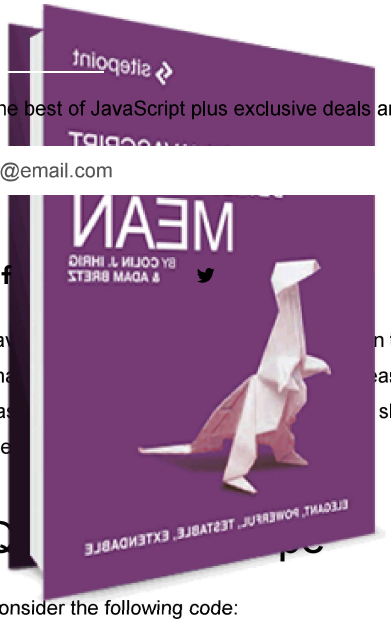
5 Typical JavaScript Interview Exercises

Aurelio De Rosa(http://www.sitepoint.com/author/aderosa/) September 18, 2014



Ja n the IT world. If this is the role that best expresses your knowledge, you have a lot of opportunities to cha ease your salary. But before you are hired by a company, you have to demonstrate your skills in order to pas show you 5 typical questions asked for a front end job to test the JavaScript skills of the candidate and the

Q

Consider the following code:


```
(function() {
   var a = b = 5;
})();

console.log(b);
```

What will be printed on the console?

Answer

The code above prints `5`.

The trick of this question is that in the IIFE there are two assignments but the variable `a` is declared using the keyword `var`. What this means is that `a` is a local variable of the function. On the contrary, `b` is assigned to the global scope.

The other trick of this question is that it doesn't use *strict mode* (`'use strict';`) inside the function. If strict mode (http://cjihrig.com/blog/javascripts-strict-mode-and-why-you-should-use-it/) was enabled, the code would raise the error "Uncaught ReferenceError: b is not defined". Remember that strict mode requires you to explicitly reference to the global scope if this was the intended behavior. So, you should write:

```
(function() {
   'use strict';
   var a = window.b = 5;
})();

console.log(b);
```

Question 2: Create "native" methods

Define a `repeatify` function on the `String` object. The function accepts an integer that specifies how many times the string has to be repeated. The function returns the string repeated the number of times specified. For example:

```
console.log('hello'.repeatify(3));
```

Should print `hellohellohello`.

Answer

A possible implementation is shown below:

```
String.prototype.repeatify = String.prototype.repeatify || function(times) {
   var str = '';

   for (var i = 0; i < times; i++) {
      str += this;
   }

   return str;
};
```

The question tests the knowledge of the developer about inheritance in JavaScript and the `prototype` property. It also verifies that the developer is able to extend native data type functionalities (although this should not be done).

Another important point here is to demonstrate that you are aware about how to not override possible already defined functions. This is done by testing that the function didn't exist before defining your own:

```
String.prototype.repeatify = String.prototype.repeatify || function(times) {/* code here */};
```

This technique is particularly useful when you are asked to shim a JavaScript function.

Question 3: Hoisting

What's the result of executing this code and why.

```
function test() {
   console.log(a);
   console.log(foo());

   var a = 1;
   function foo() {
      return 2;
   }
}

test();
```

Answer

The result of this code is `undefined` and `2`.

The reason is that both variables and functions are hoisted (http://www.sitepoint.com/back-to-basics-javascript-hoisting/) (moved at the top of the function) but variables don't retain any assigned value. So, at the time the variable `a` is printed, it exists in the function (it's declared) but it's still `undefined`. Stated in other words, the code above is equivalent to the following:

```
function test() {
   var a;
   function foo() {
      return 2;
   }

   console.log(a);
```

```
    console.log(foo());

    a = 1;
}

test();
```

Question 4: How `this` works in JavaScript

What is the result of the following code? Explain your answer.

```
var fullname = 'John Doe';
var obj = {
   fullname: 'Colin Ihrig',
   prop: {
      fullname: 'Aurelio De Rosa',
      getFullname: function() {
         return this.fullname;
      }
   }
};

console.log(obj.prop.getFullname());

var test = obj.prop.getFullname;

console.log(test());
```

Answer

The code prints `Aurelio De Rosa` and `John Doe`. The reason is that the context of a function, what is referred with the `this` keyword, in JavaScript depends on how a function is invoked, not how it's defined.

