Math object Method

12. Math.sin( ):

→ sin method is returns the sine (a value between -1 and 1) of the angle x (given in radians) if you want to use degress instead of radians, you have to convert degrees to radians

// Angle in radiusen = Angle in degrees x PI / 180

Ex:-

let math19 = ~~math~~ Math.sin(30 * 3.14 / 180);
PI Round deg

Console.log(math19); //

13. Math.cos( );

→ returns the cosine (a value between -1 and 1) of the angle x (given in radians)

→ if you want to use degrees Instead of you have to Convert degrees to radians.

→ Angle in radians = Angle in degrees x PI / 180;
3.14 ble

Ex:-

M20 = Math.cos (60 * 3.14 / 180);

Consolelog (M20); //

Syntax

14. Math.log() let x = Math.log(151);
Math.log2() let y = Math.log2(15.1);
Math.log10() let z = Mth.log10(102);

→ the returns the natural logarithm of x
the natural logarithm returns the time needed to reach a ~~certain~~ certain level of growth.

(બધા log નો output આપણા આવશે in same number but ક્યારે તેનો use method કરણ છે? આવે work થશે.)

Ex :-

```
M22 = Math.log2(11);
Console.log(M22);
```

આપેલ Number method એ દ્વારા value return કરે છે જેમાં ને Number to string માં convert થઈ શકે છે in all methods

Number Methods :-

1. toString(); :- Returns a number as a string

Ex:-

```
→ let Num1 = 155;
  Console.log (Num1);                 // 155
  Console.log ( typeof ( Num1 ) );    // Number

→ let Num2 = Num1.toString();
  Console.log (Num2);                 // 155
  Console.log ( typeof (Num2) );      // String

→ let Num3 = (155).toString();
  Conlog (Num3);                      // 155
  Conlog ( typeof ( Num3) )           // string
```

2. toExponential ( ) :- (type change in String)

→ returns a string, with a number rounded and written using exponential notation.

toExponential number ના output કરીશું

(x = 1500000
→ 1.5e+6 ← આ રીતે e value આવે છે જેમાં શરૂઆતમાં પ્રથમ 1 point બાદ ઉંડે 5 અને 5 શૂન્ય મળ્યા.)

→ toExponential method એ કોઈપણ decimal number ને Exponential number માં convert કરે છે

* આ રીતે string માં convert કરે છે.

Ex :-

```
let x = 2550000 ;
 x2 = x.toExponential( ); // 2.55e+6 ;
```

(2.55e+6 → 2550000)

આને toExponetial (X) :
અહિં X ની જગ્યાએ કોઈ number નાખવામાં આવે તો
output માં તે Point પછી એટલી જ value count કરે છે

Ex :-

```
x = 2550000 ;
x2 = x.toExponential (1); // 2.6e+6 ;
      ↓
x2 = x.toExponential (2);
        // 2.55e+6 ;
```

2.55e+6
+1
2.6e+6

3. toFixed ( );

Returns a numbs written with a specifi. number of decimal.

```
let   y = 1234.82468
Console.log (y);        // 1234.82468
                           ↓
                         (+1)  if 1234.24567
let  y2 = y.toFixed ( );   ↓     1234 < NO (+1)
Console.log (y2);       // 1235          >5

let  y3 = y.toFixed ( 2 );  Point પછીની એ સંખ્યા
Console.log (y3 );      // 1234.82
```

to fixed નો Bill system માં round
Bill amount કરવા માટે ઉપયોગ થાય છે

4. toPrecision();

Returns a numbr witten with a specified length.

→ precision method દ્વારા Bill amout ને point પછી ને કેટલી number count કરવા છે તે ના માટે યુઝ થાય છે!

Ex:-

```
let z = 1.5648

z1 = z.toprecision(3);   } 1.5648
Console.log(z1);
```

અહીં output માં base value માંથી total 3 number જ show થશે.

→ 1.56

(+1 > 5)

```
z2 = z.toprecision(2);   // 1.5648
Console.log(z2);
```

1.6

+1 (6>5)

```
→ x = z.toprescision(1);   // 1.5648
Console.log(x);
```

1+1

becouse (.5>5)

2

Value ને roundoff કરવા માં આવ્યા છે

5. Valueof():

→ Return a number as number
String as string

Valueof syntax :-

```
x = 12345;
x2 = "12345";

Console.log (x.Valueof()); // number
Console.log (x2.Valueof()); // string
```

આવી રીતે x અને x2 ની typeof check કરાવી અને
તો તો x = number જ રહે
x2 = string જ રહે.

15/2/24

function

basic function syntax

functionname

=> function name(Params) {

keyword

function body

}

Pass values Parameters

// call function
name(param);

-: Note :-
function ને નીચા Declaration પહેલા પણ call કરી શકાય છે?

---

return type argument વગરના function માં તેને Variable માં કંઈ save કરાવીને check કરી શકિયે છીએ.

// Anoymous function //

Anoymous function :
એટલે કે એવું function કે જે function ને બોલાવવા આપે ઇચ્છિ તેવા function Name આપવામાં આવ્યું ના હોય તેવા function ને Anoymous func કહેવાય છે.

=> -: Syntax :-

function (Params) {
    function body
}

-: Ex :-

function (a,b) {
    Console.log(a+b);
}

call

Anonymous function ની call કરવા માટે તેને Variable માં store કરવામાં આવે છે ત્યારબાદ જ તે call કઈ રીતે થાય છે? Ex:- below.

Ex:-

```
let x = function (a,b ) {
        console.log(a+b )
    };
// call function
Console.log ( x(10,5 )) // 15
```

// IIFE //
(function)

immediately Invoked function Expression

What is IIFE :-

in simple way :- IIFE એ છે જે function ને Declaration કરતાજ તે run થાય તેને IIFE function કહેવાય છે જેને call કરવાની જરૂર પડતી નથી.

IIFE function ને parentheses() ની બહાર લખવામાં આવે છે ત્યાર બાદ New parentheses માં ફરી Pass Values લખવામાં આવે છે

-: Syntax :-

Start ↓   Output

```
( function   sub   (a, b) {
  keyword    name  params

      Console.log (a - b)
})(10, 5);
```

↑ end   ↖ Pass Values.

Output: 5

→ આખું આ રીતે IIFE function ને Declaration કરવામાં આવે છે? ને પુરા થયા બાદ તેને Value પણ સાથે જ આપવામાં આવે છે!

Second way to declaration

with Variable :-

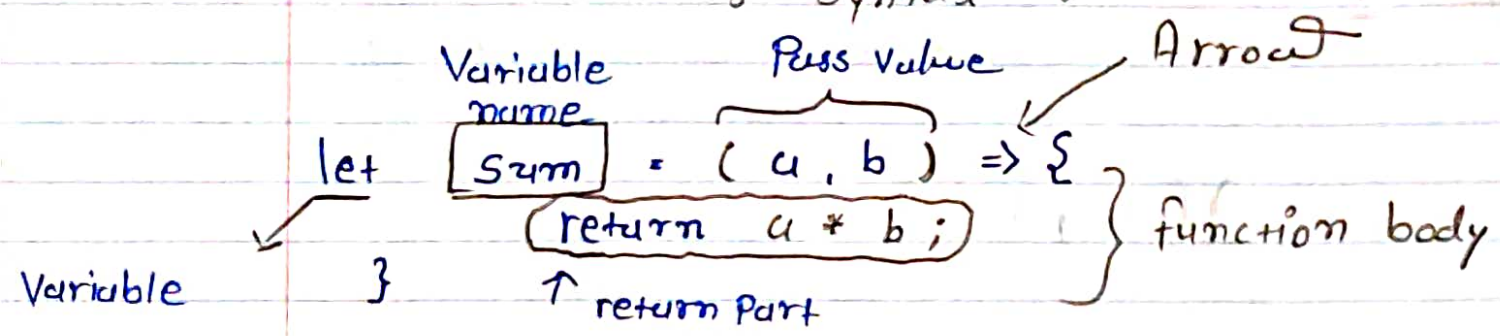
```
let x = ( function add (a, b) {
              return (a + b);
        })(10, 4);

Console.log ( x ); // 14
```

using Whit Variable.

Arrow function

:- Syntax :-




```
Console.log ( Sum (10 , 5 ) );  // 50
```

→ Arrow function માં function key word use કરવાની જરૂર નથી તેના બદલે Arrow ની mee પહેલા Declaration કરી શકાય છે?

→ Arrow function ની use કરવા અને Declaration કરવા base function કરતા વધારે છે?

