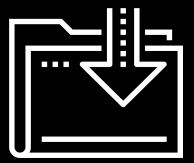
Intro to GitHub Pages and CSS

Data Boot Camp

Lesson 11.2

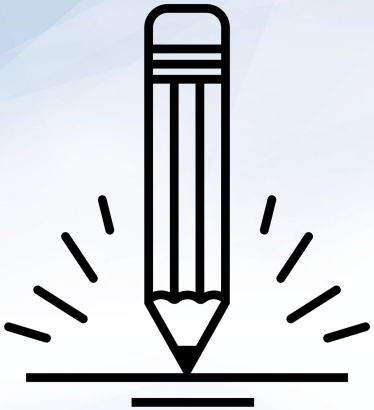


© 2019 Trilogy Education Services, Inc.

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

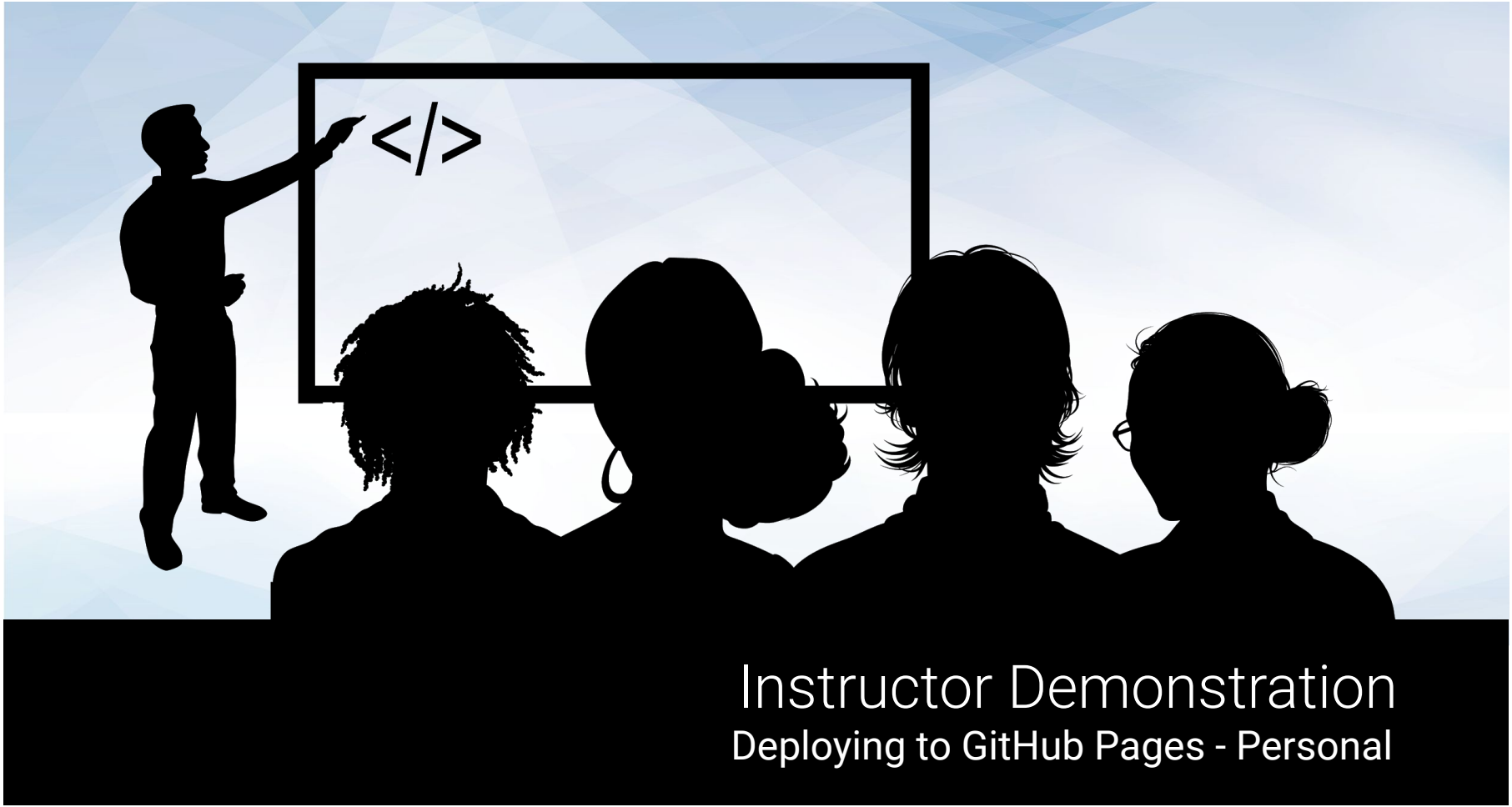
Instructions:

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 lorempixel.com and dummy text can be found at lipsum.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.

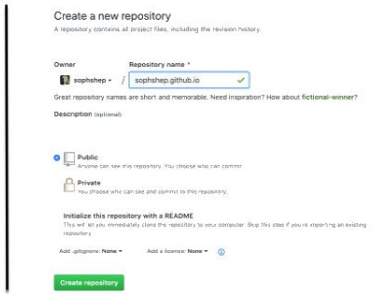
Deploying to GitHub Pages - Personal

1

Create a repository

Head over to GitHub and create a new repository named *username*.github.io, where *username* is your username (or organization name) on GitHub.

If the first part of the repository doesn't exactly match your username, it won't work, so make sure to get it right.



2

Clone the repository

Go to the folder where you want to store your project, and clone the new repository:

```
~ $ git clone https://github.com/username/username.github.io
```

3

Hello World

Enter the project folder and add an index.html file:

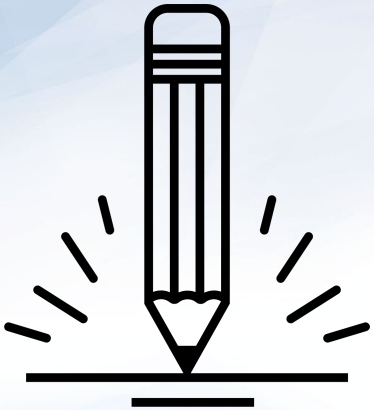
```
~ $ cd username.github.io
~ $ echo "Hello World" > index.html
```

4

Push it

Add, commit, and push your changes:

```
~ $ git add --all
~ $ git commit -m "Initial commit"
~ $ git push -u origin master
```



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

Instructions:

