Title: Why React Developers LOVE Node

Answer:

Some people mistakenly assume that Node is required in order to use React. *It is not.* You don't need Node to run a React project. You don't even need a browser.

React gives you a language to *describe a user interface (UI).* That interface could be the controls in your car, a modern fridge screen, or your microwave buttons. Basically, it could be any *interface* where a *user* needs to read and interact with an intelligent device.

React was initially made for the web user interface and it is most popular there. This is why we have the *ReactDOM* library that works with a browser's DOM. However, React today is also popular for non-Web interfaces like iOS, Android, virtual reality, and even command line interfaces.

Why Node?

Node is the most popular *platform for tools* to make working with React easier. Node is also a popular platform to run a web server that can host React applications. However, even if you do not host your React applications on Node, you can still use Node for the tools it offers. For example, you can host your React application on Ruby on Rails and use Node to build assets for the Rails Asset Pipeline.

The *Webpack* Node packages makes it super easy to bundle your multi-file React application into a single file for production and compile JSX (with Babel) during that process. This is an example of a Node tool that you can use on its own. You don't need a Node web server to work with Webpack.

However, there are so many reasons to pick Node over other options as the web server to host your React applications. Let me highlight the most important ones:

- Most of the React examples and projects you will find on the web are based on Node. Getting help is easier when your project is using Node.
- Node is JavaScript, which you are already using (with React). You do not need to invest in any other languages. Using the same language allows for true code sharing between server-side and client-side code. You will even stop saying server-side and client-side eventually. It is all just JavaScript that is used on both sides. Hiring for your project also becomes easier because, well, JavaScript is popular.
- *Most importantly:* you can execute your React front-end code in a Node environment. This is the easiest option to do server-side rendering and have Universal/Isomorphic Applications.

You can do server-side rendering with other languages but it would not be as easy as Node. The ReactDOM library has a component that is designed to work with Node to make server-side rendering as easy as a few lines of code.

Say that you have a React application that you render to the DOM with this line:

```
ReactDOM.render(<App data={DATA} />, mountNode);
```

The same React application can be rendered to an HTML string for server-sider rendering using this simple code in a Node-based web-server:

```
const ReactDOMServer = require('react-dom/server');
```

```
const html = ReactDOMServer.renderToString(<App data={DATA} />);
```

```
// In the server HTTP handler:
// Write the html variable to the Web response object
```

You can take this resulting `html` string and use it as the initial HTML of your application and you would be done. Your application will now work before the JavaScript on the client even wakes up.

Whether you do server-side rendering with Node or other options, it is a great win on two very important levels:

1. It will make your Web server SEO-friendly. Search engines will see the actual content instead of an empty HTML that loads a JavaScript bundle file. *(Note: search engines are getting better at executing JavaScript, but they are not there yet.)*
2. It will improve the performance of your Web application on slow and limited devices. Those devices will have content to start with (and interact with) while they slowly parse the client-side JavaScript application. Even after parsing that application, React on the client-side will initially do nothing because it has the exact same content as what was already mounted in the browser through server-rendered HTML.

Tags: react