



Front End Development

Session 4

Time to Add Some
Style



Learning Goals

In today's session, we will:

01

Define floats and display properties

02

Explain the difference between inline and block elements

03

Survey five common CSS position properties

04

Add CSS style to your HTML portfolio sites



Let's review



Q. What kind of code are we learning in this course?

Q. Can you name a role, job, or industry that uses this kind of code?

Q. What three “languages” do we use to create websites?

Q. What does "HTML" stand for?

Q. How can we examine website code on our computers?

Q. In HTML, what's the difference between the <head> and the <body>?

What is the difference between Git and Github?

What does the CLI stand for? What would we use it for?

How do you format a link in HTML?

How do we save our work with Git and Github?

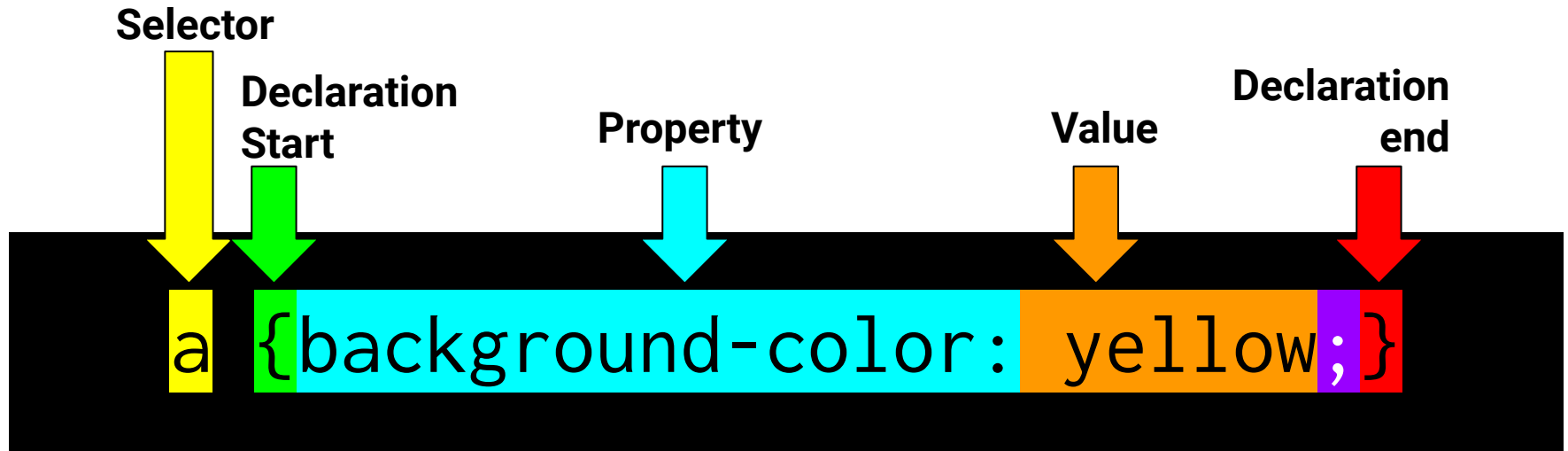
What does a `<div>` tag do?

What does CSS stand for? Why do we use CSS?



CSS Recap

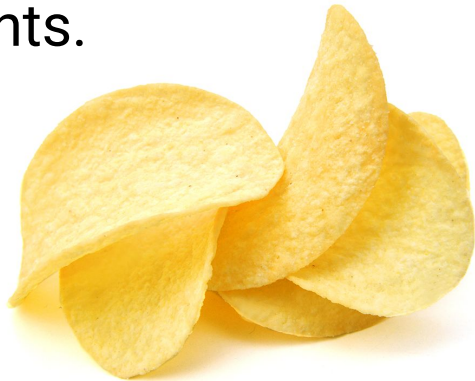
CSS Syntax



Classes vs. IDs

Classes

Classes (.classname) are used if the same style will be used on multiple HTML elements.



IDs

IDs (#idname) are used if a style is unique to a certain HTML element.



CSS Box Model:

Every HTML element you create is represented as a *rectangular box*, with the box's **content**, **padding**, **border**, and **margin** built up around one another like the layers of an onion!

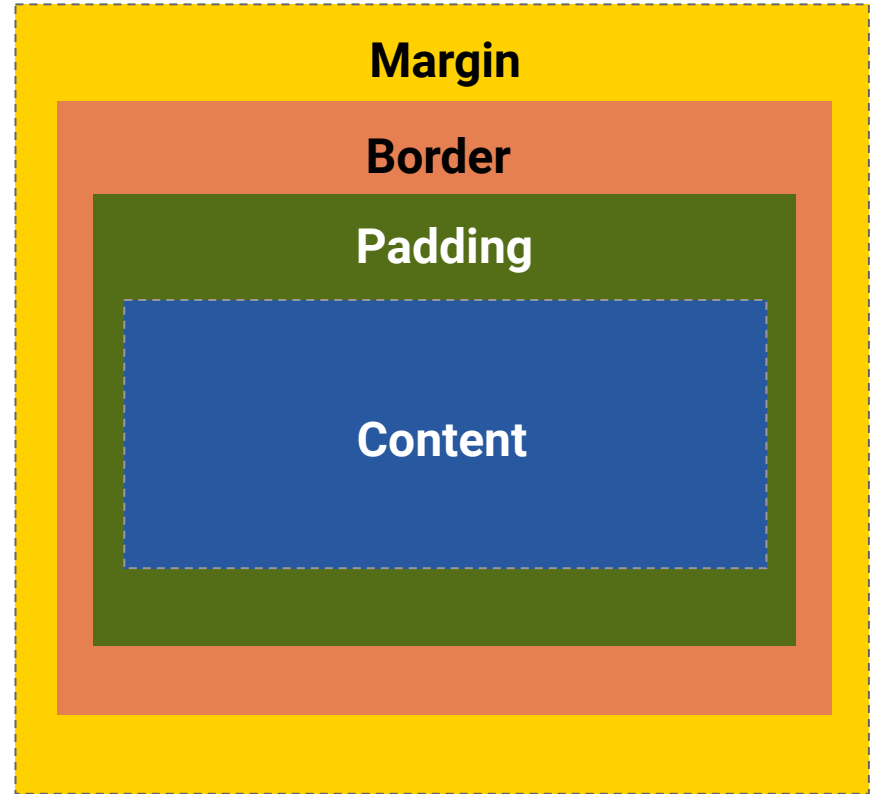
The CSS Box Model

Content: This is what's inside a container, such as text and images.

