

Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

Mar 29, 2017 · 10 min read

Simple React Development in 2018

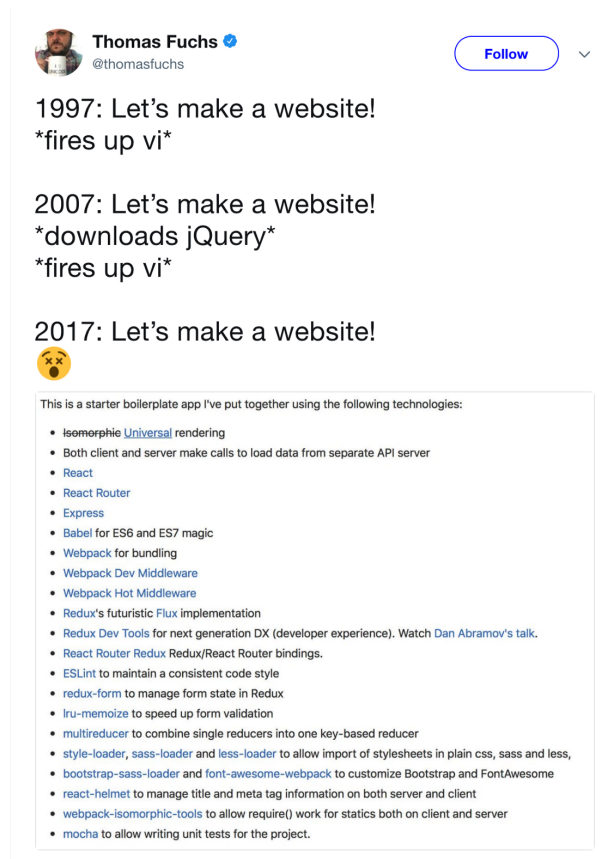
. . .

Hi, aspiring React developer! 🙌

Maybe you're a jQuery developer looking to experiment with a modern framework. Or, maybe you're an Angular developer wanting to see what all the hype is about. Maybe you're a seasoned back-end or systems programmer looking to make the jump to front-end. Maybe you're new to programming in general, but want to learn how to build dynamic web applications.

Regardless of your background, you've likely had a similar experience: React itself seems pretty straightforward, but the tooling and ecosystem is overwhelming.

You probably feel like this guy:



<https://twitter.com/thomasfuchs/status/834481271443226627>

I've seen and heard variations on this theme frequently. Especially for newcomers, React development can often seem like a complex maze of tools and libraries. Even experienced developers talk about how complicated modern web dev is, and pine for the simple days of just opening an editor and writing.

However, this is a solved problem!

Believe it or not, it is actually very simple and painless to start a new React project, thanks to amazing work by the community over the past year.

The goal of this guide is to showcase how easy it can be to start modern React development. It shares a step-by-step process, from initial system setup through to deployment, without straying into tangent explanations that aren't critical at this point in the learning process.

No prior experience with React or its related tools is required, although it does assume that you know the basics of working with a command

line and git.

This is a living document, and I'll be updating it as the landscape changes.

Last Updated: January 2nd, 2018 (Happy new year!)

. . .

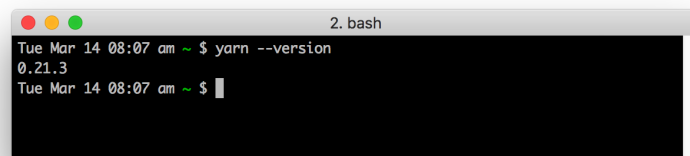
Step 1: Install Yarn

Yarn is a package manager. It helps us manage our project's dependencies.

If you're new to programming, essentially you can think of this as a tool to help us easily download and use other people's code.

Download and installation instructions are available **on the Yarn site**.

Once you've installed yarn, verify that you can use it on the command line. Open a new terminal, and run:

A screenshot of a macOS terminal window titled '2. bash'. The terminal shows the command 'yarn --version' being executed, which returns the output '0.21.3'. The prompt is 'Tue Mar 14 08:07 am - \$'.

yarn — version

But what about NPM?

