In [1]:
```
var person = {
    firstName:"Yash",
    lastName:"Hardiya",
}

var person2 = person

person2.firstName = "priyanka"
person2.lastName = "Mishra"

console.log(person, person2)



//spread operator



//object literal


//Number, string, boolean, undefined, null, symbol
//Object => store => reference
```

```
{ firstName: 'priyanka', lastName: 'Mishra' } { firstName: 'priyanka', lastName: 'Mishra' }
```

In [2]:
```
var person = {
    firstName:"Yash",
    lastName:"Hardiya",
}


//...  => ES6
var person2 = {...person}

person2.firstName = "priyanka"
person2.lastName = "Mishra"

console.log(person, person2)
```

```
{ firstName: 'Yash', lastName: 'Hardiya' } { firstName: 'priyanka', lastName: 'Mishra' }
```

In [3]:
```
//befor spread operator

var person = {
    firstName:"Yash",
    lastName:"Hardiya",
    getFullName : function(){

    }
}


var person2 = Object.assign({}, person)


person2.firstName = "priyanka"
person2.lastName = "Mishra"


console.log(person, person2)
```

```
{ firstName: 'Yash', lastName: 'Hardiya' } { firstName: 'priyanka', lastName: 'Mishra' }
```

In [4]:
```
//this keyword. =>  context represent


console.log(this)
```

```
Object [global] {
  global: [Circular],
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Function]
  },
  queueMicrotask: [Function: queueMicrotask],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Function]
  },
  __filename: '[eval]',
  exports: {},
  module: Module {
    id: '[eval]',
    path: '.',
    exports: {},
    parent: undefined,
    filename: '/Users/boby/[eval]',
    loaded: false,
    children: [],
    paths: [
      '/Users/boby/node_modules',
      '/Users/node_modules',
      '/node_modules'
    ]
  },
  __dirname: '.',
  require: [Function: require] {
    resolve: [Function: resolve] { paths: [Function: paths] },
    main: undefined,
    extensions: [Object: null prototype] {
      '.js': [Function],
      '.json': [Function],
      '.node': [Function]
    },
    cache: [Object: null prototype] {}
  },
  '$$mimer$$': [Function: defaultMimer],
  '$$done$$': [Function: bound bound done],
  person: { firstName: 'Yash', lastName: 'Hardiya' },
  person2: { firstName: 'priyanka', lastName: 'Mishra' },
  console: Console {
    log: [Function: log],
    warn: [Function: warn],
    dir: [Function: dir],
    time: [Function: time],
    timeEnd: [Function: timeEnd],
    timeLog: [Function: timeLog],
    trace: [Function: trace],
    assert: [Function: assert],
    clear: [Function: clear],
    count: [Function: count],
    countReset: [Function: countReset],
    group: [Function: group],
    groupEnd: [Function: groupEnd],
    table: [Function: table],
    debug: [Function: debug],
    info: [Function: info],
    dirxml: [Function: dirxml],
    error: [Function: error],
    groupCollapsed: [Function: groupCollapsed],
    Console: [Function: Console]
  },
  '$$': [Object: null prototype] {
    async: [Function: bound async],
    done: [Function: bound done],
    sendResult: [Function: bound ],
    sendError: [Function: bound ],
    mime: [Function: bound ],
    text: [Function: bound ],
    html: [Function: bound ],
    svg: [Function: bound ],
    png: [Function: bound ],
    jpeg: [Function: bound ],
    json: [Function: bound ],
    input: [Function: bound input],
    display: [Function: bound createDisplay],
    clear: [Function: bound clear]
  }
}
```

In [44]:
```
//getter setters for object

var obj = {
   name:"suyash",

   get userName(){

       return this.name.toUpperCase()
   },

    set userName(name){
        this.name = name
    },

    get name(){
       return ""
    }

}

console.log(obj.userName)

obj.age = 24
console.log("age is ", obj.age)



obj.userName = "mohit"
console.log(obj.userName)

console.log(obj.name)
```

```
age is  24
```

In [3]:
```
class Student2{
    #name
    constructor(){
        this.name = "pankaj"
    }

    get name(){
    return ""
}


}


var std = new Student("Pankaj")
// console.log(std.name)


class Lib extends Student2{
    constructor(){
        super()
    }
}


var lib = new Lib()
console.log(lib.name)
```

```
evalmachine.<anonymous>:1
class Student2{
^

SyntaxError: Identifier 'Student2' has already been declared
    at evalmachine.<anonymous>:1:1
    at Script.runInThisContext (vm.js:120:18)
    at Object.runInThisContext (vm.js:309:38)
    at run ([eval]:1054:15)
    at onRunRequest ([eval]:888:18)
    at onMessage ([eval]:848:13)
    at process.emit (events.js:314:20)
    at emit (internal/child_process.js:876:12)
    at processTicksAndRejections (internal/process/task_queues.js:85:21)
```

In [28]:
```
obj = {
    firstName:"pankaj",
    lastName:"sharma"
}

Object.freeze(obj)

obj.firstName = "suyash"

var keyPair = [ [ 'firstName', 'pankaj' ], [ 'lastName', 'sharma' ] ]

console.log(Object.keys(obj))
console.log(Object.values(obj))
console.log(Object.entries(obj))

var newObj = Object.fromEntries(keyPair)
console.log(newObj)
```

```
[ 'firstName', 'lastName' ]
[ 'pankaj', 'sharma' ]
[ [ 'firstName', 'pankaj' ], [ 'lastName', 'sharma' ] ]
{ firstName: 'pankaj', lastName: 'sharma' }
```

In [43]:
```
obj = {
    firstName:"pankaj",
    lastName:"sharma"
}

Object.seal(obj)
Object.freeze(obj)

delete obj['firstName']


console.log("object is sealed : ", Object.isSealed(obj))
console.log("object is sealed : ", Object.isFrozen(obj))


console.log("firstName" in obj)
console.log(obj.hasOwnProperty("firstName"))


var obj1 = {
    name:"Arpita",
    age:24
}

var obj2 = {
    jd:"full stack developer",
}

var objFull = Object.assign(obj1, obj2)
console.log(objFull)

var objFull = {...obj1, ...obj2}
console.log(objFull)
```

```
object is sealed :  true
object is sealed :  true
true
true
{ name: 'Arpita', age: 24, jd: 'full stack developer' }
{ name: 'Arpita', age: 24, jd: 'full stack developer' }
```

In [45]:
```
var array = ['a', 'b', 'c', 'd']

var ind = -1
var search = 'c'

for(var i=0; i<array.length; i++){
    if(array[i] == search){
        ind = i
        break;
    }
}

console.log(ind)
```

```
2
```

In [ ]:
```
function getNumArray(){
    return [1, 2, 3]
}


getArray().map(function(el ){
    return el*el
}).filter(el=>el%2==0)




pipe(
    getArray,
    map

)
```

In [ ]:
```
1.      Check if a key is present in every segment of size k in an array.
2.      Find the minimum and maximum element in an array
3.      Find the Kth largest and Kth smallest number in an array
4.      Write a program to cyclically rotate an array by one
5.      Count Pairs with given sum
6.      Find duplicates in an array
7.      Find the first repeating element in an array of integers
8.      Find the first non-repeating element in a given array of integers
9.      Find if there is any subarray with sum equal to zero
10.     Find Largest sum contiguous Subarray of given size
```