



# GENERAL ASSEMBLY

Code in One Day:  
**HTML & CSS Crash Course**



# GENERAL ASSEMBLY

Empowering people to pursue work they **love**.

## What We Teach

Coding

UX & Design

Data

Digital Marketing

Product Management

# Continue Your Education

	Front-End Web Development	Web Development Immersive
Format	10-week part-time, evening course	12-week full-time course, Monday-Friday
Outcome	Learn to develop beautiful web pages with HTML, CSS, & JavaScript.	Learn the skills to become an entry-level web developer and the resources to get a job in this intensive program.
Tuition	\$3,950	\$14,950
Learn More	<a href="http://www.ga.co/fewdbos">www.ga.co/fewdbos</a>	<a href="http://www.ga.co/wdibos">www.ga.co/wdibos</a>

# Introductions

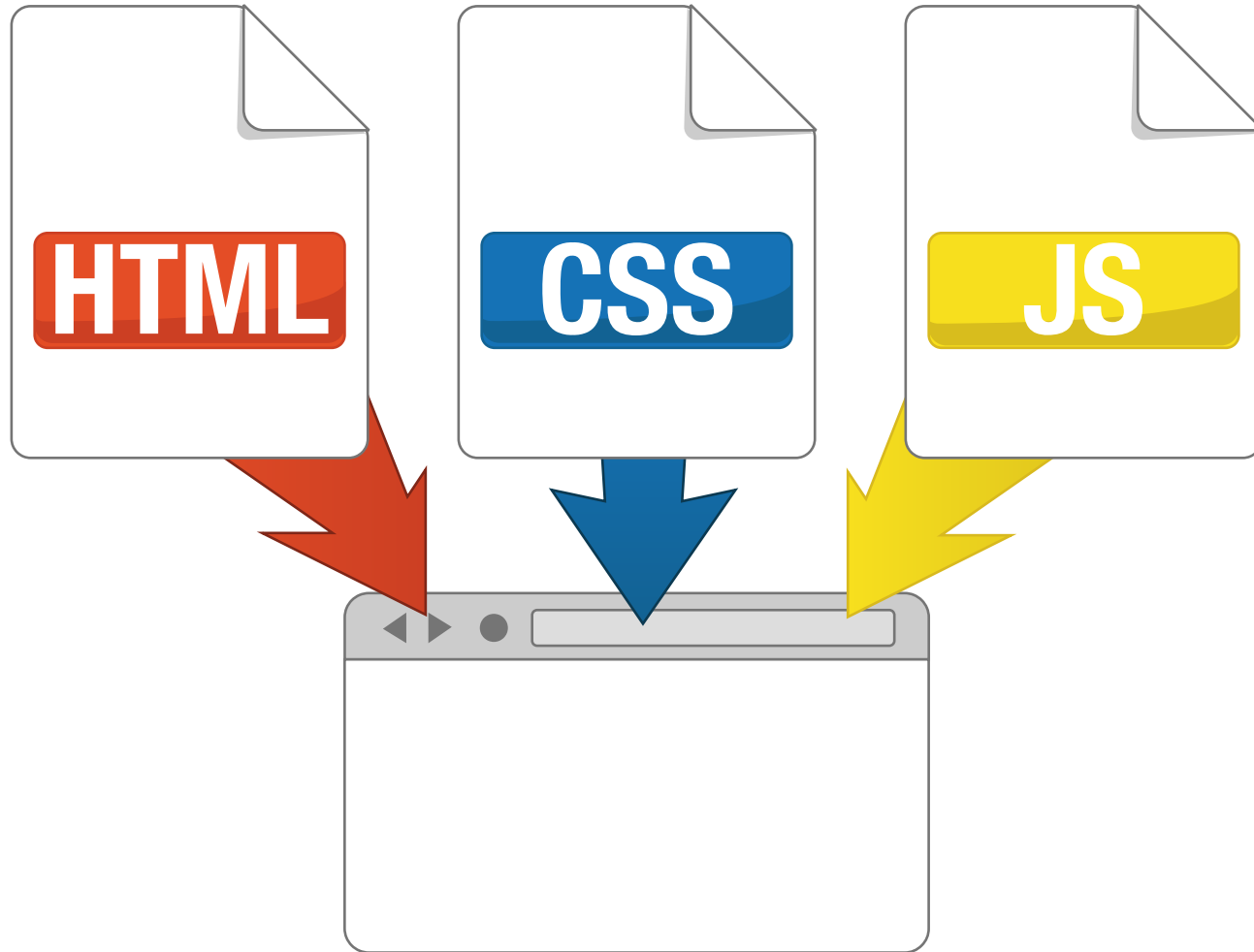


# What We'll Cover

- Building Web Pages with HTML & CSS
- How Stuff Behaves in the Weird World of the Web
- Where to Go from Here

