VueX和网站开发的若干问题

Vuex是一个专为Vue.js应用程序开发的状态管理模式+库，专为解决下面两个问题：

- 多个视图依赖于同一状态。
- 来自不同视图的行为需要变更同一状态。

核心内容（成员列表）：

State: 数据源存放状态

mutations state: 成员操作

Getters: 加工state成员给外界

Modules: 模块化状态管理

Vuex核心流程图如下

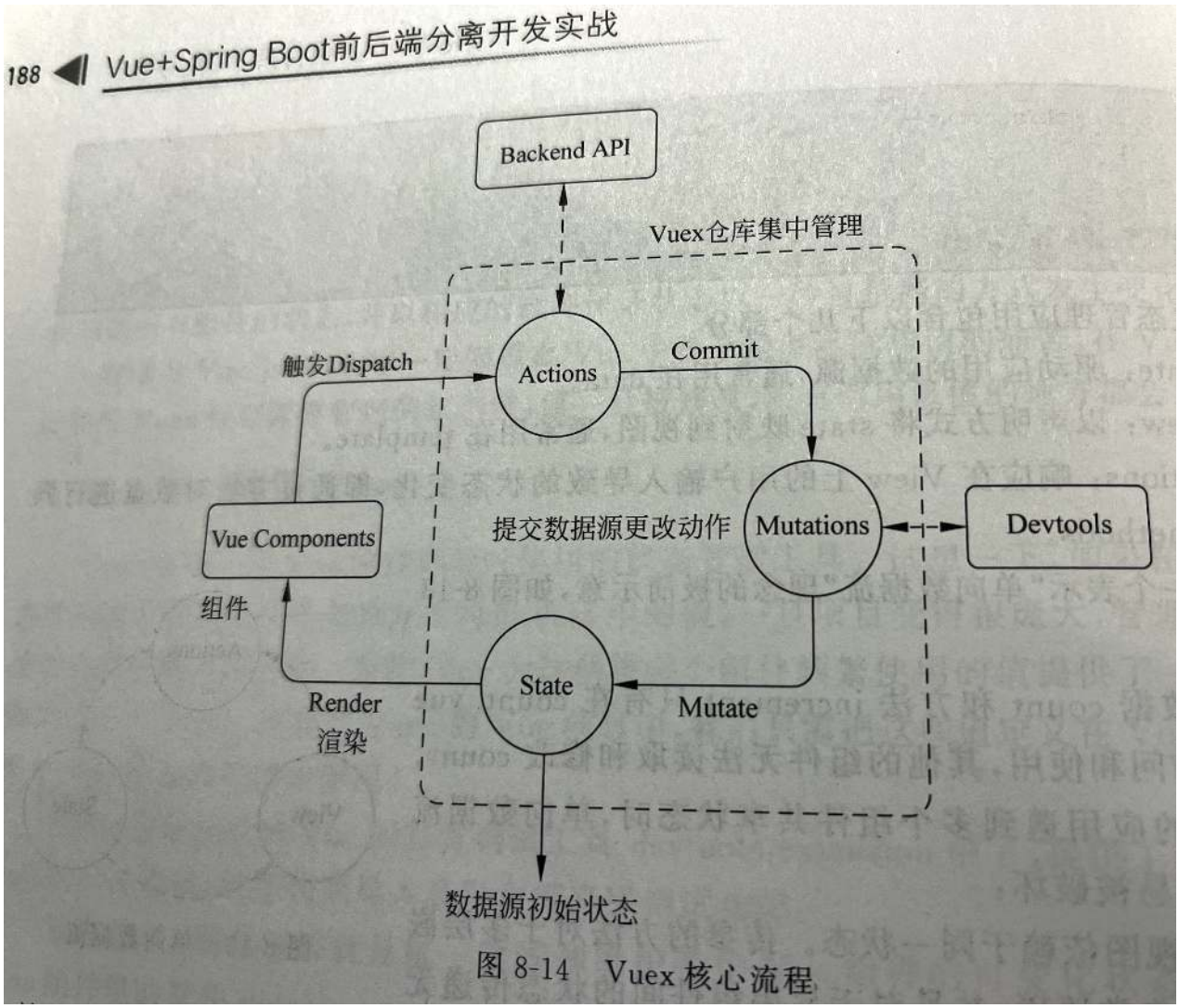


图 8-14 Vuex 核心流程

>安装Vuex

```
npm install vuex -S
```

>引用

```
import Vuex from 'vuex'
Vue.use(Vuex)
```

>创建Vuex实例对象，类比创建Vue实例对象

```
var store = new Vuex.Store({
    state: {
        //相当于Vue中的data
        num: 10
    }
})
export default store //要遵循common.js规范，使用export default导出store模块
```

>在main.js中引入模块即可

```
import store from './store'
```

>挂载到Vue实例中

```
new Vue({
    el: '#app',
    router,
    store,//将Vuex实例挂载到Vue对象
    components: {App},
    template: '<App/>'
})
```

举例：

提交单个值

```
this.$store.commit('edit',18)
```

添加成员

```
Vue.set(state,"age",18)
```

删除成员

```
Vue.delete(state,'age')
```

调用取值

```
this.$store.getters.fullInfo
```

这样所有组件都可以使用Vuex中的值，取值代码如下：

```
{{$store.state.num}}
```

Mutations事件状态

```
//mutations操作同步数据，无法操作异步数据
```

```
  mutations: {
   // 事件类型 type 为 increment
  increment (state) {
    // 变更状态
    state.count++
  }
}
```

举例：

Vuex代码

```
new Vue({
    state: {
        num: 10
    },
    getters: {
        newnum(state){
            return state.num + '元'
        }
    },
    mutations:{
        getAdd(state){
            state.num++
        }
    }
})
export default store
```

组件中代码

```
...
<input type="text" :value="$store.state.num"/>
