Javascript Interview Questions and Answers

Pratik Bandal

Javascript Interview Questions and Answers

Pratik Bandal

Javascript Interview Questions and Answers

Pratik Bandal

About Book

Javascript Interview Questions and Answers is extremely useful for beginners and professionals who are preparing for interview of Javascript or web technologies. This book covers interview questions of Javascript which is required form UI and full stack development. It is also recommended to go through other web technologies like HTML, CSS as it is difficult to cover so many technologies in this small book.

The book will prepare you for interview as well as refining your knowledge of Javascript and will make you ready for working in industry.

For any positive or negative feedback regarding this book, kindly reach me at pratikbandal007@gmail.com

About Author

I am Pratik Bandal currently working in a big multinational company and have over 6 years of experience in software industry. I have done B.E. (E&TC) and Post graduate diploma in Advanced Computing. I have experience in different domains like healthcare, payment and banking sectors.

But on top of all, I am developer like you all guys doing an 8 hour job. Writing is something I do extra and I love doing it. No one is perfect and holds true for me as well. Without all you guys I am not an author.

Writing an interview question book is a really great deal of responsibility. But huge natural variations in interview are something difficult to cover in this small book.

Contents

1. Fundamentals

1. What is Javascript?

Javascript is basically scripting language used to make web pages more interactive as it can be inserted into HTML.

Javascript is understood by all modern web browsers.

2. What are features of Javascript?

Input validation : Javascript allows you to validate user input before sending it to server for backend operations.

Control over browser : Javascript gives more control over browser due to which you can change the background colour of this page as well as the text on the browser's status bar.

Detect browser and OS : Javascript enables you to detect user’s browser and OS due to which you can perform platform dependant operations.

Handling date and time : Javascript enables you to write code based users date and time, it also enables you to capture users date and time as user and server may be in different time zone.

Generating HTML on fly : Javascript enables you to dynamically generate HTML.

3. What are advantages of Javascript?

Faster speed : JavaScript is fast because it run immediately within the client-side browser. Javascript is not dependant on network unless backend data is required to be processed. Need to compile Javascript on the client–side as it is interpreted directly by web browsers.

Interoperability : Javascript can be inserted into web page regardless of extension. Within other languages such as Perl and PHP it can be used inside the script.

Rich interfaces : Javascript has vast libraries like (charts, drag and drop,

sliders etc.) which enables you to provide attractive look to your website.

Reduction in server load : Since Javascript is client-side scripting language it reduces load on website servers as many operations can be performed at client-side which reduces load on server and enables it serve to more users.

4. What are disadvantages of Javascript?

Client-side security : JavaScript code executes in users computer hence in some cases it can be manipulated for malicious purpose.

Browser support : JavaScript is sometimes interpreted differently by different browsers.

5. What is difference between Javascript and ECMAScript?

JavaScript is a scripting language that has been formed by keeping ECMAScript specification at its core.

ECMAScript is nothing but a standard or specification defined in order to create different scripting languages and one of them is JavaScript.

Javascript, Jscript and ActionScript are few scripting languages that follow ECMAScript specifications.

6. Who developed Javascript?

JavaScript was created in 1995 by Brendan Eich during his time at Netscape Communications. It was inspired by Java, Scheme and Self.

7. How to insert Javascript in Web page?

You can use <script> tag in html. <script> tag has type attribute which defined which code is there inside the script tag.

You can use <script> element in web pages in following ways:

- In head element
- In body element

- As an external script file

To use Javascript as a scripting language for web pages in <head> or <body> you can define type as “text/javascript”.

e.g.

<script type=”text/javascript”>

</script>

Sometime you may need to use same Javascript code in several web pages, in such cases you can store Javascript code in external file and save file as <filename>.js file.

Then this script can be made available to web page using src attribute of <script> tag

e.g.

<script src=”external file URL”>

</script>

8. What are advantages of using external javascript?

Placing JavaScript code in external js files has few advantages over inline scripts:

Segregating HTML and JavaScript code helps to manage the code base better.

To improve development output designers can work along with coders in parallel without code conflicts.

This approach also works well with modern source code version control systems like GIT and SVN.

Each of these files can maintain history.

Segregating HTML and JavaScript makes code as well as HTML is easily readable.

Segregated external JavaScript files are cached by browsers and can speed up page load times

These small js files can be minified to reduce the size and make it not readable by humans, using Google closure or YUI Compressor or other.

Many popular JavaScript libraries are available as hosted on content delivery networks (cdn) and you can simply point to them using the URL in

the src, this avoids copying the js file to local folder.

Using external Js you can take benifits of advanced tools such as RequireJS or CommonJS to load these scripts logically and modularly

9. Javascript is case sensitive language?

Yes, Javascript is case sensitive scripting language. Variables, functions, keywords must have consistent casing otherwise it will not be recognized by Javascript and will generate error.

e.g.

```
var pratik;
var praTik;
```

In above case pratik and praTik will be considered as different variables.

10. Is Semicolon compulsory in Javascript at end of the statement?

No, it is not necessary to use semicolon at end of the statement still it will be considered as valid statement.

11. What are different data types in Javascript?

Below are basic data types in Javascript:

Primitive Data Types:

- Number: Represent numeric values, both integer and float.
- String: Sequence of characters are represented using String.
- Boolean: Represent Boolean value, true or false.
- Undefined: Represent undefined value.
- Null: Represent null.

Non-Primitive Data Types:

- Object: Represent more complex data structure.
- Array: Represent group of elements.
- RegExp: Represent regular expression.

12. How to add comment?

JavaScript provides two kinds of comments:

Single-line comments and multiline comments.

Single-line comments start with // and are terminated by the end of the line:

e.g.

```
x++; // single-line comment
```

Multiline comments are delimited by /* and */:

e.g.

```
/* This is
 a multiline
comment.
*/
```

13. What is Javascript engine?

JavaScript engine is a computer program used to execute Javascript code.

JS engines were developed by web browser vendors and every major browser has one.

Chrome V8 from google is most used engine, Google chrome use it.

SpiderMonkey is developed by Mozilla for use in firefox.

JavaScriptCore is Apples engine for its Safari browser.

14. What are different Javascript frameworks that you know?

Many frameworks are based on Javascript now, below are few of them:

- Angular
- React
- Vue
- Node
- Ember
- Meteor

- Backbone
- Aurelia
- Polymer
- Mithril

2. Variables, Operators and Statements

15. Explain variable in Javascript.

Basically, variable is used for temporary storage of data. It has name, value and memory address. You have to declare variable before using it for storing data. Below is syntax to declare variable:

var variablename;

Here var is keyword and variablename is name of the variable.

You can also define multiple variables using single statement as:

var variable1, variable2, variable3;

Value can be assigned to variable as:

var variablename = value;

16. What is var?

The var statement declares a variable and can also optionally initialize its value.

