CSS Box Model

Lesson 3

FEWD
GENERAL ASSEMBLY

Before Each Class

- **Open** GH Desktop
- Top toolbar: Branch → Merge into Current Branch → upstream/master
- **Click** "Merge into master"
- **Click** "Pull"
- **Open** VS Code: File → Open → Select **FEWDR-422** folder → Click "Open"

Learning Objectives

- Define CSS Box Model, and demonstrate the ability to properly manipulate the "box" around tags, including margin and padding.
- Apply inline and block attributes to a page.
- Explore the different HTML structural elements
- Troubleshoot HTML/CSS using the Chrome Developer Tool

Agenda

- Review Week 1
- CSS Box Model
    - Inline vs Block Elements
    - Padding, Border, and Margin
- Chrome Developer Tools
- Div + HTML5 Structural Elements
- Lab Time

Week 1 Review

Open: **starter_code/week_01_review**

Directions:

1. Open the **starter_code/week_01_review** folder
2. Follow the prompts to fill in the blanks

Timing:

- **15 minutes** - As a class, fill in all of the blanks to create a working web page

Tips:

- Recall the different principles we learned last week

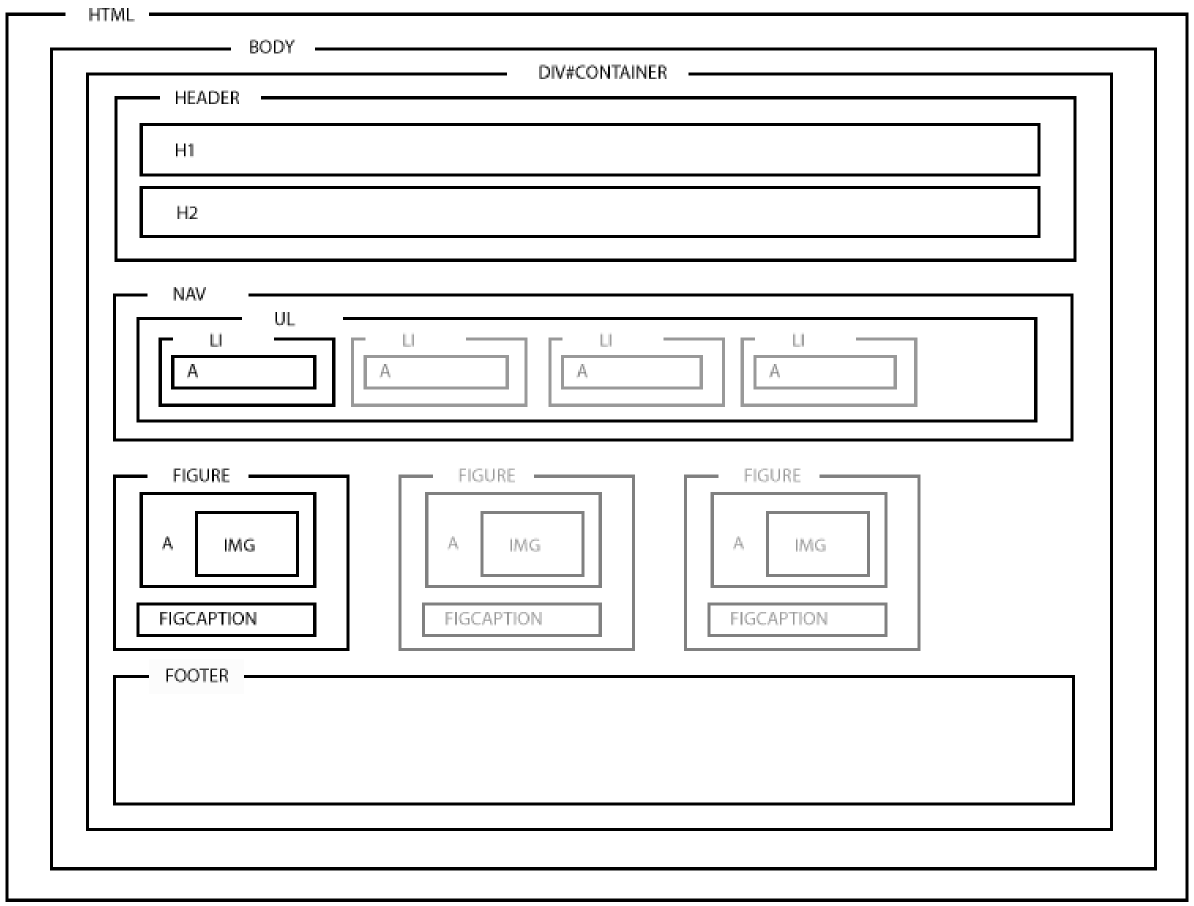
CSS Box Model

What is the CSS Box Model?

Every element in web design is a box, which can be styled using CSS.

