Creating and Building Websites

Stanford University Continuing Studies CS 21

Mark Branom
branom@alumni.stanford.edu

Course Web Site: http://web.stanford.edu/group/csp/cs21

Week 6 Agenda

- Unfinished business
- Cascading Style Sheets

CSS Review – why called CSS?

- Cascading
  - Rules are interpreted in a top-to-bottom, cascading fashion.
  - Rules that aren't conflicting are aggregated (e.g., `body { background: red; color: white; }` produces a background of red with a font color of white.
  - For rules that conflict, last one generally "wins" (e.g., `{background: red; background: white}` produces a background of white, not red.
- Style
  - Refers to the look and feel
- Sheet
  - Set of rules
  - External, Internal, or Inline

CSS Review: Sheet Types (External)

- External
  - Rules are listed in a separate text file (e.g., style.css)
  - Web pages are attached to the style using the <link> or the @import method:

```
<link href="URL-to-style-sheet-file.css"
rel="stylesheet" type="text/css"
media="all | print | only screen and
(max-width: 640px)">

<style type="text/css" media="all | print | only
screen and (max-width: 640px)">
@import url("URL-to-style-sheet-file.css")
</style>
```

CSS Review: Sheet Types (Internal)

- Internal
  - Rules are listed in the web page itself (usually in the <head>)

```
<head>
…
<style type="text/css">
   body { background: pink;}
</style>
…
</head>
```

CSS Review: Sheet Types (Inline)

- Inline
  - Rules are listed in the element (“tag”) itself

```
<p style="text-align: center;">
     paragraph that's centered
</p>
```

  - Generally used only in email newsletters and computer scripts

CSS Review: CSS Rules

- Rules are defined as a selector and declaration:
- selector { property: value; }

$$\uparrow$$

“declaration”

Comments

- CSS Comments:

```
/* Enter comment here */

/****************
  Main section
**************/
```

- HTML Comments:

```
<!-- enter comment here -->
```

CSS Selectors: Redefining Elements

- You can use CSS to redefine an element (“tag”):

```
body { background: pink;}

p { font-family: Verdana;}
```

CSS Selectors: Create Own Style (Class/ID)

- You can use CSS to create your own style:

```
.class { property: value; }

#id { property: value;}


 .highlight { background: #eee; }
 #headerarea { border: thin dotted #000; }
```

CSS Selectors: "Advanced" "Stuff": Multiple Styles/Selectors

- Multiple styles – Each rule can include multiple styles, using semicolons to separate them.
  - `h2 {color: darkblue; font-style: italic;}`
- Multiple selectors that have the same styles can be grouped, using commas to separate them.
  - `h1, h2, h3 {font-style: italic;}`

CSS Selectors: “Advanced” “Stuff”: Contextual Selectors

- Contextual selectors allow you to specify that something will occur only when it is used in conjunction with something else.
- In the style below, em will display in red, but only when it occurs within li within ul:
  - `ul li em {color: red;}`
- Elements being modified by contextual selectors need not appear immediately inside one another. For example, using the style above with the HTML below, blah would still be red text:
  - `<ul><li><strong><em> blah </em></strong></li></ul>`

CSS Selectors: "Advanced" "Stuff": Direct Child Selectors

- Direct child selectors allow you to specify that something will change, but only when immediately inside another element.
- With the style below, only those strong elements that are directly inside h1 will be purple font; no strong tags deeper within the sheet will be purple:
  - `h1 > strong {color: purple;}`

CSS Selectors: “Advanced” “Stuff”: Adjacent Selectors

- Adjacent selectors allow you to specify that something will change only when preceded by something else.

  In the style below, only those links that are preceded by an h2 will be green:
  - `h2 + a {color: green;}`
- Elements being modified by adjacent selectors appear immediately after one another.
- Using the style above, this link would be green:

  -
    ```
    <h2>Visit Stanford!</h2>
    <a href="http://www.stanford.edu">click here</a>.
    ```
- But this link would not:

  -
    ```
    <h2>Visit Stanford!
    <a href="http://www.stanford.edu">click here</a></h2>.
    ```

CSS and Fonts

- Some of the things you can do to change fonts in CSS:
  - font-family
    ```
    p { font-family: Verdana;}
    ```
  - font-style
    ```
    h1 { font-style: italic;}
    ```
  - font-weight
    ```
    .important { font-weight: bolder;}
    ```
  - font-size
    ```
    body { font-size: large;}
    ```
  - font-variant
    ```
    em { font-variant: small-caps;}
    ```
  - text-shadow
    ```
    .cooltext { text-shadow: #f00 3px 3px 5px; }
    ```

Colors/Backgrounds

- Colors:
  - 6 digit hex code: `#ff0000`
  - 3 digit hex code: `#f00`
  - rgb method (integer/percentage): `rgb(255,0,0)` or `rgb(100%,0%,0%)`
  - keyword color name: `red`

- Backgrounds:
  - `body { background-color: #f00; }`
  - `table { background-image: url("/images/bg.gif"); }`

Span/Div

- The <span> tag is an inline element and is used to apply style to a small amount of content.
  - `Only the word <span class="hightlight"> highlight</span> is highlit.`

- The <div> tag is a block element and is used to apply style to a large amount of content (e.g., regions of a page).

