**­­­­­**[**http://www.tutorialspoint.com/javascript/javascript\_interview\_questions.htm**](http://www.tutorialspoint.com/javascript/javascript_interview_questions.htm)

**ionicframework.com. Ionic is a complete open-source SDK for hybrid mobile app development. Built on top of AngularJS and Apache Cordova, Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass.**

**What is JavaScript?**

JavaScript is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages.

The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

**Name some of the JavaScript features.**

Following are the features of JavaScript −

* JavaScript is a lightweight, interpreted programming language.
* JavaScript is designed for creating network-centric applications.
* JavaScript is complementary to and integrated with Java.
* JavaScript is complementary to and integrated with HTML.
* JavaScript is open and cross-platform.

**What are the advantages of using JavaScript?**

Following are the advantages of using JavaScript −

* **Less server interaction −** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors −** They don't have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity −** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces −** You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

**What are disadvantages of using JavaScript?**

We can not treat JavaScript as a full fledged programming language. It lacks the following important features −

* Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
* JavaScript can not be used for Networking applications because there is no such support available.
* JavaScript doesn't have any multithreading or multiprocess capabilities.

**Is JavaScript a case-sensitive language?**

Yes! JavaScript is a case-sensitive language. This means that language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

How can you create an Object in JavaScript?

JavaScript supports Object concept very well. You can create an object using the object literal as follows −

var emp = {

name: "Zara",

age: 10

};

**How can you read properties of an Object in JavaScript?**

You can write and read properties of an object using the dot notation as follows −

// Getting object properties

emp.name // ==> Zara

emp.age // ==> 10

// Setting object properties

emp.name = "Daisy" // <== Daisy

emp.age = 20 // <== 20

How can you create an Array in JavaScript?

You can define arrays using the array literal as follows −

var x = [];

var y = [1, 2, 3, 4, 5];

**How to read elements of an array in JavaScript?**

An array has a length property that is useful for iteration. We can read elements of an array as follows −

var x = [1, 2, 3, 4, 5];

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

// Do something with x[i]

}

**What is a named function in JavaScript? How to define a named function?**

A named function has a name when it is defined. A named function can be defined using function keyword as follows −

function named(){

// do some stuff here

}

**How many types of functions JavaScript supports?**

A function in JavaScript can be either named or anonymous.

**How to define a anonymous function?**

An anonymous function can be defined in similar way as a normal function but it would not have any name.

**Can you assign a anonymous function to a variable?**

Yes! An anonymous function can be assigned to a variable.

**Can you pass a anonymous function as an argument to another function?**

Yes! An anonymous function can be passed as an argument to another function.

**What is arguments object in JavaScript?**

JavaScript variable arguments represents the arguments passed to a function.

**How can you get the type of arguments passed to a function?**

Using typeof operator, we can get the type of arguments passed to a function. For example −

function func(x){

console.log(typeof x, arguments.length);

}

func(); //==> "undefined", 0

func(1); //==> "number", 1

func("1", "2", "3"); //==> "string", 3

**How can you get the total number of arguments passed to a function?**

Using arguments.length property, we can get the total number of arguments passed to a function. For example −

function func(x){

console.log(typeof x, arguments.length);

}

func(); //==> "undefined", 0

func(1); //==> "number", 1

func("1", "2", "3"); //==> "string", 3

**How can you get the reference of a caller function inside a function?**

The arguments object has a callee property, which refers to the function you're inside of. For example −

function func() {

return arguments.callee;

}

func(); // ==> func

**What is the purpose of 'this' operator in JavaScript?**

JavaScript famous keyword this always refers to the current context.

**What are the valid scopes of a variable in JavaScript?**

The scope of a variable is the region of your program in which it is defined. JavaScript variable will have only two scopes.

* **Global Variables −** A global variable has global scope which means it is visible everywhere in your JavaScript code.
* **Local Variables −** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Which type of variable among global and local, takes precedence over other if names are same?

A local variable takes precedence over a global variable with the same name.

**What is callback?**

A callback is a plain JavaScript function passed to some method as an argument or option. Some callbacks are just events, called to give the user a chance to react when a certain state is triggered.

**What is closure?**

Closures are created whenever a variable that is defined outside the current scope is accessed from within some inner scope.

**Give an example of closure?**

Following example shows how the variable counter is visible within the create, increment, and print functions, but not outside of them −

function create() {

var counter = 0;

return {

increment: function() {

counter++;

},

print: function() {

console.log(counter);

}

}

}

var c = create();

c.increment();

c.print(); // ==> 1

With a function closure you can store data in a separate scope, and share it only where necessary.

Closures are necessary in javascript due to the fact that most API's that require callback functions (for instance, an "onclick" function) do not provide other mechanisms to send parameters to those callback functions (or to explicitly set the "this" pointer). Instead, you need to use closures to allow the callback to access variables in the "parent" function.

**Which built-in method returns the character at the specified index?**

charAt() method returns the character at the specified index.

Which built-in method combines the text of two strings and returns a new string?

concat() method returns the character at the specified index.

Which built-in method calls a function for each element in the array?

forEach() method calls a function for each element in the array.

Which built-in method returns the index within the calling String object of the first occurrence of the specified value?

indexOf() method returns the index within the calling String object of the first occurrence of the specified value, or −1 if not found.

**Which built-in method returns the length of the string?**

length() method returns the length of the string.

**Which built-in method removes the last element from an array and returns that element?**

pop() method removes the last element from an array and returns that element.

**Which built-in method adds one or more elements to the end of an array and returns the new length of the array?**

push() method adds one or more elements to the end of an array and returns the new length of the array.

**Which built-in method reverses the order of the elements of an array?**

reverse() method reverses the order of the elements of an array −− the first becomes the last, and the last becomes the first.

**Which built-in method sorts the elements of an array?**

sort() method sorts the elements of an array.

**Which built-in method returns the characters in a string beginning at the specified location?**

substr() method returns the characters in a string beginning at the specified location through the specified number of characters.

**Which built-in method returns the calling string value converted to lower case?**

toLowerCase() method returns the calling string value converted to lower case.

**Which built-in method returns the calling string value converted to upper case?**

toUpperCase() method returns the calling string value converted to upper case.

**Which built-in method returns the string representation of the number's value?**

toString() method returns the string representation of the number's value.

**What are the variable naming conventions in JavaScript?**

While naming your variables in JavaScript keep following rules in mind.

You should not use any of the JavaScript reserved keyword as variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.

JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character. For example, 123test is an invalid variable name but \_123test is a valid one.

JavaScript variable names are case sensitive. For example, Name and name are two different variables.

**How typeof operator works?**

The typeof is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The typeof operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

What typeof returns for a null value?

It returns "object".

**Can you access Cookie using javascript?**

JavaScript can also manipulate cookies using the cookie property of the Document object. JavaScript can read, create, modify, and delete the cookie or cookies that apply to the current web page.

**How to create a Cookie using JavaScript?**

The simplest way to create a cookie is to assign a string value to the document.cookie object, which looks like this −

Syntax −

document.cookie = "key1 = value1; key2 = value2; expires = date";

Here expires attribute is option. If you provide this attribute with a valid date or time then cookie will expire at the given date or time and after that cookies' value will not be accessible.

**How to read a Cookie using JavaScript?**

Reading a cookie is just as simple as writing one, because the value of the document.cookie object is the cookie. So you can use this string whenever you want to access the cookie.

The document.cookie string will keep a list of name = value pairs separated by semicolons, where name is the name of a cookie and value is its string value.

You can use strings' split() function to break the string into key and values.

**How to delete a Cookie using JavaScript?**

Sometimes you will want to delete a cookie so that subsequent attempts to read the cookie return nothing. To do this, you just need to set the expiration date to a time in the past.

**How to redirect a url using JavaScript?**

his is very simple to do a page redirect using JavaScript at client side. To redirect your site visitors to a new page, you just need to add a line in your head section as follows −

<head>

<script type="text/javascript">

<!--

window.location="http://www.newlocation.com";

//-->

</script>

</head>

**How to print a web page using javascript?**

JavaScript helps you to implement this functionality using print function of window object. The JavaScript print function window.print() will print the current web page when executed.

