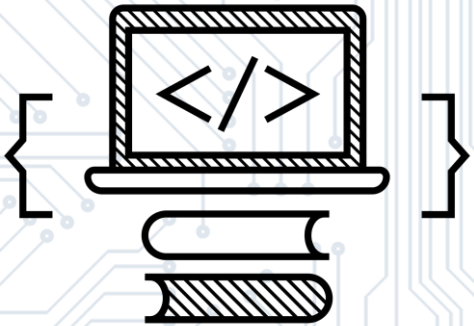


Heroes of CSS

Web Development Boot Camp

Lesson 1.3

© 2019 Trilogy Education Services, Inc.

Admin Items

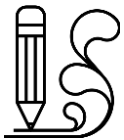
Homework #1

01 Due next Saturday – 2/1/2012

02 Homework due dates can be found in BootCampSpot. Homework should also be submitted via BootCampSpot.

03 The Homework activity can be found in the class repository on GitHub

04 Choose the easier homework if you are not quite comfortable with the more challenging assignment.



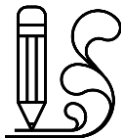
Homework Assignment Tips

01 Really work hard on this assignment! This assignment introduces you to fundamental concepts that we'll build upon during the rest of the course.

02 Review in-class material, especially activities.

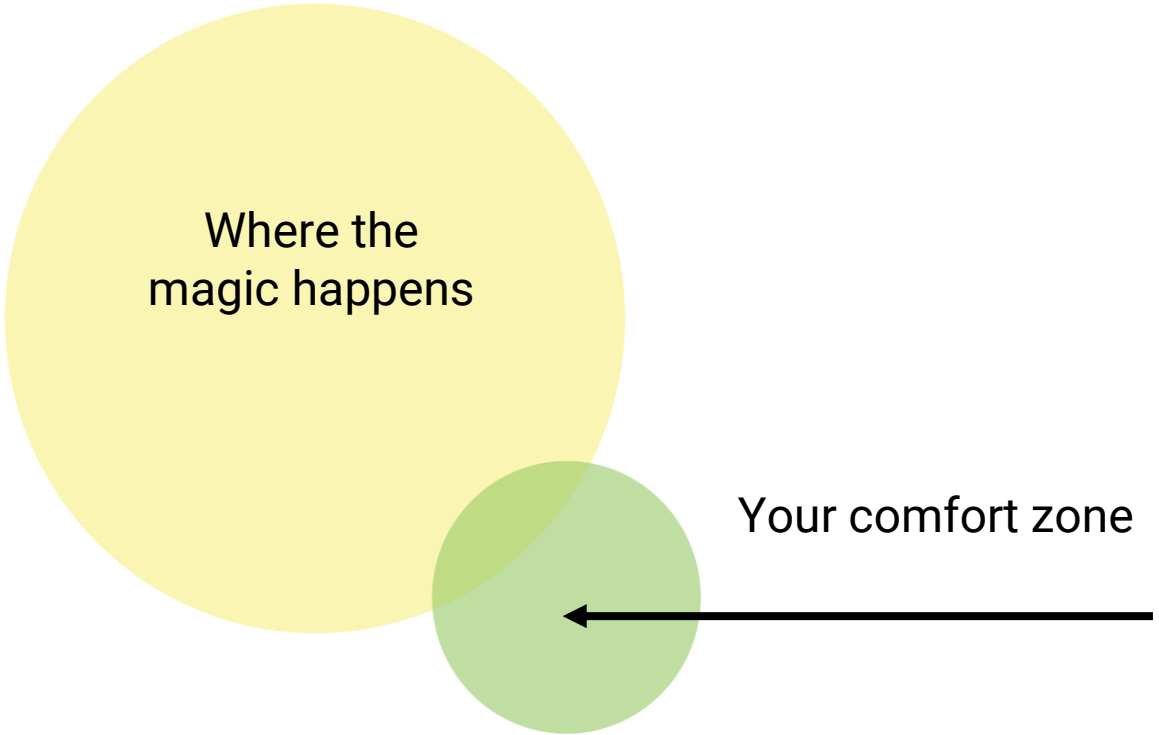
03 Work with your peers! It's much better than screaming at your computer alone.

04 Ask questions on Slack! Your peers, TAs, and instructors are all here to help.



Most Important of All

Just submit *something* (even if it seems pretty crummy)!



Warning!

Brace Yourselves

Today is going to be a bit tough. But trust us—it will all look easy a few weeks from now!

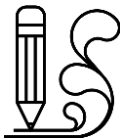
Don’t expect to understand everything at once. Today is all about getting immersed.

Class Objectives

01 To become comfortable working with HTML relative paths

02 To engage in a deep dive of HTML/CSS for layout design and element positioning

03 To complete the Bash Checkpoint.



Bash Recap

What is the command to create a file?

What is the command you use to navigate to different directories?

What is the command to create a directory?

CSS Recap

Critical Question:
What Is CSS?

Can you make a website with just HTML?

HTML and CSS Definitions

HTML: Hypertext Markup Language (Content)

CSS: Cascading Style Sheets (Appearance)

HTML/CSS are the “languages of the web.” Together they define both the content and the aesthetics of a webpage, handling everything from the layouts, colors, fonts, and content placement. (JavaScript is the language that deals with logic, animation, etc.)

HTML/CSS Analogy

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.).

Basic HTML Page (No CSS)

Awesome Header

Smaller Awesome Header

Even Smaller Awesome Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Menu Links

- **Google**
- **Facebook**
- **Twitter**

Basic HTML Page (No CSS)

Awesome Header

Smaller Awesome Header

Even Smaller Awesome Header

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Menu Links

- **Google**
- **Facebook**
- **Twitter**

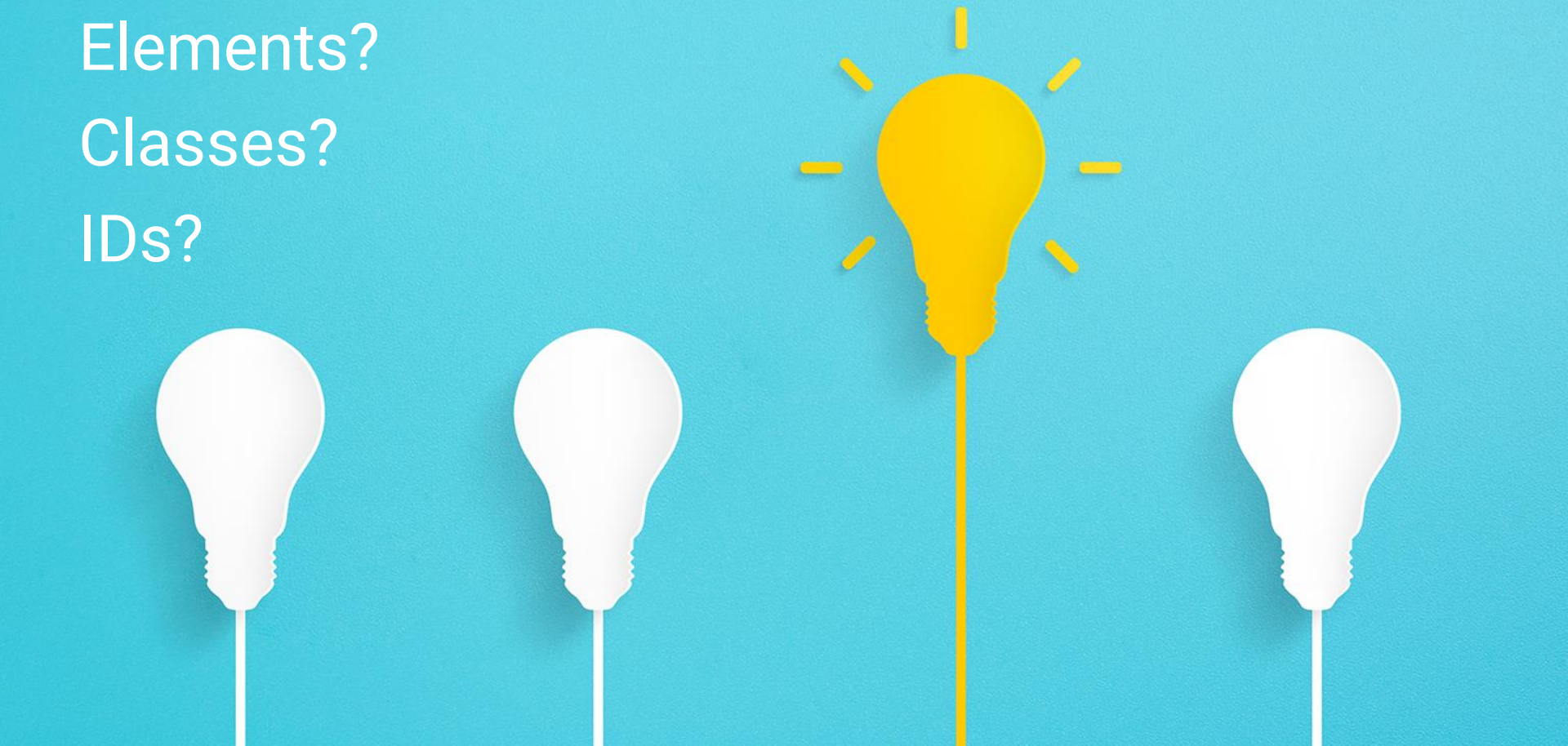
Boring

Critical Question: How Do We Style HTML?

Elements?

Classes?

IDs?

