**JavaScript Interview Questions**

**1)what is JavaScript?**

Ans) Javascript is a client side language which is understood by the browsers and will be added in the HTML Code.

a) JavaScript is the world's most popular programming language.

b)JavaScript is a scripting or programming language that allows you to implement complex things on web pages

https://www.w3schools.com/css/css3\_variables.asp

**2)what is purpose of javascript ?**

Ans)To create the logic and control the content of HTML structure and your css.

**3) Javascript Display possibilities**

Ans)

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

**3)Javascript statements?**

Ans) In a programming language, these programming instructions are called **statements**.

<p id="demo"></p>

<script>

let x, y, z; // Statement 1

x = 5; // Statement 2

y = 6; // Statement 3

z = x + y; // Statement 4

document.getElementById("demo").innerHTML =

"The value of z is " + z + ".";

</script>

The value of z is 11.

**When to Use JavaScript var?**

Always declare JavaScript variables with var,let, orconst.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browsers, you must use var.

## When to Use JavaScript const?

If you want a general rule: always declare variables with const.

If you think the value of the variable can change, use let.

In this example, price1, price2, and total, are variables:

**4)Explain dffernt data types**

Ans)Boolean ,string,number,undefind,null,function,object

**5)Why to use NaN in javscript?**

Ans)”NaN” is a not a number

NaN function is used to check the number in the argument.

If it is does not contain number then it will be return true else it will return false.

**6)Explain negative number in javascript?**

Ans)”Negative number “ is a number which is obtained from dividing the number by zero.

**7)Can javascript code be broken into multiple lines**

Ans)Yes it is possible to break the javascript code to multiple lines using ‘\’.

**8)What are the undeclared variables in java script?**

Ans)Undeclared variables are the ones which does not exit and not declared in the program.

So while running the program; runtime error will be thrown.

**9)What are the undefined variables in java script ?**

Ans)Undefined variables are the ones which does exit in the program but are not defined or not assigned any value for the variables .

**Example:**

l

             console.log(b);//not defined or undecleared

             var a;

             console.log(a);//undefined

             var c = null;

             console.log(c); //null

**10)Explain prompt box in javascript ?**

Ans)Prompt box is used to get the input from the user by popping up the window to the user.

**11)Why to use ”this” keyword in javascript?**

Ans)’this’ keyword is used to refer the current object in the program . ‘this’ keyword is used to mailnly inside the method to refer to current variable or current object.

**12)What is “setTimeOut() function in javascript?**

Ans)”SetTimeOut” function is used to call the function given as a parameter after some time dalay.

Below is the jaavscript code for the same

Syntax: setTimeOut(function(){

alert(“This is mssg box”);

},5000);

**13)What is “setInterval()” function in javascript?**

Ans)

**14)What is “clearInterval()” function in javascript?**

Ans) “clearInterval()” function is used to clear the time set from function – “setInterval” .

**15)How to give comments in javascript?**

Ans)For single line comment – “//”

For multi-line comments – “/\* \*/”

**19)Why to use “===” operator in javascript?**

Ans)”===” operator is called strick operator and it returns true when both the values are same and returns false when the values are not same.

**20)How we can submit a form using javascript?**

Ans)For submitting form from javascript we have to use

Document.form[0].submit() method.

**21)Will javascript support automatic conversion of type?**

Ans)Yes javascript will support automatic conversion of type.

**22)How we can change the style of an element from javascript?**

Ans)Below code can be used to change the style of an element –

document.getElementById(“demo”).className = “testclass”; //camelcase

**23)What is javascript hosting?**

Ans) Hoisting is JavaScript's default behavior of moving declarations to the top of the current scope.

**24)what is javascript scope?**

Ans) Scope determines the accessibility (visibility) of variables.

JavaScript has 3 types of scope:

* Block scope
* Function scope
* Global scope
* Before ES6 (2015), JavaScript had only **Global Scope** and **Function Scope**.
* ES6 introduced two important new JavaScript keywords: let and const.
* These two keywords provide **Block Scope** in JavaScript.
* Variables declared inside a { } block cannot be accessed from outside the block:

{  
  let x = 2;  
}

## Local Scope

Variables declared within a JavaScript function, become **LOCAL** to the function.

function myFunction() {  
  let carName = "Volvo";  
  // code here CAN use carName  
}

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

## Function Scope

JavaScript has function scope: Each function creates a new scope.

Variables defined inside a function are not accessible (visible) from outside the function.

Variables declared with var, let and const are quite similar when declared inside a function.

They all have **Function Scope**:

function myFunction() {  
  var carName = "Volvo";   // Function Scope  
}

function myFunction() {  
  let carName = "Volvo";   // Function Scope  
}

function myFunction() {  
  const carName = "Volvo";   // Function Scope  
}

## Global JavaScript Variables

A variable declared outside a function, becomes **GLOBAL**.

let carName = "Volvo";  
// code here can use carName  
  
function myFunction() {  
// code here can also use carName  
}

## Global Scope

Variables declared **Globally** (outside any function) have **Global Scope**.

**Global** variables can be accessed from anywhere in a JavaScript program.

Variables declared with var, let and const are quite similar when declared outside a block.

They all have **Global Scope**:

var x = 2;       // Global scope

let x = 2;       // Global scope

const x = 2;       // Global scope

**25\_Javascript error?**

## Ans) Throw, and Try...Catch...Finally

The try statement defines a code block to run (to try).

The catch statement defines a code block to handle any error.

The finally statement defines a code block to run regardless of the result.

The throw statement defines a custom error.

<p id="demo"></p>  
  
<script>  
try {  
  adddlert("Welcome guest!");  
}  
catch(err) {  
  document.getElementById("demo").innerHTML = err.message;  
}  
</script>

25)

anS)

|  |  |
| --- | --- |
| **Method** | **Description** |
| Number() | Returns a number, converted from its argument |
|  |  |
| parseFloat() | Parses a string and returns a floating point number |
| parseInt() | Parses a string and returns an integer |

|  |  |
| --- | --- |
| **Method** | **Description** |
| new Map() | Creates a new Map |
| set() | Sets the value for a key in a Map |
| get() | Gets the value for a key in a Map |
| delete() | Removes a Map element specified by the key |
| has() | Returns true if a key exists in a Map |
| forEach() | Calls a function for each key/value pair in a Map |
| entries() | Returns an iterator with the [key, value] pairs in a Map |
| **Property** | **Description** |
| size | Returns the number of elements in a Map |

## The HTML DOM Document Object

---------------------------------------------------------------------------------------

Adding and Deleting Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.createElement(*element*) | Create an HTML element |
| document.removeChild(*element*) | Remove an HTML element |
| document.appendChild(*element*) | Add an HTML element |
| document.replaceChild(*new, old*) | Replace an HTML element |
| document.write(*text*) | Write into the HTML output stream |

Finding HTML Elements

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.getElementById(*id*) | Find an element by element id |
| document.getElementsByTagName(*name*) | Find elements by tag name |
| document.getElementsByClassName(*name*) | Find elements by class name |

26)What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

The W3C DOM standard is separated into 3 different parts:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

27)What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

* The HTML elements as **objects**
* The **properties** of all HTML elements
* The **methods** to access all HTML elements
* The **events** for all HTML elements

In other words:**The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

## 28)What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

29)HTTP Methods

* **GET**
* **POST**
* **PUT**
* **HEAD**
* **DELETE**
* **PATCH**
* **OPTIONS**
* **CONNECT**
* **TRACE**

**30)What is the difference between PX, EM and Percent?**

* Pixel is a static measurement, while percent and EM are relative measurements. The size of an EM or percent depends on its parent. If the text size of body is 16 pixels, then 150% or 1.5 EM will be 24 pixels (1.5 \* 16). Look at [CSS Units](https://www.w3schools.com/cssref/css_units.asp) for more measurement units.

31)

# Attribute presence and value selectors

* Item 1
* Item 2
* Item 3

li[class] {

font-size: 200%;

}

li[class="a"] {

background-color: yellow;

}

li[class~="a"] {

color: red;

}

<h1>Attribute presence and value selectors</h1>

<ul>

<li>Item 1</li>

<li class="a">Item 2</li>

<li class="a b">Item 3</li>

<li class="ab">Item 4</li>

</ul>

<https://www.w3schools.com/css/css_combinators.asp>

32)What is difference between property binding and attr binding

Ans)1. [ngClass]=”expression” //property binding

[attr.className]=”c1” //attribute binding

2.Some attributes are not natively supported for elements

🡪[colspan]=”colVal”

🡪[attr.colspan]=”colVal”

3.Angular encourages to use property binding

🡪attrbute binding

33)Alternative of writing ngModel

🡪property binding and event binding on the same element

E.g :<input [value]=”data” (input)=”$event.target.value” />

Angular provides a built-in directive called “ngModel”

34)Why are using this multiple routers?

-Avoid this use case in applications

-you can inject components

Queryselector is defines particular section or Div

document.querySelector("p").style.backgroundColor = "red";

1)What is function In javascript?

A JavaScript function is **a block of code designed to perform a particular task.**

**Ex:**

**function greet() {**

**console.log("Hello there!");**

**}**

**// calling the function**

**greet();**

**2)** **Closures**: In a function calling object outside (global variable) and inside (local variable) , then we can call it closures.

Ex: function salutation () {  
    let name = 'Aarush';  
  
    function greet() {  
        console.log(`Hello ${name}!`);  
    }  
    return greet;  
}  
  
let wish = salutation();  
wish();

**JavaScript - Main Concepts:**

**Memorization**: optimizing techniques that can be used to reduce  
time-consuming calculations by saving previous input to something called cache and returning the result from it

**Call, Apply, Bind – Method:**

The call() method takes arguments **separately**.

The apply() method takes arguments as an **array**.

**Example of call():**

const person = {  
  fullName: function(city, country) {  
    return this.firstName + " " + this.lastName + "," + city + "," + country;  
  }  
}  
  
const person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
  
person.fullName.call(person1, "Oslo", "Norway");

**Example of apply():**

const person = {  
  fullName: function(city, country) {  
    return this.firstName + " " + this.lastName + "," + city + "," + country;  
  }  
}  
  
const person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
  
person.fullName.apply(person1, ["Oslo", "Norway"]);

Use this Reference link: <https://www.geeksforgeeks.org/explain-call-and-apply-methods-in-javascript/>

**Call() Method:**The call method is basically used to invoke the function with different**this** object. In JavaScript, **this** refers to an object. It depends on how we are calling a particular function. In the global scope, **this** refers to the global object **window**. Inside function also **this** refers to the global object **window.**

In strict mode, when we use any function then **this** refers to **undefined.**In functions like call, this could refer to a different object. With the help of the **call** method, we can invoke a particular function with different objects.

**Syntax**

object.objectMethod.call( objectInstance, arguments )

**Example:single person data**

const obj = {

        firstName: "First\_name",

        lastName: "Last\_name",

        printName: function () {

            console.log(this.firstName

                + " " + this.lastName);

        }

    };

    obj.printName();

Example:multiple persons data

const person = {  
  **fullName**: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
const person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
const person2 = {  
  firstName:"Mary",  
  lastName: "Doe"  
}  
  
// This will return "John Doe":  
person.fullName.call(**person1**);

**Call():passing arguments:**

printname:function(city,man){

console.log(this.fistname+""+this.lastname+","+city+","+man);

}

}

const person1={

fistname:"manjudasd",

lastname:"devarala"

}

const person2={

fistname:"manjula",

lastname:"devarala"

}

person.printname.call(person1,"city","man");

**Apply() method:**Just like the call method we can also bind the function to any object. Using apply( ) method also we can invoke a given function with different objects.

