HOUSECS 59.03: WEB DEVELOPMENT AND SOCIETY
Programming Assignment #3: Introduction to JavaScript

GUIDELINES:

Students are expected to adhere to the Duke Community Standard. If a student is responsible for academic dishonesty on a graded item in this course, then the student will have an opportunity to admit the infraction and, if approved by the Office of Student Conduct, resolve it directly through a faculty-student resolution agreement; the terms of that agreement would then dictate the consequences. If the student is found responsible through the Office of Student Conduct and the infraction is not resolved by a faculty-student resolution agreement, then **the student will receive a failing (unsatisfactory) grade for the final grade in the course**.

- Students may work on programming assignments with a maximum of one (1) other individual in the class. However, both individuals should contribute *equally* to the assignment and understand *all* parts of the code written.
- Students are expected to write their adherence to the Duke Community Standard in a README for every assignment. Students are allowed to consult others outside of their group—limited to Duke students and faculty—about the assignment only in a general way, but not actually provide/receive code to/from other students. If assistance is received from other individuals (excluding the instructors), it should be cited in the README. **Students should be prepared to explain any program code they submit.**
- It is acceptable to use *small* pieces of outside code (found on the Internet or otherwise) due to the nature of this course—but not entire methods or programs. Using open source libraries and packages is allowed. If you are concerned whether using a piece of code is within the Duke Community Standard, please ask. *All code used should be properly cited.*
- **All submissions are subject to automated plagiarism detection.** Assignments will be randomly checked using the MOSS Plagiarism Detector.

This assignment will be due on **Wednesday, November 1** and should be completed before the start of class. The policy for turning in late assignments is detailed in the syllabus. In order to receive a passing (satisfactory) grade, in addition to satisfying the attendance requirement, students must complete **all** assignments of this course with a total average of 70% or greater.

INSTRUCTIONS:

This assignment will assess your understanding of JavaScript structure, functions, and decisions as well as the HTML DOM. As part of this assignment, you will create one or more webpages on GitHub Pages that can be accessed at `http://[username].github.io/project3/`. You should use Git to track your changes, and your commit messages should be thoughtful and meaningful. The last commit timestamp will be used to determine the submission time of the assignment.

If you are part of a team, the webpages only need to be accessible via one team member's username, but both members should make an approximately equal number of commits to the project. Both members will earn the same score on the assignment unless the distribution of work is not equal. The assignment will be graded according to the rubric (100 points) on the following page.

RUBRIC:

Proper usage of version control systems (15 points)

- [5 points] README with adherence to Duke Community Standard
- [5 points] webpages accessible at `http://[username].github.io/project3/`
- [5 points] thoughtful and meaningful commit messages

Strong understanding of basic JavaScript structure, functions, and decisions (60 points)

- [10 points each] completion of tasks 1-3
- [15 points each] completion of tasks 4-5

Strong understanding of HTML document object model (25 points)

- [25 points] completion of task 6

TASKS:

For each of the following tasks, write a separate function, or series of functions, that accomplishes each. Partial credit *will* be awarded for incomplete or incorrect functions.

1. Take in a number input, say x. Modify a paragraph element below the input to read $f(x) = 5x - 3x^2 + 13$. Assume the input will always be a number. For example, if the input is `3`, the output is `f(x) = 1`.

2. Take in three (3) string inputs. Modify a header element below the inputs to join the inputs with a space, in order, and greet the individual. For example, if the inputs are `John`, `Jacob`, and `Smith`, respectively, the output is `Hello, John Jacob Smith!`.

3. Take in two (2) number inputs, say x and y, respectively. Modify a paragraph element below the inputs to display the output. If x is greater than y, output `bananas` in yellow. If y is greater than x, output `porcupine` in brown. If x and y are equal and negative, output `red apples` in red. Otherwise, output `green apples` in green. Assume the input will always be a number.

4. Take in a string input. Modify a header element below the inputs to display the output. If the input is a valid United States or Canadian phone number, output `valid` in green. Otherwise, output `invalid` in red. A valid phone number is defined as, after all non-numbers are removed from the input, the remaining number has ten (10) digits or eleven (11) digits with a 1 as the first digit. For example, valid phone numbers include:

`(123)-456-7890`, `123 456 7890`, `+1 123-456 7890`, `phone number: 11234567890`

5. Take in a string input, which will be a comma separated list of items. Modify a paragraph element below the input to read the third item in the list. For example, if the input is:

```
apples, bananas, oranges, pears, peaches, pineapple
```

The output is `oranges`.

6. Create a list of five (5) buttons. If the user presses any of the first four (4) buttons, the next button will turn blue. If he or she presses the fifth button, all of the buttons will turn red. The button colors reset before each press.