*/\**

*1.push - used to add element at the end*

*2.pop - used to remove the element at the end*

*3.unshift - used to add element at the start*

*4.shift - used to remove element at the start*

*5.slice - slice(index, index-excluded) - copy from original array*

*6.splice - splice(index, noofelement) - cut from original array*

*//array helper methods*

*7.reduce - reduce(function(accumulated value , current value){},initial value)*

*8.map - .map(function(value,index){return value+index}); returns manipulated array*

*9.filter - .filter(function(n){ return n%2==0})*

*10.some - .some(function(n){ return n%2==0})*

*11.every - .every(function(n){ return n%2==0})*

*6.find - .find(function(n){return n%2==0})*

*\*/*

<!**DOCTYPE** html>

<**html**>

<**head**>

    <**meta** charset**=**'utf-8'>

    <**meta** http-equiv**=**'X-UA-Compatible' content**=**'IE=edge'>

    <**title**>Page Title</**title**>

    <**meta** name**=**'viewport' content**=**'width=device-width, initial-scale=1'>

</**head**>

<**body**>

    <**form**>

        <**table** border**=**"1">

            <**tr**>

                <**td**>First Number</**td**>

                <**td**><**input** type**=**"text" id**=**"f1"/></**td**>

            </**tr**>

            <**tr**>

                <**td**>Second Number</**td**>

                <**td**><**input** type**=**"text" id**=**"f2"/></**td**>

            </**tr**>

            <**tr**>

                <**td** colspan**=**2 align**=**"center"><**button** onclick**=**"add()">OK</**button**></**td**>

            </**tr**>

        </**table**>

    </**form**>

</**body**>

</**html**>

<!**DOCTYPE** html>

<**html**>

<**head**>

    <**meta** charset**=**'utf-8'>

    <**meta** http-equiv**=**'X-UA-Compatible' content**=**'IE=edge'>

    <**title**>Page Title</**title**>

    <**meta** name**=**'viewport' content**=**'width=device-width, initial-scale=1'>

    <**script**>

**function** add(){

            f1**=**document.getElementById("f1").value;//var -> Global variable

            f2**=**document.getElementById("f2").value;

            //document.getElementById("result").innerText="<b>"+(parseInt(f1)+parseInt(f2))+"</b>";

            document.getElementById("result").innerHTML**=**"<b>"+(parseInt(f1)+parseInt(f2))+"</b>";

            //where document is a built in object whuch refer the whole document

            //value Vs innerText/innerHTML

            //value can be used with form element wheras innerText/ html can be used with other html tags

            //parseInt is used to convert string to number

        }

    </**script**>

</**head**>

<**body**>

    <**form**>

        <**table** border**=**"1">

            <**tr**>

                <**td**>First Number</**td**>

                <**td**><**input** type**=**"text" id**=**"f1"/></**td**>

            </**tr**>

            <**tr**>

                <**td**>Second Number</**td**>

                <**td**><**input** type**=**"text" id**=**"f2"/></**td**>

            </**tr**>

            <**tr**>

                <**td** colspan**=**2 align**=**"center"><**input** type**=**"button" onclick**=**"add()"/></**td**>

            </**tr**>

        </**table**>

    </**form**>

   <**div** id**=**"result"></**div**>

</**body**>

</**html**>

<!**DOCTYPE** html>

<**html**>

<**head**>

    <**meta** charset**=**'utf-8'>

    <**meta** http-equiv**=**'X-UA-Compatible' content**=**'IE=edge'>

    <**title**>Page Title</**title**>

    <**meta** name**=**'viewport' content**=**'width=device-width, initial-scale=1'>

    <**script**>

**function** demo(){

            d1**=new** Date();//It will Systems current Date

            alert(d1);

            alert(d1.getFullYear());

            alert(d1.getMonth());

            alert(d1.getDay());//day from 0 to 6

            //To fprmat the date

            alert(d1.toLocaleDateString());//d/M/YYYY

            //toLocaleTimeString

            //toLocaleString

        }

        demo();

