

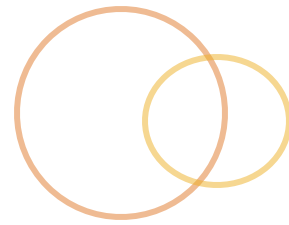
learning spike

HTML & CSS

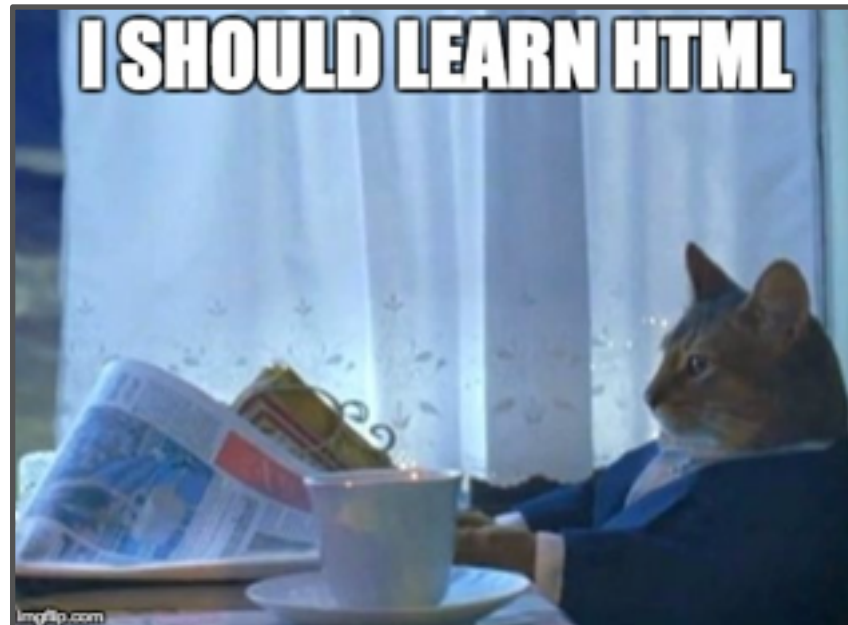
Ryan Morris
@mrmorris



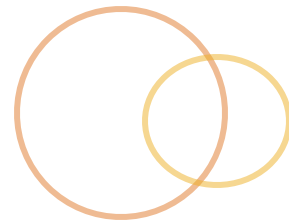
Introductions



- About me...
- About you...
 - What do you do?
 - What is your programming background?
 - What do you hope to gain from this course?



How the class works



- ⦿ Mixture of labs and lecture

- ⦿ Informal

 - ⦿ Stop me anytime

 - ⦿ Discussion > Lecture

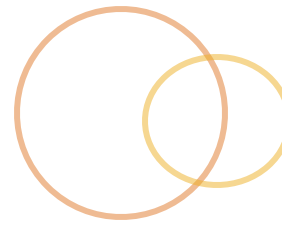
 - ⦿ Outline is flexible

 - ⦿ **There is too much to cover** so we'll adjust as needed

- ⦿ You'll help define areas of focus

- ⦿ Class assessment towards the end of the day

Goals of our spike



- Familiarity with HTML
 - Page structure
 - Lots of elements
- Familiarity with CSS
 - Selectors
 - Cascading & Specificity
- What is HTML5 and CSS3
- Some specifics:
 - Tables, Forms, Audio, Video
- Exposure to web development basics

I'm not planning to cover

- * JavaScript or the DOM
- * HTML5 in-depth
- * CSS3 flex or grids
- * LESS/SASS
- * Workflow tools
- * Web Frameworks

~Mostly for beginners~

~You should be familiar with the internet~

~let me know what you're interested in~

Resources



🕒 Guides

🕒 <http://howtocodeinhtml.com/>

🕒 Documentation

🕒 <https://developer.mozilla.org/en-US/docs/Web>

🕒 <https://www.html5rocks.com/en/resources.html>

🕒 <http://htmlreference.io/>

🕒 *Google it.*

🕒 Validator

🕒 <https://validator.w3.org/>

🕒 Compatibility

🕒 <http://caniuse.com>

Get the most out of the class



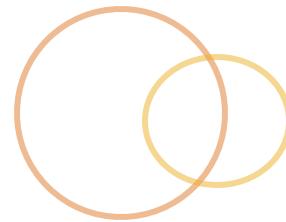
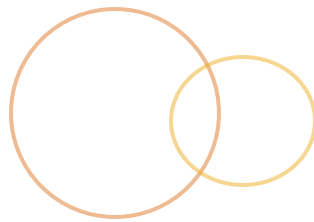
- ◎ Ask **questions!**
- ◎ Do the **labs** (pair up if needed)
- ◎ Be **punctual**
- ◎ **Avoid distractions**
- ◎ Master your **google-fu**
- ◎ **Play along**
- ◎ Don't be afraid to **break stuff**

PDF for today



- 🕒 In case you didn't quite catch something, or can't see the screen well:
 - 🕒 Head to my repository
 - 🕒 ###
 - 🕒 Go to “/docs” folder
 - 🕒 Or just download the whole repo...

Our toolkit



🕒 A browser with dev tools

- 🕒 Preference for Chrome in class
- 🕒 Open your browser and hit F12 or alt/opt/⌘ - ⌘ - i

🕒 Our **web editor**, codepen

🕒 <https://codepen.io/>

🕒 Please sign up now!

🕒 Don't like/get codepen?

- 🕒 We can also work from our own **web servers**
- 🕒 or just **locally**



Everyone OK with the above?

Wizard check

- OK with basic HTML?
 - Able to write a **<form>** from scratch?
 - Difference between **<div>** and ****?
 - When to use **id** versus **class**?
- OK with CSS?
 - **#container** vs **.container**?
 - **position: absolute**;
- What's special about HTML5?
 - Anyone using HTML5 yet?



the basics

PRIMER

The World Wide Web



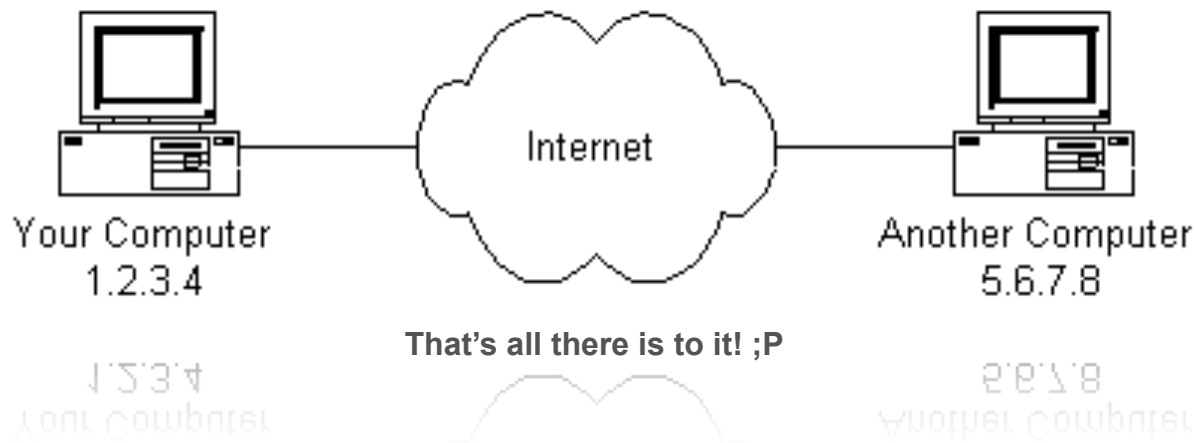
What is the web?



The World Wide Web



- ⦿ A network of **documents** linked to one another
- ⦿ With **text** and **media**
- ⦿ And **behavior** (usually JavaScript)
- ⦿ **Links** out to other pages or resources



Simple site anatomy



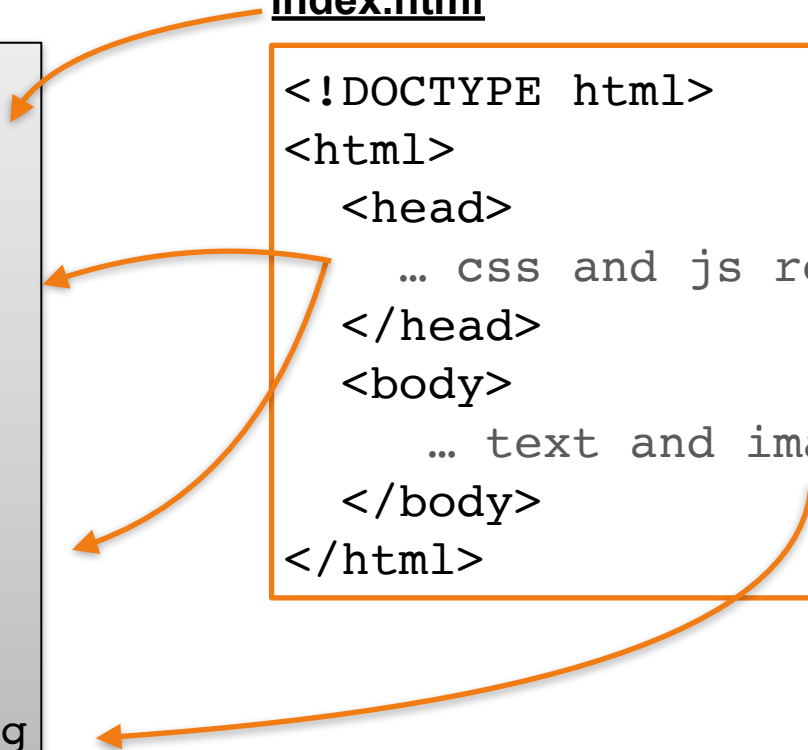
- At the simplest level our “sites” will be a collection of html, css and js files

A server

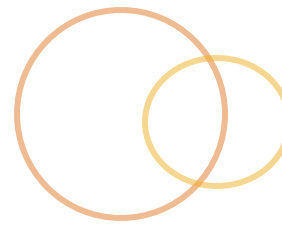
```
/var/www/  
index.html  
page2.html  
  
css/  
  site.css  
  
js/  
  site.js  
  
images/  
  banner.jpg
```

index.html

```
<!DOCTYPE html>  
<html>  
  <head>  
    ... css and js resources ...  
  </head>  
  <body>  
    ... text and images ...  
  </body>  
</html>
```



It begins with content



- ⦿ I have content that I want to make available to...
 - ⦿ My team
 - ⦿ My company
 - ⦿ The world
 - ⦿ Other machines
 - ⦿ ~~My grandparents~~
- ⦿ *I'll write up a basic piece of content*
 - ⦿ *A bio*
 - ⦿ *A page of my favorite things*
 - ⦿ *A blog*

Shortcut: <https://github.com/rm-training/html-spike/blob/master/about-me.txt>

Lab: Write some content



- In a text editor...
 - Write a ***plaintext*** bio about yourself, it should include things like
 - A title (your name)
 - A sub title (your life summed up in one line)
 - Descriptive text (a paragraph or two? fake it)
 - A list of your favorite movies with a title introducing it
 - The date you wrote this piece
 - A copyright line
 - Save it
 - `about-me.txt`
 - ***Don't write any HTML or CSS***

Reviewing our piece of content...



