DevX HTML/CSS

Class 12 - Advanced CSS

Review

We learned about *Forms and Inputs*:

- Forms & form syntax:
  - <form>, <label>, and <input> tags
- Form Tag attributes:
  - “for=”
  - “id=”
  - “name=”
  - “type=”
- <button> Tag
  - Linking <button>'s "form=" attribute to <form>'s "id=" attribute.
- Form Data
- Google Forms
  - Google Forms setup and deploy
  - Youtube video URL ids

Questions?

About Advanced CSS

The CSS that you've learned so far is the fundamentals. But with these fundamentals you can do just about everything you'd want to do. Anything else you can look up using what you know.

Occasionally you'll encounter a situation which is a little difficult using just the fundamentals. In those cases, sometimes more advanced techniques, keywords, etc… will be good to use.

We're going to cover these topics, but I want to stress that you don't *have* to use them. But having the option makes you a stronger developer.

Pseudo Classes

Pseudo Classes

```
a:first-child {
  color: green;
}
a:last-child {
  color: blue;
}
a:nth-child(odd) {
  color: red;
}
a:nth-child(even) {
  color: purple;
}
a:nth-child(10) {
  color: yellow;
}
```

Demonstrate!

Pseudo classes allow you to target elements beyond just Tags, Classes, and ID's. They allow you to select properties of elements such as *if you are hovering over them with your cursor*, *being first* or *being last* in a list, or *having visited a link*.

The syntax consists of a colon and keyword. Some keywords are:

- :hover
- :first-child
- :last-child
- :nth-child()
- :empty

There are many more, but this is a good start. The others are very specific in how you would use them.

:X-child

```
a:first-child  {
   color: green;
}
a:last-child  {
   color: blue;
}
a:nth-child(odd)  {
   color: red;
}
a:nth-child(even)  {
   color: purple;
}
a:nth-child(10)  {
   color: yellow;
}
```

Make a gallery and demonstrate!

First-child and Last-child: In a set of elements, it will apply only to first element or the last one.

Nth-child(): In this case you can write a number inside… like :nth-child(3), and it'll only affect the 3rd child element.

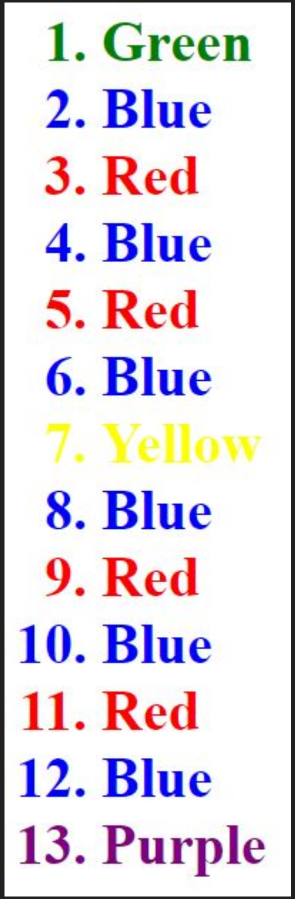
Nth-child(): can also take special keywords like "even", "odd", "3n" to select only even elements, only odd, or every 3rd. *(3n could be 4n or 5n or any number + n)*

It can even do fancy stuff like "4n+7", but that's unnecessarily complicated. So let's stick with first, last, even, odd, and some number.

Activity: Pseudo Classes List

Activity: Pseudo Classes List

```
<ol>
    <li>Green</li>
    <li>Blue</li>
    <li>Red</li>
    <li>Blue</li>
    <li>Red</li>
    <li>Blue</li>
    <li>Yellow</li>
    <li>Blue</li>
    <li>Red</li>
    <li>Blue</li>
    <li>Red</li>
    <li>Blue</li>
    <li>Purple</li>
</ol>
```



1. Green
2. Blue
3. Red
4. Blue
5. Red
6. Blue
7. Yellow
8. Blue
9. Red
10. Blue
11. Red
12. Blue
13. Purple

The goal: Match the list item names with the colors!

Normally, we would write classes that apply font colors. But this time we're using pseudo-classes.

Do not modify the HTML, use only CSS.

You'll need:

- first-child
- last-child
- nth-child(odd)
- nth-child(even)
- nth-child(#)

Remember CSS specificity!

:hover

:hover

The :hover pseudo-class targets an element when the user has a mouse hovering over it. On mobile it's for a finger press.