- `<h1>`
- `<p>`
- `<img>`
- `<div>`
- `all boxes...`

CSS Box Model Website

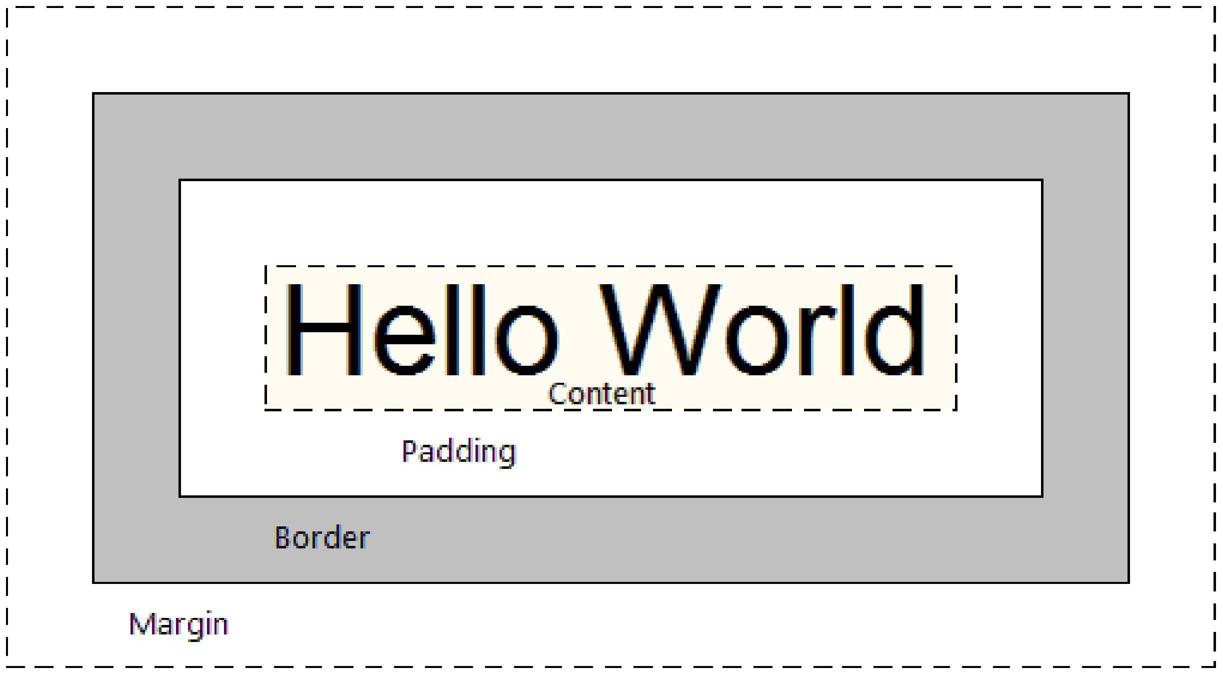


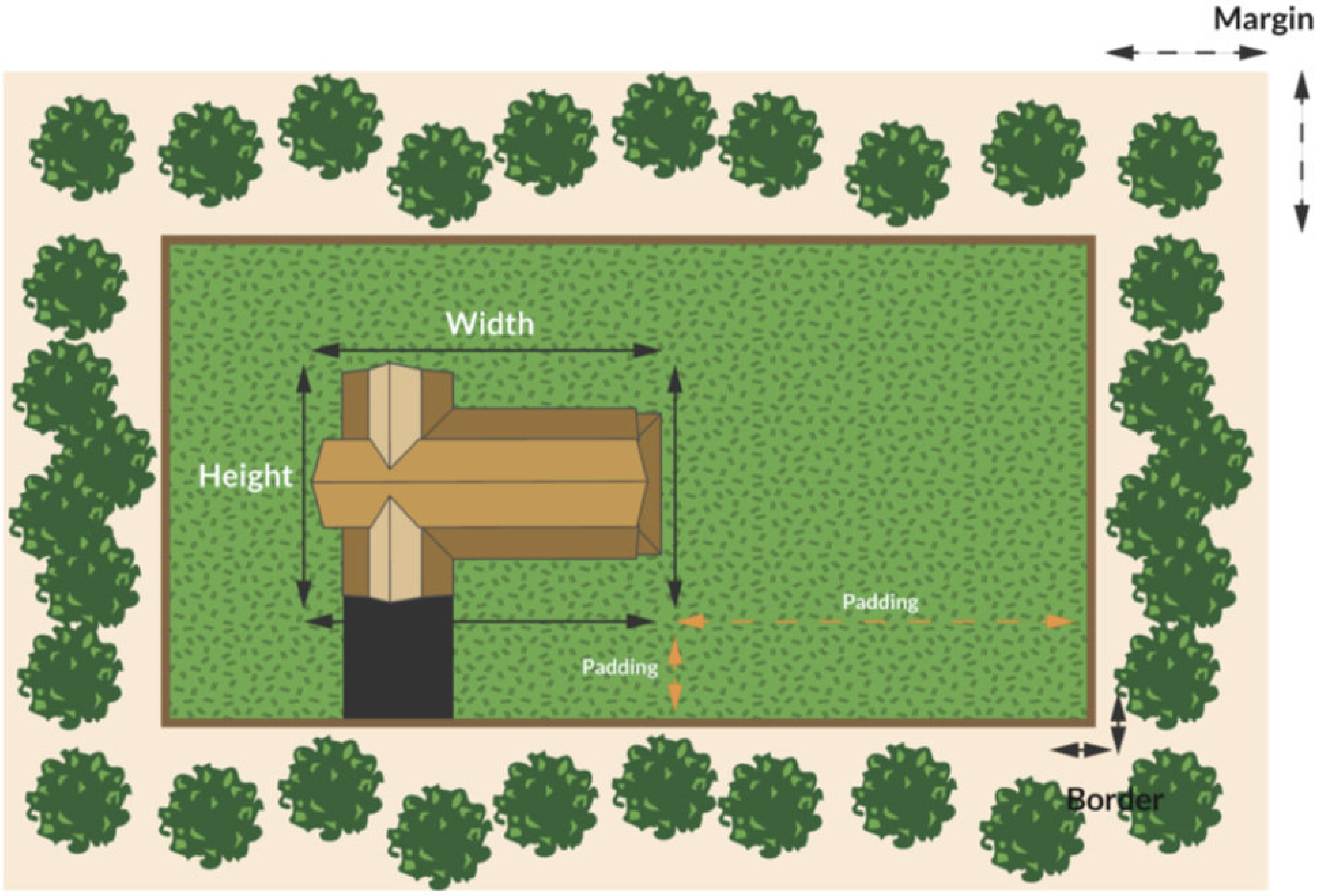
"Box" Model

From the inside out...

- content
- padding
- border
- margin

Box Model Image




Margin
Width
Height
Padding
Padding
Border

Default "Rendered" Width

Content Width

+

Padding-left

+

Padding-right

+

Border-left

+

Border-right

Default "Rendered" Height

Content Height

+

Padding-top

+

Padding-bottom

+

Border-top

+

Border-bottom

Margin

Margin is unique in that it doesn't affect the size of the box itself, per se.

Rather, it affects how this element interacts with other elements next to it, and thus is still an important part of the CSS box model.

Pulse Check

Key Objective:

Explore this interactive Box Model CodePen. Select different values for width, height, padding, margin and border. Try to predict what the rendered height/width will be.

Timing:

- **2 minutes** - Discuss with a partner in a breakout room
- **1 minute** - Volunteer to describe what you found

Bonus:

Can you figure out what the "box-sizing" option does?

Box-sizing

Box-sizing is an important CSS property that is used to alter the default CSS box model used to calculate width and height of HTML elements.

Box-sizing CSS Property

```
* {
    box-sizing: content-box;
}
```

Content-box is the default box model.

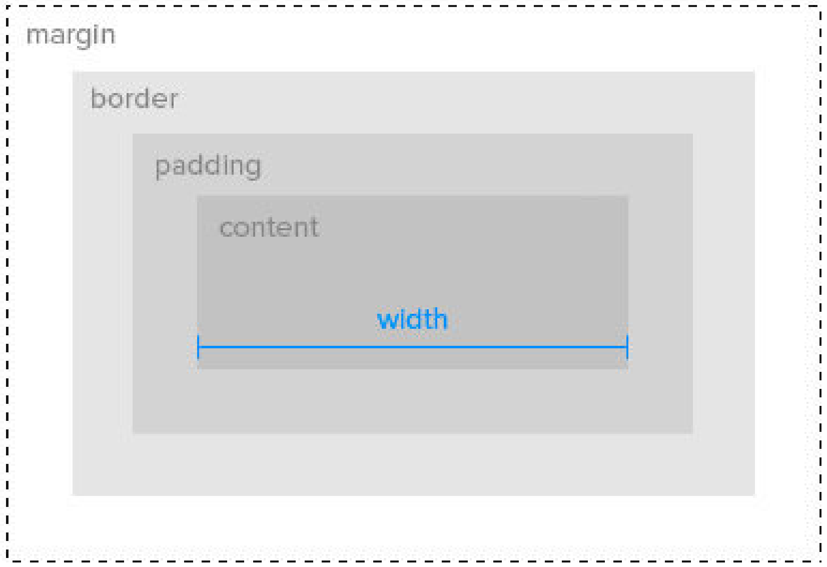
The element's width and height includes only the content. Border and padding are **not included**.

```
* {
    box-sizing: border-box;
}
```

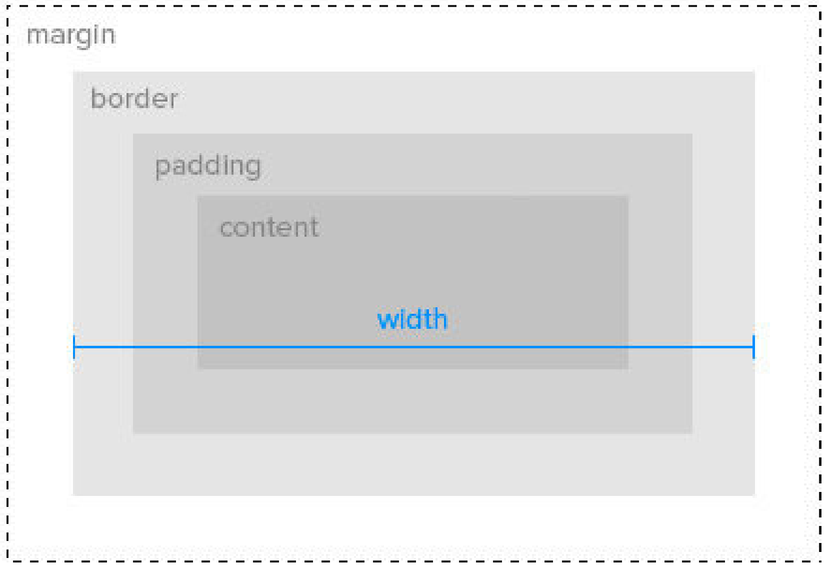
Border-box includes padding and border in the element's total width and height.

Box-sizing Example

content-box



border-box



Pulse Check

Key Objective:

Let's revisit this interactive Box Model CodePen again as a class.

