SURVIVEJS

Webpack and React

FROM APPRENTICE TO MASTER

Juho Vepsäläinen

SurviveJS - Webpack and React

From apprentice to master

Juho Vepsäläinen

This book is for sale at http://leanpub.com/survivejs_webpack_react

This version was published on 2016-02-27

ISBN 978-1523910502

Contents

Introduction

Front-end development moves forward fast. A good indication of this is the pace in which new technologies appear to the scene. Webpack[1] and React[2] are two recent newcomers. Combined, these tools allow you to build all sorts of web applications swiftly. Most importantly, learning these tools provides you perspective. That's what this book is about.

What is Webpack?

Web browsers have been designed to consume HTML, JavaScript, and CSS. The simplest way to develop is simply to write files that the browser understands directly. The problem is that this becomes unwieldy eventually. This is particularly true when you are developing web applications.

There are multiple ways to approach this problem. You can start splitting up your JavaScript and CSS to separate files. You could load dependencies through `script` tags. Even though this is better, it is still a little problematic.

If you want to use technologies that compile to these target formats, you will need to introduce preprocessing steps. Task runners, such as Grunt and Gulp, allow you to achieve this, but even then you need to write a lot of configuration by hand.

How Webpack Changes the Situation?

Webpack takes another route. It allows you to treat your project as a dependency graph. You could have an *index.js* in your project that pulls in the dependencies the project needs through standard `import` statements. You can refer to your style files and other assets the same way.

Webpack does all the preprocessing for you and gives you the bundles you specify through configuration. This declarative approach is powerful, but it is a little difficult to learn. However, once you begin to understand how Webpack works, it becomes an indispensable tool. This book has been designed to get through that initial learning curve.

What is React?

Facebook's React, a JavaScript library, is a component based view abstraction. A component could be a form input, button, or any other element in your user interface. This provides an interesting contrast to earlier approaches as React isn't bound to the DOM by design. You can use it to implement mobile applications for example.

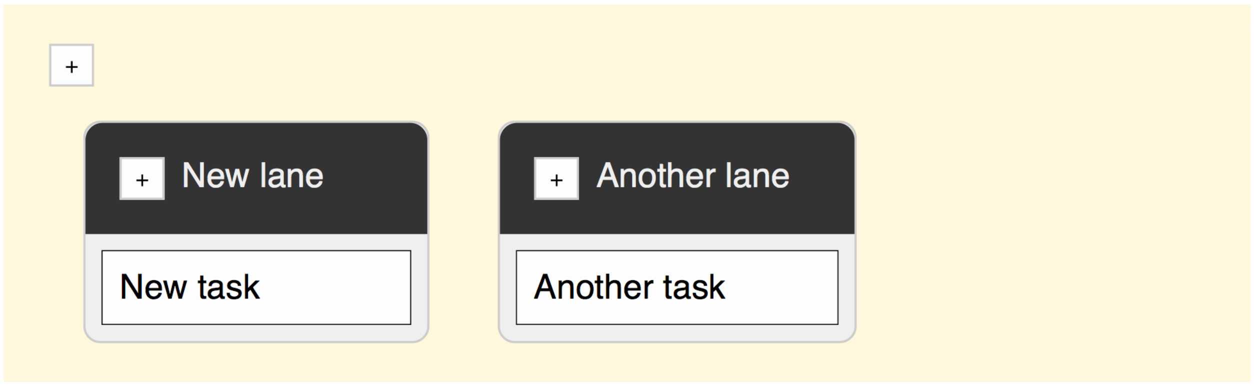
[1]https://webpack.github.io/
[2]https://facebook.github.io/react/

React is Only One Part of the Whole

Given React focuses only on the view, you'll likely have to complement it with other libraries to give you the missing bits. This provides an interesting contrast to framework based approaches as they give you a lot more out of the box. Both approaches have their merits. In this book, we will focus on the library oriented approach.

Ideas introduced by React have influenced the development of the frameworks. Most importantly it has helped us to understand how well component based thinking fits web applications.

What Will You Learn?



Kanban application

This book teaches you to build a Kanban[3] application. Beyond this, more theoretical aspects of web development are discussed. Completing the project gives you a good idea of how to implement something on your own. During the process you will learn why certain libraries are useful and will be able to justify your technology choices better.

How is This Book Organized?

