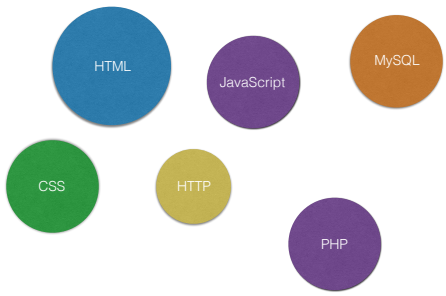
CS 337
Web Programming

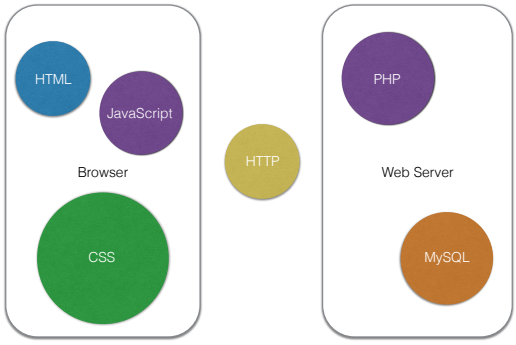
HTML/CSS/HTTP/JavaScript/PHP/MySQL/
CGI/XML/JSON/HTTPD/W3C/OMG No Moar TLAs

CSS

Doing it with Style

The Big Picture

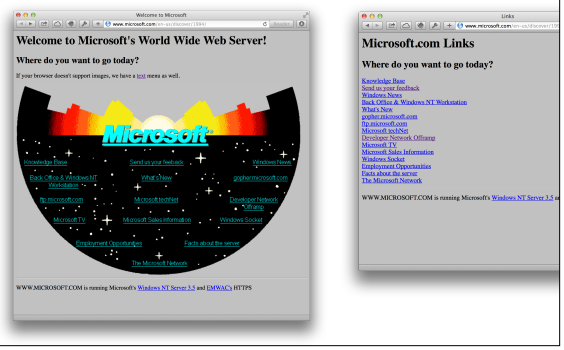




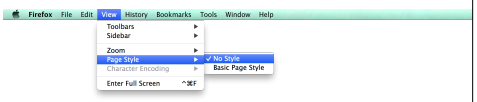
CSS

- Cascading Style Sheets
- CSS 1.0 first introduced in 1996, around HTML 3
- Early 2000s, developers began migrating from visual markup to CSS for styling
- No more tables for layout!

The Early Web



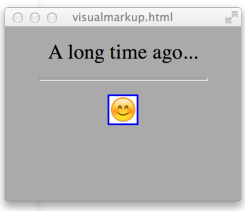
CSS



- View a webpage in Firefox
- View → Page Style → No Style

Visual Markup

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">

  <head>
    <title>visualmarkup.html</title>
  <head>

  <body bgcolor="#AAAAAA">

  <p align="center">
    <font size="+2">A long time ago...</font>
  </p>

  <hr size="4" width="75%">

  <p align="center">
    <a href="http://starwars.wikia.com">
      <img src="img/smile.png" border="2" width="32">
    </a>
  </p>

  </body>
</html>
```



Visual Markup

- Visual Markup mixed styling within the document structure.
- What's good about this method of styling?
- What are some problems with this approach?

CSS

- Separate Content and Structure from Visual Display Styles
- CSS Zen Garden
  - http://www.csszengarden.com/
  - http://www.csszengarden.com/1/

CSS Specifications

- CSS 2.x
  - http://www.w3.org/TR/CSS2/
- CSS 3+ A bit more involved than it used to be
  - http://www.w3.org/Style/CSS/current-work

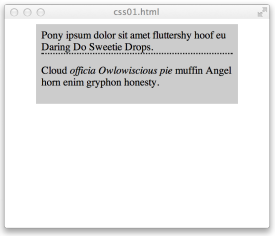
The Rules

```
p {
  color: blue;
}
```

- This is a CSS rule that makes the text of all paragraph elements blue.

More Rules

```
<!doctype html>
<head>
  <title>css01.html</title>
  <style>
    article {
      margin-left: auto;
      margin-right: auto;
      width: 75%;
      background-color: #CCC;
      padding: 0.5em;
    }

    header {
      border-bottom: 2px dotted #222;
    }
  </style>
<head>

<body>
  <article>
    <header>
      Pony ipsum dolor sit amet fluttershy
      hoof eu Daring Do Sweetie Drops.
    </header>
    <p>
      Cloud <em>officia Owlowiscious pie</em>
      muffin Angel horn enim gryphon honesty.
    </p>
  </article>
</body>
</html>
```



Playground

Where CSS Rules Live

- Within `<style>` elements
- Within a `style` attribute directly on an element
- On an external style sheet linked from a document

Inside a `<style>` Element

```
<!doctype html>
<head>
  <title>css01.html</title>
  <style>
    article {
      margin-left: auto;
      margin-right: auto;
      width: 75%;
      background-color: #CCC;
      padding: 0.5em;
    }

    header {
      border-bottom: 2px dotted #222;
    }
  </style>
<head>

<body>
  <article>
    <header>
```

Inline CSS Styles

```
<!doctype html>
<head>
  <title>css02.html</title>
<head>

<body>
  <article style="background-color: #CCC;">
    <header style="border-bottom: 2px dotted #222;">
      Pony ipsum dolor sit amet fluttershy
      hoof eu Daring Do Sweetie Drops.
    </header>
    <p>
      Cloud <em>officia Owlowiscious pie</em>
      muffin Angel horn enim gryphon honesty.
    </p>
  </article>
</body>
</html>
```

The `style` attribute on any element

Linking to Stylesheets

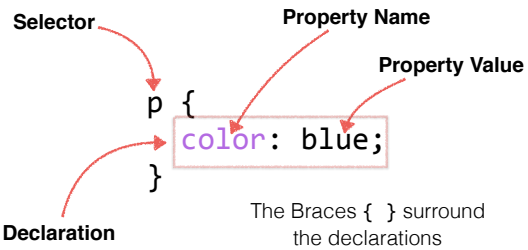
css03.html

```
<!doctype html>
<head>
  <title>css03.html</title>
  <link rel="stylesheet" type="text/css" href="css03.css"/>
<head>

<body>
  <article>
    <header>
      Pony ipsum dolor sit amet fluttershy
      hoof eu Daring Do Sweetie Drops.
    </header>
    <p>
      Cloud <em>officia Owlowiscious pie</em>
      muffin Angel horn enim gryphon honesty.
    </p>
  </article>
</body>
</html>
```

css03.css

```
article {
  margin-left: auto;
  margin-right: auto;
  width: 75%;
  background-color: #CCC;
  padding: 0.5em;
}

header {
  border-bottom: 2px dotted #222;
}
```

Anatomy of a CSS Rule



The Braces { } surround the declarations

http://www.w3.org/TR/css-syntax-3/#syntax-description

Anatomy of a CSS Rule

- Whitespace doesn't matter. The following are all equivalent

```
p
{
  color: blue;
}
```

```
p {
  color: blue;
}
```

```
p { color: blue; }
```

```
p

{

  color:
    blue;

}
```

```
p{color:blue;}
```

Multiple Declarations Per Rule

- The rule for `<article>` elements has 5 separate declarations
- The rule for `<header>` elements has one declaration

```
article {
  margin-left: auto;
  margin-right: auto;
  width: 75%;
  background-color: #CCC;
  padding: 0.5em;
}

header {
  border-bottom: 2px dotted #222;
}
```

Grouping Rules

- If we have several rules, which have the same declarations for different selectors, we can group them together in a single rule

```
h1 { font-family: sans-serif; }
h2 { font-family: sans-serif; }
h3 { font-family: sans-serif; }
```

```
h1, h2, h3 { font-family: sans-serif; }
```

CSS Keywords

- Why does the following CSS rule do nothing?

```
p {
  color: “blue”;
}
```

- Can we use browser developer tools to offer clues?

CSS Keywords

- What are language keywords?
- What are string literals?

```
p {
  color: “blue”;
}
```

String

```
p {
  color: blue;
}
```

CSS Keyword

CSS Keywords

- CSS has a ***lot*** of keywords
- All properties are keywords
- There are lots of valid property values that are also keywords

http://www.w3.org/TR/#tr_CSS

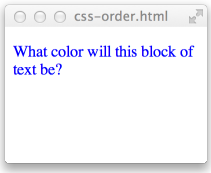
Order of Precedent

- What happens if we have two rules with identical selectors?
- Which color will the paragraph text be?

```
p {
  color: red;
}

p {
  color: blue;
}
```

Order of Precedent

```
<!doctype html>
<head>
  <title>css-order.html</title>
  <style>
    p {
      color: red;
    }

    p {
      color: blue;
    }
  </style>
<head>

<body>
  <p>
    What color will this block of text be?
  </p>
</body>
</html>
```



- Newest Rule Wins, this makes things easier for us

Selectors

- The Selector tells this rule which HTML elements to target.
- This rule targets all <p> elements on the page.