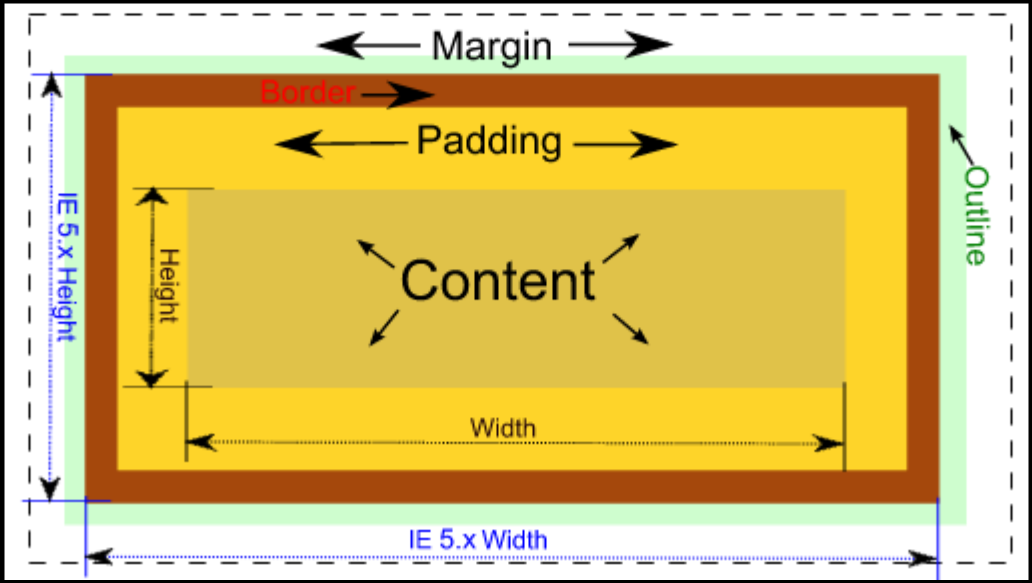
  -
    ```
    <div id="footerarea">
      [ footer content ]
    </div>
    ```

Pseudo-elements, Pseudo-classes, and Generated Content

- Pseudo-elements:
  - To affect the first letter, use the style `:first-letter`
  - To affect the first line, use the style `:first-line`
- Pseudo-classes:
  - To affect a link, use the style `a:link`
  - To affect a visited link, use the style `a:visited`
  - To affect an active link, use the style `a:active`
  - To affect a hovered link, use the style `a:hover`
- Generated Content:
  - To make content appear before an element, use the style :before with the property content
    - `a.msw:before { content: url('msword.gif'); }`
  - To make content appear after an element, use the style :after with the property content
    - `a.acrobat:after { content: url('adobeacrobat.gif'); }`

Box model

- When a browser draws an object on a page, it places it into an invisible rectangular space called a "bounding box."
- You can specify the size, look, and feel of the margins, the padding, the border, the outline, and the content of that bounding box.



Making content "centered"

- To make a block centered, you need to first give it a width, and then auto-margin the left and right margins:

  -
    ```
    #main { width: 80%;
            margin-left: auto;
            margin-right: auto; }
    ```

