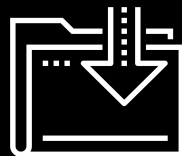




Intro to GitHub Pages and CSS

Data Boot Camp
Lesson 11.2



Class Objectives

By the end of today's class you will be able to:



Have a firm understanding of how to deploy HTML webpages to the internet using GitHub pages.



Basic understanding of CSS styling.



Basic knowledge how to position HTML elements on a web page using CSS.



Activity: HTML Warm-up

In this activity, you will create a simple HTML bio page to serve as a personal info.

Suggested Time:
10 Minutes



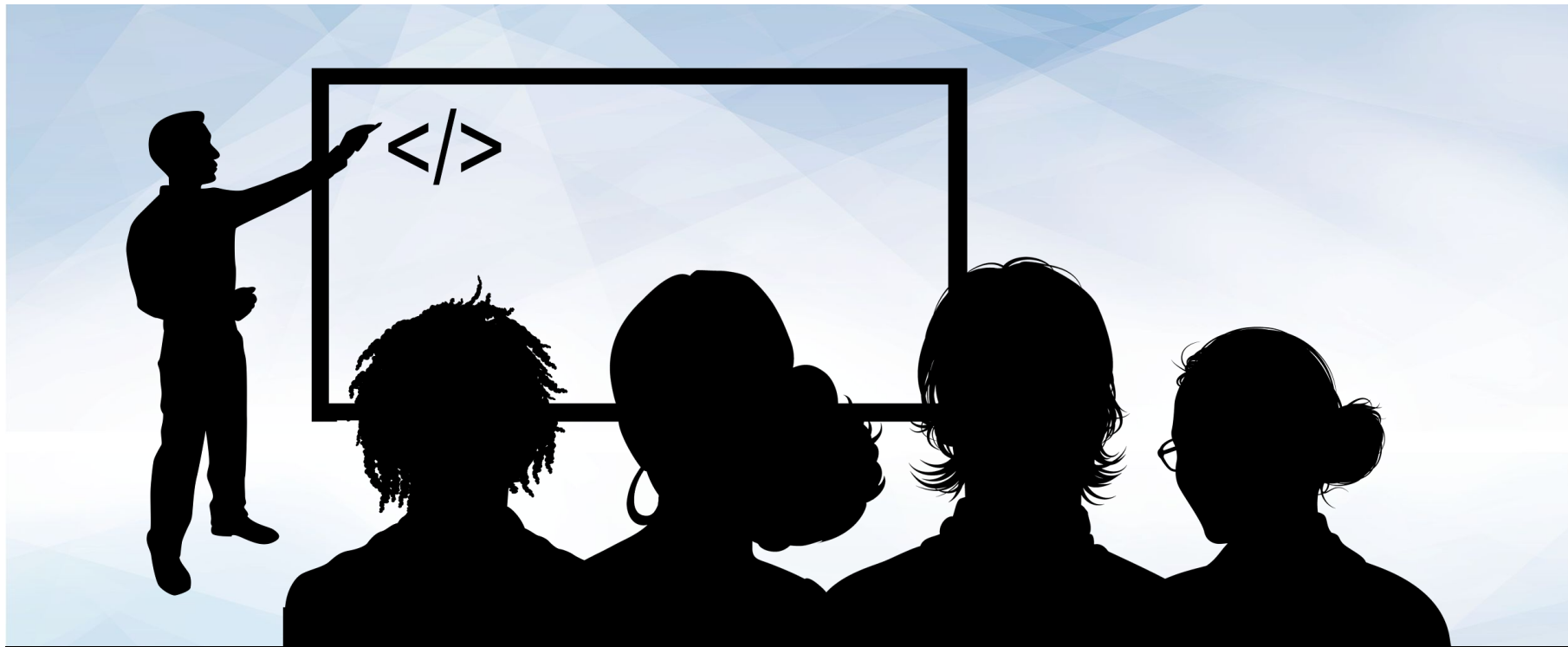
Instructions: Activity: Inspect Hello, HTML

On your own, create a simple bio page for yourself using the following HTML elements...

- Header that will store your name inside of it
- An image that will act as a stand-in for your picture with an alt attribute which gives a very basic description of the image
- Two paragraphs that will have text describing who you are
- An unordered list of links that connect to your social media pages
- A table that will contain three columns and some data on your favorite movies, songs, books, or activities
- **Hints:**
 - Dummy images can be found at loremipsum.com and dummy text can be found at loremipsum.com. Focus on getting the entire page working before diving into more specific text and images.
- **Bonus:**
 - Look into how one might go about linking one custom-made HTML page to another. Once you have found out how, try creating a link from your bio page you have just created to one of the pages you created last class.



Time's Up! Let's Review.



Instructor Demonstration

Deploying to GitHub Pages - Personal

Deploying to GitHub Pages - Personal

The concept of a 'host'

- A web host is the activity or business of providing storage space and access for websites. You cannot put a website online without it being hosted on a server somewhere.

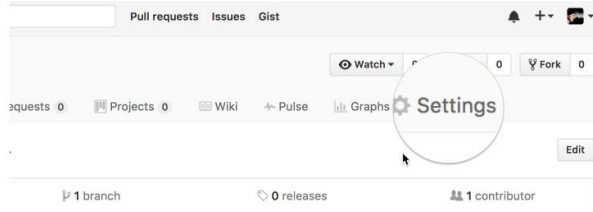
Deploying to GitHub Pages - Projects

1

Repository Settings

Head over to [GitHub.com](https://github.com) and create a new repository, or go to an existing one.

