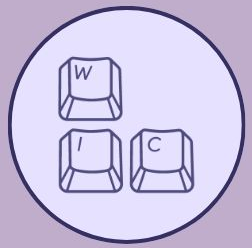
Intro to CSS

Fall 2024

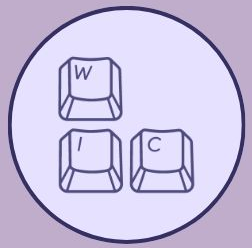
Event Type: Tech Dev

Agenda

- What is CSS?
- Applying CSS
- Simple Selectors
- Combinator Selectors
- User-Action Pseudo-Classes
- CSS Box Model
- CSS Positioning
- CSS Flexbox Layout



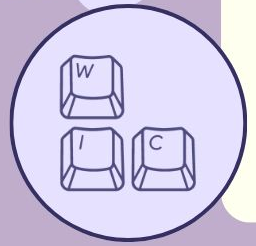
Access the Code



The folder with the starter code and the slides are in the WIC Discord in the **#tech-dev channel**

https://github.com/WIC-UCSD-Project-Teams/css-workshop

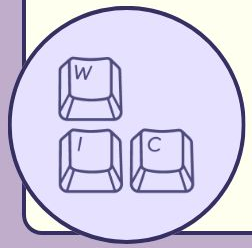
What is CSS?



https://russmaxdesign.github.io/maxdesign-slides/02-css/203-css-rulesets.html#/

What is CSS?

- CSS is the language we use to **style a Web page**.
- CSS stands for **Cascading Style Sheets.**
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS is written using a series of rules - sometimes called rulesets or statements.

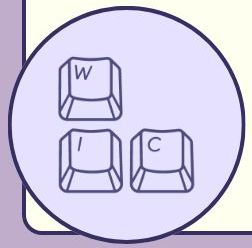


Vocabulary: Selector

- The selector **selects the elements** in an HTML document that you want to target for styling.
- In the following example, the p selector will select **every paragraph element** in the HTML document.

```
/* Selector */
p {
  color: red;
}
```

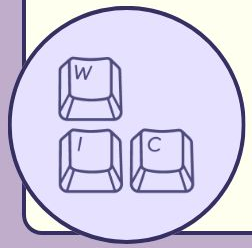
```
<!doctype html>
<html>
  <head>
    <title>Document title</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>A paragraph element.</p>
    <p>A second paragraph element.</p>
  <body>
</html>
```



Vocabulary: Declaration Block

- The **declaration block** is a container that consists of anything between (and including) the start brace "{" and end brace "}".

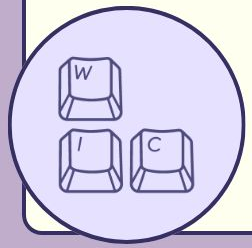
```
/* Declaration block */
p {
  color: red;
}
```



Vocabulary: Declaration

- The **declaration** consists of a property and a value. It tells browsers how to render any element that is selected.

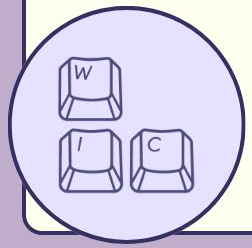
```
/* Declaration */
p {
  color: red;
}
```



Vocabulary: Property

- The **property** defines what aspect of the element will be styled.
- In the following example, the color property **will be styled.**

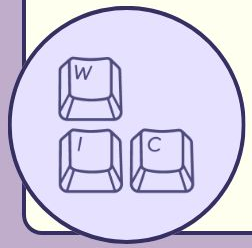
```
/* Property */
p {
  color: red;
}
```



Vocabulary: Value

- The **value** is the exact style you wish to set for the property. In the following example, all paragraph elements will be styled with a value of red.

```
/* Value */
p {
  color: red;
}
```

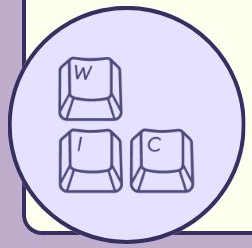


Vocabulary: Comments

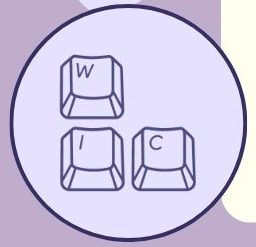
- You can **insert comments in CSS**. CSS comments must begin with a forward slash and an asterisk `"/*"` and end with an asterisk and a forward slash `"*/"`.

```
/* a CSS comment used in CSS files */
```

```
/* --------------------
HEADER STYLES
-------------------- */
```



Applying CSS

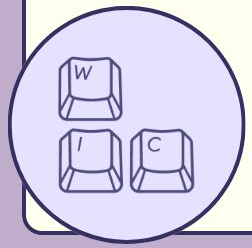


https://russmaxdesign.github.io/maxdesign-slides/02-css/204-applying-css.html#/

Inline Styles

- **Inline styles** are applied to elements directly in the HTML markup using the style attribute.
- Inline styles **should be avoided** as they are inefficient.
- For example, if you wanted to style every p element in a document, you would need to use **multiple inline styles**.

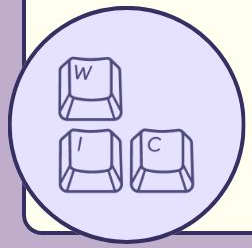
```
<html>
  <head>
    <title>Document title</title>
  </head>
  <body>
    <h2 style="color: red;">
      Heading here
    </h2>
  <body>
</html>
```



Header Styles

- **Header styles** are written inside the `<head>` element of HTML documents.
- **CSS rules** are added inside the `<style>` element.
- Like inline styles, header styles **should be avoided on any real website** as they are inefficient and hard to maintain.

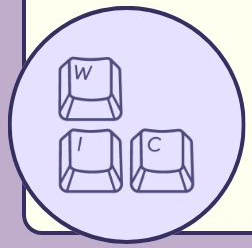
```
<html>
  <head>
    <title>Document title</title>
    <style>
      h2 { color: blue; }
    </style>
  </head>
  <body>
  <body>
</html>
```



External Style Sheets

- **External style sheets** are separate CSS documents that can be “linked to” from HTML documents using the <link> element.
- Authors can link to **more than one CSS document** from the HTML document.
- External style sheets are **far more efficient and effective** than inline or header styles as you can use a single CSS file for multiple HTML documents.

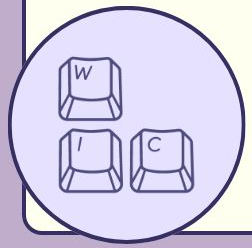
```
<html>
  <head>
    <title>Document title</title>
    <link rel="stylesheet" href="a.css">
  </head>
  <body>
  <body>
</html>
```



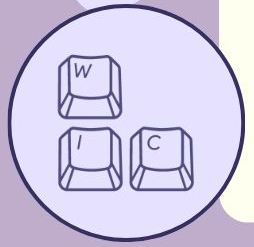
External Style Sheets

- The `rel` attribute is **required** within the `<link>` element.
- It is used to **define the relationship** from the current HTML document to the linked file.
- In this case `rel="stylesheet"` informs the browser that **the file being linked to is a stylesheet**.

- The `href` attribute is also **required** within the `<link>` element.
- It is used to **define the location** of the linked document.



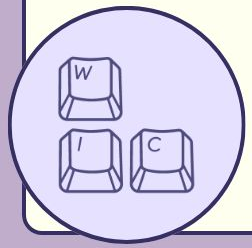
Simple Selectors



https://russmaxdesign.github.io/maxdesign-slides/02-css/205-01-selectors-simple.html#/

Simple Selectors

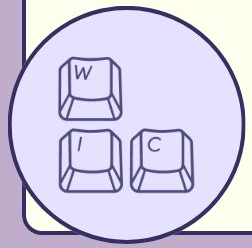
- **Simple selectors** allow you to target HTML elements directly, as well as targeting elements that contain class or ID attributes.



Type Selectors

- The **type selector** is written using an element type.
- The type selector **targets every instance of the element type** regardless of their position in the document tree.

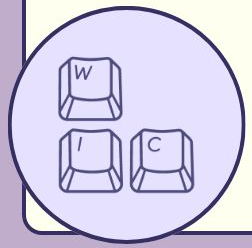
```
/* syntax */
E { }

/* example */
h1 { }
```



Type Selectors

- In the following example, the `p` selector targets **any instance of the `<p>` element**.

