HTML and CSS Background

Overview

Through summaries of key concepts and CSS examples, this article aims to describe how you can override FSS styles and classes with your own CSS styles.

More technical details and background information regarding the CSS concepts are provided at the W3C.

FSS Documentation

- Fluid Skinning System (FSS)
- FSS Naming Conventions
- FSS Cheat Sheet
- Mobile FSS Cheat Sheet

- Demos: Columns
- Demos: Tabs, Menus, etc.
- Demos: Text
- Demos: Themes

- HTML and CSS Background
- CSS Bugs, Gotchas

Still need help?

Join the infusion-users mailing list and ask your questions there.

Key Concepts

The following are HTML and CSS concepts that you may find helpful to have knowledge of before proceeding to learn how to override FSS styles.

Inheritance

The structure of HTML is hierarchical and can be compared to a tree structure, where one element, or HTML tag (eg. <body>, <p>) has the following characteristics:

1. It can contain several "child" elements
2. It belongs to only one "parent" element
3. It can have several "ancestor" elements (parents of its parent element)
4. It can have several "descendent" elements (children of its child elements)

CSS can support such a structure by allowing child elements to inherit certain properties of their parents when specified.

Example: You can try this out yourself.

style.css

```
body { font-family: "Times New Roman"; font-size: 0.9em; }
p { font-family: inherit; font-size: 1.2em; }
div { font-family: "Arial"; font-size: inherit; }
```

This style sheet specifies the font and text size for <body>, <p>, and tags. The `inherit` value indicates that the child element inherits the same property style as its parent element. In this case, any <p> tag will inherit the *font* of its parent, while any tag will inherit the *text size* of its parent.

doc.html

```
<html>
  <head>
    <link rel="stylesheet" href="style.css" type="text/css">
    <title>Inheritance Example</title>
  </head>
  <body>
    <p>P1 - This is a child paragraph inside the body tag.</p>
    <p>P2 - This is another child paragraph inside body.</p>
    <div>DIV1 - This is another child element inside body.
      <p>P3 - This is a child paragraph inside the div (DIV1) tag.</p>
    </div>
  </body>
</html>
```

This HTML document uses the sample style sheet above. It has two <p> tags (P1 & P2) and a tag (DIV1) with its own <p> tag (P3). According to `style.css`, P1 and P2 inherits its font from <body>, which is *Times New Roman*, while the size of its text is *1.2em*. DIV1, however, uses the *Arial* font, but inherits its text size of *0.9em* from <body>. P3, being a child of DIV1, will inherit its *Arial* font from DIV1, but have the text size of *1.2em*.

> **Caveat when using INHERIT**
>
> Not all CSS attributes can use inherit, and those that should use inherit dont do so properly in all browsers.

Style sheet origins

One thing to keep in mind when creating style sheets for your website is that visitors (users) can load their own style sheets to change the display of your design (not permanently though--this is just for *their* viewing). The reason may be that these users have certain needs, and so would use a personal style sheet specifying their preferences. For instance, users who are color-blind may want to specify their own colour scheme for differentiating headings and links.

In other words, a site's design can have more than one style sheet applied to it. There are three types:

Author: Author style sheets are those created by the person/people designing the site. These can be directly embedded within the HTML document, or access from an external CSS document.

User: User style sheets are those created by the user viewing the site. These can be applied through special features of a web browser, or other plug-in software.

User Agent (UA): User agents, such as web browsers, can have default styles that come into play when web authors or users don't specify the styles for certain elements in their style sheets.

!importance declarations

The distinction between origins of style sheets come into play with `!importance` declarations. Basically, `!importance` declarations specify which properties **can**-absolutely-**not** be overrided, depending on the type of style sheet it's in. The rules are as follows:

1. UA styles are always the bottom losers.
2. When there are no `!importance` declarations, Author styles always trump User styles.
3. When there are Author `!importance` declarations, the declared Author style trumps the undeclared User style.
4. When there are User `!importance` declarations, the declared User style trumps the Author style, declared or undeclared.

Example:

Author CSS

```
p {
  font-family: "Times New Roman" !important;
  font-size: 0.9em !important;
  color: green;
  text-indent: 2em;
}
```

User CSS

```
p {
  font-family: "Arial";
  font-size: 1.2em !important;
  color: blue !important;
  text-indent: 1.5em;
}
```

In this example, the Author's `font-family` (*Times New Roman*) wins (Rule #3), while the User's `font-size` (*1.2em*) and `color` (*green* text colour) wins over the Author's, all because of the `!important` declarations (Rule #4). Finally, in the case that neither has an `!important` declaration, the Author's `text-indent` (*2em*) wins (Rule #2).

Specificity of selectors

First, a small refresher on CSS terminology:

```
DIV P > #id .class INPUT[attribute=value] {
    font-size: 1em;
    font-color: #999;
}
```

This whole block of code is called a CSS rule set. It is made up of multiple peices

encapsulates the entire CSS selector

encapsulates the entire CSS declaration

`DIV P` is a contextual selector

`>` is a parent-child selector

`#id` is an ID selector

`.class` is a class name selector

`[attribute=value]` is an attribute selector

`font-size: 1em;` is the CSS rule

In CSS, a selector identifies an where a give set of CSS rules is applied to. Elements can range from HTML tags (eg. `body`, `p`), classes of those tags (eg. `p.left`, `p.right`), or IDs (eg. `#logo`). The example below shows where selectors can be found in a CSS document:

```
selector { /* style goes here */ }
selector { /* style goes here */ }
```

The specificity of a selector is essentially how specific that selector is. For instance, the * selector that applies to any element is the least specific, while ID selectors are the most specific. The order from least to most specific can be seen in this example:

```
* { /* style goes here */ }
ul { /* style goes here */ }
ul li { /* style goes here */ }
p.left { /* style goes here */ }
#logo { /* style goes here */ }
```

To understand specificity in more depth and detail, refer to the W3C's documentation.

Order of overrides

Once you have understood all the previous concepts, you can follow the sorting order to understand how and which styles are overridden.

1. Check to see which HTML element matches which selector in the style sheet - If there's a match, styles specified in the style sheet override the element's default styles.
2. Check for `!important` declarations amongst Author and User style sheets - Winners override losers.
3. Check for the specificity of selectors - The more specific overrides the less specific.
4. In the case that overriding cannot be resolved by the first three steps, compare the order in which the specifications are made - The latest stated specification overrides previously stated specifications. However, regardless of the order, embedded style sheets are always considered to be "later" than (and therefore override) imported style sheets.

   > This is why ordering the code for importing CSS files is critical.

Example for *#4*:

style.css

```
p.left { font-family: "Times New Roman"; text-align: left; }
div.footer { font-family: "Arial"; }
```

doc.html

```
<html>
  <head>
    <title>Order of Overrides - Example for #4</title>

    <!-- Embedded style sheet -->
    <style>
    div.footer { font-family: "Verdana"; }
    </style>

    <!-- Imported style sheet -->
    <link rel="stylesheet" href="style.css" type="text/css">

  </head>
  <body>
    <p class="left" style="text-align: right;">PARA - This is a paragraph.<
/p>
    <div class="footer">DIV - This is a section separate from the
paragraph.</div>
  </body>
</html>
```

In this example, PARA is displayed in *Times New Roman*, but instead of aligning to the left, as it was specified in `style.css`, it is aligned to the right because the <p> tag specification (`style="text-align: right;"`) occurs much later than the imported style sheet. However, DIV is displayed in *Verdana*, rather than *Arial*, even though `style.css` is imported after the embedded style sheet. This is because embedded style sheets are always considered to be "later" than imported style sheets (*#4*).