

首页 / 技术 / 正文

搜索站内资讯、视频或用户

Vue真是太好了 壹万多字的Vue知识点 超详细!

Echa攻城狮 2020-04-02 11:40:08



让你减少加班的15条高效JS技巧【实践】

详细讲解移动端开发的屏幕、布局的兼容适配【图文】

手把手教你深入讲解前端缓存策略

推荐7个很棒的JavaScript产品



1□、Vue和其他两大框架的区别

- Angular 学习成本太高
 - React 代码可读性差
 - Vue 学习成本较低 很容易上手
- VUE官方: <https://cn.vuejs.org/v2/guide/comparison.html>

2□、Vue是什么

- Vue是一套用于构建用户界面的渐进式框架 "前端框架"
- 让程序员脱离自己操作DOM 专注于写逻辑和操作数据
- Vue的核心库只关注视图层 易上手 便于与第三方库或既有的项目整合
- 当与现代化的工具链以及各种支持的库结合使用时 Vue也完全能够为复杂的单页应用提供驱动

3□、MVVM

- M
model 数据
- V
view 页面
- VM
viewModel Vue实例



DOM监听(DOM Listeners)
数据绑定(Date Bindings)

4□、Vue指令

4□.1□ v-text

说明: 文本数据渲染 **用法:** v-text = "Vue实例中的数据" => 简写 {{Vue实例中的数据}}

相当于JavaScript中的innerText

4□.1□.2□ v-text指令 和 {{ }}插值表达式 的区别

- v-text 会直接替换元素中的所有内容
- {{ }} 只会替换插值表达式中的占位符

4□.2□ v-html

说明: HTML渲染数据 **用法:**v-html = "Vue实例中的数据" 会解析html结构 渲染至页面

相当于JavaScript中的innerHTML

4□.2□.1□ v-html指令 和 v-text指令的区别

- v-html 会将数据解析成html 渲染至页面
- v-text 只会输出成文本(字符串形式)

注意: 在网站上动态渲染任意的 HTML 是非常危险的!!! 因为容易导致 XSS 攻击 只在可信内容上使用 v-html 绝对不要用在用户通过(提交)的内容上使用

4□.3□ v-on

说明: 事件绑定(绑定事件监听器) **用法:** v-on:事件名 = "事件处理函数" => 简写 @事件名 = "事件处理函数"

4□.3□.1□ 详细用法

- @事件名.修饰符 = "事件处理函数"
- 逻辑比较少的可以直接写在行内
- 逻辑较多的写到 methods 中 注意: 操作Vue实例上的数据要跟上 this
- 可以通过实参传递(\$event) 获取事件参数e

\$event.target 获取当前事件触发的DOM元素 \$event.path[0](el.path[0]) 也可以获取当前事件触发的DOM元素 path数组中有从触发事件源的元素的所有上一级元素 直到window
实参不传递(没有任何参数) 默认在形参中第一个就是事件参数
实参传递 就必须传递\$event 来获取获取事件参数

4□.3□.2□ 事件修饰符



320



转发



微博



Qzone



微信

- .stop
阻止事件冒泡
- .prevent
阻止事件默认行为
- .once
只触发一次回调
- .native
监听组件根元素的原生事件
很重要! 有些第三方组件可能内部并没有设置原生的事件 就可以通过.native来触发事件

面试问及

之前在使用饿了么UI的时候给一个组件添加一个原生的事件 但是一直触发不了 后面查阅文档发现这个组件内部并没有注册我使用的原生事件 事件修饰符.native就可以直接监听并触发组件的原生事件

- .capture
添加事件侦听器时使用 capture 模式
- .{keyCode | keyAlias}
只当事件是从特定键触发时才触发回调
- .left
(2.2.0版本) 只当点击鼠标左键时才触发
- .right
(2.2.0版本) 只当点击鼠标右键时才触发
- .middle
(2.2.0版本) 只当点击鼠标中键时才触发
- .self
只当事件使用侦听器绑定的元素本身触发时才触发回调
- .passive
(2.3.0版本)以{ passive:true } 模式添加侦听器

