CSC 213: Web Application Development

HTML PAGE LAYOUT DESIGN: INTRODUCTION TO CSS

Lecture 4

Introduction to CSS

- Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language.
- Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

- CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the **layout, colors, and fonts**.
- This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

- CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices.
- CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element.
- In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.
- The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318.

CSS Variations

- CSS has various levels and profiles. Each level of CSS builds upon the last, typically adding new features and typically denoted as CSS 1, CSS 2, and CSS 3.
- Profiles are typically a subset of one or more levels of CSS built for a particular device or user interface.
- Currently there are profiles for mobile devices, printers, and television sets.

1. CSS 1

The first CSS specification to become an official W3C Recommendation is CSS level 1, published in December 1996. Among its capabilities are support for:

- Font properties such as typeface and emphasis
- Color of text, backgrounds, and other elements
- Text attributes such as spacing between words, letters, and lines of text
- Alignment of text, images, tables and other elements
- Margin, border, padding, and positioning for most elements
- Unique identification and generic classification of groups of attributes

2. CSS 2

- CSS level 2 specification was developed by the W3C and published as a Recommendation in May 1998.
- A superset of CSS 1, CSS 2 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and z-index, the concept of media types, support for aural style sheets and bidirectional text, and new font properties such as shadows.
- CSS level 2 revision 1 or CSS 2.1 fixes errors in CSS 2, removes poorly-supported or not fully interoperable features and adds already-implemented browser extensions to the specification

3. CSS 3

- Instead of defining all features in a single, large specification like CSS 2, CSS 3 is divided into several separate documents called "modules".
- Each module adds new capability or extends features defined in CSS 2, over preserving backward compatibility.
- Work on CSS level 3 started around the time of publication of the original CSS 2 recommendation.
- The earliest CSS 3 drafts were published in June 1999.
- Due to the modularization, different modules have different stability and are in different status.
- As of March 2011, there are over 40 CSS modules published from the CSS Working Group.

CSS Limitations

1. Poor controls for flexible layouts - While new additions to CSS 3 provide a stronger, more robust feature-set for layout, CSS is still at heart a styling language (for fonts, colours, borders and other decoration), not a layout language (for blocks with positions, sizes, margins, and so on).

- These limitations mean that creating fluid layouts generally requires hand-coding of CSS, and has held back the development of a standards-based WYSIWYG editor.

2. Selectors are unable to ascend -

- CSS offers no way to select a parent or ancestor of an element that satisfies certain criteria.
- A more advanced selector scheme (such as XPath) would enable more sophisticated style sheets.
- However, the major reasons for the CSS Working Group rejecting proposals for parent selectors are related to browser performance and incremental rendering issues.

3. Vertical control limitations

- While horizontal placement of elements is generally easy to control, vertical placement is frequently unintuitive, convoluted, or impossible.
- Simple tasks, such as cantering an element vertically or getting a footer to be placed no higher than bottom of viewport, either require complicated and unintuitive style rules, or simple but widely unsupported rules.

4. Absence of expressions

- There is currently no ability to specify property values as simple expressions (such as margin-left: 10% – 3em + 4px;).
- This would be useful in a variety of cases, such as calculating the size of columns subject to a constraint on the sum of all columns.
- However, a working draft with a calc() value to address this limitation has been published by the CSS WG

5. Lack of column declaration

- While possible in current CSS 3 (using the column-count module), layouts with multiple columns can be complex to implement in CSS 2.1.
- With CSS 2.1, the process is often done using floating elements, which are often rendered differently by different browsers, different computer screen shapes, and different screen ratios set on standard monitors.

6. Cannot explicitly declare new scope independently of position

- Scoping rules for properties such as z-index look for the closest parent element with a position:absolute or position:relative attribute.
- This odd coupling has undesired effects such as it is impossible to avoid declaring a new scope when one is forced to adjust an element's position, preventing one from using the desired scope of a parent element.

7. Pseudo-class dynamic behavior not controllable

- While possible in current CSS 3 (using the column-count module), layouts with multiple columns can be complex to implement in CSS 2.1.
- With CSS 2.1, the process is often done using floating elements, which are often rendered differently by different browsers, different computer screen shapes, and different screen ratios set on standard monitors.

Advantages of CSS

- **1. Accessibility**
- Without CSS, web designers must typically lay out their pages with techniques that hinder accessibility for vision-impaired users, like HTML tables

2. Flexibility

- By combining CSS with the functionality of a Content Management System, a considerable amount of flexibility can be programmed into content submission forms.
- This allows a contributor, who may not be familiar or able to understand or edit CSS or HTML code to select the layout of an article or other page they are submitting on-the-fly, in the same form.

2. Flexibility

- When working with large-scale, complex sites, with many contributors such as news and informational sites, this advantage weighs heavily on the feasibility and maintenance of the project.

