Rules for naming variables

26/11/2021

You can't start with number

ex :- 1value ✗

value1 ✓

Variable → जो चीज change हो सकती है।

```
var value1 = 2;
console.log(value1 + 1)   → output = 3
   " "  (value1 * 3)   → 6
   " "  (value1 ** 2)  → 4          → power
   " "  (value1 ** 0.5) → 1.4142135623730951  → square root
```

* from special symbol we can use only underscore _ or dollar symbol

// first_name (✓)

// _firstname (✓)

// first$name (✓)

// $firstname (✓)

* we can't use spaces

// first name (✗)

* var first_name = "Vivek"

↓

अगर हम बीच में underscore लगाते हैं, तो उसे कहते हैं Snake case writing

* var firstName = "Vivek" // camelCase writing

// convention

Start with small letter and use camel case

firstName

✓ ES comp

(Let keyword)

हम अब तक var keyword का use कर रहे थे

```
var firstName = "Vivek";
```

But we can also use

```
let firstName = "Vivek";
```

* Always use let keyword for variable

// declare variable with let keyword

Difference betn var & let

```
var firstName = "Vivek"
var firstName = "Gupta"
console.log (firstName) → Gupta
```

यानी var वाला variable का नाम same रखने पर change allowed है।

```
let firstName = "Vivek"
let firstName = "Gupta"
console.log (firstName) → Error firstName is already defined.
```

But we can

```
let firstName = "Vivek"
firstName = "Gupta"
console.log (firstName) → Gupta
```

* block scope Vs function Scope

Constant

Variable → जो चीज change हो सकती है।

Constant → जिसकी value change नहीं कर सकते हैं।

```
const pi = 3.14;
console.log(pi);
```
→ output = 3.14

> Note:- बोलते वक्त भी बोले हम constant declare कर रहे हैं variable नहीं कर रहे हैं।

Note:- pi के साथ हम हर mathematical op^ns कर सकते हैं। but pi की value change नहीं कर सकते है वो fix है।

console.log(pi + 4) → 7.14 ✓

console.log(pi) → 3.14 ✓

String Indexing

```
let firstName = "harshit";
```

//	h	a	r	s	h	i	t
//	0	1	2	3	4	5	6

हर एक String में हर character की ~~value~~ index होती है or आप बोल सकते हैं position होती है।

h का index 0 है

a का index 1 है।

* Suppose हमें sting के character के by index number print करवाना है।

```
let firstName = "harshit";
console.log (firstName[6]);  → t
```

* How to find sting length ****

```
console.log (firstName.length);  → 7
```

> Note:- Index 0 से start होता है and length में हम number of character count करते हैं।

* Last Index : length - 1

Suppose हम sting लेते हैं काफी बड़ा जिसका हमें last character बताना है

```
console.log( firstName[ firstName.length - 1 ])
```

(Useful String Method)

(*) 1) trim() → ये space etc अगर हमारे string में रहता है तो ये उसे trim कर देता है।
space

Ex → let firstName = " harshit ";

console.log (firstName.length) → 13

let newStr = firstName.trim();

console.log (newStr.length) → 7

2) toUpperCase()

let upper = firstName.toUpperCase(); → HARSHIT

3) toLowerCase()

let lower = firstName.toLowerCase(); → harshit

4) slice() → Start index, end index

let slice = firstName.slice (0, 4)

console.log (slice); → hars

Note → slice जो है 0,4 तो है But ये 4th index को नहीं लेगा 0 to 3 तक को ही print करवाएगा

* अगर slice (1) → जिसे start index दे देंगे तो 1 से Last तक सारा print हो जाएगा arshit

[Typeof, string to number, number to string conversion]

1) typeof

```
let age=22;
typeof(age); → number

age= age + " ";
typeof(age); → string
```

convert number to string

Convert string to number

```
let mystr = "34"
typeof(mystr) → string

let pstr= +"34";
typeof (pstr) → number
```

→ normal string से आगे + लगा दें तो number में बदल जाएगा

दूसरा तरीका

```
let age = "18"
age = Number(age);
console.log(typeof (age)); → string
```

String Concatenation

```
let string1 = "harshit";
let string2 = "vashistha";
let fullName = string1 + string2
console.log(fullName);
```

→ harshitvashistha

(2 string को concatenate कर रहे हैं।)

```
let exactName = string1 + " " + string2
console.log(fullName);
```

→ harshit vashistha

(3 string को concatenate)

```
let string1 = "17"
let string2 = "10"
let new = string1 + string2
console.log(new)
```

→ 1710

(ये string है)

(ये भी string है)

New अगर हम चाहते हैं "17" ये string को number में बदले तो

```
let new = +string1 + +string2
console.log(new);
```

→ 27

Template String

27/11/2021

let age = 22;

let firstName = "harshit"

Now हम "my name is harshit and my age is 22" को print करवाना है।

So (1) तरीका let aboutMe = "my name is" + firstname + " and my age is " + age

But it is tough

(2) तरीका

let aboutMe = `my name is ${firstName} and my age is ${age}`

Backtike

dollar

placeholder

Undefined null bigint

(14) Topic

1) undefined let and var की variable only define करोगे and value assign नहीं करोगे तो वह undefined होगा

const में करोगे तो uncaught error देगा।

Ex → let firstName;

console.log (firstName); → undefined

var firstname;

console.log (firstName); → undefined

const first;

console.log (first) → uncaught error

Important → हम variable पहले बना सकते हैं and value बाद में डाल सकते हैं and console में एक से ज्यादा value print करवा सकते हैं।

Ex:-

```
let firstName;
console.log (firstName); → undefined

firstName = "Vivek";
console.log (typeof firstName, firstName); → string, Vivek
```

2) null

ये एक bug है Java Script का

```
let myvariable = null;
console.log (myVariable); → null
console.log (typeof null); → object
```

null का typeof null होना चाहिए but ये object दे रहा है

जो कि JavaScript का एक bug है।

TC39 ने change नहीं किया Bcos change से कोई Lanch website में दिक्कत आ सकती है So इसे bug ही रहने दिया।

3) BigInt

ये 2020 में ही आया है new Primitive type है।

let myNumber = BigInt(12);

console.log(myNumber) → 12n

let myNumber1 = 123n;

console.log(myNumber + myNumber1) → 135n

BigInt बनाने के 2 तरीके हैं = BigInt(12) or 123n

किसी भी के max reach जाकर हो

console.log(Number.MAX_SAFE_INTEGER) → _____

Note:- BigInt को BigInt type से mathematical op perform कर सकते हैं।

वरना explicit conversion का error आता है।

BigInt को दूसरे data type से मत जोड़ना।

15 Topic

"Boolean & Comparison operator"

1) Boolean → या तो true Return करेगा या false return करेगा

```
let num1 = 5
let num2 = 7

console.log(num1 > num2); → false
```

2) == Vs ===

```
let num1 = "7"; → ये int है
let num2 = 7; → ये भी int है

console.log(num1 == num2); → true
```

दोनों int था तो true दिया। अब हम 1 को String कर दे है and 1 को int

```
let num1 = "7"; → String
let num2 = 7; → int
console.log(num1 == num2); → true
```

So == (double equal) सिर्फ value check करता है datatype नहीं।

3) ===

```
console.log(num1 === num2) → false;
```

(4) // != vs. !===

```
let num1 = "7";
let num2 = 7;
console.log(num1 != num2); → false
```

यानी ये बोल रहा है की num1, num2 बराबर है Becos ये value check कर रहा है।

So

```
console.log(num1 !== num2); → true
```

ये अब बोल रहा की num1, num2 हैं अलग है

Becoss !== type and value दोनों को check करता है।

If else Truthy / False Values

16 Topic

If else

```
let age = 17;
if (age >= 18) {
    console.log("User play DDLJ");
}
else {
    console.log("User play mario");
```

user play mario

```
let num = 14;
if (num % 2 === 0) {
    console.log("even");
else {
    console.log("odd");
}
```

even

2) Falsy Values

नीचे लिखे 5 सारे falsy values हैं

```
// false
// " "
// null
// undefined
// 0
```

```
let firstName = "";
if (firstName) {
    console.log(firstName);
}else {
    console.log("firstName is empty");
}
```

