Guided Inquiry: JavaScript Week 3

Overview

Welcome to week three of JavaScript. At this point you should begin feeling more comfortable with many of the foundational concepts previously introduced. You should be combining concepts to solve more challenging problems. These notes cover the foundational JavaScript skills necessary to start utilizing the language and to become familiar with key terms. We are expounding and adding to our previous list of objectives in order to help you reinforce and grow your skill set. Repetition is key.

Learning Objectives

Basic objectives: Each student is responsible for gaining proficiency with each of these tasks prior to engaging in class discussions, through the use of the learning resources below, additional research and through the working through exercises/ challenge portion in this document:

- Continue building proficiency with key concepts
    - Variables, Data Types, Type Coercion, Basic Operators and Operator Precedence
- Understand the use of Logical Operators
    - Boolean, If/Else, Ternary and Switch Statements
- Become comfortable creating and working with Functions, Arrays and Objects
- Differentiate between the terms: *Objects, Properties* and *Methods*
- Demonstrate an understanding
    - Logical Operators, Functions and Arrays
- Understand the importance and use of the following in JavaScript
    - Loops and Iteration Events, Execution Context, Hoisting, Scope and 'this'
- Practice advanced JavaScript concepts including
    - Inheritance, Prototype Chain, First Class Functions, Closures, Immediately Invoked Function Expressions and Closures
- Become comfortable working with ES6 additions including
    - Const, Let, Template Literals, Arrow functions, Destructuring, Spread Operators, Rest Parameters, Default Parameters, Maps, and Classes
- Understand Asynchronous JavaScript
- Make AJAX calls to 3rd party APIs using Fetch and Async/ Await
- Practice by coding alongside of the assigned videos

Learning Resources

To gain proficiency in the learning objectives, use the following resources:

- Udemy Video: CSS The Complete JavaScript Course (Build Real Projects!) Sections 1- 11
  *Note: Section 6 (HTML & CSS Crash Course) is optional
  https://redventures.udemy.com/course/the-complete-javascript-course

See next page...

Exercises/ Challenge

The exercises below are to be done during and following your viewing of the resources. Please push your completed project to a repository on Github. Provide the link to your repo. Each activity should be appropriately labeled in its own folder.

EASY: Write a function that would allow you to do this:

```
var run = exercise('running');

console.log(run()); // prints "Today's exercise: running"

var swim = exercise('swimming');

console.log(swim()); // prints "Today's exercise: swimming"
```

MEDIUM: Write a function that would allow you to do this:

```
var sharePizza = cutPizzaSlices(8);

console.log(sharePizza(2));

  // prints "Each person gets 4.00 slices of pizza"

console.log(sharePizza(3));

  // prints "Each person gets 2.67 slices of pizza"
```

HARD: Data security exercise. Inside of a closure, create an object called pii (Personally Identifiable Information)that cannot be accessed directly. The object should have at least two properties: name and ssn. Only the name property should be accessible, and it should be called through a public function. The ssn property should not be accessible at all.

Creating private objects and private properties helps you control who has access to what data and helps you prevent people who shouldn't see important info like social security numbers from getting access to the data.

You can use 'getName' or other get methods to access data that people might need. For example, people addressing a package or email may need a customer's name, but they definitely shouldn't have access to their ssn.

VERY HARD: Object prototype chain and prototypal inheritance exercise.

1. Create a Person constructor that has three properties: name, job, and age.

2. Give the Person an 'exercise' method that console logs whatever you want, e.g. "Running is fun! - said no one ever".

3. Give the Person a 'fetchJob' method that console logs the person's name and job, e.g. "Brad is a back-end developer".

See next page…

4. Create a Programmer constructor that inherits all the members from Person with an additional 'languages' property that is passed in and a busy property that is NOT passed in and is set to true by default.

5. Give the Programmer a 'completeTask' method that updates the busy property on that programmer to be false. Also give the Programmer an 'acceptNewTask' method that updates the busy property on that programmer to be true.

6. Give the Programmer an 'offerNewTask' method that console logs one thing if the programmer is busy and another if the programmer is not, e.g. should initially log out "Mark can't accept any new tasks right now." and "Mark would love to take on a new responsibility." if the programmer is not busy.

7. Give the Programmer 'learnLanguage' and 'listLanguages' methods that add new languages to the programmer and list off all languages the programmer knows.

8. Test it out - can you create different programmers and run all the methods on them? Does each programmer maintain their own properties properly and independently of the other programmers?

Bonus - ES6 Syntax: Use ES6 Syntax in your answer. Feel free to add on new methods or properties to incorporate the syntax.

```
function Person(name, job, age) { }

function Programmer(name, job, age, languages) { }
```

Submission

Submit the URL to your individual repository that contains your assignment to Canvas. Make sure to check that your content was successfully pushed /uploaded. The deadline is 11:59pm EST, Monday, 5/2/2022.