Variables are declared using var as below,

var varname1 [= value1] [, varname2 [= value2] ... [, varnameN [= valueN]]];

17. What is let?

The 'let' allows you to declare variables that are limited in scope to the particular block, expression on which it is used or statement.

Variables are declared using let as below,

let var1 [= value1] [, var2 [= value2]] [, ..., varN [= valueN];

18. What is const?

Constants are block-scoped, much like variables defined using the let statement. The value of a constant cannot change through reassignment, and it can't be re-declared.

19. What is difference between let and var?

The variable defined with var is available anywhere within the function hence ’var’ keyword has function scope.

The let has a Block Scope. A variable declared with ‘let’ keyword has a scope only with in that block.

20. What is difference between let and const?

‘let’ allows you change the value of variable any number of times.

Using ‘const’, after the first assignment of the value we cannot redefine the value again.

21. What is automatic type conversion?

When Javascript tries to operate on a wrong date type it will try to convert the value to a “right” type.

22. What are operators in Javascript?

An operator is a symbol or word which is used to perform particular operation.

Arithmetic, Assignment, Comparison, Logical and Conditional Operators are types of operators.

23. Explain different types of operators in Javascript.

Below are types of operators:

- **Arithmetic operators** : Arithmetic operators are used to perform arithmetic between variables and values. Addition (+), Subtraction (-), Multiplication (*), Division (/), Modulus (%), Increment (++) and Decrement (--) are arithmetic operators.
- **Assignment operators** : Assignment operators are used to assign values to variables. =, +=, -=, *=, /=, %= are Assignment operators.
- **Comparison operators** : Comparison operators are used to

compare variables or values. Equal to (==), equal value and type (===), not equal (!=), not equal value or type (!==),less than (<), greater than (>), less than or equal to (<=) and greater than or equal to (>=) are comparison operators.

- **Logical operators** : Logical operators allow program to make decision based on multiple conditions logic. Logical operators used for decision making are And (&&), or (||), not (!).
- **Conditional operator** : Conditional operators are used to assign values to variable conditionally. (condition) ? value1: value2 is conditional operator.

24. What are control flow statements?

You can change the sequence in which Javascript statements are executed by using control flow statements.

Below are types of control flow statements:

Selection statements : Selection statements use condition to determine which group of statements should be executed, if….else, if and switch are selection statements.

Loops : Loops allow you to execute group of statements repeatedly till condition is satisfied, while, do…while and for are loops.

Jump statements : Jump statements are used to break or exit loop, break and continue are jump statements.

25. What is break statement?

Break statement stops execution of loop entirely.

26. What is continue statement?

Continue statement stops execution of current iteration in a loop and continues with next iteration of loop.

27. What is the difference between comparing variables using

"==" and "===" operator?

The ‘==’ operator tests for abstract equality i.e. it does the required type conversions before doing the equality comparison.

But the ‘===’ operator tests for strict equality i.e. it will not do the type conversion thus if the two values are not of the same type, when compared, it will return false.

28. What is typeof operator?

The typeof operator is used to get the data type of its operand. The operand can be either a literal or a data structure such as variable, function or an object.

e.g.

console.log(typeof somevar);

The typeof operator returns below values as string: object, Boolean, function, number, string and undefined.

29. What is variable hoisting?

In Javascript regardless of where the actual declaration has been made, all variable declarations that are using var, are hoisted/lifted to the top of their functional/local scope (if declared inside a function) or to the top of their global scope (if declared outside of a function).

This lifting of scopes is called hoisting.

Hence,

```
bla = 2;
var bla;
// ...is implicitly understood as:
var bla;
bla = 2;
```

30. What is difference between undefined and null?

The undefined means a variable has been declared but has no value has yet been assigned.

On the other hand, null is basically a value which has been assigned.

Also, undefined is a type itself (undefined) while null is an object.

Unassigned variables are initialized with a default value of undefined by JavaScript or undefined can be assigned to variable through code.

Whereas JavaScript never sets a value to null.

That must be done programmatically.

31. What is output of null == undefined?

null == **undefined** will return **true** .

However, **null** === **undefined** will return **false** .

32. What are escape characters?

Escape characters (backslash) is used before special characters like ampersand, single quotes, double quotes and apostrophes to display them.

e.g.

console.log('I\'m Pratik Bandal'); ☐ Correct syntax

console.log('I'm Pratik Bandal'); ☐ Syntax error

In above example, if backslash is not used before single quotes this line will give syntax error.

33. How to create array in javascript?

You can define arrays using the array literal as follows-

var a = [];

var b = [1, 2, 3];

34. How to create three dimensional array?

You can define three dimentionalarray arrays using the array follows:

var threedimentionalarray = [[[]]];

35. What are the variable naming conventions in JavaScript?

The following rules are to be followed while naming variables in JavaScript:

You are not allowed to use any of the reserved keyword as variable name.

JavaScript variable names should not begin with a numbers (0-9).

They must start with a letter or the underscore character.

JavaScript variable names are case sensitive.

36. Why you should not prefer to use global variables?

Global variable can be created by many developers resulting in duplicate global variables.

Duplicate variable can overwrite the value of your variable.

3. Functions

37. What are functions?

Function is a collection of statements which can be used anywhere in program, it is used to perform specific task.

38. What are types of functions in Javascript?

Below are the types of functions:

Named: Functions which have name at the time of definition are named functions.

e.g.

```
function print() {
    console.log(“This is named function!!!”);
}
```

Anonymous: Functions which do not have names are anonymous functions.

```
var print=function() {
    console.log(“This is anonymous function!!!”);
}
```

39. What are frequently used built-in global functions?

Alert(), prompt(),isNan(), eval(), isFinite(),confirm(), parseInt(), parseFloat(), escape(), unescape() are most frequently used built-in global functions.

40. What is isNaN?

It is a function which determine whether or not value is an illegal number. The isNan() method returns true if the passed value is NaN(Not a number) and is of type number, else it returns false.

e.g.

Input: ‘213’

Output: false

Input:’hello’

Output: true

41. What is parseInt?

The parseInt is a function which parses the string and returns the integer value found in string.

42. What is alert?

The alert() function is used to display information in message box.

43. What is confirm?

The confirm() function displays a message box with two buttons, Ok and cancel. When you click the Ok button, the function returns true. When you click cancel button function returns false.

44. What is charAt?

The charAt() function returns character from specified index.

e.g.

```
var str=”Pratik”;
console.log(str.charAt(0));

Output :
P
```

45. What is indexOf?

This function returns the index within the calling string object of first occurrence of the specified value and returns index of found occurrence or -1 if not found.

e.g.

```
var str=”This is javascript book”;
```

```
console.log(str .indexOf(“javascript”));
```

Output:

8

46. What are function scopes?

Scope defines accessibility of function and its variables. In Javascript scope is divided into two categories:

- **Global** : Function with global scope can be accessed anywhere in the program.
- **Local** : Function with local scope can be accessed only within its parent function.

