CSS3

Three Parts of a Web Page

- HTML: provides content and structure
- CSS: provides formatting/presentation
- JavaScript/jQuery: provides behavior/interaction

CSS

- Cascading Style Sheets affect the *appearance* of the page
- site design can be completely changed without modifying the HTML
- resource: www.w3schools.com/css/

A Very Simple Example

```
<head>
    <!-- ... -->
    <style>
    /* Styling for all paragraphs */
    p {
        font-family: Verdana, Helvetica, sans-serif;
        text-align: justify;
        background-color: lightblue;
    }
    </style>
</head>
```

Selectors

- we use a *selector* to specify what element(s) we want to style
- there are a number of selectors we can use, depending on the situation

Type	Description
element	add styling to all instances of an element (most general)
class	add styling to all elements in a class (somewhat specific)
ID	add styling to one element with a unique ID (most specific)

HTML Attributes for Styling

There are two HTML attributes that help us with styling.

Attribute	Description
ID	add styling to one element with a unique ID
class	add styling to a class of elements

```
<p id="intro">This is an introduction.</p>
<h2 class="mktg">Company Benefits<h2>
```

Element Selector

Style every occurrence of a given element type.

```
/* Make all level-1 headings red */
h1 { color: red; }

/* Make all paragraphs right-aligned */
p { text-align: right; }
```

ID Selector

Style the single element with the given ID. In the <style> section:

```
/* Display the site title twice as big as everything else */
#site-title { font-size: 200%; }

/* Display single paragraph containing legal stuff in all caps */
#legalese { text-transform: uppercase; }
```

And in the <body> section:

```
<div id="site-title">Joe's Awesome Site</div>
<!-- ... -->
<p id="legalese">Our lawyers made us write this...</p>
```

Class Selector

Style all elements in the given class. In the <style> section:

```
/* Display all warnings in red */
.warning { color: red; }
```

And in the <body> section:

```
<p class="warning">Do not pass Go.</p>
<p id="legalese">Our lawyers made us write this...</p>
<p class="warning">Do not collect $200.</p>
<p>Here is a perfectly ordinary (and classless) paragraph...</p>
<p class="warning">In fact, you owe $200.</p>
```

Styling Groups of Elements

Use commas to specify groups of elements to style.

```
h3, .warning, #fatal {
    color: red;
}
```

The CSS above will make the following elements red:

- all level-3 headings
- all elements that belong to class `warning`
- the element with ID `fatal`

Exercise C1: Selectors

1. make a copy of file `H1-structure.html`, and save it as `C1-selectors.html`
2. style the level-2 headings to be (color) 'green'
3. give the second level-2 heading an ID of 'sl2' and style that ID to be 'orange'
4. style the level-3 heading to be (text-decoration) 'underline'
5. give the first and third paragraphs a class of 'odd-para' and style only those paragraphs to be (text-align) 'justify'
6. (the words in parentheses above are the property names; they are described here)
7. validate the file with the CSS validator

Color

Colors can be specified in numerous ways. The most common ones are:

- named colors, such as `green` or `white` or `lawngreen`
- RGB (red/green/blue) triplets, specified like `rgb(0,255,0)` (for green)
- hexadecimal values, like `#ffff00` (for yellow)
- examples of the different ways are shown here

More Color

The most common properties involving color are:

Property	Description
`color`	the foreground color of the element
`background-color`	the background color of the element
`border`	a rectangular border around an element

Color Examples

The following color definitions:

```
#p1 { color: magenta; }
#p2 { color: rgb(255,0,0); } /* red */
#p3 { background-color: #48d1cc; } /* turquoise */
```

Will have the following effect:

This is paragraph with ID `p1`.

This is paragraph with ID `p2`.

This is paragraph with ID `p3`.

Color Picker

- except for the most basic colors (with names), how do we know what RGB or hex values to use?
- a color picker (like www.colorpicker.com) can help a lot
- it can also help us find colors that go well together

Fonts

- we can choose different font faces (families), sizes, etc. to apply to our text
- it's common to choose one font for all headings, and another font for all body text
- for each one we specify a desired font, as well as some "fallback" ones in case a font isn't found on the computer
- *web fonts* give us even more flexibility, using fonts loaded from other web sites, but we don't have time for that here...

More Fonts

Here are some examples:

Name	Font Type	Sample
Georgia	serif	The dog strolled through the woods
Verdana	sans-serif	The dog strolled through the woods
Courier	monospace	`The dog strolled through the woods`

Specifying a Font Family

```
h1, h2, h3 { font-family: Georgia, "Times New Roman", serif; }
p { font-family: Verdana, Helvetica, sans-serif; }
```

- for level 1-3 headings, first Georgia will be looked for, then Times New Roman, and if not found, a generic system serif font will be used
- for paragraphs, first Verdana will be looked for, and if not found, Helvetica will be looked for, and if not found, a generic system sans-serif font will be used
- font family names with spaces must be enclosed in double quotes

Specifying Font Size

There are many good and bad ways to specify font sizes. Here are some of the better ways:

Units	Example	Description
percentage	120%	a percentage relative to the font size of the parent element
ems	1.2em	a unit of measure used in typography, also relative to the size of the parent element (and in this case equivalent to 120%)
pixels	16px	an absolute number of pixels

More Font Size

- percentages and ems are preferred for text because it allows the user to zoom in and out
- pixels do not zoom, and are more useful for defining sizes for things that are not text (we'll get to this soon)
- advice: don't spend too much time making fonts small, because users will zoom in/out based on their eyesight, and if the text isn't zoomable, they may just leave

Font Size Example

Here's a good rule of thumb for defining font sizes in your documents:

```
body { font-size: 100%; } /* default size for entire document */
h1 { font-size: 150%; } /* 50% bigger than parent (<body>) */
p { margin: 10px; } /* the space around a paragraph */
```

- set the font size in the <body> tag to 100%
- set the font size of other text elements (h1, p, etc) as a percentage relative to the parent
- use pixels only for non-text sizes (we'll get to this soon...)

Exercise C2: Colors and Fonts

1. make a copy of file `H1-structure.html`, and save it as `C2-color-font.html`
2. in the <body> element, style all text to use the Times New Roman font, with Georgia and generic serif as fallbacks
3. style the headings for levels 1-3 to use the Verdana font, with Arial and generic sans-serif as fallbacks
4. set the default font size for the entire document
5. style the level-1 heading to be two times as big as the parent's font size
6. make the level-2 headings green, using `rgb()`
7. style the paragraphs to have a 'lightblue' background
8. validate your CSS

Style Sheet Types

There are two types of style sheet.

Type	Description
internal	Style rules go in <style> element, nested in <head> element. Good for styles that only apply to one file.
external	Style rules are stored in a separate file, and (probably) shared by multiple HTML pages. Good for site-wide styling.

Internal Style Sheet Example

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Internal Style Sheet Example</title>
    <style>
        h1 { color: green; }
    </style>
</head>
<body>
    <h1>Internal Style Sheet Example</h1>
    <p>I am a lonely paragraph.</p>
</body>
</html>
```

External Style Sheet Example

File: `styles.css`

```
h1 { color: green; }
```

HTML file:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>External Style Sheet Example</title>
    <link ref="stylesheet" href="styles.css">
</head>
<body>
    <h1>External Style Sheet Example</h1>
    <p>I am a lonely paragraph.</p>
</body>
</html>
```

The Cascade

- rules *cascade* and can overwrite each other
- if (for example) two color rules exist for the same element, the second one will take effect
- external rules are read before internal rules, so internal ones can overwrite external ones
- more specific rules (with ID or class) override more general rules (without them)

The Cascade: Example

For this example, assume that an external style sheet contains:

```
h1#urgent { color: yellow; }
```

The internal style sheet contains:

```
h1 {
    color: green;
    font-size: 120%;
    color: blue;
}
h1.warning { color: red; }
```

- the level-1 heading with ID `urgent` will be yellow
- level-1 headings that belong to class `warning` will be red
- all other level-1 headings will be blue

Exercise C2: Cascade

1. load `H0-starter.html`, and save it as `C2-cascade.html`
2. add five level-2 headers to the file, giving the first one an ID of `first` and the first two a class of `first-two`
3. define an internal style rule to set the color to brown for all elements in class `first-two`
4. create an external stylesheet called `C2-cascade.css`, and add a rule to set the color of the element with ID `first` to orange
5. add a link in the <head> element of the HTML file to load the external stylesheet created in the prior step
6. add a rule to the bottom of the internal style sheet to set all level-2 headings to green

The Box Model

- every element occupies a rectangular area on the page
- by default, every element has a certain amount of space around it, so that unstyled pages are still readable
- we can use CSS to modify the spacing around objects
- to do this well, we need to understand the box model

Box Model Properties

These are the main properties we'll look at.

Name	Description
`padding`	the space between the content and the border
`border`	the border itself, possibly visible or possibly not
`margin`	the space outside the border

Box Property Units

The properties use the following units.

Name	Description
`pixel`	specified in absolute pixels, not zoomable (and this is typically desireable for the spaces between element boxes)
`em`	the typographic 'em', useful for having space above/below an element that grows/shrinks with zoomed text

Specifying the Border Property

The border property takes three arguments:

1. border width (normally in pixels)
2. border style (for example: solid, dotted)
3. border color

HTML

```
#p1 { border: 1px solid red; } /* 1-pixel solid red */
#p2 { border: 5px dotted rgb(0,0,255); } /* 2-pixel dotted blue */
```

Display

This is paragraph #p1.

This is paragraph #p2.

Variable Number of Values for Margin and Padding

You can specify one, two or four values when defining the margin or padding:

```
/* 5-pixel margin on all sides */
#p1 { margin: 5px; }

/* blank line on top/bottom (zoomable!), no margin on left/right */
#p2 { margin: 1em 0; }

/* margins of 0, 1, 2 and 3 pixels on the top, right, bottom and
left, respectively (i.e. clockwise starting at top) */
#p3 { margin: 0 1px 2px 3px; }
```

Specifying Only One Side

With borders, margins and padding, you can specify a single side instead of all four, by appending the side to the property name:

```
/* 5-pixel left margin */
#p1 { margin-left: 5px; }

/* 2-pixel solid black underline */
h1 { border-bottom: 2px solid black; }

/* 5-pixel right padding */
#p2 { padding-right: 5px; }
```

Try It Out

This page at w3schools.com lets us interactively experiment with all of the box model properties. Try it out!