```
p {
  color: blue;
}
```

http://www.w3.org/TR/css3-selectors/

Selectors

`*`	any element
`E`	elements of type `E`
`E[foo]`	elements of type `E` with an attribute named “foo”
`E F`	an element `F` who is some descendant of `E`
`E > F`	an element `F` which is a direct child of `E`
`E.arrg`	an element `E` which has a `class` attribute value of “arrg”
`E#woot`	an element `E` who’s `id` attribute value is “woot”

http://www.w3.org/TR/css3-selectors/

Selectors

- Universal Selector is Optional
- `*.navigation` and `.navigation` are equivalent
- `*#entry003` and `#entry003` are equivalent

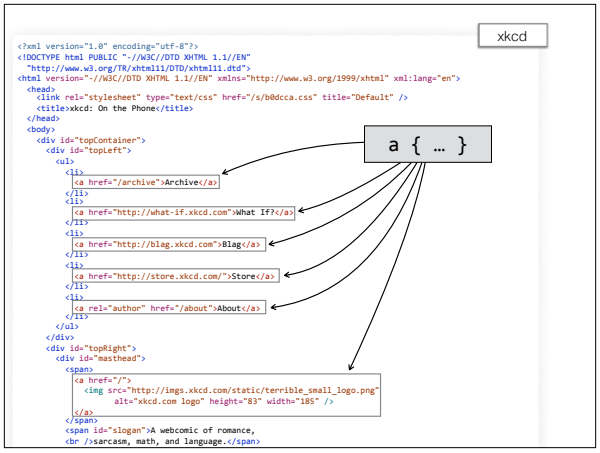
`.warg`	all elements with a `class` attribute who’s value contains “warg”
`#something`	all elements with an `id` value of “something”

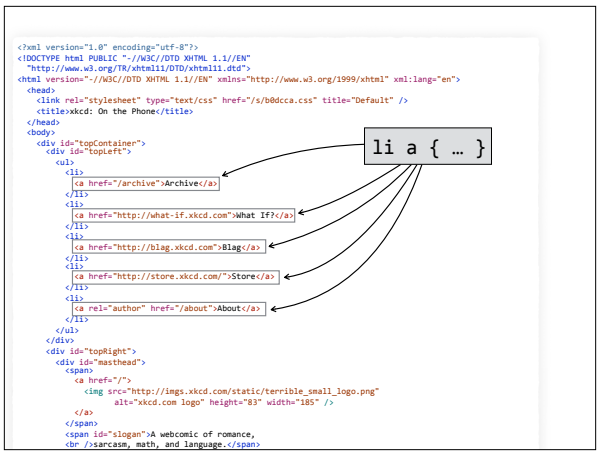
Selectors

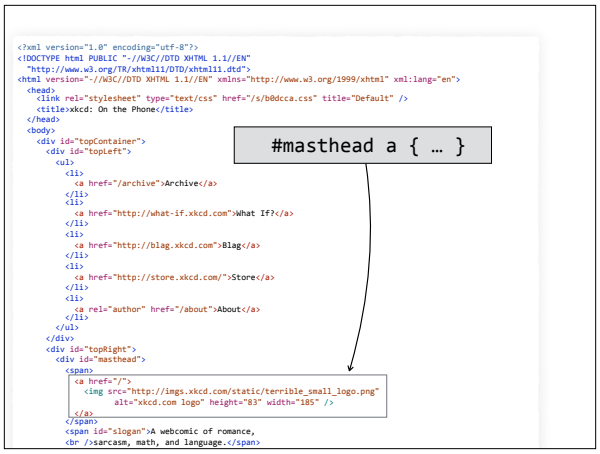
- Correctly targeting HTML elements is a very valuable skill
- Lets look at some examples.

xkcd

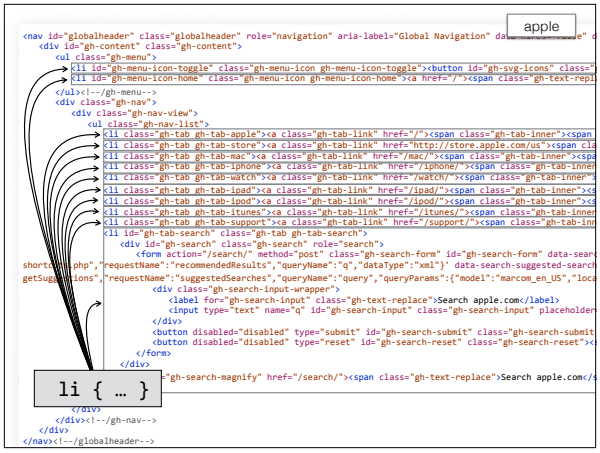
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html version="-//W3C//DTD XHTML 1.1//EN" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="/s/b0dcca.css" title="Default" />
    <title>xkcd: On the Phone</title>
  </head>
  <body>
    <div id="topContainer">
      <div id="topLeft">
        <ul>
          <li>
            <a href="/archive">Archive</a>
          </li>
          <li>
            <a href="http://what-if.xkcd.com">What If?</a>
          </li>
          <li>
            <a href="http://blag.xkcd.com">Blag</a>
          </li>
          <li>
            <a href="http://store.xkcd.com/">Store</a>
          </li>
          <li>
            <a rel="author" href="/about">About</a>
          </li>
        </ul>
      </div>
      <div id="topRight">
        <div id="masthead">
          <span>
            <a href="/">
              <img src="http://imgs.xkcd.com/static/terrible_small_logo.png"
                   alt="xkcd.com logo" height="83" width="185" />
            </a>
          </span>
          <span id="slogan">A webcomic of romance,
          <br />sarcasm, math, and language.</span>
```

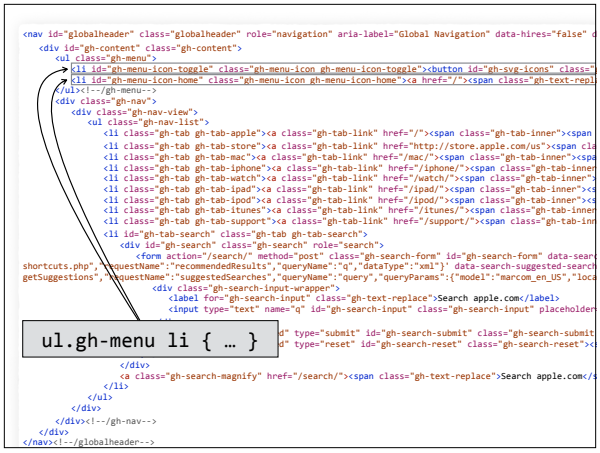

xkcd
a { … }

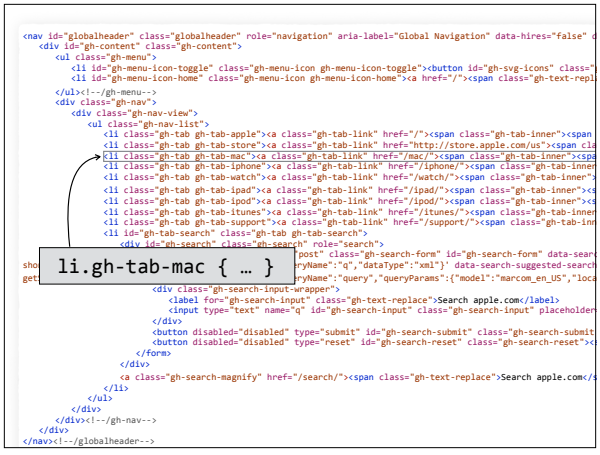

li a { … }


#masthead a { … }

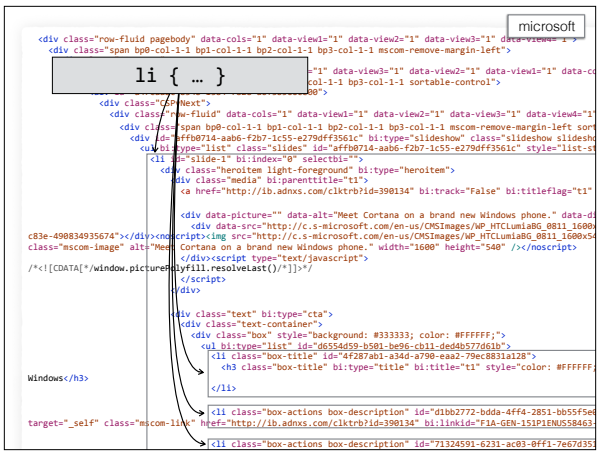

apple


apple
li { … }


ul.gh-menu li { … }


li.gh-tab-mac { … }

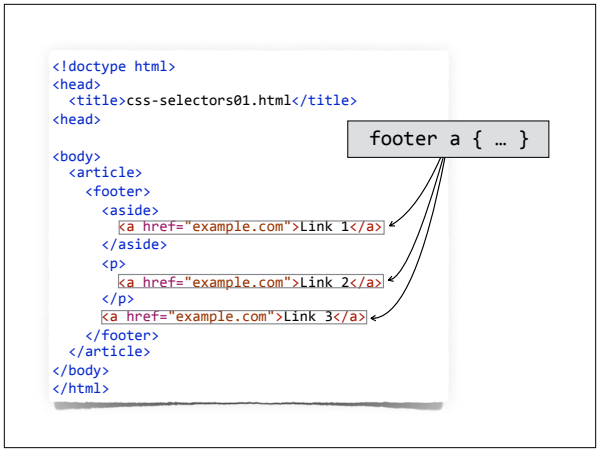

microsoft

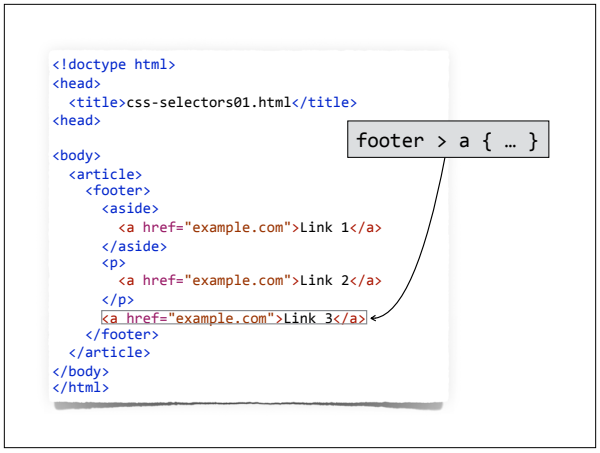



```
<!doctype html>
<head>
  <title>css-selectors01.html</title>
<head>

<body>
  <article>
    <footer>
      <aside>
        <a href="example.com">Link 1</a>
      </aside>
      <p>
        <a href="example.com">Link 2</a>
      </p>
      <a href="example.com">Link 3</a>
    </footer>
  </article>
</body>
</html>
```

```
<!doctype html>
<head>
  <title>css-selectors01.html</title>
<head>

<body>
  <article>
    <footer>
      <aside>
        <a href="example.com">Link 1</a>
      </aside>
      <p>
        <a href="example.com">Link 2</a>
      </p>
      <a href="example.com">Link 3</a>
    </footer>
  </article>
</body>
</html>
```

