J220: Intro to Coding

Class 10
Monday April 11

Note to lecturers: start Zoom recording (and transcription!)

Use iClicker.com to log attendance

Agenda

- Quick review
- Pseudo-classes
- Project Presentations

Break

- CSS Transitions
- Pseudo-elements
- CSS Display
- CSS Position
- Extra Credit Homework

Quick Review

Quick Review: Common mistakes

Using &nbsp, an HTML entity for creating a non-breaking space.

> Used in the navigation, instead of wrapping the words in a list, around a tags or spans.

Don't use &nbsp, use the box model! Apply a padding or margin to the inline HTML elements

Quick Review

Your HTML code is divided into two sections. The <head>, which is read by the browser and has information that cannot be seen by the user (unless viewing via the inspector tool) and the <body> which has all visible parts of the page.

This mean your <footer>, which is seen by the user, should be nested inside of your body. It should probably be the last HTML elements in your document, but still inside of body.

Browsers are smart enough to figure this out and still show the footer correctly, but you want your code to be as clean as possible.

Quick Review

Media queries:

We always build for mobile first! When using media queries, it is good practice to have those styles apply to window sizes bigger than mobile (like tablets or desktop)

You will still find examples on stackoverflow and other coding sites that show queries being used for mobile, but we are trying to move away from this.

This means using min-width instead of max-width.

Quick Review

Some of you had multiple css files for your pages, with specific files for tags, headings or images. I can understand the confusion, we created separate files in class, but mostly to have you be able to import them and see how they are different

Ideally, you keep all your css together in one file. So don’t break it out into multiple. Except if you’re using simple-grid.css. This is a framework and it should be kept separately.

Pseudo-Classes

What are pseudo-classes?

We’ve used some of these before but haven’t properly explained them. Let’s look at how they work in context of the a tag.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>



    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

1. :link (only used with a tag), used when a link has not been visited by the user

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:link {
        color: yellow;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

1. :link (only used with a tag), used when a link has not been visited by the user
2. :visited (only used with a tag), when the user has visited the link

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:visited {
        color: purple;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

1. :link (only used with a tag), used when a link has not been visited by the user
2. :visited (only used with a tag), when the user has visited the link
3. :hover, used when the mouse is over the element

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:hover {
        color: seagreen;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

1. :link (only used with a tag), used when a link has not been visited by the user
2. :visited (only used with a tag), when the user has visited the link
3. :hover, used when the mouse is over the element
4. :focus, used when selected with the tab key

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:focus {
        color: orange;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

1. :link (only used with a tag), used when a link has not been visited by the user
2. :visited (only used with a tag), when the user has visited the link
3. :hover, used when the mouse is over the element
4. :focus, used when selected with the tab key
5. :active, used when the element being clicked on

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:active {
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

:hover, :focus, :active

Think about these pseudo-classes as a way to manipulate the styling depending on the ‘special state’ of the element.

They can be used on **any** HTML element on the page, not just <a> tags.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      a:active {
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Common pseudo-classes

Let's see these classes in action! Open the in-class example zip shared in Slack

j220_april11.zip

Quick detour into anchor links

When creating a one-page site, help the user navigate by making use of anchor links in your navigation.

Anchor links

Let’s review this simplified HTML page.

```html
  <body>
    <nav>
      <ul class="navLinks">
        <a href="#about"><li>About</li>
         </a>
        <a href="#work"><li>Work</li></a>
      </ul>
    </nav>
    <section id="about">
      <h3>About Yoli Martinez</h3>
      <p>Yoli is a Software Engineer at
The Washington Post, working mostly with
visual storytelling teams.</p>
    </section>
    <section id="work">
      <h3>A selection of work</h3>
      <p>Here's a list of stories I
contributed to recently</p>
    </section>
  </body>
```

Anchor links

First, notice the navigation. The <li> are surrounded by <a> tags.

```
  <body>
    <nav>
      <ul class="navLinks">
        <a href="#about"><li>About</li>
         </a>
        <a href="#work"><li>Work</li></a>
      </ul>
    </nav>
    <section id="about">
      <h3>About Yoli Martinez</h3>
      <p>Yoli is a Software Engineer at
The Washington Post, working mostly with
visual storytelling teams.</p>
    </section>
    <section id="work">
      <h3>A selection of work</h3>
      <p>Here's a list of stories I
contributed to recently</p>
    </section>
  </body>
```

Anchor links

First, notice the navigation. The <li> are surrounded by <a> tags.