Arrow function ને single line માં પણ લખી શકાય જો one argument વાળા function હોય તો

Ex:- let Sum = (a , b) => a * b ; // 50
(10, 5)
Sem out

arrow function ને single line માં લખવામાં આવે તો Return શબ્દ લખવાની જરૂર પડતી નથી

Function in Constructor

Constructor નો use થતો નથી

→ function માં constructor બનાવવા માટે new key coord નો use કરવામાં આવે છે અને ત્યાર બાદ તેની અંદર argument અને Pass paramet કરે અને double inveted Commas માં લખવામાં આવે છે?

→ જેમાં પહેલા બે તો Passing Parameters કહેવાય છે અને thred number પર Return એ Argument કહેવાય છે!

-: Syntax :-

```
let Add = new Function ("a", "b", "return a+b");
```

Variable — Keyword — Parameter — Argument

```
Console.log (Add (10, 30)) // 40
```

17/02/24

— x —

2 code

— x —

Solving

// Leet code //

1 * Return Hello word program

```
function CreateHelloword ( ) {
    return function ( ) {
        " Hello word " ;
    }
};

Let f = CreateHelloword ( );
Console.log ( f ); // Helloword;
```

2 * Count :-

```
function Counting ( n ) {
    let num = n ;
    return function ( ) {
        Console.log (num );
        num ++ ;
    }
}

let Comter = Counting ( 10 );

Counter ( ); // 10
counter ( ); // 11
counter ( ); // 12
```

} call three times

// functions are object //

Javascript માં function બનાવ્યા બાદ આપણે કોઈ input parameters નક્કિ કર્યા બાદ એક કે તેથી વધુ Parameters તેમા Pass કરવામાં આવે તો તે input તરીકે Accept કરે છે. પરંતુ જે આપણે

→ Arguments.length method નો use કરીએ તો આ method દ્વારા આપણે function માં કેટલી વાર Param. Pass કરવામાં આવ્યા છે તે નુ number show કરે છે?

Ex :-

```
function countParam (a, b, c, d) {
        return arguments.length ;
}

// call function
countParam ( 2, 3, 5, 6 ) // 4
```

1 2 3 4 ✓ Parameters.

→ Argument object :-

Argument એ જ્યારે parameteres input તરીકે લઈએ છીએ ત્યારે તે તેને array ના form મા save કરે છે. જેની તેમાં એક થી વધુ arguments save કરવામાં આવે છે.

→ Ex :-

```
function sum ( ) {
    let sum = 0;
for (let i = 0; i < argument.length ; i++)
{   sum = sum + argument [ i ]; }
    return sum ;
}
```

Call function sum (2 , 3 , 4 , 6 , 8) // 23

આમ sum function માં એકથી વધુ arguments pass કરી શકાય છે. જેમાં parameters બનાવવાની જરૂર પડતી નથી.

=> Function Rest Parameters :-

The rest Parameters (...) allows a function

-> (...) rest Parameters નો use three dote ની use કરીને decler કરવામાં આવે છે)

Ex :-

```
function total (...args) {
    let total = 0;
for(let i=0; i < args.length ; i++)
 {    total = total + ~~args.length~~; args[i]; }

 return total;
}

Console.log(total (1, 2, 3, 4)); // 10
```

* Generator Function

generator function નો use * star નો ~~simbol~~ ~~symbol~~ Symbol ના મદદ થી થાય છે. function ના પછી ~~fun~~ Star મૂકવામાં આવે છે.

અને generator function માટે yield keyword નો use કરવામાં આવે છે.

yield : આ keyword ની મદદ થી function માં એક થી વધારે વાર Return કરવા માટે use કરવા માં આવે છે. yield keyword નો use return જેવુ જ work કરે છે!

Ex :-

```
Function* Countes ( ) {
    yield 1 ;
    yield 2 ;
    yield 3 ;
    return 4 ;
}
```

અહીં જ્યારે First time counts function call કરવામાં આવે ત્યારે તે 1 print કરે છે! જ્યારે second time counts કરે 2 print કરે છે.

Ex :- (જ્યારે counts ને 5 વખત call કરવામાં આવશે ત્યારે તે undefined Return કરશે)

Countes

```
( ~~Countes( )~~ ) ; // 1
" ( ~~Countes( )~~ ) ; // 2
" ( ~~Countes( )~~ ) ; // 3
" ( ~~Countes( )~~ ) ; // 4
```

Call the generator

```
console.log ( counter.next().value ); // 1
console.log ( counter.next().value ); // 2
console.log ( counter.next().value ); // 3
```

આમ જેમ આપણે આ function call કરતાં જઈશું તેમ તેની અંદર રહેલ yield અને Return વાળી value print કરે છે!

// String it's method //

> Note :- String નુ વેરિયેબલ એ એક Array જેવુ હોય છે ને index પર Value Return કરે છે।

⭐ ⇒ .length method

```
{
let strt = " Hello Javascript ";
console.log( strt.length ); // 16
}
```

* અહિ .length method એ string માં store કરેલી Value ના space પણ Count કરીને આપે છે।

⭐ ⇒ .trim() method

→ .trim() એ string ની અંદર રહેલી string ની left side અને Right side માં વધુ unwanted spue ને remove કરે છે?

→ .trimStart()

એ String માં only એની Strat એટલે કે left side પર space Remove કરે છે?

→ .trimEnd()

એ String માં only End એટલે કે Right side પર space ને Remove કરે છે!

★ -> split method ()

let stxt = " This is Javascript";

=> split method એ જ્યારે string સાથે use કરવામાં આવે ત્યારે તે string ને array ના form માં Print કરીને આપે છે? તે string માં રહેલા એક એક word ને array ની જેમ index પર ~~sdo~~ Stroe કરે છે।

(1) -> split() method માં જો કોઈ Parametr Pass કરવામાં આ string સાથે અને જો તે કોઈ ~~ap~~ alphabet ~~box~~ હોય તો તે તે alphabet ને તે String માંથી Remove કરીને Return કરે છે!

(2) => અને જો તેમાં માત્ર ત્યાંથી જ એક space મુકવામાં આવે તો તે string ના અંદર જ્યાંથી space હોય છે. ત્યાંથી ~~તેને~~ તેને 'devied' કરી દેય છે.

Ex:-

(1)

```
Console.log (stxt.split(a)); // Remov "a"
```

out put

-> ["This", "is", "J, V script";

Ex:-

```
Consolelog (stxt.split('space');
```

output

-> ["This", "is", "Javascript"]

★

→ toUpperCase() method

→ toUppercase method એ આપણી string ને uppercase માં convert કરીને show કરે છે!

★

→ toLowerCase() method

→ toLowerCase() method એ આપણી String ને lower case માં convert કરીને show કરે છે

★ // PadStart & PadEnd method — આ મેથડ એ જરૂરી આંકડા ને છુપાવવા માટે use કરવામાં આવે છે!

—: Syntax :—

```
let num = "12345"; // use in String
let PadNum = num.PadStart(10, "X");
                .PadEnd(10, "X");
Console.log(PadNum); XXXXX12345
```

End out put // 12345XXXXX ← End

→ Pad method માં string નો સીધો use કરવામાં આવે છે અને તેમાં જે number આપવામાં આવેલા હોય છે કે number માં Variable ના input માં જેટલા number ઘટતા હશે તેના બદલામાં તે "X" Print કરશે

→ total 10 અંકમાંથી જો 5 હશે તો બાકી 5 માં X આવશે.

હાલે આ બધી method all ways new String Return કરશે.

* // Slice ( ) ;

→ Slice method માં input તરીકે Parameters Pass કરવામાં આવે છે.

Ex :- → String ના index Value :-

let txt = " H e l l o T h i s a J a v a s c r i p
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 . . .

let Newtxt = txt. slice (2 , 8)

Start index — End index

Here the last index letter is not show in console so it's only Print to " 7 " index

⇒ Sclice ( ) માં જો input Pass તરીકે જો આપતા one number હોય તો sclice (10) તો 0 થી લઈ ને 10 સુધી index Value ને Print કરે છે.

⇒ જો તે negetive હોય તો -1 સુધી Print કરશે.

→ ~~Sclice (nagetive, positev)~~

→ Sclice (nagetive , Nageteive)

ઉપર મુજબ Value પણ Pass કરી શકાય છે?

Ex :- H e l l o t h i s i s a J S .
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

. Sclic (-10 , -4) ;

Print

Strting Point of string from last side

Ending Point of string

SubString() :

subString() method એ slice જેવુ જ work કરે છે

But

substring method માં જો Nagetive Value pass કરવામાં આવે તો તે ક્રમિકા તેને 0 Count કરે છે અને string ને 0 થી Point પર હોય ત્યાંથી શરૂ થવાના તો End કરશે.

