# Why Parcel Has Become My Go-To Bundler for Development

Author
Mats Mazoumi
12 Comments
Join the Conversation →
Published
Oct 21, 2019
Updated
Oct 21, 2019

VALIDO
Build custom online forms in minutes
Start free
Multi-page and one-page forms, payments, file uploads, and more. No code or coding required.

In depth front-end engineering courses

DIGITAL OCEAN · DYNOLUNE · PAINTER · PROFESSIONALFEEDS

Deploy your apps with DigitalOcean's Cloning Lab and Droplet Snapshots. Get a free trial of DigitalOcean and receive $100 in free credits.

FAUNA
If JavaScript, then FaunaDB.
Start building →
Fauna's flexible, cloud native database, GraphQL, transactions.

Master Writing Professional and Modern JavaScript
Frontend Masters
Frontend Masters is the best place for advanced JavaScript training. Learn from the best instructors with Frontend Masters.

Project management is now cheap!
Nuclino
Nuclino's fast, secure wiki, documents, and project management tool. Collaborate on anything. Free for teams of up to 5. Unlimited for individuals.

## What makes Parcel special

The thing I love the most about this bundler: **it doesn't need any configuration**. Literally, none at all! Compare that to webpack, where configuration can be strewn across several files all containing tons of code, that you may have picked up from other people's configurations or inherited from other projects. Sure, configuration is only as complex as you make it, but even a modest webpack configuration can be quite unwieldy.

We all use different tools to simplify our job. There are things like preprocessors, post-processors, compilers, transpilers, etc. It takes time to set these up, and often a pretty decent amount of it. Wouldn't you rather spend that time developing?

That's why Parcel seems a good solution. Want to write your styles in SCSS or LESS? Go it! Want to use the latest JavaScript syntax? Included. Need a server for development? You got it. That's barely scratching the surface of the large list of other features it supports.

Parcel allows you to simply start developing. That's the biggest advantage of using it as a bundler — alongside its blazing fast compilation thanks that utilizes multicore processing where other bundlers, including webpack, work off of complex and heavy transforms.

## Where using Parcel makes sense

Parcel, like any tool, is not a golden pill that's designed as a one-size-fits-all solution for everything. It has use cases where it shines most.

I've already mentioned how Parcel is great for a project up and running. That makes it ideal when working with light deadlines and prototypes, where time is precious and the goal is to get in the browser as quickly as possible.

That's not to say it isn't up to the task of handling complex applications or projects where lots of developers might be touching code. It's very capable of that. However, I realize that those projects may vary and benefit from more granular control.

It's sort of like the difference between driving a car with an automatic transmission versus a stick shift. Sometimes you need the additional control and sensitivities you don't.

I've been working on a commercial multi-page website with a bunch of JavaScript under the hood, and Parcel is working out very well for me. It's providing my server, it compiles my Sass to CSS, it adds vendor prefixes when needed, and it allows me to use import and export in my JavaScript files out of the box without any configuration. All of this allowed me to get my project up and running with ease.

## Let's create a simple site together using Parcel

Let's take Parcel for a test drive to see how relatively simple it is to make something with it.

We're going to build a simple page that uses Sass and a bit of JavaScript. We'll fetch the current day of the week and a random image from Unsplash Source.

## The basic structure

There's no scaffolding we're required to use or framework needed to initialize our project. Instead, we're going to make three files that they ought to look super familiar: `index.html`, `style.scss` and `index.js`. You can set that up manually or in Terminal:

```
Terminal
mkdir simple-site
cd simple-site
touch index.html js/main.js styles/main.scss
```

Let's sprinkle some boilerplate markup and the basic outline into our HTML file:

```
index
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link id="favicon"rel="shortcut icon"type="image/x-icon"href="data:image/svg+xml,<svg xmlns='http://www.w3.org/2000/svg'viewBox='0 0 100 100'><text y='.90em'fontsize='90'>🌱</text></svg>">
    <title>Parcel Tutorial</title>
    <link rel="stylesheet"href="./styles/main.scss">
  </head>
  <body>
    <div id="today">Today Is:
      <span id="day">Day</span>
    </div>
    <div id="image">and the image of the day is:
      <br />
      <img id="image"src="https://source.unsplash.com/random/600x400"alt="unsplash random image">
    </div>
    <script src="./dist/main.js"></script>
  </body>
</html>
```

You may have noticed that I'm pulling in a web font (Lato) from Google, which is totally optional. Otherwise, we're doing is linking up the CSS and JavaScript files and dropping in the basic HTML that will display the day of the week and a link from Unsplash that will serve a random image. This is all we really need to our baseline.