We will start by building a Webpack based configuration. After that, we will develop a small clone of a famous Todo application[4]. This leads us to problems of scaling. Sometimes, you need to do things the dumb way to understand why better solutions are needed after all.

We will generalize from there and put Flux architecture[5] in place. We will apply some Drag and Drop (DnD) magic[6] and start dragging things around. Finally, we will get a production grade build done.

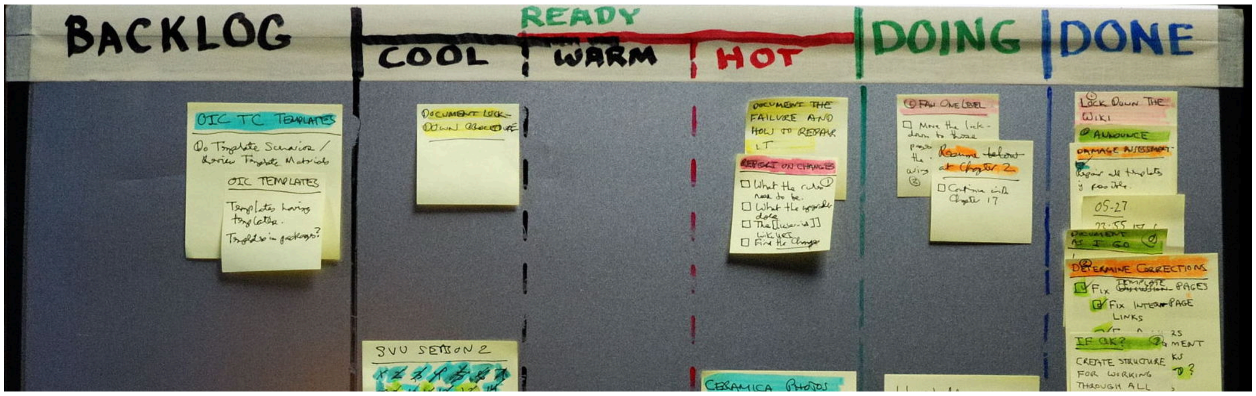
[3]https://en.wikipedia.org/wiki/Kanban
[4]http://todomvc.com/
[5]https://facebook.github.io/flux/docs/overview.html
[6]https://gaearon.github.io/react-dnd/

The final, theoretical part of the book covers more advanced topics. If you are reading the commercial edition of this book, there's something extra in it for you. I will show you how to deal with typing in React in order to produce higher quality code. You will also learn to test your components and logic.

I will also show you how to lint your code effectively using ESLint[7] and various other tools. There is a chapter in which you learn to author libraries at npm[8]. The lessons learned there will come in handy for applications as well. Finally, you will learn to style your React application in various emerging ways.

There are a couple of appendices at end. They are meant to give food for thought and explain aspects, such as language features, in greater detail. If there's a bit of syntax that seems weird to you in the book, you'll likely find more information there.

What is Kanban?



Kanban by Dennis Hamilton (CC BY)

Kanban, originally developed at Toyota, allows you to track the status of tasks. It can be modeled in terms of `Lanes` and `Notes`. `Notes` move through `Lanes` representing stages from left to right as they become completed. `Notes` themselves can contain information about the task itself, its priority, and so on as required.

The system can be extended in various ways. One simple way is to apply a Work In Progress (WIP) limit per lane. The effect of this is that you are forced to focus on getting tasks done. That is one of the good consequences of using Kanban. Moving those notes around is satisfying. As a bonus you get visibility and know what is yet to be done.

Where to Use Kanban?

This system can be used for various purposes, including software and life management. You could use it to track your personal projects or life goals for instance. Even though it's a simple tool, it's

[7]http://eslint.org/

[8]https://www.npmjs.com/

quite powerful, and you can find use for it in many places.

How to Build a Kanban?

The simplest way to build a Kanban is to get a bunch of Post-it notes and find a wall. After that, you split it up into columns. These `Lanes` could consist of the following stages: Todo, Doing, Done. All `Notes` would go to Todo initially. As you begin working on them, you would move them to Doing, and finally, to Done when completed. This is the simplest way to get started.

This is just one example of a lane configuration. The lanes can be configured to match your process. There can be approval steps for instance. If you are modeling a software development process, you could have separate lanes for testing and deployment for instance.