Timing:

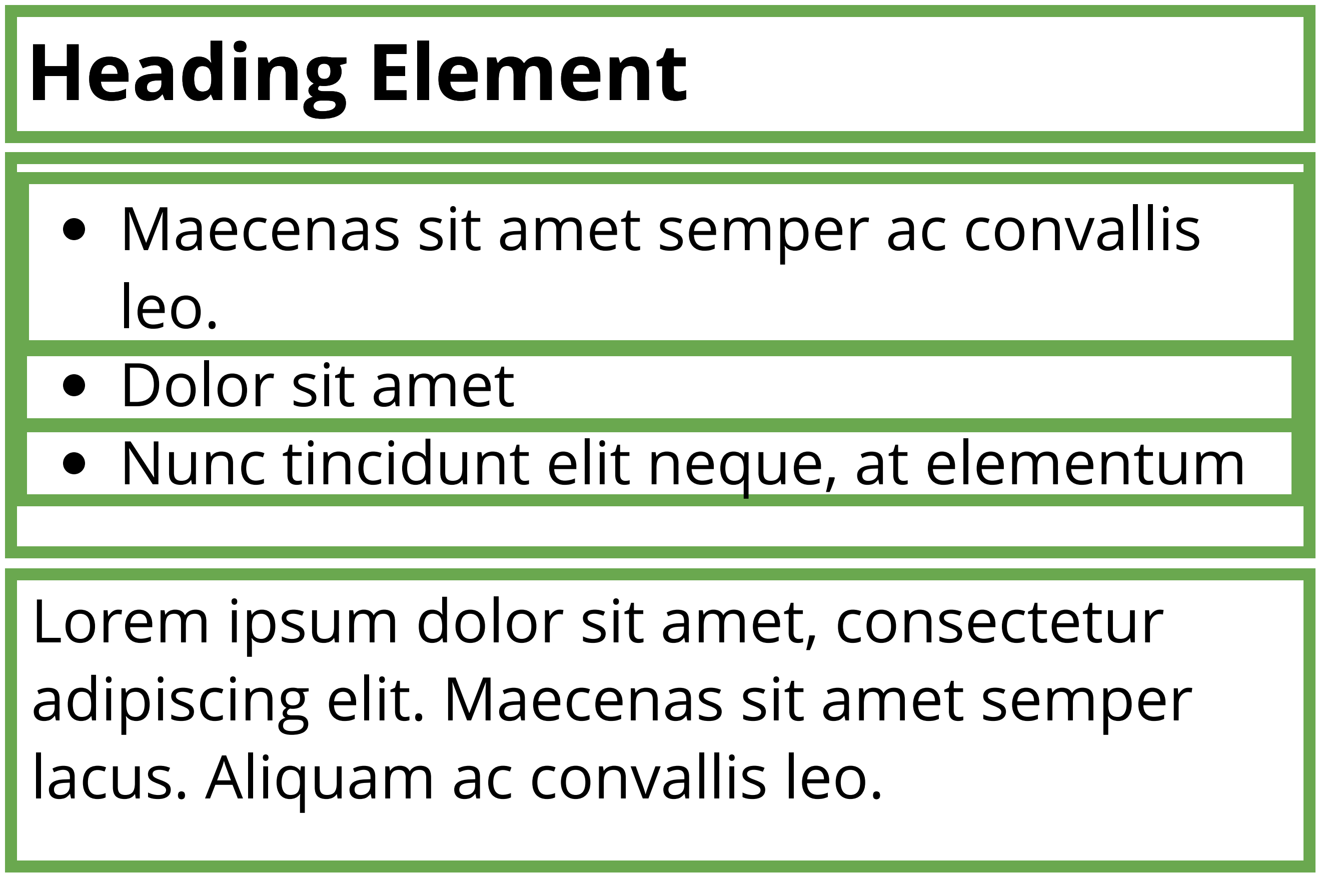
- **2 minutes** - Discuss what happens when switching between the different box-sizing properties.

Inline vs Block

Block Elements

- By default, **block-level elements** begin on new lines
- Essentially, this means the "boxes" for block-level elements extend the full-width of the available space (even if the content does not)
- They can contain inline elements, as well as other block-level elements

Block Elements (cont'd)



Heading Element

- Maecenas sit amet semper ac convallis leo.
- Dolor sit amet
- Nunc tincidunt elit neque, at elementum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet semper lacus. Aliquam ac convallis leo.

Block Element Examples

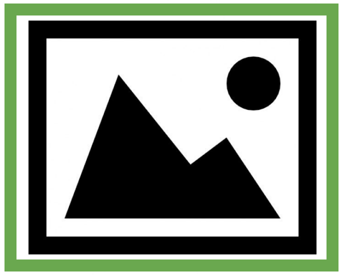
- `<p>`
- `<h1>, <h2>, ... <h6>`
- `<ol>`
- `<div>`
- `<blockquote>`
- more...

Inline Elements

- By default, **inline elements** do not begin on new lines
- Essentially, this means the "boxes" for inline elements will only take up as much space as necessary
- These should only contain data and other inline elements -- not block-level elements.

Inline Elements (cont'd)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet semper lacus. Aliquam ac convallis leo. Proin a lorem eu eros tristique vehicula. Nam purus est, pretium ac tincidunt id

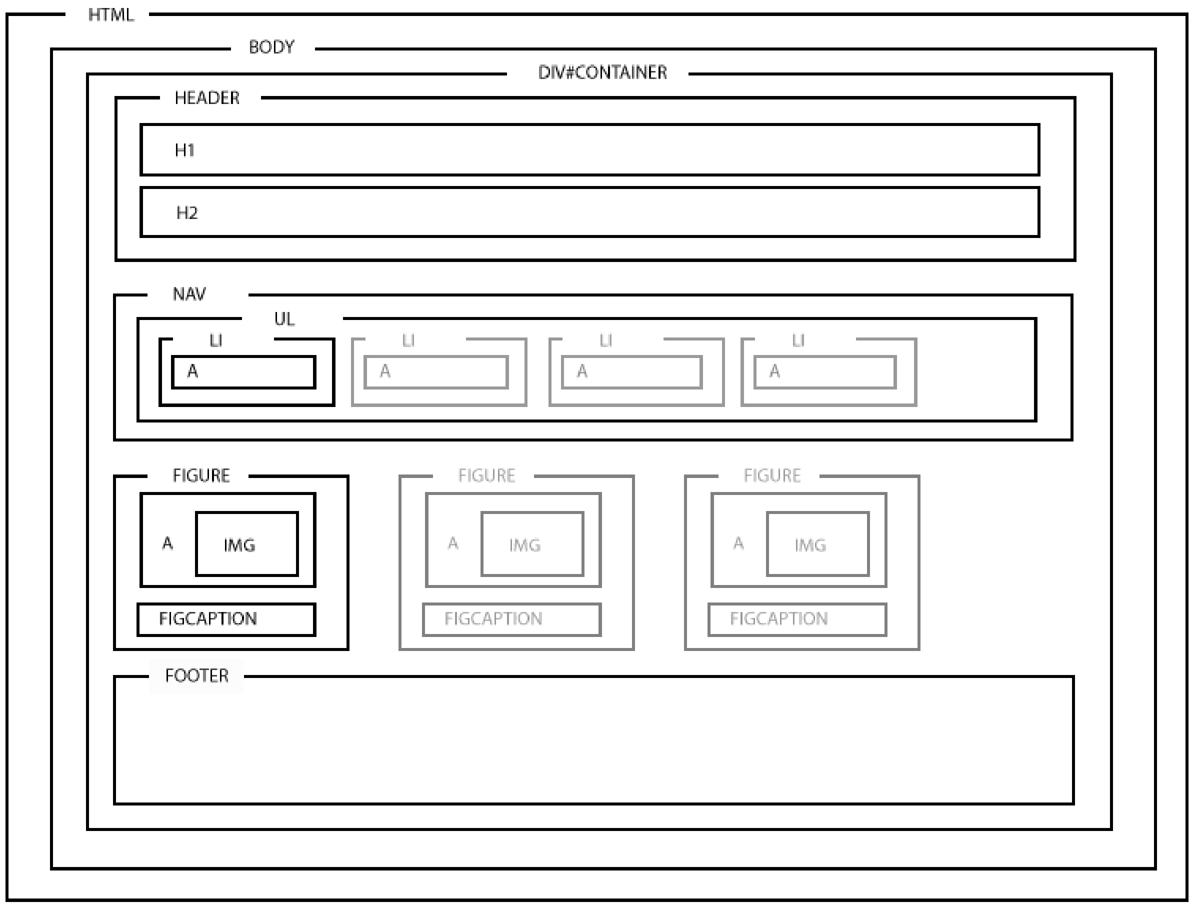


Suspendisse dolor risus, pulvinar vitae iaculis quis, sollicitudin et urna. Vestibulum sollicitudin ultrices nunc ut iaculis.

