1. What are the data types available in javascript?

Six data types that are [primitives](https://developer.mozilla.org/en-US/docs/Glossary/Primitive):

[Boolean](https://developer.mozilla.org/en-US/docs/Glossary/Boolean)

[Null](https://developer.mozilla.org/en-US/docs/Glossary/Null)

[Undefined](https://developer.mozilla.org/en-US/docs/Glossary/Undefined)

[Number](https://developer.mozilla.org/en-US/docs/Glossary/Number)

[String](https://developer.mozilla.org/en-US/docs/Glossary/String)

[Symbol](https://developer.mozilla.org/en-US/docs/Glossary/Symbol) (new in ECMAScript 6)

and [Object](https://developer.mozilla.org/en-US/docs/Glossary/Object)

Primitive values

All types except objects define immutable values (values, which are incapable of being changed).

1. What is DOM?  
   The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of the document and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods. Essentially, it connects web pages to scripts or programming languages.  
   The Document Object Model (DOM) provides another way to represent, store and manipulate that same document. The DOM is a fully object-oriented representation of the web page, and it can be modified with a scripting language such as JavaScript.
2. What is Shadow DOM?  
   Shadow DOM provides encapsulation for DOM and CSS in a [Web Component](https://developer.mozilla.org/en-US/docs/Web/Web_Components).   
   We already using shadow DOM like <img><section><mark><header><video>etc
3. How can you declare a class in Javascript?  
     
   This is probably one of the most common ways. You define a normal JavaScript function and then create an object by using the new keyword. To define properties and methods for an object created using function(), you use the this keyword, as seen in the following example.

function Apple (type) {

this.type = type;

this.color = "red";

this.getInfo = getAppleInfo;

}

// anti-pattern! keep reading...

function getAppleInfo() {

return this.color + ' ' + this.type + ' apple';

}

// you may end up defining a lot of these functions and they are all in the "global namespece". This means you may have naming conflicts if you (or another library you are using) decide to create another function with the same name.

function Apple (type) {

this.type = type;

this.color = "red";

this.getInfo = function() {

return this.color + ' ' + this.type + ' apple';

};

}  
A drawback of above is that the method getInfo() is recreated every time you create a new object. Sometimes that may be what you want, but it's rare. A more inexpensive way is to add getInfo() to the prototype of the constructor function.  
function Apple (type) {

this.type = type;

this.color = "red";

}

Apple.prototype.getInfo = function() {

return this.color + ' ' + this.type + ' apple';

};

To instantiate an object using the Apple constructor function, set some properties and call methods you can do the following:

var apple = new Apple('macintosh');

apple.color = "reddish";

alert(apple.getInfo());  
  
**Literals are shorter way to define objects and arrays in JavaScript.**

create an empty object using you can do:

var o = {};

instead of the "normal" way:

var o = new Object();

For arrays you can do:

var a = [];

instead of:

var a = new Array();

So you can skip the class-like stuff and create an instance (object) immediately. Here's the same functionality as described in the previous examples, but using object literal syntax this time:

var apple = {

type: "macintosh",

color: "red",

getInfo: function () {

return this.color + ' ' + this.type + ' apple';

}

}

In this case you don't need to (and cannot) create an instance of the class, it already exists. So you simply start using this instance.

apple.color = "reddish";

alert(apple.getInfo());

**Such an object is also sometimes called singleton**. **singleton means that you can have only one single instance of this class at any time, you cannot create more objects of the same class. In JavaScript (no classes, remember?) this concept makes no sense anymore since all objects are singletons to begin with.**

**Singleton using a function**

**var apple = new function() {**

**this.type = "macintosh";**

**this.color = "red";**

**this.getInfo = function () {**

**return this.color + ' ' + this.type + ' apple';**

**};**

**} but the way to use the object is exactly like in 2.**

**apple.color = "reddish";**

**alert(apple.getInfo());**

**new function(){...} does two things at the same time: define a function (an anonymous constructor function) and invoke it with new.**

1. What is Difference between null and undefined?  
     
   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.

1. How can you add a method to a already defined class?  
   function DefinedClass(){  
   }  
   DefinedClass.protoype.somemethod= function(){  
   alert somethig  
   }
2. What is Javascript closures and what are the disadvantages of closures? Give an Example.  
   Closures are functions that refer to independent (free) variables (variables that are used locally, but defined in an enclosing scope). In other words, these functions 'remember' the environment in which they were created.

function init() {

var name = "Mozilla"; // name is a local variable created by init

function displayName() { // displayName() is the inner function, a closure

alert (name); // displayName() uses variable declared in the parent function

}

displayName();

}

init();

What is difference between == and === in javascript?  
  