### Marvel at Parcel's quick set-up!

Let's run the application using with Parcel as the bundler before we get into styling and scripts. Installing Parcel is like any thing:

```
Terminal
npm install -g parcel-bundler
# or
yarn global add parcel-bundler
```

Let's also create a `package.json` file should we need any development dependencies. This is also where Parcel will include anything it needs to work out of the box:

```
Terminal
npm init -y
# or
yarn init -y
```

That's it! No more configuration! We only need to tell Parcel which file is the entry point for the project so it knows where to point its server. That's going to be our HTML file:

```
Terminal
parcel index.html
```

If we open the console we'll see something like this indicating that the server is already running:

```
Terminal
Server running at http://localhost:1234
```

Parcel's server supports hot reloading and rebuilds the app as changes are saved.

Now, heading back to our project folder, we'll see additional stuff that Parcel created for us:

```
.cache
dist
  index.html
  parcel-bundler-tutorial.6bdf8f45.js
  parcel-bundler-tutorial.6bdf8f45.js.map
  style.87fb1538.css
  style.87fb1538.css.map
  style.87fb1538.js
  style.87fb1538.js.map
```

What's important for us here is the `dist` folder, which contains all our compiled code, including source maps for CSS and JavaScript.

### Now all we do is build!

Let's go to `styles.scss` and see how Parcel handles Sass. I've created variables to store some colors and a width for the container that holds our content:

```
styles
$container-size: 700px;
$bg: #ffffff;
$text: #000000;
$primary-color: #ff9f00;
```

Now for a little styling, including some nesting nested rulesets. You can do your own thing, of course, but here's what I cooked up for demo purposes:

```
styles
a, a:visited, a:active {
  text-styling: underline;
}

body {
  background: $bg;
  color: $text;
  font-family: 'Lato', sans-serif;
  margin: 0;
  padding: 0;
}

.container {
  margin: 0 auto;
  max-width: $container-size;
  text-align: center;
}

.a {
  display: inline-block;
  text-size: 20px;
}

.open {
  color: $primary-yellow;
  text-size: 30px;
  margin-left: 10px;
}
```

Once we save, Parcel's magic is triggered and everything compiles and reloads in the browser for us. No command needed because it's already watching the files for changes.

This is what we've got so far:

The only thing left is to show the current day of the week. We're going to use imports and exports so we get to see how Parcel allows us to use modern JavaScript.

Let's create a file called `today.js` and include a function that reports the current day of the week from an array of days:

```
main
export function getDay() {
  const days = new Date();
  const daysArr = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Sat
  return daysArr[days.getDay()];
}
```

NOTE

It's worth a note to remember that the `getDay` function returns Sunday as the first day of the week.

Notice we're exporting the `getDay` function. Let's go into our `index.js` file and import it there so it gets included when compiling happens:

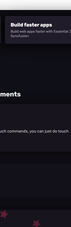
```
main
import { getDay } from \./today';
```

We can import/export files, because Parcel supports ES6 module syntax right out of the box — again, no configuration needed!

The only thing left is to select the `<span>` element and pass the value of the `getDay` function to it:

```
main
const day = document.querySelector('.today');
day.innerHTML = getDay();
```

Let's see the final result:




### Last thing is to build for production

We've created the app, but we want to serve it somewhere — whether it's your personal server or some zero-configuration deployment like Surge or Now — and we want to serve compiled and minified code.

Here's the one and only command we need:

```
Terminal
parcel build index.js
```

```
✨ Built in 640ms.

dist/style.5a41ad8f.fonts.svg   1.98 KB    24ms
dist/style.1240a813.svg         2.5 B     5ms
dist/parcel-bundler-tutorial.6bdf8f45.js.map   223 B     2ms
dist/index.html                 193 B     2ms
dist/style.1240a813.css         153 B     13ms
```

This gives us all of the production-ready assets for the app. You can read more about Parcel's product mode for some tips and tricks to get the most from your environment.

I've said it several times and I'll say it again: Parcel is a great tool. It bundles, it compiles, it serves, it lints — and post-processes, if minifies and uglifies, and more. We may have looked at a pretty simple example, but hopefully you now have a decent feel for what Parcel offers and how you might start to use it in your own projects.

I'm interested if you've already using Parcel and, if so, how you've been using it. Have you found it works better for some things more than others? Did you discover some neat trick that makes it even more powerful? Let me know in the comments!

The Divi Theme
Download the Ultimate WordPress Theme including our advanced Drag & Drop builder.

Build faster apps
Build web apps faster with Essential 26 by Sandbaxx.
Sandbaxx
Learn more →

## Comments

**Evan**
October 21, 2019

In the beginning, instead of chaining multiple touch commands, you can just do touch index.html style.scss index.js

Reply

Evilfers
October 21, 2019
Cool article

Reply

**Chris Pamaytoff**
October 22, 2019

The parcel build command does not reduce source maps, so here is some simple npm scripts I get started

```
"scripts": {
  "prebuild": "rm -rf dist",
  "build": "parcel build index.html --no-source-maps",
  "serve": "parcel index.html --open"
}
```

I use Parcel for quick prototyping, it is really good for this, but misses critical functionality like copying assets, proxy ( as I not reloading wordpress site etc ), SVG, great bundler for small projects.