Inline Element Examples

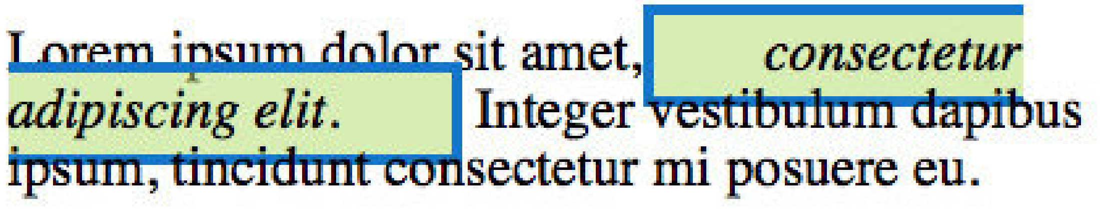
- `<span>`
- `<img>`
- `<strong>`
- `<a>`
- `<input>`
- more...

CSS Box Model Revisit



Inline Element Key Difference

When trying to add dimensions to inline elements:

- Width and height will have **no effect**
- Any vertical **padding**, **border**, or **margin** applied to an element will not push away elements above or below it.
- 

Lorem ipsum dolor sit amet, *consectetur adipiscing elit.* Integer vestibulum dapibus ipsum, tincidunt consectetur mi posuere eu.

Pulse Check

Key Objective:

Experiment with CSS Display using this CodePen

Timing:

- **4 minutes** - Discuss with a partner in a breakout room
- **1 minute** - Volunteer to describe what you found

Directions:

- Follow the steps outlined in the CSS comments.
- What do you notice about setting the height and width on inline elements (e.g. anchors) versus block elements (e.g. list items)?

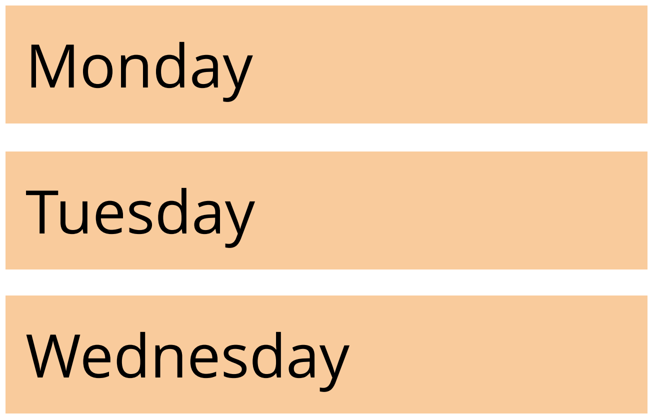
Changing the Display Property

Display: inline

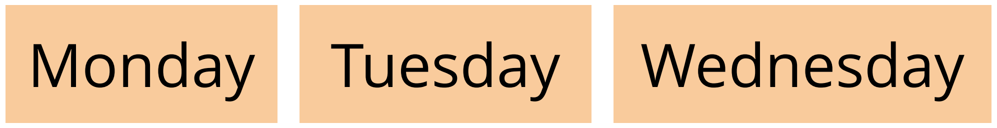
Display **inline** makes a block element act like an inline element.

```
li {
    display: inline;
}
```

BEFORE:



AFTER:



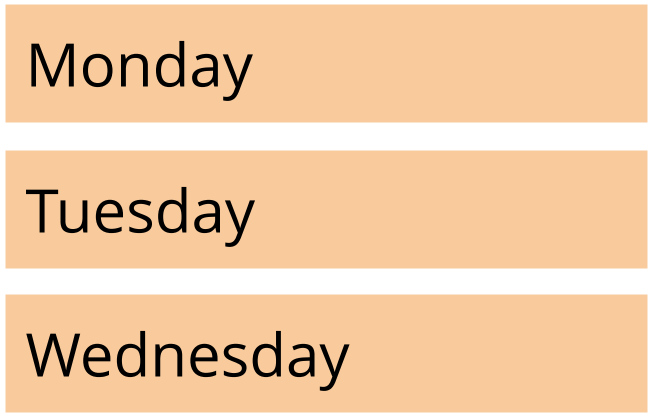
- Elements will now sit next to each other
- **Still can't set a width, height, or margin and padding on top and bottom**

Display: inline-block

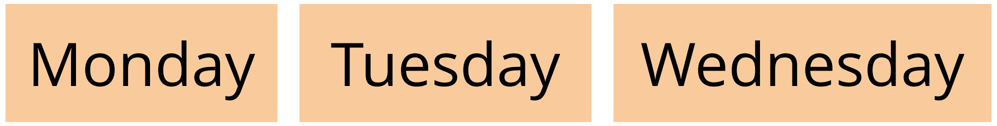
Display **inline-block** makes a **block** *or* **inline** element flow like an **inline** element, while allowing us to set a width, height, padding, and margin

```css
li {
     display: inline-block;
}
```

BEFORE:



AFTER:



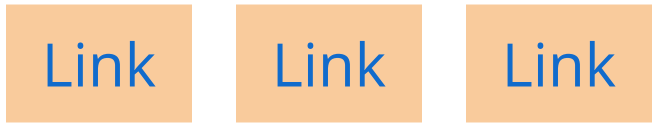
- Elements will sit next to each other
- **We can now set a width, height, and margin/padding on top and bottom!**

Display: block

Display **block** makes an inline element act like a block-level element

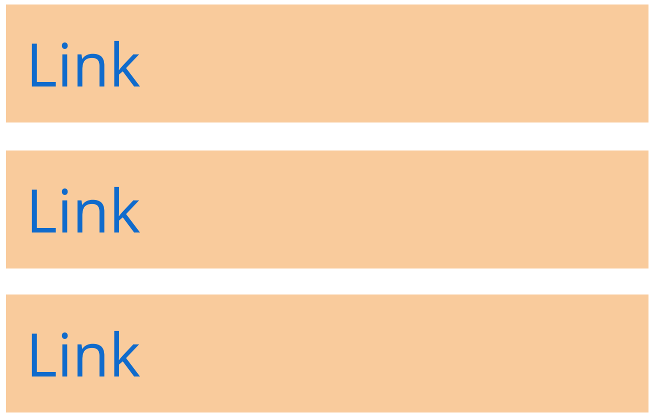
```
a {
    display: block;
}
```

BEFORE:



- Elements will stack on top of each other
- We can add all dimensions (e.g. width, height, margin, and padding)

AFTER:



Display: none

Display **none** hides an element from the page.

```css
li {
    display: none;
}
```

BEFORE:

Monday

Tuesday

Wednesday

AFTER:

- Elements will be hidden from the page.

Text-align Property

The **text-align** property in CSS is used for aligning the inner content of a block element.

	TEXT-ALIGN
BLOCK	yes
INLINE / INLINE-BLOCK	no

Centering Images & Containers

Images

```
img {
    display: block;
    margin: 0 auto;
}
```

Containers

```
div {
    width: 500px;
    margin: 0 auto;
}
```

Pulse Check

Key Objective:

Together, let's change the default display properties to see what happens: CodePen

Timing:

- **3 minutes** - code along in the previous CodePen

Padding, Border, Margin

Padding Area

The **padding area** contains the space between the content and its border.

```
element {
    padding-top: 10px;
    padding-right: 30px;
    padding-bottom: 10px;
    padding-left: 30px;
}

/* Alternatively, you could use the 'padding' shorthand property */

element {
    padding: 20px; /* apply to all four sides */
    padding: 10px 30px; /* vertical | horizontal */
    padding: 10px 30px 10px; /* top | horizontal | bottom */
    padding: 10px 30px 10px 30px; /* top | right | bottom | left */
}
```

Margin Area

The **margin area** contains the space outside of an element -- between the border and its neighboring elements.

```
element {
    margin-top: 5%;
    margin-right: auto;
    margin-bottom: -10px;
    margin-left: 30px;
}

/* Alternatively, you could use the 'margin' shorthand property */

element {
    margin: 20px; /* apply to all four sides */
    margin: 1em auto; /* vertical | horizontal */
    margin: 10px 30px 10px; /* top | horizontal | bottom */
    margin: 10px 30px 10px 30px; /* top | right | bottom | left */
}
```

Border

The **border** separates the padding and the margin.

```
element {
    border-width: 5px;
    border-style: dashed;
    border-color: black;
}

/* Alternatively, you could use the 'border' shorthand property: */

element {
    border: solid; /* border style */
    border: 2px solid; /* border width | border style */
    border: dotted #ff0000; /* border style | border color */
    border: 10px ridge rgba(0,0,0,.5); /* border width | border style | border color */
}
```

Pulse Check

Slack Check

I want to add some styling to a few elements; how might I do that using CSS? Answer the following questions in a Slack thread, one at a time:

1. Add 5px padding to **a**
2. Add a solid, green border that is 10px wide to a **p**
3. Center an **img** with 10px margin top/bottom

Code Along

Open: **starter_code/tags_boxes**

Directions:

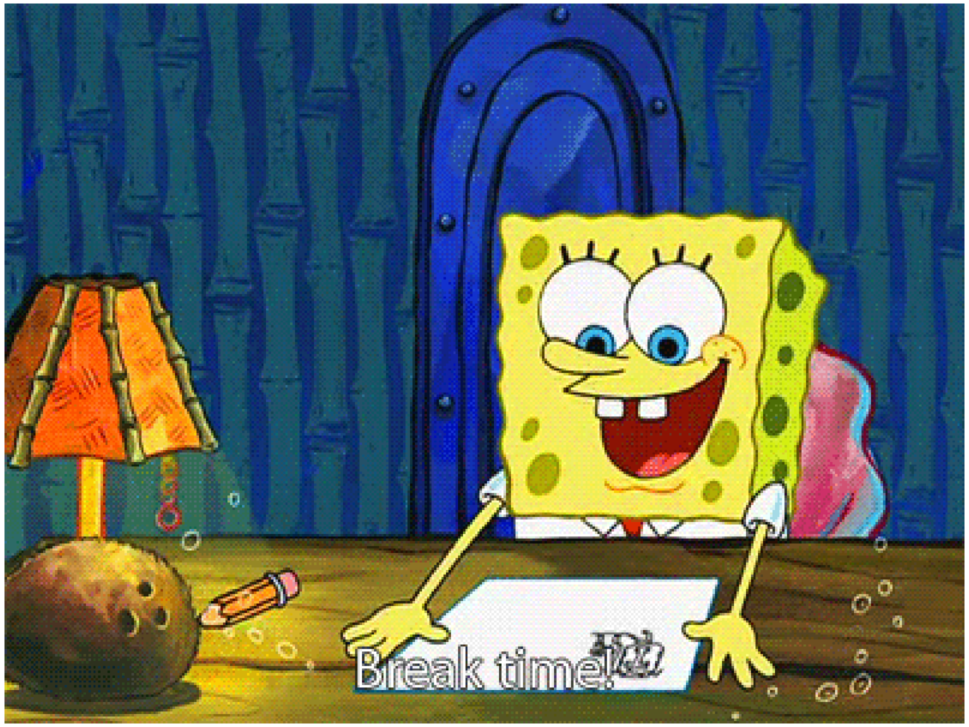
1. Open the index.html file in the **Week_2_Styling/starter_code/tags_boxes** folder
2. Follow the directions to complete the code along

Timing:

- **~10 minutes**

Bonus:

- Make it your own! Customize it as much as you want



Break time!

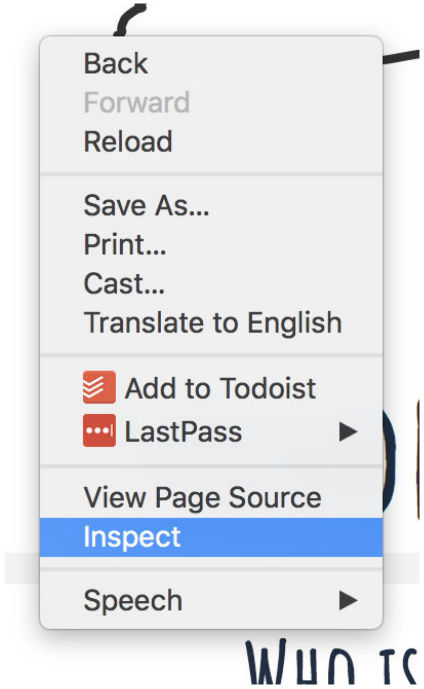
Let's take 5-10 minutes to decompress...

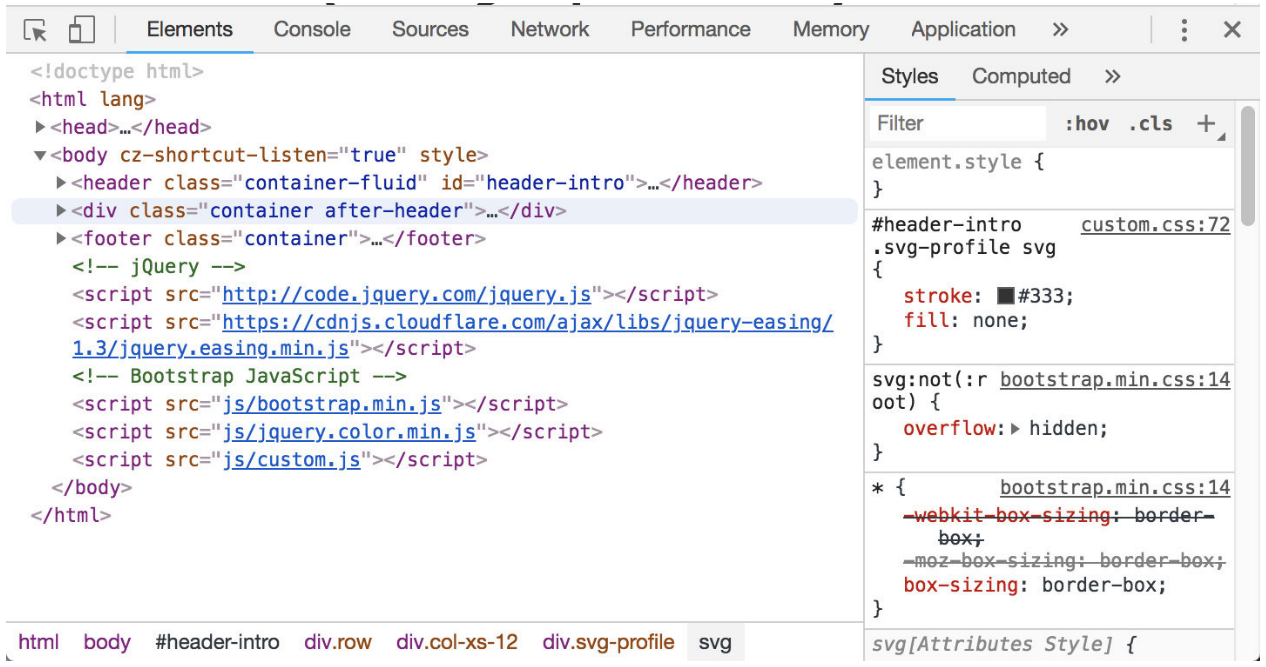
Chrome Developer Tools

Troubleshoot HTML/CSS

Accessing Chrome Developer Tools

- **Right-click** on the page → click **Inspect**



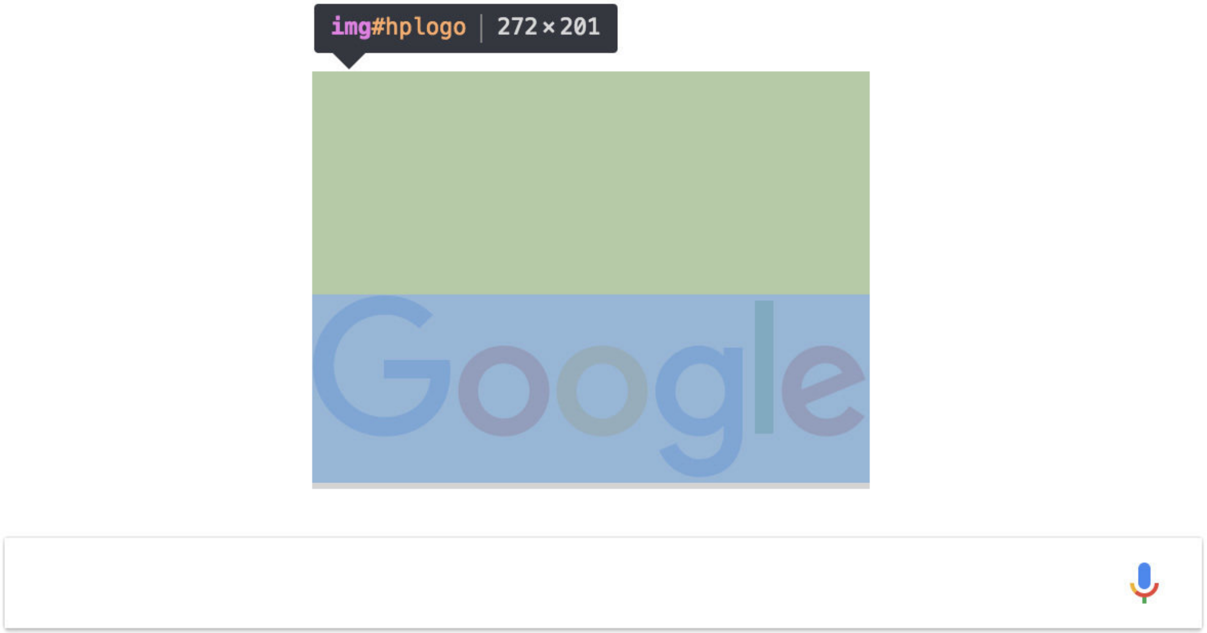


```
<!doctype html>
<html lang>
▶<head>…</head>
▼<body cz-shortcut-listen="true" style>
  ▶<header class="container-fluid" id="header-intro">…</header>
  ▶<div class="container after-header">…</div>
  ▶<footer class="container">…</footer>
   <!-- jQuery -->
   <script src="http://code.jquery.com/jquery.js"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/
   1.3/jquery.easing.min.js"></script>
   <!-- Bootstrap JavaScript -->
   <script src="js/bootstrap.min.js"></script>
   <script src="js/jquery.color.min.js"></script>
   <script src="js/custom.js"></script>
  </body>
</html>
```

