Chapter 4: Introduction to CSS

CSS Defined:

- Short for "Cascading Style Sheets".
- Determines how the elements in our XHTML documents are displayed and formatted.
- Designed to separate the content of a web page from the presentation of that content.
- Enables us to make all pages of our website look similar and consistent (font, color, etc.).
- Allows us to make site-wide formatting changes from a single location (rather than having to edit each page individually).

- *Initial few slides are taken from CSS slides uploaded by Grace Bautista Ursua on Scribd:* https://www.scribd.com/user/41316337 4/Grace-Bautista-Ursua

Three Ways to Use CSS:

1) Inline Style - CSS is placed directly into the HTML element.
2) Internal Style Sheet - CSS is placed into a separate area within the <head> section of a web page.
3) External Style Sheet - CSS is placed into a separate computer file and "connected" to a web page.

CSS Format Conflicts:

- It's possible for CSS formatting to be defined in all three locations at the same time.
- For example, a paragraph element could contain an inline style (color:red) but the internal style sheet (color:blue) and the external style sheet (color:green) give conflicting instructions to the web browser.
- Web browsers need a consistent way of "settling" this disagreement.
- Within this cascade of style declarations, the closest rule wins.
- An inline style overrules an internal style, which overrules an external style.

What is Meant by "Cascading"?

- We use the term cascading because there is an established order of priority to resolve these formatting conflicts:
  1) Inline style (highest priority)
  2) Internal style sheet (second priority)
  3) External style sheet (third priority)
  4) Web browser default (only if not defined elsewhere)

Example: Inline Style

```
<h2 style="font-family:georgia; color:red;">
CAUTION: Stormy Weather!
</h2>
```

PREVIEW:

A semicolon must follow each style declaration.

Example: Internal Style Sheet

```
<head>
<style type="text/css">
h2 {font-family:georgia; color:red;}
</style>
</head>
```

- For internal style sheets, all formatting declarations are placed inside the <style> element within the <head> section of the document.
- An element is listed and all the styling information follows, surrounded by opening and closing curly brackets, { }.
- A semicolon must still follow each style declaration.

Example: External Style Sheet

```
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

style.css (separate file):

```
h2 {font-family:georgia; color:red;}
```

- For external style sheets, a <link> tag is placed at the beginning of the <head> section of the document specifying the external style sheet (with a .css extension) to be used for formatting.
- The external style sheet uses the same syntax as the internal style sheet when listing elements and their styling.
- Styles declared in an external style sheet will affect all matching elements on all web pages that link to the stylesheet.
- In this example, all <h2> elements on all pages using this style sheet will be displayed in Georgia font and in red color.

Internal vs. External Style Sheets

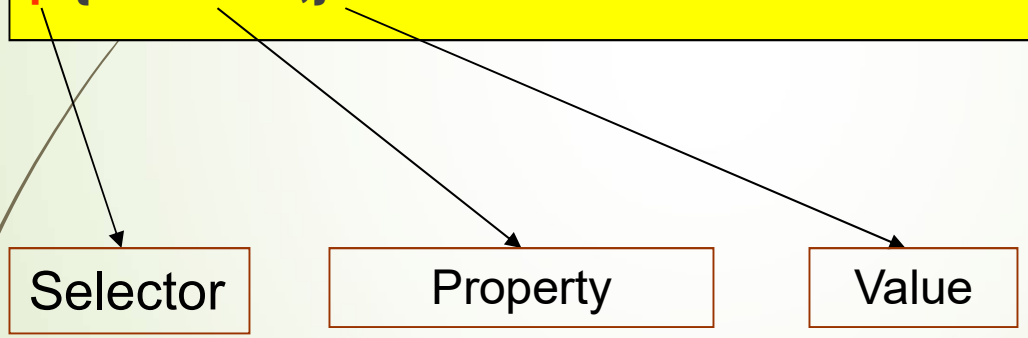
- Internal style sheets are appropriate for very small sites, especially those that have just one page.
- Internal style sheets might also make sense when each page of a site needs to have a completely different look.
- External style sheets are better for multi-page websites that need to have a uniform look and feel to all pages.
- External style sheets not only make for faster-loading sites (less redundant code) but also allow designers to make site-wide changes quickly and easily.

CSS Terminology and Syntax:

Correct syntax:

selector {property:value;}

```
p {color:red;}
```



Some Examples

Background Picture

```
body {
   background-image:url('picture.gif');
   background-repeat:repeat-x;
   background-color:red;
 }
```

Paragraph Properties

