**Chapter 1:**

1. Why was LiveScript renamed JavaScript even though its functionality had nothing to do with Java?
   1. To answer this question, We need to take a look at JavaScript history and how it evolved with all of it’s great Libraries & Frameworks such as React.js, Angular.js, etc.…
   2. The beginning of JavaScript:
      1. In mid-1990s, it was an important time for the internet and the development of browsers e.g. “internet explorer” by Microsoft for instance. In 1995, a Netscape developer developed a new scripting language. It was originally named Mocha, soon after it was renamed LiveScript and Later, it was known as JavaScript.
      2. LiveScript is still being used but not as JavaScript since it eventually compiles to JavaScript
      3. T here is an NPM (Node Package Manager) library called LiveScript (npm install livescript), of course it is not as popular as react-router-dom, but it is an evidence that it is not dead & the last commit in it’s github repository was 2 years ago.
   3. As for the 2nd portion of the question “Even though its functionality had northing to do with Java”, I would agree, JavaScript is a scripting language and has it’s usages however, Java is another animal, it is an OOP (Object Oriented Programming) Language just like how LiveScript is functioning.
   4. JavaScript was renamed that way because the plan in the beginning to evolve the webpages development using Java. That’s why Netscape and Sun (a computer services companies) had a license agreement
   5. What is common about Java & JavaScript, that both can be used to create native Android Apps using Kotlin or React Native. Other than that, I haven’t seen much in common.
   6. A lot could be stated here in this question, but I tried to make it as comprehensive and short as possible.
2. Write a line of code to initialize a variable for a dog named Fido with a weight of 22 lbs.

Shape, rectangle

Description automatically generated with medium confidence

1. Write the HTML syntax for a blank web page with a title of “Chapter 1” for use as a wrapper for JavaScript.

Graphical user interface, text

Description automatically generated

1. Write a while loop that will scoop 3 scoops of ice cream for me. Output the results to the browser.
   1. Check Github repository (<https://github.com/AhmedAbdelRazak/RCC/tree/master/WhileLoop>)

Text

Description automatically generated

1. Write a while loop that will scoop 3 scoops of ice cream for me and will let me know when I have run out of ice cream. Output the results to an alert window.
   1. Check Github Repository (https://github.com/AhmedAbdelRazak/RCC/tree/master/AlertWindow).
2. <html>
3. <head>
4. <link rel="stylesheet" href="styles.css" />
5. <title>While Loop window alert!</title>
6. </head>
7. <body>
8. <script>
9. //Creating an unordered list element to store the array in it
10. var ulElement = document.createElement("ul");
11. document.body.appendChild(ulElement);
12. //Creating Scoop Array To iterate
13. const scoopArray = ["Scoop1", "Scoop2", "Scoop3"];
14. //loop should end after iterating through array scoopArray
15. var loopEnd = scoopArray.length;
16. //initial value
17. var i = 0;
18. while (i <= loopEnd) {
19. let liElement = document.createElement("li");
20. let text = document.createTextNode(
21. scoopArray[i]
22. ? scoopArray[i] + " Delicious Icecream ;)"
23. : "No More Ice Cream",
24. );
25. liElement.appendChild(text);
26. liElement.style.listStyle = "none";
27. liElement.style.marginTop = "10px";
28. liElement.style.animation = `show 600ms ${i}000ms cubic-bezier(0.38, 0.97, 0.56, 0.76) forwards`;
29. ulElement.appendChild(liElement);
30. i++;
31. if (liElement.textContent === "No More Ice Cream") {
32. liElement.style.color = "red";
33. }
34. if (scoopArray[i] === undefined && loopEnd === i) {
35. setTimeout(() => {
36. //launching the alert window after 3 seconds since this is what how I did the animation in CSS ;)
37. window.alert("We don't have ice cream anymore my friend!");
38. }, 3000);
39. }
40. }
41. </script>
42. </body>
43. </html>
44. Explain the fundamental differences between output to the browser, to an alert window, and to the console. When would it be best to use each function? Show me an example in code of each function.
    1. Output to the browser:
       1. Creating html element in javascript
       2. The example below is creating <li> or list tag using javascript



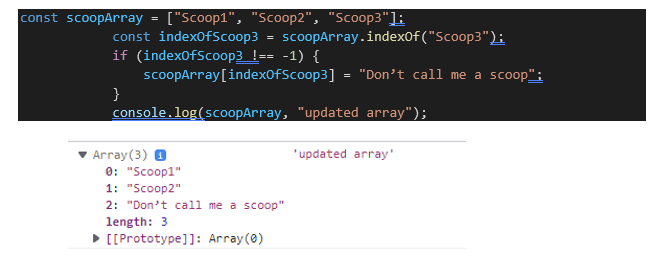
* + 1. Then we can append text to it as demonstrated in the example below



* + 1. The best scenario to use Output to the browser usually when you’re 100% sure that the variables applied are good to go. It is not recommended though to use output in the browser right away.
  1. Output to an alert window:
     1. It is recommended to use the alert window if you’re creating an app that has a delete button or filling a form, if the user tried to delete something or leave the page without completing a form, an alert window should pop.