You may have seen that NPM is the de facto package manager in the Javascript community, and may wonder why I'm suggesting Yarn.

*Without getting too far off-topic, Yarn is a concerted effort by the community to improve package management. Internally it still uses the NPM package repository, so you can **still install all the same packages**.*

In rare cases, you may run into issues with Yarn, specifically with `yarn global add`. If you happen to run into any problems, substitute with `npm install -g`.

If you're curious, you can read more about the differences between Yarn and NPM, but it's not critical knowledge at this point.

. . .

• Step 2: Create a React application

When people talk about complexity in React development, often they're referring not to React itself, but in the build system and development environment.

They have a point—as React developers, we tend to build pretty complex dev environments and build systems. We suffer this complexity because it makes React development quicker and more enjoyable.

However, that complexity is overwhelming for those just getting started, and worse, it gets in the way of learning what's really important: React itself.

Thankfully, there is a have-your-cake-and-eat-it-too solution. It's called **create-react-app**.

create-react-app is a command-line tool built by bright people at Facebook, and it's *amazing*. It abstracts away all the complexity and difficulty of implementing Webpack, Babel, a dev server, a production build process, and a thousand other tedious-but-critical things.

It's zero-configuration, and doesn't clutter up your project directory with a bunch of files you don't understand.

***Note:** Eventually, you will want to spend the time learning how this works internally. There's a lot of neat stuff you can do with Webpack, for example. Our top priority at the moment, though, is getting you excited*

about building with React, so we can defer this stuff for later.

Yarn comes with a built-in helper for using create-react-app:

```
yarn create react-app [your-app-name]
```

I'd like my application to be named *cats-n-stuff*, so I can simply run:



```
Mon Oct 23 07:05 am work $ yarn create react-app cats-n-stuff
yarn create v0.25.4
warning No license field
[1/4] ? Resolving packages...
[2/4] ? Fetching packages...
[3/4] ? Linking dependencies...
[4/4] ? Building fresh packages...
success Installed "create-react-app@1.4.1" with binaries:
- create-react-app
warning No license field

Creating a new React app in /Users/joshu/work/cats-n-stuff.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

yarn add v0.25.4
info No lockfile found.
[1/4] ? Resolving packages...
[2/4] ? Fetching packages...
[3/4] ? Linking dependencies...
[4/4] ? Building fresh packages...
Done in 46.09s.

Success! Created cats-n-stuff at /Users/joshu/work/cats-n-stuff
Inside that directory, you can run several commands:

  yarn start
    Starts the development server.

  yarn build
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd cats-n-stuff
  yarn start

Happy hacking!
Done in 51.55s.
Mon Oct 23 07:06 am work $
```

```
yarn create react-app cats-n-stuff
```

I believe `yarn create` is available in Yarn 0.25 and higher. Try upgrading

Yarn if you can't use this command.

This command does quite a few things:

Downloads and installs `create-react-app`

Creates a new project folder, `cats-n-stuff` (or whatever you name your app)

Scaffolds out a new project

Installs dependencies

You'll notice the output towards the end gives you a set of commands. Let's start by following its directions:

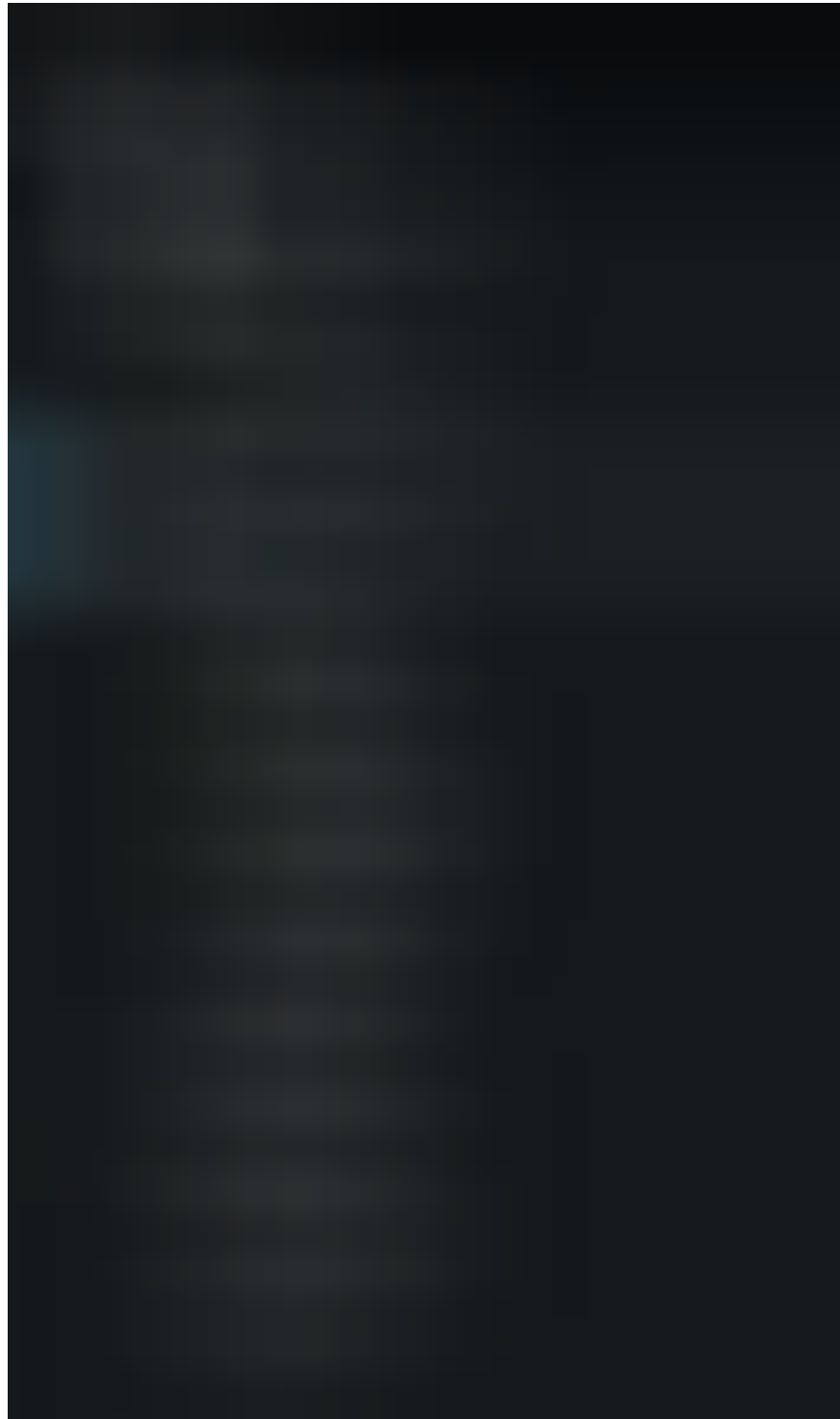


```
cd cats-n-stuff && yarn start
```

Bam! If we navigate in our browser to the address provided, we see a swanky demo page:



If we open our new directory in an editor, you'll see that create-react-app has scaffolded out everything we need (and nothing superfluous).



node_modules holds all of our third-party code, such as React.

public is where our static assets go, like our favicon. It also houses our project's only HTML file. Because React is all-javascript, you only really need the HTML file to update the `<head>` stuff, like

title and meta tags.

`src` is where our application actually lives. I'd recommend exploring all these files, to get a sense of how they work together.

Finally, the root directory holds stuff related to dependencies (`package.json` and `yarn.lock`), as well as a comprehensive documentation (`README.md`) for `create-react-app`.

. . .

Is this only for beginners? Will I be handicapping myself down the line?

create-react-app abstracts away all configuration, to help you focus on writing the application itself. At some point, though, you may want to access that configuration.