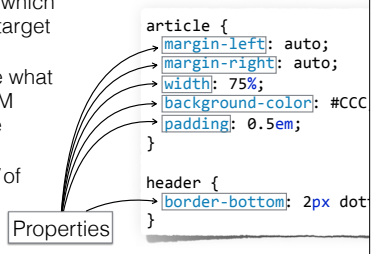
a { … }





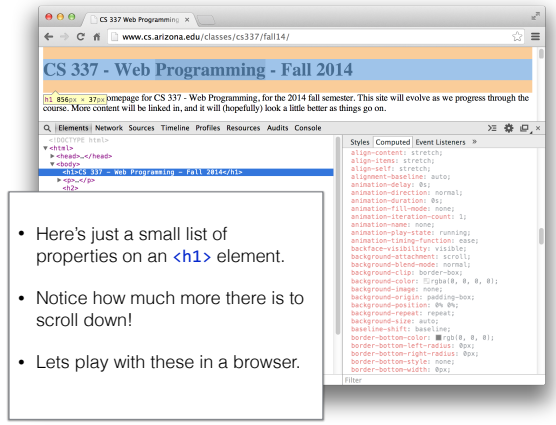
Properties

- Selectors define which DOM objects to target
- Properties define what aspects of a DOM object to change
- There are a ***LOT*** of properties



Properties

- CSS2 Complete Set of Properties
  - http://www.w3.org/TR/CSS2/propidx.html
- CSS3 Properties
  - https://developer.mozilla.org/en-US/docs/Web/CSS/Reference



- Here's just a small list of properties on an `<h1>` element.
- Notice how much more there is to scroll down!
- Lets play with these in a browser.

CSS Text Properties

- `font-family`
- `font-size`
- `color`
- `font-weight`
- `text-decoration`

font-family

```
<!doctype html>
<head>
  <title>css-font-family.html</title>
  <script src="http://use.typekit.net/moq4rmb.js"></
  <script>try{Typekit.load();}catch(e){}</script>
<head>

<body>
  <h1>Default Font Family</h1>
  <h1 style="font-family: sans-serif">Sans-Serif Fon
  <h1 style="font-family: monospace">Fixed-Width Fon
  <h1 style="font-family: cursive">Script Font</h1>
  <h1 class="comicbook">Comic Book Font</h1>
</body>
</html>
```

font-size

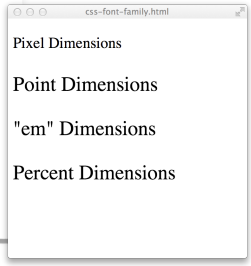
- Units of measurement
  - em - 1 'em' is the height of the current font settings as determined by your browser
  - px - Pixels on your screen (depending on your screen resolution…)
  - pt - 1pt is 1/72 of an inch. Traditional unit for print design work.
  - % - 100% is the default font size, and you can go up and down from there

font-size

- What should we use?
- px and pt are fixed, and used to not scale
- em and % will scale based on browser settings
- Browsers pretty much scale everything these days
- Lets look at an example

font-size

```
<!doctype html>
<head>
  <title>css-font-family.html</title>
  <style>
    .sz1 { font-size: 24px; }
    .sz2 { font-size: 24pt; }
    .sz3 { font-size: 2em; }
    .sz4 { font-size: 200%; }
  </style>
<head>

<body>
  <p class="sz1">Pixel Dimensions</p>
  <p class="sz2">Point Dimensions</p>
  <p class="sz3">&amp;quot;em&amp;quot; Dimensions</p>
  <p class="sz4">Percent Dimensions</p>
</body>
</html>
```



font-weight

- Determines how heavy the font stroke is.
- Descriptive
  - Bold, Normal
- Relative
  - Lighter, Bolder
- Absolute
  - 100, 300, 800

http://www.w3.org/TR/css-fonts-3/#font-weight-prop

font-weight

- Determines how heavy the font stroke is.
- Descriptive
  - Bold, Normal
- Relative
  - Lighter, Bolder
- Absolute
  - 100, 300, 800

http://www.w3.org/TR/css-fonts-3/#font-weight-prop

font-weight

```
<!doctype html>
<head>
  <title>css-font-weight.html</title>
  <style>
    body {
      font-family: sans-serif;
      font-size: 2em;
    }
    .w1 { font-weight: normal; }
    .w2 { font-weight: bold; }
    .w3 { font-weight: 500; }
    .w3a { font-weight: lighter; }
    .w3b { font-weight: bolder; }
    .w4 { font-weight: 700; }
  </style>
<head>

<body>
  <p class="w1">Normal Weight</p>
  <p class="w2">Bold</p>
  <p class="w3">
    Weight 500,
    <span class="w3a">Lighter</span>,
    <span class="w3b">Bolder</span>
  </p>
  <p class="w4">
    Weight 700,
    <span class="w3a">Lighter</span>,
    <span class="w3b">Bolder</span>
  </p>
</body>
</html>
```

color

- The `color` property changes the color of the foreground text. It's not `font-color`, just `color`. Also not `colour`, sorry UK.
- Can be specified as named keywords, Hex values, or RGB values.

```
p {
  color: blue;
}
```

http://www.w3.org/TR/css3-color/#svg-color

color

```
<!doctype html>
<head>
  <title>css-color.html</title>
  <style>
    body {
      background-color: #ccc;
      font-weight: bold;
      font-size: 1.5em;
    }
    .c1 { color: red; }
    .c2 { color: purple; }
    .c3 { color: #076873; }
    .c4 { color: rgb(180, 255, 80); }
  </style>
<head>

<body>
  <p class="c1">Red</p>
  <p class="c2">Purple</p>
  <p class="c3">UA River</p>
  <p class="c4">Pale Yellow</p>
</body>
</html>
```

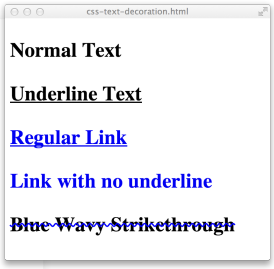
text-decoration

- The `text-decoration` set of properties defines various types of lines that can be associated with text.
- Mostly this is underlines, but could be other things.

http://www.w3.org/TR/css-text-decor-3/

text-decoration

```
<!doctype html>
<head>
  <title>css-text-decoration.html</title>
  <style>
    body { font-size: 2em; font-weight: 600;}
    .d1 { text-decoration: underline; }
    .d2 { text-decoration: none; }
    .d3 {
      text-decoration-line: line-through;
      text-decoration-color: blue;
      text-decoration-style: wavy;
    }
  </style>
<head>

<body>
  <p>Normal Text</p>
  <p class="d1">Underline Text</p>
  <p><a href="#">Regular Link</a></p>
  <p>
    <a href="#" class="d2">Link with no underline</a>
  </p>
  <p class="d3">Blue Wavy Strikethrough</p>
</body>
</html>
```



Shorthand Properties

- There are explicit properties for controlling just about every possible attribute of a DOM element.
- Often for related properties, there is a 'shorthand' property which makes writing rules a little easier.
- `text-decoration` was an example of a shorthand property.

Shorthand Properties

Explicit Properties	Shorthand Equivalent
`text-decoration-line: underline;`	`text-decoration: underline;`

`text-decoration-line: underline;`
`text-decoration-color: blue;`
`text-decoration-style: wavy;`
	`text-decoration: underline blue wavy;`

Vendor Properties

- There's standards, and then there are browsers.
- Browsers tend to support standards in development before they're "official"
- To avoid conflicting with the standard once its final, there are vendor prefixes.

Vendor Properties

WebKit	-webkit
Gecko	-moz
Trident	-ms
Opera	-o

Vendor Properties

- This is cool, because we get to play with new shiny toys before they're quite ready.
- The bad thing is that it leads to a bit of 'declaration spam'

```
div {
  -webkit-box-shadow: 7px 7px 5px 0px rgba(50, 50, 50, 0.75);
  -moz-box-shadow:    7px 7px 5px 0px rgba(50, 50, 50, 0.75);
  box-shadow:         7px 7px 5px 0px rgba(50, 50, 50, 0.75);
}
```

- The -webkit and -moz prefix declarations work today, and the box-shadow declaration is what the final standard will be. So that's there for future proofing

Inheritance

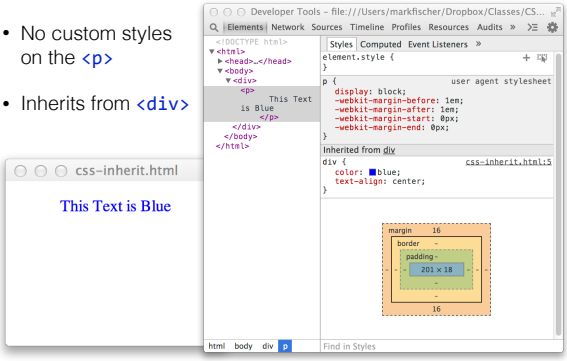
- Very different from OOP
- CSS Inheritance flows from parent elements
- Object Inheritance flows from parent class

Inheritance

```
<!doctype html>
<head>
  <title>css-inherit.html</title>
  <style>
    div {
      color: blue;
      text-align: center;
    }
  </style>
<head>

<body>
  <div>
    <p>
      This Text is Blue
    </p>
  </div>
</body>
</html>
```

Inheritance

- No custom styles on the `<p>`
- Inherits from `<div>`



Inheritance

- Do all properties inherit?
- Do all elements have the same properties?
- This is why understanding the DOM tree is so important.

http://www.w3.org/TR/css-cascade-3/#inheriting

Inheritance

- Do all properties inherit?
  - Nope!
- How do we know what inherits and what doesn't?

Inheritance

- Do all properties inherit?
  - Nope!
- How do we know what inherits and what doesn't?
  - The Docs of course!

