COMP 4021

Internet Computing

Cascade Style Sheets (CSS)

Why Cascading Style Sheets (CSS)?

- CSS separates visual parameters (color, spacing, etc.) from actual content
  - Define a style rule containing a set of style parameter/value settings
  - Elements which want to use the style setting can "import" the rule
- You have already seen how style can be used for individual elements:

```
<p style="color:red; background-color:yellow; font-size:46pt;
   font-style:italic">A pretty paragraph.</p>
```

- But what if you want the same style to be used for *all* paragraphs in the web page?

CSS is Not just for HTML

- CSS can be applied to all XML based languages, i.e., tags with element names:
  - XML and any XML based language:
    - XML
    - HTML
    - SVG
    - MathML
    - ChemML
    - And so on…
- Just associate an Element with a style rule, whether the element is an HTML or SVG element does not matter

Typical Style Properties

- Style parameters that can be controlled with CSS:
  - Text font
  - Text size
  - Text colour
  - Background colour
  - Background image
  - Margins
  - Padding (space between element and margins)
  - Borders (including colour, style, width)
  - Word spacing
  - Letter spacing
  - Text decoration (such as underline and blink)
  - Vertical alignment
  - Control over capitals (upper case, lower case)
  - Text indentation
  - List styles (many parameters)

How Styles are Connected to Content?

Inline: Embed Style in Elements

- Inline style:

  ```
  <h1 style="font-size:48pt; font-family:Arial; color:red;" >
    This is My Report</h1>
  ```

- These style parameters will apply only to this single instance of h1, not to other instances of h1
- What if you want the same visual information to be used for *all* paragraphs in the web page?
- Inline style is ‘bad’; CSS provides a *central* set of style rules that can be easily applied to sets of elements
- A web site designer wants to find all style setting in the <style> section or a separate "style" file so the ‘look and feel’ can easily be changed

Using CSS Style Rules

- A style rule:

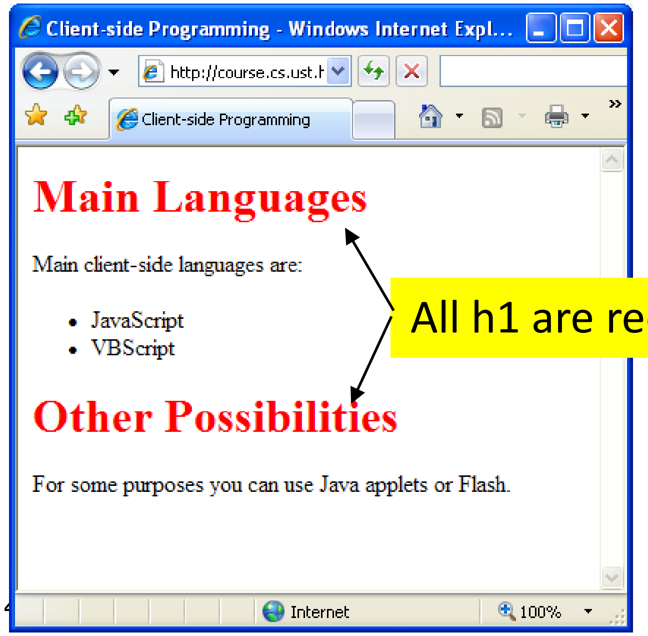
$$\overbrace{\text{h1}}^{\textit{Selector}}\ \{\ \underbrace{\overbrace{\text{color:}}^{\textit{Property}}\ \ \overbrace{\text{red}}^{\textit{Value}}\ \}}_{\textit{Declaration}}$$

- You can define a rule for:

	Selector Syntax	Examples
Element Type	Element_name	h1, div, p
Element ID	#ID	#myDiv
Class	.class_name	.highlight

Embedded/Internal CSS

- Styles are embedded in the HMTL document; typically put inside the <head> element
- Still not very good: You need to repeat the styles in each HTML pages that use them!



Main Languages

Main client-side languages are:

- JavaScript
- VBScript

Other Possibilities

For some purposes you can use Java applets or Flash.