4.4 v-bind

说明: 属性绑定(行内属性) **用法:** v-bind:属性名="Vue实例中的数据" => 简写 :属性名="Vue实例中的数据" **当Vue实例中的数据改变之后 页面会同步更新**

4.4.1 属性绑定修饰符

- .prop
被用于绑定 DOM 属性 (property)
- .camel
(2.1.0+) 将 kebab-case 特性名转换为 camelCase. (从 2.1.0 开始支持)
- .sync
(2.3.0+) 语法糖, 会扩展成一个更新父组件绑定值的 v-on 侦听器

4.4.2 对象的方式绑定class

- :class = "{ 'red' : isRed }"

isRed = true 就有red这个类 isRed = false 就没有red这个类 isRed 在 Vue 实例data中声明



320



转发



微博



Qzone



微信

- 默认的class 和 :class(绑定的class) 一起使用不会冲突 后面的会作为追加或者移除来解析

```
class = "red" :class = "{yellow' : isYellow}"
```

4.4.3 对象的方式绑定style

- :style = "{fontSize : mySize + 'px'}"
- 样式名需要使用驼峰命名法
- 后面的mySize需要在Vue实例data中定义

4.5 v-model

说明: 双向数据绑定 **用法:** v-model = "Vue实例中的数据"

4.5.1 双向

- 视图层
 - 数据层
- 数据能够自动的从内存中显示到页面上去

4.5.2 双向绑定修饰符

- .lazy
懒载入 当表单属性失去光标或按下回车键后 再将页面中的数据和Vue实例中的数据双向绑定
- .trim
输入的数据首位空格过滤
- .number
输入字符串转为有效的数字

注意: v-model 只能设置给form表单属性

4.6 v-for

说明: 循环渲染 **用法:** v-for = "(item,index) in items" :key = "index"

items是源数据数组 item是被迭代的数组元素的别名 index是被迭代的数组元素的下标(索引)

4.6.1 :key

- 数据唯一标识绑定
- v-for默认行为试着不改变整体 而是替换元素 迫使其重新排序的元素需要提供一个key(用户删除数据后 数据需重新排列序号 就可以使用key来实现)
- 数据实现重用和重新排序现有的元素
- 值最好为字符串或数值类型(唯一的)

 320

 转发

 微博

 Qzone

 微信

4.6.2 遍历数组注意点

- Vue本身只是监视了Vue实例Data中数组的改变(监视内存地址) 没有监视数组内部数据的改变
- (变异方法)Vue重写了数组中的一系列改变数组内部数据的方法(先调用原生的方法 再更新页面) 所以实现了使用数组提供的方法 数组内部改变 界面自动改变

push() pop() shift() unshift() splice() sort() reverse() ...

- this.arr[index] = 新值

这种修改数组中的元素是无法实现数据改变后页面会同步改变(只会修改data中的数据 但是页面不会同步改变)

- splice()的增删改
增 this.arr.splice(index,0,新值)
删 this.arr.splice(index,1)
改 this.arr.splice(index,1,新值)

4.7 v-if, v-else, v-else-if

说明: 条件(表达式或布尔值)判断渲染 **用法:** v-if = "表达式或布尔值" v-else-if = "表达式或布尔值" v-else

4.7.1 注意

v-if 和 v-else-if 后面必须跟表达式或布尔值 v-else-if 和 v-else 不能独立使用 必须跟在 v-if 后面

4.8 v-show

说明: 条件渲染 **用法:** v-show = "表达式或布尔值" 根据表达式或布尔值的真假切换元素的display属性

4.8.1 注意

v-show 不支持 <template>元素 也不支持 v-else

4.9 v-if vs v-show

都是用来判断渲染数据的

- v-if

1.有较高的切换性能消耗 2.惰性渲染 第一次如果条件为false 则不会渲染到页面 需要等后面条件切换后才会进行第一次渲染 3.条件切换是切换DOM数中这个元素的移除或添加 4.如果运行时条件很少改变则使用v-if

- v-show

1.有较高的初始渲染消耗 2.初始渲染 不管条件 第一次加载页面的时候都会渲染到页面 3.条件切换只是切换元素的display属性 4.如果运行时条件会非常频繁的切换则使用v-show



