<https://www.toptal.com/javascript/interview-questions>

<https://www.ajaymatharu.com/javascript-difference-between-undefined-and-null/>

var a=null;

>>undefined

if(a){console.log("True")} else{console.log("False")}

>> False

var f;

>>undefined

if(f){console.log("True")} else{console.log("False")}

>>False

Typeof  => returns string

typeof a

"undefined"

var a;

undefined

typeof a

"undefined"

undefined

>>undefined

typeof undefined

>>"undefined"

null

>>null

typeof null

>>"object"

NULL

>> NULL is not defined

3<2   => False => 0

3>2  => True => 1

**== Vs ===**

<https://codeburst.io/javascript-double-equals-vs-triple-equals-61d4ce5a121a>

Strict equality === both type and value

Loose equality ==  type coercion and then checking

There are only six falsy values in JavaScript

false, 0, “”, null, undefined NaN

NaN === NaN // false

**& (bitwise and) &&(logical AND):**

<https://stackoverflow.com/questions/7310109/whats-the-difference-between-and-in-javascript>

Operator chaining:

It expects two arguments and returns:

* First term that evaluates to false
* Last term otherwise (if all are true-y)

Here are some examples:

0 && false 0 (both are false-y, but 0 is the first) true && false false (second one is false-y) true && true true (both are true-y) true && 20 20 (both are true-y)

This definition conforms to the definition of logical and from the field of mathematics.

“”/2  gives 0

**object constructor function**

<https://www.w3schools.com/js/js_object_constructors.asp>

**NAN**

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN#Confusing_special-case_behavior>

Number.isNAN()- use this

isNaN() - does a type coercion before checking for equality

The **Number.EPSILON** property represents the difference between 1 and the smallest floating point number greater than 1.

x = 0.2;  
y = 0.3;  
z = 0.1;  
equal = (Math.abs(x - y + z) < Number.EPSILON);

function isInteger(x) { return (x ^ 0) === x; }

function isInteger(x) { return Math.round(x) === x; }

function isInteger(x) { return (typeof x === 'number') && (x % 1 === 0); }

>>b=[1,2,3,4]

(4) [1, 2, 3, 4]

>>b.slice(-3)

(3) [2, 3, 4]

>>b.slice(-1)

[4]

>>b.slice(-2)

(2) [3, 4]

<https://www.smashingmagazine.com/2011/05/10-oddities-and-secrets-about-javascript/>

<https://stackoverflow.com/questions/500431/what-is-the-scope-of-variables-in-javascript>

**Higher Order function:**

<https://medium.com/humans-create-software/a-dirt-simple-introduction-to-higher-order-functions-in-javascript-b33bf9e19056>

Map, reduce, filter

Array.filter(func(individual ele){return value to decide for HOF})

Predict the output:

### 3. What would the output of following code ?

function getNumber(){  
 return;  
}  
  
var numb = getNumber();  
console.log(numb);

1. null
2. undefined
3. ""
4. 0

Answer: 2) undefined

### 2. What would the output of following code ?

function getNumber(){  
 return (2,4,5);  
}  
  
var numb = getNumber();  
console.log(numb);

1. 5
2. undefined
3. 2
4. (2,4,5)

Answer: 1) 5

**Call vs Apply vs Bind:**

Call - comma, Apply - Array (immediate function execution)

Bind - to execution function later

This variable is given some value while they are called

func.call(obj, param1,param2...)

As a result -=> the passed obj can be changed in that function

<https://stackoverflow.com/questions/15455009/javascript-call-apply-vs-bind>(3rd answer)

Javascript prototype

Scope

Prototype scope

Catch claued scope

**JS javatpoint:**

JavaScript is *an* ***object-based scripting language*** that is lightweight and cross-platform.

JavaScript is **not compiled but translated. The JavaScript Translator (embedded in browser)** is responsible to translate the JavaScript code.

JavaScript is used to create interactive websites. It is mainly used for:

* **Client-side validation**
* **Displaying dynamic contents**

## 3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

**<script** **type="text/javascript">**

alert("Hello Javatpoint");

**</script>**

There are two types of comments in JavaScript.