47. What is strict mode?

Strict mode prevents certain actions and throws more exceptions. The statement “use strict” orders browser to use the Strict mode, which is a reduced and safer feature set of JavaScript.

Strict mode eliminates some silent errors in JavaScript by changing them to throw errors.

Strict mode resolves mistakes that make it difficult for JavaScript engines to perform optimizations hence strict mode code can sometimes run faster than identical code that’s not strict mode.

Strict mode forbids some syntax likely to be defined in future versions of ECMAScript.

It prevents, or throws errors, when unsafe actions are taken (such as gaining access to the global object).

It disables features that are confusing or poorly thought out.

Due to Strict mode it becomes easier to write secure JavaScript.

Strict mode applies to individual functions or to entire scripts. It doesn't apply to block statements enclosed in {} braces; attempting to apply it to such contexts does nothing.

To invoke strict mode for an entire script, put statement "use strict" before any other statements.

To invoke strict mode for a function, put statement "use strict" in the function's body before any other statements.

48. What is function closure?

A closure is a feature in JavaScript where an inner function has access to the outer (enclosing) function’s variables.

49. What is callback function?

A function passed into another function as an argument, which is then invoked inside the outer function to complete some kind action is called as callback function.

e.g.

```
function showName(name) {
  alert('User name is:' + name);
}
function displayName(callback) {
  var name = prompt('Please enter name to be displayed');
  callback(name);
}
displayName(showName);
```

50. What is setTimeout function?

The setTimeout() function executes function at specified interval.

```
setTimeout(expression, timeout);
```

Here, expression is the function/code that is called only once and timeout is number of milliseconds to wait before calling the function.

The clearTimeout() function is used to deactivate or cancel timer set by setTimeout() fuction.

51. What is setInterval function?

The setInterval() function executes a function after a specified time interval.

```
setInterval(expression, timeout);
```

Here, expression specifies function/code to be called after particular time interval and timeout specifies the time interval between function calls.

The clearInterval() function is used to cancel or deactivate the timer set by setInterval() function.

52. What is difference between setInterval and setTimeout functions?

The setTimeout(expression, timeout) runs the function once after timeout

The setInterval(expression, timeout) runs the function in intervals repeatedly, with length of timeout between them.

53. What is encodeURI() method?

The encodeURI() method encodes a Uniform Resource Identifier by replacing each instance of particular characters by one, two, three or four escape sequences representing UTF-8 encoding of character.

54. What is decodeURI() method?

The decodeURI() method decodes a Uniform Resource Identifier previously created by encodeURI().

4. Events

55. What are Events?

Events are actions or occurrences that happen in the system you are programming to which you can respond in some way. Events are handled by function know as event handler.

56. What are different events in Javascript?

Few of the important events are listed below:

Input Events

- onsubmit - triggers on submitting a form.
- onselect - triggers on selecting an element.
- onchange - triggers when changes happen to an element.
- onfocus - triggers when windows gets focus.
- onreset - triggers when user clicks reset button.
- onblur - triggers when window loses focus.
- onkeyup - triggers on releasing a key.
- onkeydown - triggers on pressing a key.

Click Events

- onclick - trigger on clicking a mouse button.
- ondblclick - triggers on double clicking mouse button.

Mouse Events

- ondrag - triggers when element is dragged.
- ondragend - triggers when drag ends.
- ondragstart - triggers when drag starts.
- ondragenter - triggers when dragged element is dropped.
- ondragleave - triggers on leaving target while dragging element.
- onmouseover - triggers when mouse pointer moves over element.
- onmousedown - triggers on pressing mouse button.
- onmouseup - triggers on releasing mouse button.
- onscroll - triggers on scrolling a scroll bar of an element.

Load Events

- onload- triggers when page has been loaded.
- onerror- triggers when an error occurs when loading an image.
- onunload- triggers when browser closes document.

57. What are event handlers?

Event handler is a routine that is used to deal with event, allowing programmer to write code that will be executed when event occurs.

58. What is addEventListener() method?

The addEventListener() method attaches an event handler to specified element.

You can add multiple event handlers to one element.

It is possible to add event listener to any DOM object.

e.g.

```
document.getElementById(“someUniqueDivId”).addEventListener(“click”, respondtoclick);
function respondtoclick() {
        console.log(“Do some stuff!!!”);
}
```

59. How to remove event listener from any element?

The removeEventListener() is an inbuilt function in JavaScript which removes an event handler from an element for attached event.

Below example show how to remove event listener which was added in previous example.

e.g.

```
document.getElementById(“someUniqueDivId”).removeEventListener("click respondtoclick);
```

60. What are different key codes?

It is necessary to know the codes associated with keys to identify which key is pressed in the code.

Below table gives key codes:

Key	Code	Key	Code	Key	Code	Key	Co
backspace	8	A	65	numpad 0	96	semi-colon	
Tab	9	B	66	numpad 1	97	equal sign	
Enter	13	C	67	numpad 2	98	comma	
Shift	16	D	68	numpad 3	99	dash	
Ctrl	17	E	69	numpad 4	100	period	
Alt	18	F	70	numpad 5	101	forward slash	
pause/break	19	G	71	numpad 6	102	grave accent	
caps lock	20	H	72	numpad 7	103	open bracket	
escape	27	I	73	numpad 8	104	back slash	
page up	33	J	74	numpad 9	105	close braket	
page down	34	K	75	multiply	106	single quote	
End	35	L	76	add	107		
home	36	M	77	subtract	109		
left arrow	37	N	78	decimal point	110		
up arrow	38	O	79	divide	111		
right arrow	39	P	80	f1	112		
down arrow	40	Q	81	f2	113		
insert	45	R	82	f3	114		
delete	46	S	83	f4	115		
0	48	T	84	f5	116		
1	49	U	85	f6	117		
2	50	V	86	f7	118		
3	51	W	87	f8	119		
4	52	X	88	f9	120		
5	53	Y	89	f10	121		
6	54	Z	90	f11	122		
7	55	left window key	91	f12	123		

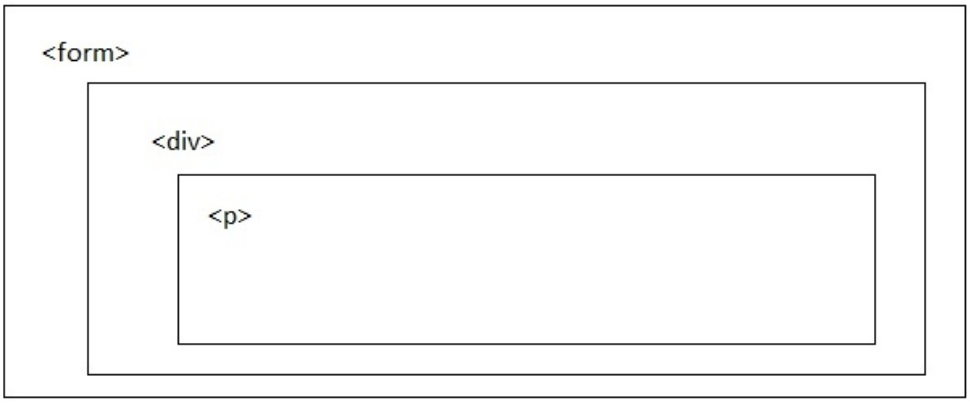
8	56	right window key	92	num lock	144		
9	57	select key	93	scroll lock	145		

61. What is event bubbling?

When an event happens on an element, it first runs the handlers on that particular element, after that handlers on its parent runs, this happens all the way up on all other ancestors.

