Web Development Workshop

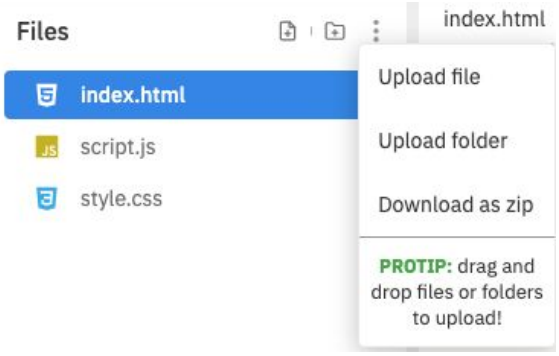
Day 3

Creating an Image that Increases in Size (Zoom in)

- We will create an image that when you click on it, it zooms in, increasing in size. Make sure you have an image ready!

  How to insert your image on repl.it

- Look on the left side where it says “Files”, where it lists “index.html”, “script.js” and “style.css”
  - Next to the title “Files”, you should see three icons, click on the far right one (the 3 vertical dots). A menu should dropdown. Click on “Upload file”. Then, select an image that you had downloaded from the internet or image you already have.



  You can also open your image on your desktop and drag it into the Files column!

- Go to your **index.html** file.
  You are going to write the following code, but instead of `"pizza.png"`, you are going to write what your image is called.

```
<img src="pizza.png" id="photo" onclick="zoom()" width="100" height="100">
```

  An explanation of this code:

  `<img src="pizza.png"`
  - `<img>` tag = tells the computer that we are linking an image to this website. `Src` specifies where the computer can find this image.

  `id="photo"`
  - `id="photo"` identifies the element.

  `onclick="zoom()”`
  - `onclick="zoom()”`= this is a function that allows you to click on the image and makes the image zoom in

  `width="100"`
  - `width="100"` = his makes the image 100px width

`height="100">`

- `height="100"` = this makes the image 100px tall
- `>` = closes our `img` tag

Go to your script.js file and we are going to type this out:

```
var start=0;
function zoom(){
 if(start==0){
   w=photo.width;
   h=photo.height;
   photo.width=photo.width*2;
   photo.height=photo.height*2;
   start=1;
 }else{
   photo.width=w;
   photo.height=h;
   start=0;
 }
}
```

An explanation of this code:

`var start=0;` a variable for when your image is not being clicked on

`function zoom(){` our function to allow us to zoom into our picture

`if(start==0){` when your image is being clicked on

`w=photo.width;` the width of the photo

`h=photo.height;` the height of the photo

`photo.width=photo.width*2;` When we click on image, width 2x as big

`photo.height=photo.height*2;` When we click on image, height 2x

`start=1;` Now the computer knows that the image has been clicked on

`}else{` if we don't click on the image

`photo.width=w;` the photo's width will stay the same

`photo.height=h;` the photo's height will stay the same

`start=0;` The computer knows that the image hasn't been clicked on

`}`

`}`

Change the Color of the Button when Clicked On

- Now, go to your **index.html** file. We are going to type

```
<button id="btn-1" onclick="changeColor(this)"> Click me! I change color!</button>
```

An explanation of this code:

```
<button id="btn-1" onclick="changeColor(this)">
```

  - `button id="btn-1" =` we created a button and identified it as a button, calling it “btn-1”
  - `onclick="changeColor(this)=` we created a function that will allow us to change the color when we click the button

```
Click me! I change color!</button>
```

  - `Click me! I change color!` = label for our button
  - `</button>` = the closing button tag

- Go to the **script.js file**. We are going to create the function that is going to actually change the color of the button. We will create a variable for the color that we want our button to. Create the **var** color.
- You should have this typed out:

```
var color=;
```

- Now, var color has to equal a color. So, we need the hex code of a color. Pick any color you like! You can use this website to pick a color: https://htmlcolorcodes.com/ . After you pick a color that you like, copy the hex code.
- It should look like this:

```
var color="#F8C7CC";
```

- Then, let’s create the function **changeColor()** that will actually change the color of our button.
- It should look something like this now:

```
var color="#F8C7CC";
function changeColor(){
}
```

- We need to tell the computer what we want to change the color of. So, between the parentheses of **changeColor()**, type in **btn,** which we already identified as being a button in the html file.
- It should look something like this:

```
var color="#F8C7CC";
function changeColor(btn){
```

```
}
```

