序言

1、同是mvvm框架，为什么选择vue?

2、angularjs、vuejs数据更新的核心原理是什么?

目录结构

```
├── .babelrc                babel转义配置
├── .editorconfig           编辑格式配置
├── .gitignore              git管理忽略
├── .postcssrc.js           postcss配置
├── index.html              页面基础layout
├── package.json            npm包管理文档
├── build                   webpack打包配置管理
|   ├── build.js            线上打包配置
|   ├── check-versions.js   检车依赖node,npm版本号
|   ├── dev-client.js       本地开发连接客户端
|   ├── dev-server.js       本地开发服务
|   ├── utils               打包常用工具函数
|   ├── vue-loader.conf     vue-loader配置
|   ├── webpack.base.conf   webpack基础配置
|   ├── webpack.dev.conf    webpack dev配置
|   └── webpack.prod.conf   webpack prod配置
├── config                  配置相关
|   ├── dev.env.js          dev环境配置
|   ├── index.js            webpack配置文件
|   ├── mock-filter.js      mock数据过滤接口配置
|   └── prod.env.js         prod环境配置
├── mock                    mock数据
|   └── user
|       └── listpage.json
├── static                  不需要编译涉及的静态资源
|   └── ...
└── src                     前端项目主要入口
    ├── components          公共vue组件
    |   └── breadcrumb
    |         └── index.vue
    ├── imgs                公共图片资源
    |    └── element-logo
    ├── router              vue路由相关
    |   └── index.js
    ├── store               vuex相关
    |   └── index.js
    ├── styles              公共样式相关
    |   └── index.sass
    ├── views               页面相关
    |   └── 页面名称
    |         ├── components  页面级组件
    |         |       └── index.vue
    |         ├── imgs        页面级imgs
    |         |     └── xxx.jpg
    |         └── 页面.vue
    ├── app.vue             vue页面layout
    └── main.js             vue打包入口
```

规范

> 1.新增的页面和二次开发的组件，按目前的文件夹对应放置。
> 2.文件命名采用驼峰式命名(camelCase)，组件命名采用短横线分隔命名(kebab-case)，定义组件的时候尽量避免与现有组件同名。
> 3.每个.vue文件，包含template、script、style三个部分，没有的可以不写。
> 4.每个<template>下只能有且只有一个元素。
> 5..vue文件中style需要注意的地方：
> 新建的vue文件，样式可以设置lang属性，不设置此属性的话默认是css，建议用less、scss、的语法，这样会使代码看起来更简洁，如lang="scss"。 无论是less、sass、还是scss， 都要安装相应的编译器才能够识别。
> 添加"scoped"属性：让样式只对当前文件有效，防止全局污染。
> 6.vue文件中script需要注意的地方：
> 框架中基本上使用的是es6、es7的语法，好处是代码看起来更简洁、清晰易读。
> 7.避免 this.$parent，谨慎使用 this.$refs
> 8.注释块必须以/**（至少两个星号）开头**/;
> 9.单行注释使用//;
> 10.调试信息console.log() debugger 使用完及时删除

组件的添加和使用

全局注册：

```
<!--html-->
<div id="example">
  <my-component></my-component>
</div>

<!-- js -->
// 注册
Vue.component('my-component', {
  template: '<div>A custom component!</div>'
})

// 创建根实例
new Vue({
  el: '#example'
})

<!-- html -->
<div id="example">
  <div>A custom component!</div>
</div>
```

局部注册：

```
<template>
        <navmain></navmain>
</template>
<script>
export default {
  components: {
    'navmain': require('@/views/layout/components/navmain.vue')
  }
}
</script>
```

详细请看：https://cn.vuejs.org/v2/guide/components.html

路由的自定义添加

1.安装vue-router

2.引入：`vue-router`

```
import Router from 'vue-router'
```

3.使用

```
Vue.use(Router)
```

4.自定义路由

```
export default new Router({
  routes: [
    {
      path: '/',
      name: '用户管理',
      component: index,
      children: [
        {
          path: 'userList',
          name: '用户列表',
          component: userList
        },
        {
          path: 'addUser',
          name: '添加用户',
          component: addUser
        }
      ]
    }
  ]
})
```

