# ES6 = ECMA Script 6

Strict Mode

Use JS BIn

//Classes in JAvascript with inheritance

class Human{

gender = "Male";

printGender=()=>{

console.log(this.gender);

}

}

class Person extends Human{

name="Max";

gender="Female";

printmyName=()=>{

console.log(this.name)

}

}

const person = new Person();

person.printmyName();

person.printGender();

//Spread

const number=[1,2,3];

const newnumber=[...number,4];

console.log(newnumber)

// Spread Object

const person={

name:'Max'

}

const newPerson = {

...person,

age:28

}

console.log(newPerson);

//Rest

const filter=(...args)=>{

return args.filter(el=>el===1);

}

console.log(filter(1,2,3))

//Destructuring

const numbers = [1,2,3];

[num1,num2]=numbers;

console.log(num1,num2);

//Primitive Type

const numbers = 1;

const num1=numbers;

console.log(num1);

//Reference Type

const person = {

name:'max'

};

const secPerson = person;

person.name='tink'

console.log(secPerson)

//Copying of Object

const person = {

name:'max'

};

const secPerson = {

...person

};

person.name='tink'

console.log(secPerson)

//Update String Literals

let name="Laurence";

let message =`${name} was here today`;

console.log(message);

name="John"

message =`${name} was here today`;

console.log(message);

////////////////////////

function updateMessage(p){

let message =`${p} was here`;

console.log(message);

}

updateMessage("John");

//New With Functions

const updateMessage=(p)=>{

let message =`${p} was here`;

console.log(message);

}

updateMessage("John");

//When No Value passed so setting default value

const updateMessage=(p="No Value")=>{

let message =`${p} was here`;

console.log(message);

}

updateMessage();

//arguments

const updateMessage=(p="No Value",b=

show(message,"blank","new"))=>{

let message =`${p} was here`;

console.log(message);

}

function show(){

console.log(arguments);

}

updateMessage("John");

//Rest

function myFunction(...p){

console.log(p);

}

myFunction(1,2,3)

myFunction(2,4,55,3444)

//Looping through Rest

function myFunction(...p){

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

{

console.log(p[i])

}

console.log(p);

}

myFunction(1,2,3)

//Looping Using Argument Object

function myFunction(){

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

{

console.log(arguments[i])

}

console.log(arguments);

}

myFunction(1,2,3)

//Function Fat Arrow

let mes0 = (value)=>{

console.log(value);

}

mes0('test1')

let mes1 = value=> "Hello " + value

console.log(mes1('test1'));

let mes2 = value=> {

console.log ("Hello " + value);

}

mes2('test2');

//Destructing

let person ={

fname:"priya",

lname:"patra"

}

console.log(person);

let {fname,lname} = person;

console.log(fname);

console.log(lname);

let myArry=["js","java","sd"];

//console.log(myArry);

let [a, b, c, d = "edfe"]=myArry;

console.log(a);

console.log(b);

console.log(c);

console.log(d);

//Generators and Iterators

Use in javascript mode

"use strict";

function\* pggen()

{

let counter =0;

while(counter<10){

yield counter++;

}

}

let pg =pggen();

for(let i=0;i<10;i++){

let holder= pg.next();

if(holder.done){

break;

}

console.log(holder.done);

console.log(holder.value)

}

console.log(pg);

//Datastructure

let testSet =Object.create(null);

testSet.id=0;

testSet[10]="Hello World"

if(testSet.id){

console.log("its there");

}

else{

console.log("not there")

}

console.log(testSet['10'])

let a= null;

let b= null;

testSet[a]="Whatever";

console.log(testSet)

console.log(testSet[b])

let testSet2 = new Set();

testSet2.add("hello world")

testSet2.add("hello world1")

testSet2.add("hello world2")

console.log(testSet2)

testSet2.forEach(function (value,key){

console.log(key+" "+value)

})

//Removing Duplicate

let myArr=[1,1,22,22,33,44,,3,22,11,11,2,2,3,4,44,44,4];

function removeDuplicate(a){

return[...new Set(a)];

}

console.log(removeDuplicate(myArr))

//Maps And DataStructure

//Map

let map =myArr.map(function(a){

console.log(a\*10);

})

//Map

let map =myArr.map(function(a){

console.log(a\*10);

})

let myMap =new Map();

myMap.set("Hello","World");

myMap.set("Hello1","World1");

console.log(myMap["Hello"]);//undefined

console.log(myMap.get("Hello1"));

console.log(myMap.has("Hello"));// Checks only the key not the value