**IIFE (Immediately Invoked Function Expression):**

* Using this we can create function in which we can declare variables and this variables will not be accessible outside the function so variables remain private (because of scope)

Syntax:

* With the parentheses the JavaScript will consider this as an expression and not a statement

(function (parameters) //anonymous function

{ //code

}

)

(parameters); //function calling

**Closures:**

* An inner function has always access to the variables and parameters of its outer function even after the outer function has returned.
* Even after the execution context is removed the variable object of that function is still there in the memory and accessible from inner functions (in the scope chain and execution stack)

**ES6:**

Let and const variable declarations:

In ES5:

* Variables in ES5 are **function scope**
* Variables declared using var in a function inside a conditional statement is accessible anywhere in that function and not outside the function
* Without declaring a variable we can use it

In ES6:

* **let**: mutable variables
* **const**: immutable variables
* Variables in ES5 are **block scope** (A block is simply a code which is wrapped in the curly brackets)
* Variables declared using let or const in a function inside a conditional statement are not accessible anywhere in the function but only inside the statement and outside of the block
* **Temporal dead zone**: Without declaring a variable we cannot use it
* **Template literals (``):** While writing text inside this we can write variables just like

console.log(`This is ${firstName} ${lastName}`); using ${variablename} for getting values of variables