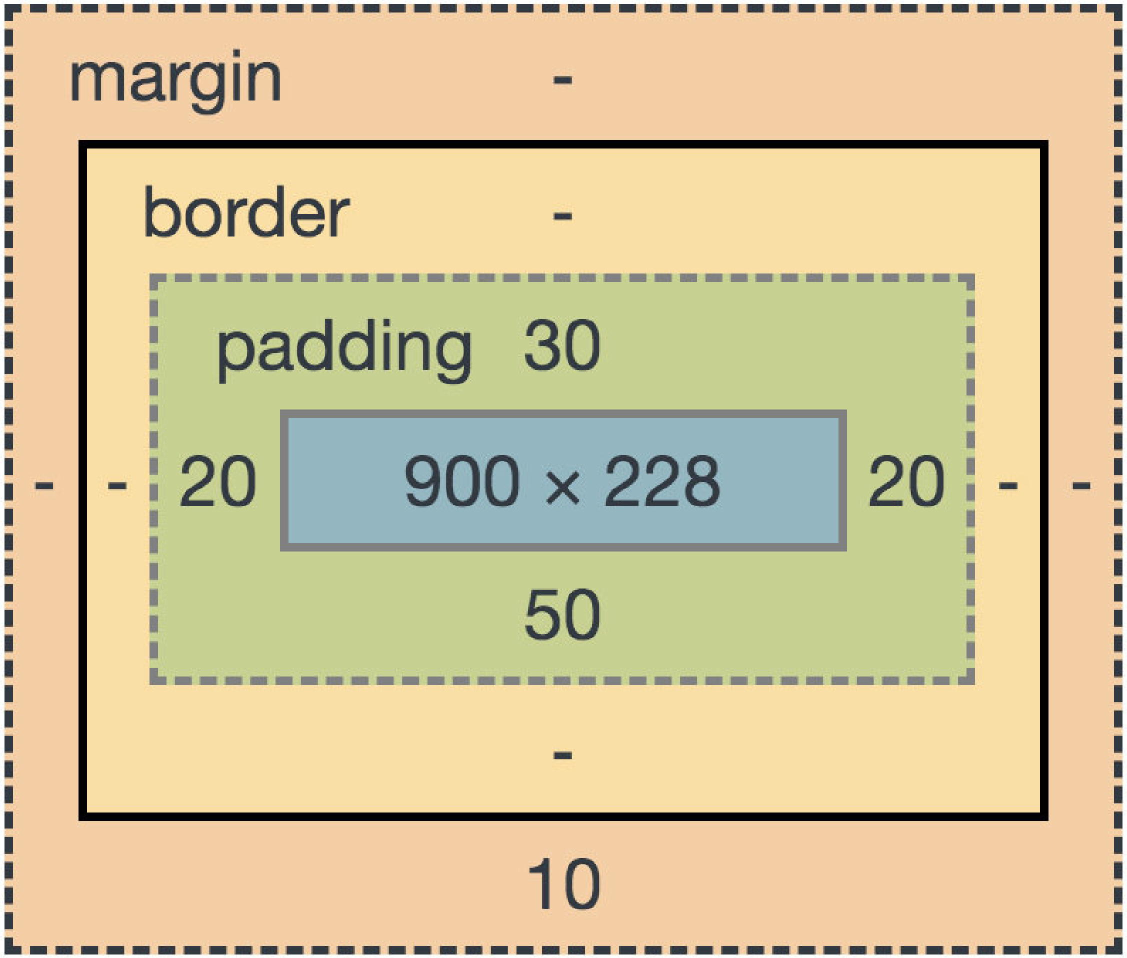
```
element.style {
}
#header-intro          custom.css:72
.svg-profile svg
{
   stroke: ■#333;
   fill: none;
}
svg:not(:r bootstrap.min.css:14
oot) {
   overflow:▶ hidden;
}
* {             bootstrap.min.css:14
   ~~-webkit-box-sizing: border-box;~~
   ~~-moz-box-sizing: border-box;~~
   box-sizing: border-box;
}
svg[Attributes Style] {
```

html   body   #header-intro   div.row   div.col-xs-12   div.svg-profile   svg

Live Box Model



Live Box Model Values



Pulse Check

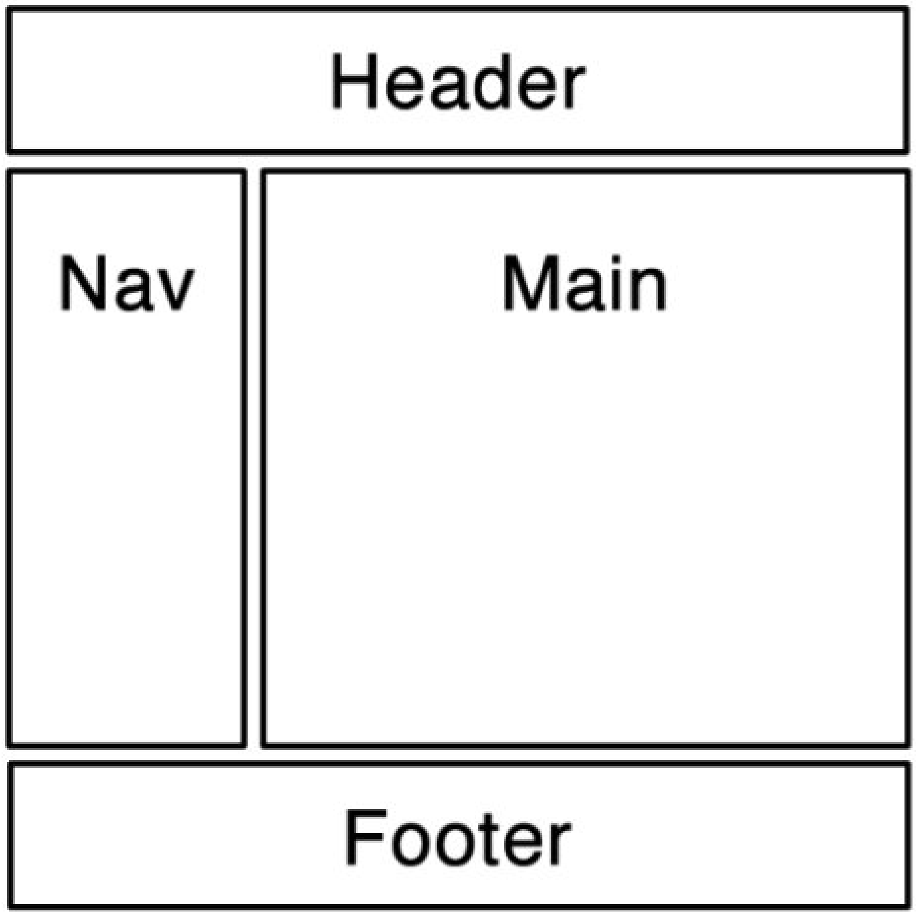
Zoom Check

Navigate to Msiddeeq.com. Together, lets use the Chrome Dev Tools to find the following information about my page:

1. How much padding is on the **p** elements?
2. Does the **header** have a border? If so, how much?
3. How much margin is on the intro **svg** element?