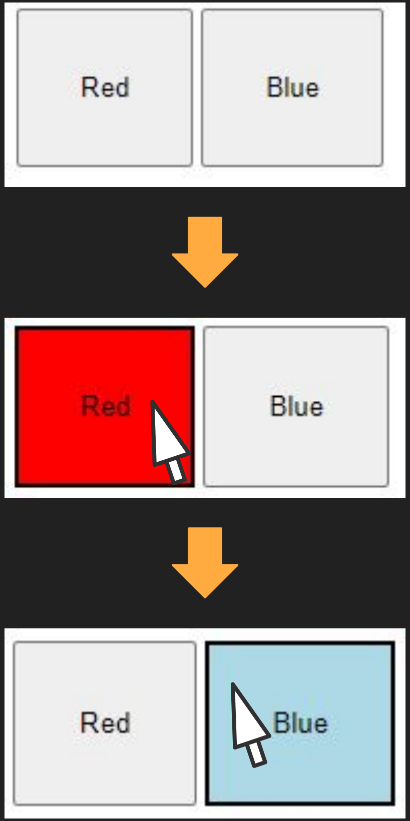
This mostly works for desktop, but can provide a slight effect in mobile too.

```
<button class="red">Red</button>
<button class="blue">Blue</button>
```

+

```
button { padding: 30px; }
button.red:hover { background-color: red; }
button.blue:hover { background-color: lightblue; }
```

=

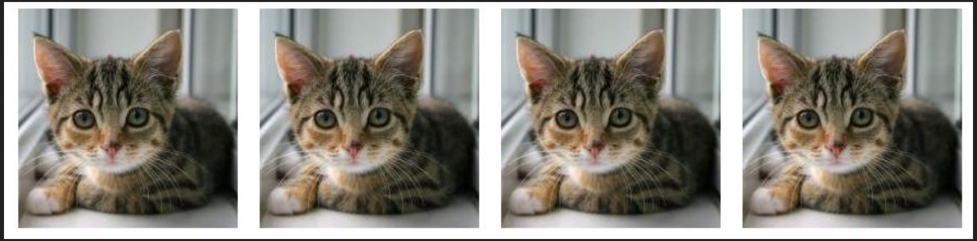


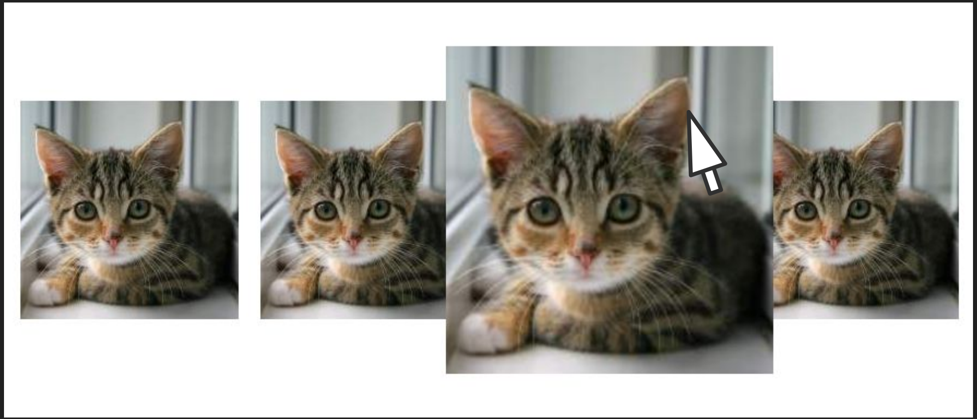
Demonstrate!

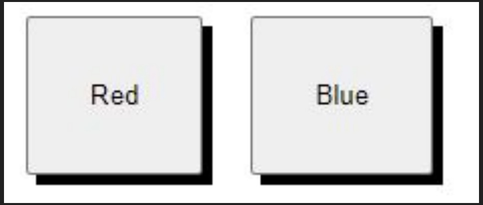
:hover

Hover is an interesting pseudo-class since it allows the user to interact with the element.

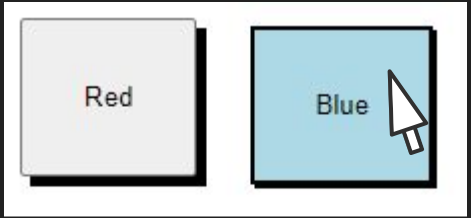
This allows you to do things like giving buttons a clickable reminder. Or expanding images.







Notice the button is pressed ->



Demonstrate!

Activity: Hover

Activity: Hover

In this exercise, we're going to make some elements hoverable!

Use the starter code below and on the right:

```html
<button>Hover me!</button>
<h1>Hover me!</h1>
<p>Hover me!</p>
<div class="blue">Hover me!</div>
```

```css
button {
}
h1 {
}
p {
}
.blue {
  height: 100px;
  width: 100px;
  text-align: center;
  background-color: lightblue;
}
```

I want the font size of each of these elements to increase when you hover!

Do it!

