JavaScript

The Right Way

Hey, you!

This is a guide intended to introduce new developers to JavaScript and help experienced developers learn more about its best practices.

Despite the name, this guide doesn't necessarily mean "the only way" to do JavaScript.

We just gather all the articles, tips, and tricks from top developers and put it here. Since it comes from exceptional folks, we could say that it is "the right way", or the best way to do so.

CHOOSE YOUR PATH

GETTING STARTED

ABOUT

Created by Netscape in 1995 as an extension of HTML for Netscape Navigator 2.0, JavaScript had as its main function the manipulation of HTML documents and form validation. Before winning this name so famous nowadays, JavaScript was called Mocha. When it first shipped in beta releases, it was officially called LiveScript and finally, when it was released by Sun Microsystems, was baptized with the name by which it is known today. Because of the similar names, people confuse JavaScript

with Java. Although both have the lexical structure of programming, they are not the same language. Different from C, C# and Java, JavaScript is an interpreted language. It means that it needs an "interpreter". In case of JavaScript, the interpreter is the browser.

CURRENT VERSION

The JavaScript standard is ECMAScript. As of 2012, all modern browsers fully support ECMAScript 5.1. Older browsers support at least ECMAScript 3. A 6th major revision of the standard is being worked on.

A good reference to versions, references and news about JavaScript can be found at the Mozilla Developer Network.

THE DOM

The Document Object Model (DOM) is an API for HTML and XML documents. It provides a structural representation of the document, enabling you to modify its content and visual presentation by using a scripting language such as JavaScript. See more at Mozilla Developer Network - DOM.

JS CODE STYLE

CONVENTIONS

As every language, JavaScript has many code style guides. Maybe

the most used and recommended is the Google Code Style Guide for JavaScript, but we recommend you read Idiomatic.js.

LINTING

Nowadays the best tool for linting your JavaScript code is JSHint. We recommend that whenever possible you verify your code style and patterns with a Lint tool.

THE GOOD PARTS

OBJECT ORIENTED

JavaScript has strong object-oriented programming capabilities, even though some debates have taken place due to the differences in object-oriented JavaScript compared to other languages.

Source: Introduction to Object-Oriented JavaScript

ANONYMOUS FUNCTIONS

Anonymous functions are functions that are dynamically declared at runtime. They're called anonymous functions because they aren't given a name in the same way as normal functions.

Source: JavaScript anonymous functions

FUNCTIONS AS FIRST-CLASS OBJECTS

Functions in JavaScript are first class objects. This means that JavaScript functions are just a special type of object that can do all the things that regular objects can do.

Source: Functions are first class objects in JavaScript

LOOSE TYPING

For many front-end developers, JavaScript was their first taste of a scripting and/or interpretive language. To these developers, the concept and implications of loosely typed variables may be second nature. However, the explosive growth in demand for modern web applications has resulted in a growing number of back-end developers that have had to dip their feet into the pool of client-side technologies. Many of these developers are coming from a background of strongly typed languages, such as C# or Java, and are unfamiliar with both the freedom and the potential pitfalls involved in working with loosely typed variables.

Source: Understanding Loose Typing in JavaScript

SCOPING AND HOISTING

Scoping: In JavaScript, functions are our *de facto* scope delimiters for declaring vars, which means that usual blocks from loops and conditionals (such as if, for, while, switch and try) DON'T delimit scope, unlike most other languages. Therefore, those blocks will share the same scope as the function which contains them. This way, it might be dangerous to declare vars inside blocks as it would seem the var belongs to that block only.

Hoisting: On runtime, all var and function declarations are moved to the beginning of each function (its scope) - this is known as Hoisting. Having said so, it is a good practice to declare all the vars altogether on the first line, in order to avoid false expectations with a var that got declared late but happened to hold a value before - this is a common problem for programmers coming from languages with block scope.