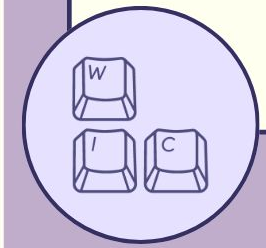
```
/* CSS selector */
p { }

<!-- HTML markup -->
<p></p>
<div></div>
<p></p>
```



Exercise: Write Some Type Selectors

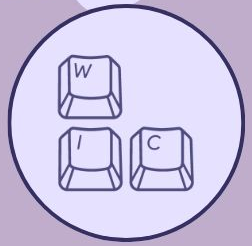
Go to the exercise-205-01-01.html file

- **Write a CSS rule** to set the <h1> element to color: blueviolet.
- **Write a CSS rule** to set the <b> element to color: chocolate.
- **Write a CSS rule** to set the element to color: cornflowerblue.
- **Write a CSS rule** to set the <blockquote> element to color: darkcyan.



Exercise Answer

```
/* Add styles here */
h1 { color: blueviolet; }
b { color: chocolate; }
em { color: cornflowerblue; }
blockquote { color: darkcyan; }
```



Exercise 205-01-01: Write some type selectors

Lorem ipsum dolor sit amet, **consectetur adipisicing** elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, *repudiandae error architecto* a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

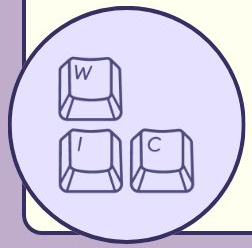
> This is a blockquote.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Class Selectors

- The **class selector** is written using an optional element, followed by a ".", followed by the class name.

```
/* syntax */
.class-name { }
E.class-name { }

/* example */
.intro { }
```



Class Selectors

- Whitespace is **not permitted** within the selector.
- Class values are **case-sensitive**. In the following example, browsers interpret .intro and .Intro as different classes.
- Classes can begin with letters, **digits, hyphens and underscores**. They can also include colons and periods.
- However, if class values start with a number or a special character, they **must be escaped** when writing CSS selectors.

```
/* Valid */
.class-name { }

/* Invalid */
. class-name { }
```

```
/* case sensitive classes */
.Intro { }
.intro { }
```

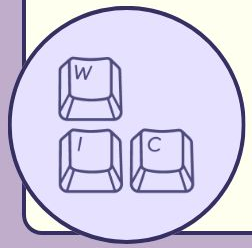
```
/* escaped number */
.\34 80wide { }

<!-- HTML markup -->
<div class="480wide"></div>
```

```
/* class names */
.active { }
.480wide { }
.-members { }
._classname { }
```

Class Selectors

- The class selector targets any HTML element **that has the relevant class value.**
- In the following example, the .intro selector targets any element that **contains a class** of intro.

```
/* CSS selector */
.intro { }

<!-- HTML markup -->
<p class="intro"></p>
<div class="intro"></div>
```

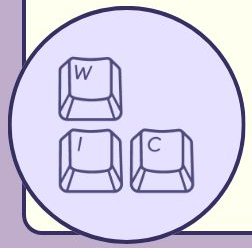


Class Selectors

- Elements can be added before the "." to make a selector more specific.
- In the following example, the `p.intro` selector **targets only** `<p>` elements that contain a class of `intro`. The `<div>` with a class of `intro` is not targeted.

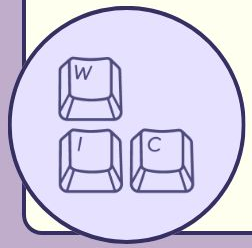
```
/* CSS selectors */
div.intro { }
p.intro { }
```

```
/* CSS selector */
p.intro { }

<!-- HTML markup -->
<div class="intro"></div>
<p class="intro"></p>
```



Class Selectors

- **Classes can be added together** to make a selector more specific. The joined class selectors cannot contain whitespace.
- In the following example, the .one.two selector **targets any element that has both class attribute values** of one and two.
- The class attribute values within the HTML document can be **written in any order**. The value one two is the same as two one.
- Multiple classes can also be written in any order within the CSS. The selector .one.two is the **same as** .two.one.



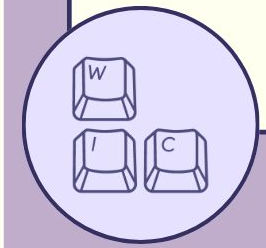

```
<!-- HTML markup -->
<p class="one"></p>
<p class="one two"></p>
<p class="two"></p>
```

```
/* CSS selector */
.one.two { }

<!-- HTML markup -->
<p class="one two"></p>
<p class="two one"></p>
```

Exercise: Write Some Class Selectors

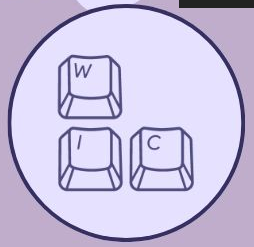
Go to the exercise-205-01-02.html file

- **Write a CSS rule** to style the class of intro with a color of steelblue.
- **Write another CSS rule** to target only <p> elements with a class of intro. Set the color to violet.



Exercise Answer

```
/* Add styles here */
.intro {
  color: steelblue;
}

p.intro {
  color: violet;
}
```



Exercise 205-01-02: Write some class selectors

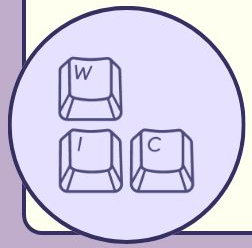
Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos?
Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam,
consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos?
Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam,
consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

ID Selectors

- The **ID selector** is written using an optional element, followed by a "#", followed by the ID name.

```
/* syntax */
#id-name { }
E#id-name { }

/* example */
#sidebar { }
```



ID Selectors

- Whitespace is **not permitted** within the selector.
- ID names are **case-sensitive**. In the following example, browsers will interpret #nav and #Nav as different IDs.
- ID attribute values are allowed to have any characters or strings, as long as the value is **unique** and does not contain whitespace.
- However, if ID values start with a number or a special character, these characters or numbers **must be escaped.**

```
/* Valid */
#id-name { }

/* Invalid */
# id-name { }
```

```
/* Case sensitive */
#nav { }
#Nav { }
```

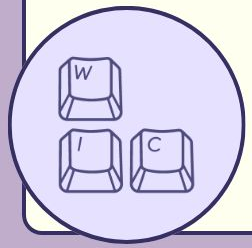
```
/* escaped number */
#\34 80wide { }

<!-- HTML markup -->
<div id="480wide"></div>
```

```
/* ID names */
#active { }
#480wide { }
#-members { }
#_classname { }
```

ID Selectors

- The ID selector targets any HTML element **that has the relevant ID value.**
- In the following example, the #nav selector targets **any instance** of an element that contains an ID of nav.

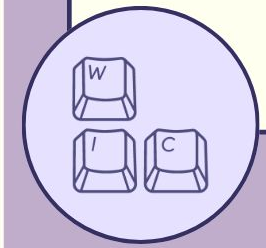
```
/* CSS selector */
#nav { }

<!-- HTML markup -->
<p id="nav"></p>
```



Exercise: Write Some ID Selectors

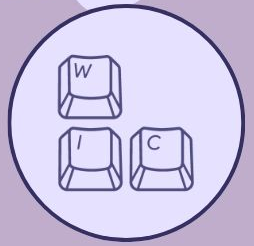
Go to the exercise-205-01-03.html file

- **Write a CSS rule** to style the element with an ID of title with color: darkviolet.
- **Then write another CSS rule** to style the element with an ID of pullout with color: deeppink.



Exercise Answer

```
/* Add styles here */
#title {
  color: darkviolet;
}

#pullout {
  color: deeppink;
}
```



Exercise 205-01-03: Write some ID selectors

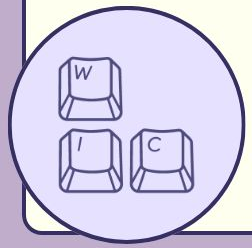
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Universal Selectors

- The **universal selector** is written using a "*".

