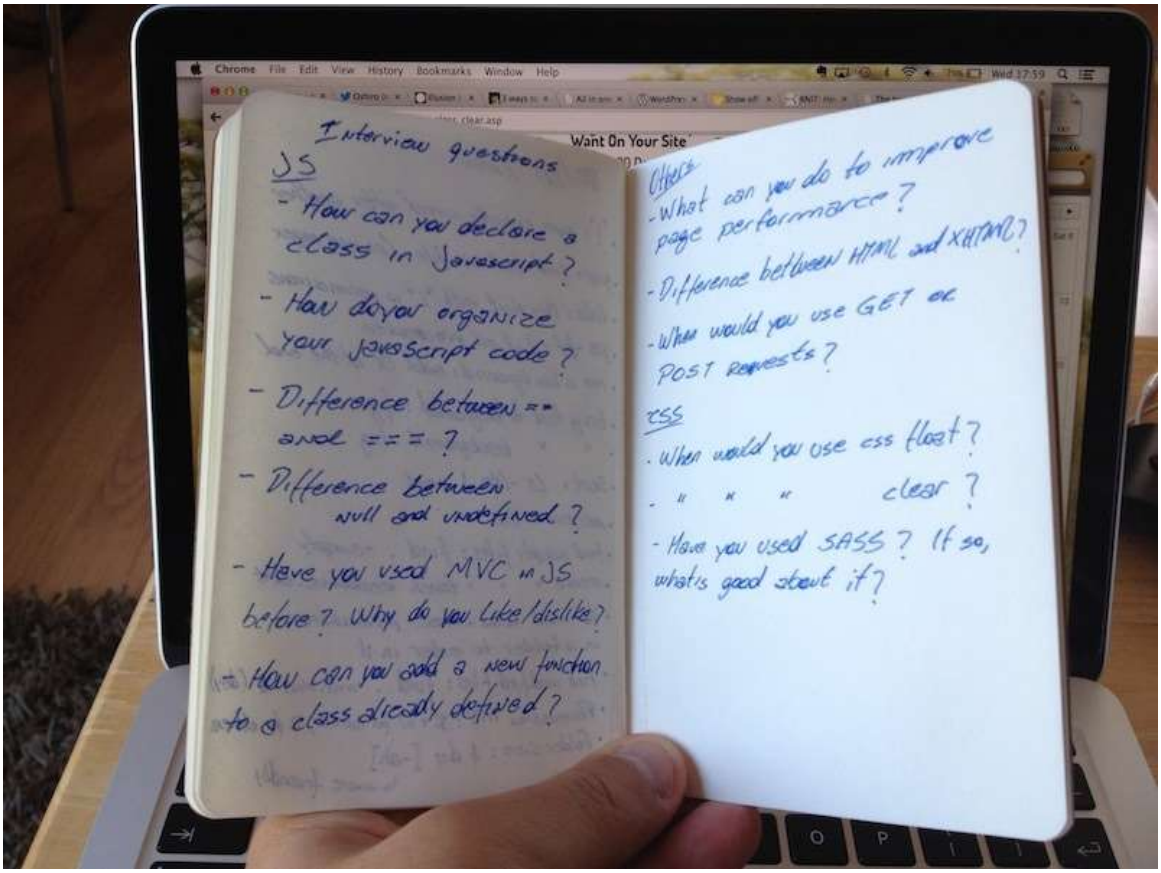
FrontEnd Journal

Most Common Technical Interview Question for FrontEnd Developers

16 Jul 2013

on javascript | backbone.js | interview | questions and answers | web developer interview questions



This picture above is from my notes taken from technical interviews I've had lately. If you're a frontend developer this post may be of your interest.

Even senior javascript developers can be trapped by some of the questions asked by interviewees. So even if you can code comfortably in your computer with the help of google, there are some concepts that are always good to have in mind when you're about to face a technical interview.

The idea here is not to go through the questions and look for the answer. The idea for this post is **learning** instead, because doing that you'll be prepared for any variation of those questions or if you're not going to face an interview, just to review what would be your answer and leave a comment.

For every question there'll be a brief explanation and some examples.

Let's start with the Javascript questions

1. How can you declare a class in Javascript?

In javascript there's no classes like in Java, what we actually call a class is in reality a function simulating a class behaviour. For being so flexible, there are many ways to create a class in javascript, below you'll find 3 ways of doing that.