# Primer

# How Web Pages Work



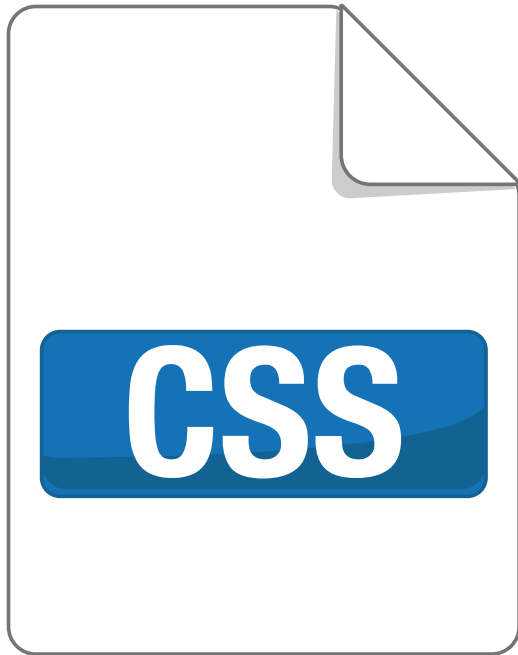
# Hypertext Markup Language



HTML provides the *structure* for your web pages

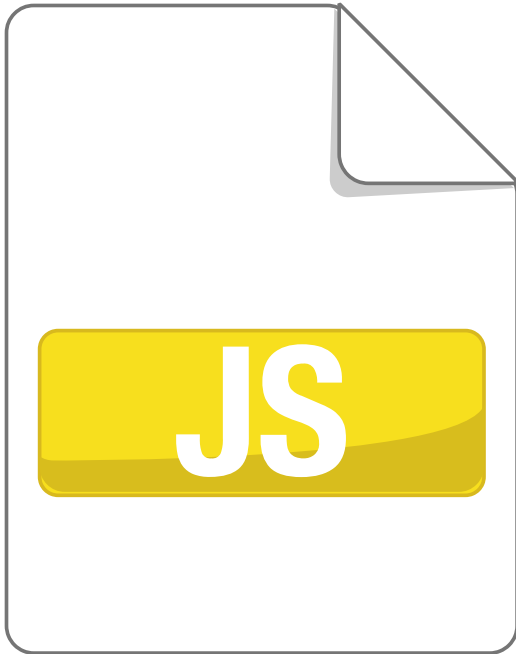


# Cascading Style Sheets



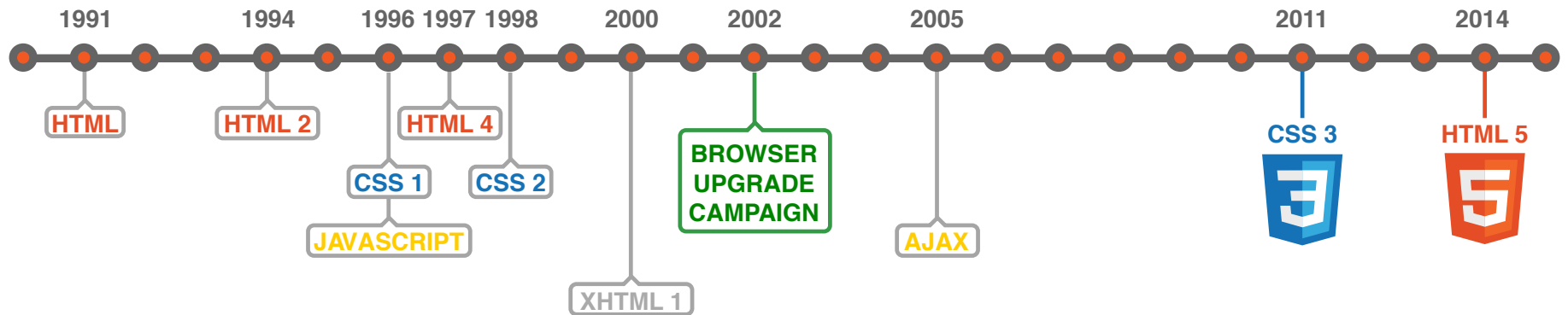
CSS defines how the elements of the web page should *look*