    </**script**>

</**head**>

<**body**>

</**body**>

</**html**>

/\* 1.Data type by Javascript

   2.Process

      Operators  ?,==,===

      Looping

      Conditional

   3. function

   4. Arrays-> Arrsay helper methods

   5. All apis

   6. Regularexp

   7. ES6

\*/

//console.log(a);//without declaring we will get runtime error

console.log(**typeof**(a));//without declaring we can use the variable

**var** s**=**10;

console.log(**typeof**(s));

s**=**"mmm";

console.log(**typeof**(s));

//Variable hoisting

console.log(**typeof**(s));//When we are trying to use any variable in any exression

// Variable hoisting will happen(It will declare the variable but intialization not done)

s**=**10;

 VAR VS LET VS CONST

// /\*

// let Vs var Vs const

// here let and const introduced in ES6

// \*/

// var x=10;

// x=90;//x can be  reassigned

// var x="lk";//redeclare

// let y=90;

// y=90;// we can reassign

// //let y=89;//can't redeclre

// const z=90;

// z=989;

// //const z="lk";// we can't redeclare and reassign

**for**(**var** i**=**1;i<=10;i**++**){

    console.log(i);

}

console.log(i)////var is global scope , const and let have local scope

 Operators:

== ===

Var x=23;

Var y=”23”;

Console.log(x==y); //op:true will convert the type and check , it is not strict equality;

Console.log(X===y );//op:false strict equality, will not convert the type;

| **Comparison** | **Conversion Rule** | **Result** |
| --- | --- | --- |
| int == double | Convert int to double | Compares values |
| char == int | Convert char to int (ASCII value) | Compares values |
| boolean == int | No conversion (not comparable) | Error/false |
| float == double | Convert float to double | Compares values |
| byte == int | Convert byte to int | Compares values |
| short == int | Convert short to int | Compares values |
| long == float | Convert long to float | Compares values |
| double == boolean | No conversion (not comparable) | Error/false |

Type coercion

Var x=10;

X=’a’;

// since js is loosely typed language

// based on the given value, type of variable will be changed is called type coercion in js

One type is converted to another

5==’5’ // type coercion will convert any if there are two operands if one operand is object it converts that to primitive

String is converted to number in this example (obj to primitive)

Ternary operators

Write a program to display the result based on the score

Score>=90 distinction

score>=80&&score<90 first class

score>=70&&score<80 pass

score<70 fail

score=89;

(score>=90)?console.log("distinction"):(score>=80&&score<90)?console.log("first class"):(score>=70&&score<80)?console.log("pass"):console.log("fail");

Function

/\*

1.Function declaration

2.function expression

3. Anonymous function(Function without name)

\*/

//Before declaration if we call function function hoisting will happen

greetMsg();//hoisting happens

**function** greetMsg(){//function declaration

  console.log("welcome");

}

//2.Function Expression(We can store function in a variable)

calculate();//Function expression will not be hoisted

//We can pass the function as an argument to another function

**let** calculate**=function**(){

    console.log("calculation");

}

**let** calculate1**=function**(a,b){

**return** a+b;

}

calculate();

calculate**=**10; //We can reuse the variable

console.log(calculate1(12,3));

console.log(**typeof**(calculate));

console.log(**typeof**(calculate1));

 //function reusability

//the same function with different name

Let add=calculate;

**let** calculate**=function**(){

    console.log("calculation");

}

console.log(calculate);//[Function: calculate]

/\*

1.Function declaration

2.function expression

3. Anonymous function (Function without name)

4.Callback Function

\*/

**4.Callback Function**

//A function as an argument to another function

**let** f1**=function**(){

    console.log("Welcome");

}

**let** f2**=function**(){

    console.log("to javascript");

}

**function** display(v){ //where vtype is fuction that is callback function

  v();