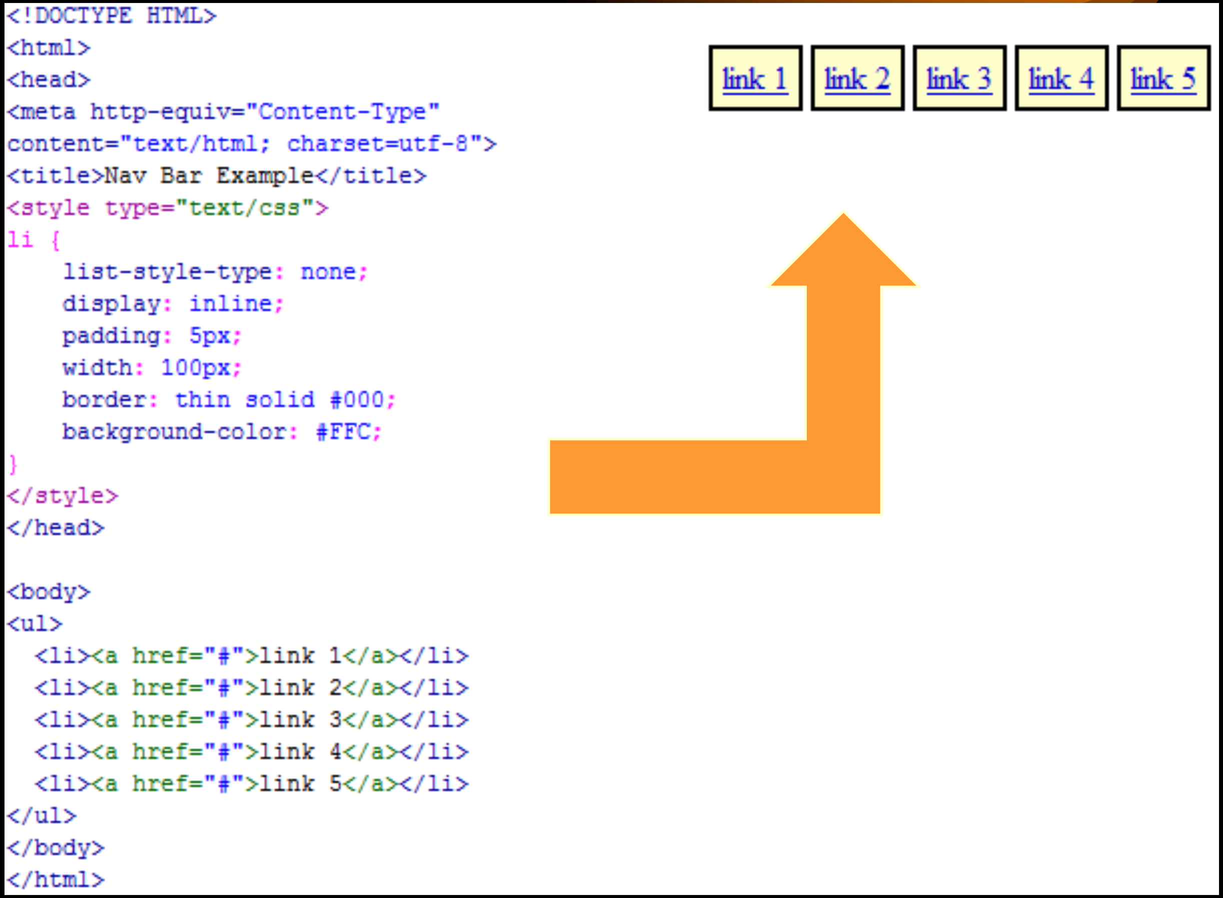
  -
    ```
    nav#mainnavigation
          { width: 50%;
            margin: 0px auto 0px auto;}
    ```

Block vs. Inline

- Elements are either `block` or `inline`.
- Using CSS, you can change their default status:
  - `em { display: block; }`
  - `p { display: inline; }`
- You can even make them disappear:
  - `.hidden { display: none; }`

Creating a Nav Bar

- One way to create a nav bar using a list of links and CSS:



```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
<title>Nav Bar Example</title>
<style type="text/css">
li {
    list-style-type: none;
    display: inline;
    padding: 5px;
    width: 100px;
    border: thin solid #000;
    background-color: #FFC;
}
</style>
</head>

<body>
<ul>
  <li><a href="#">link 1</a></li>
  <li><a href="#">link 2</a></li>
  <li><a href="#">link 3</a></li>
  <li><a href="#">link 4</a></li>
  <li><a href="#">link 5</a></li>
</ul>
</body>
</html>
```

Absolute Positioning

- You can also use CSS to define exactly where you want an object to be located, using the property position with a value of absolute:

```
#headersection { position: absolute;
                 top: 0px; left: 0px;
                 width: 100%; }
```

Fixed Positioning

- You can also use CSS to keep content static and never move:

```
#linksarea { position: fixed;
             top: 20px;
             left: 0px;
             width: 200px;
             height: 400px; }
```

Layering

- If two objects are positioned using *absolute* or *fixed* positioning, you can layer one on top of the other. To determine which is on top, use the property *z-index*. The rule with the higher z-index will be on top.

- In this example, #box1 is on top of #box2, even though #box2 comes after #box1 in the code:

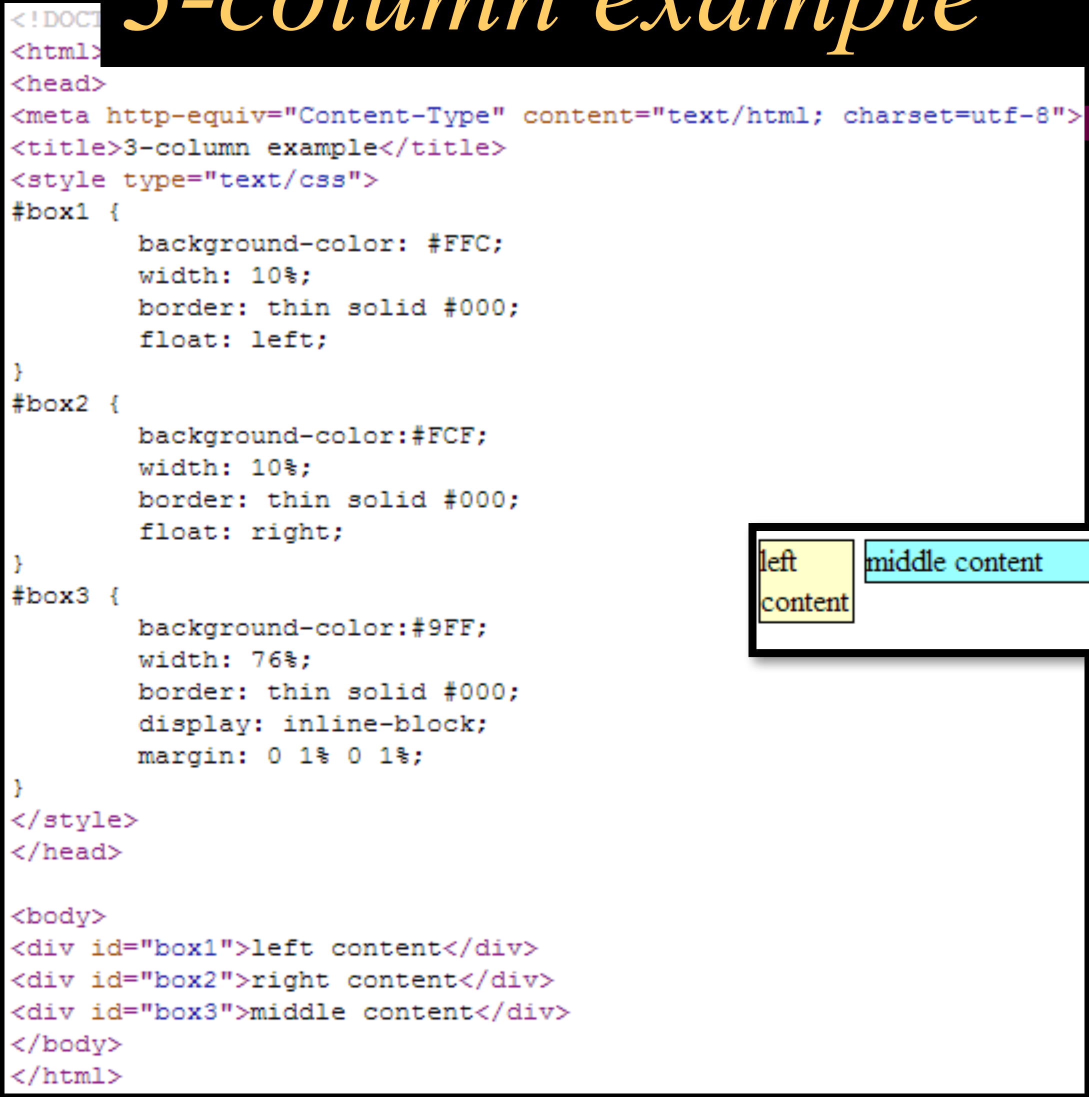
```
#box1 { position: absolute; bottom: 0px;
        left: 0px; z-index: 5; }

#box2 { position: absolute; bottom: 3px;
        left: 3px; z-index: 1; }
```

Float/Clear

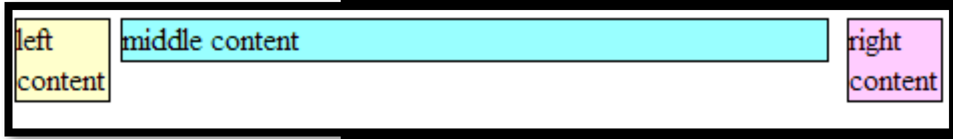
- You can also have content float to the right or left of another object.
  - `float: left;`
  - `float: right;`

- To end a float, use the *clear* property:
  - `clear: right;`
  - `clear: left;`
  - `clear: both;`

3-column example



```
<!DOCT
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>3-column example</title>
<style type="text/css">
#box1 {
        background-color: #FFC;
        width: 10%;
        border: thin solid #000;
        float: left;
}
#box2 {
        background-color:#FCF;
        width: 10%;
        border: thin solid #000;
        float: right;
}
#box3 {
        background-color:#9FF;
        width: 76%;
        border: thin solid #000;
        display: inline-block;
        margin: 0 1% 0 1%;
}
</style>
</head>

<body>
<div id="box1">left content</div>
<div id="box2">right content</div>
<div id="box3">middle content</div>
</body>
</html>
```



Next week…

- Forms
- Basic CGI programming