This bubbling of events from child to parent is called event bubbling.

A click on inner <p> first runs onclick on that <p>, then on outer , then on outer <form> and so on till document object.



This process is called “bubbling”, because events “bubble” from the inner element up through parent like a bubble in the water.

62. Is it possible to stop event bubbling?

Yes, by using method event.stopPropagation().

If you want to stop the event flow from event target to top element in DOM, event.stopPropagation() method stops the event to travel to the bottom to top.

63. What is event capturing?

In the capturing phase:

The browser checks to see if the element's outer-most ancestor (<html>) has an onclick event handler registered on it during the capturing phase, and runs it if so.

Then it moves on to the next element inside <html> and does the same thing, then the next one, and so on until it reaches the element that was actually clicked on.

The capturing phase is not often used. Usually it is invisible to us.

64. What is event delegation?

Event delegation concept relies on the fact that if you want some code to run when you click on any one of a large number of grouped child elements, you can set the event listener on their parent and have events that happen on them bubble up to their parent, instead of having to set the event listener on every child individually.

A good example is a series of list items <li> — You can set the click event listener on the parent <ul>, if you want each one of them to pop up a message when clicked and it will bubble to the list items.

5. Objects

65. What is Object?

The object is collection of properties and methods.

Object in Javascript are variables as well. Object can have properties any data types (String, Number, Boolean etc.).

Object properties can be primitive values, other objects and functions.

e.g.

```
var book = {
        name : “Javascript Book”,
        auther: “Pratik Bandal”,
        pages: 100
}
```

66. What are ways to create objects?

In Javascript you can create objects through following ways:

Using literals:

Using object literal you can basically create object by using name value pairs inside curly {} braces.

e.g.

```
var book = {
        name : “Javascript Book”,
        auther: “Pratik Bandal”,
        pages: 100
}
```

Using new keyword with built-in object constructor function:

Using new keyword you can create object and then set its properties as below,

e.g.

```
var book= new Object();
```

```
person.name= "John";
person.auther= "Doe";
person.pages= 50;
```

Using new keyword with built-in user defined constructor function:

We first create a constructor function and then get objects using ‘new’ keyword. .

e.g.

```
function Book(name, author, pages) {
        this.name = name;
        this.author = author;
        this.pages = pages;
}
```

And then we create Book object as below,

```
var book = new Book(“Javascript Book”, ”Pratik Bandal”, 100);
```

Using Object.create():

The Object.create() method creates a new object, using an existing object as the prototype of the newly created object.

e.g.

```
const person = {
      isHuman: false,
      printIntroduction: function () {
            console.log(`Am I human? ${this.isHuman}`);
      }
};
const me = Object.create(person);
```

67. Which are built-in or native objects?

JavaScript provides Number, Boolean, String, Array, Date, Math, RegExp which are built in objects.

68. What is ‘this’ keyword?

The this keyword refers to the object it belongs to.

In an object method, this refers to the object to which method belongs.

When used alone, the owner is the Global object, so this refers to the Global object (Windows object).

In a function, this refers to the Global object (Windows object).

In strict mode, when used in a function, this is undefined.

In HTML event handlers, this refers to the element in html that received the event.

69. What is String object?

String object is basically sequence of characters. String object provides number of methods to perform required operations on String object.

It provides methods like charAt(), charCodeAt(), concat(), indexOf(), match(), slice(), split(), substr(), toLowerCase(), toUpperCase(), valueOf(), toString() which provides important functionalities that can be performed on string.

70. What is Number object?

The number is Javascript wrapper object which allows you to work with numerical values.

Number object is created as,

var numberVar = new Number([value]);

It provides methods like isNaN(), isFinite(), isInteger(), isSafeInteger(), parseInt(), parseFloat() to work with numbers.

71. What is Boolean object?

The Boolean object wraps a boolean value.

Boolean object is created as,

var booleanVar = new Boolean([value]).

If the value is omitted or is 0, -0, null, false, NaN, undefined, or the empty

string (""), the object has an initial value of false. All other values, including any object or the string "false", create an object with an initial value of true.

72. Explain about Array object.

The Array is a global object that is used to store different elements for the construction of arrays.

Array object is created as,

new Array(ele0, ele1[, ...[, eleN]])

new Array(arrayLength)

73. Explain about Date object.

The JavaScript Date object represents a single moment in time. Date objects use a unix timestamp which is a integer value that is number of milliseconds since 1 January 1970.

Date object is created as,

new Date();

new Date(value);

new Date(dateString);

new Date(year, monthIndex [, day [, hours [, minutes [, seconds [, milliseconds]]]]]);

74. Explain about Math object.

Math is a built-in object that has methods and properties for mathematical constants and functions.

Unlike the other global objects, Math does not have a constructor. All methods and properties of Math are static.

It provides methods like sin(x), cos(x), tan(x), exp(x), floor(x), max([x[, y[, …]]]), min([x[, y[, …]]]), pow(x, y), random(), round(x), trunc(x) for mathematical operations.

75. Explain about RegExp object.

The RegExp constructor is used to create a regular expression object for matching text with a pattern.

Date object is created as,

new RegExp(pattern[, flags])

76. What is namespace?

In JavaScript, namespace is a single global object which will contain all our functions, methods, variables.

Javascript don’t provide default namespace you have to create it, so all functions, variables and object in Javascript are by default global.

77. How to create namespace?

Below is example to create namespace and access function within it:

```
var myProjectNameSpace = {
    projectfunctionone: function() {
    },
    projectfunctiontwo: function() {
    }
}
.
.
.
myProjectNameSpace. Projectfunctionone();
```

78. What is prototype in javascript?

All objects in Javascript have property called as prototype, the prototype is an object which has a constructor properties by default.

The prototype object is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property is not visible.

The prototype property allows you to add properties and methods to any

object.

e.g.

somecustomcreationobject.prototype.age=29;

79. What is prototypal inheritance?

Object have property called as prototype which can refer to other object.

