JS Versions

Javascript ES6

- JavaScript was invented in 1995, and became an ECMA standard In 1997. ECMAScript is the standardization of Javascript.

- ECMAScript is the official name of the language.

- ECMAScript versions have been abbreviated to ES1, ES2, ES3, ES5, and ES6.

Key Features

ES6 introduced several key features like

- const,
- let
- arrow functions
- template literals
- default parameters

const

The **const** keyword is mainly used to store that variable who's value is not going to be changed. In javascript, const is considered to be more powerful than var. Once used to store a variable it can't be reassigned. In simple words, it is an *immutable* variable except when used with objects.

Example:

```
const name = 'Mukul';
document.write(name); // Will print 'Mukul' to the console.

// Trying to reassign a const variable
name = 'Rahul';
document.write(name); // Will give TypeError
```

let

In case of let keyword, the variables declared will be mutable i.e their values can be changed. It works similar to the var keyword with some key differences like scoping which makes it a better option as when compared to var.

Example 1:

```
let name = 'Mukul';
document.write(name); // Mukul

name = 'Rahul';
document.write(name); // Rahul
```

Example 2

```
<html>  <body>
<h2>Redeclaring a Variable Using let</h2>
<p id="demo"></p>
<script>
let  x = 10;
// Here x is 10
{    let x = 2;
  // Here x is 2 }
// Here x is 10
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

The `let` keyword allows you to declare a variable with block scope.

arrow functions

- Introduced in ES6, arrow functions are definitely one of the most impactful changes in javascript.
- These function expressions makes your code more readable, more modern.
- Arrow functions allows a short syntax for writing function expressions.
- You don't need the function keyword, the return keyword, and the curly brackets.
- Using const is safer than using var, because a function expression is always a constant value.

Example

```
// ES5
var x = function(x, y) {
return x * y;
}

// ES6
const x = (x, y) => x * y;
```

```
<script>
const x = (x, y) => x * y;
document.getElementById("demo").innerHTML = x(5, 5);
</script>
```

Template literals

- Template literals are a feature of ES6 which allows us to work with strings in a better way
- Before ES6 we made use of '+' operator whenever we wanted to concatenate strings
- Template literals are easier to read and write than traditional string concatenation, especially for long strings or strings with multiple embedded expressions.

```
// ES5
var name = 'Mukul Latiyan';
document.write('My name is '+ name);

// ES6
const name1 = 'Mukul Latiyan';
document.write(`My name is ${name1}`);
```

Default Parameter Values

- ES6 allows function parameters to have default values.

```
function myFunction(x, y = 10) {
// y is 10 if not passed or undefined
return x + y;
}
myFunction(5); // will return 15
```