**JAVASCRIPT QA**

**1) What is the difference between undefined and null in Javascript?**

**Ans:** At first blush these two things seem to be similar but are actually quite different. First of all, they have different types:

null is a value that can be assigned to a variable and means “no value”. However, variables that have been declared but not yet assigned a value are “undefined”. This is true not only of variables but also array keys and object members that do not exist, function parameters that have not been given and the result of calling a function that does not return a value.

### 2) What is the difference between == and ===?

**Ans:** The answer to this question is rather simple but an incorrect answer by a candidate should be considered a red flag. The answer is that == performs any necessary type conversions before doing the comparison whereas === does not. The [Javascript equality table](https://dorey.github.io/JavaScript-Equality-Table/) is rather infamous and it’s considered a good practice to always use === when comparing values for equality

### 3) What does the use strict; directive do?

**Ans:** The purpose of the use strict; directive is to ensure that code is executed in strict mode. So what is “strict mode”? [Strict mode](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode) helps developers avoid a handful of pitfalls in Javascript. Enabling strict mode means that the following will result in an error.

1. Not declaring a variable. This helps avoid mistyping a variable name as the mistyped variable will appear to be a new, undeclared variable.
2. Deleting a variable, i.e., delete x;
3. Writing to a read-only property (read-only properties are those defined via Object.defineProperty()).
4. Using variables with names like arguments or eval.
5. Use of the with statement.

An answer to this question should include one or more of the above. Bonus points can be given if the candidate says something about the scope of strict mode. In particular, if use strict; is at the top level, then it applies to the entire module but if it is found inside a function definition, it applies only to the code in the body of the function.

### 4) What are Promises used for?

**Ans:** Promises are used for asynchronous programming. They allow Javascript programs to be written in a non-blocking fashion just as using normal callbacks do, but without the mental overhead of that technique. People often refer to the [callback hell](http://callbackhell.com/) that often occurs when using callbacks to achieve asynchrony. Promises were created to alleviate this problem.

### 5) What is the ****DOM****?

**Ans:** **DOM** stands for **Document Object Model** is an interface (**API**) for HTML and XML documents. When the browser first reads (parses) our HTML document it creates a big object, a really big object based on the HTML document this is the **DOM**. It is a tree-like structure that is modeled from the HTML document. The **DOM** is used for interacting and modifying the **DOM structure** or specific Elements or Nodes.

### 6) What is ****Event Propagation****?

**Ans:** When an **event** occurs on a **DOM** element, that **event** does not entirely occur on that just one element. In the **Bubbling Phase**, the **event** bubbles up or it goes to its parent, to its grandparents, to its grandparent's parent until it reaches all the way to the window while in the **Capturing Phase** the event starts from the window down to the element that triggered the event or the [event.target](https://dev.to/macmacky/70-javascript-interview-questions-5gfi#12-what-is-eventtarget-).

### 7) What does the **!!** operator do?

**Ans:** The **Double NOT** operator or **!!** coerces the value on the right side into a boolean. basically it's a fancy way of converting a value into a boolean.

### 8) How to evaluate multiple expressions in one line?

**Ans:** We can use the, or comma operator to evaluate multiple expressions in one line. It evaluates from left-to-right and returns the value of the last item on the right or the last operand.

**9) What is Scope?**

Ans: **Scope** in JavaScript is the **area** where we have valid access to variables or functions. JavaScript has three types of Scopes. **Global Scope**, **Function Scope**, and **Block Scope(ES6)**.

* **Global Scope** - variables or functions declared in the global namespace are in the global scope and therefore is accessible everywhere in our code.
* **Function Scope** - variables,functions and parameters declared within a function are accessible inside that function but not outside of it.
* **Block Scope** - variables **(let,const)** declared within a block {} can only be access within it.

### 10) What are ****Closures****?

**Ans:** This is probably the hardest question of all these questions because **Closures** is a controversial topic. So I'm gonna explain it from what I understand.

**Closures** is simply the ability of a function at the time of declaration to remember the references of variables and parameters on its current scope, on its parent function scope, on its parent's parent function scope until it reaches the global scope with the help of **Scope Chain**. Basically it is the **Scope** created when the function was declared.