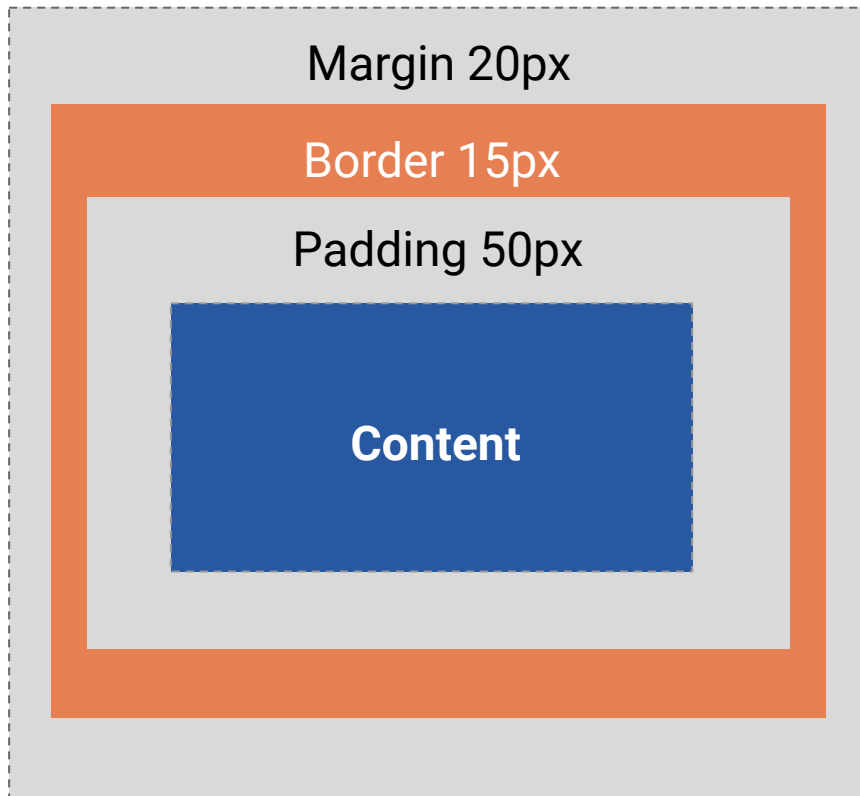
Padding: Padding is used to expand the space *inside* a box, between our content and our border.

Border: A border surrounds your content and distinguishes it from other elements around it.

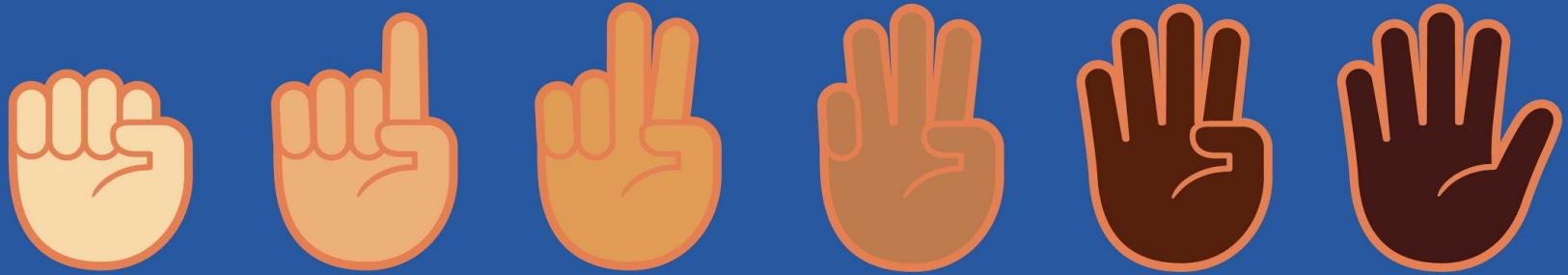
Margin: Margin is used to create additional space *outside* of your border. Margin spacing increases the distance between your box and other elements on the page.



The CSS Box Model



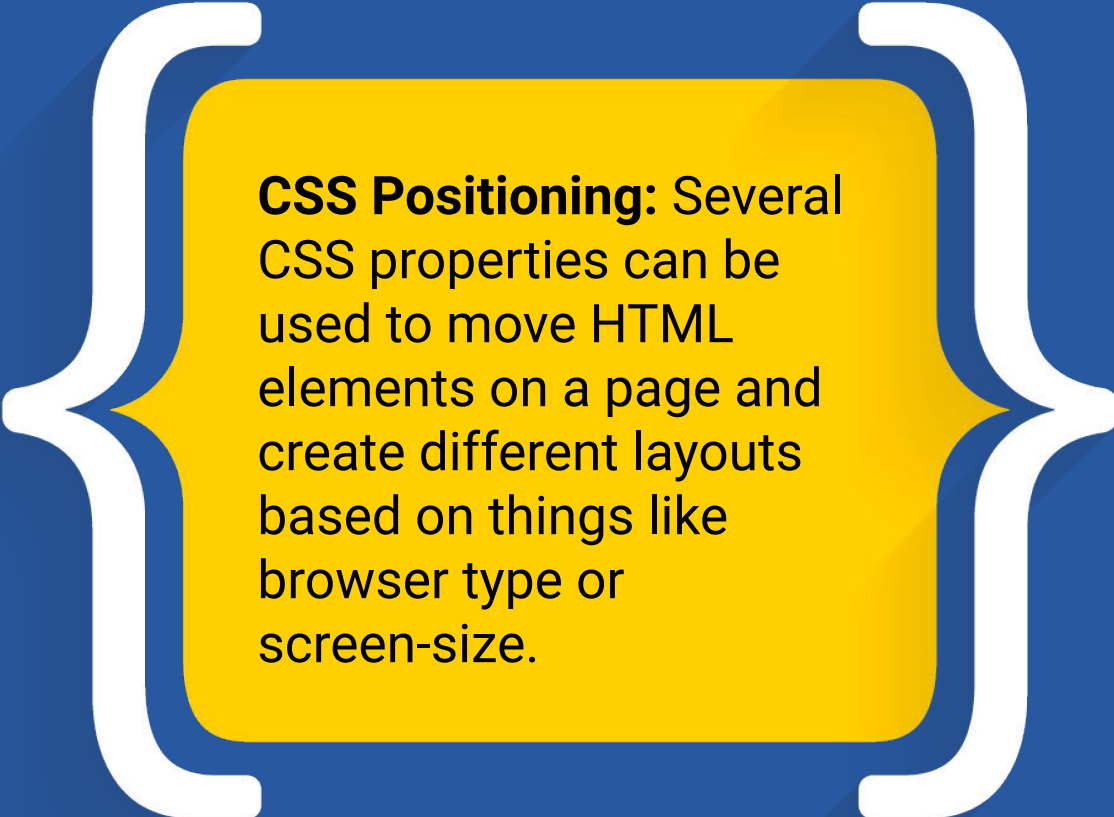
```
div {  
  background-color: navy blue;  
  padding: 50px;  
  border: 15px orange;  
  margin: 20px;  
}
```