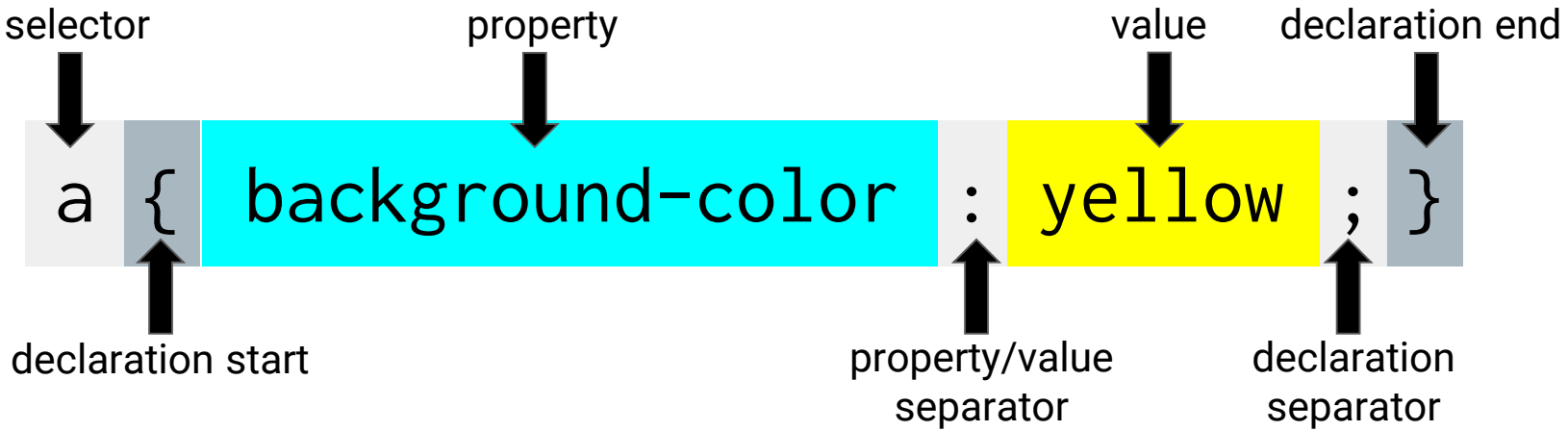


CSS Syntax

CSS works by hooking onto **selectors** added into HTML using **classes** and **identifiers**.

Classes use **.classname**, IDs use **#idname**, and elements use just their name.

Once hooked, we apply **styles** to those HTML elements using CSS.



Selectors

Element selector	Element name
	(p, a, div, span, etc.)
		Applies to all <p> elements
		
		`p {`
		`  background-color: blue;`
		`}`

Class selector	Period (.) + variable name
	(.myDiv, .phoneNumber, etc.)
		Applies to all elements with class="classItem"
		
		`.classItem {`
		`  background-color: orange;`
		`}`

ID selector	Hash (#) + variable name
	(#myDiv, #phoneNumber)
		Applies to all elements with id="idItem"
		
		`#idItem {`
		`  background-color: green;`
		`}`

CSS Selectors

```
p {
    background-color: blue;
}

.classItem {
    background-color: orange;
}

#idItem {
    background-color: green;
}
```

```
<p>
   A paragraph with a blue background.
</p>

<div class="classItem">
   A div with an orange background.
</div>

<div id="idItem">
   A div with a green background.
</div>
```

A paragraph with a blue background.

A div with an orange background.

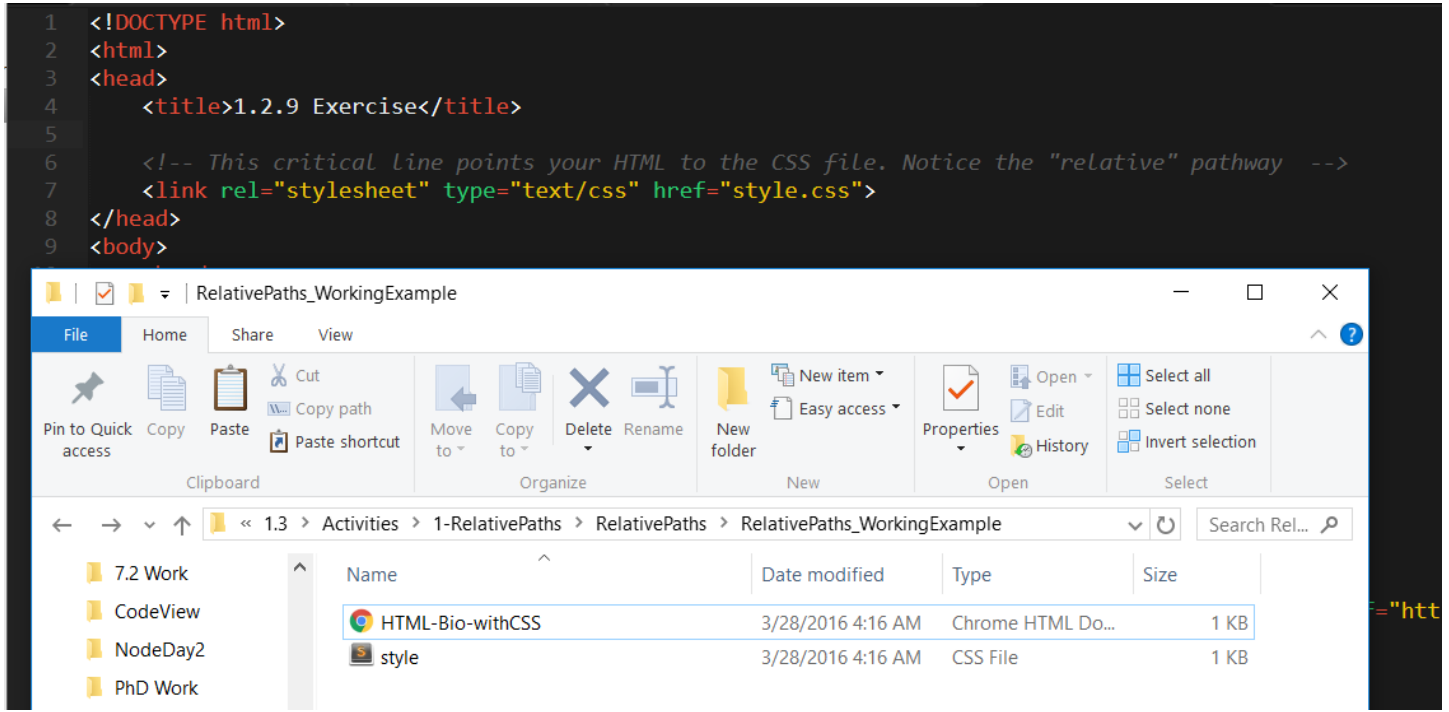
A div with a green background.

Questions?

Relative File Paths

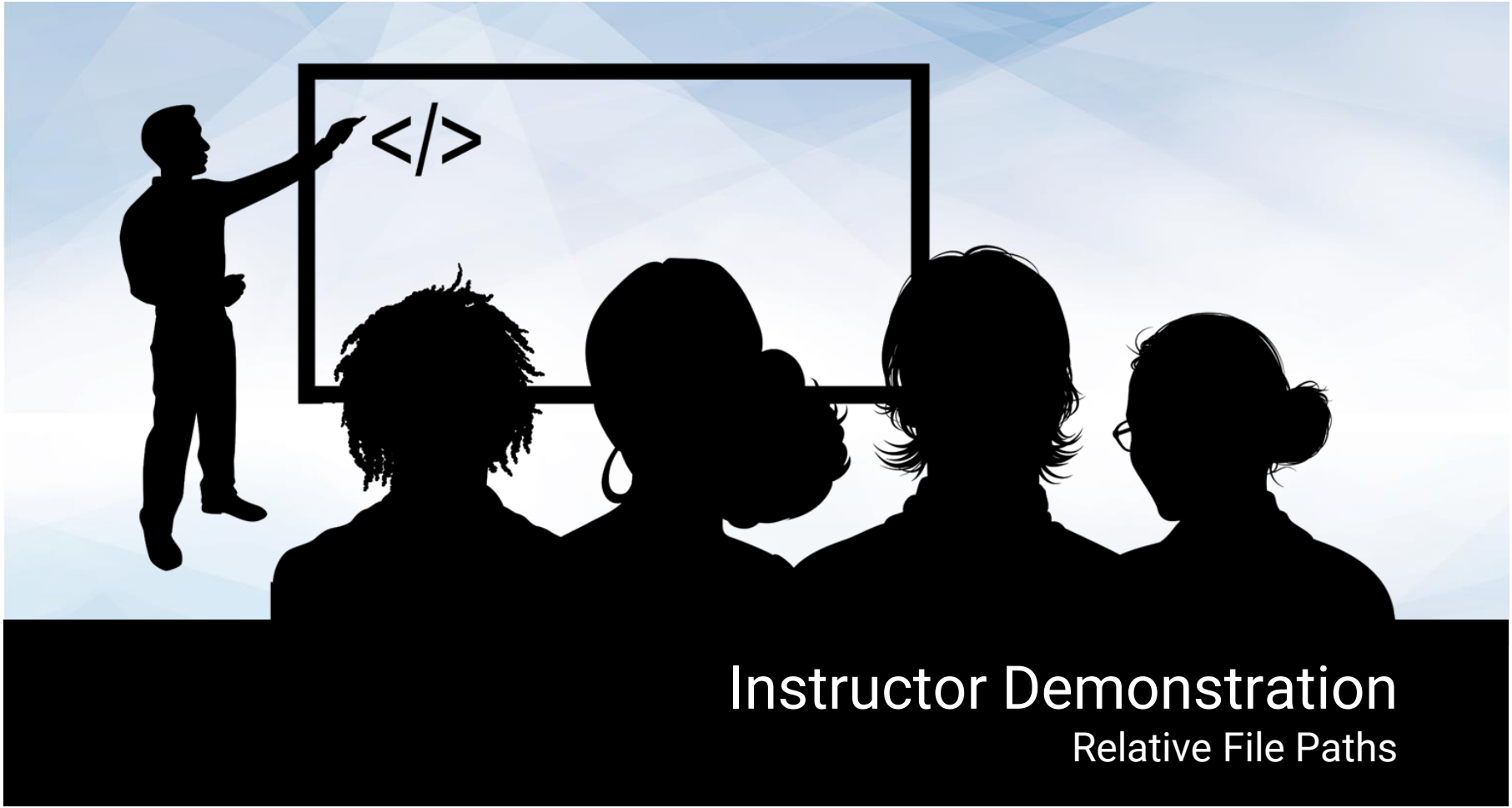
Relative File Paths

Relative file paths connect us with other files in our working directory.
In this case, style.css is in the same folder as our HTML document.



