JavaScript –ES6

Day 2

Agenda

- New ES6 syntax
- Arrow Functions
- Destructuring
- ES6 Modules
- ES6 Classes
- Promises
- ES6 collections

Agenda

- Array extensions
- Object extensions
- String extensions

Day 1 -Recap

- Let ,var, const
- Default Parameter Values
- Function Rest Parameter
- Spread operator
- for…of Loop in ES6
- Template Literals

day 2

- Arrow Functions
- Destructuring
- classes

Arrow function

- ES6 arrow functions provide you with an alternative way to write a shorter syntax compared to the function expression.

- The following example defines a function expression that returns the sum of two numbers:

```
let add = function (x, y) {
    return x + y;
};
console.log(add(10, 20)); // 30
```

- The following example is equivalent to the above add() function expression but use an arrow function instead:

```
let add = (x, y) => x + y;
console.log(add(10, 20)); // 30;
```

- Use the (...args) => expression; to define an arrow function.
- Use the (...args) => { statements } to define an arrow function that has multiple statements.

ES6 Destructuring-Array

- ES6 provides a new feature called destructing assignment that allows you to destructure properties of an object or elements of an array into individual variables.
- Assuming that you have a function that returns an array of numbers as follows:

```
function getScores() {
  return [70, 80, 90];
}
let scores = getScores();
let x = scores[0],
   y = scores[1],
   z = scores[2];
```

- Prior to ES6, there was no direct way to assign the elements of the returned array to multiple variables such as x, y and z

```
let [x, y, z] = getScores();
console.log(x); // 70
console.log(y); // 80
console.log(z); // 90
```

- The variables x, y and z will take the values of the first, second, and third elements of the returned array.

- Note that the square brackets [] look like the array syntax but they are not.

- If the getScores() function returns an array of two elements, the third variable will be undefined

ES6 Destructuring-Object

- Suppose you have a person object with two properties: firstName and lastName.

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

- Prior to ES6, when you want to assign properties of the person object to variables, you typically do it like this:

```
let firstName = person.firstName;
let lastName = person.lastName;
```

- ES6 introduces the object destructuring syntax that provides an alternative way to assign properties of an object to variables:

let { firstName: fname, lastName: lname } = person;

In this example, the firstName and lastName properties are assigned to the fName and lName variables respectively.

Syntax:

let { property1: variable1, property2: variable2 } = object;

Classes

- A JavaScript class is a blueprint for creating objects. A class encapsulates data and functions that manipulate data
- Use the keyword class to create a class.
- Always add a method named constructor()

Syntax

```
class ClassName {
  constructor() { ... }
}
```

example

- class Car {
  constructor(name, year) {
  this.name = name;
  this.year = year;
  }
  }
- Using classes

```
let myCar1 = new Car("Ford", 2014);
let myCar2 = new Car("Audi", 2019);
```

The Constructor Method

- The constructor method is a special method:

It has to have the exact name "constructor"

It is executed automatically when a new object is created

It is used to initialize object properties

Inheritance example

```
class human{
        constructor(legs){
           this.legs=legs;
        }
    }
    class Person extends human {
        constructor(firstname,lastname,age,leg){
           super(leg);
           this.fname=firstname;
           this.lname=lastname
           this.age=age
        }
        display=()=>{console.log('in display ');}
        displaylanme=()=>{console.log('lname');}
    }
p1=new Person('abdullah','shaikha',10,4);
console.log(p1.legs);
```