}

display(f1);//welcome

display(f2);//to javascript

**another example:**

**function** result(){

**return** 90;

}

**function** msg(){

**return** "Hi Successfully submitted"

}

**function** display(f){

   console.log(f());

}

display(result);

display(msg);

//Immediatly Invoked Function

(**function**(){

    console.log("welcome");

}

)();

 WebApi Function:

console.log("Welcome");

//Get data from database

setTimeout(**function**(){

    console.log("fetching data");

},5000)

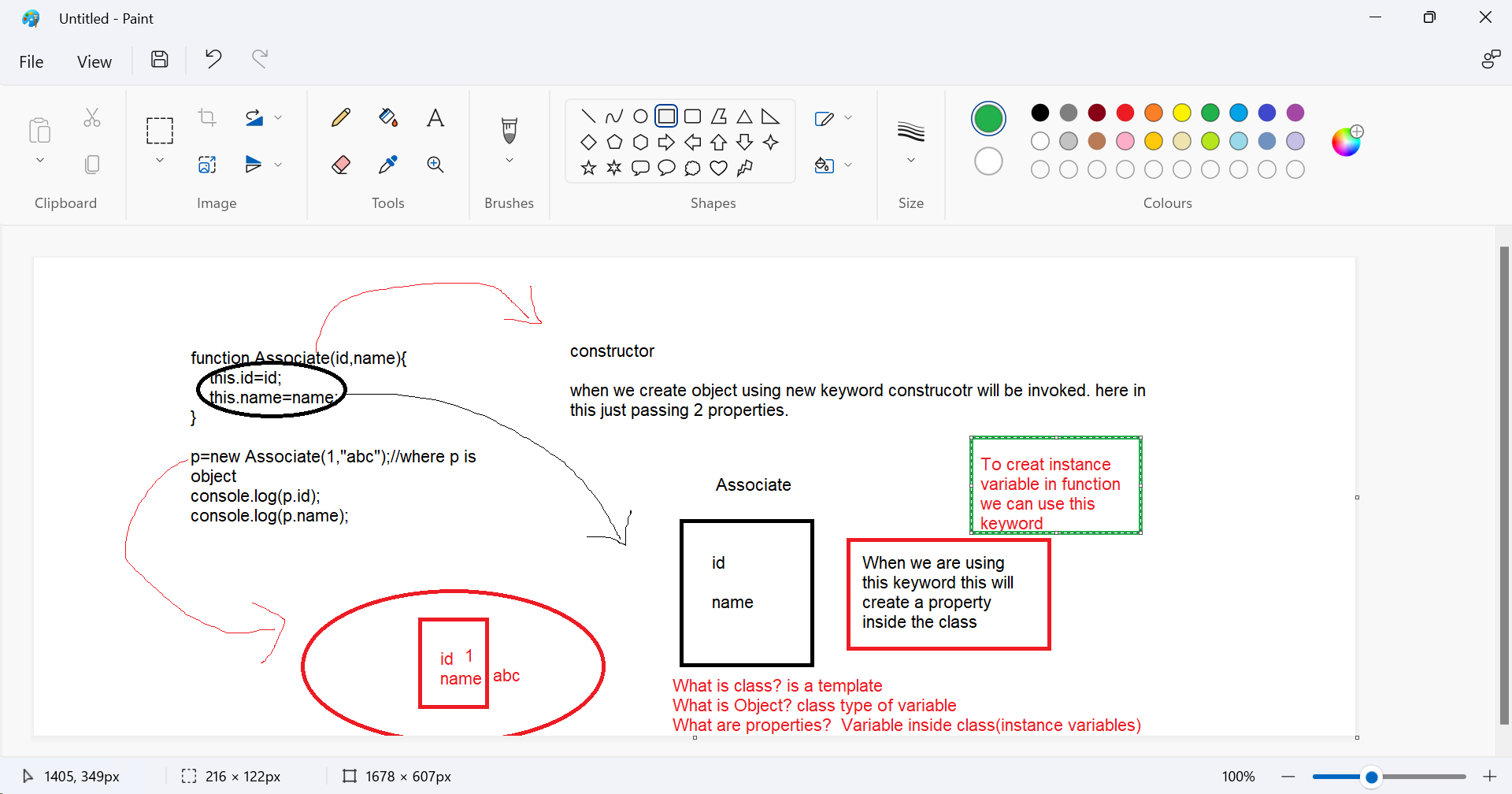
setInterval(**function**(){

    console.log("fetching data");