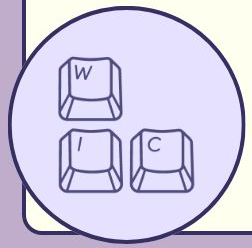
```
/* syntax */
* { }
```



Universal Selectors

- The universal selector targets **all elements within the document.**
- In the following example, the universal selector targets **every element in the document.**

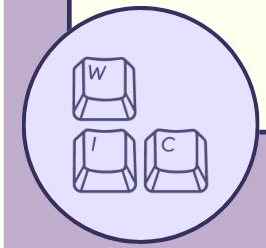
```
/* CSS selector */
* { }

<!-- HTML markup -->
<body>
  <div></div>
</body>
```



Exercise: Write the Universal Selector

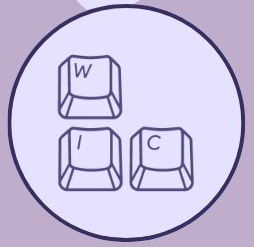
Go to the exercise-205-01-04.html file

- **Write a CSS rule** to style all elements in the document using the universal selector. Use color: darkmagenta.



Exercise Answer

```
/* Add styles here */
* {
  color: darkmagenta;
}
```

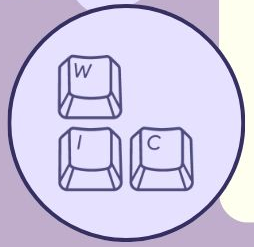


Exercise 205-01-04: Write the universal selector

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

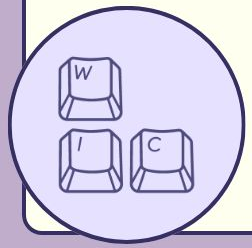
Combinator Selectors



https://russmaxdesign.github.io/maxdesign-slides/02-css/205-02-selectors-combinator.html#/

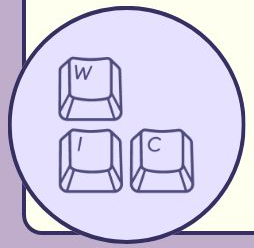
Combinator Selectors

- **Combinators** allow you to combine individual selectors into new types of selectors.



Descendant Combinators

- **Descendant combinators** are written using two or more selectors separated by whitespace.
- Descendant combinators target **only elements that are descendants of other elements.**
- In the following example, the `<a>` that is a **descendant** of the `<p>` will be selected, but not the `<a>` that is a descendant of the `<div>`.



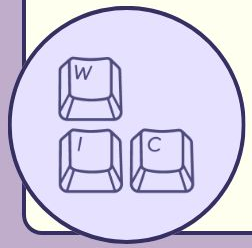

```
/* syntax */
E F { }

/* example */
ul li a { }
```

```
/* CSS selector */
p a { }

<!-- HTML markup -->
<p>
  <a></a>
</p>
<div>
  <a></a>
</div>
```

Descendant Combinators

- The key to descendant combinators is understanding **paths to elements**. There is a path to every element starting with the `<html>` element.
- Paths can be **written in full or in part to the element** or a partial path, depending on your need.

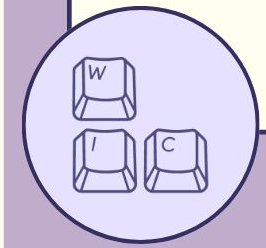
```
/* example paths */
html body .container .nav ul li a { }
body .container .nav ul li a { }
.container .nav ul li a { }
.nav ul li a { }
ul li a { }
ul a { }
li a { }
a { }
```



Exercise: Write a Descendant Combinator

Go to the `exercise-205-02-01.html` file

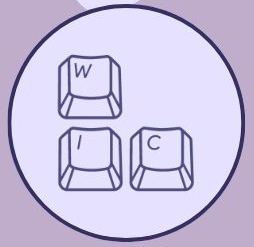
- **Write a CSS rule** to style the `<a>` element to be `color: firebrick`**.** However, **only style the `<a>` element inside the `<p>` element** (Link 2), not the `<a>` element inside the `<div>` element (Link 1).




```
<div>
  <a href="#">Link 1</a>
</div>
<p>
  <a href="#">Link 2</a>
</p>
```

Exercise Answer

```
/* Add styles here */
p a {
    color: firebrick;
}
```



Exercise 205-02-01: Write a descendant combinator

Link 1
Link 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

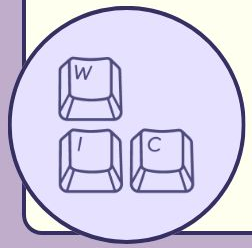
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Child Combinators

- **Child combinators** are written using two selectors separated by a ">".
- Child combinators target **any element that is a direct child of another element.**
- **Only child elements**, rather than descendant elements, will be selected.
- In the following example, the <a> that is **a child** of the will be selected, but not the <a> that is a descendant of the .

```
/* syntax */
E > F { }

/* example */
p > a { }
```

```
/* CSS Selector */
div > a { }

<!-- HTML markup -->
<div>
  <a></a>
</div>
<div>
  <p> <a></a> </p>
</div>
```

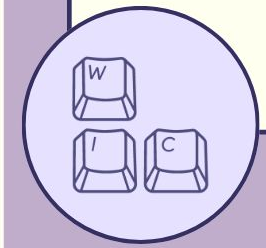


Exercise: Write a Child Combinator

Go to the exercise-205-02-02.html file

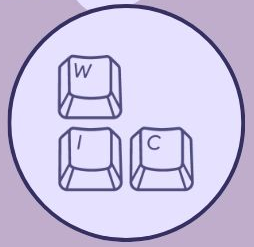
- **Write a CSS rule** to style the <a> element to be color: forestgreen. However, **only style the <a> that is a child of the element** (Link 1), not the <a> that is a descendent of the element (Link 2).

```
<div>
  <a href="#">Link 1</a>
</div>
<div>
  <p>
    <a href="#">Link 2</a>
  </p>
</div>
```



Exercise Answer

```
/* Add styles here */
div > a {
  color: forestgreen;
}
```



Exercise 205-02-02: Write a child combinator

Link 1
Link 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

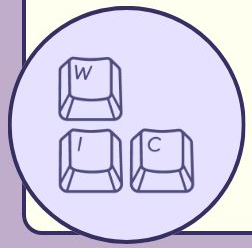
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Adjacent Sibling Combinators

- **Adjacent sibling combinators** are written using two selectors separated by a `"+"`.
- Adjacent sibling combinators target **the sibling immediately following a defined element.**
- In the following example, only the `<h3>` **that is adjacent to** (or comes directly after) the `<h2>` will be selected.

```
/* syntax */
E + F { }

/* example */
h2 + h3 { }
```

```
/* CSS selector */
h2 + h3 { }

<!-- HTML markup -->
<h2>Content</h2>
<h3>Content</h3>
<h3>Content</h3>
```

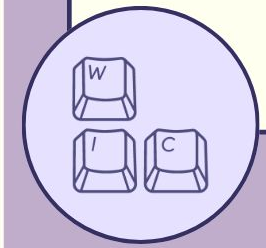


Exercise: Write an Adjacent Sibling Combinator

Go to the `exercise-205-02-03.html` file

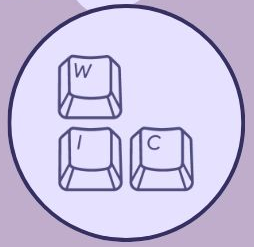
- **Write a CSS rule** to style the `<h3>` element to be `color: hotpink`. However, **only style the** `<h3>` that comes directly after the `<h2>` element.

```
<h2>Heading level 2</h2>
<h3>Heading level 3</h3>
<h3>Heading level 3</h3>
```



Exercise Answer

```
/* Add styles here */
h2 + h3 {
  color: hotpink;
}
```



Exercise 205-02-03: Write an adjacent sibling combinator

Heading level 2

Heading level 3

Heading level 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

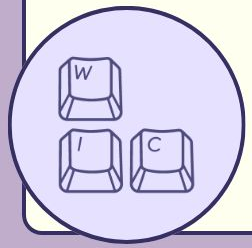
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

General Sibling Combinators

- **General sibling combinators** are written using two selectors separated by a "~".
- General sibling combinators target **any sibling that follows a defined element.**
- In the following example, **any** <h3> **that appears after** the <h2> in source order will be selected - as long as they share the same parent.

```
/* syntax */
E ~ F { }

/* example */
h2 ~ h3 { }
```

```
/* CSS selector */
h2 ~ h3 { }

<!-- HTML markup -->
<h2>Content</h2>
<h3>Content</h3>
<h3>Content</h3>
```

