COMP2132 PROJECT

OBJECTIVES

Implement a Javascript web application game for the web browser.

DUE DATE

Assigned during session10, due 11:59pm the night before session12.

REQUIREMENTS

You may team up with one other classmate for this project, or you may complete it yourself.

Include a prominent heading in the HTML indicating who the author(s) are.

Select **ONE** of the following two options to implement:

- Dice game (easier)
- Hangman game (harder)

The dice game will be easier to code, the hangman game will be more challenging. Both of them will be marked the same., so make your choice according to your strength with Javascript.

GENERAL REQUIREMENTS

Details for the rules and requirements of each game will be provided on the following pages, but no matter which game you implement:

- The game page should be carefully designed and styled with CSS to present a high-quality user experience. Submitting a project with poor or little styling will result in a greatly reduced final mark.
- There must be at least 6 images used.
- All paths used in HTML, CSS and Javascript files must be relative paths. Do NOT use server root paths that begin with / or client specific paths like **C:/**
- HTML, CSS and Javascript files must be free of serious errors (warnings are ok).
- Code must be well tabbed and use descriptive variable names.
- Javascript code must include the use of one or more **function**s authored by you, and one or more **Object**s authored by you.
- Must include at least one Javascript animation (for example, a fade in effect).
- jQuery may be used if desired.
- CSS should be compiled from SASS. SASS file(s) should demonstrate the use of SASS variables and at least one SASS mixin. Both .css and .scss files should be included with the project submission.
- Project must be published to a public repository on Github.com

DICE GAME REQUIREMENTS

Create a dice game where a user plays against the computer. The user and the computer each roll a pair of dice 3 times. After the third roll of the dice the player with highest score wins.

The scoring for the game works as follows:

- If any of the players two dice comes up as a 1 then the score for that round for the player is 0. eg: if the player rolls a 6 and 1, they get a score of 0
- If the player rolls a pair of the same numbers then the players score is the total of the two dice times 2. eg: if he player rolls 5 and 5, they get a score of (5+5)*2=20
- If the player rolls any other combination of dice other than the ones mentioned above then the players score is the total value of the two dice, eg: player rolls a 3 and 2, player gets a score of 3+2=5

The game should provide a text or graphical output showing the following:

- The current rolled dice values for the player and the computer.
- The score for this round for the player and the computer.
- The accumulated total score for the player and computer

The game should provide a button that will do the following: roll a pair dice for the player and another pair of dice for the computer, calculate the score for each of the player's then update the browser display to reflect the state of the game.

After three rolls of the dice the game should total up the scores and display a message displaying who the winner was.

The game should provide a button that will reset the game and start a new game

NOTE: A video demonstration of the basic required functionality for the Dice Game can be found from **learn.bcit.ca > Content > Session10 > Project > dice-game-demo.mp4**

HANGMAN GAME REQUIREMENTS

Create your own version of the standard hangman game.

- The hangman game should randomly select a word and hint from a collection of words and helpful hints.
- The user must guess the correct word by entering letters into an input box.
- If the user guesses a letter that is contained in the selected word than the game should display the correctly guessed letters in the position of the word where they are located.
- If the user makes an incorrect guess, then the program should display part of the hangman graphic.
- After the user guesses a letter, disable the option so they cannot choose the same letter more than once per game.
- If the user makes too many incorrect guesses and the entire hangman graphic is displayed, then the user loses the game. Most hangman games allow 6 incorrect guesses
- If the user correctly guesses all the letters in the selected word than the user wins the game.
- When the game is over either from making too many incorrect guesses or correctly guessing the word, then the game should display the results (Tell them if they Won or Lost the game) and give the user the option to **'Play Again'**.
- When a game is over, ensure the user cannot keep guessing letters, but must choose a **'Play Again'** option before they can play a new game.
- If the user chooses to **'Play Again',** reset everything (eg enable all buttons), and start a new game.

NOTE: A video demonstration of the basic required functionality for the Hangman Game can be found from **learn.bcit.ca > Content > Session10 > Project > hangman-game-demo.mp4**

SUBMISSION

Before the due date, publish all project resuorces to a public repository on **Github.com**. Email your instructor the Github.com URL before the due date:

> **To: jeffrey_parker@bcit.ca** or **jparker6@my.bcit.ca** (send to one, not both)
>
> **From:** Use your **@my.bcit.ca** email accessible from **learn.bcit.ca**
>
> **Subject:** COMP2132 Project
>
> **Body:** The URL to your project repository on Github.com

If completed with a partner, only one public repository is required with one email notification.

As a backup, also compress all project resources (HTML, CSS, JS, images, folders, etc) into a single **.ZIP**. Upload **.ZIP** to

> **learn.bcit.ca > Content > Session10 > Project Dropbox**