**ECMAScript 2015 Goodies – Part 1 – let keyword**

ECMAScript 2015, formerly known as ECMAScript 6 is finally here. TC39 committee has finally approved the specification which is available [here](http://www.ecma-international.org/ecma-262/6.0/) . The new standard brings many useful features to our beloved language – JavaScript and its good time to start a new series to understand these features one by one.

Let’s start this series with one of the simplest but most useful feature to the language, which is the **let** keyword. The let statement allows you to declare a block scope local variable, unlike **var** which defines a variable globally. Additionally variable declared using var hoists, however that’s not the case when declared using let keyword. Variable scoping and hoisting has always troubled a lot to almost every JavaScript developer [definitely including me], but now we have a solution to address these problems.

We will look into different code snippets to understand <code>let</code> keyword and its usefulness. Let’s start with the declaration first.

Below code snippet defines a variable c and its available within if block only. Unlike variable d, which is declared using var keyword, variable c is not accessible outside if block.

let a = 10;

let b = 20;

if (a < b) {

let c = 30;

var d = 40;

}

console.log(c); //error

console.log(d);

You can also declare a variable and scope it to a particular loop, for in this case as shown below

let numbers = [1, 2, 3, 4, 5];

for (let i = 0; i < numbers.length; i++) {

console.log(numbers[i]);

};

// console.log(i) //error

Variables declared using let in a given scope does not override the variable declared with the same name outside of that scope.

function letTest(){

let x = 30;

if(true){

let x = 40; // different variable

console.log(x); //40

}

console.log(x); //30

}

Unlike var, you cannot declare multiple variables with same name using let keyword

function duplicateTest() {

let x = 10;

let x = 20; //error

}

Finally, as mentioned earlier variables declared using let does not hoist in a given scope

function hoistTest() {

console.log(foo); // ReferenceError

let foo = 2;

}

That’s all I have for now on let keyword. Hope this article helped you to understand let keyword and its usefulness. We will continue to explore other features in future articles, till the time write clean JavaScript code!