**ESLint Configuration**

It is a pluggable linter that’s going to tell you if you’ve imported something and not used it, if your function could be short-handed, and loads of other little gotchas that you can fully configure. I really think that it makes me a better developer — I know the things it normally picks me up on, so I learn not to do them! Let us go through how I get up and running in a new React project with ESLint installed. Though, this setup could be used for any Javascript application. I’m going to assume you already have [npm and node](https://www.npmjs.com/get-npm) installed.

**Starting a New React App**

There are a whole bunch of React starter projects out there, but Facebook’s own — create-react-app is one of the simplest. All you need to do is create-react-app app-name and you have a React starter with a local server, Babel, Webpack and Jest. It's a super easy way to get setup, and naturally [Facebook have some great documentation](https://github.com/facebook/create-react-app).

**Installing ESLint**

Installing our linter could not be easier. Straight to the terminal and npm install -g eslint. This installs ESLint globally on your machine, so you can use it on any future project. If you just want to run it on one project the commands are a little different, but they're available to copy and paste in the [ESLint documentation](https://eslint.org/docs/user-guide/getting-started).

Now it is installed, we need to configure it. The config file will let us define what features and rules we use. If we want our linter to recognise ES6 syntax but break when we have an undefined variable, we need a config file. To create one, we can run eslint --init.

This will create a file in our root called .eslintrc.json, with some default rules such as disallowing the use of console and debugger. We'll mess with these later.

That’s it for the default setup! You can now run eslint to get configuration suggestions, or start linting files with eslint \*\*/\*.js. But we aren't quite finished yet. Let's say we have a React component that looks like this:

import React from 'react';  
import Header from './components/Header';

const App = () => {  
 return (  
 <div>  
 <Header />  
 </div>  
 );  
};

export default App;

As you can see, all we want to do is to render that one component. But if we run eslinton this file, we will get a couple of errors.

1:8 warning 'React' is defined but never used no-unused-vars  
2:8 warning 'Header' is defined but never used no-unused-vars

This is because ESLint does not know that this is a React app, and it doesn’t know that the things we imported are components, not variables. We need a little extra plugin in here to fix this.

**Linting a React App**

In our terminal again, we need to npm install -g eslint-plugin-react. Alternatively, if you're running ESLint locally, we run npm install eslint-plugin-react --save-dev. This will install the plugin we need, but we need to tell ESLint that we want this plugin to help us out. In our ESLint config file, we just need a few extra lines.

"extends": [  
 "eslint:recommended",  
 "plugin:react/recommended"  
]

By adding this to our JSON object, we can run ESLint again and the plugin will tell the linter that it is looking at React code, not regular JavaScript. You can also add various settings to this plugin including what version of React you are using and other things such as Flow version. Learn more about these on the [npm plugin page](https://www.npmjs.com/package/eslint-plugin-react).

Just to test it is working, go into any of your React files and at the top, add a line like this:

import Blank from './';

Blank is not a real component, so when we run our linter, we’ll receive an error for this, but no other errors for importing React or other real components, and we know our linter is working.

3:8 warning 'Blank' is defined but never used no-unused-vars

Now you can lint your entire React app! Wasn’t that easy? If you’re looking for more plugins, checkout [awesome-eslint](https://github.com/dustinspecker/awesome-eslint). This is a big ol’ list of plugins, parsers and tools for ESLint that are all installed in the same way as eslint-plugin-react.

Before I finish, here is a few extra tips:

**#1: npm script**

That’s all pretty good, but typing out eslint src/\*\*/\*.js src/\*\*/\*.jsx is a real hassle. I prefer to have one script to run that'll lint our app without us writing more stuff! So head to your package.json file, and in there you should see some scripts that create-react-app gave us such as, start and build. We can add one to do our linting for us like so:

"scripts": {  
 "start": "react-scripts start",  
 "build": "react-scripts build",  
 "test": "react-scripts test --env=jsdom",  
 "eject": "react-scripts eject",  
 "lint": "eslint src/\*\*/\*.js src/\*\*/\*.jsx"  
}

You’ll notice that the script finds all of the .js and .jsx files in there. I like to use the .jsx extension as my IDE then gives me things like Emmet in return statements. You can customise this as much as you want.

Finally, with our new script, we can go to our terminal and run npm run lint just like we would for any other script.

**#2: ESLint Rules**

Lastly, let’s look at the rules we can give to ESLint. They have a [big list of rules](https://eslint.org/docs/rules/) for their plugin, and you can see which ones are turned on by default. They organise them nicely into things that might cause errors, best practices, variables, stylistic choices (see for indentation freaks), ES6 and more! Each of them has a page where you can take a look at some examples that will fail and succeed, and I’m sure if you looked through them all there’d be some really useful stuff… But who’s got the time to look through all that?

One option we have is to use a preset configuration. [Awesome ESLint](https://github.com/dustinspecker/awesome-eslint#configs) has a bunch of these including configurations from AirBnb, Facebook and Google. If those guys use them, they can’t be bad, right?

Reference

https://medium.com/@RossWhitehouse/setting-up-eslint-in-react-c20015ef35f7

https://eslint.org/docs/rules/