.js */
```

This syntax will include the given file where the comment is placed. The filename is relative to the file containing the include markup unless it starts with a slash which indicates an absolute URL path.

The file will only be included if not a file of the same base name was previously loaded through the include_once statement. This name is shared over all script files (from all plugins), so you should use a meaningful file name.

Using this statement makes sense if you write multiple independent plugins all using the same JavaScript library. Including it with `include_once` using the same basename will make sure the library is loaded only once even if multiple of your plugins are installed.

Coding Guidelines

When writing JavaScript for the use within DokuWiki you should follow a few rules. Because of the nature of JavaScript, failing to do so might result in not only breaking your script but all scripts in DokuWiki.

Validate your Code

As mentioned above, DokuWiki will shrink the JavaScript code when the compress option is enabled (which it is by default). To do this without introducing syntax errors, the JavaScript has to be checked more strictly than it might be when run uncompressed.

To check your code you should use the JSLint online service.

- debug your code with compress disabled but
- verify your code still works with compress enabled

Use unobtrusive JavaScript

Do not assume people have JavaScript enabled, when writing new DokuWiki functionality. Instead use JavaScript as enhancement of the user interface only, when JavaScript is not available your code should fallback to normal page reload based behavior.

jQuery makes this very easy.

Avoid Inappropriate Mixing

The old way of doing things is to embed JavaScript directly in the HTML. However, JavaScript and (X)HTML shouldn't be mixed, and indeed with DokuWiki there are many cases where they *cannot* be mixed. Here are some examples of **INAPPROPRIATE MIXING**[1]:

```
<body onload="refreshPage()">

<p>some HTML</p>

<script language="JavaScript">
  doSomethingHere();
</script>

<p>more <a href="http://example.com"
onclick="doSomethingElse()">HTML</a></p>

</body>
```

This isn't just a matter of philosophical purity: some of the JavaScript may not work. In the above example, it turns out that both DokuWiki and the <body> tag are trying to assign the page's `onload` handler to different JavaScript functions. Browsers cannot handle this conflict and the results are unpredictable.

Strictly speaking, it is possible to embed JavaScript in your HTML, but only if you know that the JavaScript has no conflict with DokuWiki. Because this requires knowledge of DokuWiki's implementation, and because DokuWiki's implementation can change, this is still not a good idea. It's wiser to be philosophically pure.

Using IDs

To modify a DOM object the JavaScript must be able to locate the object. The easiest way to locate the object is to give the associated HTML tag an ID. This ID must be unique among all IDs on the page so that referencing this ID produces exactly the right DOM object.

When you are producing your own HTML (e.g. from a template or plugin) that should be accessed from JavaScript later, be sure that the ID does not conflict with an existing ID. In particular, be sure that it won't conflict with the IDs automatically assigned to section headers. The easiest way to ensure this is to use two adjacent underscores (__) in your ID. Because section IDs are always valid pagenames, they will never contain adjacent underscores.

Inline scripts

As said before you should avoid mixing JavaScript and XHTML. However if you need to use inline JavaScript, you should wrap it like this:

```
<script type="text/javascript">/*<![CDATA[*/
...
/*!]]>*/</script>
```

If you need to add inline JavaScript to the <head> section you should write an action_plugin and handle the TPL_METAHEADER_OUTPUT event. When you define only some variables see the JSINFO variable below.

jQuery

Since the October 2011 release «Angua», DokuWiki ships with the jQuery and jQuery UI libraries.

Please follow these coding conventions when working with jQuery in DokuWiki. Please also refer to our JQuery FAQ for Plugin Developers.

No $()

jQuery is only used in compatibility mode. There is no `$()` method[2]. Use the `jQuery()` method instead.

Do not map `$()` to `jQuery()`, not even within your own (anonymous) functions.

Prefix jQuery object variables

To make it clear that a variable contains an instance of the jQuery object all these variables should be prefixed by a $ character:

```
var $obj = jQuery('#some__id');
```

DokuWiki JavaScript Environment

Predefined Global Variables

DokuWiki defines certain JavaScript variables for the use in your script:

- `DOKU_BASE` – the full webserver path to the DokuWiki installation
- `DOKU_TPL` – the full webserver path to the used Template
- `DOKU_COOKIE_PARAM` – parameters required to set similar cookies as in PHP
    - `path` – cookie path
    - `secure` – whether secure cookie
- `LANG` – an array of languagestrings
- `JSINFO` – an array of useful page info (see the section below)
- `NS` – $INFO['namespace'] passed through the function `tpl_metaheaders()`

JSINFO

DokuWiki passes the global $JSINFO to JavaScript (see mailinglist discussion). This variable is an associative array usually containing the keys:

- `id` – the current page's ID
- `namespace` – the current namespace.

And after 2018-04-05:

- `ACT` – the current mode
- `useHeadingNavigation` – whether to use the first title for Navigation links (the former global constant `DOKU_UHN` has been deprecated)
- `useHeadingContent` – whether to use the first title for Content links (the former global constant `DOKU_UHC` has been deprecated)

Other keys can easily be added from within PHP code. The usual way is using an action plugin like this:

```
function register(Doku_Event_Handler $controller) {
    $controller->register_hook('DOKUWIKI_STARTED', 'AFTER',  $this,
'_adduser');
}
function _adduser(&$event, $param) {
    global $JSINFO;
    $JSINFO['user'] = $_SERVER['REMOTE_USER'];
}
```

If you want to make sure that your plugin's data don't interfere with other plugins or DokuWiki itself consider using `plugin_<pluginName>` as prefix/top-level key.

The contents of the `$JSINFO` php variable are sent to the browser in the tpl_metaheaders() function which is called from within the used template.

If you need JSINFO in the pop-up media manager or in the media detail page you have to use respectively MEDIAMANAGER_STARTED or DETAIL_STARTED in stead of DOKUWIKI_STARTED.

[1]
Please note as well that there is no `language` attribute of the `script` tag! Instead use `type=«text/javascript»` to be standards compliant.
[2]
it might be defined but will **not** do what you expect