* 1. Console logging:
     1. This is the most common and effective way to see outputs before adding it to the browser.
     2. it is usually being used to check output in a given function such as a function the renders the sum or another that renders a filtered array. A lot of examples could be written but the above 2 just to mention a few.
     3. The example below is simply a trial to substitute element 3 (index 2) in the array from “Scoop3” to “Don’t call me a scoop”:



1. What is the Document Object Model used for in JavaScript?
   1. Document Object Model also known as the “DOM”. A document object is being created when an html document gets loaded in the browser.
   2. It is simply a tree of multiple methods or attributes. Each branch of the tree ends with an event of action AKA node. DOM methods or properties allow programmers to manipulate with document’s structure, style or content e.g., onClick event listener, so programmers use the DOM to deploy an action when a user clicks on a button. E.g. onClick => submit a form to a database.
   3. Document Object properties:
      1. Document:
         1. Anchor
         2. Form
            1. Text
            2. textArea
            3. Checkbox
            4. Button
   4. Document Object Methods:
      1. getElementById() which returns element or an html tag with a given Id.
      2. getElementByClassName() which returns all elements with the same class
2. What does this code do? setTimeout (wakeUpUser, 4000);
   1. Assuming that wakeUpUser is a function, so the above code will deploy/ launch this function after 4000 ms or 4 seconds.
3. Write a line of code that will calculate the area of a circle. (Area = PI \* Radius Squared)
   1. Please check GitHub repository (<https://github.com/AhmedAbdelRazak/RCC/tree/master/CalculateCircleArea>)

<html>

    <head>

        <link rel="stylesheet" href="styles.css" />

        <title>Circle Area</title>

    </head>

    <body>

        <div

            class="formWrapper"

            style="

                margin-top: 100px;

                text-align: center;

                font-size: 1.5rem;

                font-weight: bold;

            "

        >

            <label> Please fill in the circle radius? </label>

            <br />

            <input

                type="number"

                id="radius"

                placeholder="Circle Radius"

                style="width: 50%; padding: 10px; text-align: center; font-size: 1.2rem"

            />

            <div>

                <button onclick="buttonClicked()">Get Circle Area</button>

            </div>

            <div id="finalOutput" style="display: none; margin-top: 20px">Hi</div>

        </div>

        <script>

            //a function that has a parameter of circleRadius

            function circleAreaCalculation(circleRadius) {

                return circleRadius \* circleRadius \* Math.PI;

            }

            //this is an even listener onclick from the html

            const buttonClicked = () => {

                //storing the radius in a variable from the input field of id radius

                var radius = document.getElementById("radius").value;

                var finalOutputDiv = document.getElementById("finalOutput");

                finalOutputDiv.textContent =

                    "Area of the Circle: " +

                    radius +

                    "\*3.14 = " +

                    circleAreaCalculation(radius).toFixed(2);

                finalOutputDiv.style.display = "block";

            };

            var finalOutput = document.getElementById("finalOutput");

        </script>

    </body>

</html>

1. Explain how JavaScript works and fits into the web page ecosystem.
   1. First the web page is being created using HTML and CSS elements that are already being defined to the Document Object Model AKA the DOM.
   2. After the web page loads the HTML and CSS, it executes the JavaScript code in order (top left, to bottom right)
   3. As for how it fits into the web page, JavaScript is mandatory for any dynamic web page. For instance, if you looked at the program in the previous question, you will find there is a button, This button is useless without using JavaScript since I added an event listener once the button is clicked, the radius should be calculated, not only this, after it is calculated, I want to re-style the element to change it from display:”none” to display:”block”

**Chapter 2:**

1. When is a logical AND operator true? When is it false?
   1. AND operator is true if and only if the conditions is TRUE.
      1. E.g. var x=5; var y=7
      2. Console.log(x<y && x==5) => This statement should return true.
   2. Otherwise, it will return false
   3. Summary:
      1. TRUE && TRUE => TRUE
      2. TRUE && FALSE => FALSE
      3. FALSE && TRUE => FALSE
      4. FALSE && FALSE => FALSE
2. When is a logical OR operator true? When is it false?
   1. OR operator is true if at least one condition is true.
      1. E.g. var x=5; var y=7
      2. Console.log(x<y || x!==5) => This statement should return true despite the 2nd condition is false
   2. OR operator is FALSE if and only if both conditions are FALSE
   3. Summary:
      1. TRUE && TRUE => TRUE
      2. TRUE && FALSE => TRUE
      3. FALSE && TRUE => TRUE
      4. FALSE && FALSE => FALSE
3. Write a line of JavaScript code that will return a random number between 1 and 100
   1. We usually should use Math.random() built in function to generate a random number from 0.0000001 to 0.999999

Graphical user interface, text, application, email

Description automatically generated

* 1. Of course the digits after the decimal point should be infinity because it is considered to be irrational number
  2. Now to get a digit w/n 1 and 100 interval, we will need to put into consideration that the numbers from 1 to 100 are integers so there is no decimal should be there.
  3. To return an integer, we should consider using functions Math.ceil() or Math.floor()

Graphical user interface, text, application, email

Description automatically generated

* 1. Since the return value will always be 0 or 1 only if I used the functions stated in point d, therefore the work around should be **multiplying by 100 and add 1 to the value** this is if I decided to use Math.floor() however, Math.ceil() couldn’t be used in this scenario because using Math.random with Math.ceil() can return 0 eventually so the random number will be out of the required interval

Graphical user interface, text, application, email

Description automatically generated

1. What is the value of a variable, if it is not initialized?
   1. I genuinely never tried it, but I used the “Battleship Assignment” and I used the variable “guess” since it was not initialized.

A screenshot of a computer

Description automatically generated with medium confidence

* 1. So after looking at the console, I saw that not initialized variables should return value of undefined.