5.html中添加视图

```
<div id="app">
  <router-view></router-view>
</div>
```

官网教程：https://router.vuejs.org/zh-cn/

性能优化-懒加载

组件之间的数据通信

在vue2.0中分为：

父传子：Props
子传父：子：$emit(eventName) 父$on(eventName)
父访问子：ref
非父子组件通信：

vue2.0 移除：1.$dispatch() 2.$broadcast() 3.events

父组件传向子组件

父组件

```
<template>
        <hamburger class="hamburger-container" :toggleClick="toggleSideBar"
:isActive="sidebar.opened"></hamburger>
</template>
<script>
import Hamburger from '@/components/Hamburger'
export default {
  components: {
      Hamburger
  }
}
</script>
```

子组件

```
<template>
  <div>
    <svg t="1492500959545" @click="toggleClick" class="hamburger" :class="
{'is-active':isActive}">
      ...
    </svg>
  </div>
</template>

<script>
export default {
  name: 'hamburger',
  props: {
    isActive: {
      type: Boolean,
      default: false
    },
    toggleClick: {
      type: Function,
      default: null
    }
  }
}
</script>
<style scoped>
.hamburger {
        display: inline-block;
        cursor: pointer;
        width: 20px;
        height: 20px;
        transform: rotate(90deg);
        transition: .38s;
        transform-origin: 50% 50%;
}
.hamburger.is-active {
        transform: rotate(0deg);
}
</style>
```

子组件传向父组件

子组件$emit()触发，父组件$on()监听

```
<template>
  <div class="hello">
    <button v-on:click="onClickMe">telltofather</button>
  </div>
</template>

<script>
export default {
  methods: {
    onClickMe () {
      this.$emit('child-tell-me-something',this.msg)
    }
  }
}
</script>
```

父组件

```
<template>
<div id="app">
<p>child tell me: {{childWords}}</p>
<component-a v-on:child-tell-me-something='listenToMyBoy'></component-a>
</div>
</template>

<script>
import Store from './store'
import ComponentA from './components/componentA'
export default {
  components: {
    ComponentA
  },
  data () {
    return {
      childWords: ''
    }
  },
  methods: {
    listenToMyBoy (msg) {
      this.childWords = msg
    }
  }
}
</script>
```

父组件直接访问子组件

给子组件指定一个索引ID

```
<div id="parent">
  <user-profile ref="profile"></user-profile>
</div>
```

```
var parent = new Vue({ el: '#parent' })
// 访问子组件
var child = parent.$refs.profile
```

`$refs` 只在组件渲染完成后才填充，并且它是非响应式的。它仅仅作为一个直接访问子组件的应急方案 - 应当避免在模板或计算属性中使用 `$refs`。

非父子组件间的通信

使用一个空的 Vue 实例作为一个事件总线中心(central event bus):

```
var bus = new Vue()
```

```
// 在组件 A 的 methods 方法中触发事件
bus.$emit('id-selected', 1)
```

```
// 在组件 B 的 created 钩子函数中监听事件
bus.$on('id-selected', function (id) {
  // ...
})
```

复杂的场景应该考虑使用专门的状态管理模式。

vue方法放置顺序

```
1. components
2. props
3. data
4. created
5. mounted
6. activited
7. update
8. beforeRouteUpdate
9. filter
10. computed
11. watch
12. methods

*** filter、computed、methods里面都是定义方法，注意区分三者之间的用法和区别！！！
```

生命周期

调试 vue.js devtool

1.直接下载

2.github上下载源码

文档汇总

http://element-cn.eleme.io/#/zh-CN/component/installation element UI

https://cn.vuejs.org/v2/guide/installation.html vue2.x 教程

https://router.vuejs.org/zh-cn/ vue router

https://www.vue-js.com/ vue 中文社区

https://vuex.vuejs.org/zh-cn/index.html vuex文档

https://www.kancloud.cn/yunye/axios/234845 axios中文文档