<input type="button" :value="+1" @click="addnum"/>
...
methods: {
    addnum(){
        this.$store.commit('getAdd')
        //带参数传递
        //this.$store.commit('getAdd',100)
        //传递多个参数（对象）
        //this.$store.commit('getAdd',{id:1,name:'小明'})
    }
}
```

Vuex-Actions异步操作数据

```
actions:{
    newgetAdd(content,data){
        setTimeout(()=>{
            content.commit('getAdd',data)
```

```
        },1000)
    }
}

methods: {
    add(){
        this.$store.dispatch('newgetAdd',1000)
    }
}
```

辅助函数mapState、mapGetters、mapMutations和mapActions

它们对应操作state、getters、mutations和actions中的数据或方法，简化调用方式，mapState和mapGetters在computed中使用。

mapMutations的作用是把store中mutation内的方法映射到组件methods属性中，可以在组件中直接使用mutation里面的方法。mapActions类似mapMutations，把store中actions内的方法映射到组件methods属性中，可以在组件中直接使用actions里边的方法，mapMutations和mapActions在methods中使用。

定义响应式数据

1、ref函数：将普通数据变成响应式数据。

在数据层（js中），通过.value获取数据；在视图层（html），不需要.value就能获取数据。

2、reactive函数：将对象和数组这类复合数据类型数据变成响应式数据。

> 为了提高代码的复用性，在Vue3.x新增的setup()函数，是Composition API的入口函数，可将每个功能写在一起。

依赖注入

在父组件中使用provide()函数，向后代传递数据；

在后代组件中使用inject()函数，获取后代传递过来的数据。

跨域问题

前端跨域配置如下：

8.3.3 跨域处理

跨域是指浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器对 JavaScript 实施的安全限制。

同源策略：是指协议、域名、端口都要相同，其中有一个不同便会产生跨域。

如我们调用百度音乐的接口，以此获取音乐数据列表，这必然会出现跨域问题。百度音乐的接口完整网址：http://tingapi.ting.baidu.com/v1/restserver/ting?method=baidu.ting.billboard.billList&type=1&size=10&offset=0，如图 8-11 所示。