```
p {
   color:red;
   font-style:italic;
   text-align:center;
}
```

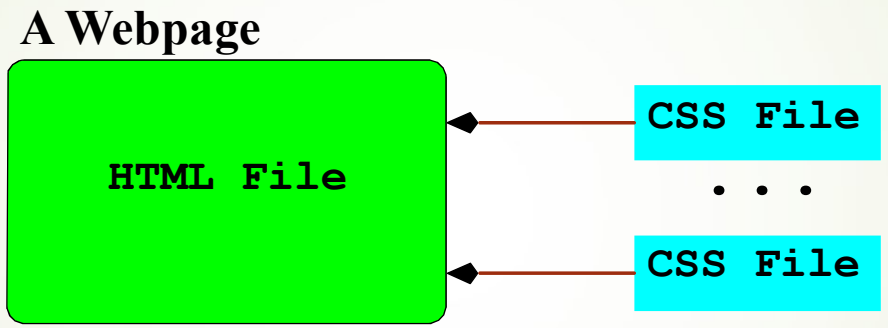
CSS Text Properties:

The following properties can be specified for any element that contains text, such as <h1> thru <h6>, <p>, <ol>, <ul>, and <a>:

Property	Some Possible Values
text-align:	center, left, right, justify
text-decoration:	underline, line-through, blink
color:	blue, green, yellow, red, white, etc.
font-family:	Arial, Verdana, "Times New Roman"
font-size:	large, 120%, 20px (pixels)
font-weight:	bold, normal
font-style:	italic, normal

HTML + CSS = Webpage

A webpage consists of two basic parts: HTML5 code and CSS code.

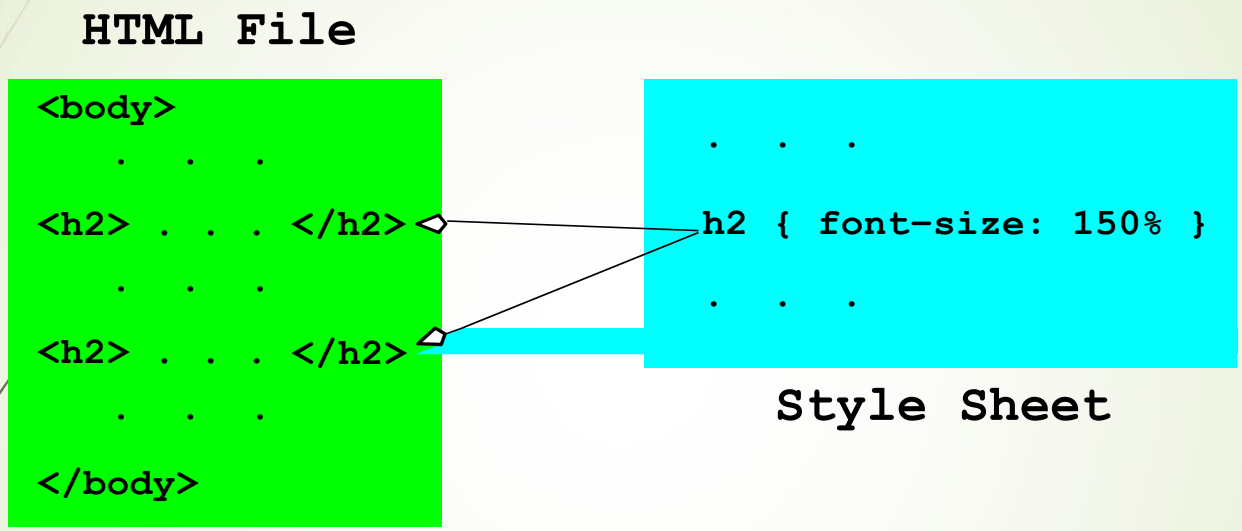


- Listings of CSS properties are available on the Web.
- Many properties can be associated with all HTML elements when appropriate. Some, such as `text-align`, apply only to block elements. Others, such as `vertical-align`, apply only to inline elements.
- A few styles apply only to specific elements. For example, `list-style-type` is only for list items. CSS documents the applicability of each style property.
- Browsers provide default presentation styles to all HTML elements. By associating your own style declarations, you can control the presentation style of an entire page and how any element in it is displayed.

Style Sheets

- A set of style rules placed in a file (usually with the `.css` suffix) become a style sheet.
- In a style sheet, comments may be given between `/*` and `*/`.
- A style sheet may also contain *at-rules* for including other style sheets, indicating media targets, and so on.

Attaching a Style Sheet



Whole-Page Styling

- Because most style properties set for the `body` element apply to the entire page through inheritance, CSS makes whole-page styling easy to enforce.