1. Single-line Comment // It is single line comment
2. Multi-line Comment  /\* \*/

Global variable can also declared by window.variable\_name

**Primitive data type:**

String, Number, Boolean, Undefined, Null

**Non-primitive (reference) data type:**

Object, Array, RegExp

|  |  |
| --- | --- |
| ?:) | Conditional Operator returns value based on the condition. It is like if-else. |
| , | Comma Operator allows multiple expressions to be evaluated as single statement. |
| delete | Delete Operator deletes a property from the object. |
| in | In Operator checks if object has the given property |
| instanceof | checks if the object is an instance of given type |
| new | creates an instance (object) |
| typeof | checks the type of object. |
| void | it discards the expression's return value. |
| yield | checks what is returned in a generator by the generator's iterator. |

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

Array methods (<https://www.javatpoint.com/javascript-array>)

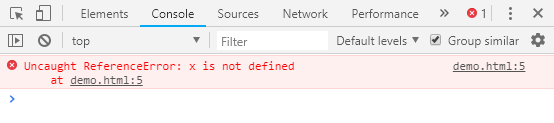
Javascript Strict Mode:

Being a scripting language, sometimes the JavaScript code displays the correct result even it has some errors. To overcome this problem we can use the JavaScript strict mode.

The JavaScript provides "use strict"; expression to enable the strict mode. If there is any silent error or mistake in the code, it throws an error.

1. <script>
2. x=10;
3. console.log(x);
4. </script>

If we add “use strict”



<https://stackoverflow.com/questions/824234/what-is-a-callback-function>

In JavaScript, almost "everything" is an object.

* Booleans can be objects (if defined with the **new** keyword)
* Numbers can be objects (if defined with the **new** keyword)
* Strings can be objects (if defined with the **new** keyword)
* Dates are always objects
* Maths are always objects
* Regular expressions are always objects
* Arrays are always objects
* Functions are always objects
* Objects are always objects

All JavaScript values, except primitives, are objects.

<https://www.w3schools.com/js/js_object_definition.asp>

Any function can be used as object constructore function

The object created using that function can have properties only if its properties are assigned to this.prop=val like this

If not yu cannot access

You can add prop=val to the the objects that are created using new keyword but cannot propertieds to it

It can be done by func.prototype.prop=val

A **web farm** scales across multiple *servers*. A **web garden** scales across multiple *processors*.

[LambdaTest](https://www.lambdatest.com/index-new-landing.html?utm_source=Medium&utm_medium=Blog&utm_campaign=Blogpost&utm_term=indexnewlanding) is a cross browser compatibility testing tool that can help you ensure that your website works fine for those other browsers too.

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.

The escape () function is responsible for coding a string so as to make the transfer of the information from one computer to the other, across a network.

The unescape() function was deprecated in JavaScript version 1.5. Use [decodeURI()](https://www.w3schools.com/jsref/jsref_decodeuri.asp) or[decodeURIComponent()](https://www.w3schools.com/jsref/jsref_decodeuricomponent.asp) instead.

Strict Mode adds certain compulsions to JavaScript.

**Closure is a locally declared variable related to a function which stays in memory when the function has returned.**

**30. What is an undefined value in JavaScript?**

**Undefined value means the**

* **Variable used in the code doesn’t exist**
* **Variable is not assigned to any value**
* **Property doesn’t exist**

**31. What are all the types of Pop up boxes available in JavaScript?**

* **Alert**
* **Confirm and**
* **Prompt**

**32. What is the use of Void(0)?**

**Void(0) is used to prevent the page from refreshing and parameter “zero” is passed while calling.**

**Void(0) is used to call another method without refreshing the page.**

[**https://www.quackit.com/javascript/tutorial/javascript\_void\_0.cfm**](https://www.quackit.com/javascript/tutorial/javascript_void_0.cfm)

**Inner html - a bad practice**

* **The entire innerHTML content is re-parsed and build into elements, therefore its much slower**
* **The innerHTML does not provide validation and therefore we can potentially insert valid and broken HTML in the document and break it**

**16. What is the difference between ViewState and SessionState?**

**‘ViewState’ is specific to a page in a session.**

**‘SessionState’ is specific to user specific data that can be accessed across all pages in the web application.**

**Netscape is the software company who developed JavaScript.**

**var newP = document.createElement("p");**

**var textNode = document.createTextNode(" This is a new text node");**

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/isNaN#Confusing\_special-case\_behavior**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN#Confusing_special-case_behavior)

**isNaN(NaN);       // true  
isNaN(undefined); // true  
isNaN({});        // true  
  
isNaN(true);      // false  
isNaN(null);      // false**

**isNaN(37);        // false  
  
// strings  
isNaN('37');      // false: "37" is converted to the number 37 which is not NaN  
isNaN('37.37');   // false: "37.37" is converted to the number 37.37 which is not NaN  
isNaN("37,5");    // true  
isNaN('123ABC');  // true:  parseInt("123ABC") is 123 but Number("123ABC") is NaN  
isNaN('');        // false: the empty string is converted to 0 which is not NaN  
isNaN(' ');       // false: a string with spaces is converted to 0 which is not NaN  
  
// dates  
isNaN(new Date());                // false  
isNaN(new Date().toString());     // true  
  
// This is a false positive and the reason why isNaN is not entirely reliable  
isNaN('blabla');**