In the last section we got webpack’s output configured - but to use babel we had to add a loader to our webpack config. We used it then without knowing what it was, but now we can revisit it.

Let’s take another look at that loader.

```
module: {
    rules: [
        {
            test: '/\.js$/',
            exclude: /node_modules/,
            loader: "babel-loader"
        }
    ]
}
```

Now take a look at how Webpack describes loaders:

> Out of the box, webpack only understands JavaScript and JSON files. Loaders allow webpack to process other types of files and convert them into valid modules that can be consumed by your application.

So loaders allow us to transform files of one type into another type so that webpack can work with them. It might not have looked like we were transforming file types in the step shown above, but we were actually taking vanilla js files and running the babel-loader over them to turn them into es6 files.

There are all sorts of loaders for webpack - take a look at **this list**. In webpack, most of the things you need to do will end up needing a loader.

We will visit a few more loaders later, but for now, just notice how they work.

The `rules` array will contain all of our loaders, each loader specifies what types of files it will run on by running a regex matcher - in the case above we are looking for all .js files - the `$` at the end simply means that nothing comes after that.

But simply looking for all the .js files in our project would be problematic, as we don’t want to run this on all the files we have in our node modules. For that kind of use case, we also have an exclude option available to us, and then we simply name the loader to be run on the selected files. Some loaders will have different options, you can always look it up in the loader documentation.

Quiz

Being able to read documentation is a super power that will allow you to rise quickly in your skills as a developer. Go to the **webpack loader documentation** and recap one or two things that you learn, or that confuse you.

Your reflection

Loaders is like plugins which are used to work with different formats and languages

Things to think about

Thanks for your response.

Quiz

What webpack loader regex rule would you write to find all css files?

/\.css$/

Interview Question

What are some examples of es6 syntax? Do you have a favorite?

Your reflection

import { functionName } from 'javascriptFileFullOrRelativePath' export { functionName }

Things to think about

When you are going for a front end job interview, it’s always good to have some of these on the top of your head.

Resource for ES6

Check out the **ECMAScript 6 — New Features: Overview & Comparison** with respect to ES5.