Note :-

substring (-20, -10) બંને Value nagetive હોય તો તે ને ક્રમિકા તે 0 count કરશે અને console માં કંઈ Print કરશે નહિ.

SubStr ()

substr() method is deprecated now

substr() method એ string ને slice કરીને Print કરવામાં use થાય છે. But substr method માં End Point ની જગ્યા એ આપણે કેટલા word ની length સુધી Print કરવુ છે તે લખવામાં આવે છે

Ex :-

txt = "Hello my name is Ram"

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

txt.subStr (3 , 10) → અહિં તે total 10 word Print કરશે index ને ધ્યાન માં લેશે નહિ.

↓ This is a length of print string

★ Concat();

Concat method એ બે string ને જોડવાનું work કરે છે

Ex:-

```
let txt = Hello ; }
let txt2 = word ;
```

```
Console.log ( txt.concat (txt2));
```

↳ Hello word ;

txt.concat (txt2, " ", Hello, Hill)
આવીને New string પણ Add કરી શકિયે છીએ

---

txt = ' H e l l o W o r d i t S J a V a scri
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 :-

⇒ CharAt();

* CharAt() method એ આપેલા Pass કરેલ index પર રહેલ character Value ને Return કરે છે & console પ

Ex:-

```
text = txt.charAt ( 5 ); // W
                  (12 ); // J
```

Charcode();

=> આ method એ તેમાં આપેલ index number પર રહેલ character ની ascii Value Return કરે છે

Ex:- txt = t = index is 10

```
text = txt.charcode(10); // 116
       charcode("); // 79
       charcode(0); // 48
```

—x—————x—————x—————x—————x~~x

આ method એ case-sensitive છે.

replace();

-: Syntex :-

```
txt = "Hello word";
New_txt = txt.replace("Hello", "Hill");
// Hill word;   } output
```

("Hill" = New Replace word)

=> replace method એ string ની અંદર letter કે word ને બીજા શબ્દ replace કરવા માટે use થાય છે.

Note:-
replace method એ માત્ર string માં એક જ વાર word કે letter ને Replace કરે છે. જ્યારે તેને first time એ match થાય ત્યાર બાદ એ word બીજી વાર હોય તો તેને Replace કરતું નથી.

* જો Replace method ને આપણી વધારે પણ string ની અંદર સુધારા કરવા માટે use કરવી હોય તો તેમાં નીચે મુજબ ની જેમ રીતે કરી શકાય.

→ /word/g Key word ની મદદથી

txt = "Hello my name is Jayan";

New = txt.replace(/a/g , 'A'); ← global keyword

→ આપણે આ Replace string માં જેટલા same letter ને change કરી દેશે.

—x—x—x—x—x—

replaceAll();

આ method માં replace ની જેમજ work કરે છે. but આ માં /g keyword વગર વાર આપણી string word change કરી શકાય છે?

Date object

→ Declaresoin of Date function

⇒ let Dates = new Date();

↳ This object create with new keyword and use.
also This keyword is Return full time date and year value in console.

⇒ output

```
          Date           Hours      second
           ↓               ↓           ↑
Wed   Dec  28    2024   17:50:34   GMT+0530
 ↑     ↑           ↑       ↑       (indian standard Times)
Day   Month       year   minutes
```

→ Date object માં Default current Date and time Value ને console કરે છે

// Date objectને ત્રણ રીતે create // કરી શકાય છે?

```
let x = new Date();  // Default

                        ┌──── Date ────┐ ┌─ Time ─┐
x = new Date("January  01, 2023, 9:30:00");
          // custome time and date add ; in String

               year month Date  Hours minutes second
x = new Date(2018, 11, 24,  10, 33, 30, 0);
             └── Date ──┘   └── Time ──┘  ms second
```

⇒ in Previous Country GMT -

x = new Date (99 , 11 , 24);

↓ year ↓ Month ↓ Date ↳ time is null = 0;

⇒ :- Method of Date :-

⇒ Fri Feb ૨૩ ૨૦૨૪ , 04:58:14:589 GMT+0530 (indian stendd Time)

હવે આ full time object માંથી જો આપણે અમુક ટુકડા ની જરૂર હોય ત્યારે નીચેના method use થાય છે જેમ કે only Time, Date , Time and Date.

create Date object :-

⇒

```
let xTime = new Date ( );
```

1. `Console.log ( xTime . toDatestring ( ) );`

⇒ toDateString() માં only Fri Feb 23 2024 show થશે.

2. `Consolelog ( xTime . bISOString ( ) );`

⇒ toISOString () માં only 2024-02-23T11:45:52.726Z show

↓ year ↓ month ↓ Date ↓ Time but it's →4TC Time←

This method is Print only UTC Time.

3. Console.log (xTime.toLocaleDateString());

toLocaleDateString() આ only Date ↳ 2/23/2024 show
↑ month, ↑ Date, ↑ year

4. Console.log (xTime.toLocaleString());

toLocalestring() આ only Frist Date and second Time show કરશે.

Ex:-
→ 2/23/2024 , 5:23:03 PM
Date , time

5 Console.log (xTime.toLocaletimeString());

toLocaletimestring() only LocaleTime show કરે છે

Ex:- 5:24:05 PM <=

6. Console.log (xTime.toUTCString());

toUTCstring only Time and Date show કરશે in 24 hours ની count પ્રમાણે with GMT.

Ex:- Fri, 23 Feb 2024 . 11:56:30 GMT,
↳ it's UTC Time)

7. Console.log (xTime.toTimeString());

toTime string() પણ 24 Hours ની count પ્રમાણે time show કરશે =>locale

Ex:- 17:30:12 GMT+0530 (India standard time)

Date object get Method

let nTime = new Date(); // Base obj

year ↓ , minutes ↓

Fri Feb 23 2024 17:38:43 GMT+0530 (India Sta... Time)

↑ Day ↑ month ↑ Date ↓ Hours ↑ Seco

1. getDate()

Here in getDate() method is take value from base object and show only Date in our console.

nTime.getDate(); // 23

2. getDay()

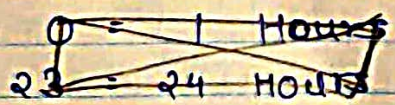
Here in getDay() method is take value from base object and return week days in number between 0 to 6

→ Here 0 = Sunday
6 = Saturday

nTime.getDay(); // 5 becouse it's Friday

3. getHours();

This method is return Hours beHween 0-23 Hours



nTime.getHours(); // 17 Hours

4. getFullYear();
This method is only Return full yeure
Year is 2024 in buse object

5. getMillisecond();
in This method is only Return Three numbs of mseond
⟶ if millisecond is > Three numbs it's Return "0" ~~234~~
Ex :- 234 it's show but 2345 not show it's "0".
Three | Four
becous Mscond is count 0 to 999 only

6.

6. getMinutes()
This method Return only ~~min~~ minutes in numbes ~~is~~
nTime.getMinutes(); // 28

7 getSeconds()

This method Return only ~~second~~ Second Value in Time.

nTime.getseconds(); // 43

8. getMonth()
This method Return month in number. Type

in month
0 = January } Count between 0 - 11
11 = December

nTime.getMonth(); // "1" == February

9. getTime(); // Total of mseconds

getTime() method માં January 01 1970 થી milliscond count થતી હોય છે અને ત્યારથી લઈ ને curent Time નુ Total show કરે છે!

=> nTime.getTime(); // 1708690123345 — This change always

10. getTimezoneoffset();

Timezoneoffset is Return in minutes but it also show defrense between UTC Time and GMT Time.

using "-330" numbr of base.

Date object Set method.

Note:-

⟹ Set method એ Date(); object નો ડાયરેક્ટ change કરતું નથી પરંતુ તેમાં Time, Date, Hours Yeare ની change કરી user પોતાની મનગમતી value ને set કરવા માટે use થાય છે!

⟹ This method takes input from user side.

⟹ set ની કોઈ પણ method માં input આપવા ફરજીયાત મુકવામાં આવે છે અને જો empty રાખો તો તે work કરતી નથી.

અહિં એ કોઈ Parameters Value તેના Base Value કરતા વધુ આપવામાં આવે તો તે તેના Next Day, Time, Yeare, Hour માં સુધારો કરે છે!

—: Syntax only :—

—x———x———x—

and Paremeters of inputes

1. SetFullYear();

Para is : setFullYear(year); ⊛

optional { setFullYear(year, Month);
set full year(year, Month, Date);

2. Set Date();

Peramet is : setDate(it's take Date acoding Month); ⊛

This Date value is impact on month value.