# Javascript



Javascript is used to add *interactivity*

# History of Web Standards



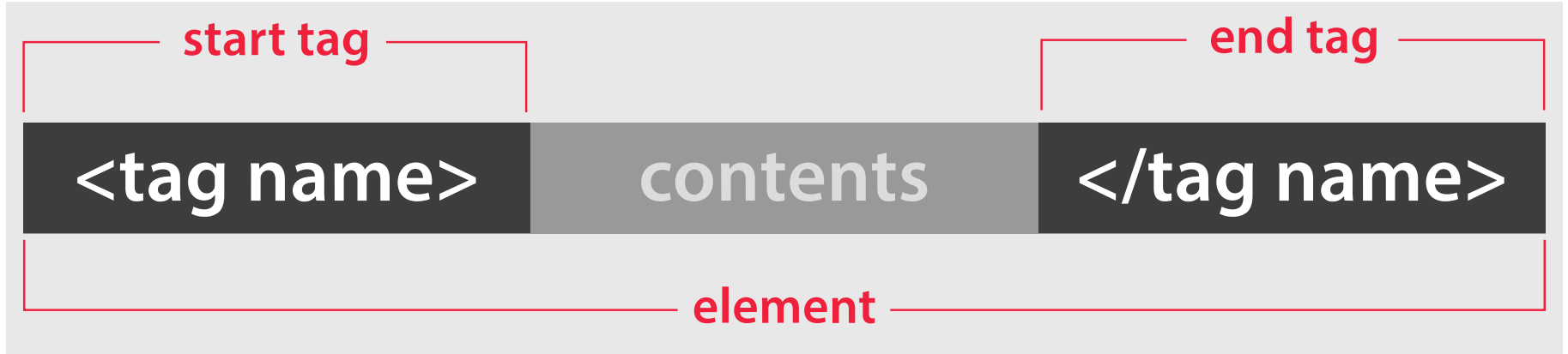
# Part 1

# Objectives: Part 1

- Describe the anatomy of a web page and create the basic structure of a page in HTML
- Apply HTML tags to web page content
- Style web pages with CSS
- Link external CSS files

# HTML Basics

# HTML Syntax



► **HEADS UP:** Some tags only have a start tag.

# HTML Syntax





# What's up DOCTYPE?

- The HTML5 doctype tells the browser to behave **strictly** according to the standards
- Case-insensitive but conventionally written as:

```
<!DOCTYPE html>
```

# Anatomy of a Web Page

Every HTML page has the same foundational structure

```
<!DOCTYPE html>
<html>
<head>
  <title>Your Title Here</title>
</head>
<body>
  Your Content Here
</body>
</html>
```

# Content Tags

Your content needs structure too! Content tags are used to delineate different page contents. They go in between the `<body> ... </body>` tags in your html page.

## Content Tags

# Heading Tags

```
<h1>Largest Heading</h1>
```

```
<h2> . . . </h2>
```

```
<h3> . . . </h3>
```

```
<h4> . . . </h4>
```

```
<h5> . . . </h5>
```

```
<h6>Smallest Heading</h6>
```

## Content Tags

# Text Elements

```
<p>paragraph text</p>  
<code>computer code</code>
```

## Content Tags

# Unordered list

```
<ul>...</ul>
```

## Content Tags

# List item

```
<ul>  
  <li>Item</li>  
  <li>Another Item</li>  
</ul>
```

## Content Tags

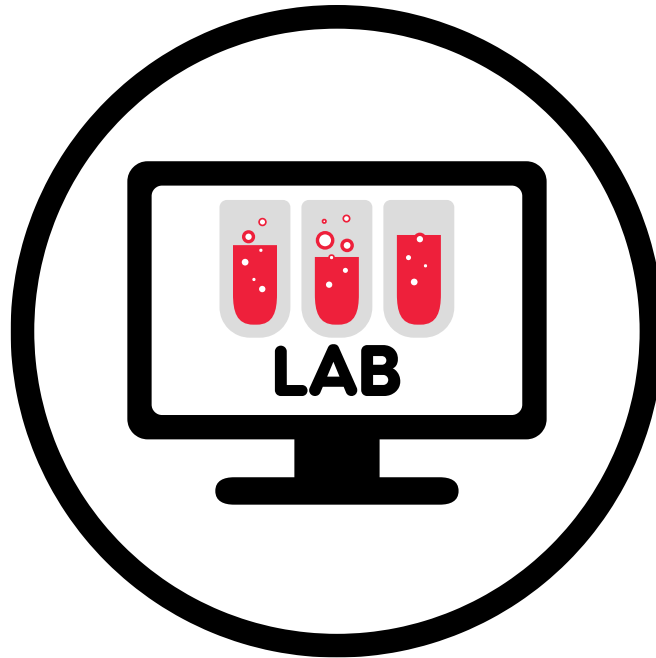
# Links

```
<a href="url">Link Text</a>
```





# General Assembly Press Release



**Cookie Recipe HTML**

# Lab Setup

<https://github.com/jmeade11/crashcourse>

1. Go to the URL above
2. Click the green **Clone or download** button
3. Click the link to **Download ZIP**
4. Uncompress the crashcourse-master.zip file and move it to your desktop
5. Open the crashcourse-master and navigate to the labs folder

lab1

# Lab Tasks

<https://github.com/jmeade11/crashcourse/labs>

1. In Sublime Text, choose **File>Open** and open the the labs folder
2. Expand the lab folder, then expand the **lab1** folder and click on the index.html file to open it
3. Add your HTML boilerplater
4. Copy the recipe from the online instructions and paste it in the body
5. Use what you've learned to add tags to the recipe
6. Control + click on the file in Finder and choose **Open With > Chrome** to review your work