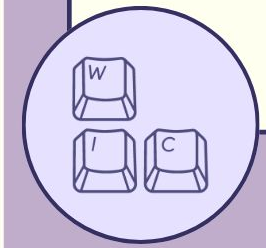


Exercise: Write a General Sibling Combinator

Go to the `exercise-205-02-04.html` file

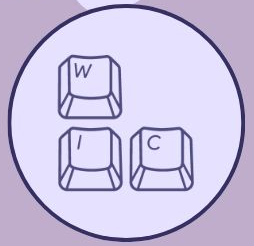
- **Write a CSS rule** to style the `<h3>` element to be `color: lawngreen`. However, **only style all** `<h3>` elements that come after the `<h2>` element, rather than those that come before.

```
<h3>Heading level 3</h3>
<h2>Heading level 2</h2>
<h3>Heading level 3</h3>
<h3>Heading level 3</h3>
```



Exercise Answer

```
/* Add styles here */
h2 ~ h3 {
  color: lawngreen;
}
```



Exercise 205-02-04: Write a general sibling combinator

Heading level 3

Heading level 2

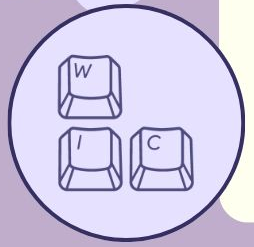
Heading level 3

Heading level 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

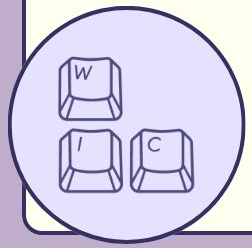
User-Action Pseudo-Classes



https://russmaxdesign.github.io/maxdesign-slides/02-css/205-04-selectors-user-action.html#/

User-Action Pseudo-Classes

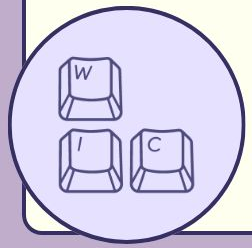
- The **user-action pseudo-classes** allow you to style elements based on the way that users interact with these elements.
- Super useful and commonly used in buttons.



:focus pseudo-class

- The **:focus pseudo-class selector** is written using an element, followed by ":", followed by "focus".
- Whitespace is **not permitted** within the selector.
- The :focus pseudo-class selector targets **elements that have focus** (ones that accept keyboard events).

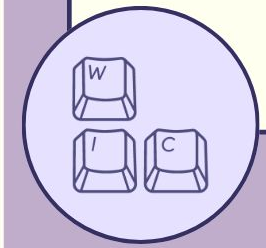
```
/* syntax */
E:focus { }

/* example */
a:focus { }
```



Exercise: Write a :focus Selector

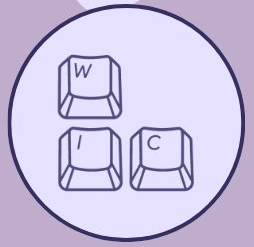
Go to the exercise-205-04-01.html file

- **Write a CSS rule** to style the <a> element to be color: olivedrab, but only when it is in the focus state.
- You will need to use *TAB* keystrokes to focus on the link **to see the :focus state in action**.



Exercise Answer

```
/* Add styles here */
a:focus {
  color: olivedrab;
}
```



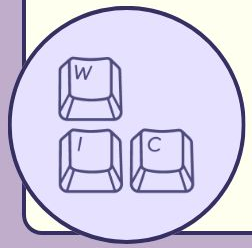
Exercise 205-04-01: Write a :focus selector

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

:hover pseudo-class

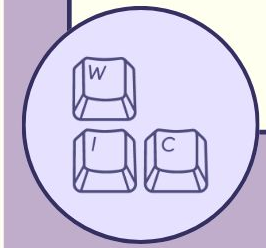
- The **:hover pseudo-class selector** is written using an element, followed by ":", followed by "hover".
- Whitespace is **not permitted** within the selector.
- This selector targets **any element when the user's cursor is over the element**, but the user has not activated it.



Exercise: Write a :hover Selector

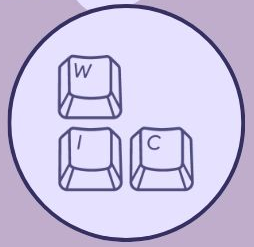
Go to the exercise-205-04-02.html file

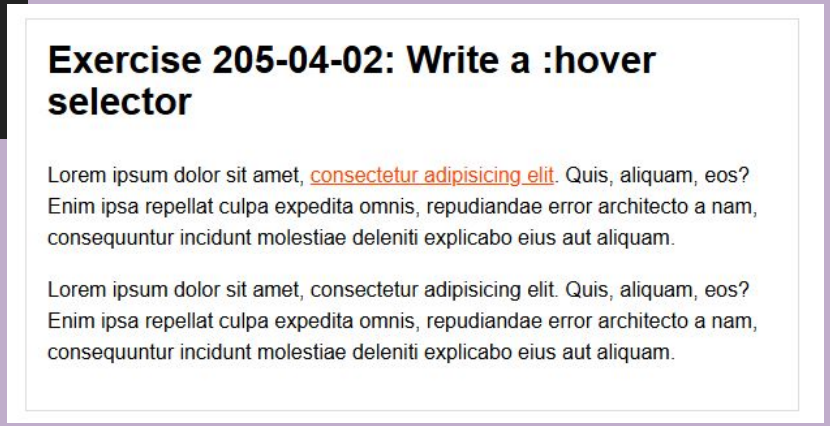
- **Write a CSS rule** to style the <a> element to be color: orangered, but only when it is in the hover state.
- You will need to use hover over the link **to see the :hover state in action**.



Exercise Answer

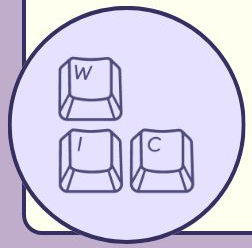
```
/* Add styles here */
a:hover {
  color: orangered;
}
```





:active pseudo-class

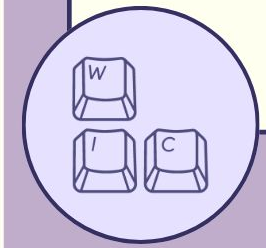
- The **:active pseudo-class selector** is written using an element, followed by ":", followed by "active".
- Whitespace is **not permitted** within the selector.
- The :active pseudo-class selector targets **any element that is currently being activated by the user.**



Exercise: Write an :active Selector

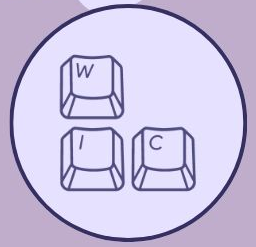
Go to the exercise-205-04-03.html file

- **Write a CSS rule** to style the <a> element to be color: orchid, but only when it is in the active state.
- You will need to click on the link **to see the :active state in action** - just as the link is activated.

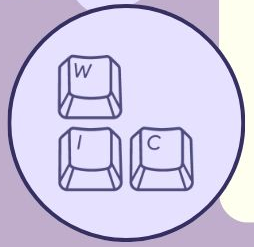


Exercise Answer

```
/* Add styles here */
a:active {
  color: orchid;
}
```



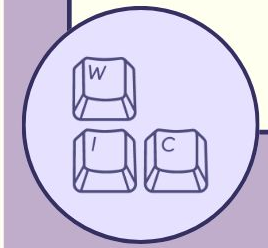
CSS Box Model



https://russmaxdesign.github.io/maxdesign-slides/02-css/208-css-box-model.html#/

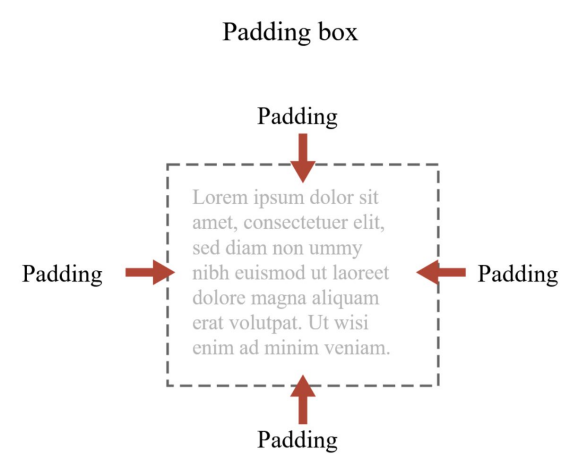
CSS Box Model

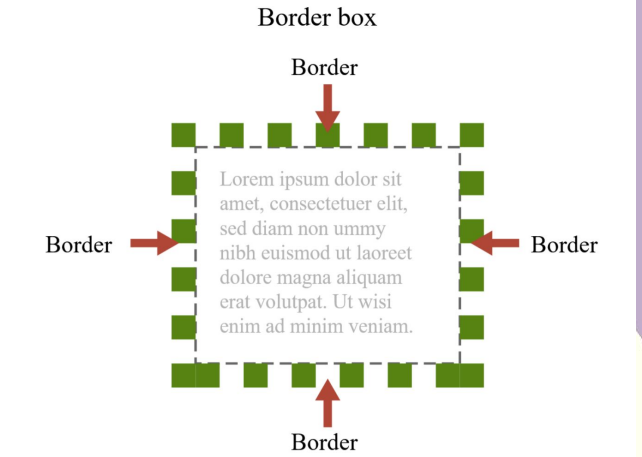
The **CSS box model** describes the rectangular boxes that are created for every element in the document tree.