Click on the Settings tab.



2

Theme chooser

Scroll down to the **GitHub Pages** section. Press **Choose a theme**.

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more](#).

None ▾

Save

Theme chooser

Select a theme to build your site with a Jekyll theme using the `master` branch. [Learn more](#).

Choose a theme



Activity: Deploying Personal Bios to GitHub Pages

In this activity, you will be deploying the bio pages you created on the previous activity to GitHub pages.

Suggested Time:
10 Minutes



Activity: Deploying Personal Bios to GitHub Pages

- Now that we have gone over how to create a personal website using GitHub Pages, it is your turn to publish the bio you created!
- Once your personal webpage is live, slack it out so that everyone can see what you have made.

Deploy Guide:

1. Create a new repository that is named *bio_page*
2. Navigate into a folder and clone the repository into it
3. Create a folder named docs and add your html bio to that folder
4. Rename the html file to index.html
5. Add, commit, and push your changes into the repository
6. Go to the repository on GitHub and enter the settings menu
7. Go to the Pages tab and setup github Pages for that repo:
 - a. Pick the master branch
 - b. Select the docs folder
 - c. Save



Instructor Demonstration

Introduction to Basic CSS Styling

CSS: Cascading Style Sheets

Introduction to Basic CSS Styling



- CSS describes how and where elements should appear on the page. It defines things such as color, placement, fonts, sizes, and more.

CSS



HTML/CSS Analogy

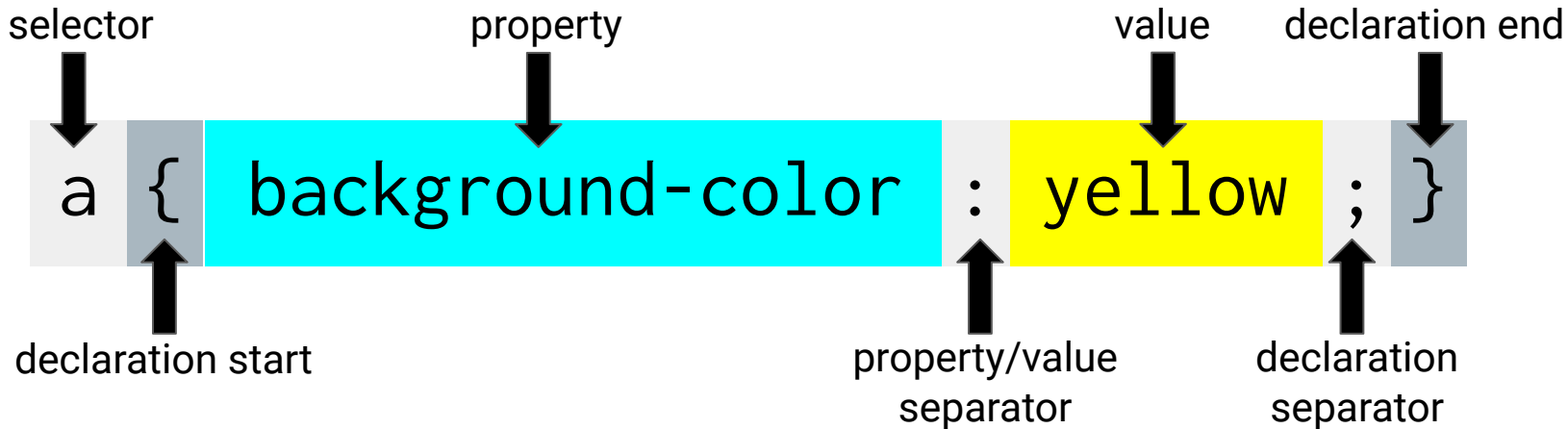
Introduction to Basic CSS Styling

HTML Alone	HTML and CSS
Like writing papers in Notepad.	Like writing papers in Microsoft Word.
Used to write unformatted text (i.e, content only).	Used both to write the content <i>and</i> format it (color, font, alignment, layout, etc.).
	

CSS Syntax

Introduction to Basic CSS Styling

- CSS works by hooking onto **selectors** added into HTML using **classes** and **identifiers**.
- Once hooked, we apply **styles** to those HTML elements using CSS.



<Time to Code>





Activity: Dull Corp

In this activity, you will be updating the *DULL Corporation's* website so that it is not nearly so...Dull. To do so, you will be creating an external stylesheet and linking it to pre-made HTML.

Suggested Time:
10 Minutes



Instructions: Activity: Dull Corp

- Open the [Unsolved Starter](#).
- Using external CSS and your imagination, update the "DULL Corp" website to be more interesting.
 - Center the `header` element to the page.
 - Set each `h1`, `h2`, and `h3` element to be a different color.
 - Set `img` element to have a shadow.
 - Give the `section` element a background color.
 - Change the font size of both `paragraph` elements.
 - Place a border around the `ul` element.



Time's Up! Let's Review.