3. SetHours():

⇒ This method takes 0 to 23 Hours Value

Peramets is : setHours (Hours); ①

opes { setHours (Hours, minutes) ;
setHours (Hours, minutes, seconds
Set Hours (Hours, minutes, seconds, Msecond

4. Set Minutes():

This Peramets is : SetMinutes (Minutes); ②

opes { Set Minutes (Minutes, second
Set Minutes (Minutes, seconds, Mse

5. Set Seconds ()

This Peramets is : Set Seconds (seconds); ③

opes { Set seconds (Seconds, Msecond

6. Set Milliseconds():

This Peromets is : Set Milliseconds (Msecon

1000 Ms = 1 Second
60,000 ms = 1 Minetes

7 Set Month( ): it's between 0 to 11 month's

Pass Peramites is : setMonth (Month);
ઉપર } setMonth (Months , Dates value);

8. SetTime( )
it's only Returen Base Time of india
"Jan 01 1970 . 5:30:00 GMT"

" UTC Time "

UTC Time એ universal Time ghour કરે છે એમાં

⇒ UTC Time GMT Time વચ્ચે 24 Hours no gap
ફરિયા છે. જે 530 અને -330 છે આવી રીતે ;
છે।

Note: → UTC Time એમાં get and set Mathod એ same
work કરે છે।

leet code Practice

4 = IV I < V
-1 < 5 TRUE
= V base

6 VI V ≤ I false
+

I = 1
II = 2
III = 3
IV = 4
V = 5
VI = 6
VII = 7
VIII = 8
IX = 9
X = 10

XI = 11
XII = 12
XIII = 13
XIV = 14
XV = 15
XVI = 16
XVII = 17
XVIII = 18
XIX = 19
XX = 20

LVIII

L < V
50 < 5

```
Const romanNum = new Map([
  { ["I", 1],
    ["V", 5],
    ["X", 10],
    ["L", 50],
    ["C", 100],
    ["D", 500],
    ["M", 1000], }
```

Array &

* Object destructring *

object destructring માં object ની Value ની new Variable માં store કરવા માં આવે છે. તેને obj destructring કહેવામાં આવે છે!

```
Const bands = {
    bandName : "one direction",
    Member1 : "Harry styles",
    Member2 : "Zayn",
    Member3 : "Niall Horan"
}
```

Var → Const; obj name → bands; key → bandName, Member1, Member2, Member3; Values → "one direction", "Harry styles", "Zayn", "Niall Horan"

* First Method

→ Syntax :-

```
let { bandName , Member1 , Member2 } = bands ;
```

Variable key word → let; same name as key in obj → bandName, Member1, Member2; Object name → bands

- ઉપર મુજબ જે first let , const , var એમ મુજબ Variable key word લખવામાં આવે છે. ત્યાર બાદ Curly Brackets માં તેને આપણા Variable name જે bandName , Member1 , meber..... etc same as objet key મુજબ લખવામાં આવે છે and last માં object નો name કે જેની Value Variable માં store કરવાની હોય છે. તે લખવામાં આવે છે.

★ console.log માં bandName only લખ-
વામાં આવે તો direct object માંથી તેની Value
print થશે.

Ex :-

```
Console.log (bandName); // one direction
Console.log (Member2); // zayn
```

★ Second Method

—: syntax :-

→ અહીં First method માં આપણે Name ની જગ્યા એ
કોઈ બીજા Name આપવા હોય variables
ને તો તે નીચે મુજબ રીતે આપી શકાય છે.

Val1 in→ onedirection / Val2 in Harry style / Val3 in→ za

```
let { bandName: Val1 , Member1: Val2 , Member2 : Val3
```

↑ Variable Name ↑ ↑ binds

∴ Val1, Val2, Val3 તે માં હવે object ના Value
તેની બીજી colon ની પહેલા આપણે key પ્રમાણે save
થઈ શકે છે.

Now object ની Value ને Print કરવા માટે

```
console.log (Val1); // one direction
Console.log (Val2); // Harry style
Console.log (Val3); // zayn
```

Output

* Thried method with rest peremetrs *

```
let { bandNome, Members1, ...restpop } = bands
                                ↑
                          restperameters
```

↳ :. Frist key અને Value અને
bandNome માં Frist Key Value save થશે.
Member1 માં second key Value save થશે.
અને બાકીની જે Value છે તે restpop માં store થશે

અને restpop ને console કરવામાં આવે તો તેમાં
એ new object return કરશે જેમાં બાકી ની
Value key સાથે show કરશે.

* Array *

Note → Array Distructuring એટલે કે array ની Value ને new Variables માં stroe કરવાની method ને Array Distructuring કહેવામાં આવે છે.

(I) Frist Method :-

:- Syntax :-

```
const fruits = ["apple", "banana", "mongo", "orang"]

let [ Fruits1, fruit2, fruit3 ] = Fruits;

log ( Fruits1 ); // apple
```

:. fruits1, 2, 3 માં array index મુજબ Value Variables માં store થઈ જશે.

Distructring માં Variables ના name જે પણ
આપવા હોય.

② Second Method :-

With rest parameters

let [f1, f2, ...restfruits] = fruits;

f2 Variables માં apple store થશે,
f2 Variables માં banana store થશે,
And
restfruits માં બાકી of Array
Value store થશે અને new Array ret
કરશે.

↓

["mango", "oranye"];

[Array]

What is Array :-

⇒ array is object
⇒ array is reference type Datatype.
⇒ array is oreded of collection of items of elements

★ Array is ~~im~~mutable also we are change array value using it's index value :-

Ex :-

```
let Colors = [ "Red", "Blue", "Yellow" ]
index          [0]     [1]      [2]

Colors[1] = "pink";
```

→ અહીં colors Array માં તેની [1] index પર જે રહેલ Blue value એ chang થઈને હવે Pink થઈ જશે.

★ how chack in javascript is object is Array of not

Note :-

Js માં ઘણા બધી defined કરેલા opretores and variables ની type object હોય છે. જે console માં object show કરે છે! but તે કેર ના type array છે કે કોઈ Map, set etc છે. Check કરવા માટે ની નીચે મુજબ એક્સ્ટ્રા Method છે?

★ Syntax of Method ★

```
Array.isArray( Name of * );
```

// Base Example array //

items = ["item-1", "item-2", "item-3", 3, 5]

→ આપણે આ Array.isArray method માં Pass કરેલ Param એ જો Array ના form માં હશે તો આ આપેલ Method એ "TRUE" Return કરશે નહિ તો એ કોઈ બીજા form હશે તો તે "Flase" Return કરશે.

Ex :- ⇒

```
let Color = [ ];  // This array
let Colors = { 3; // fo object

Console.log ( Array.isArray (Color) ); // True
Console.log (Array.isArray (Colors )); // Flase
```

① Push( ):

Push() Method એ array ની અંદર New element or Value ને Add કરવા માટે use થાય છે.

* Push() Method એ array માં last index પર values ને Add કરવા માટે વપરાય છે!

```
items.Push ("item-4");
```
} Syntax :-

-) અહિ item-4 એ Ex Array માં last index એટલે કે 5 પછી Add કરશે.

② PoP( ):

PoP() Method એ array ની અંદર ની last index પર રહેલ Value કે element ને Remove કરવા માટે use થાય છે

Note:-

POP() Method એ Remove કરેલ element કે value ને Return પણ કરે છે. જે ને New Variable માં store કરીને use કરી શકાય છે.

items.pop(); // This Remove "5";

let popValue = items.pop();
console.log(popValue); // 5 output

(3) unshift():
unshift() એ array માં value of element Add કરવાનું work કરે છે. same as Push() method But unshift method એ array ને start index પર value ને Add કરે છે

items.unshift("item-0"); // આ items array માં 0 ind
પર value Add થશે.

(4) shift();
Shift() Method એ pop() Method ની જેમ Value Remove કરે છે? array ની start index પર થી એને ને પણ Remove કરેલ Value ને Return કરે છે?

items.shift();

⑤ Concat();

Concat() method એ અલગ અલગ array ને Merging અથવા તો એક કરવાનું work કરે છે. જેમાં કોઈપણ કોઈ array પણ Pass કરી શકાય છે!

Ex:-

Concatarray = itemes.Concat(Newarray , ["Hello" , "word"] , [1,2,3])
New
↳ log (Concatarray)
concat થયેલ array print કરશે.

⑥ Slice ();

→ Slice() method એ Array માંથી અમુક ચોક્કસ ઇન્ડેક્સને Cut કરીને ને Show કરે print કરવા માટે use થાય છે.

→ slice() method એ Array માંથી Value Remove કરતી નથી

→ slice() method એ હંમેશા New Array Return કરે છે.

⇒ Syntax :-

.Slice(start index , End index);

Ex:-

itemes.Slice(3 , 5); // 3 ,5

→ અહિ Itemes array માંથી index 3 and 4 ની value Show થશે અને last index Count થશે નહિ.