Instead of an outside link, the href's point to internal navigation. An id that has been applied to another HTML element.

```html
  <body>
    <nav>
      <ul class="navLinks">
        <a href="#about"><li>About</li>
         </a>
        <a href="#work"><li>Work</li></a>
      </ul>
    </nav>
    <section id="about">
      <h3>About Yoli Martinez</h3>
      <p>Yoli is a Software Engineer at
The Washington Post, working mostly with
visual storytelling teams.</p>
    </section>
    <section id="work">
      <h3>A selection of work</h3>
      <p>Here's a list of stories I
contributed to recently</p>
    </section>
  </body>
```

Anchor links

First, notice the navigation. The <li> are surrounded by <a> tags.

Instead of an outside link, the href's point to internal navigation. An id that has been applied to another HTML element.

Anything can get an id applied to it. Here, we are using the id to correspond with different sections of the site.

```
  <body>
    <nav>
      <ul class="navLinks">
        <a href="#about"><li>About</li>
         </a>
        <a href="#work"><li>Work</li></a>
      </ul>
    </nav>
    <section id="about">
      <h3>About Yoli Martinez</h3>
      <p>Yoli is a Software Engineer at
The Washington Post, working mostly with
visual storytelling teams.</p>
    </section>
    <section id="work">
      <h3>A selection of work</h3>
      <p>Here's a list of stories I
contributed to recently</p>
    </section>
  </body>
```

Anchor links

When a link gets clicked, the hash gets added to the url

If this was your website:

ymartinez.github.io

After the click turn the url into this:

ymartinez.github.com/#about

Let’s look at the in-class examples again. This is easier to understand when seeing in action.

```
  <body>
    <nav>
      <ul class="navLinks">
        <a href="#about"><li>About</li>
         </a>
        <a href="#work"><li>Work</li></a>
      </ul>
    </nav>
    <section id="about">
      <h3>About Yoli Martinez</h3>
      <p>Yoli is a Software Engineer at
The Washington Post, working mostly with
visual storytelling teams.</p>
    </section>
    <section id="work">
      <h3>A selection of work</h3>
      <p>Here's a list of stories I
contributed to recently</p>
    </section>
  </body>
```

Advanced Pseudo-Classes

Advance pseudo-classes

These classes are mostly based around the position of the element in the page.

1. :first-of-type
2. :last-of-type

You can find more here, though most are used for forms: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>



    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

These classes are mostly based around the position of the element in the page.

1. :first-of-type
2. :last-of-type

Pay attention to the CSS syntax, these also start with one colon.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>




    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:first-of-type

Selects the first of a group of sibling elements. But what are siblings?

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:first-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:first-of-type

Selects the first of a group of sibling elements. But what are siblings?

In the code on the right, the **<article>** element is the parent. The <p> tags are nested inside.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:first-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:first-of-type

Selects the first of a group of sibling elements. But what are siblings?

In the code on the right, the <article> element is the parent. The <p> tags are nested inside.

That makes the <p> tags the children.

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:first-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:first-of-type

Selects the first of a group of sibling elements. But what are siblings?

In the code on the right, the <article> element is the parent. The <p> tags are nested inside.

That makes the <p> tags the children.

The first <p> would be marked as the first-of-type element.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:first-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:last-of-type

Works just like first-of-type, but in reverse.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:last-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:last-of-type

Works just like first-of-type, but in reverse.