# CSS Basics

# Selectors, Properties and Values (Oh My!)

```
selector {  
  property: value;  
}
```

# CSS Selectors

Selectors target the elements you want to style

- **Element Tags**
- Classes & IDs
- Attributes
- Pseudo Classes
- Combinators

# Properties

Properties are the styles you want to apply

- border
- color
- background-color



# Values

Properties each have a set of specific values that they can accept

- border has values for the width, color and style
- color and background-color accept a color value

# Style Tag

CSS can be added to your HTML page with a style tag.

```
<!DOCTYPE html>
<head>
  <title>My Page Title</title>
  <style><!--Style tags only go inside your head tags-->
    h1 {
      color: blue;
    }
  </style>
</head>
<body>
  ...
```

# CSS Example

```
<style>
  h1 {
    color: darkblue;
    font-family: sans-serif;
  }
  h2 {
    color: maroon;
    font-family: sans-serif;
  }
  body {
    background-color: lightgray;
    border: 5px solid darkblue;
    padding: 5px;
  }
</style>
```

► **HEADS UP:** Property names and values are case-sensitive!

## CSS Properties

# Color Properties

The `color` property sets the font color.

```
/* make the fonts on the  
page blue */
```

```
body {  
  color: blue;  
}
```

`background-color` sets the background color.

```
/* make the page background  
a light blue */
```

```
body {  
  background-color: aliceblue;  
}
```

## CSS Properties

# Border Properties

```
/* border is shorthand for:  
   border-width  
   border-style  
   border-color */  
  
ul {  
  border: 1px solid darkgray;  
}
```

Borders can be set for all sides or individually. For example, `border-bottom` can be used as the shorthand for the bottom border only. Properties can be specified individually as well, such as: `border-top-style`.

## CSS Properties

# Font Properties

```
/* font is shorthand for:  
   font-style, font-variant,  
   font-weight, font-size,  
   line-height, font-family; */  
p {  
  font: bold 16px sans-serif;  
}
```

Several font-related properties can be set. The font property combines them into a single shorthand property.

► **HEADS UP:** The shorthand notation requires at least the font-family and font-size, *and* font-family must be specified last.



# CSS Basics

# External Style Sheets

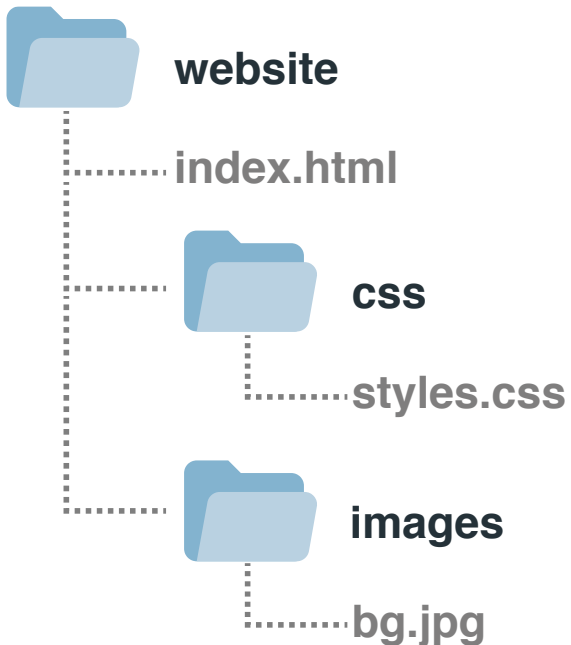


# Linking Files

```
<head>
...
<link rel="stylesheet" href="css/styles.css">
...
</head>
```

► **HEADS UP:** No <style> tags are used!

# Linking Files Relatively

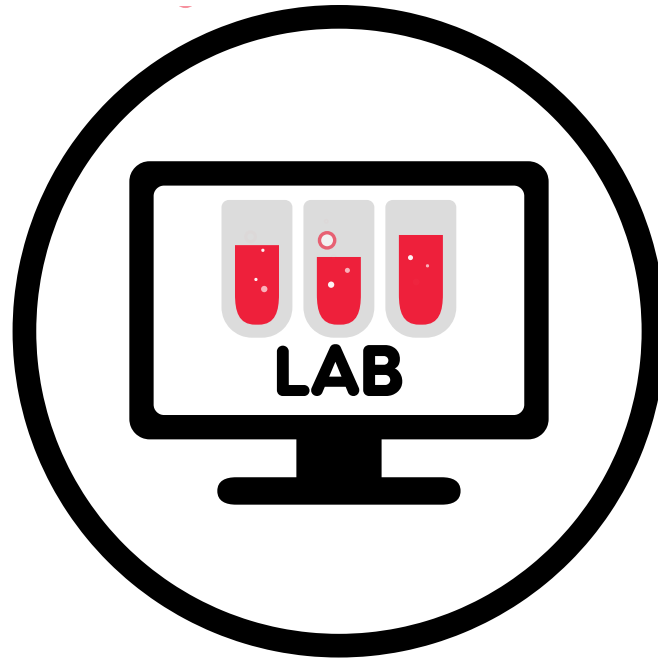


- index.html → styles.css:

```
href="css/styles.css"
```

- styles.css → bg.jpg:

```
"../images/bg.jpg"
```



# Cookie Recipe CSS Basics

lab2

# Lab Task

<https://github.com/jmeade11/crashcourse/>

1. We're going to make Grandma's cookies look better, so open the lab2 folder and use the starter index.html file.
2. Make the background for the html element the color **moccasin**.
3. Set the font for the page to **16px** and **sans-serif**.
4. Style the link color as **chocolate**
5. Add a **solid**, **15px** wide, **chocolate** colored border around the body.
6. The body should also have its background set to **oldlace**, font color set to **saddlebrown**.
7. **BONUS** Change the h1 to **40px** and **center** align it.