- ⦿ These things have semantic meaning
 - ⦿ A **title**
 - ⦿ A **sub title**
 - ⦿ Descriptive **text**
 - ⦿ A **list** of your favorite movies with the **title** introducing it
 - ⦿ These could **reference** those movies,
ie: *Point to a synopsis*
 - ⦿ The **date** you wrote this piece
 - ⦿ A copyright **line**

Reviewing our piece of content...



🕒 These things have semantic meaning ***and structure***

1. A **title**

1. A **sub title**

2. Descriptive **text**

3. A **title** for the list

1. A **list** of **references** to your favorite movies

4. The **date** you wrote this piece

2. A copyright **line**

HTML adds semantics



- 🕒 HTML is responsible for providing **structure** and **semantic meaning** to content
- 🕒 *Let's convert my plaintext profile to html*
 - 🕒 codepen.io
- 🕒 And inspect it with the dev console

Lab: Make your content HTML



🕒 In codepen.io — copy and convert your *plaintext* bio to HTML

🕒 A title

🕒 `<h1></h1>`

🕒 A sub title

🕒 `<h2></h2>`

🕒 Descriptive text

🕒 `<p></p>` per paragraph

🕒 Add a `` tag to indicate importance of a word or phrase

🕒 A list of your favorite movies =>

🕒 `Movie1Movie2`

🕒 And make each Movie Title a link (swap # with any url)

🕒 `Movie Title 1`

🕒 The list title

🕒 `<h3></h3>`

🕒 The date you wrote this piece

🕒 `<p></p>`

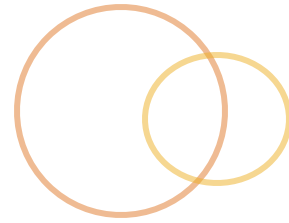
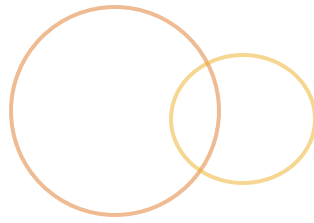
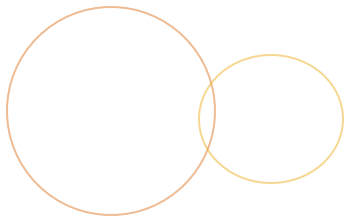
🕒 A copyright line (html entity or utf8)

🕒 `<p>© or ©</p>`

Is `<p>` really appropriate?

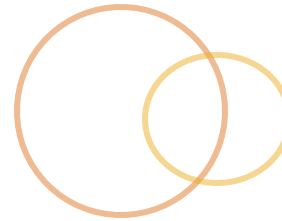
Solution

<https://github.com/rm-training/html-spike/blob/master/about-me-text-to-html.html>



html

HTML BASICS



- ◎ HyperText Markup Language
- ◎ Main language for info on the web
- ◎ Makes a network of hyperlinked documents
- ◎ Uses tags
 - ◎ `to structure`
 - ◎ and `<mark>mark</mark>` content
- ◎ Spec is maintained by the W3C.org
- ◎ HTML5 is the latest
- ◎ Plain text - edit in anything
- ◎ Typically lowercase
- ◎ Whitespace doesn't matter; indent for readability

History of HTML



- Created by Sir Tim Berners-Lee in 1991
 - The idea of tags is from SGML (standard generalized markup language) format
- HTML 2.0 in 1995
- HTML 4.01 in 1999
- XHTML 1.0 (extensible html)
 - stricter standards
- XHTML 1.1 by W3C
 - moving to XML
- XHTML 2.0
 - no backwards compatibility
- HTML5** (versus HTML 5)
 - paving the cowpaths
 - a living standard

Browser Wars



- ◎ Browsers tended to implement their own features
 - ◎ Pick and choose from specs
 - ◎ For a while they were diverging...
 - ◎ IE (once great) became the bane of web dev existence
- ◎ Now things are pretty stable and up-to-spec
- ◎ Still need to be aware of cross-browser issues

HTML5 is as strict as you make it



- ⦿ No real error reporting
- ⦿ Browser will fill in the gaps
- ⦿ Use a validator!
 - ⦿ <https://validator.w3.org/>
 - ⦿ *try it out...*

HTML Tags



- HTML is made of up **tags**

- `<h1>Hi</h1>`

- `<input type="text">`

- Tags represent **elements**

- h1, p, div, article

- Tags are supposed to be “semantic”

- h1 is a heading

- p is a paragraph

- div is...?

- article is...?

Lots of tags



- ⦿ `<h1>Heading 1</h1> <!-- through h6 -->`
- ⦿ `<p>Paragraph</p>`
- ⦿ `<a>A link`
- ⦿ `<div>A div</div>`
- ⦿ `A span`
- ⦿ `
 <!-- a line break -->`
- ⦿ `<title>The page title</title>`
- ⦿ `<input>`
- ⦿

Attributes



- Modify the meaning, data or behavior of a tag

```
<input type="text" name="username">
```

- Can be anything you want

```
<div data-junk="Hi">Bla bla...</div>
```

- Boolean tags are true by their presence, ex: required

- A few common attributes:

- id

- class

- type

- href

- We'll reference these in our css, too*

Anatomy of an element



⦿ `<element attrName="attrValue">`
 Content of element
`</element>`

⦿ Block vs inline

⦿ `<p></p>`

⦿ ``

⦿ Self closing elements

⦿ `<input type="text" name="username">`

Block Elements

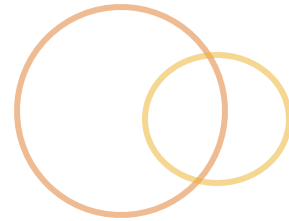


- Take up the full width of the page
- Content-level semantics
 - Organizational and structural
- Such as:
 - `div`
 - `p`
 - `h1`, `h2`, `h3`, `h4`, `h5`, `h6`
 - `ul`, `ol`
 - `section`, `article`, `header`, `aside`, `footer`
- <https://codepen.io/mrmorris/pen/EoYVWz?editors=1100#0>

Inline Elements



- Do not take up the full width of the page
- Text-level semantics
- Such as:
 - span
 - em — emphasis
 - strong — importance
 - mark — relevance
 - b, i — no longer convey meaning, only style
 - time



Ability to reference and display a visual resource

```

```

Formats are up to the browser

- jpg, png, gif, bmp, svg...

Attributes

- alt

- src

- width

- height



⦿ Anchor element

- ⦿ creates a hyperlink to other pages, files or locations

⦿ Attributes

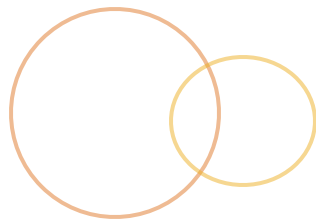
- ⦿ href
 - ⦿ relative vs absolute
- ⦿ title
- ⦿ rel
- ⦿ target

```
<a href="page.html" title="View my page">Go</a>
```

```
<!-- link within a page -->
```

```
<a href="#some-id">Jump to #some-id</a>
```


Comments



⦿<!-- this is an html comment -->



html

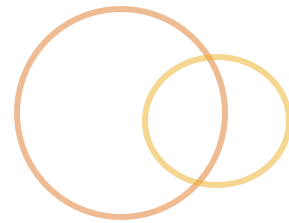
BUILDING HTML

Semantics



- ◎ The meaning of each piece of content (or block)
 - ◎ Is it a header?
 - ◎ Is it a section or an article?
 - ◎ Is this whole section of content related?
 - ◎ A paragraph in an article?
 - ◎ What is the description of this photo?
- ◎ Semantics in the web helps machines and browsers understand your content

Our Audience



- ◎ Consider who will be your audience
 - ◎ Normal Human Visitors (NHVs)
 - ◎ Robots (google) and other machines
 - ◎ Syndicators (facebook, pinterest)
 - ◎ People with disabilities (screen readers)

Organizing HTML



- ◎ HTML is what a **browser will expect to parse**
- ◎ Our pages will be built by blocks, like **legos**
- ◎ When approaching a **design**, consider:
 - ◎ How do i...
 - ◎ **break this down** into blocks?
 - ◎ communicate that **content is related**?
 - ◎ arrange my blocks nicely in **many devices**?
 - ◎ style these blocks to look **super pretty**?
 - ◎ *Don't forget...*
 - ◎ write code that's easy to **maintain**?
 - ◎ keep things **fast** for the visitor?

Content Weight



- ◎ The order of your content in the page matters
- ◎ The weight you give content matters
 - ◎ `<h1>` through `<h6>`
 - ◎ `` and ``
 - ◎ Don't overdo it
 - ◎ ie: `<h1>` on everything
- ◎ Some things don't affect weight
 - ◎ `<div>` and ``
 - ◎ `<i>` and ``

<div>, , class and id



- Two super helpful but generic tags

- <div> to organize related content

- to wrap inline text content

- And two attributes

- class

- id

- Help content like this...



```
<h1>Welcome</h1>

<h2>Post 1</h2>
<p>Bla bla...</p>

<h2>Post 2</h1>
<p>Bla bla...</p>

<h3>End of page</h3>
<p>Copyright</p>
```

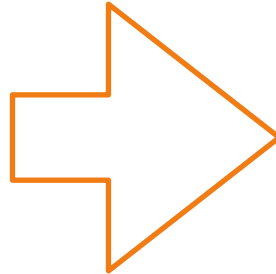
Become...

```
<h1>Welcome</h1>

<h2>Post 1</h2>
<p>Bla bla...</p>

<h2>Post 2</h1>
<p>Bla bla...</p>

<h3>End of page</h3>
<p>Copyright</p>
```



```
<div id="header">
  <h1>Welcome</h1>
</div>

<div id="posts">
  <div class="post">
    <h2>Post 1</h2>
    <p>Bla bla...</p>
  </div>

  <div class="post">
    <h2>Post 2</h2>
    <p>Bla bla...</p>
  </div>
</div>

<div id="footer">
  <h3>End of page</h3>
  <p>Copyright</p>
</div>
```

But is this semantic?