320



转发



微博



Qzone



微信

4.1.0 v-cloak

说明: 这个指令保存在这个元素上 直到关联实例结束编译

4.1.0.1 详细说明

插值表达式在网络较满的情况下可能会出现数据闪烁问题 可以通过给实例(#app)盒子添加一个 v-cloak 指令 通过这个指令的特性(如页面中还有插值表达式就会存在这个指令 如果页面的插值表达式都被替换成数据 就会自动移除这个标签) 配合 css [v-cloak] {display:none|opacity:0}来解决数据闪烁的问题

4.1.1 v-once

说明: 这个指令添加的元素 内部的胡子语法只会在第一次渲染的时候执行解析一次 后面数据发生改变后也不会触发更新

4.1.1.1 用途

某些元素只需要解析一次 后续不需要更新 就可以使用这个指令 提升性能

5、Vue实例

5.1 el

- 与页面中的元素绑定
- 指定根element(选择器)
- 可以写id、class、标签选择器
建议使用id 因为id是唯一的 一个Vue实例绑定一个页面元素
- **注意:** 不支持绑定body和html

5.2 data

- 数据对象
- 初始化数据(页面可以访问)
- 可以在里面写对象、字符串、数值、数组、...

5.3 methods

- 方法对象
- 可以在里面声明一些方法
可以通过this.xxx 获取Vue实例上的数据或方法
- **注意:** 不要使用箭头函数 会改变this指向

5.4 computed

- 计算属性
- 可以在里面声明一些函数
必须要有 return 值



320



转发



微博



Qzone



微信

- 计算属性函数中如果使用到了data中的数据 这些数据发生改变后 就会重新执行这个计算属性函数 将最新的计算结果返回出去

执行时机:初始化显示执行和函数中用到了data中的数据变化后会执行

- 在页面中直接用插值表达式使用计算属性({计算属性函数名}) 计算属性本质就是一个方法 但是使用的时候是将这些方法的方法名当作属性使用 计算属性的值就是return出来的值
- getter 和 setter

1.使用get和set函数 需要把计算属性函数改成计算属性对象 2.计算属性默认执行get方法 (根据相关的数据计算返回当前属性的值) 3.当计算数值自身改变后执行set方法 4.可以用来计算税前和税后的互推算

- 计算属性存在缓存 多次读取只执行一次getter函数
缓存 => {计算属性名:"数据结果"} 键值对

5.5 watch

- 侦听属性
- 可以侦听data中的属性和一些非DOM元素的改变
- 可以获取数据改变前的值和改变后的值
形参(newVal,oldVal) => (改变后的值,改变前的值)
- 深度侦听
watch默认无法侦听复杂数据类型 需要侦听复杂数据类型 得使用深度侦听

```
watch:{
  XXX:{
    deep:true,
    handler(newVal,oldVal){
      // 处理代码
    }
  }
}
```

- 侦听路由 hash

```
watch: {
  // watch里面的 $router 这些对象前面不要带this
  "$route.path"(newVal, oldVal) {
    if (newVal.indexOf("/login") >= 0) {
      this.welcom = "欢迎登陆";
    }
    if (newVal.indexOf("/register") >= 0) {
      this.welcom = "欢迎注册";
    }
  }
}
```

- 子组件侦听路由的变化

```
watch: {
  $route: function(newVal,oldVal) {
    console.log(this.$route.path);
  }
}
```



转发



微博



Qzone



微信

```

    }
  }
}

```

• 面试问及:

使用或侦听器吗 在Vue中碰到过什么bug

1. 侦听器用来检测数据的改变
2. 当侦听的那个数据发生改变后就会触发侦听器中的对应函数
3. 一般我更多的使用是用侦听路由的变化 => 重新获取数据(如搜索在对应的router-view中显示对应的数据)
4. 之前碰到过一个坑点 侦听器默认无法侦听复杂数据类型
5. 后面使用深度侦听 `depp:true` 来解决了这个问题
6. 或者侦听精确到对象中的那个值也可

