

//Developers_Institute_
TLV Coding Bootcamp

<Welcome to our>
</COURSE: CSS>



Developers.Institute

Intense, career-oriented courses

What Will be Covered?

— — —

- CSS Overview
- Selectors
- Identifiers
- Properties and manipulation
- Position and Display
- Combinators
- Pseudo Class Selectors
- Practical Exercises

What is CSS?

— — —

- CSS stands for Cascading Style Sheets. While HTML defines the structure of a web page, CSS describes how elements are displayed on the screen.
- HTML represents content, and CSS represents the appearance of the content.

CSS Syntax

A CSS rule-set consists of a selector and some declarations.

The selector points to the HTML element you want to style. It can be a *html tag*, *class* or *id*.

Declarations are defined in a declaration block, surrounded by curly brackets `{ }`, each declaration includes a CSS property name and value in this format `name: value`, and ends with a semicolon `;`.

You can also add comments, surrounded by `/*` and `*/`.

For example:

```
h1 {                                /* Selector */
  color: blue;                      /* Declaration 1 */
  font-size: 12px;                 /* Declaration 2 */
}
```

Where do we write CSS?

— — —

CSS can be written in three different places.

1. First, it can be written in any HTML element:

- `<h1 style="color:blue; font-size:12px;">Hello World</h1>`

2. Second, we can use `<style>` tags in the head of our HTML document -->

- However, we generally avoid this option in favor of option 3

3. Third, the best option, is to write CSS in a stylesheet

- This .css file will be read by the browser and the style will be applied on the web page.
- You need to specify the path of your stylesheet in your html code if you want it to be executed (see image on next slide)

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {color:red;}
p {color:blue;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
<html>
  <head>
    <link rel="stylesheet" href="/path/to/stylesheet.css"></link>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

CSS Simple Selectors

CSS selectors are used to “find” (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- **Simple selectors** (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)

Element Selector

Element selector selects HTML elements based on the element name (tag).

For example, this style will be applied on every `<p>` tag:

```
p {  
  color: red;  
}
```


HTML - ID and Class

- HTML elements can have a `class` attribute, used to define equal styles for elements with the same class name.
 - ``
- Elements can also have a unique `id` attribute.
 - `<p id="my_paragraph">Hello World</p>`

Note that multiple elements can have the same `class` name, but an `id` can never be given to more than one element.

Class and ID Selectors

The **id selector** uses the `id` attribute of HTML element, to use it, write a hash `#` character, followed by the id of the element.

For example, to style this paragraph `<p id="my_paragraph">Hello World</p>`, use:

```
#my_paragraph{  
  color: red;  
}
```

While the **class selector** uses the `class` attribute of the elements, to use it, write a dot `.` followed by the class of the element.

For example, to style this image ``, use:

```
.my_img{  
  width: 200px;  
  height: 200px;  
}
```

Universal and Grouping Selectors

Universal selector is also available, in css (and in a lot of computer languages), `*` means “Everything”

This style will be applied on every html elements of the page:

```
* {  
  color: blue;  
}
```

Grouping Selector

You can style more than one element at the time, just separate each selector with a comma `,`:

```
h1, h2, p {  
  color: red;  
}
```

CSS Properties - Colors and Backgrounds

Colors

In CSS, colors can be Hexadecimal Color Codes (`#ffffff`), rgb (`rgb(255,255,255)`), or color names (`black`).

Backgrounds

You can use CSS to add a background to your sections, it can be a `background-color` or a `background-image` .

`background-image` should receive a url to work.

```
body{  
    background-image: url("my_img.png")  
}
```

CSS Properties - Borders and Margins

Borders

The CSS `border` properties allow you to specify the `border-style`, `border-width`, and `border-color` of an element's border.

```
p {  
  border-style: solid;  
  border-width: 5px;  
  border-color: green;  
}
```

Margins

The CSS `margin` properties are used to **create space around elements, outside of any defined borders**.

You can specify the width of `margin` for every side, but you can also specify the margin for each side of the element.

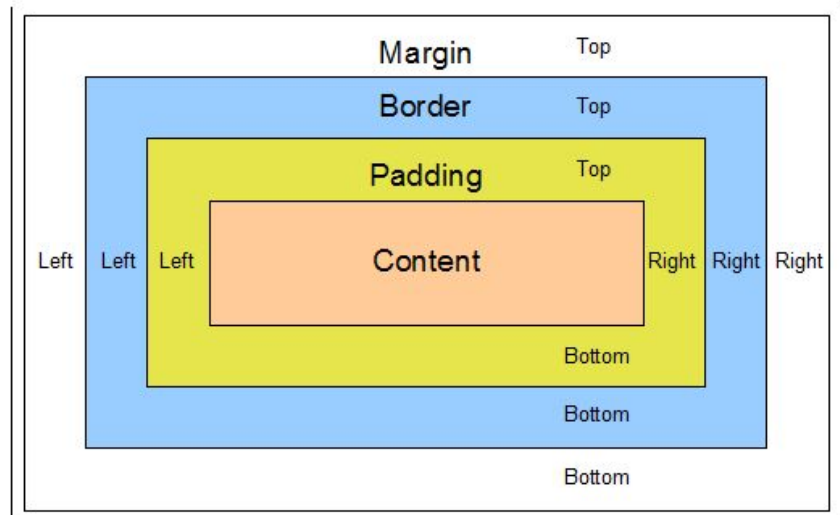
```
p {  
  margin-top: 100px;  
  margin-bottom: 100px;  
  margin-right: 150px;  
  margin-left: 80px;  
}
```

