React-Router在单界面开发中的应用

## 概念

在React组件化开发的设计模式中，通过管理URL的方式，实现组件的切换和状态的变化。和以往单界面开发模式中通过隐藏和显示DOM元素的方式有所不同，React-Router能够根据当前URL自动的选择所要在界面中显示的组件，并通过Router嵌套的方式，实现不同子组件在父组件下进行切换。对于不属于当前URL的组件并不会显示（不是隐藏该组件（Hidden），而是没有此组件）。

## 实现

React-Router是React官方维护的事实上也是唯一可选的路由库。使用React-Router需先下载安装该库。

### 安装



### 引入

在项目入口文件中进行引入操作





## React-Router的路由规则

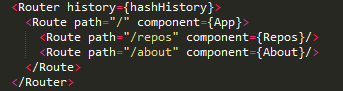
### 根路由：



如上，在访问web的根目录时，显示App组件

### 嵌套路由：

在以App为根组件的web应用中，App下包含多个子组件（子模块），因此，应在根Router下嵌套其他Router：



此路由方式和上图中将根路由和子模块路由平行设置的区别在于：

当url访问值/repos的时候，会先加载App组件，再加载Repos组件，App组件和Repos组件是嵌套的关系。

### Path属性



如图，在上述的path中，可以匹配/repos/zhangyiqiang/repo1或者/repos/admin/ad等多个url，repos是固定的，而userName和repoName则是可变的。

### 通配符



### 默认子组件的设置

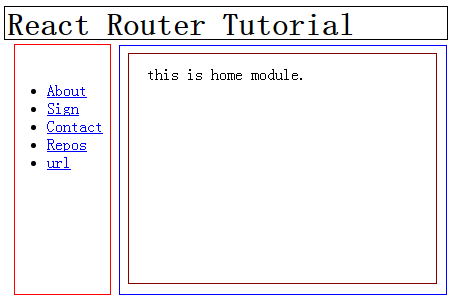


如图，IndexRoute表示在访问根组件App（url为’/’）的时候默认显示父组件App和子组件

Home。

## 根据React-Router开发单界面web应用

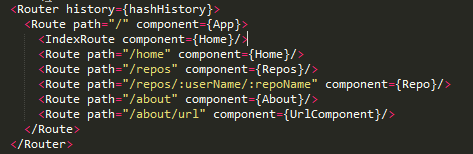
### 整体设计图



如上图所示我们的web应用总体包括标题栏（黑色框）、导航栏（红色框）、内容块（蓝色框和棕色框）。在单界面开发模式中，通过导航栏的点击事件，切换不同内容视图（棕色框区域），实现不同模块的内容展示。

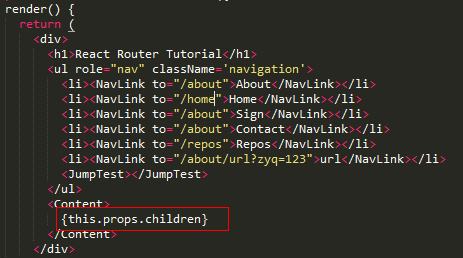
### URL划分

在上述设计图中，在不同模块切换的时候，标题栏和导航栏是不变的，以根路由‘/’对应整个web应用为基础，不同路由对应不同组件的嵌套关系如下：

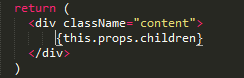


其中，在组件App中完成对于标题栏和导航栏的渲染（标题栏和导航栏由相应的子组件渲染）；在根路由‘/’下设置默认子组件Home显示web应用的主页；以不同路由（’/repos’、‘/about

’）等对应切换不同组件（Repos和About组件）。这些组件均在App组件的节点下渲染（根据React-Router的嵌套规则，在由’/repos’切换至‘/about’时，根目录下’/’对应的App组件依旧会被渲染，因为在渲染’/about’对应的About组件时，需先渲染其父组件App）



