使用 Webpack 創建 React 應用程序

jsramblings.com/creating-a-react-app-with-webpack

2022年6月6日

2022 年 10 月 11 日 • 6 分鐘閱讀
有時你想開始使用 React 但不想要所有的膨脹 `create-react-app` 或其他一些樣板。這是如何設置僅使用 Webpack 設置 React 的分步演練！

項目完成後的樣子

本指南將遵循已經建立的約定 `create-react-app` - 例如調用構建文件夾 `build` ，靜態資產位於 `public` 等。

最後，這是我們將擁有的文件夾結構：

```
- public
   - index.html
- src
   - App.jsx
webpack.config.js
.babelrc
package.json
```

我們將一步一步檢查每個階段後是否一切正常：

1. 基本腳手架：創建項目文件夾並提供純 HTML
2. 添加 Webpack 並捆綁一個簡單的 JS 文件
3. 添加 Babel 以支持 ES6
4. 添加反應

事不宜遲，讓我們開始吧！

第 1 步：基礎腳手架

第一步是創建項目文件夾並添加一個純 HTML 文件。

要創建項目並初始化 `package.json` ，請運行以下命令：

```
mkdir react-webpack
cd react-webpack
npm init -y
```

然後，添加主索引 `html` 文件——它通常位於一個 `public` 目錄中，所以讓我們這樣做：

```
mkdir public
touch public/index.html
```

💡

提示：您可以通過在項目文件夾中鍵入並從那里 `code .` 手動創建文件夾來在 VSCode 中打開項目 `public`

將 `index.html` 只包含基本的 HTML5 樣板文件：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>React + Webpack</title>
</head>
<body>
    <h1>Hello React + Webpack!</h1>
</body>
</html>
```

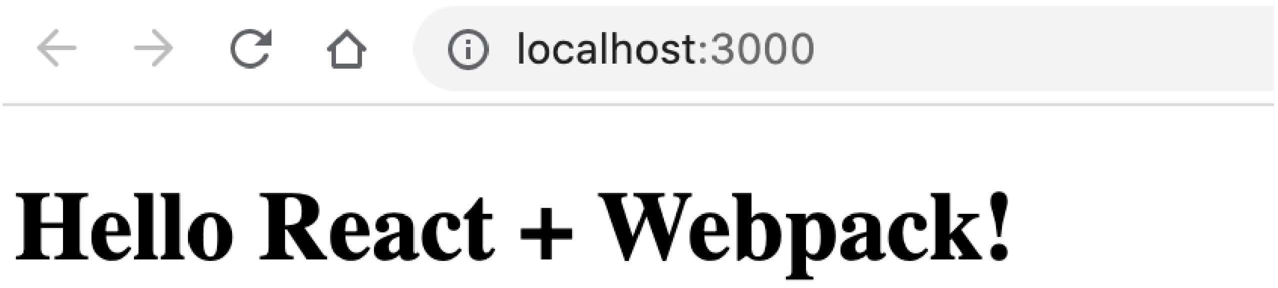
💡

提示：在 VSCode 中，如果您鍵入 `html:5` 並點擊 Tab，VSCode 將為您創建 `index.html` 內容！

現在讓我們通過僅提供我們剛剛使用 npm serve創建的 HTML 來檢查它是否運行到現在。

```
npx serve public
```

確實如此！如果您導航到http://localhost:3000，您應該會看到我們剛剛添加的問候消息：



Hello React + Webpack!

💡

Tip: If `npx serve public` fails for you with `Must use import to load ES Module` error, check your Node version and make sure you're using at least Node 16 (latest LTS).

Step 2: Adding Webpack

For this section, it's best to just follow the latest official Webpack docs.

First, install Webpack:

```
npm install webpack webpack-cli --save-dev
```

Next, let's create a simple JavaScript file that we can configure Webpack to bundle:

```
mkdir src
touch src/index.js
```

We can just create a div with a hello message and add it to the document:

```
// index.js

const helloDiv = document.createElement("div");
helloDiv.innerHTML = "Hello from Javascript!";
document.body.append(helloDiv);
```

The, we need to configure Webpack by creating a `webpack.config.js` file in the root of the project:

```
// webpack.config.js

const path = require("path");

module.exports = {
  entry: "./src/index.js",
  output: {
    filename: "main.js",
    path: path.resolve(__dirname, "build"),
  },
};
```

Finally, in `package.json`, add a new "build" script:

```
// ...
"scripts": {
    "build": "webpack"
},
```

Now, let's try it out! After running `npm run build`, you should see a new folder was created, called `build`, with a `main.js` file in it! 💪

💡

Tip: Add the `build` folder to `.gitignore` to not commit it by accident. And if you haven't already, make sure to also ignore the `node_modules` folder.

Next, we need to move the static assets to the bundle.

More specifically, we want to also include the `index.html` file in the `build` folder.

Easiest way to do this is with the HtmlWebpackPlugin.

