CSS

Setting up Projects

We've worked with **HTML** previously to make our websites, and **HTML** will be something we will continue to use. While **HTML** provides us a way to put our words and images onto the page and determine the order and rough layout, it doesn't provide us a way to efficiently change the style of the page. This is the job that **CSS** (Cascading Style Sheets) has been designed for. As we start adding in more technologies and languages into our projects we will often need to deal with multiple files.

When setting up our projects now, we will want to make one overall folder, let's call this one Example for now. Inside of the Example folder we can then make an **HTML** and a **CSS** file. We can choose to name them anything we like, but for now let's go with index.html and style.css, these are common choices.

Example/

index.html

style.css

copy

**Inside of** index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Example</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Example</h1>

</body>

</html>copy

**Inside of** style.css

h1 {

color: rebeccapurple;

font-size: 50px;

}

copy

**Visually in browser**

Example

This will change the color of the h1 text from the default of "black" to a particular shade of purple. It will also change the size of the text from the default 32px (pixels) to 50px. There are many different ways to define colors and to define sizes in **CSS** and we'll cover them more later--but for the curious, W3 schools has fantastic references for both.

* [Color Names](https://www.w3schools.com/colors/colors_names.asp)
* [More Colors](https://www.w3schools.com/html/html_colors.asp)
* [Sizes](https://www.w3schools.com/cssref/css_units.asp)

When writing **CSS** we need to be aware of the selector (which elements we are selecting) and inside of the curly brackets the properties that we are changing from their defaults. We'll talk more about which properties to learn in a later chapter, but for now we'll turn our attention to the selector.

*If you're curious why there is a named CSS color in the official standard called rebeccapurple read*[*this*](https://medium.com/@valgaze/the-hidden-purple-memorial-in-your-web-browser-7d84813bb416)*(****warning****, very sad story).*

# Selector Basics

### **Objectives:**

* Learn about selecting **HTML** in our **CSS** using tags, classes, and ids

## **Selecting an element**

Any HTML element in our code can be selected like h1, a, table, button. We can even select some tags you might not expect like body (useful if we want to define an overall padding or background color to the website.

When we write code like the following...

h1 {

color: red;

}

p {

text-align: center;

}copy

We are telling every h1 tag on the page to have red text color, and every paragraph on the page to have its text centered (instead of the default left alignment). What if we want to only style one particular h1 tag to be red or a few of our paragraphs to be centered?

## **Selecting a class**

Another way we can select a tag is by its class. In the **HTML** we can add a class attribute with the name of the class we want to select (class names cannot contain a space, spaces are used to separate multiple classes).

<p class="is-centered">This text should be center aligned</p>

<p>This text should remain left-aligned</p>

<p class="is-red">This text should be red</p>copy

In the css we can then select a class by using a . followed by the class name.

.is-centered {

text-align: center;

}

.is-red {

color: red;

}

copy

This text should be center aligned

This text should remain left-aligned

This text should be red

Classes are incredibly flexible and we will encourage you to use them frequently. Classes are allowed to be applied to multiple elements even if they aren't the same type. We can also apply more than one class at a time to an element if we so choose.

## **Selecting an id**

Sometimes there only needs to be one of a given element, in those cases we can use id as well.

<div id="game">

<img id="player" src="walk-animation.gif" alt="player sprite">

</div>copy

In the css we can then select an id by using a # followed by the class name.

#game {

width: 320px;

height: 240px;

overflow: hidden;

background: linear-gradient(#1293f5, #bae8ff 80%, #35b236 80%);

}

#player {

margin-top: 140px;

margin-left: 50px;

}

/\* Don't worry about knowing what all of these properties are yet \*/

/\* we'll come back to them later, we promise! \*/copy

ninja

In the example above, we should only ever have one #game field and one #player in the scene at a time. So id will be a fine choice here. If we decided to add a second player we'd need to give it a unique id.

In general we'll find that classes can encourage us to reuse our code (a good thing) and are more flexible. So unless we really need to know an element is unique we should use classes for now.