**Syntax:**

object.objectMethod.apply(objectInstance, arrayOfArguments)

**Example:**

 const obj1 = {

        firstName: "First\_name",

        lastName: "Last\_name"

    };

    const obj2 = {

        firstName: "Sachin",

        lastName: "Tendulkar"

    };

    function printName() {

        console.log(this.firstName + " " + this.lastName);

    }

    printName.apply(obj2);

**Example:apply()**

const person={

printname:function(city,man){

console.log(this.fistname+""+this.lastname+","+city+","+man);

}

}

const person1={

fistname:"manjudasd",

lastname:"devarala"

}

const person2={

fistname:"manjula",

lastname:"devarala"

}

person.printname.apply(person2,["city","man"]);

**output:”**manjuladevarala,city,man”

**Bind():**

With the bind() method, an object can borrow a method from another object.

The example below creates 2 objects (person and member).

The member object borrows the fullname method from the person object:

**Example:**

const person = {  
  firstName:"John",  
  lastName: "Doe",  
  fullName: function () {  
    return this.firstName + " " + this.lastName;  
  }  
}  
  
const member = {  
  firstName:"Hege",  
  lastName: "Nilsen",  
}  
  
let fullName = person.fullName.bind(member);

fullName();

**Example:**

const person ={

fistname:"manjudasd",

lastname:"devarala",

fullName:function(){

console.log(this.fistname+""+this.lastname);

}

}

const member ={

fistname:"manjula",

lastname:"devarala"

}

/\* let printname=person.printname.bind(person2); \*/

fullName = person.fullName.bind(member);

fullName();

Math.max():

Math.max(1,2,3);  // Will return 3

Math.max.apply(null, [1,2,3]); // Will also return 3

<https://www.geeksforgeeks.org/javascript-function-binding/>

HTTP request methods are **GET, POST, PUT, PATCH, and DELETE**

Closures, Hoisting

Difference between Let and const

Map, Filter, Reduce – Methods

**Q) What are different DOM methods?**

A) HTML DOM getElementsByName() Method. HTML DOM getElementsByTagName() Method. HTML DOM query Selector () Method. HTML DOM querySelectorAll () Method.

**Q) Difference between Undefined and not defined?**

A) The main difference between "undefined" and "not defined" is that **"undefined" is a value that can be assigned to a variable, while "not defined" indicates that a variable does not exist**

**Q) Call-backs?**

A) A call back is **a function passed as an argument to another function**. This technique allows a function to call another function. A call back function can run after another function has finished.

**Q) What is Propagation?**

A) Propagation refers to **how events travel through the Document Object Model (DOM) tree**.

**Q)Event bubbling**?

A) **Event Bubbling** is a concept in the DOM (Document Object Model). It happens when an element receives an event, and that event bubbles up (or you can say is transmitted or propagated) to its parent and ancestor elements in the DOM tree until it gets to the root element.

const body = document.getElementsByTagName("body")[0]

const div = document.getElementsByTagName("div")[0]

const span = document.getElementsByTagName("span")[0]

const button = document.getElementsByTagName("button")[0]

body.addEventListener('click', () => {

console.log("body was clicked")

})

div.addEventListener('click', () => {

console.log("div was clicked")

})

span.addEventListener('click', () => {

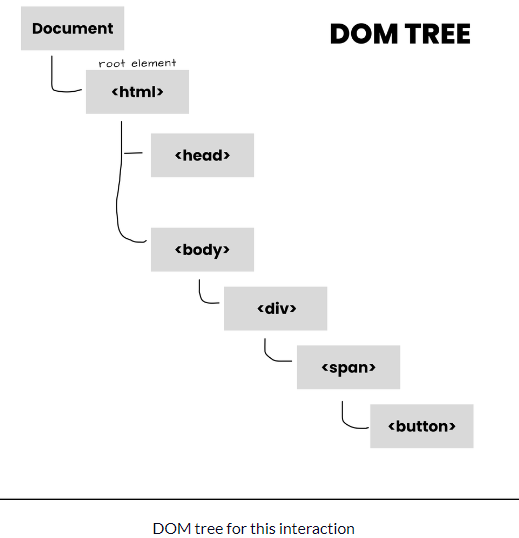
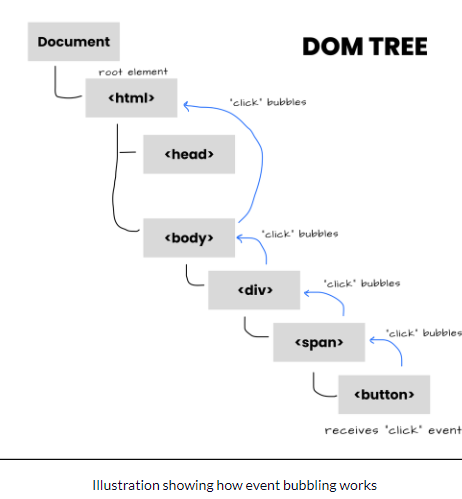
console.log("span was clicked")

})

button.addEventListener('click', () => {

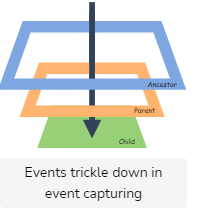
console.log("button was clicked")

})

**Q)event Capturing?**

A) **Event capturing** is one of two ways to do event propagation in the HTML DOM. In event capturing, an event propagates from the outermost element to the target element. It is the opposite of [**event bubbling**](https://www.educative.io/edpresso/javascript-event-bubbling), where events propagate outwards from the target to the outer elements.



<html>

  <head></head>

  <body>

    <article id="ancestor" >

      article element

      <div id="parent" >

        div element

        <p id="child" >

          p element

        </p>

      </div>

    </article>

  </body>

</html>

<script>

  // Script to click event handler to capture on each element

  for(let elem of document.querySelectorAll('\*')) {

    elem.addEventListener("click", e => console.log("Capturing:", elem.tagName), true);

  }

</script>



**Q) What is Currying?**

A) Currying is a technique in functional programming that performs the transformation of a function with multiple arguments into several functions containing a single argument in a sequence.

function curry(f) { // curry(f) does the currying transform

return function(a) {

return function(b) {

return f(a, b);

};

};

}

**Q)**  <script>

        console.log('10'+10+10);

        console.log(10+10+'10');

    </script>

**Ans)** **101010**

**2010**

**Q)** **<div>**

**<div class="name">**

**</div>**

**</div>**

**Ans)** **div {**

**display:flex;**

**justify-content:center;**

**align-self:center;**

**align-items:center;**

**height:100vh;**

**}**

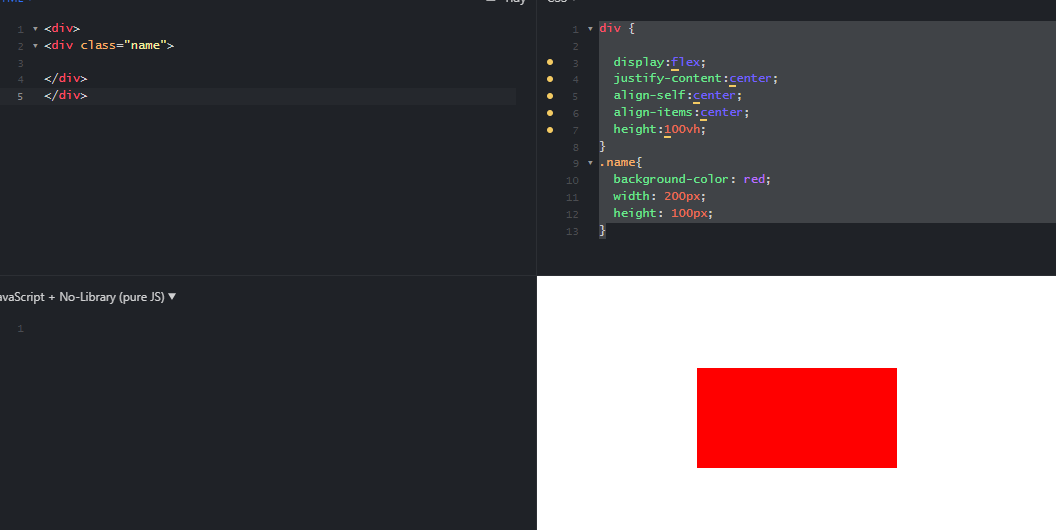
**.name{**

**background-color: red;**

**width: 200px;**

**height: 100px;**

**}**

****

**Q)**     const num = [1,2,3,4,5];

         for(let i in num){

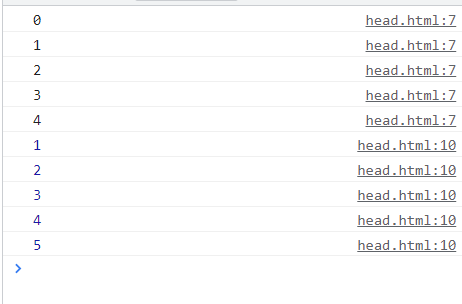
            console.log(i);

         }

         for(let i of num){

            console.log(i);

         }

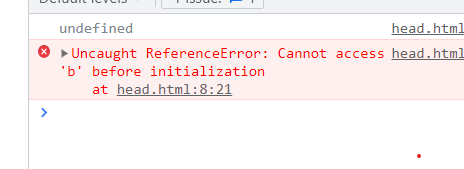
**Ans)** ****

 console.log(a);

        var a=5;

        console.log(b);

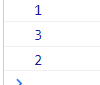
        let b =10;

Ans) 

**Q)**  console.log(1);

        setTimeout(() => console.log(2),0);

        console.log(3);

**Ans)** ****

**Q)**     const obj = {a:5,b:10};

        let obj1 = obj;

        obj1.c =15;

        console.log(obj.c);

**Ans)15**

**Q) What is VCS?**

A)Vcs Defines Version control system tool and called as Git. Which means it store the history whatever changes we have done In Git

Denouncing and throttling?

If, else, if else, switch conditions

**q)** <html>

    <footer>2</footer>

    <header>1</header>

</html>

### Ans) 2

1

### **The Object Datatype**

The object data type can contain:

1. An object  
2. An array  
3. A date

**Javascript Onclick functionality**

<h2>My First JavaScript</h2>

<button type="button" onclick="document.getElementById('demo').innerHTML = Date()">

Click me to display Date and Time.</button>

<p id="demo"></p>

**What is the javascript event flow?**

**Event Target**

**Event Bubbling**

**Event Capturing**

**Q) Why we Use JavaScript?**

JavaScript is used by programmers across the world **to create dynamic and interactive web content like applications and browsers,**

JavaScript is **commonly used for creating web pages**.

**Q)What is JavaScript Throttling?**

Throttling is used to call a function after every millisecond or a particular interval of time only the first click is executed immediately.

35)call ,apply ,bind javascript?

36)let ,var,const?

37)map?

38)hosting?

39)closures?

40)undefined ?

41)Not defined ?

42)memozation?

43)callback function?

44)trim()

45)positions differnce

46)flex properties

47)meta tags

48)Diff between “==” and “===”’ ? Which is true?