# Part 2

# Quick Review

- Describe the different parts of an HTML page
- What is the doctype used for?
- What tag is used to link to an external stylesheet?

# Objectives: Part 2

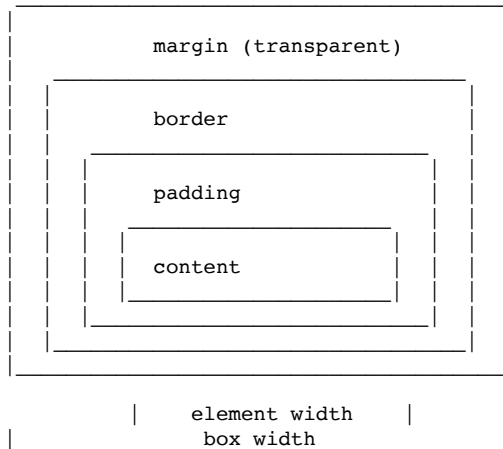
- Understand the CSS box model
- Identify block and inline elements
- Use fixed and relative units of measurement
- Add colors with transparency
- Add images to the page and background

# CSS Box Model



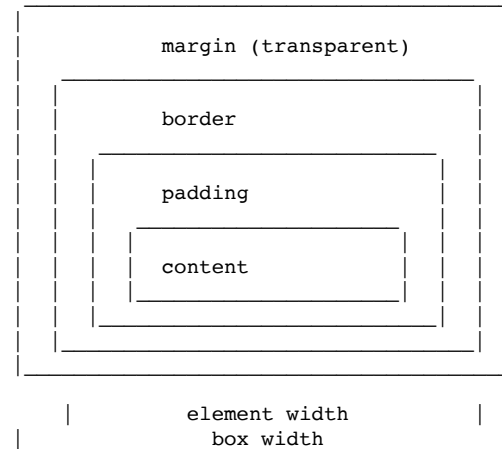
# The Box Model

## Box Model Standard



*Cascading Style Sheets, level 1*, W3C Recommendation  
17 Dec 1996, revised 11 Apr 2008

## Microsoft Box Model 🍌

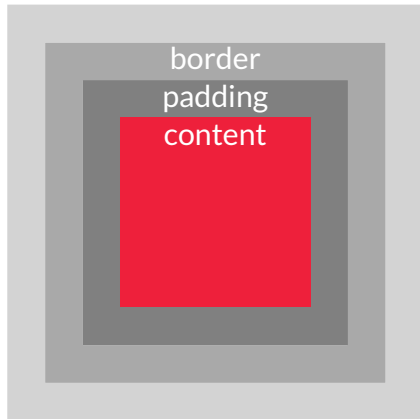


*Revenge of the IE Box Model?* Jeff Kaufman, February  
18, 2012

# Box-Model Mayhem

Content Box

220px



Border Box

140px



Each box height and width is set to **100px** with padding and margin set to **20px**.

The difference is **box-sizing**!



# CSS Box Model

# Block and Inline Elements

# Block vs Inline Elements

## Block Elements

- ***Block*** elements from sitting next to them
- Size can be **specified** with height and width
- Can enclose other block or inline elements

## Inline Elements

- Wrap inside containing elements
- Size is the height and width of contents only
- ***Should not*** enclose block elements

# Block Example

```
<div style="height: 100px; width: 50%;">  
  <p>contents</p>  
</div>  
<span>I've been blocked!</span>
```



contents

blocked area

I've been blocked!

# Inline Example

```
<span style="height: 100px; width: 25%; background: red;">Empty</span>  
<span style="height: 100px; width: 25%; background: red;">spans</span>  
<span style="height: 100px; width: 25%; background: red;">disappear</span>  
<span style="height: 100px; width: 25%; background: red;"></span>
```

Empty spans disappear



# Block and Inline Elements



# Units of Measure & CSS Colors

# Units of Measure

Value	Description
<b>px:</b>	A fixed (aka absolute) value in pixels
<b>em:</b>	Relative to the font-size of the element (2em = 2 x the size of the current font)
<b>rem:</b>	Relative to the root element font-size
<b>vh:</b>	% of the viewport height (50vh = 50% of the viewport height)
<b>vw</b>	% of the viewport width
<b>vmax</b>	% of viewport's larger dimension
<b>vmin</b>	% of viewport's smaller dimension
<b>%</b>	It depends 🤖

\* Not comprehensive.



