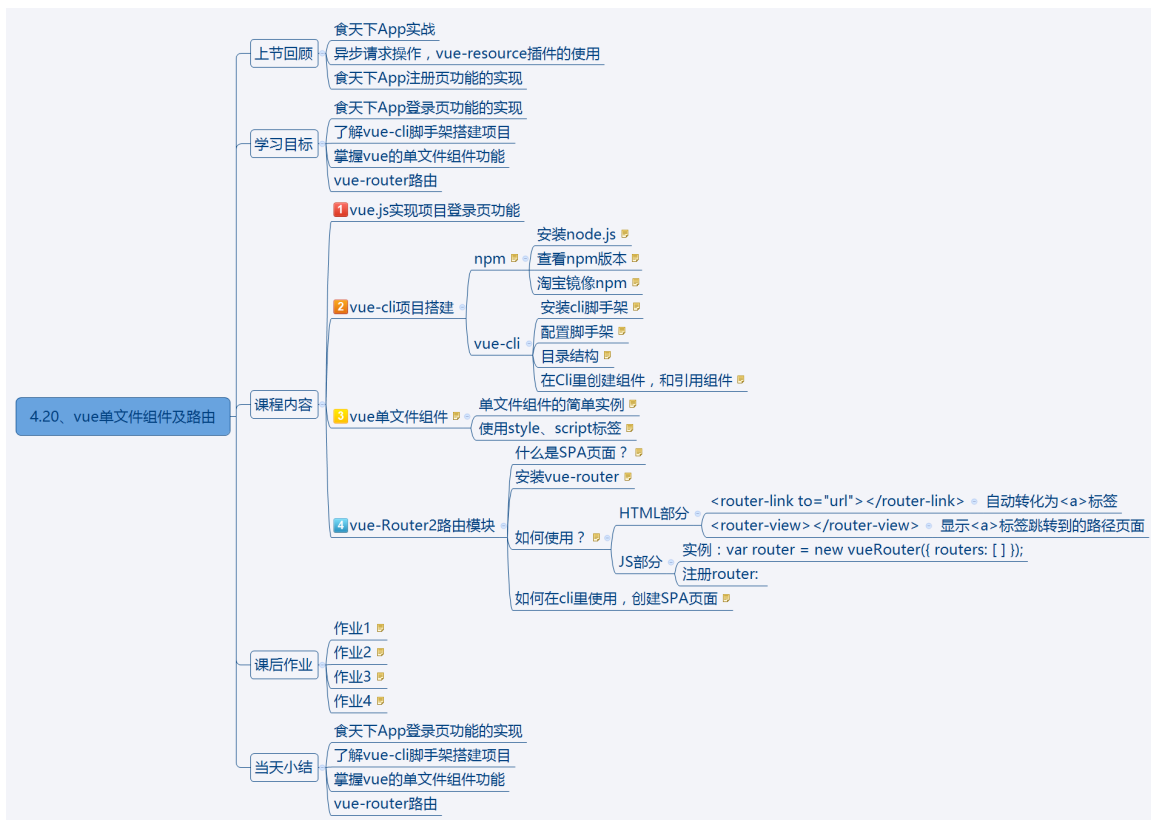


## 4.20、vue单文件组件及路由

4.20、vue单文件组件及路由 .....	1
1. 上节回顾 .....	3
1.1. 食天下App实战 .....	3
1.2. 异步请求操作，vue-resource插件的使用 .....	3
1.3. 食天下App注册页功能的实现 .....	3
2. 学习目标 .....	3
2.1. 食天下App登录页功能的实现 .....	3
2.2. 了解vue-cli脚手架搭建项目 .....	3
2.3. 掌握vue的单文件组件功能 .....	3
2.4. vue-router路由 .....	3
3. 课程内容 .....	4
3.1. vue.js实现项目登录页功能 .....	4
3.2. vue-cli项目搭建 .....	4
3.2.1. npm .....	4
安装node.js .....	4
查看npm版本 .....	4
淘宝镜像npm .....	4
3.2.2. vue-cli .....	5
安装cli脚手架 .....	5
配置脚手架 .....	5
目录结构 .....	7
在Cli里创建组件，和引用组件 .....	8
3.3. vue单文件组件 .....	11
3.3.1. 单文件组件的简单实例 .....	12
3.3.2. 使用style、script标签 .....	13
3.4. vue-Router2路由模块 .....	14
3.4.1. 什么是SPA页面？ .....	14
3.4.2. 安装vue-router .....	14
3.4.3. 如何使用？ .....	15
HTML部分 .....	16
JS部分 .....	17
3.4.4. 如何在cli里使用，创建SPA页面 .....	17
4. 课后作业 .....	20
4.1. 作业1 .....	21
4.2. 作业2 .....	21