In the first `console.log()` call, `getFullname()` is invoked as a function of the `obj.prop` object. So, the context refers to the latter and the function returns the `fullname` property of this object. On the contrary, when `getFullname()` is assigned to the `test` variable, the context refers to the global object (`window`). This happens because `test` is implicitly set as a property of the global object. For this reason, the function returns the value of a property called `fullname` of `window`, which in this case is the one the code set in the first line of the snippet.

Question 5: `call()` and `apply()`

Fix the previous question's issue so that the last `console.log()` prints `Aurelio De Rosa`.

Answer

The issue can be fixed by forcing the context of the function using either the `call()` or the `apply()` function. If you don't know them and their difference, I suggest you to read the article What's the difference between function.call and function.apply? (http://www.sitepoint.com/whats-the-difference-between-function-call-and-function-apply/). In the code below I'll use `call()` but in this case `apply()` would produce the same result:

```
console.log(test.call(obj.prop));
```

Conclusion

In this article we've discussed five typical questions that are asked at interviews to test a JavaScript developer. The actual questions may differ from interview to interview but the concepts and the topics covered are usually pretty similar. I hope you had fun testing your knowledge. In case you didn't know some of all of the answers, don't worry: there is nothing that studying and experience can't fix.

If you have been asked some other interesting questions at interviews, don't hesitate to share them with us. It'll help a lot of developers.

Aurelio De Rosa (http://www.sitepoint.com/author/aderosa/)

(https://twitter.com/AurelioDeRosa) (https://plus.google.com/+AurelioDeRosa/) (aurelioderosa/en) (https://github.com/AurelioDeRosa)

I'm a (full-stack) web and app developer with more than 5 years' experience programming for the web using HTML, CSS, Sass, JavaScript, and PHP. I'm an expert of JavaScript and HTML5 APIs but my interests include web security, accessibility, performance, and SEO. I'm also a regular writer for several networks, speaker, and author of the books *jQuery in Action, third edition* (http://www.manning.com/derosa/) and *Instant jQuery Selectors* (http://www.packtpub.com/learn-how-to-use-jquery-selectors-effectively/book).

71 Comments SitePoint Login

Recommend 3 Share Sort by Best

Join the discussion...

Vincent Voyer • a year ago

Another great exercice is this:

```
var items = document.querySelectorAll('.items');
for (var i = 0, max = items.length; i < max ; i ++) {
  var element = items[i];
  element.addEventListener('click', function() {
    console.log('you clicked on element number ' + i);
  });
}

// Given 10 items, what's the output on the console if I click on the third item?
// Why?
// How to fix this behavior?
```

What do you think?

20 • Reply • Share ›

welbornio ➚ Vincent Voyer • 2 days ago

That's a great one.

• Reply • Share ›

Aurelio De Rosa ➚ Vincent Voyer • a year ago

I think I'll use it in an upcoming sequel.

• Reply • Share ›

Aurelio De Rosa ➚ Vincent Voyer • a year ago

That's a good exercise too.

• Reply • Share ›

Vincent Voyer ➚ Aurelio De Rosa • a year ago

Thanks for reply.

I found out that nowhere in your article your were talking about closures, nor in question 4.

Question 4 is about `this` and context. While my exercice is about scope (problem) and closures (solution).

• Reply • Share ›

dotDeeka ➚ Vincent Voyer • 5 months ago

Isnt the easiest way to just assign i to a local variable in your callback

super readable very simple.

• Reply • Share ›

Aamir Afridi ➚ Vincent Voyer • a year ago

I understand the problem but do you mind explaining the solution.

• Reply • Share ›

Vincent Voyer ➚ Aamir Afridi • a year ago

So the solution to the problem:

```
var items = document.querySelectorAll('.items');
for (var i = 0, max = items.length; i < max ; i ++) {
  var element = items[i];
```

```
  element.addEventListener('click', createLog(i));
}



function createLog(i) {
  return function logLocal() {
    console.log('you clicked on element number' + i);
  }
}
```

By calling a function (createLog) that will return a function (logLocal), you are able to reference `i` at the value it had when the loop iterate on it.

Basically you are creating a new scope where `i` has the value you expect. Not just the last value of the loop.

As in JavaScript number and strings are copied by value, you get a new var.