# Setting the Size of Things

# Color Values

- Keyword
- RGB & RGBA
- HSL & HSLA
- Hexadecimal

► **Alpha channels** are in values from 0 to 1, where zero is transparent and 1 is 100% opaque.

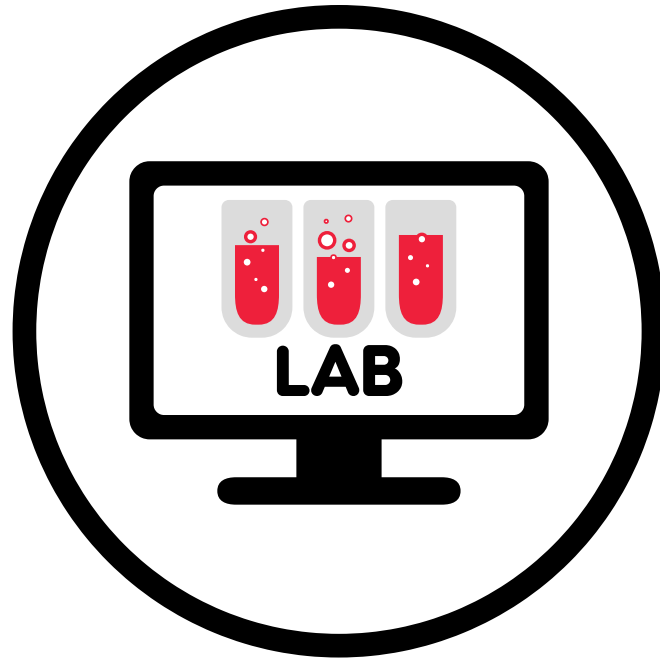
```
/* Keyword Syntax */
h1 {
  background-color: gray;
}

/* RGB & HSL Syntax */
p {
  color: rgba(0,0,0,1);
  border: 2px solid hsl(0,0%,0%);
}

/* Hexadecimal Syntax */
div {
  background: #ff0000;
}
```



**Colors**



**Cookie Recipe More CSS**

lab3

# Lab Task

<https://github.com/jmeade11/crashcourse/>

1. Grandma's recipe needs more work so open the the index.html and the styles.css files in the lab3 folder.
2. The body needs 20px of space between the border and the text.
3. Change the h1 font-size to 8% of the width of the smaller of the height or width of the viewport so it scales with the screen size!
4. Select just the words: "MMMMMM COOKIES!!!!" in the Nutrition Information and make them **bold**
5. **BONUS** Can you make more of the html background-color show around the outside of the body border?

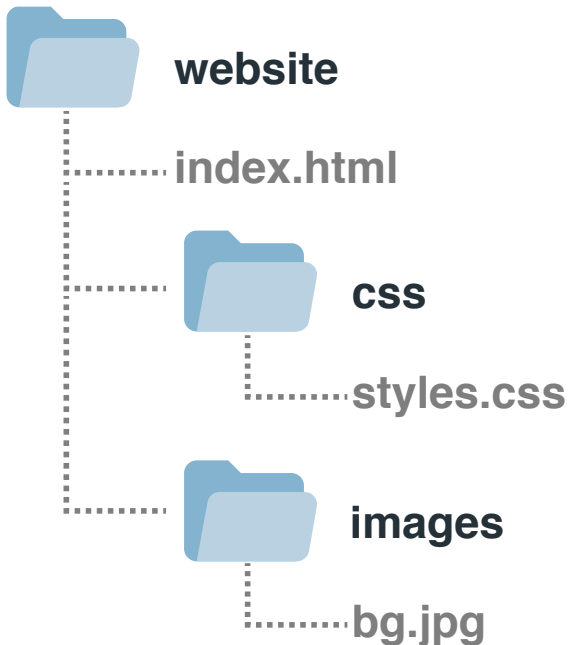
# Working with Images



# Adding Images to Your Page

- Images can be added with the `<img>` tag in HTML or in the background via CSS
- Background images are design elements only and are ignored by screen-readers

# Linking Files Refresher



- index.html → bg.jpg:

```
<img href="images/bg.jpg">
```

- styles.css → bg.jpg:

```
url("../images/bg.jpg")
```

# Content Images

- The `<img>` tag has no closing tag
- The `href` attribute links the file and is required
- The `alt` attribute is used by screenreaders and for SEO

```
<img href="images/dogs.jpg" alt="A picture of my dogs">
```

# Background Images

Background images are added through CSS

```
//The background-image property places the image in the background  
  
selector {  
    background-image: url('path/to/file');  
}
```

