

## Project 6: Game Show App

### Sections of this Guide:

- **How to approach this project** includes detailed guidance to help you think about how to organize your code, project and files.
- **How to succeed at this project** lists the grading requirements for the project, with hints, links to course videos to refresh your memory and helpful resources.

### How to Approach this Project

For the sixth project, you will use JavaScript to create the interactivity portion of a word guessing game. The HTML and CSS will already be written for you, and you will interact with elements that are created in the HTML file, as well as create new elements dynamically inside of your JavaScript. You will also write out the logic for displaying a phrase on screen, handling a user selecting a letter, and determining if the user won or lost the game.

- ❑ Download the project source files from the [Game Show App project instructions page](#) in your Techdegree curriculum.
- ❑ Set up a new GitHub repo and push the project files to it.
  - ❑ *Related video:* [Share Your Projects with GitHub](#)
- ❑ As you write your JavaScript code, be sure to save and test often. Code a small section and then test. This will help make troubleshooting much easier
- ❑ Double check everything, validate your files, request an informal review in Slack, and then submit.

### How to succeed at this project

Here is one approach to completing the things you need to pass this project. Make sure you complete them **before** you turn in your project.

- ❑ **Add needed variables**
  - ❑ Get the element with the **id** of `qwerty` and save it to a variable.
  - ❑ Get the element with the **id** of `phrase` and save it to a variable.

- ❑ Get the element with a class of `btn__reset` and save it to a variable
- ❑ Create a missed variable, initialized to 0, that you'll use later to keep track of the number of guesses the player has missed (remember, if the player guesses wrong 5 times, they lose the game)
- ❑ **Create an array named `phrases`.**
  - ❑ Declare and initialize the `phrases` array, storing at least five strings that contain only letters and spaces, no punctuation.
  - ❑ Related video: [Practice Basic Arrays in JavaScript](#)

```
1  // return a random phrase from an array
2  const getRandomPhraseFromArray = arr => {
3
4  }
5
6  // adds the letters of a string to the display
7  const addPhraseToDisplay = arr => {
8
9  }
10
11 // check if a letter is in the phrase
12 const checkLetter = button => {
13
14
15 }
16
17 // check if the game has been won or lost
18 const checkWin = () => {
19
20 }
21
22 // listen for the start game button to be pressed
23 startButton.addEventListener('click', () => {
24
25 });
26
27 // listen for the onscreen keyboard to be clicked
28 qwerty.addEventListener('click', e => {
29
30 });
```

*Example of all of the project's function headers, or "stubs"*

- ❑ **Attach an event listener to the "Start Game" button to hide the start screen overlay.**
  - ❑ Add the event listener to the variable you created for the `btn_reset` element
  - ❑ Hide the overlay by changing its `display` property.

## ❑ Create a `getRandomPhraseAsArray` function.

- ❑ Start by creating a function “stub”, where you declare the function and its parameters, but leave the function body blank. Add a code comment that describes the purpose of the function.
- ❑ Create a variable to store a random number based on the length of the array
- ❑ Use the variable to select an index inside of the array.
- ❑ After you create the `getRandomPhraseAsArray`, you will need to “call” it, and pass the `phrases` array to it.
- ❑ Return the array element at that index
- ❑ *Related video:* [Practice Basic JavaScript Functions](#)
- ❑ *Related video:* [JavaScript Numbers: Create a Random Number](#)

## ❑ Create an `addPhraseToDisplay` function.

- ❑ Create an `addPhraseToDisplay` function that loops through an array of characters. You will need to write it so that it can take *any* array of letters and add it to the display.
  - ❑ Inside the loop, for each character in the array,
    - ❑ Create a list `li` item
    - ❑ Put the character inside of the list item
    - ❑ Append that list item to the `#phrase` `ul` in your HTML
    - ❑ If the character in the array is a letter and not a space, the function should add the class `“letter”` to the list item. If not, add the `“space”` class.
- ❑ To use the function, you’ll get the value returned by the `getRandomPhraseAsArray`, save it to a variable, and pass it to `addPhraseToDisplay` as an argument.
- ❑ *Related video:* [Practice Basic JavaScript Functions](#)
- ❑ *Related video:* [Creating New Dom Elements](#)
- ❑ *Related video:* [Appending Nodes](#)

## ❑ Create a checkLetter function

- ❑ Create a function “stub” for the `checkLetter` function
  - ❑ Include a parameter in the function head for the button that gets clicked
- ❑ Store all of the `li` elements in a variable inside `checkLetter`
- ❑ Create a variable to store if a match is found and give it an initial value of null
- ❑ Loop through all of the `li` elements. *Remember: arrays start with index 0!*
  - ❑ Create a conditional that compares the text of the button parameter to the text of the `li` at the current index of the loop
    - ❑ If they match, add the “show” class to the `li`
    - ❑ If they match, store the button text in the `match` variable
- ❑ Once the loop completes, `return` the `match` variable
- ❑ *Related video:* [Practice Basic JavaScript Functions](#)

## ❑ Add an event listener to the keyboard

**Note:** the event listener should be listening for a user to press a button on the on screen keyboard, not the physical keyboard of the computer.

- ❑ Start by creating an event listener for the `qwerty` element that listens for the “click” event.
- ❑ Use a conditional to filter out clicks that don’t happen on the buttons or if the button already has the “chosen” class
  - ❑ Add the “chosen” class to the button that was pressed.
  - ❑ Call the `checkLetter` function and store the results in a variable.
  - ❑ If the `checkLetter` function does not find a letter, remove one of the heart images and increment the `missed` counter
- ❑ *Related Video:* [JavaScript and the Dom: What is an Event?](#)
- ❑ *Related Video:* [JavaScript and the Dom: Listening for Events with `addEventListener`](#)

## ❑ Create a checkWin function

- ❑ Create a variable to store the `li` elements that have the class name “letter”
- ❑ Create a variable to store the `li` elements that have the class name “show”

- ❑ Check if the length of the 2 variables are the same. If they are, display the win overlay
  - ❑ Create the win overlay by adding the “win” class to the start overlay.
  - ❑ Change the headline text of the start overlay to show a person won.
  - ❑ Change the `display` property of the overlay to “flex”
- ❑ Check if the missed counter is greater than 4. If they are, display the lose overlay
  - ❑ Create the lose overlay by adding the “lose” class to the start overlay.
  - ❑ Change the headline text of the start overlay to show a person lost.
  - ❑ Change the `display` property of the overlay to “flex”
- ❑ *Related video:* [Practice Basic JavaScript Functions](#)

**Good luck! And remember if you need any help or have any questions, the Slack community is here for you!**