Lab: Organize your HTML w/ Divs

🕒 Add some organizational divs to your bio HTML

- 🕒 `<div id="container"></div>`
- 🕒 `<div id="header"></div>`
- 🕒 `<div id="content"></div>`
- 🕒 `<div id="about-me" class="section"></div>`
- 🕒 `<div id="movies" class="section"></div>`
- 🕒 `<div id="footer"></div>`

🕒 Add a menu div just above #content

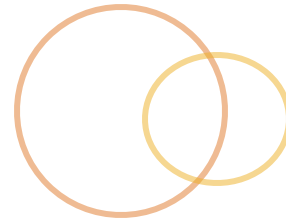
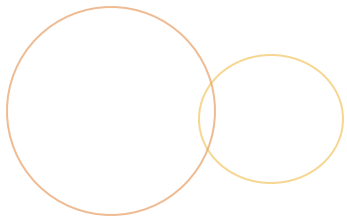
- 🕒 `<div id="menu">`
 - `Link 1`
 - `Link 2`
- `</div>`

🕒 What scenarios would deserve a `` or the use of the `class` attribute?

- 🕒 `` maybe?
- 🕒 `class` for sections or headers?

Solution

<https://github.com/rm-training/html-spike/blob/master/about-me-organize-with-divs.html>



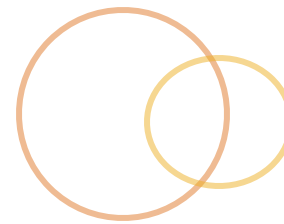
being semantic

HTML5



- Introduced more semantic elements
 - `<article>`, `<section>`, `<header>`, `<nav>`
- ...and a large set of Browser APIs
 - media, geolocation, history, drag and drop, etc...
- A living standard
- Should I use it?
 - Widely supported at this point
 - Browser auto-update FTW!

Before HTML5





🕒 Identify structural content with class and id

```
<div id="header">  
<div id="container">  
<div class="section">  
<div class="article">  
<div id="footer">  
<div id="sidebar">  
<div id="nav">
```

- 
- `<header>`

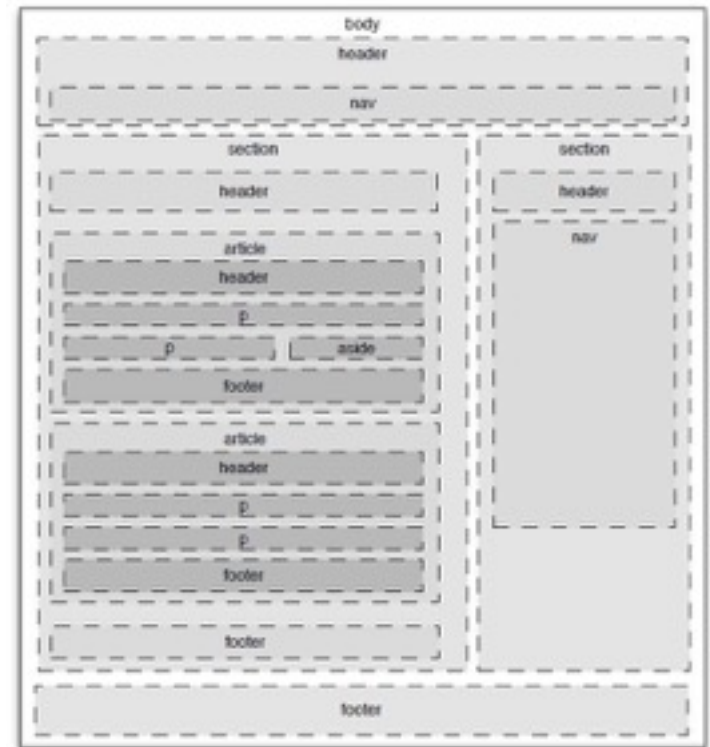
 **<footer>** **<nav>** **<main>** **<section>** **<article>** **<aside>**

 <figure>, <figcaption>

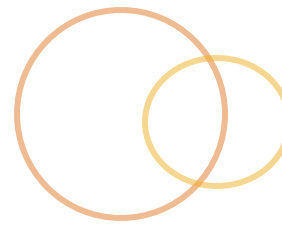
 `<time>` **<mark>** `<details>`, `<summary>`

 Example:

<https://codepen.io/mrmorris/pen/rYXJdv?editors=1100>



Essential Semantics



⦿ <header>

- ⦿ Page or section header
- ⦿ Sections can have their own headers, each with their own h1

⦿ <section>

- ⦿ Grouping thematically related content
- ⦿ <div> -> section

⦿ <article>

- ⦿ Specialized section - can this be syndicated?

⦿ <footer>

- ⦿ Information about the element that contains it

⦿ <aside>

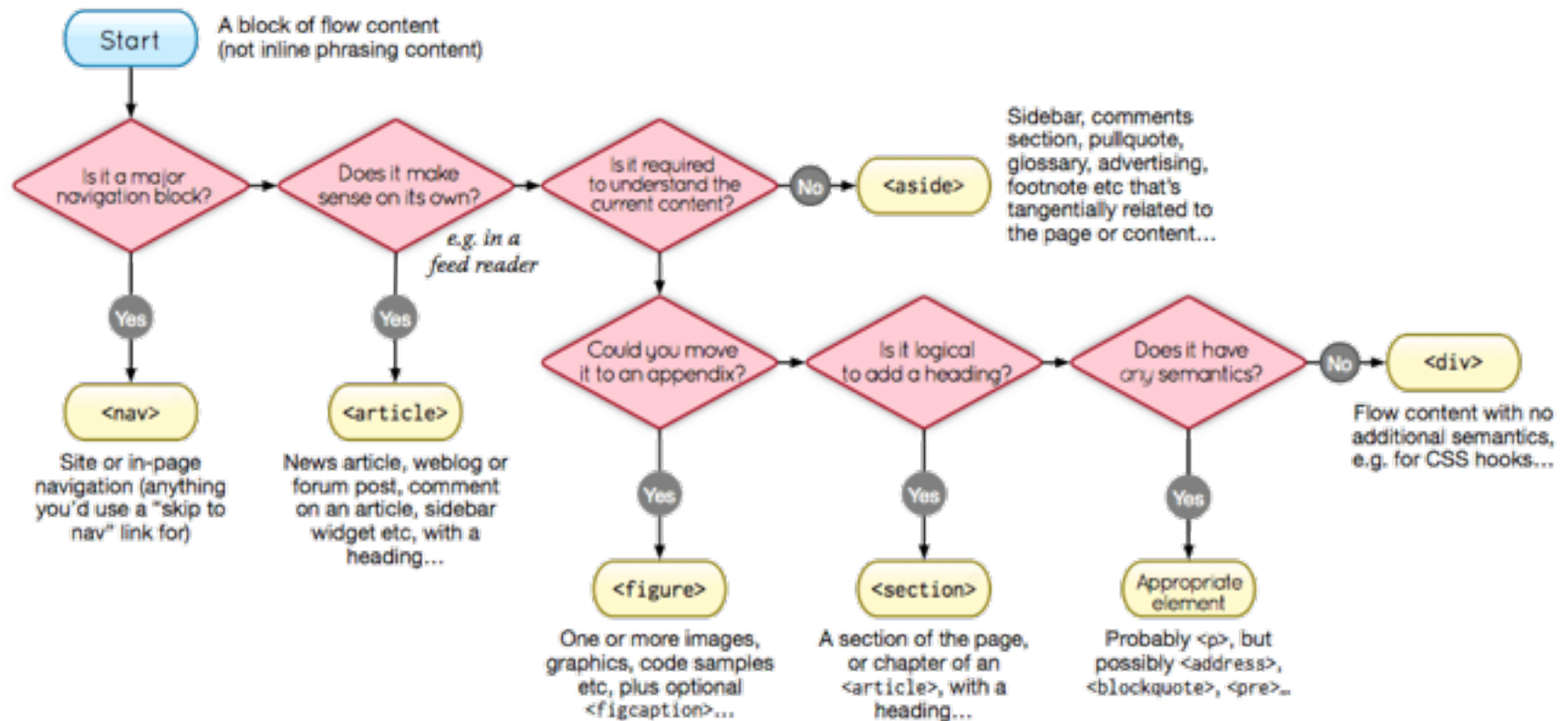
- ⦿ Sidebar or content that is related but doesn't belong into the main flow

⦿ <nav>

- ⦿ Major navigation information

⦿ All have their own <header>, <footer>, <main> and <h1> — <h16>

Which element is right?



...now with HTML5



- 🕒 We'll still want to use class and ids to help identify special content

note: div still has a place at the HTML5 table!

```
<div id="header">
<div id="container">
<div class="section">
<div class="article">
<div id="footer">
<div id="sidebar">
<div id="nav">
```

```
<header id="main">
<div id="container">
<section class="container">
<article class="cats">
<footer>
<aside>
<nav id="main">
```


Lab: Make your content semantic

- Make your **HTML** bio use HTML5 semantics
- Replace the `<div>` wrappers with the appropriate semantic element
 - `<nav>`
 - `<header>`
 - `<section>` or `<article>`?
 - add headers and footers within sections where appropriate
 - `<footer>`
 - `<time datetime="yyyy-mm-dd"></time>`
- The nav links should be wrapped in a ``
- Are ids still relevant?

Solution

<https://github.com/rm-training/html-spike/blob/master/about-me-semantic.html>



CSS

CSS INTRODUCTION

Styling our page



Our page is looking pretty boring 😴

- Has structure

- And some styles from the browser

We'll use CSS to style it so that it:

- Visually flows

- Has a more appropriate layout

- Is easier to read and use

- Is pleasant to look at and use

Let's style my page

- <https://github.com/rm-training/html-spike/blob/master/css/1-basic.css>

Lab: Style your page

- Give heading 1s:
 - A font family of Times
 - 18px font size
- Headings 2 and 3s:
 - Font of Arial
 - 11px size
 - italicize it
- The paragraphs:
 - Arial font
 - 10px size
- The date (time):
 - Uppercase it
- Anything else you want!
 - background-color
 - padding and margin

```
h1 {  
    font-family: Times, serif;  
    font-size: 18px;  
}  
h2, h3 {  
    font-family: Arial, sans-serif;  
    font-size: 11px;  
    font-style: italic;  
}  
p {  
    font-family: Arial, sans-serif;  
    font-size: 10px;  
}  
time {  
    text-transform: uppercase;  
}
```

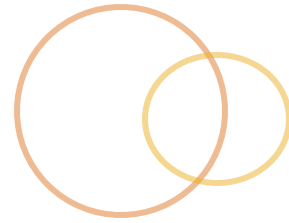
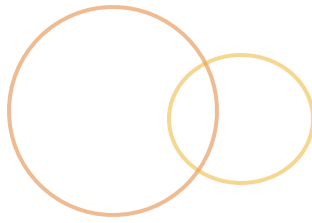
Solution

<https://github.com/rm-training/html-spike/blob/master/about-me-styled.html>



- ◎ Cascading Style Sheets
- ◎ Defines visual style of the page
- ◎ Specs are maintained by the W3C.org
- ◎ It's easy to use but hard to master
 - ◎ Not all browsers adopt at the same rate
 - ◎ Lots of old hacks are still out there
 - ◎ Devs are still figuring out how to best organize it

Why CSS?