When you want to read a property from object, and it’s missing, JavaScript automatically takes it from the prototype. This is called “prototypal inheritance”.

80. What is difference between call() and apply()?

The Function.prototype.call() method calls a function with a provided this value and arguments provided individually.

It is necessary to know arguments of function when using call() method.

The Function.prototype.apply() method calls a function with a provided this value, and arguments provided as an array (or an array-like object).

It is necessary to know arguments of function when using apply() method.

81. What is Promise?

The Promise object represents the eventual completion (or failure) of an asynchronous operation along with its resulting value.

A Promise is a proxy for a value which may or may not be known when the promise is created.

It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.

This enables asynchronous methods return values like synchronous methods: instead of immediately returning final value, the asynchronous method returns a promise to provide the value at some point in the future.

6. Browser Object Model

82. What is Browser Object Model?

The browser object model (BOM) is a hierarchy of browser objects that are used to manipulate methods and properties associated with the Web browser itself.

The default object of browser is window means you can invoke all the functions of window by specifying window or directly

Objects that make up the BOM include the window object, navigator object, screen object, location object, history, and the document object.

83. What is Window Object?

In a tabbed browser, each tab is represented as Window object.

Every object, variable, and function defined in a web page uses of the window as its Global object.

Window object provides methods like alert(), blur(), close(), confirm(), print(), prompt(), open().

84. What alert method in window object?

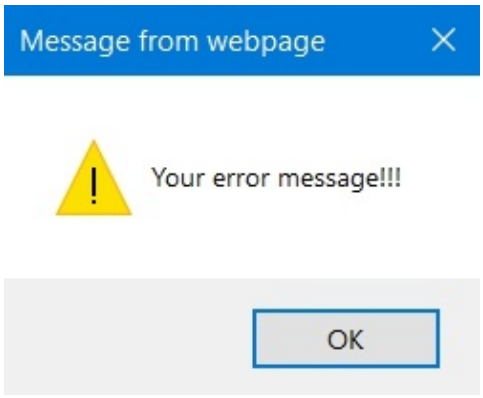
Alert dialogs are typically used when users has to be made aware of something that they have no control over, such as errors.

Often alert dialogs pops up when the user enters invalid data into a form.

When alert() is called,the browser creates a system message box that displays the given text with an OK button.

For example, the following line of code causes the message box in to be displayed:

alert(“Your error message!!!”);



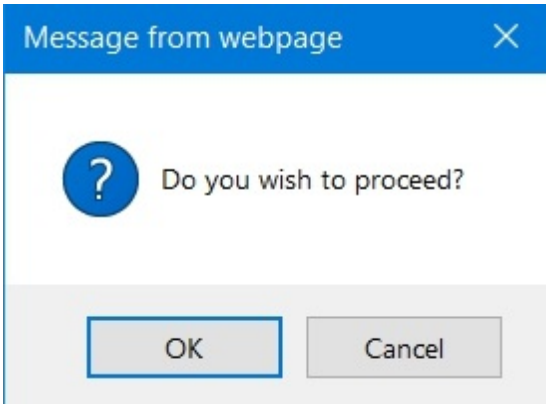
85. Explain confirm method of window object.

A confirm dialog appears similar to an alert dialog.

The main difference between them is the presence of a Cancel button along with the OK button in the confirm dialog, which allows the user to confirm if a given action should be taken.

For example, the following line of code displays the confirm dialog shown in Figure

confirm(“Do you wish to proceed?”);



86. Explain prompt method of window object.

The prompt method is used to display dialog with input from user.

Along with OK and Cancel buttons, prompt dialog also has a text box where the user is asked to enter some data.

The prompt() method accepts two arguments: the text to display to the user and the default value for the text box (which can be an empty string if you so

desire).The following line results in the window displayed:

prompt("Enter the country","USA");

87. How to redirect other webpage?

It is possible to redirect to other webpage in javascript by directly assigning value to window.location or by using location.assign(),location.replace() and location.reload() methods.

88. What is Navigator Object?

The navigator object is used to get browser information like name, version, type, language. It has methods like javaEnabled() and taintEnabled().

89. How to identify operating system of client device?

“Navigator.appVersion” is used to find operating system of client device.

90. What is History Object?

The History object consist of array of URLs which are visited by a user in browser.

History object provides method like back(), forward() and go().

91. How to load previous page in browser programmatically?

history.back() can be used to load previous page in browser through code.

92. How to load next page in browser programmatically?

history.forward() can be used to load next page in browser through code.

93. What is go method of history object?

The go() method loads a specific URL from the history list.

history.go(number|URL)

number|URL parameter can either be a number which goes to the URL within the specific position (1 goes forward one page, -1 goes back one page), or a string. The string has to be a partial or full URL, and the function will go to the first URL that matches the string.

94. What is Screen Object?

The Screen object consist of information about display screen like height, width and colour bits of screen.

95. What is Location Object?

The Location object consist of information about current URL of window object.

Location object provides methods like assign(), reload and replace().

96. How to print a web page?

The window.print() will print the current web page when invoked.

7. Document Object Model

97. What is Document Object?

The Document object represents HTML document that is displayed in window. Document object has properties which allows access and modification of document content.

The way document is accessed and modified is called Document Object Model or DOM.

Document object provides methods like open(), close(), write(), getElementById(), getElementByName(), getElementByTagName().

98. What is DOM?

The Document Object Model (DOM) represents HTML or XML page in such a way that programs can change document structure, content and style.

The DOM represents the document as nodes as well as objects.

The DOM is object oriented representation of web page, which can be modified by scripting language like Javascript.

99. What are DOM nodes?

According to the W3C HTML DOM standards, everything in HTML can be represented as nodes.

Document node represents whole document.

Element node represents every HTML element such as HTML, HEAD, BODY, A, H1 etc.

Text node represents text content inside the element.