```
body
{   font: small Verdana, Geneva, Arial,
          helvetica, sans-serif;
    color: black;
    background-color: white;
    border: 0px; padding: 0px;
    margin: 0px   0px   30px   0px;
    /*       top right bottom left       */
```

The `font` Property

The `font` style property allows you to specify all font-related properties in one place in the general form:

`font`: *style variant weight size* / *line-height family*

Only the *size* and *family* are required.
Normally, `line-height` is 120% of `font-size`. To improve readability of textual materials on screen, we recommand

```
h1, h2, h3, p, li { line-height: 150% }
```

to set line spacing to 1.5 × `font-size` for these elements.

Centering

```
text-align: center

<h2 class="center">Topic of The Day</h2>

h2.center { text-align: center; color: #006600 }

h1.center, h2.center, h3.center, h4.center,

h5.center, h6.center

{ text-align: center; color: #006600 }

.center { text-align: center; color: #006600 }

<h1 class="center">Topic of The Day</h1>

<h3 class="center">Lunch Menu</h3>

<p class="center">Some text</p>
```

To center a block element with a fixed width or a table, use the style rules

```
margin-left: auto; margin-right: auto
```

Demo: **Ex:** `CenterStyle`

HTML Class Attribute

- The class attribute specifies one or more classnames for an element.
- The class attribute is mostly used to point to a class in a style sheet.
- The general form of a class selector is

  element . *class*
- The rule matches *element* in that class. If *element* is omitted, then the selector matches all elements in that class.

https://www.w3schools.com/tags/att_class.asp

Indenting

```
p { text-indent: 3em }

p.abstract { margin-left: 5em;
             margin-right: 5em }
```

Multicolumn Layout

```
body
{  margin: 50px;
   column-count: 2;
   column-gap: 2em;
   column-rule: thin solid black; }
```

Within a multicolumn layout, child elements are flowed from one column to the next automatically. But, you can set the style `column-span: all` for a child element for it to span all columns.

```
h1 { column-span: all }
/* the only other value 1, the default */
```

Demo: **Ex:** TwoColumn

CSS Selectors

Type selector

A type selector is the simplest selector. It specifies an HTML element tag name and associates the rule with every instance of that element in the HTML document. For example, the rule

```
h3 { line-height: 140% }
```

Universal selector

The symbol * used as a selector selects every HTML element, thus making it simple to apply certain styles to all elements, all element in a class (*.*class*) or all child/descendant elements.

Class selector

The *.className* selector selects elements in the named class. An element is in class *xyz* if its `class` attribute contains the word *xyz*. For example,

```
.cap { text-transform: uppercase }
```

makes elements in the `cap` class ALL CAPS. And

```
.emphasis { font-style: italic; font-weight: bold }
```

makes the attribute `class="emphasis"` meaningful for many elements.

Id selector

The #*idName* selector associates the rule with the HTML element with the unique `id` attribute *idName*. Hence, the rule applies to at most one HTML element instance. For example,

```
#mileageChart{ font-family:Courier, monospace; color:red }
```

applies to `<table id="mileageChart"> ... </table>` only.

Concatenated (conjunction) selector

When a selector is the concatenation of two or more selectors, it selects elements satisfying each and every selector included. For example,

`span.highlight`	(`span` elements in class `highlight`)
`nav.main.mobile`	(`nav` elements in class `main` and class `mobile`)
`table#mileageChart`	(`table` element with id `mileageChart`)

Selector Grouping

Selectors sharing the same properties can be grouped together in one rule to avoid repeating the same rule for different selectors. To group selectors, list them separated by commas. For example,

```
h1, h2, h3, h4, h5, h6 { color: blue }
```

Pseudo-class selectors

- The *pseudo-class* is a way to permit selection based on conditions at run-time or on the hierarchical structure in the document.
- The eight most widely used pseudo-classes are the selector suffixes:
- `:link` (a valid link)
- `:visited` (a visited link)
- `:hover` (mouse over element)
- `:active` (element being clicked)
- `:focus` (UI element gained focus),
- `:enabled` (UI element usable)
- `:disabled` (UI element unusable)
- `:checked` (element selected).
- `:target` selects target of an in-page link action by the user. For example,

```
section:target { border: thin solid black }
```

CSS Selector Examples

```
body { background-color: white }
```
Element

```
*.fine or .fine { font-size: x-small }
```
Universal + Class

```
h2.red { color: #933 }
```
Class

```
a.box:hover
{ border: #c91 1px solid;
  text-decoration: none; }
```
Pseudo-class in Class

