Creating and publishing React custom library

by Dr. DSR Murthy

```
C:\>npx create-react-app  react-npm-library
## then enter the new directory

cd react-npm-library

## then start the dev server
npm start
```

npx installs the latest versions of react-scripts and does not install any thing globally.

Then enter the src directory and create a new directory where *to* place component library

```
cd src
mkdir react-library
```

Creating the library

Inside the react-library directory , create a file for the component.

```
button.jsx
index.css
```

Write button.jsx code and index.css code

Now import it from App.js and test it.

Initializing the library

Now go to react-library directory for publishing

```
cd react-library
```

initialize an npm package

```
npm init –y
```

This will create a package.json file in the root directory. It should look like this

```
{
  "name": "react-library",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Bundling the library

Now lets get ready to bundle the library

Re-arranging the package directory

Arrange the react-library directory for bundling.

Go to terminal and type these commands inside the react-library directory

```
mkdir src
move button.jsx src
move index.css src
cd src
create index.js file here
```

The above commands will move the button.jsx and index.css files to a new src directory and also create a new file called index.js

Project structure now:

```
|
 - src
   | - index.js
   | - index.css
   | - button.jsx
 - package.json
```

Inside the index.js file add the following code

```
import AwesomeButton from  './button.js'

const returnLibrary = () => {
    return {
        AwesomeButton: AwesomeButton
        // you can add here other components  to export
    }
}
export default returnLibrary()
```

Installing bundlers

Install rollup for bundling. You can also use webpack.

So in the root of the react-library directory install rollup.

- npm install rollup --save-dev

Rollup will be used to compile our code. Install babel for es5 transpilation.

- npm install @babel/cli @babel/core @babel/preset-env @babel/preset-react @rollup/plugin-babel --save-dev

Since we are also bundling css then we shall have to install a styles bundler for rollup we shall use rollup-plugin-styles

- npm install rollup-plugin-styles autoprefixer --save-dev

Also add babel runtime helpers. this is important if we are bundling a library with babel.

- npm install @babel/runtime
- npm install @babel/plugin-transform-runtime --save-dev

Configuration

Now lets configure the rullop and babel for compiling our code.

In the root directory create these to files.

- rollup.config.js
- .babelrc

Inside rollup.config.js add the following code.

```
import styles from "rollup-plugin-styles";
const autoprefixer = require('autoprefixer');
import { terser } from 'rollup-plugin-terser'
import babel from '@rollup/plugin-babel';

// the entry point for the library
const input = 'src/index.js'

//
var MODE = [
  {
    fomart: 'cjs'
  },
  {
    fomart: 'esm'
  },
  {
    fomart: 'umd'
  }
]
```

```
var config = []


MODE.map((m) => {
   var conf = {
      input: input,
      output: {
         // then name of your package
         name: "react-awesome-buttons",
         file: `dist/index.${m.fomart}.js`,
         format: m.fomart,
         exports: "auto"
      },
      // this externelizes react to prevent rollup from compiling it
      external: ["react", /@babel\/runtime/],
      plugins: [
         // these are babel comfigurations
         babel({
            exclude: 'node_modules/**',
            plugins: ['@babel/transform-runtime'],
            babelHelpers: 'runtime'
         }),
         // this adds sourcemaps
         sourcemaps(),
         // this adds support for styles
         styles({
            postcss: {
               plugins: [
                  autoprefixer()
               ]
            }
         })
      ]
   }
   config.push(conf)
})

export default [
  ...config,
]
```

Also add this to .babelrc

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

Editing package.json scripts

Now got to package.json and edit the scripts section and change it to this.

```
// package.json
...
"scripts": {
   "build": "rollup -c"
}
...
```

Build package

Now that everything is set run

```
npm run build
```

This will compile package into the dist directory.

Editing package.json

Now that our library has been built lets edit package.json to make our library ready for publishing.

```
{
   "name": "react-library",
   "version": "1.0.0",
   "description": "",
   "main": "index.js",
   "directories": {
      "test": "test"
   },
```

```
  "scripts": {
    "build": "rollup -c"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@babel/runtime": "^7.12.5"
  },
  "devDependencies": {
    "@babel/cli": "^7.12.10",
    "@babel/core": "^7.12.10",
    "@babel/plugin-transform-runtime": "^7.12.10",
    "@babel/preset-env": "^7.12.11",
    "@babel/preset-react": "^7.12.10",
    "@rollup/plugin-babel": "^5.2.2",
    "rollup-plugin-sourcemaps": "^0.6.3",
    "rollup-plugin-styles": "^3.12.2",
  }
}
```

Final package.json

Now edit it to look like this

```
{
  "name": "react-library",
  "version": "1.0.0",
  "description": "your description",
  "main": "dist/index.cjs.js",
  "scripts": {
    "build": "rollup -c"
  },
  "peerDependencies": {
    "react": "^17.0.1",
    "react-dom": "^17.0.1"
  },
  "dependencies": {
    "@babel/runtime": "^7.12.5"
  },
  "keywords": [
    "react",
    "keywords"
  ],
```

```
    "author": "Your name",
    "license": "MIT",
    "devDependencies": {
        "@babel/cli": "^7.12.10",
        "@babel/core": "^7.12.10",
        "@babel/plugin-transform-runtime": "^7.12.10",
        "@babel/preset-env": "^7.12.11",
        "@babel/preset-react": "^7.12.10",
        "@rollup/plugin-babel": "^5.2.2",
        "rollup-plugin-sourcemaps": "^0.6.3",
        "rollup-plugin-styles": "^3.12.2",
    }
}
```

Publishing

Now you are ready to publish.
First create an npm account.

Creating .npmignore

```
|
| - dist
    | - index.esm.js
    | - index.cjs.js
    | - index.umd.js
| - src
    | - index.js
    | - index.css
    | - button.jsx
| - .babelrc
| - package.json
| - rollup.config.js
```

add this to .npmignore file.

```
## the src folder
src
.babelrc
rollup.config.js
## node modules folder
node_modules
## incase you have a git repositiory initiated
```

```
.git
.gitignore
CVS
.svn
.hg
.lock-wscript
.wafpickle-N
.DS_Store
npm-debug.log
.npmrc

config.gypi
package-lock.json
```

Finding a name

Sometimes you might try to publish a package and find that the name is either already taken or the name is almost identical to another package so its better to first search and see if the package name is already taken. So type the following command in the command line.

```
npm search [package name]
```

if you find that nobody is using it them you can use the name.

Testing your package

To test package go to another projects on computer and type

```
npm link /path/to/your/package
```

Adding README.md

You should also add a Readme.md file that will be displayed on npm having a description of your package. You might be familiar with it if you have ever created a repository on GitHub

Publishing

If all works well then you can publish it by typing

```
npm publish
```

Happy Future