Activity: Deploying Personal Bios to GitHub Pages

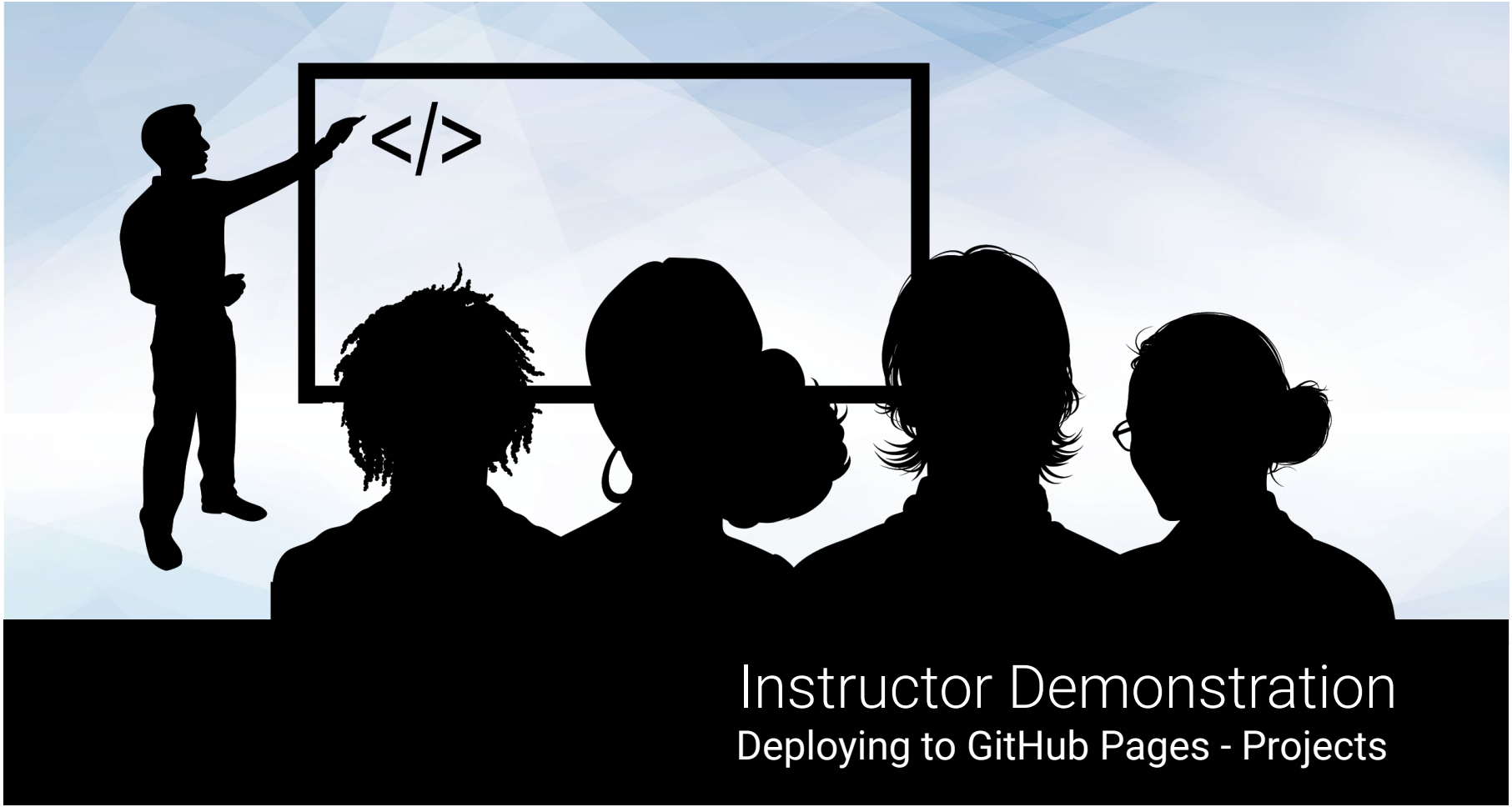
- Now that we have gone over how to use GitHub Pages, it's your turn to publish your bio
- Once your personal webpage is live, slack it out so that everyone can see what you have made.

Deploy Guide:

1. Create a new repository that is named *username*.github.io
2. Navigate into a folder and clone the repository into it
3. Add an HTML file named index.html and code out a basic webpage (or use a previous page)
4. Add, commit, and push your changes into the repository
5. Navigate to *username*.github.io and you will find that your new web page has gone live!

Bonuses:

- Add a few new pages to your website
- Add extra tags and flesh your page out some more - give it some pop