49)What is throttling?

A) Throttling is **a mechanism in Intel® Processors to reduce the clock speed when the temperature in the system reaches above TJ Max (or Tcase)**. This is to protect the processor and to indicate to the user that there is an overheating issue in their system that they need to monitor.

50)why we use javascript?

51)Javascript event flow?

52)javascript json?

**JSON** is a format for storing and transporting data. **JSON** is often used when data is sent from a server to a web page.

53)version

54)angular use

55)commands

56)pure and impure pipes?

Ans)A **pure pipe** is only called when Angular detects a change in the value or the parameters passed to a pipe. An **impure pipe** is called for every change detection cycle no matter whether the value or parameter(s) changes.

57)immutable?

58)annotions?

Ans) An annotation is extra information associated with a particular point in a document or other piece of information. It can be a note that includes a comment or explanation. Annotations are sometimes presented in the margin of book pages

59) annotations in angular?

Ans) Annotation is **a block of text that can be displayed over a node or connector**. Annotation is used to textually represent an object with a string that can be edited at runtime. Multiple annotations can be added to a node/connector.

60)semantic tags?

* <header>: t defines a header for a web page.
* <nav>: It defines a container for navigation links.
* <section>: This defines a section in a web page.
* <article>: This element contains the main part, containing information about the web page.
* <aside>: The <aside> content is often placed as a sidebar in a document.
* <footer>: It defines a footer for a document or a section.

Ans) Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

* <article>
* <aside>
* <details>
* <figcaption>
* <figure>
* <footer>
* <header>
* <main>
* <mark>
* <nav>
* <section>
* <summary>
* <time>

## 61) Nesting <article> in <section> or Vice Versa?

The <article> element specifies independent, self-contained content.

The <section> element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.

**62) What are mixins in CSS?**

The @mixin directive **lets you create CSS code that is to be reused throughout the website**. The @include directive is created to let you use (include) the mixin.

### **[Sass @mixin and @include - W3Schools](https://www.w3schools.com/sass/sass_mixin_include.php)**

**63) event bubbling?**

[**https://javascript.info/bubbling-and-capturing**](https://javascript.info/bubbling-and-capturing)

**64)box model?**

     .cleardu{

                width: 100px;

                height: 100px;

                padding:10px;

                box-sizing: border-box;

                border:2px solid #000;

                background-color: rgb(66, 194, 7);

            }

            .cleardu1{

                width: 100px;

                height: 100px;

                padding:10px;

                box-sizing: content-box;

                border: 2px solid #000;

                background-color: aqua;

            }

            .cleardu2{

                width: 100px;

                height: 100px;

                padding:10px;

                border: 2px solid #000;

                background-color: rgb(177, 58, 58);

            }

        </style>

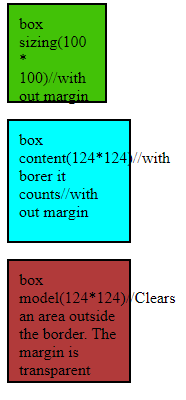
    </head>

    <body>

        <p class="cleardu">box sizing(100 \* 100)//with out margin</p>

        <p class="cleardu1">box content(124\*124)//with borer it counts//with out margin</p>

        <p class="cleardu2">box model(124\*124)//Clears an area outside the border. The margin is transparent</p>

****

**65)box sizing?**

# **66) CSS Pseudo-classes?**

Ans) A pseudo-class is used to define a special state of an element.

/\* unvisited link \*/  
a:link {  
  color: #FF0000;  
}  
  
/\* visited link \*/  
a:visited {  
  color: #00FF00;  
}  
  
/\* mouse over link \*/  
a:hover {  
  color: #FF00FF;  
}  
  
/\* selected link \*/  
a:active {  
  color: #0000FF;  
}

## Pseudo-classes and HTML Classes

a.highlight:hover {  
  color: #ff0000;  
}

## CSS - The :first-child Pseudo-class

p:first-child {  
  color: blue;  
}

## CSS - The :lang Pseudo-class

q:lang(no) {  
  quotes: "~" "~";  
}  
</style>  
</head>  
<body>  
  
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>  
  
</body>

## 67) What are Pseudo-Elements?

## Ans) A CSS pseudo-element is used to style specified parts of an element.

## 68) What are Pseudo-Elements?

## Ans) selector::pseudo-element {   property: value; }

## The ::first-line Pseudo-element

## p::first-line {   color: #ff0000;   font-variant: small-caps; }

## p::first-letter {   color: #ff0000;   font-size: xx-large; }

## h1::before {   content: url(smiley.gif); }

## h1::after {   content: url(smiley.gif); }

## ::selection {   color: red;   background: yellow; }

## 69)px ,%, vw/vh ,rem, em?

## Px(Absolute size)

## % ( Realated to parent width/height)

## Vw/vh (Realated to sreensize)

## Rem ( related to root element font)

## Em ( related to its font size)

**70)** **The flex container properties are:**

* [flex-direction](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-direction)
* [flex-wrap](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-wrap)
* [flex-flow](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-flow)
* [justify-content](https://www.w3schools.com/css/css3_flexbox_container.asp#justify-content)
* [align-items](https://www.w3schools.com/css/css3_flexbox_container.asp#align-items)
* [align-content](https://www.w3schools.com/css/css3_flexbox_container.asp#align-content)

## 71) conditional ternary operator?

## Ans) The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark ( ? ), then an expression to execute if the condition is truthy followed by a colon ( : ), and finally the expression to execute if the condition is falsy.

**72)Data types?**

**JavaScript Data Types** · 1. String · 2. Number · 3. Bigint · 4. Boolean · 5. Undefined · 6. Null · 7. Symbol · 8. Object ...

**73) Doctype?**

### **[HTML doctype declaration - W3Schools](https://www.w3schools.com/tags/tag_doctype.asp)**

[https://www.w3schools.com › tags › tag\_doctype](https://www.w3schools.com/tags/tag_doctype.asp)

All **HTML** documents must start with a <!**DOCTYPE**> declaration. The declaration is not an **HTML** tag. It is an "information" to the browser about what document type ...

**What is <! DOCTYPE with example?**

DOCTYPE> tag is used to inform the browser about the version of HTML used in the document. It is called as the document type declaration (DTD).  
...  
Syntax.

|  |  |
| --- | --- |
| **Display** | **None** |
| Start tag/End tag | Start tag only |
| Usage | Structural |

## 74) ****What are some of the common lists that can be used when designing a page?****

**Ans)** You can insert any or a combination of the following list types:  
– ordered list  
– unordered list  
– definition list  
– menu list  
– directory list

– ordered list

 <ul>

        <li>C</li>

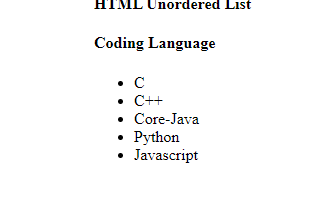
        <li>C++</li>

        <li>Core-Java</li>

        <li>Python</li>

        <li>Javascript</li>

    </ul>



<h4>Coding Language</h4>

    <ul style="list-style-type:square;">

        <li>C</li>

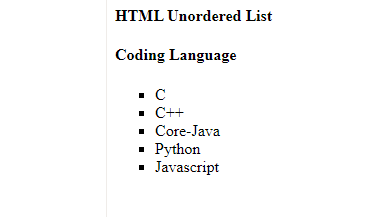
        <li>C++</li>

        <li>Core-Java</li>

        <li>Python</li>

        <li>Javascript</li>

    </ul>



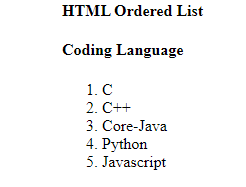
<ol>

<li>Item1</li>

<li>Item2</li>

<li>Item3</li>

</ol>



 <ol type="I">

        <li>C</li>

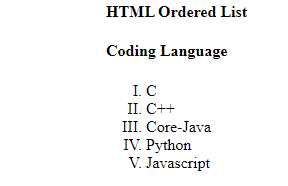
        <li>C++</li>

        <li>Core-Java</li>

        <li>Python</li>

        <li>Javascript</li>

    </ol>



 <ol start="12">

        <li>C</li>

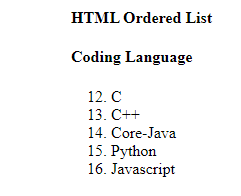
        <li>C++</li>

        <li>Core-Java</li>

        <li>Python</li>

        <li>Javascript</li>

    </ol>



**Description List:**A description list is a type of list where each item has a description. It is also known as a definition list. The <dl> tag is used to create description list, the <dt> tag defines the item, and the <dd> tag describes each item in list.

**Syntax:**

<dl> Contents... </dl>

The HTML definition list contains following 3 tags:

* [**<dl>**](https://www.geeksforgeeks.org/html-dl-tag/): It defines the start of the list.
* [**<dt>**](https://www.geeksforgeeks.org/html-dt-tag/): It defines a item.
* [**<dd>**](https://www.geeksforgeeks.org/html-dd-tag/)**:** It defines the description of each item.

 <h4>Description List</h4>

    <strong>Developers Life</strong>

    <dl> <dt>Code</dt>

        <dd>- Code all day!</dd>

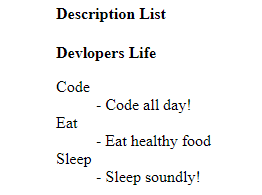
            <dt>Eat</dt>

        <dd>- Eat healthy food</dd>

            <dt>Sleep</dt>

        <dd>- Sleep soundly!</dd>

    </dl>



**75)** function cName(){

            return;

            {

                mes: "javascript"

            }

        }

        console.log(cName());

**Ans)undefined (y because return after semicolon )**

**76)**  function cName(){

            return{

                mes: "javascript"

            }

        }

        console.log(cName());

**Ans)** ****

**77)**  function cName(){

            return

{//if I use flower bracket is in down

                mes: "javascript"

            }

        }

        console.log(cName());

**Ans)undefined (y because**

**76)**  const numbers =[

            1,

            [3, [2, 8, [12]], 9]

            [5],

            [12 [[5]]],

            [100, [23, 45]]

        ];

**Ans)[1, 3, 2, 8, 12, 9, 5, 12, 5, 100, 23, 45]**

**77)what is Software**

**Ans)**Software is a program or collection of programs which will solve particular problem.

Types of software:

1.standalone app –in our system only login done (means it will work only our system ).

2.client-server – some of the application run to another system is called client ap

**78)** **Definition of HTTP**

**Ans)** (**HyperText Transfer Protocol**) The communications protocol used to connect to Web servers on the Internet or on a local network (intranet).

**79)** Bootstrap CDN

If you don't want to download and host Bootstrap yourself, you can include it from a CDN (**Content Delivery Network**).

**80) Top 10 Features of ES6**

* let and const keywords :
* Arrow Functions.
* Multi-line Strings.
* Default Parameters.
* Template Literals.
* Destructuring Assignment.
* Enhanced Object Literals.
* Promises.

**81)Type infernce**

Ans) The type of the x variable is inferred to be number. This kind of inference takes place when initializing variables and members, setting parameter default values, and determining function return types.

In most cases, type inference is straightforward.