```
p, ul, nl { line-height: 150%; }
```
Grouping

Link Styles

```
/* shaded blue for unvisited links */
a:link { color: #00c;  }
/* dark red for visited links */
a:visited { color: #300; }
/* when link is clicked */
a:active
{   background-image: none;
    color: #00c; font-weight: bold; }
/* when mouse is over link */
a:hover
{ background-color: #def; background-image: none; }
```

Sometimes it is useful to have different classes of links (e.g., external and internal links). In that case, you can use selectors in the form

```
a.external:link
a.external:hover

a { text-decoration: none }
/* removing underline */
```

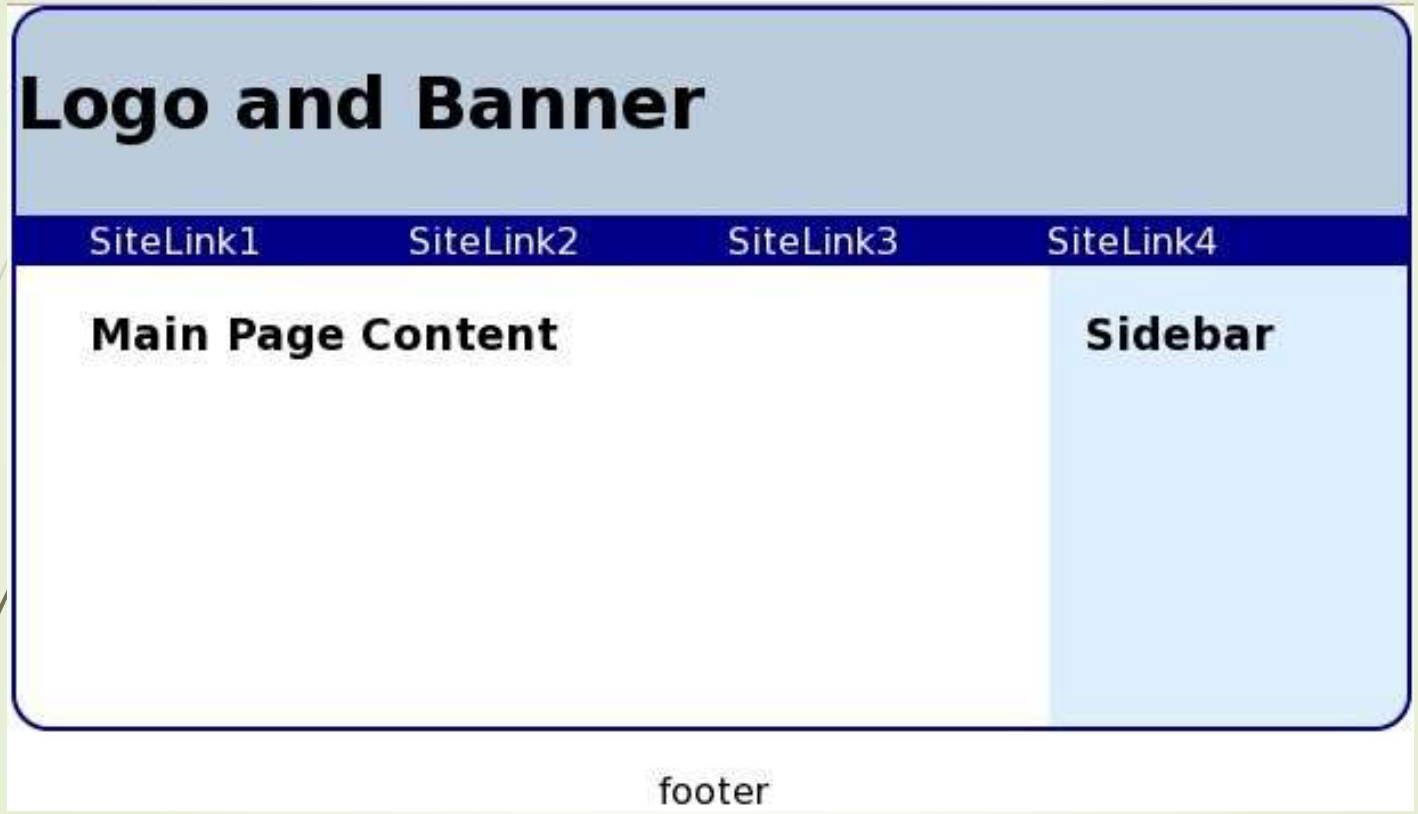
Webpage Layout with CSS

- A critical task in developing a new website is creating a good visual design and page layout.
- A layout grid is a set of invisible vertical and horizontal lines to guide content placement. It is the primary way designers organize elements in a two-dimensional space.
- A grid aligns page elements vertically and horizontally, marks margins, and sets start and end points for element placement.
- A consistent page layout also helps to create unity throughout the site.
- A fixed-width (or ice) layout can be easier to implement but a fluid layout that adjusts to varying page width and screen resolution is more desirable.

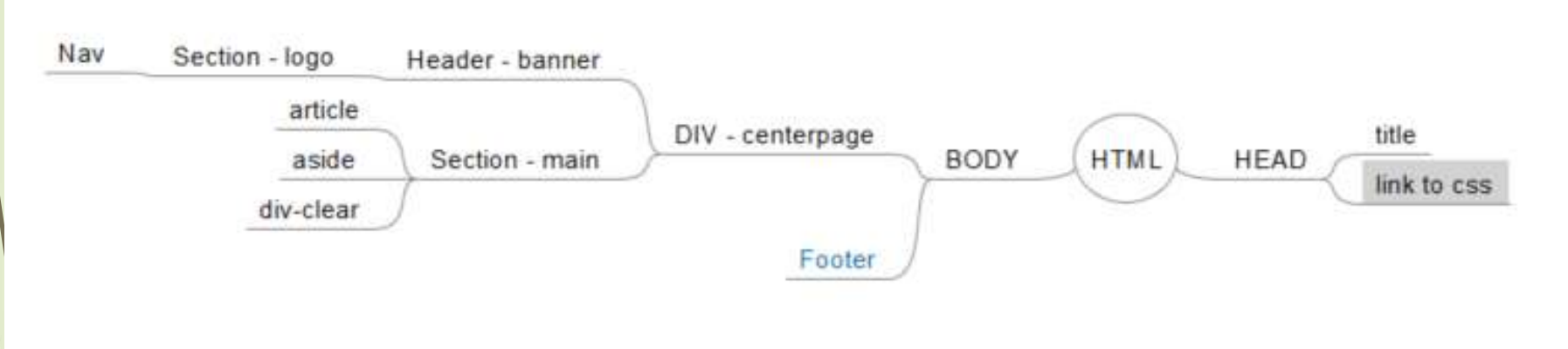
Fluid Float Layout

- An example that can be used as a template for a simple webpage.
- We'll look at the HTML page structure and the CSS styling for this example.

A Fluid Float Layout



Structure


Nav
Section - logo
Header - banner
article
aside
div-clear
Section - main
DIV - centerpage
Footer
BODY
HTML
HEAD
title
link to css

Fluid Float Layout Page Structure

1. A `header` for top banner and navbar
2. A `section` containing an `article` (for the main content), an `aside` (for the sidebar), and an empty `div` to end all previous floats.