子组件Content中也需相应设置children填充：



## React根组件渲染节点：



在React应用的入口文件，以render()函数作为整个根组件的容器，需要注意的是，document.getElementById(‘app’)获取的app节点必须是在DOM中已经完成渲染。可以通过以下三个方式构造该节点：

①在html文件中直接写入

②javascript的createElement方法

③jQuery

## Webpack打包文件过大问题：

使用webpack打包工程所有依赖的js文件、css样式、第三方库等，最终形成了bundle.js文件。这样做的好处显而易见，它帮助了我们再编写js代码时能够模块化地引入我们所需要的资源，至于这些资源如何引入界面，则由webpack帮助我们打包。



如上图，但是这样不可避免的存在bundle.js文件过大的问题，上图只是简单的引用了react、react-dom、react-router、extract-text-webpack-plugin这四个资源库，加上自己写的demo所包含的js文件以及样式文件，经过webpakc的打包，最终的文件大小达到了863KB。可想而知，在项目中应用webpack打包，最终形成的文件可能达到几兆甚至几十兆的大小，如果整个web应用必须等到该文件加载完毕才开始渲染初始化，那将会导致糟糕的用户体验。

解决该问题主要通过以下几种方式解决：

1、提取第三方库从bundle.js中剥离

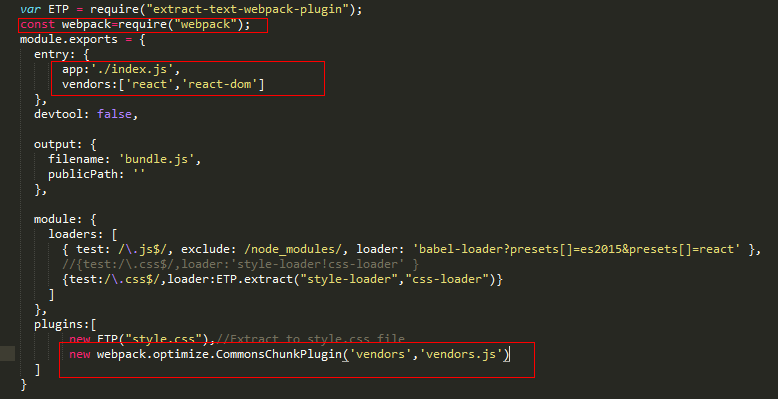
2、压缩代码

3、按需加载

4、css样式文件单独打包加载

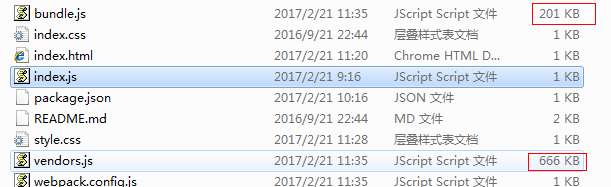
### 提取第三方库从bundle.js中剥离

bundle.js过大有很大一部分是因为把第三方库的文件也打包进来了，比如react.js和react-dom.js这两个文件的大小就有660KB左右，因此，需要把这两个文件单独打包，单独引入。Webpack.config.js的配置如下：



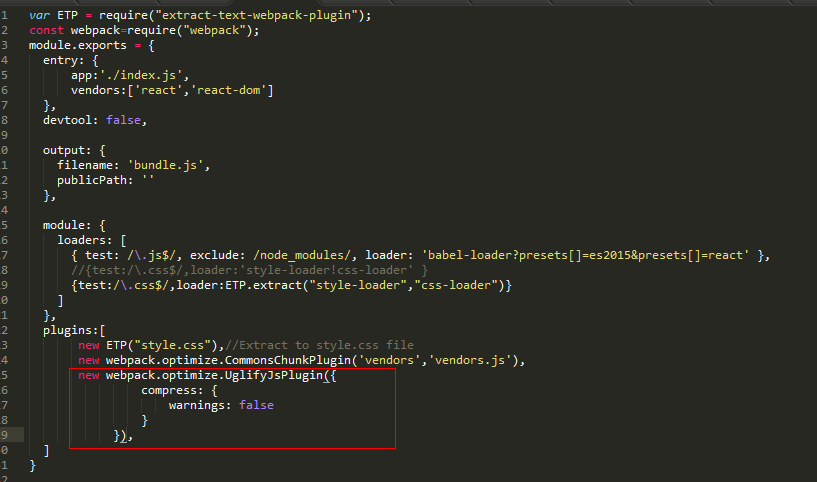
通过该配置，将会打包react和react-dom生成vendors.js文件，在我们的应用中先加载vendors.js再加载bundle.js。





