JavaScript Grammar Checker Instructions

In this activity, you will use what you know about iterating over arrays to gather information and improve the quality of a paragraph.

1	**Getting Started** In **grammarChecker.js**, the `story` variable holds the paragraph you will be editing. In order to edit the story, turn it into an array on line 3. The `.split()` method separates the `story` string by the space character (`' '`) and stores each word as an element of the array. To see the array you will be working with throughout the lesson, log `storyWords` to the console. After you've viewed the `storyWords` array, comment out the `console.log()` statement before moving to the next task.
Hint	The `' '` argument of the `.split()` method indicates that the split should occur any time a space occurs in the string. This splits the string into a new array with each word being an element of the new array. Remember that you can log `storyWords` to the console with `console.log()`, and comment it out using a double forward slash (`//`).
2	For a better visual comparison of the original and edited stories, you want to view the edited `storyWords` array as a string. To change the `storyWords` array back into a readable string, you can invoke the `.join()` method on `storyWords`. Give the `.join()` method an argument of an empty space character (`' '`) to separate each array element with a space in the string. Place the `.join()` method invocation as an argument of a `console.log()` statement to log the final story to the console.
Hint	The final syntax should look like: `console.log(array.join(separator));`
3	**Counting Words** Now it's time to start editing the story by manipulating the `storyWords` array. You want to be able to see the changes, so be sure your `console.log()` of the joined story is the last line of code in your editor. First, above the `console.log()` statement that uses the `.join()` method, create a variable named `count` that stores the number `0`. Directly below `count`, use a `.forEach()` method to iterate over the `storyWords` array. As an argument of the `forEach()` method, create an empty function to be used as the callback function. While ES6 arrow syntax is recommended for the callback function, feel free to use any syntax you're comfortable with.
Hint	Prefix `.forEach()` with the name of the array you wish to iterate over. Inside the parentheses of the `.forEach()` method, create an empty function to be used as the callback function.
4	For each word in the `storyWords` array, you want the `count` variable to increment by one.

	Add a parameter named word to the callback function of the .forEach() method to be used to store the current element when iterating over the storyWords array. Each time storyWord iterates, increment count by one. Below the .forEach() method, log count to see how many words are in the story.

Hint	The increment operator (++) is a great way to increment a value by one.
	
	Your .forEach() method should resemble the following if using the arrow function syntax.
	
	`array.forEach((element) => {`
	`  // do something with each element of the array`
	`});`
	Remember to keep the console.log() statement of the joined storyWords array as the last line of your code.

5	**Filtering Words** A word count of 181 is a bit long for this story. Let's filter out all instances of the word "literally" to shorten the story and remove the unnecessary word. You will reassign the filtered story to the same storyWords variable by applying the .filter() method! Throughout the project, you will use this approach of reassigning the storyWords variable for each revision of the story. Below where you logged the count variable, reassign the storyWords variable to equal the invocation of the .filter() method on the storyWords array. Give the .filter() method a callback function with a parameter of word.
Hint	Since storyWords was declared using the let keyword, we can reassign the value. You can provide the callback function for the .filter() method as its argument like the following: `array = array.filter(callback);`
6	Below the storyWords variable declaration, notice the variable unnecessaryWord on line 4. You want to filter out the value of unnecessaryWord from the story. Within the filter() method's callback function body, return word only if it is NOT equal to unnecessaryWord. Check the story string in the console to make sure it doesn't include the word "literally". The first instance of "literally" was previously in the first sentence.
Hint	The strict inequality operator (!==) can be used to check if two values are not equal. Be sure to use the return keyword in the body of the callback function to return the necessary words!
7	**Replacing Words** Now that you've removed the unnecessary words, let's take care of any misspelled words in the story. You can reassign storyWords to a new array of spell-checked words using the .map() method! Reassign storyWords to equal the invocation of the .map() method on the storyWords array. Setword as a parameter of .map()'s callback function.

	In the callback's body, define a conditional statement to check if the word argument is equal to the misspelledWord variable. If it is, return the correct spelling of the string, "beautiful". If not, return word. Take a look at the story string in the console to see the correct spellings! You can see one instance of the spellchecked word in the first sentence of the story.
Hint	A quick way to return the result of an if...else conditional is by using the *ternary operator*. The ternary operator is just a shortcut for an if...else statement. The syntax is as follows: `conditionToCheck ? expressionIfTrue : expressionIfFalse`
8	Uh oh, your great grandmother is going to read the story and there's a "bad" word in it! Let's apply the .findIndex() method to storyWords to find the index of the bad word. Start by declaring a variable called badWordIndex and setting it to the invocation of .findIndex() on the storyWords array. The .findIndex() callback function should check each word to see if it equals the badWord variable declared on line 6, and return the index of the found word. Then, log badWordIndex to the console.
Hint	Be sure to use the return keyword in the callback function's body to return the index.
9	Now that we have the index of the bad word, we can easily replace it with something more appropriate. Access the element inside the storyWords array that has the index of badWordIndex using bracket notation. Set the accessed element equal to the more appropriate string, 'really'. Save the code and check that the bad word has been replaced.
Hint	Accessing an element using bracket notation is done by placing brackets [ ] after the array's name, with a the number representing the index of the element inside the brackets. In our case, this number is stored as badWordIndex.
10	Finally, let's simplify the words in the story to appeal to a broader audience. We can make sure every word in the story is less than 10 characters using the .every() method. The .every() method uses a callback function to test if every element in an array passes a specified condition. It returns true if all elements pass, and false if there is an element that does not pass. To start, define a variable called lengthCheck and set it to the invocation of the .every() method on storyWords. In the callback function, test whether every word is less than 10 characters. Log lengthCheck to the console to see the result. If true is logged, every word in the story is less than 10 characters. Otherwise, one or more words are longer than 10 characters.

Hint	The syntax for the .every() method is similar to other iterator methods we've been working with:
	
	`array.every((word) => {`
	`  return // condition`
	`}`
	You can check the length of a word by using the .length property, and compare the numbers using the less than (<) operator.

11	Hmm, it looks like there's at least one word longer than 10 characters.

	Use an iterator method of your choice to access the word (there is only one) in the `storyWords` array that is greater than 10 characters. Then, manually replace the word in the original `story` string with a shorter word. Some ideas for replacement words that can be used are: stunning, dazzling, or glorious.
Hint	A couple options that can be used to access the element are `.filter()` and `.forEach()`. Feel free to use any method you like. As long as you're able to find the word, you've accomplished the task!
12	Stretch & Challenge Great work! You've vastly improved the story using some of the most important iterator methods available to us to use as JavaScript developers. Feel free to continue using iterator methods to make further edits, or click *Next* to continue your learning journey.
Hint	Some additional improvements could be: • Removing the word "very". • Replacing "GW" with "George Washington". • Changing the imperial units of measurement (feet and miles) to metric units (meters and kilometres)