यहाँ जो भी false value दोगे finally program false value देगा तो if block नहीं चलेगा and else ही चलेगा।

firstName is empty

3) Truthy values

ये 5 falsy के अलावा सारा truthy है।

Ternary Operator

17 Topic

Ternary opr मे line of code if के comparison में काफी कम हो जाता है।

```
let age = 3;
let drink = age >= 5 ? "coffee" : "milk";
console.log(drink);
```

milk

for true

for false

And Or Operator

18 Topic

AND

```
let firstName = "Harshit";
let age = 16;
if (firstName [0] === "H" && age > 18) {
    console.log("Name starts with H and above 18");
} else {
    console.log("inside else");
}
```

inside else

2) <u>OR</u> → OR के case मे दोनो सही to सही

1 सही 1 गलत तो सही

1 गलत 1 सही तो सही

2 दोनो गलत तो गलत

```
let firstName = "Harshit";
let age = 22;
if(firstName[0] === "H" || age>18) {
  console.log("inside if");
} else {
  consolelog("inside else");
```

inside if

एक सही एक गलत तो सही आएगा

OR मे एक भी condition true to output true आएगा

Nested if else

18 Topic

if के अंदर if else लिख दिया and else के अंदर if else लिख दिया तो ये known as nested if else

```
let winningNumber = 10;
let userGuess = +prompt("Guess a number");

if (userGuess === winningNumber) {
    console.log("Your guess is right!!");
}else {
    if (userGuess < winningNumber) {
        console.log("too low!!!");
    }else {
        console.log("too high !!!");
    }
}
```

+ लगाकर हमने string को number में change कर दिया

If elseif

(19 Topic)

```
let tempInDegree = 50;
  if(temInDegree < 0) {
      console.log("extremely cold outside");
  } else if( tempInDegree < 16) {
      console.log( 'It is cold outside);
  }
  else if( tempInDegree < 25) {
      console.log ("whether is okay ");
  } else if (tempInDegree < 35) {
      console.log ( " let's go for swim");
  } else if ( tempInDegree < 45) {
      console.log ("turn on AC");
  } else {
      console.log ("too hot!! ");
  }
  console.log ( "hello");
```

too hot
hello

Note:→ इसमें पहले if case check होता है then else if के बाद condition check होते हैं कोई भी else if का case fulfill हुआ तो उसके बाद control block से बाहर आ जाएगा

कोई condition satisfy नहीं तो else case चलेगा।

Switch Statement

20 Topic

```
let day = 2;
switch(day) {
    case 0:
        console.log ("Sunday");
        break;
    case 1:
        console.log ("Monday");
        break;
    case 2:
        console.log ("Tuesday");
        break;
    case 3:
        console.log("wednesday");
        break;
    case 4:
        console.log ("Thursday");
        break;
    case 5 :
        console.log ("Friday");
        break;
    case 6:
        Console.log("Saturday");
        break;
    default :
        Console.log("Invalid Day");
    }
```

Note:- अगर break नहीं लगाएंगे तो day का value 2 है So case 2 से Tuesday wed, Thurs Frid Sata print हो जाएगा।

So जैसे ही case Satisfy होगा then control switch के case से बाहर आ जाएगा।

While Loop

21 Topic

* Program to print 0 to 9

* एक तरीका

```
console.log(1)
console.log(2)
console.log(3)
:
console.log(9)
```

But not good approach

1
2
3
4
5
6
7
8
9

* दूसरा तरीका

```
let i = 0;
console.log(i);
i++;
console.log(i);
i++;
console.log(i);
i++;
console.log(i);
:
console.log(i);
```

i की value print की फिर increment किया फिर print किया and so on

0
1
2
3
4
5
6
7
8
9

But this doesn't fulfill DRY → Don't repeat Yourself

```
let i = 0;
while (i <= 3) {
    console.log(i);
    i++;
}
console.log("hello");
```

पहले i की value 0 थी then condition check हुआ i की value 3 था 3 से छोटा है अगर है तो i की value print 0 then increment 1 फिर 1 से 2 से 3 जैसे ही 3 से 4 हुआ condition check होगा तो false and loop बाहर आकर hello print कर देगा

0
1
2
3
hello

* First n natural number का sum

पहला तरीका

```
let num = 10;
let total = 0;

total = total + 1
total = total + 2
      :
total = total + 10
```

45+10=55

दूसरा तरीका

```
let num = 10;
let total = 0;
let i = 0;
while ( i <= 100) {
        total = total + i;
        i++;
  }
console.log( total);
```

5050

तीसरा तरीका

```
let total = (num * (num+1)) / 2;
console.log (total);
```

→ 55 आएगा

Becos num की value 10 है

For Loop

(22 Topic)

* Print 0 to 9

```
for ( let i = 0; i <= 9 ; i++)
    console.log ( i );
}
```

Note:- i की value आप for Loop की body के बाहर use नहीं कर सकते हैं।

Like आपको बाहर i की value console करवाना है, तो आप नहीं करवा सकते हैं।

Because let use किया है var use करते तो for की body के बाहर भी i को console करवा सकते।

* first n natural number sum using for loop

```
let total = 0;
let num = 10;
for( let i = 1; i <= num ; i++) {
    total = total + i
}
consolelog ( total);
```

break / continue key word

23 Topic

break

```
for (let i = 1; i<=10; i++) {
    if(i===4) {
        break;
    }
    console.log(i);
}
```

1
2
3

Note: Program में हमने जब i की value 4 हो जाए तो break लगा दिया जब i की value 1 हुआ तो print हुआ, 2, 3 But as i की value 4 हुई तो break चलेगा and control for के body के बाहर चला जाएगा

Continue

```
for (let i = 1; i<=10; i++) {
    if(i===4) {
        continue;
    }
    console.log(i);
}
```

1
2
3
5
6
7
8
9
10

i की value 3 तक तो सही print हुआ। when i की value 4 हुआ तो continue उसे skip कर देगा and control फिर for loop में चला जाएगा So फिर 5 printt then 10 तक हो गया print.

Do while Loop

24 Topic

Jai.nandMatrix1400008
gmail.com

```
let i = 10;
do {
    console.log(i);
    i++;
}
while (i <= 9);
```

10

do while Loop में एक बार तो चलता ही चलता है then condition check होता है।

do first time execute होगा then while चलेगा।

Array

Array → Ordered collection of items

ordered मतलब index

```
let fruits = ["apple", "mango", "grapes"];
let numbers = [1, 2, 3, 4];
let mixed = [1, 2, 2.3, "string", null, undefined]

console.log(mixed);
console.log(numbers);
console.log(fruits[2]);
```

[1, 2, 2.3, "string", null, undefined]

[1, 2, 3, 4]

grapes

* array के elements को square bracket में रखते हैं।
* array में ek type के data type को रख सकते हैं।
* array में element's index के basis पे set रहते हैं
* जैसे fruits array के 2nd index को call किया है।
* array के अंदर array भी store कर सकते हैं।

* Change of element in array. because array is mutable

```
let fruits = ["apple", "mango", "grapes"];
console.log(fruits);
fruits[0] = "banana";
console.log(fruits);
```

["apple, "mango", "grapes"]

["apple", "banana", "grapes"]

* array एक reference type है। and सारे reference type को object बोलते हैं।

Ex → console.log(type of fruits); > object

Note → अगर हमें पता करना है कोई चीज array है या नही

तो method है Array.isArray

```
console.log (Array.isArray(fruits));  > true
```

आप दूसरे को देख सकते है उसमें false होगा

```
let obj = { }
console.log (Array.isArray (obj));  > false
```

Becos obj array नही है।

Array push/pop/shift/unshift

26 Topic

(1) push → ये array के last में कुछ add करेगा।

Becos array mutable होता है हम इसके method के through original array में change कर सकते है।

```
let fruits = ["apple", "mango", "grapes"];
console.log (fruits);  > ["apple, mango", "grapes"];

fruits.push ("banana");
console.log(fruits);  > ["apple, "mango", "grapes", "banana"];
```

