JavaScript –ES6

Day 1

Agenda

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

Agenda

- Array extensions
- Object extensions
- String extensions

Introduction

- ECMAScript 2015 or ES2015 is a significant update to the JavaScript programming language. It is the first major update to the language since ES5 which was standardized in 2009. Therefore, ES2015 is often called ES6.

ES versions

Edition	Official name	Date published
ES11	ES2020	In progress
ES10	ES2019	Summer 2019
ES9	ES2018	June 2018
ES8	ES2017	June 2017
ES7	ES2016	June 2016
ES6	ES2015	June 2015
ES5.1	ES5.1	June 2011
ES5	ES5	December 2009
ES4	ES4	Abandoned
ES3	ES3	December 1999
ES2	ES2	June 1998
ES1	ES1	June 1997

Var keyword

- The scope of the variable is either global or local. If you declare a variable outside of a function, the scope of the variable is global. When you declare a variable inside a function, the scope of the variable is local.
- When you declare a global variable using the var keyword, you add that variable to the property list of the global object. In the case of the web browser, the global object is the window
- var a = 10;console.log(window.a); // 10

let keyword

- ES6 provides a new way of declaring a variable by using the let keyword.
- The let keyword is similar to the var keyword, except that these variables are blocked-scope
- In JavaScript, blocks are denoted by curly braces {} ,

for example, the if else, for, do while, while, try catch and so on

- Variables are declared using the let keyword are block-scoped, and are not attached to the global object
- Redeclaring a variable using the let keyword will cause an error

const keyword

- ES6 provides a new way of declaring a constant by using the const keyword. The const keyword creates a read-only reference to a value.
- Like the let keyword, the const keyword declares blocked-scope variables. However, the block-scoped variables declared by the const keyword can’t be reassigned
- const x=20 //this value cannot be changed

Default Parameter Values

- ES6 allows function parameters to have default values.
- function add(x, y = 10) {

```
 // y is 10 if not passed or undefined
 return x + y;
}
```

Function Rest Parameter

- The rest parameter (...) allows a function to treat an indefinite number of arguments as an array.
- ES6 provides a new kind of parameter so-called rest parameter that has a prefix of three dots (...). A rest parameter allows you to represent an indefinite number of arguments as an array.
- See the following syntax:

```
function fn(...args) {
   //function statement }
```

The last parameter (args) is prefixed with the three-dots (...). It’s called a rest parameter (...args).

Spread operator

- ES6 provides a new operator called spread operator that consists of three dots (...)
- The spread operator allows you to spread out elements of an iterable object such as an array,a map, or a set.
- example:let odd = [1,3,5];

```
let combined = [2,4,6, ...odd];
console.log(combined);
```

for…of Loop in ES6

- ES6 introduced a new construct for...of that creates a loop that iterates over iterable objects such as: Array, String, Map, Set, …

- for (variable of iterable) {

  // statements

}

- In each iteration, a property of the iterable object is assigned to the variable. You can use var, let, or const to declare the variable.
- The iterable is an object whose iterable properties are iterated.

For of example

- <script>
  let p=["junaid","sam","sonu","abdullah"];
  for(let x of p){
  console.log(x);
  }
  </script>

Output in console Junaid
Sam
Sonu
abdullah

Template Literals

- Template literals that allow you to work with a string template easier.
- Prior to ES6, you use single quotes (‘) or double quotes (“) to wrap a string literal. And the strings have very limited functionality.
- To enable you to solve more complex problems, ES6 template literals provide the syntax that allows you to work with strings in a safer and cleaner way.
- In ES6, you create a template literal by wrapping your text in backticks (`) as follows:
- let simple = `This is a template literal`;

and you get the following features:

A multiline string: a string that can span multiple lines.

String formatting: the ability to substitute part of the string for the values of variables or expressions. This feature is also called string interpolation.

HTML escaping: the ability to transform a string so that it is safe to include in HTML.

Template Literal example

- <script>
  let x="hi this is junaid double quotes";
  let y='hi this is junaid single quotes';
  let z=`hi this is junaid back tick`;
  //multiline can be used
  let p =`This text
  can
  span multiple lines`;
  console.log(x);
  console.log(y);
  console.log(z);
  console.log(p);
</script>

Template literal String function

```
28
29 // es6 strings.METHODS
30
31 console.log(`${firstName}`.startsWith('V'));
32 console.log(`${firstName}`.endsWith('d'));
33 console.log(`${firstName}`.includes('ino') );
34
35 console.log(`${firstName}${lastName} `.repeat(10));
36
```