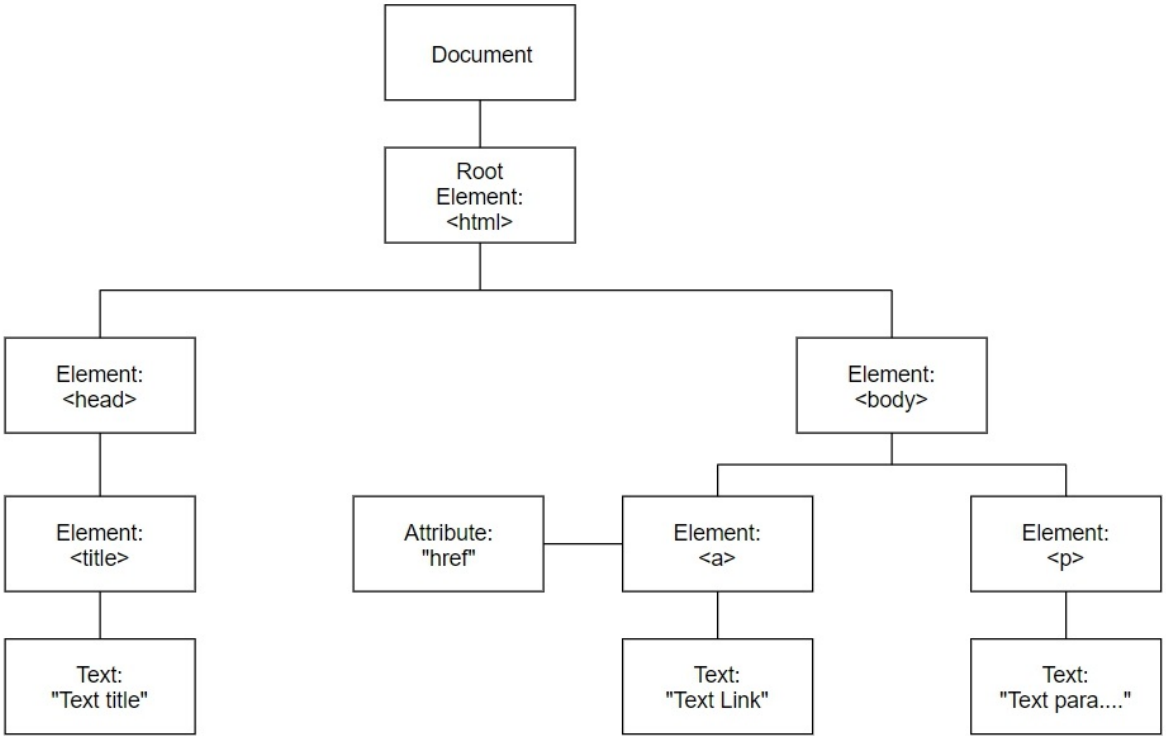
Attribute node represents every HTML attribute.

Node has node properties that contain information about the node

The nodeName property specifies name of the node.

The nodeValue property specifies value of the node.

The nodeType property specifies type of the node.



100. How to get element with id in DOM?

The getElementById() method of document object can be used to get element using id.

e.g.

document.getElementById(“myUniqueId”);

101. How to get element using class in DOM?

The getElementByClassName () method of document object can be used to get element using class.

e.g.

document.getElementByClassName(“myClass”);

102. How to get content of any element?

The innerHTML property is useful for getting or replacing the content of HTML elements.

103. What are DOM levels?

The W3C DOM specifications are divided into different levels where each level contain some required and optional modules.

Level 0: Provide low-level set of interfaces.

Level1: DOM level 1 can be described in two parts: CORE and HTML.

CORE provides a low level interfaces that can be used to represent any structured document.

HTML provides high-level interfaces that can be used to represent HTML document.

Level2: Consist of six specifications:

CORE2, VIEWS, EVENTS, STYLE, TRAVERSAL and RANGE.

CORE2: extends functionality of CORE specified by DOM level 1.

VIEWS: views allow programs to dynamically access and manipulate content of document.

EVENTS: events are scripts that are executed when user reacts to web page.

STYLES: allow programs to dynamically access and manipulate content of style sheets.

TRAVERSAL: allows programs to dynamically traverse the document.

RANGE: allows programs to dynamically identify range of content in document.

Level3: consists of five different specifications: CORE3, LOAD, SAVE, VALIDATIONS, EVENTS and XPATH.

CORE3: extents functionality of CORE specified by DOM level 2.

LOAD and SAVE: allows program to dynamically load the content of XML document into DOM document and save DOM document into XML document by serialization.

VALIDATION: allows program to dynamically update the content and structure of document while ensuring the document is valid.

EVENTS: extents functionality of Events specified by DOM level 2.

XPATH: XPATH is a path language that can be used to access DOM tree.

104. What are deferred scripts?

By default, Javascript files will interrupt parsing of HTML document in order for them to be fetched and executed.

The defer attribute tells browser to only execute the script file once the HTML document has been fully parsed.

<script defer src=”myscript.js”>

This reduces loading time of web page and web page is displayed faster.

105. What are asynchronous scripts?

By default, Javascript files will interrupt parsing of HTML document in order for them to be fetched and executed.

The async attribute is used to indicate browser that script file can be executed asynchronously.

<script async src=”somescript.js”>

The HTML parser does not have pause at the point it reaches the script tag to fetch and execute, the execution can occur whenever the script becomes ready after being fetched in parallel with document parsing.

106. What is difference between attribute and property?

Attributes: Provide more details on an element like id, type, value etc.

Property: Value assigned to the property like type=”text”, value=’Name’ etc.

107. What is the difference between innerHTML & innerText?

innerHTML: It will process an HTML tag if found in a string

innerText: It will not process an HTML tag if found in a string

108. What is the difference between textContent & innerText?

The textContent returns every element in the node.

The innerText is aware of styling and won't return the text of “hidden” elements.

109. What is HTMLCollection?

The HTMLCollection interface represents a generic collection of elements (in document order) and offers methods & properties for selecting from the list.

HTMLCollection has length property which returns the number of items in the collection.

It is not possible to iterate over HTMLCollection list using forEach by default.

110. What is NodeList?

NodeList objects are collections of nodes which are usually returned by properties such as Node.childNodes and functions such as document.querySelectorAll().

NodeList has length property which returns the number of nodes in nodelist.

for...of loops will loop over NodeList objects accurately.

111. What are frames?

Frame divides page into section and in each section different page can be displayed.

8. Cookies

112. What is cookie?

The Cookies are small items of data that consists of name and value pair. Cookies are stored on your computer so that it can be accessed by your web browser.

A web browser and server communicate through HTTP which is stateless protocol. Stateless protocol treats each request independently, so server does not keep data after sending it to browser. With cookies such data can be fetched directly from stored cookie file instead of communicating with server.

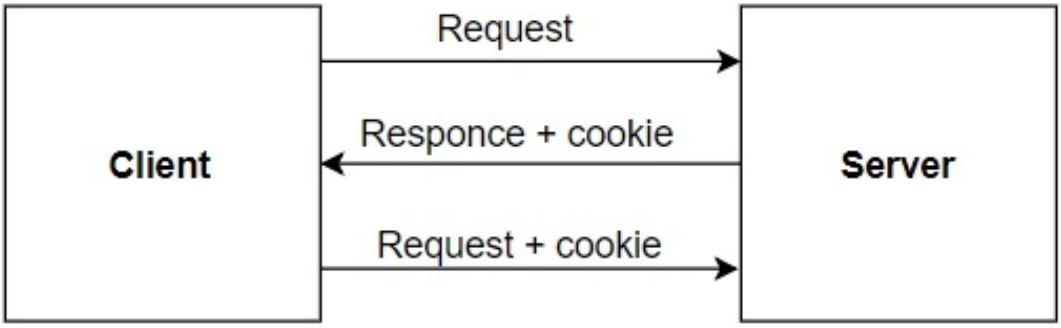
For example when user visits web page, user name can be stored in cookie. Now when next time user visits the page cookie belonging to the page is added to the request. This way server gets necessary data “remembered” by cookie.