在 Vue 中使用本地代理的方式进行跨域处理。首先在配置文件 config/index.js 里设置代理，在 proxyTable 中添加如下代码：

```
//config/index.js 部分代码省略
module.exports = {
  dev: {
    proxyTable: {
      '/api': {
    //目标路径，别忘了添加 http 和端口号
        target: 'http://tingapi.ting.baidu.com',
        changeOrigin: true,     //是否跨域，true 为跨域
        pathRewrite: {
          '^SymbolYCp/api': ''   //重写路径
        }
      }
    }
  }};
```

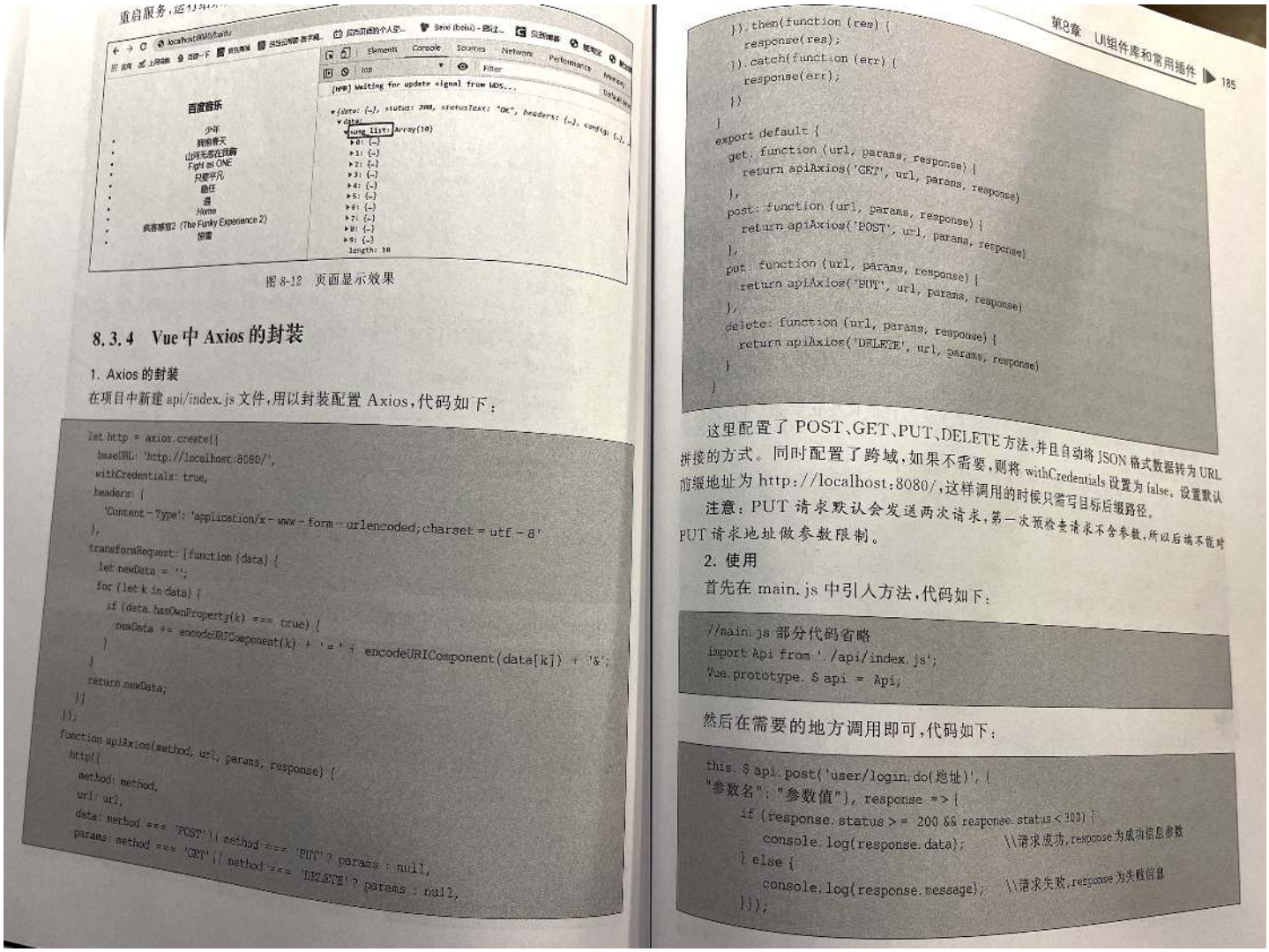


图 8-12　页面显示效果

8.3.4　Vue 中 Axios 的封装

1. Axios 的封装

在项目中新建 api/index.js 文件，用以封装配置 Axios，代码如下：

```
let http = axios.create({
  baseURL: 'http://localhost:8080/',
  withCredentials: true,
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8'
  },
  transformRequest: [function (data) {
    let newData = '';
    for (let k in data) {
      if (data.hasOwnProperty(k) === true) {
        newData += encodeURIComponent(k) + '=' + encodeURIComponent(data[k]) + '&';
      }
    }
    return newData;
  }]
});
function apiAxios(method, url, params, response) {
  http({
    method: method,
    url: url,
    data: method === 'POST' || method === 'PUT' ? params : null,
    params: method === 'GET' || method === 'DELETE' ? params : null,
  }).then(function (res) {
    response(res);
  }).catch(function (err) {
    response(err);
  })
}
export default {
  get: function (url, params, response) {
    return apiAxios('GET', url, params, response)
  },
  post: function (url, params, response) {
    return apiAxios('POST', url, params, response)
  },
  put: function (url, params, response) {
    return apiAxios('PUT', url, params, response)
  },
  delete: function (url, params, response) {
    return apiAxios('DELETE', url, params, response)
  }
}
```

