ICP 3: RESPONSIVE WEB DESIGN

Joe Moon
Email: jmn5y@umsystem.edu
Github: https://github.com/joemoon-0/WebMobile-2022Spring/tree/main/2022Spring-Web/Web_ICP3

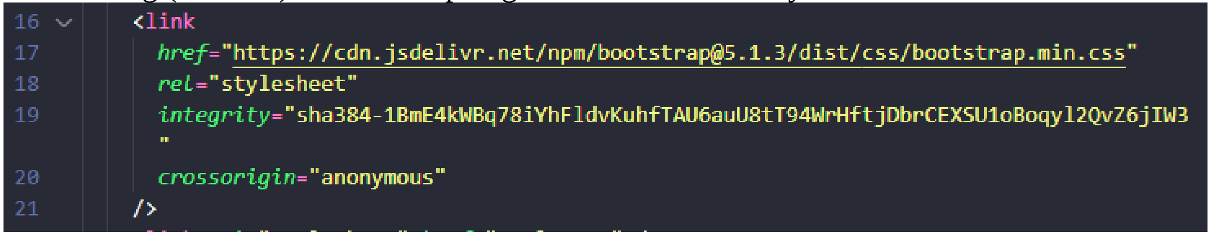
Nathan Cheney
Email: ncxn8@umsystem.edu
Github: https://github.com/nathancheney/Web-Mobile-Programming/tree/main/ICP-3/

Introduction

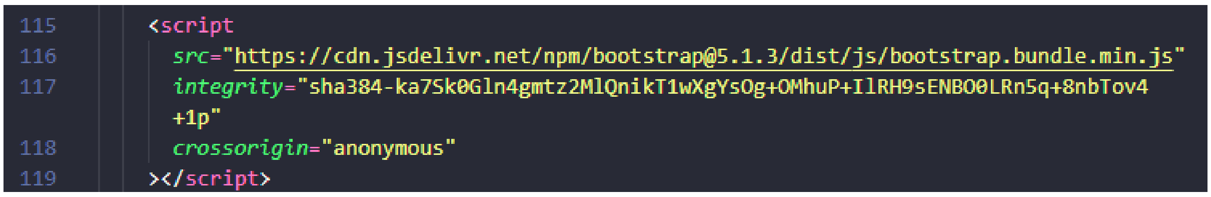
As mobile device use continues to rise, overtaking traditional desktop and laptop viewing platforms, the need for responsive designs becomes equally important so that a webpage's content will render correctly regardless of screen size. Bootstrap 5 is one tool that can help achieve such results by using predefined classes which were made with responsiveness in mind. Combined with JavaScript, which gives a webpage its functionality, web apps can be created for use on all platforms.

RWD Task

This task was intended to recreate a given mock up using Bootstrap classes along with provided images and a color palette. Bootstrap was integrated using the CDN links placed in the head tag (for CSS) and the script tag at the end of the body.

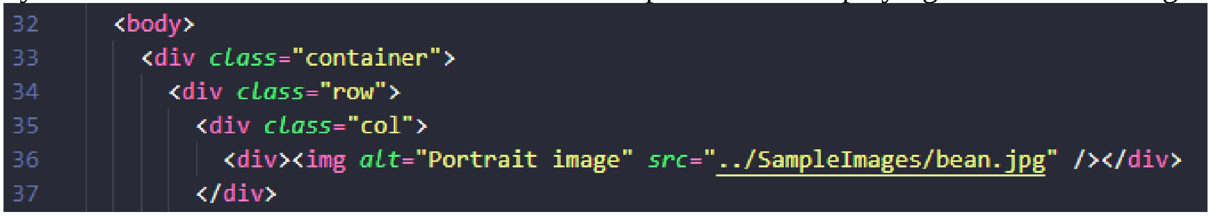


```
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3
  "
  crossorigin="anonymous"
/>
```



