

CSSDOC

putting together style, docblocks and tags

CSSDOC Second Public Draft

Author: Tom Klingenberg <tklingenberg@lastflood.net>

Version: 0.2.22

Date: 2008-11-16

Table of Contents

CSSDOC Second Public Draft	1	2.2.5. Reference (Tag Value)	11
Table of Contents	2	2.2.6. CSSDOC command (Tag Value) ..	12
Introduction	3	2.2.7. Section Name (Tag Value)	12
CSSDOC Project	3	2.3. Group of Tags	12
Feedback	3	2.3.1. File Comment Tags	12
1. Basic Elements	4	2.3.2. Section Comment Tags	13
1.1. DocBlock	4	2.3.3. Tags related to CSS-Hacks	13
1.1.1. DocBlock Examples	4	2.3.3.1. Bugfix	13
1.2. Comment Types	5	2.3.3.2. Workaround	14
1.2.1. File Comment	5	2.3.4. Versioning and Packaging Tags ...	14
1.2.2. Section Comment	6	2.3.5. Application defined Tags	14
1.2.2.1. Hierarchical Structure of		2.3.6. Parser and Control Tags	15
Sections	7	3. Tag Reference	17
1.2.3. Standard Comment	7	3.1. Proposed Tags (1.0-PRE)	17
1.2.4. Standard CSS Comment	7	3.2. Obsolete Tags	29
1.3. Tags	7	Appendix A: Index of Tags	31
1.4. Syntax	8	Appendix B: CSS that looks like Tags	32
1.4.1. Tag Syntax	9	List of CSS at-rules	32
1.4.2. Tagname Syntax	9	Appendix C: Tags used by Third Parties	33
1.4.2.1. Vendor-specific Tagnames	9	@group	33
1.4.3. Tagvalue Syntax	9	Appendix D: List of Browsers	34
1.5. Parser	10	Appendix E: Real Life Examples	35
2. Tags	11	Publications	35
2.1. Tag Names	11	Appendix F: Specification Resources	36
2.2. Tag Values	11	Appendix G: Index of Examples and	
2.2.1. Text (Tag Value)	11	Illustrations	37
2.2.2. Yes/No (Tag Value)	11	Index of Examples	37
2.2.3. Date (Tag Value)	11	Index of Illustrations	37
2.2.4. Timestamp (Tag Value)	11	Appendix Z: Todo	38

Introduction

CSSDOC is a convention to comment Cascading Style Sheets (CSS). It is an adoption of the well known JavaDoc / DocBlock based way of commenting source-code.

Because CSS has not such an explicit commanding character then program-source-code that will be compiled or executed later, the scope what is documented differs compared to the programming languages. Anyway, the benefits of using CSSDOC is mutual and will hopefully help to further develop:

- CSS-Authors can use it to beautify their CSS files and to find a convention they can refer to while sharing files with other authors, writers, designers, webapp devs, creative-theme-directors,
- Software developers can use it to provide more comfortable capabilities inside their software for CSS-Authors and vice versa.
- Application and tools can use CSSDOC to gather additional data from CSS files an author has written down.

This Draft will describe the elements CSSDOC consists of and acts as a reference to a first set of tags that were developed from Summer 2007 to March 2008. It is the **Second Public Draft** and some tags are now obsolete, have been renamed or added compared to the **First Public Draft**. The overall document structure has been reworked and appendices have been added for a faster read and better understanding.

This file is intended to become 1.0-PRE which acts as a blueprint for the first technical implementations. From time to time you find editorial notes within the text, most often @edit tagged. These parts are not fully finished and there is some work to do. But a reader should feel her- or himself free enough to drop feedback to those parts as well as to those without any notes.

As always, any kind of feedback to this draft is very welcome. If you are interested to help with the project, do not hesitate to contact us. See [Feedback](#) for more info.

Tom Klingenberg <tklingenberg@lastflood.net>

CSSDOC Project

The CSSDOC project is done together with Timo Derstappen and Dirk Jesse. I'd like to thank them for the ongoing support and for taking care. For example Dirk Jesse did a great job with CSSDOC inside his mature **YAML CSS Framework**¹ and this specification benefits a lot from his input.

<http://www.cssdoc.net/>

Feedback

The CSSDOC Project is planing to re-work the website infrastructure to better support the specification process and to provide community support. As long as we still use the first and password protected website due to our limited resources, please contact us by email.

¹ <http://www.yaml.de/en/home.html>

1. Basic Elements

The main elements CSSDOC consists of is both human- and machine-readable data written into comments of CSS files. These comments follow a special format and are so called **DocBlock** comments or **documentation comments** (“doc comments”). This concept has been borrowed from programming languages like Java or PHP, but you do not need to be a programmer to understand this easy concept. The name is DocBlock because of its nature being a comment-block used for documentation – that is DocBlock in short.

Documentation comments in a CSS file are a CSSDOC comment then. A CSSDOC comment can be of one of three types: **file comment**, **section comment** and **standard comment**. Each DocBlock then contains certain tags which can be used to put structured data into a comment and therefore to the whole file or a section of it.

A tag is a short word prefixed by the @-symbol placed inside the comment. This is exactly the same concept like with Javadoc or PHPDocumentor. In the following paragraphs written DocBlocks and tags will be describden in more detail including some examples.

For basic comment usage in CSS², please see the official CSS specification³.

1.1. DocBlock

These are comments spanning multiple lines written in a specific convention. They start with a `/**` character-sequence and are followed by one or more nicely intended lines beginning with a `*`. The DocBlock is closed on the last line with the `*/` character-sequence which has to be intended as well.

NOTE: Javadoc 1.4 does partially lower the strictness on how to write the DocBlock. CSSDOC parser authors might want to take this into account. Please check the Javadoc specification for this.

Inside this “Block marker” there is space for a title (normally the first line) and a human readable description, which can be followed separated by an empty line in the DocBlock. The **Larger DocBlock example** shows this.

1.1.1. DocBlock Examples

```
/**
 * DocBlock Comment
 */
```

Example 1: Small DocBlock example

² See “4.1.9 Comments” (<http://www.w3.org/TR/CSS21/syndata.html#comments>)

³ For example Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (<http://www.w3.org/TR/CSS21/>).

```

/**
 * Larger Comment
 *
 * This DocBlock comment is even longer an contains
 * even more Text.
 */

```

Example 2: Larger DocBlock example

Both examples show the part of a doc comment that is called the **main description**.

1.2. Comment Types

Each DocBlock is used in CSSDOC as a certain comment type. A CSS file must have at least one comment block on top which is the **file comment** block. It can be followed by any amount of **section comment** blocks and **standard comments** including none.