Pseudo Elements

:before and :after

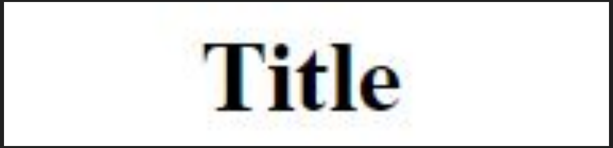
```
<h1>Title</h1>

h1:before {
  content: "before";
}
h1:after {
  content: "after";
}
```

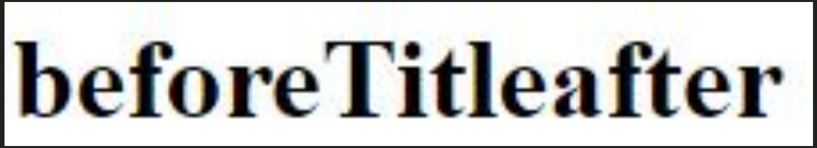
These are weird. CSS is able to create an element on the page, inside of the selected element. Here are the weird parts:

- You cannot click on it
- You cannot highlight it
- You cannot manipulate it with Javascript
- It can only display image or text.

The syntax consists of a colon and the word “before” or “after”.

They also NEED a “content” property to display. NEED IT.





Demonstrate with a simple element!

:before and :after Quirks

Some other important details about working with :before and :after.

Aside from the quirks I mentioned on the previous page, they behave like normal HTML elements. You can apply CSS like border, background-color, or background-image.

They NEED a content property. Even if it reads content: “ ”; It won’t display without it!

I’ve forgotten to add content many times. So try and remember!

Also, Keep it Simple!

Before/After is an advanced concept. They are confusing and weird! But they can also be a nice touch, or solve a specific layout problem for you. Don’t go trying to use them all the time. Keeping things simple is usually the way to go.

Demonstrate with custom bullets!

Attribute Selectors

Attribute Selectors

```
a[title]
a[href="https://example.org"]
a[href*="example"]
a[href$=".org"]
a[class~="logo"]
```

Examples above, some are more complex, like $ and ~. Google those!

You can ALSO select elements based on their attributes! Both having the attribute and what the attribute's value is.

This has limited use. But it's good to know that it is an option.

You could, for example, target only `input[type=”radio”]` . This would avoid other inputs, only styling radio `<button>`.

Most of the time you don't need this feature. But knowing it exists might help you when working with forms, links, images, or otherwise.

Read more about them here:

https://www.w3schools.com/css/css_attribute_selectors.asp

https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors

Demonstrate!

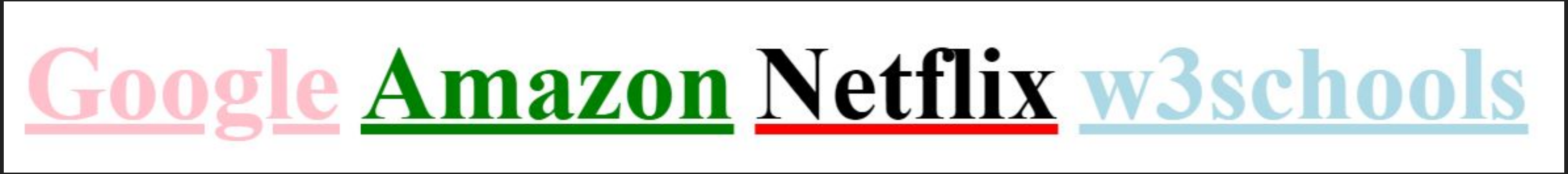
Activity: Attribute Selectors

Activity: Attribute Selectors

We’re going to make some links, and then style them differently depending on which site they go to!

Use [href*=”something”] to target them and give them different colors!

```
<a href="http://www.google.com" >Google</a>
<a href="http://www.amazon.com" >Amazon</a>
<a href="http://www.netflix.com" >Netflix</a>
<a
href="http://www.w3schools.com" >w3schools</a>
```



Give it a try! Google how to change underline color!

calc()

calc()

calc() is an alternative way to work with measurements in CSS.

It allows you to perform math operations with different CSS units of measurement.

In this example, you want the paragraph tag to be almost as wide as the screen, but 40 pixels less. This calc does just that.

You can use any measurement units (pixels, em, rem, %, vh, vw, etc), with simple math operations (plus, minus, multiply or divide).

Note: There MUST be spaces around the operators:
+ or - or * or /

Demonstrate!

```
p {
  width: calc( 100vw - 40px );)
}
h1 {
  width: calc( 100vh - 10% );)
}
h2 {
  width: calc( 2em - 5px );)
}
```

The above are just examples. Not real world examples.