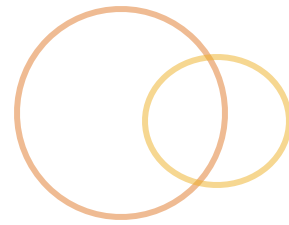
- ◎ **Separation** of presentation from content
- ◎ Pages can **share** the same formatting rules
- ◎ Formatting can be **replaced** easily
- ◎ Improved **accessibility**
- ◎ Ease of **maintenance**

History of CSS



- Developed around the same time as HTML but didn't gain adoption by browsers until later in the game
- CSS1** completed in 1996
 - IE3 had limited support
 - By 2000 IE5 supported it best (!)
 - Lots of inconsistencies across browsers...
- CSS2** around 1998
- CSS2.1**
 - fixing bugs in css2
 - was finally "published" in 2011
- CSS3** went modular; started in 1999 - still going
- CSS4** is moving to "level 4" per module

Where CSS lives



◎ Inline

```
<span style="color:red">RED</span>
```

◎ CSS block

```
<style type="text/css">  
    span {color: red;}  
</style>
```

◎ External file (.css)

```
<!-- in the html <head> -->  
<link rel="stylesheet" href="theme.css">
```

```
/* in the theme.css file*/  
span {color: red;}
```


Anatomy of a css declaration



```
◎ /* declaration block */
  selectors {
    property: value;
    property: value;
    /* shorthand */
    property: val1 val2 val3 val4;
  }
◎ div {
  color: #f90;
  border: 1px solid #000;
  padding: 10px;
  margin: 5px 10px 3px 2px;
}
```

CSS Selectors



By element

`h1 {color:#f90;}`

`<h1></h1>`

By id

`#header {`

`<div id="header"></div>`

By class

`.main {`

`<div class="main"></div>`

By attribute

`div[name="user"] {`

`<div name="user"></div>`

By relationship to other elements

`li:nth-child(2) {`

``

`p span {`

`<p></p>`

`p > span {`

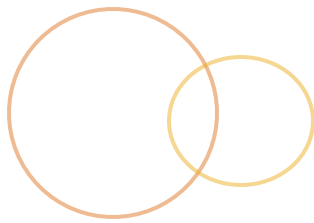
`<p></p>`

Multiple selectors

- ⦿ We can have one style declaration apply to many selectors at once

```
⦿ /*  
   * every div,  
   * every span inside a paragraph,  
   * every h1  
   */  
div,  
p span,  
h1 {  
    color: red;  
}
```

Cascading



- ◎ The mechanism by which elements inherit css properties from their parents

- ◎ Not all properties cascade

- ◎ color and font do but position and margin do not

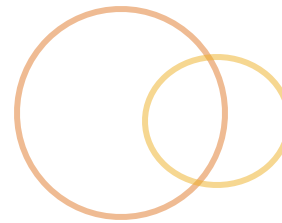
- ◎ Inherits in order of priority

- 1. Importance

- 2. Specificity

- 3. Source Order

Specificity



- Selectors apply styles based on its **specificity**
 - Order of priority: inline, id, pseudo-classes, attributes, class, type, universal
 - Higher specificity wins

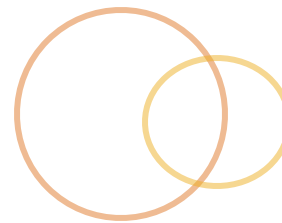
html:

```
<div id="main" class="fancy">  
  What color will I be?  
</div>
```

css:

```
#main{  
  color: orange;  
}  
.fancy{  
  color: blue;  
}  
#main.fancy{  
  color: red;  
}
```

Specificity in-depth



- Built on four values — highest result wins
 - Thousands: inline
 - Hundreds: one for each id used in the selector
 - Tens: one for each class, attribute or pseudo-class
 - Ones: one for each element or pseudo-element

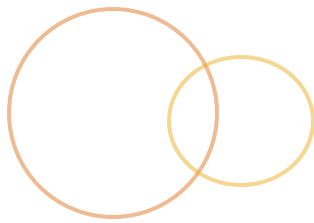
```
/* specificity: 0101 */
#outer a {
    background-color: red;
}
```

```
/* specificity: 0201 */
#outer #inner a {
    background-color: blue;
}
```

```
/* specificity: 0104 */
#outer div ul li a {
    color: yellow;
}
```

```
/* specificity: 0113 */
#outer div ul .nav a {
    color: white;
}
```

Importance



!important is a special declaration that overrides specificity

html:

```
<div id="main" class="fancy">  
    What color will I be?  
</div>
```

css:

```
#main{  
    color: orange !important;  
}  
.fancy{  
    color: blue;  
}  
#main.fancy{  
    color: red;  
}
```

Source Order



- When multiple selectors apply to the same element at the same specificity, the latest rule will win

html:

```
<div id="main" class="fancy">  
  What color will I be?  
</div>
```

CSS:

```
#main {  
  color: orange;  
}
```

```
#main {  
  color: blue;  
}
```


Cascading properties (not rules)



Property values cascade

```
html:  
<div id="main" class="fancy">  
    Will I be bold and blue?  
</div>
```

```
CSS:  
#main {  
    font-weight: bold;  
    color: orange;  
}
```

```
#main {  
    color: blue;  
}
```

blue wins but the original
font-weight will not be overwritten



More CSS Selectors



Selector

Name/Description

Example

*

Star selector

*

#id

By id

#my-id

elementname

By element name

ul

.class

By class

.my-class-name

element subElement

By hierarchy

div p

element:pseudo

:hover, :checked, etc..

a:hover

element + nextEl

Adjacent sibling

div + div.special

element > child

Directly descendant

div > p

element ~ sibling

Any sibling

div ~ div

element[attribute]

Has an attribute

a[title]

element[attribute=val]

Attribute equals

input[type=checkbox]

element:nth-child(i)

Nth child

li:nth-child(5)

:first-child, :last-child

Order as child

ul li:first-child

Exercise: Explore selectors



- Take a gander, let's get familiar
 - An interactive selector map:
 - <http://www.w3schools.com/jquery/tryselect.asp>
- Browse a site with the inspector

Lab - What will these select?



- 1.p
- 2.p p
- 3.header
- 4.header h1
- 5.header > h1
- 6..header h1
- 7.body
- 8.h1, h2
- 9.article p span strong
- 10.#container
- 11.#container footer

```
<body>
  <header>
    <h1>Welcome!</h1>
  </header>

  <p>To my page.</p>

  <div id="container">
    <article>
      <header>
        <h2>Recent posts</h2>
      </header>
      <p><strong>Yo!</strong></p>
      <footer>10 min ago</footer>
    </article>
  </div>

  <footer>
    <div class="header">
      <h1>Thanks for joining!</h1>
    </div>
  </footer>
</body>
```

Lab - How can I select...



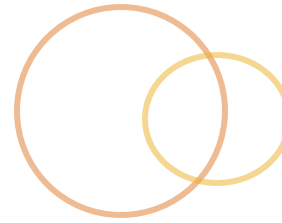
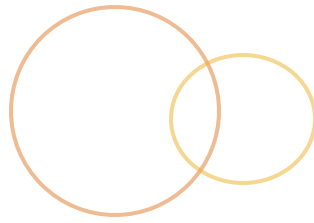
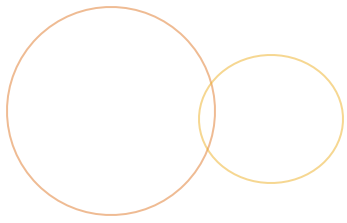
1. The container div
2. Every article
3. Article headers
4. The entire document
5. The last footer
6. The “header” div
7. All h1s
8. All paragraphs
9. Paragraphs in articles
10. Paragraphs at the top level
(not in article)

```
<body>
  <header>
    <h1>Welcome!</h1>
  </header>

  <p>To my page.</p>

  <div id="container">
    <article>
      <header>
        <h2>Recent posts</h2>
      </header>
      <p><strong>Yo!</strong></p>
      <footer>10 min ago</footer>
    </article>
  </div>

  <footer>
    <div class="header">
      <h1>Thanks for joining!</h1>
    </div>
  </footer>
</body>
```



CSS

CSS STYLING

Oh, the things we'll style



- ◎ Text!
 - ◎ Font, Color, Lists
- ◎ Blocks!
- ◎ Positioning!
- ◎ Layouts!
- ◎ State change and behavior!

Units for size and position



● Absolute Units

- **px** (`width: 10px;`)
- mm, cm, in
- pt, pc

● Relative Units

- Relative to viewport size or a font-size
- **percentages** (`width: 100%;`)
- **em**
 - Size of the current element font-size — the width of character M
 - Default is 16px in browsers
- **rem** (root em)
 - always is the base font-size of the document (not current element)
- **vw, vh** (viewport units)
 - 1/100th the width or height of the *viewport*
 - **vmin, vmax**
 - 1/100th of viewport height or width, whichever is smaller

● Computed

- `calc(25% - 4px);`
- Any expression with units



Hexadecimal

- #RRGGBB with values between 00 and FF

RGB and HSL

- `rgb(0,0,255);` /* rgb(red , green, blue) */
- `hsl(120, 100%, 10);` /* hue, saturation, lightness */

RGBA and HSLA

- Include an alpha channel
- `rgba(255, 0, 0, 0.3);`

Predefined strings

- red, blue, yellow, etc...
- https://www.w3schools.com/colors/colors_names.asp

Text styles



- color

- font-family

- font-size

- font-style

 - ex: italic

- font-weight

- text-transform

 - ex: uppercase, lowercase

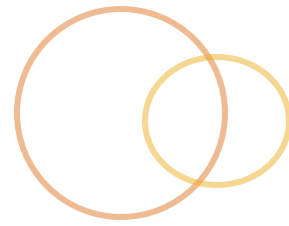
- text-decoration

 - ex: underline

- text-shadow

```
/* offset-x | offset-y | blur-radius | color */  
text-shadow: 1px 1px 2px black;
```