*Thankfully, create-react-app has **an eject feature** that removes this abstraction and exposes the underlying configuration. This means that create-react-app is a brilliant tool not just for newcomers, but also for experienced React devs.*

I've been using React for years, and I use create-react-app on all new side projects :) I try to go as long as I can without ejecting.

. . .

Step 3: Write your application

Ok, we're looking pretty good here! In less time than it takes to write up an HTML template, we've scaffolded out an entire front-end project.

Now we just have to, you know, write our actual application.

Start with *just* React.

If you follow many of the tutorials intended for intermediate or advanced users, you may think you need to learn about and install a

dozen “supplementary” packages. After all, React is just the view layer, right?

Wrong. Facebook used to say that React was the “V” in “MVC”. They stopped using that language, because it isn’t true; React is an entirely different paradigm, not simply a template renderer. The truth is that React is surprisingly capable *on its own*.

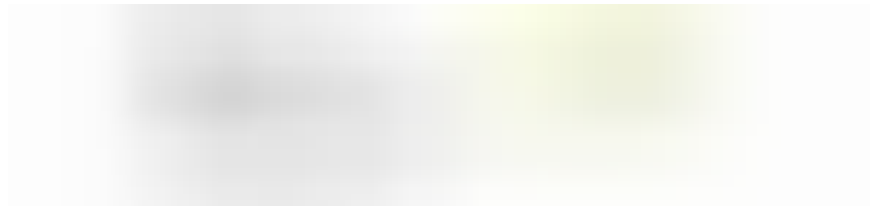
For example, the most critical package people typically recommend is React Router. Even the most trivial app will need some kind of routing built-in, so clearly we’ll need to use it in our first application, right?

Well, first of all, I believe the best way to learn is to make several small “weekend project” applications. Those types of applications don’t often need more than a single route: your goal is to build something quick and neat in React, do you really need a bunch of different pages? Start small, work your way up to more complex projects.

Additionally, there is tremendous educational value in learning how to solve these kinds of problems yourself. Why not try to implement your own basic router? The React API is powerful—React Router is still just React, after all!

You may discover, after your second or third small React project, that certain things are more painful than you’d like. Maybe your homegrown router is proving to be too much trouble, or maybe you feel like your application’s state is hard to manage. This is when you should be searching for pre-existing solutions; *after* you’ve encountered the problem, not before.

Spend time learning React on its own. Add dependencies as you need them. Try to solve problems yourself; if nothing else, you’ll understand the benefits of things like Redux or React Router.



Dan Abramov, co-creator of Redux, explaining that Flux libraries like Redux aren't always needed.
https://twitter.com/dan_abramov/status/704304462739939328

. . .

Resources for learning React

After spending some time googling for React tutorials, I've realized exactly why newcomers find React so confusing.

It's actually pretty tough to find reliable, up-to-date tutorials that cover *just* React, without delving into webpack and redux and a million other things.

Here are the resources I've found that do the job:

1. **Official React Tutorial**—Straight from the horse's mouth. Build a Tic Tac Toe game.
2. **The Road to Learning React**—A pay-what-you-want ebook that builds a fully-functional HackerNews clone, using create-react-app and minimal third-party libraries. Fantastic resource.
3. **FullStack React 30 Days of React**—The first 15 days contain a ton of great info for the core React API and how to use it.

. . .

Step 4: Deploy

Alright, you've built an awesome little toy app. Now you want to show your creation to the world. How do we get this thing online?

Web development is fun, and it's even better when you wind up with something neat to show off. There's an incredible amount of

satisfaction achieved by showing the world your random side projects.

The first thing we'll need to do is build our project.



yarn run build

What does this do?

'build' is an NPM script provided by create-react-app. It bundles up your code into ready-to-deploy static files. If you check the 'build' directory in your project, you'll see what it generates.

Deployment services

As with everything else in modern web development, there are a bunch of options for how to actually get your built files onto the internet.

Create-react-app's documentation covers several possible deployment options, and in my experience, Surge.sh is the quickest, simplest way to get code online quickly.