In this case, the second <p> tag is considered last.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:last-of-type {
        margin-bottom: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

:first-of-type, :last-of-type

I end up using these pseudo-classes mostly to add box-model spacing (padding/margins) around the page or when styling needs to be a bit more predictable.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p:first-of-type {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with <a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
      <p>Followed by a second paragraph
with no links to a cute kitten. Sad.</p>
    </article>
  </body>
</html>
```

Advance pseudo-classes

Let’s see these classes in action! Open the in-class example zip shared in Slack. Look at the “contact” section and uncomment lines 92-103 in the CSS file.

j220_april11.zip

CSS Transitions

CSS Transitions

Mostly used when thinking about UI/UX (user interface, user experience)

It gives a finished look to interactive elements.

We'll focus on using transitions when :hover pseudo-class is activated.

CSS Transitions

What do you need to make a transition?

1. Name of the CSS property the transition will be applied to

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  color: white;
  transition-property: color;



}
```

CSS Transitions

What do you need to make a transition?

1. Name of the CSS property the transition will be applied to
2. The amount of time it should take to transition

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  color: white;
  transition-property: color;
  transition-duration: 500ms;


}
```

CSS Transitions

What do you need to make a transition?

1. Name of the CSS property the transition will be applied to
2. The amount of time it should take to transition
3. The animation effect

```css
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  color: white;
  transition-property: color;
  transition-duration: 500ms;
  transition-timing-function: linear;

}
```

CSS Transitions

What do you need to make a

1. Name of the CSS pro
   transition will be app
2. The amount of time i
   to transition
3. The animation effect

Timing function options:
https://developer.mozilla.org/en-US/docs/Web/CSS/transition-timing-function

I usually end up using linear or ease-out

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
            30px;
          adius: 10px;
            e: 40px;
           ttom: 20px;
            block;


          :hover {
            te;
  transition-property: color;
  transition-duration: 500ms;
  transition-timing-function: linear;

}
```

CSS Transitions

What do you need to make a transition?

1. Name of the CSS property the transition will be applied to
2. The amount of time it should take to transition
3. The animation effect
4. The amount of time a user will wait before seeing a transition in action (I have rarely used)

```css
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  color: white;
  transition-property: color;
  transition-duration: 500ms;
  transition-timing-function: linear;
  transition-delay: 1s;
}
```

CSS Transitions

Just like with padding and margin, there is a shortened way to set this up. Just used the transition CSS property:

transition: <transition-property> <transition-duration> <transition-timing-function> <transition-delay>

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  color: white;
  transition: color 500ms linear;
}
```

CSS Transitions

If you are applying a transition to several properties, you have two options.

1. List them out, with a comma separating each property.

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  font-size: 50px;
  color: white;
  transition: font-size 500ms linear,
color 1s linear;
}
```

CSS Transitions

If you are applying a transition to several properties, you have two options.

1. List them out, with a comma separating each property.
2. If changing everything with the same duration, use all.

```css
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
}

.hoverPsuedo:hover {
  font-size: 50px;
  color: white;
  transition: all 500ms linear;
}
```

CSS Transitions

You can add transitions on any selector, whether is using a pseudo-class or not.

So, if there has been a change in how something looks before/after a :hover, :focus, :active, you can apply a transition to the element so it can also ease back pleasantly!

```
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
  transition: all 1s linear;
}

.hoverPsuedo:hover {
  font-size: 50px;
  color: white;
  transition: all 500ms linear;
}
```

CSS Transitions

Things to keep in mind:

- Not all CSS properties can have a transition applied. For full list, visit here: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties

(This doc mentions CSS animations, which we'll learn in our next CSS class!)

```css
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
  transition: all 1s linear;
}

.hoverPsuedo:hover {
  font-size: 50px;
  color: white;
  transition: all 500ms linear;
}
```

CSS Transitions

Things to keep in mind:

- Play around while you’re learning, but be thoughtful about when you apply transitions. Movement on the page can be very disorienting.
- Best part is learning new things, like this: https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-reduced-motion

Let’s review this doc quickly!

```css
.hoverPsuedo {
  background-color: seagreen;
  border: none;
  width: 50%;
  padding: 30px;
  border-radius: 10px;
  font-size: 40px;
  margin-bottom: 20px;
  display: block;
  transition: all 1s linear;
}

.hoverPsuedo:hover {
  font-size: 50px;
  color: white;
  transition: all 500ms linear;
}
```

CSS Transitions

Let’s go back to our example and see how we can apply transitions over buttons.

Uncomment lines 98-110

CSS Transitions

Let’s quickly review this setup. Some buttons have two classes applied to them.

hoverPsuedo is used to give each button the same styling (background-color, width, margin-bottom, ects.)

```
<section id="transitionsSection"
class="block">
  <div class="row">
    <div class="col-12">
      <h3>Transitions</h3>
      <button class="hoverPsuedo">This
is a button</button>
      <button class="hoverPsuedo
halfSecTransition">This will transition
in 0.5 seconds</button>
      <button class="hoverPsuedo
twoSecTransition">This will transition
in 2 seconds</button>
      <button class="hoverPsuedo
fourSecTransition">This will transition
in 4 seconds</button>
    </div>
  </div>
</section>
```

CSS Transitions

We're using the second classes (halfsec, twosec, foursec) just to illustrate different applications of transitions.

If a normal scenarios, you wouldn't have multiple transitions, you would just apply them to one CSS selector.

```
<section id="transitionsSection"
class="block">
  <div class="row">
    <div class="col-12">
      <h3>Transitions</h3>
      <button class="hoverPsuedo">This
is a button</button>
      <button class="hoverPsuedo
halfSecTransition">This will transition
in 0.5 seconds</button>
      <button class="hoverPsuedo
twoSecTransition">This will transition
in 2 seconds</button>
      <button class="hoverPsuedo
fourSecTransition">This will transition
in 4 seconds</button>
    </div>
  </div>
</section>
```

CSS Transitions

Things you can try:

- Change the font-size
- Increase the height
- Add a border-left

BREAK!

Pseudo-Elements

What are pseudo-elements?

Pseudo-elements are applied the same way as pseudo-classes, but these focus on styling specific parts of the selected HTML element and has nothing to do with its location on the page.

Pseudo-elements

There are a few ways to select a portion of an element:

1. ::first-letter
2. ::first-line
3. ::before
4. ::after

You can find more here, though these are not used too much: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>



    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Pseudo-elements

There are a few ways to select a portion of an element:

1. ::first-letter
2. ::first-line
3. ::before
4. ::after

Notice that these start with two colons. While one would work, this is used to help distinguish between pseudo-classes and pseudo-elements

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>




    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::first-letter

Selects the first letter in block-level element (doesn’t work with span, a, strong, code and other inline HTML elements)

Usually used for designing drop caps. Let’s look at the in-class examples again.

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p::first-letter {
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::first-line

Work similarly to first-letter, but does have some drawbacks. It is not selecting the first sentence, only the first visible line a in block element

```html
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p::first-line {
        text-transform: uppercase;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before

Creates a element, outside of the DOM. It is mostly used for design touches and to help with user experience.

First, let’s break down how it is applied.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {


      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before

All it technically requires is the content property. It can be a string, a symbol or left empty.

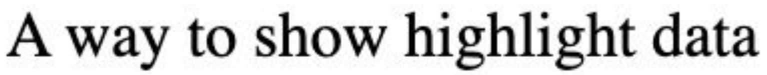
```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "";

      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before

For now, without adding anything, this is what the text looks like.


A way to show highlight data

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "";

      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before

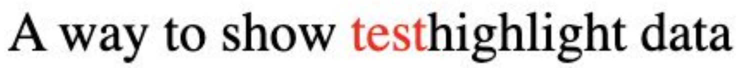
For the purpose of this example, let's use a string ("test") and add color so it's easy to see.