CSS Properties - Padding

The CSS `padding` properties are used to **generate space around an element's content, inside of any defined borders.**

It works the same way as `margin`.

See the [box model](#) to understand `padding` and `margin`



Elements and Dimensions

The `height` and `width` properties are used to set the height and width of an element.

Text styling

You can specify the `color` of the text, but also the alignment (with `text-align`), the `text-decoration`, the `text-transformation`, the `text-indent` and [much more](#)

```
p {  
  color: blue;  
  text-align: justify;  
  text-indent: 50px;  
  text-decoration: underline;  
  text-transformation: uppercase;  
}
```

List styling

You can modify the icon in front of each list elements, with `list-style-type` to use a built in icon, or with `list-style-image` to add an image of your own.

Exercise

Create a .css stylesheet and link it to your menu.html file that we worked on last class.

Choose one of the tables that you made for your menu, and begin experimenting with the style properties we've learned so far.

- Can you figure out how to put space between the columns and the rows?
- Don't forget that classes/IDs can make your job easier!

Elements Display

— — —
You can modify the way elements are displayed.

The `display` property is controlling the layout of an element.

Block level: `display: block;` will display the element as a block, meaning stretch it out to the left and right as far as possible, and the next element will be placed below it.

Inline: `display: inline;` elements don't start on a new line and only take up as much width as necessary.

Inline block: `display: inline-block` allows to set a width and height on the element, and respect the padding, but doesn't add a line break after the element.

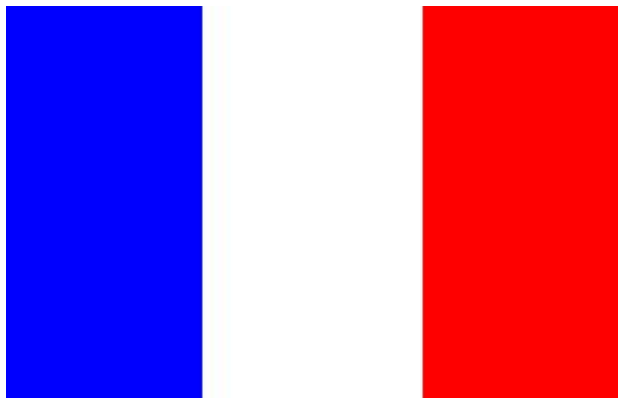
None: `display: none` won't display the element.

Careful: It will not hide it (meaning it won't keep a blank place for the element), to hide an element, use `visibility: hidden`

For more on these, see: [This Course on CSS Layout](#) / [Example with squares](#) / [Example within texts](#)

Exercise

Recreate this flag with the display property:



Elements Position

— — —
The `position` property specifies the type of positioning method used for an element (static on the page, relative to his parent, etc..)

It can be:

- `static`: Always positioned according to the normal flow of the page
- `relative`: Position the element relative to its normal position, you can add `left`, `right`, `top` or `bottom` arguments to add a gap by the element.
- `fixed`: Always stays in the same place even if the page is scrolled, `left`, `right`, `top` and `bottom` are used to position the element
- `absolute`: Always stays in the same place relative to his parent.
- `sticky`: The element won't leave the page, it will be positioned relative until he reaches the edge of the window, then it "sticks" to it.

For more on these, see: [More on CSS Position/ CSS Position Tricks](#)

Exercise

Recreate this image with the position property:



The elements are : a parent box (grey), a blue box and an orange box.

Overflow

— — —

The `overflow` property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

It can be:

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

Combinator Selectors

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element (meaning they are nested in it).

The following example selects all `<p>` elements inside `<div>` elements:

```
div p {  
  background-color: yellow;  
}
```

Child Selector

The child selector selects all elements that are the direct descendant of a specified element.

The following example selects all `<p>` elements that are children of a `<div>` element:

```
div > p {  
  background-color: yellow;  
}
```

```
<div>  
  <p></p>  
  <p></p>  
  <p></p>  
</div>
```

```
<p></p>|
```

```
<div>  
  <p></p>  
  <p></p>  
</div>
```

```
<p></p>  
<p></p>
```

Combinator Selectors

Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

Sibling means “that have the same parent”.

