Date:2021-01-16

* First class function returning a function:
  + - Create a main function and based on some condition we return a function.
    - In case of main function contain an argument as well as subfunction also contain a argument then we can call it like….
      * + Main\_function(argument\_of\_main)(argument\_of\_sub\_Function);
* Immediately invoked function expression (IIEE):
  + - IIFE are generally used for a data privacy.
    - If we want to hide a particular variable for not access outside function we can use it.
    - Ex:
      * + (function(){

//code

})();

Here we create a anonymous function and call it in one line. A variable which is in the function are not accessible outside the function.

* Closures:
  + - An inner function always access to the variables and parameter to its outer function, even after the function returned.
    - We are not create closures manually. Its automatically create by JS.
    - See the practical in 16th day folder.
* Bind, call and Apply:
  + - call(var1,,,,,,)-contain one compulsory argument “this” and other are as per function argument.
    - apply(var1,[]) – same like a call but function argument are must be in array.
    - bind() – this is used to save a copy of function for future use…and also allow to pre set argument.
    - Ex:

var jay = {

name : 'jay',

age :21,

job :'student',

presentation :function(style,timeofday){

if(style=='formal')

{

console.log("good "+timeofday+" Ladies and gentlemen ! i am "+this.name+" and i am a "+this.age+" years old "+this.job);

}

else if(style=='frirndly')

{

console.log("hi gays !!!good "+timeofday+" i am "+this.name+" and i am a "+this.age+" years old "+this.job);

}

}

}

jay.presentation('formal','morning');

var jimmy={

name : 'jimmy',

age :22,

job:'designer'

}

//use of call()

jay.presentation.call(jimmy,'frirndly','evening');

//use of apply

jay.presentation.apply(jimmy,['formal','night']);

//use of apply

a = jay.presentation.bind(jimmy);

a('formal','evening');

b=jay.presentation.bind(jay,'frirndly');

b('evening');

* Variable declaration using let and const:
  + - A variable declared using const we can’t change value of it. we can assign it only once.
    - A variable declared with const it is a function scope.
    - But variable declared with let it is a block scope only.
    - Ex:

//var is used in function scope

//let is used in block scope

//const is assigned only once

function abc()

{

var a='this is accessible in abd() and all inner function of abc';

const con = 11;

let c;

console.log(a);

subabc();

function subabc()

{

c=21;

let b='this is accessible only in a subabc() even we cannot use it in inner function of subabc()';

console.log(b+c);

}

console.log(c);

}

abc();

* Blocks and IIFEs:
  + - About blocks.
    - Where to variables are accessible or not.
    - Diff bet. Var, const, and let.
* Strings in ES6 ES2015:
  + - Use of (``) write in between and we can use a $variable name foe access a value.
    - Some methods….
      * Objname.startsWith(‘’); return true or false
      * Objname.endsWith(‘’); return true or false
      * Objname.includes(‘’); check string is in or not..
      * Objname.repeat(how many time you want to repeat)
      * Ex:

//some new string function in ES2015

let firstName = 'jay';

let lastName = 'piparava'

//new way to concate a string.....

let fullName = `${firstName} ${lastName}`;

console.log(fullName); //jay piparava

console.log(fullName.startsWith('j')); //true

console.log(fullName.startsWith('a')); //false

console.log(fullName.endsWith('j')); //false

console.log(fullName.endsWith('a')); //true

console.log(fullName.includes('pip')); //true

console.log(fullName.repeat(3)); //jay piparavajay piparavajay piparava