(7) splice():

* splice() method માં index value ના સ્થાન માં રાખીને array માંથી elements Remove અને Remove elements ની જગ્યા પર New elements Add કરવા માટે use થાય છે!

→ આ splice() method માં First time માં index start and End value input કરેલ છે. ત્યાર બાદ તે new Add element value Pass કરવામાં આવે છે.

Ex:-

```
itemes = [ "item-1", "item-2", "item-3", "item-5" ];
            [0]       [1]       [2]       [3]

itemes.splice( 1 , 2 , "Hello", "word" );
```

Remove element index value

* અહિં આ itemes Array માં "1" and "2" index પરથી item-2, item-3 Remove કરશે and Hello, word ને Add કરશે.

* જો Remove element ના બદલામાં કોઈ Add value ના બદલામાં આવે ના તો કોઈ Add કરશે નહિ માત્ર Remove કરશે.

Output

↳ ["item-1", "Hello", "word", "item-5"]

chang

(8) Reverse(); only use.

=> Reverse() method એ array ને Reverse ઓર્ડર માં print કરવા માટે થાય છે.

÷ Syntax :-

items.reverse();
Array

(9) Sort(): // num = [33, 55, 40, 11, 115]

Sort() Method એ Array માં numbre Value ને ascending and Descending order માં a print set કરવા માટે use થાય છે.

num.sort();

But :- // output => 11, 115, 33, 40, 55

Normal Sort method માં જે ત્રણ અંક હશે અને 115 જેમાં First અંક એક હોવાથી તેને તે ઓછી આંકડા ના order માં print કરશે તો only sort Method એ Perfect Sorting કરતું નથી.

num. sort(((a,b) => a-b));

if :- // => 11, 33, 40, 55, 115

Perfect Sorting કરવા માટે Sort((a,b) => a-b);
└ arrow funct

arrow function Pass કરવામાં આવે છે. જે array માં એ index Value ને comper કરવી ને તેને ascending or descending order માં Perfect રીતે store કરે છે.

Num = [11 , 5 , 50 , 40 , 33]

10 Max() & min():

=> in array, max() and min() Method આ .apply(null, array) આવો use કરીને તેનો work શું છે?

-: syntax of use. :-

50 // function Mymax (array) { return Math.max.apply(null, array); }

5 // Math.min.apply(null, array);

-: syntax :-

pass value

any array name

=> આમાં max value Method આ array પર work કરવા માટે તેનો ઉપર મુજબ :apply(null, ___) આવો use કરવામાં આવે છે.

11. forEach()

=> this method આપણ print Array in simple txt form format in console.

Ex :- -: Syntax :-

let text = ["Hello", "World", "This", "is", "JavaS", 6];

This is Array from // let txt = " ";

text.foreach(function (Val, index) { txt += Val + " " ; } function body);

This foreach method store array in empty string and print to string in simple txt form.

main array Ex:-
text [1, 20, 45, 60, 75, 90, 100, "Hello"];

Note:-
every(), some(), find(), filter(), findIndex()
reduceRight(), reduce(), map()

This method Call back function પર work કરે છે.
all-~~every~~ method ની અંદર માં second function Pass કરવામાં આવે છે. જે function અલગ માં Arrow function પણ હોઈ શકે છે! અને આ function માં condition return કરવામાં આવે છે
=> This method all store in new Variable

(12.) every()
This method huse to chack all array element with given condition ને in function.

```
let Reszult = text.every( (Val, index) => {
    return Val > 50;}
);
console.log (Reszult); // Flase
```

=> Becouse in this array " 1, 20, 45 " are less than " 50 " that's way it's Return flase Val.
=> if is Val > 0 it's //true

(13.) Some ();
This method huse to chack all array element and Return Boleon Value.

```
let Result = text.some( (Val, index) => {
    return Val > 50; }
);
Console.log (Result); // True
```

= This method is Return True Becouse in This array some element greter than 50: " 60, 75, 90, 100 "

// Some Method માં જો કોઈ એક element જો function ની Condition થી mach થશે તો તે true Return કરશે.

(14) Find ();

find () method chack and Return only First element in array it's work only on time

```
let Result = text.find( (Val, index) => {
   { result Val > 50; }
});
Console.log (Result) // 60 => only Val.
```

Note → find method એ array કે જેને give condition સાથે mache થતું હોઈ તેના First element એ છે. તો તે ને તે Return કરે છે. આગળ આગળથી ચાલ છે.

(15) filter ();

Filter method એ new array create કરીને આપે છે.

Filter method માં function ના condition mach થતા elements ને Return કરીને new array આપે છે.

Ex:-

```
let NewFilter = text.filter( (Val, index) => {
   Return (Val > 20 || Val == "Hello") }
});
Console.log( NewFilter ); // 45, 60 ..100 and Hello
```

Note :-
findindex method માં જો કોઈ given condition match ના થાય તો તે "-1" return કરશે.

(16) findIndex ( ) ;

=> findindex માં function માં જે કોઈ condition match પણ કોય તો તેની index Value Return કરે છે!

=> This method only Return First Value were it's match with condition

Ex :-

```
let ReturnsIndex = text.findindex( (Val , index) => {
        return Val > 20 } );

Console.log (Returnsindex); // 2 "index"
```

(17) map ( );

map ( ) method એ new array create કરવા માટે use થતી method છે! and This method also Chang array

=> maP ( ) method માં array માં ભાગાકાર, ગુણાકાર, plue etc elements આપે work કરાવી શકિએ છીએ.

Ex :-

```
let NewmapA = text. map( (Val , index) => {
        return Val + 1 ; } );
```

=> NewmapA ને Console કરતા new array print થશે જેમાં array ની દરેક Value માં + 1 થઈ ને show થશે.

(18) reduce() અને reduceRight():

⇒ This method આ full array ને single one variable માં stroe કરીને print કરવા માટે use થાય છે!
જેમાં reduce and reduceRight method દ્વારા તેમાં +, -, *, / વગેરે opression કરી શકાય છે!

Note:- reduceRight આ array માં left Right થી
શરૂ left side સુધી work કરે છે!
* reduce આ left side થી Right સુધી ના element count's કરે છે!

(* requred pass values)

Ex:-

```
let red = text.reduce( (total, index) => {
    result total + index ; }, 0);
```

{ inesuled total value of start * }

⇒ This method is giva use a total of text array in return is (391)

{ ⇒ same us reduceRight work }

(19) entries():
This method wus shove in console index and it's value pare in screen usin next().value mathod.

Ex:-

```
let show = text.entries();

Console.log( show.next().value ); // [0, 1]
                                  // [1, 20]
                                  // [2, 45]
```

Output

(20) includes( ):

includes method આ Direcut console.log કરાવી શકાય છે. includes માં pass કરેલ element જો array માં હશે તો તે true Return કરશે નહિ તો એ false Return કરશે.

Ex :-

Console.log(text.includes('60')); // true

(21) Copywithin()

Copywithin() method દ્વારા arrayમાં element copy કરવા માટે use કરવામાં આવે છે.

:- Syntax :-

Copywithin(targetValue, start ind, end ind)

→ Copywithin() method માં ત્રણ Value pass કરવામાં આવે છે. First એ

First :- જે index પરથી element copy થશે.
Second :- element copy કરવાની બાદ ચાલુ તે index Val
Thired : element copy કરવાની End index Val

Ex :- index 0 1 2 3 4 5 6

copy = [10, 20, 30, 40, 50, 60, 70]

chang

log(Copy.Copywithin(3, 0, 2)); ← target, start, end

અહીં last end index count નહીં થાય.