```
npm install html-webpack-plugin --save-dev
```

And update the `webpack.config.file`:

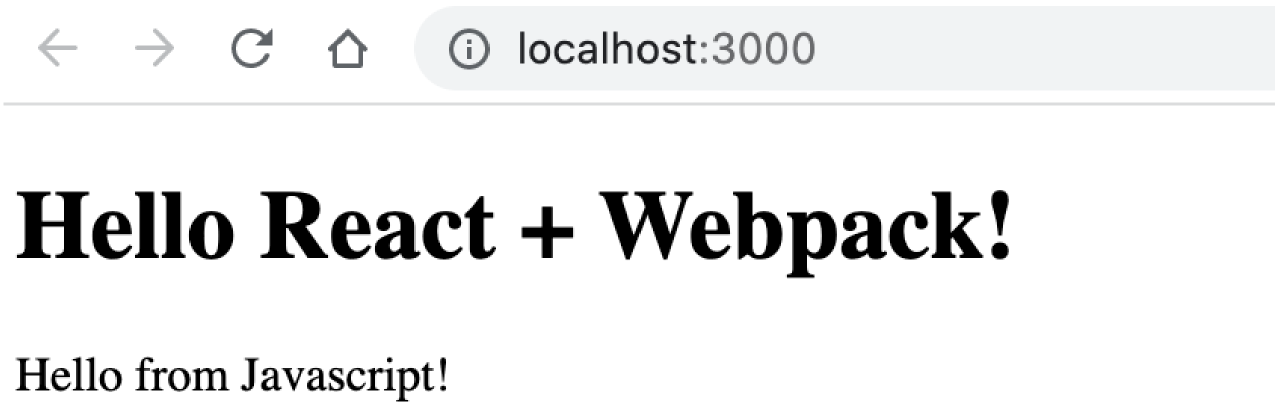
```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  entry: "./src/index.js",
  output: {
    filename: "main.js",
    path: path.resolve(__dirname, "build"),
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: path.join(__dirname, "public", "index.html"),
    }),
  ],
};
```

This will copy the file under `public/index.html`, copy it to the `build` folder and inject a link to the bundled JS file (`main.js`).

Let's try it out!

```
npm run build
npx serve build
```

And it works! You should now also see the message "Hello from Javascript" :D


localhost:3000

Hello React + Webpack!

Hello from Javascript!

Adding Webpack dev server

So far, it was ok to just use `npx serve` to check our app works, but in real life, it's easier to just use the `webpack-dev-server`, so let's add that as well.

```
npm install --save-dev webpack-dev-server
```

Then, configure it in the Webpack config:

```
{
  // ...,
  devServer: {
    static: {
      directory: path.join(__dirname, "build"),
    },
    port: 3000,
  }
}
```

... and then add `npm run start` script to package json; and while we're there, pass in the right "mode":

```
{
  // ...,
  "scripts": {
    "build": "webpack --mode production",
    "start": "webpack serve --mode development"
  }
}
```

Finally, check that it works: run `npm run start`, open http://localhost:3000 and check the app still works as before.

Step 3: Adding Babel

This is useful for allowing us to use all ES6 features and having them transpiled down to JS versions that all browsers can understand. (If you want to learn more about what Babel `preset-env` is and why it's useful, this article by Jacob Lind is a great overview).

First, let's install the required packages:

```
npm i @babel/core @babel/preset-env babel-loader --save-dev
```

Next, update the Webpack config to tell it to pass the files through Babel when bundling:

```
{
 // ...,
 module: {
    // exclude node_modules
    rules: [
      {
        test: /\.(js)$/,
        exclude: /node_modules/,
        use: ["babel-loader"],
      },
    ],
  },
  // pass all js files through Babel
  resolve: {
    extensions: ["*", ".js"],
  }
}
```

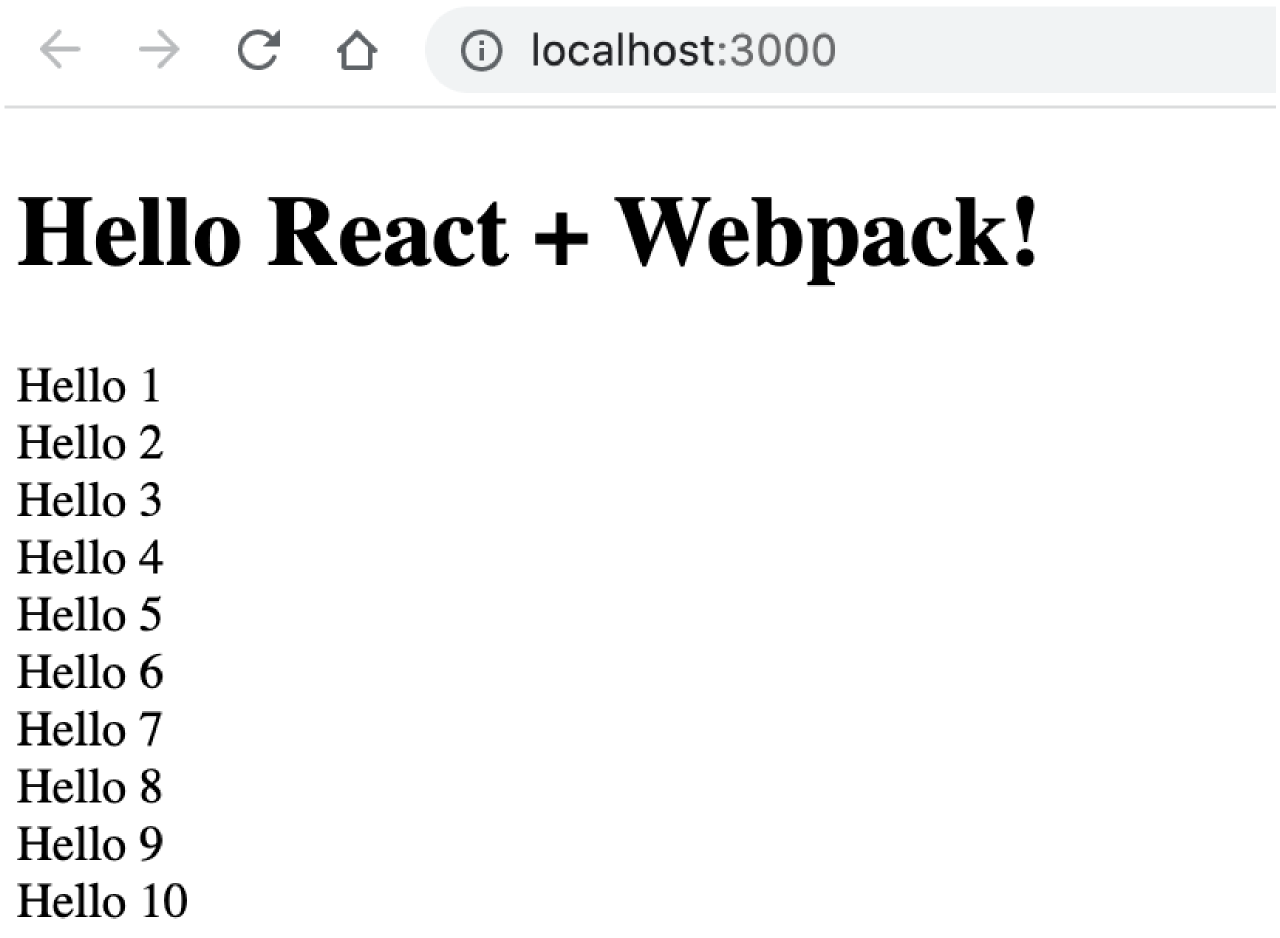
Then, create the Babel config file - `.babelrc`. This is where we configure Babel to apply the `preset-env` transform.

```
// .babelrc

{
  "presets": [
    "@babel/preset-env"
  ]
}
```

Optionally, update the `index.js` to contain some ES6 features that wouldn't work without Babel 😀 (actually they do work in most browsers, but for example IE still doesn't support Array.fill).

```
// Use a feature that needs Babel to work in all browsers :)
// arrow functions + Array fill

const sayHelloManyTimes = (times) =>
  new Array(times).fill(1).map((_, i) => `Hello ${i + 1}`);

const helloDiv = document.createElement("div");
helloDiv.innerHTML = sayHelloManyTimes(10).join("<br/>");
document.body.append(helloDiv);
```

Finally, let's check everything works - run `npm run start` and check the app correctly runs:



Hello React + Webpack!

Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10

Step 4: Add React

Finally, we can add React 😅

First, install it:

```
npm i react react-dom --save
npm i @babel/preset-react --save-dev
```

Then, update the `.babelrc` file to also apply the `preset-react` transform. This is needed, among other things, to support JSX.