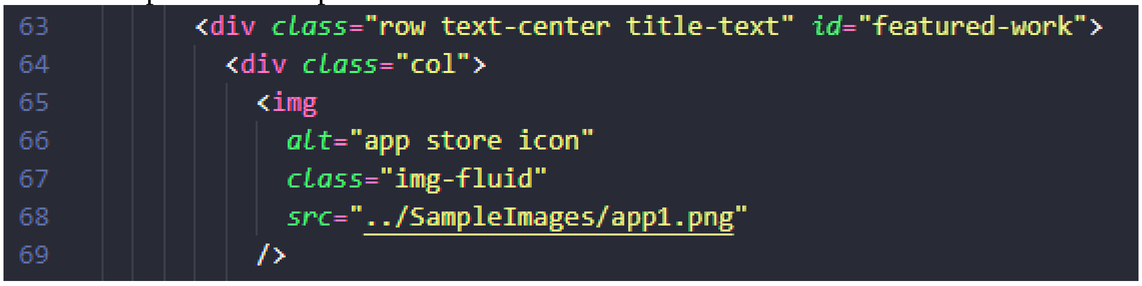
```
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4
  +1p"
  crossorigin="anonymous"
></script>
```

In order to recreate the given mock-up, the grid layout system was used in combination with containers, rows, and columns to give the page it's structure for the content that it would host (mostly images). Each container would consist of several rows, followed by a column for an item. Below is one such example used for displaying the Mr.Bean image:



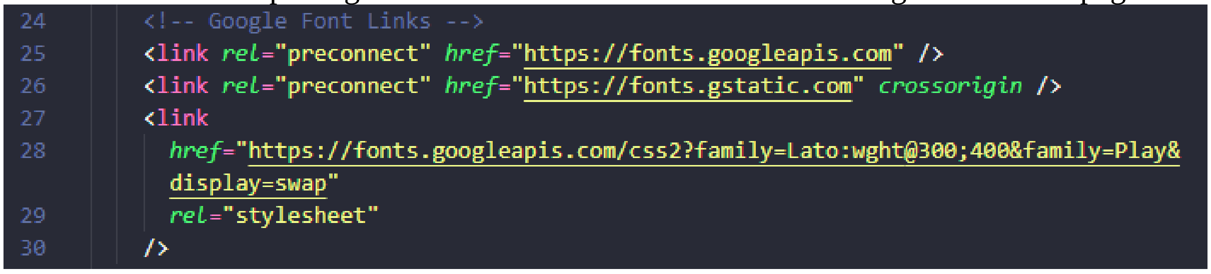
```
<body>
  <div class="container">
    <div class="row">
      <div class="col">
        <div><img alt="Portrait image" src="../SampleImages/bean.jpg" /></div>
      </div>
```

Since the webpage primarily consisted of images, it was important to ensure that the images remained responsive to changing screen sizes. As such, the "img-fluid" class was used which gives the image the properties of **max-width: 100%** and **height: auto** to ensure that it occupies the full space of the column that it is located in as done on line 67.



```
63            <div class="row text-center title-text" id="featured-work">
64              <div class="col">
65                <img
66                  alt="app store icon"
67                  class="img-fluid"
68                  src="../SampleImages/app1.png"
69                />
```

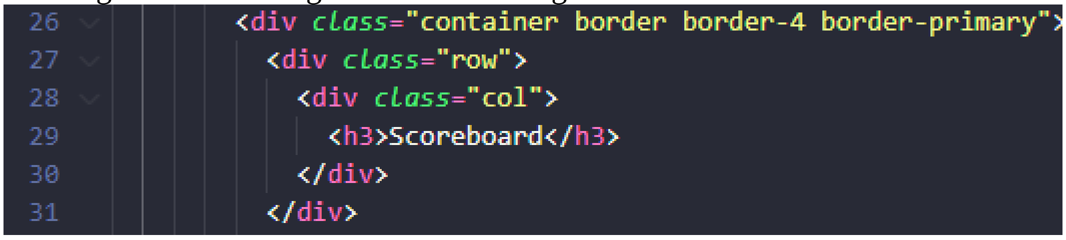
A custom style sheet was used to implement various other features such as text coloring, however the fonts were imported using Google fonts. Specifically, API links were used so that the font packages would not have to be downloaded along with the webpage.