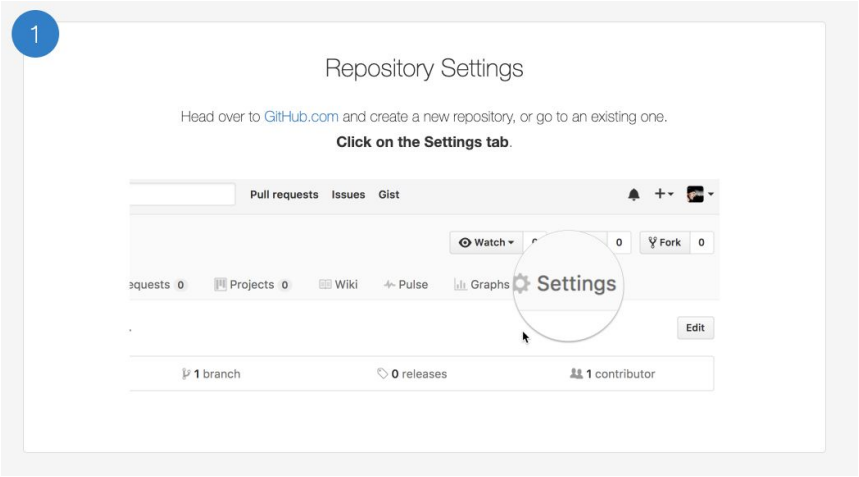


Instructor Demonstration

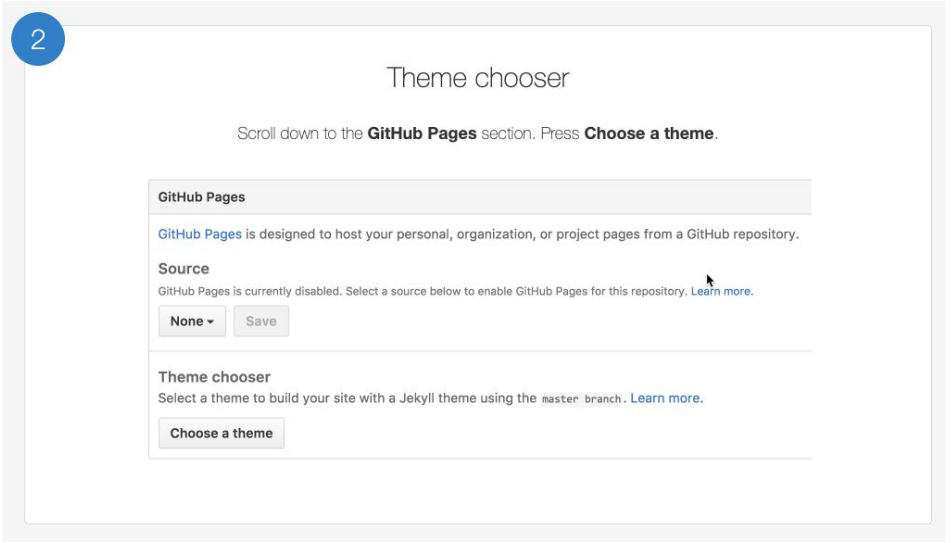
Deploying to GitHub Pages - Projects

Deploying to GitHub Pages - Projects

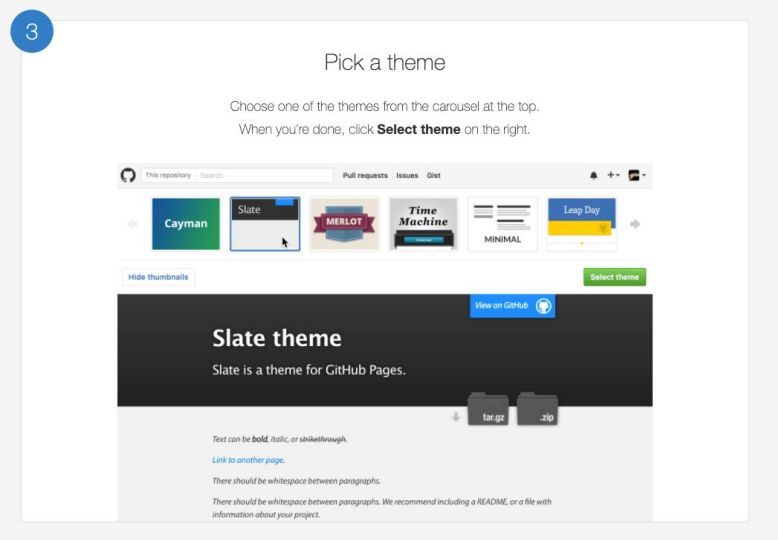
1

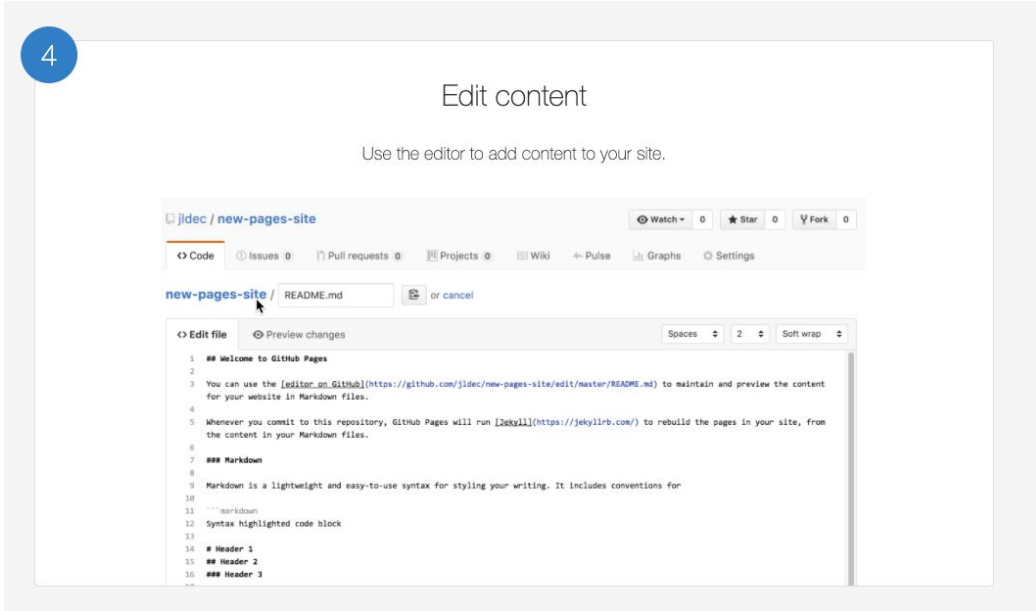


2



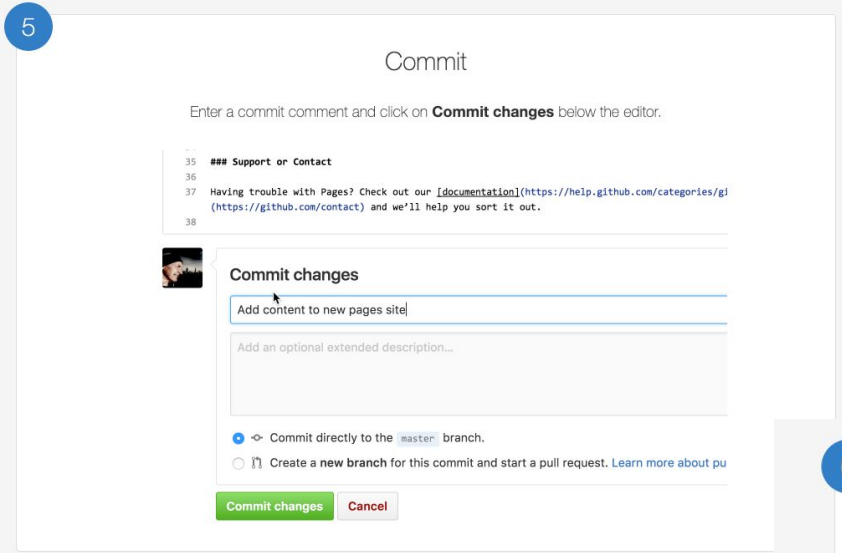
Deploying to GitHub Pages - Projects



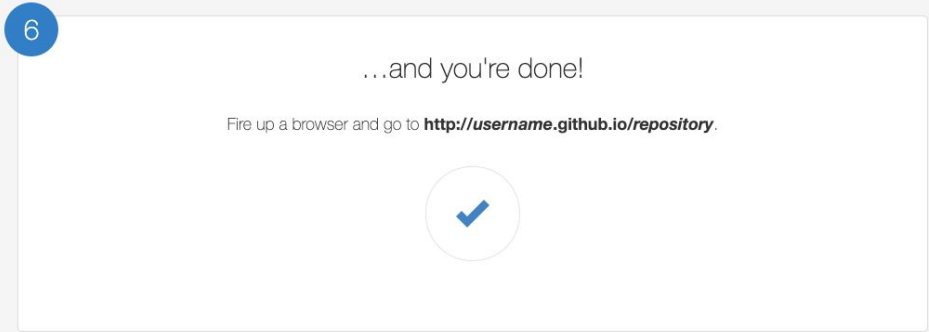


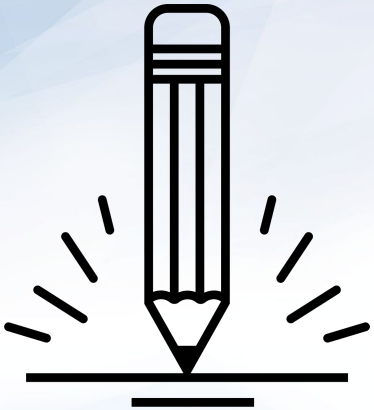
Deploying to GitHub Pages - Projects

5



6





Activity: Creating a Project Site

In this activity, you will be creating a web page to display and explain a data science project you already completed and will deploy the HTML to a github pages project page.

Suggested Time:
10 Minutes

Instructions:

Activity: Deploying Personal Bios to GitHub Pages

- Now that you know how to create webpages for specific projects, navigate into the repository of one of your previous activities or homework assignments and create a basic webpage for that project.
- Make a simple website that explains what the purpose of the activity was and shows off some of the work that you have done.
    - Create a header that will act as a title for the page
    - Create a few short paragraphs describing the project's purpose
    - Try to include at least one table or a picture of a graph that shows off some of the data you collected
    - Have some fun! Test your HTML skills by going above and beyond with your page!

- Once you have created your HTML, add, commit, and push your files up to GitHub Pages.
    - Make sure to navigate to your page in the web browser to ensure your website is live.

Deployment Guide:

Activity: Deploying Personal Bios to GitHub Pages

1. Create a new repository on your GitHub account. You can name this repository whatever you would like.
2. Once inside of the repository, create a new file and name it index.html
3. Add your HTML into this file, save it, and then navigate into your repository's Settings tab.
4. Scroll down to the GitHub Pages section and then, in the section labeled Source, select that you would like to use the master branch as your source.
5. Navigate to *username*.github.io/*repositoryname* and you will find that your new web page has gone live!