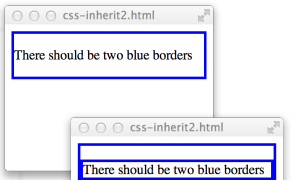
http://www.w3.org/TR/css-fonts-3/#font-size-prop

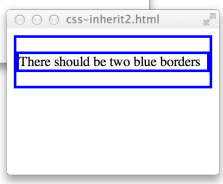
http://www.w3.org/TR/css3-background/#the-border-style

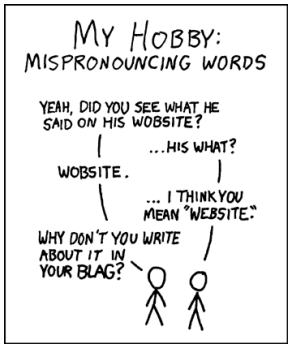
Inheritance

- You can force inheritance on properties that don't automatically inherit

```
<!doctype html>
<head>
  <title>css-inherit2.html</title>
  <style>
    div { border: 3px solid blue; }
    /* p { border: 3px solid blue; } */
    /* p { border: inherit; } */
  </style>
<head>

<body>
  <div>
    <p>
      There should be two blue borders
    </p>
  </div>
</body>
</html>
```







The Cascade

Precedent and Specificity

Specificity

- Each Selector has associated with it a particular weight, and this is called the Selector's **Specificity**
- When two rules conflict, the rule with the greater Specificity wins
- If rules have identical Specificity, the newest rule wins

Calculating Specificity

- Specificity can be represented as a 4 element tuple
  - 0,0,0,0
- Different types of selectors add to different parts of the tuple.
- Highest tuple wins
  - 1,99,0,0 > 0,1,99,0 > 0,0,1,99 > 0,0,0,99

http://www.w3.org/TR/2009/CR-CSS2-20090908/cascade.html#specificity

Calculating Specificity

Selector	Specificity	Example
Element	0,0,0,1	`H1`
Class	0,0,1,0	`.button`
ID	0,1,0,0	`#main`
Inline Style	1,0,0,0	`style="color: red;"`

Specificity Examples

Selector	Specificity
`li p`	0,0,0,2
`p.first`	0,0,1,1
`div#main`	0,1,0,1
`ol li ol li ol li`	0,0,0,6
`body div#content p.entry`	0,1,1,3
`style="color:red;"`	1,0,0,0

Specificity Example

```
<!doctype html>
<head>
  <title>css-specificity.html</title>
  <style>
    body { color: black; }
    li p { color: red; }
    div ul li p { color: purple; }
    #homeTab { color: blue; }
    div.content ul.tabs p { color: green; }
    body div.content ul li p { color: orange; }
  </style>
<head>

<body>
  <div class="content">
    <ul class="tabs">
      <li id="homeTab">
        <header>Home</header>
        <p>
          The Home Screen
        <p>
      </li>
    </ul>
  </div>
</body>
</html>
```

- What Color will the <p> content be?

Specificity Example

Selector	Specificity	Targets Element	Wins
`body`	0,0,0,1	`<body>`	
`li p`	0,0,0,1	`<p>`	
`div ul li p`	0,0,0,4	`<p>`	
`#homeTab`	0,1,0,0	`<li>`	
`div.content ul.tabs p`	0,0,2,3	`<p>`	✔
`body div.content ul li p`	0,0,1,5	`<p>`	

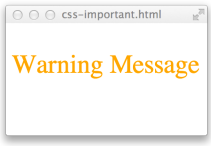
!important

- The `!important` argument is sort of separate from Specificity, although it relates to which directive wins.
- It is usually bad form to rely on `!important`. It makes people's lives very difficult.
- It only applies to individual declarations, not entire rules.

`!important` Example

```
<!doctype html>
<head>
  <title>css-important.html</title>
  <style>
    #content p {
      color: blue;
      font-size: 2em;
      font-weight: normal;
    }
    .warning {
      color: orange !important;
      font-weight: bold;
    }
  </style>
<head>

<body>
  <div id="content">
    <p class="warning">
      Warning Message
    <p>
  </div>
</body>
</html>
```

- The first rule has a specificity of 0,1,0,1 which makes it very hard to override with a class selector
- Only the `!important` declaration is overridden. The text is not Bold



Precedent

- If two rules have the same Specificity, then their Precedent wins.
- Source First
  - Browser
  - User-Stylesheet
  - Page-Author
- Then Order (Newest Rules win)

Boxes and Borders

- HTML elements are largely rectangular (although this is changing)
- We have a lot of tools at our disposal for how we display and arrange our boxes

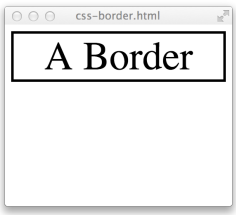
Borders

- border-color
- border-style
- border-width
- border-radius
- box-shadow
- border (shorthand property)

http://www.w3.org/TR/css3-background/#borders

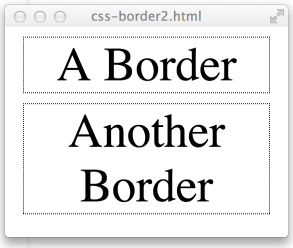
Borders

```
<!doctype html>
<head>
  <title>css-border.html</title>
  <style>
    div {
      font-size: 3em;
      text-align: center;
      border-style: solid;
    }
  </style>
<head>

<body>
  <div>A Border</div>
</body>
</html>
```



Borders

```
<!doctype html>
<head>
  <title>css-border2.html</title>
  <style>
    div {
      font-size: 3em;
      text-align: center;
      margin: 10px;
    }
    .box1 {
      border-style: dotted;
      border-width: 1px;
      border-color: blue;
    }
    .box2 {
      border: dotted 1px blue;
    }
  </style>
<head>

<body>
  <div class="box1">A Border</div>
  <div class="box2">Another Border</div>
</body>
</html>
```



border-radius

```
<!doctype html>
<head>
  <title>css-border-radius.html</title>
  <style>
    div {
      border: 3px solid purple;
      -webkit-border-radius: 15px;
      -moz-border-radius: 15px;
      border-radius: 15px;
      font-size: 2em;
      padding: 15px;
    }
  </style>
<head>

<body>
  <div>
      Rounded Rects!
  </div>
</body>
</html>
```

http://border-radius.com

https://developer.mozilla.org/en-US/docs/Web/CSS/border-radius

box-shadow

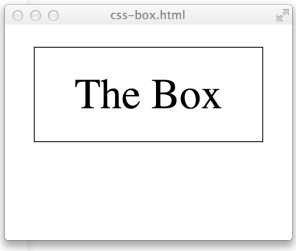
```
<!doctype html>
<head>
  <title>css-box-shadow.html</title>
  <style>
    div {
      border: 3px solid purple;
      -webkit-border-radius: 15px;
      -moz-border-radius: 15px;
      border-radius: 15px;
      -webkit-box-shadow: 7px 7px 5px 0px rgba(50, 50, 50, 0.75);
      -moz-box-shadow:    7px 7px 5px 0px rgba(50, 50, 50, 0.75);
      box-shadow:         7px 7px 5px 0px rgba(50, 50, 50, 0.75);
      font-size: 2em;
      padding: 15px;
    }
  </style>
<head>

<body>
  <div>
      Drop Shadows!
  </div>
</body>
</html>
```

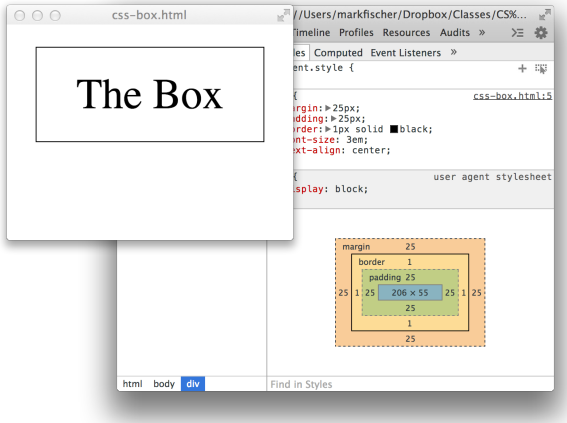
http://css3gen.com/box-shadow/

http://www.w3.org/TR/css3-background/#the-box-shadow

Box Model

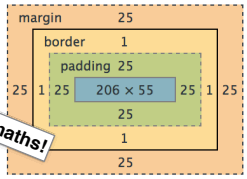
```
<!doctype html>
<head>
  <title>css-box.html</title>
  <style>
    div {
      margin: 25px;
      padding: 25px;
      border: 1px solid black;
      font-size: 3em;
      text-align: center;
    }
  </style>
<head>

<body>
  <div>The Box</div>
</body>
</html>
```





Box Model

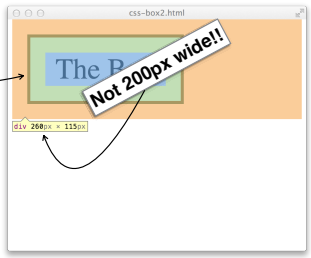
- How wide is this box?
- Content + Borders + Padding
  - 1 + 25 + 206 + 25 + 1 = 258
- What about margins?
  - 25 + 258 + 25 = 308
- If we change the border width, will this box get bigger?



Box Model

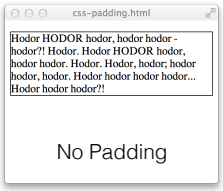
- This is pretty clunky, especially when specifying a width.

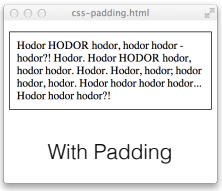
```
<!doctype html>
<head>
  <title>css-box2.html</title>
  <style>
    div {
      width: 200px;
      margin: 25px;
      padding: 25px;
      border: 5px solid black;
      font-size: 3em;
      text-align: center;
    }
  </style>
<head>

<body>
  <div>The Box</div>
</body>
</html>
```



`padding`

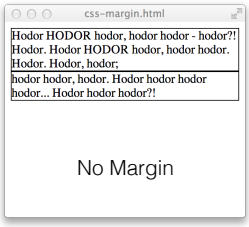
- Padding is the distance from the border to the content.
- Padding is important to legibility.