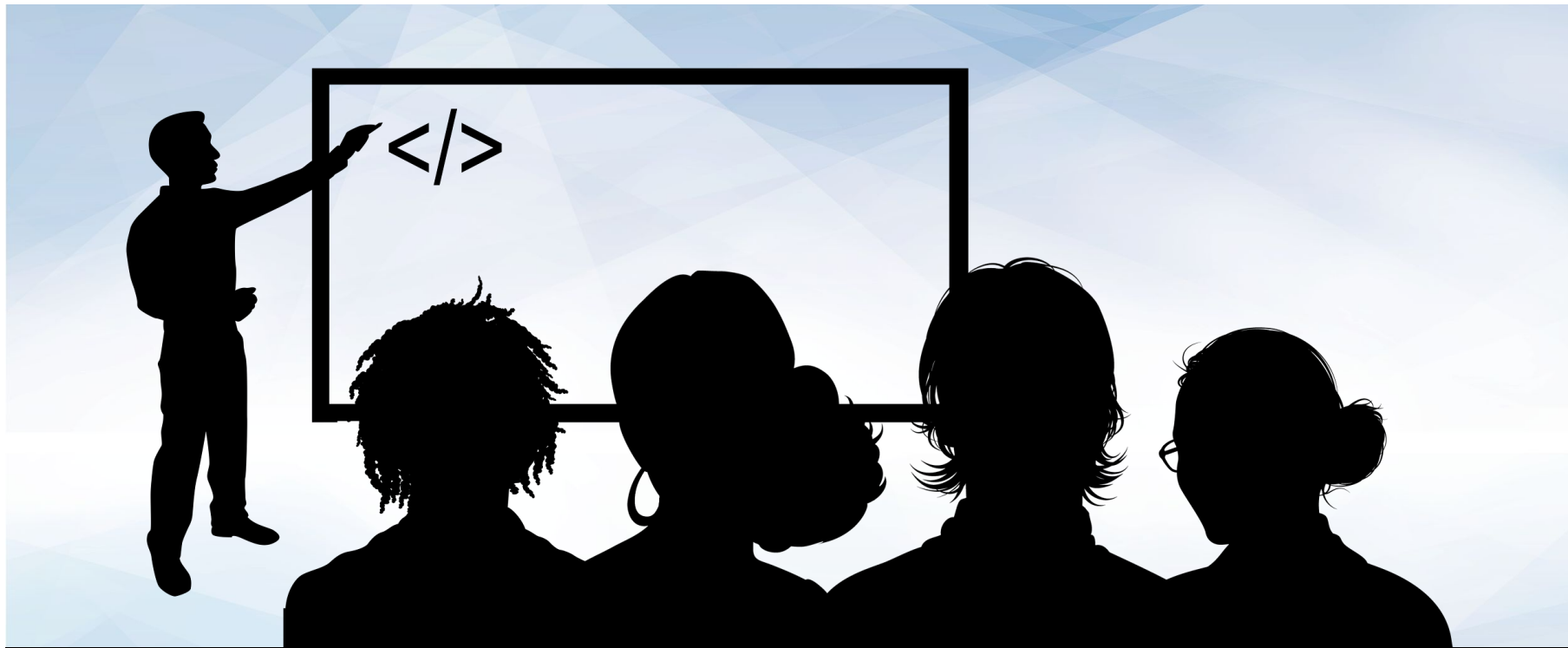




Countdown timer

15:00

(with alarm)