Fist to five

A large yellow semi-circle is positioned on the left side of the slide, partially overlapping the blue background. The text 'CSS Positioning' is centered within this semi-circle.

CSS Positioning



CSS Positioning: Several CSS properties can be used to move HTML elements on a page and create different layouts based on things like browser type or screen-size.



CSS display properties

Display Values

Block vs Inline



Block

- `<div>` and `<section>`
- `<header>`
- `<table>`
- `<blockquote>`
- `` and ``
- `<video>`

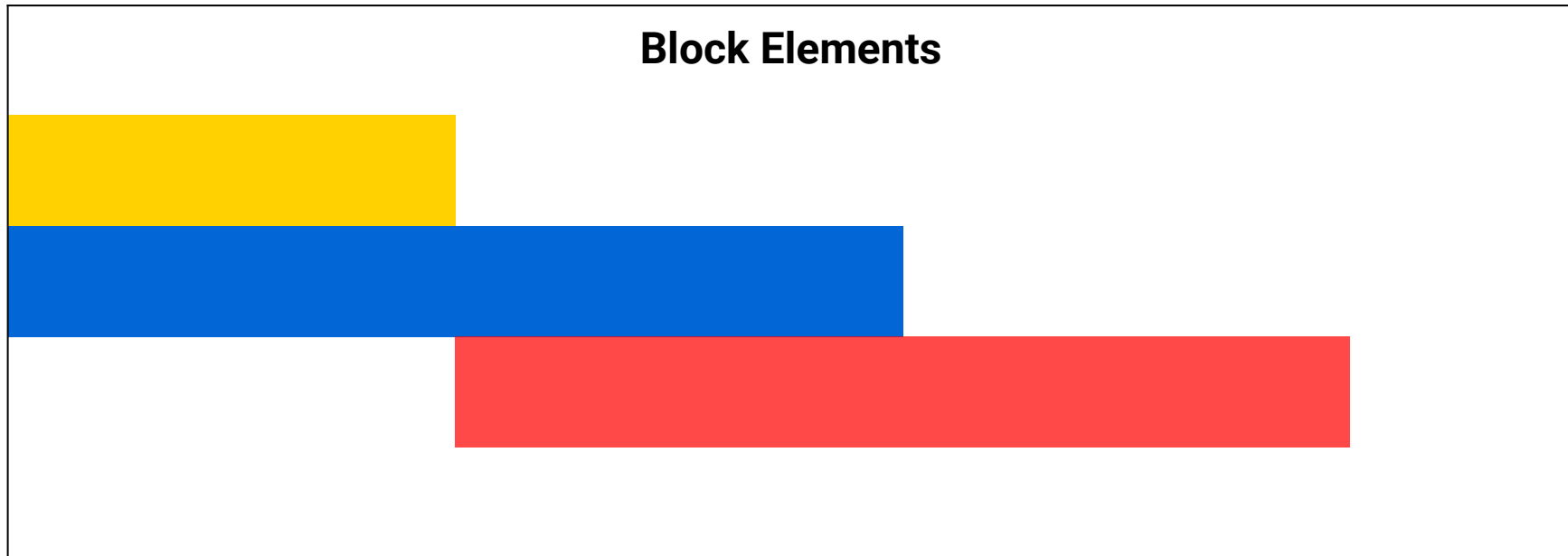


Inline

- `<a>`
- ``
- `<button>`
- `<script>`
- `` and ``
- `<code>`

display:***block***;

Displays an element as a **block** element. Blocks *always* start on a new line and take up the *full width*, stretching out to the left and right as far as it can. A <div> is a block-level element!



$$\text{display:}\textit{inline};$$

Displays an element as **inline**. An inline element starts *wherever you put it* and takes up *only as much space* as necessary. HTML links are inline elements!

The diagram shows a row of four horizontal bars of different colors: yellow, white, blue, and red. The yellow bar is the topmost and is labeled "Inline Elements". Below it are three more bars: a white bar, a blue bar, and a red bar. The blue and red bars are positioned below the white bar, indicating they are inline elements. The blue bar is on the left, and the red bar is on the right, separated by a gap. The white bar is positioned below the blue and red bars, indicating it is a block element that spans the width of the container.



Block elements are normally used as containers for *inline* elements.

Floats

The Float property is used to position and format content.


For example, an image can ***float*** around the text in a container.



