GYMNASIUM

JAVASCRIPT FOUNDATIONS

Lesson 3 Assignment

The DOM (Document Object Model)

CORE CONCEPTS

1. HTML elements on a page are represented by JavaScript element objects.
2. Selecting an element doesn't do anything to that element. It merely finds a matching element and gives you a JavaScript element object that represents that element.
3. Element objects have various properties that allow you to inspect the attributes of the represented HTML element.
4. Changing the values of those properties instantly updates that HTML element and re-renders the page.
5. The textContent property allows you to set the text that appears inside of an element. However, this will replace any previous text or child elements, so be careful.
6. The innerHTML property is similar to textContent but allows you to pass a string of HTML. This will create new child element for the selected element.
7. The style property is an object. It has many properties of its own, which represent the various CSS styles that can be applied to that element.
8. An application generally gets input, often by reading the values of existing HTML elements, validates that input, processes it somehow and outputs the result, often by setting the values of other HTML elements.

ASSIGNMENT

1. Quiz
2. Go to two or three pages of your choice, preferably ones with a good amount of complex text. Open the console and do several queries for common elements, such as h1, p, div, etc. Try reading and setting various properties of the elements you get, such as text content, innerHTML, and any styles you want. Remember that some selector functions will give you back an array of items, so you'll have to choose which element to work with by specifying an index. Repeat this until you are comfortable selecting different types of elements, and modifying them.
3. In the sample application we created, we are validating each numeric property and setting the border style of the input related with it like so:

   ```
   if(isNaN(miles)) {
       distanceInput.style.borderColor = "red";
       return;
   }
   else {
       distanceInput.style.borderColor = "initial";
   }
   ```

   Create a new function in that project called validateInput. It should take a number value and a text input element as parameters, like so:

```
function validateInput(value, input) {
}
```

Fill in the body of that function so that it performs the same action as the original code. Replace each original if/else statement with a call to that new function, and see how this makes the overall code smaller and easier to read. Like so:

```
validateInput(miles, distanceInput);
```

4. For an extra challenge, have the validateInput function return true or false based on whether or not the input is valid. Use that in the original code to jump out of the click event handler function if the validation fails. This might look something like this:

```
if(validatedInput(miles, distanceInput) == false) {
    return;
}
```

   But there is an even more concise way to do it using the "not" operator, which is the exclamation point "!".

RESOURCES

- Again, more documentation. The objects that represent HTML elements on a page have a type of "HTMLELement". But they are also a type of more general object just called "Element". So, they have properties and methods from both of these, and you might need to look in two places to see all of properties. This is the page that lists all the properties and methods of element objects:

  https://developer.mozilla.org/en-US/docs/Web/API/element

  And this lists the properties of HTMLElement objects:

  https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement

- Read through these and find some other interesting things you can do with elements. As you do this, think of some other ways you could use these properties or methods in an application.

  And here is the reference for the style property:

  https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement.style