Differences between Node.js and the Browser

Both the browser and Node.js use JavaScript as their programming language. Building apps that run in the browser is a completely different thing than building a Node.js application. Despite the fact that it's always JavaScript, there are some key differences that make the experience radically different.

From the perspective of a frontend developer who extensively uses JavaScript, Node.js apps bring with them a huge advantage: the comfort of programming everything - the frontend and the backend - in a single language.

You have a huge opportunity because we know how hard it is to fully, deeply learn a programming language, and by using the same language to perform all your work on the web - both on the client and on the server, you're in a unique position of advantage.

> **What changes is the ecosystem.**

In the browser, most of the time what you are doing is interacting with the DOM, or other Web Platform APIs like Cookies. Those do not exist in Node.js, of course. You don't have the `document`, `window` and all the other objects that are provided by the browser.

And in the browser, we don't have all the nice APIs that Node.js provides through its modules, like the filesystem access functionality.

2. Another big difference is that in Node.js you control the environment. Unless you are building an open source application that anyone can deploy anywhere, you know which version of Node.js you will run the application on. Compared to the browser environment, where you don't get the luxury to choose what browser your visitors will use, this is very convenient.

This means that you can write all the modern ES2015+ JavaScript that your Node.js version supports. Since JavaScript moves so fast, but browsers can be a bit slow to upgrade, sometimes on the web you are stuck with using older JavaScript / ECMAScript releases. You can use Babel to transform your code to be ES5-compatible before shipping it to the browser, but in Node.js, you won't need that.

3. Another difference is that Node.js supports both the CommonJS and ES module systems (since Node.js v12), while in the browser we are starting to see the ES Modules standard being implemented.

In practice, this means that you can use both `require()` and `import` in Node.js, while you are limited to `import` in the browser.

Copyright OpenJS Foundation and Node.js contributors. All rights reserved. The OpenJS Foundation has registered trademarks and uses trademarks. For a list of trademarks of the OpenJS Foundation, please see our Trademark Policy and

Trademark List. Trademarks and logos not indicated on the list of OpenJS Foundation trademarks are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

The OpenJS Foundation | Trademark Policy | Privacy Policy | Code of Conduct | Security Reporting