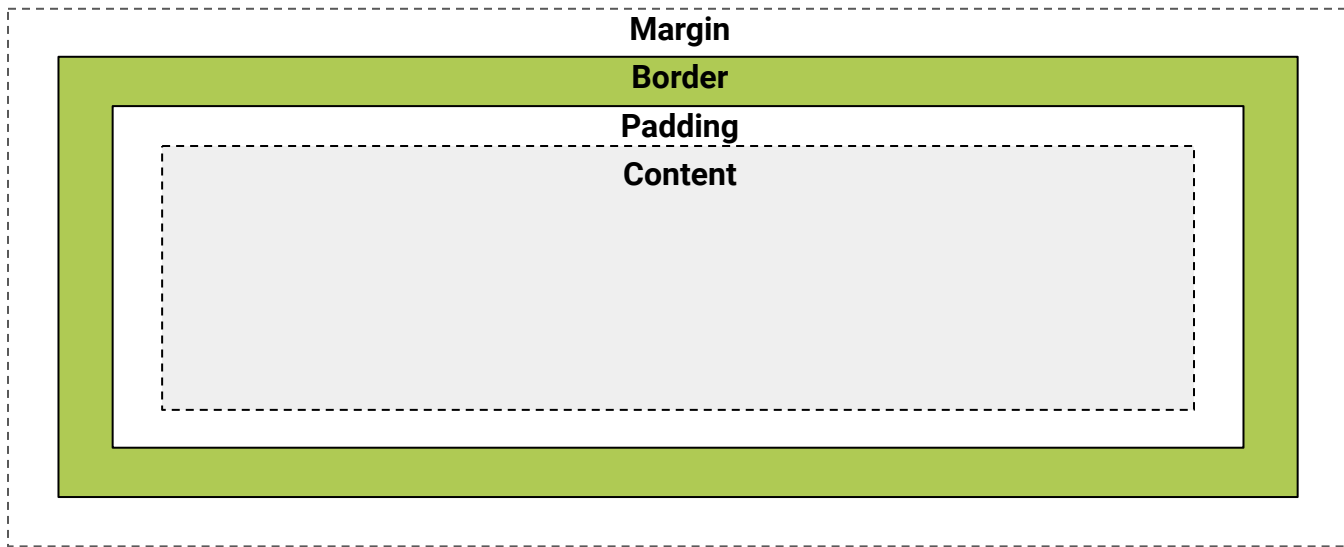


Instructor Demonstration

Box Model and CSS Positioning

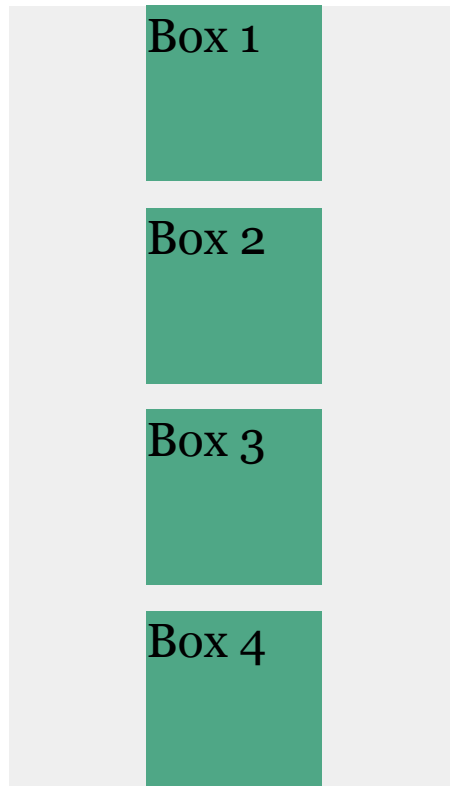
Boxes Upon Boxes Box Model and CSS Positioning

- In CSS, every elements rests within a series of Boxes.
- Each box has customizable space properties: margin, border and padding.
- Typical spacing value: 20px 10px 10px 20px (top, right, bottom, left).



Position: Static Box Model and CSS Positioning

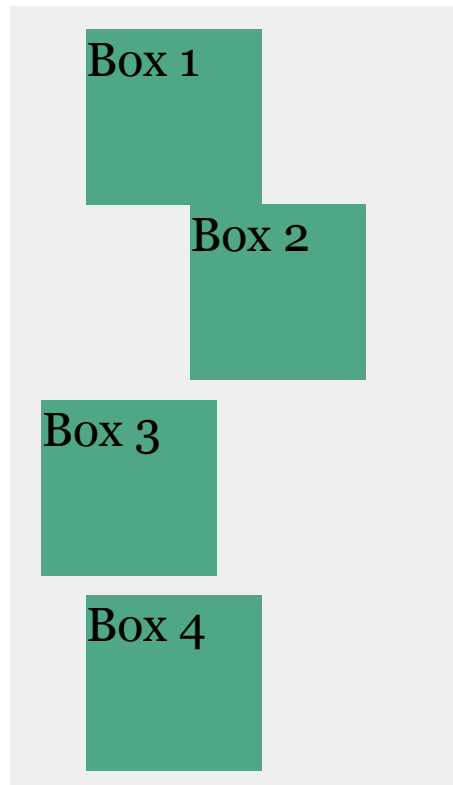
- Four boxes placed statically (default):



Position: Relative Box Model and CSS Positioning

- Switching the boxes to relative will nudge the boxes in relation to their “original” location:

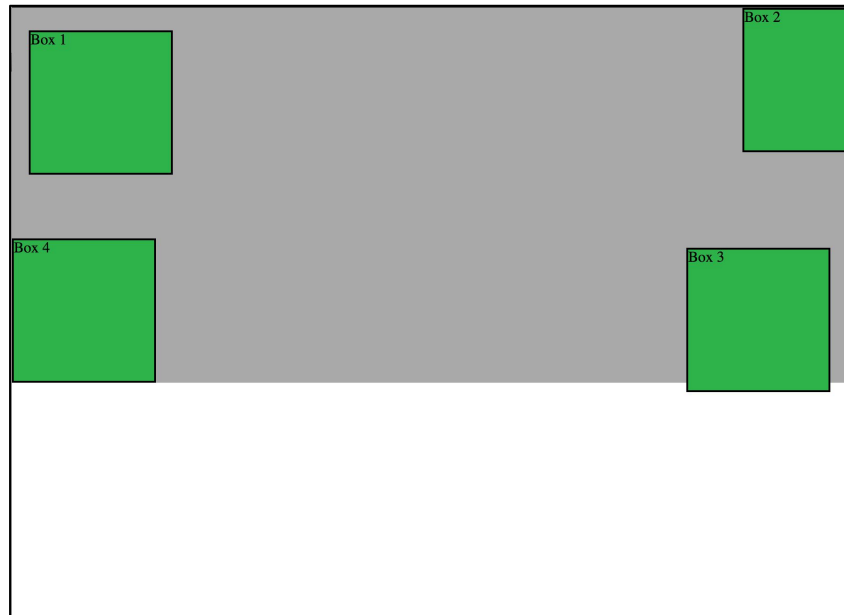
```
.box {  
  background: #2db34a;  
  height: 80px;  
  position: relative;  
  width: 80px;  
}  
.box-1 {  
  top: 20px;  
}  
.box-2 {  
  left: 40px;  
}  
.box-3 {  
  bottom: -10px;  
  right: 20px;  
}
```