If `i` was an object, you would get a reference to the `i`at the time of the iteration, which is what you want.

This exercise test the ability of the candidate to understand that a lot of the power of JavaScript comes from the fact that you can have functions creating and returning new functions. Coupled with how scope works, you can have great solutions to common problems.

3 ^ | v • Reply • Share ›

Aaron ➦ Vincent Voyer • a year ago

@Vincent, this is an unnecessary detail and the problem was scope. I think mine is cool solution than yours. Check it below...

```
var items = document.querySelectorAll('.items');
for (var i = 0, max = items.length; i < max; i++) {
    (function (n) {
        var element = items[i];
        element.addEventListener('click', function () {
            console.log('you clicked on element number ' + n);
        });
    }(i));
}
```

5 ^ | v • Reply • Share ›

Vincent Voyer ➦ Aaron • 9 months ago

It's a lot less readable than externalizing your closure to a named function.

I used to write code like this untill I really understood functions in JavaScript

It does not allows you to transform your closure in a module.

2 ^ | v • Reply • Share ›

Tom ➦ Vincent Voyer • 7 months ago

I don't think it's less readable, and we are not talking about reuse the code, in fact to me is more intuitive since demonstrate clousures better ;)

^ | v • Reply • Share ›

Ian ➦ Aaron • 9 months ago

It will work if capturing every <n> in the loop..

var items = document.querySelectorAll('.items');
for (var i = 0, max = items.length; i < max; i++) {
var element = items[i];
element.addEventListener('click', (function(n){
return function () {
console.log('you clicked on element number ' + n);
};
}(n)));
}

1 ^ | v • Reply • Share ›

Long Huynh ➦ Aaron • 3 months ago

why not just use .bind()?

^ | v • Reply • Share ›

WiseFundManager ➦ Long Huynh • 13 days ago

I like this approach better:

'use strict';

var items = document.querySelectorAll('.item');

for (var i = 0, max = items.length; i < max; i++) {

var element = items[i];

this.response = function (n) {

console.log('you clicked on element number ' + n);

};

element.addEventListener('click', this.response.bind(this, i));

element.addEventListener(click , this.response.bind(this, i));

}

• Reply • Share ›

Vadim Gutman ➔ Long Huynh • 16 days ago

Common guys lets go ECMA 6
why not just use 'use strict';

and add a 'let _i = i;' in the loop

problem solved

• Reply • Share ›

Jonatan Brown • a year ago

Wouldn't a more elegant solution to the last problem be to use .bind(obj.prop) on the getFullName function?

3 • Reply • Share ›

Jorge Bucaran ➔ Jonatan Brown • 7 months ago

Certainly. Or just **this**.

```
var fullname = 'John Doe';
var obj = {
  fullname: 'Colin Ihrig',
  prop: {
    fullname: 'Aurelio De Rosa',
    getFullname: function() {
      return this.fullname;
    }.bind(this)
  }
};
```

• Reply • Share ›

Nyasro • a year ago

Very nice :)
Today I learned new thing about Javascript `this` keyword.

I liked "`this` keyword, in JavaScript depends on how a function is invoked, not how it's defined".. Nice one :)

3 • Reply • Share ›

Tom Hicks ➔ Nyasro • a year ago

I've done a similar write up of questions I use when testing for javascript ability here: http://tomhicks.github.io/code...

I don't cover hoisting explicitly which you do well, but I think I cover the nature of prototypes in more depth.

I would be interested to hear what kinds of things you do for testing higher level concepts like api design and clean coding as I find those things harder to test for without having to spend a long time doing it. I like that these kinds of tests are quick and pretty accurate.

3 • Reply • Share ›

Gleb Bahmutov • a year ago

I prefer to see the candidate first formulate the answer, then write unit tests, then write code that makes the tests pass. Then iterate. This could apply to these questions, but I prefer functional-lite javascript questions because they map naturally to english and to unit tests, see http://bahmutov.calepin.co/fun...

2 • Reply • Share ›

Frederik Krautwald • a year ago

Die, JavaScript. Die!

4 • Reply • Share ›

Aurelio De Rosa ➔ Frederik Krautwald • a year ago

Why such hate?

1 • Reply • Share ›

Frederik Krautwald ➔ Aurelio De Rosa • a year ago

Oh, did I mention...

