

# Cascading Style Sheets

CS174

Chris Pollett

Sep 10, 2008.

# Outline

- Server Side Includes
- Cascading Style Sheets

# Server Side Includes (SSI)

- So far in this class, we've learned a little about how the web server works and XHTML.
- We'll talk a little more about XHTML in a moment...
- But first, let's consider a minimal way to make web-pages more dynamic.
- Server Side Includes are a minimal programming language (not Turing complete) that is supported by both Apache and IIS.
- They illustrate several of the concepts we'll see later for more supped up server side languages.

# Getting (SSI) running

- To get Apache to use its processor for SSI directives one needs the lines  
AddType text/html .shtml  
AddHandler server-parsed .shtml  
in the httpd.conf file
- At the directory level to say SSI is allowed, you, within <directory> tags you add the line:  
Options +Includes
- This line can also be added within .htaccess files
- The default extension for files containing SSI directives is .shtml . If you'd like to use .html instead, then in your httpd.conf file you need the line:  
XBitHack on
- The file that contains the SSI directives need also to have execute privileges set for the WebServer user.

# The SSI Commands

- A basic SSI directive has the syntax:

```
<!--#element attribute=value attribute=value ... -->
```

- element can be one of config, cmd, echo, elsif, else, endif, exec, if, flastmod, include, set
- cmd and exec are for executing shell command or scripts and are typically disabled.
- echo and set are used for printing and setting a variable

```
<!--#set var='bob' value='hello' -->
```

```
<!--#echo var='bob' -->
```

- The server also automatically sets some variables according to the Common Gateway Interface (CGI):

```
<!--#echo var='QUERY_STRING' -->
```

```
<!--#set var='bob' value='hello${DATE_LOCAL}' -->
```

# More on SSI command

- The command config can be used to format dates as well as error messages:

```
<!--#config errmsg="[This is what the SSI error message will look like]" -->
```

```
<!--#config timefmt="%d, %Y" -->
```

This file was last modified <!--#flastmod file="ssi.shtml" -->.

- By the way this also show what flastmod is for.
- include can be used to include one file within another and can allow for simple templating:

```
<!--#include virtual="footer.html" -->
```

- if, elif, else operate like in similar to in Java but can't nest:

```
<!--#if expr="\${QUERY_STRING}" = "\\" ||
```

```
  \${QUERY_STRING}" = "\"print\" " -->
```

```
<!--#include virtual="classpage.html" -->
```

```
<!--#else -->
```

```
<!--#include virtual="\${QUERY_STRING}" -->
```

```
<!--#endif -->
```

# Stylesheets

- We now return to talking about XHTML, in particular, how to control the presentation of XHTML documents with stylesheets.
- Stylesheets are used to specify the look of the page and its elements.
- For instance, one can globally control things like margins, indentation, etc.
- They can be used to support the idea of separating structure of content from how it is presented.
- Cascading Style Sheets (CSS) are the standard way to do this for XHTML documents.
- CSS comes in three specs: CSS1, CSS2, CSS3, each adding more features to the last.
- Most modern browsers support CSS1 and parts of CSS2.
- The basic concept in a stylesheet is that of the value of a property that a tag has.
- Cascading refers to how settings of this value in high level stylesheets can be overridden in lower level style sheets.

# Levels of Style Sheets

- So what are the levels of stylesheets?
  - *inline*, *document*, *external*.
- *inline* --sets property value for single tag. (deprecated XHTML1.1) For example,  

```
<p style="color: red">red paragraph</p>
```
- *document* -- sets property value for the whole document.
- *external* -- sets property value for several documents till value is changed.
- If no style information is available for a given property the browser will use a default value.
- It is often useful to use the same stylesheet for several documents. The MIME type for stylesheets is text/css. You can link a stylesheet file into an xhtml file with a line like:  

```
< link rel="stylesheet" type="text/css" href="mystyles.css" />
```
- Styles can be validated at the W3C site.

# Basics of Styles

- The basic inline style command looks like:  
`<tag style="property_1: value1 ; property2: value2; ..." >`
- The basic document level style in the head of the document looks like  
`<style type="text/css" >`  
`/* here is a comment */`  
`rule_list`  
`</style>`
- Each rule has the format  
`selector {property_1: value1 ; property2: value2; ...}`
- External style sheets are similar to document level styles except you don't need the style tags.

# Examples of Simple Selectors

`h1 {font-size: 24pt} /* would apply to all h1 tags in the document */`

`h2, h3 {font-size: 14pt} /* notice applies to both h2 and h3 tags */`

You can also specify that styles should only apply to elements in certain positions within the file:

`body b i {font-size: 30pt;} /* only for bolded italic'd text within file, doesn't work NS7*/`

# Class selectors

A class is defined in a style element by putting a period with a name after it:

```
p.normal {prop_list1}
```

```
p.narrow {prop_list2}
```

To use we do:

```
<p class="normal">normal look text</p>
```

One can also have generic selectors:

```
.red {color:red}
```

These can be used with multiple tags

```
<h3 class="red"></h3> <p class="red"></p>
```

# ID Selectors

- In a similar way to class selectors, one can use a “#” to specify an id selector

p#sec1 {prop\_list2}

To use we do:

```
<p id="sec1">section1 text</p>
```

- Although only one id attribute (for instance, sec1) of a given value can appear in a document, such a selector may occur in an external style and thus be applied to id tags with that name in several documents.
- For instance, one might specify p#abstract. A typical document has only one abstract but you could use the same style for several documents.