Graphical user interface, text, application

Description automatically generated

1. What function do you use to get input from a user?
   1. Based on chapter 2 and specially in the battleship assignment, you can take input from a user using the prompt() function.
   2. E.g. prompt( “React, aim, fire! (enter a number from 0-6)” )
   3. This line is a bit away from chapter 2 but you can get user input by using <input /> tag by simply assign an Id to it to use getElementById(“TagId”).value and bam, you got the value.
   4. Thank you Dr. Ian for your advise in the email though, the response to this question could’ve been in only 1 line but I’m still learning and will always learn 😊
2. Compare and contrast the use of equal (=) and the use of double equal (==) in programming.
   1. The use of (=) is only when I assign a value to a specific variable (e.g. var x = 2) which means that the value of x is 2
   2. The use of (==) is a comparative operator and it usually returns a Boolean (true or false).
   3. E.g. var x=2; if(x==2){console.log(true)} else {console.log(false)}
3. Two or more nested if statements can be rewritten using which logic operator?
   1. AND Operator
   2. Example: var x = 5; var y=9
      1. If(x==5) {

If(y==9) {

Console.log(true);

}

};

* 1. The example above could be rewritten as:
     1. If(x==5 && y==9) {

Console.log(true);

};

1. Two or more if-else if statements can be rewritten using which logic operator?
   1. OR Operator.
   2. Example: var x = 7;
      1. If(x==2){

Console.log(true);

} else if (x==7) {

Console.log(true);

} else {

Console.log(false);

};

* 1. The example above could be:
     1. If(x==2 || x==7) {

Console.log(true);

} else {

Console.log(false);

};

1. Write a program that checks to see if it will rain. The user will input the data and the program will decide if it will rain or not. In this universe, it always rains when the temperature is above 60 degrees Fahrenheit and lower than or equal to 80 degrees Fahrenheit. Paste the code here.
   1. Here is a Github repository for the program (<https://github.com/AhmedAbdelRazak/RCC/tree/master/TemperatureProg>).

Text

Description automatically generated

1. Include a screenshot for the above rain program here.

Graphical user interface

Description automatically generated

**Chapter 3:**

**Ch 3 Written Assignment**

1. In your own words, state what you think the thing-a-ma-jig is supposed to do?
   1. It did teach me how the code should be executed since it is not necessarily that the code should be executed in order.
   2. It did teach me the difference between Global vs Local variables and how the Global variables are being shadowed when used in a function (e.g. clunkCounter variable was constantly changing inside of the display function)
   3. As for the actual code, when passing 1 or 2 to the thingamajig function, it returned clunkCounter as 1, 2 respectively however, when passing value of 3, it returned 6, value of 4 returned 24, value of 5, returned 120, value of 6, returned 720, value of 7 returned 5040.
      1. I tried to see the sequence of the output “1, 2, 6, 24, 120, 720, 5040,…”
      2. With that being said, it looked like the output usually is the factorial of the argument passed to the function thingamajig.
         1. 1! = 1;
         2. 2! = 2\*1 = 2;
         3. 3! = 3\*2\*1 = 6;
         4. 4! = 4\*3\*2\*1 = 24;
         5. 5! = 5\*4\*3\*2\*1 = 120;
         6. 6! = 6\*5\*4\*3\*2\*1 = 720;
         7. 7! = 7\*6\*5\*4\*3\*2\*1 = 5040;
   4. In the text book page 121 it is stated that the Thing-A-Ma-Jig was invented by a curious chap who was fascinated by rearranging words. It is like DOG rearranged is God, OGD, DGO, GDO and ODG. So if a word has three letters, the Thing-A-Ma-Jig says you can make six total combinations from those letters. In the end of the paragraph, it looks like my assumption of rendering mathematical factorial was correct!
   5. So the final answer to this question away from what skills I gained is that it always retrieves the factorial of the argument passed, so if I passed 5 for instance, the thing-a-ma-jig program will render factorial 5 which is 120
2. What is the purpose of the variable, facky? Why do we multiply facky by size?
   1. Assuming that my response and analysis in question 1 is correct, facky variable is a must in this program to calculate the factorial of size.
   2. The value of size is being passed when executing line code thingamajig().
   3. If we decided to pass value of 3 as an argument, then size=3; facky=1; Now we will need to get factorial 3 which should be 3\*2\*1, we cannot do this without a helper local variable which in our case here is variable facky
   4. When executing the while loop in function thingamajig, we will notice that an expression facky = facky\*size, and here, all the magic happens.
   5. For this example, I passed value of 3 so there will be two iterations in the while loop to break out.
   6. The first iteration will be facky = 1\*3 the second iteration will be facky = 3\*2 which will return eventually facky as value of 6 which is 3!
   7. I hope I was able to elaborate and demonstrate properly 😊
3. Compare and contrast global variables with local variables in JavaScript. What is the scope of each? Show me an example in code of both a global variable and a local variable.
   1. As for comparing global vs local variables:
      1. Global variables could be used inside any function within the code and it should be defined/ declared in the body of the script tag and outside of the function.
      2. Local variables can only be used in the same function it was identified in (locally scoped).
      3. When a global variable is being used in a function, it will shadow the original value it was identified with.
   2. As for the scope of each:
      1. The variables defined outside a function are globally scoped
      2. The variables defined inside a function are locally scoped.
      3. We should consider identifying a global variable when we know that the same variable will be used in multiple different functions in a JavaScript code to reduce redundancy and avoid re-identifying new variables in all functions.
      4. We should consider identifying a local variable when we need a helper variable that won’t be used outside of the function. The best example for this is the facky variable in function thingamajig.
   3. As for the example in a code:
      1. A good example should be found in the thing-a-ma-jig code
      2. Please read the comments in green in the code below.