1.2.1. File Comment

The **file comment** DocBlock holds the comment of the CSS file as a whole. It can contain a shorter or longer **main description** as well as a **tag section** for this file meta data. This can be something as simple as the authors name as well as things like version numbering, a certain style the file is part of etc. . Often it is nice to name a project with `@project` or use `@colordef` to define used colors inside the file comment:

```

/**
 * Homepage Style
 *
 * Standard Layout (all parts) for Big Little Homepage
 *
 * This style has been designed by Mina Margin. It reflects
 * the composition of colors through the years of the
 * customers project as well as the boldness it implies.
 *
 * @project    Big Little Homepage
 * @version    0.2.8
 * @package    xhtml-css
 * @author     Mina Margin
 * @copyright  2008 by the author
 * @cssdoc    version 1.0-pre
 * @license    GPL v3
 *
 * @colordef   #fff; white
 * @colordef   #808080; standard grey
 */

```

Example 3: File Comment DocBlock example

The **file comment** DocBlock is placed on top of the file. It can be preceded by the `@charset` rule⁴ which needs to be on the very top of each CSS file.

4 See <http://www.w3.org/International/questions/qa-css-charset>

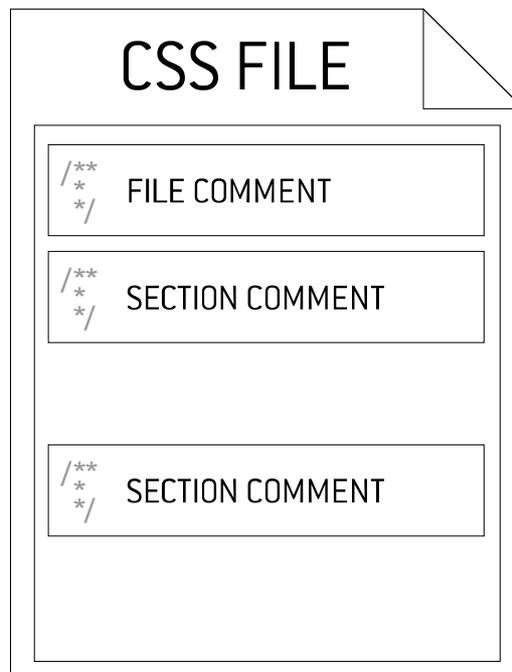
NOTE: By definition, the file comment is the first CSS comment written as DocBlock in the file.

A parser must identify it as the file comment but a non-docblock-styled comment proceeding it should be treated as a standard CSS comment by the parser. Even if the file comment should not be proceeded by any other comment. This should be considered as bad practise. A parser might throw a warning then.

A parser must find at least the file comment. If it does not exists, the file must be classified as CSS file without CSSDOC.

1.2.2. Section Comment

A **section comment** opens a new section in a CSS file that can be used to structure CSS code. The meaning of section is quite broad and very limited at once⁵, there is no specified meaning of what a section is by this specification at all. Common section examples in CSS are: **reset**, **layout**, **typography** or **content** but each CSS author does such definitions and maintains them primary on its own⁶.



To open a new section, the `@section` tag must be placed into the section comment. Infact the `@section` tag turns a docblock into a section comment. Any docblock not being the first one without the `@section` tag is just a standard CSSDOC comment.

The following example shows a simple section comment for a CSS-reset block.

⁵ “A section is a section is a section.”

⁶ Interviews made with CSS-users of various professional-levels made this clear, including the documented ones made Devhouse Cologne 2007.

```

/**
 * Reset
 *
 * @section reset
 * @see      YUI Reset CSS, http://developer.yahoo.com/yui/reset/
 */

```

Example 4: Section Comment DocBlock (Reset section)

The Illustration **Usagescheme of file and section comment** shows the usage of File- and Section Comments within a CSS file.

To add even more structure to a document, sub- and subsub-sections can be used for that.

1.2.2.1. Hierarchical Structure of Sections

Complex CSS-Files might benefit from subsections. Therefore a first Idea was to propose tags like `@subsection` and `@subsubsection` to offer a way to create a finer structure. See [@subsection](#) and [@subsubsection](#) in the Tag Reference for more Info. This concept has not been tested roughly right now and Feedback is much appreciated.

|| *[@edit: currently we are testing @subsection and @subsubsection to create a hierarchical structure of sections. For larger files this is a must but these tagnames aren't very much tested right now.]*

1.2.3. Standard Comment

|| *[@edit: it is an idea to name the CSSDOC standard comment **inline comment**.]*

Sometimes it does not make sense to open a new section only to make a comment. In these cases, a CSSDOC comment can be opened and tags can be put in. The following Example does demonstrate this:

```

/** Non Standard CSS follows.
 * @cssdoc parsing off */

```

Example 5: Non-file and non-section CSSDOC comment.

This can be further shortened by letting a CSSDOC comment spanning only one tag:

```

/** @cssdoc parsing on */

```

Example 6: Very short CSSDOC comment

1.2.4. Standard CSS Comment

A **standard CSS comment** is - as the name says - a standard CSS comment by the CSS specification. These comments can be used to put additional comments into CSS files, the CSSDOC specification does not limit its usage. As by definition, even a DocBlock comment is a standard CSS comment.

1.3. Tags

Tags are one-word-identifiers that start with the `@`-symbol.

```

/**
 * Comment with Tags
 *
 * This DocBlock comment has tags.
 *
 * @author Mina Margin
 * @version 1.0
 * @see http://mina.blogsaroundthe.net/
 */

```

Example 7: File comment with three tags.

Tags are used in an area inside the DocBlock comment after the title and the human readable comment (**main description**), that is called tag section. A tag can have a value which should be intended properly if the preceding or following line contains a tag and value pair as well.

1.4. Syntax

At the lexical level, CSSDOC must match with the standard sequence of tokens for CSS. In CSS21 the definition uses lex-style regular expressions. Octal codes refer to ISO 10646⁷.

Therefore anything - and that means **main description**, tag naming and values as well - must match the comment description⁸ without ending it:

Token	Definition
COMMENT	<code>\/\ * [^*] * \ * + ([^/*] [^*] * \ * +) * \ /</code>

This might look quite complicated and the CSS specification misses to reference **lex-style regular expression**⁹ or even define it at all. So it is advisable to get a closer look inside the CSS specifications:

“Comments begin with the characters ‘/*’ and end with the characters ‘*/’. They may occur anywhere between tokens, and their contents have no influence on the rendering. Comments may not be nested.” - 4.1.9 *Comments* in *CSS 2.1 Specification*¹⁰

“Textual comments in CSS style sheets are similar to those in the C programming language [7]:

```
EM { color: red } /* red, really red!! */
```

Comments cannot be nested. For a CSS1 parser, a comment is equivalent to whitespace.” - 1.7 *Comments* in *Cascading Style Sheets, level 1*¹¹

That specified, CSSDOC can create its own syntax within this domain. It is great to see, that the