Text

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

Use . for classes

{} for regular heads like <h1> <p>

Use # for div

# Selectors Advanced

### **Objectives**

* Learn about Wildcard, Descendant, Direct Descendant, and Attribute Selectors
* Learn how combining selectors affects Selector Specificity

## **The Wildcard \***

If you ever want to select all the elements, the wildcard selector can do just that.

\* {

outline: 1px solid red;

}copy

This will draw a red outline around every element on the page which is really useful useful when we start to work on positioning elements where we want them on the page.

## **Selector Specificity**

What would happen if we were to have the following **HTML** and **CSS**?

<h1 id="red" class="green">Hello World</h1>copy

h1 {

color: rebeccapurple;

}

.green {

color: mediumseagreen;

}

#red {

color: tomato;

}copy

The h1 selector wants to make the tag rebeccapurple, the class wants to make the tag mediumseagreen and the id wants to make the tag tomato. Which color will it actually be?

Click to see the answer

In rough terms, selecting a tag like h1 or p has a strength like **1**, selecting a class like .green has a strength like **10**, and selecting an id like #red has a strength like **100**. So ids will be strongger than classes which will be stronger than elements. If we're wondering about the wildcard selector \*, it has a strength like **0.1**.

## **Descendant Selector**

Sometimes we know the element we want to select is inside of another element. If the "parent" element is one we can easily select (maybe it has a class on it), then we can use this to our advantage to access the child element easily.

<p>This is a quote about CSS</p>

<div class="quote">

<p>"CSS is like chess. You learn the basics in a day and spend a lifetime mastering it."</p>

</div>

<p>Chris Coyier (css-tricks.com)</p>copy

.quote p {

color: darkslategrey;

font-style: italic;

}copy

This is a quote about CSS

"CSS is like chess. You learn the basics in a day and spend a lifetime mastering it."

Chris Coyier (css-tricks.com)

If we want to select the paragraph that is inside of the "quote" class and not the paragraphs that are outside of the "quote" we can use a space in our selector between the parent element and the child we want to select. This can save us from having to make additional classes to select just the element we want.

## **Direct Descendant Selector**

Similar to the Descendant Selector, we can select a child element with the Direct Descendant Selector `>`, but unlike the Descendant Selector it won't select grandchildren.

<div class="header">

<img src="brand-img.png" alt="brand image"class="support tag close from-rainbow">>

<a href="#">Home</a>

<a href="#">About</a>

<a href="#" class="cart">

<img src="cart.png" alt="cart-icon">

<span>Cart (2)</span>

</a>

</div>copy

.header > img {

height: 50px;

}copy

In the above example, the brand image will be given a height of 50px and the cart icon won't be affected.

## **Combining Selectors**

selector | strength

-------------------

div | +1

#game | +100

p | +1

.score | +10

-------------------

combined | 112copy

We talked earlier about selector specificity and mentioned how id has a relative strength like **100**. We can actually make even stronger selectors by combining them. If we write a selector like div#game p.score (select a paragraph with the class "score" that is a child of a div with the id "game") it will have a strength like **112**.

## **Attribute Selector**

Sometimes we have a situation where we want to style two similar elements that rely on an attribute for how they behave. For instance if we are making a form we likely use `<input type="text">` for the user to input data into and `<input type="submit">` which acts as the button the user clicks to submit the form. CSS we would want to use on one we might not want to use on the other. While we \*could\* use a class to differentiate between the two we could also use a selector that looks at the attribute.

<form action="/process" method="post">

<input type="text" name="username" value="Rebecca">

<input type="submit" value="Enter Username">

</form>copy

input {

padding: 5px;

border-color: rebeccapurple;

}

input[type="text"] {

border: none;

border-bottom: 2px solid rebeccapurple;

}

input[type="submit"] {

color: white;

background-color: rebeccapurple;

}copy

We can apply some styles to both inputs using the `input` selector and follow up with attribute selectors to style the text input and submit button differently.