Source: JavaScript Scoping and Hoisting

FUNCTION BINDING

Function binding is most probably the least of your concerns when beginning with JavaScript, but when you realize that you need a solution to the problem of how to keep the context of this within another function, then you might realize that what you actually need is **Function.prototype.bind()**.

Source: Understanding JavaScript's Function.prototype.bind

CLOSURE FUNCTION

Closures are functions that refer to independent (free) variables. In other words, the function defined in the closure 'remembers' the environment in which it was created in. It is a important concept to understand as it can be useful during development, like emulating private methods. It can also help to learn how to avoid common mistakes, like creating closures in loops.

Source: MDN - Closures

STRICT MODE

ECMAScript 5's strict mode is a way to opt in to a restricted variant of JavaScript. Strict mode isn't just a subset: it intentionally has different semantics from normal code. Browsers not supporting strict mode will run strict mode code with different behavior from browsers that do, so don't rely on strict mode without feature-testing for support for the relevant aspects of strict mode. Strict mode code and non-strict mode code can coexist, so scripts can opt into strict mode incrementally.

Source: MDN - Strict mode

IMMEDIATELY-INVOKED FUNCTION EXPRESSION (IIFE)

An immediately-invoked function expression is a pattern which produces a lexical scope using JavaScript's function scoping. Immediately-invoked function expressions can be used to avoid variable hoisting from within blocks, protect against polluting the global environment and simultaneously allow public access to methods while retaining privacy for variables defined within the function.

This pattern has been referred to as a self-executing anonymous function, but @cowboy (Ben Alman) introduced the term IIFE as a more semantically accurate term for the pattern.

Source: Immediately-Invoked Function Expression (IIFE)

PATTERNS

DESCRIPTION

While JavaScript contains design patterns that are exclusive to the language, many classical design patterns can also be implemented.

A good way to learn about these is Addy Osmani's open source book Learning JavaScript Design Patterns, and the links below are (in the majority) based on it.

DESIGN PATTERNS

- Factory
- Prototype
- Mixin
- Singleton

Creational Design Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Module
- Proxy
- Revealing Module

Structural Design Patterns

- Chain of Responsibility
- Command
- Mediator
- Observer

Behavioral Design Patterns

MV* PATTERNS

There are some implementations of the traditional MVC Pattern and its variations in JavaScript.

- MVC Pattern
- MVP Pattern
- MVVM Pattern

TESTING TOOLS

DESCRIPTION

Various libraries and frameworks to do tests in JavaScript.

LINKS

- Mocha

Maintained by TJ Holowaychuk

- QUnit

Maintained by jQuery

- Jasmine

Maintained by Pivotal Labs

- Karma

Maintained by the team behind AngularJS. Mostly by Vojta Jina

- Intern

Maintained by Sitepen

- Istanbul

A JavaScript code coverage tool written in JavaScript, maintained by Krishnan Anantheswaran

- Sinon.JS

Standalone test spies, stubs and mocks for JavaScript. No dependencies, works with any unit testing framework. Created by Christian Johansen, maintained by Sinon.JS community.

- DexterJS

A test helper to mock functions and the XHR object, maintained by Leo Balter

FRAMEWORKS

GENERAL PURPOSE

- jQuery

jQuery is a fast, small, and feature-rich JavaScript library. Built by John Resig.

- YUI

Built by Yahoo!, YUI is a free, open source JavaScript and CSS library for building richly interactive web applications. New development has stopped since August 29th, 2014.

- ZeptoJS

Zepto is a minimalist JavaScript library for modern browsers with a largely jQuery-compatible API. If you use jQuery, you already know how to use Zepto.

- Dojo Toolkit

Dojo is a free, open-source JavaScript toolkit for building high performance web applications. Project sponsors include IBM and SitePen.

MV*

- Backbone.js

Very popular JavaScript client-side framework, built by @jashkenas.

- Ember.js

Built by @wycats, jQuery and Ruby on Rails core developer.

- Knockout.js

Simplify dynamic JavaScript UIs by applying the Model-View-View Model (MVVM).

