JS Basic IV

Function — Block of code that fulfills a specific task

Syntax

```
function f_name(){
    // Block of code
}
```

Need of function :-
- readability improve
- to keep away bulky code
- to reduce bug changes
- code will be reuse

Declaration of function :-

```
function f_name(){
    // Block of code
}
```

hoisting :- It is proseess that it will take all function declaration to the top of file.
- ↳ It is done automatically by JS engines
- ↳ That means we can call function before is declaration. of

Function Assignment

Named function assignment

```
let ans = function sum(){
    console.log('2+5=7');
}
```

Anonymous function assignment

```
let ans = function(){
    console.log('2+5=7');
}
```

Calling a function : sum()

Note :- We can not call the function before its declaration because hoisting only works with function initialization not with function assignment.

Rest Operator → ...

Sometimes we need multiple parameters in function to handle this we use rest operator.

```
function sum (...args) {
    // Block of code
}
```

console.log(argument) → prints object
console.log(...args) → prints array
} both use to print argument of function

Default parameter

You have to set from right side

When user does not pass any value to the function then the function takes the value by itself.

```
function f_name (p, r = 5, n = 5) {
    // Block of code
}
```

Getter & Setter

Getter

to access properties

```
get fullName() {
    return `${per.fname}
    ${per.LName}`
}
```

Setter

to change or match properties

```
set fullName(val) {
    let parts = value.split(' ');
    this.fName = parts[0];
    this.lName = parts[1];
}
```

```
let peer = {
    fname: 'Khushi'
    LName: 'Modi'
}
```

Try & catch

Try -> The try statement defines a code block to run.
catch -> Statement defines a code block to handle any error.
finally -> Statement defines a code block to run regardless of the result.

```
try {
    //Block of code to try
}
catch (error) {
    //Block of codes to handle error
}
finally {
    //Block of code to run regardless of the try & catch result
    //
}
```

Reduce method in Array

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

0 → initial value of accumulator