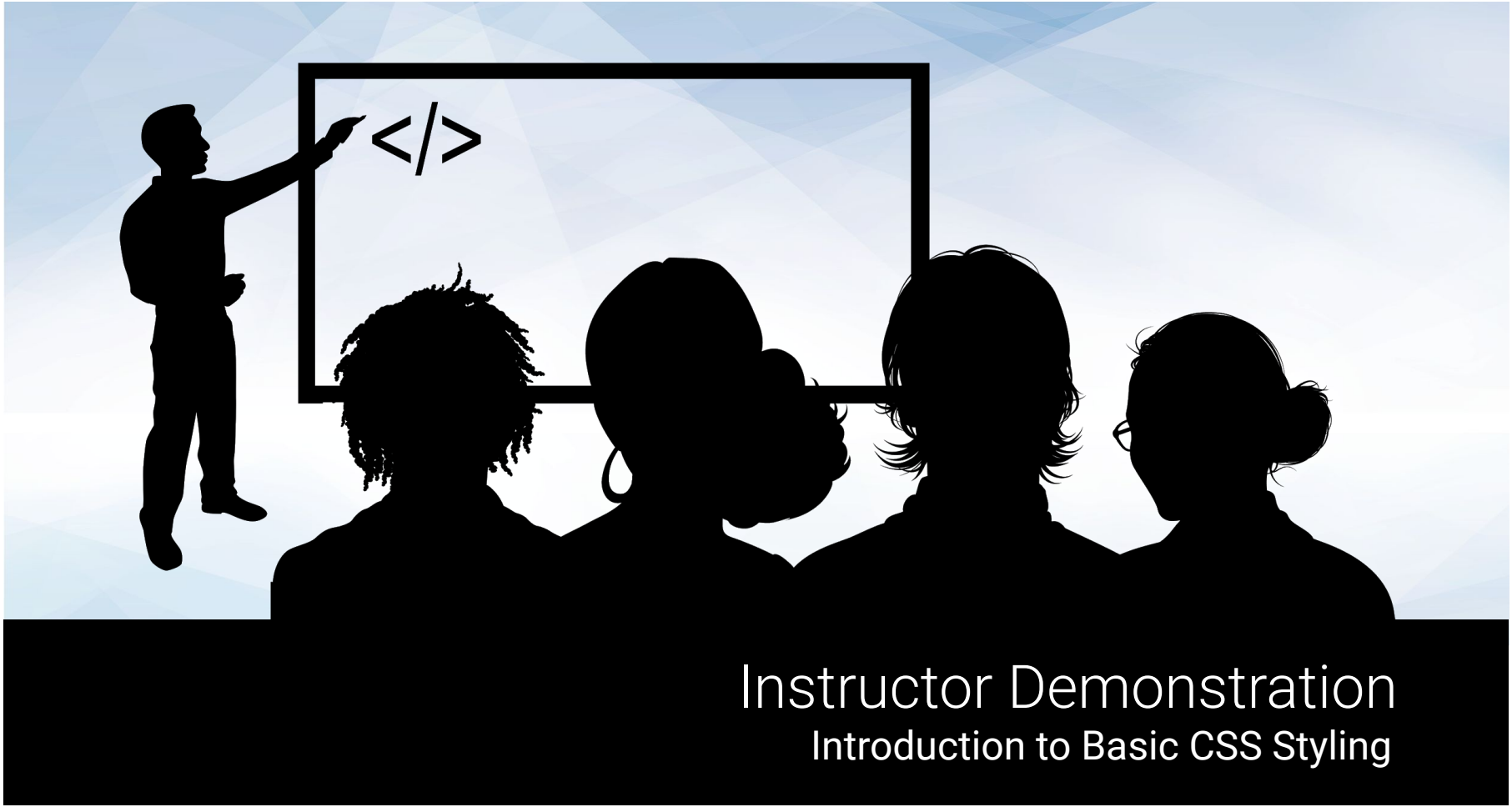
Bonuses:

- Try to create a navigation system which links from your personal website to this new project site and vice-versa.
- See if you can create something of a portfolio that can be used to show off all of the work you have done in the past.

Countdown timer

15:00

(with alarm)



Instructor Demonstration

Introduction to Basic CSS Styling

Introduction to Basic CSS Styling

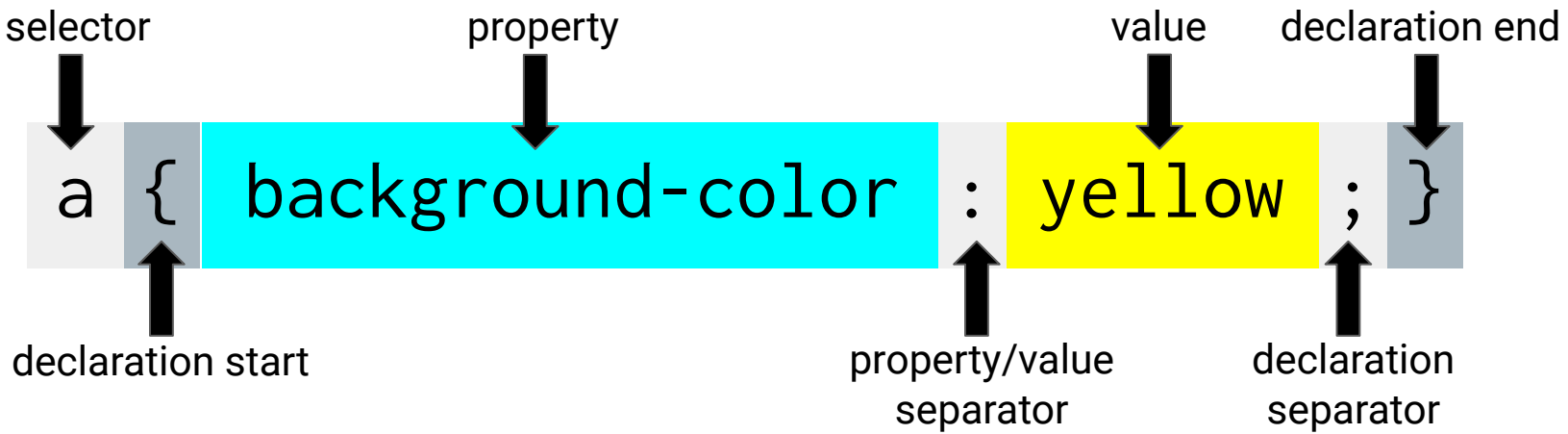
- CSS describes how and where elements should appear on the page. It defines things such as color, placement, fonts, sizes, and more.

Introduction to Basic CSS Styling

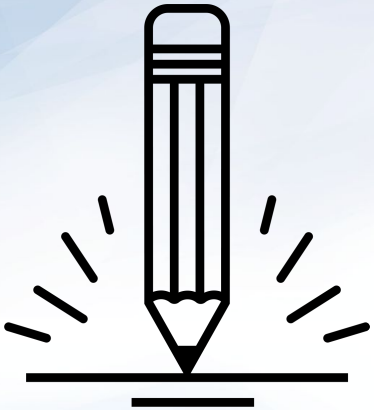
HTML Alone	HTML and CSS
Like writing papers in Notepad.	Like writing papers in Microsoft Word.
Used to write unformatted text (i.e, content only).	Used both to write the content *and* format it (color, font, alignment, layout, etc.).

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.