All h1 are red

```
<html>
<title>Client-side Programming</title>
<style>
h1 { color: red }
</style>
<body>
<h1>Main Languages</h1>
<p>Main client-side languages are:</p>
<ul>
<li>JavaScript</li>
<li>VBScript</li>
</ul>
<h1>Other Possibilities</h1>
<p>For some purposes you can use Java
    applets or Flash.</p>
</body>
</html>
```

All h1 in the web page will be red

External Style File

- Styles are put in a separate “CSS” files

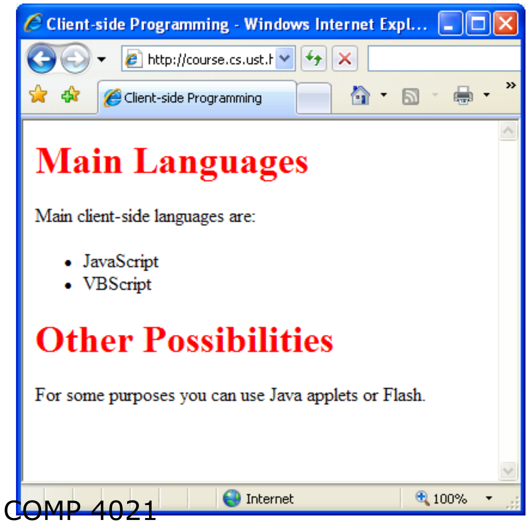
File: my_style.css

```
h1 { color: red }
```

File: css_simple.html

```
<html>
<title>Client-side Programming</title>
<link rel="stylesheet" href="my_style.css"
          type="text/css"/>
<body>
<h1>Main Languages</h1>
<p>Main client-side languages are:</p>
<ul>
<li>JavaScript</li>
<li>VBScript</li>   </ul>
<h1>Other Possibilities</h1>
<p>For some purposes you can use Java applets
or Flash.</p>
</body> </html>
```

- The visual result is the same as before:



Main Languages

Main client-side languages are:

- JavaScript
- VBScript

Other Possibilities

For some purposes you can use Java applets or Flash.

ID Selector

- Define a rule for a particular element using element **ID**, e.g.,

```
#big_title { font-size: 48pt;
             font-family: Arial; color: red }

<h1 id="big_title">My Report</h1>

<h2 id=“small_title”>Conclusion</h2>
```

Class Selector

- Create a rule for a **class** of heterogeneous elements (having different element names):

  .zappy { font-weight: bold; font-family: Impact; color: blue }

  The rule will be applied to both of the followings:
  <p class="zappy">Hi! This is my zappy style!</p>

  <div class="zappy">My name is Zebedee!</p>
- Class can be restricted to a particular set of elements:

  p.zappy { … declaration …}

  div.zappy { … declaration … }
  - p.zappy is applied to <p class="zappy"> …
  - div.zappy is applied to <div class="zappy">…

Nice Way to Style a Div

- Typically you would first define the style information for a div (such as the position and colours):

```
<style type="text/css">

    .layer_style1 {
        position:absolute; top:20px; left:5px;
        color:#CC00EE; width:200px;
    }

  </style>
```

A style class is created. Note the dot before the class name

Declaring the Div

- The div is defined using the style rule:

Style class created in the last slide is used

```
<div id="layer_name1" class="layer_style1“ >
     <h1>Layer 1</h1>
     <p>Content for layer 1 goes here.</p>
</div>
```

Pseudo Classes

- An element can go through several “states”: the initial state when nothing has happened yet, then it could be in the “hover”, “active” states, etc.
  - States are properties or pseudo classes of an element
- Which pseudo class an element is in (e.g., whether a link has been visited) is determined by the browser
  - It is not defined by web designer with CSS such as <p class=“blue” …>
- Pseudo class can be used with most elements

Pseudo Class Examples

- Style anchor text to distinguish it from normal text

```
A:link        { text-decocoration: underline }  // not visited
A:visited     { text-decocoration: none}  // visited
A:hover       { background: black}      // mouse above text
A:active      { blackground: yellow}  // mouse button down
```