Example1

let x = 3;

let x: number

Example2

let x = [0, 1, null];

let x: (number | null)[]

**82) What are annotations in Angular?**

Annotation is **a block of text that can be displayed over a node or connector**. Annotation is used to textually represent an object with a string that can be edited at runtime. Multiple annotations can be added to a node/connector.

Example:

@ComponentAnnotation

import {

  ComponentAnnotation as Component,

} from '@angular/core';

export class ComponentAnnotation extends DirectiveMetadata {

  constructor() {

  }

}

**83)enum**

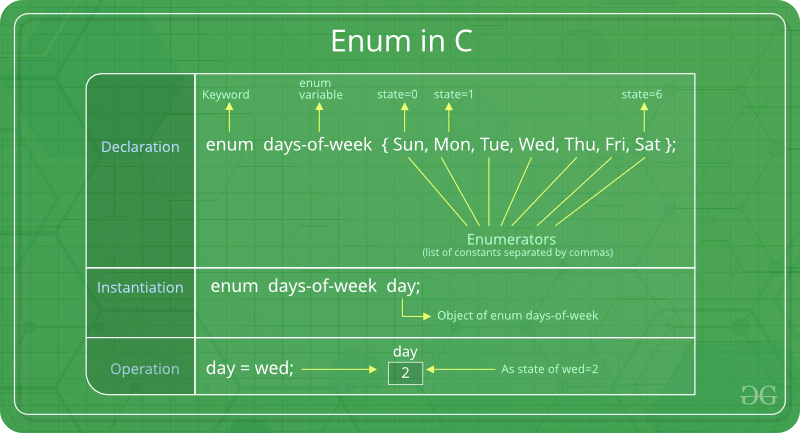
enum emp\_typ { designer=92,developer=94,tester=95,architecture=97,sysadmin=98,netadmin=99};

        var myDes=emp\_typ.netadmin;

        console.log(myDes);

        console.log(emp\_typ[95]);//output tester:95

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.



**84)function passeing in annotation?**

Ans)

function add(x:number,y:number,z?:number):number{

        if(z!=undefined){

            return x+y+z;

        }

        else{

        return x+y;

        }

    }

    var res = add(10,20);

    console.log(res);

    var res2:number;//output:30

    res2=add(45,56,55);

    console.log(res2)//output:156

85) What is void in Angular?

Ans)Similar to languages like Java, void is **used where there is no data**. For example, if a function does not return any value then you can specify void as return type. There is no meaning to assign void to a variable, as only null or undefined is assignable to void.

# 86)TypeScript Data Type – Void

Ans) Similar to languages like Java, void is used where there is no data. For example, if a function does not return any value then you can specify void as return type.

Example:

function sayHi(): void {

console.log('Hi!')

}

let speech: void = sayHi();

console.log(speech); //Output: undefined

There is no meaning to assign void to a variable, as only null or undefined is assignable to void.

Exampple:

let nothing: void = undefined;

let num: void = 1; // Error

87)

//suppose if i give more parameters is called rest paramaters

    function add(...x){//three dots 3 called spread operators

        console.log(x);//output (3) [false, 34, 55]

    }

    add(12, 34, 55);

    add(11,22,33,44,55,66,77);

88)spread operator compare to normal hard code If I use es6 property spread operator easy

 //very hard code

    var x=[11,22,33];

    var y=[44,55,66];

    var z=[x,y];

    console.log(z);//output[ [11, 22, 33], [ 44, 55, 66] ]

 //(2) [Array(3), Array(3)]

    0: (3) [11, 22, 33]

    1: (3) [44, 55, 66]

    //

    //hard code

    var x=[11,22,33];

    var y=[44,55,66];

    var z=[x[0],x[1],x[2],y[0],y[1],y[2]];

    console.log(z);//output[ [11, 22, 33, 44, 55, 66] ]

//

    //

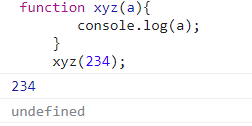
    //pure spread operator(very simple)

    var w = [...x,...y];

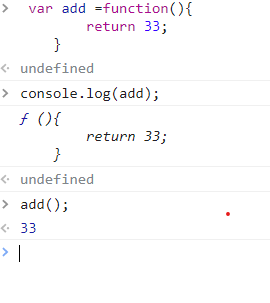
    console.log(w);//output[ [11, 22, 33, 44, 55, 66] ]

//(7) [11, 22, 33, 44, 55, 66, 77]

89)



90)



91)call back function?

Def:a callback function is a function passed into another function as an argument, which is then invoked inside the outer function come kind of router or action.

  //we can pass functions to another function and exceuted

     var add =function(){

        consle.log("hello")

     }

     function xyz(a){

        a();//is called call back function

     }

     xyz(add);

     //output:hello

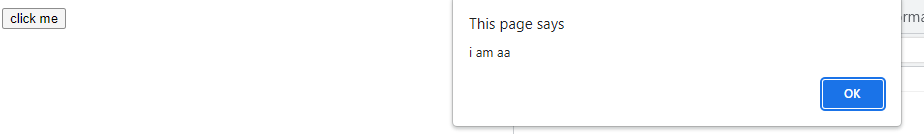
92) callback function examples

Ans) when I click alert mssg will come

Example1: function aa(){

        alert("i am aa");

     }



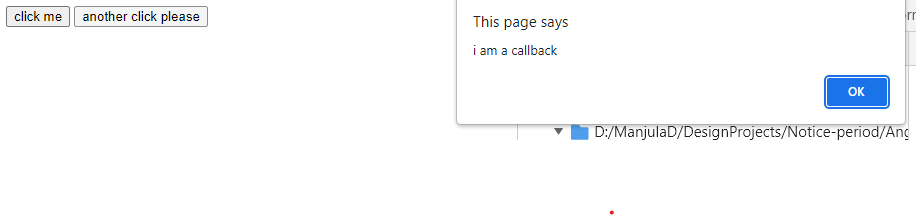
Example:2

document.getElementById('b1').addEventListener('click',function(){

        alert("i am a callback");

     });

Output:



Example3: passing arguments to callback function

var x=[10,20,30,40,50,60];

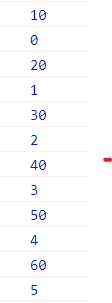
     var y=x.map(function(a,b){

        console.log(a);

        console.log(b);

     })

Output:



Example :using map in callback function

Ans)

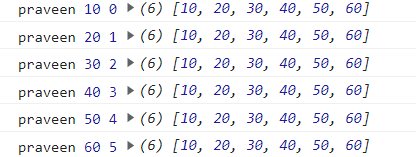
 var x=[10,20,30,40,50,60];

     var y=x.map(function(a,b,c){

        console.log("praveen",a,b,c);

         })

Output:



93)Higher order functions in javascript?

Ans) **a function which takes another function as an argument or returns a function** is known as a higher order function. Let's deep dive a bit to see both types of implementation, that is: Passing a function as an argument to another function. Returning a function from another function.

1)every()

2)filter()

3)ForEach()

4)reduce()

5)map()

6) setTimeout()var

**a)every():**Checks if every element in an array pass a test

Notes: for every value of x this function will called.

every function based on return type it desides iterate or not iterate alo desided

## Definition and Usage

The every() method executes a function for each array element.

The every() method returns true if the function returns true for all elements.

The every() method returns false if the function returns false for one element.

The every() method does not execute the function for empty elements.

The every() method does not change the original array

1) var x=[12,23,34,45,56,67,78]

     var k=x.every(function(){

        console.log('hello');

     })

     //output: hello

2)     var x=[12,23,34,45,56,67,78]

     var k=x.every(function(){

        console.log('hello');

        return true;//7 times its executed that is y 7 hello will get

     })

     //output: 7hello

3)

     var x=[12,23,34,45,56,67,78]

     var k=x.every(function(a,b,c){

        console.log('hello',a,b,c);

        return true;

     })

     console.log(k);//true

4)  var x=[12,23,34,45,56,67,78]

     var k=x.every(function(a,b,c){

        if(a==34){

            return false;

        }

        return true;

     })

     console.log(k);//false

5) var x=[12,23,34,45,56,67,78]

     var k=x.every(function(a,b,c){

        if(a==NaN){

            return false;

        }

        return true;

     })

     console.log(k);//true

6)

var x=[12,23,34,'pp',45,56,67,78]

     var k=x.every(function(a,b,c){

        if(isNaN(a)){

            return false;

        }

        return true;

     })

     console.log(k);//false pp

**b) forEach() :** Calls a function for each array element

 1) var x=[12,23,34,'pp',45,56,67,78]

     var k=x.forEach(function(a,b,c){

       console.log(a);

     })

Output:



2)  var x=[12,23,34,'pp',45,56,67,78]

     var k=x.forEach(function(a,b,c){

       console.log(a);

       return true;

     })

Output:



**c)filter():** Creates a new array with every element in an array that pass a test

1)  var x=[12,23,34,'pp',45,56,67,78]

     var k=x.filter(function(a,b,c){

        if(isNaN(a)){

            return false;

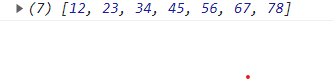
        }

        return true;

     })

     console.log(k);

output:

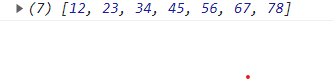
2)

 var x=[12,23,34,'pp',45,56,67,78]

     var k=x.filter(a=>isNaN(a)?false:true);

     console.log(k);

output:



**c)reduce():** Reduce the values of an array to a single value (going left-to-right)

## Definition and Usage

The reduce() method executes a reducer function for array element.

The reduce() method returns a single value: the function's accumulated result.

The reduce() method does not execute the function for empty array elements.

The reduce() method does not change the original array.

1)  var x=[12,23,34,45,56,67,78];

    var k=x.reduce(function(a){

        console.log(a);

    })

Output:



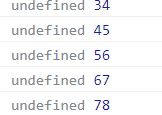
2)  var x=[12,23,34,45,56,67,78];

    var k=x.reduce(function(a,b){

        console.log(a,b);

    });

Output:



3)  var x=[12,23,34,45,56,67,78];

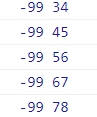
    var k=x.reduce(function(a,b){

        console.log(a,b);

        return -99;

    });

Output:



4) var x=[12,23,34,45,56,67,78];

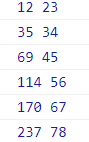
    var k=x.reduce(function(a,b){

        console.log(a,b);

        return a+b;

    });

Output:



5) var x=[12,23,34,45,56,67,78];

    var k=x.reduce(function(a,b){

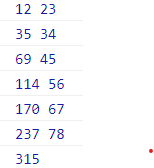
        console.log(a,b);

        return a+b;

    });

    console.log(k);

output:



6)  var w = x.reduce((a,b)=>a+b);

    console.log(w);

output:315

**lamda function:**

var x = [12,23,45,67,88];

var k = x.reduce((a,b)=>{

console.log(a,b);

return a+b;

})

**Example:**

var x = [12,23,45,56,67]

var k = x.reduce((a,b)=>a+b);

console.log(k)//203

**d)map():** Creates a new array with the result of calling a function for each array element

important notes:

a)it will be stored in new array

b)Map returns an array

1) var x=[12,23,34,45,56,67,78];

    var k=x.map(function(a,b){

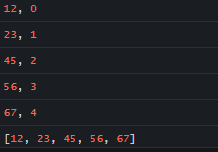
        console.log(a,b);

        return a;

