10 September 2024 15:24

MERN

JS

M → DB

E → Framework (Nodejs)

N → Runtime Environment (JS)

R → Library.

ES6 (2015) ECMA → European Computer Manufacturing Association

ECMA 6

ES 6

Engine → Compiler + Interpreter

Safari → JS loops

IE → Chakra

Firfox → Spider Monkey

Brave / Crome → V8 Engine

JS says

Functions are the No. 1 citizen.

Object

object

var a;

arguments

function add(x, y)

int, float, double, long char, string

function add(int x, float y)

{ }

Loosely Typed

var a = 10;

a = " ";

int a;

a = 10;

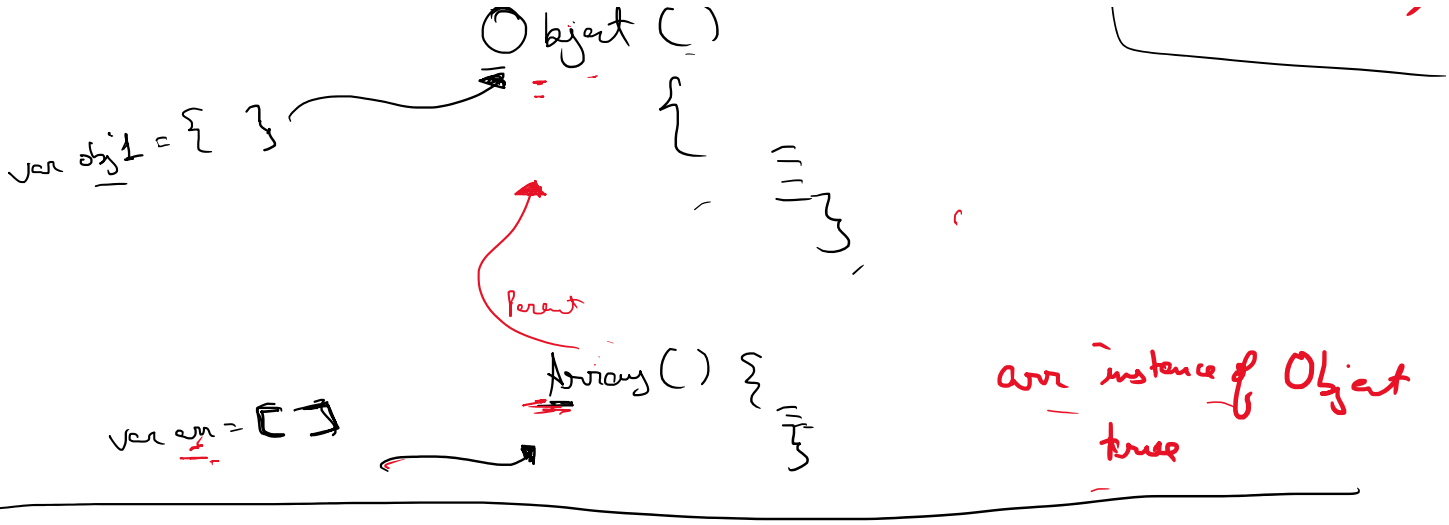
a = 10000;

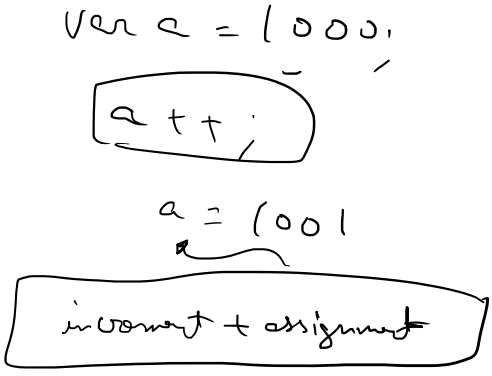
a = "Brijesh";