⁷ No real data, only a ISO-Number specified: "Information Technology - Universal Multiple- Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:2003. Useful roadmap of the BMP and plane 1 documents show which scripts sit at which numeric ranges.”; compare <http://www.w3.org/TR/CSS21/refs.html#ref-ISO10646>

⁸ <http://www.w3.org/TR/CSS21/syntax.html#tokenization>

⁹ Lex - A Lexical Analyzer Generator by M. E. Lesk and E. Schmidt; <http://dinosaur.compilertools.net/lex/index.html>

¹⁰ <http://www.w3.org/TR/CSS21/syntax.html#comments>

¹¹ <http://www.w3.org/TR/CSS1#comments>

generous comment usage definition in CSS and the ability to use unicode in cssfiles leave a lot of space for CSSDOC comments and data. But even so any character sequence but not `*/` could be used, CSSDOC syntax has been defined as a very simple one.

|| *[@edit: there should be the aim to define cssdoc syntax in lex style as well sothat it is compatible to the definition of the css syntax / tokenization.]*

1.4.1. Tag Syntax

Single tag without a value (empty value):

```
\s*\s{tagname}
```

Tag with a value

```
\s*\s{tagname}\s*{tagvalue}
```

Tag with a value spanning multiple lines:

```
\s*\s{tagname}\s*{tagvalue}({nl}  
\s*\s+{tagvalue})+
```

As where `nl` is `{\n|\r\n|\r|\f}`.

1.4.2. Tagname Syntax

Tag names (`{tagname}`) follow a very simple syntax:

```
[@] [_a-z-] [_a-z0-9-]+
```

1.4.2.1. Vendor-specific Tagnames

In CSSDOC a tagname can begin with '-' or '_' (a dash or an underscore). Those are reserved for vendor-specific extensions. This is the same concept as with vendor-specific extension CSS is using¹².

|| *[@edit: check against vendor-specific tagnames with phpdoc and javadoc]*

1.4.3. Tagvalue Syntax

Tag values (`{tagvalue}`) have a simple syntax as well:

```
([a-z0-9A-Z] | [$_.\+!*'\(\), :\[ \]-] | [\s\t])+
```

For a simple value of smaller size which fits into the current line, it is simply anything from the first non-space character (`[^\s]`) to the end of the line minus spaces at the end.

An empty line in the next line within the comment is not part of the last value. Empty lines empty values.

NOTE: Next to tagvalue syntax, a specific **notation of a value** might apply depending on the kind of tag used.

¹² See 4.1.2.1 *Vendor-specific extensions* in Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (<http://www.w3.org/TR/CSS21/syntax.html#vendor-keywords>)

*[@edit: this concept does not allow unicode stuff for values. That's a pity. Maybe create a first parser as a test to re-think the possibilities. See encoding of URLs¹³ and ID, IDREF or Name: [a-zA-Z][a-zA-Z0-9\-_\:\.]*¹⁴*

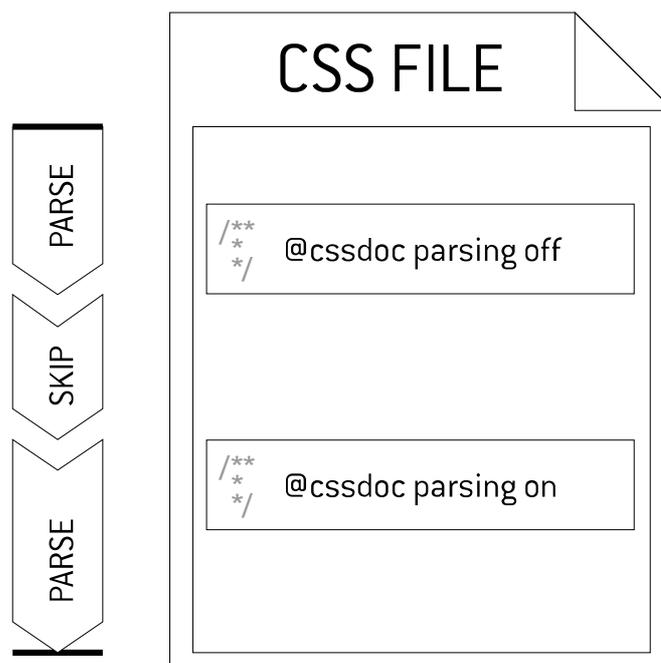
1.5. Parser

A Parser is a program reading the machine-readable parts from a CSSDOC CSS file. CSS parsers already do exist, CSSDOC enabled CSS parsers not yet.

A CSSDOC enabled parser must parse as intended. That means, it handles the CSS file properly and is taking care of the comments as well. That is called **normal parsemode**. The CSSDOC command **parsing off** must throw the parser into **skipping parsemode** which enables the parser to exclude a range of the file from syntactical correct CSS parsing. This is a feature a CSSDOC parser must have.

NOTE: **skipping parsemode** has been defined to work around useragent CSS-parsing-bugs that would stop a strict-standards-working CSS parser.

This solves problems many CSS tools are facing. Non-standard CSS code can not be properly optimized nor excluded. Such tools can now refer to CSSDOC to include such a functionality.



More Information can be found in [2.3.6. Parser and Control Tags](#) and in the Tag Reference for Tag [@cssdoc](#) .

13 See <http://www.rfc-editor.org/rfc/rfc1738.txt>

14 See <http://www.w3.org/TR/html401/types.html#h-6.2>

2. Tags

Tags have a specific meaning and are machine-readable. Some tags must be used on a per file basis, some tags are optional and some tags can only be used inside the one or other comment type (file or section). The name, meaning and location of each tag is written down in the [Tag Reference](#) .

Tags can be used for various meanings. The name of a tag defines the meaning of it. General purpose tags mainly go inside the **file comment**. Those related to parts of the document into **section comments** and sometimes a whole section might be defined only to separate a complicated CSS hack from well written and easy readable layout rules.

There is a set of tags that is designed for CSSDOC related parsers and applications as well as a set that can be used by authors of web applications to place meta data into CSS-Files. Some group of tags are discussed in [2.3. Group of Tags](#).

2.1. Tag Names

Valid characters for tag names are specified in [1.4.2. Tagname Syntax](#). Each CSSDOC tag name is named in [3. Tag Reference](#).

2.2. Tag Values

Next to the syntax of tag values, specific notations of values might apply. The following tag value types are a first compilation of different types.

2.2.1. Text (Tag Value)

Simple text that might span over multiple lines.

2.2.2. Yes/No (Tag Value)

As a rudimentary datatype, CSSDOC supports a kind of bool / booleanstate or yes/no value. It is 0 or 1, yes or no and true or false.

2.2.3. Date (Tag Value)

Some Tags need to specify a Date Value. The problem of writing dates down in a consistent way has been already solved. ISO 86011 should be preferred. If this is unfitting in bigger, international Projects, Coordinated Universal Time (UTC)¹⁵ is suggested. Might contain a placeholder for versioning systems as well (Subversion etc.).

