Production ready Electron app using React and Parcel web bundler.

Yogesh Kumar Follow
Jan 15 · 3 min read

There are a few examples out there to build an Electron app using create-react-app. But, ever since I came across Parcel, it just blew me off. Parcel is blazingly fast web application bundler, since it uses worker processes to enable multicore compilation, and has a filesystem cache for fast rebuilds even after a restart.

Let's begin.

```
yarn init -y
```

Let's get dependencies out of the way, will explain relevant dependencies later.

```
yarn add react react-dom electron-is-dev semantic-ui-css

yarn add -D concurrently cross-env electron electron-builder parcel-
bundler wait-on
```

Babel dependencies.

```
yarn add -D babel-core babel-plugin-transform-object-rest-spread
babel-plugin-transform-react-jsx babel-preset-env babel-preset-react
```

1. Creating a React App

Create *index.html* file in root directory.

```
<!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" />
    <title>Electron React Parcel</title>
    <link
      rel="shortcut icon"
      href="./src/assets/logo.ico"
      type="image/x-icon"
    />
  </head>
  <body>
    <div id="root"></div>
    <script src="./src/index.js"></script>
  </body>
</html>
```

index.html hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/blob/master/index.html

Create a source directory, *src/*

Inside src/, create index.js and App.js

```
import React from "react";
import ReactDOM from "react-dom";
import "semantic-ui-css/semantic.min.css"; // optional
import App from "./App";

```

```
6  ReactDOM.render(<App />, document.getElementById("root"));
```

index.js hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/blob/master/src/index.js

```
1  import React, { Component } from "react";
2  import Navbar from "./components/Navbar";
3  import Custom from "./components/Custom";
4  import "./custom.css";
5  class App extends Component {
6    render() {
7      return (
8        <div>
9          <Navbar />
10         <Custom />
11       </div>
12     );
13   }
14 }
15 export default App;
```

App.js hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/blob/master/src/App.js

Add some custom components inside src/components

```
1  import React from "react";
2
3  const Custom = () => {
4    return (
5      <div className="ui container">
6        <h3>
7          React{" "}
8          <span>
9            <img
10             src="https://cdn4.iconfinder.com/data/icons/logos-3/600/React.js_logo-32.png
11             width="32"
12           />
13         </span>
14         + Parcel{" "}
15         <span>
16           <img src="https://parceljs.org/assets/parcel@2x.png" width="32" />{" "}
17         </span>
18         + Electron{" "}
```

```
          <span>
            <img src="https://electronjs.org/images/favicon.ico" width="32" />
          </span>{" "}
          = Awesomeness ! 🎉
        </h3>
      </div>
    );
  };

  export default Custom;
```

Custom.js hosted with ❤ by **GitHub** **view raw**

```
import React from "react";

const Navbar = () => {
  return (
    <div className="ui secondary menu">
      <div className="header item">Our Company</div>
      <a className="right item">Products</a>
      <a className="item">Locations</a>
      <a className="item">About Us</a>
    </div>
  );
};

export default Navbar;
```

Navbar.js hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/tree/master/src/components

Now, add some babel presets, create *.babelrc* in root directory. This will take care of React jsx,

```
{
  "presets": ["env", "react"],
  "plugins": ["transform-object-rest-spread", "transform-react-jsx"]
}
```

.babelrc hosted with ❤ by **GitHub** **view raw**

Time to add some scripts in package.json

```
"scripts": {
  "react-start": "parcel -p 3000 index.html",

  "react-build": "parcel build index.html"
}
```

Here, “parcel -p 3000” will override parcel’s default port 1234 to port 3000.

2. Adding Electron

Create *electron.js* in src directory.

```
const electron = require("electron");
const app = electron.app;

const BrowserWindow = electron.BrowserWindow;

const path = require("path");
const isDev = require("electron-is-dev");

let mainWindow;

function createWindow() {
  mainWindow = new BrowserWindow({
    width: 900,
    height: 680
  });

  mainWindow.loadURL(
    isDev
      ? "http://localhost:3000"
      : `file://${path.join(__dirname, "../build/index.html")}`
  );
  mainWindow.on("closed", () => (mainWindow = null));
}

app.on("ready", createWindow);

app.on("window-all-closed", () => {
  if (process.platform !== "darwin") {
    app.quit();
  }
});
```

```
32
33  app.on("activate", () => {
34    if (mainWindow === null) {
35      createWindow();
36    }
37  });
```

electron.js hosted with ❤ by **GitHub** **view raw**

See, we are adding electron.js in src/ directory, so that it gets copied to the “build” folder.

