VARIABLES:

1. using "var":

```
<html>
<body>
<script language="javascript">
var a = 10;
document.write("Value of a="+a);
{
        var a = 20;  //var doesn't support block scope
        document.write("<br>Inside block value of a:"+a);
}
document.write("<br>Outside block value of a:"+a);
</script>
</body>
</html>
```

Output:

Value of a=10
Inside block value of a:20
Outside block value of a:20

2. using "let":

```
<html>
<body>
<script language="javascript">
let a = 10, b= 20
{
        let a = 30; //let support block scope
        document.write("<br>Inside block value of a:"+a);
        document.write("<br>Inside block value od b:"+b);
        document.write("<br>Sum = "+(a+b));
}
document.write("<br>Outside block value of a:"+a);
document.write("<br>Outside block value of b:"+b);
document.write("<br>Sum = "+(a+b));
</script>
</body>
</html>
```

Output:

Inside block value of a:30
Inside block value od b:20
Sum = 50
Outside block value of a:10
Outside block value of b:20

Sum = 30

3. using "const":

```
<html>
<body>
<script language="javascript">
        const pi = 3.14; //constant variable
        const a = [1,2,3,4,6];
        document.write(a);
        a[0] = 30;
        document.write("<br>"+a);
        //a = "csit";    results in error
</script>
</body>
</html>
```

Output:

1,2,3,4,6

30,2,3,4,6

DATATYPES:

```
<html>
<body>
<script language="javascript">
        var a = 10; //number
        document.write(typeof a);
        var a = 12.45; //number
        document.write("<br>"+typeof a);
        var b = true; //boolean
        document.write("<br>"+typeof b);
        var c = "CSIT"; //string
        document.write("<br>"+typeof c);
        var bi = 23542524634655474676753735 76n; //bigint
        document.write("<br>"+typeof bi);
        var n = null; //null
        document.write("<br>"+typeof n);
        var n1; //undefined
        document.write("<br>"+typeof n1);
        var s1 = Symbol("CSIT"); //symbols
        var s2 = Symbol("CSIT");
        document.write(s1 == s2);
</script>
</body>
</html>
```

Output:

```
number
number
boolean
string
bigint
object
undefined
false
```

FUNCTIONS:

- ***Function without any parameters:***

```
function message()
{
        document.write("function with zero parameters");
}
message(); //function call
```

Output: function with zero parameters

- ***Function with parameters:***

```
function sum(a,b,c)
```

```
{
        document.write("Sum="+(a+b+c));
}
sum(5,19,34.6); //function call
```
Output: Sum = 58.6

- ***Function with default parameters:***

```
<html>
<body>
<script language="javascript">
    function def(a, b=10)
    {
        document.write("<br>value of a:"+a);
        document.write("<br>value of b:"+b);
    }
    def(5); //a=5, b=10
    def(5, 25); //a=5, b=25
</script>
</body>
</html>
```
Output:
```
    value of a:5
    value of b:10
    value of a:5
    value of b:25
```

- ***Function with rest parameters:***

```
<html>
<body>
<script language="javascript">
    function sum(...ar) //... indicates rest parameter array with name ar
    {
        let r=0;
        for(let i=0;i<ar.length;i++)
        {
            r = r+ar[i];
        }
        document.write("<br>SUM="+r);
    }
    sum(5,12.5);
    sum(14,15,17);
    sum(5,7.5,10,20);
</script>
</body>
</html>
```
Output:
```
    SUM=17.5
```

SUM=46
SUM=42.5

ANONYMOUS FUNCTIONS:

```
<html>
<body>
<script language="javascript">
        let p1 = function (x,y) //anonymous function
        {
                return x*y;
        }
        document.write("Product:"+p1(5,8)); //call to anonymous function
</script>
</body>
</html>
```

ARROW FUNCTIONS (or) FAT ARROWS:

1. **With no parameters:**

   Example 1:

   ```
   let h = () => 'Hello CSIT';
   alert(h());
   ```

   Example 2:

   ```
   let a = 4;
   let b = 2;
   let c = () => a + b + 100;
   alert(c());
   ```