Available Kanban Implementations

Trello[9] is perhaps the most known online implementation of Kanban. Sprintly has open sourced their React implementation of Kanban[10]. Meteor based wekan[11] is another good example. Ours won't be as sophisticated as these, but it will be enough to get started.

Who is This Book for?

I expect that you have a basic knowledge of JavaScript and Node.js. You should be able to use npm on an elementary level. If you know something about Webpack, React, or ES6, that's great. By reading this book you will deepen your understanding of these tools.

One of the hardest things about writing a book is to write it on the right level. Given the book covers a lot of ground, there are appendices that cover basic topics, such as language details, with greater detail than the main content does.

If you find yourself struggling, consider studying the appendices or seeking help from the community around the book. In case you are stuck or don't understand something, we are there to help. Any comments you might have will go towards improving the book content.

How to Approach the Book?

Although a natural way to read a book is to start from the first chapter and then read the chapters sequentially, that's not the only way to approach this book. The chapter order is just a reading suggestion. Depending on your background, you could consider the following orders or even skip some portions altogether:

[9]https://trello.com/
[10]https://github.com/sprintly/sprintly-kanban
[11]https://github.com/wekan/wekan

- From start to end - This would be the traditional way to approach a book. It will also require the most amount of time. But on the plus side you get a steady progression.
- React first, Webpack after - An alternative is to skip the early chapters on Webpack, download a starting point[12] from the repository, and go through the Kanban demonstration first. Follow the Webpack chapters after that to understand what the configuration is doing and why. The *Advanced Techniques* part and appendices complement this content well.
- Webpack only - If you know React very well, maybe it makes sense to go through the Webpack portions only. You can apply the same skills beyond React after all.
- Advanced techniques only - Given React ecosystem is so vast, the *Advanced Techniques* part covers interesting niches you might miss otherwise. Pick up techniques like linting or learn to improve your npm setup. It may be worth your while to dig into various styling approaches discussed to find something that suits your purposes.

The book doesn't cover everything you need to know in order to develop front-end applications. That's simply too much for a single book. I do believe, however, that it might be able to push you to the right direction. The ecosystem around Webpack and React is fairly large and I've done my best to cover a good chunk of it.

Given the book relies on a variety of new language features, I've gathered the most important ones used to a separate *Language Features* appendix that provides a quick look at them. If you want to understand the features in isolation or feel unsure of something, that's a good place to look.

Book Versioning

Given this book receives a fair amount of maintenance and improvements due to the pace of innovation, there's a rough versioning scheme in place. I maintain release notes for each new version at the book blog[13]. That should give you a good idea of what has changed between versions. Also examining the GitHub repository may be beneficial. I recommend using the GitHub *compare* tool for this purpose. Example:

```
https://github.com/survivejs/webpack_react/compare/v1.9.10...v1.9.17
```

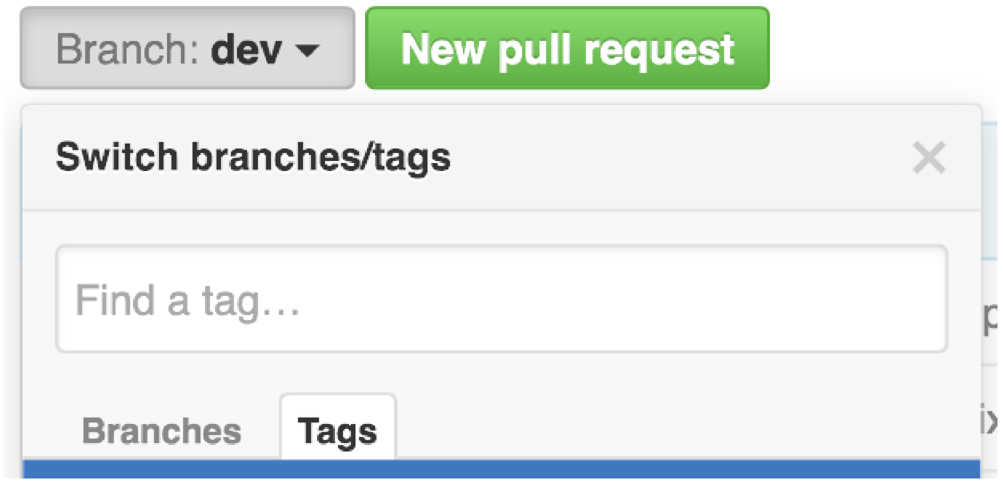
The page will show you the individual commits that went to the project between the given version range. You can also see the lines that have changed in the book. This excludes the private chapters, but it's enough to give you a good idea of the major changes made to the book.