The following example selects all `<p>` elements that are siblings of `<div>` elements:

```
div ~ p {  
  background-color: yellow;  
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

“Adjacent” means “immediately following”.

The following example selects all `<p>` elements that are placed immediately after `<div>` elements:

```
div + p {  
  background-color: yellow;  
}
```

```
<div>  
  <p></p>  
  <p></p>  
  <p></p>  
</div>
```

```
<p></p>|
```

```
<div>  
  <p></p>  
  <p></p>  
</div>
```

```
<p></p>  
<p></p>
```

Pseudo Class Selectors

A pseudo class is used to style an element when he is in a special state (for example, when the user hovers over it)

The syntax is : `element:pseudo-class { ... }`

Pseudo Class Selectors

hover

`hover` means “mouse is over it”, for example, to change the color of a `<div>` when the mouse go over it:

```
div:hover {  
  background-color: blue;  
}
```

children

You can target elements only if they are the n th child of their parent. For example, to style `<p>` tags that are the first child of another element, use:

```
p:first-child{  
  color: blue;  
}
```

The same thing works with `last-child` or `nth-child(n)`

not

The `not` pseudo class allows you to exclude elements, for example, select all the div that are not of the `ignore` class

```
div:not(.ignore){  
  background-color: yellow;  
}
```

Exercise

— — —

Using the code that I will provide you, do the following steps:

1. Change the second li of the 1st div to blue
2. Make all the li of the first div in uppercase and in bold when you hover over them
3. Change the 1st paragraph of the second div so its background color will be pink and its color white
4. Make the 2nd paragraph of the second div with a background color light blue, with padding and margin.

Bonus: Try to find a few ways to achieve each of the steps

Code

```
-- -- --
<div>
  <ul>
    <li>What is Lorem Ipsum?</li>
    <li>Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,</li>
    <li>when an unknown printer took a galley of type and scrambled it to make a type specimen
book. </li>
  </ul>
</div>
<div>
  <p>It has survived not only five centuries, but also the leap into electronic
typesetting,</p>
  <ol>
    <li> remaining essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets </li>
  </ol>
  <p> containing Lorem Ipsum passages, and more recently with desktop publishing software like
Aldus PageMaker including versions of Lorem Ipsum</p>
</div>
```

Remember this?

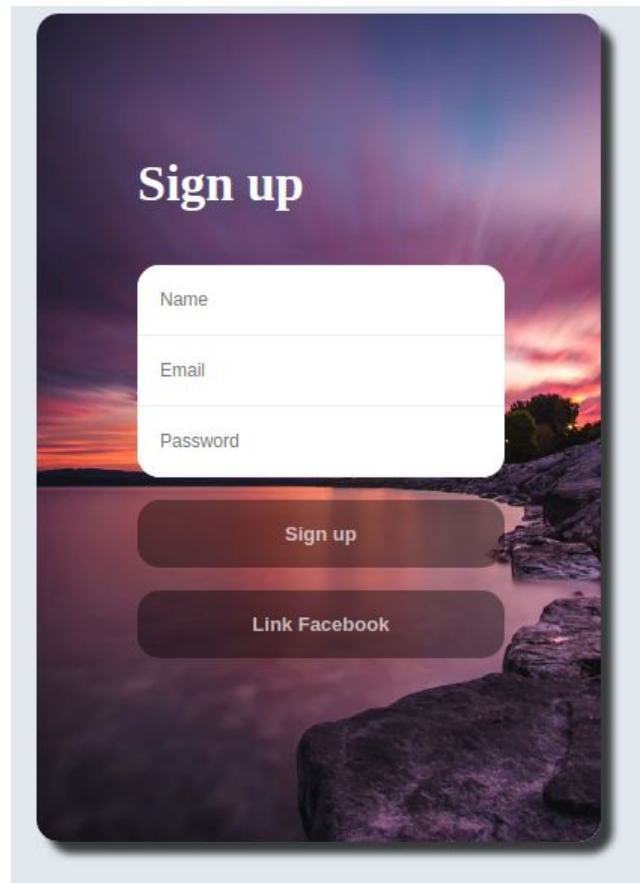
Going from this:

Sign up

<input type="text" value="Name"/>	<input type="text" value="Email"/>	<input type="text" value="Password"/>
<input type="button" value="Sign up"/>	<input type="button" value="Link Facebook"/>	

Let's see how it's done!

To this:



A modern, visually appealing sign up form. The background is a vibrant sunset over a body of water with rocks in the foreground. The form is a white rounded rectangle with three input fields: Name, Email, and Password. Below the fields are two dark, rounded buttons: 'Sign up' and 'Link Facebook'.

Sign up

Conclusion

- We can now use CSS to style and customize our pages!
- Don't spend time memorizing each style property
 - Rather, spend time learning what tools are out there
 - All you need is the idea of what you want to do... syntax can be googled
- Resources: Google, W3Schools, Codepen, Youtube
- Menu project - attack this like a developer
 - Find inspiration online, look up new CSS properties that you can implement, link multiple pages... you'll surprise yourself what you're able to come up with after some practice
- Next class: Bootstrap + taking it to the next level