**What is Date object in JavaScript?**

The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ).

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

**What is Number object in JavaScript?**

the Number object represents numerical date, either integers or floating-point numbers. In general, you do not need to worry about Number objects because the browser automatically converts number literals to instances of the number class.

Syntax −

Creating a number object −

var val = new Number(number);

If the argument cannot be converted into a number, it returns NaN (Not-a-Number).

**How to handle exceptions in JavaScript?**

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the try...catch...finally construct as well as the throw operator to handle exceptions.

You can catch programmer-generated and runtime exceptions, but you cannot catch JavaScript syntax errors.

**What is purpose of onError event handler in JavaScript?**

The onerror event handler was the first feature to facilitate error handling for JavaScript. The error event is fired on the window object whenever an exception occurs on the page.

The onerror event handler provides three pieces of information to identify the exact nature of the error −

* **Error message −** The same message that the browser would display for the given error.
* **URL −** The file in which the error occurred.
* **Line number −** The line number in the given URL that caused the error.

**Question: What is the importance of the HTML DOCTYPE?**  
DOCTYPE is an instruction to the web browser about what version of the markup language the page is written. Its written before the HTML Tag. Doctype declaration refers to a Document Type Definition (DTD).  
  
  
**Question: Explain the difference between visibility:hidden; and display:none?**  
Visibility:Hidden; - It is not visible but takes up it's original space.  
Display:None; - It is hidden and takes no space.

**Question: How do you clear a floated element?**  
clear:both  
  
  
**Question: What is the difference between == and === ?**  
== is equal to   
=== is exactly equal to (value and type)  
  
  
  
**Question: What is a java script object?**  
A collection of data containing both properties and methods. Each element in a document is an object. Using the DOM you can get at each of these elements/objects.  
  
  
  
**Question: Describe what "this" is in JavaScript?**  
this refers to the object which 'owns' the method.  
  
  
  
**Question: What is a closure?**  
Closures are expressions, usually functions, which can work with variables set within a certain context.

**Question: How to use a function a Class?**

function functionName(name) {

this.name = name;

}

// Creating an object

var functionName = new functionName("WTEN");

console.log(functionName.name); //WTEN

**Question: What is Difference between null and undefined?**  
**null** is an object with no value. **undefined** is a type. 

typeof null; // "object"

typeof undefined; // "undefined"

**Question: What is the difference between HTML and XHTML?**  
HTML is HyperText Markup Language used to develop the website.  
XHTML is **modern version of HTML 4**. XHTML is an HTML that follows the XML rules which should be well-formed.

**Question: How to set a default parameter value for a JavaScript function?**

/\*\* Here email is parameter in which we have set the default value i.e email@domain.com \*\*/

function function1(name, email)

{

email = typeof email !== 'undefined' ? email : 'defaultemail@domain.com';

console.log('name='+name+', Email= '+email);

}

function1('john','myname@gmail.com');

function1('john');

**Queston: How to convert a string to lowercase?**

var str='This is testing String';

str = str.toLowerCase();

console.log(str);

**Question: How to modify the URL of page without reloading the page?**  
use pushState javascript function.   
**For Example:**

window.history.pushState('page2', 'This is page Title', '/newpage.php');

**Question: How to convert JSON Object to String?**

var myobject=['Web','Technology','Experts','Notes']

JSON.stringify(myobject);

**Question: How to convert JSON String to Object?**

var jsonData = '{"name":"web technology","year":2015}';

var myobject = JSON.parse(jsonData);

console.log(myobject);

**Question: How to check an variable is Object OR String OR Array?**  
Use below function to get Data type of javascript variable.

function checkDataType(someVar){

result ='String';

if(someVar instanceof Object){

result ='Object'

}

if($.isArray(someVar)){

result = 'Array';

}

return result;

}

var someVar= new Array("Saab", "Volvo", "BMW");

console.log(result);

**Question: Can i declare a variable as CONSTANT like in PHP?**  
**No,** I think cosntant not exist in javascript.   
But you can follow same type convention to declare constant.

var CONSTANT\_NAME = "constant value";

**Question: How to open URL in new tab in javascript?**  
use javascript, window.open function.

window.open('http://www.web-technology-experts-notes.in/','\_blank');

**Question: What is difference between undefined and object?**  
**undefined**means some variable's value is not defined yet.  
**object**means variables's value is defined that is either function, object OR array.  
  
**With use of below, you can easily determine whether it is object OR NULL.**

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

console.log(typeof(undefined)); // undefined

**Question: How to get current date in JavaScript?**

var today = new Date();

console.log(today);

**Question: How do I declare a namespace in JavaScript?**

var myNamespace = {

function1: function() { },

function2: function() { }

function3: function() { }

};

myNamespace.function3();

**Question: What is the best way to detect a mobile device in jQuery?** 

if( /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i.test(navigator.userAgent) ) {

}

**Question: How to detect mobiles including ipad using navigator.useragent in javascript?**

if(navigator.userAgent.match(/Android/i) || navigator.userAgent.match(/webOS/i) || navigator.userAgent.match(/BlackBerry/i) || navigator.userAgent.match(/iPhone/i)){

console.log('Calling from Mobile');

}else{

console.log('Calling from Web');

}

**Question: How to detect mobiles including ipad using navigator.useragent in javascript?**

if(navigator.userAgent.match(/Android/i) || navigator.userAgent.match(/webOS/i) || navigator.userAgent.match(/BlackBerry/i) || navigator.userAgent.match(/iPhone/i)){

console.log('Calling from Mobile');

}else{

console.log('Calling from Web');

}

**Question: What is JavaScript closures? Give an Example?**  
A closure is an **inner function** that has access to the **outer function's variables** known as Closure.

function function1(x) {

var tmp = 3;

function function2(y) {

console.log(x + y + (++tmp)); // will console 7

}

function2(1);

}

function1(2);

**Question: What is the function of the var keyword in Javascript?**  
**var**is used to create the local variables.  
If you're in a function then **var** will create a local variable.

var x =10;

function function1(){

var x=20;

}

function1();

alert(x); //10

var x = 10 declares variable x in current scope.  
**If the declaration appears in a function It is a local variable.**  
if it's in global scope - a global variable is declared.

x =10;

function function1(){

x=20;

}

function1();

alert(x); //20

x=10 declare a global variable.  
  
  
**Question: How can I make a redirect page using jQuery?**

window.location.href = "http://www.web-technology-experts-notes.in/p/sitemap.html";

**Question: How can I check if one string contains another substring?**  
you can use indexOf function,   
If string found - It will returns the **position of the string** in the full string.  
If string not found- **it will return -1**  
**See Example**

var haystack = "full-string-here";

var needle = 'string';

if(haystack.indexOf(needle)>=0){

console.log('String found');

}else{

console.log('String Not found');

}

**Question: What is difference between == and === in javascript?**  
Both are used to check the equality only difference === check with both data type.  
**For Example**

2=='2' // will return true;

2==='2'// will return false;

**Question: Can I comment a JSON file?**  
No, but you can add a node on root where you can display the information.  
  
  
**Question: How to Check checkbox checked property?**

var checkboxStatus = $('#checkMeOut').prop('checked');

if(checkboxStatus){

console.log('Checkbox is Checked');

}else{

console.log('Checkbox is Not Checked');

}

**Question: How to include a JavaScript file in another JavaScript file?**  
With use of jQuery, its quite simple, **See below:**

$.getScript("my\_lovely\_script.js", function(){

});

**Question: How to remove single property from a JavaScript object?**

var myobject=['Web','Technology','Experts','Notes']

delete myobject['Technology'];

**Question: How to add single property from a JavaScript array?**

var myobject=['Web','Technology','Experts','Notes']

myobject.push(' Web Development');

**Question: How to empty an array?**

var myobject=['Web','Technology','Experts','Notes']

myobject.length = 0

**Question: How to trim string in JavaScript?**  
You can do in very simple way using jQuery.   
**See Below:**

$.trim(' Web Technology ');

**Question: How do you get a timestamp in JavaScript?**

new Date().getTime()