Sed nisl est, luctus id venenatis ut

Nullam id lectus orci. Fusce eget erat a augue tincidunt rhoncus. Curabitur diam risus, pellentesque ut dui ac, egestas varius mauris. Nunc eleifend, eros eu consectetur rhoncus, ligula augue scelerisque neque, eu ornare massa velit at justo. Etiam et hendrerit purus. Phasellus sapien felis, malesuada a mollis non, pulvinar vitae nulla. Quisque ultricies id nisl a dictum. Sed vehicula elementum massa id convallis. Curabitur iaculis ex est, eget accumsan ante venenatis id. Nunc sed interdum erat, sed ullamcorper massa.

Ut convallis mi vel justo dictum tincidunt. Mauris ac ullamcorper ante. Morbi euismod tortor id urna vulputate finibus. Duis fringilla massa lectus, id rhoncus est laoreet nec. Donec dui dui, euismod sollicitudin fermentum quis, porta vitae dolor. Mauris blandit eu tortor ac venenatis. Sed non imperdiet nunc, id rutrum erat.

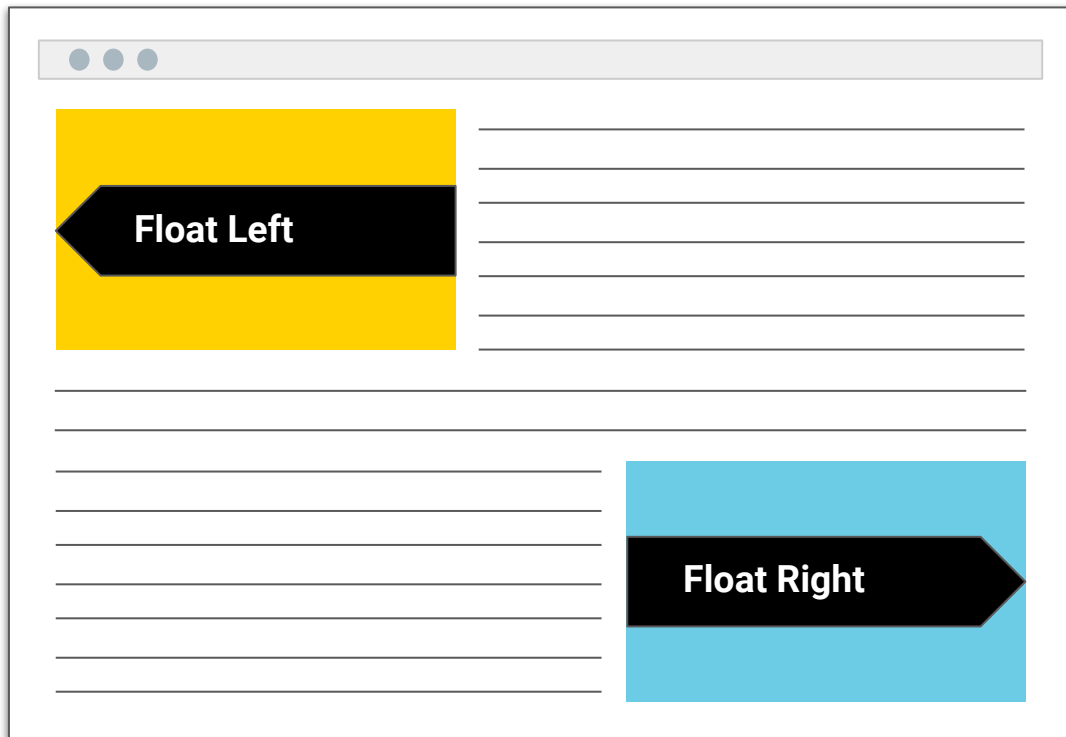


The CSS “Float” property
places an element on the
left or right side of its
container, allowing text
and inline elements to
wrap around it.

The float CSS property

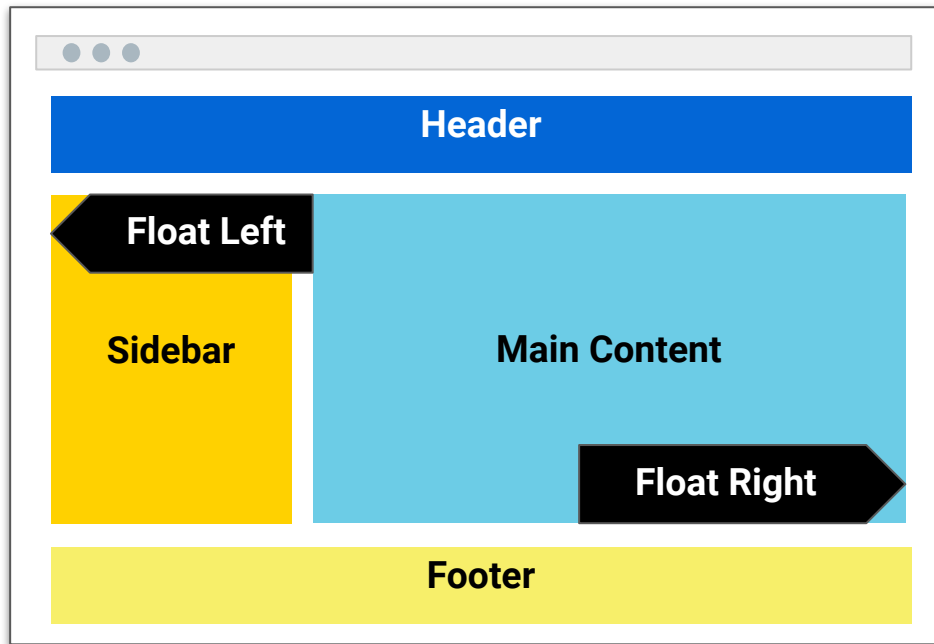
Floats were designed to wrap **text** around **images**.

However, floats can be used to make **any element** move to the farthest *left* or *right* within its “parent” container.



