Group Members: Parabjot Chander, Benjamin Mushyakov, and Andrew Zheng
CSCI 355: Internet and Web Technologies
Professor Fried
March 28, 2023

Gray Paper

<u>5 Salient Points from the Code (challenges you encountered and how you solved them):</u>

1. **Clothing Items Display (Image Slider)**
   Items that our business sells (Basketball Apparel) in the Mens and Womens webpage are displayed using a CSS grid layout with 4 columns and 8 rows, with div elements nested inside a big div representing the grid container. Each div within the grid container contains a category or the image of the item (html img tags), price and sizes available (html paragraph tags). Each category of an item takes the whole row (e.g. tops, bottoms, sneakers and accessories).

   <u>Important CSS Code Used:</u>

   ```
   Grid-Container (big div html element):
           display: grid
           grid-template-rows: 1fr 1fr 1fr 1fr
           grid-template-column: 0.3fr 1fr 0.3fr 1fr 0.3fr 1fr 0.3fr 1fr

   Image-Containers (divs containing item image, price and size)
           Border: 2px solid black;
           Image CSS =>  .mdisplay { width: 80%; height: 250px;}
   Category-Containers (divs with category of clothing displayed below)
           text-decoration: underline;
           grid-column: 1/-1;
           text-align: center;
   ```

2. **Udemy Idea: Incrementing Counter**
   When the home page loads, the dates for when basketball, NBA and WNBA were created or founded are displayed, but the dates/numbers start at zero and increment until the target score which are the actual dates reach the data target html div attribute. The function updateCounter() is repeatedly running while the page is loaded. The counters are updated every 30 ms by target divided by 200;

   <u>Important JavaScript Shown</u>:

   ```
    const updateCounter = () => {
           const target =  +counter.getAttribute('data-target');
           const c = +counter.innerText;
           const increment = target /200;
           if ( c < target){
   ```

```
                counter.innerText = `${Math.ceil(c + increment)}`;
                setTimeout(updateCounter, 30);
        }
        else{
                counter.innerText = target;
        }
}
```

3. **Udemy Idea: Scrolling Animation**
   There are 6 box div's, either with an image or a quote that as the user scrolls down the page, an event listener is triggered where 4 boxes appear either from the left (box is even number) or the right (box is odd number) appear on the screen on top of each other. The event listener calls the checkBoxes() function, the trigger point is calculated using window.innerHeight /5 *4 and the div box's positions relative to the viewport are known by the getBoundingClientRect() helper method. The logic is that for each div box, is the viewport position is less than the trigger point, we show the box or else we hide the boxes by making them translateX() by -400%.

   <u>Important JavaScript Shown:</u>

```
box.forEach( box => {
        const boxTop = box.getBoundingClientRect().top;
        if( boxTop < triggerBottom ){
                box.classList.add('show');
        }
        else{
                box.classList.remove('show');
        }
})
```