Structural Elements

Traditional Layout

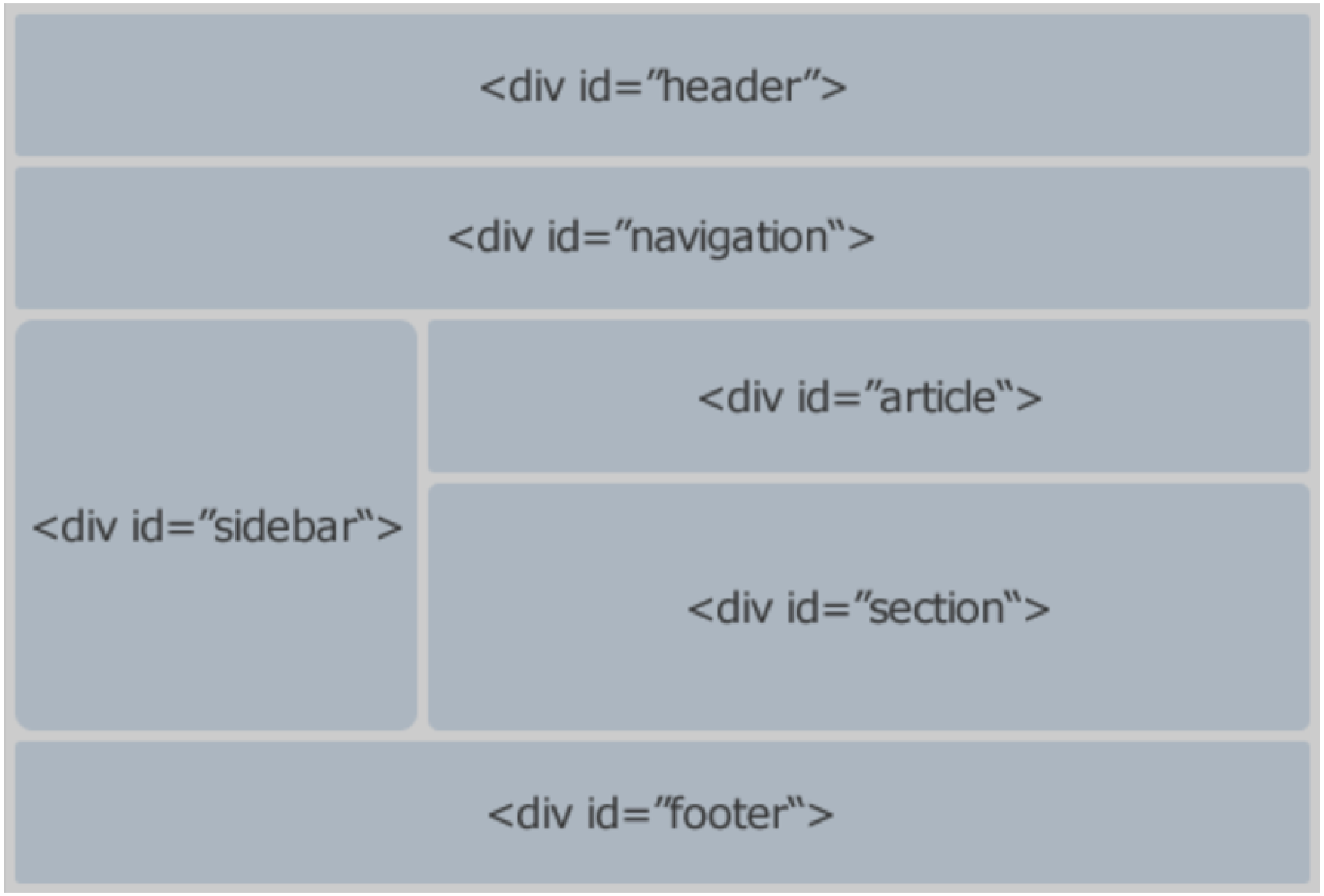


DIV Element

The **HTML Content Division element** (aka **div**) is a block-level element that defines a division or a section in an HTML document.

It is a generic container element that is primarily used to group content so it can be easily styled in CSS.

HTML Structure w/ DIVs

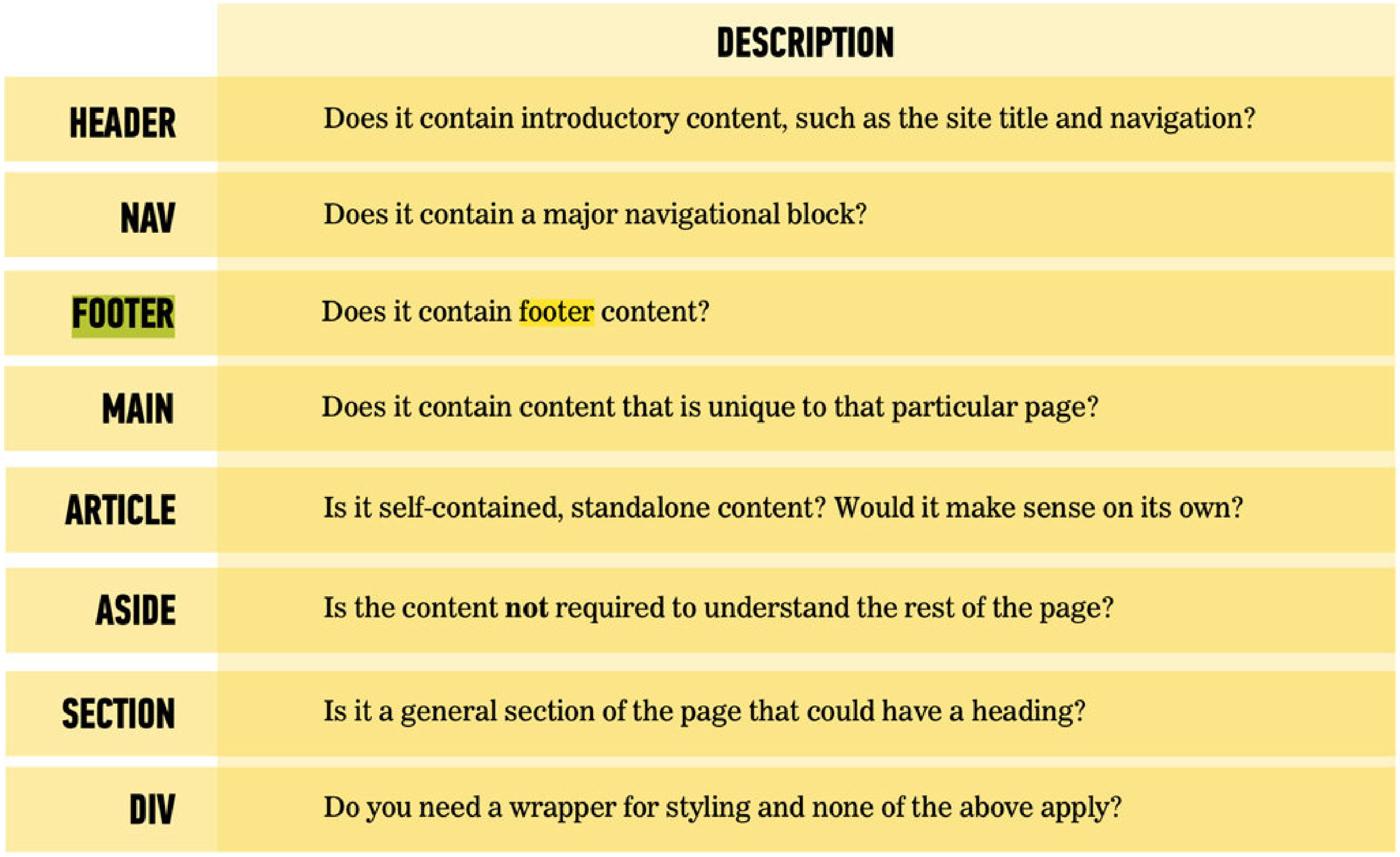


HTML5 Structural Elements

However, with the advent of **HTML5**, we now have structural elements specific to the content we expect to be placed within it.