The current version of the book is **2.0.5**.

[12]https://github.com/survivejs/webpack_react/tree/master/project_source/03_webpack_and_react/kanban_app
[13]http://survivejs.com/blog/

Extra Material

The book content and source are available at book's repository at GitHub[14]. Please note that the repository defaults to the `dev` branch of the project. This makes it convenient to contribute. To find source matching the version of the book you are reading, use the tag selector at GitHub's user interface as in the image below:



GitHub tag selector

The book repository contains code per chapter. This means you can start from anywhere you want without having to type it all through yourself. If you are unsure of something, you can always refer to that.

You can find a lot of complementary material at the survivejs organization[15]. Examples of this are alternative implementations of the application available written in mobservable[16], Redux[17], and Cerebral/Baobab[18]. Studying those can give you a good idea of how different architectures work out using the same example.

Getting Support

As no book is perfect, you will likely come by issues and might have some questions related to the content. There are a couple of options to deal with this:

- Contact me through GitHub Issue Tracker[19]

[14]https://github.com/survivejs/webpack_react
[15]https://github.com/survivejs/
[16]https://github.com/survivejs/mobservable-demo
[17]https://github.com/survivejs/redux-demo
[18]https://github.com/survivejs/cerebral-demo
[19]https://github.com/survivejs/webpack_react/issues

- Join me at Gitter Chat[20]
- Follow @survivejs[21] at Twitter for official news or poke me through @bebraw[22] directly
- Send me email at info@survivejs.com[23]
- Ask me anything about Webpack or React at SurviveJS AmA[24]

If you post questions to Stack Overflow, tag them using **survivejs** so I will get notified of them. You can use the hashtag **#survivejs** at Twitter for same effect.

I have tried to cover some common issues at the *Troubleshooting* appendix. That will be expanded as common problems are found.

Announcements

I announce SurviveJS related news through a couple of channels:

- Mailing list[25]
- Twitter[26]
- Blog RSS[27]

Feel free to subscribe.

Acknowledgments

An effort like this wouldn't be possible without community support. There are a lot of people to thank as a result!

Big thanks to Christian Alfoni[28] for starting the react-webpack-cookbook[29] with me. That work eventually lead to this book.

The book wouldn't be half as good as it is without patient editing and feedback by my editor Jesús Rodríguez Rodríguez[30]. Thank you.

[20]https://gitter.im/survivejs/webpack_react
[21]https://twitter.com/survivejs
[22]https://twitter.com/bebraw
[23]mailto:info@survivejs.com
[24]https://github.com/survivejs/ama/issues
[25]http://eepurl.com/bth1v5
[26]https://twitter.com/survivejs
[27]http://survivejs.com/atom.xml
[28]http://www.christianalfoni.com/
[29]https://github.com/christianalfoni/react-webpack-cookbook
[30]https://github.com/Foxandxss

Special thanks to Steve Piercy for numerous contributions. Thanks to Prospect One[31] and Dixon & Moe[32] for helping with the logo and graphical outlook. Thanks for proofreading to Ava Mallory and EditorNancy from fiverr.com.

Numerous individuals have provided support and feedback along the way. Thank you in no particular order Vitaliy Kotov, @af7, Dan Abramov, @dnmd, James Cavanaugh, Josh Perez, Nicholas C. Zakas, Ilya Volodin, Jan Nicklas, Daniel de la Cruz, Robert Smith, Andreas Eldh, Brandon Tilley, Braden Evans, Daniele Zannotti, Partick Forringer, Rafael Xavier de Souza, Dennis Bunskoek, Ross Mackay, Jimmy Jia, Michael Bodnarchuk, Ronald Borman, Guy Ellis, Mark Penner, Cory House, Sander Wapstra, Nick Ostrovsky, Oleg Chiruhin, Matt Brookes, Devin Pastoor, Yoni Weisbrod, Guyon Moree, Wilson Mock, Herryanto Siatono, Héctor Cascos, Erick Bazán, Fabio Bedini, Gunnari Auvinen, Aaron McLeod, John Nguyen, Hasitha Liyanage, Mark Holmes, Brandon Dail, Ahmed Kamal, Jordan Harband, Michel Weststrate, Ives van Hoorne, Luca DeCaprio, @dev4Fun, Fernando Montoya, Hu Ming, @mpr0xy, David "@davegomez" Gómez, Aleksey Guryanov, Elio D'antoni, Yosi Taguri, Ed McPadden, Wayne Maurer, Adam Beck, Omid Hezaveh, Connor Lay, Nathan Grey, Avishay Orpaz, Jax Cavalera, Juan Diego Hernández, Peter Poulsen, Harro van der Klauw, Tyler Anton, Michael Kelley, @xuyuanme, @RogerSep, Jonathan Davis, @snowyplover, Tobias Koppers, Diego Toro, George Hilios, Jim Alateras, @atleb, Andy Klimczak, James Anaipakos, Christian Hettlage, Sergey Lukin, Matthew Toledo, Talha Mansoor, Pawel Chojnacki, @eMerzh, Gary Robinson, Omar van Galen, Jan Van Bruggen, Savio van Hoi, Alex Shepard, Derek Smith, and Tetsushi Omi. If I'm missing your name, I might have forgotten to add it.