Output [10, 20, 30, 10, 20, 60, 70]; (copy)

// object //

What is object

⇒ An object is Collection of properties.
⇒ An object has (key) name and values pares
⇒ in object value can be a function in which case the Property is know as a method
⇒

⇒ Syntax of object :-

Variable ⇒ Const obj = { }; // it's empty object
obj ↑ object name
{ } → object body

// dot notation in object

⇒ object ની key ની અંદર ની Console ની અંદર show થાય
Ex!
Ex :- ⊛

```
Const Profile = {
(key) → Fname : "John",  ← (Value)
        Lname : "Don",
        age : 50,
        email : "JohnDon@gmail"
}
```

Profile.fname → John

⇒ We add new key and Value pare. using dot notation

Ex: { Profile.Mo.num = 7435961978;
      profile.age = 44;

→ આવો આ method વાપરી ને object માં ફેરફાર કરી શકાય છે.

object call method

=> Profile['Fname']; / Profile['age'];

↳ આ રીતે square brecat ની અંદર single qutes ની અંદર key value આપીને એના call કરાવી શકાય

① * delete funcrimlity of object

→ object માં કોઈ key and value ને delete કરવા માટે

:- Syntux :-

delete object name . key name

- delete ← keyword
- object name ← name
- . ← dot notation
- key name ← key name

② object માં કોઈ variable Trow key asscess

Ex :-

```
const key = "Email";
let person = {
    name : "Hello";
    age : 22
}
```

Add New Value → Person.key = "Hello@gmail.com"; ← This not work

↑ with dot noteson

→ in obj: key : "Hello@gmail"

---

with squre breacets:

Person[key] = "@gmail.com";

↓

{ email : "@gmail.com" }

Class syntax :-

```
class Person {
    Constructor (name) {
        This.name = name;
    }
    display = function () {
        return "Hello" + this.name;
    }
}
```

class → key word for class

// using new key word

```
let obj = new person ('John');
```

create object and pass value

→ Call class :-

```
Console.log (obj); //
```

↳ full show in console as a object

```
Console.log (obj.display());
```

↳ Here show Display function as object properties

આ બંને loop Map & Set પર work કરતા નથી.

for Array and object

for of & for in loop

Notes:- for of & for in loop આ Array પર work કરી output show કરે છે!

* for of loop :-

→ for of loop ને Array પર ઓછું પણ કરાવી શકાય છે because for of loop માં આ Direct Value print કરવા માટે use થાય છે

* Syntax of loop *

```
for (const i of array & obj name )
```

- for → keyword
- i → Variable
- of → key word

Ex:- let Products = ['Mobile', 'laptops', 'TV', 'Tablets']

```
for ( const i of Products )   ← Name
{
   Console.log ( i );   // alway print Variable in loop
}
```

```
for (const key of Person.keys())
{
  Console.log( key )
}
```

Array ને iterate કરીને print કરશે.

key → object પર iterator કરવું

→ output

ID
Name
age

} map key

Example →

```
...ap( );
...1]);
..., 'Herray']);
..., 21]);
...Person.keys() )
...key} : ${Person.get(key)`);
```

→ for of loop newMap() in iteration.

* for in

for in loop ની Array and object વગેરે ને iterate કરવા માટે use કરી શકાય છે!

→ Syntax of loop :-

```
for ( const  i  in  Array & obj name );
```

for → keyword
const → variable
in → key word
Array & obj name → iterator Value name

For in loop માં only key Value જ iterat કરી શકે. એટલે object ની key and array ની index Value જ only console માં print કરી શકે!

```
Const Prof = {
    Name : "Hello",
    useN : "@Hello21",
    Post : 201.
}
```

keys ← Name, useN, Post
Values → "Hello", "@Hello21", 201

array

```
let Product = [ "TV", "Mobile", "laptop", "Tab" ]
```

index
0
1
2
3

Ex :-

```
for ( const i in Porf )
{
    Console.log ( i );// Name, useN, Post
    Console.log ( Porf[ i ] ); Hello, @Hello21, 201
}
```

i → this key only
Porf[i] → Value only

→ Sum us array :-

```
for ( const i   in  Product ):
{                              only index Value
    Console.log ( i ). // 0 , 1 , 2 , 3

    Console.loy (Product [ i ] ); //
                        // TV , Mobile , laptop , Tab
}                          only Value
                              Sum us for of loop
```

☆ Cello copy & deep copy ☆
with rest Parametres

Note :-

① Cello copy :-

Cello copy માં કોઈપણ Array કે object ને બીજા કોઈ Variable માં copy કરવામાં આવે તો તેમા માત્ર main Variable નુ momrey Addres copy થાય છે. new array & object create થતા નથી

જેથી જો main array & object માં ફેરફાર કરવામાં આવે તો copy થયેલા array & object માં પણ તેનો ફેરફાર જોવા મળે છે.

=> Cello copy એટલે કે rest Parametes વગર કરવા માં આવતી copy.

★

② deep copy :-

in deep copy we use rest paramets જે (... > Three dot) કોઈ પણ copy કરવા માં આવતા array & object ની name ની આગળ three dot લગાવી ને જો તે copy કરવામાં આવે તો ત્યાર બાદ તે માં new array & object ની copy બને છે જે main array અથવા તો object ને ફેરફાર પર આધાર રાખતી નથી.

Ex :-

	Cello copy	Deep copy
Array →	arry = [1, 20, 30, 40]	array = [10, 20, 50, 40]
	arraycopy = arry ;	Newarray = [...array] ;
	↑ Cello copy	↑ deep copy
Object →	Profile = { }	object Myobj = { x y z }
	NewProfile = Profile	Newobj = {...Myobj}
	↑ Cello copy	↑ deep copy
	જ્યારે કોઈ પણ copy કરેલા object માં કે arry માં change કરવામાં આવે તો બન્ને માં change થઈ જશે.	જ્યારે જે obj કે array માં change કરવામાં આવે તેમાં જ માત્ર ફેરફાર થાય છે. બન્ને માં એક સરખો ફેરફાર થતો નથી.

→ Asynchronous ←
&
→ synchronous ←

✱ Asynchronouse :-

Note → Asynchronous code એ છે કે જેમાં પહેલાંની line માં code કે function આવે છે તેને stop કરતુ નથી ને તેને Complie કરી આપે છે

for objects

* Optional chaining *

(?.) key word :-

→ optional chaning માં (?.) આ રીતે કોઈ પણ object ના key and value call કરવામાં આવે ત્યારે તેને આવી use કરવા માં આવે છે।

Ex:- use of syntax :-

```
Const users = {
    fname : "Harray",
    age : 21
}

Console.log (user.age); // 21
```

↳ (આવી object માંથી console માં 21 print થશે।)

```
Console.log (user.ID); // Error
```

↳ આવી object-માં ID name ની કોઈ key અને તો console માં Error show થશે।

→ But
```
console.log (user?.age); // 21
Console.log (user?.ID) // undifined
```
(?.)

↳ optional chaning આવી use કરવામાં આવે તો Error ની જગ્યાએ undifind show થશે। And જ્યાં key object માં create કરવામાં આવેલ હોય fin Value print કરશે।

:- Methods :-

This, call(), bind(), apply()

* This keyword

This keyword એ object ની અંદર મધ્યમાં આવે છે object ની અંદર બીજી key ને Acess કરવા માટે used કરવામાં આવે છે.

Note :-
This key coord ને જે object ની અંદર મધ્યમાં આવે ત્યાં જ present હોવું જોઈ

Ex :-

```
user = {
    FristName : "Herray",
    age       : "21",
    about     : function ()
                { console.log (`Name is ${this.fristName}
                                age is ${this.age}`);
                }
},
```

⇒ user.about(); // Name is Herray, age is 21

This key coord એ about key ની અંદર બીજી function માં fristName and age key ને Access કરવા માટે This નો use કરવા માં આવ્યો છે.

* Call ()

Call Method માં કોઈ એક object માં કોઈ function બનાવવામાં આવેલા, હોય તે sem function એ કોઈ બીજા object માટે પણ work કરતા હોય તો આપણે call Method દ્વારા એક object પર બીજુ object ની function apply કરી શકીએ છીએ.

Ex :-

```
Student = {
  Name : "Ram",
  class : 10
  date : function ()
  { Console.log(`Name is ${this.Name}\n
                 Class is ${this.class}`);}
}
```

```
Student2 = {
  Name : "shayam",
  class : 12
}
```

```
Student3 = {
  Name : "Nike",
  class : 9
}
```

→ Student જે object છે તેમાં તેના key Value print કરવા માટે this Method દ્વારા function બનાવામાં આવ્યુ હતું but Student 2 / 3 માં કોઈ function create કરવામાં આવ્યુ નથી તો તેના પર તેની key Value print કરવા માટે call Method નો use થાય છે.