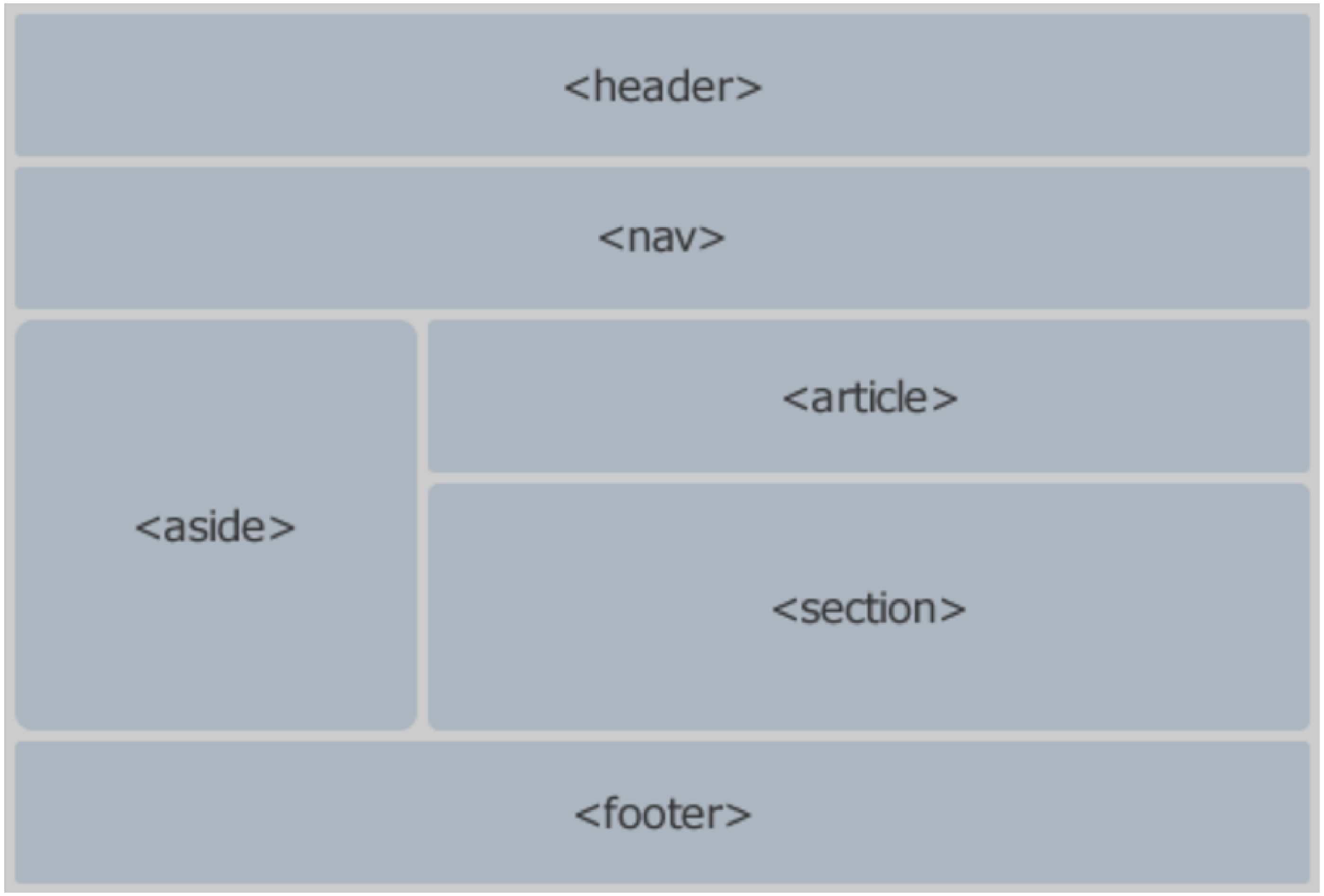
It gives our structure a bit of meaning.

HTML5 Structure Examples



	DESCRIPTION
HEADER	Does it contain introductory content, such as the site title and navigation?
NAV	Does it contain a major navigational block?
FOOTER	Does it contain footer content?
MAIN	Does it contain content that is unique to that particular page?
ARTICLE	Is it self-contained, standalone content? Would it make sense on its own?
ASIDE	Is the content **not** required to understand the rest of the page?
SECTION	Is it a general section of the page that could have a heading?
DIV	Do you need a wrapper for styling and none of the above apply?

HTML5 Structure



Nav

The **nav** is used to group together major navigational blocks on a page.

Example content:

- Groups of 2 or more links
- Can be images or text

```
<nav>
    <a href="#">Home</a>
    <a href="#">Pricing</a>
    <a href="#">About</a>
    <a href="#">Contact</a>
</nav>
```

Header

The **header** is used to group elements in the opening section of a page.

Example content:

- any main headings on your site
- the main navigation
- the logo
- search bar
- any other intro content

```
<header>
    <h1>My Favorite Element</h1>

    <nav>
        <a href="#">Home</a>
        <a href="#">Pricing</a>
        <a href="#">About</a>
        <a href="#">Contact</a>
    </nav>
</header>
```

Footer

The **footer** is used for the concluding section of a page.

Example content:

- copyright info
- authorship info
- contact info
- secondary navigation
- social media links

```
<footer>
    <nav>
        <a href="#">Terms</a>
        <a href="#">Privacy</a>
        <a href="#">Sitemap</a>
    </nav>

    <p>Copyright 2018</p>
</footer>
```

Section

The **section** represents a general section of an HTML document (typically with a heading) which doesn't have a more specific semantic element to represent it.

EXAMPLE:

a portfolio page could be split into an about section, a portfolio section, and a contact section.

```
<section>
    <h2>This is my Section</h2>

    <article>
        <p> ... </p>
    </article>
</section>
```

Main

The **main element** groups content that is unique to a particular page. This would exclude any content on a site that might be repeated on multiple pages, such as the navigation and footer.

```
<main>
    <h2>My Background</h2>

    <p>It was the best of times, it was the worst of times...</p>
</main>
```

Article

The **article element** represents a self-contained, standalone section of an HTML document.

This element could be used just once on a page -- for example, a blog post may be a single article -- or multiple times -- for example, an online newspaper page may have numerous articles.

```html
<article>
    <h2>My First Blog Post</h2>

    <p>Dear diary...</p>
</article>
```

Pulse Check

Key Objective:

Navigate to Uber.com. Go through the page and discuss which structural element you would use for each section of the website.

Timing:

- **5 minutes** - Discuss with a partner in a breakout room
- **3 minutes** - Discuss as a group

Tip:

Don't look at the source code!

Key Objective:

Practice structuring a page with HTML5 elements in our **week_01_review** folder from earlier

Timing:

- **2 minutes** - Add a header around the h1 and h2 elements
- **~1 minutes** - Style the header in CSS by giving it a white background-color

Lab Time

Open: **Week_01_Basics/Assignment/starter_code/**

Directions:

1. Let's continue working on last week's homework
2. Open the **Week_01_Basics Assignment** folder

Timing:

- **10 minutes** - Discuss how to start a project (together)
- **45 minutes** - Begin developing site based on attached *relaxr_landing.jpg* image

Tips:

- Use your resources!
- Priya and I are here for any questions you might have

Submit HW to GHE

- All of your code should be in the appropriate folder within the **FEWDR-422** folder
- Remember to save all of your changes/close any open files
- **Rename** "Assignment" folder to "Assignment-{INITIALS}"
- **Open** GH Desktop
- Add a brief description to the "Summary" field
- **Click** "Commit to master"
- **Click** "Push Origin"
- **Share** URL to "Assignment-{INITIALS}" folder with TA

IMPORTANT!!

Double check your assignment has been submitted by navigating to **your forked repo** in your browser and reloading the page.

If you do not see your files, they were not uploaded.

Send Slack message to TA or myself, immediately.

Exit Tickets

Take 5-10 minutes to give us some

(Link is in Slack Room)

Learning Objectives Review

- We defined the CSS Box Model, and demonstrated the ability to properly manipulate the "box" around tags, including margin and padding.
- We applied inline and block attributes to a page.
- We explored the different HTML structural elements
- We troubleshooted our HTML/CSS using the Chrome Developer Tool

FEWDR Homework Recap

- All homework files will be included in the week's Assignment folder in GHE
- Look for this folder when you pull the remote class repo Monday before class
- Grades and feedback will be provided directly on the **rubric.md** file in your Assignment folder
- Once graded, we will push the updated file to your repo

Next Class...

Lesson 4 - CSS Box Model Lab