- Angular.js

Built by Google, Angular.js is like a polyfill for the future of HTML

- Cappuccino

Cappuccino is an open-source framework that makes it easy to build desktop-caliber applications that run in a web browser.

- JavaScript MVC

JavaScriptMVC is an open-source framework containing the best ideas in jQuery development.

- Meteor

Meteor is an open-source platform for building top-quality web

apps in a fraction of the time, whether you're an expert developer or just getting started.

- Spice.js

Spice is a super minimal (< 3k) and flexible MVC framework for javascript. Spice was built to be easily added to any existent application and play well with other technologies such as jQuery, pjax, turbolinks, node or whatever else you are using.

- Riot.js

Riot is an incredibly fast, powerful yet tiny client side (MV*) library for building large scale web applications. Despite the small size all the building blocks are there: a template engine, router, event library and a strict MVP pattern to keep things organized.

- CanJS

CanJS is a JavaScript framework that makes developing complex applications simple and fast. Easy-to-learn, small, and unassuming of your application structure, but with modern features like custom tags and 2-way binding.

LIBRARY

- React

Built by Facebook. React is a JavaScript library for creating user interfaces by Facebook and Instagram. Many people choose to think of React as the V in MVC.

- Handlebars

Handlebars provides the power necessary to let you build semantic templates effectively with no frustration.

- Dust.js

Asynchronous templates for the browser and node.js.

GAME ENGINES

- MelonJS

MelonJS is a free, light-weight HTML5 game engine. The engine integrates the tiled map format making level design easier.

- ImpactJS

ImpactJS is one of the more tested-and-true HTML5 game engines with the initial release all the way back at the end of 2010. It is very well maintained and updated, and has a good-sized community backing it. There exists plenty of documentation - even two books on the subject of creating games with the engine.

- LimeJS

LimeJS is a HTML5 game framework for building fast, native-experience games for all modern touchscreens and desktop

browsers.

- Crafty

Crafty is a game engine that dates back to late 2010. Crafty makes it really easy to get started making JavaScript games.

- Cocos2d-HTML5

Cocos2d-html5 is an open-source web 2D game framework, released under MIT License. It is a HTML5 version of Cocos2d-x project. The focus for Cocos2d-html5 development is around making Cocos2d cross platforms between browsers and native application.

- Phaser

Phaser is based heavily on Flixel. It is maintained by Richard Davey (Photon Storm) who has been very active in the HTML5 community for years.

- Goo

Goo is a 3D JavaScript gaming engine entirely built on WebGL/HTML5

- LycheeJS

LycheeJS is a JavaScript Game library that offers a complete solution for prototyping and deployment of HTML5 Canvas, WebGL or native OpenGL(ES) based games inside the Web Browser or native environments.

- Turbolenz

Turbulenz is backed by $5M in venture funding. The engine is very polished, and primarily focused on high-quality HTML5 games. There appears to be less support for mobile devices, but they will soon catch up when all mobile devices support webGL.

- Quintus

Quintus is an HTML5 game engine designed to be modular and lightweight, with a concise JavaScript-friendly syntax.

- KiwiJS

Kiwi.js is a fun and friendly Open Source HTML5 Game Engine. Some people call it the WordPress of HTML5 game engines

- PandaJS

Panda.js is a HTML5 game engine for mobile and desktop with Canvas and WebGL rendering.

