Using JSDoc for organizing our CSS code

In this article I'll explain how to create JSDoc-like tags for keeping our CSS organized. A basic knowledge of JSDoc is not required. This is just a proposal, not a recommended best-practice.

In Java there is a tool called *javadoc*. This tool creates the API's documentation in HTML format by using comments inserted in the source code. JSDoc is a similar tool for JavaScript. The following table shows the special tags used for creating the HTML documentation. Each comments block must start with `/**` and end with `*/`.

JSDoc tags

Tag	Description
`@param` `@argument`	A parameter of a function, with its name and description.
`@return` `@returns`	The returned value of the function.
`@author`	The author of the code.
`@deprecated`	The function is not approved and can be removed.
`@see`	Creates a HTML link with a description of the specified class.
`@version`	Specifies the released version.
`@requires`	Creates a HTML link with the specified class required by that class.
`@throws` `@exception`	Describes the type of exception generated by a function.
`@link`	Creates a HTML link with the specified class. Similar to `@see`, but embedded in the comment's text.

`@fileoverview`	Special tag. When used in the first documentation's block, specifies that this block should be used for providing an introduction to the file itself.
`@class`	Provides information about the class and it's used in the constructor's documentation.
`@constructor`	Specifies a function as a class constructor.
`@type`	Specifies the type returned by a function.
`@extends`	Specifies that a class works as a sub-class of another one.
`@private`	Specifies that a class or function is private.
`@final`	Specifies that a value is a constant type.
`@ignore`	JSDoc ignores the functions marked with this tag.
`@member`	Shows that a function is a member of a given class.
`@base`	Forces JSDoc to view the current class constructor as a subclass of the class given as the value to this tag.
`@addon`	Marks a function as being an *addon* to a core JavaScript function that isn't defined within your own sources.
`@exec`	Forces JSDoc to "execute" this method as part of its preprocessing step, in the same way that class contructors are executed. This can allow attributes to be added to a class from within a function.

Now we can start from the above tags and build up our new tags for a CSS layout. Let's start with some general parameters for a hypothetical layout. The code could be the following.

```
/*
@overview Wordpress Template
@author Gabriele Romanato
@version 1.0
```

```
@link http://gabrieleromanato.name
*/
```

As you can see, this is a simple introduction that we can put at the very beginning of our style sheet. In that vein, we can declare the type of our layout and the sections contained in the document.

```
/*
@type 2-columns layout with header and footer
@sections #header, #content, #sidebar, #footer
*/
```

We can do the same for each section:

```
/*
@section #header
@elements h1
*/
```

If an element is selected by a class selector, we can specify it as follows:

```
/*
@class .left, .right
@elements img, span
*/
```

Obviously we can add a simple CSS comment to each section, as usual. This is only a matter of personal choices and we are free to use other techniques as well.