`|| [@edit: check subversion, check ISO 86011]`

2.2.4. Timestamp (Tag Value)

`|| [@edit: check date, check this. definition only rn.]`

2.2.5. Reference (Tag Value)

`|| [@edit: Check Javadoc amd phpdoc @see tag. check @ref,@see, check this. definition only rn.]`

¹⁵ See <http://en.wikipedia.org/wiki/UTC>

2.2.6. CSSDOC command (Tag Value)

The value of a CSSDOC command is one of a defined command. Some of the commands have parameter with their own syntax. List of named CSSDOC commands so far:

- parsing on
- parsing off
- version

NOTE: Previous Command-Suggestions as parsingon and parsingoff have been replaced by this more modular command and attribute approach.

Please see [Parser and Control Tags](#) for more information.

|| *[@edit: Command name and parameters value(s) have to be lexicially defined.]*

2.2.7. Section Name (Tag Value)

|| *[@edit: This should be defined while the first parser is created.]*

2.3. Group of Tags

It makes sense to group tags into specific topics. These groups help to better understand the usage-ideas behind certain tags:

- Grouped by structure/location:
 - [File Comment Tags](#) are tags used in the **file comment**.
 - [Section Comment Tags](#) are tags used in a **section comment**.

NOTE: There are tags that can be used in both locations.

- Grouped by usage:
 - [Tags related to CSS-Hacks](#) are tags used to describe CSS hacks and workarounds.
 - [Versioning and Packaging Tags](#) are tags that can be used for versioning and to add a CSS file to packages (of an Application etc.).
 - [Application defined Tags](#) are tags used for (web-) application metadata
 - [Parser and Control Tags](#) deliver information for a CSSDOC parser (CSSDOC commands)

2.3.1. File Comment Tags

Tags that are specially designed for the **file comment** do offer information related to the whole file. Basically this is information about the author (@author) as well as version or licensing information (@version, @revision, @license). See [Tag Reference](#) for all available file comment tags.

2.3.2. Section Comment Tags

Tags for a **section comment** are used to tell something about the section started by the comment the are part of. First of all, this is `@section` which defines the **section comment**. See [Tag Reference](#) for all available section comment tags.

2.3.3. Tags related to CSS-Hacks

CSS-Hacks is CSS written in such a way that only specific browsers will render it or that it will circumvent parsing or display bugs of specific browsers.

“CSS hacks take advantage of browser bugs to perform magic such as "hiding" `CssRules` from specific `WebBrowser`'s, or kicking browsers that don't follow the specs into line. There is a long running occasional debate over whether or not these hacks should be used [...]” - *CssHack in the CSS-Discuss Wiki*¹⁶

It is a complex topic in CSS and the usage of CSSDOC will reflect it with some tags designed for this problematic topic in mind. Mostly these tags are informative.

CSS-Hacks can be classified and explained with CSSDOC. They can be marked as valid or invalid CSS (`@valid`), affected browsers can be listed and browsers for which the hack were written for (`@affected`, `@css-for`) can be written down.

CSSDOC offers two classifications for CSS-Hacks: Bugfix (`@bugfix`) and Workaround (`@workaround`):

Bugfix	Workaround
Should remove CSS-Rendering Problems without having bothersome adverse effects.	Some browsers need a reduced rendering because they are full of errors. A Workaround works around those Issues.

Table: Bugfix and Workaround Comparison Table

2.3.3.1. Bugfix

A Bugfix should “fix” CSS-Rendering Problems (Bugs) and lead to an error-free display so that it can be compared to other Browsers output.

A good Example is the fix for the “Disappearing List-Background Bug”¹⁷. Lists in IE 6 do partial loose their Background-colors.

```
/**
 * Disappearing List-Background Bug
 *
 * @bugfix
 * @affected IE 5.x/Win, IE6
 * @valid    yes
 */

* html ul, * html ol, * html dl { position: relative; }
```

¹⁶ <http://css-discuss.incutio.com/?page=CssHack>

¹⁷ <http://www.positioniseverything.net/explorer/ie-listbug.html>

Example 8: Bugfix Example

In the Example, `position: relative` instead of the default `static` does not have any visible or otherwise negative effects. But by using these rules the faulty display of background colors on lists is removed. A perfect Bugfix.

2.3.3.2. Workaround

A workaround are rules that lead to a more error-free display but it implies that visual design goals or the CSS intention can not be matched up with. The quite famous Guillotine Bug¹⁸ for Example. It is not possible to gain a correct display in internet explorer 6 or 7. Instead a workaround is chosen to prevent an erroneous display. This is done with the cost of a “correct” display, so it is classified as workaround.

```
/**
 * IE/Win Guillotine Bug
 *
 * @workaround
 * @affected IE 5.x/Win, IE6
 * @css-for IE 5.x/Win, IE6
 * @valid yes
 */

* html body a,
* html body a:hover { background-color: transparent; }
```

Example 9: Bugfix Example¹⁹

NOTE: Using invalid CSS can break the concept of CSSDOC. Please see [Parser and Control Tags](#) for signalling a parser that some part of a file is not to be parsed because it is not valid CSS.

2.3.4. Versioning and Packaging Tags

If your CSS file is something that evolves from time to time, you can think about using tags from this group to create versioning information and package management.

Examples of those tags are: `@date`, `@lastmodified`, `@package`, `@revision`, `@since`, `@subpackage`, `@version`

2.3.5. Application defined Tags

NOTE: This concept is obsolete. Please see the suggestion of [1.4.2.1. Vendor-specific Tagnames](#).

¹⁸ <http://www.positioniseverything.net/explorer/guillotine.html>

¹⁹ Please compare to the YAML documentation and sourcecode for a broad range of bugfix and workaround cases. <http://www.yaml.de/>

Application developers can use CSSDOC to put data into CSS files. To not affect other applications, an application defines its namespace with the `@appdef` tag first. The `@appdef` registers the specific **appname** so that this name can be used in own `@app-appname-name` tags later on.