► **FYI:** Elements can have multiple stacked backgrounds

# Background Properties

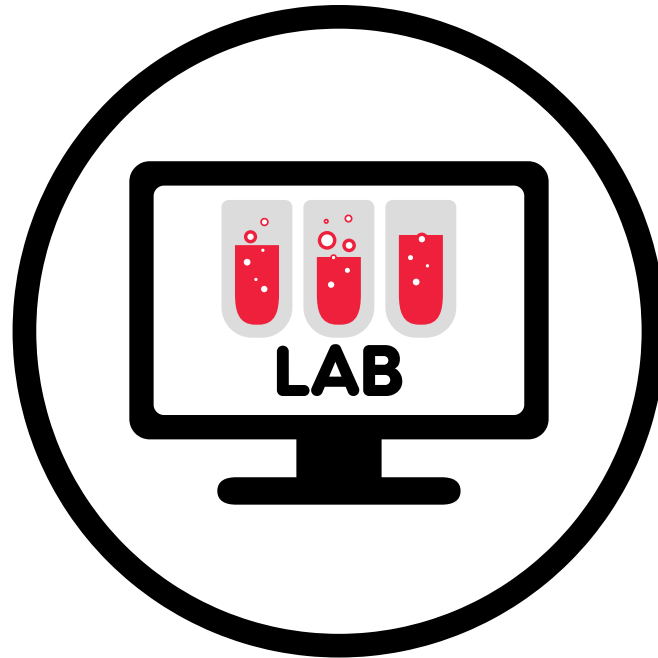
- `background-repeat`: tile the image or place once
- `background-position`: position the image in its containing element
- `background-attachment`: scroll with the page or remain fixed in one place
- `background-size`: the size of the image in the background
- `background-clip`: crop the image at the content-box, border-box or padding-box
- `background-origin`: place the image origin at the content-box, border-box or padding-box

# Background Image Syntax

```
header {  
    background-image: url('../images/bg.jpg');  
    background-size: cover;  
}  
section { /* Multiple Backgrounds */  
    background-image: url('../images/logo.png'), url('../images/bg.jpg');  
    background-repeat: no-repeat, repeat;  
    background-position: bottom right, top left;  
}  
main { /* Shorthand Format */  
    background: url('../images/logo.png')  
                no-repeat  
                bottom right / 30%  
                fixed;  
}
```



# Working with Images



**Cookie Recipe Images**



lab4

# Lab Task

<https://github.com/jmeade11/crashcourse/>

1. Grandma is pretty happy but thinks her recipe page could use some pizzaz. Open the the index.html and the styles.css files in the lab4 folder.
2. Insert the cookie image into the page above the first heading.
3. Set the image width to be 100% so it scales to fit the width of its container.

# Part 3

# Quick Review

- What is one type of color value?
- What's the difference between the content-box and border-box properties?
- What tag do you use to add an image to your HTML?

# Objectives: Part 3

- Use classes, IDs, and combinators to apply styles
- Understand how styles inherit/cascade
- Describe how conflicting styles are applied

# CSS Classes & IDs

# Selector Types

- Element Tags
- **Classes & IDs**
- Combinators
- Attributes
- Pseudo Classes & Pseudo Elements

# Classes & IDs

- Classes and IDs allow us to target elements without having to use the tag as a selector
- You can combine them with other selectors

# IDs

- An ID may only be used once on a page
- An element may only have one ID

```
<div id="extra-special">
```

```
#extra-special {  
  ...  
}
```



# Classes

- Classes are reusable as many times as you want
- An element can have as many classes as you want

```
<div class="big primary">
```

```
.primary {  
  ...  
}  
.big {  
  ...  
}
```



# Using Classes & IDs

# Combinators

# Using Combinators

- Descendant: space
- Child: >
- General Sibling: ~
- Adjacent Sibling: +



# Using CSS Combinators



**Game Time! CSS Diner**

# Cascading & Inheritance

# CSS Cascades

- CSS properties inherit their values from their ancestors
- Properties can be overridden when you provide a rule that has more specificity
- Any rules that are not specifically overridden continue to be inherited



# Selector Weighting

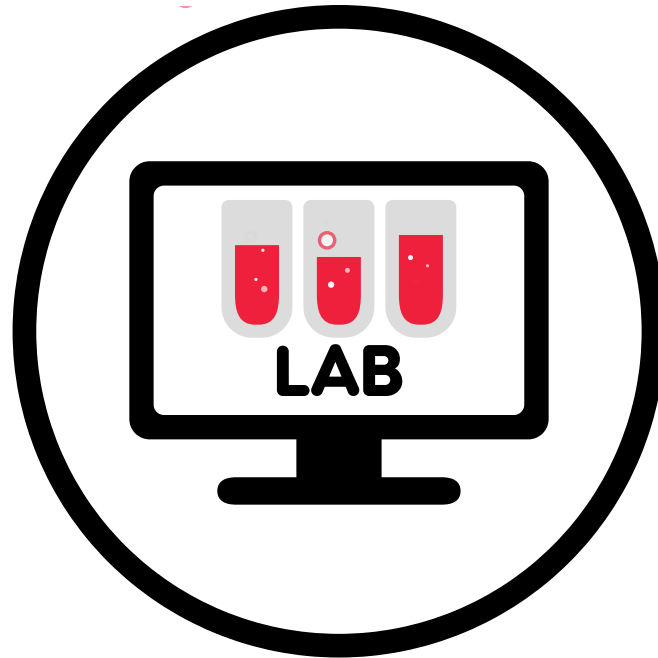
1. Inline Styles (highest)
2. IDs
3. Classes & Attributes
4. Element Tags

# When Specificity Matters

- For identical rules, the last declaration for wins!
- Styles added directly to an element are the last that will be rendered.
- One exception is the special `!important` attribute.
- The more specific the rule the more importance it is given.
- When there are multiple rules that contradict one another, the specificity and weighting rules are applied.