**Question: How to use javaScript Loop?**

var myobject=['Web','Technology','Experts','Notes']

for (index = 0; index < myobject.length; ++index) {

console.log(myobject[index]);

}

**Question: How to detect an undefined object property in JavaScript?**

if (typeof myobject === "undefined") {

console.log("myobject is undefined");

}

**Question: How to validate email address in JavaScript?**

function validateEmail(email) {

var re = /^([\w-]+(?:\.[\w-]+)\*)@((?:[\w-]+\.)\*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i;

return re.test(email);

}

validateEmail('contactuse@web-technology-experts-notes.in'); // Will return true;

**Question: How to capitalize the first letter of string?**

function capitalizeFirstLetterOfString(string) {

return string.charAt(0).toUpperCase() + string.slice(1);

}

capitalizeFirstLetterOfString('web-technology-experts-notes'); //Web-technology-experts-notes

**Question: How to get current url in web browser?**

window.location.href

**Question: How can I refresh a page with jQuery?** 

window.location.reload();

**Question: How to pass multiple parameter in setTimeout function?**

function myFunctionName(var1,var2){

console.log('called after 2 sec of page load '+var1+' '+var2);

}

setTimeout(myFunctionName,2000,'value1','value2');

**Question: How to enumerate the properties of js objects?**

var myObject = {name1: 'Value1',name2: 'Value2'};

//console.log(myObject); //It will print all the values

for (var name in myObject) {

//console.log(name+'=>'+myObject[name]);

}

**Question: How to measure the execution time of JavaScript?**

var startTime = new Date().getTime();

/\*

Write here you script

\*/

for(i=1;i<=500000; i++){

}

var endTime = new Date().getTime();

var time = endTime - startTime;

console.log('Execution time in Milli Seconds: ' + time);

console.log('Execution time in Seconds: ' + time/1000);

**Question: How to listen (Do some changes) the window.location.hash change?**

$(window).on('hashchange', function() {

callNewFunction(window.location.href)

});

function callNewFunction(url){

console.log('Hash URL is called');

}

After appling above code, whenever you add/update the hash value, **callNewFunction**will called automatically.  
  
**hashchange event** is **HTML5**feature and supported by all modern browsers and support is added in following browser.

1. Internet Explorer 8
2. Firefox 3.6
3. Chrome 5
4. Safari 5
5. Opera 10.6

**Question: How to add class in an element?**  
HTML Part

<div id="myDivId">

</div>

javaScript Part

var d = document.getElementById("myDivId");

d.className += " newClass";

**Question: How to get the list of classes for an element?**

d = document.getElementById("myDivId");

console.log(d.className);

**Question: Can we compare two javaScript objects?**  
Yes, We can compare two javascript objects. See Following examples.

var myObject1 = {name1: 'Value1',name2: 'Value2'};

var myObject2 = {name1: 'Value111',name2: 'Value222'};

if(JSON.stringify(myObject1) === JSON.stringify(myObject2)){

console.log("Both object are same");

}else{

console.log("Both object are different");

}

**Question: What is difference between Array(3) and Array('3') in javascript?**  
new Array(3), means declare the 3 elements and each have value "undefined". new Array('3'), means declare the 1 element and have value 3. 

console.log(new Array(3)); // [undefined, undefined, undefined]

console.log(new Array('3')); // ["3"]

**Question: How to get browser URL for all browser?**

console.log(window.location.href);

**Question: How to remove an element(string OR object ) from javascript Array/Object?**

var myObject = {name1: 'Value1',name2: 'Value2'};

delete myObject['name1']

console.log(myObject);

**Question: What does jQuery.fn mean?**  
jQuery.fn is just an prototype for defining the new functions.  
fn is an alias to the prototype property.  
For Example, 

$.fn.newFunctionName = function(val){

console.log('something '+val); //something Test Value

};

$.fn.newFunctionName('Test Value');

**Question: How to remove an empty elements from an Array?**

var simpleArray = [1,3,,3,null,,0,,undefined,4,,6,,];

var cleanArray = simpleArray.filter(function(n){ return n != undefined });

console.log(cleanArray);

**Question: How to add 5 days in JavaScript?**

var result = new Date();

result.setDate(result.getDate() + 5);

console.log(result);

**1. Difference between window.onload and onDocumentReady?**

The onload event does not fire until every last piece of the page is loaded, this includes css and images, which means there’s a huge delay before any code is executed.  
That isnt what we want. We just want to wait until the DOM is loaded and is able to be manipulated. onDocumentReady allows the programmer to do that.

**2. What is the difference between == and === ?**

The == checks for value equality, but === checks for both type and value.

**3. What does “1”+2+4 evaluate to? What about 5 + 4 + “3”?**

Since 1 is a string, everything is a string, so the result is 124. In the second case, its 93.

**4. What is the difference between undefined value and null value?**

undefined means a variable has been declared but has not yet been assigned a value. On the other hand, null is an assignment value. It can be assigned to a variable as a representation of no value.  
Also, undefined and null are two distinct types: undefined is a type itself (undefined) while null is an object.  
Unassigned variables are initialized by JavaScript with a default value of undefined. JavaScript never sets a value to null. That must be done programmatically.

**5. How do you change the style/class on any element?**

document.getElementById(“myText”).style.fontSize = “20”;  
-or-  
document.getElementById(“myText”).className = “anyclass”;

**6. What are JavaScript closures? When would you use them?**

Two one sentence summaries:

\* A closure is the local variables for a function – kept alive after the function has returned, or  
\* A closure is a stack-frame which is not deallocated when the function returns.

A closure takes place when a function creates an environment that binds local variables to it in such a way that they are kept alive after the function has returned. A closure is a special kind of object that combines two things: a function, and any local variables that were in-scope at the time that the closure was created.

The following code returns a reference to a function:

function sayHello2(name) {  
var text = ‘Hello ‘ + name; // local variable  
var sayAlert = function() { alert(text); }  
return sayAlert;  
}

Closures reduce the need to pass state around the application. The inner function has access to the variables in the outer function so there is no need to store the information somewhere that the inner function can get it.

This is important when the inner function will be called after the outer function has exited. The most common example of this is when the inner function is being used to handle an event. In this case you get no control over the arguments that are passed to the function so using a closure to keep track of state can be very convenient.

**7. What is unobtrusive JavaScript? How to add behavior to an element using JavaScript?**

Unobtrusive JavaScript refers to the argument that the purpose of markup is to describe a document’s structure, not its programmatic behavior and that combining the two negatively impacts a site’s maintainability. Inline event handlers are harder to use and maintain, when one needs to set several events on a single element or when one is using event delegation.

|  |  |
| --- | --- |
| 1 | <input type="text" name="date" /> |

Say an input field with the name “date” had to be validated at runtime:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | document.getElementsByName("date")[0].                     addEventListener("change", validateDate, false);    function validateDate(){  // Do something when the content of the 'input' element with the name 'date' is changed.  } |

Although there are some browser inconsistencies with the above code, so programmers usually go with a JavaScript library such as JQuery or YUI to attach behavior to an element like above.

**8.  What is JavaScript namespacing? How and where is it used?**

Using global variables in JavaScript is evil and a bad practice. That being said, namespacing is used to bundle up all your functionality using a unique name. In JavaScript, a namespace is really just an object that you’ve attached all further methods, properties and objects. It promotes modularity and code reuse in the application.

**9.  What datatypes are supported in JavaScript?**  
Number, String, Undefined, null, Boolean

10. What is the difference between innerHTML and append() in JavaScript?

InnerHTML is not standard, and its a String. The DOM is not, and although innerHTML is faster and less verbose, its better to use the DOM methods like append Child(), firstChild.nodeValue, etc to alter innerHTML content.

**Question: Who developed the Bootstrap?**  
Mark Otto and Jacob Thornton at Twitter  
  
  
**Question: What are the key components of Bootstrap?**

* Plenty of CSS files
* Scaffolding
* List of layout components
* JavaScript Plugins
* Customize your components

**Question: Why Use Bootstrap?**

* Easy to use
* Responsive features
* Mobile-first approach
* Browser compatibility
* Fluide & Fixed Layout available

**Question: From where we can download the Bootstrap?**  
<http://getbootstrap.com/>  
  
  
  
**Question: Can we include bootstrap CDN instead of download the Bootstrap?**  
Yes sure, we can do from following

<!-- Latest compiled and minified CSS -->

<link href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css" rel="stylesheet"></link>

**Question: What is class loaders in Bootstrap?**  
Class loader is a part of Java Runtime Environment which loads Java classes into Java virtual environment.  
  