[31]http://prospectone.pl/

[32]http://dixonandmoe.com/

I Setting Up Webpack

Webpack is a powerful module bundler. It hides a lot of power behind configuration. Once you understand its fundamentals, it becomes much easier to use this power. Initially, it can be a confusing tool to adopt, but once you break the ice, it gets better.

In this part, we will develop a Webpack based project configuration that provides a solid foundation for the Kanban project and React development overall.

1. Webpack Compared

You can understand better why Webpack's approach is powerful by putting it into historical context. Back in the day, it was enough just to concatenate some scripts together. Times have changed, though, and now distributing your JavaScript code can be a complex endeavor.

1.1 The Rise of the SPAs

This problem has escalated with the rise of single page applications (SPAs). They tend to rely on numerous hefty libraries. The last thing you want to do is to load them all at once. There are better solutions, and Webpack works with many of those.

The popularity of Node.js and npm[1], the Node.js package manager, provides more context. Before npm it was difficult to consume dependencies. Now that npm has become popular for front-end development, the situation has changed. Dependency management is far easier than earlier.

1.2 Task Runners and Bundlers

Historically speaking, there have been many build systems. Make[2] is perhaps the best known, and is still a viable option. To make things easier, specialized *task runners*, such as Grunt[3] and Gulp[4] appeared. Plugins available through npm made both task runners powerful.

Task runners are great tools on a high level. They allow you to perform operations in a cross-platform manner. The problems begin when you need to splice various assets together and produce bundles. This is the reason we have *bundlers*, such as Browserify[5] or Webpack[6].

Continuing further on this path, JSPM[7] pushes package management directly to the browser. It relies on System.js[8], a dynamic module loader. Unlike Browserify and Webpack, it skips the bundling step altogether during development. You can generate a production bundle using it, however. Glen Maddern goes into good detail at his video about JSPM[9].

[1] https://www.npmjs.com/
[2] https://en.wikipedia.org/wiki/Make_%28software%29
[3] http://gruntjs.com/
[4] http://gulpjs.com/
[5] http://browserify.org/
[6] https://webpack.github.io/
[7] http://jspm.io/
[8] https://github.com/systemjs/systemjs
[9] https://www.youtube.com/watch?t=33&v=iukBMY4apvI

1.3 Make

You could say Make goes way back. It was initially released in 1977. Even though it's an old tool, it has remained relevant. Make allows you to write separate tasks for various purposes. For instance, you might have separate tasks for creating a production build, minifying your JavaScript or running tests. You can find the same idea in many other tools.

Even though Make is mostly used with C projects, it's not tied to it in any way. James Coglan discusses in detail how to use Make with JavaScript[10]. Consider the abbreviated code based on James' post below:

Makefile

```
PATH  := node_modules/.bin:$(PATH)
SHELL := /bin/bash

source_files := $(wildcard lib/*.coffee)
build_files  := $(source_files:%.coffee=build/%.js)
app_bundle   := build/app.js
spec_coffee  := $(wildcard spec/*.coffee)
spec_js      := $(spec_coffee:%.coffee=build/%.js)

libraries    := vendor/jquery.js

.PHONY: all clean test

all: $(app_bundle)

build/%.js: %.coffee
    coffee -co $(dir $@) $<

$(app_bundle): $(libraries) $(build_files)
    uglifyjs -cmo $@ $^

test: $(app_bundle) $(spec_js)
    phantomjs phantom.js

clean:
    rm -rf build
```

With Make, you model your tasks using Make-specific syntax and terminal commands. This allows it to integrate easily with Webpack.