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

# CSS Properties - Color

### **Objectives**

* Learn about named , rgb, and hexadecimal colors
* Learn the difference between color and background-color

Color is one of the best properties to work with when setting look and feel of the website. There are some tricky things to be aware of when working with it. Color can be set using a few different systems.

## **Named colors**

We can see a full list [here](https://www.w3schools.com/colors/colors_names.asp)

black

white

red

green

blue

magenta

## **RGB**

Each number must be between 0 and 255 and they are set for **red**, **green**, and **blue** respectively.

rgb(0,0,0)

rgb(255,255,255)

rgb(255,0,0)

rgb(0,128,0)

rgb(0,0,255)

rgb(255,0,255)

## **Hexadecimal**

Hexadecimal works very similarly to **rgb** with each pair of numbers/letters representing the **red**, **green**, and **blue** respectively.

#000000

#ffffff

#ff0000

#008000

#0000ff

#ff00ff

We can think of hexadecimal like switching to letters when we run out of numbers, in hexadecimal

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, fcopy

are all valid values for 0 through 15. The hexadecimal number FF when expressed in decimal (base 10) is 255.

## **Color vs Background Color**

When we apply **CSS** we need to keep in mind that color changes the color of text, and background-color changes the color in the background of whatever element we have selected

<h1>Coding Dojo</h1>copy

h1 {

color: cyan;

background-color: grey;

}copy

# Coding Dojo

Background color can be applied to a multitude of elements (body, div, thead, etc...), not just ones obviously containing text.

# CSS Properties - Text

### **Objectives**

* Learn about text-align and text-decoration
* Learn about font-weight, font-style and font-family

## **Text Alignment**

We can adjust the text alignment using text-align

#### **text-align: left;**

#### **text-align: center;**

#### **text-align: right;**

## **Text Decoration / font style / font weight**

We can apply (or remove) an underline to text using the text-decoration property

#### **text-decoration: underline;**

#### **~~text-decoration: line-through;~~**

#### **text-decoration: none;**

We can apply or remove italics using font-style

#### **font-style: italic;**

#### **font-style: normal;**

We can make text **bold** or narrow with font-weight

font-weight: normal;

font-weight: light;

**font-weight: bold;**

The font-weight property also accepts number values: 100, 200, 300, 400, 500, 600, 700, 800, 900

Note that font-weight will only be applied if the [font-family](https://www.w3schools.com/cssref/pr_font_font-family.asp) contains styles for it

## **Font Family**

There is a lot to know about fonts, but for now let's focus on the basic types: serif, sans-serif, monospace, and cursive.

#### **font-family: serif;**

#### **font-family: sans-serif;**

#### **font-family: monospace;**

#### **font-family: cursive;**

# Text Styling

### **Objectives:**

* Use what we've learned about the color and background-color CSS properties
* Use what we've learned about: text-align, text-decoration, and font-style CSS properties

Recreate the look of the h1, h2, and h3 tags in the image below

**Hint:** we may need to use an additional <span></span> element to wrap text inside of an <h1></h1> that we would like to apply a different color to.

# Sizes

Below are some of the most common styling properties that will affect all elements. This is not a definitive list and you should try doing a search for other properties that can help you to add style to your documents.

### **width | height:**

The width and height properties are used to determine the size of your elements. The values can be expressed in pixels (px) and percentage (%). When working with static content using px is suitable as you will be defining your page to not change. If you are working with responsive design, you will want to use %.

Be careful when setting your height property as this will determine how much content the element can hold. If you leave it unset, your element will expand to fit the content it holds. If you set it and you have more content than can be displayed then you will need to either manually adjust the height each time you change the content or utilize the overflow property.