4.3.	作业3.....	21
4.4.	作业4.....	21
5.	当天小结 .....	21
5.1.	食天下App登录页功能的实现 .....	21
5.2.	了解vue-cli脚手架搭建项目 .....	21
5.3.	掌握vue的单文件组件功能.....	21
5.4.	vue-router路由 .....	21



## 1. 上节回顾

### 1.1. 食天下App实战

### 1.2. 异步请求操作，vue-resource插件的使用

### 1.3. 食天下App注册页功能的实现

## 2. 学习目标

### 2.1. 食天下App登录页功能的实现

### 2.2. 了解vue-cli脚手架搭建项目

### 2.3. 掌握vue的单文件组件功能

### 2.4. vue-router路由

## 3. 课程内容

### 3.1. vue.js实现项目登录页功能

### 3.2. vue-cli项目搭建

#### 3.2.1. npm

NPM是随同NodeJS一起安装的包管理工具, 能解决NodeJS代码部署上的很多问题, 常见的使用场景有以下几种:

允许用户从NPM服务器下载别人编写的第三方包到本地使用。

允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。

允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。

由于新版的nodejs已经集成了npm, 所以之前npm也一并安装好了。

#### 安装node.js

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型, 使其轻量又高效。

Node.js 的包管理器 npm, 是全球最大的开源库生态系统。

官网下载node.js: <https://nodejs.org/en/>

里面自带了npm

接着cmd进入命令行, 输入

npm -v, 看是否安装成功

#### 查看npm版本

cmd进入命令行:

输入 npm -v 查看npm版本信息

如未安装npm: 则显示

“npm”不是内部或外部命令

否则, 显示版本信息

#### 淘宝镜像npm

你可以使用我们定制的 cnpm (gzip 压缩支持) 命令行工具代替默认的 npm:

安装方法:

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

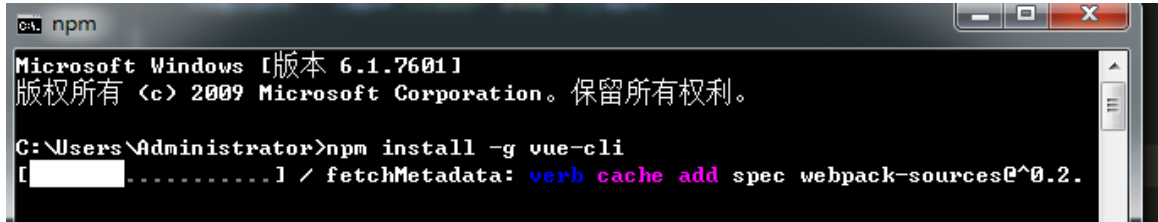
### 3.2.2. vue-cli

#### 安装cli脚手架

安装vue-cli 官方脚手架搭建工具

##### 1、直接全局安装

```
npm install -g vue-cli
```



2、然后打开npm命令行工具, cd到你想要的某个目录中, 并输入:

```
vue init <template-name> <project-name>
```