113. How cookie helps client server HTTP communication?

When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.

When receiving an HTTP request, a server can possibly send a Set-Cookie header with the response.

Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users



114. Where are cookies stored?

During browsing session browser stored cookies in memory, at the time of quitting they go to file called as cookies.txt.

Different browser store cookies file in a different location on disk.

For instance, on windows chrome stores the cookies in below location, C:\Users\<YourUser>\AppData\Local\Google\Chrome\User Data\Default\

As cookie expires it is no longer saved on hard drive.

115. Where are parameters of cookie?

There are six parameters of cookie: name, value, expires, path, domain and security.

The name and value are required whereas all other parameters are optional.

document.cookie="name=VALUE;expires=DATE;path=PATH;domain=DOM

Name and Value : The first part of cookie must have name and value. The entire name/value must be a single string with no commas, semicolons or whitespace charactors.

Expires : The cookie will disappear when user exits the browser, to give more life to the cookies you must set an expiration date in the following format.

DD-Mon-YY HH:MM:SS GMT

Path : Usually the path is set to root level directory ('/'), which means the cookie is available for all the pages of your site. If you want the cookie to be readable in specific directory <directoryname>, path should be specified as path=/< directoryname>.

Domain : Some websites have lots of domains. The purpose of the 'domain' is to allow cookies to other subdomains. In case, if website is http://www.<domain>.com with subdomains http://www.<subdomainone>.<domain>.com and http://www.<subdomaintwo>.<domain>.com. If web page on subdomainone set a cookie pages on subdomaintwo cannot read that cookie. But if you add domain=<domain> then all subdomains ending with <domain> can read the cookie.

Secure : The last parameter of cookie is secure which is a Boolean value. Its default value is false. If cookie is marked as secure then cookie will be sent to

web server and try to retrieve it using secure communication channel.

116. Can user disable cookies?

Yes, user can disable cookies from browser.

In chrome, you can go to settings-->Advanced Settings-->Privacy and security-->Content setting-->Cookies and disable cookies.

117. How to create cookie?

When visitor visits web page for first time he enters his or her name. This name will be stored in cookies as below,

```
function createCookie(username, value) {
    document.cookie=username + ”=” + value;
}
```

118. How to read cookie?

Javascript cookies can be read like this,

```
var x = document.cookie;
```

119. How to delete cookie?

While deleting cookie you don’t have to specify value.

Javascript cookies can be deleted by specifying expires parameter to a past date.

```
document.cookie=”username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path= /; ”;
```

Some browsers will not let you delete cookie if you don’t specify the path.

120. What is difference between local storage and session storage?

Local Storage : For every HTTP request, the data is not sent back to the server (HTML, images, JavaScript, CSS, etc) reducing the total traffic between client and server. Data will stay until it is manually cleared using

settings or through program.

Session Storage : It is similar to local storage; the only difference is data stored in local storage has no expiration time whereas data stored in session storage gets cleared when the page session ends. Session Storage will cleared when the browser is closed.

9. Form validation

121. What is form validation?

Form validation verifies whether all fields in form are filled according to required format.

If data entered by user is not according to format then appropriate error message is displayed to the user.

Forms can be validated using server-side as well as client-side validations.

Server-side validation is more secure and required server connection to validate, whereas client-side validation is quicker and doesn't require server connection but it is less secure.

Javascript is used for client side validation.

Advantages of client side validation is that, it saves time, reduces load on server and can validate form element even before form is submitted.

122. What is required attribute?

The required attribute in HTML element prevents that element being submitted as blank.

e.g.

<input type="text" name="employeename" required>

In above case, as long as text field employeename is blank form submission will be prevented.

123. What is pattern attribute?

The pattern attribute specifies a regular expression against which elements value is checked.

If element value does not match the regex pattern form submission will be prevented.

e.g

<input type="text" name="employeename" pattern="[A-Za-z]">

In above case, if employee name contains value which is not alphabet then form submission will be prevented.

124. How to validate form using Javascript function?

Below example shows validation of form using Javascript.

Here, we have created form having name input and then when while saving save validateName function will be called and it will validate if name is blank.

```
<script>
function validateName() {
    var name = document.nameform.name.value;
    if(name==undefined || name=="") {
        alert("Kindly enter the name!!!");
        return false;
    }
    Return true;
}
</script>
<form name="nameform" method="post" onsubmit="return validateform()">
Name: <input type="text" name="name">
<input type="submit" name="save">
</form>
```

125. How to validate email in the form?

Below function can be used to validate email in the form.

```
function validateEmail(emailField){
    var reg = /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$/;
    if (reg.test(emailField.value) == false)
    {
        alert('Invalid Email Address');
        return false;
    }
```

```
    return true;
}
```

126. How to validate field without submitting form?

To validate field without submitting form you can use validation function in onblur event of input field.

<input type="text" onblur="validateEmail(this);" />

Here, input field will be available to validateEmail function.

127. What is .test method?

The .test() API runs a search for a match between a regex and a string.

The .test() API returns a Boolean(true/false), returns true if test passes and false if it doesn't.

Using .test() returns no data, so don't expect any.

128. What is .match method?

Using .match() is best when you are expecting data back in test result, .match() returns an array with matches or simply null if there are none.

With match you won't just be testing for presence of data, you will also see if data pattern exist and return that data.

129. How to validate Date?

```
function validateDate(dateField) {
    var reg = /^([0-9]{2})\/([0-9]{2})\/([0-9]{4})$/
    if (reg.test(dateField.value) == false) {
        alert("Invalid Date!!!");
        return false;
    }
    return true;
```

```
}
```

130. How to allow number only in input field?

```
function validateNumber(numField) {
        var reg = /^[0-9]+$/
        if (reg.test(numField.value) == false) {
            alert(“Invalid Number!!!”);
            return false;
        }
        return true;
}
```

10. Error and Exception Handling

131. What is error object?

When an exception occurs, an object representing the error is constructed and thrown.

The Error constructor creates an error object.

When runtime errors occur, instances of Error objects are thrown.

The Error object can be used as a base object for user-defined exceptions.

var error = new Error("error message");

“Error” objects contain two properties, “name” and “message”. The “name” property specifies the type of exception The “message” property provides a more detailed description of the exception.

The “message” gets its value from the string passed to the constructor of exception.

132. What are different error types in Javascript?

Below are primary error types in javascript:

SyntaxError : Raised when syntax error occurs while parsing the Javascript code.

RangeError : Raised when numeric value exceeds allowed range.

EvalError : Raised when the eval() function is used in an incorrect manner.