5□.6□ computed和watch的区别

- watch里面的方法只能对那个方法名的属性名做侦听
- computed里面可以对那个方法中所有使用到了的data中的属性名做侦听
- watch里面无须return
- computed需要return

5□.7□ components

- 私有组件
- 后面会有全局组件 更详细

5□.8□ filters

- 过滤器
 - 声明全局过滤器 一定要在实例化Vue之前声明
- 全局过滤器

```

Vue.filter("formatData", (形参=管道符前面的数据, 形参=想要传入的数据...) =>
{
  处理数据; `返回` 处理之后的数据
});

```

- 局部过滤器

```

filters:{
  formatTime(形参=管道符前面的数据, 形参=想要传入的数据...){
    处理数据; `返回` 处理之后的数据  }
}

```

- 在页面中使用


```
{{data | formatTime | formatTime1 | ...}}
```
- 一个过滤器是一个函数 需要接收一个参数 参数就是处理的数据 (可以不传 默认是 管道符前面的数据) 可以做一些逻辑处理后 再将处理好的数据返回出去
- 不修改数据的情况下 修改数据的显示效果 过滤器不会修改源数据



320



转发



微博



Qzone



微信

6□、Vue实例中的this

- this就是当前实例化出来的Vue对象(Vue实例)
- Vue构造函数内部解析时 会把data、methods等中的值直接设置给这个实例化出来的Vue实例对象
- 最终直接通过这个Vue实例对象即可访问data中的数据以及methods中的方法等

7□、Vue实例属性

7□.1□ vm.\$data

获取data中的所有数据

7□.2□ vm.\$options

用于当前Vue实例的初始化选项 可以获取自定义选项

```
new Vue({
  customOption: 'foo',
  created: function () {
    console.log(this.$options.customOption) // => 'foo'
  }
})
```

7□.3□ vm.\$refs

- 通过给元素设定ref属性在Vue实例中获取这个元素
- 页面元素 ref="jack"
- Vue实例获取 this.\$refs.jack
返回的是一个DOM对象
- 如果ref重名后面的会把前面的覆盖

8□、Vue组件

8□.1□ 什么是组件

组件的出现就是为了拆分Vue实例的代码量 能够让我们以不同的组件来划分不同的功能模块 需要什么功能就去调用对应的模块即可 局部功能界面

8□.2□ 组件化和模块化的区别

- 模块化
是从代码的逻辑角度去划分的 方便代码分层开发 保证每个功能模块的职能单一
- 组件化
是从UI界面的角度进行划分的 前端的组件化方便UI组件的重用



8.3 全局组件

8.3.1 用法

`Vue.component('组件名',{参数})`

8.3.2 注意

组件名请使用小写 大写需要驼峰命名法

```
Vue.component('sayHiBaby',{参数})
页面中标签使用 <say-hi-baby></say-hi-baby>
```

8.3.3 组件参数

- `props:['xxx']`

父向子传参 通过 `props` 向子组件传递数据 可以在组件实例中通过 `this.xxx` 拿到传递过来的值 高级写法(props验证)

```
props:{
  xxx:{
    // 数据类型
    type:"String",
    // 必须传递
    required:"true",
    // 默认值
    default:"mystring"
    ....
  }
}
```

props可以传递任何数据类型 包括函数

- `data(){ return{ } }`
组件中的数据data是一个函数 将数据写在返回出去的对象中即可

为什么组件的data是一个函数 而Vue实例的data是一个对象?

1.每一个Vue组件都是一个Vue实例 2.都是通过`new Vue()` 创建出来的 3.如果data是一个对象的话 那么这些组件的引用指向就会相同 4.一个的值修改后 其他的值也会修改(这是JavaScript语法的特性) 5.如果data是一个函数 函数中再返回一个对象 6.那么每个Vue组件都有了各自的作用域 互不干扰

- `template`
模版
动态的HTML页面 包括一些JavaScript语法代码
注意: 如果模版中包含多个标签 需要在最外层加一个

- 其他参数和Vue实例的一致

9、Vue生命周期钩子



320



转发



微博



Qzone



微信

Vue生命周期钩子 || Vue生命周期函数 **Vue提供给开发者一系列的回调函数 让我们可以在Vue的各个阶段添加自定义的逻辑处理**

9□.1□ 三大阶段

9□.1□.1□ 初始化显示

创建期间的生命周期函数(1次)