这里配置了 POST、GET、PUT、DELETE 方法，并且自动将 JSON 格式数据转为 URL 拼接的方式。同时配置了跨域，如果不需要，则将 withCredentials 设置为 false。设置默认前缀地址为 http://localhost:8080/，这样调用的时候只需写目标后缀路径。

注意：PUT 请求默认会发送两次请求，第一次预检查请求不含参数，所以后端不能对 PUT 请求地址做参数限制。

2. 使用

首先在 main.js 中引入方法，代码如下：

```
//main.js 部分代码省略
import Api from './api/index.js';
Vue.prototype.$api = Api;
```

然后在需要的地方调用即可，代码如下：

```
this.$api.post('user/login.do(地址)',{
"参数名":"参数值"}, response => {
    if (response.status >= 200 && response.status < 300) {
        console.log(response.data);        \\请求成功,response为成功信息参数
    } else {
        console.log(response.message);     \\请求失败,response为失败信息
    }});
```

在java后端可用配置类设置跨域

```java
public class WebMvcConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
//                .allowedOrigins("http://119.37.199.26", "http://localhost", "http://127.0.0.1")
                .allowedOrigins("*")
                .allowedMethods("*")
                .allowedHeaders("*");
    }
}
```

前端数据格式转化工具qs

下载npm i qs -D

引入import qs from ‘qs’

方法一：

```
qs.stringify() 转换成查询字符串
let comments = {content: this.inputValue}
let comValue = qs.stringify(comments)
```

方法二：

```
qs.parse() 转换成json对象
let comValue = qs.parse(comments)
```

sessionStorage和localStorage

sessionStorage的生命周期是在浏览器关闭前。

使用window.open打开页面和改变location.href方式都可以获取到sessionStorage内部的数据。

localStorage的生命周期是永久性的。关闭浏览器也不会让数据消失， 除非主动的删除数据。

> 事件修饰符
>
> event.offsetX
>
> event.offsetY
>
> 以上两个结合获取当前鼠标定位坐标
>
> event.stopPropagation()阻止冒泡事件方法

路由和请求router.js

路由传参

1、router-link标签

```
{path:"/",component:ShowBlogs}
   <router-link to="/"></router-link>
<router-link v-bind:to="'/blog'+blog-id">
```

```
<router-link to="/login?id=1">登录</router-link>
```