<script>

        function clunk(times) {

          //num is a local variable and couldn't be used anywhere else in the code except w/n the clunk() function

          var num = times;

          while (num > 0) {

              display("clunk");

              num = num - 1;

          }

        }

        function thingamajig(size) {

          // facky is a local variable and couldn't be used anywhere else in the code except w/n the same function {thingamajif()}

          var facky = 1;

          clunkCounter = 0;

          if (size == 0) {

                display("clank");

            } else if (size == 1) {

                display("thunk");

            } else {

                while (size > 1) {

                    facky = facky \* size;

                    size = size - 1;

                }

                clunk(facky);

            }

        }

      function display(output) {

          console.log(output);

          //this global variable will shadow the original value since it was  used in this function {display()}

          clunkCounter = clunkCounter + 1;

        }

        // this is a global variable used in the display function as well as in the thingamajif function

        var clunkCounter = 0;

        thingamajig(7);

        console.log(clunkCounter);

    </script>

1. What happens to the scope of a variable if you leave off “var” when declaring it?
   1. JavaScript will deal with it as a global variable (stated in lecture at “23:50).
   2. Here is an example of a code:

Graphical user interface, text, application, chat or text message

Description automatically generated

* 1. We should pay close attention to variable z, it is being added inside function {trial()} however when I console logged z, you will notice that it returned 9 which is the expected value

1. What gets returned from a function without a return statement?
   1. As stated in the lecture (at 12:40), if I don’t return a value in a function, it will be a procedure which means it performs a code out of order but it doesn’t return anything back in terms of value. it doesn’t give a value back to the function itself.
   2. So in order to pass a value back and put it into memory once more in the main program, then a return statement should be included.
   3. And if I want to see the output without a return statement, the function should include console.log(); or document.window() which means it will output but not return anything.
2. What happens to the global and local variables when you press the reload button in the browser?
   1. Stated in the textbook page 108, Reloading a page is like starting over from scratch as far as the variables are concerned. And if any code was in the middle of executing when you reload the page, any local variables will disappear.
   2. So global variables will be in their initial value.
   3. Local variables will disappear and being re-evaluated based on the code itself.
3. What happens when a local variable shadows a global variable?
   1. Stated in the lecture (24:36), the scope global will become local and the local will gain focus from the global.
   2. And it is important to know that the local and global variables have no effect on each other, if you change one, it has no effect on the other. They are independent variables.
4. Is it dangerous to use all global variables in a program? Explain.
   1. Stated in the textbook page 108 that global variables are often being overused in JavaScript as JavaScript doesn’t enforce a lot of structure or overhead on you. However, the downside is when writing serious code this way which is declaring everything as global variables, and it has to be changed and maintained over the long term. So it is highly recommended to organize the code in a modular way.
   2. So it is not dangerous but not recommended either.
5. What happens to the third argument when you pass 3 arguments into a function with 2 parameters?
   1. As stated in the lecture at (5:45), the 3rd argument will just get dropped, so JavaScript will take the first 2 and leave the 3rd one.
6. When would you use a function that has no parameters?
   1. As stated in the very beginning of the lecture, the purpose of functions or being functional is to avoid redundancy and stop re-writing code over and over again.
   2. Few scenarios you might will need to use/ define a function that has no parameters which for instance if the function returns a prompt with hard coded value such as “Thank you”. Now, we can call this function every time we want to send a Thank you message, instead of coding prompt(“Thank you ……”) we can just call the function.
   3. It is important to know few built-in functions could be used without passing any parameters such as (Math.random() OR Math.PI).

**Chapter 4:**

**Ch. 4 Written Assignment**

1. What is an array with undefined values called?
   1. **Sparse arrays** which occur when there are undefined items/ values in the middle of an array
2. Code a for loop that iterates and prints out the individual scores of this array: var scores = [4, 5, 6, 8, 2, 5, 9, 3]

**for(var i=0; i<scores.length; i++) {**

**console.log(“Score #” + i + ”: ” + scores[i]);**

**document.write(“Score #” + i + ”: ” + scores[i]);**

**};**

1. Code a for loop that iterates and prints out the highest score of this array: var scores = [4, 5, 6, 8, 2, 5, 9, 3]

var highestScore = 0;

**for (var i=0; i<scores.length; i++) {**

**if (scores[i] > highestScore) {**

**highestScore = scores[i];**

**};**

**};**

console.log(“Highest Score is “ + highestScore);

document.write(“Highest Score is “ + highestScore);

1. What is the index of the first item in an array?
   1. Index of 0.
   2. E.g. var myArray = [4,5,10,15], to get the first item which is 4, I should trigger it by simply calling myArray[0] 🡺 4.
2. What 2 main types of variables can be stored in an array?
   1. In the lecture at (3:10), it is mentioned that arrays can store strings, numbers, (true, false) Boolean, anything you like.
   2. **Since the question has specified 2 main types, I will go with the most used types of variables which are Strings & Numbers.**
3. What is the relationship between the length of the array and an index of the array.
   1. The index of the array: it is the address/ position of an item w/n the array.
      1. var myArray=[6,3,7,9,1]; myArray[4] 🡺 1
   2. The length of the array is the total count of items in the array
      1. Taking the same array myArray => myArray.length 🡺 5
   3. **Now to get the relationship, you will notice that the last element/ item in an array has an index of array.length – 1**
4. An array is a(n) \_\_\_\_\_\_\_\_\_ data structure.
   1. **Ordered** data structure.
   2. Arrays are a data structure for **ordered data.**
5. What is the value of an item in an array if you don’t specify one in the code?
   1. For this example var myArray = [1, , 9 , “”]

Graphical user interface

Description automatically generated

* 1. As demonstrated in the screenshot above, the value of an unspecified item in an array will be undefined.
  2. This array is called Sparse array.

1. When we use a loop to iterate over an array, what is the property (of the array) that we use to tell the for loop to stop?
   1. For loop: this part is called conditional test. Each time we loop, we perform this test, and if it is false, we stop.
   2. Reference, textbook page 140.
2. What is the name of the type of operator that is used to increment a loop variable? (Hint: the answer is a single word that starts with a “P”)
   1. **Plus**.
   2. **PlusPlus (if you consider it a single word)**

**Chapter 5:**

**Ch. 5 Written Assignment:**

1. What is an object?
   1. Object is a collection of properties.
   2. Those properties or elements hold a value that can be string, number or Boolean.
   3. Not only this, we can embed a function in an object but in this case it will be called a method.
   4. There are multiple built in objects in JavaScript e.g. Date, Math, Exp, JSON
2. How do you access a property?
   1. To access a property in an object, I should use the dot notation.
   2. The name of the variable containing the object then a period, then the name of the property.
3. Compare and contrast methods to functions.
   1. Methods: they are functions that are been assigned to a property name in an object

e.g. var car = {

make: ”Toyota”,

year: 2000,

//chosenCar will be considered a method.

chosenCar: function(){

//here will be the function

}

}

To call a method in the code, I should use the “dot” notation (car.chosenCar())

* 1. Functions: they are being defined independently away from an object and could be called without the dot notation.
  2. So overall, a method is just like a function except that it is in an object.