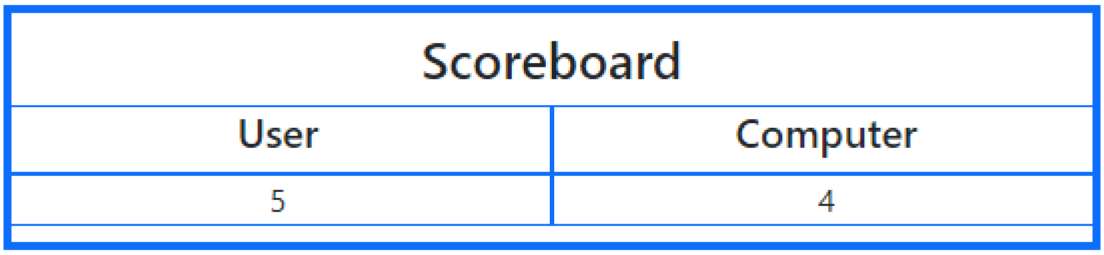
```
    <!-- Google Font Links -->
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
      href="https://fonts.googleapis.com/css2?family=Lato:wght@300;400&family=Play&display=swap"
      rel="stylesheet"
    />
```

Rock, Paper, Scissor (RPS) Game

The RPS game employs Bootstrap classes for the grid layout system and its responsiveness to screen sizes while using JavaScript to provide it gameplay functionality. Much like the RWD task, the RPS game uses containers, rows, and columns to create the UI for the game while using border classes to give it visual structure:

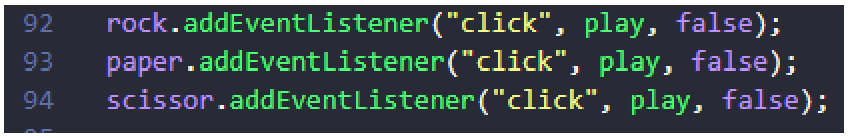


```
        <div class="container border border-4 border-primary">
          <div class="row">
            <div class="col">
              <h3>Scoreboard</h3>
            </div>
          </div>
```



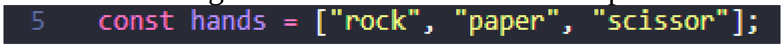
Scoreboard	
User	Computer
5	4

For gameplay, event listeners are used to determine which hand the user selects from the UI before calling the **play** function which drives the game.

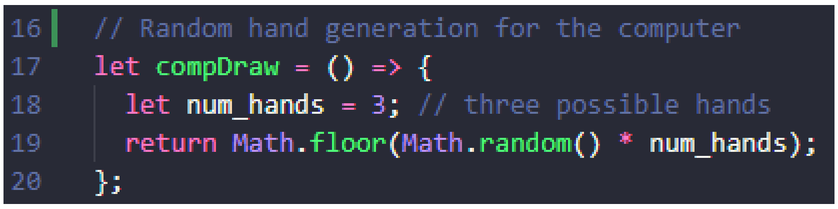


```
rock.addEventListener("click", play, false);
paper.addEventListener("click", play, false);
scissor.addEventListener("click", play, false);
```

The game uses a random number generator to determine the computer's hand from a predefined array:

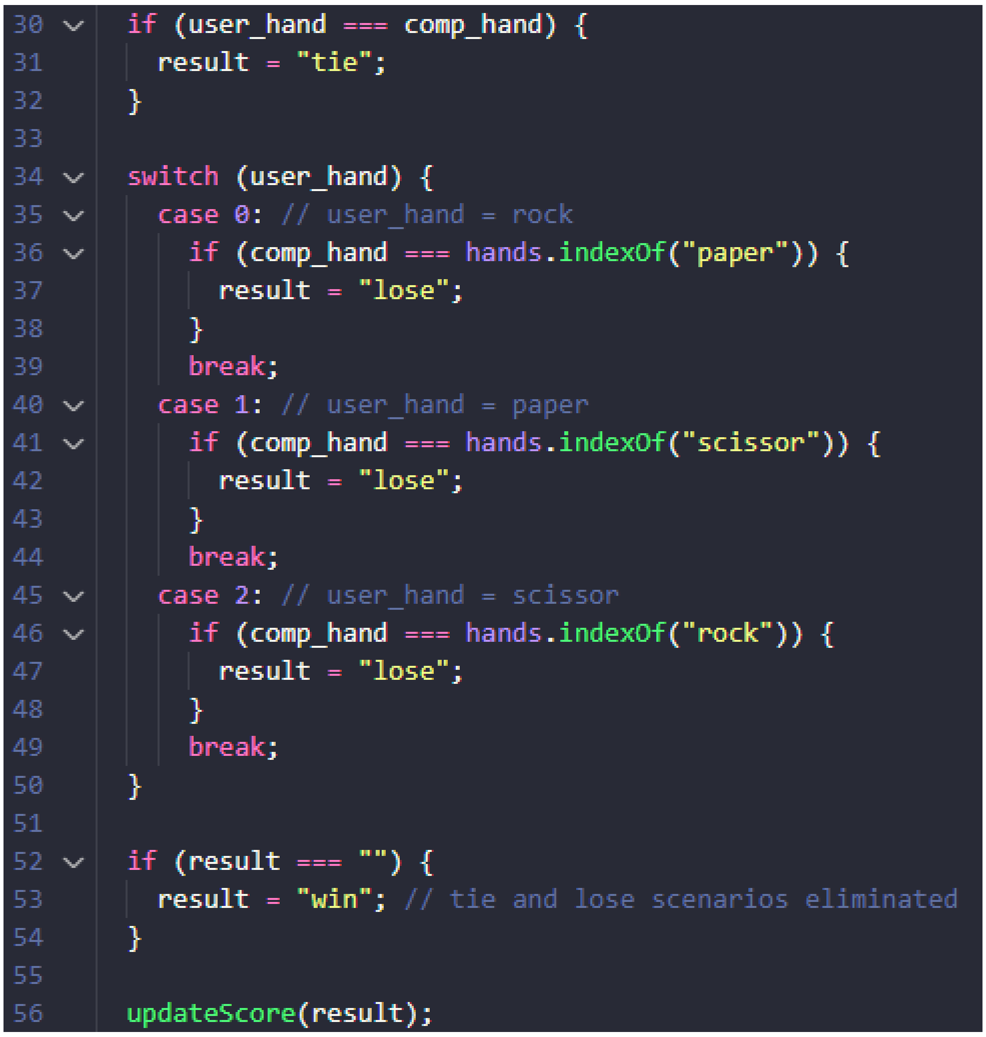


```
const hands = ["rock", "paper", "scissor"];
```



```
// Random hand generation for the computer
let compDraw = () => {
  let num_hands = 3; // three possible hands
  return Math.floor(Math.random() * num_hands);
};
```

Once the user and computer's hand is known, a series of comparisons are performed in order to determine the winner/loser (or tie) before such information is displayed on the UI by calling the **updateScore()** function.



```
if (user_hand === comp_hand) {
  result = "tie";
}

switch (user_hand) {
  case 0: // user_hand = rock
    if (comp_hand === hands.indexOf("paper")) {
      result = "lose";
    }
    break;
  case 1: // user_hand = paper
    if (comp_hand === hands.indexOf("scissor")) {
      result = "lose";
    }
    break;
  case 2: // user_hand = scissor
    if (comp_hand === hands.indexOf("rock")) {
      result = "lose";
    }
    break;
}

if (result === "") {
  result = "win"; // tie and lose scenarios eliminated
}

updateScore(result);
```

Contribution

All members contributed equally to this report and ICP.