Project 2

Due Oct 17 by 11:59pm **Points** 100 **Available** after Oct 4 at 12:01am

Overview

In this project, you will get experience working with JavaScript and the DOM by creating a digital bingo card using JavaScript and the DOM. Players will be able to start a new game with either a random card or by entering a card as a string. The digital bingo card will also keep track of the players wins and losses in local storage.

About Bingo

- Every player gets at least one bingo card.
- The format of a bingo card is as follows:
  - The card is made up of 5 columns which each column representing the letters 'B', 'I', 'N', 'G', and 'O'.
  - Each column is labeled by the letter it represents.
  - Each column is defined as follows:
    - B contains 5 numbers between 1 and 15 with no duplicate numbers.
    - I contains 5 numbers between 16 and 30 with no duplicate numbers.
    - N contains 4 numbers between 31 and 45 with no duplicate numbers. In addition to the numbers, the third item in the list is a free space. The player can automatically mark off the free space. You can designate the free space however you would like on your board.
    - G contains 5 numbers between 46 and 60 with no duplicate numbers.
    - O contains 5 numbers between 61 and 75 with no duplicate numbers.
  - Example: **https://commons.wikimedia.org/wiki/File:Bingo_card_-_02.jpg (https://commons.wikimedia.org/wiki/File:Bingo_card_-_02.jpg)**
- A caller will then call out numbers like "B5".
- If a player has a square with the number 5 in the B column, they will mark it off.
- A player wins when they have 5 marked squares in a row horizontally, vertically or diagonally. When they win, they shout "BINGO!"

Required Packages

- Google Chrome (All grading will be done in Google Chrome, so make sure your digital bingo card works in Google Chrome.
- This project is to be done using vanilla JavaScript with the standard library. This means that you *cannot* use external libraries like jQuery.

GitHub Classroom Link

Please click on the following link to get your project repository.
https://classroom.github.com/a/dzsC8H3U (https://classroom.github.com/a/dzsC8H3U)

Specification

Below are the specifications for the application. I split the specification into UI elements and Events. Under UI Elements, I will list what needs to be part of the player interface. Under Events, I will list what should be done when the different UI elements are clicked on.

UI Elements

- A place to add the bingo card to the DOM when it has been generated to allow it to be displayed.
- The following buttons. I will use the names below when referring to these buttons in the rest of the specification. You can name these buttons whatever you want as long as its clear what they do.
  - new-game-random: start a new game with a randomly generated bingo card
  - new-game-specified: start a new game with a player specified bingo card
  - i-won: end the game because the player won
  - i-lost: end the game because the player lost
- A score board showing the players record of wins and losses.

Events

- When the page first loads, the player's total wins and losses are loaded from localStorage and displayed to the player in the scoreboard. The new-game-random and new-game-specified buttons are displayed. The i-won and i-lost buttons are hidden.
- When the player clicks on new-game-random, a random bingo card should be generated and displayed following the card format under 'About Bingo'.
- When the player clicks on new-game-specified, a prompt is displayed asking the player to enter a string representing a bingo card. If an invalid string is entered, alert the player that the string is invalid and display the prompt again. If a valid string is entered, a bingo card should be generated and displayed using the numbers in the player's string.
  - The string format will be B(15,9,8,7,14)I(25,21,20,22,29)N(38,41,f,34,31)G(60,57,48,56,49)O(69,70,72,64,71) where B(15,9,8,7,14) means that the B column on the board contains 15, 9, 8, 7, and 14. I(25,21,20,22,29) means the I column contains 25, 21, 20, 22 and 29. And so forth. 'f' is used in the string to represent the free space.
  - The string should be validated as follows:
    - The string format must be validated with a regular expression.
    - The string must also be validated to ensure that each of the columns only contain numbers that are allowed in that column based on the card format under 'About Bingo'.

  - Example strings that should be accepted are
    - B(15,9,8,7,14)I(25,21,20,22,29)N(38,41,f,34,31)G(60,57,48,56,49)O(69,70,72,64,71)
    - b(15,9,8,7,14)i(25,21,20,22,29)n(38,41,F,34,31)g(60,57,48,56,49)o(69,70,72,64,71)
  - Example strings that should fail validation are:
    - B(15,9,8,7,14)I(25,21,20,22,29)N(38,41,f,34,31)G(60,57,48,56,49)
    - B(15,9,8,7,1)B(25,21,20,22,29)N(38,41,f,34,31)G(60,57,48,56,49)O(69,70,72,64,71)
    - B(15,9,8,7,14)I(25,21,20,22,29)N(38,41,f,34,31)G(61,57,48,56,49)O(69,70,72,64,71)
    - B(9,9,8,7,14)I(25,21,20,22,29)N(38,41,f,34,31)G(60,57,48,56,49)O(69,70,72,64,71)
- The free space should be automatically marked when the card is generated.
- When the card has been generated and displayed to the player, the new-game-random and new-game-specified buttons should be hidden from the player and the i-win and i-lost buttons should be made visible to the player.
- The player will mark called numbers by clicking on the square containing the number. When the player clicks on a square, the following should happen:
  - The background color of the square should be changed so the player can track which squares they have marked.
  - The game should check to see if the player has bingo by testing if the player has marked 5 squares in row horizontally, vertically or diagonally. If they player has bingo, the game should alert "BINGO!"
- The player should also be able to unmark a square. When the player unmarks a square, the following should happen:
  - The background color of the square should change back to the default "unmarked" color.
- The player should not be able to unmark the free space.
- The player can end the game at any time by clicking on the i-win or i-lost buttons.
  - If the player clicks on the i-win button, test that they have bingo.
    - If they don't have bingo, alert the player and stop the game from ending.
    - If they have bingo, end the game and add one to their wins and store the result in localStorage and update the total displayed in the scoreboard.
  - If the player clicks on the i-lost button, end the game and add one to their losses total and store the result in localStorage and update the total displayed in the scoreboard.
  - When the player ends the game, the following should happen:
    - The bingo card should be removed from the DOM.
    - The i-win and i-lost buttons should be hidden from the player and the new-game-random and new-game-specified buttons should be made visible to allow the player to start a new game.

Notes & Hints

- No preference will be given when grading between using onclick or addEventListener for attaching event listeners to your UI elements.
- Use a tool like regex101.com to help you develop your regular expression.

- The bingo card does not have to survive a page refresh or closing the browser tab *BUT* the wins and losses totals must survive.
- You can keep your HTML, CSS and JavaScript in one file or split them among three separate files. It is up to you.

Submission Guidelines

- Place any comments you have for the grader to help them grade your assignment in the FOR_GRADER file
- Name the main HTML file bingo.html
- Do not submit any IDE files, feel free to add them to the .gitignore file under "# Editor files"
- Be sure to remember to push the latest copy of your code back to your GitHub repository before the the assignment is due. GitHub Classroom will not stop you from pushing additional commits after the deadline but it will make the latest commit before the deadline the submission that the grader will grade. It will also notify us if previous work is edited after the deadline.