vue cli3搭建组件库并实现按需引入

因为需要按需引入，所以在打包的时候是多入口文件页面打包的

打包配置

1. 配置entry

   新建一个getEntries(path)，其中path是组件代码所在的文件夹名称，返回一个对象entries，key为每个组件文件夹的名称，值为每个组件文件夹中入口文件index.js的绝对路径。

   首先使用`fs.readdirSync(resolve(path))`获取到组件代码所在的文件夹目录下所有文件名称，存在files变量中。

   然后用数组`reduce()`方法循环files，先将每个文件名（item）利用`join(path, item)`转成路径存到itemPath变量。

   用`fs.statSync(itemPath).isDirectory()`对每个文件进行判断是不是文件夹。

   如果是文件夹，先把itemPath和入口文件index.js拼接成一个地址，再转成绝对路径，将item作为key，赋值到返回对象上

   ```
   entries[item] = resolve(join(itemPath, 'index.js'))。
   ```

   如果不是文件夹，直接把itemPath转成绝对路径，将item去除后缀作为key，赋值到返回对象上

   ```
   const [name] = item.split('.')
   entries[name] = resolve(`${itemPath}`)
   ```

   完整代码如下：

   ```
   function getEntries(path) {
       let files = fs.readdirSync(resolve(path));
       const entries = files.reduce((ret, item) => {
           const itemPath = join(path, item)
           const isDir = fs.statSync(itemPath).isDirectory();
           if (isDir) {
               ret[item] = resolve(join(itemPath, 'index.js'))
           } else {
               const [name] = item.split('.')
               ret[name] = resolve(`${itemPath}`)
           }
           return ret
       }, {})
       return entries
   }
   ```

利用对象展开运算符，配置entry

```
entry: {
    ...getEntries('packages'),
},
```

2. 配置output

   filename: 配置每个组件打包后生成对应文件名称,多入口文件配置为`[name].index.js`，为什么配置这个名称后面会解释。

   libraryTarget：配置为`commonjs2`，入口文件的返回值将分配给module.exports对象，使其组件库在webpack构建的环境下使用，这个是关键。

```
output: {
    filename: '[name]/index.js',
    libraryTarget: 'commonjs2',
}
```