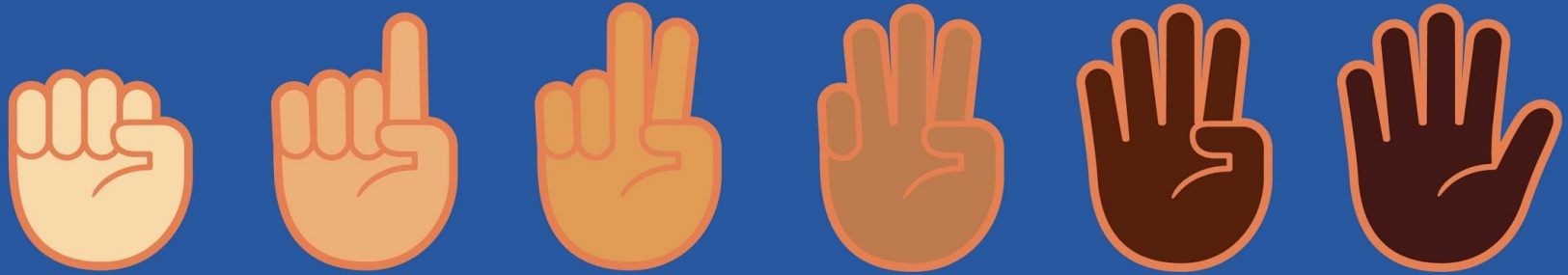
The float CSS property

Float can also be used to position containers ***relative** to each other*; however, this is a bit tricky to in practice and may take some trial & error!



CSS

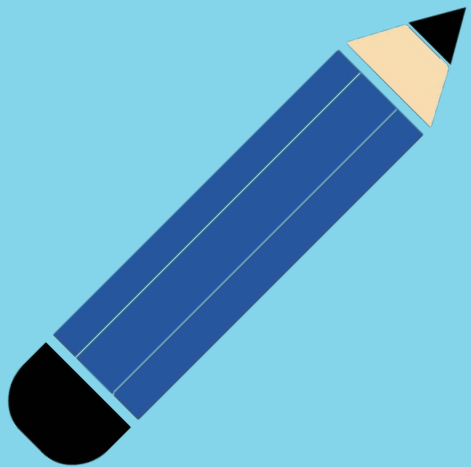
```
#sidebar {  
    float: left;  
}  
#main-content {  
    float: right;  
}
```



Fist to five

Let's





Activity instructions are in Canvas for: Practicing Float and Display

Suggested Time:
30 minutes





Let's review



Questions?





Five Properties

Five Properties

There are five position properties that you should know:

01

Static

02

Relative

03

Fixed

04

Sticky

05

Absolute

Static

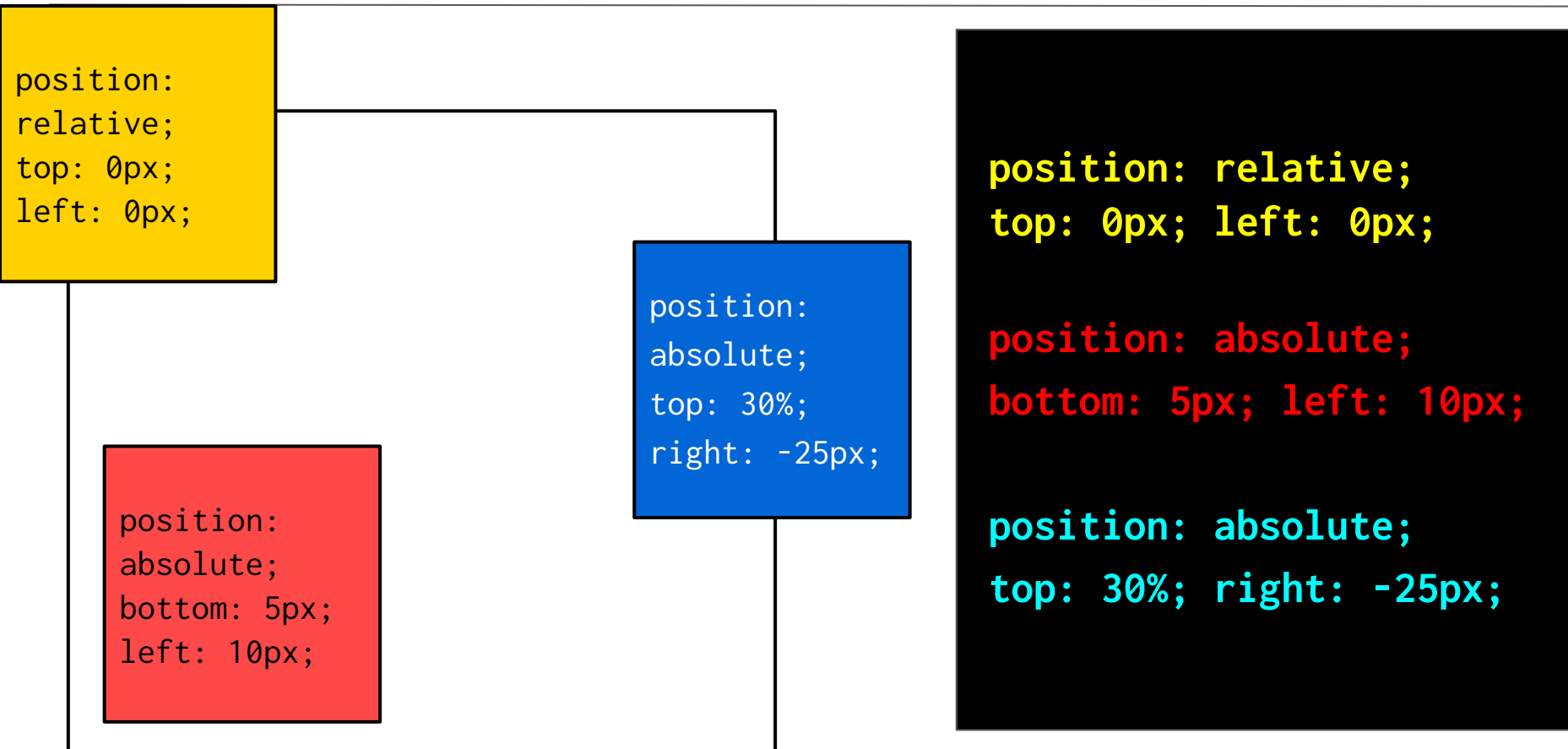
Positions an element according to the normal flow of the document.

```
.myClass {  
position: static;  
}
```