When you want your element to resize ie: images, you can size just one property (width or height) and the other will adjust appropriately to keep the relative dimensions of the image.

a {

width: 25px;

}

div {

width: 100%;

height: 200px;

}

img {

width: 250px;

}

copy

### **font-size:**

The font-size property values can be expressed in four different units pt, px, em, %. pt and px (point and pixel) are considered static size values and will not adjust properly when resizing your page. em and % (responsive measure and percent) however will resize and are what most developers recommend using. Below is a drop-down chart showing the approximate equivalents between the four units.(\*\*Note\*\* not intended to update code)

 Point | Pixel | Em | Percent 6pt | 8px | 0.5em | 50% 7pt | 9px | 0.55em | 55% 7.5pt | 10px | 0.625em | 62.5% 8pt | 11px | 0.7em | 70% 9pt | 12px | 0.75em | 75% 10pt | 13px | 0.8em | 80% 10.5pt | 14px | 0.875em | 87.5% 11pt | 15px | 0.95em | 95% 12pt | 16px | 1em | 100% 13pt | 17px | 1.05em | 105% 13.5pt | 18px | 1.125em | 112.5% 14pt | 19px | 1.2em | 120% 14.5pt | 20px | 1.25em | 125% 15pt | 21px | 1.3em | 130% 16pt | 22px | 1.4em | 140% 17pt | 23px | 1.45em | 145% 18pt | 24px | 1.5em | 150% 20pt | 26px | 1.6em | 160% 22pt | 29px | 1.8em | 180% 24pt | 32px | 2em | 200% 26pt | 35px | 2.2em | 220% 27pt | 36px | 2.25em | 225% 28pt | 37px | 2.3em | 230% 29pt | 38px | 2.35em | 235% 30pt | 40px | 2.45em | 245% 32pt | 42px | 2.55em | 255% 34pt | 45px | 2.75em | 275% 36pt | 48px | 3em | 300%

h3 {

font-size: 10pt;

}

p {

font-size: 14px;

}

a {

font-size: 1.2em;

}

span {

font-size: 80%;

}

copy

### **overflow:**

This property determines what should happen when the content inside of a container is too much for the container size. You can set the container to hide the additional information that does not fit, show the information no matter what, or have a scroll bar added to the element so the content is contained but still viewable.

div {

overflow: scroll;

}

copy

### **background:**

The background property can modify the background of an element all in one line. This is a shorter version than splitting each property on its own line of code. The color can be defined using hex, rgb or semantic code.

p {

background: #ffffff url("cherries.png") no-repeat fixed center;

}

copy

### **background-color | background-image | background-position | background-size | background-repeat:**

These background properties adjust the background by property type. Like background, background-color can be defined using hex, rgb or semantic code.

p {

background-color: blue;

}

div {

background-image: url("cherries.png");

background-position: center;

background-size: auto;

background-repeat: no-repeat;

}

copy

### **border:**

This property adjusts all border elements in one line. The first value is the border thickness. The second value is the border type. The third value is the border color. The color can be hex, rgb or semantic code.

The border property can also be broken down into separate lines using border-width, border-style and border color. Additionally, you can select very specifically which border you want to style by using border-top, border-bottom, border-right, border-left.

button {

border: 2px dotted green;

}

div {

border: 1px solid #000000;

}

p {

border-right: 1px groove rgb(100,100,100);

border-left: 1px groove rgb(200,200,200);

}

copy

### **border-radius**

This property allows the corners of your border to be given a rounded appearance. The values can be set using px or %.

button {

border-radius: 5px;

}

copy

# The Box Model

\*Copy of the code demonstrated above can be downloaded [here (layouts.zip)](https://s3.amazonaws.com/General_V88/boomyeah/company_209/chapter_2135/handouts/chapter2135_3329_layouts.zip).

All web layouts are accomplished with block elements. Designers use blocks, most often <div> elements, to create rectangular areas into which all content fits. There are only three rules:

* **Total area**: the space an element occupies and affects.
* **Float, clear and overflow**
* **Nested elements**

## **Total Area**

Total width is how much horizontal space a block occupies. This includes the block’s margin, border, and padding. Calculating width, padding, and margin is often the biggest headache for designers, but it's easy to see how they work if you use the **box model**.

The box model consists of the properties **margin, border,** and **padding**.

A picture containing chart

Description automatically generated

Margin is **outside** block elements, while padding is **within** them. This means that we use margin to separate a block from things around it, and padding to move a block's content away from its edges.

