In this lab we are going to take what we have learned about both HTML and JavaScript and use it to create an attractive interactive page.

Today we will create a web page that lets you battle against the Joker. In this game we will be the Justice League consisting of Batman, Superman, Flash and Wonder women who are the last defendants against the evil Joker. We will be able to select among the four fighters each time to attack. To do this, we will explore the use of JavaScript functions as well as revisit HTML and CSS.

A. Setting up the HTML

We’re going to set up our basic page HTML. Most of the tags should be familiar to you, as you have used them in previous labs and homework assignments. If you are having trouble, revisiting these assignments may be useful.

1. Create a new “Lab8” folder inside your CSE3 folder. With Notepad++, create a file named JusticeLeague.html
2. Set up the HTML skeleton as we have done in previous labs and homework assignments.
3. Make the title of the page “Justice League Battle”.
4. Center the code that will be in the body by putting in <center> tags.
5. Create a heading with the <h1> tag with the words “Justice League Vs. Joker”.

Up until this point, your HTML should look something like this:

```html
<html>
<head>
    <title>Justice League Battle</title>
</head>
<body>
    <center>
        <h1>Justice League Battle: Justice League vs. Joker</h1>
    </center>
</body>
</html>
```
Now we are going to create the game screen, starting with displaying the Justice League and the Joker and their stats:

7. Create `<table>` tags.
8. Create two rows with the `<tr>` tags. We will use the `<td>` tags nested in row tags to create two columns for each row.
   a. The first row is for Joker, who starts with 1000 health points. The second row will be used for Justice League, who start with 1700 combined health points.
   b. The first row, first column and second row, second column will contain the Hero’s and villain’s name and its current health points (HP). For the HP, use a set of `<p>` tags, each with `id` attributes. The first id should be `jokerHP` and the second id should be `justiceHP`. This will be explained later.
c. The remaining cells will be used to display the images. Download the two images linked under Lab 8 on the CSE3 website and save in your Lab8 folder them as “joker.jpg” and “league.jpg”. Use the <img> tag to display the images in the correct cells.

Your Table in HTML should look as follows:

```html
<table>
  <tr>
    <td>
      The Joker
      <p id="jokerHP">HP 1000</p>
    </td>
    <td>
      <img src="joker.jpg"/>
    </td>
  </tr>
  <tr>
    <td>
      <img src="league.jpg"/>
    </td>
    <td>
      Justice League
      <p id="leagueHP">HP 1700</p>
    </td>
  </tr>
</table>
```

Next we will create a dialogue box to display the events of our attacks in the battle.

a. Underneath your table, use a set of <div> tags and give it an id attribute of screen. You do not have to worry about what this tag is or what it does for this class. Just think of it as a container on a webpage.

b. Inside the <div> tags, use a set of <p> tags with the following text: Which League member should attack?

```html
<div id="screen">
  <p>Which League Member should attack?</p>
</div>
```
Finally, we will add the buttons to our game

1. Create another table with two rows and two columns.
2. To create the buttons, we will use the `<button>` tag to place a button in each of the table cells. The button tag has a special attribute called `onclick`. We will leave this blank for now, but we will use it to add functionality later.
3. In between each of the button tags, we will put the name one of League attacks.
   a. In the top row you should have **Batman** and **Superman**.
   b. The second row should have **Flash** and **Wonder Women**.

Your second table will now look like this:

```html
<table>
  <tr>
    <td>
      <button onclick=""">Batman</button>
    </td>
    <td>
      <button onclick=""">Superman</button>
    </td>
  </tr>
  <tr>
    <td>
      <button onclick=""">The Flash</button>
    </td>
    <td>
      <button onclick=""">Wonder Women</button>
    </td>
  </tr>
</table>
```
B. Formatting with CSS

As we did before, we will use CSS to efficiently format and style our web page.

1. In between the `<head>` tags, create a set of `<style>` tags as before:

   ```html
   <style type="text/css">
   </style>
   ```

2. We want the `body` background color to be Black and the font color to be white.
3. We want to use the CSS to set the height and width of our images. Set the `img height` and `width` to 200px.
4. We will also set the `height` of the `div` to 120px. It’s `width` to 400px and give it a `border` that is 2px solid white.
5. Lastly, we will make `button` have a `height` of 50px and `width` of 200px.