Since the information is automatically parseable and a parser can reveal tags by their application specific names, applications can but meta data into css files without interfering standard procedures.

|| *[@edit: discuss vendor specific tagnames in more detail and re-think @appdef / remove it.]*

2.3.6. Parser and Control Tags

Parser and Control Tags offer the possibility to leave information for a CSSDOC parser to skip specific parts of a CSS-File because complicated hacks are used that will affect strict parsing. CSSDOC provides an easy to use and to integrate solution for these kind of problems. This is done with the `@cssdoc` tag which can signal commands to a parser.

```
/* @cssdoc parsing off */
[...]
```

Example 10: Small Parser Control Example

In this small example `parsing off` signals a CSSDOC parser to stop parsing **right after** the current comment. The parser will then stop normal, syntactical correct parsing and will skip all text until the next `@cssdoc parsing on` tag/value pair. The parser will then backtrace (by searching backwards for “`/**`”) to the beginning of the comment which contains that character sequence and will switch back to CSSDOC parsemode **right at the beginning** of that comment.

```
/**
 * @css-for      ie5
 * @workaround  Faux Box Modell
 * @valid       false; This is a IE parser bug for comments
 * @cssdoc      parsing off
 */
.box {width:280px; padding:20px;} /* * /.box {width:240px;
padding:20px;} */
```

```
/**
 * @section    plain, erroneous texts
 * @cssdoc     parsing on
 * @cssdoc     parsing off
 */
```

This is not valid CSS at all but it will never be a problem for a browser to handle this file nor for a strict CSSDOC parser.

```
/* @cssdoc parsing on */
```

Example 11: Detailed Parser Control Example

The detailed example above displays the flexibility of this Idea of CSSDOC parser control. Two Non-Standard CSS-Parts can follow each other and having metadata.

This tag can be used for other parser control jobs as well. Please see tag reference for more info.

3. Tag Reference

The Tags Reference provides information about each tag. You can find out in which location a tag can be used as well as getting more detail about its' value and usage.

NOTE: See [Appendix A: Index of Tags](#) for a list of all tags.

3.1. Proposed Tags (1.0-PRE)

@affected

- Location: **File Comment, Section Comment**
- Usage: Name a browser version for a bugfix or workaround.
- Value: Browser and version, or a group of useragents.
- Note: Should be used together with @bugfix or @workaround only.
- See: @bugfix, @workaround, [Appendix D: List of Browsers](#)
- Example:

```
Ex.1  /**
      * My CSS Workaround
      *
      * @workaround wrong border display
      * @affected   opera3-6.1 win
      */
```

```
Ex.2  /**
      * Avoid unneeded page breaks of #col3
      * content in print layout.
      *
      * @bugfix
      * @affected   ie7
      * @css-for   ie5.x win, ie6, ie7
      * @valid     yes
      */
```

```
#col3 { height: 1% }
```

@author

- Location: **File Comment**
- Usage: Name the author.
- Value: Identity of the author. Probably email as well. For multiple authors use multiple tags, one per author.
- Note: Before putting an email address into the file ensure that it will be removed before the file is made public. Some spambots do parse CSS files as well.
- Example:

```
Ex.1  /**
      * My CSS File
      *
      * @author Mina Margin <mm@example.com>
      * @version 1
      */
```

@bugfix

- Location: **File Comment, Section Comment, Standard CSS Comment**
- Usage: Documentation of CSS bugfixes
- Value: *None*
- Note: A Bugfix is CSS to fix (not circumvent) display- or rendering-bugs of a browser.
- See: @affected, @css-for, @workaround
- Example:

```
Ex.1  /**
      * Doubled Float-Margin Bug
      *
      * @see http://...
      *
      * @bugfix
      * @affected ie5 win, ie6
      * @css-for all browsers
      * @valid yes
      */
```

```
Ex.2  /**
      * Avoid unneeded page breaks of #col3
      * content in print layout.
      *
      * @bugfix
      * @affected ie7
      * @css-for ie5.x win, ie6, ie7
      * @valid yes
      */
```

```
#col3 {height: 1%}
```

@colordef

- Location: **File Comment**
- Usage: Definition of a color.
- Value: **Colordef string** consisting of a standard **CSS color value**, the name of a color (defined by the author) and a more generic usage identifier followed in brackets – if applicable.
- Info: The colordef tag was created to enable a CSS Author to systematically document used colors. If consistently used, a script or tool should be able to verify used colors and to change the Palette.
- Example:

Ex.1 /**

```
[...]  
* @colordef #7b633a; Bronze (Border)  
* @colordef rgb(24,59,68); Dark Satin (Lter Shadow)  
* @colordef rgb(12,47,56); Darker Satin (Shadow)  
* @colordef red; Red  
*/
```

Ex.2 /**

```
[...]  
* @colordef rgb(45,71,108); Camou d'accord (activeBackground)  
* @colordef rgb(142,155,161); Rhyme Dump Down (activeBlend)  
* @colordef rgb(251,251,231); Sellout Gay Gray (activeForeground)  
* @colordef rgb(142,155,161); Less Red* (activeTitleBtnBg)  
* @colordef rgb(242,242,223); Snowflash (alternateBackground)  
* @colordef rgb(240,240,214); Pay for Grey (background)  
* @colordef rgb(230,230,205); Silver Snow Boots (buttonBackground)  
* @colordef rgb(0,0,0); Black (buttonForeground)  
* @colordef rgb(0,0,0); Black (foreground)  
* @colordef rgb(157,168,169); Tripple Sec Dry Grey (frame)  
* @colordef rgb(157,168,169); Tripple Sec Dry Grey (handle)  
* @colordef rgb(95,114,135); Minor Mist (inactiveBackground)  
* @note glad-beige-colors  
* @see http://www.kde-look.org/content/show.php?content=75982  
*/
```

@copyright

- Location: **File Comment**
- Usage: Copyright Information of file.
- Value: Copyright string
- Example:

Ex.1 /**

```
* [...]  
* @copyright Copyright(c) 1992, 2006-2007  
* by My Little Big Corp Inc.  
* [...]  
*/
```

@creator

- Location: **File Comment**
- Usage: A CSS-Generating Application or Tool can leave it's fingerprint here
- Value: Name of creating Application or Tool
- See: @author
- Example:

```
Ex.1  /**
      * [...]
      * @creator HTML Plus 2 Reflexive CSS (WIN 3.1)
      * [...]
      */
```

@css-for

- Location: **File Comment, Section Comment**
- Usage: Naming the browser rules have been written for.
- Note: Related to @bugfix and @workaround.
- Value: Name of fixed browsers
- See: @bugfix, @workaround, [Appendix D: List of Browsers](#)
- Example:

```
Ex.1  /**
      * [...]
      * @css-for IE 6 Win32.
      * [...]
      */
```

@cssdoc

- Location: **File Comment, Section Comment**
- Usage: Controlling and signalling a CSSDOC parser
- Value: A [CSSDOC command \(Tag Value\)](#)
- See: [2.3.6. Parser and Control Tags](#)
- Example:

```
Ex.1  /**
      * [...]
      * @cssdoc version 1.0-pre
      * [...]
      */
```

```
Ex.2  /* @cssdoc parsing off */

      [...]

      /* @cssdoc parsing on */
```

@date

- Location: **File Comment**
- Usage: Document creation date of file.
- Value: [Date \(Tag Value\)](#).
- See: [@lastmodified](#), [@version](#)
- Example:

```
Ex.1 /**
 * [...]
 * @date 2007-07-24
 * [...]
 */
```