2) Pop → यह method array के last element को delete करता है।

```
fruits.pop();
console.log(fruits);
```
> ["apple", "mango"]

3) unshift → यह array के starting में add करेगा

```
fruits.unshift("banana");
console.log(fruits);
```
> ["banana", "apple", "mango", "grapes"]

4) shift → यह starting से element को delete करेगा

```
let removedFruit = fruit.shift();
console.log(fruits);
console.log("removed fruits is", removedFruit);
```
> ["[illegible]", "mango", "grapes"]

removed fruits is apple

* Push, Pop fast है as compared to shift and unshift

Because push pop में last से add delete हो जाता है

but shift unshift में सारे elements को memory में आगे

shift करना पड़ेगा फिर new को insert करना पड़ेगा।

27 Topic

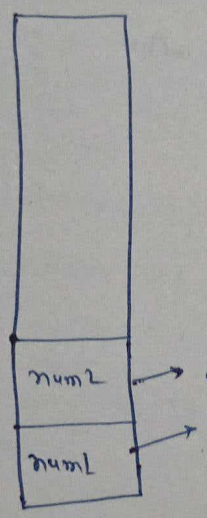
Primitive vs Reference Types

1) Primitive types

```
let num1 = 6;
let num2 = num1;
console.log ("num1 is", num1);
console.log ("num2 is", num2);
num1 ++;
console.log (after incrementing");
console.log ("value of num1", num1);
console.log ("value of num2", num2);
```

```
6
6
after incrementing
7
6
```

primitive types में



दोनों का address different होता है So पहले दोनों में 6,6 था but increment में हमने num1 में दिया So num1 = 7 हुआ जब num2 वही रह गया।

2) Reference Types

```
let array1 = ["item1", "item2"];
let array2 = array1;
console.log("array1", array1);
console.log ("array2", array2);
array1.push ("item3");
console.log ("after pushing element to array1");
console.log ("array1", array1);
console.log ("array2", array2);
```

Output of Reference Type

array 1 > (2) ["item 1", "item 2"]

array 2 > (2) ["item 1", "item 2"]

after pushing element to array 1

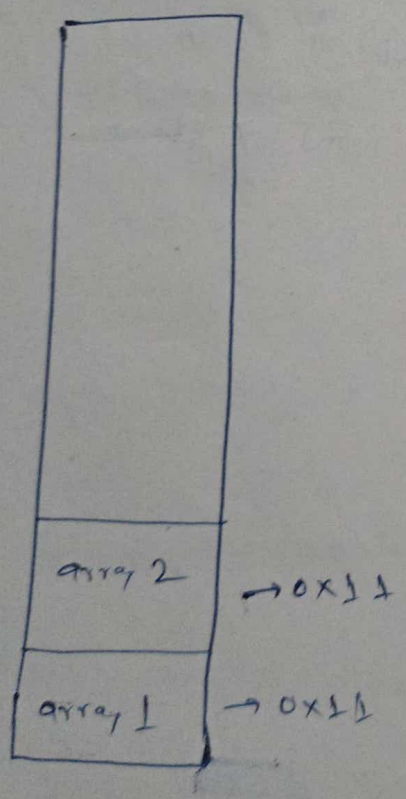
array 1 > (3)

["item 1", item 2", "item 3"]

array 2 > (3)

["item 1" "item 2", "item 3"]

So इसमें array 1 में जो change कर रहे array 2 में भी change हो रहा है।



But reference Types के case में जो array 1 का address है same array 2 को भी assign होता है pointer के through हम element को access करते हैं जो same address पे point करता है

Clone Array, Concat, Spread Operator

```
let array1 = ["item1", "item2"];
let array2 = array1;
console.log(array1 === array2);
```

true

इसमें array1, array2 दोनों का pointer किसी एक ही array को point कर रहा होगा। सो दोनों का item भी same हुआ इस console print होने से true आया

Now Clone का तरीका

```
let array1 = ["item1", "item2"];
let array2 = ["item1", "item2"];
array1.push ("item3");
console.log (array1 === array2);
console.log (array1)
console.log (array2)
```

false

["item1", "item2", "item3"]

["item1", "item2"]

इसमें array1 में जो change किया वो array2 में reflect नहीं हुआ and दोनों array भी same नहीं थे।

console में false आया मतलब दोनों अलग - अलग array हैं।

2) method

```
let array1 = ["item1", "item2"];

let array2 = array1.slice(0);

array1.push("item3");
```

यानी array1 के 0 index वाले को slice से हटा देगा and value को array2 में डाल देगा।

इससे array1 में हम new item last में push कर रहे हैं।

```
console.log(array1 === array2);
```
→ false

यानी दोनों अलग array हैं।

```
console.log(array1)
```
→ [item1, "item2", "item3"]

```
console.log(array2)
```
→ ["item1", item2"].

3) method

```
let array1 = ["item1", "item2"];

~~array1~~.

let array2 = [].concat(array1);
```

हमें खाली array लेना है उसमें array1 को concat करना है।

Spread Operator

```
let array1 = ["item1", "item2"];

let array2 = [...array1]
```

3 dot लगाना है then array1 मिलता है उसके array1 के सारे element 3 dot जो है वो spread कर देगा

3 dot अपने पुराने चीज को New चीज में spread कर देता है।

---

23 Topic

For Loop in array

```
let fruits = ["apple", "mango", "grapes", "banana"];

console.log (fruits.length);  > 4
```

Note length 1,2,3,4 तक count होता है।

```
console.log (fruits[fruits.length -1]);
```

अगर हमे last element delete करना हुआ तो length जो 4 है उसमें से 1 minus करेंगे जिसको 3 होगा and 3 index वे last element है and fruit से banana access ~~delete~~ हो जाएगा।

```
let fruits = ["apple", "mango", "grapes", "banana"];
for (let i = 0; i < fruits.length ; i++) {
    console.log (fruits [i]);
}
```

apple
mango
grapes

---

30 Topic

use const for creating array

const की value हम change नहीं कर सकते अभी const से दुबारा assign नहीं कर सकते हैं।

```
const fruits = ["apple", "mango"];
fruits.push ("banana");
Console.log ( fruits);
```

Const में हम change तो नहीं कर सकते But push तो हो रहा है।

Note:- push से new element insert कर रहे हैं जो तो हो जाएगा ही but const में new value assign नहीं कर सकते हैं मतलब रखा element change नहीं कर सकते हैं।

So const use करें

31 Topic

(While loop in array)

```
const fruits=["apple", "mango", "grapes"];
let i=0;
while ( i< fruits.length) {
  console.log ( fruits [i] )
  i++;
}
```

apple
mango
grapes

---

32 Topic

For of loop

```
const fruits =["apple", "mango", "grapes" ]
for ( let fruit of fruits) {
    console.log (fruit);
}
```

apple
mango
grapes

For of loop → it is good iterate

जब भी array को iterate करना हो, तो for of का use करते हैं।

for in Loop

33 Topic

```
const fruits = ["apple", "mango", "grapes", "fruit4"]

for (let index in fruits) {
    console.log (index);
}
```

let के बाद index याद किया अब कुछ भी रख सकते हैं

अगर सिर्फ index रखोगे तो

output
0
1
2
3
4

and अगर

console.log (fruits[index])

तो output = apple
mango
grapes
fruit4

Array के साथ for loop and for of loop ज्यादातर यूज करते हैं।

Array Destructuring

34 Topic

* Task array के items/elements को अलग - अलग variable में Store करना है।

```
const myArray = ["value1", "value2"];
let myVar1 = myArray[0];
let myVar2 = myArray[1];
console.log(myVar1, myVar2);
```

> value1, value2

Best array में एक shortcut है इसे करने का
By Destructuring

```
const myArray = ["value1", "value2"];

let [myVar1, myVar2] = myArray;
console.log(myVar1, myVar2)
```

> value1, value2

अभी हमने Array को destructuring किया है।

तरीका है `let [myVar1, myVar2, .....] = myArray`

जिस variable में रखना है

जिस array को करना है

destruce के variable को कहीं भी यूज कर सकते हैं।

Object

35 Topic

* array are good but not sufficient for real world data
* Objects store key value pairs
* objects don't have index
* Object are reference type

Object बनाना

const person = { } ⟶ array के लिए use करते थे square bracket
object के लिए curly brackets.