Content Boxes

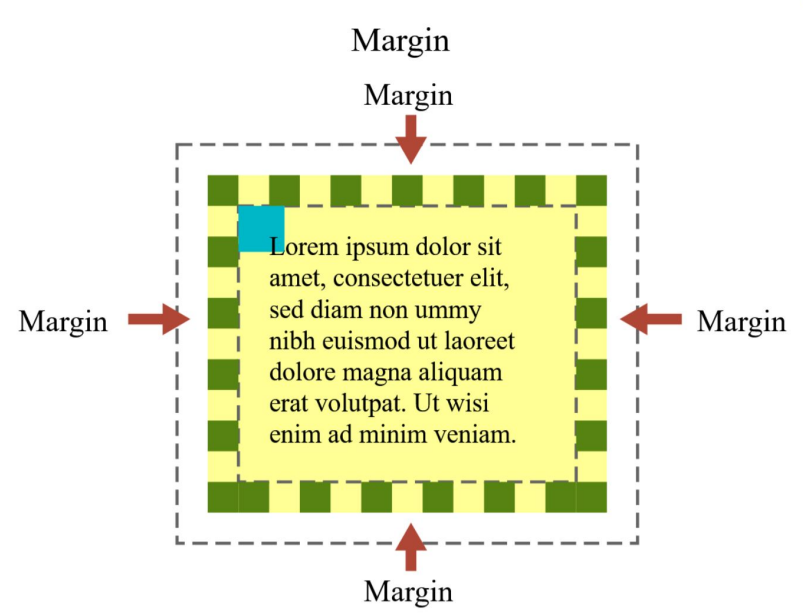
- **Content box:** Invisible box around contents
- **Padding box:** Adds padding around contents
- **Border box:** Describes style of border around padding

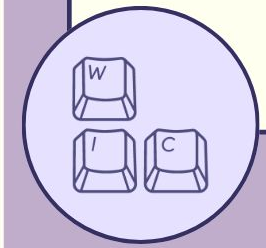




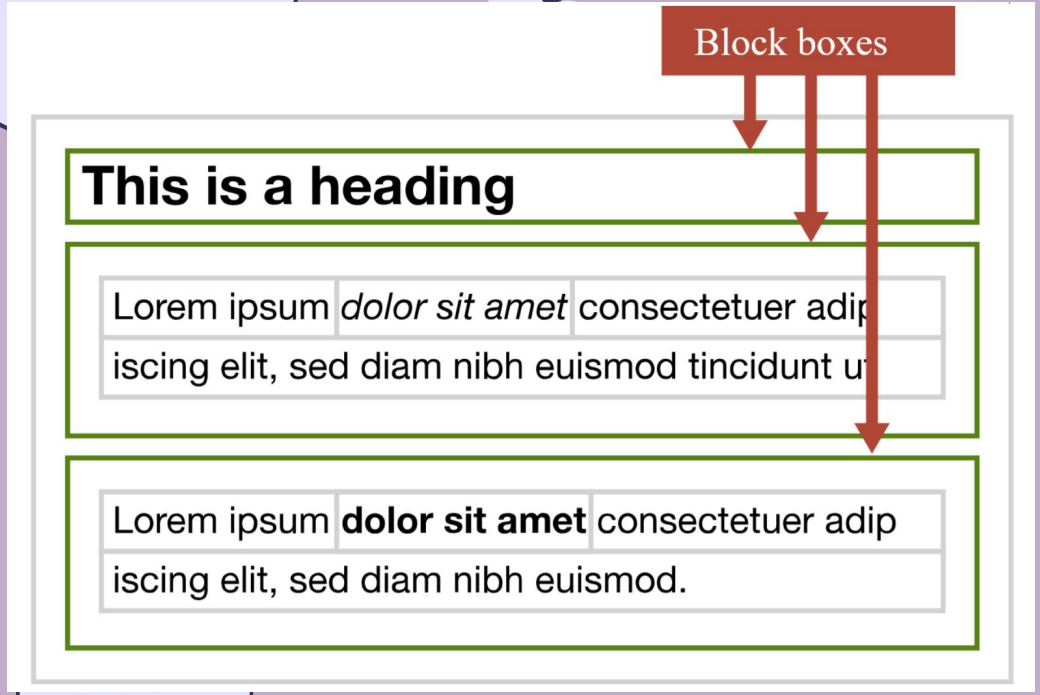
Background and Margins

- **Background color/image:** Up to and including border
- **Margin:** Around outside of border





Block boxes



- **Block boxes** stretch horizontally across the page
- Stack on top of each other
- By default, width is the width of the *containing block* or the *viewport*
- Height is *collapsed*

Block boxes

- Can **specify width/height** by pixel size or percentage
- Or make a block box behave like an inline box
  - I.e. **collapse** the dimensions to the size of the **content**

```
/* Block box with length width */
p {
  width: 200px;
}
```

```
/* Block box with percentage height */
p {
  height: 30%;
}
```

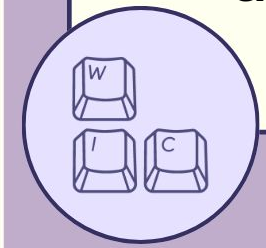
```
/* Block box with percentage width */
p {
  width: 50%;
}
```

```
/* Collapsing a block element */
p {
  display: inline;
}
```

Exercise: Style block elements

Go to the exercise-208-01.html file

1. Write a CSS rule to set a **width of 200px** for the element with a class of **example1**.
2. Write a second CSS rule to set a **width of 60%** for the element with a class of **example2**.
3. Write a third CSS rule to set the element with a class of **example3** to **display: inline**.
4. Write a fourth CSS rule to set a **height of 80px** for the element with a class of **example4**.



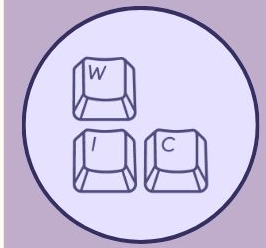
Exercise Answer

```
/* Add styles here */
.example1 {
  width: 200px;
}
```

```
.example2 {
  width: 60%;
}
```

```
.example3 {
  display: inline;
}
```

```
.example4 {
  height: 80px;
}
```