},5000)

console.log("End");

What is Class ,Object , Constructor function..?



**Constructor function:**

**function** Associate(id,name){

    this.id**=**id;

    this.name**=**name;

}

p**=new** Associate(1,"abc");//where p is object

console.log(p.id);

console.log(p.name);

**function** Associate(id,name){

    this.id**=**id;

    this.name**=**name;

    this.display**=function**(){

        console.log(this.id+" "+this.name);

    };

}

p**=new** Associate(1,"abc");//where p is object

p.display();

**let** a**=**[];

**let** b**=**[2,3,4,6];

a.push(1);

a.push(2);//add element at the end

a.unshift(8);//add element at the begining

console.log(a);

console.log(b.pop());//pop will return value from end

console.log(b);

console.log(b.shift());//shift will return value from start

console.log(b.length);

**let** a**=**[];

**let** b**=**[2,3,4,6];

a.push(1);

a.push(2);//add element at the end

a.unshift(8);//add element at the begining

console.log(a);

console.log(b.pop());//pop will return value from end

console.log(b);

console.log(b.shit());//shift will return value from start

console.log(b.length);

**let** a**=**[1, 4, 5, 9];

**let** b**=**a.slice(1,3);//will take copy of the array from original array starts from begining index and

// ends with end position

console.log(a);

console.log(b);

**let** a**=**[1, 4, 5, 9,7];

**let** b**=**a.splice(2,2);

//It will remove the values from an array get two arguments

// firat argument is startr index

//second argument number of elelemtn

console.log(b);

/\*

Array helper Methods

1. filter

2. map

3. reduce

4. some

5. every

6. find

\*/

/\*

Get values from an array which are divisible by 3

\*/

**let** a**=**[2,3,6,5,15,18];

**let** isDivisible**=function**(n){

**return** n%3**==**0;

}

//a=a.filter(isDivisible);

a**=**a.filter(**function**(n){

**return** n%3**==**0;

})

console.log(a);

/\*

Array helpwer Methods

1. filter

2. map

3. reduce

4. some

5. every

6. find

\*/

/\*

Get values from an array which are divisible by 3

\*/

**let** a**=**[2,3,6,5,15,18];

**let** isDivisible**=function**(n){

**return** n%3**==**0;

}

a**=**a.filter(isDivisible);

/\*

Flter method accept one function type of that function is predicate(which accept single value and return

true or false called

predicate)

\*/

a**=**a.filter(**function**(n){

**return** n%3**==**0;

})

console.log(a);

**var** a**=**[34,5,6,7];

a**=**a.map(**function**(v,i){

    console.log(v+" "+i);

**return** v+i;

});

console.log(a);

/\*

Array helpwer Methods

1. filter

2. map

3. reduce

4. some

5. every

6. find

\*/

var a=[34,5,6,7];

let sum=0;

for(let i=0;i<a.length;i++){

    sum+=a[i];

}

console.log(sum);

a=a.reduce(function(acc,v){//acc=acc+v

   return acc+v;

},0);

console.log(a);

//What if we 500 values in an array

str="())";

let a=str.split("");

let n=a.reduce(function(a,c){

if(a==-1){

return a;  
}

    if(c==='(')

        return a+1;

    else

        return a-1;

},0);

console.log(n===0?'valid':'Invalid');

 const a=[2,4,3,6,7];

/\*

find method argument is

predicate(will accept single any type value and returns boolean)

It iterate over the array and evaluate the element bassed on given predicate

if the predicate return true then it will return that element

It will return the first value which satisfy the predicate condition.