1. What is encapsulation? How is it used in JavaScript?
   1. It is putting/ packing all data into a single unit or component as it is considered to be a programming technique.
   2. Usage in JavaScript: Objects encapsulate, or hide, the complexity of the state and behavior in that object.
   3. And thank you for the support Prof. 😊
2. What is the operator that deletes properties from objects?
   1. We can delete properties from objects using the delete operator.

e.g. var car = {

make:”Toyota”,

Year: 2000

};

//the line below will delete the year property from the car object.

delete car.year

1. What is a behavior?
   1. Stated in the lecture at (32:00) as well as implemented in lab 2b that a behavior is simply calling/executing a method/ function in an object to implement/ render set of actions or events.
   2. A behavior is part of Object Oriented Programming in JavaScript.
   3. A behavior could be adjusted by adjusting the method in the object.
   4. A behavior can affect the values of the same object.
      1. As deployed in lab 5b, the “fuel” element/ property was changing when calling method fiat.addFuel(2) in the fiat object.
   5. An object has both state and behavior. State can affect behavior, and behavior can affect state.
   6. We also learned in Ch. 1 that JavaScript is used to add behavior to web pages but is also used to script applications like Adobe Photoshop, OpenOffice and Google Apps, and is even used as a server-side programming language.

**Chapter 6:**

1. What does the getElementbyId () method do?
   1. It is used to grab an element from the DOM. The getElementbyId method takes an id and returns the element that has that id.
   2. Element/ element object: it is the browser’s internal representation of what type into your HTML file, such as <p>…text….</p>. When the browser loads and parses the HTML file, it creates an element object for every element in the page and adds all those element objects to the DOM.
2. What does the innerHTML element do?
   1. The innerHTML property is used to read or replace the content of an element. When looking at the value of innerHTML, it will be noticed the content contained within the element, not including the HTML element tags.
   2. E.g.

<script>

          var access = document.getElementById("code9");

          var code = access.innerHTML;

          code = code + " midnight";

      alert(code);

      </script>

* 1. In the example above, I will update the content within the element that has Id “code9” by simply adding the text “midnight” and show the result in an alert box.

1. What does the document object do?
   1. The document object is used to get access to the DOM from our code. The document object is a built-in object that comes with a bunch of properties and methods, including getElementById.
2. Review the code on page 230. How would I alter the code.js file to get the code8 element and add the word “rise” to it and return it in an alert window?
   1. Here is the code:

<script>

            var access = document.getElementById("code8");

            var code = access.innerHTML;

            code = code + " rise";

alert(code);

      </script>

* 1. Here is the output

Text

Description automatically generated

1. Review the code on page 236, draw a DOM and insert it here.

Diagram, box and whisker chart

Description automatically generated

1. For the code on page 236, how would I alter the code to make sure that the movie2 element displays the text color in red for this line: “Playing at 5:00 pm, 9:00 pm”?
   1. As in page 257, I will assign an id name to the required paragraph tag.
   2. From this id, I will use setAttribute method to insert a class name to this specific element that contains the required text.
   3. Here is an example for the code:

Text

Description automatically generated

* 1. And here is the output:

Text

Description automatically generated

1. Describe how an event handler works in the browser.
   1. If there is a big important event that’s going to occur, and you definitely want to know about it, Say that event is the “page is loaded” event. A common way to deal with that situation is through a callback, also known as an **event handler**
   2. A callback works like this: give a function to the object that knows about the event. When the event occurs, that object will call you back, or notify you, by calling that function. You’re going to see this pattern in JavaScript for a variety of events.
   3. You can use the window object’s onload property to set an **event handler** or callback function for the load even.
2. How does the getAttribute method work? Show me an example of it in a line of code in JavaScript.
   1. You can get the **value** of an element’s attributes using the **getAttribute** method.
   2. Here is a code example:

Text

Description automatically generated

* 1. Here is the output:

Graphical user interface, text, application

Description automatically generated

1. How does the setAttribute method work? Show me an example of it in a line of code in JavaScript.
   1. You can set the value of an element’s attributes using the setAttribute method
   2. There is a pretty descriptive example was provided in this assignment in question #6 but here is another one inserting a src attribute for an img tag:

Text

Description automatically generated

* 1. Here is the output:

Graphical user interface

Description automatically generated

1. What is the DOM?
   1. The **DOM** or **Document Object Model** is the browser’s internal representation of your web page.
   2. The browser creates the DOM for your page as it loads and parses the HTML.
   3. You can get access to the DOM in your JavaScript code with the document object. e.g. document.getElementById(“id”).