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before

The element is added before the word highlight.


A way to show testhighlight data

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::after

This pseudo-elements works the same way as ::before, it just changes position to after the HTML element selected.

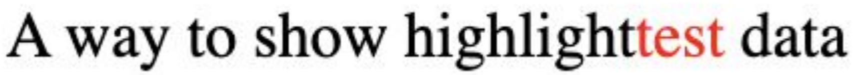
```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::after {
        content: "";

      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::after

The element is added after the word highlight.


A way to show highlighttest data

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::after {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before, ::after

WHY????

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before, ::after

WHY????

It’s actually quite useful!

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before, ::after

Both of these elements are automatically inline

The content within these elements is not accessible to screen readers.

But this is not necessary bad!

```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      .redCircle::before {
        content: "test";
        color: red;
      }
    </style>
  </head>
  <body>
    <article>
      <p>A way to show <span
class="redCircle">highlight</span>
data</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before, ::after

They help add visual cues that are not necessary for screen readers, like a green checkmark next to correctly filled form input or quotes image around <blockquotes>

To add emojis!

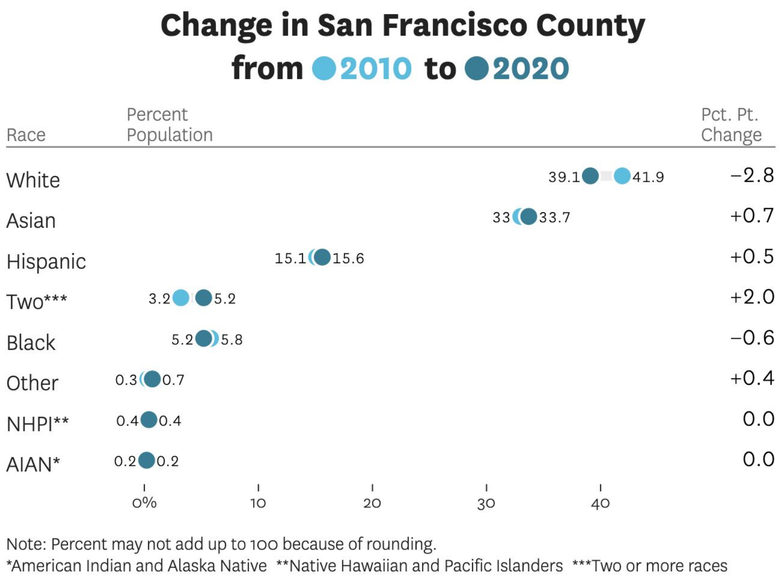
```
<!DOCTYPE html>
<html class= "html" lang="en">
  <head>
    <!-- meta data -->
    <style>
      article p::first-line {
        text-transform: uppercase;
      }
    </style>
  </head>
  <body>
    <article>
      <p>Here is a simple paragraph with
<a
href="https://placekitten.com/300/300"> a
link to a cute kitten</a> photo.</p>
    </article>
  </body>
</html>
```

Pseudo-elements

::before, ::after

I’ve mostly used them for labeling for data viz.

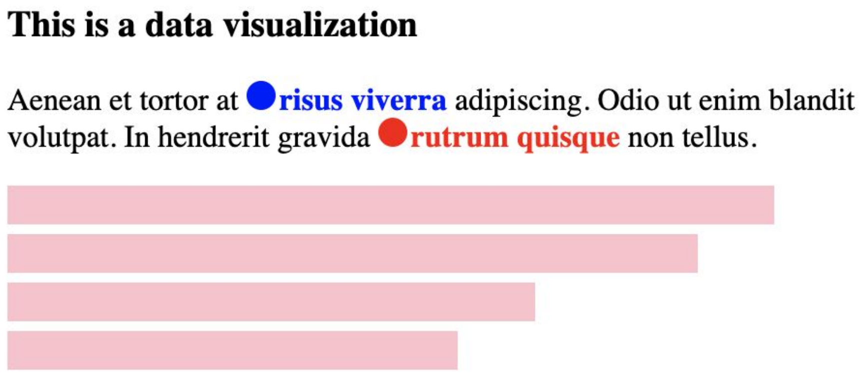
Example: https://www.sfchronicle.com/projects/2021/census-california/