Let's install it with Yarn:



yarn global add surge

We install it globally so that we can use its CLI tool. Let's run it now:



surge

Running *surge* will walk us through the deployment process. Enter an email and password to create a new account.

The third step, *project path*, is important. It should auto-fill the current location, but you'll need to add */build* to the path, so that it knows to serve only the built project files, and not the source files.

Once you enter these 3 fields and select a domain for your project to live at, Surge does all the work to get it online.



surge (continued)

Amazingly, this is all we have to do. If we go to our supplied domain, we see our project:



cats-n-stuff.surge.sh

Simplifying repeat deploys

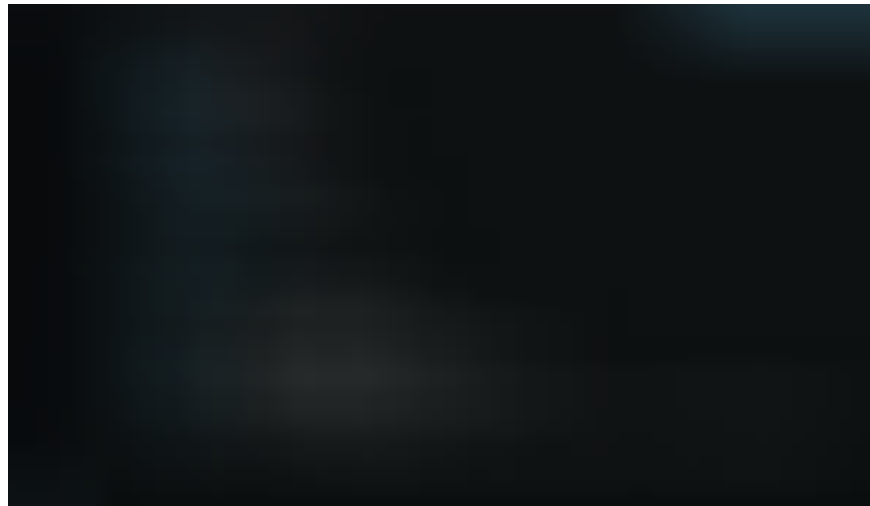
If we wanted to, we could just repeat this process every time; build, run *surge*, and then specify a project path and domain name.

We can use NPM scripts to be a little fancy though. Add this line to your `package.json`, under “scripts”:

```
"deploy": "yarn run build && surge -p build -d your-  
domain.surge.sh"
```

Terminal commands can be stringed together with `&&`, and so we can create a single NPM script to do all the stuff we need. We build a fresh copy of our project, and then deploy it with Surge, pre-filling some of the required fields.

This command starts by building the latest version of our project, and then invoking Surge with some default arguments. Here’s my `package.json`, with the deploy script:



Line 17 is the ‘deploy’ line.

Now, all we have to do is run a single command, and in under 20 seconds, our glorious project is up for anyone to see.



yarn run deploy

. . .

Conclusion

There are two misconceptions that newcomers often have about React:

- It takes forever to set up a decent build system and dev environment. You can't start writing your actual application code until you spend *hours* getting that set up.
- React is a small piece of the toolset needed to create basic web applications, and you have to learn a dozen other things before you can get started.

I hope I've done a good job disproving these theories. You can get started *in seconds* building a glorious front-end web application, and you can start without any extra dependencies.

Addendum: Community Resources

By far the most popular community hub for React developers is **Reactiflux**. It's a great place to meet React developers, and get help if you find yourself stuck or confused.

For myself personally, Twitter has been an amazing source to learn more about React and meet the community. Dan Abramov, a member of the React core team and co-creator of Redux, has a **great list of people to follow**.

Also, Facebook maintains a list of **React conferences**. Speaking from experience, React conferences are awesome.

. . .

Thanks for reading!

*I'm keen to hear your feedback on this article!
Anything I could do to make it more beginner-
friendly? Let me know on twitter.*

*Finally, if you found this post helpful, please give it
a recommendation by clicking the clap icon down
there :)*