[10] https://blog.jcoglan.com/2014/02/05/building-javascript-projects-with-make/

1.4 Grunt

Grunt

Grunt went mainstream before Gulp. Its plugin architecture, especially, contributed towards its popularity. At the same time, this architecture is the Achilles' heel of Grunt. I know from experience that you **don't** want to end up having to maintain a 300-line *Gruntfile*. Here's an example from Grunt documentation[11]:

```
module.exports = function(grunt) {
  grunt.initConfig({
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        globals: {
          jQuery: true
        }
      }
    },
    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');
```

[11]http://gruntjs.com/sample-gruntfile

```
  grunt.registerTask('default', ['jshint']);
};
```

In this sample, we define two basic tasks related to *jshint*, a linting tool that locates possible problem spots in your JavaScript source code. We have a standalone task for running jshint. Also, we have a watcher based task. When we run Grunt, we'll get warnings in real-time in our terminal as we edit and save our source code.

In practice, you would have many small tasks for various purposes, such as building the project. The example shows how these tasks are constructed. An important part of the power of Grunt is that it hides a lot of the wiring from you. Taken too far, this can get problematic, though. It can become hard to thoroughly understand what's going on under the hood.

Note that the grunt-webpack[12] plugin allows you to use Webpack in a Grunt environment. You can leave the heavy lifting to Webpack.

1.5 Gulp

Gulp

Gulp takes a different approach. Instead of relying on configuration per plugin, you deal with actual code. Gulp builds on top of the tried and true concept of piping. If you are familiar with Unix, it's the same idea here. You simply have sources, filters, and sinks.

Sources match to files. Filters perform operations on sources (e.g., convert to JavaScript). Finally, the results get passed to sinks (e.g., your build directory). Here's a sample *Gulpfile* to give you a better idea of the approach, taken from the project's README. It has been abbreviated a bit:

[12]https://www.npmjs.com/package/grunt-webpack

```
var gulp = require('gulp');
var coffee = require('gulp-coffee');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
var sourcemaps = require('gulp-sourcemaps');
var del = require('del');

var paths = {
    scripts: ['client/js/**/*.coffee', '!client/external/**/*.coffee']
};

// Not all tasks need to use streams
// A gulpfile is just another node program and you can use all packages availabl\
e on npm
gulp.task('clean', function(cb) {
  // You can use multiple globbing patterns as you would with `gulp.src`
  del(['build'], cb);
});

gulp.task('scripts', ['clean'], function() {
  // Minify and copy all JavaScript (except vendor scripts)
  // with sourcemaps all the way down
  return gulp.src(paths.scripts)
    .pipe(sourcemaps.init())
      .pipe(coffee())
      .pipe(uglify())
      .pipe(concat('all.min.js'))
    .pipe(sourcemaps.write())
    .pipe(gulp.dest('build/js'));
});

// Rerun the task when a file changes
gulp.task('watch', function() {
  gulp.watch(paths.scripts, ['scripts']);
});

// The default task (called when you run `gulp` from CLI)
gulp.task('default', ['watch', 'scripts']);
```

Given the configuration is code, you can always just hack it if you run into troubles. You can wrap existing Node.js packages as Gulp plugins, and so on. Compared to Grunt, you have a clearer idea of what's going on. You still end up writing a lot of boilerplate for casual tasks, though. That is where some newer approaches come in.

gulp-webpack[13] allows you to use Webpack in a Gulp environment.

1.6 Browserify

Browserify

Dealing with JavaScript modules has always been a bit of a problem. The language itself actually didn't have the concept of modules till ES6. Ergo, we have been stuck in the '90s when it comes to browser environments. Various solutions, including AMD[14], have been proposed.