Text Layout



- text-align

- ex: left, right, justify, center

- line-height

- letter-spacing

- word-spacing

- text-indent

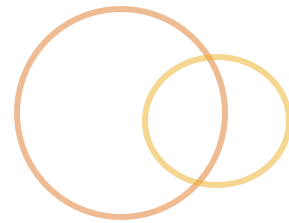
- white-space

- ex: normal, nowrap

- word-wrap

- ex: break-word

Block content



border

border: {width} {style} {color};

margin

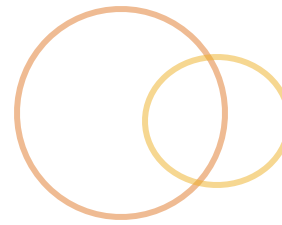
margin: {top} {right} {bottom} {left};

padding

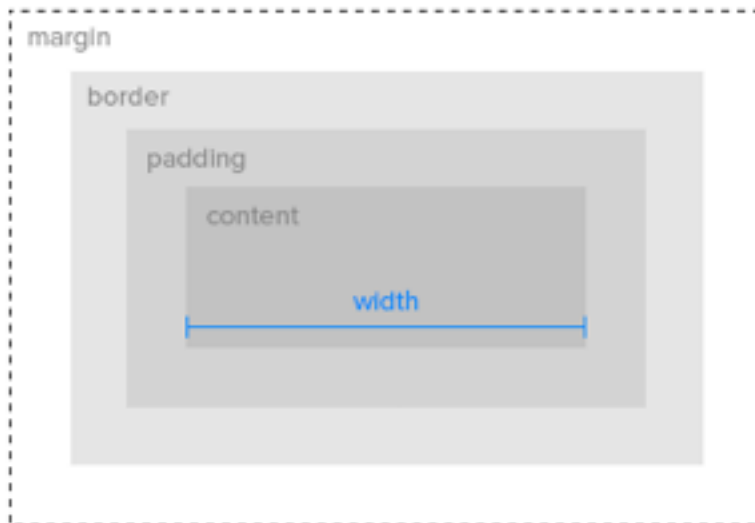
width



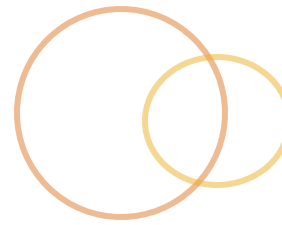
The Box Model



- How the size of a box is computed
- box-sizing
 - Property to control the box model of an element
 - content-box (default)
 - width & height of an element is applied to its “content box” only
 - Border & padding are added to the overall width & height
 - border-box
 - Include border & padding in overall width & height that you specify



Block position



- position
 - static (default)
 - relative
 - absolute
 - fixed
 - sticky (css3)
- top, left, bottom, right
 - any dimension or “auto”
- display
 - block
 - inline-block
 - none <!-- hide stuff! -->
- float
 - left, right, none
- clear
 - none, left, right, both

<https://codepen.io/mrmorris/pen/MrgOEq?editors=1100>

Pseudo-selectors



- Styling things not contained in the content

- State

 - :hover

 - :active

 - :invalid

- Or implied in the structure

 - :last-child

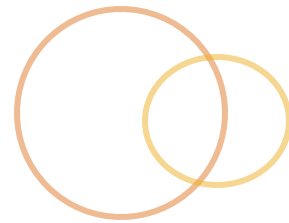
 - :nth-child

Attribute Selectors



- ◎ Select by the value of an html attribute
 - ◎ `input[type="text"]`
 - ◎ `[data-junk="bla"]`
 - ◎ `a[title]`
- ◎ Some pattern matching is possible

CSS Comments



🕒 `/* I'm a comment in css! */`

What is CSS3

○ Lots of new styling properties, such as

- border-radius

- text-shadow

- box-shadow: 1px 1px 2px #999;

- opacity

○ Animation options

- transform

- translate

- animate

○ Layouts

- CSS Grids

- Flexbox

- Columns

Browser Prefixes



🕒 Browsers experimentally support new or test features by including their own prefix

- 🕒 -moz- mozilla (firefox)
- 🕒 -webkit- safari, google
- 🕒 -o- opera
- 🕒 -ms- microsoft
- 🕒 -khtml- conqueror

```
#container {  
  display: -webkit-box;  
  display: -ms-flexbox;  
  display: -webkit-flex;  
  display: flex;  
}
```

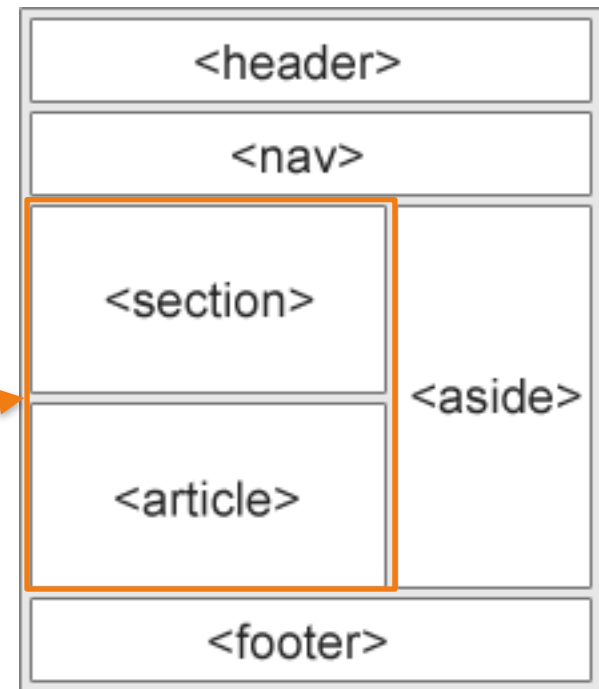
🕒 <http://shouldiprefix.com/>

Building a layout

With everything we have we can build up a decent page layout

- Header up top
- Footer on the bottom
- Sidebar to the right
- Content area to the left

`<div id="content">`



Lab: Styling a layout



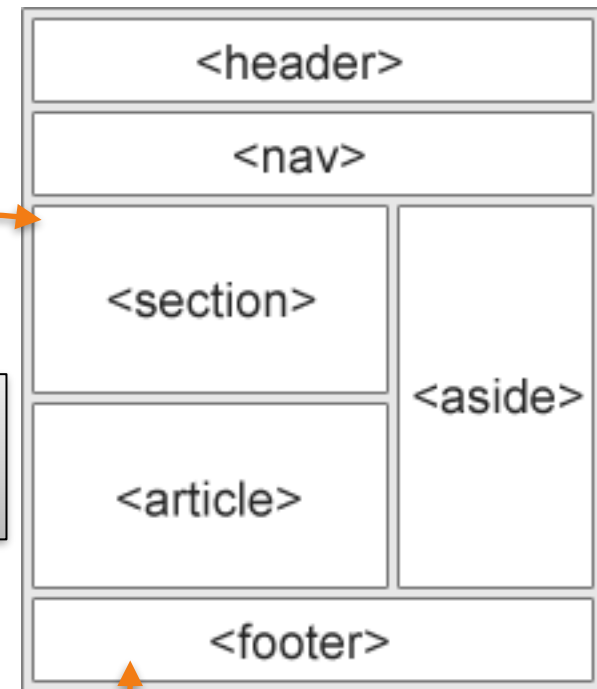
Build a basic page layout for your bio page

- Add an `<aside>` with an image and caption
- The `<aside>` should be a right column
- The `<footer>` is full width on the bottom
- Add padding, margins and borders
- You will likely want to add ids or classes to assist with selecting to style
- You'll use styles like:

- `float: right;`
- `float: left;`
- `clear: both;`
- `padding: 5px;`
- `margin: 5px;`
- `box-sizing: border-box;`
- `width: 30%;`
- `border: 1px solid #ccc;`

```
float: left;  
width: 70%;  
box-sizing: border-box;
```

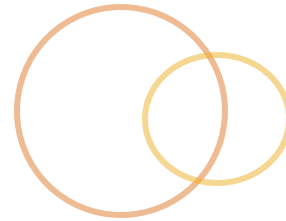
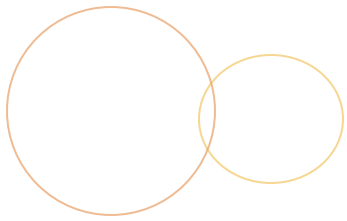
```
clear: both;
```



```
float: right;  
width: 30%;  
box-sizing: border-box;
```

Solution

<https://github.com/rm-training/html-spike/blob/master/about-me-basic-layout.html>



HTML & CSS

THE DOCUMENT

How it's fitting together



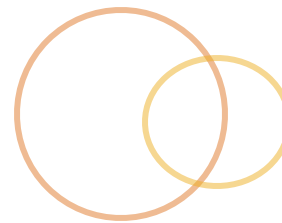
Why HTML

- Runs everywhere

How it fits

- HTML for view data & ui structure
- CSS for presentation
- JavaScript for behavior

Anatomy of a page



```
<!doctype html>
<html lang="en">
  <head>
    <title>My page!</title>
    <meta charset="utf-8">
    <link href="style.css"
      rel="stylesheet">
    <script src="script.js"></script>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

<http://jsbin.com/?html,output>

The HTML Document



- ⦿ `<!doctype html>`
 - ⦿ doctype declaration
- ⦿ `<html>`
- ⦿ `<head>`
 - ⦿ `<title>`
 - ⦿ `<meta>`
 - ⦿ info about the page
 - ⦿ `<scripts>` and css `<link>`
- ⦿ `<body>`
 - ⦿ content the user sees
 - ⦿ maybe js at the bottom
 - ⦿ and non-critical css

Publishing



Write your html files

- index.html
- bio.html

Reference css, javascript and any other resources (ie: images)

- styles.css
- scripts.js

Upload it all to a server

To consider:

- Relative vs absolute links
- Blocking vs non-blocking scripts
- Page performance
- What files does a user download for each page?

Testing Locally



- Just visit it in your filesystem
 - Dynamic languages won't work
 - Async requests won't work
 - Absolute references won't work as you expect
- or.. run a web server
 - Just like the real thing
 - Scripted (python, node)
 - Built-in
 - Apache, Nginx
 - or other super simple server apps

Working with the dev tools

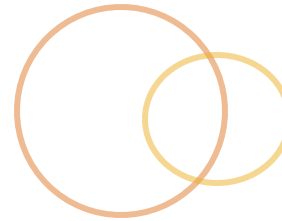
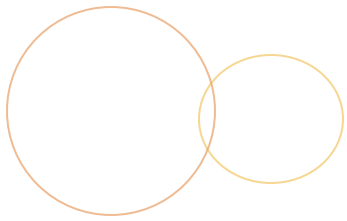


- ⦿ Inspect and edit HTML
- ⦿ Inspect and edit CSS
- ⦿ View Network happenings, Ajax, etc
- ⦿ Simulate mobile devices
- ⦿ View memory usage/debug issues
- ⦿ Inspect and debug JavaScript
- ⦿ Manage/Delete cookies, sessions, data

Lab: End simulation

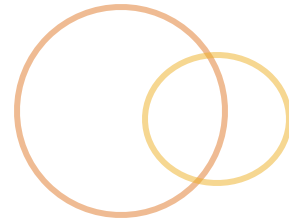
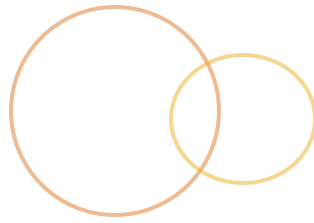
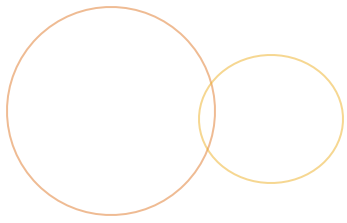


- ⦿ Break our “bio” page out of codepen.io
- ⦿ Anywhere on your local system
 - ⦿ Create an `index.html` file
 - ⦿ Write a full HTML document structure
 - ⦿ Insert your bio html into the `<body>`
 - ⦿ Create a `styles.css` file
 - ⦿ Insert your styles into this file
 - ⦿ `<link>` to the css file from `index.html`
 - ⦿ Visit `index.html` in your browser (double click)
- ⦿ Add one more file (page-to-be) - `contact.html`
 - ⦿ It will have the same document structure but a different `<body>`
 - ⦿ Update your `<nav>` to include two links
 - ⦿ One to “Contact Me!” - `contact.html`
 - ⦿ `Contact me!`
 - ⦿ Another to “Home” - `index.html`
 - ⦿ Should these be absolute or relative?



break

LOTS OF MODULES...



module

BUTTONS & HOVERS



- Generates a browser-styled button
- Used in forms or outside of forms
 - Easier to style than `<input type="submit">`
 - Can include inner HTML elements
- But lacks an href