We can specifically set the margin, padding, or border of any side of an element.

div{

padding-top: 10px;

padding-right: 10px;

padding-bottom: 10px;

padding-left: 10px;

}

copy

You can also use the shorthand property:

div{

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 100px;

}

copy

is equivalent to:

div{

padding: 25px 50px 75px 100px;

}

copy

And

div{

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 50px;

}

copy

is equivalent to:

div{

padding: 25px 50px 75px;

}

copy

And

div{

padding-top: 25px;

padding-right: 50px;

padding-bottom: 25px;

padding-left: 50px;

}

copy

is equivalent to:

div{

padding: 25px 50px;

}

copy

And

div{

padding-top: 25px;

padding-right: 25px;

padding-bottom: 25px;

padding-left: 25px;

}

copy

is equivalent to:

div{

padding: 25px;

}

copy

(The order flows clockwise, top -> right -> bottom -> left.)

Now, according to the box model, **the total width of an element is:**

**(Margin x 2) + (Border x 2) + (Padding x 2) + Content Width**

Calculating the height is trickier. Why? Because **vertical margins collapse**.

Ex:

HTML:

<div id="box-1">

</div>

<div id="box-2">

</div>

<div id="box-3">

</div>

copy

CSS:

#box-1, #box-2, #box-3{

height: 100px;

width: 100px;

background-color: red;

}

#box-1{

margin: 20px;

}

#box-2{

margin: 30px;

}

#box-3{

margin: 40px;

}

copy

You might think that this will look like this:

Treemap chart

Description automatically generated with medium confidence

But this code will actually result in this:

Chart, treemap chart

Description automatically generated

When the vertical margins of two elements are touching, **only the margin of the element with the largest margin value will be honored**, while the margin of the element with the smaller margin value will be collapsed to zero.

There are other situations where elements do not have their margins collapsed:

* floated elements
* absolutely positioned elements
* inline-block elements
* elements with overflow property set to anything other than visible (They do not collapse margins with their children.)
* cleared elements (They do not collapse their top margins with their parent block’s bottom margin.)

# Inline and Block

* Learn how display: inline; and display: block; differ
* Learn about display: inline-block;

We may have noticed when working with our HTML that some elements allow other elements next to them and others take all of the space next to them regardless of how much content they have. The ones that take up all of the space are called "block" and the ones that share space are called "inline".

<!-- some block elements -->

<h1>Hello World</h1>

<h2>Goodbye World</h2>

<!-- some inline elements -->

<label>First Name:</label>

<input type="text" name="first\_name">

<label>Last Name:</label>

<input type="text" name="last\_name">copy

# Hello World

## **Goodbye World**

First Name:  Last Name:

Every HTML element has a default display property of either "block" or "inline", but we aren't forced to just stick with the default as we can change it in our CSS.

h1, h2 {

display: inline;

background-color: cyan;

}

label, input {

display: block;

}copy

# Hello World

## **Goodbye World**

First Name:Last Name:

### **Inline-Block**

If we want to be able to change the width and allow the elements to be next to one another we have a third option "inline-block".

Inline seems really cool, but inline elements will shrug off our attempts to apply width to them.

h1, h2 {

display: inline;

background-color: cyan;

width: 200px; /\* this won't work!!! \*/

}copy

Let's try this instead!

h1, h2 {

display: inline-block;

background-color: cyan;

width: 200px; /\* this will work! \*/

}copy

# Hello World

## **Goodbye World**

We can make use of this to create columns on our webpages like so...

<p>Lorem ipsum dolor sit amet.</p>

<p>Lorem ipsum dolor sit amet.</p>copy

p {

display: inline-block;

width: 160px;

box-sizing: border-box;

padding: 20px;

background-color: lightcyan;

margin: 10px;

}copy

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.

**NOTE:** we need to ensure that the width of each individual column is narrow enough they will fit within their parent element... often a div or the body tag. Just applying inline-block won't ensure the elements are the size we want.