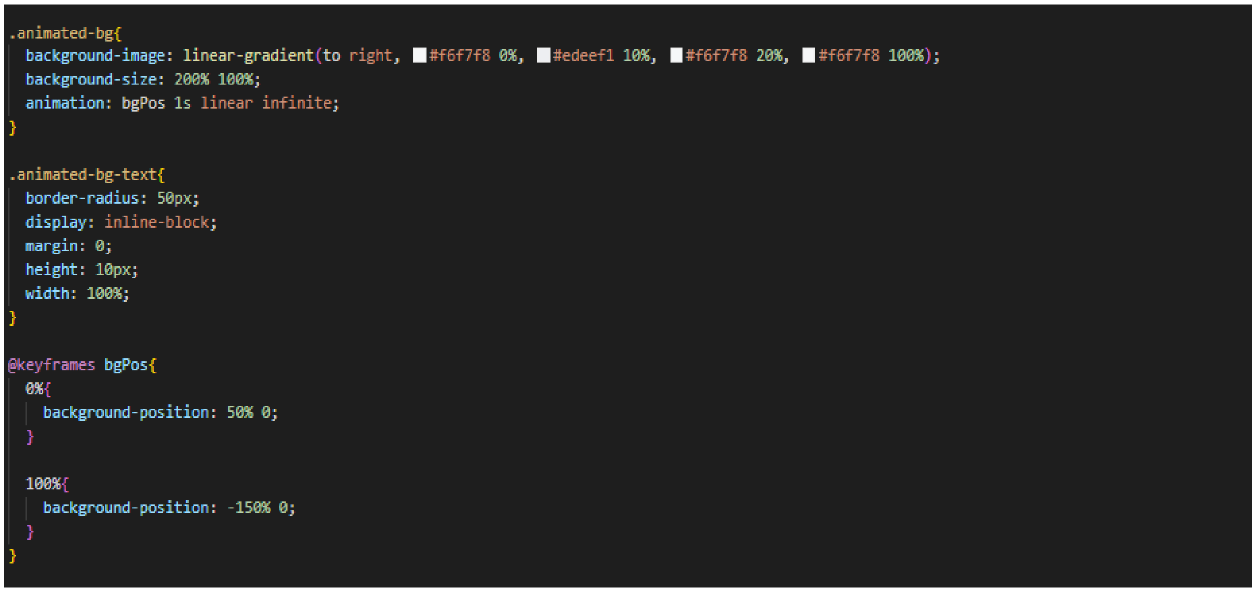
4. **Udemy Idea: Kinetic CSS Loader**
   For this udemy idea, this is strictly in CSS, it is loop based animation so it used before: and after: elements to show the start and end of it. There's a div for the basketball and the lines of ball for elements of the ball. It used animation-iteration is set to infinite to repeat. The @keyframe is used to define the animation behavior/positioning by setting each %. For example at 25%, it is 185px and at 90 degree angle and within 2 seconds, it repeats and goes through each frame of the basketball making the ball bounce up and down.
   CSS Loader Code:

```
.basketball {
   position: relative;
   top: 100px;
   right: -2px;
 }
 .ball {
```

```css
  position: relative;
  background-color: #e76f51;
  border-radius:50%;
  width:100px;
  height:100px;
  overflow: hidden;
  border: 3px solid #333;
  animation: bounceball 2s ease-in infinite;
}
.ball:before, .ball:after {
  content:"";
  position: absolute;
  background-color: #333;
  width:110px;
  height:3px;
  top:50px;
  left:-5px;
}
 .ball:before {
  transform: rotate(45deg);
}
 .ball:after {
  transform: rotate(-45deg);
}
 .lines {
  position: absolute;
  border-radius:50%;
  border: 3px solid #333;
  width:70px;
  height:70px;
  left:-20px;
  top:-20px;
}
 .lines:before {
  content:"";
  position: absolute;
  border-radius:50%;
  border: 3px solid #333;
  width:70px;
  height:70px;
  top:65px;
  left:60px;
```

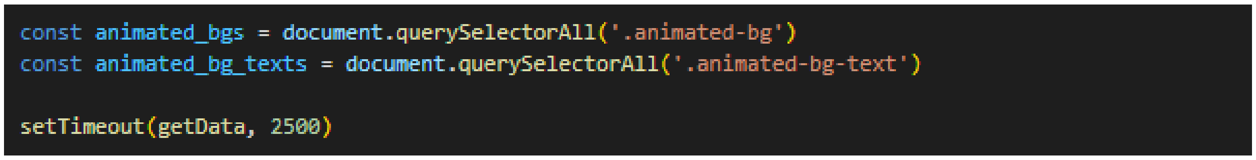
```
        }
        @keyframes bounceball {
          0% {transform: translate3d(0, 0, 0) rotate(0deg);}
         25% {transform: translate3d(0, -185px, 0) rotate(90deg);}
         50% {transform: translate3d(0, 0, 0) rotate(180deg);}
         75% {transform: translate3d(0, -185px, 0) rotate(270deg);}
         100% {transform: translate3d(0, 0, 0) rotate(360deg);}
       }
```

5. **Udemy Idea: Animation Timeout**
   This udemy idea was used for the Women's page, and displays a loading animation over the image and description of each item in said page for 2.5 seconds. CSS is used to create the animation, via keyframes, that lasts one second and is infinite (continuously repeats itself). JS is used to load the images and descriptions back in once 2.5 seconds pass via a call to setTimeout. It does this by finding all elements with the animated-bg and animated-bg-text classes and removing those classes from those elements with a for each loop.
   Noteworthy CSS:

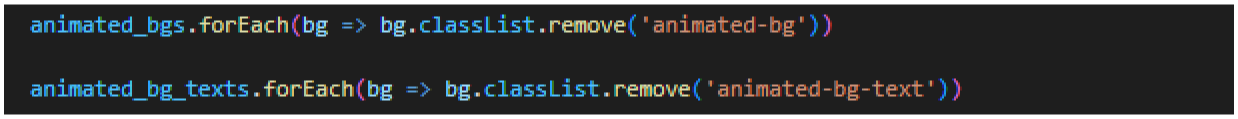


```
.animated-bg{
  background-image: linear-gradient(to right, #f6f7f8 0%, #edeef1 10%, #f6f7f8 20%, #f6f7f8 100%);
  background-size: 200% 100%;
  animation: bgPos 1s linear infinite;
}

.animated-bg-text{
  border-radius: 50px;
  display: inline-block;
  margin: 0;
  height: 10px;
  width: 100%;
}

@keyframes bgPos{
  0%{
    background-position: 50% 0;
  }

  100%{
    background-position: -150% 0;
  }
}
```

   Noteworthy JS:



```
const animated_bgs = document.querySelectorAll('.animated-bg')
const animated_bg_texts = document.querySelectorAll('.animated-bg-text')

setTimeout(getData, 2500)
```



```
animated_bgs.forEach(bg => bg.classList.remove('animated-bg'))

animated_bg_texts.forEach(bg => bg.classList.remove('animated-bg-text'))
```

(Note: there is more JS but it is very dense, yet trivial to explain. Specific html elements are found by their given IDs and in the getData function have their unique content returned after the 2.5 second animation delay. The content is returned using the innerHTML property.)

5. Challenges

1. **Node Express JS (**Package.json and node module were self generated)**:**

When implementing a node server for our website to connect to a localhost, there were some issues in terms of connecting to the host. There were problems such as port number already being used or conflicts of too many redirects and the module not being available. So for the port number problem, I try to use "kill to terminate any process going" but it didn't work initially. Therefore, I changed the port number to 5555 instead of 5000. Then, there was a problem of too many redirects of the page, I'm assuming it's trying to get all html pages at once and sometimes, the module isn't being located. Therefore, I added an route to each of the files to our directory in order for the host to get a response and request from the server side of each html with their css/js to the client side.

Server.js Code:

```
const express = require('express');
const path = require('path');
const app = express();

// Using the files from the directory
app.use(express.static(path.join(__dirname, 'WebSite Files')));

//Route to each html pages
app.get('/', (req, res) => {
 res.sendFile(path.join(__dirname, 'WebSite Files', 'home.html'));
});

app.get('/men', (req, res) => {
 res.sendFile(path.join(__dirname, 'WebSite Files', 'men.html'));
});

app.get('/women', (req, res) => {
 res.sendFile(path.join(__dirname, 'WebSite Files', 'women.html'));
});
```

```
// Port 5555 is where we will test our node app on localhost
app.listen(5555, () => console.log('Ready on port 5555'));
```

6. Contact Form Information

When the user submits the contact form in the About us page, the info gets sent to https://httpbin.org/get using a GET method. The get method is used to request or send information from a specific source (URL), since the form doesn't ask for a password, GET method is used rather than POST (not ideal to use GET when we have sensitive information).

Important HTML Form Code:
<form action="https://httpbin.org/get " method="GET" **…**>
………..
</form>

7. BasketBall Animation CSS

At the bottom of the Women's page, an animation div container is created where we have a woman shooting a basketball repeatedly. The basic explanation is that a CSS animation is created where at 25%, 50%, 75%, 100% time intervals of the animation, the basketball (div element rounded with border radius set to 50px) changes its position so when the animation plays, the basketball moves from one place to another. The female basketball player and the basketball hoop are positioned relative to the animation div container. The animation occurs 5 times, so if you load the page, and don't scroll down for a minute or two, the animation will finish and you won't see it.