Benefits of Using Style Sheets

- Can be reused
- Greater Accessibility
- Easier to maintain and update
- Faster download times
- SEO (search engine optimization) benefits



Instructor Demonstration

Relative File Paths

Absolutely No Absolute Paths

Always use relative file paths!

- If you deploy websites without them, **all of your links will fail.**
- The same will happen if you move your project from one folder to another.
- Remember, there is no such thing as a C: drive on the internet.

VERY, VERY BAD

```
<!-- BAD!!!! -->
<link rel="stylesheet" href="D:/trilogy/FullStack-Lesson-Plans/02-lesson-plans/01-
html-css-three-days/1-Class-Content/1.3/Activities/1-RelativePaths/RelativePaths/
RelativePaths_WorkingExample/style.css">
```

Absolute vs Relative Paths

Absolute Path	**Relative Paths**
Absolute path is the exact address of the file in the filesystem, starting from the root.	Relative path writes it with respect to another point in the file system
Drive + {folder(s)} + {file} C:\Projects\Project1\Images	No drive or machine name

Activity:
Relative File Paths

Suggested Time:
12 minutes

Activity: Relative File Paths

01 Unzip the folder sent to you via Slack.

02 Edit the HTML files in all of the RelativePaths folders. Write relative paths that link the HTML documents with CSS stylesheets.

HINT: Check out the RelativePaths_WorkingExample folder.

Suggested Time: 12 minutes

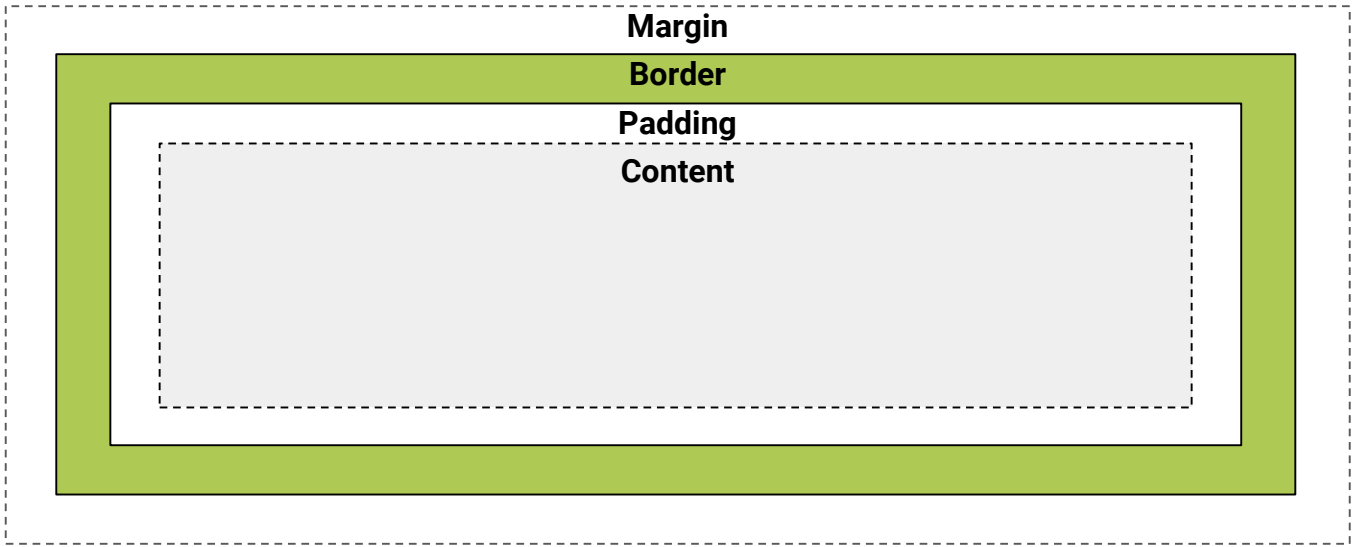
Box Model

Boxes Upon Boxes

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, bottom, left)



Activity: Box Model

```
#box {

    background-color: #1E5792;
    width: 400px;
    height: 440px;
    margin: 10px 30px 20px 50px;
    color: #fff;
    padding: 25px 10px 30px 20px;
    border-style: solid;
    border-width: 22px;
    border-color: #113152;

}
```

How wide is the blue #box?

How tall is the blue #box?

Total element width = content width + left padding + right padding + left border + right border + left margin + right margin

Total element height = content height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Suggested Time: 10 Minutes

Activity: Boxes Upon Boxes

Answer:
Width: 474 px (no margin), 554 px (with margin)
Height: 539 px (no margin), 569 px (with margin)

```
#box {

    background-color: #1E5792;
    width: 400px;
    height: 440px;
    margin: 10px 30px 20px 50px;
    color: #fff;
    padding: 25px 10px 30px 20px;
    border-style: solid;
    border-width: 22px;
    border-color: #113152;

}
```

How wide is the blue #box?

How tall is the blue #box?

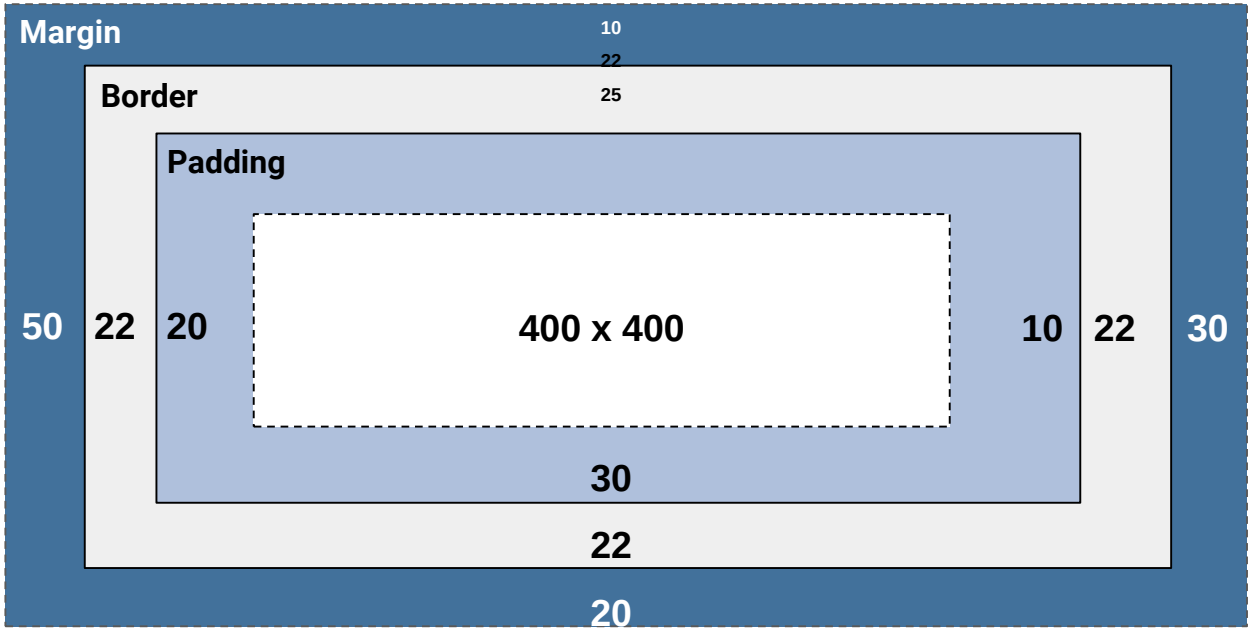
Total element width = content width + left padding + right padding + left border + right border + left margin + right margin

Total element height = content height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Suggested Time: 10 Minutes

Activity: Box Model

Answer:
Width: 474 px (no margin), 554 px (with margin)
Height: 539 px (no margin), 569 px (with margin)



Suggested Time: 10 Minutes

We Be Floatin’

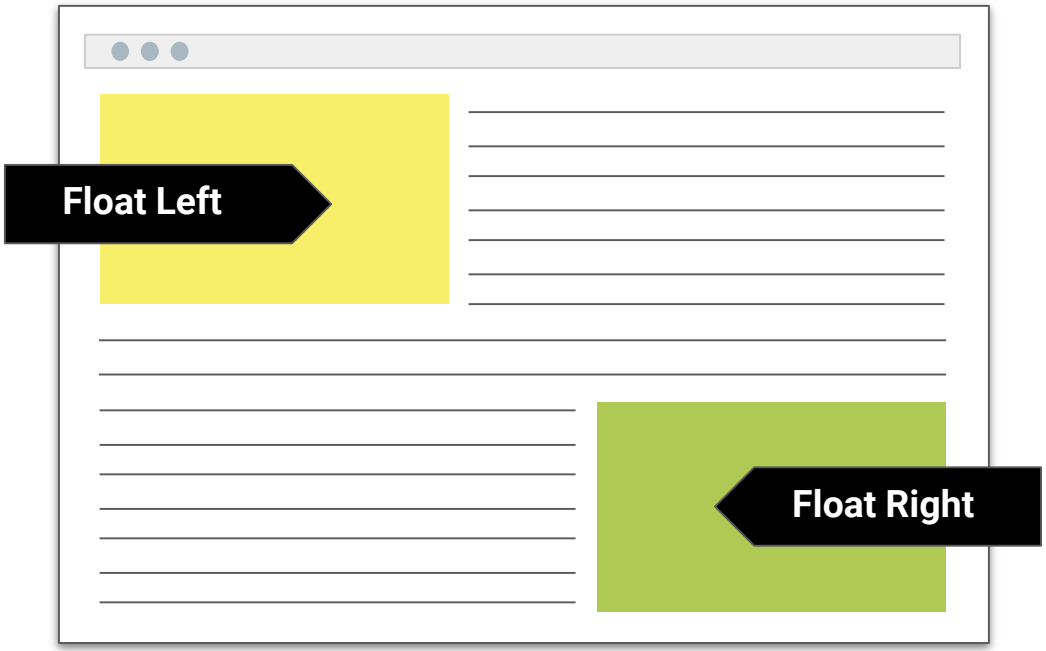
Warning

The next few topics are fairly tricky but also *very important*. Time to channel that inner genius!