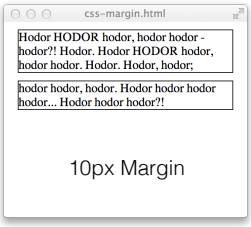




`margin`

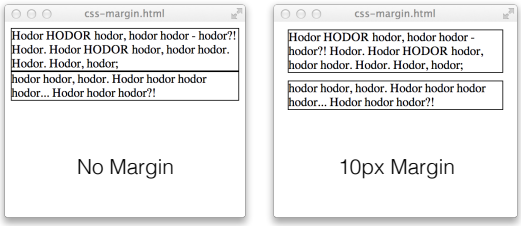
- Margin is the distance from the border out to the edge of another element.





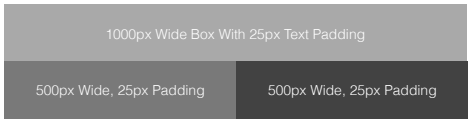
padding cont'd

- Margin is the distance from the border out to the edge of another element.



Box Model Example

- Consider the following mock-up. How would you make this with the CSS Box Model?



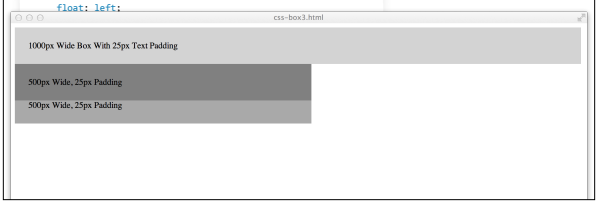
Box Model Example

```
<!doctype html>
<head>
  <title>css-box3.html</title>
  <style>
    .box1 {
      width: 1000px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 500px;
      padding: 25px;
      background-color: gray;
      float: left;
    }
    .box3 {
      width: 500px;
      padding: 25px;
      background-color: darkgray;
    }
  </style>
<head>

<body>
  <div class="box1">1000px Wide Box With 25px Text Padding</div>
  <div class="box2">500px Wide, 25px Padding</div>
  <div class="box3">500px Wide, 25px Padding</div>
</body>
</html>
```

- This is a pretty good first pass
- How does it look?

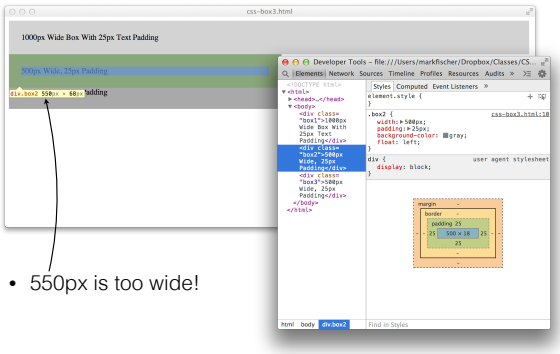
Box Model Example

```
<!doctype html>
<head>
  <title>css-box3.html</title>
  <style>
    .box1 {
      width: 1000px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 500px;
      padding: 25px;
      background-color: gray;
      float: left;
```



- Hmm, not exactly what we're going for…
- What happened?

Box Model Example



- 550px is too wide!

Box Model Example

```
<!doctype html>
<head>
  <title>css-box3.html</title>
  <style>
    .box1 {
      width: 950px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 450px;
      padding: 25px;
      background-color: gray;
      color: white;
      float: left;
    }
    .box3 {
      width: 450px;
      padding: 25px;
      background-color: #333;
      color: white;
      float: left;
    }
  </style>
<head>

<body>
  <div class="box1">1000px Wide Box With 25px Text Padding</div>
  <div class="box2">500px Wide, 25px Padding</div>
  <div class="box3">500px Wide, 25px Padding</div>
</body>
</html>
```

oh noes, maths!

- Second Pass
- Subtract the padding (x2) from the width
- 500 - 25 - 25 = 450
- How's this look?

Box Model Example

```
<!doctype html>
<head>
  <title>css-box3.html</title>
  <style>
    .box1 {
      width: 950px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 450px;
      padding: 25px;
      background-color: gray;
      color: white;
      float: left;
    }
    .box3 {
      width: 450px;
      padding: 25px;
      background-color: #333;
```

- Better!
- Now they're actually 500px wide
- What happens when we want to change the padding?

`box-sizing` to the rescue

- CSS3 introduced the `box-sizing` model to help with this insanity.

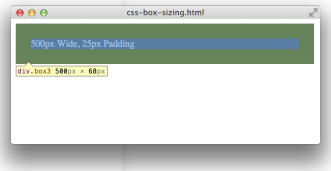
Property	Value	Notes
`box-sizing:`	`content-box`	This is the old 'Box Model'
	`padding-box`	Content+Padding is included in 'width'
	`border-box`	Everything up to the border included
		^— yay!

http://www.w3.org/TR/css3-ui/#box-sizing

`box-sizing` to the rescue

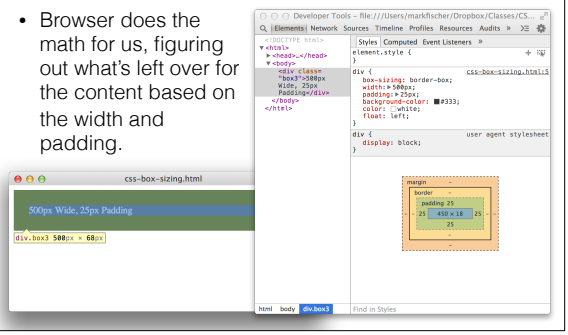
- 500px width results in a box 500px wide!

```
<!doctype html>
<head>
  <title>css-box-sizing.html</title>
  <style>
    div {
      box-sizing: border-box;
      width: 500px;
      padding: 25px;
      background-color: #333;
      color: white;
      float: left;
    }
  </style>
<head>

<body>
  <div class="box3">500px Wide, 25px Padding</div>
</body>
</html>
```



`box-sizing` to the rescue

- Browser does the math for us, figuring out what's left over for the content based on the width and padding.

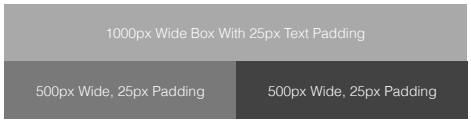


`box-sizing` to the rescue

```
<!doctype html>
<head>
  <title>css-box-sizing.html</title>
  <style>
    div {
      width: 500px;
      padding: 25px;
      border: 10px solid blue;
      margin: 25px;
      background-color: #888;
      color: white;
    }
    .b1 {
      box-sizing: content-box;
    }
    .b3 {
      box-sizing: border-box;
    }
  </style>
<head>

<body>
  <div class="b1">Content is 500px Wide, 25px Padding, 10px Border</div>
  <div class="b3">Box is 500px Wide, 25px Padding, 10px Border</div>
</body>
</html>
```

`box-sizing` to the rescue

- Lets try our earlier example with the **`box-sizing`** property.



`box-sizing` to the rescue

```
<!doctype html>
<head>
  <title>css-box-sizing2.html</title>
  <style>
    div {
      box-sizing: border-box;
      float: left;
    }
    .box1 {
      width: 1000px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 500px;
      padding: 25px;
      background-color: gray;
    }
    .box3 {
      width: 500px;
      padding: 25px;
      background-color: darkgray;
    }
  </style>
<head>

<body>
  <div class="box1">1000px Wide Box With 25px Text Padding</div>
  <div class="box2">500px Wide, 25px Padding</div>
  <div class="box3">500px Wide, 25px Padding</div>
</body>
</html>
```

- We just add one new rule for all elements that sets the box-sizing value to border-box.
- The rest of our rules are nice and simple, and make sense.

`box-sizing` to the rescue

```
<!doctype html>
<head>
  <title>css-box-sizing2.html</title>
  <style>
    div {
      box-sizing: border-box;
      float: left;
    }
    .box1 {
      width: 1000px;
      padding: 25px;
      background-color: lightgray;
    }
    .box2 {
      width: 500px;
      padding: 25px;
      background-color: gray;
```

- It works! Woot!


```
</html>
```

`box-sizing` to the rescue

- This seems too good to be true, can we actually use this?

Browser compatibility

Desktop | Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
Basic support	1.0 `-webkit` [1] 10.0	1.0 (1.7 or earlier) `-moz` [1] 29.0 (29.0)	8.0 [1]	7.0	3.0 (522) `-webkit` 5.1 (534.12)
`padding-box`	Not supported	1.0 (1.0)	Not supported	Not supported	Not supported

https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing

The `width` property

- The `width` property defines the width of an element, but which width depends on the `box-sizing` model.
- In the old days of the original Box Model, `width` always specified the width of the content area.

box-sizing	width
`content-box`	Width of the content area
`padding-box`	Width of content + padding
`border-box`	Width of content + padding + border

The `width` property

- The `width` property can be specified in units, or as a percentage of the enclosing container.

```
.d1 { width: 200px; }
.d2 { width: 4em; }
.d3 { width: 50%; }
```

Display

Now You See It…

The `display` Property

- Dictates how a given element is … displayed
- Most elements have a default display mode
- But we can change them!