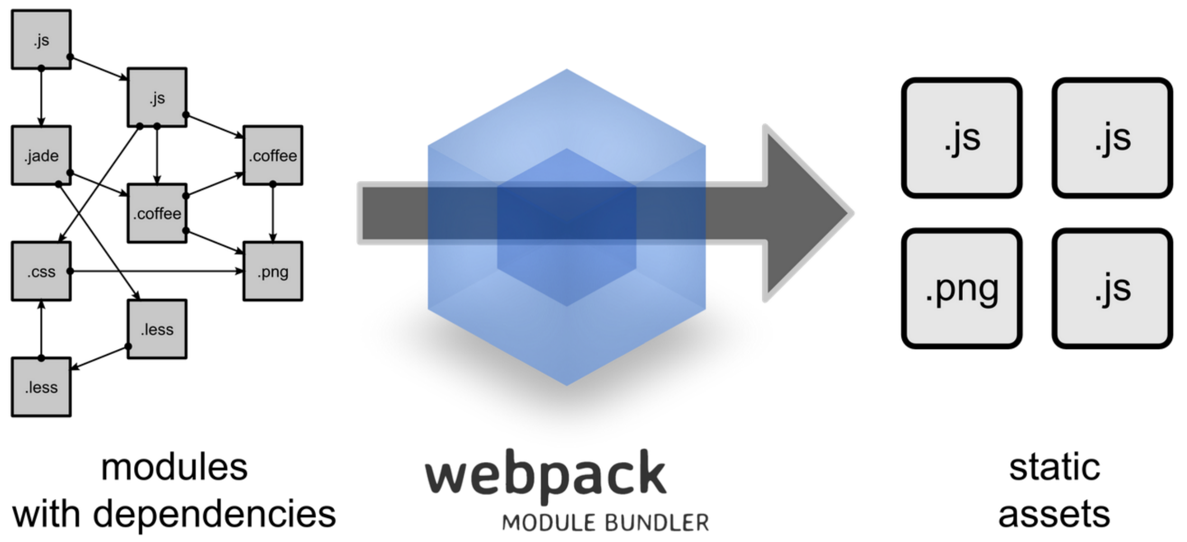
In practice, it can be useful just to use CommonJS, the Node.js format, and let the tooling deal with the rest. The advantage is that you can often hook into npm and avoid reinventing the wheel.

Browserify[15] is one solution to the module problem. It provides a way to bundle CommonJS modules together. You can hook it up with Gulp. There are smaller transformation tools that allow you to move beyond the basic usage. For example, watchify[16] provides a file watcher that creates bundles for you during development. This will save some effort and no doubt is a good solution up to a point.

[13]https://www.npmjs.com/package/gulp-webpack
[14]http://requirejs.org/docs/whyamd.html
[15]http://browserify.org/
[16]https://www.npmjs.com/package/watchify

The Browserify ecosystem is composed of a lot of small modules. In this way, Browserify adheres to the Unix philosophy. Browserify is a little easier to adopt than Webpack, and is, in fact, a good alternative to it.

1.7 Webpack



webpack

You could say Webpack (or just *webpack*) takes a more monolithic approach than Browserify. Whereas Browserify consists of multiple small tools, Webpack comes with a core that provides a lot of functionality out of the box. The core can be extended using specific *loaders* and *plugins*.

Webpack will traverse through the `require` statements of your project and will generate the bundles you have defined. You can even load your dependencies in a dynamic manner using a custom `require.ensure` statement. The loader mechanism works for CSS as well and `@import` is supported. There are also plugins for specific tasks, such as minification, localization, hot loading, and so on.

To give you an example, `require('style!css!./main.css')` loads the contents of *main.css* and processes it through CSS and style loaders from right to left. Given that declarations, such as this, tie the source code to Webpack, it is preferable to set up the loaders at Webpack configuration. Here is a sample configuration adapted from the official webpack tutorial[17]:

webpack.config.js

[17]http://webpack.github.io/docs/tutorials/getting-started/

```
var webpack = require('webpack');

module.exports = {
  entry: './entry.js',
  output: {
    path: __dirname,
    filename: 'bundle.js'
  },
  module: {
    loaders: [
      {
        test: /\.css$/,
        loaders: ['style', 'css']
      }
    ]
  },
  plugins: [
    new webpack.optimize.UglifyJsPlugin()
  ]
};
```

Given the configuration is written in JavaScript, it's quite malleable. As long as it's JavaScript, Webpack is fine with it.

The configuration model may make Webpack feel a bit opaque at times. It can be difficult to understand what it's doing. This is particularly true for more complicated cases. I have compiled a webpack cookbook[18] with Christian Alfoni that goes into more detail when it comes to specific problems.

[18]https://christianalfoni.github.io/react-webpack-cookbook/