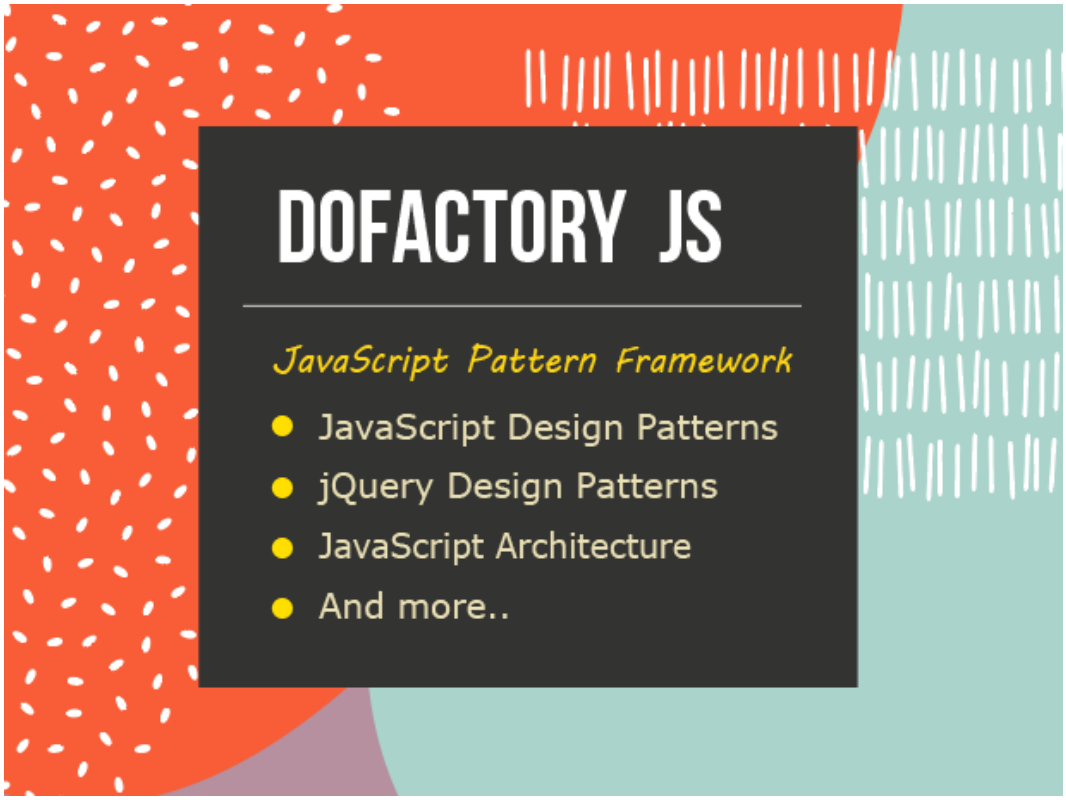
Dofactory JS 6.0

37 Ways to becoming a JavaScript Rockstar



by

Data & Object Factory, LLC

www.dofactory.com

9. 37 Ways to becoming a JavaScript Rockstar

37 Ways to becoming a JavaScript Rockstar

The JavaScript community is moving fast, and it takes considerable effort to keep up with the latest developments in this area. Here are 37 ideas and tips that will support your journey to becoming the next JavaScript Rockstar:

1. Explore and learn from open source JavaScript frameworks and libraries. You'll be amazed how this will focus and accelerate your learning of a language that will only become more important over time. Then extract best practices and patterns from these frameworks and put these into your own library or framework (we call it a personal briefcase that you take with you wherever your travels take you). It makes you a better JavaScript programmer overall.
2. Minimize your global footprint, meaning keep your variables and functions in modules. This will avoid name collision within your code and with other JavaScript libraries.
3. Before starting a new project, you and your team should agree on a set of conventions, which include coding standards, naming conventions, namespaces, file structures, file naming, version control, unit testing, 3rd party libraries used, etc.
4. If available, join your local JavaScript meetups (at meetup.com). It is a fun way to learn what's happening with JavaScript in your community.
5. Start a GitHub open source project. If your project is successful it is the fastest way to JavaScript Rockstar fame.
6. Attend a JavaScript conference once a year (either physical or online). JavaScript is very widely used and becoming more and more popular.

