Turwash Notes

12th February 2023

JavaScript - Class 4

Basics - 4

Functions :-

↳ a block of code that fulfills a Specific task.

Syntax :- (keyword: function; Functionname: printCounting)

1)
```
function printCounting () {
        console.log ("Counting")     } Fnc. body
}
```

* Why Functions ?

↳ Reusability
to reduce Bulky Codes
Remove / Reduce Bugs

Function Declaration :-

(1) (I/p parameter goes inside the parentheses)
```
function run () {
    console.log ("running")
}
```

To call → run()

✓ Hoisting in JavaScript is concept where, Process of moving function declaration to the top of file. done by JS Engine

↳ by the help of this we can call Function anywhere.

only works for Function Declaration.

② Function Assignment

↳ giving variable to a function. (assigning)

```
let Stand = function walk() {
    console.log("Walking");
}
```

To call → Stand() Not walk();

Hoisting Doesn't works here

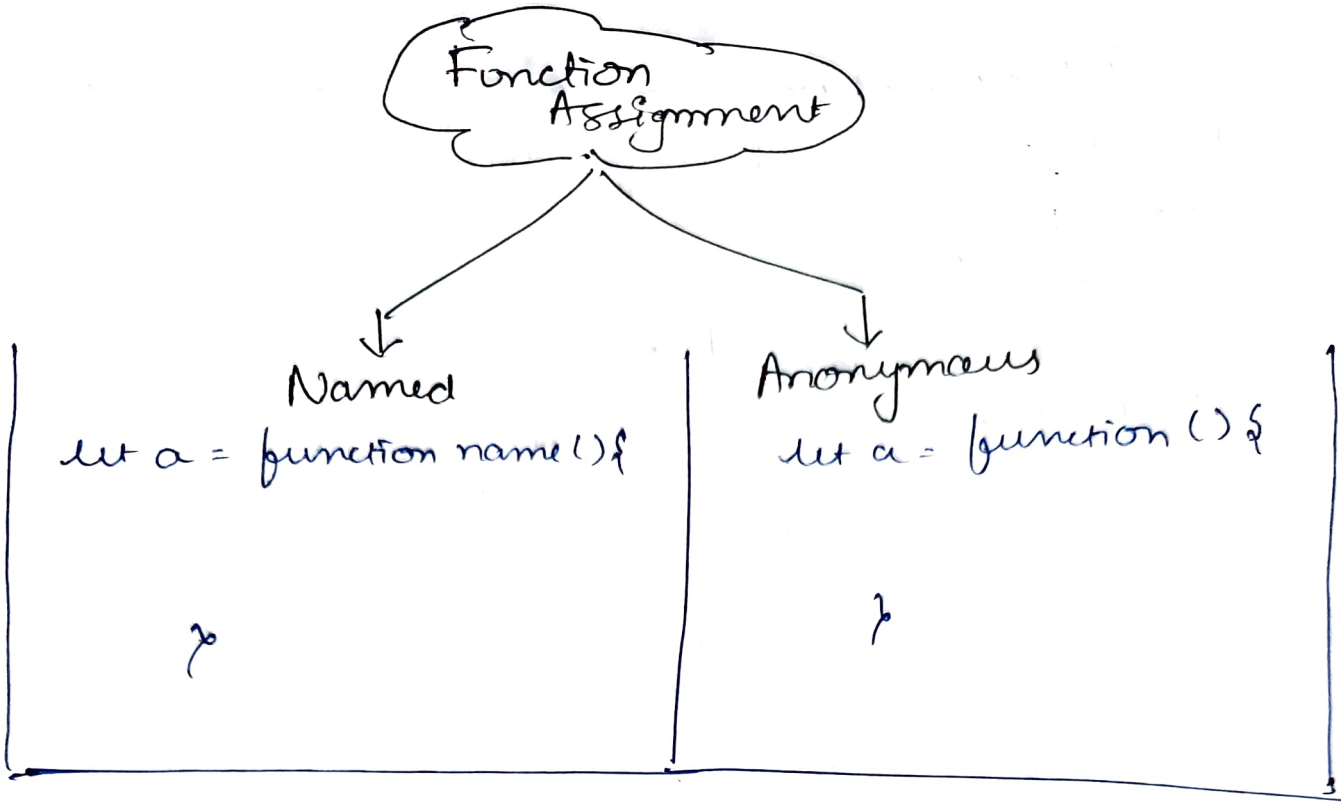
↳ only for function declaration.

③ Anonymous

↳ Name of function not present

```
let Jump = function() {
    console.log("Walking")
}

call → Jump();
```



Dynamic Function :-

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

1) clg (sum (1)); // will give NaN (undefined for b)

2) clg (sum()); // will give NaN (undefind for a & b)

3) clg (sum (1, 2, 3, 4, 5) // only 1 & 2 will be taken rest will be waste ans - (3)
Stored in Argument Object in JS.

Special Object → Arguments (for multiple passing of argument)

```
let sum (a, b) {
    let total = 0
    for (let value of arguments)
      total = total + value;
      return total;
}

let ans = sum (1, 2, 3, 4, 5)

console.log (ans)
```

← we can increase this to get new value.

Rest Operator :- ...

