## Switch the Git Branch

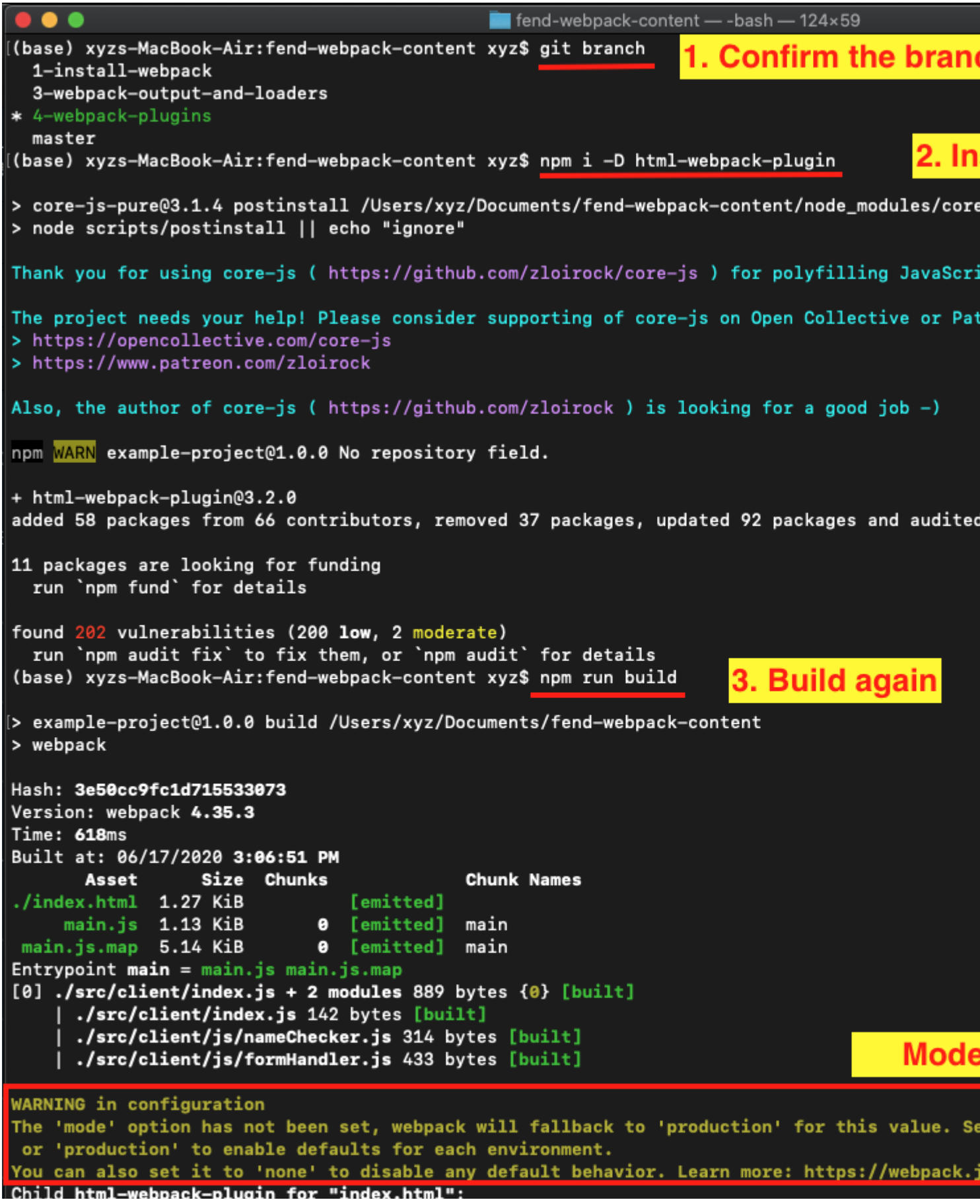
If you wish to practice the steps mentioned in the video demonstration below, use the `4-webpack-plugins` branch.