Exercise C3: Box Model

1. load `H0-starter.html`, and save it as `C3-box-model.html`
2. add four level-2 headers to the file, and give them IDs from `box1` to `box4`
3. style them as described below
4. remember: unstyled elements may still have default margins or padding!

ID	Styling
`box1`	default margin and padding, a one-pixel solid orange border
`box2`	no margin, no border, no padding
`box3`	1em top/bottom margin, two-pixel dotted red border, 5px padding
`box4`	2em top margin, no other margins, 20px padding, 'lightblue' background color, 1-pixel solid black border

Block vs Inline Display

Every HTML element has a default value for the `display` property, either `block` or `inline`

Value	Description
`block`	always starts on a new line, breaking the flow. examples: `<h1>, <p>, <ul>`
`inline`	preserves the normal flow of text. examples: `<em>, <a>`

More information on the w3schools Block and Inline Elements page.

Two More HTML Tags

We use two more HTML tags that don't have any semantic meaning but that are very useful for applying styles to elements.

Tag	Description
`<div>`	a container with block display (that breaks flow)
`<span>`	a container with inline display (that preserves flow)

and Example

Style:

```
.warning { color: red; }
```

HTML:

```
<p>This para has a <span class="warning">warning</span> in it.</p>
<div class="warning">Here is a div with a warning.</div>
<p>This para has nothing special.</p>
```

Display:

This para has a warning in it.

Here is a div with a warning.

This para has nothing special.

Width and Height Properties

Sometimes we want to specify the width and/or height of an element, instead of having it "shrink" around its content. These properties can be used with the tag to create regions of specific sizes on the page.

ID	Styling
`width`	the width of an element
`height`	the height of an element
`min-width`	the smallest width an element can have
`max-width`	the largest width an element can have

More info here.

Width/Height Example

Here are examples how we specify the width or height of an element.

```
/* Make the aside 200px wide. (It will be as high as
needed to hold the content.) */
aside { width: 200px; }

/* Make the footer 50px high. (It will be as wide
as the browser window.) */
footer { height: 50px; }
```

Page Layout

To design a page, we break up the page into regions. (Show it on the whiteboard.) We can use tags along with HTML5 semantic tags to create the regions, like this.

Note however that we haven't applied any styling yet.

Exercise C4: Width/Height

In this exercise you will give some styling to existing HTML.

1. load `C-layout-before.html`, and save it in your folder
2. give the <header> tag a background color of 'lightgray' so we can see how big its box is
3. give the <aside> tag a width of 200 pixels, and give it a background color of hexadecimal #aaa (which is shorthand for #aaaaaa)
4. give the <article> tag a width of 70%, so it will use 70% of the window width, and give it a background color of 'lightblue'

Floating

- normally elements occupy the entire width of the browser window
- we can use the `width` property to limit their width
- but since they are block elements, we can't (yet) position them next to each other; each new element is placed below the last one
- the `float` property lets us do it

More Floating

In order to "float" an element:

1. in your HTML, specify the element to be floated first, followed by the non-floated element
2. in the floated element's CSS, specify whether to float left or right
3. in the non-floated element's CSS, specify a left or right margin at least as big as the width of the floated element, in order to make room for it
4. to end floating, specify 'clear: both' in the CSS of the element where floating should stop

Even More Floating

Here's an example of the CSS:

```
aside {
    float: left; /* position column on left */
    width: 200px; /* column has this width */
}
main { margin-left: 210px; } /* leave 10px space between */
footer { clear: both; } /* stop floating element at footer */
```

And the HTML:

```
<aside>(Stuff in column)</aside>
<main>Content in the main element...</main>
<footer>The Footer</footer>
```

And here we have a better view of the example.

Exercise C5: Floating

1. download file `C-floating.html` and save it in your folder as `C-floating1.html`
2. your mission is use the instructions from the slide before last to make the <aside> element float to the right instead of to the left

Menu Styling

- although most behavior is done by JavaScript, we can do some menu interaction with pure CSS
- in our HTML we use an unordered list to hold the menu items
- the rest of the "magic" is done with styling
- this link gives more info about menu styling

Pseudo-Classes

In order to get interaction, we need to use a *pseudo-class* with the <a> tag that specifies the state of the link.

```
/* Normal styling for links */
a {
    padding: 5px;
    background-color: green;
}
/* Change the background to red when hovered over */
a:hover { background-color: red; }
```

Hover over this link to see it work.

Menu Example

This file gives a complete example for a menu... with ample commentary. ;-) View the source code (in the browser) to see how it's done.

Walk-Thru Example

Now, if time permits, I will take an unstyled document and transform it before your eyes... I'll use C-layout-before.html as a starting point.