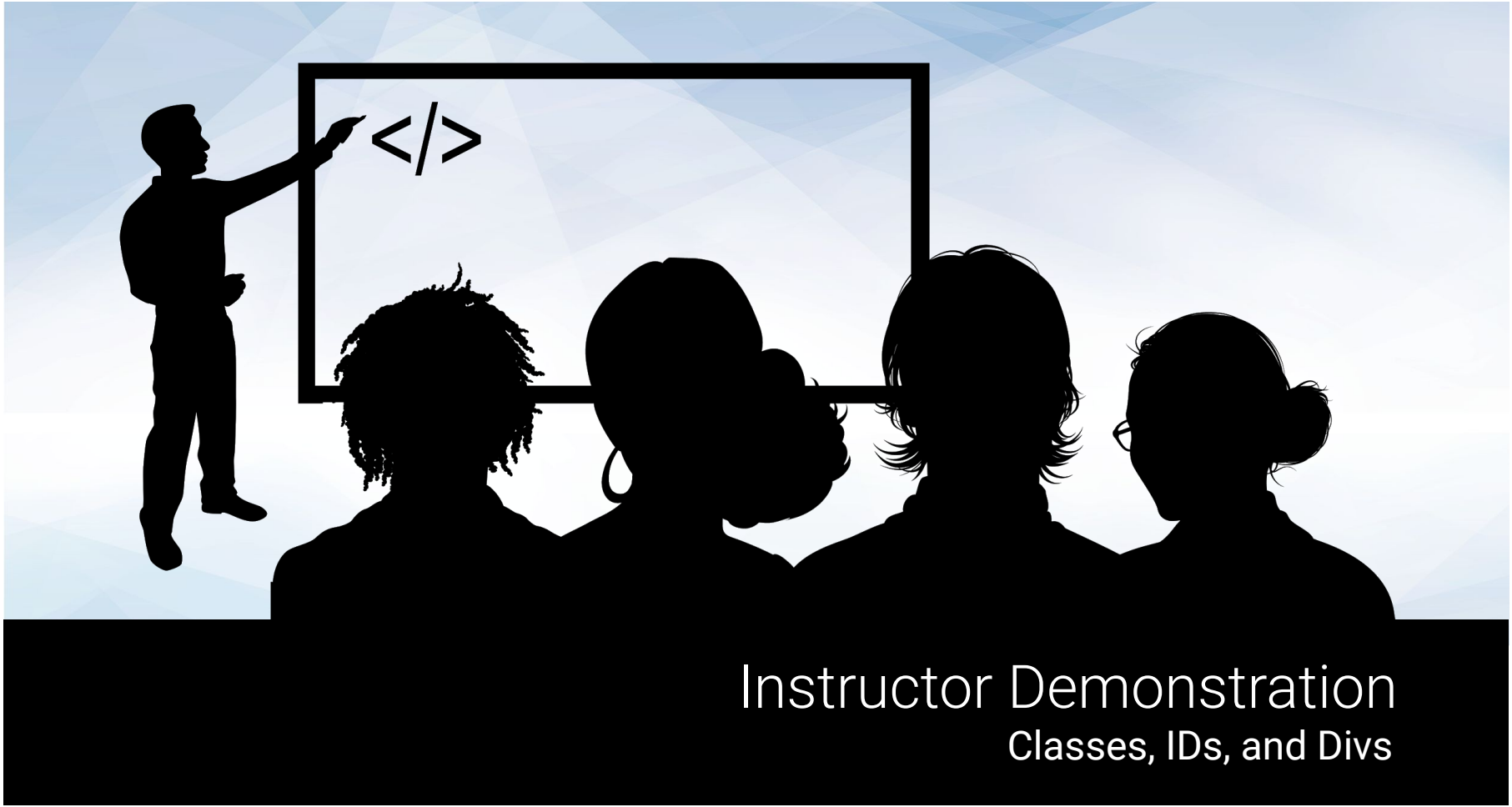
Suggested Time:
10 Minutes

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.



Instructor Demonstration

Classes, IDs, and Divs

Classes, IDs, and Divs

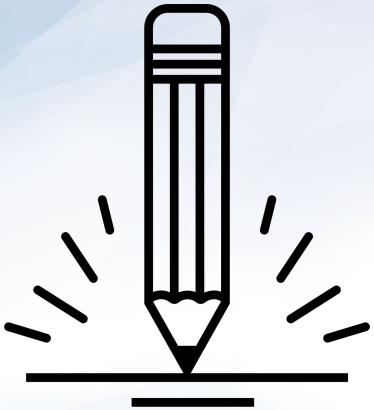
- To create an HTML class, place a `class=”className”`
- Classes can be reused as needed.
- Reference classes in stylesheets with ‘.’

```
<style type="text/css">

  .bigHeader{
    font-size: 40px;
  }

  .smallerHeader{
    font-size: 20px;
  }

  .tinyHeader{
    font-size: 10px;
  }

</style>
</head>

<body>

<h1 class="bigHeader">Awesome Header</h1>
<h1 class="smallerHeader">Smaller Awesome Header</h2>
<h1 class="tinyHeader">Even Smaller Header</h3>
```

Classes, IDs, and Divs

- To create an HTML id, place an `id="idName"` attribute within an HTML element.
- HTML id attributes must be unique!
- Id attributes are referenced in stylesheets with #

```
    #blueText {
      color: blue;
    }

    #redText {
      color: red;
    }

    #purpleText {
      color: purple;
    }

  </style>
</head>

<body>

  <h1 class="bigHeader" id="blueText">Awesome Header</h1>
  <h1 class="smallerHeader" id="redText">Smaller Awesome Header</h2>
  <h1 class="tinyHeader" id="purpleText">Even Smaller Header</h3>
```



Activity: Targeted CSS

In this activity, you will be given a very basic HTML file and will have to create an external CSS stylesheet changes the page's styling.

Suggested Time:
5 Minutes

Instructions:

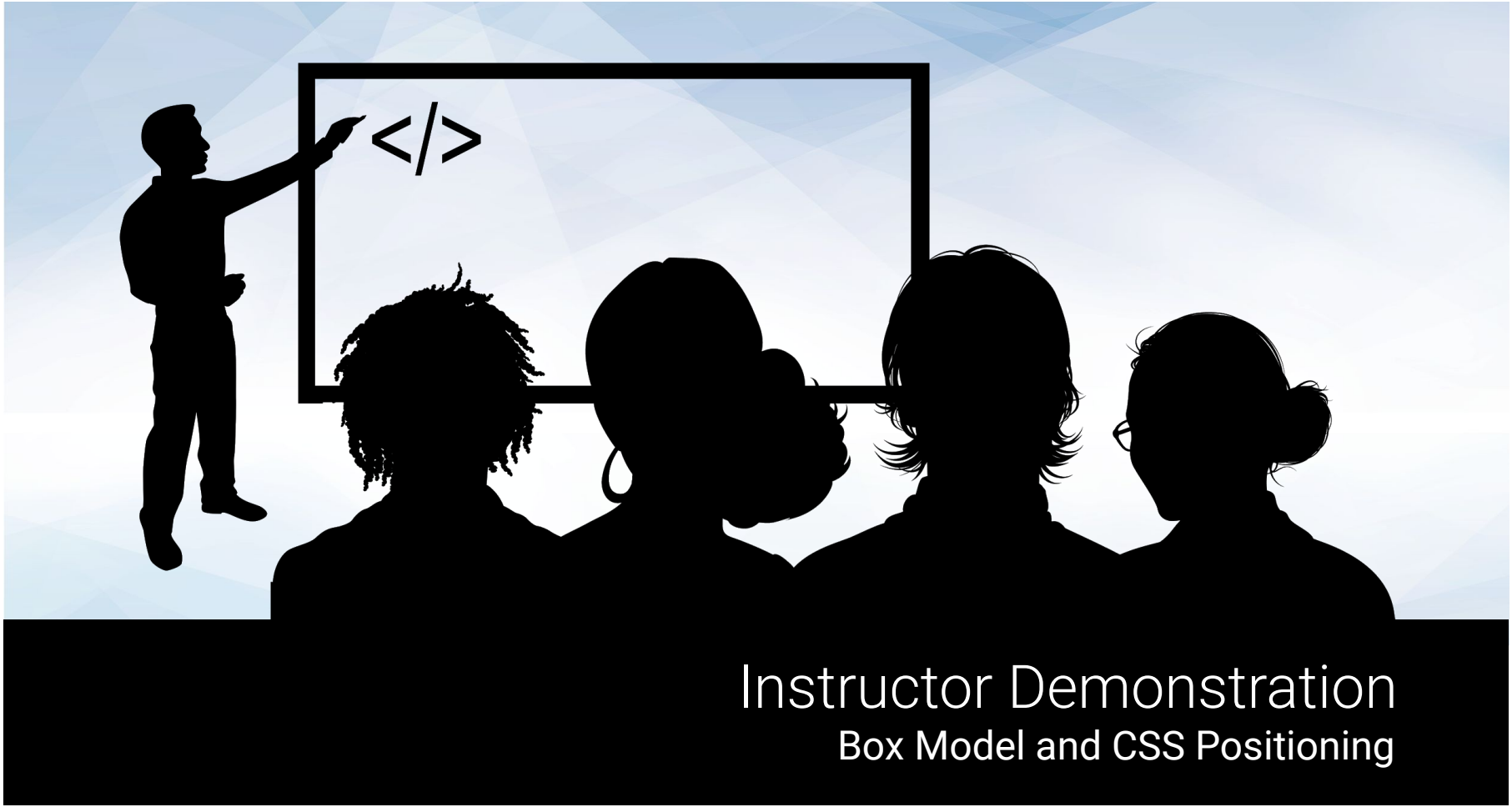
Activity: Dull Corp

- Open the Unsolved Starter.
- Using an external stylesheet, classes, and IDs accomplish the following...
  - Style the first element so that it has a color of blue
  - Style the second and third elements so that they have a color of purple
  - Style the fourth element so that it has a color of red

Bonus

- Using only additional classes and IDs, attempt to accomplish the following...
  - Set the background color of the first and last elements to black
  - Set the font-size of the first element to 40px
  - Set the font-weight of the second element to bold
  - Set the font-size of the third and last elements to 10px

Time’s Up! Let’s Review.