\*/

let val=a.find(function(n){

    return n%2==0;

});

console.log(val);

const a=[2,4,3,6,7];

/\*

Find whether giiven array contains atleast an even number

\*/

let v=a.some(function(n){

    return n%2==0;

});

let v1=a.every(function(n){

    return n%2==0;

});

console.log(v1);

let s1=1;

let s1name="abc"

let s2=2;

function getMaxScoreDetails(s1id,s1name,s1score,s2id,s2name,s2score){

    if(s2score>s1score){

    }

    return

}

In this example we cnt return more than one value from function

and also we needs to pass six arguments

better we can group the related data into single unit

/\*

Object

\*/

let person={id:1,name:"abc"};//Javascript Object

/\*

data can be  transsferd between two heterogenous application in the form of

JSON in webservice

\*/

let p={"id":"1","name":"abc"};//JSON Object

//Stringify will convert Object(JSOn/Havascript) to string

let p1=JSON.stringify(p);

console.log(typeof(p1));

console.log(p)

//parse will convert string to Object(Javascript)

let p2=JSON.parse(p1);

console.log(p2);

console.log(typeof(p2));

//Why we need class

let person={id:1,name:'abc'};

console.log(person);

console.log(person.id);

person.email="abc@gmail.com";//we are changing the template of Object

console.log(person);

//here is a code to retiurve data from api and trying to store in p1

// but from api the template is only id and name so will get data inconsistencies .

//So it is very important to define Object Structure

 //To define Obejct Structure we need class

//class is a template

//we can group related data into one class(Properties)

class Person{

    //we need to create properties(Instaance variable)

    constructor(id,name){//will be invoked while creating object

      //using this keyword(reference to the current object) we can define properties

      this.id=id;

      this.name=name;

    }

}

p=new Person(1,"abc");//just passing values to instance variable

// like this we can group related data into single unit

console.log(p.id);

console.log(p.name);

function Person(id,name){

    this.id=id;

    this.name=name;

}

p=new Person(1,"abc");

console.log(p.id);

Object Literals:

An object literal in js allows us to generate basic js objects.

**const** person={

    fname:"Deepak",

    lastname:"kumar",

    age:50,

    fullname:**function**(){

    return this.fname+" "+this.lastname;

    }

    };

    console.log(person.fullname());

    console.log(person["age"]);

**Class expression:**

A class can be defined inside an expression by using the class keyword

The class expression is another way of creating classes in js and they can be named or unnamed . if named, the class name is used internally, but not outside of the class.

syntax : Class Expression

**const** variable\_name= **class** Name{

**constructor**(var){

        this.var=**var**;

    }

    method(){

*//code here*

    }

}

**Static keyword:**

A class static method or field is defined with the static keyword

Rather than being defined on each instance, static is defined on the class itself

While static fields are helpful for caching , constant settings or any other data that doesn’t need to be copied between instances, static methods are frequently used to provide utility functions for an application.

class Book{

    constructor(bookName){

        this.bookName=bookName;

    }

    static namee="kumar";

    static author(){

        return "raj";

    }

    static print(obj){

        console.log("book called "+obj.bookName+" is written by "+this.author);

    }

}

const b=new Book("LIONS");

const c=new Book("Tiger");

// b.print(c); //TypeError: b.print is not a function

Book.print(b); // book called LIONS is written by raj

Regular expressions

**var str="322";**

**//to check the given string contains any number**

**var pattern=/[0-9]+/;**

**console.log(pattern.test(str));**

**//to check the give string is a number**

**//exact match**

**pattern=/^[0-9]+$/**

**var ans=pattern.test(str);**

**console.log(ans);**

**pattern=/^[A-Z]/ //starting with caps**

**console.log(pattern.test(str));**

**pattern=/^[0-9]{3}$/ // exact three digit**

**console.log(pattern.test(str));**

**pattern=/^[0-9]{1,3}$/ // upto 3 digits**

**console.log(pattern.test(str));**

**// The product code should follow a specific format consisting of eight characters, starting with the special character "#" followed by a dash "-", four uppercase letters, and then three digits from 0 to 9.**