@fontdef

- Location: **File Comment**
- Usage: Definition of a font
- Value: Fontdef (Tag Value)

|| *[@edit: Specify and discuss with font-knowing CSSers]*

- See: [@colordef](#)
- Example:

```
Ex.1 /**
 * [...]
 * @fontdef freesans, arial, sans-serif, sans;
 *         serifenlose classique (headlines)
 * [...]
 */
```

@group

- Location: **File Comment, Section Comment, Standard CSS Comment**
- Status: **Reserved**; Third Party Tag, used by CSSEdit
- Note: Usage of this tag was not being specified so far. Feedback from vendor pending.
- See: [Appendix C: Tags used by Third Parties](#)

@lastmodified

- Location: **File Comment**
- Usage: Document last date of modification..
- Value: Timestamp of the last Modification.
- See: [@date](#)
- Example:

```
Ex.1 /**
 * [...]
 * @lastmodified 2007-07-24 12:34
 * [...]
 */
```

@license

- Location: **File Comment**
- Usage: To specify the Licence the File is released under.
- Value: Text if license identification. This might be a URL followed by a human readable description or vice versa.
- Example:

```
Ex.1 /**
 * [...]
 * @license Creative Commons Attribution 2.0 Licence
 *          http://creativecommons.org/licenses/by/2.0/
 * [...]
 */
```

@link

- Location: **File Comment**
- Usage: File referencing hyperlink. Example: If it is part of a Theme, this could be the Homepage of the theme.
- Value: Hyperlink
- See: @author
- Example:

```
Ex.1 /**
 * [...]
 * @link http://www.example.com/
 * [...]
 */
```

@media

- Location: **File Comment**
- Usage: Media descriptor for a file.
- Value: CSS media descriptor
- Note: Same wording is a CSS Selector
- See: [Appendix B: CSS that looks like Tags](#)
- Example:

```
Ex.1 /**
 * [...]
 * @media print
 * [...]
 */
```

@note

- Location: **File Comment, Section Comment**
- Usage: Drop a note
- Value: Text
- See: @todo
- Example:

```
Ex.1 /**
 * [...]
 * @note this is not for production use
 * [...]
 */
```

@package

- Location: **File Comment**
- Usage: Associate one or multiple files to a package
- Value: Package name
- See: @version, @subpackage
- Example:

```
Ex.1 /**
 * [...]
 * @package      xhtml-css
.* @subpackage  css
 * [...]
 */
```

@revision

- Location: **File Comment**
- Status: **Unstable**, might become obsolete because the @version is capable already and there is no use with standard versioning systems.
- Usage: Documentation of the file revision. Useful for systems like svn²⁰ or similar.
- Value: Revision number
- See: @version
- Example:

```
Ex.1 /**
 * [...]
 * @version 0.2.8
 * @revision 193
 * [...]
 */
```

20 Subversion, Source Code Management System, <http://subversion.tigris.org/>

@section

- Location: **Section Comment**
- Usage: Open and name a new section
- Value: Section name
- See: @subsection, @see
- Example:

```
Ex.1 /**
 * Reset Block
 *
 * Reset to defaults
 *
 * @section reset
 * @see      YUI Reset CSS,
 *           http://developer.yahoo.com/yui/reset/
 */
```

@see

- Location: **File Comment, Section Comment**
- Usage: Reference more Info for a section or the file.
- Value: Text, URL
- See: @section, @link
- Example:

```
Ex.1 /**
 * Reset
 *
 * Reset to own Defaults
 *
 * @section reset
 * @see      YUI Reset CSS,
 *           http://developer.yahoo.com/yui/reset/
 */
```

@since

- Location: **File Comment, Section Comment**
- Usage: Reference since which version a section or file exists.
- Value: Version number
- See: @version
- Example:

```
Ex.1 /**
 * Reset
 *
 * Reset to own Defaults
 *
 * @section reset
 * @see      YUI Reset CSS,
 *           http://developer.yahoo.com/yui/reset/
 * @since    0.2.8
 */
```

@since-revision

- Location: **File Comment, Section Comment**
- Usage: Reference since which revision a section or file exists.
- Value: Revision number
- See: @since, @revision
- Example:

```
Ex.1 /**
 * [...]
 * @since      0.2.8
 * @since-revision 193
 */
```

@site

- Location: **File Comment**
- Usage: Name a site this style has been made for
- Value: Text.
- See: @project, @link, @package
- Note: @pacakge might be an option as well
- Example:

```
Ex.1 /**
 * couleurs de l'été
 *
 * MyWebsite Style for Summer 2008
 *
 * @site      mywebsite
 * [...]
 */
```

@style

- Location: **File Comment**
- Usage: Name of this style / style this file is part of.
- Value: Text
- See: @project
- Example:

```
Ex.1 /**
 * [...]
 * @style    couleurs de l'été
 * [...]
 */
```

@subpackage

- Location: **File Comment**
- Usage: Packaging
- Value: Subpackage Name
- See: @package
- Example:

```
Ex.1 /**
 * [...]
 * @package  xhtml-css
 * @subpackage css
 * [...]
 */
```

@subsection

- Location: **Section Comment**
- Usage: Adding structure to a CSS-File
- Value: Subsection Name
- See: @section, @subsubsection
- Example:

```
Ex.1 /**
 * [...]
 * @section  reset
 */
[...]
/**
 * [...]
 * @subsection forms
 */
```

@subsubsection

- Location: **Section Comment**
- Usage: Adding structure to a CSS-File
- Value: Subsubsection Name
- See: @section, @subsection
- Example:

```
Ex.1 /**
     * [...]
     * @subsubsection forms
     */
[...]
```

```
/**
 * [...]
 * @subsubsection labels
 */
```

@tested

- Location: **File Comment, Section Comment**
- Usage: Note which browsers a section has been tested with.
- Value: List of tested browsers
- Note: For ordering reasons multiple @tested tags can be used, one for each user agent. Instead of this a comma-separated list can be used as well. Or both.
- Example:

```
Ex.1 /**
     * My CSS Workaround Testsuite
     *
     * @workaround wrong border display
     * @affected opera3-6.1 win
     * @tested ie4, ie5, ie6, i7
     * @tested opera3-6.1 win
     * @tested moz7
     */
```

@todo

- Location: **File Comment, Section Comment**
- Usage: Create a todo list on the fly.
- Value: Text
- Note: A software can use the todo tag and it's values to create a todo list for a file or project.
- See: @version
- Example:

```
Ex.1  /**
      * [...]
      * @todo   fix hover problem on nested lists
      * [...]
      */
```

@valid

- Location: **File Comment, Section Comment**
- Usage: Flag validity of CSS.
- Value: Bool (see [2.2.3. Yes/No \(Tag Value\)](#))
- Note: If a section or a Bugfix is known not to be valid CSS, this can be flagged with this @valid tag. Keep in mind that this is no replacement for switching the parser off and on again for proper parsing.
- See: @bugfix, @cssdoc
- Example:

```
Ex.1  /**
      * [...]
      * @valid   false
      * [...]
      */
```

