前端开发系列

学习参考网址

主题	相关链接
前端基础	基础教程:html、css、js、jquery 阮一峰 js 教程:个人推荐
vue 相关	vue.js: 官网，熟悉基础语法 vue-cli:官方脚手架 vue-router:路由 vuex:状态管理 element-ui:pc 端 UI 框架 vant-ui:h5UI 框架
拓展	es6:熟悉基础语法 webpack:模块化打包工具 axios:数据请求 scss:CSS 扩展语言

vue 培训

在传统 web 开发中，我们搭建项目都以 html 结构为基础，需要操作 dom 元素然后通过 jquery 或者 js 来添加各种交互功能，一旦项目改动或者项目工程较大，代码的修改将是复杂繁琐的。

vue 主要优点有两个：

1. 数据绑定：vue 会根据对应的元素，进行设置元素数据，通过输入框，以及 get 获取数据等多种方式进行数据的实时绑定，进行网页及应用的数据渲染
2. 组件式开发：通过 vue 的模块封装，它可以将一个 web 开发中设计的各种模块进行拆分，变成单独的组件，然后通过数据绑定，调用对应模版组件，同时传入参数，即可完成对整个项目的开发。 我们大多数系统都是表单项繁多且内容需要根据用户的操作进行修改的，使用 vue 就可以不用像之前使用 jquery 那样频繁操作 dom 元素，关注数据变化，开发起来也更容易一些。

==前提==:已了解关于 HTML、CSS 和 JavaScript 的中级知识,同时对 `es6` 的语法有一定了解。

根据实际开发，做了这个demo 代码，采用 vue-cli3 官方脚手架(之前重构使用的 vue-cli2，直观区别是不需要关注 webpack 配置,以及框架目录结构的区别),接口请求使用 axios

```
# 前提：安装nodejs和git
- node: https://nodejs.org/en/download/
- git: https://git-scm.com/downloads

# 下载代码
git clone https://github.com/JMjiayou8/vue-example

# 安装依赖包
cd vue-example
```

```
npm install

# 安装淘宝镜像，可以加快安装依赖包的速度，后期使用cnpm isntall xxx
npm install -g cnpm --registry=https://registry.npm.taobao.org
cnpm install

# 启动项目
npm run serve

# 浏览器运行
http://localhost:8080/
```

上手开发

1. 工具

日常编辑器我使用的是vscode,谷歌浏览器安装Vue Devtools插件

1. 从零安装

如果下载了 demo 可以跳过这一步，也可以根据以下步骤自己实现。

```


# 安装最新稳定版vue
npm install vue -g

# 安装vue-cli3
npm install -g @vue/cli

# 创建一个项目
# 会提示选择一个preset,第一次先选择default熟悉一下
vue create vue-example
cd vue-example

# 安装依赖(create时选择default则需要自己装)
# 可以利用官方Vue CLI插件安装一些包，使用vue add xxx，会调用插件的文件生成器自动生成一套文件，但是器很有可能更改你现有的文件内容。所以调用前最好提交本地代码。
# 或者直接npm install xxx;这样需要自己创建依赖的文件
# 安装淘宝镜像，可以加快安装依赖包的速度，后期直接使用cnpm isntall xxx
npm install -g cnpm --registry=https://registry.npm.taobao.org

vue add route 或者 cnpm install vue-router --save
vue add vuex 或者 cnpm install vuex --save
vue add element  或者 cnpm install element-ui --save

# 其它
# 安装css预处理器sass
cnpm install -D sass-loader node-sass

# 启动项目
npm run serve
```

1. 目录结构

一般开发过程中只需要在`src`目录下操作

```
<!--默认有下面四个文件-->
assets:资源文件夹，包括图片、css样式文件、字体文件等
components:自定义组件
App.vue：主页面
main.js：页面js入口
<!--开发过程中按需增加-->
views:存放页面vue文件
apis:存放数据接口请求相关文件
route:存放路由文件 运行vue add router自动生成
store:存放store配置 运行vue add vuex自动生成
plugins:存放第三方插件，运行vue add element自动生成，后期可加入其它插件
...
```

`public`是静态资源目录，一般不太需要管，多数情况是把一些需要在根目录访问的文件多放置到它里面，打包后直接复制到 dist 中，不经过 webpack 处理