Instructor Demonstration

Box Model and CSS Positioning

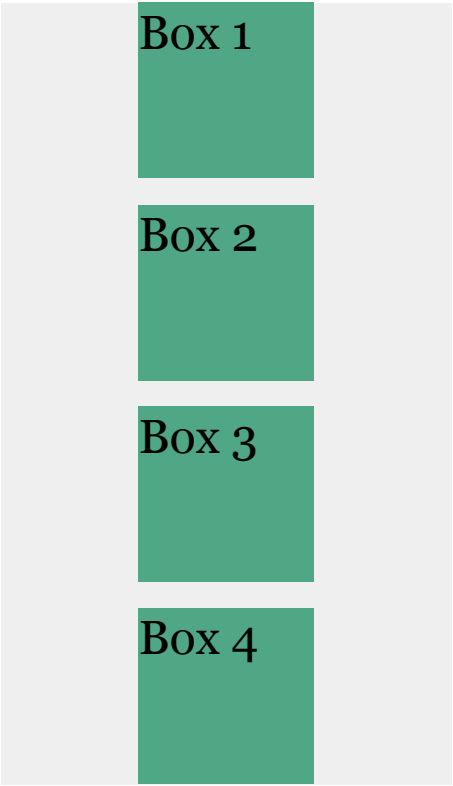
Boxes Upon Boxes

Box Model and CSS Positioning

- In CSS, every element 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, botton, left).

Position: Static
Box Model and CSS Positioning

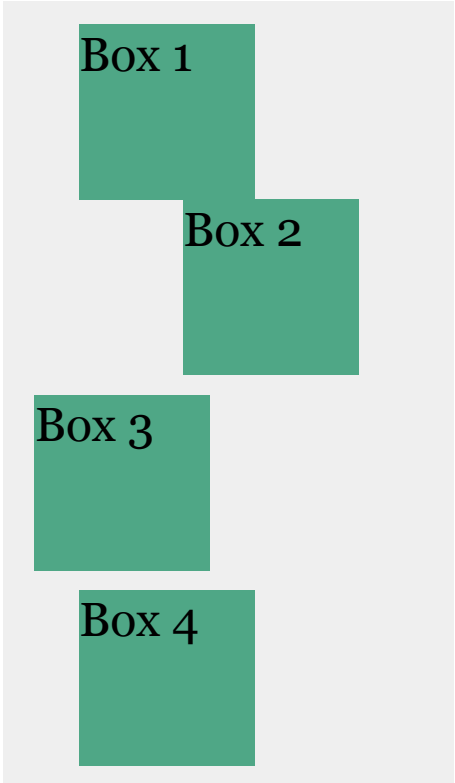
- Four boxes placed statically (default):



Position: Relative
Box Model and CSS Positioning

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

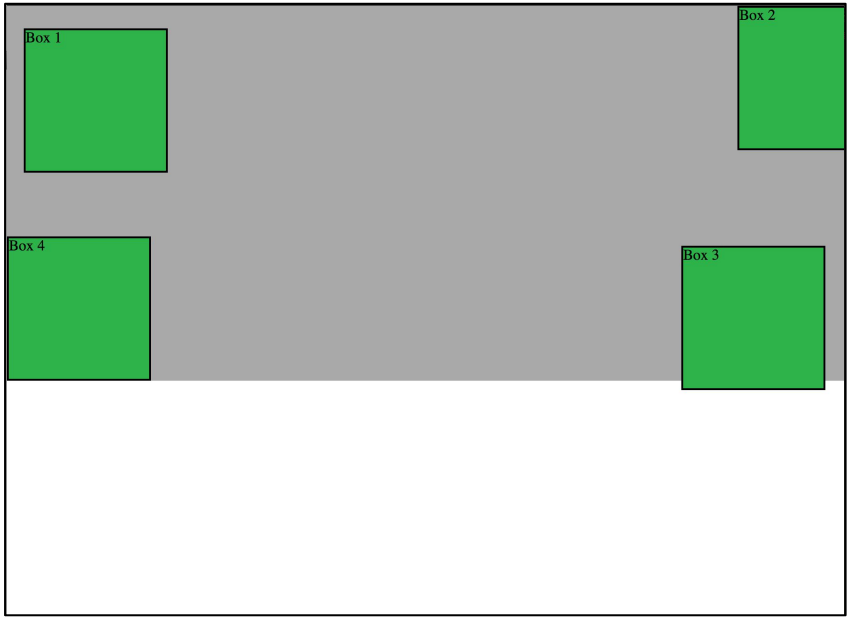
```css
.box {
  background: #2db34a;
  height: 80px;
  position: relative;
  width: 80px;
}
.box-1 {
  top: 20px;
}
.box-2 {
  left: 40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;
}
```



Position: Absolute
Box Model and CSS Positioning

- Positioned relative to nearest positioned ancestor:

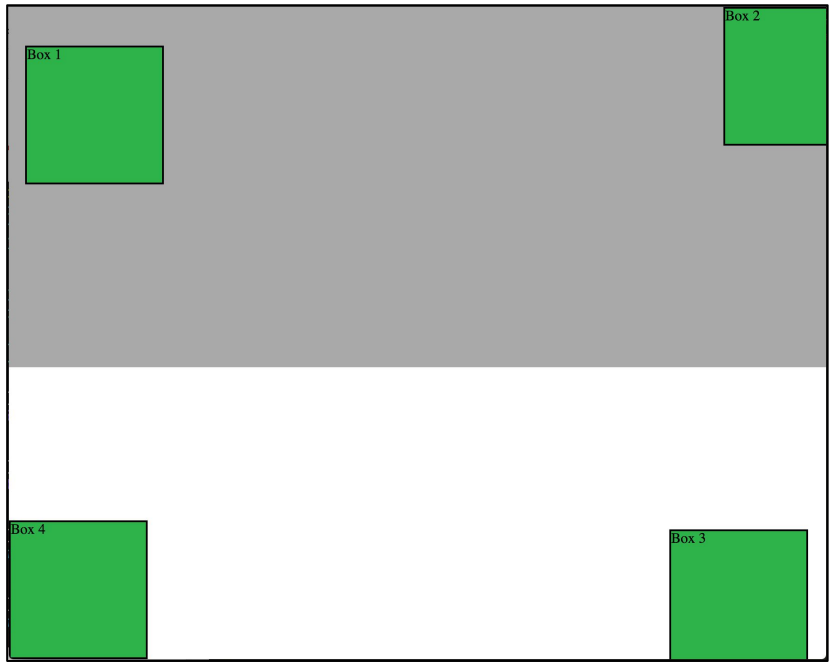
```
.box-set {
  height: 400px;
  background: darkgray;
  position: relative;
}
.box {
  position: absolute;
  height: 150px;
  width: 150px;
  background: #2db34a;
  border: 2px solid black;
}
.box-1 {
  top: 6%;
  left: 2%;
}
.box-2 {
  top: 0;
  right: -40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;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



Position: Fixed
Box Model and CSS Positioning

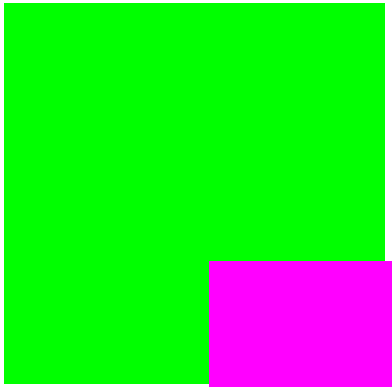
- Position with exact coordinates in the browser window:

```css
.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;
}
```

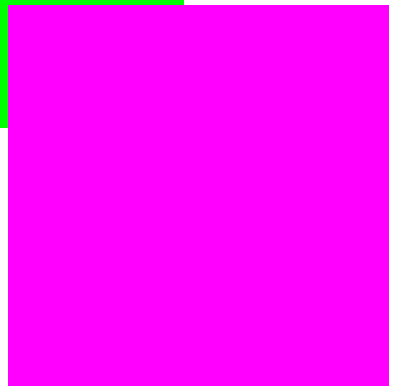


Box Model and CSS Positioning

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



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

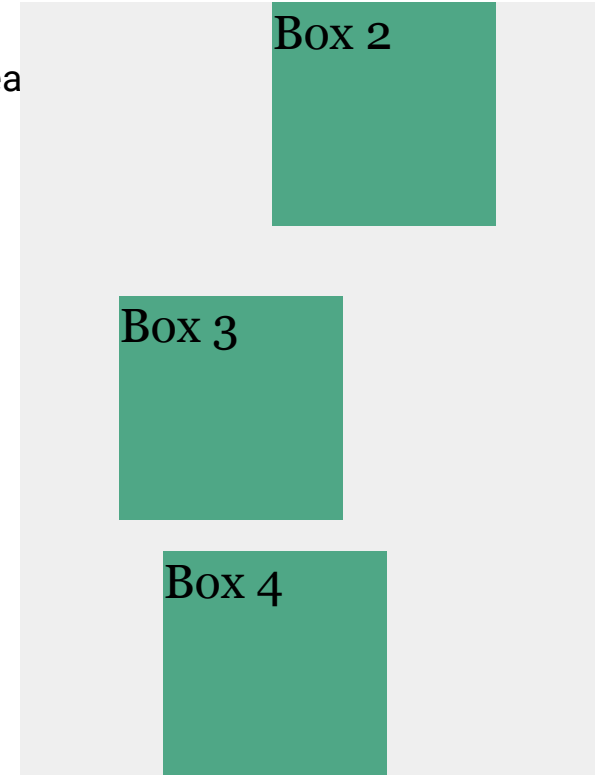