↳ we can handle multiple parameters in function using Rest operator.
This will create Array.

```
function sum (...args) {
    clg (...args)
}

sum (1, 2, 3, 4, 5, 6);
```

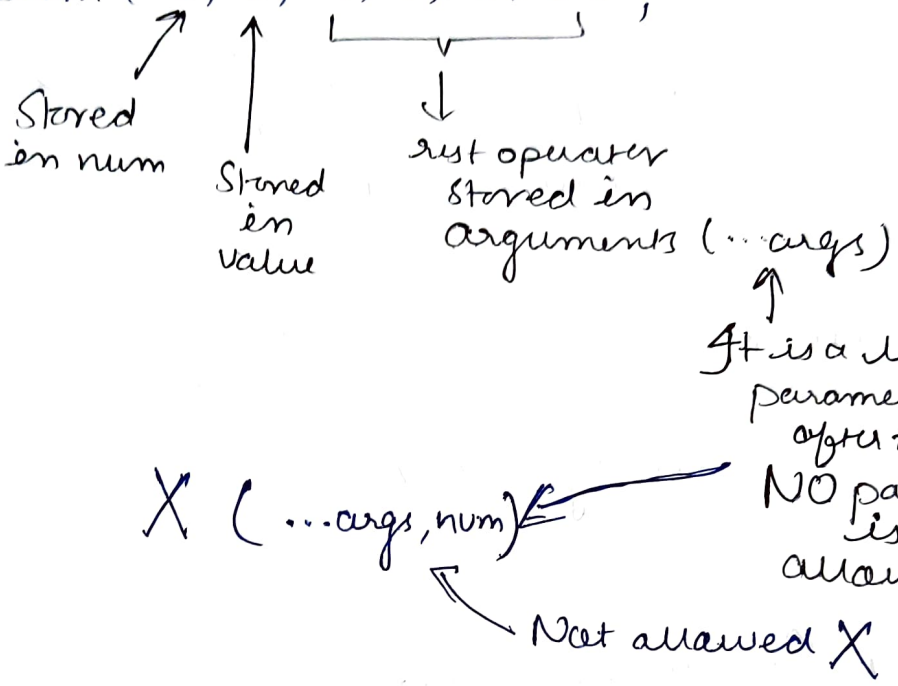
↳ [1, 2, 3, 4, 5, 6.] will be stored in array.

2) function sum(num, value, ...args) {

clg(...args);

}

sum(1, 2, 3, 4, 5, 6);



Default parameters :-

default default

function intrest (p, r=10, y=2) {

all rest have to be default

return p*r*y/100;

}

clg (intrest (1000, 5));

will give 100 taking y = 2 default.

if user gives input then input will be taken if not default will be taken.

```
let person = {
    fname : 'Tuwash',
    lname : 'Chakraborty',
};

function fullname() {
    return `${person.fname} ${person.lname}`;
}

clg (fullname());
```

↖ This is only read only function.

to manipulate.

Getter, Setter :-

```
let person = {
    fname : 'Love',
    lname : 'Babbar',
    get fullName() {
        return `${person.fname} ${person.lName}`
    },
    set fullName(value) {
        let parts = value.Split(' ');
        this.fName = parts[0];
        this.lName = parts[1];
    }
};
```

To call :-

```
clg (person.fullName);
        ↑ getter
person.fullname = 'Rahul Kumar');
clg (person.fullName); ← Sitter.
```

ERROR HANDLING

using Try & Catch block.

```
Try {
    code, if error goes to catch
}
catch (e) {
    // custom error message.
}
```

```
let person1 = {
    fname : 'Love';
    lname : 'Babbar',
    get fullName() {
        return `${person1.fname}
                ${person1.lname}`;
    },
    set fullName (value) {
        if typeof value !== String) {
            throw new Error ("Not a String");
        let parts = value.split(' ');
        this.fname = parts[0];
        this.lname = parts[1]; } };
```

error handling

```
try {
person1.fullName
    = true;
}
catch (e){
    alert(e);
}
```

Scope:-

↳ lifetime or lifespan of variable is scope

```
{
    let a = 5;
}
console.log(a); → error
```

let a = 5; } block scope

// Sorting:-

```
let a = [10, 30, 50, 60, 20, 80]
a.sort(function(a, b) {
    return a - b;
});
console.log(a);
```

return a - b; → For ascending

b - a → For descending.

a - b

- -ve → a will be placed before b
- 0 → a == b → a
- +ve → a will be placed after b

// Reducing an Array :-

using reduce method.

```
let arr = [1, 2, 3, 4];
  let total = 0,
For (let value in arr)
  total = total + value;
   console.log (total);
```

// to reduce we write call back function using 2 parameter

accumulator (total) — currentValue (loop)

```
let totalSum = arr.reduce ((accumulator,
    currentValue) => accumulator +
    currentValue, 0);
```

↑ accumulator initialized to 0

```
console.log (totalSum); → (10)
```

Working:- [1, 2, 3, 4]

accumulator = 0
currentValue = 1
accumulator = 0 + 1
= 1
current = 2
accumulator = 1 + 2
= 3
current = 3
accumulator = 3 + 3

current = 4
accumulator = 6 + 4
= (10)