  
  
**Question: What are different types of layout available in Bootstrap?**

* Fluid Layout
* Fixed Layout

**Question: What is Fluid Layout in Bootstrap?**  
Fluid layout adapts itself to different browser. Means design automatic adjust according to browser size.  
  
  
**Question: What is Fixed Layout in Bootstrap?**  
Fixed layout doesn't adapts itself to different browser but it can be responsive.   
  
  
  
**Question: What is responsive layout?**  
Responsive layout which is able to adapt itself to different sizes as well, but when resizing, the number of columns changes according to the available space.  
  
  
  
**Question: What is difference between Fluid Layout and responsive Layout?**  
Fluid layout adapts itself to different browser window sizes, all the values used are calculated proportionally to the viewport size, so when resizing, all the columns are resized.  
  
Responsive layout is able to adapt itself to different sizes as well. When resizing, the number of columns changes according to the available space.  
  
  
  
**Question: What function you can use to wrap a page content?**

.container

**Question: How to classified pagination in bootstrap?**  
Add class "pagination" on your page for pagination.  
.disabled, .active are available  
.pagination-Ig, .pagination-sm to get different sizes  
  
  
  
**Question: What is Jumbotron?**  
Jumbotron is used for content that you want to highlight like some slogan OR marketing headline.  
  
  
**Question: What to display code in bootstrap?**  
you can use following tags

<code></code>

**Question: What is Modal plugin used for in Bootstrap?**  
Modal Plugin is a child window that is layered over its parent window  
  
  
**Question: What is Bootstrap Container in Bootstrap?**  
Bootstrap container is a class which is useful and creating a centered area in the page for display.  
  
  
**Question: What is Bootstrap collapsing elements?**  
Bootstrap collapsing elements enables you to collapse any element without using external JavaScript.  
  
  
**Question: How to add badge to list group in Bootstrap?**

<span class="badge"></span> in LI Tag

**Question: What is Media Object?**  
Media objects in Bootstrap enables to put media object like image, video or audio.

**Question: What are different types of Media Object?**

.media

.media-list

**Question: What is Bootstrap well?**  
Bootstrap well is a container which makes the content to appear sunken.  
  
  
  
**Question: What are current Stable version of Bootstrp**  
Version: **3.3.6**, Dated: **November 24, 2015**   
  
**Question: In which language Bootstrap is written?**  
HTML, CSS, LESS, Sass and JavaScript

**Question: How can you declare a class in JavaScript?**  
In JavaScript, classes declare is quite different from C++, PHP and java.  
**Following are three different ways to declare the JavaScript.**  
**1. Class using function as a constructor** 

function Person(name) {

this.name = name;

}

/\* Create an Object an object \*/

var person = new Person("Web Technology experts Notes");

person.name; // Web Technology experts Notes

**2. Class Literal notation**

var person = {

name: "",

setName: function(name) {

this.name = name;

}

}

person.setName("Web Technology experts Notes");

person.name; // Web Technology experts Notes

**3. Singleton through a function** 

var person = new function() {

this.setName = function(name) {

this.name = name;

}

this.printData = function() {

return this.name;

}

}

person.setName("Rafael");

console.log(person.printData()); // Web Technology experts Notes

**Question: What is Difference between null and undefined?**  
NULL is **object**  
undefined is a **datatype**

console.log(null);

console.log(undefined);

**Question: How can you add a method to a already defined class?**   
using **prototype** you can achieve the same. 

function Person(name) {

this.name = name;

}

Person.prototype.newfunction = function() {

console.debug("new function is called.");

}

//create an object

var person = new Person("Web Technology");

// Calling the new method

person.newfunction(); // new function is called.

**Question: What is Sass?**   
Sass is the modern way of doing CSS.   
Sass works with variables, nested syntax and mathematical operations   
  
  
**Question: How to improve the performance of page from Frontend?**

* Use sprite images
* Javascripts should be at the bottom of the page
* Ensure parallel requests
* Compress images
* Browser Caching

**Question: What is XHTML?**   
XHTML is an HTML that **follows the XML rules**.   
  
  
**Question: Why Recommended external CSS or Javascript versus inline?**  
1. **Hard to maintain** the Inline JS/CSS.  
2. Inline js/css **load slower**.  
3. **No caching benefits** in Inline JS/CSS.  
  
  
**Question: Why do we need to use W3C standard code?**   
W3C standards are to ensure **cross-platform compatibility** and more compact file sizes.  
  
  
**Question: What is lazy loading?**  
Lazy loading is a design pattern commonly used in computer field.  
Lazy loading is loading code only once user needs it.  
  
  
**Question: What is the difference between HTML elements and tags?**  
HTML elements communicate to the browser how to render text.  
Aftr that they are surrounded by angular brackets <> they form HTML tags.  
  
  
**Question: What does DOCTYPE mean?**  
DOCTYPE tells the browser which type of HTML is used in current webpage.  
  
  
**Question: Tell the name of few new tags in HTML5?**   
datalist, datetime, output, keygen, date, month, week, time, number, range, email, and url.   
  
  
**Question: What is the purpose of the z-index and how is it used?**  
The z-index helps specify the positioned elements that may overlap one another.  
**z-index can take following values**.  
**Auto**: Sets the order equal to its parents.  
**Number**: Set Orders the Number specified.  
**Initial**: Sets this property to its default value which is zero.  
**Inherit**: Inherits from parent element.  
  
  
**Question: List the main CSS style sheet properties?**

* Background
* Text
* Font
* List
* Outline
* Margin
* Border
* Padding
* Table

**Question: What are some of the new features and properties in CSS3?**

Box model 

* New Web fonts
* Rounded corners
* Box Shadows,
* Transform property
* Border Images
* Text Shadows
* New Color schemes
* Multi-column layout
* New Pseudo-classes
* New Gradients

JQuery is a fast, small, and rich-featured JavaScript library.  
  
It makes things like HTML **document traversal** and **manipulation**, **event handling**, **animation**, **Ajax**and much simpler with an easy-to-use jQuery. JQuery works across a multiple of browsers.   
  
With a combination of versatility and extensible, jQuery has changed the way that millions of people write JavaScript. It help for Fast development.

***From:***[***https://jquery.com/***](https://jquery.com/)

**Question: What is current stable version of jQuery?**  
Version: 1.11.3 (April 28, 2015)  
  
  
**Question: In Which language jQuery is written?**  
JavaScript  
  
  
**Question: What is offical website of jQuery?**   
<https://jquery.com/>  
  
  
**Question: Which file need to include to use jQuery functions?**  
We must include a jQuery file.  
We can also include jQuery from their official website.  
<script src="//code.jquery.com/jquery-1.11.2.min.js"></script>  
  
  
  
**Question: What are other jQuery Foundation Projects?**

* jQuery User Interface
* jQuery Mobile
* QUnit
* Sizzle

**Question: What is JQuery UI?**  
JQuery UI is a**jQuery library** where UI comes with cool **widgets**, **effects**and **interaction**mechanism. Whether you're building highly interactive web applications OR just need to add a date picker to a form control, jQuery UI is the perfect choice.  
  
  
**Question: From where I can get List of jQuery UI Demos?**  
<http://jqueryui.com/demos/>  
  
  
  
**Question: What are the different type of selectors in Jquery?**  
Following are 3 types of selectors in Jquery

1. CSS Selector
2. XPath Selector
3. Custom Selector

**Question: What is the difference between jQuery-x.x.x.js and jQuery.x.x.x-min.js?**  
In terms of functionality, there is no difference between the jQuery-x.x.x.js and jQuery-x.x.x-min.js. **jQuery-x.x.x-min.js also called minified version because in this file there is no space, no tab, no newline, small variable name and very less file size**. Minified version load more faster as compare to normal jquery, that's why minified version used in production environment. However this can play a vital role in th  
  
  
**Question: What is the use of Delegate() Method in jQuery?**  
**1**. Attach an parent event to each one of its child elements.  
**2**. Attach the event to the elements which is not available at the time of page loading (element will after page load).  
  
