



10 Advanced JavaScript Interview Questions

October 20, 2011

Whether you're looking to prepare for an interview for JavaScript role, you're an interviewer looking for inspiration or you just want to evaluate your JavaScript knowledge here are 10 advanced topics that a software engineer working with JavaScript should be able to answer. Bear in mind that answers here are kept minimal, there are lots of caveats not mentioned.

When would you use var in your declaration and when you wouldn't?

Always use var. Not using var for variable declaration will traverse scopes all the way up till the global scope. If variable with that name is not found it will declare it in the global scope. Therefore not using var implicitly declares variable in the global scope (which, let me remind you, is a bad practice).

```
(function() {  
    baz = 5;  
    var bar = 10;  
})();
```

```
console.log(baz); // outputs 5  
//console.log(bar); // error: bar is not defined
```

Try it: <http://jsfiddle.net/tnajdek/AKxn9/>

A common mistake is to not use var in loops (e.g. `for(i=0; i` when `i` has not been previously declared) which will pollute the global scope and in some cases might bear unexpected results.

What does the attribute defer/async do when added to the script tag?


The defer attribute will cause browser to execute script after the document has been parsed. This attribute was first implemented in Internet Explorer 4, then added to HTML 4 and more recently HTML 5 spec . You might not

have heard of it as it has not been supported by other browsers (Firefox support came in version 3.5 - Gecko 1.9.2). Async is another attribute that can affect how a script is loaded and executed, here is a quote from HTML 5 spec on how this is expected to work:

There are three possible modes that can be selected using these attributes. If the `async` attribute is present, then the script will be executed asynchronously, as soon as it is available. If the `async` attribute is not present but the `defer` attribute is present, then the script is executed when the page has finished parsing. If neither attribute is present, then the script is fetched and executed immediately, before the user agent continues parsing the page.

Note: A somewhat (but not exactly) similar defer behavior can be achieved by placing your script tags at the end of the body tag and that's what is considered to be modern 'best practice'

What is the difference between `==` and `===`? Which one would you use?

The equality (`==`) operator will compare for equality after doing necessary type casting, the identity operator (`===`) doesn't do any conversions. A good practice suggested by Douglas Crockford is to always use strict equality, couple of examples from Douglas' book [JavaScript: The Good Parts](#) 

```
'' == '0'           // false
0 == ''             // true
0 == '0'            // true

false == 'false'    // false
false == '0'        // true

false == undefined // false
false == null       // false
null == undefined   // true
```

How would you check if a variable is null/undefined?

```
//check if bar is null
bar === null
```

```
//check if bar is undefined
typeof bar === "undefined"
```

How do you check if a variable is an object


You can use `typeof` to determine if variable is an object, however bear in mind that null is actually an object! However null object is 'falsy' thus the following will work:

```
if(bar && typeof bar === "object") {
    console.log('bar is object and is not null');
}
```

Discuss scoping in JavaScript.

JavaScript has lexical scoping based on functions but not blocks. Therefore:

```
//global scope
(function() {
    //anonymous function scope
    var foo = 1;
    function bar() {
        //bar function scope
        var foo = 2;
    }
    bar();
    console.log(foo); //outputs 1
    if(true) {
        var foo = 3; //redeclares foo
    }
    console.log(foo); //outputs 3
})();
```

Try it: <http://jsfiddle.net/tnajdek/8y3XC/> . Note: from within function scope everything in above scope(s) is available (see closures below)

Explain hoisting in JavaScript.