ReferenceError : Raised when an invalid reference is used

TypeError : Raised when type of variable is not as expected.

URIError : Raised when the encodeURI() or decodeURI() functions are used in an inaccurate manner.

InternalError : Raised when internal error in the javascript engine is thrown.

133. How to handle exceptions in JavaScript?

JavaScript uses the try...catch...finally statement as well as the throw operator to handle exceptions.

You can catch user-defined and runtime exceptions, but you cannot catch JavaScript syntax errors.

134. Explain try…catch…finally.

Try : wraps suspicious code that may throw an error in try block.

Catch : Write code to do something when error occurs in catch block. The catch block can have parameters which will give you error information. Usually, catch block is used to log an error or display specific messages to the user.

Finally : code in finally block will always be executed regardless of the occurrence of an error. The finally block is usually used to complete the remaining task or reset variables that might have changed before error occurred in try block.

135. How to throw exceptions programmatically?

It is possible to throw exceptions programmatically using “throw” statement.

There is no restriction on data type that can be thrown as an exception.

e.g.

throw “error occurred!!”;

throw new SyntaxError(“syntax error occurred!!”);

11. Debugging

136. What is debugging?

Debugging is the process of detecting and fixing existing and potential errors in software code that can cause it to behave unexpectedly.

To debug a program, programmer has to start with problem, identify source of the problem and then fix it.

Sometimes it takes more time debugging a program than coding it.

137. What is debugger keyword?

Debugger statement stops the execution of Javascript. If debugging functionality is not available, this statement has no effect.

Debugger keyword is like breakpoint in script source code.

e.g.

```
function someErroraniousFunction() {
    debugger;
    code;
}
```

138. What is console object?

Console object provides access to browsers debugging console.

If browser supports debugging you can use console.log() method to display required text in debugging window.

Console object provides methods like debug(), log(), error(), info(), trace(), warn() which are useful for code debugging.

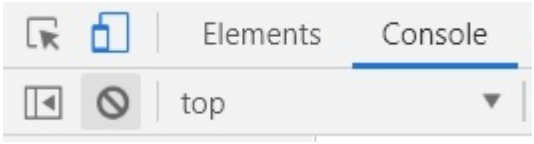
139. How to activate debugging in browser?

You can activate debugging in browser by pressing F12 and then select console in debugger menu.

140. How to get mobile devices view of webpage in desktop browser?

In browser press F12,

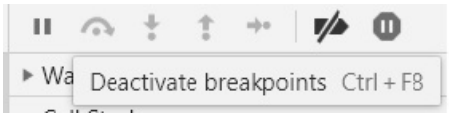
Then click on toggle device toolbar,



Then, select device for which you want webpage view.

141. How to deactivate breakpoint in browser?

This can be done by clicking on “deactivate breakpoints” icon in “Sources” tab of browser developer tool.



142. How to pause script execution?

This can be done by clicking on “pause script execution” icon in “Sources” tab of browser developer tool.

143. How to execute function line by line while debugging?

This can be done by clicking on “Step into next function call” icon in “Sources” tab of browser developer tool.

144. How to execute function without stepping into it while debugging?

This can be done by clicking on “Step over next function call” icon in “Sources” tab of browser developer tool.

145. What is code smell?

In computer programming, a code smell is any characteristic in the source code of a program which possibly indicates a deeper problem.

Determining what is a code smell and what is not a code smell is subjective, and varies by language, developer, and development methodology.

The two main know open source tools used for JavaScript code analysis are JSLint and JSHint, the second being a fork of the first one. There are however many different tools that try to achieve the same goal and you might find something more suited to your own needs

12. AJAX overview

146. What is AJAX?

AJAX stands for Asynchronous Javascript and XML. It is collection of related technologies like Javascript, XML, JSON, HTML and XMLHttpRequest etc.

AJAX allows you to send and receive data asynchronously without reloading web page and hence makes web pages more fast and interactive.

147. What is difference between GET and POST?

	GET	**POST**
History	Parameters remain in browser history as they are part of URL.	Parameters are not saved in browser history.
Security	GET is less secure as data sent as part of URL.	POST is more secure than GET because parameters are not stored in browser history.
Parameters	Parameter data is limited to what you can stuff in URL. Safest to use less than 2K of parameters.	Parameters can contain uploaded data files and larger data than GET.
Caching	Can be cached.	Not cached.

148. What is XMLHttpRequest object?

XMLHttpRequest object is used for asynchronous communication between client and server.

It provides methods like open(), send(), setRequestHeader() for exchanging data between client and server.

149. How to make HTTP GET call using AJAX?

To make HTTP call in AJAX, you first need to initialize a new XMLHttpRequest() object.

Specify URL endpoint, HTTP method (GET) to open() method of XMLHttpRequest() object.

Then call send() method to hit the request.

Receive the response using XMLHttpRequest.onreadystatechange property.

e.g.

```
const xmlHttpRequest = new XMLHttpRequest();
xmlHttpRequest.open("GET","http://some.domain.com/method");
xmlHttpRequest.send();
xmlHttpRequest.onreadystatechange=(e)=>{
    console.log(xmlHttpRequest.responceText);
}
```

150. How to make HTTP POST call using AJAX?

To make HTTP call in AJAX, you first need to initialize a new XMLHttpRequest() object.

Specify URL endpoint, HTTP method (POST) to open() method of XMLHttpRequest() object.

Then call send() and pass data to send() method to hit the request.

Receive the response using XMLHttpRequest.onreadystatechange property.

e.g.

```
const xmlHttpRequest = new XMLHttpRequest();
xmlHttpRequest.open("POST","http://some.domain.com/method");
xmlHttpRequest.send("fname=Pratik&lname=Bandal");
xmlHttpRequest.onreadystatechange=(e)=>{
    console.log(xmlHttpRequest.responceText);
}
```

151. What are HTTP status codes?

HTTP status code are the standard response code given by web site servers. These codes help identify the cause of problem when web page or other resource does not load properly.

4xx Client Error:

This category of HTTP status code includes those where request for a web page or other resource contains bad syntax or cannot be filled for some other reason, presumably due to fault of client.

Some common client error HTTP status codes are 404 (Not Found), 403 (Forbidden) and 400 (Bad request).

5xx Client Error:

This category of HTTP status code include those where the request for a web page or other resource is understood by the websites server but is incapable of filling it for some reason.

Some common server error HTTP status codes are 500 (Internal server error), 503 (Service Unavailable) and 502 (Bad Gateway).

There are also 1xx, 2xx and 3xx code that are informational, confirm success or dictate redirection which are not errors, so you shouldn’t be alerted about them.

You may also like to read (available on amazon):

Angular Interview Questions and Answers: Includes Angular 8, 7, 6, 5, 4 and 2

Core Java Interview Questions and Answers: Includes Java 12, 11, 10, 9, 8, 7, 6, 5