打包之后的目录结构如下：

按需引入配置

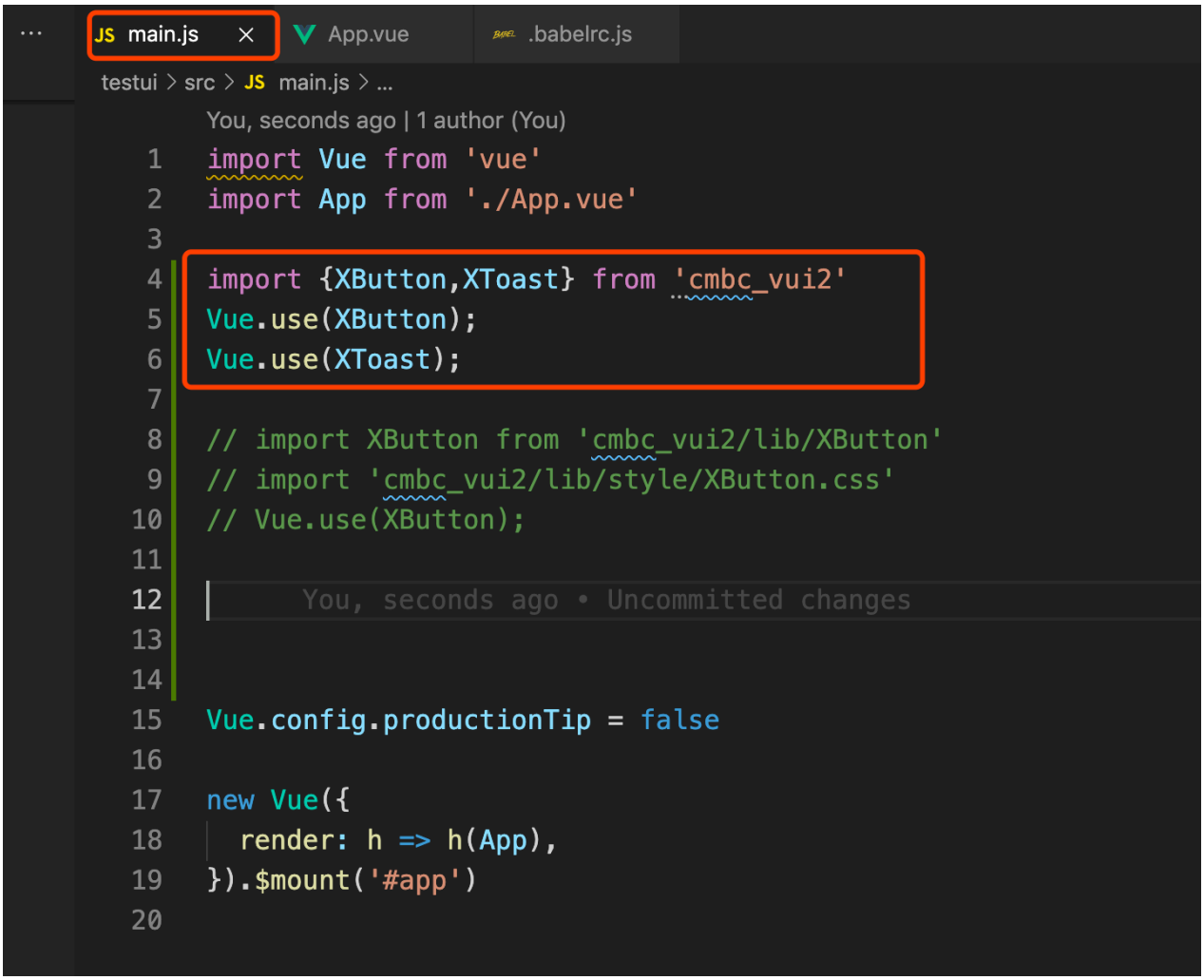
在引用组件库的项目中，执行`npm install cmbc_vui2 — save`安装组件库

在执行`npm install babel-plugin-import --save-dev`安装babel-plugin-import插件，利用它实现组件按需引入。

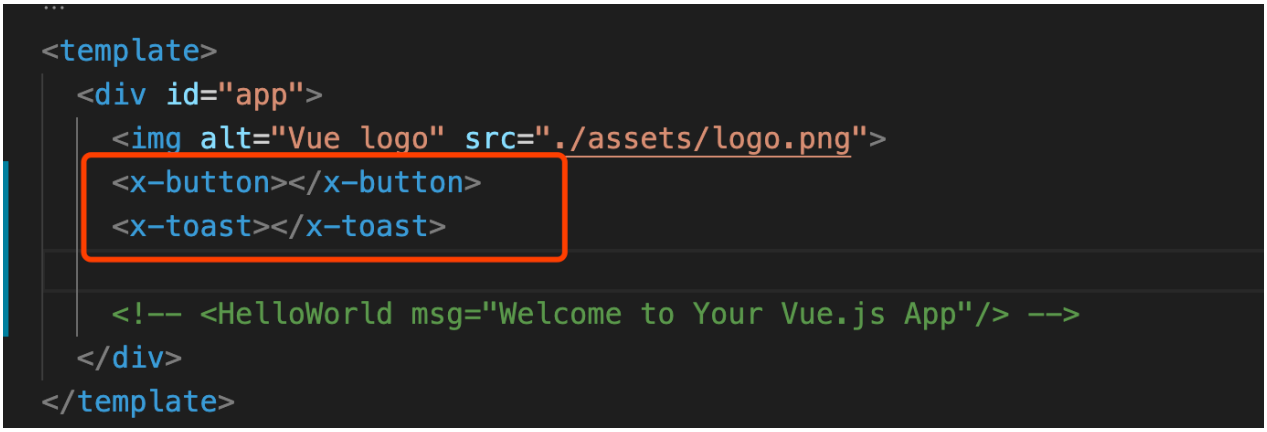
在项目的根目录下创建.babelrc.js文件，配置如下：

```
module.exports = {
    "presets": ["@vue/app"],
    "plugins": [
        [
            "import",
            {
                "libraryName": "cmbc_vui2",
                "camel2DashComponentName": false,
                "camel2UnderlineComponentName": false,
                "style": (name) =>{
                    const cssName = name.split('/')[2];
                    console.log(cssName)
                    return `cmbc_vui2/lib/style/${cssName}.css`
                }
            }
        ],
    ]
}
```

