09 May 2024 10:38

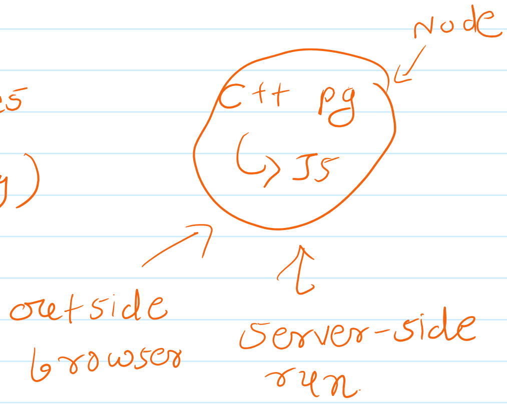
Javascript → logic

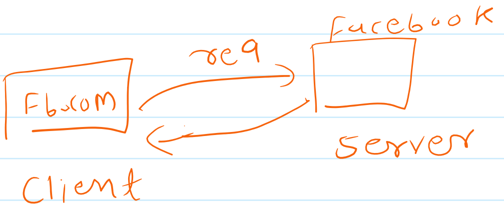
↳ light weight, Scripting

↳ web app, mobile app, games

↳ browser run (client-side lang)

Engine
↓
Crome :- V8





→ head, & body me add kar shakate ho.

→ body me last me add karnea.

↳ why because the dynamic and some are not render that reson.

↳ user exprinse good.

↳ How to run in terminal

→ node file name (JS file) (index.js)

Variable

↳ let a = 5 var a = 5

let name = "meet" var name = "meet"

scope — var ← globle
scope — let ← public

{
let a=5; } scope.
}

ex let a = 5; duplication not allow
let a = "meet"

Const :- not any value change. fixed value.

primitive type

primitive type

String
Number
Boolean
undefined → let a; → console.log(a) = undefined
Null
↳ empty value

Dynamic typing
↳ let a = 5;
a = "meet" ;

Reference type

objects , Arrays , functions
↳ let person = {
{ fName: "meet",
age : 18
};

key-value pair

Access → dot notsion
person. age

↳ person[age']
bracket notsion

Arrays :- D.S. used to contain a list of items. (all type data store)

let names = ['Meet', `Pankaj, 'Raj']

Operator

Arthmetic : + - * / % ** = x^y

Pre/post Inc/Dec

++x; → Pahele value incrment then print
x++
--x; → first print then incrment
x--;

Comparsion :

===
↑
Strict Equelity

datatype
value

both same

==
↑
loose Equelity

data → same

let num = 1

let str = '1'

✗ num === str

num == str ✓

With Non-Booleans

Falsy
↓
undefid
null
0
false
' '
NaN

Truthy
↓
any of not a falsy

objects :

Function → function name : function() {

console.log('name');

}

How to create a objects :-

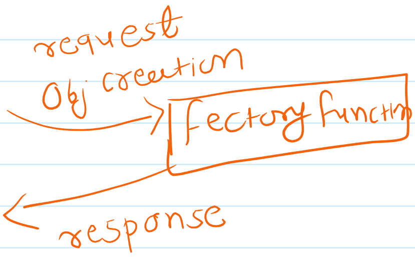
↳ (1) Fectory function ← object create

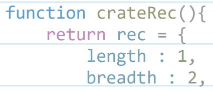
↳ function name() {

object

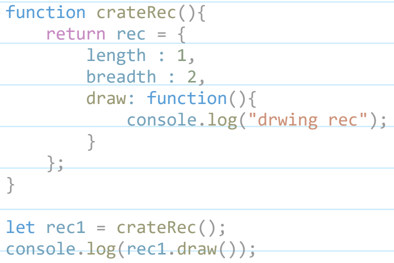
return object

}





```
function crateRec(){
    return rec = {
        length : 1,
        breadth : 2,
```



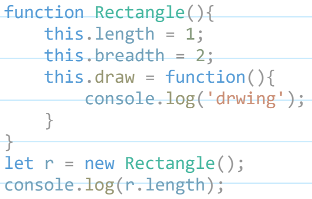
```
function crateRec(){
    return rec = {
        length : 1,
        breadth : 2,
        draw: function(){
            console.log("drwing rec");
        }
    };
}

let rec1 = crateRec();
console.log(rec1.draw());
```

← response object

let obj = createObject();

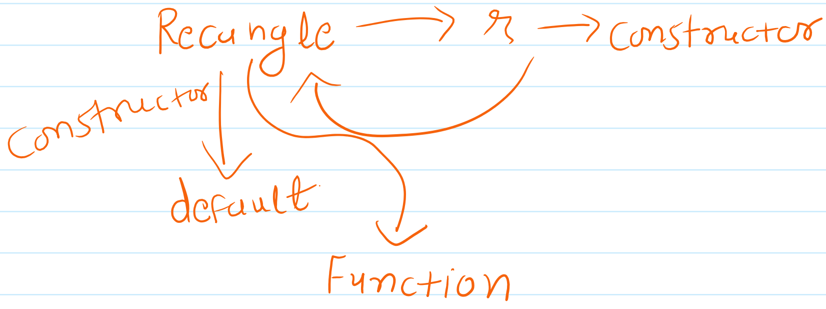
↳ function call

② Constuor Function



```
function Rectangle(){
    this.length = 1;
    this.breadth = 2;
    this.draw = function(){
        console.log('drwing');
    }
}
let r = new Rectangle();
console.log(r.length);
```

function is also object



Parameter

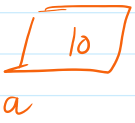
let r = new Function(_ , _ , `entire code`)

functions are objects

Note : Primitive are Copied by their value

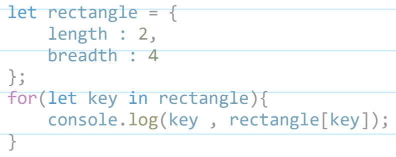
References are copied by their address

```
let a=10;
function inc(a)
{
    a++;
}
```