```
// .babelrc

{
    "presets": [
      "@babel/preset-env",
      ["@babel/preset-react", {
      "runtime": "automatic"
    }]
    ]
}
```

💡

Tip: Specifying the `preset-react` runtime as `automatic` enables a feature that no longer requires importing React on top of every file.
Also, we need to update the Webpack config to pass `jsx` files through Babel as well:

```
// webpack.config.js

{
  // ...,
  module: {
    // exclude node_modules
    rules: [
      {
        test: /\.(js|jsx)$/,         // <-- added `|jsx` here
        exclude: /node_modules/,
        use: ["babel-loader"],
      },
    ],
  },
  // pass all js files through Babel
  resolve: {
    extensions: ["*", ".js", ".jsx"],    // <-- added `.jsx` here
  },
}
```

Next, let's create a React component, so we can check that everything works:

```
// src/Hello.jsx

const Hello = () => <h1>Hello from React!</h1>;

export default Hello;
```

And add it to the main app file:

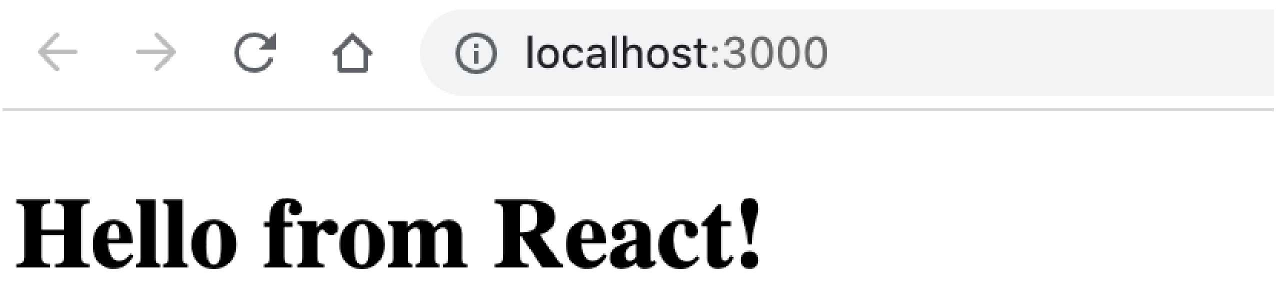
```
// index.js

import React from "react";
import { createRoot } from "react-dom/client";
import Hello from "./Hello";

const container = document.getElementById("root");
const root = createRoot(container);
root.render(<Hello />);
```

💡

Note we're using the new React 18 syntax with `createRoot`.
Finally, we also update the `index.html` to provide a "root" node for the app:

```
<!-- index.html -->
<!-- ... -->
<body>
    <div id="root"></div>
</body>
```

Let's check that it works - run `npm run start` and you should see "Hello from React!":



Hello from React!

Also, check there are no errors in the console.

You can also check that the app correctly runs in production, by running `npm run build` and then `npx serve build`.

That's it!

See the full code for this tutorial on Github: https://github.com/jsramblings/react-webpack-setup

Clear, straight to the point tutorials, straight to your inbox

Enter your email below to receive new articles as soon as they are published!

Login

M↓ Markdown
UpvotesNewestOldest
Thank you! I've gone through several tutorials to set this environment up, and yours is the very first that has actually worked as-is, without a single hiccup!

Thank you!

Best article. I've been struggling with CRA5 (with Electron) for ages. This has simplified everything !

This was a fantastic tutorial!! Thank you so much!

Great article

i got same error, this doesnt work For me the very first step doesn't work... "npx serve public" blows the following: Must use import to load ES Module: C:\Users***\AppData\Roaming\npm-cache*npx\4620\node*modules\serve\build\main.js i recommended this one https://dev.to/deepanjangh/react-setup-with-webpack-for-beginners-2a8k

What version of node are you using? Could you try upgrading node to v16 (latest LTS)? I notice there was another reader who had a similar issue and who was also on Windows, so it's likely the same cause.

For me the very first step doesn't work... "npx serve public" blows the following: Must use import to load ES Module: C:\Users***\AppData\Roaming\npm-cache_npx\4620\node_modules\serve\build\main.js

OS: Windows 11

> node -v v12.16.1
>
> npm -v 6.13.4

It looks like the latest version of the `serve` package doesn't work with node v12 (a lot of new libraries switched to the new `import` node syntax instead of `require`, so they fail when using with an older version of node). Could you try upgrading node to v16 (latest LTS)?

Great article ! Thanks man. I've been struggling with CRA5 (with Electron) for ages. This has simplified everything !

我認為這使得學習 React 變得更加愉快，因為消除了內部的魔法和復雜性 `create-react-app` 。並不是說它（ `c-r-a` ）不是更高級的方法，而是我經常嘗試創建類似項目的小型實驗室，並且感覺 `c-r-a` . **無論如何，非常感謝組織良好的教程**🙅‍♂️

如此清晰易懂。謝謝！

謝謝！

我認為這是您可以在互聯網上找到的關於該主題的最佳教程。

哈哈謝謝你！很高興它有幫助🙏

謝謝，這真的很有幫助！👍