其余常用的多数是些配置文件

```
vue.config.js：配置webpack
babel.config.js：配置babel
...
```

1. 新建一个页面 pageA

2. demo 中的 pageA 大致罗列了基础语法的使用

```
# 在views文件夹中新建一个pageA.vue
# template中只包含一级dom元素，不合理的如：<template><div>...</div><div>...
</div></template>

<template>
  <div class="pageA">
    <p>{{msg}}</p>
  </div>
</template>
<script>
export default {
  data () {
    return {
      msg: 'hello pageA'
    }
  }
}
</script>

# router.js或者router/index.js新增路由配置
{
  path: '/pageA',
```

```
  name: 'pageA',
  component: () => import('@/views/pageA.vue')
},

# 浏览器访问http://localhost:8080/pageA查看实例
```

1. 新建一个组件

```
# 在components中新建compA.vue
<template>
  <div class="compA">
    <p>父组件传递的props参数:{{msg}}</p>
    <button @click="emitData">点击按钮，将子组件属性值传递给父元素</button>
  </div>
</template>

<script>
  export default {
    name: 'compA',
    props: {
      msg: String
    },
    data() {
      return {
        comData: 'compAData'
      }
    },
    methods: {
      emitData() {
        let self = this
        self.$emit('getComAData', self.comData)
      }
    }
  }
</script>

# 在views/pageB.vue引入compA
<template>
  <div class="pageB">
    <p class="page-title">基础组件用法</p>
    <compA :msg="msg" @getComAData="getComADataFunc"></compA>
    <p>显示子组件传递的值：{{comData}}</p>
  </div>
</template>
<script>
  import compA from '@/components/compA'
  export default {
    components: { compA },
    data() {
      return {
        msg: 'hello pageB',
        comData: ''
```

```
      }
    },
    methods: {
      getComADataFunc(data) {
        this.comData = data
      }
    }
  }
</script>

- 组件间传值 父组件向子组件传值，一般在子组件中定义props接收即可。
子组件向父组件传值，一般在子组件中定义一个函数再使用$emit,再在父组件中定义函数接收。
# 浏览器访问http://localhost:8080/pageB查看实例
```

1. 封装一个 form 表单组件样例

```
# src/components/formCom.vue
组件文件中对父元素传递的props进行整合

# src/views/formPage.vue
页面父元素中定义表单结构以及数据

# 浏览器访问http://localhost:8080/formPage查看实例
```

1. 通过 axios 请求数据

```
# 安装axios
npm install axios --save

#新建src/apis/config.js存放全局axios配置

#新建src/apis/index.js存放项目接口请求

# src/views/axiosDemo.vue文件模拟请求数据渲染页面
# 需要额外熟悉一下async/await语法糖

# 浏览器访问http://localhost:8080/axiosDemo查看实例
```

1. 生命周期

大致了解一下，用的比较多的是`created`,用于页面初始化发送请求;`mounted`一般用来处理一些需要 dom 的操作，比如挂在 echarts 实例等;`beforeDestroy`一般用来在离开页面时的操作，比如清除页面里的定时器操作等。

额外补充

1. 使用 UI 组件库

2. pc:element-ui

```
# 安装
npm install element-ui --save
```

```
# main.js
import Element from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

Vue.use(Element, {size: 'mini'})

# 浏览器访问http://localhost:8080/elementPage查看效果
# 引入框架自带组件、自封装组件
```

1. 路由vue-router

```
# 安装vue-router
vue add router

# main.js中会新增以下内容
import router from './router'
new Vue({
  ...
  router,
  render: h => h(App)
}).$mount('#app')

# 自动生成router.js

# App.vue定义router-view
<template>
  <div id="app">
    <router-view></router-view>
  </div>
</template>
```

1. 状态管理vuex:通俗一点讲，就是一个全局的数据管理

```
# 安装vuex
vue add vuex

# main.js中会新增以下内容
import store from './store'
new Vue({
  ...
  store,
  render: h => h(App)
}).$mount('#app')

# 自动生成store.js

# 浏览器访问http://localhost:8080/vuexPage查看实例
```

1. webpack

vue-cli3 脚手架中 webpack 的配置比较完善，日常不需要太关注。 可以新建`vue.config.js`文件进行额外配置。

```
# vue.config.js
module.exports = {

}
```

1. scss

引入 scss

基础:html+css+js+jquery

html

- 个人觉得大概掌握以下两点就够了。
- html 文档的基本组成结构,引入文件
- html 常用标签，以及标签的语义化

css

1. 基础：设置字体大小，颜色,边框、背景等
2. 常用：浮动 float,定位 position,布局 flex(兼容性)
3. 进阶：动画

js

- 基础：掌握常用的数据处理方法，判断数据类型，this 指向等
- 高阶：闭包，原型

jquery

- 基础：元素选择器，设置属性，绑定操作
- ajax 请求

es6

- 看阮一峰的文档，了解和 es5 的差异
- 基础的语法，如 let,const,扩展运算符...,箭头函数等等
- 新增的函数