① key value pair ② key value pair

const person = { name : "Harshit", age : 22 };

में हमने 2 key value pair डाला

key → name
value → Harshit
pair

console.log (person); > {name: "Harshit", age: 22}

* जब हम object को curly brackets लगाकर बनाते हैं, तो उसे object literals भी कहते हैं।

* array में जैसा index होता था object में index नहीं होता
So access कैसे करेंगे ---

* Dot लगाकर access कर सकते हैं।

```
Console.log (person.name);  → Harshit
Consle.log (person.age);  → 22
```

Note:- person object में name, age key हैं

and कभी-कभी name, age को properties भी बोलते हैं।

Like person के name properties को access करो

* Object में array भी store कर सकते हैं।

```
const person = {
    name = "harshit",
    age = 22,
    hobbies : ["guitar", "sleeping", "listening music"]
}
```

key value pairs

```
consle.log (person.hobbies);  → ["guitar, "sleeping", "listening music"]
```

* How to add key value pair to objects

```
person.gender = "male";
```

consle करके देखलो add हो गया होगा person में

* JavaScript में key जो होती है वो by default String में होती है।

(2) console.log (person ["name"]); > Harshit

How to access Data from object

(1) * पहले हमने dot का use किया था

person.name

अब bracket लगाकर किया

person["name"]

→ But name को string में डाला

So key By default string हो जाता है So string लगाकर access करें।
बिना string के error आ जाएगा।

* अगर square bracket में key का name बिना string के पास करोगे तो वह error देगा।
ये data access from object का bracket वाला तरीका है।

* अब हमें new key value pair add करना हो, तो dot से हमने देखा था But हम bracket से भी add कर सकते हैं।

person["gender"] = "male";

36 Topic

Difference between dot and bracket notation

37 Topic

How to iterate object

2 way to iterate object

1. for in loop
2. object.keys

```
const person = {
    name: "harshit",
    age: 22,
    "person hobbies": ["guitar", "sleeping", "listening"]

for (let key in person) {
    console.log (key);
}
```

इससे सिर्फ key-key print होगा

name
age
person hobbies

But पूरा key value pair चाहिए

```
for (let key in person) {
    console.log (`${key} : ${person[key]}`);
```

name : harshit
age : 22
person hobbies : guitar, sleeping, listening

38 Topic

Computed Properties

```
const key1 = "objkey1";
const key2 = "objkey2";

const value1 = "myvalue1";
const value2 = "myvalue2";
```

हमारा Task है एक ऐसा object बनाना जिसका key key1, key2 है and and key का value value1 and value2 का value हो।

~~2 method~~

```
const obj = {
    objkey1 : "myvalue1",
    objkey2 : "myvalue2",
}
```

हमें इस तरह से बनाना है मतलब Task है।

1 method

```
const obj = {
    key1 : value1,
    key2 : value2
}
console.log(obj);
```

{key1 : "myvalue1", key2 : "myvalue2"}

But हमें output में चाहिए था [objkey1 : myvalue1]

दिए Type

So value सही आ गई But हमें key को भी compute करना था

1st method

key1 को square bracket में डाल के किसी computed properties बोलते हैं।

```
const obj = {
    [key1] : value1,
    [key2] : value2
}
console.log(obj);
```

output

{objKey1: "myValue1", objKey2: "myValue2"}

2nd method

```
const obj = { }
obj[key1] = value1;
obj[key2] = value2;
```

same output

Spread Operator

39 Topic

```
const array1 = [1, 2, 3];
const array2 = [5, 6, 7];

const newArray = [...array1, ...array2];
console.log (newArray);
```

> output

[1, 2, 3, 5, 6, 7]

(आपको array1, array2 के अलावा भी 5वा element चाहिए तो [...array1, ...array2, 8, 10] कर लेंगे)

So ये तो Array का concept था now हम इसे

* Now आप इस तरह भी spread कर सकते हैं।

```
const newArray = [ ..."abc" ]
console.log(newArray);
```

यहाँ हमने array के elements नहीं लिए हैं direct string ले लिया and उसको spread किया तो वे individual element बन जाएंगे।

Output = ["a", "b", "c"]

* const newArray = [... "123456789"];

console.log (newArray);

> ["1", "2", "3", "4", "5", "6", "7", "8", "9"]

Note- अगर आप without string नहीं उस करके direct number लिख दोगे तो error देगा Becos ये iterable नहीं होता

const tip = [...123] ✗ error

↓

(direct) Numbers are not ~~iterable~~ iterable so no used as Spread. Use it in string.

Spread opr in objects

* Note object के अंदर एक key एक बार से ज्यादा use नहीं हो सकता है।

like

```
const obj1 = {
    key1: "value1",
    key2: "value2",
    key1: "value 59",
};
```

→ { key1 ने पहले use हो चुका है
not possible again
यानी same key नहीं रह सकता बार-बार }

```
const obj1 = {
    key1: "value 1"
    key2: "value 2",
};
const obj2 = {
    key1: "value Unique",
    key 3: "value 3",
    key 4: "value 4",
};
const newobject = {...obj1, ...obj2};
console.log(newObject);
```

Note:- हमने newobject बनाया जहाँ obj1, obj2 को spread कर रहे हैं। But obj1 में key1 है and obj2 में भी key1 है so console करने के बाद

Output

[key1: "valueUnique", key2: "value2", key3: "value3", key4: "valu

यानी obj2 को बाद में spread किया so obj2 के key1 को रखा obj1 के key1 को हटा दिया।

अगर ऐसा करें।

```
const newObject = {...obj2, ...obj1};
console.log(newObject);
```

Output

{key1: "value1", key3: "value3", key4: "value4", key2: "value2"}

यानी obj1 बाद में spread हो रहा है so key1 का value obj1 वाला होगा।

Note:- हम अपना key भी new object में बना सकते हैं / add कर सकते हैं।

```
const newObject = {...obj2, ...obj1, key69: "value69"}
```

* iterable

```
const newObject = {... "abc"};
console.log(newObject);
```

यहाँ iterable हो जाएगा and key value pair बन जाएगा

{0: "a", 1: "b", 2: "c"}

* `const newObject = {... ["item1", "item2"]};`
`console.log(newObject);`

[हमने array दिया है]

Output

{0: "item1", 1: "item2"} → But में ज्यादा यूज नहीं होता है।

* कभी कभी problem solving मे कोई character आदि से आप उठा सकते है।

```
const newObject = {... "abcdefghijklmnopqrstuvwxyz"};
console.log(newObject);
```

Output

{ 0: "a", 1: "b", 2: "c", 3: "d", 4: "e", 5: "f" ... }

40 Topic

(Object Destructuring)

```
const band = {
bandName : "led"
famousSong: "Stairway to heaven",
};
const bandName = band.bandName;
const famousSong = band.famousSong
console.log( bandName, famousSong);
```

यहाँ Destructuring तकनीक का इस्तेमाल किया गया

Output

Led, Stairway to heaven

* Shortcut method for object Destructuring

```
const { bandName, famousSong } = band;
console.log( bandName, famousSong )
```

Output

Led, Stairway to heaven

अभी तक नीचे हमने क्या किया की पहले से बने object के key को हम same use करना है अपने दूसरे object में।

जो हमने कर लिया।

Now 2nd Task की हम पहले बने हुए (obj destructuring) से पहले के key का name अपने मन से अच्छे value उसका ले, तो।

```
const band = {
    bandName : "led",
    famous Song : "stairway"
    year : 1968,
    anotherFamousSong : "kashmir",
};

let { bandName : var1, famousSong : var2 } = band;
console.log(var1)
```

> Output
>
> led

* हमने object band से bandName, famousSong ले लिया but rest नहीं लिया So rest जैसे (year, another) को भी हम ले सकते हैं।

```
let { bandName, famousSong, ...restProps } = band;
console.log(restProps);
```