- Class using function as a constructor:

```
function Person(name) {
    this.name = name;
}
// Creating an object
var person = new Person("Rafael");
person.name; // "Rafael"
```

It's very important to notice that you have to use the keyword `new` when creating a `new` instance of that class otherwise you will have logical problems regarding the `this` will reference `window` object.

- Class Literal notation:

```
var person = {
    name: "",
    setName: function(name) {
        this.name = name;
    }
}
person.setName("Rafael");
person.name; // "Rafael"
```

In this example we don't use a function to define our class, we are creating a singleton object person with one attribute and one method. You can use that object straightaway, no instantiation

in this case.
That notation is useful when you don't need to create instances of that class or you'll use it just once in your application.

- Singleton through a function:

```
var person = new function() {
    this.setName = function(name) {
        this.name = name;
    }
    this.sayHi = function() {
        return "Hi, my name is " + this.name;
    }
}
person.setName("Rafael");
alert(person.sayHi()); // Hi, my name is Rafael
```

As you can see in the code snippet above, we have a function like the first example and besides we also have the `new` keyword before the function declaration. It means that we are creating one instance of that class at the same time we are declaring it.

2. How would you organize your Javascript code?

The following pattern is the one that I personally prefer and is called 'module pattern', where we separate our javascript into logical modules, or namespaces. What you'll see below is an example of how I would separate my user module.

```
// Declaring my main namespace
var myapplication = myapplication || {};

// Declaring modules usermodule
myapplication.usermodule = (function() {
    // createMessage: only accessible inside this module
    var createMessage = function(message) {
        return "Hello! " + message;
    }

    return {
        // sayHello is a public method
        sayHello: function(message) {
            return createMessage(message);
        }
    }
})();

// Declaring another module
myapplication.adminmodule = (function(){
    // your code here
})()
// This is how we call sayHello
myapplication.usermodule.sayHello("This is my module");
```

Some explanation on the code above

Take a look at the previous code and notice how I create my module using the notation below. It makes the function to be

executed immediately because of the parenthesis at the end of the command. The result of the execution will be an object which will be set to my variable myapplication.usermodule.

```
...
myapplication.usermodule = (function() {
    // code to be executed immediately
})();
```

So applying this pattern to your code you may have multiple modules and you have the control over what you want to make public and what to keep private. Besides your code will be more organized therefore easy to maintain.

3. Difference between == and ===.

This is pretty simple but at the same time some people never came across a triple equals or never wondered what's the difference.
Double equals `==` is used to compare the value of two operands:

```
"2" == 2; // true
 2  == 2; // true
```

Triple equals `===` is used to compare the value AND type of two operands:

```
"2" === 2; // false
 2  === 2; // true
```

4. Difference between null and undefined

This can be tricky and the best way to keep in your head is to memorise because if you try to relate javascript null to other languages, it will get more confusing.
In javascript, `null` is an object with no value and `undefined` is a type.

```
typeof null; // "object"
typeof undefined; // "undefined"
```

```
var a;
var b = null;
a == b; // "true" because their values are the same
a === b; // "false". they have different types
```

5. Have you already used MVC before? What you like/dislike about it?

As the UI gets more and more complex we need some good ways to keep it more and more maintainable and reusable, and Some MVC frameworks for javascript have been widely adopted lately and it's a good plus if you have already used before and knows what's the benefits of them. The most famous MVC

frameworks are backbone.js and angular.js, it's hard to not hear about them.

There are many advantages in using these frameworks, I can point out some of them:

- **Organization:** Forces your webapp to follow a well structured pattern;
- **Maintainable:** With organization comes an easy to maintain code;
- **UI Binding:** Some frameworks allow you to do that. So everytime your model changes, the view reflects it and vice-versa;
- **Decoupled client:** MVC frameworks like backbone.js *incentivise* you to use REST API's though their urlRoot attribute in their Models;
- **Reusable components:** Create reusable visual components;
- **Single-page apps:** Build single-page apps with Ajax requests;
- **Friendly URL's:** Native support for client-side url mapping;

6. How can you add a method to a class already defined?

You can add a new method to a javascript class using prototype:

```
function Person(name) {
    this.name = name;
}
Person.prototype.walk = function() {
    console.debug(this.name + " is walking.");
}
// Calling the new method
var person = new Person("Rafael");
person.walk(); // "Rafael is walking."
```

It's worth mentioning that adding methods via prototype is the most inexpensive way in terms of performance since the method is tied to the prototype of the class. It means, for every new instance of class Person, you will have access to the prototype's walk() method. Now, if you declare walk() method inside the Person class, you will end up recreating the method for every new instance of Person.

