Quiz 2 Study Guide

Introduction

For your second quiz, you will be asked to complete a series of JavaScript tasks, and to interpret some sample programs. These tasks / programs will be similar to those found in Tutorial 11. To study for the quiz, please review the slides, required readings, and exercise files from Weeks 6-12. I have summarized the topics that you will be quizzed on below:

1. The DOM

Study the relevant slides and exercise files. Know how to:

A. Target DOM elements by tag, class, and id

document.querySelector()	document.querySelector("#my_element")
	document.querySelector("p")
	document.querySelector(".my-announcements")

B. Access and/or update a DOM element's attributes

Attribute	Example
className	document.querySelector("div").className = "panel";
innerHTML	document.querySelector("div").innerHTML = "hi!";
src (for images)	document.querySelector("img").src = "some_image_url"
href (for links)	document.querySelector("a").href = "http://site.com";

C. Access and/or update a DOM element's style properties

Property	Example
width	document.querySelector("div").style.width = "200px";
height	document.querySelector("div").style.height = "200px";
background color	document.querySelector("div").style.backgroundColor = "hotpink";
border width	document.querySelector("div").style.borderWidth = "5px";
padding	document.querySelector("div").style.padding = "10px";
display	document.querySelector("div").style.display = "none";

2. Variables and Data Types

- Study the relevant slides and exercise files.
- Know the JavaScript data types
- Know why data types matter and be able to explain it.
- Know how to write something to the console (console.log).

- Know what a variable is and how to use one.
  - What does it mean to declare a variable?
  - What does it mean to assign a value to a variable?
- Know the naming conventions (camel case, mnemonic naming).
- Know the const, let, and var keywords and when to use them (and not use them).

A. JavaScript Data Types

number	For numbers of any kind: integer or floating-point	1.4, 33, 99999999
string	For strings (text). A string may have one or more characters, there's no separate single-character type	'hello world!'
boolean	for true/false.	true, false
null	for unknown values – has a single value null	null
undefined	for unassigned values – has a single value undefined	undefined
object	for more complex data structures.	{ name: 'ralph', species: 'dog' }
symbol	for unique identifiers (we won't be using this one)	

B. Lists

- Know how to create a list.
- Know how to access items in a list (list indexing)
- Know how to add and remove items from a list (push and pop)
- Know how to loop through items in a list.

Creating a list	```const colors = ['red', 'green', 'blue', 'orange'];```
	```const emptyList = [];```
	```const nums = [44, 67, 121];```
	```const mixed = [44, 'red', [1, 2], { id: 4, name: 'Jim' }];```

Finding the length of a list	```colors.length // returns 4```
	```emptyList.length // return 0```
	```nums.length // returns 3```

Accessing items in a list	```colors[0] // returns 'red'```
	```emptyList[0] // error (there isn't a first element)```
	```nums[2] // returns 121```
	```nums[nums.length - 1] // returns 121```
	```mixed[3] // returns { id: 4, name: 'Jim' }```
	```mixed[3].name // returns 'Jim'```
	```mixed[2] // returns [1, 2]```
	```mixed[2][0] // returns 1```

Adding items to a list	```nums.push(12) // adds 12 to the end of nums:```
	``` // [44, 67, 121, 12]```
	
	```emptyList.push(12) // adds 12 to the end of emptyList```
	``` // emptyList: [12]```
	
	```mixed.push(```

| | ```
          { x: 30, y: 20 }    // adds { x: 30, y: 20 } to the end
    )                          // of the "mixed" list
``` |
| **Removing items from a list** | ```
// removes the last item from the list and stores it in the
variable called removedColor.
let removedColor = colors.pop();
console.log(removedColor);      // 'orange'
console.log(colors);            // [ 'red', 'green', 'blue' ]
``` |

C. Objects

- Know how to create an object.
- Know how to access object properties
- Know how to create new properties and assign values to these properties.
- Know how to loop through an array of objects and access properties in a list.

3. Operators

- Study the relevant slides and exercise files.
- Be able to define what an expression is.
- Be familiar with the **assignment operators**
- Be familiar with **arithmetic operators** and their return types
- Be familiar with **comparison operators** and their return types
- Be familiar with **logical operators** and their return types

A. Assignment Operators

See [Lecture 10](#).

=	Assignment operator. Puts the value on the right into the variable on the left. `    let a = 3; // after this line executes, a is holding the value 7.`

+=	Adds the value (right) to the variable (left), and assigns the result to the variable.
	`    let a = 3;`
	`    let a = 3;`
	`    a += 4; // after this line executes, a is holding the value 7.`

++	Increment. Adds 1 to the variable.

-=	Subtracts the value (right) from the variable (left), and assigns the result to the variable.
	`    let a = 3;`
	`    a -= 4; // after this line executes, a is holding the value -1.`

--	Decrement. Subtracts 1 from the variable

*=	Multiples the variable's value (right) by the value on the right, and assigns the result to the variable.
	`    let a = 3;`
	`    a *= 4; // after this line executes, a is holding the value 12.`

/=	Divides the variable (left) by the value (right), and assigns the result to the variable. `    let a = 3;`

```
        a /= 4; // after this line executes, a is holding the value 0.75.
```

B. Arithmetic Operators

See Lecture 10.