> Output
>
> {year : 1968, anotherFamousSong : "kashmir"}

key Value Pair

बाकी लिया ^ आ गया

41 Topic

[Object inside Array]

Goal → Object के अंदर Array कैसे Store करते हैं।

// very useful in real world applications

```
const users : [
    {userid : 1, firstName: 'harshit', gender : 'male'};
    {userid: 2, firstname: 'mohit', gender: 'male'};
]

for(let user of users){
    console.log(user);
}
```

output

```
{user id : 1,  — }
{user id : 2  — — }
{user id : 3  — — }
```

* हम console.log(user.userid) भी करेंगे

output

```
1
2
3
```

* हम console.log(user.firstName) भी check कर सकते हैं

output

```
harshit
mohit
nitish
```

42 Topic

[Nested Destructuring]

Goal → हमने पीछे जो बनाया था main object जिसमें user का सारा data था उससे हमें किसी username चाहिए तो भी इसके array में करना है।

```
const users = [
    {user id: 1, firstName: 'harshit', gender: 'male'},
    {user id: 2, fN: 'mohit', gender: 'male'},
]
const [user1, user2, user3] = users;
console.log(user1);
```

3 new variable लिया array में and इसमें users obj से value आ जाएगा।

Now अब हमें user 1 का firstName चाहिए and user 3 का gender चाहिए।

```
const [{firstName}, , {gender}] = users;
console.log(firstName);
console.log(gender);
```

Output

harshit
male

Note- हमारे पास 3 objects है 1st object से firstName लेना है ले लिया 2nd Object से कुछ नहीं लेना था so खाली छोड़ा and last object से gender लेना था।

43 Topic

[Function]

function → as a आज्ञाकारी नौकर उसे कुछ काम दो तो कर देगा।

Normal Example

function :-

Suppose हम Point करवाना चाहते हैं Happy birthday Song तो

(1) तरीका console.log ("Happy Birthday");

(2) तरीका हम एक function बनाए and उसे call कर दें।

```
function singHappyBirthDay () {
    console.log ("happy birthday to you ... ");
}
```

fn बना { } curly brackets में काम बताया क्या करना है।

```
singHappyBirthDay ();
```

output
happy birthday to you ...

fn call किया।

* हमें एक काम करना है 2+4 बार-बार करना है, तो fn बना लो call करते रहो।

```
function twoPlusFour () {
    console.log (2+4);
}

twoPlusFour ();
twoPlusFour ();
```

output
6
6

ये fn को call किया
or
fn से invoke किया
or
fn को run किया

But इसलिये तो सिर्फ 2+4 ही होगा। Let suppose हमें 5+8 करना है कभी 3+2 कभी कुछ then.

Function Declaration

```
* function twoPlusFour() {
        return 2+4;
  }

const returnedValue = twoPlusFour();
console.log(returnedValue);
```

→ पहले मैं fn कुछ का था like console में 2+4 But अब यह return का रहा है 2+4; Now इस return fn से हम const में store कर लेते हैं and उस constant को console करेंगे।

Output

6

Function of Sum of Two numbers

```
function sumTwoNumbers (num1, num2) {
    return num1 + num2;
}
const returnedValue = sumTwoNumbers (2,6);
console.log (returnedValue);
```

Output

8

Note:→ ऊपर में जो num1, num2 लिखा है वो parameter है and नीचे जो 2,6 है वो argument है।

अगर argument में खाली छोड़ेंगे तो Nan आएगा output Bcos

undefined + undefined = Nan

For Three numbers

```
function sumThreeNumbers(num1, num2, num3) {
    return num1 + num2 + num3;
}

const returnedValue = sumThreeNumbers(2, 3, 5);
    console.log(returnedValue);
```

Now इसे बोल कैसे सकते हैं:-

Generally,

हमने एक f^n बनाया है जो as a input लेगा 3 numbers and return output में देगा 3(तीनों) का sum.

Actually,

3 parameter pass किया and 3 argument दिया

Odd or even

```
function isEven (number) {
    if (number % 2 === 0) {
        return true;
    }
    return false;
}

console.log (isEven (3));
```

Output

false

अगर normal console में देखें, तो

console.log(12 % 2 === 0)

ये आ तो true देगा आ तो false देगा यहाँ true दिया तो same

हम isEven में भी apply करते हैं।

```
function isEven (number) {
    return number % 2 === 0;
}
console.log (isEven (3));
```

> output
> False

* एक fn बनाओ जो input में string ले and string का first character return करे।

```
function firstChar (anyString) {
    return anyString[0];
}
console.log (firstChar ("zbc"));
```

→ output

z

* एक array fn बनाओ जिसमें target element find करता है मिल गया तो index बताए
वरना नहीं तो −1 देगा

```
function findTarget (array, target) {
    for (let i=0;  i< array.length;  i++) {
     if ( array [i] = = = target) {
        return i;
     }
    }
    return − 1;
}
const myArray = [1, 3, 8, 90]
const ans = findTarget (myArray , 8) ;
Consoler.log (ans);
```

[Function Expression]

44 Topic

Function expression मतलब एक variable में f^n को assign कर देते हैं।

```
const singHappyBirthday = function(){
    console.log("Happy Birthday");
}

singHappyBirthday();
```

f^n ही बनाया But अब const keyword लगा दिया

f^n को call किया

Output

happy birthday to you...

Note:- function वैसे ही बनाया But <u>function keyword</u> हटा दिया and उसे = के बाद लिख दिया

पुराना → यानी Function Declaration

```
function isEven(number){
    return number % 2 ===0;
}
```

Fn Expression

```
const isEven = function(number){
    return number % 2 ===0;
}

console.log(isEven(2));
```

True

45 topic

Arrow functions

Old Fn

```
const singHappyBirthday = function() {
    console.log("Happy Birthday");
}
```

Arrow Fn

```
const singHappyBirthday = () => {
    console.log("Happy Birthday");
}
```

Change किया → function key word हटा दिया and () parenthesis के बाद => Sign लगा दिया

```
* const sumThreeNumbers = (number1, number2, number3) => {
    return number1 + number2 + number3;
}
const ans = sumThreeNumbers(2, 3, 4);
console.log(ans);
```

Output

9

* Now अगर parameter एक ही हो and f^n एक ही line में return कर रहा है, तो हम ऐसे भी लिख सकते हैं।

```
const isEven = number => number % 2 ===0;
```

parenthesis हटा दिया Because एक ही parameter है

return keyword भी हटा दिया

46 Topic

[Hoisting]

Hoisting → function को call हम f^n के declaration से पहले कर लेते हैं तो चल भी जाता है।

* But Function Expression and Arrow function में Hoisting काम नहीं करता है।

```
hello();
```

→ f^n call करा है

```
function hello() {
    console.log("Hellow World");
}
```

Output

Hellow World

ये function declaration है। So Hoisting चल गया।

hello(); → call पहले किया → fn expression बनाया

```
const hello = function () {
    console.log ("Hello world");
}
```

→ तो uncaught Error देगा

* Same Arrow function भी uncaught error देगा

47 Topic

[function inside function]

Normal single fn बनाता हूँ।

```
const app = () => {
    console.log ("inside app");
}
```

→ fn बनाया expression वाला arrow वाला भी बना सकते हो

```
app();
```

→ fn call किया

output

inside app

```
* const app = () => {
    const myFunc = () => {
        console.log ("Hello world");
    }

    const addTwo = (num1, num2) => {
        return num1 + num2;
    }
    const mul = (num1, num2) => num1 * num2

    console.log ("inside app");
    myFunc();
    console.log(addTwo(2,3));
    console.log(mul (2,3));
}
app();
```

→ main function

→ inside में 1st fn

→ inside में 2nd fn

→ inside में 3rd fn

→ तो console main fn का है

→ 1st fn को call किया. Same other को भी call किया

Note:- inside fn को main के ही body में call करना

49 Topic

[Block Scope Vs function Scope]