Position: Absolute Box Model and CSS Positioning

- Positioned relative to nearest positioned ancestor.
- Percentages define anchors (i.e. position of each box relative to enclosing box)

```
.box-set {  
  height: 400px;  
  background: darkgray;  
  position: relative;  
}  
.box {  
  position: absolute;  
  height: 150px;  
  width: 150px;  
  background: #2db34a;  
  border: 2px solid black;  
}  
.box-1 {  
  top: 6%;  
  left: 2%;  
}  
.box-2 {  
  top: 0;  
  right: -40px;  
}  
.box-3 {  
  bottom: -10px;  
  right: 20px;  
}  
.box-4 {  
  bottom: 0;  
}
```

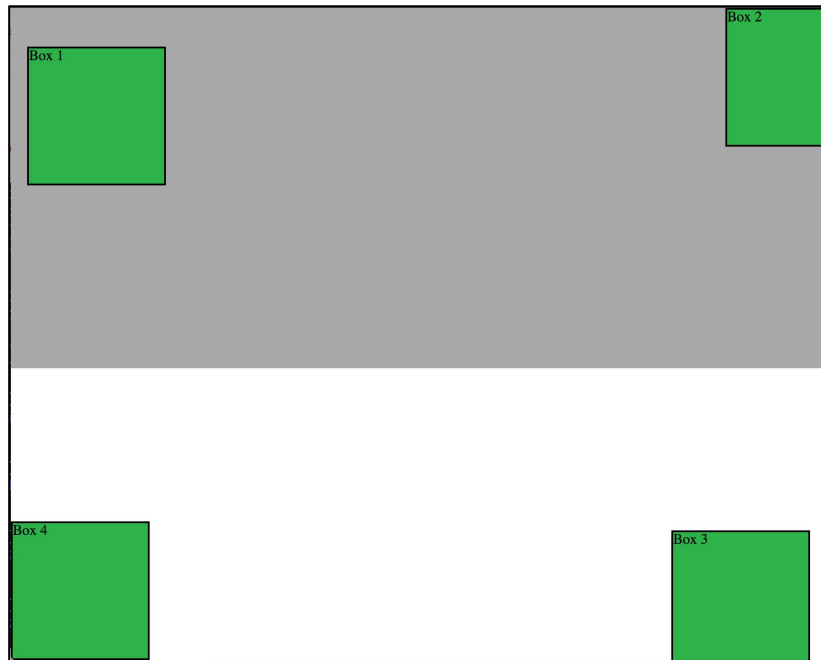


Position: Fixed

Box Model and CSS Positioning

- Position with exact coordinates in the browser window:

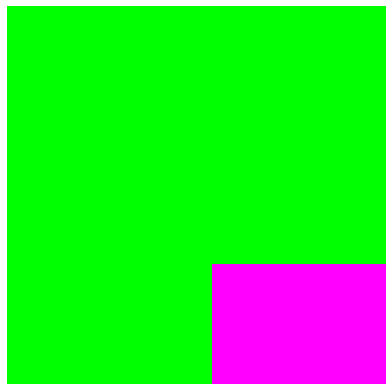
```
.box-set {  
  height: 400px;  
  background: darkgray;  
}  
.box {  
  position: fixed;  
  height: 150px;  
  width: 150px;  
  background: #2db34a;  
  border: 2px solid black;  
}  
.box-1 {  
  top: 6%;  
  left: 2%;  
}  
.box-2 {  
  top: 0;  
  right: -40px;  
}  
.box-3 {  
  bottom: -10px;  
  right: 20px;  
}  
.box-4 {  
  bottom: 0;  
}
```



Layering with Z-Index

Box Model and CSS Positioning

- The z-index property allows you to layer elements on top of each other.



```
position: absolute;  
z-index:1;
```



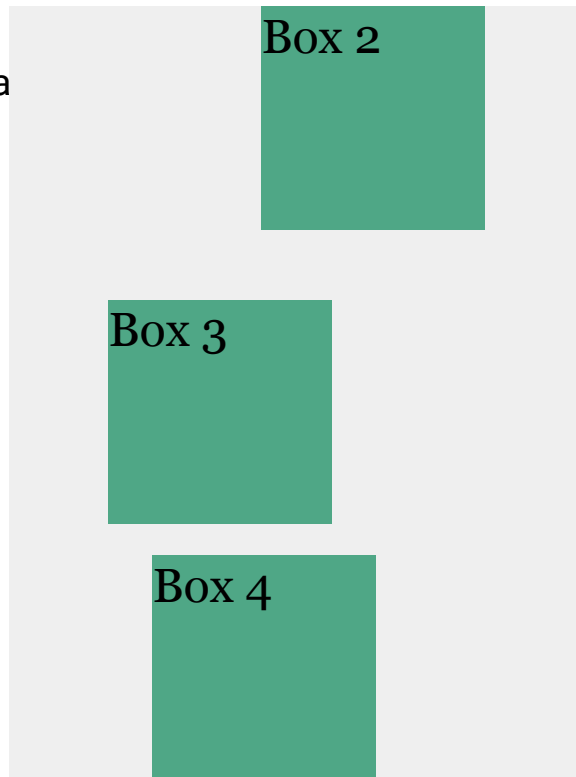
```
position: absolute;  
z-index:2;
```

Hiding things

Box Model and CSS Positioning

- Display: None allows you to hide elements from view.
- This will become useful in later sections, we'll hide and reveal specific HTML elements of our choosing.