2、修改匹配规则

```
path: '/login/:id'
```

{path:'*',redirect:'/'}写在其他路由后边，实现遇到不存在路由重定向到'/'。

> 路由嵌套（二级路由）

ES6 过滤filter

过滤器用于数据输出之前的处理，分为私有过滤器和全局过滤器

创建数组，判断数组是否包含某值

```
var newarr = [
  { num: 1, val: 'ceshi', flag: 'aa' },
  { num: 2, val: 'ceshi2', flag: 'aa2'  }
]
console.log(newarr.filter(item => item.num===2 ))
```

去除空字符串

```
var arr = ['1','2',null, '3.jpg',null]
var newArr = arr.filter(item => item)
console.log(newArr)
```

数组去重

```
var arr = [1, 2, 2, 3, 4, 5, 5, 6, 7, 7,8,8,0,8,6,3,4,56,2];
var arr2 = arr.filter((x, index,self)=>self.indexOf(x)===index)
console.log(arr2); //[1, 2, 3, 4, 5, 6, 7, 8, 0, 56]
```

> 全局过滤器在main.js中定义。

watch侦听属性，用于侦听data中数据的变化，只要有变化，就会触发watch侦听属性，侦听属性的方法名要与侦听的变量名一致。

页面局部刷新

```
<template>
  <div id="app">
    <router-view :key="key" v-if="isRouterAlive" />
  </div>
</template>
<script>
  export default {
    name: 'App',
    provide() {
      return {
        // 调用下面reload 的方法
        reload: this.reload
      };
    },
    data() {
      // 初始化isRouterAlive
      return {
        isRouterAlive: true
      };
    },
      computed: {
        key(){
            return this.$route.path+Math.random();
          }
      },
    methods: {
      reload() {
        this.isRouterAlive = false;
        this.$nextTick(function() {
```

```
            this.isRouterAlive = true;
          }
          )
        }
      }

}
</script>
```

Vue内嵌pdf

可以参考教程https://blog.csdn.net/heavenz19/article/details/121695072

step1、安装vue-pdf

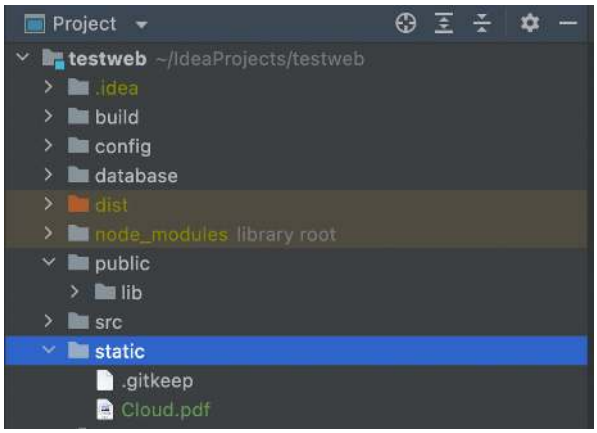
```
npm install --save vue-pdf
```

step2、在组件中使用

```
<template>
  <div id="pdf">
    <pdf
      :src="pdfUrl">
    </pdf>
  </div>
</template>

<script>
import pdf from 'vue-pdf'
export default {
  name: "animal",
  components: {
    pdf
  },
  data() {
    return {
      pdfUrl: "/static/Cloud.pdf"
    }
  }
}
```

