JavaScript

Для удобства «ДокуВики» позволяет использовать программистам JavaScript. Также как и страницы стилей CSS, все файлы JavaScript загружаются через программу-диспетчер с целью уменьшения количества HTTP-запросов, для кэширования, а также удаления пробелов и комментариев (настройка compress).

Эта статья даст общее представление о том, как скрипты JavaScript загружаются из ядра «ДокуВики», плагинов и шаблонов. Также здесь содержится информация об обработке событий и стиля программирования, когда ява скрипты используются в ДокуВики.

Загрузка JavaScript

Все яваскрипты собираются и загружаются через lib/exe/js.php, который объединяет все найденные файлы яваскрипта, убирает пробелы и комментарии (если включена опция compress) и помещает результат в кэш. Также он (`js.php`) даёт команду браузеру помещать в кэш результат, так что, когда вы разрабатываете новый яваскрит, не забывайте сбрасывать кэш браузера (например Shift+F5, Shift+CTL+R и тому подобное) всякий раз, когда ваш скрипт изменён. Если ваш сайт сидит в облаке, в отдельных случаях может потребоваться touch его настроек.

«ДокуВики» получает JavaScript из:

- автоматически сгенерированные яваскрипты (языковые строки, настройки конфигурации, панель инструментов);
- lib/scripts/*.js [1)];
- lib/plugins/*/script.js;
- lib/tpl/<currenttemplate>/script.js;
- conf/userscript.js.

Из вышеизложенного следует, что использование яваскриптов возможно в шаблонах и плагинах (с помощью файла `script.js`), а также возможно определить свои собственные скрипты в `conf/userscript.js`.

Также яваскрипт может быть добавлен в `main.php`, расположенном в `lib/tpl/<currenttemplate>`. Хорошо знакомый HTML-код <head> </head> принимает JavaScript.

Об атрибутах внешних скриптов

Однако до загрузки файла скрипта, особенно если он крупный, рендеринг сайта будет останавливаться, это стандартное поведение браузеров служит очень частой причиной для недовольства «Dokuwiki». Кроме того, скрипт исполняется немедленно после загрузки, не дожидаясь окончания сборки самой страницы. До появления атрибутов defer в HTML 4.01 (откладывающего запуск скрипта до окончания загрузки его цели) и async в HTML 5 (процесса, полностью независимого от рендеринга) для тега <script> работающие со страницей скрипты рекомендовалось размещать в конце неё. Async рекомендуется для библиотечных скриптов, а

defer - для обрабатывающих страницу, и размещать теги с такими атрибутами можно на привычном месте внутри тега head. Но если defer-скрипт опирается на async-библиотеку и та так увесиста, что не всегда успевает грузиться даже до его старта, то defer-скрипт всё же придётся поставить в конец шаблона.

Синтаксис подключения, использования (include)

Диспетчер яваскриптов «ДокуВики» позволяет использовать специальные (особые) комментарии, чтобы подгружать отличные от вышеописанных файлы скриптов. Это важно для случаев, когда в основном только один файл яваскрипт подвергается анализу (разбору) диспетчером, например в плагинах и шаблонах.

 Подключаемые файлы не проверяются механизмом кэширования на предмет обновления. Необходимо изменить основной файл, чтобы запустить механизм кэширования.

Либо использовать самообновляющий детектор `<script charset=«UTF-8» src=«/lib/tpl/шаблон/editor.js?ver=' . date ( «y-m-d_H:i:s», filemtime ( '/var/www/сайт/lib/tpl/шаблон/editor.js' ) ) . '» defer></script>`

 Подключения (includes) **не** поддерживаются внутри подключаемых файлов с целью предотвращения кольцевых ссылок (рекурсии).

 Путь для подключения (includepath) должен содержать только буквы, цифры, знак подчёркивания, знак дроби и точку.

include

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

Этот код включит указанный файл в то место, где был расположен комментарий. Путь к файлу по умолчанию относительный, но будет абсолютный, если поставить в начале знак дроби (/).

include_once

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

Этот код включит указанный файл в то место, где был расположен комментарий. Путь к файлу по умолчанию относительный, но будет абсолютный, если поставить в начале знак дроби (/).

Указанный файл будет подключен (использован), только если файл с аналогичным имененм не подгружался ранее с помощью оператора `include_once`. Данное имя файла будет доступно через любые другие скриптовые файлы (из всех плагинов), поэтому стоит давать понятное наименование такому файлу.

Использование этого оператора имеет смысл, если разрабатываются несколько независимых плагинов, каждый из которых использует одну и ту же библиотеку яваскрипт. Использование оператора `include_once` несколько раз с одинаковым именем подключаемого файла даст гарантию того, что библиотека будет подгружена только один раз, даже если будет установлено несколько плагинов одновременно.