The Concept of Flow

By default, every HTML element displayed in the browser is governed by a concept called **flow**.

This means that HTML elements force adjacent elements to flow around them.



Analogy: Flow and MS Word

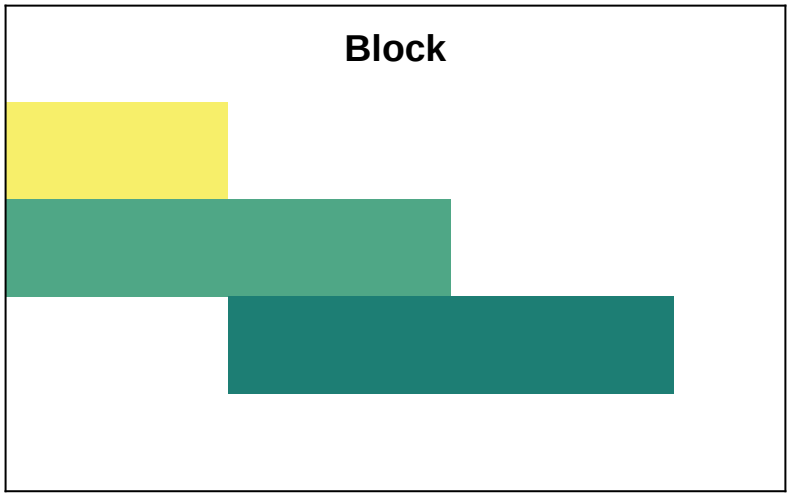
The concept of flow is similar to wrap-text in Microsoft Word.

Just as in MS Word, with CSS you can position images to be in-line with text, on top of text, and so on.

Block Elements

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

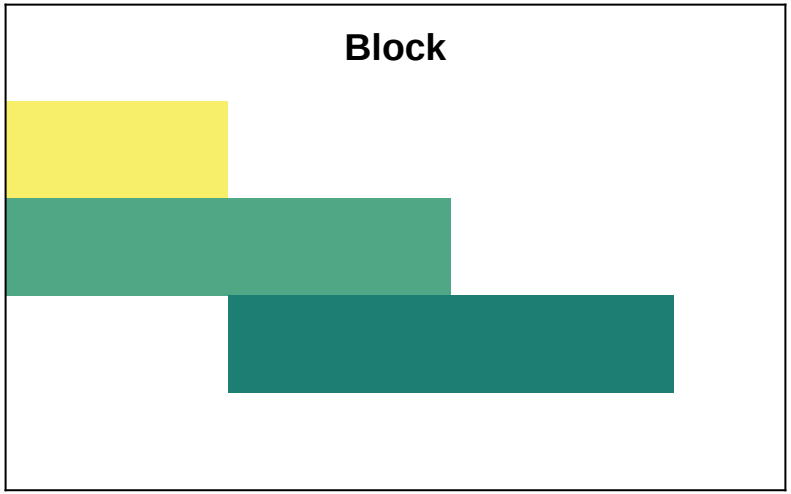
A block element will take up an entire line of space—unless you intervene with CSS properties.

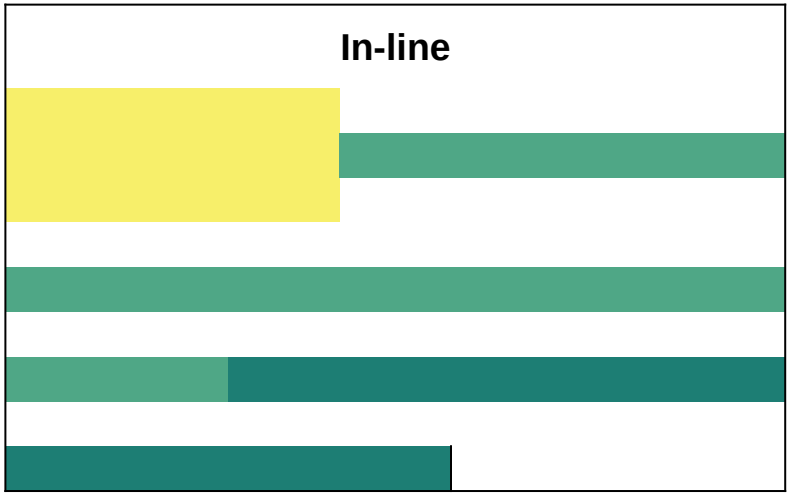


Block Elements vs. In-line Elements

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

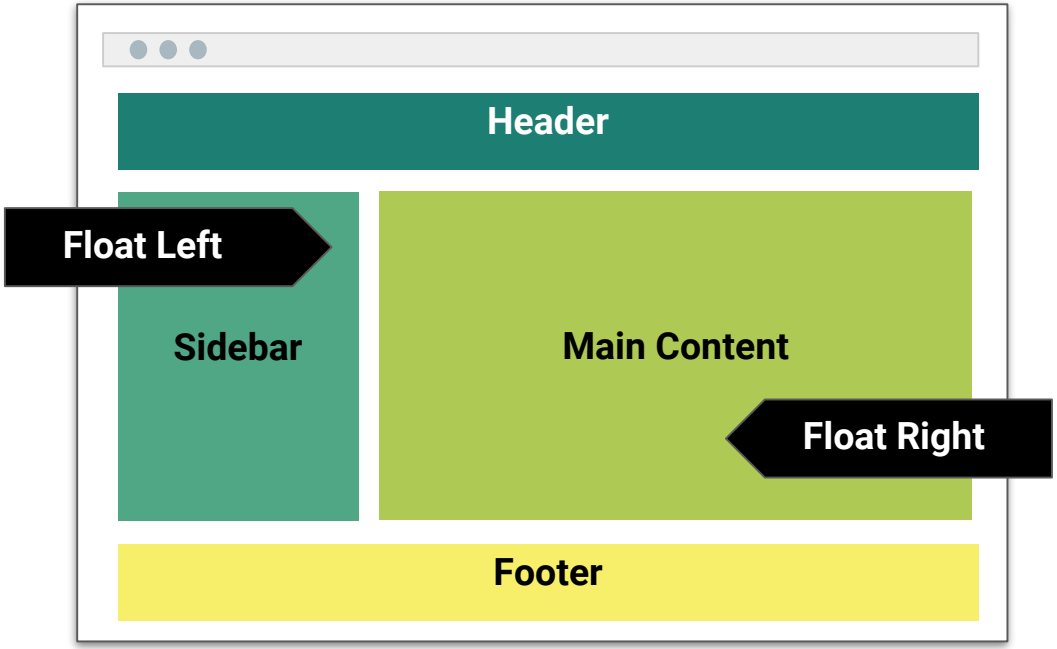
By using **float CSS properties**, we can command our website to display multiple HTML elements adjacently.





Floats

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



CSS

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

Float Activity

Suggested Time: 30 Minutes

Clearing the Float

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

Clearfix Hack

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!

Clearfix Hack

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

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

Much better!