### 11) What's the value of this in JavaScript?

**Ans:** Basically, this refers to the value of the object that is currently executing or invoking the function. I say **currently** due to the reason that the value of **this** changes depending on the context on which we use it and where we use it.

### 12) What's the difference between Function.prototype.apply and Function.prototype.call?

**Ans:** The only difference between apply and call is how we pass the **arguments** in the function being called. In apply we pass the arguments as an **array** and in call we pass the arguments directly in the argument list.

### 13) Why are functions called ****First-class Objects****?

**Ans:** **Functions** in JavaScript are **First-class Objects** because they are treated as any other value in the language. They can be assigned to **variables**, they can be **properties of an object** which are called **methods**, they can be an **item in array**, they can be **passed as arguments to a function**, and they can be **returned as values of a function**. The only difference between a function and any other value in **JavaScript** is that **functions** can be invoked or called.

### 14) What are the new features in ****ES6**** or ****ECMAScript 2015****?

* Arrow Functions
* Classes
* Template Strings
* Enhanced Object literals
* Object Destructuring
* Promises
* **Generators**
* Modules
* Symbol
* **Proxies**
* Sets
* Default Function parameters
* Rest and Spread
* Block Scoping with let and const

### 15) What's the difference between var, let and const keywords?

**Ans:** Variables declared with var keyword are function scoped.  
What this means that variables can be accessed across that function even if we declare that variable inside a block.

Variables declared with let and const keyword are block scoped. What this means that variable can only be accessed on that block {} on where we declare it.

There is also a difference between let and const we can assign new values using let but we can't in const but const are mutable meaning.

### 16) What are ****Arrow functions****?

**Ans:** **Arrow Functions** are a new way of making functions in JavaScript. **Arrow Functions** takes a little time in making functions and has a cleaner syntax than a **function expression** because we omit the function keyword in making them.

### 17) What are ****Classes****?

**Ans:** **Classes** is the new way of writing constructor functions in **JavaScript**. It is syntactic sugar for using constructor functions, it still uses **prototypes** and **Prototype-Based Inheritance** under the hood.

**18) What are ES6 Modules?**

**Ans:** **Modules** lets us split our code base to multiple files for more maintainability and this lets us avoid putting all of our code in one big file (yucksss). Before ES6 has supported Modules there were two popular module systems that were used for Code Maintainability in **JavaScript**.

* CommonJS - **Nodejs**
* AMD (Asynchronous Module Definition) - **Browsers**

Basically, the sytanx for using modules are straightforward,   
import is used for *getting* functionality from another file or several functionalities or values while   
export is used for *exposing* functionality from a file or several functionalities or values.

### 19) What is NaN? and How to check if a value is NaN?

**Ans:** NaN means **"Not A Number"** is a value in **JavaScript** that is a result in converting or performing an operation to a number to non-number value hence results to NaN.

**20) What is AJAX?**

**Ans:** **AJAX** stands for **Asynchronous JavaScript and XML**. It is a group of related technologies used to display data asynchronously. What this means is that we can send data to the server and get data from the server without reloading the web page.

Technologies use for **AJAX**.

* **HTML** - web page structure
* **CSS** - the styling for the webpage
* **JavaScript** - the behaviour of the webpage and updates to the **DOM**
* **XMLHttpRequest API** - used to send and retrieve data from the server
* **PHP,Python,Nodejs** - Some Server-Side language

### 21) What are the ways of making objects in JavaScript?

Using **Object Literal**.

Using **Constructor Functions**.

Using **Object.create** method.

### 22) What's the difference between Object.seal and Object.freeze methods?

[↑](https://dev.to/macmacky/70-javascript-interview-questions-5gfi#the-questions) The difference between these two methods is that when we use the Object.freeze method to an object, that object's properties are immutable meaning we can't change or edit the values of those properties. While in the Object.seal method we can change those existing properties.

**23) What are the ways to deal with Asynchronous Code in JavasScript?**

* Callbacks
* Promises
* async/await
* Libraries like async.js, bluebird, q, co