在入口文件中引入组件库



在需要的页面中应用

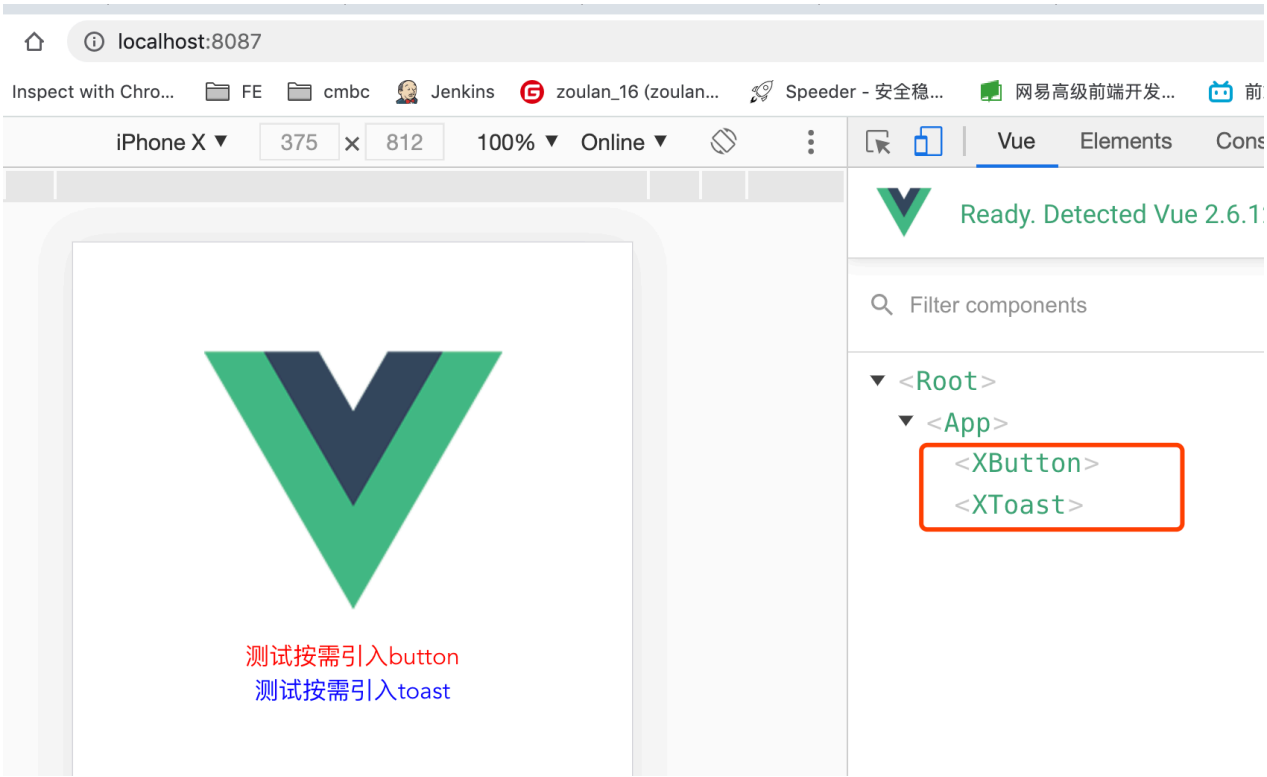


此时的样式是在配置文件中统一引入的

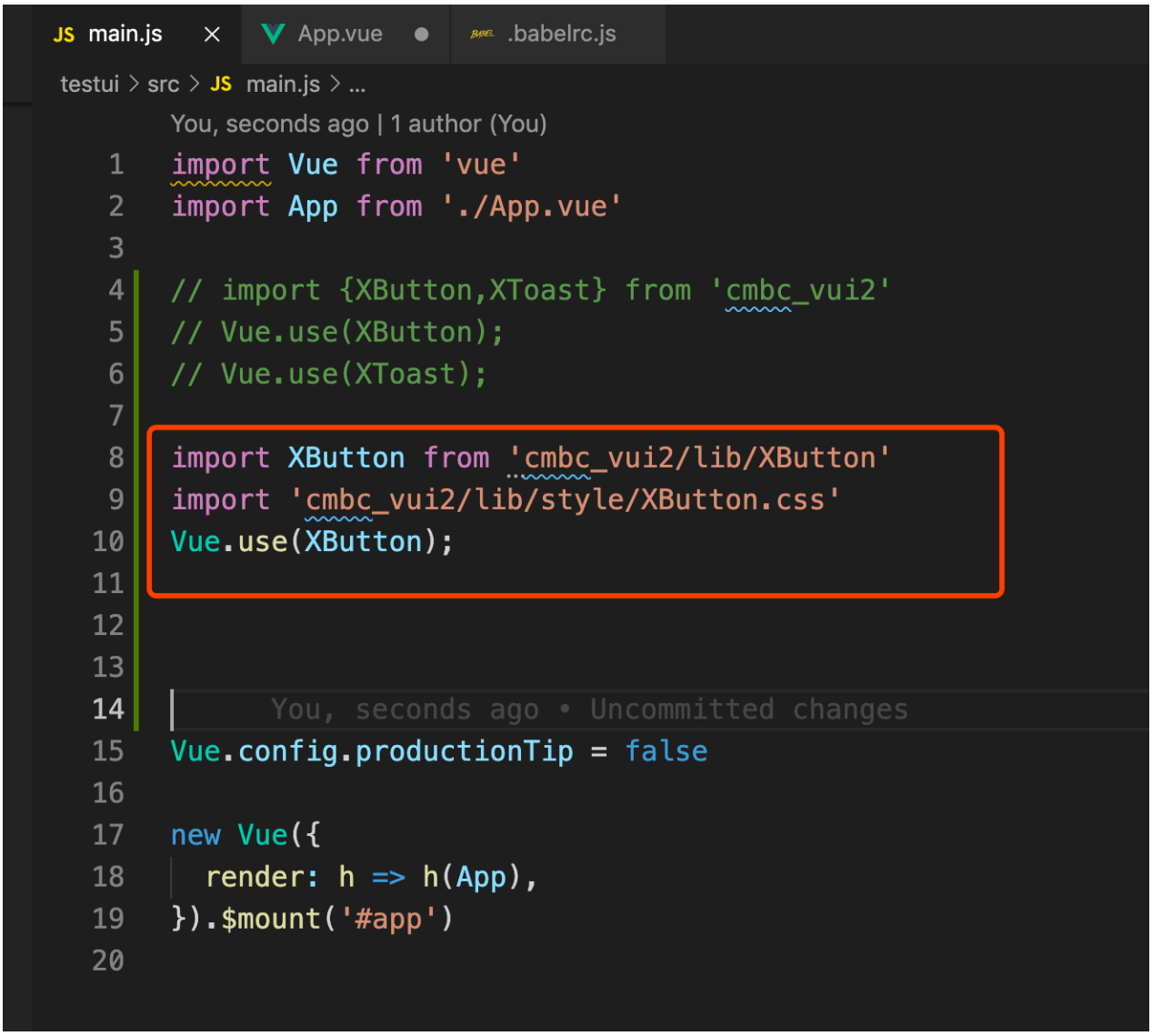
```babelrc.js
module.exports = {
    "presets": ["@vue/app"],
    "plugins": [
        [
            "import",
            {
                "libraryName": "cmbc_vui2",
                "camel2DashComponentName": false,
                "camel2UnderlineComponentName": false,
                "style": (name) =>{
                    const cssName = name.split('/')[2];
                    console.log(cssName)
                    return `cmbc_vui2/lib/style/${cssName}.css`
                }
            }
        ],
    ]
}
```