* let and const are block scope. → let and const block के अंदर कैप्चर होता है।
* var is function scope. → मतलब var पूरे fn में कहीं भी use कर सकते हैं।

```
{
  let firstName = "harshit";
}
console.log (firstName);
```

output

Uncaught Reference Error

let and const अपने block के within ही call किए जा सकते हैं जो block { curly brackets के within है। But console firstName का इस बाहर कर रहे हैं तो uncaught error देगा।

*
```
{
  const firstName = "harshit";
  console.log (firstName);
}
```
→ block 1

```
{
  const firstName = "mohit";
  console.log (firstName);
}
```
block 2

Q. इस बार block में console किया तो output

output

harshit

mohit

आ गया

* Note पूरा file 1.js भी एक block ही है कि आप यूँ कर सकते हैं।

```
const firstName = "gaurima";
consoleLog (firstName);
```

output
gaurima

Note इसमें कहीं curly Brackets नहीं लगा कि block पूरा file की तरह

* let and const तो block scope में ही रहता है।

यानी curly Brackets के अंदर ही scope रहता है।

But var curly brackets के बाहर भी यूज कर सकते हैं।

```
{
  var firstName = "harshit";
}

console.log (firstName);
```

output
harshit

* हम किसी if में var define करें and if के block के भी बाहर var को call कर सकते हैं।

```
if (true) {
    var firstName = "harshit";
    consoleLog (firstName);
}
```

if block

```
consoleLog (firstName);
```

→ harshit

50 Topic

Default Parameter

Default Parameter :→ अगर हमने parameter में 2 value pass किया But argument में एक ही value डाल रहे हैं, तो Nan आएगा इससे बचने के लिए Default हम set करते हैं parameter

```
function addTwo (a,b){
    return a+b;
}
const ans = addTwo(4)
console.log(ans);
```

> output
> Nan

4+ undefined = Nan हो गया हमें 2 argument pass करना था But एक ही pass कर रहे हैं।

Sol^n

Security Policy

① Method :- हम चाहते हैं b की value pass न करे and by default b की value 1 हो जाए, तो

```
function addTwo(a,b) {
    if (typeof b === "undefined") {
        b=1;
    }
    return a+b;
const ans = addTwo(4,5);
console.log (ans)
```

→ output

5

Note :- आप b की value pass कर भी सकते हो default set करने के बाद भी

Because b तब undefined नहीं रहेगा and तब b का default value set नहीं होगा आप जो डालोगे b का (4, 5) वो ही चलेगा

(2) Method (New)

```
function addTwo (a,b=0) {
    return a+b;
}
const ans = addTwo (4)
console.log (ans);
```

> output
> 4

[Rest Parameter]

51 Topic

Rest Parameter → Suppose 3 value है parameter में but हम argument में (3,4,5,6,7,8,9) यानी 3 से ज्यादा values डाल रहे हैं।

(1) तो बाकी 3,4,5 के अलावा argument का क्या होगा

(2) हम चाहते हैं कि 3,4 (a,b) में जाए and 5,6,7,8,9 c में as a array element चला जाए।

Soln:- तो c को आपको (...c) ऐसे parameter बनाना होगा

```
function myFunc(a, b, ...c) {
  console.log('a is {a}');
  console.log('b is {b}');
  console.log('c is {c}');
}
myFunc(3,4,5,6,7,8,9);
```

Output

a is 3
b is 4
c is 5,6,7,8,9

Note → c का जो value आया वो array format में है but हम console में template Literal जैसा डाल रहे हैं।

So c का value array format में चाहिए, तो

```
console.log('c is', c);
```

Output

c is 9 (5) [5,6,7,8,9]

Task → एक ऐसा program बनाओ जिसमें अलग argument डालो सबका ही sum हो जाए।

```
function addAll (...numbers){
    let total = 0;
    for (let number of numbers){
        total = total + number;
    }
    return total;
}
const ans = addAll ();
console.log (ans);
```

output

15

[Param Destructuring]

52 topic

```
const person = {
    firstName: "harshit",
    gender : "male",
}
```

यहाँ person constant variable में object pass किया 2 key value pair diet

```
function printDetails (obj)
    console.log (obj.firstName)
    console.log (obj.gender);
}
```

→ ये fn as a input लेता कोई भी object, and console करेगा उस objects को

उस object में से firstName print करके दे देगा।

```
printDetails (person);
```

→ fn को call किया and person object pass कर दिया।

Output

harshit
male

Note:- function printDetails ( )

यहाँ क्या आएगा = object आएगा

तो क्यों न यही Bracket में object का destructuring करते।

तो curly brackets लगाके firstName and gender लिख दो इससे local memory में firstName and gender variable बन जाएगा and उसका value इस print कर देगी।

```
function printDetails ({firstName, gender}) {
    console.log (firstName);
    console.log (gender);
}
printDetails (person);
```

> output

hamshit
male

Note:- parameter Destructuring object के साथ use करते हैं।
React में बहुत use करते हैं।

500

[(callback function)]

53 Topic

callback function → एक function बनाया myFunc (a)
जिसमें एक parameter a pass किया then argument में

हम ([1,2,3]) → array pass कर सकते हैं।
हम ("abc") → string pass कर सकते हैं।
हम (name: 'Vivek') → object भी pass कर सकते हैं।

But क्या हम a के जगह function (फंक्शन) pass कर सकते हैं ?

```
function myFunc (a) {
    console.log (a);
    ~~console.log (a);~~
    console.log ('hello world');
}
myFunc ("abc")
```

> output

abc
hello world

a parameter के लिए argument में string pass किया

```
function myFunc2() {
    console.log ("inside my func 2")
}
```

ये हमने एक simple function बनाया है।

```
function myFunc (a) {
    console.log (a);
}
```

ये a parameter वाला function है

```
myFunc (myFunc2);
```

देखो myFunc में as a argument हमने एक function को डाल दिया।

Output

```
myFunc2 () {
    console.log ("inside my func 2")
```

हम चाहे तो इस func को call कर सकते है।

So हम myFunc में fn as a input accept कर call कर सकते है।

```
function myFunc2() {
    console.log ("inside my func 2")
}

function myFunc(a) {
    ~~console.log(a)~~
    a();
}

myFunc(myFunc2);
```

हमने myFunc में input तरा accept किया एक function accept किया and function को call कर दिया

a()

So जब भी आप function accept करो तो एक convention है a के जगह callback लिखो and a() जो call किए वहाँ भी callback ही call करो

Output

inside my func2

```
function myFunc2(){
    console.log("inside my func 2")
}
function myFunc(callback) {
    callback();
}
myFunc(myFunc2);
```

Note: → अगर किसी function में parameter pass किया myFunc(a) and call करते वक्त कोई argument pass नहीं किया Like myFunc() then आपको undefined error आएगा या output में undefined print हो जाएगा

[Function returning Function]

54 Topic

हमने एक fn बनाया myFunc() जो return में array ले रहा है, string भी लेता है and number भी कभी ले लेता है return में then हमने constant ans बनाया and myFunc(); को उसमें assign कर दिया फिर console किया ans को।

```
function myFunc(){
    return 1;
}
const ans = myFunc();
console.log(ans);
```

return में 1 कर रहे हैं है So ans में 1 ही जाएगा Becos ans में myFunc(); ही assign हो रहा है then console में ans की value print हो रही है।

Output

1

But क्या भी return एक function कर सकता है।

Yes Bilkul कर सकता है।

```
function myFunc () {
    function hello () {
        console.log ("hello world)")
    }
    return hello;
}
const ans = myFunc ();
ans ();
```

→ इसने एक function hello ही return कर दिया

→ अब ans मे myFunc () assign हो रहा है and myFunc () तो function hello return कर रहा है।

यानी ans मे एक function ही (hello) assign हो रही है। So ans भी एक function बन जाएगा

→ And अब हम ans को console भी कर सकते है। और function बन ही गया है तो call भी कर लिया।

Output

hello world

इसे High order function भी कहते है जो as a input function लेगा उसे call कर सकता है and return कर सकता है function

[Array Important method]

55 Topic

// foreach

// map

// filter

// reduce

अब array का method है, तो एक array लेना ही पड़ेगा।

```
const numbers = [4, 2, 5, 8];
```

* अब एक function बनाते हैं जो इस array का first element लेगा और first वाला index पे है लेगा and उस index को print कर देगा।

Note → 2 से multiply करके print करेगा।

So fresh program

```
const numbers = [4, 2, 5, 8];
function multiplyBy2(number, index) {
console.log("index is", index);
console.log(`${number} * 2 = ${number * 2}`);
}
multiplyBy2(numbers[0], 0);
```

यहाँ हमने 0th index पे जो number है pass कर दिया and (4) number का index जो 0 है वो भी pass कर दिया तो

4×2 = 8 output आएगा ⟶ 8

So एक की तो कर दिया suppose array बहुत बड़ा है and सभी
हमें index के लिए करना है, तो thing is hard.