It should look like this:

```css
body{
    color: white;
    background-color: black;
}
img{
    height: 200px;
    width: 200px;
}
div{
    height: 120px;
    width: 400px;
    border: 2px solid white;
}
button{
    height: 50px;
    width: 200px;
}
```
C. Our JavaScript functions

Functions are used in many programming languages to encapsulate a set of instructions to be executed. Oftentimes this is code that is used multiple times with varying input. This saves us the trouble of the same set of instructions copied in different places in the program. Instead, we simply perform a function call.

In our Justice League battle game, we will implement 3 functions: (1) `attack()` to attack one round of the Justice League battle, (2) `jokerAttack()` to determine what move Joker will do, (3) `leagueAttack()` to calculate the damage that League’s attack generates

1. Put `<script>` tags in the `<head>` tags to denote JavaScript code. All our JavaScript code will reside here:

```html
<script type="text/javascript">

</script>
```

2. Next, we must declare our functions. The syntax looks similar to that of the loops; the code within the curly braces are executed each time the function is called. Functions also have a set of parenthesis. This is used to pass values to the function, but we won’t be using them in this case. Below is the `attack()` function. Write the two other functions yourself.

```javascript
function attack()
{
    // attack functions will appear here
}
```

3. For the function `attack()`, we will call our `jokerAttack()`, `leagueAttack()` and `result()` functions. Functions are called by their name, followed by a set of parentheses:

```javascript
function attack()
{
    jokerAttack();
    leagueAttack();
    result();
}
```
4. Next we will declare and initialize some variables at the top of our JavaScript code to use later in our functions. We will need:

1) `jokerHP` - initialize this to 1000 to keep track of Joker health points
2) `leagueHP` – initialized to 1700 to keep track of league’s health points
3) `leagueAttackName` – initialized to the empty string. This will be the name of league’s attack
4) `leagueAttackPower` – initialized to 0. This is the strength of the move league’s uses.
5) `leagueAttackRank` – initialized to 0. This is how attacks are ranked from strongest to weakest.
6) `jokerAttackRank` – the rank of Joker attack
7) `jokerAttackName` – initialized to the empty string. The name of Joker attack.
8) `jokerAttackPower` – initialized to 0. The strength of Joker attack.
9) `leagueName` - initialized to the empty string. This will be the name of league fighter

5. Now we will write our `JokerAttack()` function.

a) Use JavaScript’s `Math.random()` function to generate a random number between 0 and 1(exclusive). We will then multiply this number by 2 and add 1 to get a number between 1 and 2. Lastly, we use JavaScript’s `Math.floor()` function, passing in that number as a parameter to round down to the nearest whole number. This number is used to randomly choose Joker attack. Assign its value to the variable `JokerAttackRank`:

```
jokerAttackRank = Math.floor((Math.random() * 2) + 1);
```

b) We will have a set of `if` and `else` statements to check the value of `JokerAttackRank`. Depending on its value, we will assign the appropriate value of the attack’s strength to `JokerAttackPower` and the corresponding name to `JokerAttackName`.

c) Joker attacks are listed in the table below:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shock Attack</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>Gun Barrage</td>
<td>200</td>
</tr>
</tbody>
</table>

Your set of if/else statements should look somewhat like:

```
if(jokerAttackRank == 1){
    jokerAttackPower = 150;
    jokerAttackName = "Shock Attack";
}
else{
    jokerAttackPower = 200;
    jokerAttackName = "Gun Barrage";
}
```
d) Afterwards, we will want to decrement League health points by the strength of Joker attack.

\[
\text{leagueHP} -= \text{jokerAttackPower};
\]

6. After the opponent attacks, we will want to visually update League health points and print some dialog onto the screen. JavaScript has a useful command that allows us to modify existing HTML code dynamically:

```javascript
document.getElementById("leagueHP").innerHTML = "HP " + leagueHP;
```

In plain English, the command is saying:

a) Look through this document (HTML page).

b) Find the element (tag) in this document by its ID attribute (getElementById).

c) Get the HTML code within this element’s tags (innerHTML).

d) Replace the HTML code within the tag with the new value (the new value here is the string “HP” followed by the value stored in leagueHP).

