P1: Learn and Apply ECMASCRIPT Concept

- **let, var and const**
  - **Theory:**
    - There are 3 ways to declare a JavaScript variable:
      -Using var
      -Using let
      -Using const
    - The let keyword was introduced in ES6 (2015).
    - Variables defined with let cannot be Redeclared.
    - Variables defined with let must be Declared before use.
    - Variables defined with let have Block Scope.
    - The const keyword was introduced in ES6 (2015).
    - Variables defined with const cannot be Redeclared.
    - Variables defined with const cannot be Reassigned.
    - Variables defined with const have Block Scope.
    - Arrow functions were introduced in ES6.
    - Arrow functions allow us to write shorter function syntax.

  - **Code practice:**

```
//-------------------------------------------------------------------var practice
var x = 5;
var y = 6;
var z = x + y;
console.log(z);

//-----------------------------------------------------------------let practice
let x = "Yagnik Desai";
let x = 0;
console.log(x);

var x = "John Doe";
var x = 0;
console.log(x);

var x = 10;
// Here x is 10

{
```

```
   var x = 2;
   // Here x is 2
}
console.log(x);

// Here x is 2

let x = 10;
// Here x is 10

{
   let x = 2;
   console.log(x);
   // Here x is 2
}
console.log(x);

// Here x is 10


//-----------------------------------------------------------------const practice

const x = 10;
// Here x is 10
console.log(x);
{
   const x = 2;
   // Here x is 2

   console.log(x);
}

console.log(x);
// Here x is 10
```

- **Output:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.


PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
11
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js:9
let x = 0;
    ^

SyntaxError: Identifier 'x' has already been declared
    at wrapSafe (internal/modules/cjs/loader.js:1001:16)
    at Module._compile (internal/modules/cjs/loader.js:1049:27)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)
    at Module.load (internal/modules/cjs/loader.js:950:32)
    at Function.Module._load (internal/modules/cjs/loader.js:790:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:76:12)
    at internal/main/run_main_module.js:17:47
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
0
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
2
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
2
10
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
10
2
10
PS C:\Users\Yagnik Desai> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\es6.js"
Yagnik 19CE019
```

- **Destructuring**
  - **Theory:**
    Destructuring Assignment is a JavaScript expression that allows to unpack values from arrays, or properties from objects, into distinct variables data can be extracted from arrays, objects, nested objects and assigning to variables. In Destructuring Assignment on the left-hand side defined that which value should be unpacked from the sourced variable.
  - **Code:**

```
//--------------------------------------------------destructuring assignment
var names = ["alpha", "beta", "gamma", "delta"];

var firstName = names[0];
var secondName = names[1];

console.log(firstName);//"alpha"
console.log(secondName);//"beta"
//--------------------------------------------------array destructuring
var names = ["alpha", "beta", "gamma", "delta"];
var [firstName, secondName] = names;

console.log(firstName);//"alpha"
console.log(secondName);//"beta"

//Both of the procedure are same
var [firstName, secondName] = ["alpha", "beta", "gamma", "delta"];

console.log(firstName);//"alpha"
console.log(secondName);//"beta
//--------------------------------------------------object destructuring
var marks = { x: 21, y: -34, z: 47 };

var x = marks.x; // x = 21
var y = marks.y; // y = -34
var z = marks.z; // z = 47

console.log(x);
console.log(y);
console.log(z);
var z = "Yagnik 19CE019";
console.log(z);
```

- **Output:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\P1.2.js"
alpha
beta
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\P1.2.js"
alpha
beta
alpha
beta
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\P1.2.js"
21
-34
47
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\ECMASCRIPT_Demo\P1.2.js"
Yagnik 19CE019
PS D:\Acadamic\Sem 5\AWT\Assignments>
```

- **map, filter and reduce**
  - **Theory:**
    - The map() method creates a new array with the results of calling a function for every array element.
    - The map() method calls the provided function once for each element in an array, in order.
    - map() does not execute the function for empty elements.
    - map() does not change the original array.
    - The filter() method creates an array filled with all array elements that pass a test (provided by a function).
    - filter() does not execute the function for empty array elements.
    - filter() does not change the original array.
    - The reduce() method executes a reducer function for each value of an array.
    - reduce() returns a single value which is the function's accumulated result.
    - reduce() does not execute the function for empty array elements.
    - reduce() does not change the original array.

  - **Code practice:**

```
//-----------------------------------------------map
const persons = [
   {firstname : "Malcom", lastname: "Reynolds"},
   {firstname : "Kaylee", lastname: "Frye"},
   {firstname : "Jayne", lastname: "Cobb"}
  ];

  var x = persons.map(getFullName);
  console.log(x);
  function getFullName(item) {
    return [item.firstname,item.lastname].join(" ");
  }
//-----------------------------------------------filter
const ages = [32, 33, 16, 40];

var x = ages.filter(checkAdult)    // Returns [32, 33, 40]
console.log(x);
function checkAdult(age) {
  return age >= 18;
}

//-----------------------------------------------reduce
```

```
const numbers = [15.5, 2.3, 1.1, 4.7];
var x = numbers.reduce(getSum, 0);
console.log(x);
function getSum(total, num) {
  return total + Math.round(num);
}
var x="Yagnik 19CE019";
console.log(x);
```

- **Output:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.3.js"
[ 'Malcom Reynolds', 'Kaylee Frye', 'Jayne Cobb' ]
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.3.js"
[ 32, 33, 40 ]
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.3.js"
24
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.3.js"
Yagnik 19CE019
PS D:\Acadamic\Sem 5\AWT\Assignments>
```

- **callback, promises and async/wait**
  - **Theory:**
    - A callback is a function passed as an argument to another function. This technique allows a function to call another function. A callback function can run after another function has finished.
    - "Producing code" is code that can take some time. "Consuming code" is code that must wait for the result. A Promise is a JavaScript object that links producing code and consuming code.
    - async makes a function return a Promise
    - await makes a function wait for a Promise.
  - **Code practice:**

```
//-----------------------------------------------------callback
function myDisplayer(some) {
   var x = some;
   console.log(x);
}

function myCalculator(num1, num2, myCallback) {
   let sum = num1 + num2;
   myCallback(sum);
}

myCalculator(5, 5, myDisplayer);

//-----------------------------------------------------promises
function myDisplayer(some) {
   var x = some;
   console.log(x);
}

let myPromise = new Promise(function (myResolve, myReject) {
   let x = 0;

   // The producing code (this may take some time)

   if (x == 0) {
      myResolve("OK");
   } else {
      myReject("Error");
   }
```

```
});

myPromise.then(
   function (value) { myDisplayer(value); },
   function (error) { myDisplayer(error); }
);

//-----------------------------------------------------Async/Await

async function myDisplay() {
   let myPromise = new Promise(function (myResolve, myReject) {
      setTimeout(function () { myResolve("I love You !!"); }, 3000);
   });
   var x = await myPromise;
   console.log(x);
}

myDisplay();
var x = "Yagnik 19CE019";
console.log(x);
```

- **Output:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.4.js"
10
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.4.js"
OK
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.4.js"
I love You !!
PS D:\Acadamic\Sem 5\AWT\Assignments> node "d:\Acadamic\Sem 5\AWT\Assignments\P1.4.js"
Yagnik 19CE019
PS D:\Acadamic\Sem 5\AWT\Assignments>
```