The 3 equal signs mean "equality without type coercion". Using the triple equals, the values must be equal in type as well.  
Equal (==)

If the two operands are not of the same type, JavaScript converts the operands then applies strict comparison. If either operand is a number or a boolean, the operands are converted to numbers if possible; else if either operand is a string, the other operand is converted to a string if possible. If both operands are objects, then JavaScript compares internal references which are equal when operands refer to the same object in memory.

0 == false // true

0 === false // false, because they are of a different type

1 == "1" // true, automatic type conversion for value only

1 === "1" // false, because they are of a different type

null == undefined // true

null === undefined // false

'0' == false // true

'0' === false // false

1. How to empty an array?  
   a=[1,21,3];  
   a=[] //this way IT actually creates brand new array  
   You should be careful with this method because if you have referenced this array from another variable or property, the original array will remain unchanged. Only use this if you only reference the array by its original variable A.  
   Second option is a.length = 0;  
   This will clear the existing array by setting its length to 0.   
   a = []; // 37% slower

a.length = 0; // 89% slower

a.splice(0, a.length) // 97% slower

while (a.length > 0) {

a.pop();

} // Fastest

1. Can we compare two javascript objects? How?  
   function isEqual(a, b)   
   {   
   var aProps = Object.getOwnPropertyNames(a), bProps = Object.getOwnPropertyNames(b); if (aProps.length != bProps.length) { return false; } for (var i = 0; i < aProps.length; i++) { var propName = aProps[i]; if (a[propName] !== b[propName]) { return false; } } return true; }

1. Explain Promise in javascript, implementation promise in javascript.

<https://www.sitepoint.com/overview-javascript-promises/>  
<https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise>  
  
if (window.Promise) { // Check if the browser supports Promises

var promise = new Promise(function(resolve, reject) {

//asynchronous code goes here

});

}

1. What is an anonymous function? What are the advantages of anonymous function?  
   <http://thoughtsonscripts.blogspot.in/2012/01/javascript-anonymous-functions.html>
2. What is ‘use strict’?  
   Strict mode changes previously accepted "bad syntax" into real errors.

As an example, in normal JavaScript, mistyping a variable name creates a new global variable. In strict mode, this will throw an error, making it impossible to accidentally create a global variable.

1. What is typeof operator?  
   The "typeof" operator in JavaScript allows you to probe the data type of its operand, such as whether a variable is string, numeric, or even undefined.
2. What is the instanceof operator in JavaScript?
3. How to handle exceptions/errors in javascript?
4. Explain pseudo-classical inheritance vs prototypal Inheritance. Which one is better and why?
5. List ways of creating javascript classes / object
6. Difference between Object.create() vs new keyword  
   The object used in Object.create actually forms the prototype of the new object, where as in the new Function() form the declared properties/functions do not form the prototype.
7. What is hoisting?
8. Explain scope(and types of scope) and context in javascript.
9. Explain lexical scope.
10. What are the valid scopes of a variable in Javascript?
11. What is arguments object in Javascript?
12. What is callback?
13. How can you get the reference of a caller function inside a function?
14. What is IIFE(Immediately-Invoked Function Expression), What are the advantages of it? Why we use that?
15. How can you get the total number of arguments passed to a function?
16. Can you name two programming paradigms important for JavaScript app developers?  
    <https://www.webcodegeeks.com/javascript/javascript-interview-questions-answers/>
17. Explain functional programming paradigm. What is pure function?
18. What are the pros and cons of functional programming vs object-oriented programming?
19. When is classical inheritance an appropriate choice?
20. When is prototypal inheritance an appropriate choice?
21. Explain these methods call(), apply() and bind(). Explain difference between them.
22. How do identify an object is of array type?
23. What is asynchronous programming, and why is it important in JavaScript?
24. What is event-loop? Explain with diagram.
25. What are the different types of errors supported by JavaScript?
26. Explain following design patterns with implementation in javascript.
27. Singleton, Factory, Observer, Pub-Sub, Module, Prototype, Chain of responsibility.
28. What is bubbling?
29. Write a script to achieve following output.
30. sum(<number>)(<number>)(<number>)....
31. Eg. sum(1)(3)(5) = 9
32. sum(2)(1)(4)(5)(2) = 14
33. Print 1 to 10, each in some seconds(eg. 1, 2, 10) of interval.
34. Reverse a string.
35. Sort an object in both direction by key.
36. Write a script to capitalize first letter of each word from given string.
37. What would be the output of 10+20+"30"?
38. What would be the output of “10”+20+30?
39. What will be the output of the code below?
40. var y = 1;
41. if (function f(){}){
42. y += typeof f;
43. }
44. console.log(y);
45. What will be the output of the code below
46. (function(fname){
47. this.fname = fname;
48. }(“krunal”));
49. console.log(fname);
50. How to empty an array in Javascript?
51. Write an output of following code.
    1. foo();
    2. var foo = function foo(){
52. return 12;
53. };
54. Write function which prints fibonacci series
55. Using regular loop
56. Using recursive function