Exercise 208-01: Style some block-level elements

Width

Default width stretched

Example 1: Width sized with a length value

Example 2: Width sized with a percentage value

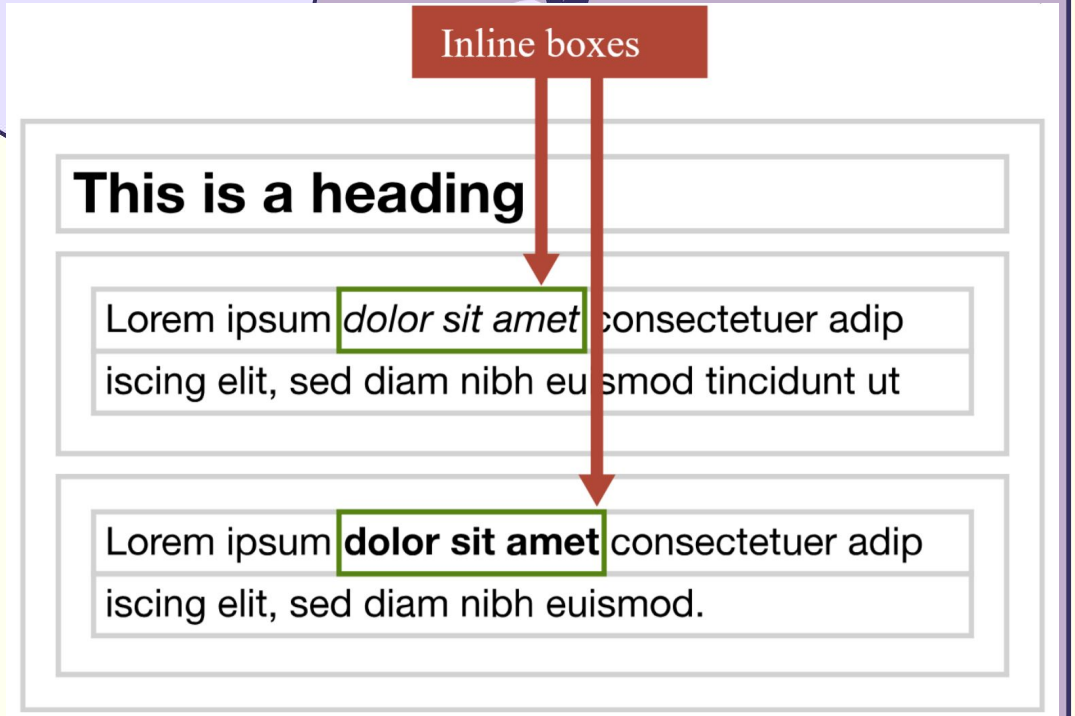
Example 3: Width collapsed via display: inline

Height

Default height collapsed

Example 4: Height sized with a length value

Inline boxes

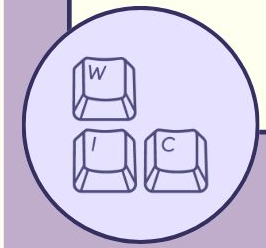


- **Inline boxes** are within one line (can also wrap to the next line)
- Default height and width is *collapsed*
- Margin and padding apply to left and right sides of inline boxes

Exercise: Style inline elements

Go to the exercise-208-02.html file

1. Write a CSS rule to set **padding of 20px** for the element with a class of **example1**.
2. Write a second CSS rule to set a **border of 10px solid red** for the element with a class of **example2**.
3. Write a third CSS rule to set a **margin of 20px** for the element with a class of **example3**.

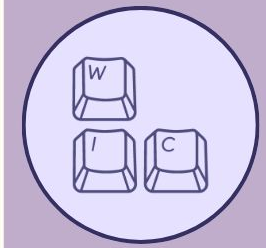


Exercise Answer

```
/* Add styles here */
.example1 {
  padding: 20px;
}
```

```
/* Add styles here */
.example2 {
  border: 10px solid red;
}
```

```
/* Add styles here */
.example3 {
  margin: 20px;
}
```



Exercise 208-02: Style some inline elements

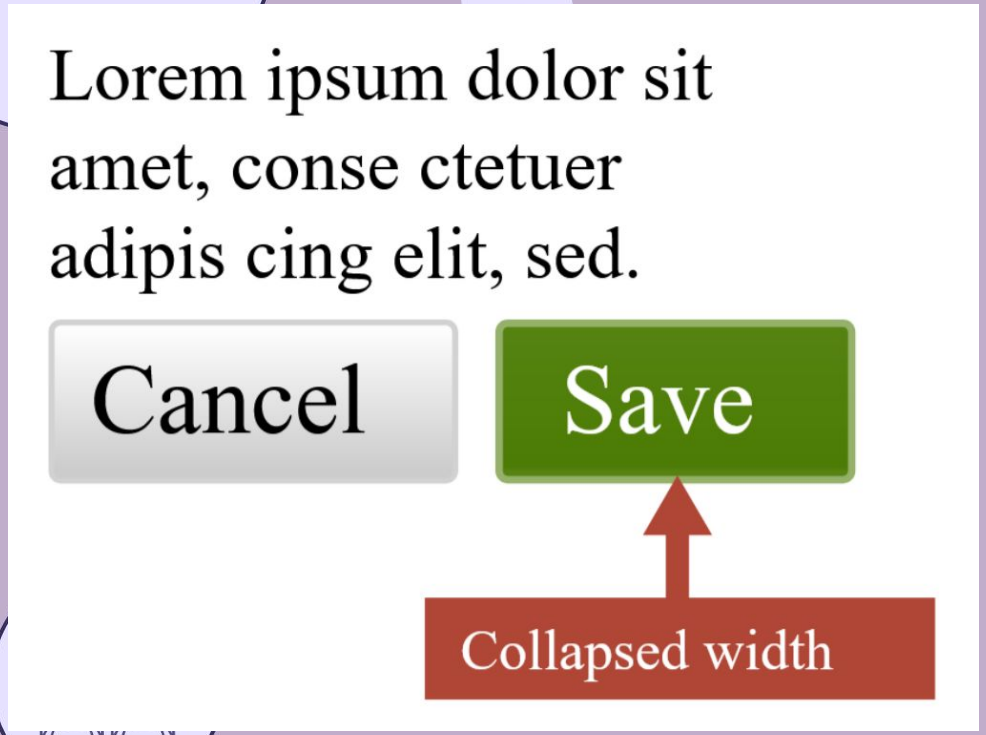
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa *Default inline element* repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa *Example 1 - padding* repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa *Example 2 - border* repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

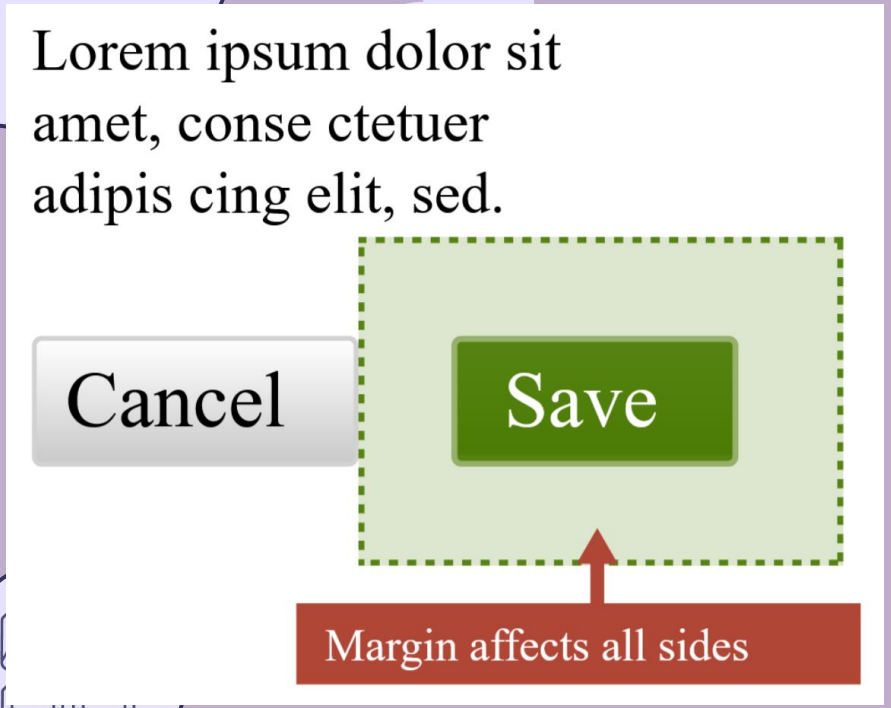
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quis, aliquam, eos? Enim ipsa *Example 3 - margin* repellat culpa expedita omnis, repudiandae error architecto a nam, consequuntur incidunt molestiae deleniti explicabo eius aut aliquam.

