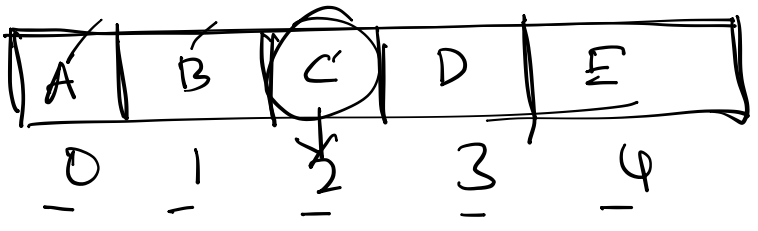
Non primitive Data types

① Array
② Object
③ Functions

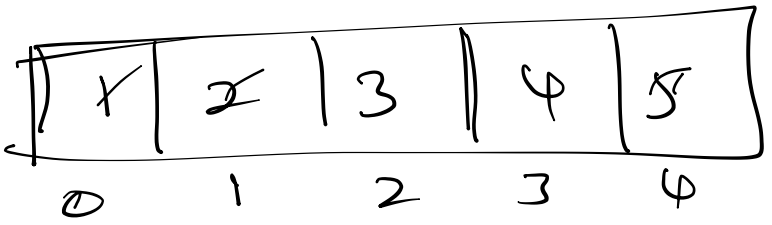
→ stored as a reference

---

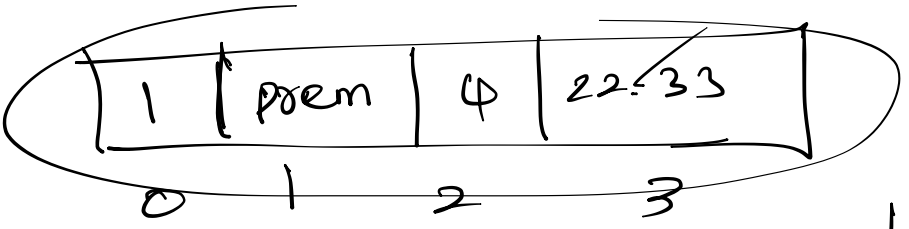
Array → collection of data
(if stored based on Indexes)




String




number




mixed of data types

var Marks = [40, 70, 80, 90, 100]
0 1 2 3 4

Read | console.log(marks[2]) → 80

Update | Marks[2] = 60

up

console.log (Marks[2]) → 60

Delete delete Marks[2]

[40, 70, Empty, 90, 100]

↓

undefined

Marks[2] → undefined.

var fruits = ["Apple", "Banana"]

fruits.push("Mango")

["A", "B", "M"] delete

fruits.pop()

fruits.shift()

["B"]

fruits.unshift('pappaya')

["P", "B"]

Splice() vs slice() → task

Var amount = [100, 2000, 1000, 1, 2]

print only <=100

[100, 1, 2]

↓

filter()

---

Objects → collection / Combination of key-value pair.

Var address = {  (key → , value →)

addressLine1 : "Door 1",

addressLine2 : "Street",

pinCode : "56003",

City : "Bangalore"

}

key should always be string

string, Number, function, Array

object
Boolean

```
var user = {
    name : "prem",
    age : 26,
    phone : 224433,
    location : "Banglore",
    address : {
        a1 : "____",
        a2 : "____",
        pincode "____",
    },
    Marks : [100, 200, 10, 20],
    Greet : function (){
        return "Hey Ram fn"
    }
}
```

Object important

Var user = {

name : "prem"

}

console.log(user.name) => "prem".

user.phone = 92345678

update →

user = {

name: "prem",

phone : 92345678

}

way1: console.log(user.phone) => 92345678

obj

key

way2: log(user["phone"]) =>

Delete: delete obj.phone.

log(user) => user = {

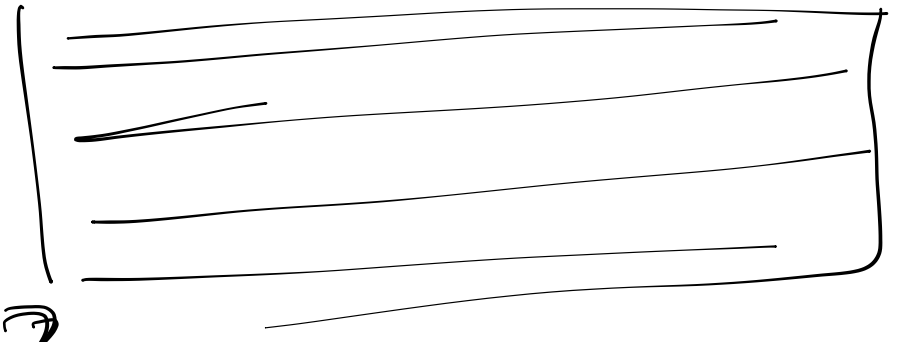
name: "prem"

}

# functions

↳ Group of code which we can reuse / maintability.

function multiple ( ) {



} Creation

}

multiple ( ) } Excewtion

parameters

function multipe( a, b, c ) {

result = a + b + c;

return result;

}

multipe ( 2, 3, 4) → 9

multiple (10, 20, 30) → // 60

↓ arguments

↳ Normal function.

Function Expression

```
var multiple = function(a, b, c){
    result = a + b + c
    return result.
}

multiple(1, 2, 3) => 6
```

(11f)

```
(function (a, b, c) {
    result = a+b+c
    return result
})(1, 2, 3) => // 6
```

---

```
function fn1((args)) {
    console.log(args)
}
fn1(1, 2, 3, 4)
```

---

```
function f1(a, b, c) {
    return a+b+c
}
f1(1, 2)
```

undefined.

$c = 0$

$1 + 2 + undefined$

↓

Error.

function fn(a, b, c=0) {

}

---

arrow functions

↳ syntax sugar

```
var multiple = (a, b) => a+b
```

```
var multiple = (a, b) => {

    return result

}
```

args wont work in arrow functions