Element	`display` value
`<div>`	block
`<span>`	inline
`<p>`	block
`<table>`	table
`<th>`	table-cell
`<li>`	list-item

The `display` Property

```
<!doctype html>
<head>
  <title>css-display.html</title>
  <style>
    .button { ••• }
    a { text-decoration: none; }
    a.button { display: block; }
  </style>
</head>

<body>

    <div class="button">
      <a href="#">Click Me</a>
    </div>

    <a href="#">
      <div class="button">
        Click Me
      </div>
    </a>

    <a href="#" class="button">
      Click Me
    <a>

</body>
</html>
```

- Setting the `<a>` element to display as a block allows us to eliminate an enclosing `<div>`

The `display` Property

- Turn an unordered list into a set of buttons

```
<!doctype html>
<head>
  <title>css-display2.html</title>
  <style>
    ul li { ••• }
    li { display: inline-block; }
  </style>
</head>

<body>

    <ul>
      <li>Button 1</li>
      <li>Button 2</li>
      <li>Button 3</li>
      <li>Button 4</li>
    </ul>

</body>
</html>
```

The `display` Property

- `display: none;` Hide Things!
- Still exists in the DOM

```
<!doctype html>
<head>
  <title>css-display3.html</title>
  <style>
    ul li { ••• }
    li { display: inline-block; }
  </style>
</head>

<body>

    <ul>
      <li>Button 1</li>
      <li>Button 2</li>
      <li style="display: none;">
        Button 3
      </li>
      <li>Button 4</li>
    </ul>

</body>
</html>
```

The `display` Property

- `block` vs. `inline-block`

```
<!doctype html>
<head>
  <title>css-display3.html</title>
  <style>
    ul li { ••• }
    .inlineblock li { display: inline-block; }
    .block li { display: block; }
  </style>
</head>

<body>

    <ul class="block">
      <li>Button 1</li>
      <li>Button 2</li>
    </ul>

    <ul class="inlineblock">
      <li>Button 1</li>
      <li>Button 2</li>
    </ul>

</body>
</html>
```

The `display` Property

```
display: none

display: inline
display: block
display: list-item
display: inline-block
display: inline-table
display: table
display: table-cell
display: table-column
display: table-column-group
display: table-footer-group
display: table-header-group
display: table-row
display: table-row-group
```

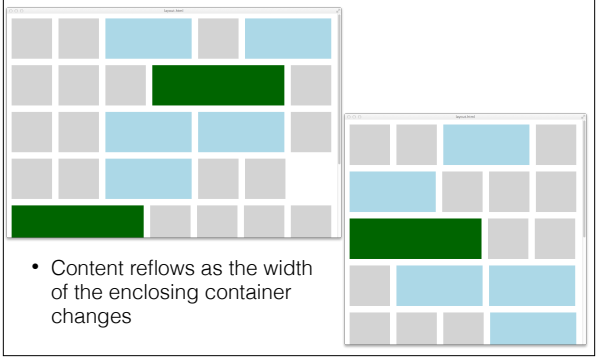
```
display: flex
display: inline-flex
display: grid
display: inline-grid
display: ruby
display: ruby-base
display: ruby-text
display: ruby-base-container
display: ruby-text-container
display: run-in

display: inherit
display: initial
display: unset
```

Positioning & Layout

Default Flow

- The default layout for HTML documents is a standard western-language format
  - Left to Right, wrapping lines from Top to Bottom
- `inline` content flows Left to Right until it hits the edge of its container, then wraps to the line below it
- `block` content puts a line break above and below it, and fills the entire width of its enclosing container

`inline` Content



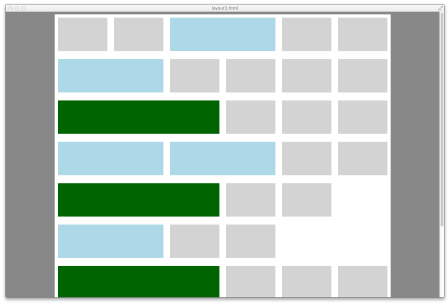
- Content reflows as the width of the enclosing container changes

`block` Content

- block content fills the whole width of the enclosing container

Fixed-Width Grid

- What if the 'enclosing container' isn't the browser window?



```
<!doctype html>
<head>
  <title>layout3.html</title>
  <style>
    body {
      background-color: #888;
    }
    section {
      width: 1020px; /* 170 * 6 = 1020 */
      background-color: #fff;
      margin: 0 auto 0 auto;
    }
    div {
      box-sizing: border-box;
      display: inline-block;
      margin: 10px;
      height: 100px;
    }
    .box1 {
      width: 150px;
      background-color: lightgray;
    }
    .box2 {
      width: 320px;
      background-color: lightblue;
    }
    .box3 {
      width: 490px;
      background-color: darkgreen;
    }
    .box4 {
      display: block;
      background-color: lightgreen;
    }
  </style>
<head>
```

- We add a `<section>` and set a width for it. Now it is the enclosing container.

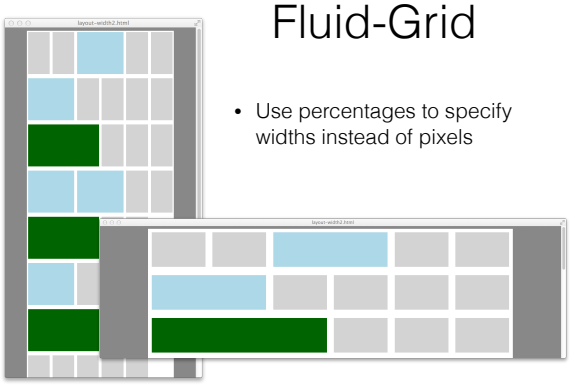
```
<body>
  <section>
  <div class="box1"></div><div class="box1"></
div><div class="box2"></div><div
class="box1"></div><div class="box1"></div><div
class="box2"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box3"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box2"></div><div
class="box2"></div><div class="box1"></div><div
class="box1"></div><div class="box3"></div><div
class="box1"></div><div class="box1"></div><div
class="box2"></div><div class="box1"></div><div
class="box1"></div><div class="box3"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box2"></div><div class="box3"></div><div
class="box1"></div>
  </section>
</body>
</html>
```

Spaces?

- Quick aside
- Why are there no spaces between `<div>` elements in this crazy block of divs?
- Setting `display: inline-block;` means that the blocks flow inline with the rest of the content. Spaces are content! Remember our DOM
- If we had spaces in here, our layout would look ever so slightly off.
- Demo

```
<body>
  <section>
  <div class="box1"></div><div class="box1"></
div><div class="box2"></div><div
class="box1"></div><div class="box1"></div><div
class="box2"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box3"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box2"></div><div
class="box2"></div><div class="box1"></div><div
class="box1"></div><div class="box3"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box1"></div><div class="box1"></div><div
class="box2"></div><div class="box3"></div><div
class="box1"></div>
  </section>
</body>
</html>
```

Fluid-Grid



- Use percentages to specify widths instead of pixels

```
<!doctype html>
<head>
  <title>layout-width2.html</title>
  <style>
    body {
      background-color: #888;
    }
    section {
      width: 80%; /* Arbitrary. Base was 1020px */
      background-color: #fff;
      margin: 0 auto 0 auto;
    }
    div {
      box-sizing: border-box;
      display: inline-block;
      margin: 0.98%; /* 10/1020 = 0.0098 => 0.98% */
      height: 100px;
    }
    .box1 {
      width: 14.706%; /* 150/1020 = 0.14706 => 14.706% */
      background-color: lightgray;
    }
    .box2 {
      width: 31.373%; /* 320/1020 = 0.31373 => 31.373% */
      background-color: lightblue;
    }
    .box3 {
      width: 48.039%; /* 490/1020 = 0.48039 => 48.039% */
      background-color: darkgreen;
    }
    .box4 {
      display: block;
      background-color: lightg
    }
  </style>
<head>
```

oh noes! maths!

- To convert our fixed-width layout to a fluid one, we unfortunately have to do some math.
- Its not too bad

Calculating Fluid Grids

- For this design, our enclosing context is the width of our `<section>` which is 1020px

```
(original_width / context_width) * 100 = percent_width

/* Margins */
10/1020 = 0.0098 => 0.98%

/* 1 Column Box */
320/1020 = 0.31373 => 31.373%

/* 2 Column Box */
490/1020 = 0.48039 => 48.039%
```

- Then we're free to change our enclosing context to anything we want, and the children will resize

Don't Forget About `box-sizing`

- Setting the `box-sizing` property to `border-box` is key to our math working out so nicely.
- If you have to support IE7 or earlier, your math gets a lot uglier, or you have to use some sort of javascript polyfil

The `position` Property

- The `position` property controls how an elements is treated within the default flow of content.

```
position: static
position: relative
position: absolute
position: fixed
position: sticky

position: inherit
```

https://developer.mozilla.org/en-US/docs/Web/CSS/position

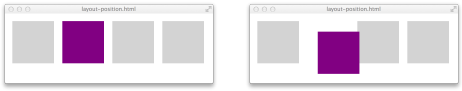
position: static

- Static positioning is sort of the normal way things are positioned. Most elements have a default position value of static.
- Static elements are placed wherever there is space next in the flow.
- `top`, `left`, `right`, `bottom` and `z-index` properties do not apply.

https://developer.mozilla.org/en-US/docs/Web/CSS/position

position: relative

- Relatively positioned elements are laid out as normal, and space for it is allocated in the flow.
- `top`, `left`, `right`, `bottom` and `z-index` properties do apply.



https://developer.mozilla.org/en-US/docs/Web/CSS/position

position: relative

```
<!doctype html>
<head>
  <title>layout-position.html</title>
  <style>
    div {
      box-sizing: border-box;
      display: inline-block;
      margin: 10px;
      height: 100px;
      width: 100px;
      background-color: lightgray;
    }
    .box2 {
      position: relative;
      top: 25px;
      left: 25px;
      background-color: purple;
    }
  </style>
<he
<bo
  <
  <
div
  <
</body>
</html>
```

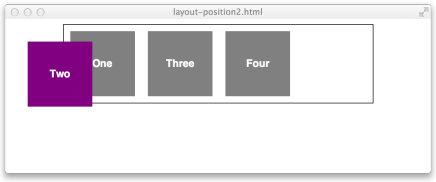
- The first image shows where the element would fall normally in the flow.
- Because it is positioned relative, top and left offsets are allowed. The element is shifted accordingly, anchored to its relative position.

position: absolute

- Absolutely positioned elements are completely removed from the flow, and no space is allocated to them.
- They are then positioned based on `top`, `left`, `right`, `bottom` and `z-index` according to their most recent positioned ancestor.
- A “positioned ancestor” is some element who’s position is not ‘`static`’.
- If no ancestor is positioned, it is displayed relative to the upper-left corner of the page.