CSS Questions

1. When would you use CSS float?

Float is used when you want to make an element of your page (usually an image) be pushed to the right or left and make other elements wrap around it.

2. When would you use CSS clear?

When you want an element on the left or right of the floating element not to wrap around it, you can use clear.

3. Have you used Sass? What's good about it?

Every web project starts with everything neat, all CSS is organized in blocks or different CSS files and you know where everything is, right?
Right, until your project gets bigger, deadlines get tight, more developers come on board and someday you notice a strange behaviour in some elements of the page. When you inspect their styles you spot lots of css overrides coming from everywhere. This is the moment you realise how messy CSS can be.

Sass is the modern way of doing CSS and can save many lines of code in your stylesheets. This is possible because Sass works with variables, nested syntax and mathematical operations.
In my opinion one of the nicest features of sass is the possibility to write a selector just once and put all styles for that inside it. Do you need a more specific selector under an existing one? Just nest the specifics into the generic one.

Check out the example below taken from their official website. It's awesome how it can "neatify" your code.

```
/* .sass */
table.hl
  margin: 2em 0
```

```
  td.ln
    text-align: right

li
  font:
    family: serif
    weight: bold
    size: 1.2em
```

```
/* .css */
table.hl {
  margin: 2em 0;
}
table.hl td.ln {
  text-align: right;
}

li {
  font-family: serif;
  font-weight: bold;
  font-size: 1.2em;
}
```

Other questions

1. What can you do to improve page performance?

In a nutshell page performance is widely understood as the page load time from the users' perspective, so below are some steps that might improve a page's performance.

- **Use sprite images** whenever possible, try to group small images commonly used in a single file to be requested just once. See how Google uses sprites in Google Maps to make one request instead of one for each small image.

- **Javascripts should be at the bottom of the page**, instead of in the head as we use to see out there;

- **Ensure parallel requests** of your JS and CSS files. In order to force the browser to do that, you can optimize the order you include resources in your page. This item can generate its own blog post or even a book so I prefer to suggest you a really good reading about it. Check this out, it's the google's best practices on page speed load.

- **Compress images** whenever possible, it makes a difference;

- **Browser Caching** is also very import to be set for static resources like JS and CSS files, images, PDFs and HTML. Caching is set in the HTTP header by informing browsers the expiry date or maximum age. Then browsers can load the last downloaded resource from the cache instead of request it again.

2. What's the difference between HTML and XHTML?

XHTML is an HTML that follows the XML rules, which means a XHTML document must have well-formed markups in order to be rendered properly in all web browsers. Differently from XHTML, the HTML document can be rendered in most of the browsers even with markup errors such as no closing tags or wrong nested tags.

And how do I create a XHTML document?

XHTML is basically a HTML document with some rules that need to be applied. Have a look at these examples below and

spot the differences.

```
<head>
   <title>This is head</title>
</head>
<BODY>
   This is the body of the document with body tag in capital
letters
   Notice that there's no close body tag and no tag as well.
```