```
<button name="button">Click <em>me</em></button>
```

```
<input type="button" value="Click Me">
```

```
<input type="submit" value="Click Me">
```

🕒 <https://codepen.io/mrmorris/pen/gXVwMd?editors=1100#0>

Anchor styling



- ⦿ A few pseudo-selectors allow us to style links based on user interaction

- ⦿ :hover

- ⦿ When the mouse hovers over the element

- ⦿ :focus

- ⦿ When the element receives “focus”

- ⦿ :active

- ⦿ When the link is clicked (“activated”)

- ⦿ :visited and :link

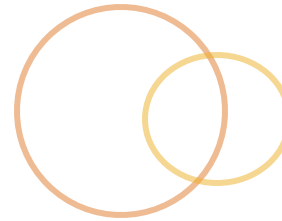
- ⦿ Anchors that have been visited or not

- ⦿ <https://codepen.io/mrmorris/pen/mqNrBz?editors=1100#0>

Lab - Mighty Fine Button, Sir



- In codepen.io:
- Write an anchor tag and style it so that it is a pretty button
- Be creative, use css like:
 - color
 - border, border-radius
 - background
 - padding
 - display: inline-block;
 - box-shadow or gradient
- Be sure to set styles for :hover
- Want to create a box shadow or gradient?
 - <https://cssgenerator.org/box-shadow-css-generator.html>



module

LISTS AND NAVS

Lists

Ordered or unordered lists of related items

ul, ol, li

Attributes

start

value

reversed

```
<ol>  
  <li>First Item</li>  
  <li>Second Item</li>  
  <li>Third Item</li>  
</ol>
```

1. First Item
2. Second Item
3. Third Item

Definition Lists



Terms + definitions

- Or anything with a direct relationships

- ex: Dialogues

- dl

- dt

- dd

```
<dl>
  <dt>First Term</dt>
  <dd>Info about it</dd>
  <dt>Second Term</dt>
  <dd>Info about it</dd>
  <dd>More info</dd>
</dl>
```

First Term

Info about it

Second Term

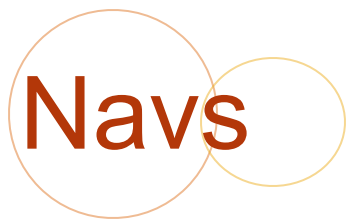
Info about it

More info

Styling Lists



- list-style: {type} {image} {position};
- list-style-type
 - disc|circle|decimal|upper-roman etc...
- list-style-position
 - inside|outside
- list-style-image
 - url(image.png);
- advanced
 - @counter style



Common pattern is to use an `` for a menu

```
<nav>
  <ul>
    <li>First Item</li>
    <li>Second Item</li>
    <li>Third Item</li>
  </ul>
</nav>
```

```
ul {
  list-style: none;
}

li {
  float: left;
}
```

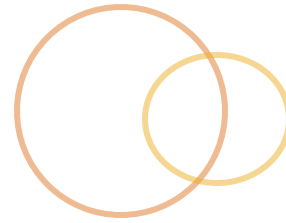
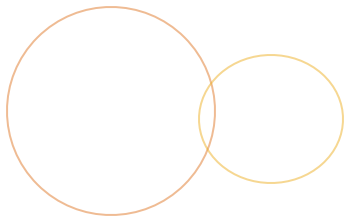
With floats: <https://codepen.io/mrmorris/pen/aEoZmx?editors=1100>

With flex: <https://codepen.io/mrmorris/pen/opvLBg?editors=1100>

Lab - May the nav be with you



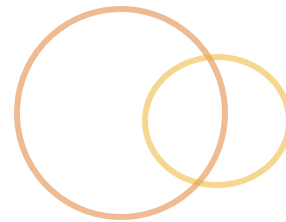
- In codepen.io:
- Write an `<nav>` menu using a styled ``
 - Include at least three links
 - Use the bio HTML we've been working from!
- Styles will include:
 - float or flex
 - padding and margin
 - text-decoration
 - background-color
 - color
 - width and height (maybe)



module

FORMS

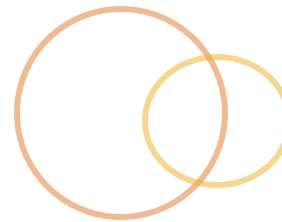
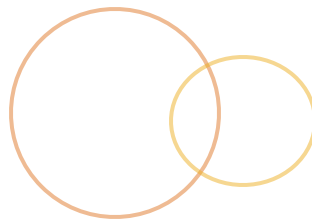
Forms



- ⦿ How we collect user input
- ⦿ On submit, makes a normal HTTP Request Methods
 - ⦿ GET, POST, PUT, DELETE
- ⦿ <form> element wraps form controls
- ⦿ Form attributes
 - ⦿ action
 - ⦿ method
- ⦿ Form can be “submitted” or “reset”

```
<form action="/edit-post" method="post">  
  <!-- form inputs -->  
</form>
```

Inputs



- How we collect data
- Lots of controls available:
 - `<input>`
 - `<textarea>`
 - `<select>`
- attributes:
 - type - text, password, radio, checkbox, email, file
 - name
 - value
- submitting
 - `<input type="submit" value="Submit me!">`
 - `<button type="submit">`
- resetting
 - `<input type="reset" value="Cancel!">`
 - `<button type="reset">`
- semantics & structure:
 - `<fieldset>`
 - `<label>`
- <https://codepen.io/mrmorris/pen/WderQL?editors=1100>

HTML5 Forms



○ New input types

- number
- range
- url
- email
- tel
- color
- search
- list/datalist

○ New element: datalist

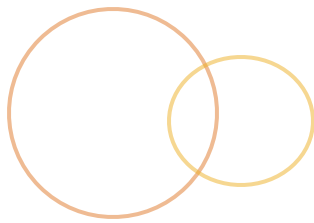
○ New input attributes

- required
- autofocus
- placeholder
- list

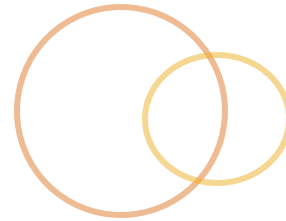
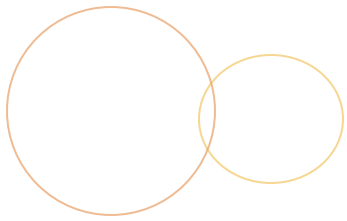
○ Browser validation

○ <https://codepen.io/mrmorris/pen/YYKwyx?editors=1100#0>

Lab: Forms



- Start a new pen in codepen.io
 - Or in your `contact.html` file
- Create a contact form
- It should collect information like:
 - name
 - email
 - personal website url
 - how they heard about you (select from a list)
 - their message (textarea)
 - confirmation that they are not a robot (checkbox)
- Use appropriate input controls, placeholders and labels
- Style it



module

TABLES

Tables



- Structured data in rows and columns
- Used to be used for layouts, too

```
<table>
  <tr>
    <th>One</th>
    <th>Two</th>
  </tr>
  <tr>
    <td>Green</td>
    <td>Yellow</td>
  </tr>
</table>
```

One	Two
Green	Yellow

Building a Table



- ⦿ wrap in `<table>`
- ⦿ Give it a header, body and footer
 - ⦿ `thead`, `tbody`, `tfoot`
- ⦿ Rows & Columns
 - ⦿ `tr` - “row of cells”
 - ⦿ `td` - “data cell”
 - ⦿ `th` - “header cell”
- ⦿ attributes like
 - ⦿ `colspan`
 - ⦿ `align`
- ⦿ <https://codepen.io/mrmorris/pen/mqZXZR?editors=1100#0>

Styling Tables



selecting rows or columns

- `tr:nth-child(n); /* nth row */`

- `td:nth-child(n); /* nth column */`

striping rows

- `tr:nth-child(odd);`

table style patterns

- `border-collapse: collapse;`

- `table-layout: fixed;`

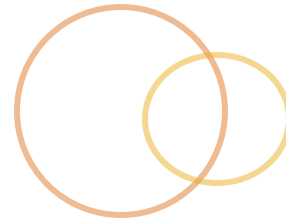
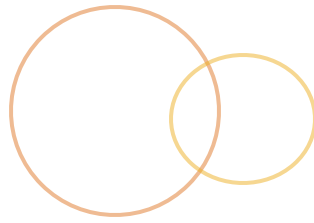
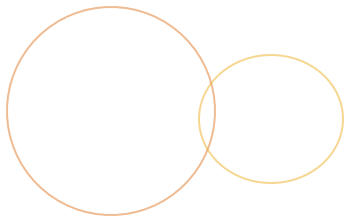
- prevent width from changing based on content

- more predictable table size

Lab: Tables



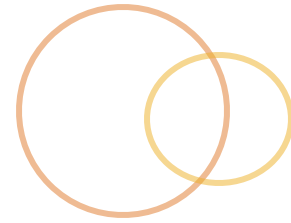
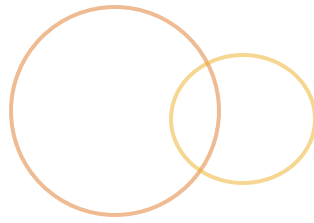
- Start a new pen in codepen.io
- Create an html `<table>` using any data
 - The first row (tr) will contain heading cells (th)
 - Add at at least 3 rows (tr) of data cells (td)
 - Wrap your heading in `<thead>` and body in `<tbody>`
- Style it so that:
 - The table width is 50% and it is centered (margin: 0 auto;)
 - The first column has a background color red
 - Every cell is padded 10 px (padding)
 - The last column's text is aligned to the right; (text-align)



module

MEDIA

Multimedia



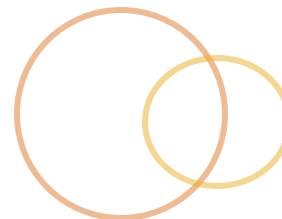
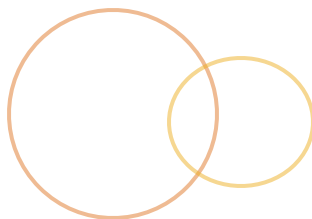
🕒 Audio

🕒 Video

🕒 Canvas

🕒 SVG

🕒 <https://codepen.io/mrmorris/pen/wPLPGm>



🕒 Embed an audio clip with a built-in browser player

🕒 Attributes:

🕒 autoplay (boolean)

🕒 controls (boolean)

🕒 loop (boolean)

🕒 muted (boolean)

🕒 volume (0.0 to 1.0)

🕒 src (string)

🕒 format

🕒 mp3 is licensed, supported in safari and i.e.

🕒 ogg is open, firefox and opera support only this

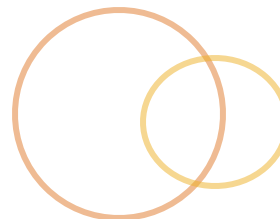
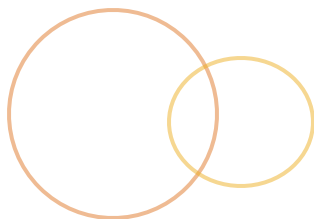
```
<audio src="bla.ogg">You don't support me</audio>
```