- beforeCreate()
Vue实例被创建 但data和methods数据中的数据还没有被设置上去(未初始化)
- created()
data和methods以及被初始化(已被设置在Vue实例上)但是还未开始模版编译
常用 如果要操作data中的数据和methods中的方法 最早只能在created中调用
- beforeMount()
开始解析模版编译 把数据和结构(模版)关联起来 但现在页面上还不能看到数据
- mounted()
数据挂载完毕 页面中可以看到数据 当这个钩子函数执行完毕 创建期间的所有的生命钩子全部执行完
常用 发送ajax请求 启动定时器等异步任务 操作DOM中的节点

Vue渲染解析插值表达式 并不是在页面中直接渲染解析 而是将整个挂载在Vue实例的模版拿到内存中去解析 等全部解析完成后 再一次性的渲染到页面(批量) (因为有可能页面中有很多一样的data中的值 如果是在页面中直接渲染解析会很耗性能)

9□.1□.2□ 更新显示

运行期间的生命周期函数(0次-多次)

- beforeUpdate()
页面中的数据还是旧的 但Vue实例中的数据已经被更新 数据还未同步
- updated()
页面和Vue实例中的数据已同步

9□.1□.3□ 销毁

销毁期间的生命周期函数(1次)

- beforeDestory()
开始进入销毁阶段 Vue实例中的方法和数据还能使用 还未真正销毁
常用 做收尾工作 如:清除定时器...
- destoryed()
已被完全销毁 Vue实例中的数据和方法不能再使用

想要销毁Vue实例 调用 vm.\$destroy() 即可 注意: 这个方法并不会销毁Vue实例的一些如 定时器或计时器等方法 会造成 '内存泄漏' 所以在完全销毁之前 需要在 beforeDestory钩子中清除定时器等...

1□0□、父子组件传参

1□0□.1□ 父传子



320



转发



微博



Qzone



微信

- 子组件中定义 props 属性 props:['num','max','min']
- 父组件中使用子组件时 <comSon num='xxx' max='xx' :min='x'></comSon>

1.0.2 子传父

- 子组件中 特定的时候 触发 this.\$emit('事件名A','参数')
- 父组件中 <com @事件名A="自己methods中的方法"></com>

1.1、VueRouter

1.1.1 前端路由

url地址和组件之间的关系

1.1.2 如果是模块工程化(VueCLI)中使用VueRouter

必须添加 Vue.use(VueRouter)

```
import Vue from 'vue'
import VueRouter from 'vue-router'
Vue.use(VueRouter)
```

1.1.3 路由起步代码

1.1.3.1 HTML

- router-link
使用 router-link 来导航 最后会被渲染成a标签
可以通过 tag属性 来设置最后被渲染的标签
to属性 指定跳转的链接

```
<router-link to="/login" tag="span">登陆</router-link>
<router-link to="/logout" tag="span">注册</router-link>
```

- router-view
router-view 路由出口 路由匹配到的组件将渲染到这里

```
<router-view></router-view>
```

1.1.3.2 JavaScript

- 定义组件

```
const login = { template: "#tempLogin" };
const logout = { template: "#tempLogout" };
```



320



转发



微博



Qzone



微信

- 定义路由

```
const routes = [
  { path: "/login", component: login },
  { path: "/logout", component: logout }
];
```

- 创建路由

```
const router = new VueRouter({
  routes
});
```

- 绑定路由

在Vue实例中 绑定router

```
new Vue({
  el: "xxx",
  router
})
```

1.1.4 路由高亮

设置路由匹配成功后 router-link 的样式属性

- active-class 模糊匹配

to设置的路由和url上面的路由只要前面的/xxx 匹配成功就会添加样式

```
/login => /login/user
触发 添加样式
```

- exact-active-class 全部匹配

to设置的路由和url上的路由必须全部匹配才会添加样式

```
/login/user => /login/user
触发 添加样式
```

1.1.5 声明式导航

点了跳转 没有任何逻辑 类似于(a标签设置了href) <router-link to="地址">XXX</router-link>

1.1.6 编程式导航

跳转的同时执行其他需要执行的逻辑 router.push('地址')