Static is the default position property applied.

position: static;



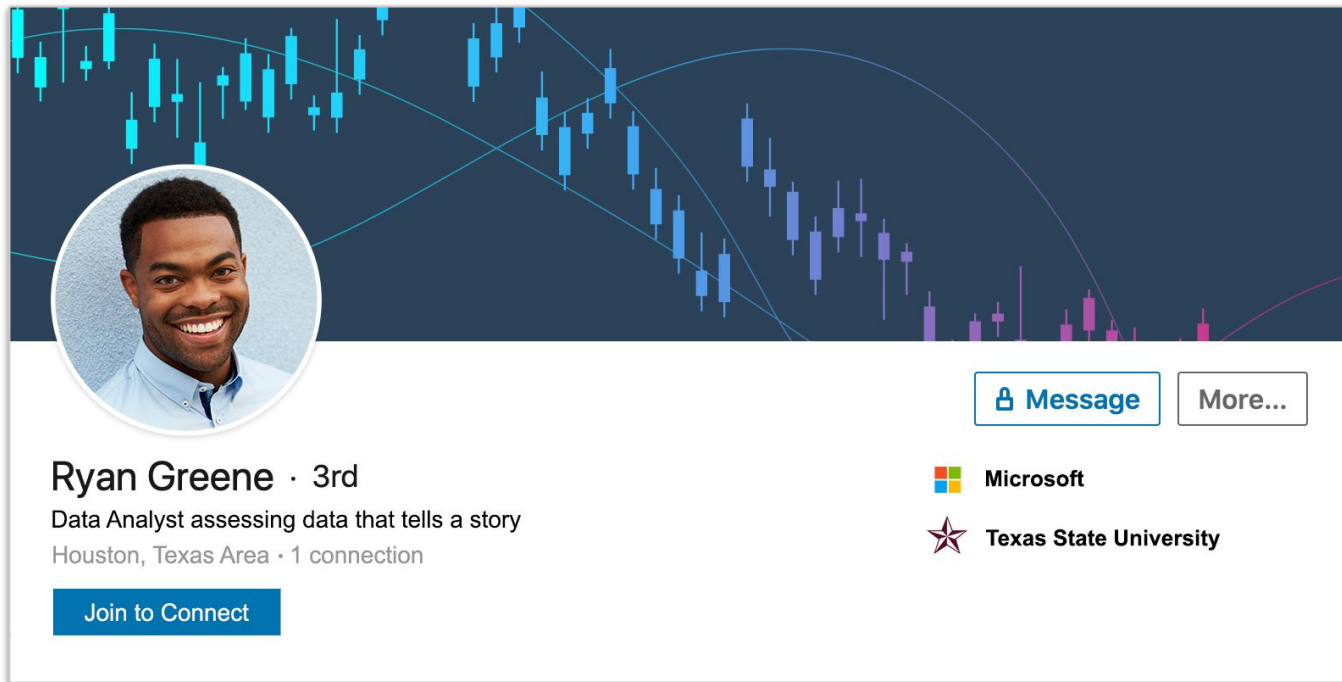
Relative

Positions the element relative to the normal position it would otherwise have (e.g. if it was left as a static element). Used to offset an element from its default, based on any values assigned (top, right, bottom, left).

```
.myClass {  
position: relative;  
}
```

Relative

An example of a relative element is the container for any profile picture that overlaps other HTML elements. Think of the container for the profile picture for LinkedIn.



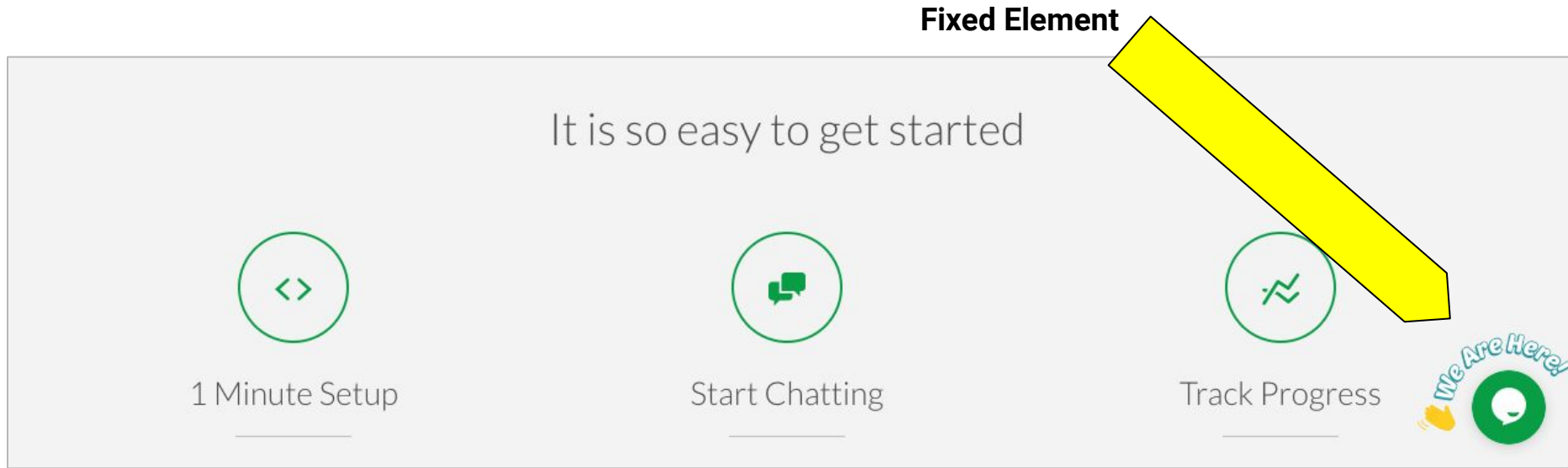
Fixed

The element is removed from the normal document flow. It will always stay where you put it even if a user scrolls down the page.

```
.myClass {  
position: fixed;  
}
```

Fixed

An example of a fixed element might be a contact us button that follows you as you scroll.