[] + {}; // [object Object]

...but

{} + []; // 0

Die, JS!

2 • Reply • Share ›

Maksim Vi. ➔ Frederik Krautwald • 9 months ago

Why would you do this on a first place?

1 • Reply • Share ›

Frederik Krautwald ➔ Maksim Vi. • 9 months ago

To know the quirks of the language.

• Reply • Share ›

Frederik Krautwald ➔ Aurelio De Rosa • a year ago

(function foo() {

```
//some stuff
}());

(function foo() {
//some stuff
})();

var foo = function () {
//some stuff
}();

var foo = (function () {
//some stuff
}());

var foo = function () {
// Works!
}();
```

see more

1 • Reply • Share ›

Frederik Krautwald ➚ Frederik Krautwald • a year ago

Try this in your console:

```
Array(16).join("wat" - 1) + " Batman!";
```

1 • Reply • Share ›

Aurelio De Rosa ➚ Frederik Krautwald • a year ago

The first example you reported are not the same thing. The syntax is different because you are talking about different concepts: IIFE vs function declarations vs function expressions.

The others are true in every decent programming language: basic types are compared based on their value, objects based on the reference.

The last examples should never be written, so there is no point in discussing them.

• Reply • Share ›

Frederik Krautwald ➚ Aurelio De Rosa • a year ago

"The last examples should never be written, so there is no point in discussing them."

- HAHAHAHA... they can be written. They are written. And the language allows them to be written. Yet, you won't discuss them. Absolutely WATMAN!

• Reply • Share ›

Aurelio De Rosa ➚ Frederik Krautwald • a year ago

I challenge you to find an expression like those you mentioned in any serious project like jQuery, Modernizr, or similar. If you want to just blame JavaScript, then you know what Bjarne Stroustrup said?

"There are only two kinds of languages: the ones people complain about and the ones nobody uses."

9 • Reply • Share ›

Oscar M ➚ Aurelio De Rosa • 10 months ago

I agree, Frederik if something like this was presented in a project I would assume it came from an intern...

There are these thinks called conventions and guidelines which developers should follow and when they don't then unexpected behaviours can occur (this applies to any language and any framework). The point is that yes there are flaws but you avoid them by following the right practices and by doing things as they where intended to be used.

Also there is really no point in hating programming languages, its just means to an end and currently there are no alternatives (at least not practical one) that you can use so I'm pretty sure it will not die any time soon. I'm aware that some alternatives do exist (like Google's Dart) but you cannot use them yet (support issue, compatibility issue, etc.).

Nevertheless Aurelio I really like your questions and those 5 follow ups, they are straightforward and to a point. Please consider making some more because they helped me a lot, also note that I used your questions and some of these web tests (HTML, CSS and JS) for a fast preparation and it turned out great.

• Reply • Share ›

Frederik Krautwald ➚ Aurelio De Rosa • a year ago

Come on, take it easy JS lover. It is an awesome but terrible language with many design flaws. How about 0.1 + 0.2?

• Reply • Share ›

Aurelio De Rosa ➚ Frederik Krautwald • a year ago

Do you know that you have just demonstrated your poor knowledge of programming languages to the world, right? Do you know that similar issues affect ALL programming languages even more "cool" one like Java? You'd better read this article:

http://docs.oracle.com/cd/E199...

Example in PHP:

```
// prints 7, not 8
echo (int) ((0.1 + 0.7) * 10);
```

5 • Reply • Share ›

Frederik Krautwald ➚ Aurelio De Rosa • a year ago

Jaysus!!! My "poor knowledge of programming languages"!!! Are you for feckin' real? I mention some design flaws and curiosities

(even funny ones) in JS, and off you wander into straw man's land, trying to slander me with an ad hominem attack.

And so what that both Java, PHP and whatever other languages have their design flaws, too? How is that going to make JavaScript's shortcomings disappear?

I think you should go and chill out a bit before you dig yourself any deeper into the world of personal attacks. Peace out, man.

3 ^ | v • Reply • Share ›

Marcello La Rocca ➚ Frederik Krautwald • a year ago

0.1 + 0.2 is not a JavaScript flaw, it is a flaw of the binary floating point notation, basically due (really in a nutshell) to the fact that it uses powers of two and can not accurately represent decimal fractions, while (shoot!) we use right a decimal system, instead.