例:

```
vue init webpack "sell-app"
```

#### 配置脚手架

.配置项目信息

Project name: 项目名字

Project description: 项目描述

Author: 项目作者

Vue build: 创建方式, 直接回车选择官方推荐的

Install vue-router?yes 是否引入vue路由, 是

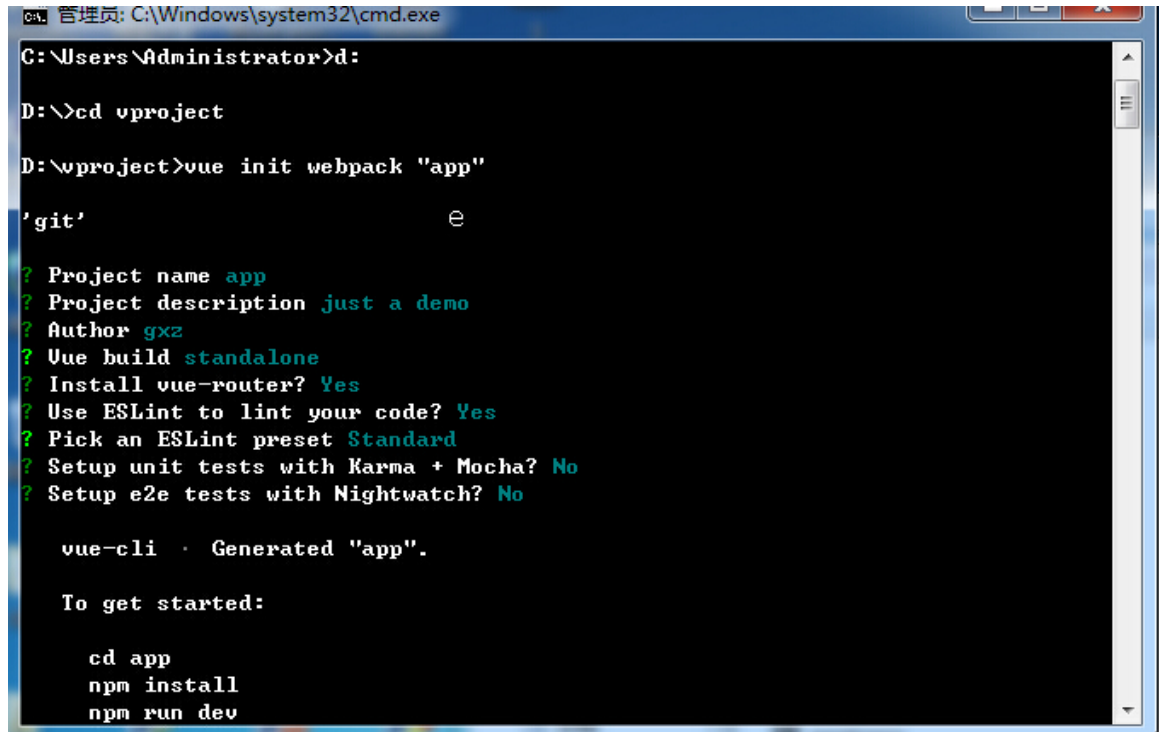
Use ESLint to lint your code?yes 是否引入eslint代码检查, 是

Pick an ESLint preset none esline风格, 选无

Setup unit tests with Karma + Mocha? no

引入单元测试工具, 否

Setup e2e tests with Nightwatch? 引入自动化测试工具, 否



```
C:\Users\Administrator>d:
D:\>cd vproject
D:\vproject>vue init webpack "app"

'git'                                e

? Project name app
? Project description just a demo
? Author gxz
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Setup unit tests with Karma + Mocha? No
? Setup e2e tests with Nightwatch? No

vue-cli · Generated "app".

To get started:

  cd app
  npm install
  npm run dev
```

配置完后会提示如何启动这个项目:

To get started:

cd app(进入项目文件夹)

npm install (安装所需依赖)

npm run dev (启动服务器)

自动打开项目页面

## 目录结构

build: 最终发布的代码存放位置。

`config`: 配置目录, 包括端口号等。我们初学可以使用默认的。

`node_modules`: npm 加载的项目依赖模块

`src`: 这里是我们要开发的目录, 基本上要做的事情都在这个目录里。里面包含了几个目录及文件:

`assets`: 放置一些图片, 如logo等。

`components`: 目录里面放了一个组件文件, 可以不用。

`App.vue`: 项目入口文件, 我们也可以直接将组件写这里, 而不使用 `components` 目录。

`main.js`: 项目的核心文件。

`static`: 静态资源目录, 如图片、字体等。

`test`: 初始测试目录, 可删除

`.xxxx`文件: 这些是一些配置文件, 包括语法配置, git配置等。

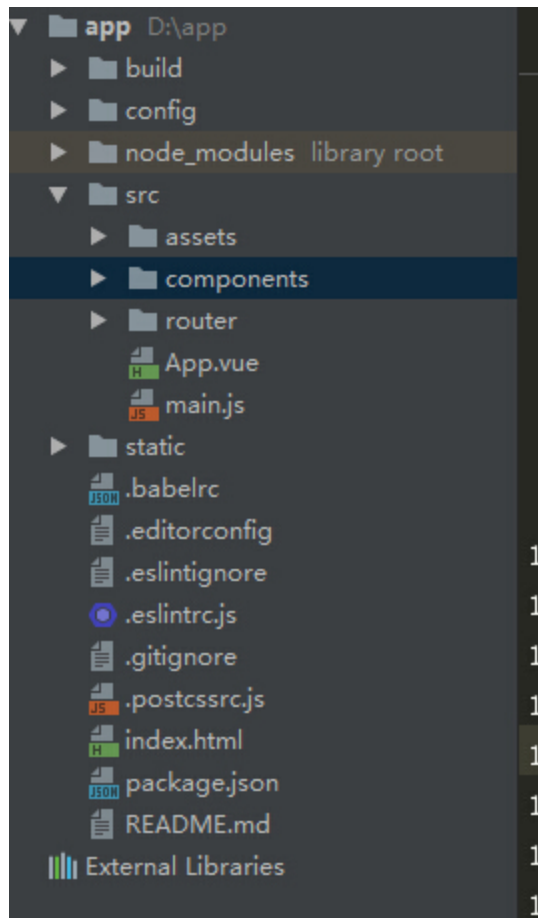
`index.html`: 首页入口文件, 你可以添加一些 meta 信息或同统计代码啥的。