Ex ⇒

First obj it's has function it's own

↳ Student.data(); → Name is Ram
class is 10

Second obj it's dont have function

pass param — obj

Student.data.Call (Student2); → Name is shayam
↓ call (Student3) class is 11

Student ના data key માં
જે function છે તે call
કરીને Student 2 obj પર apply
કરવામાં આવ્યું છે।

Call Method માં Paremetrs પણ Pass કરવામાં આવે
છે। object and etc variable to

Ex :-

↳ Call

Student.data.Call (object , "Mark's" , "RoomNo"

↑ ↑ ↑

After object → String or Number data pars pass

apply ()

→ apply method એ sem call method ની જેમ work કરે છે but તેને call માં passing params કરવા હોય તો bob object and string એમ કરવા છે પણ apply માં

```
nume..apply (object , [        ] );
```

↑ First params ⊛

↑ second params array ★ optional

→
```
function Ab () {
    console.log(` Nome : ${this.Name}
                  age: ${this.age} `);
}

const student = { Name : "Rocy" , age : 21 };
```

Mendatory ↓ array Not Mendatory

Call →
```
Ab.apply ( Student , [ ] );
```

↳ output
```
Name : Rocy
age : 21
```

Note → apply Method ને array સાથે પણ use કરી શકાય છે

trick

```
let Number = [ 5 , 10 , 50 , 40 , 55 ]

let Num = Math.Max.apply ( null , number );
// 55
```

↑ obj is null

⇒ Math.MethodName.apply(null, arrayName)
(null ↑ obj) (arrayName ↑)

→ array માં Math ની કોઈ method ને apply કરવા માટે ઉપર મુજબ syntax માં use કરવું જેમાં apply માં Frist param એ object નો હોય ત્યારે null pass કરવું.

⇒ bind()

bind Method એ છે કે તે Return જ્યારે તેને other function માંથી call કરવામાં આવે ત્યારે તે Return માં એક function આપે છે તે direct print કરવું નથી.

⇒ Syntax :-

1. FunctionName.bind(objname);

2. objectName.functionName.bind(objname);

→ bind ને print કરવા માટે Two Method છે.

<u>First</u> Ex :-

```
function Print() {
    Console.log("Name : this.Name
                 age : this.age");
}

stud : {
    Name : "Hello",
    age : 22
}
```

(2) Print.bind (Stud) — Function

this is Return funt
આ console. માં only f( )
આ રીતે function Print કરશે
Not show output

(3) print.bind (stud)( ); — function, call

this is show std Nam and age in output not function

Second.

let show = Print.bind (stud);

show( );

↳ { છે bind ના output ને show Variabl માં store કરવાથી તે મા એ function store કરશે, અને તેને output જોવા માટે તેને function ને call કરવુ પડશે।

=> Console.log (show); (x) Not Valid

★ bind ( )

bind Method એ જેમાં Pass કરવામાં આવતા function ને bind કરીને other object મા તેનું this keyword pass કરે છે!

એટલે કે bind એ object ની key value ની Acsse other function ને આપે છે!

* Synchronous & Asynchronouse

=> Javascript was synchronous programming Language.

* Asynchronous માં Javascript ની અંદર આપણે code ને daly કરવામાં આવે ત્યારે Asynchronous Javascript કરવામાં આવે છે.

Note

SetTime out only one time → execute થાય છે

જેમાં SetTimeout( ); function / method માં ફોરવી કરવામાં આવે છે.

SetTimeout( ); માં Two Parems pass કરવામાં આવે છે Frist function ફોય છે. second code daly કરવામાં આવતો time જે milliscends માં ફોય છે.

SetTimeout એ એક ID પણ Return કરતુ ફોય છે.

Ex:-

:- Syntax :-

```
SetTimeout ( function () { }  ,  1000 );
             () => { }        ,  3000, );
             Hello            ,  0     );
```

in First Part જેમાં Anonymous func, Arrow func, function by name આવી pass કરી શકાય છે.

in second part time મુકવામાં આવે છે જે millisconds માં ફોય દા. 1000, 0 etc જેનો ખ્યાલ

SetTimeout માં મુકવામાં આવેલ function એ તેમાં મુકવામાં આવેલ time પ્રમાણે executed થશે.

→ SetTimeout () માં આપવામાં આવેલ Time જ્યારે wait થતો હોય છે. ત્યારે Javascript માં તેના પછીની line માં જે કોઈ code હોય છે. તો તેને Javascript console કરી દેય છે! ત્યારબાદ E last માં setTime out વાળા functine ને Run કરે છે।

Ex :-

```
Console.log ("Hello");  ☑

SetTimeout ( () => {
    Console.log (" in setTime out");
}, 2000 );
    ↑ delay Time (2s)

Console.log ("Time word");  ☑
```

} Pass function

------ Output ------

Hello
Time word } all work complie First
in stetime out ← daly of (2s)

ClearTimeout ()

ClearTime out નો use SetTimeout માં Run ને stop કરવા માટે use કરવામાં આવે છે! SetTime out ને જ્યારે Create કરવામાં આવે છે ત્યારે કોઈ ID ને એક ID create કરે છે ને ID ચોક્કસ કે setTimeout ને એક Variable માં store કરીને ClearTimeout માં Pass કરવામાં આવે તો ને setTimeout ને ન ચિત્ર થવા થી અટકાવે છે!

Ex :-

(Variable stored with ID)

```
let ID = setTimeout ( () =>
{ console.log ("Hello"); }, 2000 );   } Not complied

console.log (" How are you ");

Cleartimeout ( ID );
```

—— output ——

How are you

— end compling —

cleartimeout માં setTimeout ની ID pass કરવા થી તેની execution stop થઈ જાય છે.

★ SetInterval ( );

→ SetInterval method એ same input setTimeout ની જેમ લેય છે. એમાં તેમાં પણ Two params pass કરવામાં આવે છે જે મા First function and second Time

→ ફરીથી!

setInterval પણ ID Return કરે છે!

★ Defreme

→ setInterval ( ) એ આપેલા Time મુજબ Rept call થયા કરે છે!.

→ SetTimeout ( ) એ only one Time ~~Rep~~ call થાય છે;

—— ClearInterval ( ) ——

ClearInterval એ તેમાં જ setInterval ની ID pass કરવાથી તે stop થઈ જાય છે.

Call back function ()

* Call back function નો અર્થ : એક function ની અંદર બીજા function pass કરવામાં આવે તેને call back function કહેવામાં આવે છે. And call back function નીચે Anonymous, Arrow, IIFE function પણ કોઈ શકે છે.

① Example :-

```
Function lineOne () {
    Console.log (" Hello word First ");
    lineTwo ();   // Second function call in First func
}

Function lineTwo () {
    Console.log (" Hello word Second");

}

lineOne ();
```

જ્યારે lineone func ની call કરવામાં આવશે ત્યારે તેની અંદર નો code Run થશે. ત્યાર બાદ તેમાં call કરવામાં આવેલ function automatic call થશે lineone() function દ્વારા.

Output :-
Hello word First
Hello word second

② Example :- (Pass function as Parames.)

```
function line1 (x) (callback) {
    Console.log (" Hello word 1 ");
    Callback ();
}

function line2 (          ) {
    Console.log (" Hello word 2 ");
}
```

// call function

line1(line2);

↑ pass the function

line1 function ની અંદર perams મારફતે function pass કરવામાં આવ્યું છે line2 નો last માં ઉપયોગમાં આવ્યો છે. line1 function ના code withe call brecket brecets. ()

|— output —|

Hello word 1
Hello word 2

③ Example:- (function withe numbres and argemets & callback function.)

first

```
function getNum ( Num1 , Num2 , callback ){
    Console.log(`Number : ${ Num1 },
                 Number : ${ Num2 }`);
    Callback ( Num1 , Num2 )
} // end function
```

call

second

```
function TotalNum ( N1 , N2 )
{
    Console.log ( N1 + N2 );}
```

This Num1,2 એ getNum દ્વારા call back માં pass કરવામાં આવશે જે TotalNum છે. તે Num1, 2 નું Total આપશે.

—— call func ——

```
getNum ( 10 , 20 , TotalNum );
```

↑ arge ↑ arge ⏟ callback

|—— output ——|

Numbr 10
Numbr 20 - 30

જે function માં callback તરીકે આપેલ છે તેથી પહેલાં બીજા function pass કરી શકાય છે!