    });

    console.log(k);

output:



2)  var x=[12,23,34,45,56,67,78];

    var k=x.map(function(a,b){

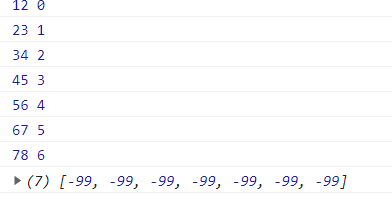
        console.log(a,b);

        return -99;

    });

    console.log(k);

Output:



3) var x=[12,23,34,45,56,67,78];

    var k=x.map(function(a,b){

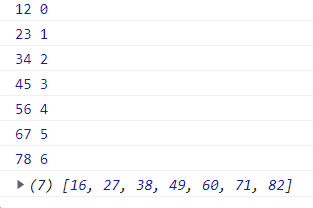
        console.log(a,b);

        return a+4;

    });

    console.log(k);

output:

4)

e)setTimeout also a higher order

1) setTimeout(function abc(){

        alert("hello");

    },5000

    )

2)  setInterval(function(){

        alert("hello");

    },5000

    )

3)    console.log("hello")

   setTimeout(function(){

    console.log("hellooooooooooooooo")

   },2300);

   var si=setTimeout(function(){

    console.log("example of setinterval")

   },4000);

   document.getElementById("b1").addEventListener("click",function(){

    clearInterval(si);

   });

**94)normal function?**

Ans)  var f1 = function(x,y){

            var z;

            z=x+y;

            return z;

        }

        var res=f1(12,23);

        console.log(res);

output:35

**95)arrow function?**

 var f1 = (x,y)=>{

            var z;

            z=x+y;

            return z;

        }

        var res=f1(12,23);

        console.log(res);

output:35

98)

  var add = (x,y)=>(return x+y;);

        var add1 = (x,y)=>x+y;

        var res2 = add(10,20);

        console.log(res2);//30

        console.log(add1(23,34));//57

        var inc = x=>++x;

        console.log(inc(12));//13

        var inc2 =  function(x){

            return x++;

        }

**99)higher order function?**

A function which can writen function then we called as high order function

var f1 = function(){

                var f11=function(){

                    console.log("i am f11");

                }

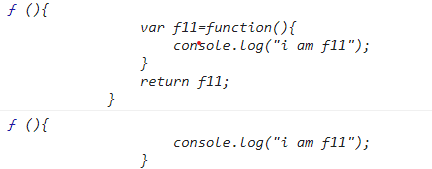
                return f11;

            }

            console.log(f1);

  console.log(f1());

**output:**

****

**100)**    var f1 = function(){

                var f11=function(){

                    console.log("i am f11");

                }

                return f11();

            }

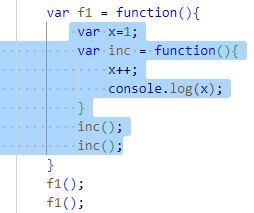
            console.log(f1);

            console.log(f1());

**Output:**

****

**101)clousers?**



Marked blue background called clouser

 /\* starts here\*/

            var x=1;

            f1 = function(){

                x++;

                console.log(x)

            }

            f1();//2

            f1();//3

            /\*ends here\*/

  1)/\* starts here\*/

            f1 = function(){

                var x=1;

                x++;

                console.log(x)//every time x is reintialzed

            }

            f1();//2

            f1();//2

            /\*ends here\*/

 2)/\* starts here\*/

             var x=1;

            f1 = function(){

                x++;

                console.log(x)

            }

            f1();//2

            f1();//3

            f2= function(){

                x++;

                console.log(x);

            }

            f2();//4

            /\*ends here\*/

**102)enhancing our components with pipes and Directives**

1. Directives in angular
2. Form related
3. Routing directives

**Directives in angular**

* + Ngclss
  + Ngstyle
  + Ngif
  + Ngfor
  + Ngswitch

**103)** onclick button textshould be hide and show?

<div id="m1">

    <h1 class="">angular26 works!</h1>

    <button (click)="chg()">click here</button>

    <h2 \*ngIf="flag">some text</h2>

</div>

**.ts**

 flag=false;

chg(){

    this.flag=!this.flag;

  }

**\*ngIf:**

<!--step1 starts here-->

<div id="m1">

  <h1>my heading</h1>

  <button (click)="chg()">click</button>

  <h1 \*ngIf="flag">Some text</h1>

</div>

<!--step1 eds here-->

**.css**

#m1 {

  border: 2px solid;

  padding: 10px;

  margin: 15px;

  background-color: blueviolet;

}

**.ts**

flag;

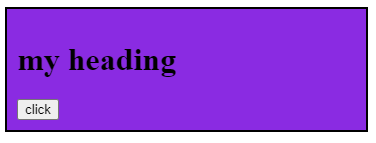
chg() {

    this.flag = !this.flag;

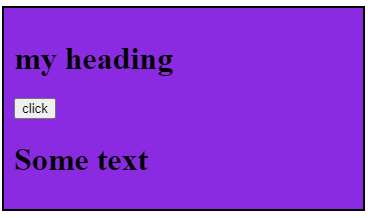
  }

**Output:**

**before clilck**

****

**After clicking**

****

<!--step2 starts here-->

<div class="m1">

  <input type="checkbox" [(ngModel)]="acc" />I agree<br />

  {{ acc }}

  <button>click to know</button>

</div>

<!--step2 eds here-->

**.ts**

 acc;

**output:**

****

  <!--step3 starts here-->

  <div class="m1">

    <input type="checkbox" [(ngModel)]="acc" />I agree<br />

    {{ acc }}

    <button [disabled]="acc" (click)="abc(acc.value)">click to know</button>

  </div>

  <!--step3 eds here-->

**.ts**

  acc;

****

<!--step4 starts here-->

  <div class="m1">

    <div>

        <input type="text" [(ngModel)]="x">

        <h1 \*ngIf="x%2==0">Its an even number</h1>

        <h1 \*ngIf="x%2!=0">Its an odd number</h1>

    </div>

</div>

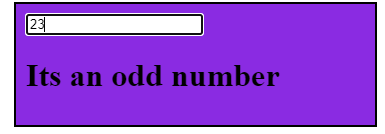
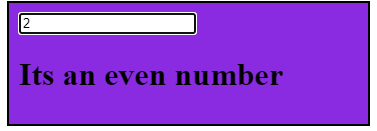
<!--step4 ends here-->

**.ts**

 abc(x: any){

    alert(x);

  }

**** ****

**Array Methods:**

**Splice():**

  //1) splice methods

            var ar1 = [10,20,30,40,50];

            ar1.splice(3,2,66);

            console.log(ar1);

            //output:(4) [10, 20, 30, 66]

  //Example:

            var ar2=[10,20,30,40,50,60]

            ar2.splice(3,2,0);

            console.log(ar2);

            //output:[10, 20, 30, ,0, 60]

            //example:

            var ar2=[10,20,30,40,50,60]

            ar2.splice(1);

            console.log(ar2);

            //output:[10]

//Example:

            var ar2=[10,20,30,40,50,60]

            ar2.splice(1,0);

            console.log(ar2);

            //output:[10, 20, 30, 40, 50, 60]

//Example:

            var ar2=[10,20,30,40,50,60]

            ar2.splice(1,3);

            console.log(ar2);

            //output:[10,50,60]

**Push():**

   //2) push methods

            var ar2 = [10,20,30,40,50,70];

            ar2.push(60);// after last index adding 60

            console.log(ar2);

             //output:(7) [10, 20, 30, 40, 50, 70, 60]

**Pop():**

 //3) pop methods

            var ar3 = [10,20,30,40,50,60,70];

            ar3.pop();//remove last index 70

            console.log(ar3);

             //output:(6) [10, 20, 30, 40, 50, 60]

**Shift():**

//4) shift methods

            var ar4 = [10,20,30,40,50];

            ar4.shift();//remove first element

            console.log(ar4);

             //output:(4) [20, 30, 40, 50]

**Unshift():**

//5) unshift methods

            var ar4 = [10,20,30,40,50,60];

            ar4.unshift(33);//adding 33 to first element

            console.log(ar4);

             //output:(7) [33, 10, 20, 30, 40, 50, 60]

notes:

            //push:adding last element(60)

            //pop:remove last element

            //shift:remove first element

            //unshift:adding first element(33)

**forEach():function for each array element**

**Notes:it wont check return true or false**

//6 forEach method

            //step1

            var ar6 = [10,20,30,40,50]

            ar6.forEach(function(){

                console.log("i am ar2");

            })

            //output:

            /\*"i am ar2"

            "i am ar2"

            "i am ar2"

            "i am ar2"

            "i am ar2"\*/

  //step2 passing arguments

            var ar6 = [10,20,30,40,50]

            ar6.forEach(function(a){

                console.log("i am ar2",a);

            })

            //output:

            /\*

            "i am ar2", 10

            "i am ar2", 20

            "i am ar2", 30

            "i am ar2", 40

            "i am ar2", 50

            \*/

//step3 passing arguments

             var ar6 = [10,20,30,40,50]

            ar6.forEach(function(a,b,c){

                console.log("i am ar2",a,b,c);

            })

            //output:

            /\*

            "i am ar2", 10, 0, [10, 20, 30, 40, 50]

            "i am ar2", 20, 1, [10, 20, 30, 40, 50]

            "i am ar2", 30, 2, [10, 20, 30, 40, 50]

            "i am ar2", 40, 3, [10, 20, 30, 40, 50]

            "i am ar2", 50, 4, [10, 20, 30, 40, 50]

            \*/

**Important Notes:**

/forEach is called callback function (with out using for loop we can process entire data)

//print even numbers

             var ar6 = [12,23,34,45,56,67,78,89,90]

            ar6.forEach(function(a,b,c){

                if(a%2==0){

                console.log(a);

                }

            })

            //output:12 34 56 78 90

//normal processs to print even numbers or this pure functioal programing language

            var ar6 = [12,23,34,45,56,67,78,89,90]

            for(i=0;i<ar6.length;i++){

                if(ar6[i]%2==0){

                console.log(ar6[i]);

                }

            }

            //output:12 34 56 78 90

**Every():it will check return true or false**

  /\* starts here\*/

     var x=[12,23,34,45,56,67,78]

     var k=x.every(function(){

        console.log('hello');

        return true;

     })

     //output: hello

     /\*ends here\*/

     var x=[12,23,34,45,56,67,78]

     var k=x.every(function(){

        console.log('hello');

        return true;

     })

     //output: 7hello

     /\*ends here\*/

    /\* starts here\*/

     var x=[12,23,34,45,56,67,78]

     var k=x.every(function(a,b,c){

        console.log('hello',a,b,c);

        return true;

     })

     console.log(k);//true

    /\*ends here\*/

     /\* starts here\*/

    var x=[12,23,34,45,56,67,78];

     var k=x.every(function(a,b,c){

        if(a==34){

            return false;

        }

        return true;

     })

     console.log(k);//false

     /\*ends here\*/

    /\* starts here\*/

    var x=[12,23,34,45,56,67,78]

     var k=x.every(function(a,b,c){

        if(a==NaN){

            return false;

        }

        return true;

     })

     console.log(k);//true

     /\*ends here\*/

     /\* starts here\*/

     var x=[12,23,34,'pp',45,56,67,78]

     var k=x.every(function(a,b,c){

        if(isNaN(a)){

            return false;