- We are almost done! Just one more line of code to go! Now, in between our function **changeColor(),** we need to tell the computer to switch the color of the button to what we want. So, between the curly brackets `{}`, type in `btn.style.backgroundColor= color;`
- It should look something like this:

```
var color="#F8C7CC";
function changeColor(btn){
 btn.style.backgroundColor= color;
}
```

Having an Image that Changes into Another Image

- We are now going to create an image that when you hover over it, it changes into another image. So, go to your **html file** once again. You are going to type this line of code:

```
<img src="goals.png" id="image" height="100" width="100"
onmouseover="newPicture()" onmouseout="oldPicture()">
```

  - `onmouseover="newPicture()"` = when the mouse is on the image, you will see the new picture
  - `onmouseout="oldPicture()"` = when the mouse is not on the image, you will see the old picture.

For the img src part, make sure you put the name of the image that you are going to use!

- Go to your **javascript file** now. We are going to create two functions. One for the picture we will see without any hovering and another function for the picture that we want to see when we hover over the first picture.
- Our first function will be for the new picture (the picture that we want to see when we hover over the first picture).

  Create your **function newPicture().**

  - It should look like this:

```
function newPicture(){
}
```

- Then, in order for the computer to know what the new picture is supposed to be, we will type in this in between the curly brackets:

```
document.getElementById("image").src="books.png";
```

For the img src part, make sure you put the name of the image that you are going to use!

- It should now look like this:

```
function newPicture(){
 document.getElementById("image").src="books.png";
}
```

For the img src part, make sure you put the name of the image that you are going to use!

- Our second function will be for the old picture (the picture we will see without any hovering).

  Create your **function oldPicture().**

  - It should look like this:

```
function oldPicture(){
}
```

- Then, in order for the computer to know what the new picture is supposed to be, we will type in this in between the curly brackets:

```
document.getElementById("image").src="goals.png";
```

For the img src part, make sure you put the name of the image that you are going to use!

- It should now look like this:

```
function oldPicture(){
 document.getElementById("image").src="goals.png";
}
```

- It should all look like this now:

```
function newPicture(){
 document.getElementById("image").src="books.png";
}
function oldPicture(){
 document.getElementById("image").src="goals.png";
}
```

Adding Animation Effects

- In your CSS file, we can create *global* animations — any element can use them!
- Some key components to recognize in the code snippet below:

- `@keyframes` is a keyword (more formally called a rule) that lets the CSS compiler know that you're defining an animation
- `bounceIn` is the name of your animation (call it anything you want!)
- `0%`, `60%`, and `100%` are selectors that define what the animation will look like. At the start of the animation, the element will be transparent and 0.1 of its original size. When the animation is 60% done, the element will be fully visible and 1.3 of its original size. At the end of the animation, the element will have transformed into its original size.

```
@keyframes bounceIn {
 0% {
   transform: scale(0.1);
   opacity: 0;
 }
 60% {
   transform: scale(1.3);
 }
 100% {
   transform: scale(1);
 }
}
```

- Then, a class can be created to apply the animation to any element

```
.animated {
  animation: bounceIn 2s;
}
```

- Finally, the class can be added to the HTML

```
<h1 class="animated">All About Me</h1>
```

Adding Hover Effects to Images

- We can use CSS to give images effects

```
.mainImage:hover {
 transform:scale(.99);
 opacity:.7;
 transition: all 0.5s ease;
}
```

- Above, the class `.mainImage` has a `:hover` selector that defines what actions will be performed when the user hovers over the element.
- To make the transition smooth, we can modify the code to look like:

```
.mainImage {
 transition: all 0.5s ease;
}
.mainImage:hover {
 transform:scale(.99);
 opacity:.7;
 transition: all 0.5s ease;
}
```

Linking Elements Together

- We'll now create a text box that appears when the user hovers over the main image by creating a CSS class like the one below. You may need to adjust the values of `width`, `height`, `top`, and `left` depending on your screen size.

```
#text-box {
 position: absolute;
 opacity:0;
 width:300px;
 height:200px;
 top:55%;
 left:20%;
}
```

- Next, we need to link main to text. The code snippet below is NOT an extension to the previous `.mainImage:hover` block; they need to be separate.

```
.mainImage:hover + #text-box {
 opacity:1;
}
```

