**A Step-by-Step Guide to Building a React Universal Blog App** - [Neeraj Dana](https://neerajdana.medium.com/?source=post_page-----d1bdfc4877a3--------------------------------) Dec 26, 2021

Developing single-page apps with [React Web Development](https://www.biztechcs.com/reactjs-development/) is all about JavaScript, browsers, speed, design, and low visibility in search engines. It is because SPAs (single page apps) use JavaScript for loading page content. As web crawlers don’t use browsers, they are unable to view and index the content.

To deal with it, developers try various methods such as,

1. Adding the website’s fragment version that requires pages to be in static form. It only adds a lot of work.
2. Un browserify SPAs into static pages so that web spiders can crawl them. Of course, it’s a paid service.
3. Trusting that search engines can read our JavaScript only page content.

However, nothing works. But now you have a solution for ReactJS application development. Using Node.js on the server and React on the client side, you can build a universal JavaScript app. It would benefit on both side, browser and client side rendering. Humans and crawlers can view the SPA’s content.

So, how do you build such an app?

Let’s find out.

**Build A React Universal Blog App**

Before you start implementing the steps below, let me tell you there are two ways to build a React Universal App. Either you can start building the app from scratch. Or, you can use an existing codebase and curtail it in your way. Choice is yours!

**React Web Development App from Scratch**

Developing a React universal app will first render the markup on the server-side. It will let search engines view the content. Next, it will allow browsers to take the content as an app that’s fast and responsive.

Alright!

Our universal blog will use the following technologies.

* Node.js for package management and server side rendering
* React for UI
* Express for easy backend JS server framework
* React Router for routing
* React Hot Loader for hot loading in development
* Flux for data flow
* Cosmic JS for content management

Get Started!

**1. Run these Commands**

mkdir react-universal-blog  
cd react-universal-blog

**2. Create a Package**

Now, create a ***package.json*** and add the following components to the content.

* Babel to package our CommonJS modules and convert ES6 and React JSX to JavaScript.
* Cosmic JS official Node.js to serve our content from Cosmic cloud-hosted API.
* Flux to manage the app’s data flow.
* React for managing UI on server and browser.
* React Router to help in routing on server and browser.
* Webpack to bundle everything in bundle.js file.

After adding these details, add a script in ***the package.json*** file. When you run ***the npm run development*** command, the script will copy ***index.html*** content from the **view**folder and paste it into the **public**folder. From there, it will set our content base from ***webpack-dev-server*** to public. It will enable hot reloading and help debug components at the source. This functioning will also notify us about the fallbacks if any.

**3. Setup Webpack Configuration File**

After setting the package, configure your webpack by editing ***the webpack.config.js*** file.

While editing,

* Add the entry property with ***the app-client.js*** file name. It will act as an entry point, and webpack will bundle our application here and output it to ***/public/dist/bundle.js.***
* You can also use loaders. Hot loaders will help you with no-page refresh loading during ReactJS Application Development.

**4. Design your Theme/ Use Pre-built**

The next part is to design your theme. Either you can custom design the blog’s feel and look, or you can use the predefined themes. For now, let’s use the pre-designed theme.

**5. Create a View Folder**

Create a folder named “***views***” in the ***index.html*** file. Add the code to it.

For styling, you can add all JS and CSS files in the Publicfolder. You can get the files from the GitHub repository.

Remember, you can refrain from using JQuery by using the React Bootstrap package.

**6. Using the Pre-built JQuery Functionality**

To give your app a structure, get in your***index.html*** file. And there, set React Mount Point to ***div***where ***id= “app.”*** It will convert our template variable to server render markup. And when the browser comes in, React will take over the control and mount to ***div***where ***id=” app.”***

To enhance the user experience, add ***class = “hidden”*** to the body. And remove it once the React is mounted.

With this, the static pieces are done.

**7. Build React Components**

Now, let’s start developing the UI of our React Web Development App. For this, we need to set up the pages of our blogs. Therefore, let’s add the following pages

* Home
* About
* Work
* Contact

First, create an ***app-client.js*** file. Make sure it has a ***Router***component in it. This component consists of browser history for our client-side routing. As server render markup doesn’t need browser history, let’s create a separate route.js file to share between server and client entry points.

With it, your basic blog is ready. It’s time to run your application and test it.

**8. Run the application on localhost**

To run the app on localhost, run these commands:

mkdir public  
npm install  
npm run development

Then, navigate to [http://localhost:8080](http://localhost:8080/) to view your app in action.

**9. Run the App on the Server**

Congratulations on app development!

To run it on the server, create an ***app-server.js*** file. In it, load the basic routes that you’ve set up. It will convert the rendered markup into a string and pass it as a variable to the template.

You’re ready to run the app on the server. Before that, create a script so that you can run the code on the server and client side, both.

To do this,

* Open the ***package.json*** file
* Edit the script part in it
* In your terminal, execute “***npm start***”

**10. Run the App**

Your app is ready. Navigate to the localhost in your browser. See your simple blog and navigate in the SPA mode.

**Add and Edit Content**

If you wish to add and edit the content, follow these steps.

1. Add a store by including AppStore.
2. Create a store folder and add an event emitter. It will allow you to edit data in the store.
3. Build some components. Add an event listener to re-render the state. Add getstore method to render data on the client side.
4. Create page components. They will serve as templates for blog pages. Next, add the page title.
5. Create AppDispatcher. It will accept information from high-level components and distribute it evenly to the store.
6. Add actions now.
7. To receive data from cloud API, configure your Cosmic JS CMS.After setting all the React and Flux architecture, edit the ***app-server.js*** file to render the data on the server-side.

**React JS Application Development using Existing CodeBase**

Now, let’s see how to develop a React Universal App using codebase.

1. Sign up for Cosmic JS.
2. In Cosmic JS, create a new bucket. Here, the bucket name represents your website name, project name, or client application that you are building.
3. After signing up and creating a bucket, you’ll find two options. One is to build the app from scratch. Another is “see some apps.”
4. As you are building using existing modules, hit the right option, i.e., “see some apps.”
5. Now, Cosmic JS provides you with different programming languages like Node.js, PHP, React, Angular, and more. Hit the install button underneath the React Universal Blog icon.
6. After installation, click on the “Deploy to Web” button. And confirm the deployment.
7. Once it’s done, you can start editing global objects, objects, and object types in the Cosmic JS dashboard.

That’s it!

Your React Web Development App is ready.

*There is a great article about commenting in React JSX on John Kavanagh’s*[*Front end developer’s blog*](https://johnkavanagh.co.uk/writing/commenting-in-jsx/)

**It’s a Wrap**

Building SPA gets comfortable with React. Follow any of these two procedures and build a React Universal app that uses React and Node.js. The best part of using this technology for React JS Application Development is that your app is fast, loads instantly, and runs on both, client and server side. Also, search engines will be able to rank it now.

**Summary**

There are two ways to build a React Web Development App. Check out this exclusive guide and build your React Universal Blog App today.