+	Addition	Adds values on either side of the operator
-	Subtraction	Subtracts right hand operand from left hand operand
*	Multiplication	Multiplies values on either side of the operator
/	Division	Divides left hand operand by right hand operand
**	Exponent	Performs exponential (power) calculation on operators
%	Modulus	Divides left hand operand by right hand operand; returns remainder

B. Comparison Operators

See Lecture 13.

- Note that all comparison operators evaluate to either true or false (boolean data type).
- Used for loops and if / else statements.

Operator	Description
===	Strict Equality. Both values and data types are equal.
==	Value Equality: If the values of two operands are equal, then the condition becomes true.
!=	If values of two operands are not equal, then the condition becomes true.
>	If the value of the left operand is greater than the value of the right operand, then the condition becomes true.
<	If the value of the left operand is less than the value of the right operand, then the condition becomes true.
>=	If the value of the left operand is greater than or equal to the value of the right operand, then the condition becomes true.
<=	If the value of the left operand is less than or equal to the value of the right operand, then the condition becomes true.

C. Logical Operators

See Lecture 13.

- Note that all logical operators evaluate to either true or false (boolean data type).
- Used for loops and if / else statements.

Operator	Description
&&	If both operands are true then the expression evaluates to true. Otherwise, the expression evaluates to false
||	If either or both operands are true then the expression evaluates to true. If both operands are false, the expression evaluates to false
!	If the operand is false than the expression evaluates to true (and vice versa)

4. Template Literals

See [Lecture 10](#).

Template literals" are "smart strings" that allow you to embed expressions.

- They're convenient for generating larger chunks of HTML that depend on variables or other data.
- They uses the "backtick" character (instead of regular single or double quotes) to indicate that you are specifying a template (above the tab key):

```
const name = "Jane";
const pic = "http://website.com/avatar.png";
const score = 300;
const html = `
    <div class="card">
        <img src="${pic1}">
        <p> ${name}'s high score is: ${score}
        </p>
    </div>`;
```

5. Functions

See [Lecture 11](#).

- Why are functions useful and what is their purpose?
- Know both syntaxes (arrow functions and function declarations).
- What is a function definition?
- What is a function call / invocation?
- What are parameters?

- What are arguments?
- What is a return value and why can it be necessary sometimes?
- What is the "function header" and why is it important?
- What is the "function body"?

6. Event Handlers

See [Tutorial 6](#) and [Tutorial 8](#).

- Know how to attach an event handler to a button.
- Know some of the events (onclick, onchange, etc.). Feel free to look them up on W3Schools or elsewhere. There are a lot of great examples.

7. Conditional Statements

See [Lecture 13](#).

- Know how to use the comparison and logical operators to determine whether something is true or false.
- Know the if, if/else, and if/else if…/else syntax.
- Be able to nest conditional statements within a for or while loop.
- Be familiar with different contexts for which conditional statements might be used.

A. If statement
```
const balance = 500;
const phone = 600;

// Check if there is enough funds to purchase item
if (phone <= balance) {
    console.log("You have enough money to purchase the item!");
}
```

B. If / else statement
```
const balance = 500;
const phone = 600;

// Check if there is enough funds to purchase item
if (phone <= balance) {
    console.log("You have enough money to purchase the item!");
} else {
    console.log("You do not have enough money to purchase the item!");
}
```

C. If / else if…/else statement

```javascript
// Set the current grade of the student
let day = "Mo";

// Check if grade is an A, B, C, D, or F
if (day == "Mo" || day == "Tu" || day == "We" || day == "Th") {
    console.log("It's a week day");
} else if (day == "Th") {
    console.log("Almost Friday...");
} else if (day == "Fr") {
    console.log("Yay! I love Friday!");
} else {
    console.log("It's the weekend!");
}
```

8. Loops

See lectures 14-16 and [these slides](#).

- Know how to use the comparison and logical operators to determine whether to enter the loop, and when to exit the loop.
- Know the syntax for **for loops** and **while loops**
- Be familiar with different contexts for which loops might be used.
- Be able to iterate through all the items in a list using a loop.
- Be familiar with "famous" applications of looping algorithms (printing all the items in a list, calculating the smallest number, calculating the largest number, etc.).

A. While Loop Syntax

Here is an example of "while loop" syntax:

```javascript
// Set population limit of aquarium to 10
const popLimit = 10;

// Start off with 0 fish
let fish = 0;

// Initiate while loop to run until fish reaches population limit
while (fish < popLimit) {
    // add one fish for each iteration
    fish++;
    console.log("There's room for " + (popLimit - fish) + " more fish.");
}
```

B. For Loop Syntax

Here is an example of "for loop" syntax:

```javascript
// Set population limit of aquarium to 10
const popLimit = 10;

// Initiate for loop to run until fish reaches population limit
for (let i = 0; i < popLimit; i++) {
    // add one fish for each iteration
    console.log("There's room for " + (popLimit - i) + " more fish.");
}
```