What if we have columns that don't all have the same amount of content?

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Maiores quas quis, nemo dicta explicabo modi amet tenetur corrupti iste ducimus, aut totam.</p>copy

p {

display: inline-block;

width: 200px;

box-sizing: border-box;

padding: 20px;

background-color: lightgreen;

margin: 10px;

}copy

Without vertical-align

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Maiores quas quis, nemo dicta explicabo modi amet tenetur corrupti iste ducimus, aut totam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Notice how they don't all line up at the top but instead seem to line up at the bottom. This is the default vertical-align behavior for inline-block and we can fix it by applying a different setting... vertical-align: top. While we could try fixing this with margins or use of the position property, vertical-align is the best approach.

With vertical-align: top;

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Maiores quas quis, nemo dicta explicabo modi amet tenetur corrupti iste ducimus, aut totam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

What if we have elements that are traditionally inline like this collection of links.

<a href="#">Home</a>

<a href="#">About</a>

<a href="#">User Profile</a>

<a href="#">Contact Us</a>copy

a {

text-decoration: none;

color: darkslategrey;

background-color: blanchedalmond;

padding: 10px;

margin: 10px;

}copy

Normally anchor tags will display as inline elements... if we make them block elements they can stack one on top of each other.

With display: inline;[Home](https://login.codingdojo.com/m/283/8968/60804) [About](https://login.codingdojo.com/m/283/8968/60804) [User Profile](https://login.codingdojo.com/m/283/8968/60804) [Contact Us](https://login.codingdojo.com/m/283/8968/60804)

If we add on display:block; to our a tag styling...

With display: block;[HomeAboutUser ProfileContact Us](https://login.codingdojo.com/m/283/8968/60804)

Block, inline, and inline-block are basic ways we can alter the normal layout of the HTML tags on our page. Let's test our new ability to create custom layouts in the next assignment.

# The Flex Advantage

### **Objectives**

* Learn how display: flex; is applied to a parent element to style its children
* Learn how display: flex; ignores whitespace in our **HTML** when positioning elements

We have shown one way to change the position of the elements on our pages using the display property with inline, block, and inline-block. While we can certainly attain whatever layout we want using that approach there are newer and perhaps more convenient/powerful options we can use.

One thing we may have noticed laying out our website using inline-block is how sensitive it is to the hard-coded sizes we apply to it. While we can include percentages sometimes there is phantom extra space that is added into the layout that can be hard to deal with. Consider the following three columns of text.

<div class="container">

<div class="col">

<h3>Lorem</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

<div class="col">

<h3>Ipsum</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

<div class="col">

<h3>Dolor</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

</div>

copy

.container {

width: 720px;

height: 250px;

background-color: silver;

}

.col {

width: 220px;

padding: 10px;

background-color: blanchedalmond;

display: inline-block;

}

copy

Default Indentation

### **Lorem**

Lorem ipsum dolor sit amet consectetur.

### **Ipsum**

Lorem ipsum dolor sit amet consectetur.

### **Dolor**

Lorem ipsum dolor sit amet consectetur.

### **Trimming the whitespace**

When using inline-block to have our columns display next to each other, the whitespace (extra spaces, tabs, and newlines) all count as a blank "text" element that takes up space. If we rewrite our code so the next div begins after the previous div this will eliminate this extra space.

<div class="container">

<div class="col">

<h3>Lorem</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div><div class="col">

<h3>Ipsum</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div><div class="col">

<h3>Dolor</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

</div>

copy

.container {

width: 720px;

height: 250px;

background-color: silver;

}

.col {

width: 220px;

padding: 10px;

background-color: blanchedalmond;

display: inline-block;

}

copy

Whitespace removed between column divs

### **Lorem**

Lorem ipsum dolor sit amet consectetur.

### **Ipsum**

Lorem ipsum dolor sit amet consectetur.

### **Dolor**

Lorem ipsum dolor sit amet consectetur.

### **Using Flex**

If we change how we write the code to use display: flex; on the parent "container" div then we can write our **HTML** with our usual indentation and we can avoid having the newline and tab between divs take up space.