**Chapter 7:**

1. var addit = 6 + “5”;

**It will be a concatenation not addition.**

**It should return “65”**

1. var addit = “6” + “5”;

**It will be a concatenation not addition.**

**It should return “65”**

1. var x=5; x += "2";

**x+ = “2” should be evaluated as x = x + “2”**

**since x = 5 therefore it should evaluate x = 5 + “2”**

**Now, I should expect it will return a concatenation not addition.**

**The final output should be “52”**

**Graphical user interface, text, application, email

Description automatically generated**

1. 1 == “1”

**== operator won’t consider the data type**

**It will evaluate 1 == 1 (convert “1” to 1)**

**It should return true**

1. 1 === “1”

**=== operator is a strict equality operator so it will consider the data type**

**It won’t convert the “1” to 1.**

**So the result/ output should be false.**

1. “42” === “42”

**=== is a strict equality operator so data type is considered.**

**String of 42 the same as string of 42 seems truthy 😊**

**So the final result/ output should be true.**

1. 42 !== “42”

**There is no data type conversion in this scenario 42 !== “42”**

**The final result should be true because number 42 is not equal to string 42**

1. c = Math.sqrt(-9);

**As mentioned in the lecture, this will return an imaginary number.**

**Imaginary numbers cannot be represented in JavaScript.**

**Thus, the final output/ result will be NaN**

1. var sub = “1” - “1”;

**With minus, the “1” will be converted to number 1 (stated in page 286)**

**It will evaluate 1 – 1**

**So the final output/ result will be 0**

1. if ( [ ] ) { //code }; //assume that there is code in the braces and the array is defined.

**This is like saying if true.**

**The code block w/n the if statement will be executed since [] is always true.**

1. if ( 0 ) { //code }; //assume that there is code in the braces.

**0 is falsey just like (undefined, null, empty string, false, NaN)**

**Therefore, I should expect that the code block w/n the if statement won’t be executed because the condition will always return false**

1. 8675309 == ""

**It will evaluate 8675309 == false (since I have an empty string here)**

**8675309 == false is false**

**So the return/ output will be false.**

1. false == (5==="5")

**First, the logic between the parenthesis will be executed**

**It is a strict equality operator between the parenthesis so no data type conversion.**

**5===”5” should return false**

**Second, we will have false == false.**

**So the final output/ result for this will be true**

**Chapter 8:**

1. What does the <td> element correspond to on the game board?
   1. The <td> tag represents the cell’s location in the grid, so we have 7\*7 grid which is 49 squares.
   2. Top left square will be “00” so I should make the first <td> element has an id of “00” and the list goes on by order for the 49 square grid for the game (“01, “02”,…..).
2. What is the collision function responsible for?
   1. The collision function/ method takes a ship and checks to see if any of the locations overlap or collide – with any of the existing ships already on the board:

Diagram

Description automatically generated

1. How can you cheat and get the locations of the ships during runtime in the final game?
   1. To cheat, open up the developer console, and type model.ships.
   2. Then press return and you should see the three ship objects containing the locations and hits arrays.
   3. Now you have the inside scoop on where the ships are sitting in the game board.
2. We represent each ship in the game with a/an \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
   1. We represent each ship in the game with an **object** that holds the locations it sits in.
   2. An example of the object is stated in question 9.
3. To add a “hit” to the game, what do we add to the corresponding <td> element?
   1. Based on the game model, the attributes for element <td> are added dynamically when we have a hit.

function displayHit(location) {

var cell = document.getElementById(location);

cell.setAttribute(**“class”, “hit”)**

}

* 1. So the answer should correspond to the <td> tag, we have to add a class “hit” to add a hit to the game.

1. What are the 3 objects that are used in Battleship 2.0?
   1. Model :
      1. Which will hold the state of the game, like where each ship is located and where it’s been hit.
   2. View:
      1. Which is responsible for updating the display with hits, misses and messages for the user.
   3. Controller:
      1. Which glues everything together by handling the user input, making sure the game logic gets played and determining when the game is over.
2. What method is used to set the class of an element?
   1. setAttribute method
3. Each object in the game has \_\_\_\_\_\_\_\_\_\_\_\_ primary responsibility.
   1. Each object in the game has **its own** primary responsibility.
   2. The details of each object in the Battleship game is stated in question 6
4. How would you implement chaining in JavaScript? Explain what chaining is and what it does. Show me an example in code from Battleship 2.0.
   1. Chaining isn’t necessary nor much more efficient, but it does make the code shorter and not redundant which make it easier to read than multiple lines of code.
   2. It is preferable to use chaining wisely though, so we shouldn’t code really long chains, they will be harder to read and understand that way.
   3. With chaining, you can string together object references (using the dot operator), thus combining statements and eliminating temporary variables.
   4. Example from the code, is simply defining all 3 ships in one array of objects variable instead of defining three different variables (var ship1={}; var ship2={}; ……)

A screenshot of a computer

Description automatically generated with medium confidence

1. How are do-while loops and while loops similar? How are they different?
   1. The do whi
   2. le loop is similar to the while loop **except that the condition is checked after the statements in the body of the loop have executed once**.
   3. Example from the code for the generateShipLocations function:

Text

Description automatically generated

**Chapter 9:**

1. What is a target?
   1. The target tells us what element generated the event.
   2. A target holds the object on which the event occurred, it can be different kinds of objects, but most often it is an element object.
   3. Simple example: when clicking on the button:

Graphical user interface

Description automatically generated

1. What does this line of code do: getElementsByTagName(“img”); ?
   1. We should expect to get a list of all <img> tags in our HTML code.
   2. So that line of code can return many elements, one element or even zero elements depending on how many images we have in our page.
   3. That list can be treated like an array, but it’s actually an object called a NodeList.
2. How many threads of control does a browser have?
   1. Page 381-383, we should know that most JavaScript is written to react to events, those kind of events could be user clicking on a page, data arriving from the network, timers expiring in the **browser**, changes happening in the DOM and may more other threads of control the **browser** can have.
   2. Understanding that the **browser** retrieves a page and all of that page’s contents and then renders the page is not what the **browser** only doing, but behind the scene the **browser’s** doing a lot more than just that such as:
      1. Knows when the page is fully loaded.
      2. Keeps track of all clicks.
      3. Knows when a user submits a form
      4. Watches the click and manages timers and timed events
      5. Watches all mouse movement
      6. Identifies the geo location of the browser
   3. **So we can say we have a single thread to run all JavaScript in a given page, and that single thread is being distributed to sub threads that would be responsible for all the variety of events that the browser renders.**
3. What is the name of the property of an event object to know when an event happened?
   1. The **target (event.target)** event property is an event object to know when an event happened.
4. Are events handled synchronously or asynchronously? Why?
   1. Event handled asynchronous.
   2. Because code written to handle events is different from code that executes top to bottom and then completes. Event handlers can run at any time and in any order.
5. What is an event handler’s main purpose?
   1. Event handlers purpose is to handle events. Handlers are typically small pieces of code that know what to do when an event occurs. In terms of code, a handler is just a function. When an event occurs, its handler function is called
6. List and define all the events discussed in Chapter 9. (Hint: Event Soup)
   1. click
      1. Get this event when you click (or tap) in the page.
   2. load
      1. The event you get when the browser has completed loading a web page.
   3. mousemove:
      1. when you move your mouse over an element you’ll generate this event.
   4. keypress:
      1. This event is generated every time you press a key.
   5. unload:
      1. This event is generated when you close the browser window, or navigate away from a web page.
   6. mouseover:
      1. When you put your mouse over an element, you’ll generate this event.
   7. mouseout:
      1. When you move your mouse off an element, you’ll generate this event.
   8. resize:
      1. Whenever you resize your browser window, this event is generated.
   9. dragstart:
      1. If you drag an element in the page, you’ll generate this event.
   10. touchstart:
       1. On touch devices, you’ll generate a touchstart event when you touch and hold an element.
   11. play:
       1. You will get this event when you click the play button in your <video> tag.
   12. pause:
       1. This one when you click the pause button.
   13. drop:
       1. You’ll get this event when you drop an element you’ve been dragging.
   14. douchend:
       1. You’ll get this event when you stop touching
7. Older versions of Internet Explorer have a different event model from other browsers. Discuss what they are and how they work.
   1. IE8 and older browsers do support most of the “on” properties you can use to assign event handlers.
   2. IE8 and older browsers use a method named attachEvent instead of the addEventListener method.
   3. When an event is triggered and your event handler is called, instead of passing an event object to the handler, IE8 and older versions store the event object in the window object
   4. And here is a simple example to demonstrate how they work:

Diagram

Description automatically generated with medium confidence

1. Consider the code: for (var i = 0; i < images.length; i++) { images[i].onclick = showAnswer; } How would you alter this to set the normal (not blurred) image on mouse over and reblur the image on mouse out? (It’s 2 lines of code)

**for (var i = 0; i < images.length; i++) {**

**images[i].onmouseover = showAnswer;**

**images[i].onmouseout = reblur;**

**//We should consider removing the timer (setTimeout(reblur, 2000, image) from the // showAnswer function since it won’t be needed after refactoring the code as required in // this question.**

**}**

1. Write a line of JavaScript code that sets the interval of function ticker( ) to 5 seconds.

The required line assuming that function ticker() is defined ( **setInterval( ticker, 5000)** )

Here is the full code:

A screenshot of a computer

Description automatically generated with medium confidence

Here is the output:

Graphical user interface, text, application, email

Description automatically generated

Here is the GitHub repo:

<https://github.com/AhmedAbdelRazak/RCC/blob/master/Chapter9/WrittenAssign_Q10/index.html>

**Chapter 10:**

1. Show me an example of code using a function expression. Explain how this is different than the standard function declaration.
   1. A function expression returns a reference to the new function created by the expression.
   2. The biggest difference is that function expression are being invoked or called whenever the developer decides unlike function declaration which is being evaluated and the browser assigns the resulting reference to a variable in case it is going to be used.
   3. Here is a quick example:

Diagram

Description automatically generated with low confidence

1. When a browser parses a page, before it evaluates any code, it looks for \_\_\_\_\_\_\_\_\_ .
   1. When the browser parses the page, before it evaluates any code, it’s looking for **function declarations.** When the browser finds one, it creates a function and assigns the resulting reference to a variable with the same name as the function.
2. Function declarations are evaluated \_\_\_\_\_ the rest of the code is evaluated.
   1. Function declarations are evaluated **Before** the rest of the code is evaluated or handled.
3. Can you hold function references in variables?
   1. **Yes**, it is possible to hold function references in a variable
   2. A function expression returns a **reference** to the new function created by the expression
   3. Furthermore, a **function reference** is a first class value, which means it can be assigned to variables, included in data structures, passed to functions or returned from functions.
4. Why is a function declaration not an expression?
   1. A function declaration is described as a statement rather than an expression.
   2. It has a hidden assignment that assigns the function reference to a variable.
5. You need to start thinking of a function as a \_\_\_\_\_\_\_, just like other objects and primitive types.
   1. You need to start thinking of a function as a  **Value.**
6. Implement the Shell Game on page 440 (Sharpen Your Pencil). Show screenshots and code.
   1. The code in the GitHub link: <https://github.com/AhmedAbdelRazak/RCC/blob/master/Chapter10/shellGame.html>
   2. Here is the final output:

Graphical user interface, application, website, Teams

Description automatically generated

* 1. Here is the code:

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="utf-8" />

        <title>Shell Game | Q7</title>

        <script>

            //The variables below hold references to the winner and loser functions.