  
  
**Question: What does .size() method of jquery?**  
Return the number of elements in node. 

$("div.webtechnologyexpert").size();

**Question: What is the use of jQuery Connect?**  
It is used to connect or bind a function to another function.It is use to execute a function whenever a function from another object is executed. To use this you need to download the jquer.connect.js file. 

$.connect('fun1',fun2)

$.connect(null,'fun1',fun2)

$.connect(self,'fun1',fun2)

$.connect('fun1',null,fun2)

**Question: What is the use of jQuery disconnect?**  
It is used to disconnect a function to another function.It is the opposite of $.connect  
  
  
  
**Question: What is the use of jQuery disconnectAll?**  
It is used to disconnect all the connected functions.  
  
  
**Question: What is the purpose of jquery-x.x.x-vsdoc.js?**  
Generally we will use jQuery-x.x.x-vsdoc.js to provide the intellisense support. We can even delete this file. But the thing is that it won't provide the intellisense support if we delete that file.  
  
 **Question: How to hide and show a div?**

$(document).ready(function(){

$('div#mydiv').hide();

$('div#mydiv').show();

});

**Question: How to add data in empty div?**

$(document).ready(function(){

$('div#mydiv').html();

});

**Question: How to add click event on div in jQuery**

$(document).ready(function(){

$("div#mydiv").click(function(){

alert('mydiv is clicked')

});

});

**Question: How to add double-click event on div in jQuery**

$(document).ready(function(){

$("div#mydiv").dblclick(function(){

alert('mydiv is double clicked')

});

});

**Question: How to add hover event on div in jQuery?**

$(document).ready(function(){

$("div#mydiv").hover(function(){

alert('Hover on mydiv')

});

});

**Question: Is jQuery dependened on operating system?**  
No, jQuery is independent of any operating system. It works similar in all operating system.  
  
  
  
**Question: Explain the features of jQuery?**

1. Effects and animations on html
2. Ajax to send the server call
3. Extensibility
4. Add/Change on DOM
5. Add, Update, delete Events
6. CSS manipulation with use of jquery
7. We can add JavaScript Plugins
8. DOM traversal and modification with use of jquery

**Question: Can constructors be parameterized?**  
Yes, It can be.

Invoking a Function with a Function Method

In JavaScript, functions are objects. JavaScript functions have properties and methods.

**call()** and **apply()** are predefined JavaScript function methods. Both methods can be used to invoke a function, and both methods must have the owner object as first parameter.

function myFunction(a, b) {  
    return a \* b;  
}  
myObject = myFunction.call(myObject, 10, 2);     // Will return 20

function myFunction(a, b) {  
    return a \* b;  
}  
myArray = [10, 2];  
myObject = myFunction.apply(myObject, myArray);  // Will also return 20

Both methods take an owner object as the first argument. The only difference is that call() takes the function arguments separately, and apply() takes the function arguments in an array.

In JavaScript strict mode, the first argument becomes the value of **this** in the invoked function, even if the argument is not an object.

In "non-strict" mode, if the value of the first argument is null or undefined, it is replaced with the global object.

With call() or apply() you can set the value of **this**, and invoke a function as a new method of an existing object.

Bind: It other words, *bind ()* allows us to easily set which specific object will be bound to *this* when a function or method is invoked.

This might seem relatively trivial, but often the *this* value in methods and functions must be set explicitly when you need a specific object bound to the function’s *this* value.

The need for *bind* usually occurs when we use the *this* keyword in a method and we call that method from a receiver object; in such cases, sometimes *this* is not bound to the object that we expect it to be bound to, resulting in errors in our applications. Don’t worry if you don’t fully comprehend the preceding sentence. It will become clear like teardrop in a moment.

The map() method creates a new array with the results of calling a function for every array element.

The map() method calls the provided function once for each element in an array, in order.

**Note:** map() does not execute the function for array elements without values.

**Note:** map() does not change the original array.

In the JavaScript, the variables can be used before declared, this kinds of mechanism is called [Hoisted](https://www.code-sample.com/2015/06/hoisted-in-javascript.html). It's a default behavior of [JavaScript](https://www.code-sample.com/2015/06/hoisted-in-javascript.html).

var obj = {num : 2};

var addtoThis = function(a,b,c) {

return this.num + a + b + c

}

console.log(addtoThis.call(obj, 3,3,3)); //functionname.call(obj, functionarguments)

var arr = [1,2,3];

console.log(addtoThis.apply(obj, arr)); //functionname.call(obj, functionarguments)

var bound = addtoThis.bind(obj)

console.log(bound(1,3,3));

Callback

let add = function (a,b) {

return a+b;

};

let multiply = function (a,b) {

return a\*b;

};

let doWhatever = function (a,b) {

console.log(“here are your two number $(a), $(b)”)’

};

let calck = function (num1, num2, callback){

if(typeof callback ==”function”){

return callback (num1, num2);

}

}

Console.log(calc(2,3, doWhatever ))

Console.log(calc(2,3, multiply ))

//simple way

let calck = function(num1, num2, calcType) { if (calcType ==="add"){ return num1 + num2; }else if (calcType === "multiply"){ return num1\*num2; } }; console.log(calck(2,3, 'add'));

Callback

var myArry = [{

Num:5,

Str: ”Apple”

}, {

Num:7,

Str: ”Cabbage”

}, {

Num:1,

Str: ”ban”

}];

myArr.sort(function(val1, val2){

if(val1.Str > val2.Str){

return -1;

}else {

return 1;

}

});

console.log(myArr);

The **Object.getPrototypeOf()** method returns the prototype (i.e. the value of the internal[[Prototype]] property) of the specified object.

var proto = {};

var obj = Object.create(proto);

Object.getPrototypeOf(obj) === proto; // true

var baseWidget = {

length:12,

width:24

},

widget = Object.create(baseWidget, {

color: {value:'blue', writeable:true, enumerable:true},

qty: {value:5, writeable:true, enumerable:true}

});

for (var i in widget ) {

if(widget.hasOwnProperty(i)){

console.log(i +":"+ widget[i]);

}

}

//output will be -- color:blue, qty:5

Another Way

Object.getOwnPropertyNames(widget).forEach(function(prop){

console.log(prop +":"+ widget[prop]);

})

//output will be -- color:blue, qty:5

var widgetProto = Object.getPrototypeOf(widget);

for (var i in widgetProto) {

if(widgetProto.hasOwnProperty(i)){

console.log(i +":"+ widgetProto[i]);

}

}

//output will be -- length:12 , width:24

A **callback** function, also known as a higher-order function, is a function that is passed to another function (let's call this other function “other Function”) as a parameter, and the **callback** function is called (or executed) inside the other Function.

## Factories vs. Services

First, right off the bat I’ll say they’re pretty much equivalent. Why do we have them both, then? That’s for the gods of Angular to know. They both allow us to create an object that can then be used anywhere in our app.

Most important is to realize that both are **singletons** in your app, even though the name “factory” might imply differently.

Essentially, factories are functions that return the object, while services are constructor functions of the object which are instantiated with the new keyword.

angular.module('app').factory('SomeService', function() {

return {

someFunction: function() {}

};

});

angular.module('app').service('SomeService', function() {

this.someFunction = function() {};

});

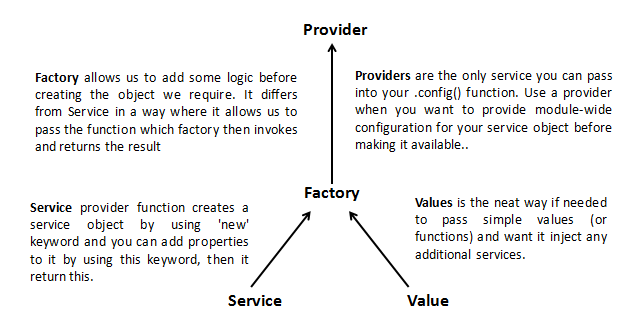
Service provider function creates a service object by using 'new' keyword and you can add properties to it by using this keyword, then it return this.

factory allows us to add some logic before creating the object we require. it differs from service in a way where it allows us to pass the function which factory then invokes and return the result.