★ Example of nested callback funct ★

{ Pyramid of doom in JS
&&
Callback hell }

```
//main function that we call all time's

function stylechange ( elem , txt , color , time ,      } Pass
                       onSuccesscallback ,               } argues
                       onfuilurecallback )               } in func
{ // start
  SetTimeout ( ( ) => {
    if (elem) {
        elem.textcontent = txt;          } first task done
        elem.style.color = color;        }
      if (onSuccessCallback)             } as par if
      {                                  } Second task done
        onSuccessCallback();             }
      }
    }
    else {                               } second work
      if (onfuillere callback)           } if
      {                                  } first work is
        onfuillere callback();           } not done
      }
    }
  }, Time )
} //end func
```

→ stylechange ← માં ય argument and ૨ callback func pass કરાયાં છે

// Call Stylechang function

```
Stylechang ( h2 , line1 , "green" , 1000,

// on Successcallback funct creatin as Arrow
( ) => //2
{  Stylechang ( h2 , line2 , "blue" , 1000 ;
    ( ) => //3
   {  Stylechang ( h1 , line3 , "pink" , 2000,
      ( ) => //4
       Stylechang ( h2 , line4, "yellow", 1000,
       ( ) => //5
      { stylechang ( h1 , line5, "Red", 2000 , ( ) =>
       ( ) => //empty
       {}
      },
     ( ) => { .log ( 'h1 is not for line5 ) } ) //end 5
},
( ) => {.log ('h1 is not for line4 ) } ) //end of 4
},
( ) => {.log ( 'h1 is not for line3 ) } ) //end of 3
},
( ) => {.log ( h1 is not for line 2 ) } ) //end of 2
},
( ) => { .log ( h2 , is not for line1 ) } ) //end of 1
```

// mine function end

— Promise —

★ Creating promise

```
new promise ( );
 ↑        ↑
keyword   Promises
for obj   keyword
```

* Promise એ new keyword દ્વારા create કરવામાં આવે છે.
→ promise માં callback function પણ pass કરવામાં આવે છે.
→ Promise માં બે ટાઈપ state હોય 1. resolve &
2. reject (Error);

(1) resole એટલે કે જ્યારે Promise માં લખવામાં આવે code કોડ success full Run થશે ત્યારે resole વાળો part run થશે અને output આવશે.

2. reject :-
માં હંમેશા Error store થાય છે And Return પણ Error મળે છે, જો resole માં success થતા માં અટકાવતી હોય છે reject માં આપણા પોતાની custom Error પણ create કરી શકીએ છે

—: Syntax :—

```
=> Const Tea = [ "Tea Powder" , "sugar" , "Milk" ];
=> const promies1 = new promies( (resolve , reject) =>
   {
       if (Tea.includes("Milk"))
   { resolve(" Your Tea is ready in 2 min "); }
   else
   { reject(" Your Tea is not ready "); }
   }// end of Arrow ); // end of promies
```

Console.log ("promise1"); // condition True so The promise ક્યાં resole થશે અને resole part print થશે.

("Your Tea is Ready in 2 mins")

Note:- જો Promies માં if else condition create કરવા વગર Promies માં resole & reject બંને Pass થતા હોય Promies માં resole & reject ને Pass કરવામાં આવે તો હંમેશા promies First one એટલે કે resole stat ને જ print કરશે એટલે check કરવા માં આવે છે. if resole is done or not with if else statement.

* resolve & reject એ બંને Promise ની Method છે! જે કોઈ પણ data. variable, obj, function, string) Pass કરી શકીએ છીએ.

Print of promise in console with than() & catch ()

→ Promise ને Console. માં Print કરવા આપવા તેનું output show કરવા માટે than & catch ની method નો use થાય છે!

→ આ બંને method માં callback function આપવા માં આવે છે!

① .than(() => { });
than એ હંમેશા resolve state ને જ get કરીને print કરે છે!

> then() method
> નો always Return
> promie that we can use for next then()

એટલે કે જો promies એ resolve થશે ત્યારે .then method work & run થશે જ નહિ તો આગળ નહિ ચાલે.

Note:-
Then() method ની chaing પણ બનાવી શકાય છે
એટલે કે જો first then() resolve થશે તો second
then થઈ rsolve થશે & threed then() resolve
થશે એવી રીતે ex:-

```
Promies.then(() =>
        {.log("frist then()");}):
        .then(() =>
        {.log("second then()");})
        .then(() =>
        {.log("threed then()");});
```

Promies name ↗ (points to "Promies")

} પહેલા first then after second...

→ જો first Promis resolve નહિ થાય તો second & threed પણ resolve થશે નહિ.

(3) Catch(() => {});
catch() method માં હંમેશા Error એટલે કે reject state ને print કરવામાં આવે છે!
જો promie માં then() method નહિ થાય resolve થાય તો તે catch method માં Error એટલે કે reject Part show કરશે.

Ex:-

```
.then(() => {}).catch(() =>
                    { console.log("rejected");});
```

finally()

finally method નો mostly not use but in sum case like
જ્યાં finally નું work એ છે કે promies થાય resolve થાય અથવા તો reject થાય તો પણ finally માં જે કાંઈ હશે તે print થશે. તે promis ના resolve કે reject સાથે conection હોતું નથી.

Ex: of than(), catch(), finally()

```
const Tea = ["Tee powder", "suger", "milk"];

Const MackTea = new promies (
    (resolve, reject) =>
    {
        if (Tee.includes("suger"))
        { resolve(" your tee is redy soon...");}
        else
        { reject(" your tee is not redy plase waite");}
    } //end of funct
) //end of promi

// print promies

MackTee.than ( (Val) => { Console.log(Val) }
    .than ( ( ) => { console.log(" Thanks") }
    .catch ( (Val) => { Console.log (Val ) }
    .finally ( ( ) =>
        { Console.log ("sir"); );

        // end //
        output →
```

{Ext - chaing than()}

or Tea નામના Array માં suger છે જે થી Promies if condinstion માં જશે And resolve થશે and than () method work કરશે first and second પછી & last માં finally પણ run થશે

Output is then
you Tea is redy soon.. ← first Promies of
Thanks ← (second chainge than() ext)
sir ← finally who alway Run.

Synchronous && Asynchronous

Synchronous :-

in Javascript જે code line by line code exicut થાય તેવા code ને synchronous code કહેવામાં આવે છે!
જે માં કોઈ function હોય ની first function Run થાય ત્યારબાદ તેની પછીનાં funcri આવી રીતે line માં "syn" માં Run કે exicut થતા code synchronous Code કહેવાય છે

=> Normal function, loop, if, else, else if etc

Aynchronous :-

in Javascript જે code હોય ને line by line exicut થતાં નથી એક સાથે Run થાય અને જે code first end થાય તેનુ output first આપે ત્યાર બાદ secode હોય ને જ થાય ની thread નુ આપવી.

=> SetTimeout, SetInterVal, Promise, fetch, axiox

① async :-

→ async એ કોઈ sync function ને async way માં convert કરવા માટે લખવામાં આવતો keyword છે.

→ જે function keyword ની આગળ લખવામાં આવે છે.

Ex :-

```
async function name () {};
   ↑
```

→ async function ને call કરવામાં આવે તો તે Promise Return કરે છે જે આ async function ની બહાર લખવી પડતી નથી

② await :-

await એ async function હોય છે તેની અંદર લખવામાં આવતો Keyword છે. Fund Await એ then જે Promises માં આવે છે. તેની જગ્યા લખી શકાય છે

Ex :-

```
async function getdata () {

Const respo = await fetch ("https://xyz.typicode.com/post");

Const data = await respo.json()

Console.log ( data );

}
```

respo.json() — This json() metho convert Json data in Js data

↑ after convert — This was Js (obj / array / strin etc.

શ્રી

③ fetch( )

fetch એ backend માંથી JSON data frontend માં connect કરવા માટે નો new method છે. કે જેમાં તે બે argues input તરીકે લેય છે

→ first "conction માટે ની link" ; https://....

→ Second" { જે backend ના data માં POST, Push, Delete etc કરવા માંગતા હોય તેની method or Data હોય છે! }

Second method માં જ્યારે function Pass કરવામાં આવે ત્યારે તે Json formet માં convert થઈ ને જાય છે!

Note :-

fetch એ Promiesse And Asynchronous code માં જ work કરે છે! becouse જ્યારે backend માંથી data get કરતી વખતે Time લાગે છે. ત્યારે તે wait માં રહે છે! જેથી તેની async & await ની key સાથે use કરવાથી તેની then & catch use કરવાની જરૂર પડતી નથી.

|— JSON —|

⇒ JSON એ JS નુ એક રૂપ છે જેમાં data store કરવા માટે use થાય છે. backend side થી data get થાય ત્યારે તે Json formet માં હોય છે.

file save extanction

(.JSON)

⇒ Json માં JS ની જેમ object હોય છે but JSON માં object ની key double quotes માં મુકવામાં આવે છે.

જ્યારે

⇒ JS માં object ની key ને quotes માં મુકવામાં આવતી નથી.

Json to JS

Json data ને JS માં Converte કરવા માટે Method

let x = JSON.Parse( ); // const json = '{"age": 21, "Name": x}'

⇒ JSON.Parse() ની અંદર જો કોઈ Json object Pass કરવામાં આવે તો તે Return માં JS object તરીકે Return કરશે. જે ને આપણા JS file માં use કરી શકીએ છે.

{ age : 21, Name : x }; // output

(convert)

Js to Json

જ્યારે backend side પર કોઈપણ data મોકલવામાં આવે ત્યારે તેને Js to Json માં convert કરવા પડે છે જેની માટે Method

upper object

⇒ JSON.Stringify (x);

↳ '{"age": 21, "Name": x}'; // output