Clearfix Hack

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

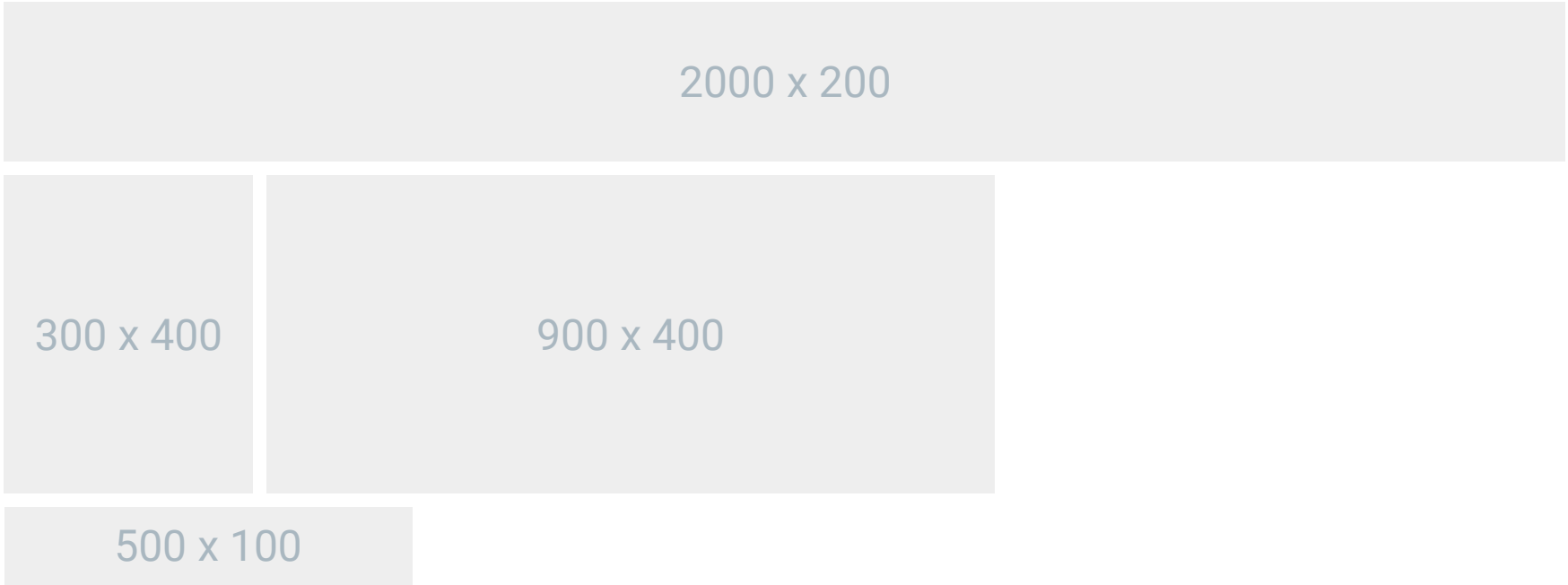
This will add an HTML element, hidden from view, after the content of the `.clearfix` element. This clears the float.

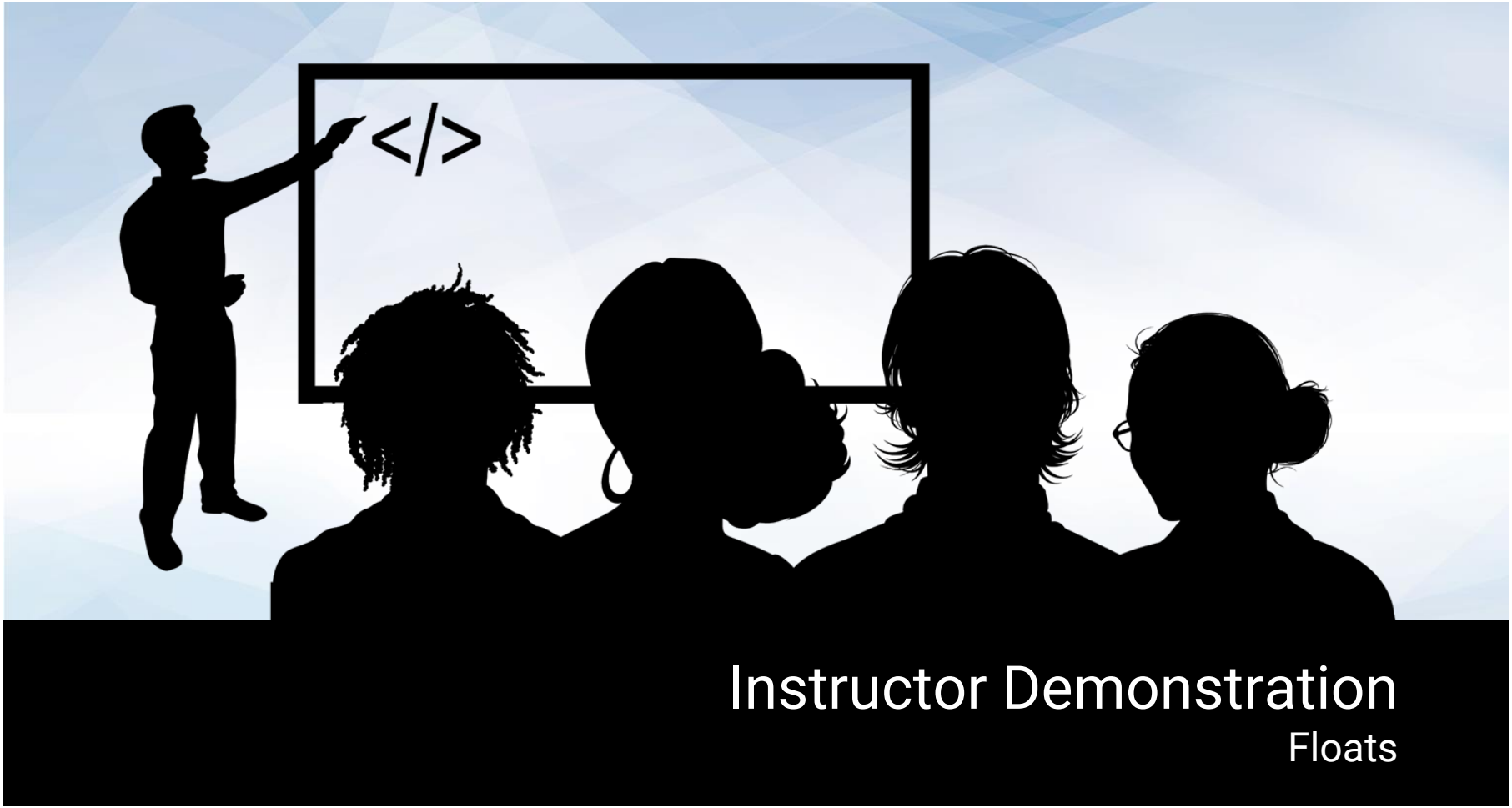
```
.clearfix::after {
   content: "";
      display: block;
   clear: both;
}
```

Quick Demo

Dummy Webpage!
OMG. The Greatest Website of All Time!!!
Item One
Item Two
Item Three
Item Four
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
Footer + Copyright

Quick Demo



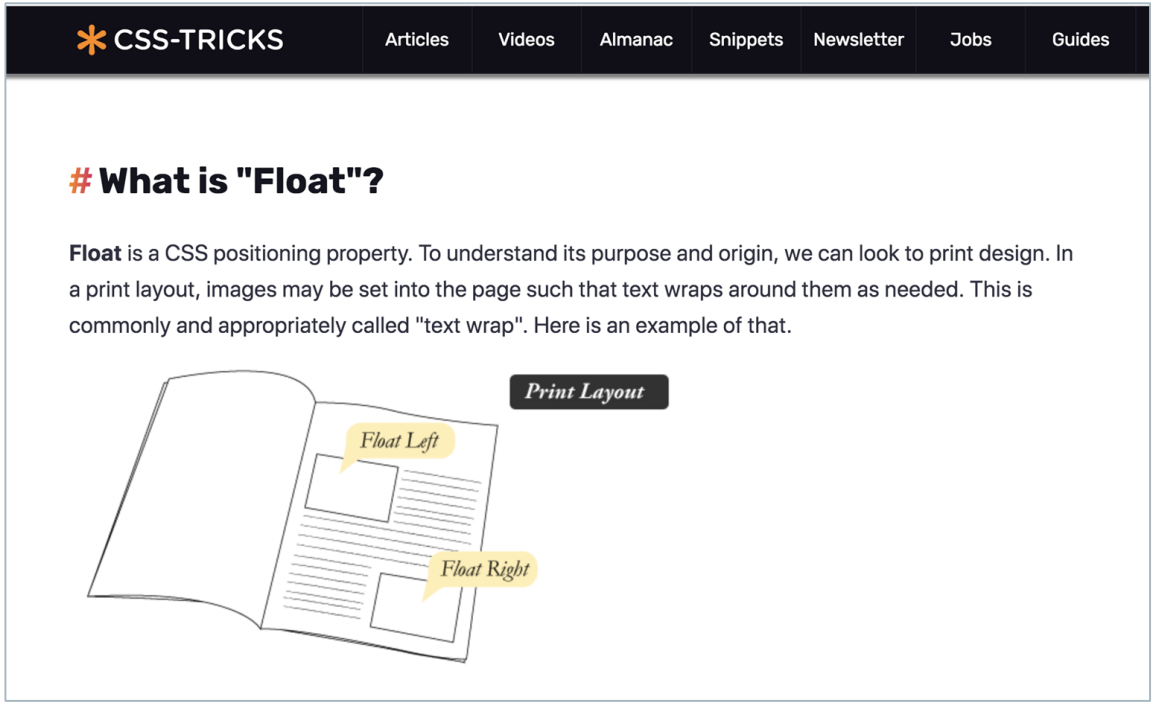


Instructor Demonstration

Floats

A Fantastic Guide to Floats

To all serious frontend developers, this is a necessary read:
css-tricks.com



Challenge: Float Layout

In this activity, you'll flex your newfound floating skills by creating a conceptual layout. Eyeball the design to your best ability.