Inline-block boxes



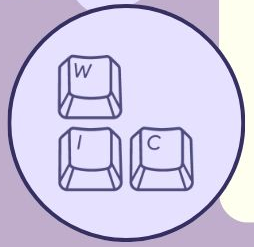
- **Inline boxes** are within one line (can also wrap to the next line)
- Default height and width is *collapsed*
- Height and width can be **sized/stretched** similar to block boxes

Inline-block boxes



- Margin and padding affect **all sides** of inline-block boxes

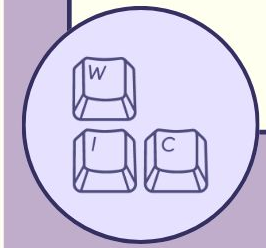
CSS Positioning



https://russmaxdesign.github.io/maxdesign-slides/02-css/209-css-positioning.html#/

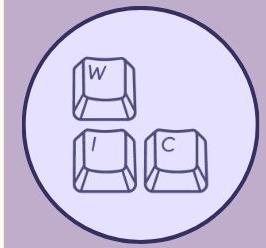
CSS Positioning

- The **position** property specifies how **HTML elements** are positioned.
- The possible values are: **static**, **relative**, **absolute**, **fixed**, and **sticky**.



position: static

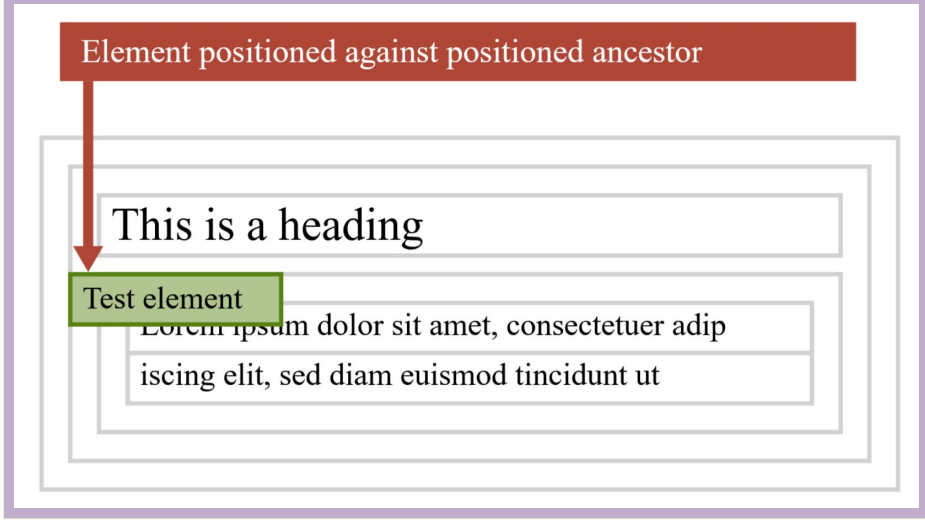
```
/* Static */
p {
  position: static;
}
```



- static is the default value
- Lays element out according to normal flow of document

position: absolute

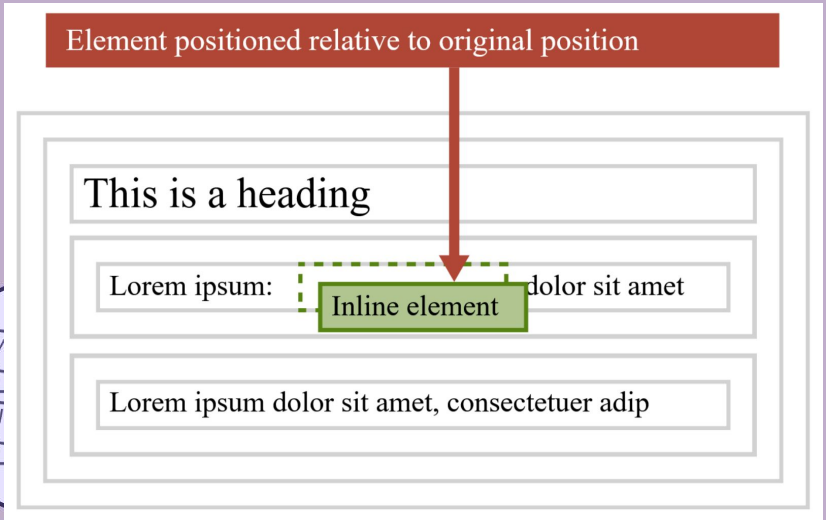
```
/* Absolute */
p {
  position: absolute;
}
```



- Absolute **removes element** from normal document flow
- Absolutely-positioned elements are positioned **relative** to their **closest positioned ancestor** or their **initial containing block**.
- Other elements ignore absolutely positioned element

position: relative

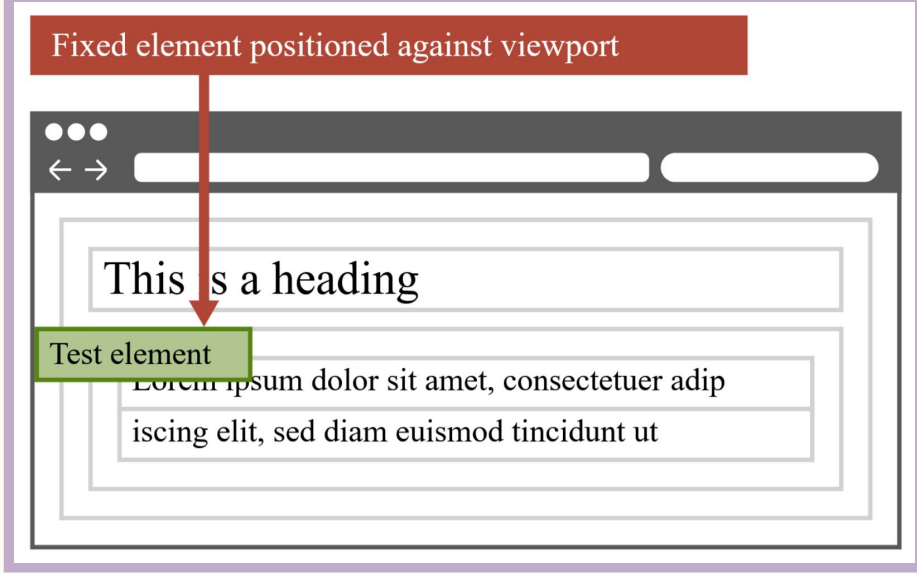
```
/* Relative */
p {
  position: relative;
}
```



- Relative positions elements **relative** to their **original position**

position: fixed

```
p {
  position: fixed;
}
```

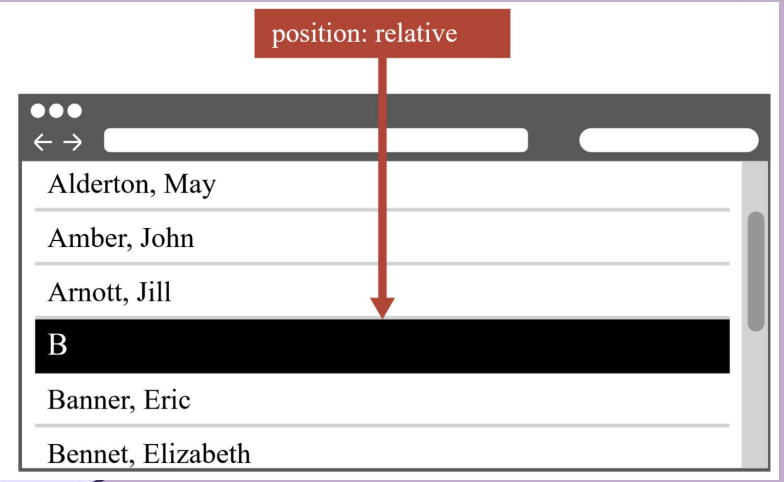


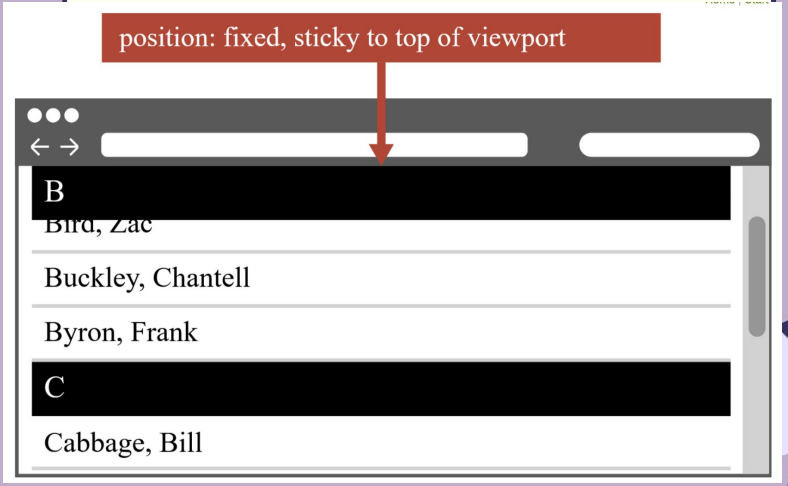
- Fixed elements are positioned against the **viewport**
- These elements will not move when the page is scrolled

position: sticky

```
p {
  position: -webkit-sticky;
  position: sticky;
}
```

- Sticky elements treated as **relative** until their containing block **crosses a threshold**
- Then treated as **fixed**





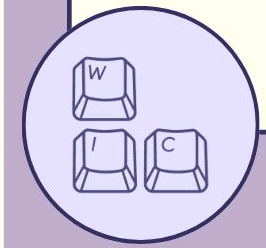
Exercise: Positioning

Go to the exercise-209-04.html file

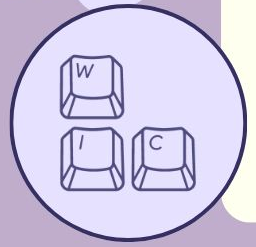
Try out each of the values for the positioning attribute! See what changes when you change the positioning CSS rule for the class contacts-heading, or any other classes that you make.