1.1.7 动态路由匹配



320



转发



微博



Qzone



微信

- 将原本 router-link to 中的地址 /user => /user/:id
:id 只是一个占位
- 切换路由 /user => /user/123
- 可以在组件中通过 this.\$route.params(id) 获取传递过来的数据

1.1.8 路由重定向

- redirect : "跳转的新地址"
- {path:"地址1",redirect:"跳转到的新地址2"}
- 能够实现 匹配到地址1之后 立刻跳转到地址2

1.1.9 前置导航守卫

- 导航<路由> 当路由发生改变后 就会触发导航守卫再进行路由跳转

```
const router = new VueRouter({ ... })
router.beforeEach((to, from, next) => {
  // ...
  next()
})
```

to

去哪个路由 一般通过to来判断对应路由做对应处理

from

来自哪个路由

next()

必须next()才可以继续跳转页面(像node"express"里面的中间件)

- 执行时机 比组件的beforeCreate还要早

1.判断登陆状态 如判断token... 2.可以在跳转路由时先判断这个页面这个用户是否有权限访问... 3.可以每次路由跳转前都提示用户跳转至什么页面...

1.1.10 路由元信息

- 给路由配置一个标示 配合导航守卫定位到这个路由 做一些对这个路由的逻辑处理
- 给路由规则中配置一个meta对象
meta: { requiresAuth: true }
- 在导航守卫中就可以通过 to.meta.requiresAuth 是否为true 来判断是否是对应路由

1.1.11 嵌套路由

- 在需要使用嵌套路由的组件中 找到需要动态替换的位置
<router-view> </router-view>
- 修改路由规则
在需要嵌套路由的组件的路由规则中添加 children:[]
注意 嵌套路由的path不要加 /



```
{ path: '/user', component: User,
  children: [
    {
      path: 'index',
      component: Index
    }
  ]
}
```

1.1.2 统一404页面

配置错误路由规则

- 错误路由
`{path:"/NotFound",component:NotFound}`
- 配置404
`{path:"*",redirect:".NotFound"}`

注意：这个匹配必须写在路由规则的最下面 (当上面的路由匹配规则都不成功就会执行这个路由 然后跳转到404错误页面)

1.1.3 路由懒加载

- 当打包构建应用是 JavaScript 包会变得非常大 影响页面加载 路由懒加载 把不同路由对应的组件分割成不同的代码块 当路由被访问的时候才加载对应组件
- 将原先在router.js 中的 `import Foo from './Foo.vue'`; 修改成 `const Foo = () => import('./Foo.vue')` 其他不变 即可
- 路由懒加载并不能提速 只是把时间分开了 总时间没变

1.2、资源请求

1.2.1 Axios

1.2.1.1 Axios是什么?

Axios是一个基于 promise(实现了链式调用) 的 HTTP 库 可以用在浏览器和 Node.js 中 专注于发请求 专注于网络请求的一个库`

1.2.1.2 CDN

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

1.2.1.3 .then

成功回调

1.2.1.4 .catch

失败回调



1.2.1.5 get请求

```
// 为给定 ID 的 user 创建请求
axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

// 可选地，上面的请求可以这样做
axios.get('/user', {
  params: {
    ID: 12345
  }
})
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

1.2.1.6 post请求

```
axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

1.2.1.7 基地址

基础的地址应用 没有必要每次都写 可以直接抽离出来

```
axios.defaults.baseURL = '设置的基地址'
Vue.prototype.$axios = axios
```

axios填写路径时后面直接写对应的路径即可 前面的公共部分不需在写(写了也不会影响)

1.2.1.8 设置跨域携带cookie

```
axios.defaults.withCredentials = true
```

1.2.1.8.1 面试问及

- 坑点 axios 跨域默认不携带cookie
- 某个项目 登陆状态的判断 后端用的是cookie
- 登陆成功之后 服务器返回了cookie 标记登陆
- 但axios在请求不同源的接口时 默认不会携带cookie
- 所以后端接口后续一直无法获取登陆状态