(Additional instructions sent via Slack)

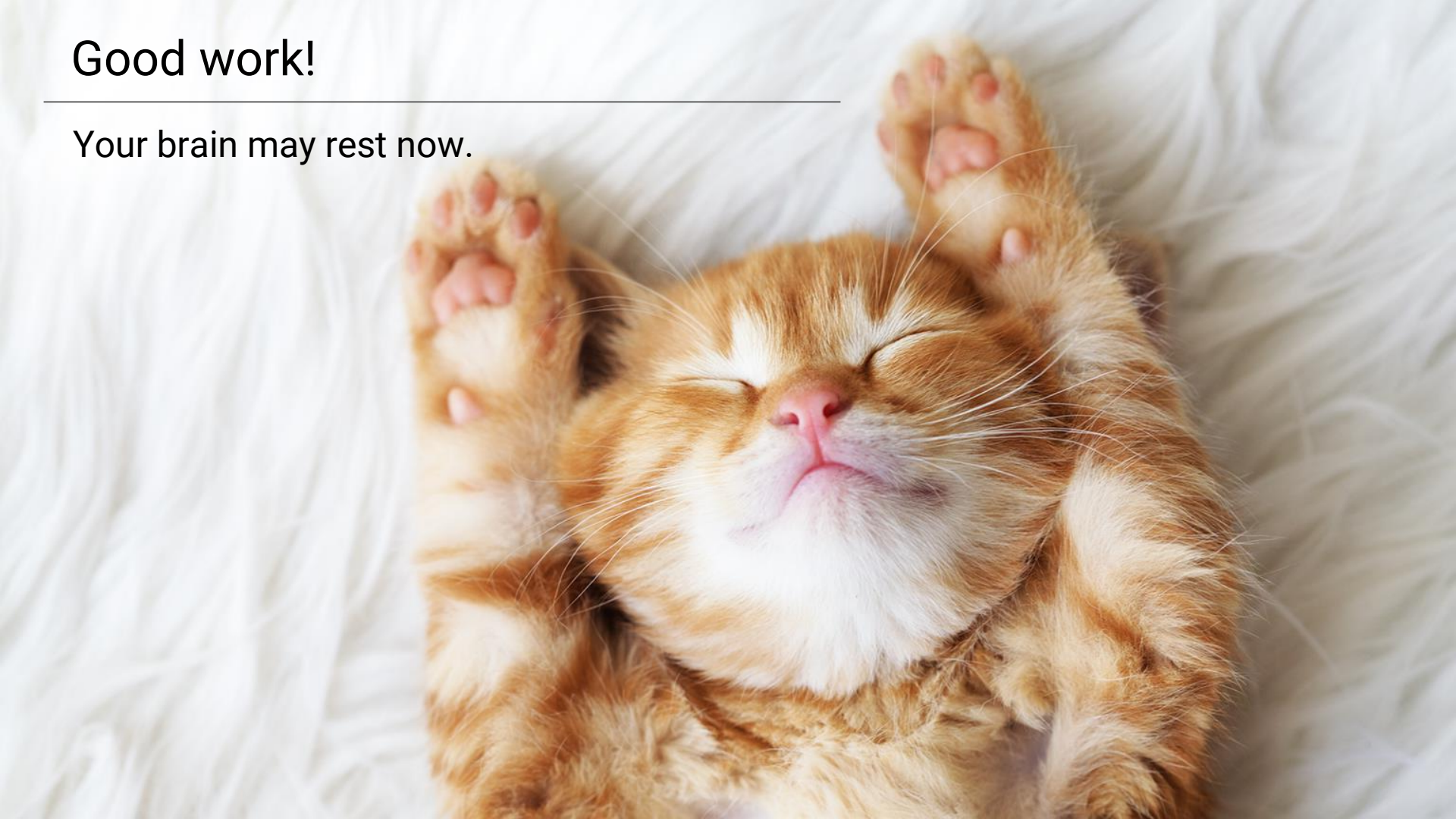
Suggested Time:
30 minutes

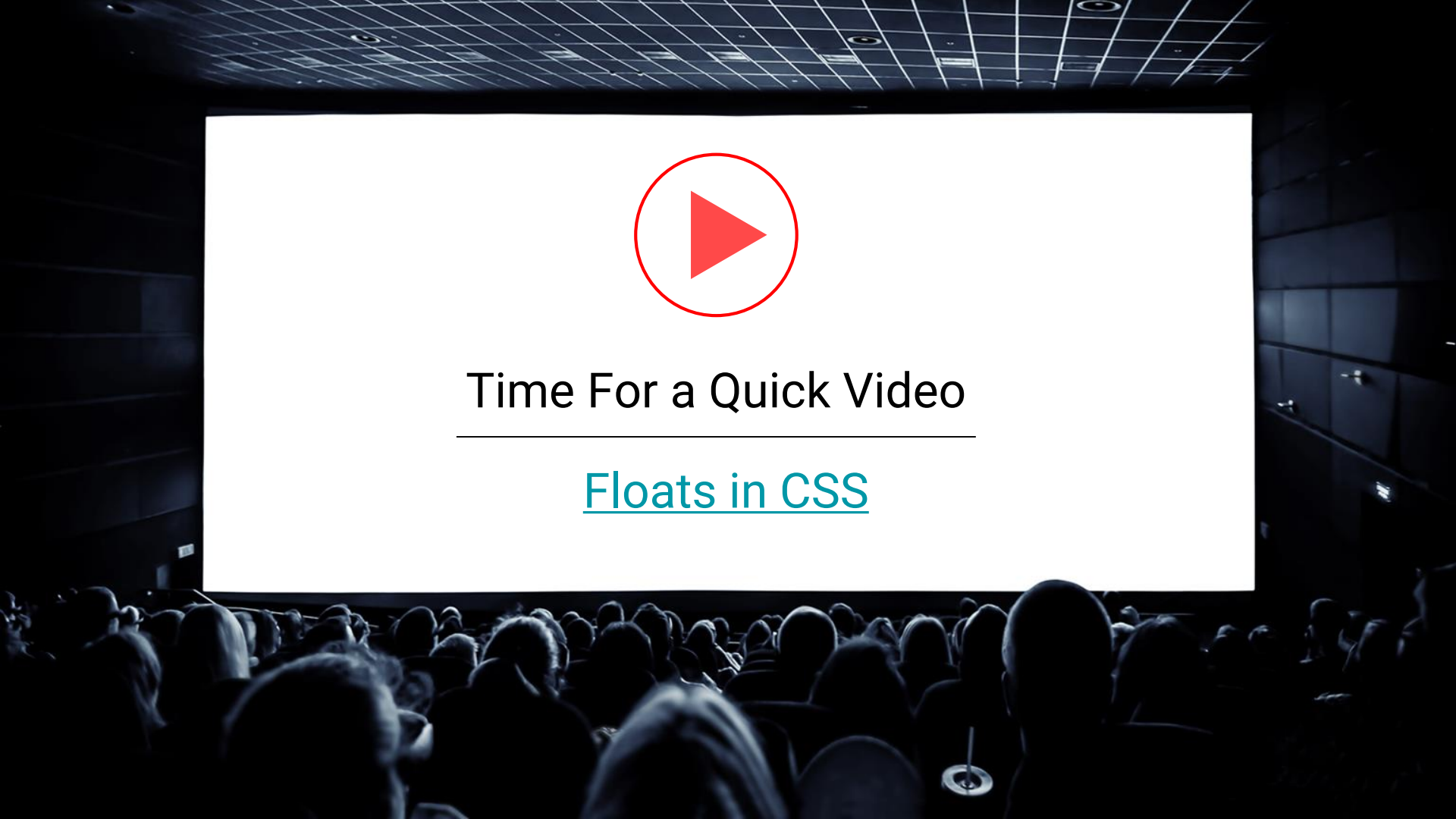
Activity: Float Layout

Suggested Time: 30 Minutes

Good work!

Your brain may rest now.



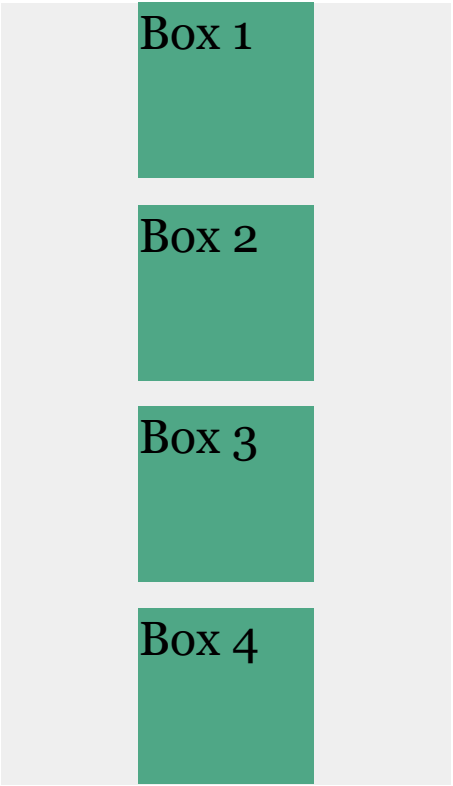

Time For a Quick Video
Floats in CSS

Take a Break!