So हम for method को करेंगे

```
const numbers = [4, 2, 5, 8];

function myFunc (number, index) {
  console.log (` index is ${index} number is ${numbers}
}

for (let i=0; i< numbers.length ; i++) {
    myFunc (numbers[i], i);
}
```

Output

index is 0 number is 4

index is 1 number is 2

index is 2 number is 5

index is 3 number is 8

सारे element में बारी-बारी से पहुंचने के लिए for loop चलाया then console (i)
भी कर सकते थे उसमें 0, 1, 2, 3 print हो जाता So i की value
को my func में pass किया

But हमें Loop चलाने की जरूरत नहीं है

Note→ Callback क्या होता है?

A:- जब कोई f^n as a input दूसरा f^n को argument में लेता है and call कर देता है तो वह callback होता है।

```
const numbers = [4, 2, 5, 8];          → array

function myFunc (number, index) {      f^n c
  Console.log (`index is ${index} number is ${number}`)

numbers.forEach (myFunc);              → myFunc
```

इस Program में एक array है एक f^n है and एक myfunc है।

for each क्या करेगा जो हमने for Loop लगाकर काम किया न ये वाला काम for each खुद कर देगा।

for each myfunc को बार-बार call करेगा and myfunc तो number array में पड़े कर रहा है।

जब पहली बार call करेगा तो array के first element को and usके index को pass करेगा। ये खुद pass करेगा हमें नहीं करना

Output

index is 0 number is 4

index is 1 number is 2

index is 2 number is 5

index is 3 number is 8

for each को as a for Loop समझो Loop की तरह समझो

हम ये approach भी use कर सकते हैं,

```
const numbers = [4, 2, 5, 8]

numbers.foreach (function (number, index) {
    console.log (' index is ${index} number is ${number}
});
```

ये हमने anonymous function बनाया है। anonymous → जिस fn का कोई नाम नहीं होता है।

Task:- const numbers = [4, 2, 5, 8];
में foreach लगाकर हर number को 2 से multiply करना है।

A:→

```
const numbers = [4, 2, 5, 8];
numbers.foreach (function (number) {
    console.log (number * 2);
})
```

output
8
4
10
16

Note → याद रखे forEach क्या करेगा
forEach के bracket में जो करेंगे वो forEach के पहले जो भी array होगा उसके each element के लिए कर देगा।
foreach call back ही करता है and input में argument लेने का काम करता है।

Realistic Example of for each

एक array है जिसमें बहुत सारे object हैं। हमें हर object के firstName को print करना है।

```
const users = [
  {firstName: "harshit", age: 23},
  {firstName: "mehit", age: 21},
  {firstName: "nitish", age: 22},
  {firstName: "nalin", age: 20},
users.forEach(function(user) {
  console.log(user.firstName);
});
```

anonymous fn as a input user fn को दिया

But यही काम हम for of से भी कर सकते हैं। for each पुराना है for of नया है। so ज्यादा use for each हुआ है।

```
for (let user of users) {
  console.log(user.firstName);
}
```

---

for each को arrow fn जैसा भी use कर सकते थे।

```
users.forEach(user) => {
  console.log(user.firstName);
})
```

[Map Method]

56 topic

पहले तो map method array का method है so array बनाना पड़ेगा

ये foreach जैसा ही है so map method भी एक f^n को as a input

callback f^n लेगा।

Note:- callback f^n हम बाहर भी बना सकते हैं and map( ) के अंदर

भी यानी bracket के अंदर

Map Method:→ ये एक नया array बना देता है

```
const numbers = [3,4,6,1,8];
```

map method array को देता है so एक array लिया

```
const square = function (number) {
    return number * number;
```

ये f^n हमने बनाया जो square करता है। हम callback बनाना चाहते हैं but अभी ये f^n को बाहर ही रख रहे हैं।

```
const squareNumber = numbers.map (square);
    console.log (squareNumber);
```

ये पूरे map f^n को const में रखा।

map method हर number को लेगा square करेगा and new array बनाके डालता जाएगा and ये array const squareNumber में जाएगा उसे console कर दिया।

Output = { 9, 16, 36, 1, 64 }

Note अगर square fn कुछ return नहीं करता Let suppose only console करता तो console value print हो जाता है but map method को कुछ मिलता नहीं and वह by defaut undefined print करता।

```
const numbers = [3, 4, 6, 1, 8];
const square = function (numbers) {
  console.log ( numbers * numbers);
const squareNumber = numbers. map(square);
console. log (squareNumbers);
```

Output

9
16
36
1
64

[undefined , undefined, undefined, undefind, undefined]

Note! → So अगर आप map function use करते हैं, तो callback fn में return भी जरूर करें

प्रश्न function map के अंदर

```
const numbers = [3, 4, 6, 1, 8];
const squareNumbers = numbers.map(function (number) {
        return number * number;
    });
console.log (squareNumber);
```

हम arrow fn भी लगा सकते थे →

```
const sqNum = numbers.map ((number) => {
        return num*num;
    });
console. log (sqNum);
```

Realistic example of map method हमें एक array दिया है उसके firstName का array print करना है।

```
const users = [
    { fN : "harshit", age: 23 },
    { fN : "mohit", age: 21 },
    { fN : "nitish", age: 22 },
    { fN : "gaurav", age : 20 },

const userNames = users.map((user) => {
        return user.firstName;
    });
    console.log(userNames);
```

[Filter Method]

51 Topic

Task → एक number का array दिया है and मुझे वाले even या odd number filter करने है।

ये भी as a input callback function लेता है।

But filter का callback function हमेशा return करेगा Boolean value i.e true or false

```
const numbers = [1, 3, 2, 6, 4, 8];
```
→ array दिया है

```
const isEven = function (number) {
    return number % 2 == 0;
}
```
→ एक f^n है जो para मे number ले रहा and return boolean कर रहा है।

```
const evenNumbers = numbers.filter (isEven);
console.log (evenNumbers);
```
→ final evenNumbers f^n const बनाया जिसमे numbers मे filter कर रहे है।

filter numbers के हर एक element को check करेगा जो boolean मे true return करेगा उसका एक नया array बना देगा।

Output

[2, 6, 4, 8]

Note- filter method का callback f^n हमेशा true या false return कर रहा होगा।

[Reduce Method]

ये array का method है।
ये as a input callback fn लेता है.
Callback fn आप अलग से बना सकते हैं reduce के अंदर भी बना सकते हैं
and arrow fn भी बना सकते हैं।
अलग से बनाओगे callback fn तो anonymous भी हो सकता है e.g.
चारों दिखते जैसा ही होगा map, filter, जैसा।

AIM:- एक array दिया है उसके सारे element को sum करना है।

```
const numbers = [1, 2, 3, 4, 5, 10];

const sum = numbers.reduce ((accumulator, currentValue) =>
        return accumulator + currentvalue;
3);
```

देखो reduce method का accumulator 0th index value लेगा यानी 1
and current value 1st index value लेगा यानी 2
and return करेगा 3 (1+2)
ये first call हुआ।
2nd call में accumulator का value 3 and current value भी 3 and return
6 so on...