@version

- Location: **File Comment**
- Usage: Specify the fileversion.
- Value: Version Number
- See: @revision
- Example:

```
Ex.1  /**
      * [...]
      * @version  1.2
      * [...]
      */
```

@workaround

- Location: **File Comment, Section Comment**
- Usage: Flag a workaround
- Value: **None** or a short Description of the Bugfix.
- Note: A Workaround is CSS to circumvent (not to fix or even not fixable) display- or rendering-bugs of a browser.
- See: @bugfix
- Example:

```
Ex.1 /**
 * 3 Pixel Jog Bug
 *
 * @workaround 3 Pixel Jog Bug
 * @affected ie5.x, ie6
 * @css-for ie5.x, ie6
 */
```

3.2. Obsolete Tags

@app-

- Location: **File Comment**
- Status: Obsolete (was: “Unstable, might become obsolete because of Vendor Specific Tags ([1.4.2.1. Vendor-specific Tagnames](#)).”)
- Usage: Tag of a previously defined tag-namespace. Used to create an own definition of tags.
- Value: Defined by Application
- See: @appdef
- Example:

```
Ex.1 /**
 * My CSS File
 *
 * @appdef myapp My Application
 * @app-myapp-theme Kubrik 3D
 * @app-myapp-used True
 */
```

@appdef

- Location: **File Comment**
- Status: Obsolete (was: “Unstable, might become obsolete because of Vendor Specific Tags ([1.4.2.1. Vendor-specific Tagnames](#)).”)
- Usage: Define application based Tag-Namespace that can be used later on with the @app- tag
- Value: Lower-case-String of a new Tag-Namespace Definition.
- Example:

```
Ex.1 /**
 * "Yet Another Multicolumn Layout"
 * (X)HTML/CSS Framework
 *
 * @appdef yaml
 */

[...]

/**
 * debugging of the background
 *
 * @section          debug-background
 * @app-yaml-default disabled
 */
```

@debug

- Location: **File Comment, Section Comment**
- Status: Obsolete, was once bugfix related
- See: [Tags related to CSS Hacks](#)

@ref

- Location: *File Comment, Section Comment*
- Status: Obsolete (was: “Unstable; might become obsolete or future, see @see”)
- Usage: Reference something: A File, a Website, another section
- Value: Reference
- See: @see
- Note: For a strict reference to something existing (is it really?) so that an editor can acquire it (e.g. open that file). Concept for reference is missing, see [Reference \(Tag Value\)](#). @see is doing this in javadoc. @see should do it in cssdoc as well.
- Example:

```
Ex.1 /**
 * [...]
 * @ref file:///home/mot/.defaults
 * @ref file://./print.css
 * [...]
 */
```

Appendix A: Index of Tags

Proposed Tags (1.0-PRE)	16	@section.....	23
@affected.....	16	@see.....	23
@author.....	17	@since.....	24
@bugfix.....	17	@since-revision.....	24
@colordef.....	18	@site.....	24
@copyright.....	18	@style.....	25
@creator.....	19	@subpackage.....	25
@css-for.....	19	@subsection.....	25
@cssdoc.....	19	@subsubsection.....	26
@date.....	20	@tested.....	26
@fontdef.....	20	@todo.....	27
@group.....	20	@valid.....	27
@lastmodified.....	20	@version.....	27
@license.....	21	@workaround.....	28
@link.....	21	Obsolete Tags	28
@media.....	21	@app-	28
@note.....	21	@appdef.....	29
@package.....	22	@debug.....	29
@revision.....	22	@ref.....	29

Appendix B: CSS that looks like Tags

CSS has some tokens that look like CSSDOC tags because they start with the @-symbol. This is called at-rule²¹ or rule in short. @media or @charset are the two most well known ones.

While normally those are not used inside comments, this might be the case if a CSS author uses comments to comment something out. E.g. for doing some tests. A CSSDOC parser might then accidentally run into this “tag” and tries to parse its value.

This leads to the need of a CSSDOC parser to identify commented out CSS. After some discussion it is quite clear that a robust CSSDOC parser can not and should not identify those commented out at-rules as those.

NOTE: The best-practise suggestion to solve this problem is to “comment out” at-rules by adding an undefined prefix in front of the at-rule ident part like @-disabled-media instead of @media²².

NOTE: Using comments to comment out at-rules must be considered bad practise with CSSDOC.

List of CSS at-rules

A list of known CSS at-rules in use:

- @charset <http://www.w3.org/International/questions/qa-css-charset>
- @font-face <http://www.w3.org/TR/REC-CSS2/fonts.html#font-descriptions>
- @import <http://www.w3.org/TR/REC-CSS2/cascade.html#at-import>
- @media <http://www.w3.org/TR/REC-CSS2/media.html>
- @page <http://www.w3.org/TR/REC-CSS2/page.html#page-box>

21 See <http://www.w3.org/TR/REC-CSS2/syndata.html#at-rules>

22 Disabling at-rules can be easily done by renaming it to an undefined at-rule. See <http://www.w3.org/TR/CSS21/syndata.html#parsing-errors> for more infos.

Appendix C: Tags used by Third Parties

@group

The @group Tag is used by a CSS editing software CSSEdit²³. It uses the @group Tag to create a kind of section that is reflected in an outline afterwards.

An Idea might be to integrate the @group tag into CSSDOC as well.

|| [*@edit: CSS-Edit Authors have been contacted for a Statement of how @group works, answer is pending.*]

²³ CSSEdit for Macintosh by MacRabbit software; <http://macrabbit.com/cssedit/>

Appendix D: List of Browsers

CSS authors might ask themselves how to name a browser in a short and convenient way. CSSDOC might take part in the process suggesting such notations. There are many different webbrowsers²⁴, so there can not be one Notation defined once. This table might suggest some of them:

Notation	Browser	Version	Operating System
fx 2 nix	Firefox ²⁵	2	Linux/Unix
	Icab		
ie 6 win	Internet Explorer	6	Windows
ie 3.2 os9	Internet Explorer	3.2	Apple OS9
	Konqueror		
lynx	Lynx	All Versions	All Platforms
moz 1 osx	Mozilla	1	Apple OSX
	Netscape		
opera 6.5 win	Opera	6.5	Windows
	Safari		

NOTE: Keep in mind that there is no rule in CSSDOC that forces a CSS author to write down browser names in a specific way.