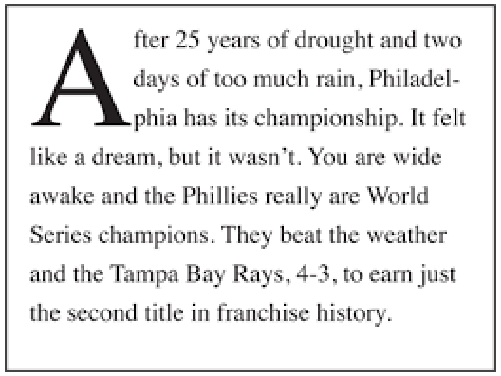
Pseudo-class names

See Demo

Pseudo Elements

- Pseudo Elements refer to parts of an element's content, e.g.,
  - FIRST-LETTER and FIRST-LINE
- To achieve the "drop letter" effect:

  <span class="drop-letter">A</span>fter …
  and define the style for <drop-letter>
- Using CSS built-in pseudo element:


After 25 years of drought and two days of too much rain, Philadelphia has its championship. It felt like a dream, but it wasn't. You are wide awake and the Phillies really are World Series champions. They beat the weather and the Tampa Bay Rays, 4-3, to earn just the second title in franchise history.

Class name (defined by a rule not shown here)

$\overbrace{\text{P.INITIAL}}\underbrace{\text{::FIRST-LETTER}}_{\text{Pseudo-class names}}\text{ \{ font-size: 200\%; }\underbrace{\text{float:left}}_{\text{Display at the left of the parent element (i.e., P)}}\text{\}}$

See Demo

Other Pseudo Elements

```
<style>
p::after {
  content: " new!";
  font-size: 80%;
  font-weight: bold;
  color: red;
}
</style>

<h1>What would p::after do to <p>?</h1>
<p>Cascade Style Sheet can be very powerful</p>
```

CSS For HTML

- Style parameters that can be controlled with CSS:
  - Text font
  - Text size
  - Text colour
  - Background colour
  - Background image
  - Margins
  - Padding (space between element and margins)
  - Borders (including colour, style, width)
  - Word spacing
  - Letter spacing
  - Text decoration (such as underline and blink)
  - Vertical alignment
  - Control over capitals (upper case, lower case)
  - Text indentation
  - List styles (many parameters)

Setting Multiple Attributes in a Rule

```
h1 {color: maroon;
        font: italic 1em Times, serif;
        text-decoration: underline;
        background: yellow url(titlebg.png) repeat-x;
        border: 1px solid red; padding: 5px; }
```

- All h1 in the web page will use dark red, the most commonly used font for paper, is italicized, is underlined, has a background image that is repeated horizontally (not vertically) but will use yellow for the background image if the image cannot be loaded, uses a 1 pixel red border that is separated from the text by 5 pixels

CSS - Large Example 1/3

```
<head><title>Basic CSS Example</title>

<style type="text/css">
body          {background-color: black;}
h1            {font-size: 24pt;
              font-family: Comic Sans Ms, Cursive;
              text-align: center;}
p             {font-family: Arial, Sans-serif; font-size: 16pt;
              line-height: 100%;
              text-align: justify;
              text-indent: 20px;}

#letterspace{letter-spacing: 3px;}
```

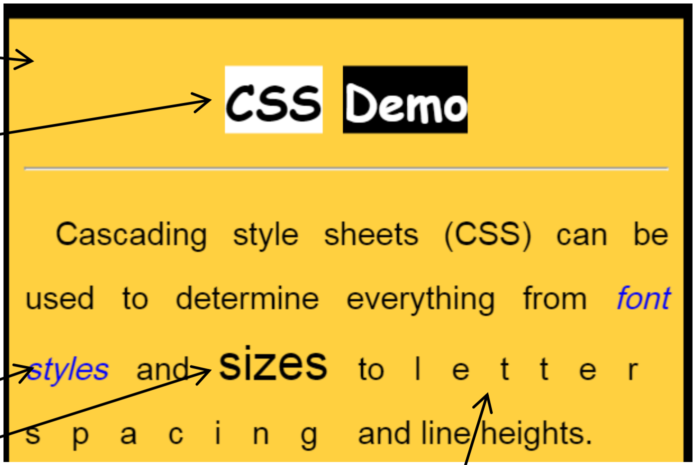
CSS - Large Example 2/3

```
.blackonwhite        {color: black;
                   background-color: white;}
.whiteonblack        {color: white;
                   background-color: black;}
.style            {color: blue; font-family: Arial; font-style:oblique;}
.size              {font-size: xx-large;}
.lineheight        {line-height: 500%;}

/* Define class "page", which applies only to div */
div.page           {background-color: #FFD040; color: black;
                   margin: 50px 10px 50px 10px;
                   padding: 10px 10px;
                   width: 90%; height: 90%;}
</style> </head>
```