引入样式

至此，组件已引用完成，结果如下：



还有一种引用方式，就是单个组件引用，如下所示：

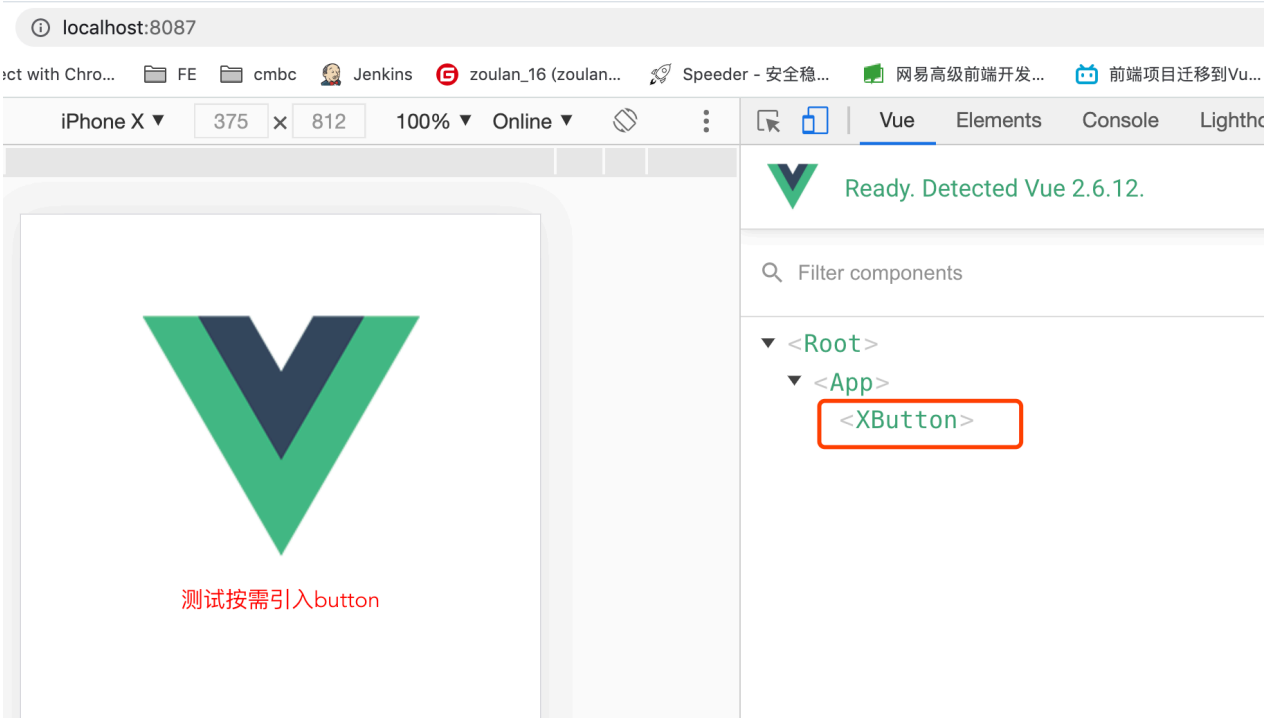


此时样式需要手动引入（此处可研究下样式自动引入，不用每次都手动引入）

```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <x-button></x-button>
    <!-- <x-toast></x-toast> -->      You, seconds ago • Uncomm

    <!-- <HelloWorld msg="Welcome to Your Vue.js App"/> -->
  </div>
</template>
```

结果如下：



注意：

```
import {XButton,XToast} from 'cmbc_vui2'
Vue.use(XButton);
Vue.use(XToast);
```

这个名称一定要是新建组件文件夹的名称，因为在多入口文件页面打包配置中配置entry时，entry对象每个key是每个组件代码所在的文件夹名，而output.filename是`'[name]/index.js'`。每个组件是独立打包，打包生成文件夹名称是原来每个组件代码所在文件夹名称。