        }

        return true;

     })

     console.log(k);//false pp

/\*ends here\*/

**Filter():for**

//filter() array methods starts here

            /\*starts here\*/

            var x=[12,23,34,'pp',45,56,67,78]

            var k=x.filter(function(a,b,c){

                if(isNaN(a)){

                    return false;

                }

                return true;

            })

            console.log(k);

            /\*Ends here \*/

            /\*starts here\*/

            var x=[12,23,34,'pp',45,56,67,78]

            var k=x.filter(a=>isNaN(a)?false:true);

            console.log(k);

            /\*Ends here \*/

            //filter() array methods ends here

**Reduce():**

//reduce() array methods starts here

            /\*starst here\*/

            var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a){

                    console.log(a);

                });

            /\*ends here\*/

            /\*starst here\*/

            var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a,b){

                    console.log(a,b);

                });

            /\*ends here\*/

            /\*starst here\*/

            var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a,b){

                    console.log(a,b);

                    return -99;

                });

            /\*ends here\*/

                /\*starst here\*/

            var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a,b){

                    console.log(a,b);

                    return a+b;

                });

            /\*ends here\*/

                /\*starst here\*/

                var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a,b){

                    console.log(a,b);

                    return a+b;

                });

                console.log(k);

            /\*ends here\*/

                /\*starst here\*/

                var x=[12,23,34,45,56,67,78];

                var k=x.reduce(function(a,b){

                    console.log(a,b);

                    return a+b;

                });

                console.log(k);

                var w = x.reduce((a,b)=>a+b);

                console.log(w);//output:315

            /\*ends here\*/

              /\* starts here\*/

                var x=[12,23,34,45,56,67,78]

                var k=x.every(function(){

                    console.log('hello');

                    return true;

                })

                //output: hello

                /\*ends here\*/

                var x=[12,23,34,45,56,67,78]

                var k=x.every(function(){

                    console.log('hello');

                    return true;

                })

                //output: 7hello

                /\*ends here\*/

                /\* starts here\*/

                var x=[12,23,34,45,56,67,78]

                var k=x.every(function(a,b,c){

                    console.log('hello',a,b,c);

                return true;

            })

            console.log(k);//true

            /\*ends here\*/

            //reduce() ends here

**Map():**

//map() array methods starts here

            /\*starts here \*/

            var x=[12,23,34,45,56,67,78];

            var k=x.map(function(a,b){

                console.log(a,b);

                return a;

            });

            console.log(k);

            /\*ends here\*/

            /\*starts here \*/

            var x=[12,23,34,45,56,67,78];

            var k=x.map(function(a,b){

                console.log(a,b);

                return -99;

            });

            console.log(k);

            /\*ends here\*/

            /\*starts here \*/

            var x=[12,23,34,45,56,67,78];

            var k=x.map(function(a,b){

                console.log(a,b);

                return a+4;

            });

            console.log(k);

            /\*ends here\*/

            //map() array methods ends here

**Length:**

//7 length method

             var ar7 = [10,20,30,40,50,60,70];

            console.log(ar7.length);

            //output:6

**Important Notes: length only length method not length()**

**toString():**

//8 toString() methods

            var ar8 = [10,20,30,40,50,60,70];

            //The toString() method returns an array as a comma separated string:

            console.log(ar8.toString());

            //output::(7) “10, 20, 30, 40, 50, 60, 70”

**Join():**

    //9 join() methods

            var ar9 = [10,20,30,40,50,60,70];

           //The join() method returns an array as a \* separated string:

            console.log(ar9.join("\*"));

//output:10\*20\*30\*40\*50\*60\*70

**concat():**

  //10 concat() starts here

            const mygirls = ["manju", "sravanthi", "anju", "abanthi"];

            const myboys = ["ashok", "mallesh", "venky", "malli"];

            const mychildren = mygirls.concat(myboys);

            console.log(mychildren);

//output:manju,sravabthi,anju,abanthi,ashok,mallesh,veky,malli

**Constructor:**

//12 constructor method starts here

            const fruits1 = ["Banana", "Orange", "Apple", "Mango"];

            let text=fruits1.constructor;

            console.log(text);

            //function Array() { [native code] }

**copyWithin():**

 //13 copyWithin() method starts here

            const ar13 = ["Banana", "Orange", "Apple", "Mango","hjhjdjh","alkak"];

            ar13.copyWithin(2,0);

            console.log(ar13);

            //output:["Banana", "Orange", "Banana", "Orange", "Apple", "Mango"]

**Entries():**

  //14 entires() method starts here

            const ar14 = ["Banana", "Orange", "Apple", "Mango","hjhjdjh"];

            const f= ar14.entries();

            for(let x of f){

            console.log(x);

            }

            //

            /\*

            [0, "Banana"]

            [1, "Orange"]

            [2, "Apple"]

            [3, "Mango"]

            [4, "hjhjdjh"]

            \*/

**Fill():**

 //fill()

        const ar19=[10,20,30];

        ar19.fill("kiwi");

        console.log(ar19);

        //output:["kiwi", "kiwi", "kiwi"]

**Find():**

## Definition and Usage

The find() method returns the value of the first element that passes a test.

The find() method executes a function for each array element.

The find() method returns undefined if no elements are found.

The find() method does not execute the function for empty elements.

The find() method does not change the original array.

//find() method starts here

            const ar20=[10,20,30];

            console.log(ar20.find(checkage));

            function checkage(ar20){

                return ar20>18;

            }//output:20

            //find() method ends here

**findIndex():**

    //findIndex() method starts here

            const ar21=[10,20,30];

            console.log(ar21.find(checkage));

            function checkage(ar20){

                return ar21>18;

            }//output:1

            //find() method ends here

**From():**

  //from() starts here

           let textd = "ABCDEFG"

            const myArr = Array.from(textd);

            console.log(myArr)

            //output:["A", "B", "C", "D", "E", "F", "G"]

            //from() ends here

**Includes():**

//includes() starts here

            const ar22=[10,20,30,40,50];

            console.log(ar22.includes(10));

            //includes() ends here

Output:true

**Array.isArray():**

 //Array.isArray() starts here

            const ar23=[10,20,30,40,50];

            let result=Array.isArray(ar23);

            console.log(result);

            //output:true

            //Array.isArray() ends here

**Keys():**

  //keys() method starts here

            const fruitskey = ["Banana", "Orange", "Apple", "Mango"];

            const keys = fruitskey.keys();

            let textkey = "";

            for (let x of keys) {

            textkey += x + "<br>";

            }

            document.getElementById("demo").innerHTML = textkey;

            //output

            /\*

            0

            1

            2

            3

            \*/

            //keys() method ends here

**lastIndexOf():**

 //lastIndexOf() method starts here

            const fruitsindex = ["Apple", "Orange", "Apple", "Mango"];

            let index = fruitsindex.lastIndexOf("Apple");

            console.log(index);

            //output:2

            //lastIndexOf() method ends here

**Authentication:**

checking I am a revalsys employee or not

**Authorization:**

checking I m having permisssions for all modules or not

handiling for asynchronous operations

I particular task

**promise:**

**1.Not lazy(Eager)**

const promise=new promise(()=>{

        reolve(100);

    });

**2.data emits single value**

 const promise=new promise(()=>{

        reolve(100);

    });

**3.Data is used using then method**

const promise=new promise(()=>{

        reolve(100);

    });

    promise.then((value)=>{

        console.log(value)

    });

**4.cannot cancel**

**5.unicast**

const promise=new promise(()=>{

        reolve(100);

    });

    promise.then((value)=>{

        console.log(value)

    });

    promise.then((value)=>{

        console.log(value)

    });

**Observable:**

**1.Lazy**

    obs=new observable((observer)=>{

        odserver.next(100);

    })

**If we subscribe the code then only obseravable work.**

**2.Emits multiple values over a period of time**

  obs=new observable((observer)=>{

        observer.next(100);

        observer.next(200);

        observer.next(300);

    })

**3.Data is used using subscriptions**

  obs=new observable((observer)=>{

        observer.next(100);

    })

    const sub=obs.subscribe(value)=>{

        console.log(value);

    }

**4.Can be cancelled**

**5.multicast**

const promise=new promise(()=>{

        reolve(100);

    });

    obs=new observable((observer)=>{

        odserver.next(100);

    })

    const sub1=obs.subscribe(value)=>{

        console.log(value);

    }

    const sub2=obs.subscribe(value)=>{

        console.log(value);

    }

**1)How to pass the data from onecomponent to another component how many ways?**

Ans: There are 4 ways

              1)@Input -- this decorator to share the data parent to child component

              2)@output --- this decorator to share the data child to parent component

              3)@viewchild --ViewChild is used to select an element from component's template@ view child decorator

              while ContentChild is used to select projected content

              4)@viewchilderen -- To get the refence of one component to another component.

\*\*\*\*\*

**2)What is lazy loading in angular?**

Ans: Lazy loading is a technique in Angular that allows you to load JavaScript components asynchronously when a specific route is activated.

     This can add some initial performance during the initial load, especially if you have many components with complex routing. Improve the performance of the Angular

\*\*\*\*\*

**3)Explain the importance of components & Modules?**

Ans: Components-consists of html,css,ts,services & pipes Modules-like a big container-which has one or more components

\*\*\*\*\*

**4)How to make HTTP calls using angular?**

Ans: For making the API calls, we use HTTP like get(),post(), put(),delete()

**5)Explain the need of angular pipes?**

Ans: Transform the data to expected result.Inthis we have 2 types-pure and impure.

**6)What is Routing ?**

Ans: navigating between pages-declaration in app-routing.modules.ts

**Patterns:**

**Maxvalue:**

var x=[10,20,30,40,50];

var max=x[0];

for(i=0;i<=5;i++){

    if(max<x[i]){

    max=x[i];

  }

}

console.log(max);

//output:50

for(let i=1;i<=5;i++){

    for(j=1;j<=5;j++){

     document.write(i);

    }

    document.write("<br>");

}

document.write("<br>");

document.write("<br>");

//output:

/\*

000000

111111

222222

333333

444444

555555

\*/

for(let i=1;i<=5;i++){

    for(j=1;j<=5;j++){

     document.write(j);

    }

    document.write("<br>");

}

document.write("<br>");

document.write("<br>");

//output:

/\*

12345

12345

12345

12345

12345

\*/

//for j=1 inside forloop exicuted 5times

for(let i=1;i<=5;i++){

    for(j=1;j<=5;j++){

        if(j<=i){

        document.write(j);

        }

    }

    document.write("<br>");

}

document.write("<br>");

document.write("<br>");

//output:

/\*

1

12

123

1234

12345

\*/

//ends here

//starts here

for(let i=1;i<=5;i++){

    for(j=1;j<=5;j++){

      if(j<=i){

        document.write(i);

        }

    }

    document.write("<br>");

}

document.write("<br>");

document.write("<br>");

//output

/\*

1

22

333

4444

55555

\*/

//ends here

**Constructor function:**

     function Employee(fn,ln,a,g,l,q){

            var firstname=fn;//variable

            var lastname=ln;

            this.age=a;//property

            this.location=l;//property

        }//method

        var e1 = new Employee("ashok","mancha",28,"male","guntur","masters");

        var e2 = new Employee("manjula","mancha",26,"female","nellore","masters");

        console.dir(e1);

        console.dir(e2);