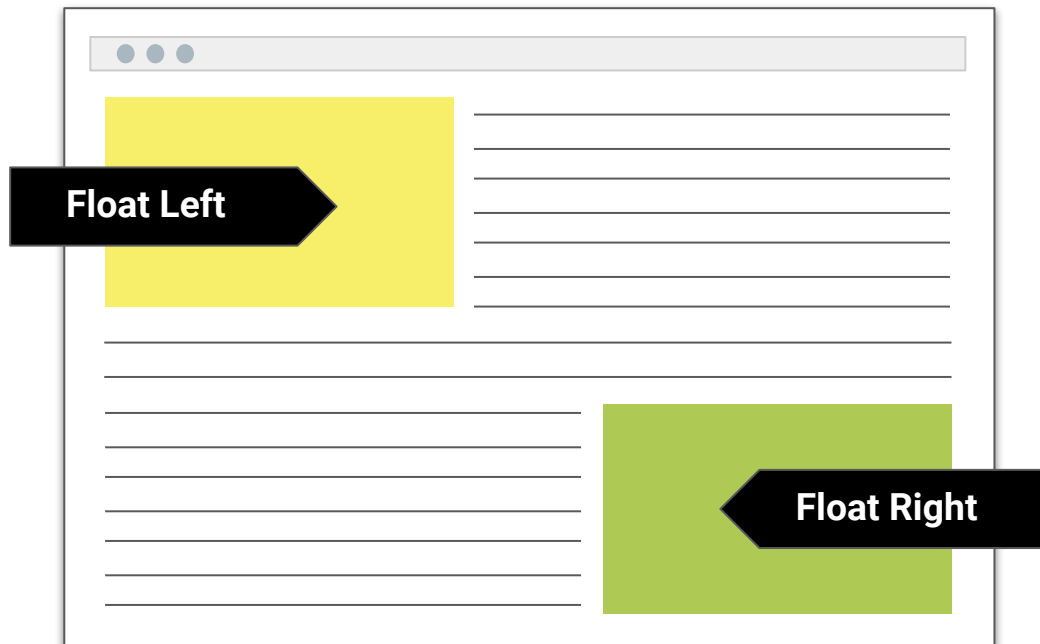
```
.box-1 {  
  display: none;  
}
```



The concept of Flow

Box Model and CSS Positioning

- By default, every HTML element displayed in the browser is governed by a concept called **flow**.
- This means that HTML elements force their adjacent elements to **flow around them**.



Flow analogy to MS Word

Box Model and CSS Positioning

- This concept of “flow” is very similar to the **wrap-text options** you may be familiar with in Microsoft Word.
- Just as in MS Word, you can have images in-line with text, on-top of text, etc.



Block Elements

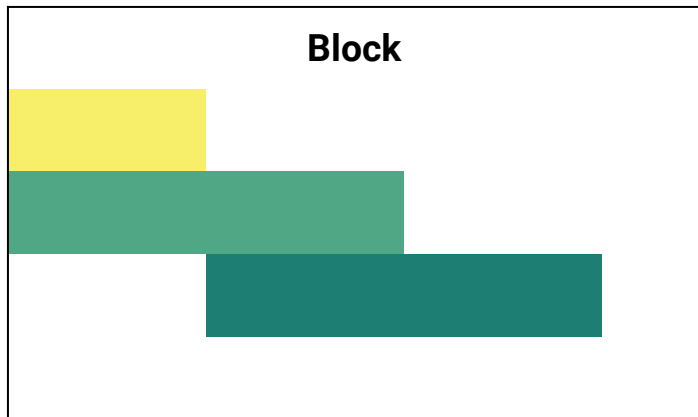
Box Model and CSS Positioning



By default, web clients render many HTML elements as **block elements**. Paragraphs, headers, divs, and more receive this treatment.



A block element will take up an entire line of space—unless you intervene with CSS properties.