`package.json`: 项目配置文件。

`README.md`: 项目的说明文档, markdown 格式

在Cli里创建组件, 和引用组件





在components下创建组件：



```
<template>
  <div class="header">
    hello world
  </div>
</template>

<script type="text/ecmascript-6">
  export default {};
</script>

<style scoped>
  .header {
    background-color: #42b983;
    font-size: 45px;
  }
</style>
```

在App.vue导入和应用组件：

```

<template>
  <div id="app">
    <!---->
    <!--<router-view></router-view>-->
    <v-header></v-header>
  </div>
</template>

<script>
  import header from './components/myFirstComponent.vue';

  export default {
    name: 'app',
    components: {
      'v-header': header
    }
  }
</script>

```

最后执行 `npm run dev`

### 3.3.vue单文件组件

在很多Vue项目中, 我们使用 `Vue.component` 来定义全局组件, 紧接着用 `new Vue({ el: '#container' })` 在每个页面内指定一个容器元素。

这种方式在很多中小规模的项目中运作的很好, 在这些项目里

JavaScript

只被用来加强特定的视图。但当在更复杂的项目中, 或者你的前端完全由

JavaScript

驱动的时候, 下面这些缺点将变得非常明显:

全局定义(Global definitions) 强制要求每个 component 中的命名不得重复

字符串模板(String templates) 缺乏语法高亮, 在 HTML 有多行的时候, 需要用到丑陋的“\”

template:"<div>\

p \

h \

span \

a \

```
</div>"
```

不支持CSS(No CSS support) 意味着当 HTML 和 JavaScript 组件化时, CSS 明显被遗漏  
没有构建步骤(No build step) 限制只能使用 HTML 和 ES5 JavaScript, 而不能使用预处理器, 如 Pug  
(formerly Jade) 和 Babel

但是文件扩展名为 .vue 的 single-file components(单文件组件)  
为以上所有问题提供了解决方法, 并且还可以使用 Webpack 或 Browserify 等构建工具。

### 3.3.1. 单文件组件的简单实例

单文件组件的格式

```
<template>  
  //组件模板  
</template>
```

```
<script>  
  //js逻辑  
</script>
```

```
<style>  
  //css样式  
</style>
```

```

<template>
  <h2>hello {{name}}</h2>
</template>

<script>
  module.exports = {
    data:function (){
      return {
        name:'斌哥'
      }
    }
  }
</script>

<style>
  h2{
    text-align: center;
    border-bottom: 1px solid #cccccc;
  }
</style>

```

现在我们获得：

完整语法高亮

CommonJS 模块

组件化的 CSS

### 3.3.2. 使用style、script标签

即便你不喜欢单文件组件，你仍然可以把 JavaScript、CSS 分离成独立的文件然后做到热重载和预编译。

```
<!-- my-component.vue -->
```

```
<template>
  <div>This will be pre-compiled</div>
</template>
<script src="/my-component.js"></script>
<style src="/my-component.css"></style>
```

### 3.4. vue-Router2路由模块

#### 3.4.1. 什么是SPA页面?

单页Web应用(single page web application, SPA), 就是只有一张Web页面的应用, 是加载单个HTML页面并在用户与应用程序交互时动态更新该页面的Web应用程序。

#### 3.4.2. 安装vue-router

Vue-router2中文文档

<http://router.vuejs.org/zh-cn/index.html>

cmd进入命令行:

输入 `npm install --save vue-router`

webpack会自动配置package.json

```
"dependencies": {
  "vue-router": "^2.3.1"
}
```

或者:

webpack里的package.json配置依赖:

```
"dependencies": {
  "vue-router": "^2.1.1"
}
```

cmd进入命令行: 输入 `npm install`

引入vue和vue-router:

```
import Vue from 'vue'
import VueRouter from 'vue-router'
```