As some might not be familiar with the term 'hoisting' yet have the relevant experience this question could be asked indirectly

In JavaScript function declarations (`function foo() {}`) and variable declarations (`var bar`) are 'hoisted' i.e. are silently moved to the very top of the scope. Consider the following code:

```
(function() {  
    console.log(bar); //returns 'undefined'  
    //console.log(baz) // error: baz is not defined  
    foo(); // outputs 'aloha' to the console  
  
    //function declaration AND its body is hoisted  
    function foo() {  
        console.log('aloha');  
    }  
    //variable declaration is hoisted but value assignment is not  
    var bar = 1;  
    baz = 2; //defines baz in global scope  
})();
```

See for yourself: <http://jsfiddle.net/tnajdek/FxDrj/> 

What are closures?

```
(function() {  
    function foo(x) {  
        var baz = 3;  
        return function (y) {  
            console.log(x + y + (++baz));  
        }  
    }  
  
    var moo = foo(2); // moo is now a closure.  
    moo(1); // 7  
    moo(1); // 8!  
})();
```

The inner function inside `foo` will close-over the variables of `foo` before leaving creating a closure.

Try it: <http://jsfiddle.net/tnajdek/Rj6mK/> 

Explain prototypal/differential inheritance

Conceptually this is very simple: A new object can inherit properties of an old object.

```
(function() {  
    var genericObject = {  
        bar : "Hello World",  
        get_bar : function() {  
            return this.bar;  
        }  
    };  
    var customObject = Object.create(genericObject);  
    customObject.bar = "Aloha folks!";  
    console.log(customObject.get_bar()); //outputs: "Alo  
    delete customObject.bar;  
    console.log(customObject.get_bar()); //fallbacks to  
})();
```

While JavaScript has always been a prototype-oriented language, tools to work with prototypes were somewhat missing. `Object.create` used in the code snippet above has been added in ECMAScript 5 and has not been supported prior to Firefox 4, Chrome 5, IE 9


What is Strict Mode in JavaScript

Strict Mode has been introduced as part of ECMAScript 5 and introduces new, restricted variant of JavaScript which has following aims:

- Throws errors for actions that are rather silly but previously didn't throw an error
- Throws errors for potentially unsafe actions
- Disables functions that are poorly thought out
- Potentially code in strict mode could run faster by eliminating mistakes that would make it difficult for JavaScript engines to perform optimizations

Strict mode can be enabled for the entire source file or on per function basis by adding a string literal "use strict" on top of the file/function i.e.

```
function foo(){  
  "use strict";  
  // ... your code ...  
}
```

For more detailed information about the strict mode consult [relevant article on MDN](#) 

Extra topics for discussion:

- What's your favorite browser, framework, JavaScript book
- How do you approach debugging in JavaScript
- What do you think of JSLint?
- Browser detection vs. feature sniffing

Anything I missed? Had a surprising interview question I haven't covered? Let me know in comments below!

If liked my JavaScript interview questions, you might also enjoy reading about [JavaScript quirks](#) and [JavaScript workflow automation](#).



Vijetak

June 18, 2012 at 17:34

Really nice article. I am deeply touched with the details of each sections.

[REPLY TO THIS COMMENT](#)



Quinn

November 5, 2014 at 01:35

Isn't 'deeply touched' a bit much? It's just an article on javascript.

[REPLY TO THIS COMMENT](#)





Joe

June 29, 2012 at 23:14

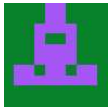
This made me poop in my pants.

[REPLY TO THIS COMMENT](#)

jane

February 19, 2013 at 04:08

This made me hose my trousers.

[REPLY TO THIS COMMENT](#)

Neo

November 28, 2014 at 12:27

po poori mone...

[REPLY TO THIS COMMENT](#)

Krishna Patni

July 13, 2013 at 19:35

simple and precise. the article content is in sync with the title.

[REPLY TO THIS COMMENT](#)

Ironic Penguin

July 23, 2013 at 23:36

How do you prevent spambot comments using JS?

[REPLY TO THIS COMMENT](#)

Sanyam Agrawal

August 4, 2013 at 21:52

Very important Questions to make your basics right and know where you exactly are with the knowledge of Javascript .

[REPLY TO THIS COMMENT](#)

Nitin Kulkarni

August 5, 2013 at 15:12

Thanks a lot..helped me preparing for interview....some of questions given here were very closed to what i was asked :-)

[REPLY TO THIS COMMENT](#)

sAu

September 13, 2013 at 13:52

different than usual.. nice content.