Block Elements vs. In-Line Elements

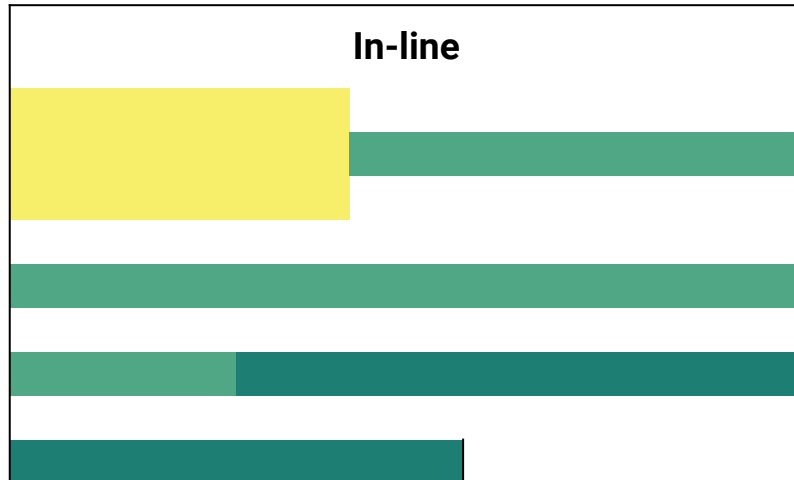
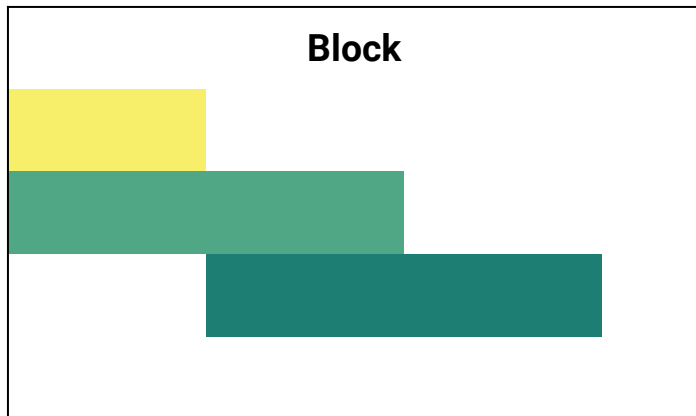
Box Model and CSS Positioning



Now, contrast block elements with **in-line elements**.



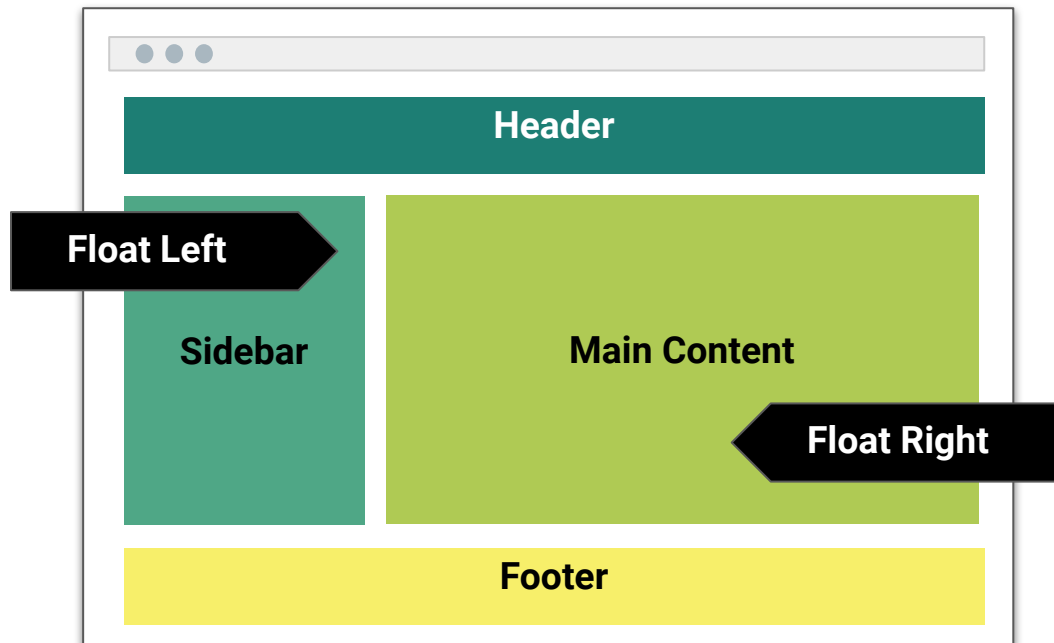
By using **float CSS properties**, we can command our website to display multiple HTML elements adjacently.



Floats

Box Model and CSS Positioning

- To transform these block elements into in-line elements, we use a CSS property called **float**. Floats are necessary when building web layouts.



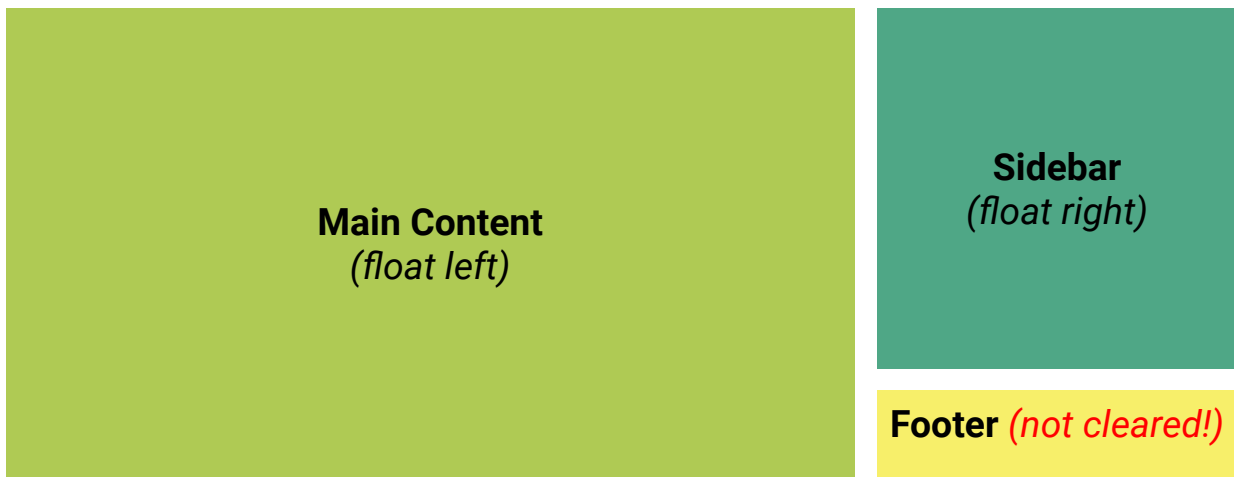
CSS

```
#sidebar {  
    float: left;  
}  
#main-content {  
    float: right;  
}
```


Clearing the Float

Box Model and CSS Positioning

- However, floats often get in the way of layouts. Sometimes we don't want to give each element the “in-line” treatment.