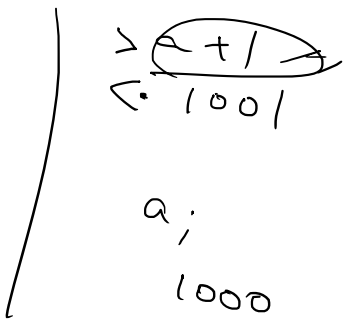
God level / TOP Level

Object ( )

{

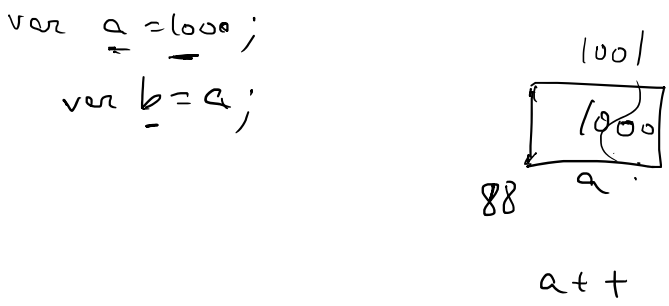


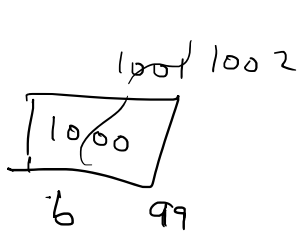




Immutability

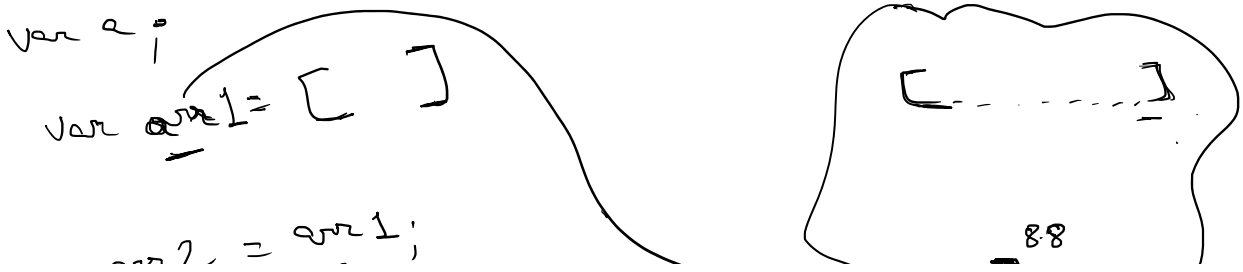
changes done in either copied value or original value, does not affect the other





Immutability

number, string, boolean, bigInt,



var arr2 = arr1;

memory address / reference.

88

arr2.pop()

Mutability.

changes done in Either original value or copied value, will ultimately affect the other.

By car.

Keys

Keys: car, Same car.

car → dent: Original car damage.

var obj1 = { }

null internally treated as {}

type of obj1 - 'object'

obj1 = null

removed

G.C | Garbage Collection | algo

{ key : value }

Heap memory

200

null

To dereference an object

acts like scissor

primitive type

'object' ← arr, obj

p = ""

p = ''

p = ``

'string'

var a = new String("priya");

new → object

String

AUTOBOXING

Powers, methods, Properties

duplicity

"priya"

var a = "priya";

a.toUpperCase();

'PRIYA'

Normal String (value type)

Properties

convert value type

undefined == null

== loose comparison

1. Type Convert
2. value checking

empty values

=== strict comparison

1. Type check
2. value check

---

1. add(1, 2); // no. of arguments → Numbers, Sum
2. add("Four", 10, 20); // 10, 30
3. add([10, 20, 30], "four", 10, 20);
4. add([10, 20, 30], "priya", 10, 20, one, two);

add(1, 2, 3, "4"); // 10

Data types:

string

number

boolean

true false

number → success value

Error value (NaN) Not a Number

total(1, 2, 3, "4", "five", one, two);

argument [5] ( )

1

one( )

argument [5] = 1