// accumulator	currentValue	return
1	2	3
3	3	6
6	4	10
10	5	15
15	10	25

output = 25

Realistic example of reduce

पीछले example में accumulator start हो रहा था 0th index यानी 1 से

But अगर हम चाहते है की हम खुद से starting value दे

accumulator को then

```
const sum = numbers.reduce((accumulator, currentValue) =>
    return accumulator + currentValue;
    3, 100);
```

100 + 1

acc current

Output = 125

```
const userCart = [
{productId: 1, productName: "mobile", price: 12000},
{productId: 2, pName: "laptop", price: 22000},
{productId: 3, pName: "tv", price: 15000},
```

aim: flipcart जैसे app में हम cart में product add करते हैं न तो total amount कैसे दिखाता है।

→ एक array जिसमें कई object डाला

```
const totalAmount = userCart.reduce((totalPrice, currentProduct) =>
    return totalPrice + currentProduct.price;
    3, 0)
console.log(totalAmount);
```

→ हमने initial value totalPrice को 0 रखा है and object से price को currentProduct.price से लिया है बल्कि कैसे बताते है की object के कौन value use करने है।

totalPrice	currentValue	return
0	{ }	12000
12000	22000	34000
34000	15000	49000 → output

पहली बार totalPrice 0 होगी and current value में पूरा का पूरा object ही आया होगा and उस object से price fetch किया and return में 0 + 12000 = 12000 हुआ so on...

[Sort Method]

50 Topic

* Sort method original array को भी change कर देता है।

```
const numbers = [5, 9, 1200, 400, 3000];
```

इस array को अब sort करो तो क्या करोगे

[5, 9, 400, 1200, 3000 (Asc)
3000, 1200, 400, 9, 5 (Des)]

Debouncing
Throttling
HSL colors

```
const numbers = [5, 9, 1200, 400, 3000];
numbers.sort();
console.log(numbers);
```

Output → [1200, 3000, 400, 5, 9]

ये wired result इसलिए आया bcos JavaScript इस array को as a number sort नहीं कर रहा बल्कि पहले इसे string बना रहा then sort कर रहा है।

ASCII Value

" 0 : 48
" 1 : 49
" 2 : 50
" 3 : 51
" 4 : 52
" 5 : 53
" 6 : 54
" 7 : 55
" 8 : 56
" 9 : 57

```
numbers.sort();
console.log(numbers);
```

[1200, 3000, 400, 5, 9]

ये

5 - 53
9 - 57
1200 - 49
400 - 52
3000 - 51

Note- 1200 वाले देखता है 1 देखता है 1st letter

यानी 1200 में 1 का 49 होगा इस तरह वे होगा

Actual Sort Program

```
const numbers= [5, 9, 1200, 410, 3000];
numbers.sort ((a,b) => {
    return a-b;
});
console.log (numbers);
```

sort (a,b) → यहां हमने callback fn पास किया and return a-b किया

a-b ही सब कुछ कर रहा है। हम 2 number को example लेते हैं

1200, 410

a-b ---> 790.

a-b ---> positive (greater than 0) ---> b, a set हो जाएगा

410, 1200 set हो गया

5, 9

a-b ---> -4

a-b ---> negative (less than 0) ---> a, b set होगा

[5, 9]

ये ascending में sort हुआ अगर descending में sort करना है, तो

return b-a;

output

[3000, 1200, 410, 9, 5]

Sort Real World Example

```
const product = [
{ productId: 1, productName: "p1", price: 300 },
{ pId: 2 , pName : "p2", price : 3000 },
{ pId: 3 , pName : "p3", price : 200 },
{ pId: 4 , pName : "p4", price : 8000 },
{ pId: 5 , pName : "p5", price : 500 },

const lowToHigh = products.slice(0).sort((a,b) => {
    return a.price - b.price
});
console.log (products);
```

Slice(0) इसलिए लगाया है कि
हमारा पूरा array न change हो।
only price change हो इसलिए

Output

pId: 3 pN = p3 price 200
300
5000
3000
8000

find Method

60 Topic

AIM: हमें कुछ string दिया है But हम find करना चाहते हैं उस string को जिसकी length 3 है।

```
const myArray = [ "Hello", "cat", "dog", "Lion" ];

function isLength (String) {
    return String.length === 3;
}
const ans = myArray.find (isLength);

console.log (ans);
```

ये find method array के हर element को check करेगा जैसे ही पहला 3 letter वाला string मिलेगा ये उसे return करेगा and control बाहर निकल जाएगा

example $\boxed{\text{cat}}$

But अगर cat → catty लिखा तो
Dog output आता

अगर Doggy लिखा, तो undefined आता क्योंकि कोई 3 वाला नहीं मिला

Real example of find

```
const users = [
  {userId: 1, uN: "harshit"},
  {userId: 2, uN: "harsh"},
  {userId: 3, uN: "nitish"},
  {userId: 4, uN: "mohit"},
  {userId: 5, uN: "aaditya"},
];
const myUser = users.find((user) => user.id === 3);
console.log(myUser);
```

12:49

[Every Method]

Let suppose किसी एक array में सारे - के - सारे element even हैं ये check करने के लिए हम even method का use करते हैं।

Every का मतलब होता है हर एक यानी हर एक method एक task या goal को fulfill कर रहा है या नहीं ये check करने के लिए every method

```
const numbers = [2, 4, 6, 8, 10];

function isEven (number) {
    return number % 2 === 0;
}
const ans = numbers.every (isEven);
console.log(ans);
```

इसे call back भी boolean return करेगा and every method भी boolean return करेगा

But callback हर element के check के बाद return करेगा and every method सारे element को check करता रहेगा जैसे ही odd मिला वहीं रुक जाएगा।

Realistic example of

हमें check करना है की हमारे सारे product का price 30000 से कम है या नहीं।

```
const userCart = [
    {pId: 1, pName: "mobile", price: 12000; 3},
    {pId: 2, pr : "laptop", price 22000, 3},
    {pId: 3, pN: "tv, price 24000 3},
]

const ans = userCart.every((cartItem) => cartItem.price < 30000)
console.log(ans);
```

true

[Some Method]

Use → Like हमें check करना है कि given array में कोई भी ऐसा ~~method~~ element है जो odd है।

```
const numbers = [3, 5, 8, 9];
const ans = numbers.some((number) => number % 2 === 0);
console.log(ans);
```

Real Example

पिछले वाले Product में check करो कोई भी product है जिसका price 100000 से ज्यादा है कि नहीं।

63 Topic

[Fill method]

Goal → आपको array बनाना है उसे fill करना है with -1, -1 -1
length है 10 array की कैसे करेंगे।

Use (1) → statement same → fill direct

```
const myArray = new Array(10).fill(0);
console.log(myArray);
```

fill करना है 0 से

myArray constant बनाया

newArray जिसका length 10 है

console करवा दिया।

output → [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Use (2) → हमारे पास array है [1, 2, 3, 4, 5, 6, 7, 8];
हम चाहते हैं 3, 4, 5 को 0 करना

Value, start, end

```
const myArray = [1, 2, 3, 4, 5, 6, 7, 8];
myArray.fill(0, 2, 5);
console.log(myArray);
```

(1) value 0 उसका की हमें क्या fill करना है
(2) start कहाँ से करना है उसे वाले 3 का index 2 है
(3) end कहाँ करना है 5 तक But एक index आगे तो तक्की 5 भी हो जाए तो 6 का index 5 तो

[Splice Method]

64 Topic

Splice → Array के बीच में से कुछ element delete करना है, insert करना है, तो splice का use होता है। ये original array को change करता

① **delete**

```
const myArray = ['item1', 'item2', 'item3'];
myArray.splice('1','1');
console.log(myArray);
```

→ यानी item2 को delete करना है So start index 1 then बताया delete कितने item करना है So एक करना है, तो 1

Output

["item1", item3"]

② **insert**

```
const myArray = ["item1", "item2", item3"];
myArray.splice(1, 0, "inserted item");
console.log(myArray);
```

→ * index 1 पे insert करना है
* 0 item delete करना है मतलब नहीं करना है delete
* value inserted item है

Output

["item1", "inserted item", "item2", "item3"]

③ **insert - delete साथ में कैसे करे**

```
const myArray = ['item1', 'item2', item3'];
myArray.splice(1, 2, "inserteditem1", "inserted item2");
console.log(myArray);
```

→ ① कौन index से start होना है 1st index
② कितने item delete करने है ②
③ value insert करा