//output:

{

age: 25,

qualification: "masters"

}

**Array constructor function and normal array:**

     var ar1 = [10,20,30,40,50]; //sort hand way of creating array //array literals

        var ar2 =  new Array(10,20,30,40,50); //creating array from array constructor function

        console.dir(ar1);

        console.dir(ar2);

        console.log(typeof(ar1));

        console.log(typeof(ar2));

console.log(typeof(document));//object

console.log(typeof(Document));//function

**http request:**

    //first start here

 function abc(){

        var http = new XMLHttpRequest();

        console.dir(http);

    }

    abc();

//first ends here

//second step starts here//

function abc(){

        var http = new XMLHttpRequest();

        http.open("get", "webdemo.html");

        console.dir(http);

    }

    abc();

//second step ends here//

//third step start here //

 function abc(){

        var http = new XMLHttpRequest();

        http.open("get", "webdemo.html");

        http.send();

        console.dir(http);

        http.onreadystatechange = function(){

            console.log("ready state changed", http.readyState)

        }

    }

    abc();

//third step ends here //

    //last step starts here

    function abc(){

        var http = new XMLHttpRequest();

        http.open("get", "webdemo.html");

        http.send();

        console.dir(http);

        http.onreadystatechange = function(){

            if(http.readyState==4){

                document.getElementById("edokati").innerHTML=http.respaonse;

            }

        }

    }

    abc();

    //last step ends here

         //start

         function abc(){

            var http = new XMLHttpRequest();

            console.log(http.readyState);

            console.dir(http);

            http.onreadystatechange = function(){

                console.log(http.readyState);

            }

            http.open("GET","http://isolveit.in/demos/angulardemo.html");

            http.send();

        }

        abc();

        //end

 <ul>

        <li>web server</li>

        <li>web client</li>

        <li>http</li>

        <p>Reloding of page chainging is called statelessness</p>

        <li>http request:

            <ul>

                <li>address bar</li>

                <li>anchor tag</li>

                <li>form</li>

                <li>javascripit:based on below request det 2 methods

                    <ul>

                        <li>XMLHttpRequest</li>

                        <li>fetch()</li>

                    </ul>

                </li>

            </ul>

            <li>http response</li>

        </li>

    </ul>

<p>Binding the data methods and properties together in a small unit is called encapsulation</p>

 <li><strong>Objects:</strong>object is collection of methods and proerties is called objects</li>

        <li><strong>Objects:</strong>To group the data in a set or block is called object literals</li>

**Arrow functions: decreasing lines of code**

**Local path:** D:\ManjulaD\DesignProjects\Notice-period\Angular-classes\Angular-23-arrow-funtions

 //normal funtion starts here

     var f1= function(x,y){

        var z;

        z=x+y;

        return z;

     }

     var res=f1(12,13);

     console.log(res);

     //output:25

      //normal functio ends here

//arrow funtions starts here

     var f1= (x,y)=>{

        var z;

        z=x+y;

        return z;

     }

     var res=f1(12,13);

     console.log(res);

     //output:25

      //arrow functions ends here

//minifined arrow funtions starts here

      var add=(x,y)=>{return x+y};

      var res2=add(10,20);

      console.log(res2);

       //output:30

      //minifined arrow functions ends here

//minifined arrow funtions starts here

      var add1=(x,y)=>

        x+y;

        console.log(add1(23,34));

        //output:57

      //minifined arrow functions ends here

    //minified arrow function starts

      var inc=(x)=>{

        return ++x;

      }

      console.log(inc(12));

      //output:13

//

      var inc=x=>

      ++x;

      console.log(inc(12));

      //output:12

    //minified arrow function ends

If suppose single parameter passing then no need to use bractes.

 //this is normal increament function starts here

   var inc2 = function(x){

        return x++;

    }

    console.log(inc2(12));

    //this is normal increament function starts here

Member functions:

 //member functions starts here

    var f1a=function(){

        this.x=100;

        var y=200;

        this.f2=function(){

            console.log("i am f11");

        }

        var f3=function(){

            console.log("i m f3");

        }

    }

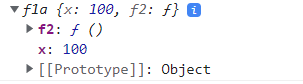
   var obj1=new f1a();

   console.log(obj1);

    //output:f1a {x: 100, f2: ƒ}

    //member functions ends here

Output:



 //member functions starts here

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11");

        }//member function or method

        var f2=function(){

            console.log("i am f2");

        }

    }

    var ob1=new fun1();

    var ob2=new fun1();

    console.log(ob1.x);

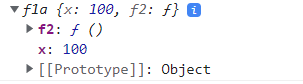
    console.log(ob2.x);

    //output:100

    //output:100

    //member functions ends here

Notes:Ob1 and ob2 same instance created like



Y is cant acceses in out side

Suppose we want acces inside only then this type is called private functions

//member funtion

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=function(){

            console.log("i am f2");

        }

    }

    var ob1=new fun1();

    ob1.f11();

    //output:i m f11 100 200

 //access in normal functions

    //note:i m accessing y but x i am not accessing y because normal functions wont access properties

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=function(){

            console.log("i am f2",this.x,y);

        }//itscreate own this

        f2();

    }

    //output:i am f2 undefined 200

//if i want to se arrow fuction i can access properies also.

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=()=>{

            console.log("i am f2",this.x,y);

        }//itscreate own this

        f2();

    }

    //output:i m f11 100 200

//if i want to se arrow fuction i can access properies also.

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=()=>{

            console.log("i am f2",this.x,y);

        }//function

        f2();

        setTimeout(function(){

            console.log("i am callback of settimeout",this.x);

        },2000);

    }

    var ob1=new fun1();

    ob1.f11();

    //output:i m f11 100 200

    //arrow-funtions.html:155 i am callback of settimeout undefined

  //if i want to se arrow fuction i can access properies also and methods.

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=()=>{

            console.log("i am f2",this.x,y);

        }//function

        f2();

        setTimeout(function(){

            console.log("i am callback of settimeout",this.x);

        },2000);

    }

    var ob1=new fun1();

    ob1.f11();

    //output:i m f11 100 200

    //arrow-funtions.html:155 i am callback of settimeout undefined

 //if i want to se arrow fuction i can access properies also.

    var fun1=function(){

        this.x=100;//property

        var y=200;//variable

        this.f11=function(){

            console.log("i m f11",this.x,y);

        }//member function or method

        var f2=()=>{

            console.log("i am f2",this.x,y);

        }//function

        f2();

        setTimeout(()=>{

            console.log("i am callback of settimeout",this.x);

        },2000);

    }

    var ob1=new fun1();

    ob1.f11();

    //output:html:175 i am callback of settimeout 100

Template strings:

//normal string print starts here

    var x=100;

    var y=200;

    console.log(x,"+",y,"=",x+y);//100 '+' 200 '=' 300

    //ends here

// normal string print starts here

    var x=100;

    var y=200;

    var s=x+"+"+y+"="+(x+y);

    console.log(s);

    //output:100+200=300

//ends here

 //using template strings starts here

    var x=100;

    var y=200;

    var k =`${x}+${y}=${x+y}`;

    console.log(k);

    //output:arrow-funtions.html:204 100+200=300

    //using template strings ewnds here

Important Notes: backtick charecters(` `) must and should use

//uisng normal

var x=100;

    var y=200;

    console.log(x,"+",y,"=",(x+y));

    //output:100, "+", 200, "=", 300

[

    //uisng normal

    var x=100;

    var y=201;

    console.log(x+"+"+y+"="+(x+y));

    //output:"100+201=301"

    //using template strings

    var x=100;

    var y=100;

    var z=`${x}+${y}=${x+y}`;

    console.log(z);

    //output:"100+100=200"

  //starts here

    var x=100;

    var y=100;

    console.log(`addition of ${x} and {y} is {x+y}`);

    //output:"addition of 100 and 200 is 300"

    //ends here

//starts here

    var e1='<h1>hello world</h1>';

    document.getElementById("d1").innerHTML=e1;

    //output:hello world

    //ends here

//multi lines strings starts here

    var e1="<h1>hello world</h1>

            <h2>manimumits</h2>";

    document.getElementById("d1").innerHTML=e1;

    output:error

    //multi lines strings ends her

    //multi lines strings starts here

    //note:suppose i use more lines of headings 1 line we cant use

    var e1="<h1>hello world</h1>    <h2>manimumits</h2>";

    document.getElementById("d1").innerHTML=e1;

    output:hello world

    manimumits

    //multi lines strings ends here

 //multi lines strings starts here

    //note:If i use template string backtick charecters if i use multiple lines also i will get output

    var e1=`<h1>hello world</h1>

     <h2>manimumits</h2>`;

    document.getElementById("d1").innerHTML=e1;

    //output:hello world

    //manimumits

    //multi lines strings ends here

**Classes in .ts:**

-----------------

    //class starts here

    class Employee{

        ename;

        eage;

        esal;

        egender;

        elocation

    }

    var e1= new Employee();

    console.log(e1);

    //output:arrow-funtions.html:286 Employee {ename: undefined, eage: undefined, esal: undefined, egender: undefined, elocation: undefined}

    //class ends here

 //class starts here

    class Employee1{

        ename;

        eage;

        esal;

        egender;

        elocation

    }

    var e1= new Employee1();

    e1.ename="praveen";

    e1.eage=35;

    console.log(e1);

    //output:arrow-funtions.html:301 Employee1 {ename: 'praveen', eage: 35, esal: undefined, egender: undefined, elocation: undefined}

    //class ends here

     //class starts here

     class Employee2{

        constructor(name,sal,age,gender,location){

        this.ename=name;

        this.eage=age;

        this.esal=sal;

        this.egender=gender;

        this.elocation=location

        }

    }

    var e1= new Employee2("praveen","1234","35","male","pune");

    console.log(e1);

    //output:arrow-funtions.html:317 Employee2 {ename: 'praveen', eage: '35', esal: '1234', egender: 'male', elocation: 'pune'}

    //class ends here

//class starts here

    //Notes:methods alsowe can access in class

       class Employee3{

        constructor(name,sal,age,gender,location){

        this.ename=name;

        this.eage=age;

        this.esal=sal;

        this.egender=gender;

        this.elocation=location;

        }

        ename;

        eage;

        esal;

        elocation;

        egender;

        getEmployeeGender(){

            return this.egender;

        }

    }

    var e1= new Employee3("praveen","1234","35","male","pune");

    console.log(e1.getEmployeeGender());

    //output:male

    //class ends here

**Import:**

Modules don’t create single ton but default modules create separate instance for every **import.**

**Decorator:**

**4 type of decorator:**

**1.class decorators**

**2.property decorators**

**3.Method decorators**

**4.parameter decorators**

**Function banana(target: Function): void{**

**target.prototyppe.banana = function(): void{**

**Console.log(‘We have banana!’);**

**)**

**}**

**@banana**

**Class Fruitbasket{**

**Constructor(){**

**//Implementation goes here…**

**}**

**}**

**Var basket = new Fruitbasket();**

**Basket.banana();//console will outpur ‘we have banana’**

**Important Notes:**

**1.Web application is nothing but single page application.**

**2.what class is to use tell us decorator**

**Decorator:decorator is used to tell us what type of template what class to apply it will tell**

**This is written in javascript or react.js:**

<div id="d1"></div>

           var Emloyees =  [{

        name: 'praveen',

        designation: 'angular teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'raju',

        designation: 'angular ',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'ashok',

        designation: 'ammulu teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'chiranjivi',

        designation: 'actor teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    }

 ]