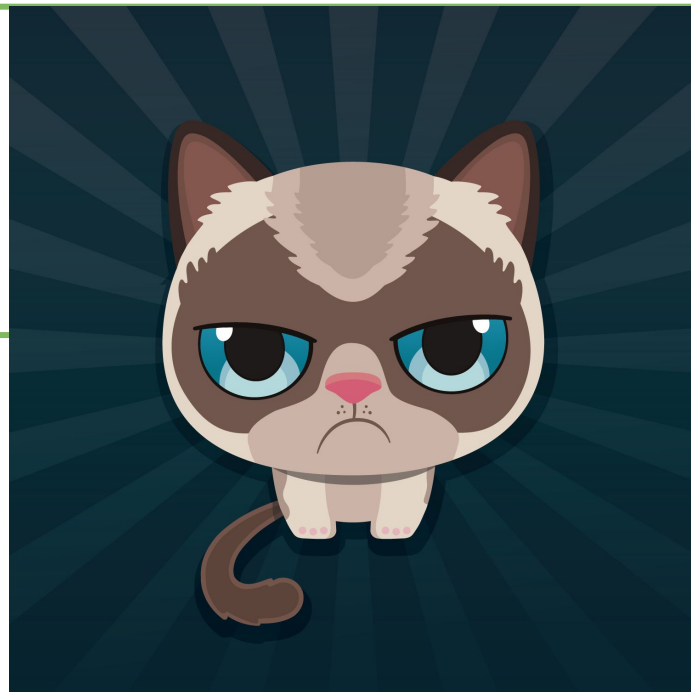
Clearfix Hack

Box Model and CSS Positioning

- Sometimes when elements don't match up in size, we get situations like this:

```
<div>
```

Uh-oh! The image is taller than the element containing it, and it's floated, so it's overflowing outside of its container!



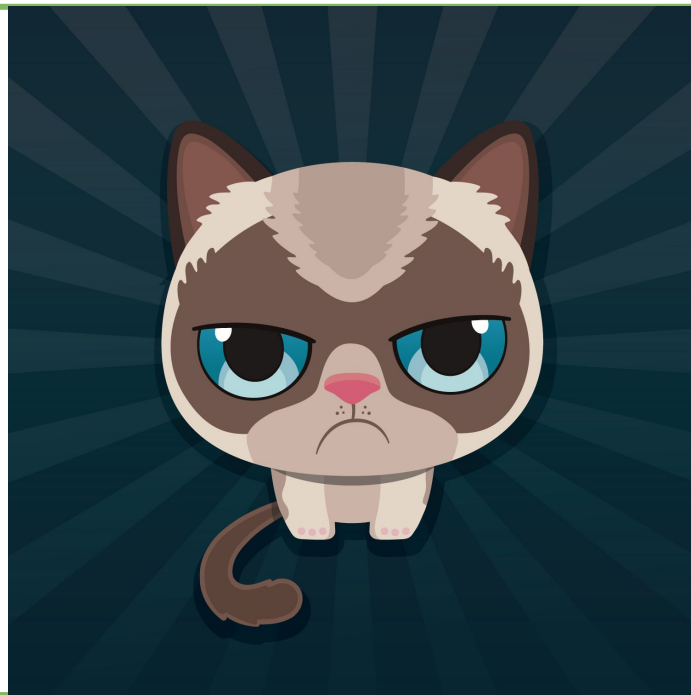
Clearfix Hack

Box Model and CSS Positioning

- We can get around this by using the **clearfix hack**.

```
<div class="clearfix">
```

Much better!



Clearfix Hack

Box Model and CSS Positioning



`::after` is what we call a pseudo-element. We use it to style specific parts of an element.



This will add an HTML element, hidden from view, after the content of the `.clearfix` element. This clears the float.

```
.clearfix::after {  
  content: "";  
  display: block;  
  clear: both;  
}
```



Activity: Aimed Positioning

In this activity, you will be given an HTML file you will style using CSS. In particular, they will be posting certain elements as described in the instructions.

Suggested Time:
10 Minutes



Instructions: Activity: Aimed Positioning

- Open the [Unsolved Starter](#).
- Using CSS, position the five headers in the starter code in their described locations.

Hints:



- You can move elements around the page using pixels OR percentages. Try out both to see how they work.
- Not every task is capable of being accomplished without changing the order of HTML elements. Even then, some positions may still be impossible.

Bonus:

- Try to move the HTML elements provided using different kinds of positioning (static, relative, and fixed).



Time's Up! Let's Review.

*The
End*