Chris Wray Project 1 Overview

Project turned out to be my Make-A-Thing weekend project with the extended functionality of another API integration into the program with the ability to spawn Meme text on the event of a subsequent button press. The main pain points with this program involved learning about how to link in a second API with out interfering with the first, getting the generate meme button press event working for both at the same time, fire modification, and linking the fire code.

Index.html:

The main "**index.html**" page is pretty spartan as you have a section for holding the 3 main portions: the button for meme generation, image containers, text container, and picture inversion button. There are also two paragraph tags for displaying the meme & text. An interesting issue is that with out the invocation of "**meme_name**" the original API cannot be invoked correctly. Using "**hidden**" as a modification of that tag may function to hide the meme text correctly, but that is not currently integrated. P5 and app.js are finally invoked at the bottom to properly link into the whole program structure.

```
#1_Me > <> index.html > ...
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Meme Teller</title>
    <link href="style.css" rel="stylesheet" />
  </head>
  <body>
    <div class="meme_container">
      <button class="button_meme" id="button">New Meme</button>

      <div class="img_container">
        <img class="memeImg" id="meme_img" />
      </div>

      <div class="text_container">
        <img class="textRead" id="text_read" />
      </div>

      <!--Container for displaying the random meme text on screen-->
      <!-- <div class="meme_text_container">
        <img class="memeTxt" id="meme_txt" />
      </div> -->

      <p id="meme_name"></p> <!-- Must be active in order to display the random meme img -->
      <p id="meme_text"></p> <!-- This is where the 2nd Meme API text will go -->
    </div>

    <!-- <button class="button_memetext" id="newmemetext_b">
      Meme Text
    </button> -->

    <button class="button_surprise button_meme" id="surprise_b">
      Surprise!
    </button>

    <script src="https://cdn.jsdelivr.net/npm/p5@1.1.9/lib/p5.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

App.js:

The core of this creation is the app.js code and everything attached to it. We start out with invocations of the core variables for the programs and the first two fetch statements that are intended to hook into "**api.imgflip.com/get_memes**" & "**meme-api.herokuapp.com**". They follow a similar path with the usage of "**data.data.memes.length**" & "**data.memes.length**" to create a random meme variable. In both cases another variable is created to connect the html page elements to the back end with "**getElementById**". In the first iteration, a call is then invoked to the new variable to generate a random image, and these are loaded with "**loadImage**".

In the second block of code, a similar process is attached to a new click event that uses "**getElementById**" to access the click event of the button on "**index.html**". This is done because the data must be loaded by the other segment of code before it can be used, but the process is mostly the same.

```
//refreshes the memeText with the a meme button press
//refreshes the meme with the meme button
let memeButton = document.getElementById("button");
memeButton.addEventListener("click", function () {

  fetch("https://meme-api.herokuapp.com/gimme/15?action=render")
    .then(response => response.json())
    .then(data => {
      console.log(data.memes[0].title); // ".title" did the trick to render the "Title" portion of the 2nd Meme API

      // Process this API portion here...
      memeTextData = data.memes.title
      randomTextMeme = Math.floor(Math.random() * data.memes.length);
      console.log(randomTextMeme);

      memeText = document.getElementById("meme_text");
      memeText.innerHTML = data.memes[randomTextMeme].title;
    })
  fetch("https://api.imgflip.com/get_memes")
    .then(response => response.json())
    .then(data => {
      console.log(data.data.memes)
      memeData = data.data.memes
      randomMeme = Math.floor(Math.random() * data.data.memes.length);

      memeName = document.getElementById("meme_name");
      memeName.innerHTML = memeData[randomMeme].name;

      memeImage = document.getElementById("meme_img")
      memeImage.src = memeData[randomMeme].url;

      memeURL = memeData[randomMeme].url
      p5Image = loadImage(memeURL)
    })
})

//creates the surprise button and links it to the inverse display function
let surpriseButton = document.getElementById("surprise_b");
surpriseButton.addEventListener("click", function () {
  clickedB = !clickedB
  console.log(clickedB)
})