   Output: 106

2. **With parameters:**

   Example 1:

   ```
   let square = n => n*n;
   alert(square(5));
   ```

   Output: 25

   Example 2:

   ```
   let sum = (a,b) => a+b;
   alert(sum(10,12.5));
   ```

   Output: 22.5

3. **With multiple statements as function body**

   ```
   let mul = (x,y) =>
   {
      document.write("<br>Value of x:"+x);
      document.write("<br>Value of y:"+y);
      return x*y;
   };
   ```

document(“
Product: “+mul(10,5));

Output:

Value of x: 10

Value of y: 5

Product: 50

4. **With explicit return**

```
let f4 = (a,b) => {
        document.write("a="+a);
        document.write("b="+b);
        return a+b;
}
var res = f4(10,12.6);
document.write("result from fat arrow:"+res);
```

Output:

result from fat arrow: 22.6

ARRAYS:

1. **Creating Arrays and accessing array elements**

```
<html>
<body>
<script language="javascript">
        const a = [10, 20, 12.5, 7, 90];
        document.write(a);

        const b = [];
        b[0] = "apple";
        b[1] = "banana";
        document.write("<br>"+b);

        const c = new Array("volvo","bmw","ferrari");
        document.write("<br>"+c);
        document.write("<br>My Favorite car:"+c[1]);

        for(let i=0;i<c.length;i++)
        {
                document.write("<br>"+c[i]);
        }
        document.write("<br>Using forEach()...<br>");
        //forEach
        c.forEach(myfun);
        function myfun(value,index,arr)
        {
                document.write(index+":"+value);
        }
</script>
</body>
</html>
```

2. **Operations on Arrays:**

```
<html>
<body>
<script language="javascript">
        const fruits = ["banana","orange","pineapple"];
        fruits.push("watermelon");
        document.write("<br>After push:"+fruits);
        fruits.unshift("apple");
        document.write("<br>After unshift:"+fruits);
        document.write("<br>Item popped:"+fruits.pop());
```

```
        document.write("<br>After pop:"+fruits);
        document.write("<br>Item removed:"+fruits.shift());
        document.write("<br>Element found
at:"+fruits.indexOf("pineapple"));
        const a = [10,3,5,6,7];
        const n = a.slice(1,4);
        document.write("<br>Sliced Array:"+n);
        fruits[0] = "pear";
        document.write("<br>After modification:"+fruits);
</script>
</body>
</html>
```

PROMISES:

1.
```
<html>
<body>
<script>
var clean = new Promise(function(resolve, reject)
{
        //cleaning the room
        let done = true;
        if(done==true)
        {
           resolve("Go to play football");
        }
        else
        {
           reject("Do laundry");
        }
});
clean.then(
function(value)
{
        //invoked when promise is fullfilled
        document.write("<h2>"+value+"</h2>");
},
function(error)
{
        //invoked when promise is rejected
        document.write("<h2>"+error+"</h2>");
}
); //consuming code
</script>
</body>
</html>
```

2.
```
<html>
<body>
<script>
var printing = new Promise(function(resolve, reject)
{
        //printer is printing the document
        let result = false;
        if(result == true)
        {
```

```
            resolve("Printed Document");
        }
        else
        {
            reject("Error!");
        }
});
printing.then(
function(value)
{
        document.write("Output:"+value);
        document.write("<br>Put it in file");
});
printing.catch(
function(error)
{
        document.write("Output:"+error);
        document.write("<br>Fix it !!!");

});
</script>
</body>
</html>
```

CALLBACKS:

1.
```
<html>
<body>
<script>
function input(call) //call = greet
{
        var n = prompt("Enter ur name:");
        call(n);  //greet(n)

}
function greet(name) //callback function
{
        document.write("<h1>Welcome:"+name+"</h1>");
}
input(greet);
</script>
</body>
</html>
```

2. <html>

```
<body>
<script>
setInterval(display,2000);
function display() //callback
{
        d  = new Date();
        var t = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
        var h = document.getElementById("i1");
        h.innerHTML = t;
}
</script>
<h1 id="i1"> Initially </h1>
</body>
</html>
```