²⁴ See http://en.wikipedia.org/wiki/List_of_web_browsers

²⁵ See FAQ in Mozilla Firefox 1.5 Release Notes Question 8. <http://www.mozilla.com/en-US/firefox/releases/1.5.html>

Appendix E: Real Life Examples

The following resources use a early version of CSSDOC:

- [PHPUGFFM goes CSSDOC](#)²⁶ - Webblog website, one of the first CSSDOC tests done.
- [YAML CSS Framework](#)²⁷ - Whole in-css-file-documentation is done with CSSDOC.
- [Trotz alledem! - Small, static website](#)²⁸ - Usage of CSSDOC in a life website project.

Publications

Even though CSSDOC is quite new, some printed publications about it are already available.

- German
 - "Praxislösungen mit YAML 3.0 YAML" by Dirk Jesse, Galileo Press 2007 (Second Edition), ISBN 978-3-8362-1135-2, *Page ff*
 - "Anmerkungen zum Stil - Stylesheets mit CSSDOC kommentieren" by Michael Jendryschik, iX Magazin für professionelle Informationstechnik (3 2008), Heise Zeitschriften Verlag Verlag GmbH & Co. KG, ISSN 0935-9680, Page 132-134

|| *[@edit: get infos about the page from dirks book.]*

|| *[@edit: eike zuleeg website as example]*

|| *[@edit: Technical reviewers should add their Examples as well]*

26 <http://phpugffm.de/index.php/phpugffm-goes-cssdoc,2007-05,198.html> , <http://phpugffm.de/wp-content/uploads/2007/05/style-before.css> and <http://phpugffm.de/wp-content/uploads/2007/05/style-after.css>

27 <http://www.yaml.de/en/home.html> and http://www.yaml.de/fileadmin/download/yaml_304_071127.zip

28 <http://www.widerstand-portrait.de/> and <http://www.widerstand-portrait.de/site/style.css>

Appendix F: Specification Resources

This specification brings together "best of breeds" things from different worlds. On the one hand there are programming languages, on the other hand there is computer based (graphic) design for hypertext documents (Webdesign). Even though everything in these two worlds has got a lot of technical impact, brewing together the different concepts to lead into a useable specification for both worlds – the programmers and designers one – is easy by idea and a lot of work in practise. One thing to do is to evaluate against the existing specs.

This is a listing of existing specifications and resources that are good to know about while using, discussing and specifying CSSDOC:

- **Programming**
 - **Java**
 - [Documentation Comments](#)²⁹ – The original specification on documentation comments, Chapter 18, Documentation Comments, in the Java Language Specification, First Edition, by James Gosling, Bill Joy, and Guy Steele. (This chapter was removed from the second edition.)
 - [Documentation Comments Article for Java Programmers](#)³⁰
 - **PHP**
 - [phpDocumentor Quickstart](#)³¹ - phpDocumentor for newbies written by Gregory Beaver.
- **Webdesign**
 - **CSS**
 - [Cascading Style Sheets, level 1](#)³² - W3C Recommendation 17 Dec 1996, revised 11 Jan 1999
 - [Cascading Style Sheets Level 2 Revision 1 \(CSS 2.1\) Specification](#)³³ - W3C Candidate Recommendation 19 July 2007
 - **HTML**
 - ["HyperText Markup Language Specification – 2.0"](#)³⁴ by Berners-Lee and D. Connolly, November 1995.
 - [XHTML™ 1.1 - Module-based XHTML - Second Edition](#)³⁵ - W3C Working Draft 16 February 2007

29 <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/javadoc.html#documentation>

30 <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/javadoc.html#documentationcomments>

31 http://manual.phpdoc.org/HTMLSmartyConverter/PHP/phpDocumentor/tutorial_phpDocumentor.quickstart.pkg.html

32 <http://www.w3.org/TR/1999/REC-CSS1-19990111>

33 <http://www.w3.org/TR/2007/CR-CSS21-20070719>

34 <http://ftp.ics.uci.edu/pub/ietf/html/rfc1866.txt>

35 <http://www.w3.org/TR/2007/WD-xhtml11-20070216/>

Appendix G: Index of Examples and Illustrations

Index of Examples

Example 1: Small DocBlock example.....	4
Example 2: Larger DocBlock example.....	5
Example 3: File Comment DocBlock example.....	5
Example 4: Section Comment DocBlock (Reset section).....	7
Example 5: Non-file and non-section CSSDOC comment.....	7
Example 6: Very short CSSDOC comment.....	7
Example 7: File comment with three tags.....	8
Example 8: Bugfix Example.....	14
Example 9: Bugfix Example.....	14
Example 10: Small Parser Control Example.....	15
Example 11: Detailed Parser Control Example.....	15

Index of Illustrations

Illustration 1: Usagescheme of file and section comment.....	6
Illustration 2: Parsermode-Scheme.....	10

Appendix Z: Todo

This ~~first~~ second Draft is work in progress, some parts of the document need some more work.

- Create a first Parser to Test and verify the Implementation (0% Done)
- Complete Tags Reference (98% Done)
 - Vendor specific tags decision needs to be discussed
 - check against revision and such tags with SVN usage should conform, unneeded tags should become obsolete.
- Concept for Parser and Control Tags has to be defined and written down (80% Done)
 - ~~Idea/Concept~~
 - ~~Work over Description~~
 - Parser concept and scheme. Move some text from “Parser and Control Tags” section to the parser description and use “Parser and Control Tags” section to better reference current commands.
- Charset/Encoding Information
 - ~~Write down a Tokenizer of this (Check CSS Tokenizer first)~~
 - ~~Triple-Check UTF-8 and CSS:~~
 - ~~Comments~~
 - ~~Selectors~~
 - ~~Values~~
 - “CSS1 style sheets could only be in 1-byte-per-character encodings, such as ASCII and ISO-8859-1. CSS 2.1 has no such limitation. In practice, there was little difficulty in extrapolating the CSS1 tokenizer, and some UAs have accepted 2-byte encodings.”
G.3 Comparison of tokenization in CSS 2.1 and CSS1
<http://www.w3.org/TR/CSS21/grammar.html>
- Tag value syntax description fixup.
- Tag value notation definitions:
 - Date
 - Timestamp
 - Reference
 - Section name
 - boolean
 - colordef
 - fontdef
 - appname
- ~~Index of All Tags (Done)~~
- It would be nice if each Section in Chapter Tags contains at least one Example (80% Done)

- ~~Create a list of all Examples~~
- Open an Appendix of Usage Examples (YAML, Own Sites...)
- [2.3.6. Parser and Control Tags](#) Example check Faux Box Model demo or take the ie comment parser bug.
- Take a deep breath of tagname specs from javadoc again, sepspecially the file comment tags.
- Renoved Notes
 - Browser Notations:
 - Take OS-Specific or Browser-Specific Themes into account
 - Maybe suggest version spanning as well? (from-to, lt, gt, not)
 - And group spanning?
- Ideas
 - @tagged selector, @tagged ruleset, @tagged rule, @tagged attribute.
 - the /* LTR */ marker