**str="#-DEEP123";**

**pattern=/^#-[A-Z]{4}[0-9]{3}$/**

**console.log(pattern.test(str));**

**// string must contain 9-12 characters**

**str="abcdefghij";**

**pattern=/^.{9,12}$/;**

**console.log(pattern.test(str));**

**str="hello world";**

**pattern=/\s/**

**console.log(pattern.test(str));**

ES6

// ES6

/\*

  1. Template Literal

  2. Array/Object Destructuring

  3. Arrow Funcion

  4. Spread and rest operator

  5. Enhanced Object literals

  6. Promises

  7. Async/await

\*/

//Template literals

var name="abc";

var score=90;

console.log("Your Name is : "+ name +" \nYour score is : "+score);

//It will accept the new line

//Without concatenating and with using expression we can emmbed the variable values

console.log(`Your Name is: ${name}

Your score is: ${score}`);

//We can pass template literal to function

function display(strings,...values){//first argument string literals,second argumets are

//  template literals where ...values -> is an array

console.log(strings);//

 console.log(values);

}

let name='abc';

let score=90;

display `Your Name is ${name} and Score is : ${score}`;

//display(array of string liuterals,string of template literals each value separated by comma )

 Arrow function

// /\*

// Javascript Arrow Function

// 1. Provide concise way to write function

// let sqr1=(n)=>n\*n;

//  var=(p)=>{}

// \*/

// let a=[1,2,3,6,8];

// a=a.filter((n)=>n%2==0);

// a.forEach((n)=>{console.log(n)});

// let users=[

//   {id:1,uname:"abc"},

//   {id:2,uname:"mnb"},

//   {id:3,uname:"lkj"}];

// users=users.map((u)=>`${u.id},${u.uname}`);

// users.forEach((u)=>{console.log(u)});

//Lexical this

function Person(){

   this.age=34;

   //Every 1sec increment the age by 1

  //  setInterval(function(){

  //   //will have its own this context

  //   this.age++;

  //   console.log(this.age);

  //  },1000);

  setInterval(()=>{

    //It will its surround this context

    this.age++;

     console.log(this.age);

  },1000)

}

p=new Person();

console.log(p.age);//where age is Person's property

*/\**

*Arrow functions*

*1. convert normal function to arrow function*

*\*/*

*// let sum=function(a,b){*

*//     return a+b;*

*// }*

*// console.log(sum(10,20));*

**let** sum=(a,b)**=>**a+b;

console.log(sum(10,20));

*// 2. no param*

**let** print=()**=>**console.log("hello world");

print();

*// 3. single param*

**let** even=(n)**=>**n%2==0;

console.log(even(20));

*//4. more than one param and statement*

**let** mul=(a,b)**=>**{

**let** c=a\*b;

    console.log(c);

}

mul(9,8);

*//5. forEach(), map(), filter() method example*

**let** arr=["hi","hello","vanakam"];

arr.forEach(n**=>**console.log(n));

*// arr=arr.map(n=>n+"1");*

arr=arr.filter(n**=>**n.includes("1"));

console.log(arr);

*//6. accessing object using arrow function*

**let** users=[

    {id:1,name:"deepak"},

    {id:2,name:"vignesh"},

    {id:3,name:"santhosh"}

]

users=users.map((n)**=>**`${n.id} ${n.name}`);

console.log(users);

**function** Person(){

    this.age=34;