7. Let’s take a look back to the table HTML and find the `<p>` tags that we defined earlier.

The last line of code we wrote will insert the value stored in `leagueHP` in between the `<p>` tags with the id attribute `leagueHP`. That line of code says to look through the document and get the tag with the id called `leagueHP`, which is the first cell in the second column of our first table. Now `innerHTML` will get the text in between the opening and closing `<p>` tag. We then set the value of “HP” + `leagueHP` as the text in between the opening and closing tag.

Similarly, we will want to use our `document.getElementById().innerHTML` call to look for the tag that has the id attribute of `screen`. We will replace the HTML in there with a message stating which attack Joker used and how much health league lost. Make sure to wrap this information in a set of `<p>` tags.

```javascript
document.getElementById("screen").innerHTML = "<p>Joker used " + jokerAttackName + "! League lost " + jokerAttackPower + "HP.</p>";
```

That’s it for the `jokerAttack()` function.

8. Now let’s go back to the HTML code for our League attack buttons. The value assigned to the `onclick` attribute is javascript code that will be executed. In between the quotations for the `onclick` attribute, put `attack()`. This is a function call, and it will call the `attack()` function to execute its code. To make any function call with no input parameters, you will simply need to type the function name followed by empty parentheses.

```
<button onclick="attack()">Batman</button>
```

Now if you run the code and click any of League member buttons, you should see the dialog box display one of Joker attacks. You should also notice that League HP has decreased.
9. Now let’s let the League retaliate in battle. In our `leagueAttack()` function, we will write code for League attack functionality.

a) We want to choose which league member will attack based on which button we press. When the user clicks on one of the buttons, we will assign the appropriate value to our `leagueAttackRank` variable. Inside or your attack buttons’ `onclick` attribute, write code for this variable assignment before your call to the `attack()` function.

Each League Members attack stats are as follows:

<table>
<thead>
<tr>
<th>Rank</th>
<th>League Member</th>
<th>Attack</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Batman</td>
<td>Batarang</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>Superman</td>
<td>Laser Beam</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>Flash</td>
<td>Speed Punch</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>Wonder Woman</td>
<td>Cuff Attack</td>
<td>120</td>
</tr>
</tbody>
</table>

Here is an example of how your code should look like:

```html
<button onclick="leagueAttackRank = 1;attack()">Batman</button>
```

b) Inside of the body of your `leagueAttack()` function, similarly to your `jokerAttack()` function, put in some conditional statements to check for the rank of League’s attack and assign the correct name and strength to the variables `leagueAttackName`, `leagueName` and `leagueAttackPower`.

```javascript
if(leagueAttackRank == 1){
    leagueAttackName = "Batarang";
    leagueName="Batman";
    leagueAttackPower = 70;
}
```

Continue to repeat the same format for all four league members.

c) Next, decrement `jokerHP` by the strength of League’s attack.

d) Update the display of information by using `document.getElementById().innerHTML` to change the HTML inside of the tag with the id `jokerHP` to display Joker’s new health points.

e) Lastly, we will want to add text to our dialog box to display which League Member attacked and what power it used and how much health Joker is lost. This time we will use the `+=` operator to append the information rather than replace it.

```javascript
document.getElementById("screen").innerHTML += "<p>"+leagueName+" used " + leagueAttackName + "! Joker lost " + leagueAttackPower + " HP.</p>";
```
Now when you run your webpage, both Justice League and Joker should be able to attack.

Now we going to have the `result()` function to let us know who won at the end of battle.

1. We will start by declaring the function `result()`.
2. We use `if` and `if-else` to check who won the war by checking if `jokerHP` is less than 0 and else if `leagueHP` is less than zero.
3. Update the display of information by using `document.getElementById().innerHTML` to change the HTML inside to display the result.

```javascript
if(jokerHP<=0)
{
    document.getElementById("screen").innerHTML = "<p>Justice League have saved the world</p>";
}
else if(leagueHP<=0)
{
    document.getElementById("screen").innerHTML = "<p>Joker has destroyed the world</p>";
}
```

D. Putting it all online

1. Modify your `CSE3Page.html` to include a link to your `JusticeLeague.html`.
2. Use Secure File Transfer Client to update your CSE3 folder online.

REMEMBER to drag ONLY your Lab8 folder into your CSE3 folder

Check-off:

Updated `CSE3Page.html` online with Lab8
Working Justice League game