When you create a new directive, you can write compile and/or linking functions for it to attach your custom behavior.

Service does not need to return anything. But you have to assign everything in this variable. Because service will create instance by default and use that as a base object.

Factory can return anything which can be a class(constructor function), instance of class, string, number or boolean. If you return a constructor function, you can instantiate in your controller.



### **45. WHAT IS LINKING FUNCTION?**

Link combines the directives with a scope and produce a live view. For registering DOM listeners as well as updating the DOM, link function is responsible. After the template is cloned it is executed.

### **46. WHAT IS PRE-LINKING FUNCTION IN ANGULARJS?**

Pre-linking function is executed before the child elements are linked. It is not considered as the safe way for DOM transformation

### **47. WHAT IS POST LINKING FUNCTION IN ANGULARJS?**

Post linking function is executed after the child elements are linked. It is safe to do DOM transformation by post-linking function.

### **48. WHAT IS COMPILE FUNCTION IN ANGULARJS?**

It is used for template DOM Manipulation and collect all of the directives.

### **49. WHAT IS LINK FUNCTION IN ANGULARJS?**

It is used for registering DOM listeners as well as instance DOM manipulation. It is executed once the template has been cloned.

**Q23. What is compilation process in Angular?**  
Ans: Once you have the markup, the AngularJS needs to attach the functionality. This process is called "**compilation**" in Angular. Compiling includes rendering of markup, replacing directives, attaching events to directives and creating a scope. The AngularJS has compiler service which traverses the DOM looking for attributes. The compilation process happens in two phases.

* **Compilation:** traverse the DOM and collect all of the directives and creation of the linking function.
* **Linking:** combine the directives with a scope and produce a live view. The linking function allows for the attaching of events and handling of scope. Any changes in the scope model are reflected in the view, and any user interactions with the view are reflected in the scope model.

When you create a new directive, you can write compile and/or linking functions for it to attach your custom behavior.

## What are Design Patterns?

## *Design patterns are solutions to general problems that software developers faced during software development. These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time.*

**Why should we use design patterns?**  
Design Patterns provides us lot of advantages. That’s why we should use them. Some common advantages are given bellow:

* It provides solution to common problems which occur in software design
* It provides common platform for developers means developer can implement these in any language
* It provides a standard terminology and is specific to particular scenario

### **3.What all are the types of Design Patterns?**

* Creational Patterns
  + This type of pattern address problems of creating an object and separating it from operations
* Structural Patterns
  + This type of pattern address problems of using object oriented constructs to organize classes and objects
* Behavioral Patterns
  + This type of pattern address problems of assigning responsibilities to classes

Creational Patterns:

* Singleton
* Factory
* Abstract Factory
* Prototype
* Builder

1. Module Design Pattern

2. Singleton Design Pattern

3. Prototype Design Pattern

4. Observer Design Pattern

5. Constructor Design Pattern

6. Revealing Module Design Pattern

7. Mediater Design Pattern

8. Command Design Pattern

9. Decorator Design Pattern

10. Flyweight Design Pattern

11. Facade Design Pattern

12. Factory Design Pattern

13. Mixin Design Pattern

// firing an event upwards

$scope.$emit('myCustomEvent', 'Data to send');

// firing an event downwards

$scope.$broadcast('myCustomEvent', {

someProp: 'Sending you an Object!' // send whatever you want

});

// listen for the event in the relevant $scope

$scope.$on('myCustomEvent', function (event, data) {

console.log(data); // 'Data to send'

});

We could fire an event down from ParentCtrl to SiblingOneCtrl using $broadcast:

app.controller('ParentCtrl',

function ParentCtrl ($scope) {

$scope.$broadcast('parent', 'Some data'); // going down!

});

app.controller('SiblingOneCtrl',

function SiblingOneCtrl ($scope) {

$scope.$on('parent', function (event, data) {

console.log(data); // 'Some data'

});

});

If we wanted to communicate upwards, from SiblingOneCtrl to ParentCtrl, you guessed it, we can use $emit.

app.controller('ParentCtrl',

function ParentCtrl ($scope) {

$scope.$on('child', function (event, data) {

console.log(data); // 'Some data'

});

});

app.controller('SiblingOneCtrl',

function SiblingOneCtrl ($scope) {

$scope.$emit('child', 'Some data'); // going up!

});

app.controller('SiblingOneCtrl',

function SiblingOneCtrl ($rootScope) {

$rootScope.$on('rootScope:emit', function (event, data) {

console.log(data); // 'Emit!'

});

$scope.$on('rootScope:broadcast', function (event, data) {

console.log(data); // 'Broadcast!'

});

$rootScope.$on('rootScope:broadcast', function (event, data) {

console.log(data); // 'Broadcast!'

});

});

app.controller('ChildCtrl',

function ChildCtrl ($rootScope) {

$rootScope.$emit('rootScope:emit', 'Emit!'); // $rootScope.$on

$rootScope.$broadcast('rootScope:broadcast', 'Broadcast'); // $rootScope.$on && $scope.$on

});

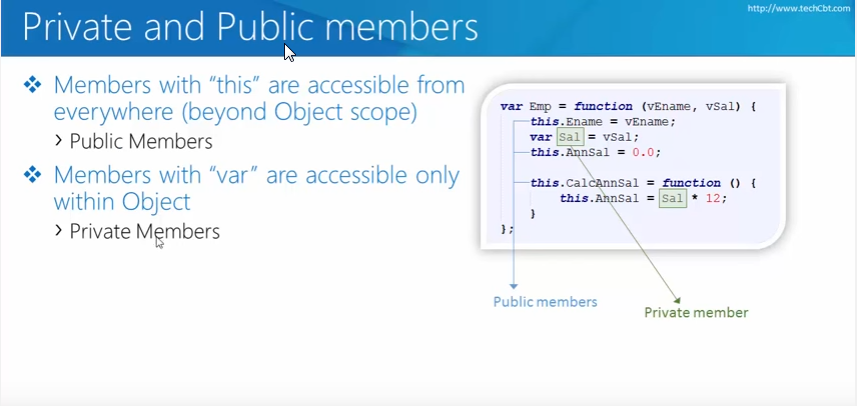
Private and Public Members

Members with “this” are accessible from everywhere (beyond Object scope)

* + - Public Members

Members with “var” are accessible only with Object

* + - Private Members



@mixin flex {

// write the css here

display: -webkit-flex;

display: flex;

.row {

@include flex;

}

$style1: 100%, 70px, #fo6d06;

$style2: (background: #bada55, width: 100%, height: 100px);

@mixin box($width, $height, $background) {

width: $width;

height: $height;

background-color: $background;

}

.fogdog {

@include box($style1...);

}

.badass {

@include box($style2...);

}

}

**Explain what is Sass? How it can be used?**

Sass stands for Syntactically Awesome Stylesheets and was created by Hampton Catlin. It is an extension of CSS3, adding nested rules, mixins, variables, selector inheritance, etc.

Sass can be used in three ways

* As a command line tool
* As a standalone Ruby module
* As a plugin for any Rack-enabled framework

**4) Explain how to define a variable in Sass?**

Variables in Sass begin with a ($) sign and variable assignment is done with a colon(:).

**7) Explain what is a @extend function used for in Sass?**

In Sass, the @EXTEND directive provides a simple way to allow a selector to inherit the styles of another one. It aims at providing a way for a selector A to extend the styles from a selector B. When doing so, the selector A will be added to selector B so they both share the same declarations.  @EXTEND prevents code bloat by grouping selectors that share the same style into one rule.