- We can now change the HTML:

```
<img src="{image link}" alt="Rutgers" class="mainImage"></img>
<div id="text-box">
 At Rutgers University, I'm a part of the Girls Who Code club!
</div>
```

- At this point, you should be able to see the text box when you hover over your main image. If you don't, try adjusting the `top` and `left` properties of `#text-box`.
- Final step: make your text box fancy!

```
#text-box {
 position: absolute;
```

```
opacity:0;
width:300px;
height:200px;
top:55%;
left:20%;
font-size:30px;
color: #000;
background-color: #fff;
border:1px dashed#000;
text-align:center;
padding:20px;
}
```

Adding a JavaScript Clock

- Let's add a clock to your website.
- In script.js, define a function called GetClock()

```
function GetClock(){
    var d = new Date();
}
```

- `new Date()` gets the current date in military time format. We can get the different components of time to make our clock more visually appealing and easy to read:

```
function GetClock(){
    var d = new Date();
    var cmonth = d.getMonth(); //current month
    var cdate = d.getDate(); //current date
    var cyear = d.getFullYear(); //current year
    var chour = d.getHours(); //current hour
    var cmin = d.getMinutes(); //current minute

    var clocktext = "Current time: "+cmonth+" "+cdate+",
       "+cyear+" "+chour+":"+cmin;
    document.getElementById('clockbox').innerHTML=clocktext;
}
```

- Adding the following snippet anywhere in the <body> of your HTML file will put the clock on your website:

```
<div id="clockbox"></div>
```

```
<script type="text/javascript">
     GetClock();
     setInterval(GetClock,1000); // The clock will run once
                                 every 1000 milliseconds (1 second)
</script>
```

- From here, we just make the clock easier to read
- Let's add AM or PM to the reading, as well as a statement that "fixes" the minutes

```
function GetClock(){
     var d = new Date();
     var cmonth = d.getMonth(); //current month
     var cdate = d.getDate(); //current date
     var cyear = d.getFullYear(); //current year
     var chour = d.getHours(); //current hour
     var cmin = d.getMinutes(); //current minute
     var AMorPM;

     //if it's 7:02, the time will read 7:2
     //the if statement below fixes that problem
     if(nmin<=9)
        nmin="0"+nmin;
     if(chour==0){
        AMorPM = " AM";
        nhour=12;
     }
     else if(chour < 12){
        AMorPM = " AM";
     }
     else if(chour == 12){
        AMorPM = " PM";
     }
     else if(chour > 12){
        AMorPM = " PM";
        nhour-=12;
     }
```

```
        var clocktext = "Current time: "+cmonth+" "+cdate+",
           "+cyear+" "+chour+":"+cmin + AMorPM;

        document.getElementById('clockbox').innerHTML=clocktext;
    }
```

- Run your code again — do you see the change?
- Finally, we can change the month number into the month name and get this as our final function:

```
    function GetClock(){
        var d = new Date();
        var cmonth = d.getMonth(); //current month
        var cdate = d.getDate(); //current date
        var cyear = d.getFullYear(); //current year
        var chour = d.getHours(); //current hour
        var cmin = d.getMinutes(); //current minute
        var AMorPM;

        var monthWords = ["Jan", "Feb", "Mar", "Apr", "May",
        "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];
        if(cmin<=9)
           cmin="0"+cmin;
        if(chour==0){
           AMorPM = " AM";
           nhour=12;
        }
        else if(chour < 12){
           AMorPM = " AM";
        }
        else if(chour == 12){
           AMorPM = " PM";
        }
        else if(chour > 12){
```

```
        AMorPM = " PM";
        nhour-=12;
    }
    var clocktext = "Current time: "+monthWords[cmonth]+"
        "+cdate+", "+cyear+" "+chour+":"+cmin + AMorPM;
    document.getElementById('clockbox').innerHTML=clocktext;
}
```

- What do these changes do?
    - **monthWords** is an array of words (AKA strings). Meanwhile, **cmonth** is the current month represented by a number (note that counts typically start at 0, so Jan = 0, Feb = 1, Mar = 3, etc.)
    - **monthWords[cmonth]** gets the **cmonth**-th word in the **monthWords**. It's currently September, and **cmonth** = 8. **monthWords**[8] is the 8th element in **monthWords** (starting at 0), which is "Sep"