Lesson Tuesday

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Separation of Logic: Adding a UI to Text Analyzer

Text

Over the last two lessons, we used test-driven development to build some basic functionality for a word counter program called "Text Analyzer". This program can count the number of words in a passage of text and also count the number of occurrences of that word.

In this lesson, we'll add a user interface for our application, paying close attention to separation of logic as we go.

Spoiler alert: because of the way we're designing our application, it's going to be much easier to keep our logic separate. That's because we've written and tested our business logic completely separately from our user interface.

Make sure to code along with this lesson and the next 3 that follow. All of these lessons focus on creating Text Analyzer's UI logic. In the upcoming practice lesson, you'll be tasked with adding more functionality to this project.

Adding a User Interface

Let's start by adding our UI files. Here are all the files that `text-analyzer` should include:

```
index.html
js/scripts.js
css/styles.css
README.md
```

Note the file paths above: the `js` directory should contain `scripts.js` and the `css` directory should contain `styles.css`. We'll incorporate Bootstrap into our UI, and we'll do so via a CDN, so we don't need to include local files for that library. You are always welcome to include Bootstrap directly in the project if you prefer.

Here's our `scripts.js` file so far:

js/scripts.js

```
// Business Logic

function wordCounter(text) {
  if (text.trim().length === 0) {
    return 0;
  }
  let wordCount = 0;
  const textArray = text.split(" ");
  textArray.forEach(function(element) {
    if (!Number(element)) {
      wordCount++;
    }
  });
  return wordCount;
}

function numberOfOccurrencesInText(word, text) {
  const textArray = text.split(" ");
  let wordCount = 0;
  textArray.forEach(function(element) {
    if (element.toLowerCase().includes(word.toLowerCase()))
{
      wordCount++;
    }
  });
  return wordCount;
}

// UI Logic
```

We have two functions that we can plug into our user interface where needed.

Note that we've also added a comment at the end of the file for UI Logic. We will put all of our UI logic below this line. Again, you won't see this comment in real-world code bases but it will help us keep things separate for now. As we've mentioned previously, we'll learn how to separate our business and user interface logic into different files in Intermediate JavaScript.

Next, let's add a basic HTML file called `index.html`.

index.html

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <title>Text Analyzer</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/
dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha
384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kad
q2F9CUG65" crossorigin="anonymous">
  <link href="css/styles.css" rel="stylesheet" type="text/c
ss">
  <script src="js/scripts.js"></script>
</head>
<body>
  <div class="container">
    <h2>Text Analyzer</h2>
    <form id="word-counter">
      <div class="form-group">
        <p>Input a text passage to get a total word count:
</p>
        <textarea id="text-passage" name="text-passage" cla
ss="form-control"></textarea>
        <p>Optionally enter a word to count the number of t
imes it occurs in the passage:</p>
        <input type="text" id="word" name="word" class="for
m-control">
        <br />
        <button type="submit" class="btn btn-success">Submi
t Survey</button>
      </div>
    </form>
    <p>Total Word Count: <span id="total-count"></span></p>
    <br />
    <p>Selected Word Count: <span id="selected-count"></spa
n></p>
    <div id="bolded-passage">

    </div>
  </div>
</body>
</html>
```

This includes a form with a `textarea` (a larger `text` field) for the text passage as well as a simple text field for inputting a word.

Note also that we've included a `div` with `id="bolded-passage"`. This is where the results will go for some functionality that we will be adding later to further demonstrate how to separate logic.

Now let's get to work on our UI logic. Add this code below the UI Logic comment in `scripts.js`:

js/scripts.js

```
// UI Logic

function handleFormSubmission() {
  event.preventDefault();
  const passage = document.getElementById("text-passage").v
alue;
  const word = document.getElementById("word").value;
  const wordCount = wordCounter(passage);
  const occurrencesOfWord = numberOfOccurrencesInText(word,
passage);
  document.getElementById("total-count").innerText = wordCo
unt;
  document.getElementById("selected-count").innerText = occ
urrencesOfWord;
}

window.addEventListener("load", function() {
  document.querySelector("form#word-counter").addEventListe
ner("submit", handleFormSubmission);
});
```

First notice that we've separated all of the code that responds to the form submission into its own UI function called `handleFormSubmission()`. We then pass that function as the callback when we create the event listener for the form's submit event.

When a form is submitted, the `handleFormSubmission()` function will be called. First, we'll grab the form input values for `passage` (the entire text) and `word`. Then we'll call our `wordCounter()` and `numberOfOccurrencesInText()` functions and update the `HTMLElement.innerText` property of two spans to update the webpage with the results.

Our code is very nicely separated. Let's try out our Text Analyzer webpage in the browser. Be forewarned, though. There is a bug! See if you can figure out what it is before moving on.

In the next lesson, we'll discuss this bug, walk through the debugging process, and then fix it. But we recommend seeing if you can find the bug, figure out what the problem is, and fix it on your own before moving to the next lesson. Don't be hard on yourself if you can't find it. Just look at it as an opportunity to practice debugging.

Previous (/introduction-to-programming/arrays-and-looping/practice-extending-text-analyzer-business-logic-with-tdd)
Next (/introduction-to-programming/arrays-and-looping/separation-of-logic-fixing-a-bug-in-text-analyzer)

disable dark mode

(http://www.epicodus.com)