//  console.log(Emloyees[0].name)

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

    var e11=document.createElement("h1");

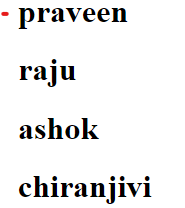
    var tn1=document.createTextNode(Emloyees[i].name);

    e11.appendChild(tn1);

    document.getElementById("d1").appendChild(e11);

}

//output:

****

           var Emloyees =  [{

        name: 'praveen',

        designation: 'angular teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'raju',

        designation: 'angular ',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'ashok',

        designation: 'ammulu teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    },

    {

        name: 'chiranjivi',

        designation: 'actor teacher',

        descript: 'he is good teacher',

        img\_url: 'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg'

    }

 ]

//  console.log(Emloyees[0].name)

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

    var e11=document.createElement("h1");

    var tn1=document.createTextNode(Emloyees[i].name);

    e11.appendChild(tn1);

    var e12 = document.createElement("img");

    e12.setAttribute('src',Emloyees[i].img\_url);

    document.getElementById("d1").appendChild(e11);

    document.getElementById("d1").appendChild(e12);

}

//output:

****

**In Angular:**

**.html**

**.ts**

export class AppComponent {

  Emloyees = [

    {

      name: 'praveen',

      designation: 'angular teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'raju',

      designation: 'angular ',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'ashok',

      designation: 'ammulu teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'chiranjivi',

      designation: 'actor teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

  ];

}

**Output:**

****

**If I want multiple employee data I want**

<div \*ngFor="let i of Emloyees">

  <h1>{{ Emloyees[0].name }}</h1>

  <h3>{{ Emloyees[0].designation }}</h3>

  <p>{{ Emloyees[0].descript }}</p>

  <img [src]="Emloyees[0].img\_url" />

</div>

**.ts**

export class AppComponent {

  Emloyees = [

    {

      name: 'praveen',

      designation: 'angular teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'raju',

      designation: 'angular ',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'ashok',

      designation: 'ammulu teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

    {

      name: 'chiranjivi',

      designation: 'actor teacher',

      descript: 'he is good teacher',

      img\_url:

        'https://s3-ap-southeast-1.amazonaws.com/tv-prod/member/photo/1787281-medium190.jpg',

    },

  ];

}

**Output:**



**Template variables:**

<div id="m1">

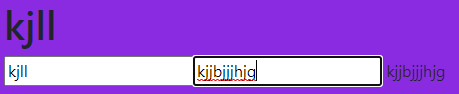
  <h1>{{x}}</h1>

  <input type="text" [(ngModel)]="x">

  <input type="text" [(ngModel)]="y">

  {{y}}

**Output:**

****

<span \*ngIf="10>3 then t1;else t2"></span>

  <ng-template #t1><div>praveen</div></ng-template>

  <ng-template #t2><div>praveen</div></ng-template>

</div>

<p>Template variables are binded to particular block</p>

**//output:** Praveen

**SASS:**

What is Sass? Sass stands for **Syntactically Awesome Stylesheet**. Sass is an extension to CSS. Sass is a CSS pre-processor. Sass is completely compatible with all versions of CSS.

Scss:

The term SCSS is an acronym for **Sassy Cascading Style Sheets**. It is basically a more advanced and evolved variant of the CSS language. Natalie Weizenbaum and Chris Eppstein created it, and Hampton Catlin designed it. It comes with more advanced features- thus often called Sassy CSS.

**The <strong> element is for content that is of greater importance, while the <b> element is used to draw attention to text without indicating that it's more important**.

**Properties:**

<img [src]="img\_url">

**.ts**

  img\_url="https://www.filmibeat.com/ph-big/2019/05/jr-ntr\_1558329150190.jpg";

**Output:**

**Bigger image**

**Staep2:**

<img [src]="img\_url" (click)="xyz()">

**.ts**

 xyz(e:any){

console.log("ABC called",e);

  }

**Output:**

**if click image then**

**I will get in console** ABC called

**Step3:**

<img [src]="img\_url" (click)="xyz(this)">

**.ts**

xyz(e:any){

console.log("ABC called",e);

  }

**Output:**

****

<img [src]="img\_url" (click)="xyz($event)">

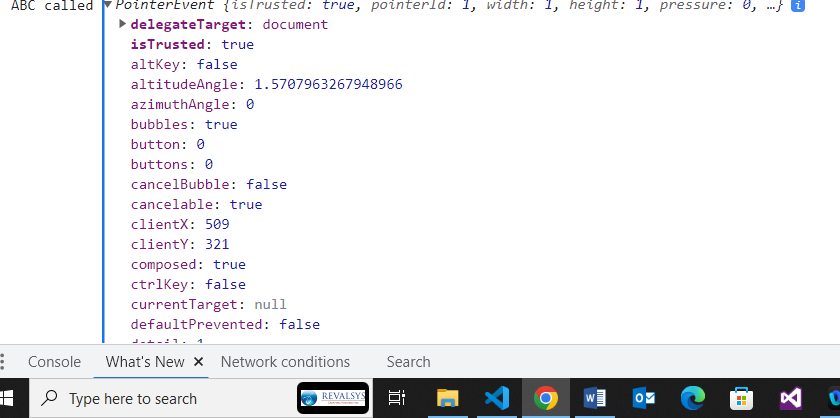
**.ts**

xyz(e:any){

console.log("ABC called",e);

  }

**Output**

****

Here I can get every thing from $event

Actually **this** and **event**s are global

Step3:

<img [src]="img\_url" (click)="xyz($event)">

 xyz(e:Event){

    e.target['style'].width=100px;

  }

Step4:

<div \*ngFor="let students of students" >

  <h1  (click)="exp($event)">{{ students.firstname }}</h1>

  <h2 style="display:none;">{{students.lastname}}</h2>

</div>

exp(a: any) {

    console.log(a);

    a.target.nextSibling.style.display=’block’;

  }

**For accordion**

**Finnaly accordion**

<!---->

<div \*ngFor="let students of students" >

  <h1  (click)="exp($event)">{{ students.firstname }}</h1>

  <h2 style="display:none;">{{students.lastname}}</h2>

</div>

<!---->

**.ts**

 students = [

    {

      firstname: 'ashok',

      lastname: 'mancha',

    },

    {

      firstname: 'manju',

      lastname: 'devarala',

    },

  ];

  exp(a: any) {

    console.log(a);

    var x=a.target.nextSibling.style.display='block';

    if(x=='none'){

      a.target.nextSibling.style.display='block';

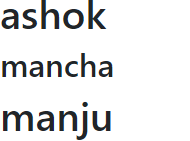
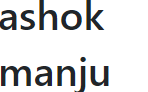
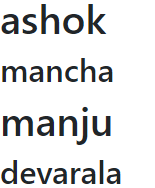
    }

    else{

      a.target.nextSibling.style.display='block'

    }

  }

**** ****

**This is one way data binding:**

<img src="{{img\_url}}"><!--interpolation to bind img-->

<img [src]="img\_url" width="100px" alt=""><!--property binding-->

<img [src]="img\_url" width="100px" alt="" (click)="pqr()"><!--event binding-->

<input type="text" [value]="name">

{{name}}

pqr(){

    console.log("i m pqr")

  }

name="angular";

img\_url="https://www.filmibeat.com/ph-big/2019/05/jr-ntr\_1558329150190.jpg";

**output:**

****

**Two way data binding:**

<!--this is 1 way data binidng strts here-->

<input type="text" [(ngModel)]="name">

{{name}}

<!--this is 1 way data binidng ends here-->

****

{{cost}}

<input type="text" [value]="cost" #c><!--#c is type of variables is called template variables-->

<button (click)="update(c.value)">click</button>

**.ts**

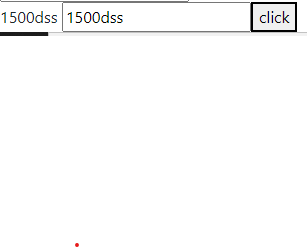
  cost=1500;

  update(t:any){

    this.cost=t;

  }

**When I enter input filed after clicking click button outside value changed**

****

**Keyup**

{{cost}}

<input type="text" [value]="cost" #c (keyup.enter)="update(c.value)"><!--#c is type of variables is called template variables-->

**Keyup:**

<input type="text" #d (keyup)="0">{{d.value}}

****

<input type="checkbox" #q (keyup)="0">{{q.value}}

<h1 #t1>manju</h1>

<h2 #t2>nfkd</h2>

<button (click)="edf(t1,t2)">lets click</button>

 edf(tv1: any,tv2: any){

    console.log(tv1,tv2)

  }

**Example1:**

<input type="radio" name="gender" value="male" [(ngModel)]="g">male<br>

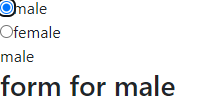
<input type="radio" name="gender" value="female" [(ngModel)]="g">female<br>

{{g}}

<h3 \*ngIf="g=='male'">form for male</h3>

<h3 \*ngIf="g=='female'" >form for female</h3>

 g:any;

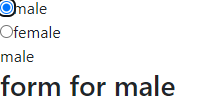
****

**Example2:**

<span \*ngIf="g=='m' then t1;else t2"></span>

<ng-template #t1>Male for form</ng-template>

<ng-template> #t2feMale for form</ng-template>

****

**How to Avoid Cross Browser Compatibility Issues?**

1. Validate HTML and CSS. ...
2. Maintain Layout Compatibility. ...
3. Use CSS Resets. ...
4. Provide Support for Basic Features of the Application. ...
5. Check JavaScript Issues to avoid the Cross-Browser compatibility issues. ...
6. Check DOCTYPE tag.

**120)**

**function outerFunc(outerParam) {**

**function innerFunc(innerParam) {**

**outerParam["b"] = innerParam;**

**}**

**return innerFunc;**

**} const obj = {a:1}**

**const example = outerFunc(obj);**

**const answer = example(2)**

**console.log(obj);**

**Ans)**

**{**

**a:1**

**b:2**

**}**

**121)**

**var y = 1;**

**if (function f() {}) {**

**y += typeof f;**

**}**

**console.log(y);**

**Ans)undefined**

**122)** **for (var i = 0; i < 4; i++) {**

**setTimeout(() => console.log(i), 0)**

**}**

**Ans)4**

**4**

**4**

**4**

**125) for(let i=0; i<10; i++){**

**setTimeout(function(){**

**console.log(i)**

**}, 1000)**

**}**

**Output:0**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

## [The prototype chain](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object_prototypes#the_prototype_chain)

In the browser's console, try creating an object literal:

const myObject = {

city: "Madrid",

greet() {

console.log(`Greetings from ${this.city}`);

},

};

myObject.greet(); // Greetings from Madrid

**123)web application?**

Ans)response to same application and same pages

**124)web service?**

Ans)Response to the other appications also

**125)API**

Ans) api is a combinations which programs or package of programs which provides us some functionalityinteract with some other external resources.

Grid and flexbox. The basic difference between CSS Grid Layout and CSS Flexbox Layout is that **flexbox was designed for layout in one dimension - either a row or a column.** Grid was designed for two-dimensional layout - rows, and columns at the same time.