应用VueRouter插件:

```
Vue.use(VueRouter);
```

### 3.4.3. 如何使用?

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <!--引用vue.js和vue-router.js-->
  <script src="https://unpkg.com/vue/dist/vue.js"></script>
  <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>

  <style>
    .black {
      width: 100px;
      height: 100px;
      background: #000;
    }
    .red {
      width: 100px ;
      height: 100px;
      background: red;
    }
  </style>
</head>
<body>
  <div id="app">
    <!--用router-link和router-view引用router-->
    <router-link to="/app/black">black</router-link>
    <router-link to="/app/red">red</router-link>
    <router-view></router-view>
  </div>
  <script>
    /* 定义两个组件*/
    const black = {
      template: "<div class='black'></div>"
    };
    const red = {
      template: "<div class='red'></div>"
    };
    /* 配置路由字典*/
    const router = new VueRouter({
      routes: [
        { path: '/app/black', component: black },
```

```
        { path: '/app/red', component: red }
      ]
    });
    /* 在实例Vue注册router,并用$mount挂载根元素#app*/
    const app = new Vue({
      router
    }).$mount("#app");
  </script>
</body>
</html>
```

## HTML部分

**<router-link to="url"></router-link>**

自动转化为<a>标签

**<router-view></router-view>**

显示<a>标签跳转到的路径页面



## JS部分

实例: `var router = new vueRouter({ routers: [ ] });`

注册router:

### 3.4.4. 如何在cli里使用，创建SPA页面

创建两个组件:red.vue和blue.vue

```
<template>
  <div class="red">
    <p>我是红色页面</p>
  </div>
</template>

<script type="text/ecmascript-6">
  export default {};
</script>

<style>
  .red {
    color: #fff;
    background-color: red;
  }
</style>
```

```
<template>
  <div class="blue">
    <p>我是蓝色页面</p>
  </div>
</template>

<script type="text/ecmascript-6">
  export default {};
</script>

<style>
  .blue {
    color: #fff;
    background-color: blue;
  }
</style>
```

跟着在router/index.js配置路由字典：

```
import Vue from 'vue';
import Router from 'vue-router';
import red from '../components/red.vue';
import blue from '../components/blue.vue';

