pre-requirements

1. install nodejs

To create react app without CRA

[To install all essential packages]

1. To initialize node project under your project folder

   `npm init (if you have no other requirements, just pressing enter)`
2. To install yarn

   `npm install --save-dev yarn`
3. To install react related packages:

   `npm install --save-dev react react-dom react-scripts`
4. To install below nodejs packages and execute audit process:

   ```
   npm install -g webpack-cli
   npm install uglifyjs-webpack-plugin --save-dev
   npm install babel-preset-react-app --save-dev
   npm audit --force
   npm audit
   ```
5. To install babel loader for jsx parsing:

   ```
   npm install babel-loader babel-core babel-preset-env -D
   npm i @babel/plugin-transform-runtime -D
   npm i @babel/runtime -D
   npm i @babel/core@^7.0.0 -D
   npm i @babel/preset-env -D
   npm i @babel/plugin-proposal-class-properties -D
   ```
6. To link babel-loader and its cores and plugins

   ```
   npm i babel-loader @babel/core @babel/runtime @babel/preset-env
   @babel/plugin-proposal-class-properties @babel/plugin-transform-runtime -D
   ```

[To create needed files for Webpack and jsx parsing]

until now, your project folder structure will be like:

1. node_modules
2. package-lock.json
3. package.json

To create babel file manually under the root path of your project:
.babelrc, which content as below

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

```
}
```

To create webpack config file manually under the root path of your project: webpack.config.js

until now, your project folder structure will be like:

1. node_modules
2. package-lock.json
3. package.json
4. .babelrc
5. webpack.config.json

Now, we have to add “scripts” (for build command), “eslintConfig” (for debug) and “browserslist” (for browser limitations) parts and modify devdependencies in our package.json for build process, which content is below:

```
{
  "name": "rectmanual",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "react-scripts start",
    "build_react": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "build": "npm run build:react && npm run build:bundle",
    "build:react": "react-scripts build",
    "build:bundle": "webpack --config webpack.config.js",
    "build:lib": "NODE_ENV=production
./node_modules/react-scripts/node_modules/.bin/webpack"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@babel/core": "^7.12.3",
    "@babel/plugin-proposal-class-properties": "^7.12.1",
    "@babel/plugin-transform-runtime": "^7.12.1",
    "@babel/preset-env": "^7.12.1",
    "@babel/runtime": "^7.12.1",
    "babel-core": "^6.26.3",
    "babel-loader": "^8.1.0",
    "babel-preset-env": "^1.7.0",
    "babel-preset-react-app": "^10.0.0",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-scripts": "^4.0.0",
    "uglifyjs-webpack-plugin": "^2.2.0",
    "webpack-cli": "^4.1.0",
    "yarn": "^1.22.10"
  },
  "eslintConfig": {
    "extends": "react-app",
    "plugins": [
      "react-hooks"
    ],
    "rules": {
      "react-hooks/rules-of-hooks": "error",
      "react-hooks/exhaustive-deps": "warn"
    }
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

now, please create 2 folders : src and public, those folders are default rule for react-webpack project. src is for your react components and entry js, public is for web server hosting (yarn start will use this as site folder)

until now, your project folder structure will be like:

1. node_modules
2. src
3. public
4. package-lock.json
5. package.json

6. .babelrc
7. webpack.config.json

now, please add index.js into src, index.js content like below, is a way to enabling your react world:

```
import Lobby from "./components/Lobby/Lobby"
import React from 'react';
import ReactDOM from 'react-dom';


ReactDOM.render(
        <React.StrictMode>
          <Lobby/>
        </React.StrictMode>,
        document.getElementById("app")
);
```

now, please add index.html and create the folder:js in the public folder, index.html's content like below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    </head>
    <body>
        <div id="app"/>
        <script src="js/bundle.infolink.min.js"></script>
    </body>
</html>
```

```
 <script src="js/bundle.infolink.min.js"></script>
```