**) Why Sass is considered better than LESS?**

* Saas allows you to write reusable methods and use logic statements, e., loops, and conditionals
* Saas user can access Compass library and use some awesome features like dynamic sprite map generation, legacy browser hacks and cross-browser support for CSS3 features
* Compass also allows you to add an external framework like Blueprint, Foundation or Bootstrap on top
* In LESS, you can write a basic logic statement using a ‘guarded mixin’, which is equivalent to Sass if statements
* In LESS, you can loop through numeric values using recursive functions while Sass allows you to iterate any kind of data
* In Sass, you can write your own handy functions

.foo {

color: red;

}

.bar {

@extend .foo;

}

@mixin stuff {

color: red;

}

.foo {

@include stuff;

}

.bar {

@include stuff;

}

### **11) What are nested rules in Sass?**

Nesting is a method of combining multiple logic structures within one another. In Sass, multiple CSS rules are combined within one another.

For example, if you are using multiple selectors then you can use one selector inside another to create compound selectors.

10) Explain what is the use of Mixin function in Sass? What is the meaning of DRY-ing out a mixin?  
  
Mixin allows you to define styles that can be re-used throughout the stylesheet without needing to resort to non-semantic classes like .float-left.  
  
DRY-ing out of a mixing means splitting it into dynamic and static parts.  The dynamic mixin is the one that the user actually going to call, and the static mixin is the pieces of information that would otherwise get duplicated.

11) Explain what Sass Maps is and what is the use of Sass Maps?  
  
Sass map is a structured data in a hierarchical way and not just a bunch of variables. It can help in organizing the code. Some great use of Sass are  
  
It is very useful when dealing with layers of elements in your project  
It can be helpful in color management when there is long list of different color and shade  
Use icon map for various social media icons for example: facebook: ‘e607’ or twitter: ‘e602’  
Unlike other programming libraries, Sass map will consist only of code that is going to be used

**placeholder selectors**

Sass 3.2 introduced a special feature called placeholder selectors. They’re also referred to as “silent classes” because they won’t appear in our CSS output unless we @extend them.

It’s usually better to extend a placeholder selector instead of a class selector, so let’s make our .btn class a silent class. This way it won’t “exist” in our CSS output until we’re ready to use it.

A silent class is defined by using a % in front of the selector name:

React Question

What is ReactJS?

React.js is a JavaScript library developed by engineers at Facebook for building user interfaces. ReactJS allows us to create reusable UI components. It is currently one of the most popular JavaScript libraries and it has strong foundation and large community behind it.

# [What are the advantages/benefits and limitations of React?](http://www.interviewquestionspdf.com/2017/01/what-are-advantagesbenefits-and.html)

Following are the advantages/benefits and limitations of React  
**Advantages/benefits:**

* ReactJS can be used on client and server side too.
* ReactJS components are highly re-usable.
* React comes with a small API. Beginners will find it easy to learn and start using it.
* ReactJS uses virtual DOM which is JavaScript object. This will improve apps performance since JavaScript virtual DOM is faster than the regular DOM.
* Component and Data patterns improve readability which helps to maintain larger apps.
* ReactJS can be used with other frameworks (Backbone.js, Angular.js) as it is only a view layer..

**React Limitations:**

* React only covers view layer of the app so you still need to choose other technologies to get a complete tooling set for development.
* React is using inline templating and JSX. This can seem awkward to some developers.
* React library is too large

# [What is the difference between ReactJS and AngularJS?](http://www.interviewquestionspdf.com/2017/02/what-is-difference-between-reactjs-and.html)

|  |  |  |  |
| --- | --- | --- | --- |
| **Reactjs** |  | **Angualrjs** | |
| Age-3 years |  | Age- 7 years |
| Facebook |  | Google |
| JSX |  | HTML/JS/CSS |
| View layer only |  | Full MVC |
| All about components |  | View, Models, controllers |
| Less to code |  | A lot to code |
| The Library |  | The Framework |
| Small |  | Big |
| Easy to learn |  | Not easy as React |
| Speed-1.34 Seconds |  | Speed-310 Milliseconds |
| Virtual DOM |  | Regular DOM |
| Rendering - Server Side |  | Rendering - Client Side |
| Abstraction- Weak |  | Abstraction- Strong |
| Fails When? - Compile Time |  | Fails When? - Runtime |
| Packaging - Strong |  | Packaging - Weak |

# [What do you understand by JSX? Explain it.](http://www.interviewquestionspdf.com/2017/01/what-do-you-understand-by-jsx-explain-it.html)

JSX is a syntax extension for JavaScript. Basically HTML that goes into a js file to be compiled into real HTML. A JSX expression must have exactly one outermost element. Every JSX element is secretly a call to React.createElement().  
**Example:** var h1 = <h1>Hello, World!</h1>

**Examples of JSX elements with the attributes:**  
<a href="http://www.yahoo.com">Welcome to the Yahoo</a>;  
var title = <h1 id="title">Introduction to React.js: Part I</h1>;  
  
**An example of a nested JSX expression being saved as a variable:**  
var theFacebook = (  
<a href="https://www.facebook.com">  
<h1>  
Click me  
</h1>  
</a>  
);  
  
\*If a JSX expression takes up more than one line, then you should wrap the multi-line JSX expression in parentheses.

# [Explain what is Prop?](http://www.interviewquestionspdf.com/2017/01/explain-what-is-prop.html)

When we use our defined components, we can add attributes called props. These attributes are available in our component as this.props and can be used in our render method to render dynamic data.  
var MyComponent = React.createClass({  
    render: function(){  
        return (

 <h1>Hello, {this.props.name}!</h1>  
        );  
    }  
});

# [Explain Virtual DOM?](http://www.interviewquestionspdf.com/2017/01/explain-virtual-dom.html)

In React, for every DOM object, there is a corresponding "virtual DOM object." A virtual DOM object is a representation of a DOM object, like a lightweight copy. Think of manipulating the virtual DOM as editing a blueprint, as opposed to moving rooms in an actual house

# [How Virtual DOM Works?](http://www.interviewquestionspdf.com/2017/01/how-virtual-dom-works.html)

When you render a JSX element, every single virtual DOM object gets updated.This sounds incredibly inefficient, but the cost is insignificant because the virtual DOM can update so quickly. Once the virtual DOM has updated, then React compares the virtual DOM with a virtual DOM snapshot that was taken right before the update.  
By comparing the new virtual DOM with a pre-update version, React figures out exactly which virtual DOM objects have changed. This process is called "diffing."  
Once React knows which virtual DOM objects have changed, then React updates those objects, and only those objects, on the real DOM.

# [What is virtual DOM diffing?](http://www.interviewquestionspdf.com/2017/01/what-is-virtual-dom-diffing.html)

When Component rendering for the second time Virtual DOM checks what element get changed and print the changed element to the page, other elements remain untouched.  
The Virtual DOM diffing allows React to minimize changes to the DOM as a result of user actions, therefore increasing browser performance. This is why it's fast!

# [What's a difference between a DOM object and a virtual DOM object?](http://www.interviewquestionspdf.com/2017/01/whats-difference-between-dom-object-and.html)

* A virtual DOM object can update much faster than a regular DOM object.
* A virtual DOM object can't directly affect HTML.
* A virtual DOM object will be updated if ANY JSX element renders.

# [Why do we capitalize components?](http://www.interviewquestionspdf.com/2017/01/why-do-we-capitalize-components.html)

Because they're a constructor. It is not a rendered DOM element YET, it just has the capacity to render out a DOM element whenever we print it (with something like <Header />).

# [What is a state? How to use state?](http://www.interviewquestionspdf.com/2017/01/what-is-state-how-to-use-state.html)

State handles data changes. State is a JavaScript object that lives inside each component. We can access it via this.state. In React we modify the DOM indirectly, by updating each component's state and letting React handle updates to the DOM.  
The state is a vital part of React apps, making user interfaces interactive. Represents data that changes over time. We declare an initial state in the component's constructor. As first line call the super function. (To communicate with other components). In order to create the initial state for a component, we must declare the property this.state as an object in the class constructor function.  
We update state by calling this.setState(), this causes our component to re-render.

# [How you will update a Component's state?](http://www.interviewquestionspdf.com/2017/01/how-you-will-update-components-state.html)

We update state by calling this.setState(), this causes our component to re-render.  
  
this.setState({showComments:true})  
Will only update the properties an argument, not replace the entire state object.