### 压缩代码

通过webpack内置的压缩插件，可以很大程度的压缩我们的js文件。Webpack.config.js的配置如下：

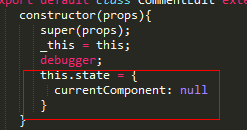


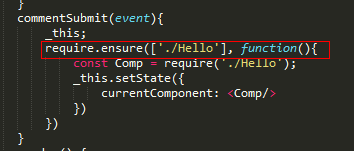


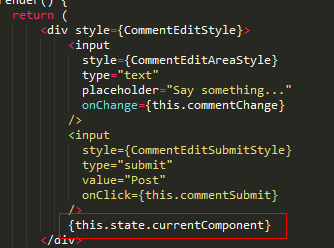
### 按需加载模块（组件）

作为单界面应用，即使做了文件压缩，一次性加载所有模块文件也是不合理的。对于我们的应用来说，某些模块的应用是需要用户触发的，如果用户没有触发动作，那这一模块则没有必要加载。因此需要采用按需加载来减小web应用初始化时所需要加载的资源大小。

Webpack提供了require.ensure来进行按需加载，具体使用demo如下：

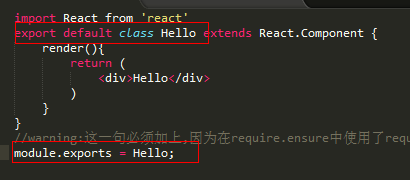




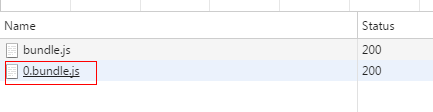


其实commentSubmit是组件被用户行为事件触发的函数（如鼠标单击事件）。

作为被按需加载的组件，应该具有如下形式：



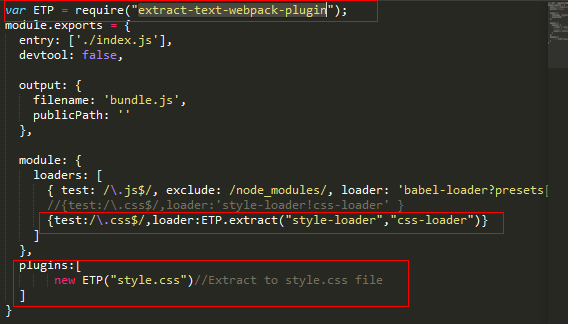
通过上述的配置，Hello组件只有在用户鼠标单机了所调用它的父组件之后，才会加载Hello.js文件并渲染其组件节点。



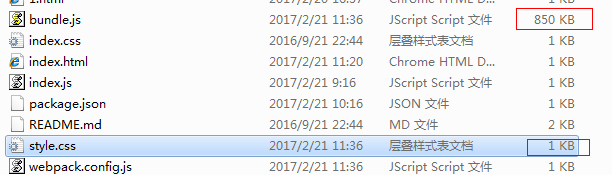
Webpack对此的处理是对需要加载的文件以区别于bundle.js的文件名进行打包加载。

### Css文件单独打包加载

Css文件单独打包加载需要使用到插件extract-text-webpack-plugin，需要先通过npm安装该插件。使用该插件的webpack.config.js配置如下：



通过以上配置，webpack将会将工程中以模块化引入的css样式打包输出为style.css文件，由index.html文件中单独调用，将不会在打包进bundle.js文件中。使用该方案能够减小bundle.js文件大小，剔除了原bundle.js中的样式文件。



### 综合上述三种方式（提取第三方库、压缩代码、css文件单独打包）：