<div class="container">

<div class="col">

<h3>Lorem</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

<div class="col">

<h3>Ipsum</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

<div class="col">

<h3>Dolor</h3>

<p>Lorem ipsum dolor sit amet consectetur.</p>

</div>

</div>

copy

.container {

width: 720px;

height: 250px;

background-color: silver;

display: flex;

align-items: center;

}

.col {

width: 220px;

padding: 10px;

background-color: blanchedalmond;

}

copy

Using display: flex; with default indentation

### **Lorem**

Lorem ipsum dolor sit amet consectetur.

### **Ipsum**

Lorem ipsum dolor sit amet consectetur.

### **Dolor**

Lorem ipsum dolor sit amet consectetur.

**Note:** the align-items: center; will prevent the columns from stretching (inheriting the height of the parent container div). Try out other values like flex-start and flex-end.

# Justify the Content

### **Objectives**

* Learn about align-items
* Learn about the different values we can apply to justify-content

Flex can also be used to position smaller elements (like images or boxes) within a larger element. Properties like align-items and justify-content can be used to position the elements exactly where we want them within some larger element.

## **Flex Basics - Ninja Example**

This **HTML** will stay in common with all of the following examples

The div with an id of #dojo will have a fixed size for each of the demos

<div id="dojo">

<img src="blue.png" alt="blue ninja" >

<img src="green.png" alt="green ninja" >

<img src="purple.png" alt="purple ninja" >

<img src="red.png" alt="red ninja" >

</div>copy

#dojo {

width: 400px;

height: 200px;

background-color: grey;

}copy

blue ninja green ninja purple ninja red ninja

## **Applying Display Flex**

At first when we apply display: flex; the ninjas will appear distorted. Don't worry this is normal and we can fix this!

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex; /\* this line is new! \*/

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Fixing the stretch**

When align-items is set it will "unstretch" our ninjas!

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center; /\* this line is new! \*/

}copy

blue ninjagreen ninjapurple ninjared ninja

The justify-content property can be applied to the parent to affect the child elements. In this case when we use justify-content on the #dojo it will reposition the ninjas!

## **Justify Content - Flex-Start**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: flex-start; /\* this line is new \*/

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Justify Content - Center**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: center;

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Justify Content - Flex-End**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: flex-end;

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Justify Content - Space Between**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: space-between;

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Justify Content - Space Evenly**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: space-evenly;

}copy

blue ninjagreen ninjapurple ninjared ninja

## **Justify Content - Space-Around**

#dojo {

width: 400px;

height: 200px;

background-color: grey;

display: flex;

align-items: center;

justify-content: space-around;

}copy

blue ninjagreen ninjapurple ninjared ninja

There are many different values we can use with justify-content. When you need it try seeing how each of them affect the children. For more background on justify-content check out this fantastic [CSS Tricks Article](https://css-tricks.com/almanac/properties/j/justify-content/).

# Being Flex-ible

### **Objectives**

* Learn about display: flex;
* Learn about the flex: 1; shorthand

## **The Flex Shorthand**

One of the most useful things we can do when using `display: flex;` is to make use of a shorthand property called flex!

This property combines: flex-grow, flex-shrink, flex-basis and can provide an easy way for us to create columns in our layout.

<div class="row">

<div class="col">Column</div>

<div class="col">Column</div>

</div>copy

.row {

display: flex;

}

.col {

flex: 1;

padding: 20px;

margin: 10px;

background-color: lightcyan;

}copy

Think of each element with flex: 1; like one slice of a pie. If another column were to have flex: 2, it will essentially take 2 slices of the available space. For a live demonstration of this, see the embedded demonstration below. Try using the "Add Column" and "Remove Column" buttons and seeing how the columns resize themselves.

## **Flexible Math**

What would happen if we mixed in a div with the following "col-2" class? What about classes with flex: 3; or higher?