            //We can assign and reassign these references to other variables just like with any value.

            var winner = function () {

                alert("WINNER!");

            };

            var loser = function () {

                alert("LOSER!");

            };

            winner();

            // let's assign to other variables for practice

            //At any time, we can invoke a reference to a function

            var a = winner;

            var b = loser;

            var c = loser;

            a();

            b();

            // now let's try your luck with a shell game

            c = a;

            a = b;

            b = c;

            c = a;

            a = c;

            a = b;

            b = c;

            a();

        </script>

    </head>

    <body></body>

</html>

1. Implement the array sort method program on pages 459 and 460. Change the numbers that are sorted; add a bigger range of numbers than what's in the book. I want the number array that is sorted to be at least 20 numbers. I need to see screenshots of the output and the code.
   1. The code in this GitHub link: <https://github.com/AhmedAbdelRazak/RCC/blob/master/Chapter10/sortN.html>
   2. Final output of sorting descendingly:

Graphical user interface, application, Word

Description automatically generated

* 1. Final output of sorting ascendingly:

Graphical user interface, application

Description automatically generated

* 1. The code:

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="utf-8" />

        <title>Sort Numbers | Q8</title>

    </head>

    <body>

        <script>

            var numbersArray = [

                60, 50, 62, 58, 54, 54, 82, 94, 22, 39, 45, 71, 18, 56, 25, 10, 35, 49,

                75, 27, 16, 70,

            ];

            function compareNumbersDesc(num1, num2) {

                if (num1 > num2) {

                    return 1;

                } else if (num1 === num2) {

                    return 0;

                } else {

                    return -1;

                }

            }

            numbersArray.sort(compareNumbersDesc);

            console.log(numbersArray, "output");

        </script>

    </body>

</html>

**Chapter 11:**

**Chapter 11 Written Assignment:**

1. Define Closure. Show an example, in JavaScript, of a program with a closure from chapter 11 that is not the lab. Explain how the program works.
   1. Closures are a function along with a referencing environment
   2. A closure captures the value of variables in scope at the time the closure is created.
   3. The closure contains the actual environment, not a copy.
   4. Example:

Graphical user interface, diagram

Description automatically generated with medium confidence

1. Implement the following function that takes a password argument and returns a function that accepts a password guess and returns true if the guess matches the password, with a closure.

function makePassword ( **password** ) {

return **function guess(passwordGuess)** {

return (**passwordGuess === password** );

};

}

**var tryGuess = makePassword(“secret”);**

**console.log(“Guessing “nope”: “ + tryGuess(“nope”));**

**console.log(“Guessing ‘secret’: ” + tryGuess(“secret”));**

1. Implement a program in JavaScript that bakes a cake and sets a timer for 10 minutes.
   1. setTimeout( function () {

alert(“Time to take the cake out of the oven!”);

} , 600000);

1. In an 80s movie, computer animated characters were forced to fight in a computerized blood sport. The ones that died were referred to as derezzed. Name the movie.
   1. I actually don’t understand the question ☹
   2. Maybe I will need to read the chapter better but it is 11:57 PM already so I have to submit.
2. How do closures affect scope? Explain.

Text

Description automatically generated

1. What is an anonymous function? Write a line of JavaScript code that exemplifies an anonymous function.
   1. An anonymous function is a function expression that has no name.
   2. We basically will put the function, inline in the call to the setTimeout built in function.
   3. setTimeout(function() {alert ( “Time to take the cookies out of the oven”); }, 600000 );
2. What is a nested function? Show me an example of a nested function from Chapter 11.
   1. Nested functions are functions defined inside another function.
   2. A nested function local scope, just like other local variables.
   3. To bind the value of a variable in a nested function, use the value that’s defined in the closest enclosing function. If no value is found, then we should look in the global scope.
   4. Example:

Text

Description automatically generated

1. Can JavaScript determine the scope of a variable by reading the structure of code? If so, what is that called?
   1. Yes JavaScript can determine whether the variable is in local or global scope from just the structure of the code.
   2. This is called **Lexical scope.**
2. What is a free variable and why would you use it when coding?
   1. A functions typically has local variables in its code body and it also might **have variables that aren’t defined locally which is called free variables. The name free comes from the fact that within the function body, free variables aren’t bound to any values (not declared locally in the function).** When we have an environment that has a value for each of the  **free variables,**  we say that we’ve closed the function. And when we take the function and the environment together, we say we have a closure.
3. When is a function declaration defined? When is a function expression evaluated?
   1. A function declaration is defined before the rest of the code is evaluated.
   2. A function expression is evaluated at runtime with the rest of the code, and so it not defined until the statement in which it appears is evaluated.

**Chapter 12:**

1. Using this object literal, create a constructor and instantiate this teacher as an instance of the constructor. var teacher = { firstname = “Ian”, lastname = “Lasky”, Degree = “MS”, Department = “BE”, Age = 47 };

Text

Description automatically generated

1. A constructor is a \_\_\_\_\_\_\_\_\_\_\_.
   1. Function that are meant to be used with the new operator. We capitalize the names of constructors by convention.
2. What happens if you forget to use the keyword new with a constructor?
   1. No object is created. This will cause errors in the code that can be difficult to debug.
3. What does a constructor function return?
   1. We can use constructors to return useful objects like date objects, regular expressions and arrays.
4. What is an object literal? Show me an example of an object literal in Chapter 12.
   1. Object literals give you a convenient way to create objects any where in the code but when I need to create lots of objects.
   2. var cadiParams = {

Make=”GM”,

Model: “Cadillac”,

Year:1955,

Color: “tan”

};

1. What would you use if you wanted to know if an object was created by a specific constructor?
   1. To know if an object was created by a specific constructor, use the **instanceof** operator.
2. List 3 builtin constructors that JavaScript comes with.
   1. new Date() – creates a new date representing the current date and time.
   2. new Array() – creates an empty array with length zero
   3. .reverse() – This reverses all the values in the array.
3. What keyword is used in a constructor to allow access to the object being constructed and adds properties to that object?
   1. The  **new** keyword.
4. How are objects customized and initialized in Chapter 12?
   1. Objects are customized by using constructors in chapter 12.
   2. To initialize an object/ constructor:

Text

Description automatically generated

1. When is it better to use a Constructor rather than an object literal?
   1. A constructor is an ideal path when we need to create many similar objects just like what we did in Lab12 with multiple different car models.