This HTML document above can be opened with no problems in Chrome, even containing many markup errors because most browsers can fix them for you automatically.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title></title>
  </head>
  <body>
  </body>
</html>
```

The code above is a well-formed XHTML document and that's the minimum you must have in order to render it. Notice the

declaration of the doctype at the top of the document and the namespace (xmlns) attribute in html open tag. These elements are mandatory as well as all the tags in lowercase.

3. When would you use GET and POST requests?

There are several technical differences between these two types of requests, regarding length limitation, security, caching and a few others. But if someone asks you WHEN would you use it, I'd say one of the most important points that any front-end developer should take into account is that we should only use GET for **idempotent** requests, it means requests that don't make significant changes in the backend system or database but if you do need to make inserts, updates or deletes in a database, trigger emails or any other major action, POST is recommended.

That's pretty much it that I wanted to share according to my recent interviews and I believe you'll come across some of these questions or a variation of them in your next technical interview.

Learn even more with my other posts:

Javascript ES6: Learn important features in a few minutes
Crud WebService for your json data in Node.JS and RethinkDB
HTML5 pushState and Single-Page apps

Rafael Oshiro

Front-End Developer working primarily with Javascript and Ruby on Rails. Talk to him http://twitter.com/roshiro.

Dublin / São Paulo • *http://rafael.oshiro@gmail.com*

Share this post

37 Comments

FrontEnd Journal - Javascript and FrontEnd Development articles

1 Login

Recommend 4 Share Sort by Best

Join the discussion…

R. I. • 2 years ago

Your answer to #3 is inaccurate. === (the identity operator) and == (the equality operator) behave identically, except == does type coercion first. So === just does the test, but === does coercion and then the test. See this little write up http://www.c-point.com/javascr... more details on how the coercion is done.

20 • Reply • Share ›

stefano • 2 years ago

I would avoid to say that Backbone.js is an MVC if you have any backbone experience. Backbone is an MV* and that make a huge difference when you organize and test your application. Backbone with Marionette get closer to an MVC standard pattern.

4 • Reply • Share ›

Chris Aquino • 2 years ago

Nice article! One note: MVC itself does not force you to use a REST architecture. REST is a style of web service. You can use MVC as a design pattern on the front end, while consuming any style of web service, including ones that are not RESTful. Many MVC frameworks are

built to support RESTful web services out of the box, but will let you customize how you interact with a web service.

4 • Reply • Share ›

Rafael Mod ➚ Chris Aquino • 2 years ago

Ok, I probably was to broad when I made a relation between MVC and REST. But in the case of backbonejs you can specify the urlRoot attribute in you Model to connect with your existing REST API.

Just edited the original post to make it more clear. Thanks

1 • Reply • Share ›

Nav • 2 months ago

Please add the following:

1. Are you familiar with the terms: Progressive enhancement and Continuous Integration?

Progressive enhancement is a strategy for web design that emphasises accessibility, semantic HTML markup, and external stylesheet and scripting technologies.

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

2. What is a UI thread?

The UIThread is the main thread of execution for your application. This is where most of your application code is run. All of your application components (Activities, Services, ContentProviders, BroadcastReceivers) are created in this thread, and any system calls to those components are performed in this thread.

For instance, let's say your application is a single Activity class. Then all of the lifecycle methods and most of your event handling code is run in this UIThread. These are methods like onCreate, onPause, onDestroy, onClick, etc. Additionally, this is where all of the updates to the UI are made. Anything that causes the UI to be updated or changed HAS to happen on the UI thread.

3. What is a closure?

A closure is an inner function that has access to the outer (enclosing) function's variables—scope chain. The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function's variables, and it has

access to the global variables.

2 ^ | v • Reply • Share ›

Oscar M • 10 months ago

Hi, nice article, just to add your "gmaps_sprite.png" image is missing.
Now I have few points to give, I hope you will find them interesting.

1. Regarding the HTTP Methods, I believe a better approach would be to explain it through CRUD and to use common conventions that are presented nowadays on web apis, like following:
GET - used for retrieving (READING) an information
POST and PUT - used for CREATING and UPDATING, unfortunately there is no universal answer to this because it can depend on various things ...
DELETE - well... for DELETING a resource

Also I believe you misunderstood the definition of idempotent, you see it does not matter what impact it has on back-end.
"Idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. This only applies to the result, not the resource itself." - see here

---

see more

2 ^ | v • Reply • Share ›

Alex • 2 years ago

Thanks, this was a nice review. Although at this point I would go SCSS or LESS over SASS, because of widespread adoption, the fact that SASS is more or less deprecated, and that regular CSS files will drop in as SCSS or LESS files without conversion.

2 ^ | v • Reply • Share ›

Rafael Mod → Alex • 2 years ago

Thanks for sharing that, Alex. I'll look into SCSS and LESS a bit more. It's always good to know what people are using more.

1 ^ | v • Reply • Share ›

Andy Feliciotti • 2 years ago

I recently had an interview and most of these questions came up, great post!

2 ^ | v • Reply • Share ›

Rafael Mod → Andy Feliciotti • 2 years ago

Good to hear that. Thanks, Andy.

2 ^ | v • Reply • Share ›

Andy Feliciotti ➦ Rafael • 2 years ago

They also had a trick question about semicolons after code in Js

13 ▲ ▼ • Reply • Share ›

Adrian Carballo • 2 months ago

"we should only use GET for idempotent requests, it means requests that don't make significant changes in the backend system or database". Idempotent actually means that the requests always return the same, regardless of how many times you query it, or what client is querying it; it has nothing to do with making changes to the backend.

1 ▲ ▼ • Reply • Share ›

develway • 23 days ago

Very nice questionnaire, but I think the question about XHTML must be changed to HTML5.

▲ ▼ • Reply • Share ›

Simon Dupon • a month ago

Question: How to remove duplicated values from an array?
Answer:

```
var testArray = [2,3,9,2,1,4,9];
var uniqueArray = [];

testArray.forEach(function(item){
if(uniqueArray.indexOf(item) === -1) {
uniqueArray.push(item);
}
});

console.log(uniqueArray);
```

▲ ▼ • Reply • Share ›

Front End Developer Hub • 6 months ago

Beside all those question I thinks the love and pation for work is very much important.

You can check it out

http://www.frontend-dev.com/fr...

▲ ▼ • Reply • Share ›

ustutorials.com • 7 months ago

Which Browser we are using how can we know with using JavaScript?
document.getElementById(“demo”).innerHTML =
“Developed by ” + navigator.appName + “. Browser Name “+ navigator.appCodeName;

Most Common Technical Interview Question for FrontEnd Developers

• Reply • Share ›

sudeep • 8 months ago

THanks it helps a alot... cheers !!!!

• Reply • Share ›

mastermalone • 9 months ago

I just went on an Interview today. Some of these questions were asked but the majority of the tech exercises where based on some oddball situations such as create a function that removed duplicate values from an array, and how would you take a array with an array as one of it's many values and flatten it out into a new array with all string values: Example:

var myArr = ["hi", "there", ["This", "is", "an", "Array"], "more", "strings"];

How would you flatten this to read as :
["hi", "there", "This", "is", "an", "Array", "more", "strings"];

I figured the answer out when I got home. Flattening arrays is something that I don't do often.

• Reply • Share ›

Simon Dupon ➚ mastermalone • a month ago

Not sure it is a best solution but it's working:

```
var myArr = ["hi", "there", ["This", "is", "an", "Array"], "more",
"strings"];
var flattenedArray = [];

myArr.forEach(function(item) {
if (!(item instanceof Array)) {
flattenedArray.push(item);
}else {
item.forEach(function(item){
flattenedArray.push(item);
})
}
});

console.log(flattenedArray);
```

• Reply • Share ›

Nipun Dutt • a year ago

Thanks for this:)

• Reply • Share ›

Sam Garg • a year ago

Nice post. Too bad I saw this AFTER my interview... I'm a front end guru/freelancer, contact me if u need some help. http://samgarg.me

• Reply • Share ›

Guest • a year ago

Nice post. Too bad I saw this AFTER my interview...

• Reply • Share ›

Amani2 • a year ago

These are actually useful and relevant questions! Unfortunately most of the things I get asked during interviews have little to do with FE stuff (algorithms….hash tables….etc). When I start interviewing candidates I may use some of these questions.

• Reply • Share ›

Rafael Mod → Amani2 • a year ago

I suppose mixing up a few question related to algorithms with some related to FE development can help you select good candidates. Good luck!

• Reply • Share ›

Aleks D. • a year ago

This helped amazingly, and thank you for not just showing the answer but explaining!

• Reply • Share ›

Rafael Mod → Aleks D. • a year ago

Glad to have your feedback, Aleks!

• Reply • Share ›

Jay • a year ago

Nice post, thanks.

• Reply • Share ›

Corie Bro • 2 years ago

Thanks for the post..I found your blog a day late...grrr.. I'm a software developer student who interviewed for Jr Web developer... Same questions asked

• Reply • Share ›

CN • 2 years ago

I would add a brief note on javascript recursion. But otherwise, dope list.

• Reply • Share ›

Rafael Mod → CN • 2 years ago

Thanks for the suggestion, CN

• Reply • Share ›

imran • 2 years ago

roshiro great job for Front End developer community, it really not only help us to prepare for interview but also it help us to understand basic Front end terminologies. Gr8 Stuff keep it up! :)

• Reply • Share ›

Rafael Mod ➚ imran • 2 years ago

Great to hear that, thanks!

• Reply • Share ›

whatsmyline • 2 years ago

Great post. Well done. If i could add to your last paragraph,i would clarify by elaborating a bit on the other http request headers as they elude to the basis of REST and autoCRUD. I.e.: Post for inserts, puts for updates, gets for queries, and delete for destroy.

• Reply • Share ›

Rafael Mod ➚ whatsmyline • 2 years ago

Thanks for the feedback! Yeah, elaborating a bit more on other