.col-2 {

flex: 2;

padding: 20px;

margin: 10px;

background-color: palegreen;

}copy

When we mix how many "slices of the pie" each of our columns ask for then we'll be able to mix in some larger columns. In those cases we'll need to add up the "slices" that are being asked for and use that at the basis for deciding the fraction of the row each column will receive.

Chart, pie chart

Description automatically generated

What's best with all of this is the ease in setting up even columns as we don't need to figure out the exact width in pixels each column needs to take up and we don't have to factor in any padding, borders, or margins in those calculations.

Read more about the flex shorthand at the fantastic [CSS tricks](https://css-tricks.com/understanding-flex-grow-flex-shrink-and-flex-basis/).

# CSS Resets

Different browsers may give default attributes to HTML elements, causing sites to look different based on the version of the browser or whether you're seeing the site in IE, Firefox, Chrome, etc.  For example in Internet Explorer, an H1 element may have certain margin and padding causing your site to look one way, while Chrome and Firefox may give the H1 a slightly different margin/padding.  These differences can cause a headache, especially if you're trying to make your site look consistent throughout different browsers/versions.

To solve many cross-browser issues, it's encouraged that you use 'RESET' and 'NORMALIZE' (discussed below) and, basically, define what the default attributes should be (instead of relying on the default values that come with the browser).  In fact, **technical recruiters love to ask you about this** and see if you know how to make your site look consistent throughout different browsers.  When they ask you this, you should be able to explain (with confidence) what reset/normalizing means and why they are used.  You should also tell them that in addition to this, it's important to validate your HTML/CSS often, as validating the HTML/CSS gives insight on how you can fix issues that may lead to cross-browser compatibility issues (and often, invalid HTML/CSS can lead to weird behavior as the browser may not 'fix' invalid HTML/CSS quite the right way).

## **Simple CSS reset**

The simplest CSS reset that web developers commonly use is to simply zero out the margin and padding for all visible elements on the page. This will make our designs consistent from one browser to another and allows us to then specify padding and margin on only the elements where they will be necessary.

\* {

padding: 0;

margin: 0;

}copy

## **More on CSS Resets**

Here is a great explanation of what CSS resets are, why we use them, and some examples:

<http://perishablepress.com/a-killer-collection-of-global-css-reset-styles/>

# Position Basics

### **Objectives**

* Learn what the position property does
* Learn about position: absolute;, position: relative;, and position: fixed;

Use the code below the video to follow along to this demo of the position property.

### **Code from the video**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>JS Bin</title>

</head>

<body>

<div class="container">

<h1>Title</h1>

<div id="message">

<div class="header"><span>I am an Alert!</span><span>×</span></div>

<p>This is some arbitrary alert text!, don't worry about what it actually says!</p>

<button>OK</button>

</div>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laborum deleniti saepe odit pariatur delectus repudiandae atque illum aperiam possimus numquam vel, quo ducimus nulla est, id sunt optio eius voluptas. Repudiandae commodi temporibus aliquid illum beatae eveniet unde cum officia rem possimus aut cupiditate, sint rerum accusantium itaque explicabo tenetur voluptate necessitatibus corrupti provident praesentium?</p>

</div>

</body>

</html>copy

\* {

margin: 0;

padding: 0;

font-family: sans-serif;

}

.container {

width: 800px;

margin: 0 auto;

padding: 20px;

box-shadow: 4px 4px 4px grey;

}

h1 {

font-size: 50px;

padding-bottom: 10px;

margin-bottom: 20px;

border-bottom: 1px solid #ccc;

}

p {

margin: 20px 0;

font-size: 1.1rem;

}

#message {

width: inherit;

border: 1px solid #eee;

border-radius: 10px;

background-color: #fff;

box-shadow: 4px 4px 4px grey;

/\* change the position properties here \*/

position: fixed;

bottom: 80px;

}

#message \* {

margin: 10px;

}

#message .header {

margin: 0px;

background-color: #ddd;

padding: 10px;

font-weight: 900;

display: flex;

justify-content: space-between;

}

#message button {

height: 30px;

width: 80px;

}copy