Else, you can switch to the branch `5-webpack-mode` corresponding to the current exercise where all the steps have already been carried out.

```
git checkout 5-webpack-mode
git branch
```

## Webpack Mode

The last core Webpack concept we’re going to cover is Mode. In the last concept, when you build, you probably noticed the warning:


fend-webpack-content — -bash — 124×59
(base) xyzs-MacBook-Air:fend-webpack-content xyz$ git branch
1-install-webpack
3-webpack-output-and-loaders
* 4-webpack-plugins
master
1. Confirm the bran
(base) xyzs-MacBook-Air:fend-webpack-content xyz$ npm i -D html-webpack-plugin
2. In
> core-js-pure@3.1.4 postinstall /Users/xyz/Documents/fend-webpack-content/node_modules/core
> node scripts/postinstall || echo "ignore"
Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScri
The project needs your help! Please consider supporting of core-js on Open Collective or Pat
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock
Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)
npm WARN example-project@1.0.0 No repository field.
+ html-webpack-plugin@3.2.0
added 58 packages from 66 contributors, removed 37 packages, updated 92 packages and audited
11 packages are looking for funding
run `npm fund` for details
found 202 vulnerabilities (200 low, 2 moderate)
run `npm audit fix` to fix them, or `npm audit` for details
(base) xyzs-MacBook-Air:fend-webpack-content xyz$ npm run build
3. Build again
> example-project@1.0.0 build /Users/xyz/Documents/fend-webpack-content
> webpack
Hash: 3e50cc9fc1d715533073
Version: webpack 4.35.3
Time: 618ms
Built at: 06/17/2020 3:06:51 PM
Asset Size Chunks Chunk Names
./index.html 1.27 KiB [emitted]
main.js 1.13 KiB 0 [emitted] main
main.js.map 5.14 KiB 0 [emitted] main
Entrypoint main = main.js main.js.map
[0] ./src/client/index.js + 2 modules 889 bytes {0} [built]
| ./src/client/index.js 142 bytes [built]
| ./src/client/js/nameChecker.js 314 bytes [built]
| ./src/client/js/formHandler.js 433 bytes [built]
Mode
WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value. Se
or 'production' to enable defaults for each environment.
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.j
Child html-webpack-plugin for "index.html":

Mode Warning in Configuration

The issue is that we haven't told webpack which mode to run in. Modes won't make sense until we delve into environments. You have probably already come across the idea of environments in code projects, or at least heard mention of a "development" or "prod" environment. But in order to use webpack to its true potential, we have to fully understand these concepts.

## Production vs. Development Environments

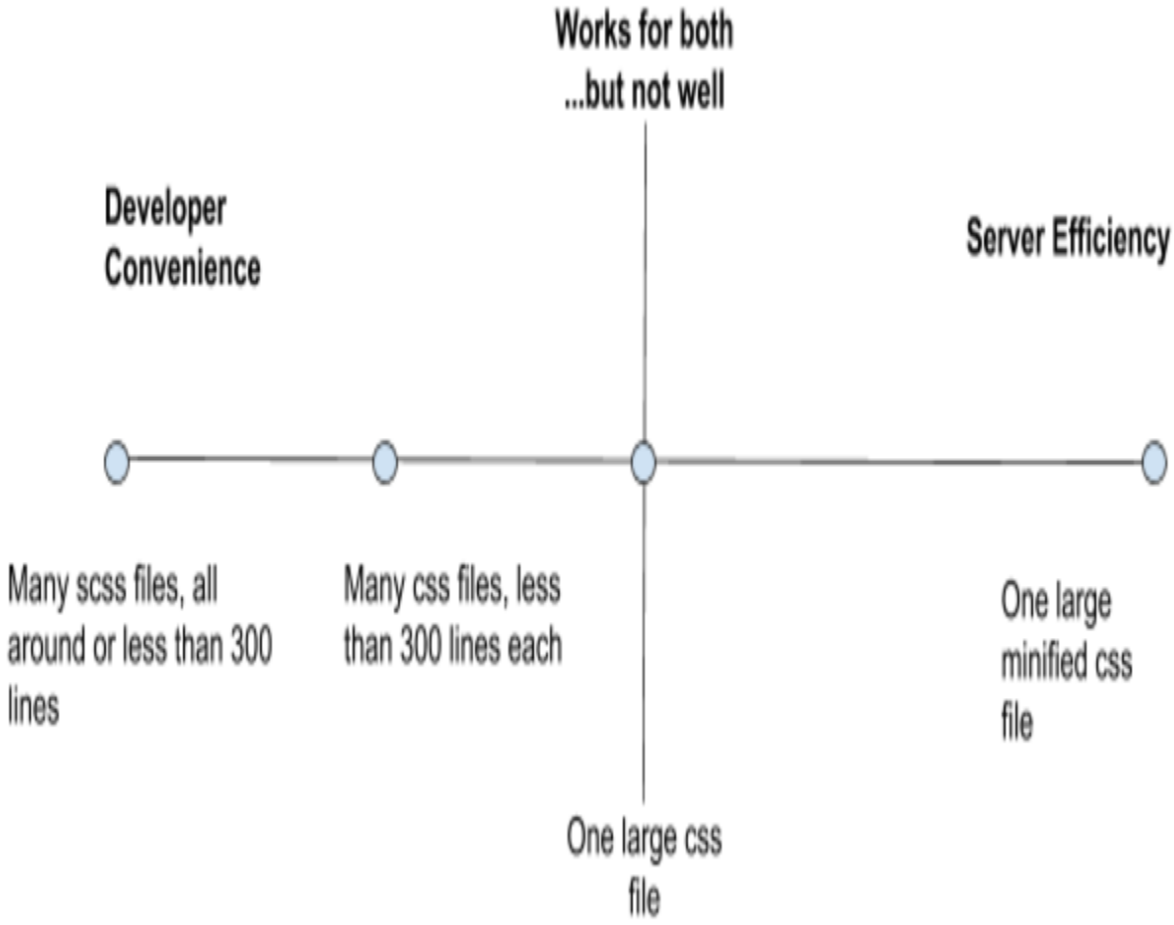
Developers refer to the various "states" of a website as environments. When we are developing a website, we call it the development environment - we run the server on localhost and use tools that are specifically convenient for us as developers. On the other hand, there is a production environment, which is our code on a server and where we can tune every tool and file for optimal efficiency, thereby giving our users the best experience when they use the webpage. There can be many environments for a project, like a testing environment or review environment, but we are going to focus on development and production for now.