position: absolute

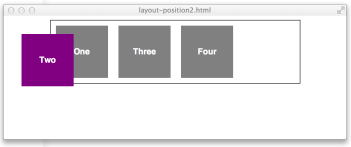
- The “Two” element will remain where it is, even as we resize the browser window.
- Notice there is no “hole” where Two would normally go.



position: absolute

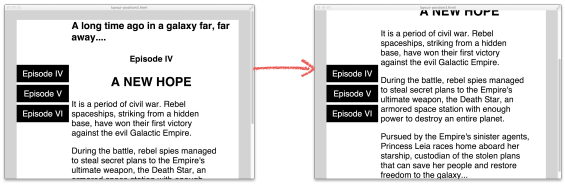
```
<!doctype html>
<head>
  <title>layout-position2.html</title>
  <style>
    * { box-sizing: border-box; }
    section {
      width: 480px;
      margin: 0 auto;
      border: 1px solid black;
    }
    div { /* Formatting */ }
    .box2 {
      position: absolute;
      top: 25px;
      left: 25px;
      background-color: purple;
    }
  </style>
<head>

<body>
  <section>
  <div>One</div><div class="box2">Two</
div><div>Three</div><div>Four</div>
  </section>
</body>
</html>
```

- Demo



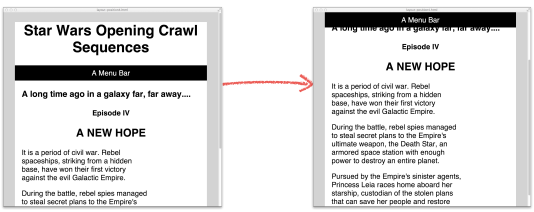
position: fixed

- Elements with a **`fixed`** position are placed relative to the browser viewport, not the page.
- They stay put as the browser content scrolls



position: sticky

- Elements with **`sticky`** positioning are placed normally within the flow, until they are about to scroll off the page. Instead of scrolling off the page, they will ‘stick’.
- Browser support for this one isn’t great yet.



`top, left, bottom, right`

- These properties define the distance between that edge of the block, and its ***nearest positioned*** ancestor.
- A ***positioned element*** is any element who's `position` property has been set to something other than `static`.
- If no ancestor is explicitly positioned, it defaults to the *window*.
- Setting `left` and `right` will implicitly define a width.
- Setting `top` and `bottom` will implicitly define a height.

`top, left, bottom, right`

- Along with `width` and `height`, these 6 properties define the position and size of a block.
- Defining any 4 of the properties is enough to implicitly define all 6, since they can be derived from one another.
- If `top`, `bottom`, and `height` are set, `bottom` is ignored.
- If `left`, `right`, and `width` are set, `right` is ignored (in left-to-right layouts).
- Example: layout-position5.html

The `float` Property

- When an element is floated it is taken out of the normal flow of the document. It is shifted to the left or right until it touches the edge of it's containing box *or another floated element*.

```
figure {
  width: 150px;
  height: 150px;
  background-color: green;
  float: left;
}
```

The `float` Property

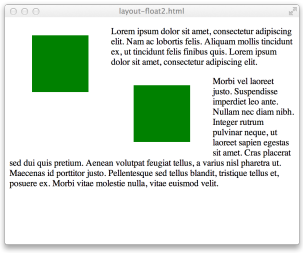
- Sometimes floats cause problems.
- Say you have a blog, where each entry has an associated image.
- Maybe you start with this HTML
- How's this look?

```
<!doctype html>
<head>
  <title>layout-float2.html</title>
  <style>
    figure {
      width: 100px;
      height: 100px;
      background-color: green;
      float: left;
    }
  </style>
<head>

<body>
  <section>
    <article>
      <figure></figure>
      <p>Lorem ipsum dolor…</p>
    </article>

    <article>
      <figure></figure>
      <p>Morbi vel laoreet justo…</p>
    </article>
  </section>
</body>
</html>
```

The `float` Property

```
<!doctype html>
<head>
  <title>layout-float2.html</title>
  <style>
    figure {
      width: 100px;
      height: 100px;
      background-color: green;
      float: left;
    }
  </style>
<head>

<body>
  <section>
    <article>
      <figure></figure>
      <p>Lorem ipsum dolor…</p>
    </article>

    <article>
      <figure></figure>
      <p>Morbi vel laoreet justo…</p>
    </article>
  </section>
</body>
</html>
```

- Hmm not so good
- We want each `<article>` to start on its own line



The `clear` Property

```
<!doctype html>
<head>
  <title>layout-float2.html</title>
  <style>
    figure {
      width: 100px;
      height: 100px;
      background-color: green;
      float: left;
    }
    article { clear: both; }
  </style>
<head>

<body>
  <section>
    <article>
      <figure></figure>
      <p>Lorem ipsum dolor…</p>
    </article>

    <article>
      <figure></figure>
      <p>Morbi vel laoreet justo…</p>
    </article>
  </section>
</body>
</html>
```

- `clear` to the rescue
- Layout keeps moving down the page until all floated elements have been cleared.

Color in CSS

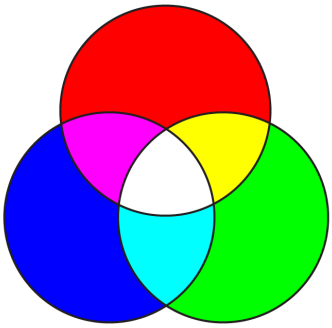
Color in CSS

- Light based color displays (CRTs, LCDs, Plasmas, etc) use a color combination system called “Additive Color”
  - http://en.wikipedia.org/wiki/Additive_color
- This differs from Ink based color models (Printing)
  - http://en.wikipedia.org/wiki/Subtractive_color

Additive Color Model

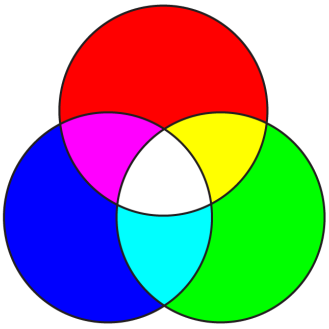
- Combine colors and intensities to produce a wide range of colors.
- Most computer displays, specifies colors in varying amounts of Red, Green, and Blue: RGB
- This is our color language for CSS

Additive Color Model



Additive Color Model

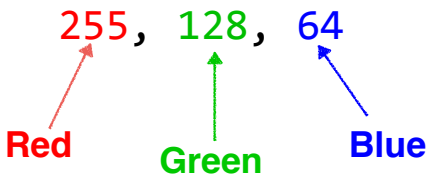
- 100% Red
- 100% Green
- 100% Blue



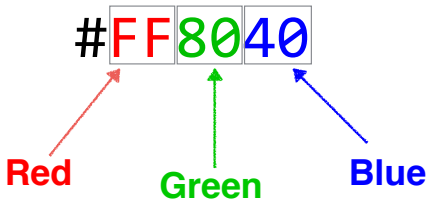
Color Notation

Color	Percent	Decimal	Hexdecimal	Name
	100% Red	255	FF	red
	100% Green	255	FF	lime
	100% Blue	255	FF	blue
	50% Blue	128	80	navy
	80% Blue	205	CD	mediumblue
	54% Blue	139	8B	darkblue

Color Mixing
Decimal Notation



Color Mixing
Hex Notation



Color Mixing

Color	Decimal	Hexdecimal	Name
	rgb(255, 0, 0)	#FF0000	red
	rgb(0, 255, 0)	#00FF00	lime
	rgb(0, 0, 255)	#0000FF	blue
	rgb(220,20,60)	#DC143C	crimson
	rgb(160,82,45)	#A0522D	sienna
	rgb(128,128,128)	#808080	gray

Hex Abbreviation

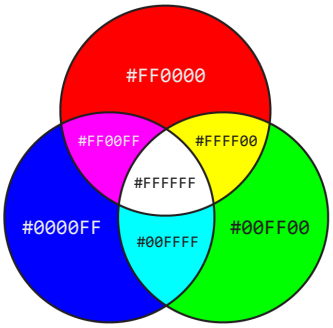
- If each color pair of digits is the same digit, you can represent the color with 3 characters instead of 6

#FFFFFF → #FFF

#88FF88 → #8F8

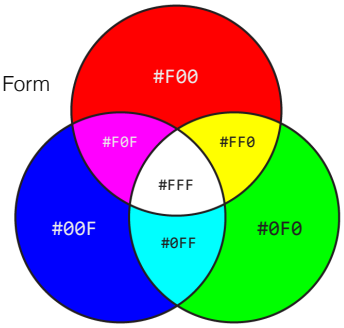
#88CDCD nope! #8CDCD

Additive Color Model



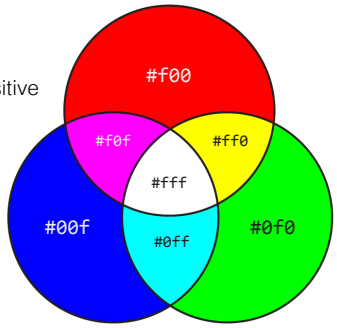
Additive Color Model

- Short Form



Additive Color Model

- Case Insensitive



Color Declarations

```
<!doctype html>
<head>
  <title>css-color.html</title>
  <style>
    body {
      background-color: #ccc;
      font-weight: bold;
      font-size: 1.5em;
    }
    .c1 { color: red; }
    .c2 { color: purple; }
    .c3 { color: #076873; }
    .c4 { color: rgb(180, 255, 80); }
  </style>
<head>

<body>
  <p class="c1">Red</p>
  <p class="c2">Purple</p>
  <p class="c3">UA River</p>
  <p class="c4">Pale Yellow</p>
</body>
</html>
```