CSS – Large Example 3/3

```
<body>
<div class="page">
 <h1>
    <span class="blackonwhite">
          CSS</span>
    <span class="whiteonblack">
          Demo</span>
 </h1>
<hr/>
<p>Cascading style sheets (CSS) can be used to control everything
    from <span class="style">font styles</span> and <span
    class="size">sizes</span> to <span id="letterspace">letter
    spacing</span> and <span class="lineheight">line
    heights</span>.</p>
</div>
 </body></html>
```


CSS Demo

Cascading style sheets (CSS) can be used to determine everything from *font styles* and **sizes** to l e t t e r s p a c i n g and line heights.

Demo

Try to map HTML codes to the display
Why there is a dark border?

Applying a Rule to Multiple Tags

```
h1 { background: yellow; color: blue }
h2 { background: yellow; color: blue }
h3 { background: yellow; color: blue }
```

- The above can be more efficiently written as

```
h1, h2, h3 { background: yellow; color: blue }
```

Applying a Rule to Multiple Tags

```
h1 { background: yellow; color: blue; font: 24pt; }
h2 { background: yellow; color: blue; font: 20pt; }
h3 { background: yellow; color: blue; font: 16pt; }
```

- The above can be more efficiently written as

```
h1, h2, h3 { background: yellow; color: blue}
h1 { font: 24pt; }
h2 { font: 20pt; }
h3 { font: 16pt; }
```

- One rule sets the common properties for all three tags
- An individual rule tailors the font size of each tag
- Two rules are defined for the same tag

CSS in HTML5

- CSS is already a powerful language, HTML5 makes it more powerful to meet the imagination of all users
- Standardization of separation of CSS into modules
- More selectors: E::nth-child(n), E::not(s)
- Color: saturation, lightness, alpha-channel
- Background and Borders: stretch a background image, box shadow, rounded box corners
- Multi-column layout
- @media rules: display size, color depth, aspect ratio

What CSS can do to a DIV?

- Rounded box corners
- How to turn a DIV into a circle?
- What about a triangle?

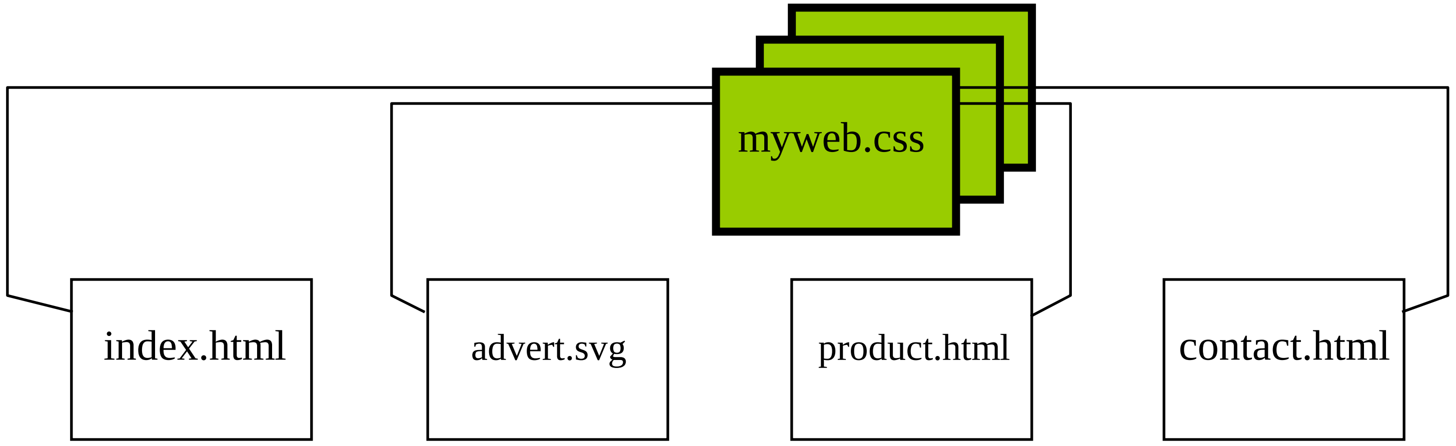
- I would rather draw the actual triangle with SVG (later lectures)!!!
- There are many CSS purists who try to do everything with CSS
- CSS becomes very complexity (and powerful)!!!

```
<style>
div {
    width: 100px;
    height: 100px;
    border-radius: 20%;
    background: red;
    align: center;
}
</style>
<div></div>
```