CSS Positioning

Position: Static (Default)

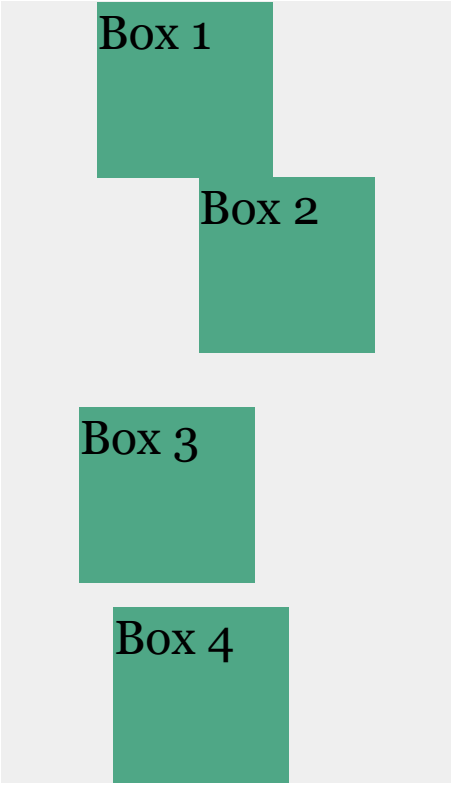
Four boxes placed statically (default):



Position: Relative

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

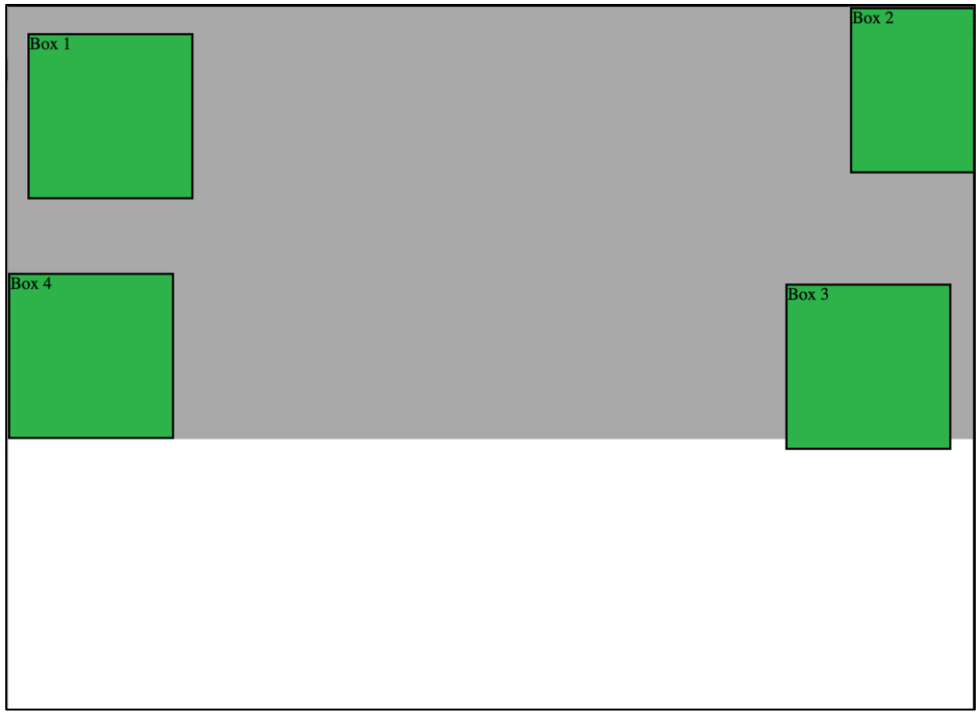
```css
.box {
  background: #2db34a;
  height: 80px;
  position: relative;
  width: 80px;
}
.box-1 {
  top: 20px;
}
.box-2 {
  left: 40px;
}
.box-3 {
  bottom: -10px;
  right: 20px;
}
```



Position: Absolute

Positioned relative to nearest positioned ancestor

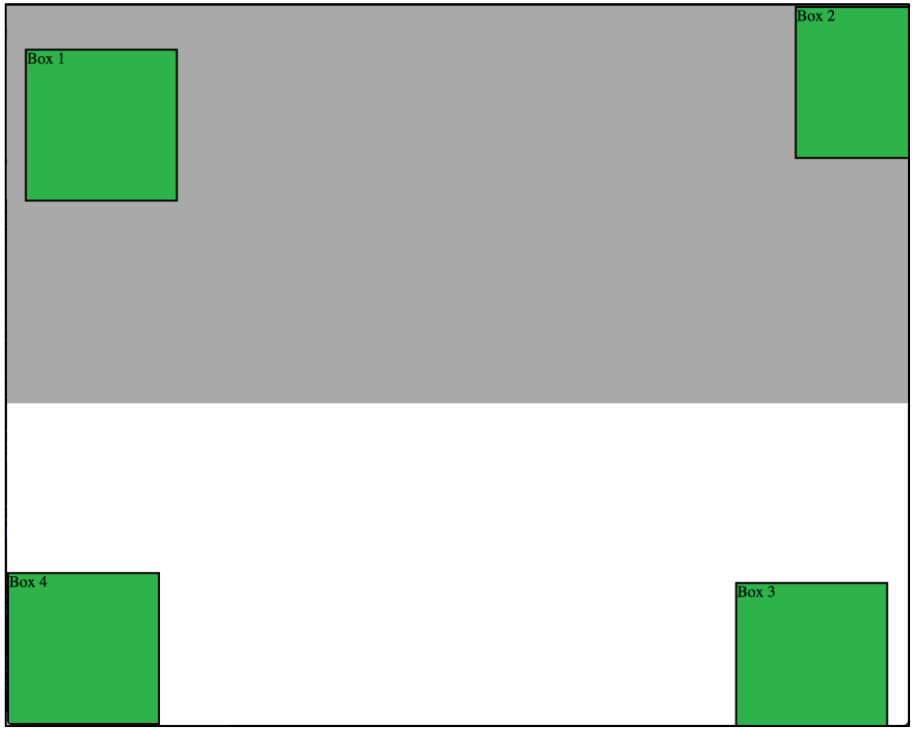
```css
.box-set {
  height: 400px;
  background: darkgray;
  position: relative;
}
.box {
  position: absolute;
  height: 150px;
  width: 150px;
  background: #2db34a;
  border: 2px solid black;
}
.box-1 {
  top: 6%;
  left: 2%;
}
.box-2 {
  top: 0;
  right: -40px;
}
.box-3 {
  bottom: -10px;
  right: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



Position: Fixed

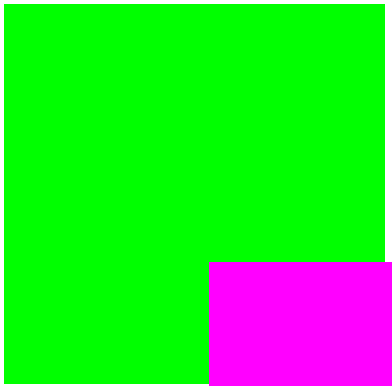
Position with exact coordinates in the browser window