Possible position values: **static**, **relative**, **absolute**, **fixed**, and **sticky**

```
/* Absolute */
p {
  position: absolute;
}
```



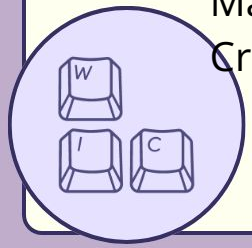
CSS Flexbox Layout



https://www.joshwcomeau.com/css/interactive-guide-to-flexbox/

Flexbox Layout

- One of the most important parts of CSS!!
- **CSS Flexbox** is a powerful layout system used to create flexible and responsive web designs. It provides an intuitive way to arrange elements within a container and control their alignment, distribution, and order.
- A Flex Container is an element to which you apply the CSS property `display: flex;` or `display: inline-flex;`. This makes it a container for Flex Items, which are the child elements within the Flex Container. Flex Items are flexibly arranged inside the container.
- Flexbox operates along two axes: the **"Main Axis"** and the **"Cross Axis."** The Main Axis represents the primary direction of Flex Items' alignment, and the Cross Axis is perpendicular to it.

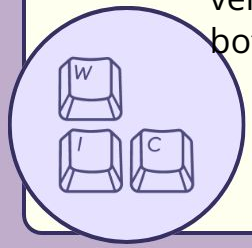


Flex Container Properties

flex-direction

- The flex-direction property determines the direction of the Main Axis. It has four possible values:

- row: Flex Items are placed horizontally in a row from left to right (default).
- row-reverse: Flex Items are placed horizontally in a reversed order from right to left.
- column: Flex Items are placed vertically in a column from top to bottom.
- column-reverse: Flex Items are placed vertically in a reversed order from bottom to top.



justify-content

- The justify-content property aligns Flex Items along the Main Axis. It offers various values to control the spacing between Flex Items:

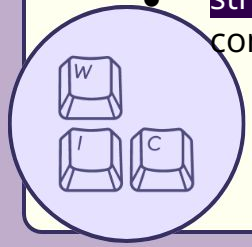
- flex-start: Items are aligned to the start of the container (default).
- flex-end: Items are aligned to the end of the container.
- center: Items are centered within the container.
- space-between: Items are evenly distributed with the first item at the start and the last item at the end.
- space-around: Items are evenly distributed with equal space around them.
- space-evenly: Items are evenly distributed with equal space around and at the start and end.

Flex Container Properties

align-items

- The align-items property aligns Flex Items along the Cross Axis. It has the following values:

- flex-start: Items are aligned to the start of the container.
- flex-end: Items are aligned to the end of the container.
- center: Items are centered along the Cross Axis.
- baseline: Items are aligned based on their text baselines.
- stretch: Items are stretched to fill the container along the Cross Axis (default).



flex-wrap

- The flex-wrap property determines whether Flex Items should wrap to a new line when they exceed the Flex Container's width. It has three possible values:

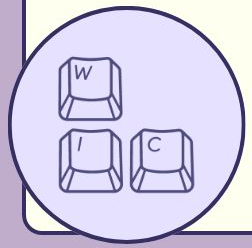
- nowrap: Items are displayed in a single line (default).
- wrap: Items wrap to a new line as necessary.
- wrap-reverse: Items wrap to a new line in reverse order.

Flex Item Properties

flex-grow, flex-shrink, and flex-basis

- These properties are collectively known as the "flex shorthand." They control the growth, shrinking, and initial size of Flex Items.

- flex-grow: Determines how much an item should grow relative to other items when extra space is available.
- flex-shrink: Determines how much an item should shrink relative to other items when there is not enough space.
- flex-basis: Specifies the initial size of an item before any available space is distributed.

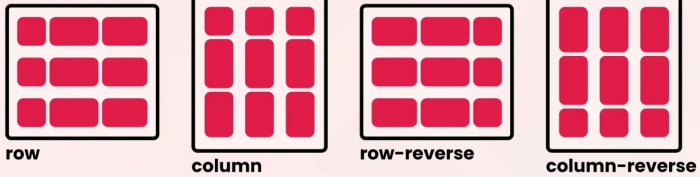


order

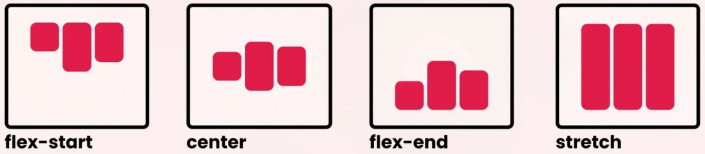
- The order property allows you to change the order in which Flex Items appear within the Flex Container without changing their source order in the HTML.

CSS Flexbox

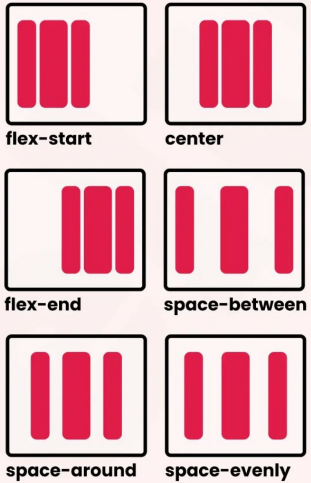
flex-direction



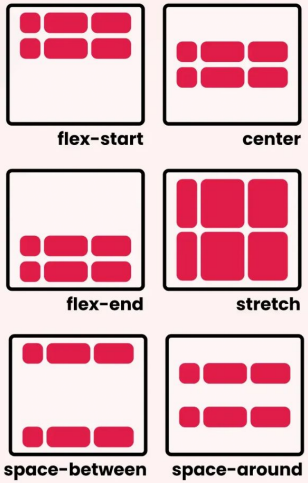
align-items

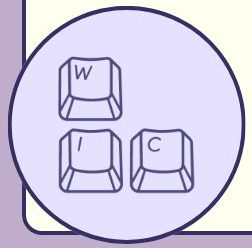


justify-content



align-content



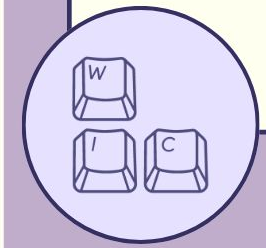


@eludadev

Exercise: Explore Flexbox

Go to the flexbox.html file and explore the different flex properties that you just learned

- Another really cool interactive guide: https://www.joshwcomeau.com/css/interactive-guide-to-flexbox/

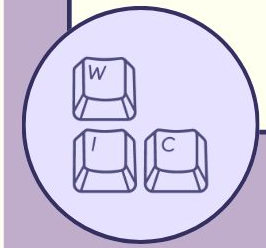


Game: Flexbox

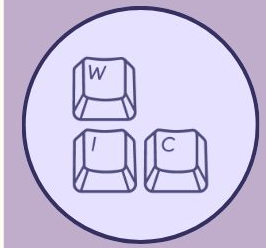
Learn how to use flex by playing these games:

https://codingfantasy.com/games/flexboxadventure/play

https://flexboxfroggy.com/



Here are more CSS Slides that cover a lot more CSS ideas and properties!



https://russmaxdesign.github.io/maxdesign-slides/index.html

WIC Events This Week!

Fall Quarter 2024 Events

Week 6 Events Nov 4-8

Mon 4	Community hours	CSE B225 4-5 pm
Tues 5	CSS Workshop	ASML 4-5 pm
	BPC Prep Day	Qualcomm 5:30-6:30 pm
Wed 6	Community Hours	Tables Outside CSE 3-4 pm
Thurs 7	First-Year Welcome to San Diego Series: UTC Run!	Trolley Station 5:30-6:30 pm
Sat 9	EDGE College Workshop	CSE B250&B260 10 am-1 pm

Thank you!

Make sure to sign in to get your membership points!