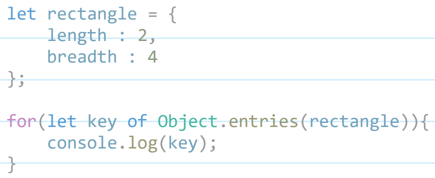
a++;
}
inc(a)
console.log(a);

for - in loop



```
let rectangle = {
    length : 2,
    breadth : 4
};
for(let key in rectangle){
    console.log(key , rectangle[key]);
}
```

For - of loop → only Arrays, Map are used.



```
let rectangle = {
    length : 2,
    breadth : 4
};

for(let key of Object.entries(rectangle)){
    console.log(key);
}
```

find a propertiy exist or not

Object cloning : (copy)   let a = {v : 10}
                                 let b = a

Iteration → for(let key in Rec)
Assign   {
Spread   Console. log(key, Rec[key])
                dest[key] = a[key];
let dest = { };   }

Assign used

empty object

let dest = Object. assign({}, src}

spread

let dest = {... Src};

let dest = {...src};

```
let src = {
    a : 10,
    b : 20,
    c : 30
};
let dest = {};
for(let key in src){
    dest[key] = src[key];
}
console.log(dest);
src.a++;
console.log(dest);
```

```
let src = {
    a : 10,
    b : 20,
    c : 30
};
let dest = Object.assign({} , src);
console.log(dest);
src.a++;
console.log(dest);
```

```
let src = {
    a : 10,
    b : 20,
    c : 30
};
let dest = {...src};
console.log(dest);
src.a++;
console.log(dest);
```

Garbage Collection : not used variable, const that memory deallocated in automatic.

↳ we have no control

↳ background run

In-built Object

Math : Math.max()
Math.min()
Math.random()

String : let lastName = new String("meet")

${ }

↑ dynamic

` `

↑ ગમે તેમ right કરી શકાય.

Date and Time

let date = new Date();

Arrays

Adding new elements

Finding element

Removing

Removing

Splitting

Combining

```
let number = [1,2,3,4,5];
console.log(number);
console.log(number[0]);
// end push
number.push(9);
// start push
number.unshift(8);
// middal push
number.splice(2 , 0 , 'a','b','c');
console.log(number);
// serching
console.log(number.indexOf(5));
// find
console.log(number.includes(5));
```

this used in object searching

```
// Serching ref for callback function used
let p = [
    {no : 1, name: 'meet'},
    {no : 2, name: 'pankaj'}
];
let p1 = p.find(function(p){
    return p.name = 'meet';
})
console.log(p1);
```

Callback Function

any name

array name . find (function(p1) {

return p. name=`meet'

}

```
let p = [
    {no : 1, name: 'meet'},
    {no : 2, name: 'pankaj'}
];
let p1 = p.find(p1 => p1.name === 'meet')
console.log(p1);
```

Arrow Function

(A) (B)

~~Function~~ ~~(~~p ~~)~~ =>

~~{~~

~~return~~ p. name= "Meet";

~~return~~ p.name= "Meet";

~~}~~

Ⓒ

p ⇒ p.name="meet"

```
// remove - end - pop
// remove - begin - shift
// remove - middale - splice(index ,
no.of.ele.delete)
let number = [1,2,3,4,5];
number.pop();
number.shift();
number.splice(2 , 1);
```

Combining & Slicing

let a = [1, 2, 3]

let b = [4, 5, 6]

let c = a.concat(b)

slice(_ , _)

starting index   ending index

both are slicine part only

Empty array → array length=0

spread operation

let a = [1, 2, 3]

let b = [3, 4, 5]

let c = [...a, ...b]

```
let a = [1,2,3];
let b = [4,5,6];
let c = [...a , ...b];
console.log(c);
```

for-of loop

```
let a = [1,2,3,4,5];
for(let key of a){
    console.log(key);
}
```

forEach loop

```
a.forEach(number =>
console.log(number));
```

joing array

[join method used

Joing array

↳ join Method used

```
let a=[1,2,3]
let b=a.join(`,');
```

Function :-

Syntax : function function_name( ) { ← input parameter

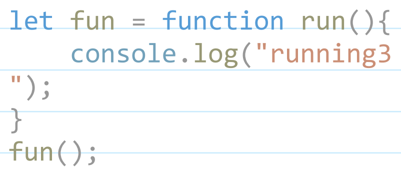
```
body → {

}
```

hosting :- JS Enging — first all function are auto Matic run.

named function assingment

```
let stand = function walk(){
        console.log('waking');
}
```

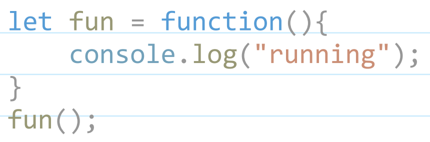
Funcation Call :- stand();



```
let fun = function run(){
    console.log("running3
");
}
fun();
```

Anonymous function assignment

```
let a = function ( ) {
body → {

}
```



```
let fun = function(){
    console.log("running");
}
fun();
```

← only two parameter

```
function sum(a,b){
    return a+b;
}
```

any number of parameter

```
function sum(a,b){
    let t =0;
    for(let val of arguments)
    {
        t += val
    }
}
```

Rest operator → . . .

function sum(_ , _ , _ , _)

function sum(...args)

```
function sum(...args){
    console.log(args);
}
sum(1,2,3,4,5);
```