// ==============================
// Use "let" everywhere in teh code... as opposed to let && var...
// ==============================
```

The next portion invokes the fire, which was a blob of code I dug up to make the page more interesting as the original context of this involved the theme of "Apocalypse" which seemed to immediately invoke fire in my minds eye. I originally tried to link the code externally, but has issues getting it working correctly. It was recommended that it just be linked into the page and it worked to my surprise. This direct linkage also made fire height easier as "**fireElemLenght**" managed its variability.

The final relevant portion was the draw section which simply invoked various portions of the fire code to properly draw it onto the screen along with the image & tint sections to inverse the pallet of the meme images, and scale it properly.

```
function draw() {
  // We clean the background each time
  background(0, 0, 0);

  // We generate a new base fire line (to make it 'move')
  initFireLine();

  // Compute the whole fire pixels
  fireGrow();

  // Display the result
  drawFire();

  //IMAGE distortion
  image(p5Image, 0, 0)
  // filter(INVERT);
  image(p5Image, 0, 200)
  tint(255, 92, 53, 126)

  push();
  scale(0.5);
  image(p5Image, 200, 0);
  filter(POSTERIZE, 15);
  pop();

  //inverts image if "Surprise" button pressed
  if (clickedB == true) {
    filter(INVERT);
  }
}

function initializePalette() {
  // generate our 255 color palette
  // try to change colors composing the fire
  for (var i = 0; i < nbColors; i++) {
    var val = exp(i / 10) - 1;

    var red = map(val, 0, exp(7.5), 0, 255);
    var green = map(val, 0, exp(10), 0, 255);
    var blue = random(50);

    if (green > 254) // check for colors range
    {
      red = 255;
      green = 255;
      blue = 255;
    }
```

Style.css:

The body section of the stylesheet for this experience uses an image that I found from "**unsplash.com**" which was open sourced from Diana Vargas, and it seemed to invoke the mood that seemed to fit the theme that I was going for. The next stylesheet containers in the upper portion establish the baselines for the core elements of the experience. That includes the frames for the memes, text, and buttons. The second section shapes the second meme text portion, button text, text read, and canvas aspects. Admittedly, there were more experiments on my part in the latter portion and so the I had to roll back some changes to keep a clean look.

```css
body {
  background-image: url("diana-vargas-ZySVEbGBNxA-unsplash.jpg");
  background-position: center;
  margin: 15px 30px;
  display: flex;
  flex-direction: row;
  align-items: flex-start;
}

.meme_container {
  width: 100%;
  height: 100%;
}

p {
  font-family: 'Impact', impact;
  font-size: 25px;
  letter-spacing: 2.5px;
  font-weight: 500;
  color: #FBF4F4;
  /*outline: 3px solid;*/
}

.memeImg {
  margin-top: 20px;
  width: 75%;
  height: 75%;
}

.img_container {
  width: 50%;
  height: 50%;
}

.button_surprise {
  margin-right: 350px;
}

/* ------------------------------------- */

.meme_text {
  font-family: 'Impact', impact;
  font-size: 25px;
  letter-spacing: 2.5px;
  font-weight: 500;
  color: #000;
  outline: 3px solid;
}

.textRead {
  margin-top: 40px;
```

```css
/* ------------------------------------- */

.meme_text {
  font-family: 'Impact', impact;
  font-size: 25px;
  letter-spacing: 2.5px;
  font-weight: 500;
  color: #000;
  outline: 3px solid;
}

.textRead {
  margin-top: 40px;
  width: 55%;
  height: 55%;
}

.button_text {
  flex-direction: row;
  padding: 10px;
}

.button_meme {
  display: flex;
  flex-direction: row;
  justify-content: center;
  padding: 10px;
}

canvas {
  margin: 50px 0px 0px -650px;
}
```

Conclusion:

The visual experience here was prompted my the Make-A-Thing project that I signed up for and my work with Mathura and Christina was core to understanding how to properly use two APIs in one program, and go about the basic API calls. I enjoyed this because while I spent far too much time playing around with it – this represents the act of this class's new-to-me portions finally starting to click into place.