Suppose you have main.js and index.js scripts placed in index.html

main.js

var thing = {'thing':'main};

console.log(''main', thing)

index.js

var thing = {'thing':'others};

console.log(''others', thing)

To stop leaking varaibles to public. IIFE resolves this issue

index.js

(function(){

    var thing = {'thing':'others};

console.log(''others', thing)

})()

main.js

(function(){

    var thing = {'thing':'main};

console.log(''main', thing)

})()

this will help not to leak thing varaible to global vaiables.

Closure is formed when foo[0]() is called .

var foo = [];

for (let i = 0; i < 10; i++) {

    foo[i] = function() {

        return i

    }

}

console.log(foo[0]());

console.log(foo[1]());

console.log(foo[2]());

 It prints 10 10 10 - Closure points to actual value of i in the outer scope

 var foo = [];

To avoid above ,

var foo = [];

for (var i = 0; i < 10; i++) {

(function(y){

foo[y] = function() { return y}  - Its pass by value hence 1 2 3

})(i)

}

console.log(foo[0]());

console.log(foo[1]());

console.log(foo[2]());

Does Javascript pass variables by refernce or pass by value

Primitive are passed by values

Var a = 1;

Function foo (a){

a= 2

}  
  
console.log(a) – it prints 1 only

Objects are passed by refernce

var a ={}

function foo(a) {  
a.moo = false  
}  
  
foo(a);

console.log(a);

if object property is changed then pass by reference otherwise its pass by value