*//Every 1sec increment the age by 1*

*//  setInterval(function(){*

*//   //will have its own this context*

*//   this.age++;*

*//   console.log(this.age);*

*//  },1000);*

   setInterval(()**=>**{

*//If we use function inside setInterval it couldn't able access this.age,*

*// if we use array arrow function it can access this.age*

     this.age++;

      console.log(this.age);

   },1000)

 }

 p=new Person();

 console.log(p.age);

*// Array helper functions as arrow function*

*// 1.reduce using arrow function*

**let** hlp=[1,2,3,4,5];

 console.log(hlp.reduce((acc,val)**=>**acc+val,0));

*// 2.map*

**let** mp=[3,4,1,2];

mp=mp.map((n)**=>**n\*2);

console.log(mp);

*// 3.filter*

**let** fil=mp.filter((n)**=>**n%3==0);

console.log(fil);

*// 4.some*

console.log(mp.some((n)**=>**n%2==0));

*// 5.every*

console.log(mp.every((n)**=>**n%3==0));

*// 6.find*

console.log(mp.find((n)**=>**n%2==0));

//Rest Operator ...

//converts multiple elements in to an array, we can use in parameter

// function sum(...values){

//     console.log(values);

// }

// sum(1,2,3);

//console.log(sum(1,2));

//console.log(sum(2,3,4,5));

function sum(...values){

  return values.reduce((s,v)=>s+v,0);

}

console.log(sum(1,2,3));

//... Sprad Operator

//expand an arryay

let odd=[1,3];

let even=[4,6,8];

let number=[...odd,...even];

console.log(number);

//Arrays Destructuring

//It allows us to write concise code

const [apiKey,baseUrl]=['1234',"https://abc.com"]

console.log(apiKey);

console.log(baseUrl);

var str="1,abc,34";

s=str.split(",")

const [id,name,score]=s;

console.log(id);

console.log(name);

console.log(score);

let a={id:1,name:'abc'};

let {id,name}=a;

console.log(id);

//Enhanced Object literal

// While creating objecr we can use only the varible name

/\*

get id and name from user and create Object using Object literal

\*/

let id=1;

let name='abc';

let obj={id,name};

console.log(obj);//{ id: 1, name: 'abc' }

//Synchronized Task

console.log("welcome");

//To make that s asynchronous task we can use setTimeout web Api method

setTimeout(()=>{

  for(let i=1;i<=10000;i++){

    console.log(i);

  }

},1000);

//code to fetch data from Database

console.log("end");

Promises:

**function** countWords(sentence){

    return new Promise((res,rej)**=>**{

**let** words="";

        setTimeout(()**=>**{

            words=sentence.split(".");

            if(words.length>5){

                res(`count:${words.length}`);

            }else{

                rej("less count");

            }

         },2000);

    })

}

countWords("hi.hello.vanakam.this").then((res)**=>**console.log(res)).catch((err)**=>**console.log(err));

Fetch:

process.env.NODE\_TLS\_REJECT\_UNAUTHORIZED='0'; // to use in company laptop

fetch("https://jsonplaceholder.typicode.com/todos")

.then(res**=>**res.json())

.then(date**=>**console.log(date))

.catch(err**=>**console.log(err));

TYPESCRIPT:

Let is converted to var when used in es5 in ts

**var** x=null;

console.log(typeof(x));

*//when data is fetched from db , it can a string or any other datatype , some time there will be null value returned ,*

*// to avoid run time error , we can define the variable as string | null ...*

*// let a:string|null=null;*

*// console.log(typeof(a));*

*// let b:any="welcome";*

*// b.disp(); // through error only during runtime*

*// let c:unknown="welcome";*

*// c.disp(); // throws error during compile time*

*// so avoid using any.....*

**var** a:any=10;

**var** s1:string=a;

**var** b:unknown=10;

**var** a="abc";

**var** s2:string=b;

**let** a:any=20;

**let** b:unknown=30;

**let** s:string=a;

**let** c=string=b;*//type 'unknown is not assignable to type 'string'*