Assuming you are not in the mood to read, you can watch this: http://youtu.be/3WgVHE5Augc?t=...
(Still, you'll need a tiny bit of patience and watch at least 4 or 5 minutes - but, come on, hang in there: who knows, you might even learn something...)

So... try again, maybe you'll be in luck next time ;-)
After all, there are many **real** flaws in JS, as there are in Java, C, Haskell (yes, even Haskell!) and basically any language.

4 ^ | v • Reply • Share ›

Frederik Krautwald ➚ Marcello La Rocca • 4 months ago

I really love Crockford. He even mentions the "wat embarrassment" to the JavaScript community. But how lucky it is that we have ClojureScript today.

^ | v • Reply • Share ›

WiseFundManager ➚ Frederik Krautwald • 13 days ago

You really need to stop hating, mister hater.

^ | v • Reply • Share ›

Frederik Krautwald ➚ WiseFundManager • 12 days ago

Thanks for the labeling roses, how mature of you.

^ | v • Reply • Share ›

vomitory • a year ago

Please explain, question **#4**, line 14. Why doesn't it just crash? I mean, how does the same property getFullName exists in two forms, function and variable? It's not a quantum mechanics! Ok, let's say, they co-exist. But how the hell, variable getFullName points to global fullName? There's no direct reference anywhere! Sorry for exclamation - there's not other way to ask that...

1 ^ | v • Reply • Share ›

scottseeker ➚ vomitory • 9 months ago

Because the context in this case is the global context, so 'this' refers to the fullName established at the global level.

^ | v • Reply • Share ›

Peter Clark • 21 hours ago

Thanks for sharing top JavaScript interview questions, and I have recently find the collection of frequently asked JavaScript interview question and answer for beginner and advanced label. these questions and answers will help you prepare for you interviews in different companies.
http://goo.gl/S7wdGg

^ | v • Reply • Share ›

Al • 23 days ago

All these questions were asked to me as-is in my yesterday's front end UI dev role.

^ | v • Reply • Share ›

Billy J Figueroa • a month ago

Good Article, these are the types of things you learn in books like 'Secrets of the JavaScript Ninja".

Thanks for the article. **@Vincent Voyer** good stuff too. I have seen the IIFE used and passed i the way **@Aaron** is doing it. I don't think it over complicates things. I am not a big fan of functions inside of functions lol that confuses me more than seeing a IIFE being passed i as a param

^ | v • Reply • Share ›

Imstillreallybored • 2 months ago

I was confused by question 4 i thought the answer you put down was wrong and it is. The second output is not "John Doe" its undefined.

http://jsfiddle.net/t4krszpb/

^ | v • Reply • Share ›

James Brown • 2 months ago

Just FYI. if you are interested in javascript quizzes, i have found a lot them here: http://quizful.com/app

^ | v • Reply • Share ›

Gaurang • 3 months ago

Hi Aurelio De Rosa,

I had a doubt in question # 4.

I kept on getting 'undefined' for line 16. I used the exact same code. Now my understanding is this is because getFullname looks for fullname value in scope of 'test' object but can't find it. I got the output that you talk about after i removed 'var' keywork to make fullname on line # 1 global. Is my understanding correct or am I missing something?

Thanks
Gaurang

^ | v • Reply • Share ›

Michael Burch • 5 months ago

Michael Burch • 5 months ago

Actually, when I run the Question 4 code in JSFiddle, it returns Aurelio De Rosa and undefined.

• Reply • Share ›

jer • 7 months ago

Actually the answer to #1 is incorrect. Since the console.log statement is outside the scope of the anonymous function it throws a ReferenceError: a is not defined. http://jsfiddle.net/Ls51jjmm/9...

To fix this, include the console.log inside the scope of the function like this:

```
(function () {
'use strict';
var a = Window.b = 5;
console.log(a);
})();
```

• Reply • Share ›

Jingqi Xie • 10 months ago

A lot easier than what I have faced!

• Reply • Share ›

Eric Njanga • a year ago

Great article. Thank you.

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

Next Article

On Our Radar This Week: Black Friday Freebies and Vanilla JS ➔ (http://www.sitepoint.com/radar-week-black-friday-freebies-vanilla-js/?utm_source=sitepoint&utm_medium=nextpost&utm_term=javascript)

About