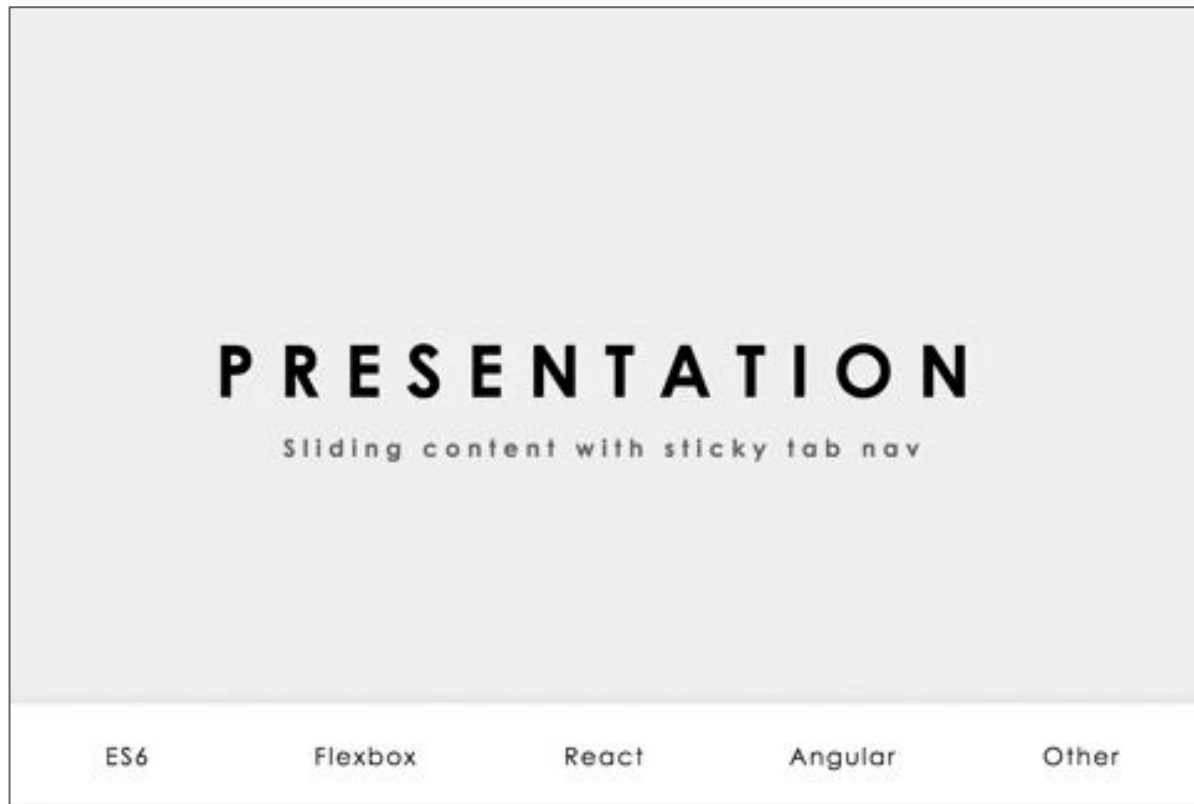
Sticky

A “sticky” element is positioned based on the user's scroll position. It appears fixed until a user tries to scroll, then moves along with the user, overriding other elements.

```
.myClass {  
position: sticky;  
}
```

Sticky

An example of a sticky element is a nav bar that scrolls with you when you hit a certain height.



Absolute

Absolute elements are removed from the document flow and are placed in absolute position based either on their parent container or the overall document body.

```
.myClass {  
position: absolute;  
}
```

Absolute Positioning vs. Relative Positioning

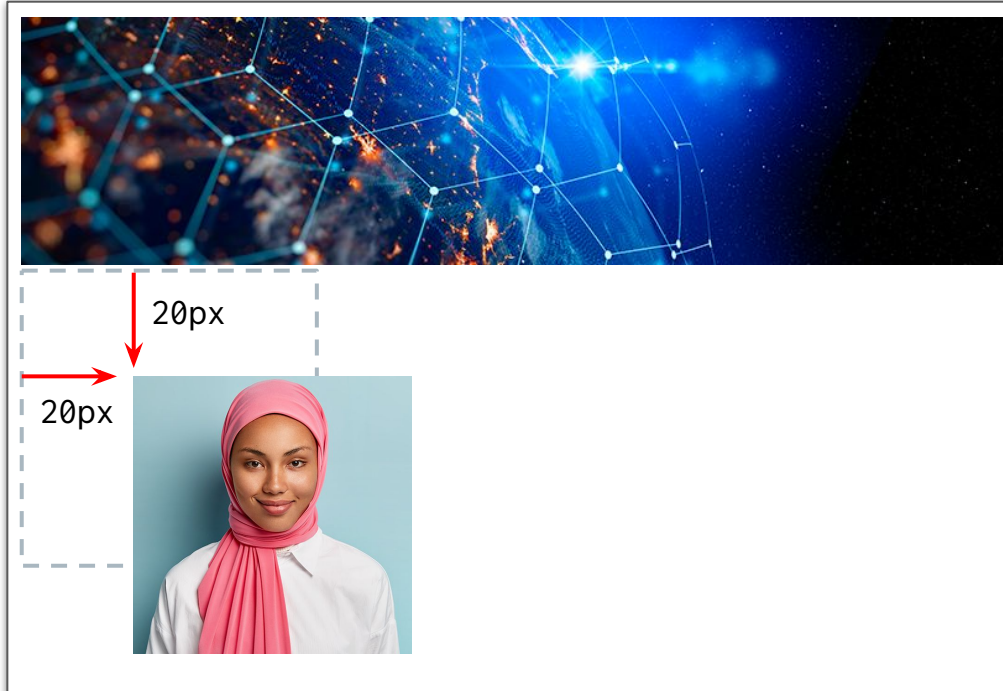
Relative

Relative positioning is just like stating no positioning at all! The left, right, top and bottom attributes "nudge" elements out of their normal layout.