Demo

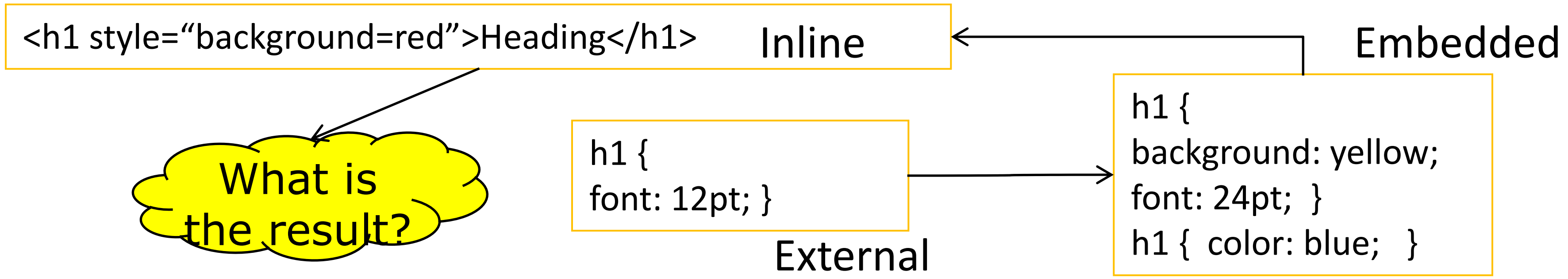
Typical Web Site Usage

- CSS means that a complete set of web pages can be developed which all point to the same CSS files
- The files can even have different languages (i.e. SVG, XHTML) all pointing to the same style information



Why is CSS called "Cascading"?

- Styles in multiple rules defined on the same element are merged
- When two rules conflict, prioritize them (1 highest; 5 lowest):
  1. Inline styles: style attribute included within a tag
  2. Embedded style: CSS rules inside the HTML itself
  3. External style sheets: CSS files referenced from the HTML itself
  4. User style: Local CSS file specified by the user on the browser
  5. User agent style: browser's default style sheet
- Given two identical embedded rules, the LAST takes precedence



```
<h1 style="background=red">Heading</h1>
```

Inline

```
h1 {
font: 12pt; }
```

External

```
h1 {
background: yellow;
font: 24pt;  }
h1 {  color: blue;   }
```

Embedded

Website Advantages

- Separation of contents and styles
- Styles can be separately managed by visual designers
  - Facilitate global controls and updates to styles
  - Cascade allows local overwrite of styles
- Every page has a consistent ‘look and feel’
- Style sheet can be altered, result is immediately seen across whole web site - for example, web site can have a different look and feel for Chinese New Year, then later change back
- Easier for debugging/ handling (just one set of style files controls everything)

Take Home Message

- CSS separates content and style, making webpages easier to read and maintain, which is the major goal in content management systems (CMS)
- How to identify which subset of elements a rule applies to?
  - Powerful "selectors" make selecting DOM elements easy
  - In jQuery, we will learn more CSS selectors
- CSS is much more powerful than covered here
  - CSS goes beyond styling to include animation, 2D/3D transformation