```
<body id="top">
<div id="centerpage">
    <header class="banner">... </header>
    <section id="main">
        <article>... </article>
        <aside>... </aside>
        <div style="clear: both"></div>
    </section>
</div>
<footer>
<p style="text-align: center">footer</p>
</footer>
```

Fluid Float Layout CSS

```
div#centerpage
{/* centering */
  margin-left:auto; margin-right:auto;
 /* fluid page width */
  width: 80%;
 /* border */
  border: 2px solid darkblue;
 /* rounded corners */
  border-radius: 16px;
  overflow: hidden;
}
```

Top Banner HTML

```
<header class="banner">
<section class="logo">Logo and Banner
</section>
<nav> <a href="#">SiteLink1</a>
    <a href="#">SiteLink2</a>
    <a href="#">SiteLink3</a>
    <a href="#">SiteLink4</a></nav></header>
```

Top Banner CSS

```
header.banner { background-color: #bcd; }

header.banner > section.logo
{   font-size: xx-large; font-weight: bold;
    height: 60px; padding-top: 30px;
}
```

Navbar CSS


```
header.banner nav
{ background-color:darkblue;/* color of navbar */
padding-left: 2em; /* lead spacing */
white-space: nowrap /* links on one line */ }
header.banner > nav a:link
{ text-decoration: none; /* no underline */
color: white; /* links in white */
margin-right: 60px; /* spacing the links */ }
header.banner > nav a:hover
{ /* mouseover effect */
text-decoration: underline; }
```

Main Content CSS

```
section#main
{  overflow: hidden; background-color: #def; }

section#main > article
{  width: 69%; float: left;
   background-color: white;
   padding-left: 2em;     }

section#main > aside
{  float: left;
   margin-left: 1em;top-margin: 2em; }
```

Demo: **Ex:** FloatLayout