```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;
}
```



Layering with z-index

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



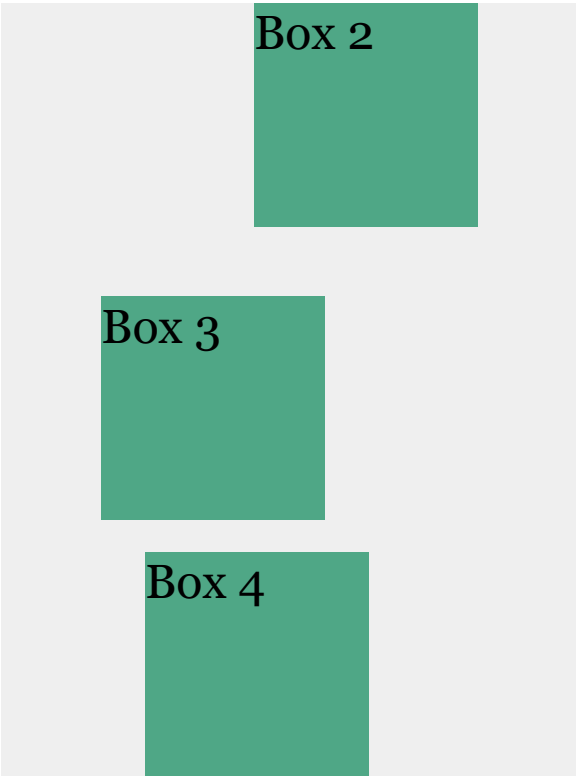
```
position: absolute;
z-index:1;
```

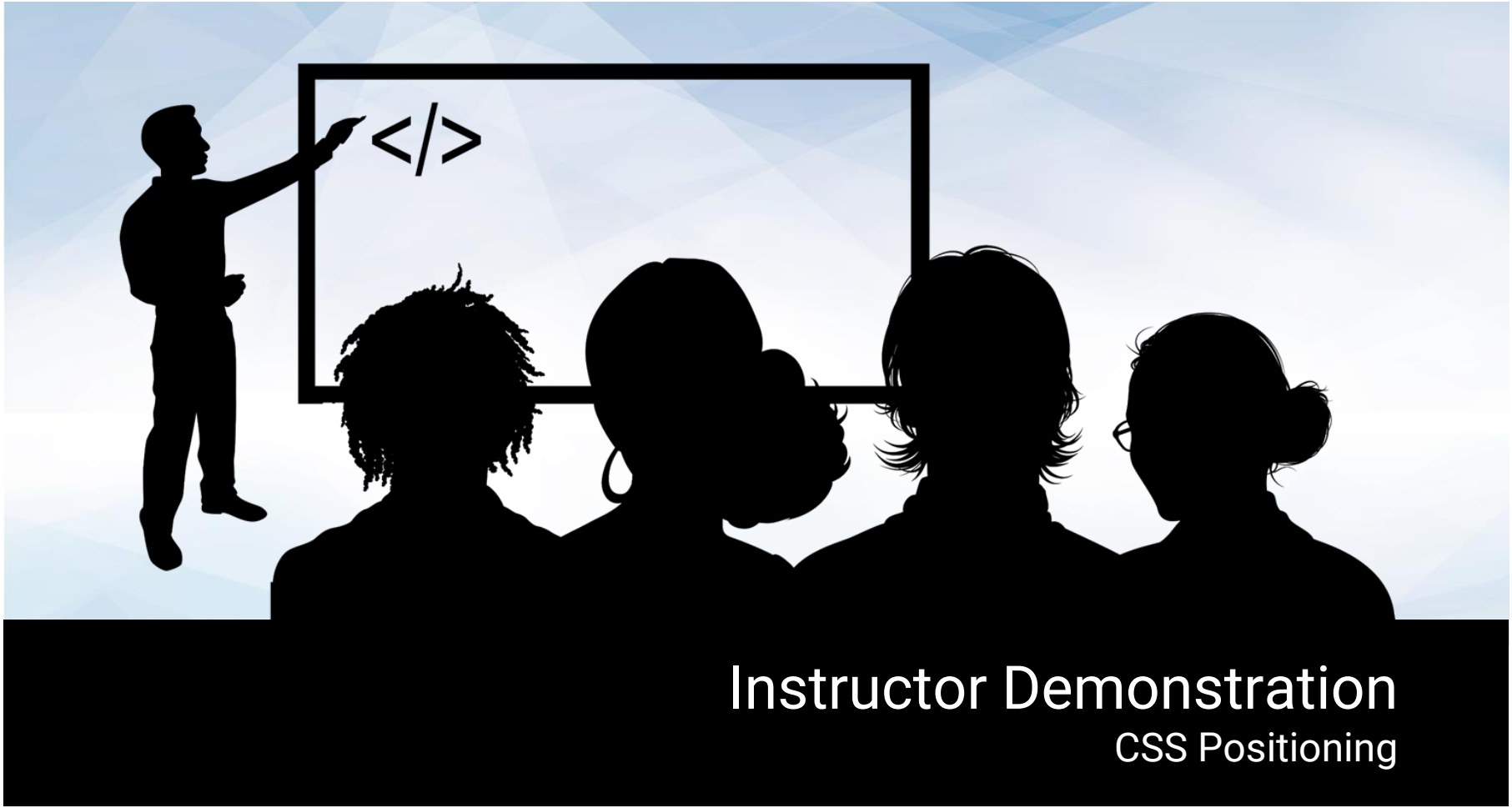
```
position: absolute;
z-index:2;
```

Hiding Things

Display: none allows you to hide elements from view.

This will become useful in later sections, when we'll hide and reveal specific HTML elements of our choosing.





Instructor Demonstration

CSS Positioning

Time For a Quick Video

Positioning in CSS

Great Resource

Another great read for frontend developers:
learn.shayhowe.com

Activity: CSS Positioning

In this activity, you'll flex your newfound positioning skills by creating another conceptual layout. Eyeball the design to your best ability.

(Check Slack for additional instructions)

Suggested Time:
30 minutes

Time For a Quick Video
CSS Positioning Layout

Advice

Tips to Keep Moving Forward

01 **Redo this at home.**

We designed the class activities to firm up your HTML/CSS skills. Try them again at home.

02 **REMEMBER!**

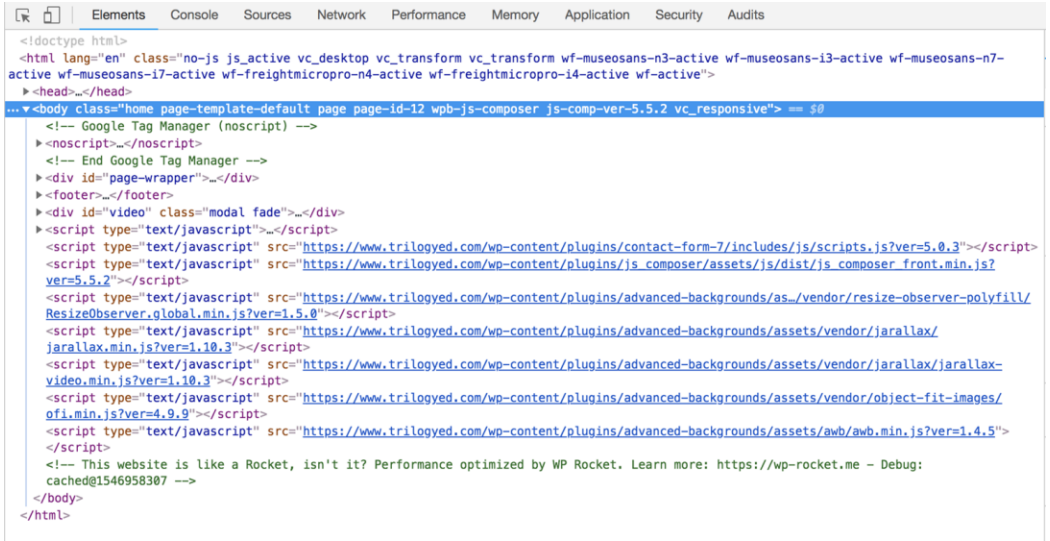
The best way to learn web development is to practice, practice, practice!

Chrome Inspector

Chrome Inspector Is Your Friend

To access the Chrome inspector, right-click on a page and select **Inspect**.

It lets you inspect the HTML, CSS, and more.



Chrome Inspector Is Your Friend

You can even edit the HTML/CSS in a webpage and instantly view your changes in the browser!

This works on any website, whether it's yours or not.

We’ll come back to this in our next class.

Recap

You Got This!

Questions?

Extra Material

Activity: Chrome Inspector

Choose a website you commonly visit (Amazon, Google, Huff Po, etc.) and heavily modify it using Chrome DevTools.

Suggested Time:
15 minutes

Activity: Chrome Inspector

Using Chrome Inspector, try to modify the following:

- Content (change text)
- Colors
- Spacing
- Any other CSS style rules

When you're done, send a screenshot to your class's Slack channel.

Suggested Time: 15 minutes

CSS Resets

Loading Multiple CSS Files

We can link our documents to more than one stylesheet at a time—one of the most powerful features of CSS/HTML.

By tapping into different stylesheets simultaneously, we can create complex layouts with plenty of design rules.

Just remember: the loading order matters!

```
<!DOCTYPE html>
<html>
<head>
          <title>Multiple CSS Files!!</title>
     <link rel="stylesheet" href="assets/style1.css">
     <link rel="stylesheet" href="assets/style2.css">
     <link rel="stylesheet" href="assets/style3.css">
</head>
<body>
     <header>
```

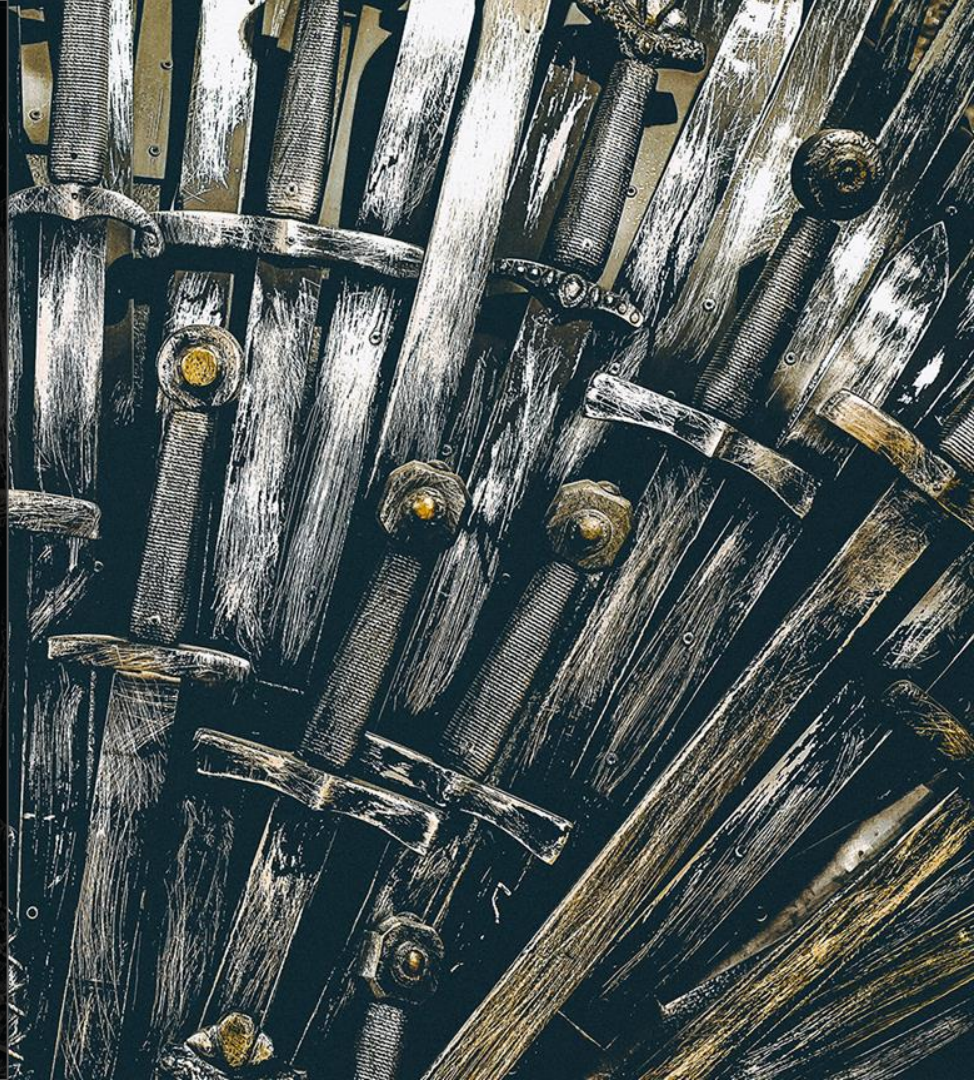
By a show of hands...
Which browser do you use?

Battle of the Browsers

Under the hood, web browsers often render webpages differently than their competition.

These disparities may result in HTML/CSS displaying differently in each web client.

Because of these potential differences, web developers need to make their websites cross-browser compatible.



Reset.css (or Normalize.css)

Reset.css will “reset” all browser-specific CSS. This means your site will appear the same in all browsers.

However, you will have to restyle everything yourself.

Why CSS Resets Matter

01

They help create browser-compatible websites.

02

They are an example of using someone else's CSS in ***your*** website!

03

They are a common topic in front-end developer interviews.

Activity: Reset.css

Incorporate a reset.css file in a basic HTML file. (Instructions sent out)

Note how the reset file impacts the styling of your HTML file.

Suggested Time:
10 minutes