Modify the “main” property to package.json and point it to the electron file:

```
"main": "src/electron.js"
```

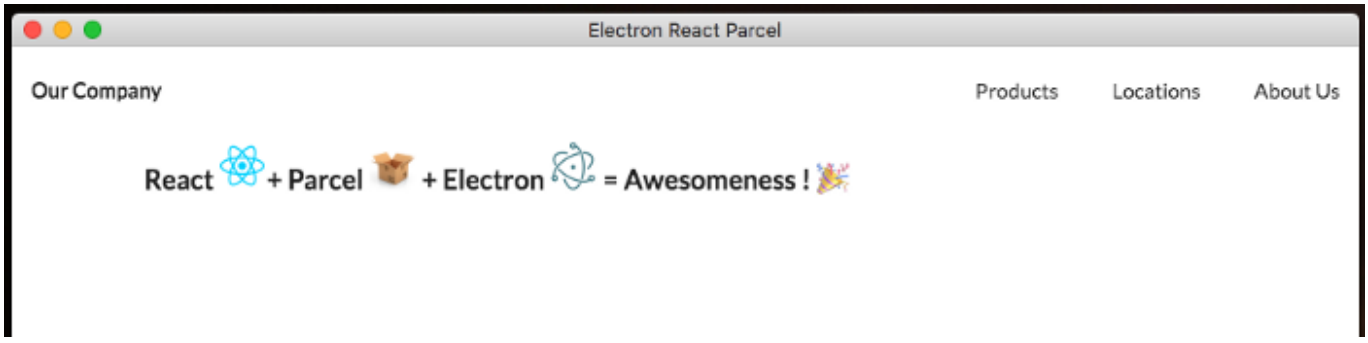
To start Electron app, we now need to modify scripts property in package.json add add following script.

```
"start": "concurrently \"cross-env BROWSER=none yarn react-start\"
\"wait-on http://localhost:3000 && electron . \""
```

This will first start React app on port 3000 and before starting the Electron app.

Now run, and boom!

```
yarn start
```



Pretty neat!

The power of Parcel

Check the build time,

```
yarn run v1.10.1
$ concurrently "cross-env BROWSER=none yarn react-start" "wait-on http
://localhost:3000 && electron . "
$ parcel -p 3000 index.html --out-dir build
[0] Server running at http://localhost:3000
✨  Built in 6.57s.
```

6.57 seconds is **fast**. The subsequent restarts will be much faster as parcel uses .cache to rebuilt.

3. Building for production

Here's the tricky part.

Both parcel-bundler and electron-builder create dist/ folder to create build files. In order to separate React app bundle and electron app packages, we need to override parcel's default "dist" directory to "build" by adding " — out-dir build" to scripts.

```
"scripts" : {
 "react-start": "parcel -p 3000 index.html --out-dir build",
 "react-build": "parcel build index.html --out-dir build --public-url
./"
}
```

Add following “build” property to package.json and point electron-builder to the correct folders.

```
"build": {
  "appId": "com.boskysoft",
  "files": [
     "src/electron.js",
     "src/assets/*",
     "build/**/*"
  ]
}
```

The above step is important. The “files” property will include bundled React app present in “build” folder to “dist” folder of electron packager.

Now modify “scripts” property in package.json to include build scripts

```
"electron-build": "electron-builder ",
"build": "yarn clean-build && yarn react-build && yarn electron-
build",
```

4. Multi-platform support

In order to build desktop apps for macOS, Windows and Linux (debian) add following properties to “build” property in package.json

```
"mac": {
  "target": "dmg",
  "icon": "src/assets/logo.png"
 },

"win": {
  "target": "nsis",
```

```
    "icon": "src/assets/logo.png"
   },

  "linux": {
    "target": "deb",
    "icon": "src/assets/logo.png",
    "category": "Development"
  }
```

The final “build” property should look like this:

```
 "build": {
     "appId": "com.boskysoft",
     "files": [
       "src/electron.js",
       "src/assets/*",
       "build/**/*"
     ],
     "mac": {
       "target": "dmg",
       "icon": "src/assets/logo.png"
     },
     "win": {
       "target": "nsis",
       "icon": "src/assets/logo.png"
     },
     "linux": {
       "target": "deb",
       "icon": "src/assets/logo.png",
       "category": "Development"
     }
   }
```

build hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/blob/master/package.json

Modify “electron-build” script in “scripts” property to support multi-platform compilation.

```
"electron-build": "electron-builder -mwl"
```

"electron-builder -mwl" will compile for

m : macOs, w: “Windows” and l : “Linux”

The final “scripts” property should look like this:

```
"scripts": {
    "react-start": "parcel -p 3000 index.html --out-dir build",
    "react-build": "parcel build index.html --out-dir build --public-url ./",
    "electron-build": "electron-builder -mwl",
    "clean-build": "rm -rf build/ .cache dist/",
    "build": "yarn clean-build && yarn react-build && yarn electron-build",
    "start": "concurrently \"cross-env BROWSER=none yarn react-start\" \"wait-on http://l
}
```

scripts hosted with ❤ by **GitHub** **view raw**

https://github.com/kumarryogeshh/electron-react-parcel-boilerplate/blob/master/package.json

Finally, add “homepage” in the package.json, otherwise the packaged app won’t find the .js and .css files.

```
"homepage": "./"
```

Complete code can be found on my GitHub repo here.

Thanks for giving your time. Please do clap to support. Do check the repo.

Let me know in comments section.

. . .

Please follow me on LinkedIn and Twitter.

If you, or someone you know is interested in having as for consultation, check out our website.

React Electron Parcel Js

About Help Legal