NEWS

WEBSITES

DailyJS Echo JS The Treehouse Show

/r/javascript on Reddit Open Web Platform Daily Digest

Badass JavaScript AngularJS Daily

NEWSLETTER

JavaScript Weekly A Drip of JavaScript

Ember Weekly Backbone Weekly Node Weekly

Meteor Weekly Grunt Weekly Gamedev.js Weekly

HTML5 Weekly UDGWebDev Weekly

READING

ARTICLES

- Understanding JavaScript OOP

---

by Quildreen Motta

- Understanding “Prototypes” in JavaScript

by Yehuda Katz

- Prototypes and Inheritance in JavaScript

by Scott Allen

- Understanding JavaScript Function Invocation and “this”

by Yehuda Katz

- Partial Application in JavaScript

by Ben Alman

- Getting Over jQuery

by Nico Bevacqua

- A Dive Into Plain JavaScript

by Ryan Burgess

- A Deeper Look at Objects in JavaScript

by Kirupa Chinnathambi

- Closures in JavaScript

by Kirupa Chinnathambi

- Introduction to Easing in JavaScript

by Kirupa Chinnathambi

- How to Learn JavaScript Properly

by JavaScript Is Sexy

BOOKS

- Eloquent JavaScript

by Marijn Haverbeke

- JavaScript: The Definitive Guide

by David Flanagan

- JavaScript: The Good Parts

by Douglas Crockford

- JavaScript Patterns

by Stoyan Stefanov

- JavaScript Testing Recipes

by James Coglan

- Professional JavaScript for Web Developers

by Nicholas C. Zakas

- High Performance JavaScript

by Nicholas C. Zakas

- Human JavaScript

by Henrik Joreteg

- Object Oriented JavaScript

by Stoyan Stefanov

- Pro JavaScript Design Patterns

by Dustin Diaz

- Secrets of the JavaScript Ninja

by John Resig

- JavaScript Application Design

---

by Nicolas Bevacqua

- Speaking JavaScript

---

by Axel Rauschmayer

- Test-Driven JavaScript Development

---

by Christian Johansen

- You Don't Know JS

---

by Kyle Simpson

- You Might Not Need jQuery

---

by Ryan Burgess

- JavaScript Garden

---

by Ivo Wetzel

FREE E-BOOKS

JSBooks DevFreeBooks

PORTALS

HTML5 Rocks W3Fools Mozilla Developer Network

Web Platform Smashing Magazine Node School

How to Node Felix's Node.js Beginners Guide

PODCASTS

Ember Hot Seat JavaScript Jabber Node Up

5 Minutes of JavaScript The Meteor Podcast

SCREENCASTS

D F C

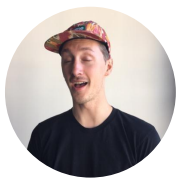
tagtree

WHO TO FOLLOW

Brendan Eich

isaacs

Richard D. Worth



fat

Michał Budzyński

Angelina Fabbro

Addy Osmani

Joe Zimmerman

Douglas Crockford

Paul Irish

TJ Holowaychuk

rauchg

Nicholas C. Zakas

John Resig

James Halliday

Dave Herman

Ricardo Cabello

Tim Caswell

Christian Heilmann

Rick Waldron

David Walsh

Nicolas Bevacqua

PAAS PROVIDERS

Heroku Modulus Nodejitsu OpenShift

dotCloud Getup Windows Azure

HELPERS


npm


YEOMAN


GRUNT


Gulp

CREA Creator of **http://jstherightw**  DD Jedi.

JavaScript Lover, VIM addicted rogrammer. ira

CONTRIBUTORS

This project wouldn't exist without these amazing contributors. Thank you guys for making this real!

gnuwilliam

zenorocha

leobetosouza

caio-ribeiro-pereira

luiztiago

bevacqua

itsryandrake

rmdias

heitortsergent

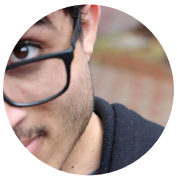
allanesquina

caiogondim

ellisonleao

alissonbovenzo

cironunes



pragmaticivan

vitorbritto

rafael-lima

yang-wei

appleboy

stackptr

djalmaaraujo

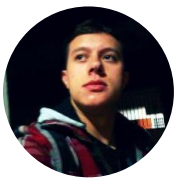
dylans

steinbitglis

mildfuzz

leobalter

rivfader



matheusazzi

mkautzmann

ryanburgess

tiagorg