[REPLY TO THIS COMMENT](#)

pb

September 30, 2013 at 05:37

Excellent stuff... thanks!

[REPLY TO THIS COMMENT](#)

LN

December 12, 2013 at 14:26

For the question on closures, how do you know what value is "y"? It's not defined anywhere :(

```
(function() {  
    function foo(x) {  
        var baz = 3;  
        return function (y) {  
            console.log(x + y + (++baz));  
        }  
    }  
    var moo = foo(2); // moo is now a closure.  
    moo(1); // 7  
    moo(1); // 8!  
})();
```

[REPLY TO THIS COMMENT](#)

Tom Najdek

December 13, 2013 at 01:32

Function `foo(x)` will return a reference to an anonymous function which takes one argument and alerts result of a mathematical expression. This reference to a function is a closure, or in other words, a function together with referencing environment. When I call `foo(2)`, what I get back is a reference to function that takes one argument (`y`) together with referencing environment (`baz = 3` defined on top of the encompassing scope and `x = 2` defined as an argument to a function). I then assign the above reference to a variable `moo`. Since `moo` is a closure (function + referencing environment) I can call it! I'm calling it with a single argument

- an integer 1 hence inside that closure's scope $y = 1$ (comes from an argument) and $x = 2$ and (initially) $baz = 3$ (coming from the referencing environment). This is why first call to `moo()` will alert 7 ($x = 2$, $++baz = 4$, $y = 1$) and second call alerts 8 ($x = 2$, $++baz = 5$, $y=1$).

[REPLY TO THIS COMMENT](#)

Rahul

December 20, 2013 at 21:11

Please explain it in detail. i don't understand the flow here.

[REPLY TO THIS COMMENT](#)

Hadar Shkolnik

January 4, 2014 at 22:34

Thanks a lot Tom! :) Was great reading your explanations, and it's also a very good example. I think it may be good to refer to this comment, or perhaps add this to the article (or at least part of it).
Cheerz, Hadar

[REPLY TO THIS COMMENT](#)

Negin

February 13, 2014 at 03:33

Thanks you! It was tricky!

[REPLY TO THIS COMMENT](#)

anbu

December 20, 2013 at 21:42

Hi Sir, I have one clarification in above answer mentioned by Tom Najdek that i think `x=2 //foo(2)`. please correct me Thank you

[REPLY TO THIS COMMENT](#)

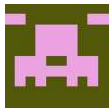


Tom Najdek

December 21, 2013 at 20:02

Yes, silly typo, thanks for pointing this out! I've corrected my comment above.

[REPLY TO THIS COMMENT](#)



samiyuru

May 9, 2014 at 07:51

Great article

[REPLY TO THIS COMMENT](#)



Ezz

May 21, 2014 at 12:20

One of the best articles, I got an advanced knowledge about JavaScript, it is like a guid for who want to learn JavaScript in the right way :) Thank you so much

[REPLY TO THIS COMMENT](#)



Nikita

May 25, 2014 at 22:42



Simply Great!!

[REPLY TO THIS COMMENT](#)

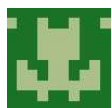


Gaurav jha

June 3, 2014 at 12:56

Thanks buddy for publishing this useful things

[REPLY TO THIS COMMENT](#)

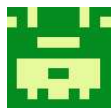


siddharth shivhare

September 3, 2014 at 23:10

Hi Tom, You have very clearly explained these important topics which are very often asked in interviews nowadays. Thanks a lot for clearing my doubts. One should go through these topics.

[REPLY TO THIS COMMENT](#)



Tung Nguyen Thanh

April 9, 2015 at 08:43

Great questions. Thanks for sharing.

[REPLY TO THIS COMMENT](#)





Atin Verma

August 19, 2015 at 21:26

Great content, thanks !!

[REPLY TO THIS COMMENT](#)

[ADD COMMENT](#)

[About Me](#) | [My Projects](#)

© 2015 Tom Najdek | CC BY-SA