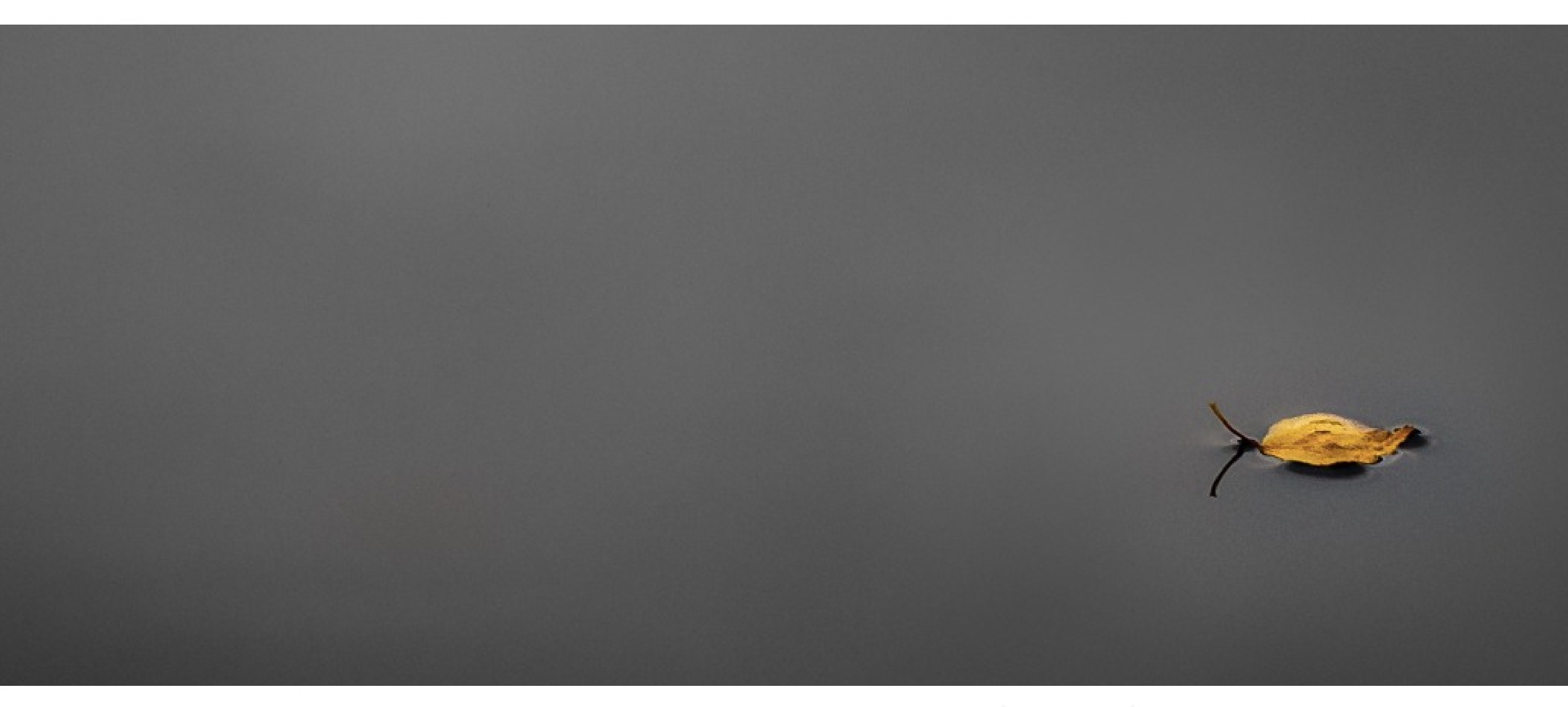
Webpack From Zero to Hero

Chapter 1: Getting Started with the Basics

Rubens Pinheiro Gonçalves Cavalcante Follow
Apr 18 · 3 min read



Nothing — Picture under the Creative Commons License (source: Flickr)

This article is part of the **Webpack from Zero to Hero series**, for more background or for the index you can check the "Chapter 0: History".

Next - **Chapter 2: Tidying Up Webpack.**

> *Disclaimer: During this series I'll be running Yarn, but if you like NPM better it is totally fine too* 😄.

Introduction

In this article we're going to start from an **empty directory** and proceed to build an application with dependencies, producing a simple bundle using Webpack with just a **few lines** of setup!

Starting a project

Open the terminal, create your project directory (or clone a empty repository) and create a new package:

```
yarn init -y
```

This will create your **npm environment.** Then create a simple file with some JS inside of it on **src/index.js**:

```
const hello = subject => console.log(`Hello ${subject}!`);
hello("OLX Dev!");
```

index.js hosted with ❤ by GitHub view raw

Install the dependencies

To install Webpack runtime and client we'll need to do:

```
yarn add webpack webpack-cli --dev
```