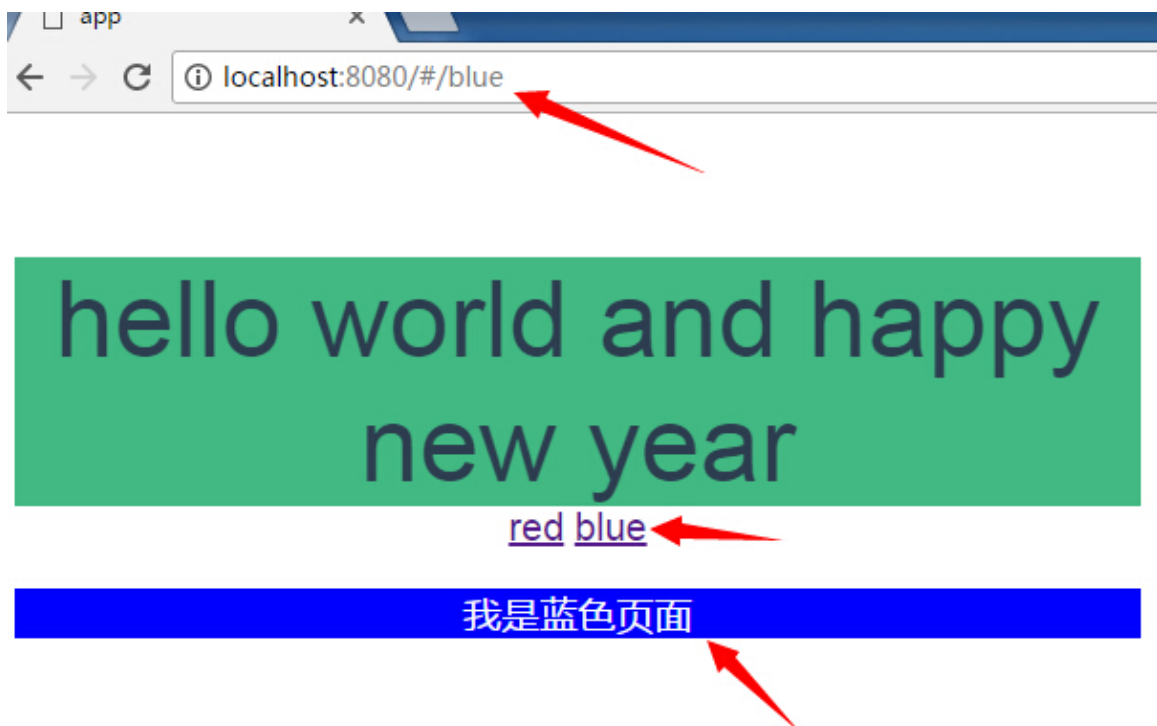
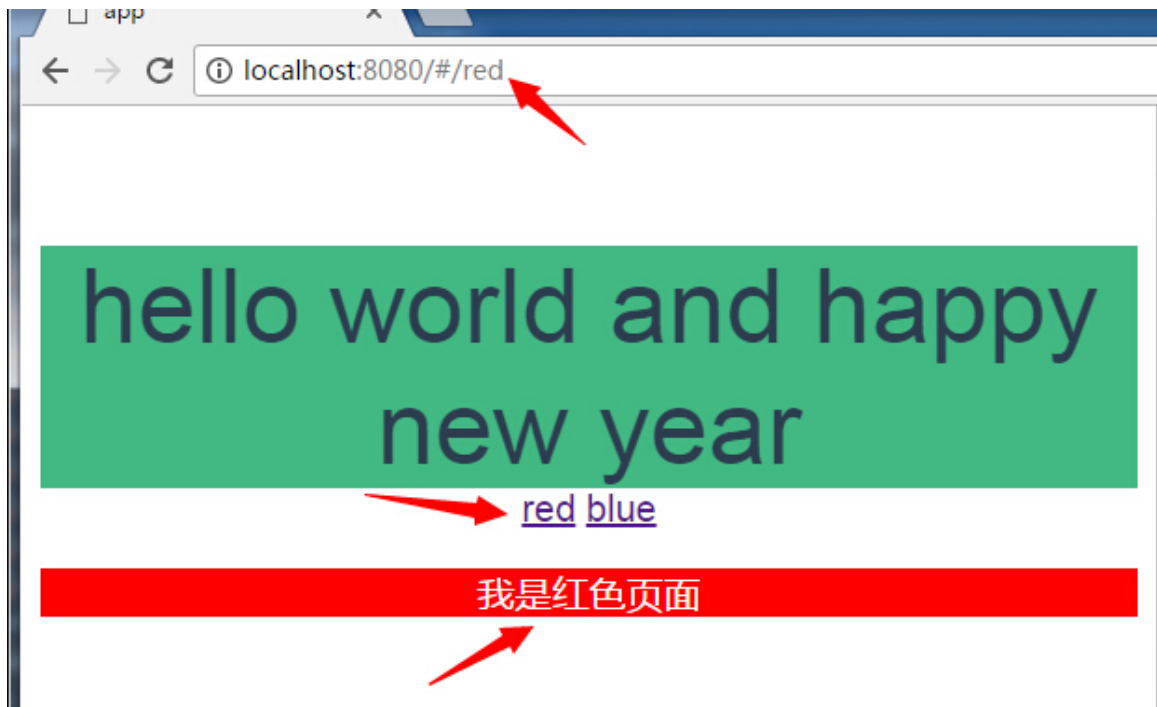
Vue.use(Router);

export default new Router({
  routes: [
    {
      path: '/red',
      component: red
    },
    {
      path: '/blue',
      component: blue
    }
  ]
})
```

在app.vue里引用:

```
<template>
  <div id="app">
    <!---->
    <!--<router-view></router-view>-->
    <v-header></v-header>
    <router-link to="/red">red</router-link>
    <router-link to="/blue">blue</router-link>
    <router-view></router-view>
  </div>
</template>
```

效果:



#### 4. 课后作业

#### 4.1. 作业1

作业1:

- 1、完成课堂项目

#### 4.2. 作业2

作业2:

- 1、了解vue-cli脚手架的搭建

#### 4.3. 作业3

作业3:

- 1、了解vue的单文件组件以及单文件组件的结构

#### 4.4. 作业4

作业4:

- 1、练习课堂代码

### 5. 当天小结

#### 5.1. 食天下App登录页功能的实现

#### 5.2. 了解vue-cli脚手架搭建项目

#### 5.3. 掌握vue的单文件组件功能

#### 5.4. vue-router路由