There are lots of tools that aim to make writing code easier for developers. One of these tools, called "sass", we will cover in a future lesson, but for now just know that it is css with some developer friendly syntax and features. It's great that we have tools that make development easier, but if you take sass as an example - we can't run sass on a server. All our sass files have to be run through a transpiler in order to become css that can go on a live webpage. No matter how awesome a development tool is, in the end our code will be judged by how well it runs on a server, and oftentimes what is best for the server is the opposite of what is convenient for developers. So how do we handle both of these environments? By utilizing build tools, we can make code that is convenient for our dev team, without sacrificing speed on the server.

One of the awesome features of webpack, is that it lets us apply configurations to our code based on the environment we are running. We can create a development

environment (MODE in webpack) and run totally different loaders and plugins than we do for production mode.

Now we have learned the second part of why we use build tools. First, we learned that build tools allow devs to use the tools that are more convenient for them. The other side of that coin is that build tools simultaneously allow devs to optimize code for the server. Build tools like Webpack are one tool we can use to help us with organization in all environments. If that doesn’t fully make sense now, don’t worry too much, it will become more clear as we go along.




You can see that a lot of times, what is best for developers is the opposite of what is most efficient for the server. Webpack helps us have the best of both worlds.

### Changes proposed for the Configuration Files

1. Create a copy of the `webpack.config.js`, and rename it as `webpack.prod.js`. This file should have `mode: 'production'` statement in `module.exports`.
2. Now, rename the `webpack.config.js` to `webpack.dev.js`. This file should have the following statements in `module.exports`

```
mode: 'development',
devtool: 'source-map',
```

### Changes proposed for `package.json`

The statements added to the `package.json` for the configuration files of production and development modes separately in the "script" block are:

```
"scripts": {
      "build-prod": "webpack --config webpack.prod.js",
      "build-dev": "webpack-dev-server  --config webpack.dev.js --open"
},
```

Note that you should remove the `"build": "webpack"` script now from `package.json`, and only have the two related to `build-dev` and `build-prod`. This also means when you build your app with npm, you should use the correct script, e.g.

```
npm run build-dev
```

**Mode Quiz**
Which environment is concerned with what happens on a server?

---

production
RESET

**QUESTION 2 OF 4**

## Select the following webpack loaders and plugins that you would expect to find in development mode and not in production mode

- CSS minifiers
- Linters
- An uglify package for javascript
- A source map mode for debugging

SUBMIT

QUESTION 3 OF 4

Which environment has the most direct effect on users?

- Development
- Production

SUBMIT

**Interview Question**
Would you minify files for development or production? Why?

---

**Your reflection**
**For production to reduce load time and enhance speed for user**
**Things to think about**
Some interview questions will feel deceptively easy. That doesn't necessarily mean that the question has an unexpected answer. With a question like this, the interviewer really wants to know that you understand the concepts and differences between development and production. In this question, you can answer by simply saying "production", but you would be missing out. The "why" part is your big opportunity to show just how much you know about the subject. Even if the interviewer does not add the "why" at the end in person. Pretend it's there. Take every opportunity to show your grasp of important concepts.