Absolute

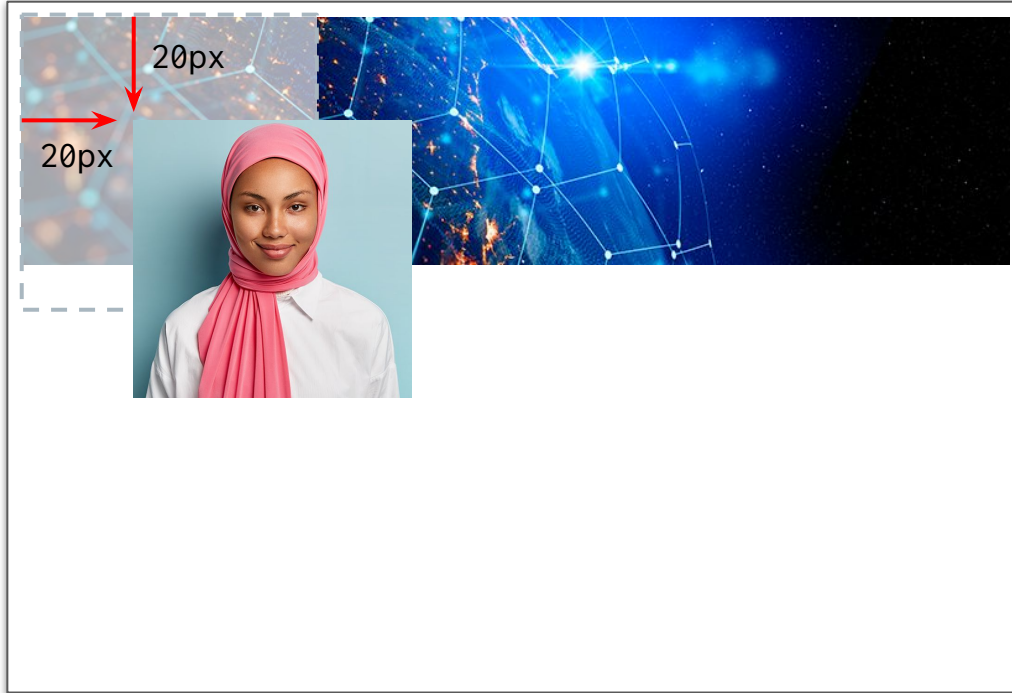
Absolute positioning allows you to place your element *precisely* where you want it - and it won't budge.

position: relative;

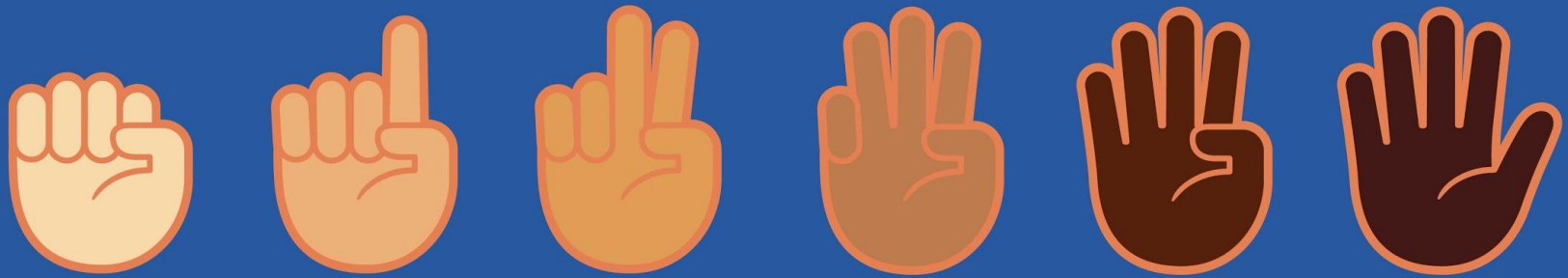


```
position: relative;  
top: 20px;  
left: 20px;
```

position: absolute;



```
position: absolute;  
top: 20px;  
left: 20px;
```



Fist to five

Let's





Activity instructions are in Canvas:

You will work with a partner to practice **CSS Positioning**

Suggested Time:
30 minutes





Let's review



Questions?



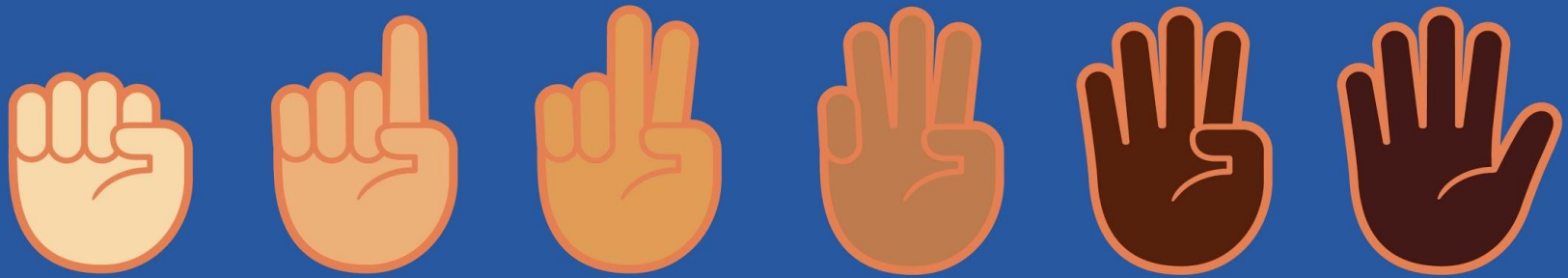
Workshop



Activity instructions are in Canvas for: Styling Our Portfolios

Suggested Time:
50 minutes





Fist to five



Let's review





**Time to
Recap**

Learning Goals

Our objectives for today's session were to:

01

Define floats and display properties

02

Explain the difference between inline and block elements

03

Survey five common CSS position properties

04

Add CSS style to our HTML portfolio sites

Reflection

What was your **favorite part** of today's session?

What was the **most interesting thing** we covered today?

What do you **still have questions** about?





Sneak Preview

Tomorrow, we'll be learning how to use a framework called "Bootstrap" that will allow us to easily design responsive, mobile friendly websites!

Questions?



The End