**Score: \_\_\_\_\_**

**WA3 – Front End: JavaScript**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

**Introduction:**

In today’s class we saw a number of ways that JavaScript can interact with content in a web page. We can run a JavaScript function when an event, such as a button click, occurs. That JavaScript code can then interact with elements in the page through the Document Object Model (DOM). The document.getElementById(…) method allows the JavaScript code to reference HTML elements by using their id attributes, in a way very similar to how CSS id selectors work. JavaScript can then access an element’s attributes (e.g. .value), change an element’s attributes (e.g. .src) and even modify the CSS styles that are applied to the element (e.g. .classList.add(…) or .classList.remove(…) ). Finally, we saw that it is also possible for JavaScript to actually create new HTML elements and add them to the DOM, causing them to appear in the rendered page. The activities below will build directly on these examples from class. Thus, you may find it helpful to review each of those examples in turn as you move through these activities.

**JavaScript Metaphor:**

🔑 1. In class we have used the metaphor of a house to explain the roles of HTML, CSS and JavaScript in web applications. In the last set of activities, you invented your own metaphor and used it to explain the roles played by HTML and CSS. Briefly reintroduce your metaphor and use it to explain the role of JavaScript.

**Events and JavaScript Functions:**

The js1.html file in the www folder of the Comp256Web container contains the source code for the first example from the class slides. The questions in this section build on that example. You will want to open the js1.html file in a text editor, and you can access the rendered version by clicking on its link in the page at http://localhost:8080/www.

🔑 2. Add to the HTML source to create another button that will call a function named goodbye. The text on the button should reflect its purpose. Give the line of HTML source that you added.

🔑 3. Add JavaScript code that defines the function goodbye and implements it so that it displays a goodbye message. Give the JavaScript for your goodbye function.

🏆 4. Like functions or methods in other languages, JavaScript functions can have parameters and the calls to them can pass parameters.

a. Add a parameter to your goodbye function so that it accepts a name as an argument. Then use that name in the message that is displayed (e.g. "Goodbye Dorothy"). Give the updated code for your goodbye function.

b. Update the call to your goodbye function in your HTML so that it passes a name as an argument to the function. Note: Because the call to the function already appears in double quotes (e.g. onclick="goodbye()") you’ll need to be careful about how you pass a string as the argument. See: the *Single quotes vs. double quotes* section on the MDN page *Handling text — strings in JavaScript* for information about how to have one string inside of another:

* <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Strings>

**Text Input Fields:**

The questions in this section build on the js2.html file in the www folder in the Comp256Web container. You will want to open the js2.html file in a text editor and add code to it to complete the questions in this section and to test your answers by using the rendered version. You can see the rendered version by clicking on the link for js2.html in the page displayed at: <http://localhost:8080/www>.

🔑 5. Add an HTML element to the body of the document to allow the user to input their last name as well as their first name. Give the HTML for the new element that you added.

🔑 6. Revise the hello function so that it uses both the first and last name entered. For example, it might now say “Hello Anita Borg, I’m JavaScript.”. Give the JavaScript for your revised hello function here.

🏆 7. Optional: Make it possible for the user to also select a prefix for their name.

a. Add an HTML element for dropdown list that allows the user to pick a prefix (e.g. Mr., Mrs., Miss., Ms., Dr., etc…). The w3Schools page on the *HTML <select> Tag* will provide some helpful information:

* <https://www.w3schools.com/tags/tag_select.asp>

Or if you are feeling adventurous, you could use a styled dropdown provided by bootstrap:

* <https://www.w3schools.com/bootstrap/bootstrap_dropdowns.asp>

Give the HTML source that you added to create the dropdown list.

b. Modify the hello function to include the prefix in the message (e.g. “Hello Dr. Anita Borg, I’m JavaScript.”). You’ll need to figure out how to get the selected item from the dropdown list. A search for something like “Get selected value in dropdown list using JavaScript” should produce some useful results. You may have to adapt a little if you’ve used the bootstrap dropdown.

**Changing Attributes with JavaScript:**

The questions in this section build on the js3.html file in the www folder of the Comp256Web container. You will want to open on the js3.html file in a text editor and add code to it to complete the questions in this section and to test your answers by using the rendered version. You can see the rendered version by clicking on the link for js3.html in the page displayed at: <http://localhost:8080/www>.

🔑 8. The button in the js3.html page is able to make the weather go from rainy to sunny, but we can’t go back. Let’s fix that.

a. Add an HTML element to the js3.html page that creates another button. This button should call a function that will change the image back to rainy. You’ll write the function in part b, but you should put the call in the element now. Be sure that your button displays appropriate text. Give the HTML source for your new element.

b. Write the function that is called by your button from part a and have it change the image back to the rainy image. Give the code for the JavaScript function that you wrote.

**Changing Styles with JavaScript:**

9. In your modified js3.html page from question #8 there are always two buttons, one to make it sunny and one to make it rainy. That seems a little silly, why have a button to make it rainy if it is already rainy? And vice versa for when it is sunny? Let’s fix that.

a. One way to fix this will be to hide the HTML element for the button that we do not want to have displayed. The examples from class used a CSS rule that was able to make things invisible. Add that style element to your js3.html page. Give the HTML source for the style element that you added including the CSS rule here.

b. The page initially displays the rainy image. Thus, the button to make it rainy should not be displayed when the page is loaded. Modify the HTML for that element so that it is not visible. Hint: Add a class attribute and use the CSS rule you added in part a. Give the HTML for the modified element.

c. Eventually, you’ll be using JavaScript to make the buttons visible and invisible as appropriate. But, before you can do that, your JavaScript will need to be able to get the button elements. To make that possible each button will need to have an id attribute. Add an id attribute to each of your button elements. Give the code for the elements that you updated here:

d. Now modify the sunny() JavaScript function so that it also does the following:

* Shows the button to make it rainy. Hint: Use the classList.remove function to remove the class that makes the button invisible.
* Hides the button to make it sunny. Hint: Use the classList.add function to add the class that makes the button invisible.

Give your updated JavaScript function here:

🏆 e. Now modify the JavaScript function that you added to make it rainy so that it also changes the visibility of the buttons so that only the necessary one is displayed. Give your updated JavaScript function. When this question is complete the page should function such that only one of the two buttons is visible. If it is rainy the button to make it sunny should be visible. If it is sunny, the button to make it rainy should be visibible.

**Modifying the DOM:**

The questions in this section build on the js5.html file in the www folder of the Comp266Web container. You will want to open the js5.html file in a text editor and add code to it to complete the questions in this section and to test your answers by using the rendered version. You can see the rendered version by clicking on the link for js5.html in the page displayed at: <http://localhost:8080/www>.

10. Recall that the DOM is a tree data structure that represents the content and structure of the HTML page. Imagine the user adds “Tacos”, “Pizza” and “Wings” to the bulleted list. Draw the DOM for this page after these additions have occurred.

🏆 11. Optional: Currently it is possible to add new foods to the list, but it is not possible to remove them. Let’s fix that.

a. Add HTML code for a new button element that will remove the last food in the list. The button doesn’t have to work yet, it should just have appropriate text and should call a JavaScript function that you will write next. Give the HTML for your new button element.

b. Write the JavaScript function that removes the last item in the list. Hint: Each li element in the list is called a *child* of the ul element. Thus, you want your code to find and then remove the last child of the ul element. Use your favorite search engine to learn more about the lastChild property and the remove function. Give your JavaScript function for removing the last child here.

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.