```
<audio controls>
```

```
  <source src="foo.ogg" type="audio/ogg"/>
```

```
  <source src="foo.mp3" type="audio/mpeg"/>
```

```
</audio>
```



- Embed a video clip with built-in browser controls

- Attributes:

 - src

 - controls

 - poster — preloaded image/screen

 - autoplay

- Formats

 - ogg - superseded by webm

 - mp4 - ie and safari — however now it's widely supported through hardware playback

 - webm - firefox and chrome

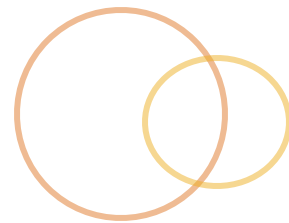
- can add subtitles (<track> element)

- can style video now -- get full page background videos

 - ex: <https://codepen.io/dudleystorey/pen/knqyK>

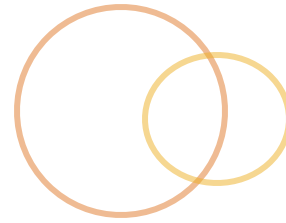
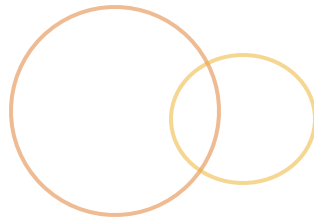
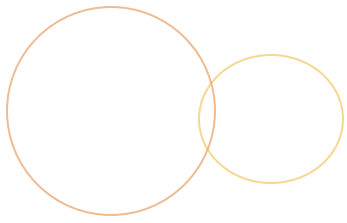
```
<video src="bla.mp4">You don't support me</video>
```

Styling Audio & Video



- ◎ Markup controls yourself
 - ◎ disable browser controls
 - ◎ use javascript to trigger the player

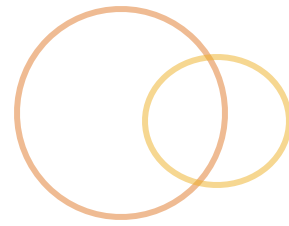
```
<audio id="player" src="vincent.mp3"></audio>
<div>
  <button onclick="document.getElementById('player').play()">Play</button>
  <button onclick="document.getElementById('player').pause()">Pause</button>
  <button onclick="document.getElementById('player').volume += 0.1">Vol+ </button>
  <button onclick="document.getElementById('player').volume -= 0.1">Vol- </button>
</div>
```



module

BEING RESPONSIVE

Viewport meta tag



- Introduced by Safari for mobile optimization
- No real standards but most browsers support it

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- device-width
 - width of the device
- initial-scale
 - Zoom level
- maximum-scale
 - Can prevent zooming

Media Queries



Conditional queries targeting media capabilities

- Screen or print
- Screen orientation, aspect ratio and resolution
- Color space
- Dimensions

Examples

- <https://codepen.io/mrmorris/pen/wpwMbV?editors=1100>
- <http://www.alsacreations.fr>

Media Queries



Based on a condition

- ② `@media screen and (min-width: 320px) {...}`
 - ② Screen devices with browsers at least 320px wide
- ② `@media (max-width: 12450px) {...}`
 - ② Affects all devices 12,450px wide or less

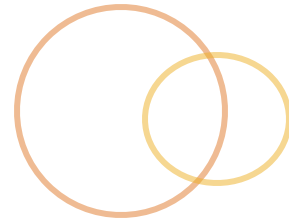
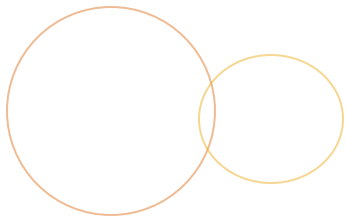
Can stack up conditions

- ② `@media (min-width: 30em) and (orientation: landscape) {...}`
 - ② If browser is at least 30em wide and device is in landscape mode
- ② `@media screen and (max-width: 699px) and (min-width: 520px) {...}`
 - ② If browser width is between 520px and 699px

Lab - Media Queries



- Update your bio page to be responsive
- When the page is less than 800px wide
 - Make the main content area full width
 - Stack the `<aside>` below the content area, full-width
 - Make the `<nav>` links stack
 - Full width
 - or... style as buttons



module

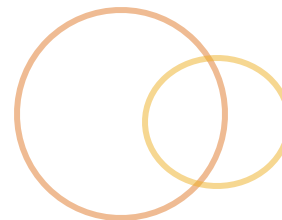
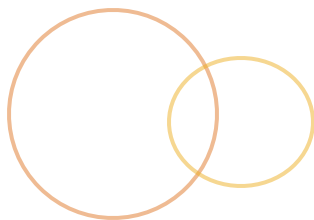
ANIMATIONS

CSS Animation



- Traditionally, animation was handled in JavaScript
 - mouseovers
 - color or size changes
 - or the :hover pseudo-selector
- Now CSS3 supports a lot of basic animations
 - Transitions
 - Transforms
 - Animate (with keyframe)
- Paving the cowpaths
- Should add to the experience but should not be necessary — degrade gracefully

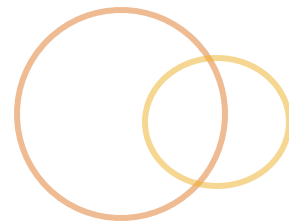
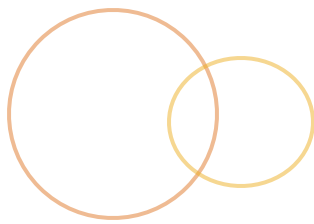
Transitions



- Specify the properties which you want to animate
 - Along with during & easing
- Then adjust the property value
 - CSS will animate the transition between values
- Properties
 - transition-property
 - “all” for all properties
 - transition-delay
 - transition-timing-function
 - ease, linear, ease-in, ease-out, ease-in-out, cubic-bezier
 - transition: {property} {timing} {easing};**

```
transition-property: background;  
transition-duration: 0.3s;  
transition-timing-function: ease;  
transition: background 0.3s ease {delay};
```

Transforms



⦿ Allow user to transform element in 2d or 3d space

- ⦿ `rotate(-10deg)`

- ⦿ `skew`

- ⦿ `translate`

- ⦿ moving position relative to it's original position

- ⦿ ex: `translate(10px -20px);`

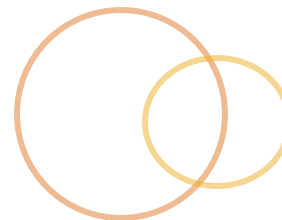
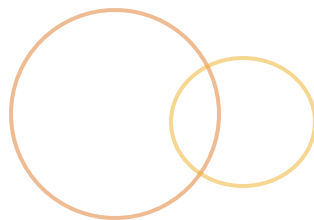
- ⦿ right 10, up 20

⦿ Then you can "transition" a transform

- ⦿ `transition: transform 1s ease-in;`

```
transform: scale(1.5); /* scale to 1.5* size */  
transform-origin: bottom left; /*bottom, center or percentage*/
```


Animation



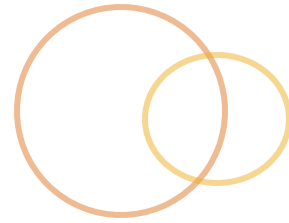
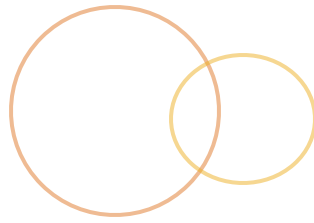
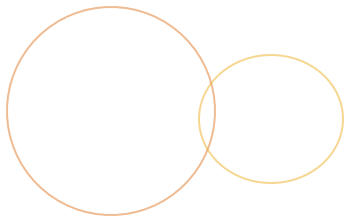
- 🕒 Set up a `@keyframe` declaration
 - 🕒 Then reference it as an `animation` for any element
- 🕒 Limited support