Pseudo-elements

::before, ::after

The code could look something like this. Review section III: Using ::before



This is a data visualization

Aenean et tortor at ●**risus viverra** adipiscing. Odio ut enim blandit volutpat. In hendrerit gravida ●**rutrum quisque** non tellus.

```css
.blueSquare {
  color: blue;
  font-weight: 700;
}
.blueSquare::before {
  content: "";
  display: inline-block;
  height: 15px;
  width: 15px;
  background-color: blue;
  margin-right: 2px;
  border-radius: 50%;
}
```

CSS Display

CSS Display...or an excuse to talk about how to hide stuff on the page

There are times when you want to temporarily hide HTML elements on a page. You might want some HTML elements to show up on hover or after a click.

There are three ways of doing this, each with their own best use case.

Let’s explore!

CSS Display

There are three CSS properties that hide HTML elements from the viewer.

1. Opacity

```css
.box {
  opacity: 0; /* to hide */
  opacity: 1; /* to show */
}
```

CSS Display

There are three CSS properties that hide HTML elements from the viewer.

1. Opacity
2. Visibility

```css
.box {
  visibility: hidden; /* to hide */
  visibility: visible; /* to show */
}
```

CSS Display

There are three CSS properties that hide HTML elements from the viewer.

1. Opacity
2. Visibility
3. Display

```
.box {
  display: none; /* to hide */
  display: inline/block/etcs /* to
show */
}
```

CSS Display

How are they different?

Opacity

- It is still part of the DOM. When you open the inspector
- You can see the HTML element
- It still has pointer events (cursor changes)

```css
.box {
  opacity: 0;
  visibility: hidden;
  display: none;
}
```

CSS Display

How are they different?

Visibility

- It is still part of the DOM. When you open the inspector
- You can see the HTML element
- There are no pointer events

```css
.box {
  opacity: 0;
  visibility: hidden;
  display: none;
}
```

CSS Display

How are they different?

Display

- It is removed from the DOM
- When you open the inspector, it is not there
- There are no pointer events
- Causes the page to shift

```css
.box {
  opacity: 0;
  visibility: hidden;
  display: none;
}
```

CSS Display

Think of what you’re trying to do and the best CSS property for the job.

Take a closer look at section IV: Learning about display

```
.box {
  opacity: 0;
  visibility: hidden;
  display: none;
}
```

CSS Position

CSS Position

There are a few CSS positions: static, relative, absolute, fixed, sticky

When to use a sticky over absolute over fixed is still something I Google!

CSS Position

Static: The default position for HTML elements.

Relative: Usually used for a parent/ancestor element that will have a child that will be using position absolute or relative

Absolute: Using it’s relative parent as it’s container, it requires CSS properties top, bottom, left, right to find its place. It doesn’t move according to the viewport.

Fixed: Also uses top, bottom, left, right, but it is positioned according to the viewport.

Sticky: Using it’s relative parent as it’s container, it requires CSS properties top, bottom, left, right to find its place. It stays with its parent.

CSS Position

Again, this is still something people have to google and look for documentation often. It's also easier to see it in action.

- Notice the fixed position on the navigation bar, it stays with the page.
- The overlays section V (all kittens, all the time!)
- BONUS example of using flexbox and sticky for a simple scrolly in scrolly.html/css

Extra Credit Homework

Extra Credit Homework

Two options:

1 - Redo assignments 5 (can now try the challenge!), 6 or 7

2 - Do Extra Credit Exercise #2 and use CSS learned during this lecture.