7. Write a simple game in JavaScript. You will learn an awful lot about JavaScript performance, the event loop, code optimization, and more. At the end you can play your game with family & friends.
8. Search for online video presentations on JavaScript. Look at YouTube, Vimeo, and other places. Some are wonderful and definitely worth watching.
9. Never stop learning. There is so much more interesting stuff to know.
10. Build yourself a starter kit that you can use in any new project. Seasoned programmers have this, either in their head or as a physical set of files. As you mature as a developer with many projects below your belt, you will find that most projects have a lot in common. Companies like to think that their apps are unique, but in reality, most have a lot in common with others.
11. Refine your coding and naming standards document up to a level that there is no doubt about how things should be formatted and look like. Again, the code produced by a team of developers should look as if it were written by a single developer. In your standards document, include all cross-cutting concerns such as error handling, error logging, authentication, testing, continuous integration, version control, etc.
12. Give back to the community and answer questions on stackoverflow.com or similar site. Your community standing and reputation will grow.
13. Visit amazon.com and keep up with the latest books and literature on JavaScript and jQuery. Even if you don't buy the books it is good to be aware where the action is.
14. Try TDD (Test Driven Development) and experience first-hand how this affects the way you organize and structure your code. It takes more time, but to many it is an eye-opening experience.
15. If job security is important to you bring your JavaScript skills up to the next level and start learning any of the latest JavaScript frameworks like Angular, React, or Vue.
16. Study other languages than JavaScript. It will broaden your horizons and will give you a better appreciation and understanding of JavaScript itself.
17. Visit websites you like and view their HTML, CSS, and JavaScript code. Many are not minified, and the JavaScript code is easy to read.
18. Avoid using JavaScript's eval. If you do, you're evil (according to JSLint).

19. Add comments where appropriate. Don't use trivial comments like 'iterate over array'. If an algorithm is complex explain it. If the business logic is not obvious explain why and how it works.
20. Keep you code DRY, but don't overdo it: keep also the Rule of Three in mind.
21. Although semicolons are optional, always use them at the end of your statements.
22. For performance reasons minimize the use of for-in loops. Regular for loops are much faster.
23. DOM access is slow. Store references to frequently used DOM objects in local variables.
24. When manipulating HTML try batching DOM manipulations because every time something changes a reflow will occur. Insert a group of row elements rather than a single row at a time.
25. Assign globals to local variables, such as, `var doc = window.document`. This will reduce the time JavaScript has to search through the scope chain (locals are searched before globals).
26. Always upper case the first letter of your constructor function names. Also, don't forget to include the new operator when creating new objects using these constructors.
27. Use try/catch sparingly. They carry overhead and should only be used if there is a way to recover from the error. Also, don't use try/catch where you can use a simple if statement.
28. Closures are a wonderful and powerful concept, but they come with some overhead. Use them when appropriate.
29. When changing the DOM use CSS classes rather than styles. Style changes cause a reflow for each change whereas a class batches a series of styles together causing only a single reflow.
30. Minimize HTTP requests. This includes script files, css files, images on your page, and also Ajax calls. Prefer *chunky* over *chatty* requests, that is, it is better to make a few large calls rather than many small requests. JavaScript and css files can be combined as are images using image maps and css sprites.
31. On the client always try to cache server data that has already been retrieved.
32. Organize biweekly brown-bag lunches and present something interesting about JavaScript to your fellow team members.

33. Use patterns where appropriate. Some developers become 'pattern happy' and force every little piece of code into a pattern. Patterns implementations can be complicated, so balance this against the KISS principle (KISS = Keep it Simple, Stupid).
34. jQuery is a very rich library and it is always good to go beyond the basics, such as selectors and event handling. Perhaps you can even dive into the source code and learn a few other tricks.
35. Keep an eye on new Internet startup companies. They are typically the ones that are pushing the JavaScript envelope the hardest. You'll be amazed at what they are able to accomplish.
36. Implement a Continuous Integration (CI) environment in which JavaScript is tested as part of a daily build.
37. Now that you have gotten this far, congratulations and we hope to hear from you as the next JavaScript Rockstar!