```
position: absolute;
z-index:2;
```

Box Model and CSS Positioning

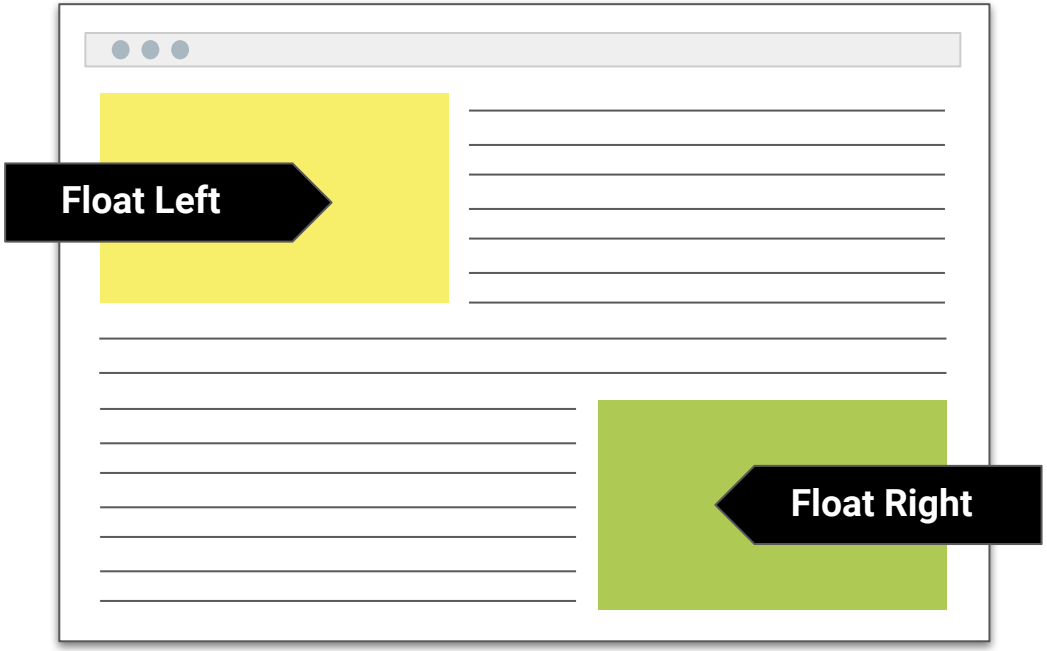
- Display: None allows you to hide elements from view.
- This will become useful in later sections, we'll hide and revea specific HTML elements of our choosing.

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



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**.



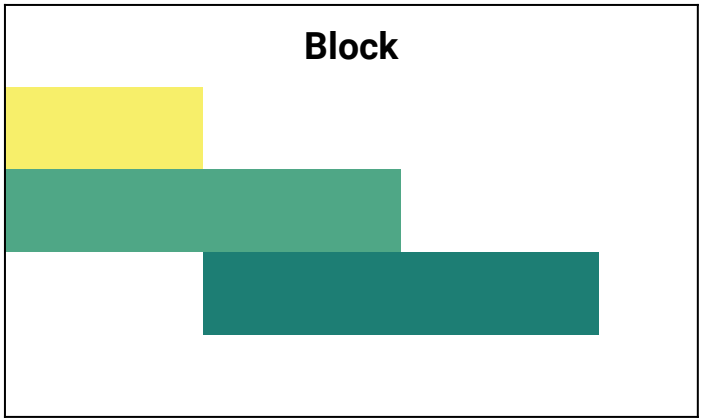
Box Model and CSS Positioning

- 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.

Box Model and CSS Positioning

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

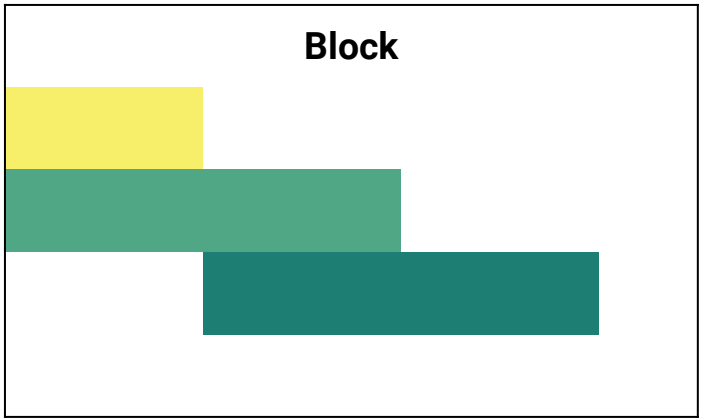
A block element will take up an entire line of space—unless you intervene with CSS properties.

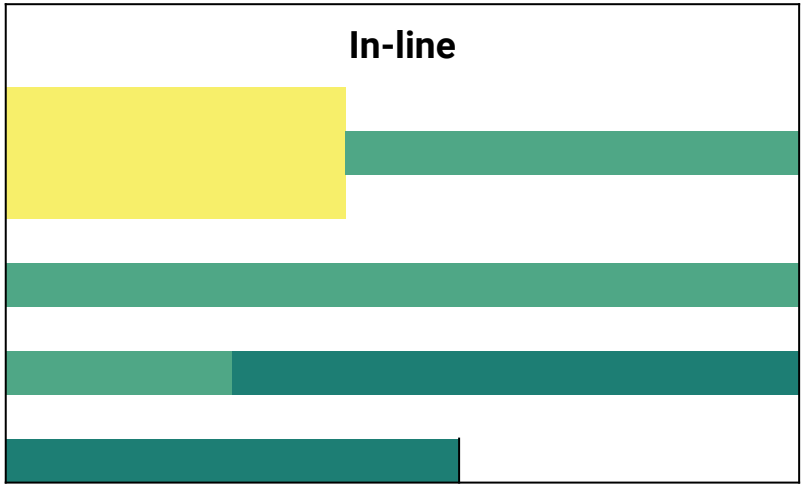


Box Model and CSS Positioning

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

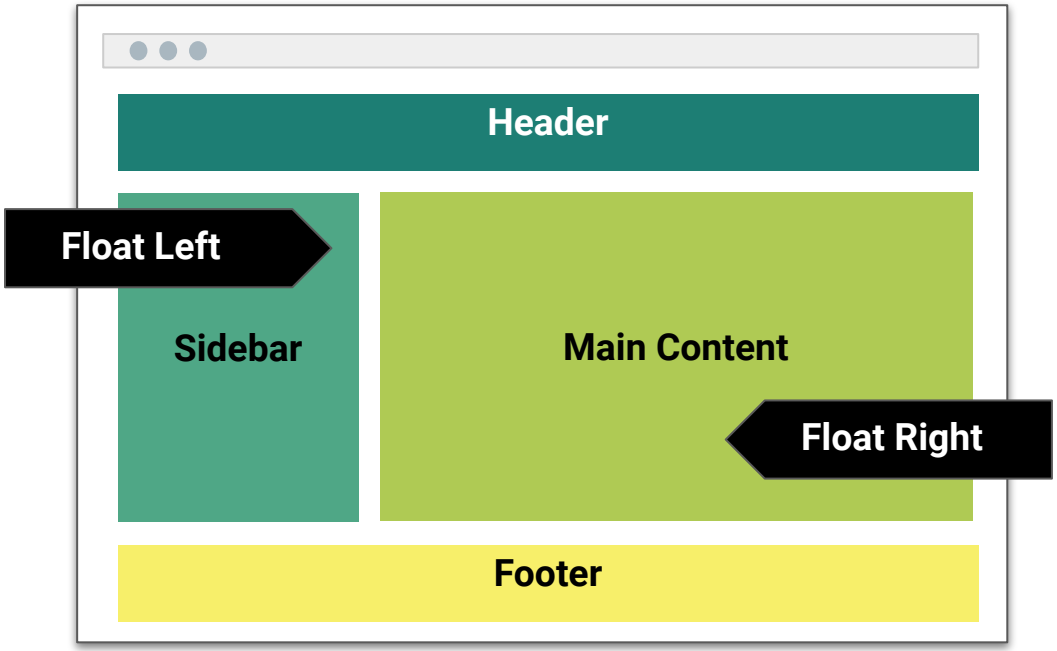
By using **float CSS properties**, we can command our website to display multiple HTML elements adjacently.





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.

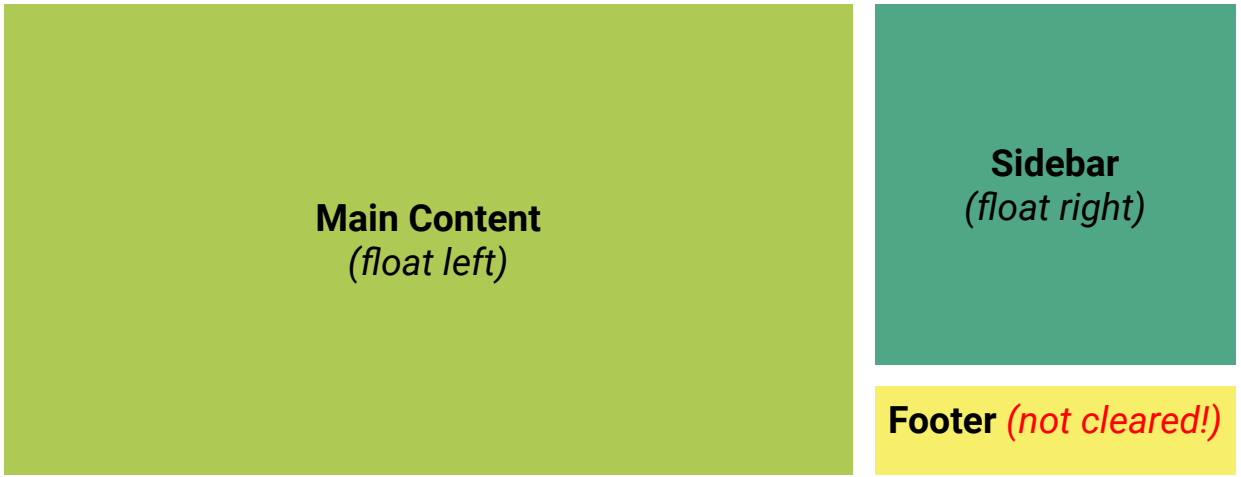


CSS

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

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.



Box Model and CSS Positioning

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

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

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.

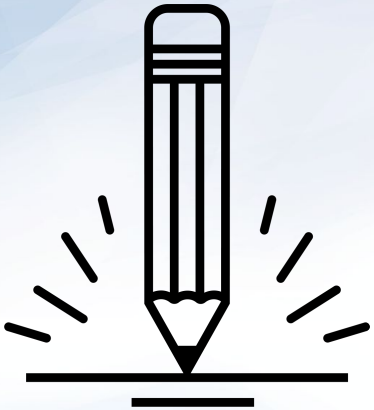
```
.clearfix::after {
    content: "";
    display: block;
    clear: both;
}
```

Box Model and CSS Positioning

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

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

Much better!



Activity: Aimed Positioning

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

Suggested Time:
10 Minutes

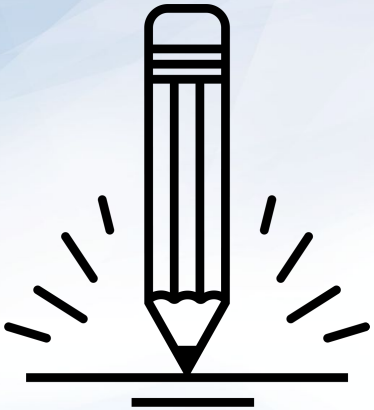
Instructions:

Activity: Aimed Positioning

- Open the Unsolved Starter.
- Using CSS, position the five headers in the starter code in their described locations.
  - Try to accomplish this using the box model
  - Try to accomplish this using absolute positioning
- Whenever you complete a step, save a unique version of your stylesheet and then create a new stylesheet for the next step.
- **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.



Activity: Student Bio

In this activity, you will be given a Bio Page HTML skeleton and you will style it with CSS so the HTML resembles the image provided in the Unsolved folder.

Suggested Time:
20 Minutes

Instructions:

Activity: Student Bio

- Open the Unsolved folder and use the files inside as a starting point.
- Style the HTML file with the following:
  - Add a class called "container" on the div tag.
  - Add an id called "main-bio" for the first section tag.
  - Add an id called "contact-info" for the second section tag.
  - Add an id called "bio-image" for the bio image.
  - Style specs:
    i. `body`
       1. The background color is `#efeee7`.
       2. The font used `"Georgia", Times New Roman, Times, serif;`.
       3. The font color is `#333333`.
       4. Be sure to zero out the body margins and padding so the page is flush to the top of the page.

    ii. `header`
       1. The background-color is `#333333`.

Instructions:

Activity: Student Bio

- h1
  - The font color is #eee.
  - The font size is 28px.
  - Look at the example on the screen, and eyeball the padding and/or margins and positioning of the text.

Instructions:

Activity: Student Bio

- `h2`
  - The font size is 24px
- Make the container have a width of 1024 pixels, and center it. You do this using margin: 0 auto;.
- Make `#main-bio`, `#contact-info`, `#bio-image` all `float: left`.
- Make the `#bio-image` have a width of 200 pixels.
- Be sure to include `alt` text in all images
- `#main-bio` should have a width of `70%`.
- Add margins to the image so there is distance between it and the bio text.
- `#content-info` should have a width of `30%`.
  - Adjust the line height so it is 1.5 times the size of the font.
- Make the link color `#d21034`.
- `#figure` should be given relative positioning
- `#bio-name` should use absolute positioning and be placed at the bottom-center of your `#bio-image`

Instructions:

Activity: Student Bio

- Make sure to replace the content inside of the HTML document with your own name, github link, etc.
- Stage, commit, and push this new file to GitHub Pages.
- Bonus:
  - Using fixed positioning, attempt to create a footer on your page that will stay positioned at the bottom of the screen even when scrolling.
  - Underneath the main content of your page, add in a "portfolio" section. Try to make it so that the projects are set up in a kind of grid with two projects per row. Each of the projects should be named with a header, include a relevant image, and have a short paragraph describing the project.

Time’s Up! Let’s Review.

The End