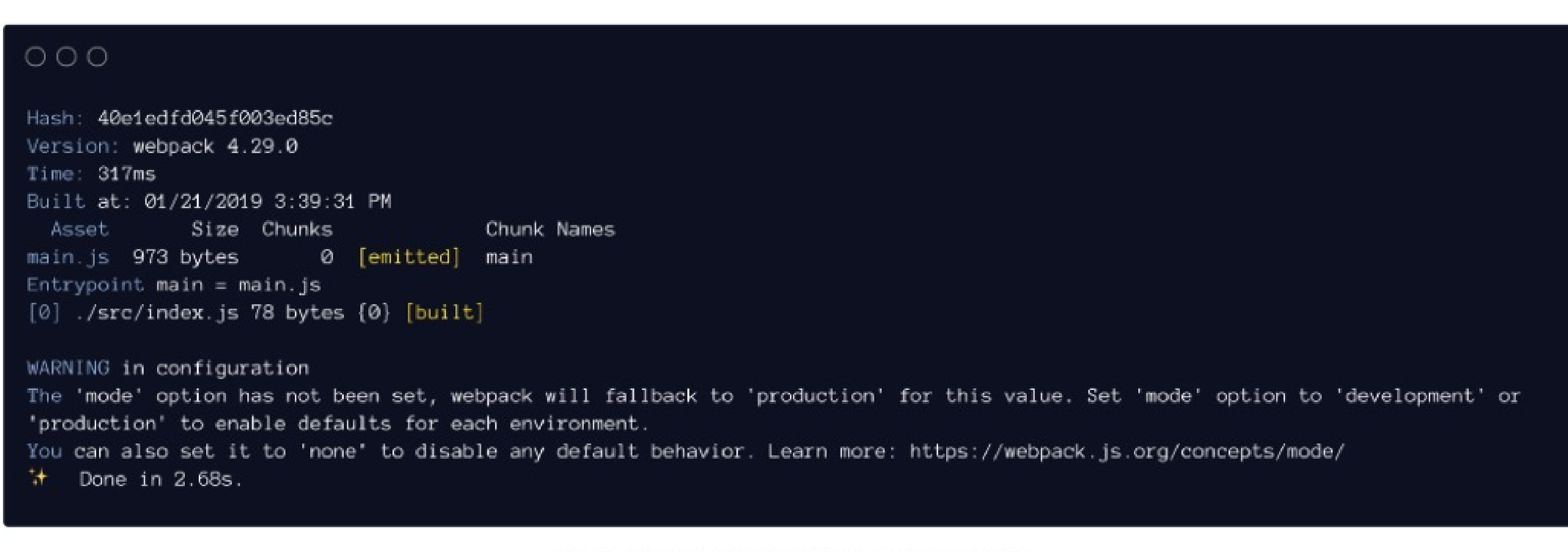
Run the build

Now you're ready to go. Just run:

```
yarn webpack
```

> *If you're running with npm > v5 you can use* ***npx webpack.***

And the result will be something like this:



```
Hash: 40e1edfd045f003ed85c
Version: webpack 4.29.0
Time: 317ms
Built at: 01/21/2019 3:39:31 PM
  Asset       Size  Chunks             Chunk Names
main.js  973 bytes       0  [emitted]  main
Entrypoint main = main.js
[0] ./src/index.js 78 bytes {0} [built]

WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value. Set 'mode' option to 'development' or
'production' to enable defaults for each environment.
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.js.org/concepts/mode/
✨  Done in 2.68s.
```

This is what you will probably see in the console

Note that webpack will **warn** you about not providing the **mode** and it's therefore assuming "production", later we'll understand what this means.

Now we can see the output at *dist/main.js*, and this is the bundle! 😎

> 😡 — *"Hold your horses partner! I don't have a clue what's happening here!"*

Wait a minute, let me explain.

Convention Over Configuration

Following Bundlers like Parcel, Webpack 4 is now designed with this in mind.

You see that we're able to run without any configuration file, and you probably have noticed that it assumed two things:

- The entry file being **src/index.js**

- The output being **dist/main.js**

And more, when it warned about the "**The 'mode' option has not been set**", it's because it assumes a different set of options when you're bundling for *production* and *development* modes.

Running With Different Modes

Let's see what happens when we run with different modes.

Open the *dist/main.js* file when Webpack assumed "production" mode. Now run:

```
yarn webpack --mode development
```

And check *dist/main.js* again. You'll see the **non-uglified unoptimized** version of the bundle.

At the start you can see the *webpack runtime* code, the one responsible for the module resolutions (imports/exports), and then your wrapped code:

```
/***/ "./src/index.js":
/*!**********************!*\
  !*** ./src/index.js ***!
  \**********************/
/*! no static exports found */
/***/ (function(module, exports) {

eval("const hello = subject => console.log(`Hello ${subject}!`);\n\nhello(\"OLX Dev!\");\n\n//#
sourceURL=webpack:///./src/index.js?");

/***/ })
```

In development mode Webpack wraps the modules on eval calls

Module resolution in action

Now move the "hello()" function to another file: *hello.js*:

```

export const hello = subject => console.log(`Hello ${subject}!`);
```

hello.js hosted with ❤ by GitHub view raw

And import it in index:

```
import { hello } from "./hello";
hello("OLX Dev!");
```

index.js hosted with ❤ by GitHub view raw

Now run the dev build again and check the result on the bundle dist/main.js.

Setting up NPM Scripts

To make our life simple, let's move these build commands to the ***package.json*** file. Add this section to it:

```
"scripts": {
  "build": "webpack --mode production",
  "build:dev": "webpack --mode development"
}
```

npm-script.json hosted with ❤ by GitHub view raw

Now we're able to run the build with just:

```
yarn build
```

or

```
yarn build:dev
```

If you're using NPM you need to do **npm run [script-name].**

"YOU LIED TO ME!"

> 😠 — *"When I checked the bundle, all ES6 code was still there! It's not transpiling!"*

Now is my time to say hold your 🐴 🐴, because as I explained in Chapter 0: History, Webpack acts only as a bundler. Its work consists only of **module resolution** and piping these modules through consumers through **loaders**.

The one responsible to *transpile* from ES6/ES7 (or ES2015+ if you prefer) to ES5 is BabelJS:

This is the solution 🔥

. . .

In this chapter we started from an empty directory and created a project with NPM dependencies and scripts already running webpack builds with just a few steps. But it is still a raw setup and we have to support more browsers than just the latest ES2015+ supporting ones. But let's see how to do it properly in the **next chapter**. See you there!

JavaScript Webpack Technology Nodejs Frontend

421 claps