The above javascript file is built by webpack, we will explain "how-to" later, here, this bundle file can be changed as your own.

until now, your project folder structure will be like:

1. node_modules

2. src
3. public
    js
    index.html
4. package-lock.json
5. package.json
6. .babelrc
7. webpack.config.json

now we have to write our own webpack.config.js, which content is below:

1. entry section: “bundle.infolink.min.js” is for our all components bundle file name, and glob.sync will let all our files (js, jsx or css) in the src folder into the build folder (this folder will be generated by webpack automatically)
2. output section: “../public/js/[name]” is a full path of our library, syntax [name] will adopt “bundle.infolink.min.js” directly or you can change to your own. if you describe “public/js/[name]”, webpack will generate it under the folder : dist (this folder will be generated by webpack automatically), umd means for general purpose module, InfoLinkComponents is a our lib. name
3. module section: is modules that we need during webpack building process, because we have css load and jsx parsing needs, so we deploy 2 modules.
4. plugin section: we need warning or hint messages during runtime, so we deploy uglifyJsPlugin

```
const path = require("path")
const UglifyJsPlugin = require("uglifyjs-webpack-plugin")
const glob = require("glob")

module.exports = {
  entry: {
    "bundle.infolink.min.js":
glob.sync("build/static/?(js|jsx|css)/main.*.?(js|jsx|css)").map(f =>
path.resolve(__dirname, f)),
    //"bundle.infolink.js": glob.sync("src/components/*.?(js|jsx|css)").map(f =>
path.resolve(__dirname, f))
  },
  output: {
    filename: "../public/js/[name]",
    libraryTarget: 'umd',
    library: "InfoLinkComponents"
  },
  module: {
    rules: [
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"],
      },
      {
        test: /\.(js|mjs|jsx|ts|tsx)$/,
        exclude: /(node_modules|bower_components)/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: ['@babel/preset-react']
          }
        }
      }
    ],
  },
  plugins: [new UglifyJsPlugin()],
}
```

for now, we prepare everything, we can start to develop our react project:

1. have some code into index.js in the folder: src

```
import Lobby from "./components/Lobby/Lobby"
import React from 'react';
import ReactDOM from 'react-dom';


ReactDOM.render(
        <React.StrictMode>
          <Lobby/>
        </React.StrictMode>,
        document.getElementById("app")
);
```

2. have some components under the src, like: Lobby and Member

until now, your project folder structure will be like:

1. node_modules
2. src

   components

      Lobby

         Lobby.jsx

      Member

         Member.jsx

```
   index.js
3. public
     js
     index.html
4. package-lock.json
5. package.json
6. .babelrc
7. webpack.config.json
```

```
import React from 'react'
import Member from "../Member/Member"


const Lobby = () =>{
    return (
        <>
            <span>hello member</span><p/>
            <Member name="hello"/>
        </>
    )
}

export default Lobby;
```

```
import React, { useEffect, useState } from 'react'


const Member = (props) => {

    const [name, setName] = useState("default name");

    useEffect(()=>{
        props.name && setName(props.name)
    }
    ,[])

    return(
        <span> name : {name}</span>
    );
}

export default Member;
```

OK, please execute: **yarn build**, if successful, we can find bundle.infolink.min.js in the folder : public/js and you still can find webpack automatically generates 2 folders for us: **build** and **dist (it should have no file, because we change output location**.)

until now, your project folder structure will be like:

1. node_modules
2. build
3. dist
4. src
   - components
     - Lobby
       - Lobby.jsx
     - Member
       - Member.jsx
   - index.js
5. public
   - js
     - bundle.infolink.min.js
   - index.html
6. package-lock.json
7. package.json
8. .babelrc
9. webpack.config.json

now executing: yarn start, and you can find your react project runs well.

demo file link:

https://drive.google.com/file/d/1QcNwTfSwjkhJnPhwOYr1D3mdPgK4-6T7/view?usp=sharing