Рекомендации по написанию кода (программированию)

Когда JavaScript используется в «ДокуВики», необходимо соблюдать несколько правил, поскольку ошибка в скрипте может привести не только к его прерыванию (остановке), но и к сбою всех скриптов в «ДокуВики».

Проверка вашего программного кода

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 you code should fallback to normal page reload based behavior.

To help you with this DokuWiki has a few predefined functions to help you with Event Handling.

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**[2)]:

```
<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>
```

This ensures your script code is served in the most compatible way. Some more info is available at the XHTML 1.0: Script and Style elements specification and CDATA section interface definition.

If you need to add inline JavaScript to the <head> section you should write an action_plugin and handle the TPL_METAHEADER_OUTPUT event.

DokuWiki JavaScript Library

DokuWiki does not use any of the bigger JavaScript libraries like Prototype, Dojo or JQuery. Instead it comes with a small set of handy classes and functions that may help you with writing JavaScript code

for DokuWiki.

Event Handling

As said in Avoid Inappropriate Mixing, event handlers should not be mixed into HTML code. Instead those handlers should be assigned when the Document Object Model (DOM) was loaded. The DOM is a tree-based object representation of the HTML that is available to the JavaScript. Your JavaScript need merely figure out which objects to attach functions to and then to attach functions to them.

To attach functions to any given DOM Object in a cross-browser compatible way, the **`addEvent()`** function is provided. It takes the DOM object, an event name (like 'click') and a callback function (the handler) as arguments. This function also takes care of multiple plugins trying to register an event on the same DOM object.

Additionally, `addEvent()` changes the properties and methods of the event in Internet Explorer - so you can use the `target` property and can call the `preventDefault()` and `stopPropagation()` methods.

Unfortunately, the JavaScript in your `script.js` loads before the HTML has finished loading and before the DOM tree has been made. The objects that the JavaScript needs don't yet exist. However, you may still run JavaScript when `script.js` loads, as long as that JavaScript doesn't require DOM.

To solve this problem DokuWiki provides the **`addInitEvent()`** function. This function will register a given callback to be run as soon as the DOM is ready.

Here's an example (from the summary_enforcement tip) using both methods:

```
function enforceSummary() {
    /*...*/
}

function installSummaryEnforcement()
{
    var summary_input = document.getElementById('edit__summary');
    if(summary_input !== null)
    {
        /*...*/
        addEvent(summary_input, 'change', enforceSummary);
        addEvent(summary_input, 'keyup', enforceSummary);
    }
}

addInitEvent(installSummaryEnforcement);
```

In this example, we need to attach `enforceSummary()` to the `onchange` and `onkeyup` handlers for the summary input field. `installSummaryEnforcement()` does this.

The call to `addInitEvent()` will run the `installSummaryEnforcement()` function as soon as the DOM is loaded.

Notice how `installSummaryEnforcement()` itself works. First it acquires a DOM object by ID

(though there are other ways to acquire it). In this case the object may not exist since the summary field is only shown when editing a page, so the function first tests to see if it got the object. If it did, it calls `addEvent()` to attach `enforceSummary()` to the event handlers.

The DokuWiki event functions were originally provided by Dean Edwards here and here.

- For deeper insight on the event system see the source

Predefined Global Variable

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
- `LANG` – an array of languagestrings

SACK (AJAX) Library

DokuWiki provides a simple AJAX library named SACK by Gregory Wild-Smith.

- See the Source tw-sack.js for details on how to use it.

$()

The $() function is a handy shortcut to the all-too-frequent `document.getElementById()` function of the DOM. Like the DOM function, this one returns the element that has the id passed as an argument. Unlike the DOM function, though, this one goes further. You can pass more than one id and $() will return an Array object with all the requested elements.

- Taken from the prototype library
- See Docs by Sergio Pereira

Additional functions

DokuWiki provides various other tool methods. Especially the following might be useful for your development: isset, getElementsByClass, findPosX, findPosY, jsEscape, prependChild.

- For details check the source of these files:
  - script.js
  - helpers.js

JSINFO

Since november 23, 2009 DokuWiki passes the global $JSINFO to javascript. (see the mailinglist mailinglist)

The usual way in a plugin is 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'];
}
```

add exact time when the array is send to JavaScript

1)

чтобы избежать излишней загрузки скриптов для тех, кто читает статью, содержимое `edit.js` и `media.js` загружается только в режиме редактирования или отображения медиафайлов

2)

Please note as well that there is no `language` attribute of the `script` tag! Instead use `type=«text/javascript»` to be standards compliant.