How is state available, and what is it by default?  
this.state == null

# Understand the concepts of ownership and children in ReactJS

Let’s explore how does the ownership works in React. What exactly is a child in React’s world by looking at the special property : this.props.children.

var MyContainer = React.createClass({

render: function() {

return <MyChild value={this.props.value} />

}

});

// Or in ES6

const MyContainer = (props) => <MyChild value={props.value} />

We don’t talk about parent/child, which refers to the DOM relationships.

If we wrap a <div> around <MyChild>:

var MyContainer = React.createClass({

render: function() {

return <div><MyChild value={this.props.value} /></div>

}

});

// Or in ES6

const MyContainer = (props) => <div><MyChild value={props.value} /></div>

<MyContainer> is not the parent of <MyChild> (the <div> is in between), but it’s more generally its owner.

In the DOM hierarchy, the <div> is the representation of <MyContainer> which is the parent of whatever <MyChild> can represent.

# [How can you set state initially (within the constructor() method)?](http://www.interviewquestionspdf.com/2017/01/how-can-you-set-state-initially-within.html)

constructor() {  
super();  
this.state = { name: "Vikas" };  
}

# [How can you access something inside state?](http://www.interviewquestionspdf.com/2017/01/how-can-you-access-something-inside.html)

this.state.\_\_\_\_\_ inside JSX {} inside render  
if this.state = { name: "Nick" }; then...  
{this.state.name}

# [What are Synthetic events?](http://www.interviewquestionspdf.com/2017/01/what-are-synthetic-events.html)

In order to ensure events have consistent properties across different browsers, React wraps the browser's native events into "synthetic events", consolidating browser behaviors into one API. Synthetic events are a cross-browser wrapper around the browser's native event system.

# [What are props? How to pass and read them? React JS interview question](http://www.interviewquestionspdf.com/2017/02/what-are-props-how-to-pass-and-read.html)

Arguments passed to components can be accessed using this this.props object. They look similar to regular HTML element attributes. We use this.props object to read parameters that were passed to the component.  
 **We pass parameters to the component:**  
<Comment name="My Name"/>

**In the component we use this.props object to read from it:**  
class Comment extends React.Component {  
render (){  
return (  
<p className="comment">{this.props.name}</p>  
)  
}  
}

# [What is a Flux action? ReactJS Interview Question](http://www.interviewquestionspdf.com/2017/02/what-is-flux-action-reactjs-interview.html)

It's a Javascript object that describes what we want to do, and the data we need to do it.

# [What does "handling state" mean? What exactly are we trying to handle? ReactJS Interview Question](http://www.interviewquestionspdf.com/2017/02/what-does-handling-state-mean-what.html)

The data that comes from the server.

# [What is redux? React Js Interview Question](http://www.interviewquestionspdf.com/2017/02/what-is-redux-react-js-interview.html)

A method of handling the state (or the data) of an app.

# [What is polling?](http://www.interviewquestionspdf.com/2017/02/what-is-polling.html)

In order to check whether new comments are added, we can periodically check the server for updates. Use setInterval on componentDidMount every 5 second call the \_FetchComment function.  
This might sound expensive, but it's not.React optimizes the rendering process by only updating the DOM when changes are detected on the resulting markup.

# [Difference between props and state? Facebook ReactJS Interview Question](http://www.interviewquestionspdf.com/2017/02/difference-between-props-and-state.html)

While a component has information from both its props and its state, props are immutable and owned by a component's parent, whereas state is mutable and owned by the component. this.state is private to the component and can be updated with this.setState(). As with props, when the state updates the component will re-render itself.

**NPM :**

**npm** , short for **Node Package Manager**, is two things: first and foremost, it is an online repository for the publishing of open-source Node.js projects; second, it is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management  
  
NPM is a Node Package Manager and it's use for

* It is an online repository for the publishing of open-source Node.js projects.
* Command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

Javascript run time engine, for packaging eco system so it is very important to download any node js library.

mkdir angularDemo

Gulp: Automate and enhance your workflow

**Gulp result in a much shorter and cleaner task configuration file.** In other words, with Gulp, there is less code to write for doing the same processes

* Grunt focuses on configuration, while Gulp focuses on code
* Grunt was built around a set of built-in, and commonly used tasks, while Gulp came around with the idea of enforcing nothing, but how community-developed micro-tasks should connect to each other

Gulp is the project management tools, this is building and deploy application

Bower is nothing it is just a packaging manager which is install for the frontend technology like html JavaScript and css

Bind: Doesn't work on dynamically added event

The jQuery .bind() method registers event type and an event handler directly to the DOM element.

**Using Live Method:**

That method is deprecated now. Advantage of using live () is that it uses the concept of event bubbling.  
The live () method attaches the event handler to the root level of the document along with associated selector and the event information. This can also respond to the events which are generated by dynamically added elements. Doesn’t support Channing events

**Using On Method:**

You can consider jQuery .on() method as being ‘overloaded’ with different signatures. This method attach the event handler function for one or more events to the selected elements. jQuery .on() function was included in jQuery 1.7 so, for early versions it will not work.

**Using Delegate Method:**

jQuery .delegate() method behaves in a similar way to the .live() method. But, instead of attaching the selector or event information to the document, you can choose where it is anchored. That is you can control on which node the events will be added. This can also respond to the dynamic element events.

What does a task runner do ?

Performs repetitive task

* + - Prefixing css rules
    - Compiling SASS files in css
    - Minifying JS / CSS file
    - Concatenating Files

Task Runner

* + Grunt
  + Gulp

NodeJs:

* + A platform which allows us to run javascipt on a computer/server
  + Read, delete and update files
  + Easily communicate with database

Why is Node.js so Popular ?

* + It uses javascript
  + Very fast (run on the V8 engine & uses non-blocking code)
  + Huge ecosystem of open source package (npm)
  + Great for real time services (like chats)

Computer do not understand javascript

A Javascript engine take javascirpt, and converts it inot something it does understand- machine code

## The Difference Between Arrays and Objects

In JavaScript, **arrays** use **numbered indexes**.

In JavaScript, **objects** use **named indexes**.

Arrays with named indexes are called associative arrays (or hashes).

The **splice()** method can be used to add new items to an array:

var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(2, 0, "Lemon", "Kiwi");

The first parameter (2) defines the position **where** new elements should be **added** (spliced in).

The second parameter (0) defines **how many** elements should be **removed**.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be **added**.

**7) explain Jquery live and bind methods ?**  
.bind() attacheds events to elements that exist or match the selector at the time the call is made.  
Any elements created afterwards or that match going forward because the class was changed, will not fire the bound event.

$(‘img’).bind(‘click’, function(){…});

.live() works for existing and future matching elements.  
Before jQuery 1.4 this was limited to the following events:  
click, dblclick mousedown, mouseup, mousemove, mouseover, mouseout, keydown, keypress, keyup  
$(‘img’).live(‘click’, function(){…});

**9) type of webservice ?**

there are two types of web service….1. SOAP [Simple Object Access Protocol] Webservice and 2. RESTful [REpresentational State Transfer] Webservice.  
SOAP is a messaging protocol , REST is a design philosophy , not a protocol.

http://csslint.net/

PageSpeed Insights

https://gtmetrix.com

|  |  |
| --- | --- |
| **SVG** | **Canvas** |
| Here’s it’s like draw and remember. In other words any shape drawn by using SVG can be remembered and manipulated and browser can render it again. | Canvas is like draw and forget. Once something is drawn you cannot access that pixel and manipulate it. |
| SVG is good for creating graphics like CAD software’s where once something is drawn the user wants to manipulate it. | Canvas is good for draw and forget scenarios like animation and games. |
| This is slow as it needs to remember the co-ordinates for later manipulations. | This is faster as there is no intention of remembering things later. |
| We can have event handler associated with the drawing object. | Here we cannot associate event handlers with drawing objects as we do not have reference of them. |
| Resolution independent. | Resolution dependent. |

var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

var centerX = canvas.width / 2;

var centerY = canvas.height / 2;

var radius = 70;

context.beginPath();

context.arc(centerX, centerY, radius, 0, 2 \* Math.PI, false);

context.fillStyle = 'green';

context.fill();

context.lineWidth = 5;

context.strokeStyle = '#003300';

context.stroke()

https://www.code-sample.com/2015/04/javascript-interview-questions-answers.html