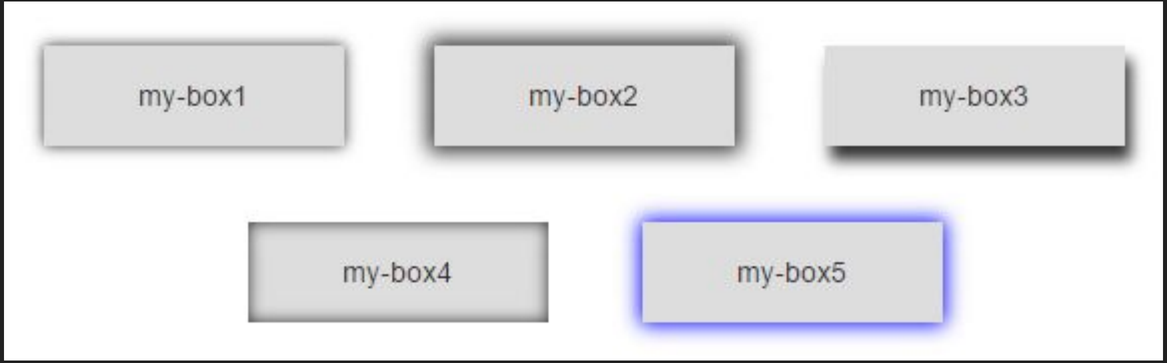
Calc can get complicated. Try and keep it simple. Only use when easier than other methods.

Shadows

Box Shadow

Box shadow applies to any element which uses the box model.

The CSS property is box-shadow, and it takes a value something like:



```
div {
   box-shadow: 3px 3px 3px black;
}
```

X axis position (left/right)

Y axis position (up/down)

“Fuzz” or “Blur” radius

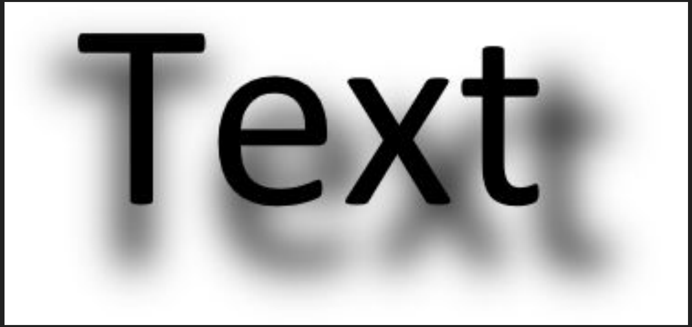
Color

Demonstrate!

Text Shadow

Text shadow works like box shadow but only applies to text.

The CSS property is text-shadow, and it takes a value something like:



```
div {
  text-shadow: 3px 3px 3px black;
}
```

X axis position (left/right)

Y axis position (up/down)

“Fuzz” or “Blur” radius

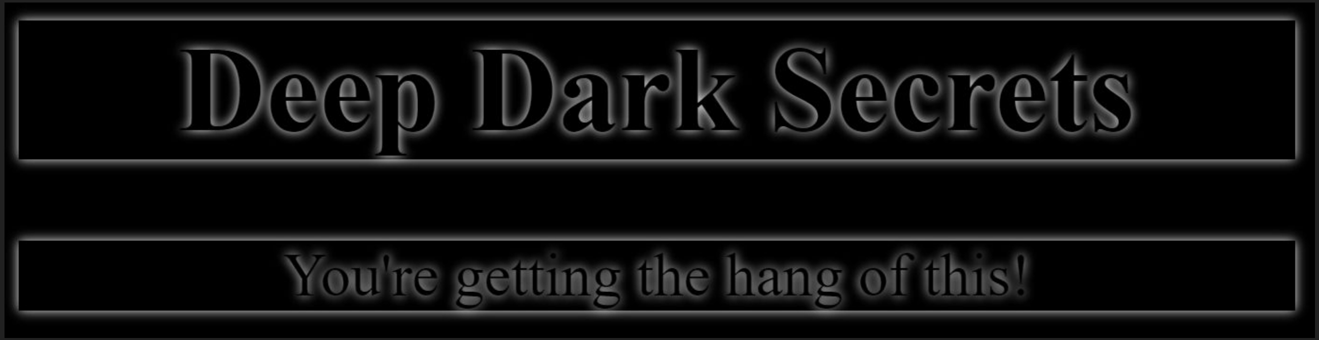
Color

Demonstrate!

Activity: Shadows

Activity: Shadows

Use the starter code below, text and box shadow to create the design over here ->



Deep Dark Secrets

You're getting the hang of this!

You'll also need text-align and font-size!

```
body {
  background-color: black;
}

<h1>Deep Dark Secrets</h1>
<p>You're getting the hang of
this!</p>
```

Homework