3. Separation of content from presentation

- CSS facilitates publication of content in multiple presentation formats based on nominal parameters.
- Nominal parameters include explicit user preferences, different web browsers, the type of device being used to view the content (a desktop computer or mobile Internet device), the geographic location of the user and many other variables.

4. **Site-wide consistency**

- When CSS is used effectively, in terms of inheritance and "cascading," a global style sheet can be used to affect and style elements site-wide.
- If the situation arises that the styling of the elements should need to be changed or adjusted, these changes can be made by editing rules in the global style sheet.
- Before CSS, this sort of maintenance was more difficult, expensive and time-consuming.

5. Bandwidth

- A stylesheet, whether internal to the source document or separate, will specify the style once for a range of HTML elements selected by class, type or relationship to others.
- This is much more efficient than repeating style information inline for each occurrence of the element.
- An external stylesheet is usually stored in the browser cache, and can therefore be used on multiple pages without being reloaded, further reducing data transfer over a network.

6. Page reformatting

- With a simple change of one line, a different style sheet can be used for the same page.
- This has advantages for accessibility, as well as providing the ability to tailor a page or site to different target devices.
- Furthermore, devices not able to understand the styling still display the content.

Styles Solved a Big Problem

- HTML was never intended to contain tags for formatting a document.
- HTML was intended to define the content of a document, like:
- <h1>This is a heading</h1>
- <p>This is a paragraph.</p>

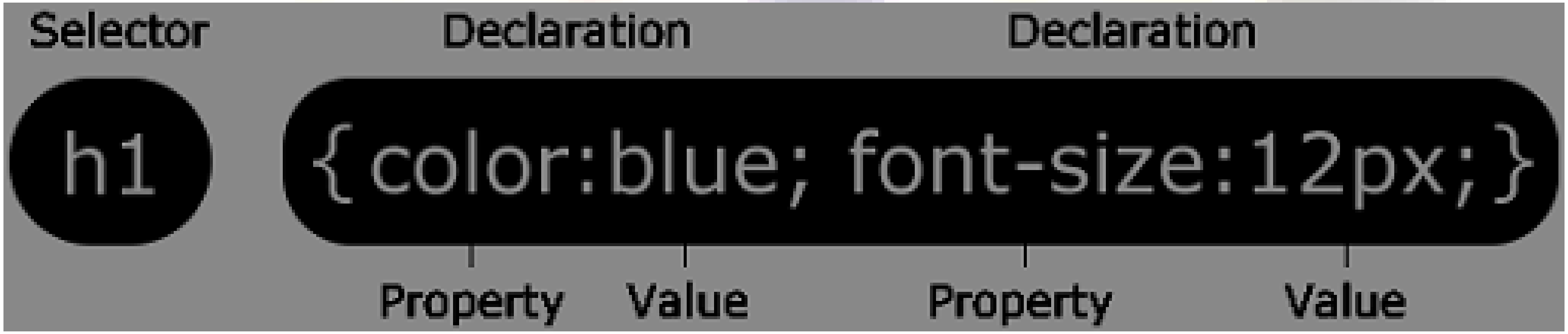
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers.
- Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS. In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file.
- All browsers support CSS today.

CSS Saves a Lot of Work

- CSS defines HOW HTML elements are to be displayed.
- Styles are normally saved in external .css files.
- External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file!

CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



The selector is normally the HTML element you want to style

CSS Syntax

- Each *declaration* consists of a property and a value.
- The *property* is the style attribute you want to change.
- Each property has a *value*.

CSS Example

- A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets:

- p {color:red;text-align:center;}

- To make the CSS more readable, you can put one declaration on each line, like this:

```
P

{

color:red;

text-align:center;

}
```

CSS Comments

- Comments are used to explain your code, and may help you when you edit the source code at a later date/time.
- Comments are ignored by browsers.
- A CSS comment begins with "/*", and ends with "*/",

CSS Comments

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
```

CSS Id and Class

- In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called **"id" and "class".**

The id Selector

- The id selector is used to specify a style for a single, unique element.
- The id selector uses the id attribute of the HTML element, and is defined with a "#".
- The style rule below will be applied to the element with id="para1":

Example

```
#para1

{

text-align:center;

color:red;

}
```

Note: Do NOT start an ID name with a number! It will not work in Mozilla/Firefox

The class Selector

- The class selector is used to specify a style for a group of elements.
- Unlike the id selector, the class selector is most often used on several elements.
- This allows you to set a particular style for many HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a "."

Example

```
.center {text-align:center;}
```

You can also specify that only specific HTML elements should be affected by a class.

In the example below, all p elements with class="center" will be center-aligned:

Example

```
p.center {text-align:center;}
```

Note: Do NOT start a class name with a number! This is only supported in Internet Explorer.