Color Declarations

- Names, Hex, and the `rgb()` notation (not a function)

```
.c1 { color: red; }
.c2 { color: purple; }
.c3 { color: #076873; }
.c4 { color: rgb(180, 255, 80); }
```

Color Declarations

- rgb( ) notation can use percentages *OR* decimals

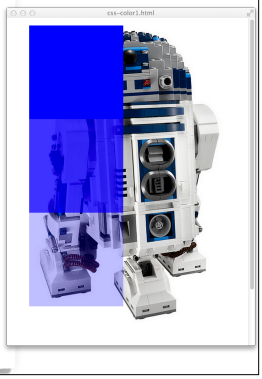
```
.c4 { color: rgb(75%, 100%, 50%); }
.c4 { color: rgb(180, 255, 80); }

 /* Cannot Mix % and decimal! */

.c4 { color: rgb(75%, 255, 80); }
```

`rgba()` notation

- `rgba()` defines a color and an opacity.
- “a” for alpha-channel
- Opacity is from 0.0 to 1.0
- `rgb(128, 128, 128)` /* 100% opaque */
- `rgba(128, 128, 128, 1.0)` /* 100% opaque */
- `rgba(128, 128, 128, 0.5)` /* 50% opaque */

`rgba()` notation

```
<!doctype html>
<head>
  <title>css-color1.html</title>
  <style>
    figure { position: relative; }
    img { position: absolute; }
    div {
      width: 200px;
      height: 200px;
      position: relative;
    }
    .c1 { background-color: rgba(0, 0, 255, 1.0); }
    .c2 { background-color: rgba(0, 0, 255, 0.75); }
    .c3 { background-color: rgba(0, 0, 255, 0.25); }
  </style>
<head>

<body>
  <figure>
    <img src="img/droid.jpg" alt="Lego R2D2">
    <div class="c1"></div>
    <div class="c2"></div>
    <div class="c3"></div>
  </figure>
  <div>

  </div>
</body>
</html>
```

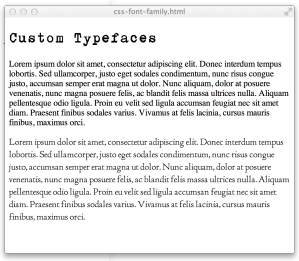


Custom Fonts

- @font-face declarations
- Load fonts directly from the web.
- Watch for licensing!
- Google Fonts: https://www.google.com/fonts
- Open Font Library: http://openfontlibrary.org/

@font-face Rules

```
<!doctype html>
<head>
  <title>css-font-family.html</title>
  <style>
    @font-face {
      font-family: 'CoelacanthLight';
          src: url('fonts/CoelacLight.otf') format('opentype');
      font-weight: normal;
      font-style: normal;
    }
    @font-face {
      font-family: 'Warenhaus';
          src: url('fonts/Warenhaus-Standard.otf')
      font-weight: normal;
      font-style: normal;
    }
    h1 { font-family: 'Warenhaus'; }
    .alt { font-family: 'CoelacanthLight'; }
  </style>
<head>

<body>
  <h1>Custom Typefaces</h1>
  <p>
  Lorem ipsum dolor sit…
  </p>
  <p class="alt">
  Lorem ipsum dolor sit…
  </p>
</body>
</html>
```



Images in CSS

CSS Images

- Mostly defined as background images and border images.
- Inline images are defined in **<img>** elements.
- Why define an image with CSS instead of using an **<img>** element?
  - Easiest way to get background images with text on top.
  - Media queries (See responsive design) can be used to select different images depending on browser conditions.

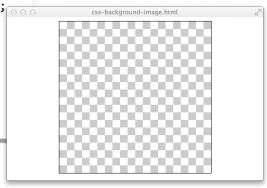
background-image

- The basic property is `background-image` for a block.
- Allows us to choose an image via a URL
  - Can be a local file, or any valid URL

background-image

```
<!doctype html>
<head>
  <title>css-background-image.html</title>
  <style>
    div {
      width: 320px;
      height: 320px;
      margin: 0 auto;
      border: 1px solid black;
      background-image: url('img/grid.png');
    }
  </style>
</head>

<body>
  <div></div>
</body>
</html>
```

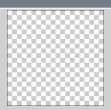
- `grid.png` Image: 32px x 32px



background-repeat

- By default, background image will tile in the X and Y directions to fill the block.
- Change this by altering the `background-repeat` properties.

Value	Behavior
`repeat`	Repeats in both the X and Y directions. This is the default
`repeat-x`	Repeats only in the X direction
`repeat-y`	Repeats only in the Y direction
`no-repeat`	Does not repeat

Value	Behavior
`background-repeat: repeat;`	
`background-repeat: repeat-x`	
`background-repeat: repeat-y`	
`background-repeat: no-repeat`	

background-position

- By default, background image will be placed at 0,0 – the top-left corner of the block
- Change this by altering the `background-position` properties.

Value	Behavior
`background-position: top center;`	
`background-position: 32 32;`	
`background-position: 50% 50%;`	
`background-position: right 50%;`	

background-size

- By default, background images will be sized at their native pixel size
- Change this by altering the `background-size` property.

Value	Behavior
`background-size: 64px 64px;`	
`background-size: 50%;`	
`background-size: 32px 100%;`	
`background-size: 100%;`	

Value	Behavior
`background-size: 32px 100%;` `background-repeat: repeat-x`	
What about Photographs?	
`background-size: cover;`	
`background-size: contain;` `background-repeat: no-repeat;`	

Pseudo-Selectors

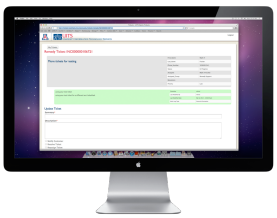
Pseudo-Classes

- Pseudo Classes are prefixed with a single colon
  - a:link
  - a:visited
  - a:hover
  - div:nth-child(…)

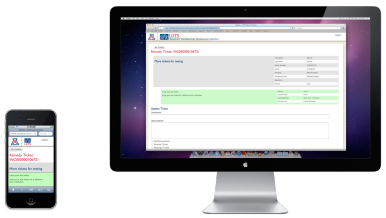
Pseudo-Elements

- Pseudo Elements are prefixed with a double colon
  - p::first-line
  - div::after
  - div::before

Responsive Design



The Good Old Days



Welcome To Mobile!

Hello iPad

Ahhh Slow Down!

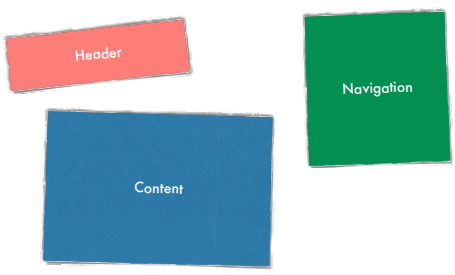
Responsive Design

- Think about mobile first
- Stop thinking in pixels
- Progressive Enhancement
- Build up features as the screen gets bigger

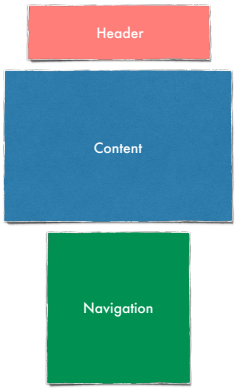
Show Me The Goods

- http://www.alistapart.com/d/responsive-web-design/ex/ex-site-FINAL.html
- http://www.css-tricks.com/
- http://bostonglobe.com/
- http://www.smashingmagazine.com/
- http://ethanmarcotte.com/

How Do We Do It?

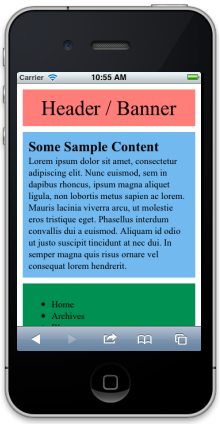


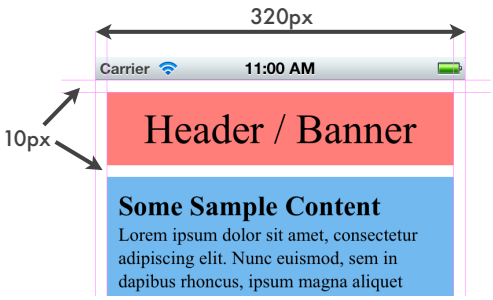
Mobile First



Design

In a browser





Maths

Pixel Based

```
{
  margin: 10px;
  padding: 10px;
}
```

Maths

Responsive

```
/*
 * Size ÷ Context = Relative
 * 10 ÷ 320 = 0.03125
 */

{
  margin: 3.125%;
  padding: 3.125%;
}
```

Media Queries

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

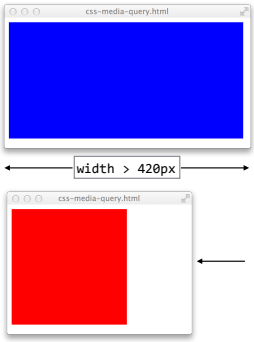
- Media Queries act like conditionals around a set of rules.
- The rules within a media query are only applied when the conditions defined in the media query are met.

This `<div>` rule applies only when the device is `screen`, and the screen width is less than 420px

```
@media screen and (max-width: 420px) {
  div {
    background-color: red;
    width: 200px;
  }
}
```

Media Queries

```
<!doctype html>
<head>
  <title>css-media-query.html</title>
  <style>
    div {
      width: 400px;
      height: 200px;
      background-color: blue;
    }
    @media screen and (max-width: 420px) {
      div {
        background-color: red;
        width: 200px;
      }
    }
  </style>
<head>

<body>
  <div></div>
</body>
</html>
```



Demo

- responsive.html

What About IE?

- Of course, Media Queries don't work in IE < 9
- But we can stand on other's shoulders
- respond.js https://github.com/scottjehl/Respond
- css3-mediaqueries-js

To Read

- http://www.alistapart.com/articles/responsive-web-design/
- http://www.alistapart.com/articles/dao/
- http://www.abookapart.com/products/responsive-web-design
- http://goldengridsystem.com/

fin