其中，pdf文件放在如下目录



但效果上只能显示第一页，后期会在项目实战教程中展示如何分页。

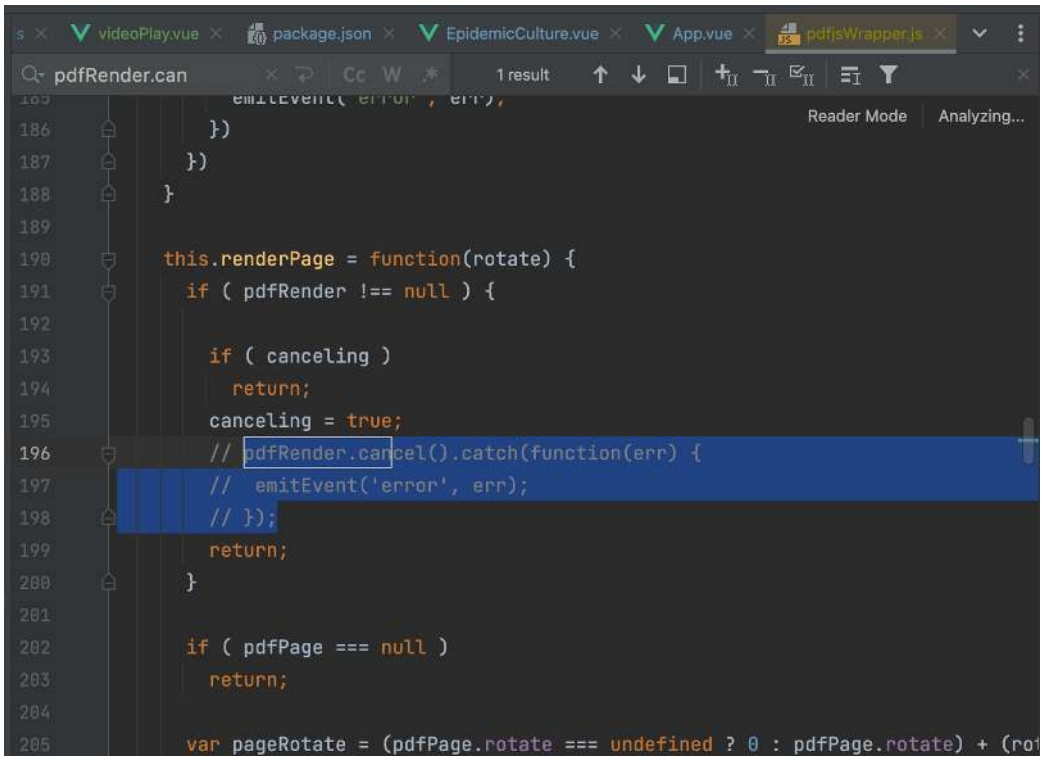
另外，进度条方式展示pdf方式如下

```
<iframe src="/system/resource/pdfjs/viewer.html?file=%2F__local%2F9%2F1C%2FC9%2FA72652E5A6B58F316A0
width="100%" height="2500" style="border:1px solid #DDDDDD"></iframe>
```