# Demystifying Cascade



# Cookie Recipe Classes & IDs

lab5

# Lab Task

<https://github.com/jmeade11/crashcourse/>

1. Grandma is pretty much over-the-moon now. Let's do a few clean up tasks in our CSS and HTML. Open the index.html and styles.css files in lab5 folder.
2. The h1 is centered, but we want to also center the h4 tag that credits grandma and the paragraph with the link at the bottom of the page.
3. Let's group each of the main areas of the recipe (ingredients, instructions, and nutrition) inside a set of section tags.
4. Give each one a corresponding id.

# Part 4

# Quick Review

- What does cascade mean?
- What selector is weighted highest?
- Is there any way to override a style attribute on a tag?

# Objectives: Part 4

- Use floats to have more control over your layout
- Be aware of more modern method for layout control
- Review some helpful shorthand properties




# CSS Floats

# What is Float?

Float places an element on the left or right of its container and allows other elements to wrap around it.

# No Float

```
div {  
  float: none; //default  
}
```




Float  
me

We were the people who were not in the papers. We lived in the blank white spaces at the edges of print. It gave us more freedom. We lived in the gaps between the stories.

# Float Left

```
div {  
  float: left;  
}
```



Float  
me

We were the people who  
were not in the papers.  
We lived in the blank  
white spaces at the edges of print. It  
gave us more freedom. We lived in  
the gaps between the stories.

# Float Right

```
div {  
  float: right;  
}
```

We were the people who  
were not in the papers.  
We lived in the blank  
white spaces at the edges of print. It  
gave us more freedom. We lived in  
the gaps between the stories.

Float  
me

# Float Property

The `float` property accepts the values:

- `right`
- `left`
- `none` (default)
- `initial` (resets to the default)
- `inherit` (gets its value from its ancestor)

# Make it Stop!

Everything after the floated element in your markup will float. To cause elements to stop wrapping, use the ***clear*** property to the first element you want to go on its own line.

# Clear Property

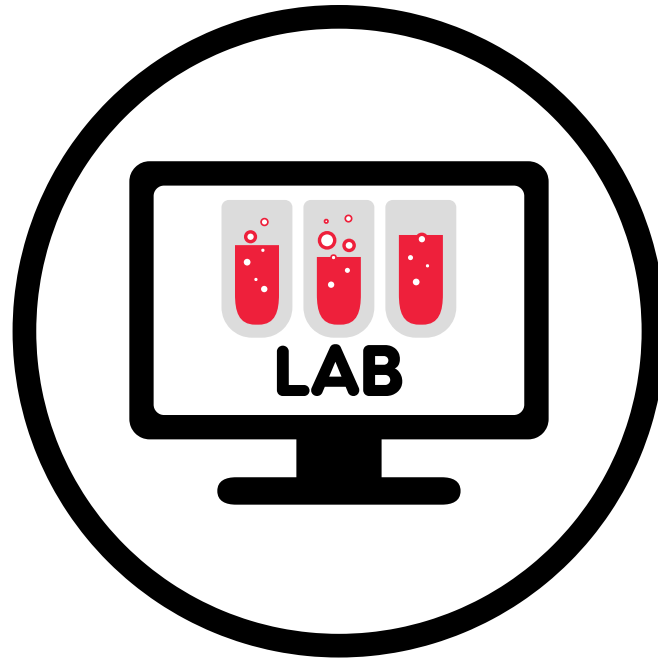
The `clear` property accepts the values:

- `right`
- `left`
- `both`
- `none` (default)
- `inherit` (gets its value from its ancestor)





# Floating Elements



# Cookie Recipe Floats

lab6

# Lab Task

<https://github.com/jmeade11/crashcourse/>

1. Grandma says it would be a lot easier to read the recipe if you could see both the ingredients list and instructions at the same time. Let's open up the css and html files in the lab6 folder and make her happy.
2. You'll need to set both the ingredients and instruction sections to be 50% in width and then float them both to the left, so they fit next to one another.
3. Make sure you address the nutrition section so that it doesn't float with the other sections.
4. Lastly, let's give the nutrition section some additional padding, but just on top! Make it 20px.

# Go Do Awesome Things!

[www.linkedin.com/in/jenniferannmeade](http://www.linkedin.com/in/jenniferannmeade)