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>