为解决如下报错：

```
Cannot read properties of undefined (reading 'catch')"
```

采用的方式是注释以下语句



页面嵌入视频播放功能

在 Vue 中通过 videojs和vue-video-player 均可以实现

```
npm i video.js
```

或者

```
npm install vue-video-player --save
```

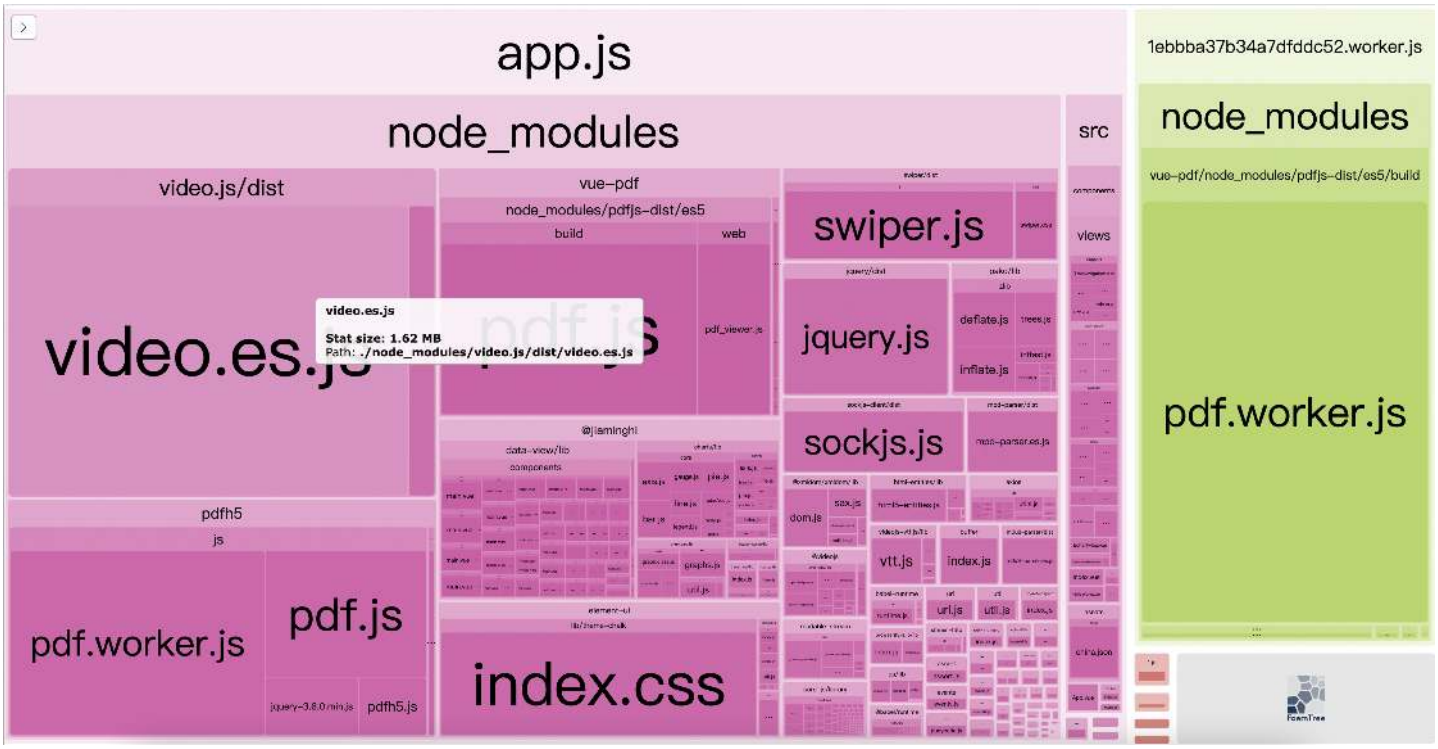
详见后期实战项目教程。

UI组件库

VUE的UI组件库可以是响应式的Bootstrap、饿了么的移动Mint UI和PC端的Element UI。

webpack-bundle-analyzer 插件的使用

在项目上线时遇到的问题：首页访问要加载app.js超过5M，加载超过8s。于是借助于webpack-bundle-analyzer 获取以下分析视图。



首先命令安装：

```
# NPM
npm install --save-dev webpack-bundle-analyzer
# Yarn
yarn add -D webpack-bundle-analyzer
```

在使用的webpack.config.js 文件（webpack.dev.conf.js）配置

```
// webpack.config.js 文件
const BundleAnalyzerPlugin = require('webpack-bundle-analyzer').BundleAnalyzerPlugin
module.exports={
  plugins: [
    new BundleAnalyzerPlugin()  // 使用默认配置
    // 默认配置的具体配置项
    // new BundleAnalyzerPlugin({
    //   analyzerMode: 'server',
    //   analyzerHost: '127.0.0.1',
    //   analyzerPort: '8888',
    //   reportFilename: 'report.html',
    //   defaultSizes: 'parsed',
    //   openAnalyzer: true,
    //   generateStatsFile: false,
    //   statsFilename: 'stats.json',
    //   statsOptions: null,
    //   excludeAssets: null,
    //   logLevel: info
    // })
  ]
}
```

配置package.json

```
{
 "scripts": {
    "dev": "webpack --config webpack.dev.js --progress"
  }

```

实际接入时：

```
"scripts": {
  "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js --host 0.0.0.0"
  "start": "npm run dev",
  "build": "node build/build.js"
},
```

通过npm run dev，以上视图在localhost:8888可以访问到。