```
@keyframes pulse {  
  0% {color: red}  
  50% {color: blue}  
  100% {color: red}  
}  
button {  
  animation: pulse 1s infinite ease-in;  
}
```

Lab - Make it animate



- Try setting up a `transition`
- Determine an event that will trigger the change
 - `:hover`?
 - We don't yet have JavaScript in our toolkit...
- When a thing is hovered over, change a style property to have it transition
 - `background-color`?
 - `box-shadow`? ← nice to create a “lift” effect
 - `text-shadow`?
- Bonus:
 - try a `2d transform` to move the element on hover



module

CSS3 LAYOUTS

MicroLayouts



- ⦿ Used to rely heavily on position & float
- ⦿ CSS3 Introduces a few powerful layout options
 - ⦿ FlexBox
 - ⦿ Multi-Columns
 - ⦿ CSS Grid
- ⦿ Avoid “div-itus”
- ⦿ Presentational logic kept where it belongs

Columns



⦿ Automatically breaks content into columns

⦿ `column-count: {n};`

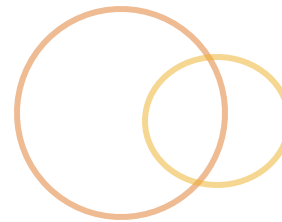
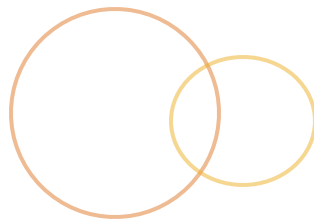
⦿ `column-gap: {n}px;`

⦿ `column-rule: 1px solid #f90;`

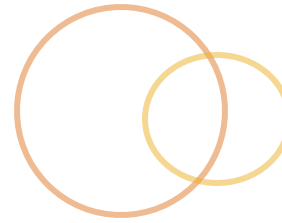
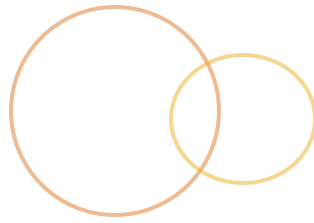
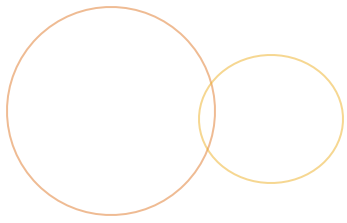
⦿ Can span across columns

⦿ `column-span: all;`

⦿ <https://codepen.io/mrmorris/pen/BJBKwN>



- Ability to specify how the child elements of a container flex to fill the space
- It's a little challenging to grasp at first
 - This is a good reference:
 - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- You specify an axis (vertical or horizontal)
 - Downside is it only runs in one dimension
- Specify how the elements align and stretch to fit
- Ability to get true vertical centering!
- Nav example:
 - <https://codepen.io/mrmorris/pen/xpKVYa?editors=1100>



module

AND BEYOND

Olden Days of Front end Dev



- Write HTML
- Write CSS
- Write JavaScript if needed
- Link them together and upload via SFTP
- Test it once in Firefox then in IE like crazy

```
/root  
/root/index.html  
/root/page2.html  
  
/root/css/site.css  
  
/root/js/site.js  
  
/root/images/...
```


Modern Front End Dev



- Write it

- Decide on

- Language enhancements
- Organization method
- Framework
- Build process
- Package management
- Deployment process
- Test framework
 - Test on many devices

CSS Performance



Browser parses selectors RIGHT to LEFT

- `.some-class ul li a {...}`

- browser finds all `a` elements...

- that are within an `li`...

- narrow that down to any `li` in a `ul`

- narrowed down to any `ul` inside `.some-class`

Descendant selector is costly because it searches the document for each

- Just use class names in full (or ids)

Don't use the universal (*) selector

CSS Preprocessors



- ◎ SASS or LESS
- ◎ Extend CSS to support more powerful features
 - ◎ Variables
 - ◎ Mixins
 - ◎ Nested rules
 - ◎ Partial
- ◎ Either JS that **runs on the fly** or a **build step**
- ◎ <http://sass-lang.com/>
- ◎ <http://lesscss.org/>

Support older browsers



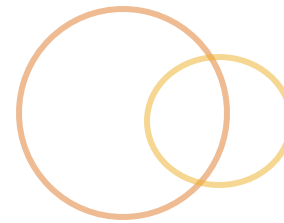
- Modernizr

- <https://modernizr.com/>

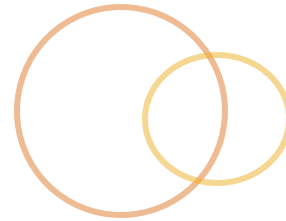
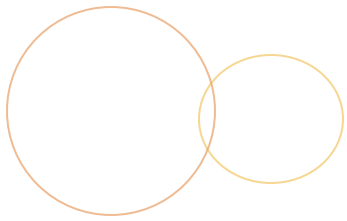
- Detect it yourself

- <http://diveintohtml5.info/everything.html>

Modern Web Dev



- Writing
 - IDE or Text Editor
 - “Auto Refresh” in my browser
 - Frameworks
 - Layouts
 - Bootstrap
 - SemanticUI
 - App
 - ReactJS
 - Vue.js
 - Package management
 - Bower, NPM
 - CSS
 - Pure CSS or a precompiler?
 - Modules
- Building
 - HTML Linting
 - CSS precompiling
- Testing
 - Desktop and Mobile
 - Lot's of browsers & versions... which are important?
- Front End Dev Checklist
 - <https://frontendchecklist.io/>



the end is hear

WRAPPING UP

Best Practices



- Write semantic HTML
- Don't write inline css
- Add progressive enhancements
 - Degrade Gracefully
- Consider your audience(s)
- Consider your devices
 - Responsive vs Targeted
- Code defensively
 - Not all browsers support the same things

Questions/Comments?

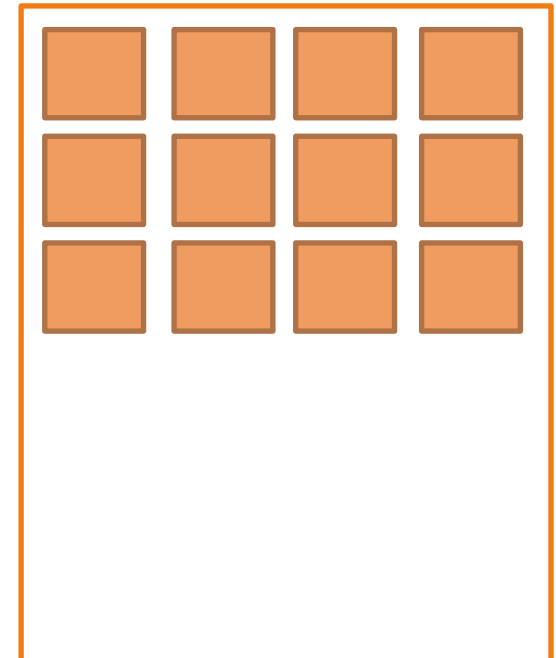


- Any topics of interest I didn't cover?
- Any specific questions?

Final Lab - Grid of Images



- ⦿ A common pattern is a grid of image “cards”
- ⦿ In a new page or codepen.io
- ⦿ Make a four column grid of image “cards” with at least 8 images
 - ⦿ `<div class=“card”></div>`
 - ⦿ <https://codepen.io/mrmorris/pen/MrgQeb?editors=1100>
- ⦿ Use a flex or float approach
- ⦿ Each “card” should occupy ~25% width
 - ⦿ 10px Gutters? You can use a calculated value
 - ⦿ `margin: 0 calc(25% - 10px);`
 - ⦿ Consider padding and margin
 - ⦿ Consider any style in the first or last “column”
- ⦿ Make each “card” a link (`<a>`)
 - ⦿ Add a `:hover transition`
- ⦿ Bonus: add varying captions
 - ⦿ how does it stack?

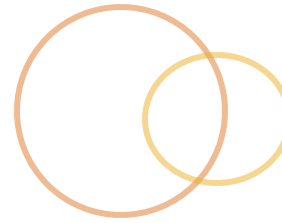
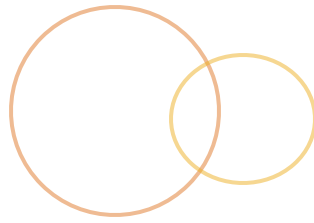
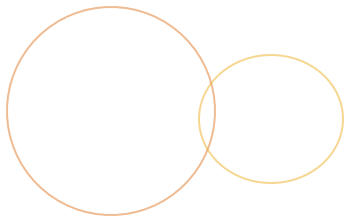


Go now and code well

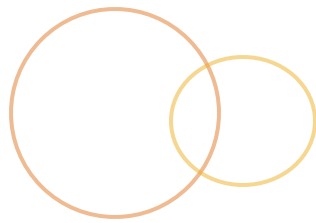
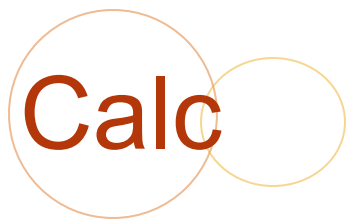


☉ That's a wrap!

- ☉ What did you enjoy learning about the most?
- ☉ What is your key takeaway?
- ☉ What do you wish we did differently?
- ☉ Any other comments, questions, suggestions?
- ☉ Feel free to contact me at mr.morris@gmail.com or my eerily silent twitter **@mrmorris**



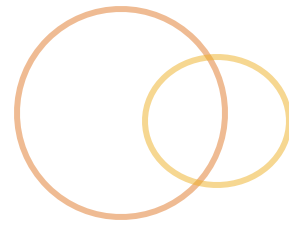
EXTRA STUFF



Adjust units dynamically

`font-size: calc(16px + 0.5vw);`

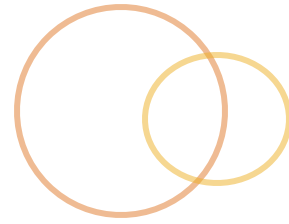
Sticky Footers



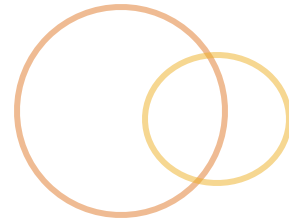
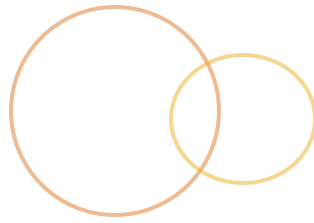
- Set your body to
 - height: 100vh
 - min-height: 100vh
- And your footer will always reach the bottom (at least)
 - Rather than have space

The pain of css

🎯 Vertical centering

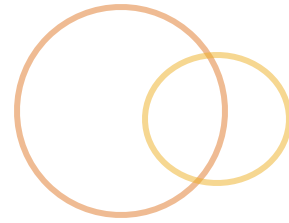
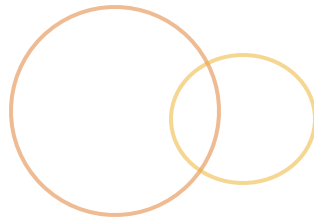


Question:



- Does the page need to look the same in every browser?
- No.

lframes

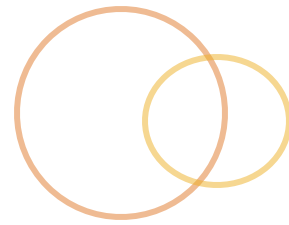


Meta Viewport



🕒 `<meta name="viewport" content="width=device-width, initial-scale=1">`

HTML5 Outlines



🕒 Outliner: <http://hoyois.github.io/html5outliner/>

HTML Entities



- Some special characters that would otherwise be part of the html structure
 - <, >, &, “
 - < > & "
 -
 - ©
- <https://developer.mozilla.org/en-US/docs/Glossary/Entity>