1:安装node

2:初始化package.json,安装webpack, webpack-cli

mkdir webpack-demo && cd webpack-demo

npm init -y

npm install webpack webpack-cli --save-dev

3: 创建以下目录结构、文件和内容：

webpack-demo

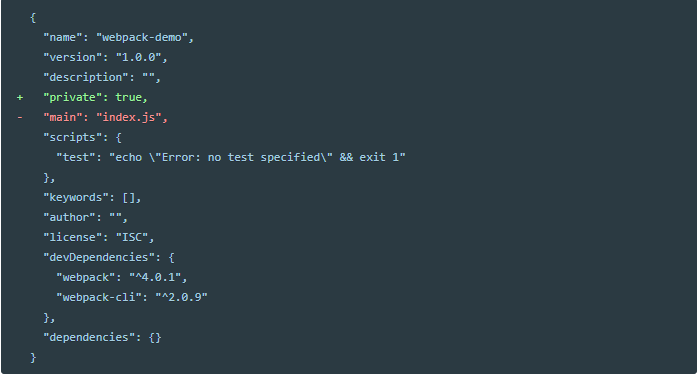
|- package.json

+ |- index.html

+ |- /src

+ |- index.js

4: 调整 package.json 文件，以便确保我们安装包是私有的(private)，并且移除 main 入口



5:  npx webpack 执行webpack编译

lodash

# Lodash

是一个一致性、模块化、高性能的 JavaScript 实用工具库。

模块:

注意，webpack 不会更改代码中除 import 和 export 语句以外的部分。如果你在使用其它 [ES2015 特性](http://es6-features.org/)，请确保你在 webpack 的 [loader 系统](https://www.webpackjs.com/concepts/loaders/)中使用了一个像是 [Babel](https://babeljs.io/) 或 [Bublé](https://buble.surge.sh/guide/) 的[转译器](https://www.webpackjs.com/loaders/#transpiling)。

webpack 4 中，可以无须任何配置使用，然而大多数项目会需要很复杂的设置，这就是为什么 webpack 仍然要支持 [配置文件](https://www.webpackjs.com/concepts/configuration)

webpack.config.js内容

const path = require('path');

module.exports = {

entry: './src/index.js',

output: {

filename: 'bundle.js',

path: path.resolve(\_\_dirname, 'dist')

}

};

npx webpack --config webpack.config.js

*注意，当在 windows 中通过调用路径去调用 webpack 时，必须使用反斜线()。例如 node\_modules\.bin\webpack --config webpack.config.js。*

*如果 webpack.config.js 存在，则 webpack 命令将默认选择使用它。我们在这里使用 --config 选项只是向你表明，可以传递任何名称的配置文件。这对于需要拆分成多个文件的复杂配置是非常有用。*

*给package.json添加编译脚本:build npm run build*