- 抓包后检测网络请求 发现浏览器并没有把服务器返回的cookie给保留
- 所以导致每次都没设置成功cookie (set-cookie)
- 设置 `axios.defaults.withCredentials = true`
- 允许跨域携带cookie 这样浏览器就能在跨域的时候存下cookie
- 浏览器中无法直接查看跨域的cookie 浏览器隐藏了 可以通过抓包查看

1.2.1.9 创建克隆对象 多基地址设置

```
const xxx = axios.create({
  // 即地址
  baseURL: 'https://some-domain.com/api/',
  // 可以统一设置请求头
  headers: {Authorization: token}
});
xxx.get()
xxx.post()
```

- 只需要使用axios的create创建多个副本 每个副本设置一个不同的基地址
- 请求某个基地址的时候 使用该副本的对应方法即可

1.2.1.10 Axios拦截器

1.2.1.10.1 请求拦截器

```
axios.interceptors.request.use(function (config) {
  // 可以在发请求之前在这里设置一些请求头
  `config.headers.Authorization=token`
  return config;
}, function (error) {
  // Do something with request error
  return Promise.reject(error);
});
```

1.2.1.10.2 响应拦截器

```
axios.interceptors.response.use(function (response) {
  // 可以在获取响应数据之后设置一些提示 如获取失败/成功
  `response.data.code == 200?`
  return response;
}, function (error) {
  // Do something with response error
  return Promise.reject(error);
});
```

- 可以在响应拦截器中判断token是否造假
- 是造假则可以直接清除本地的token...

1.2.2 vue-resource

vue-resource已经不再更新 推荐使用Axios



320



转发



微博



Qzone



微信

1□3□、Vue动画钩子

1□3□.1□ Vue动画钩子是什么?

Vue提供的让程序员可以在动画的各个时机 添加 自定义逻辑 的钩子 也可称之为 动画钩子或动画函数

1□3□.2□ Vue动画的理解

- 操作 css 和 transition 或 animation
- Vue 会给目标元素添加/移除特定的 class
- 过渡的相关类名

```
// 指定显示的transition
xxx-enter-active
// 指定隐藏的transition
xxx-leave-active
// 指定隐藏时的样式
xxx-enter/xxx-leave-to
```

1□3□.3□ 单个元素动画

transition标签包裹

1□3□.4□ 列表过渡动画

transition-group标签包裹

1□3□.5□ name

动画样式的开始类名

1□3□.6□ tag

解析为的标签名

1□3□.7□ 过渡类名参数

- v-enter:
定义进入过渡的开始状态。在元素被插入之前生效，在元素被插入之后的下一帧移除。
- v-enter-active
定义进入过渡生效时的状态。在整个进入过渡的阶段中应用，在元素被插入之前生效，在过渡/动画完成之后移除。这个类可以被用来定义进入过渡的过程时间，延迟和曲线函数。
- v-enter-to
2.1.8版及以上 定义进入过渡的结束状态。在元素被插入之后下一帧生效 (与此同时 v-enter 被移除)，在过渡/动画完成之后移除。
- v-leave
定义离开过渡的开始状态。在离开过渡被触发时立刻生效，下一帧被移除。
- v-leave-active
定义离开过渡生效时的状态。在整个离开过渡的阶段中应用，在离开过渡被触发时立



320



转发



微博



Qzone



微信

刻生效，在过渡/动画完成之后移除。这个类可以被用来定义离开过渡的过程时间，延迟和曲线函数。

- `v-leave-to`
2.1.8版及以上 定义离开过渡的结束状态。在离开过渡被触发之后下一帧生效 (与此同时 `v-leave` 被删除)，在过渡/动画完成之后移除。

1.3.8 动画时机

- 条件渲染 (使用 `v-if`)
- 条件展示 (使用 `v-show`)
- 动态组件
- 组件根节点
- 动态的增删元素 就会触发进入动画 以及移除动画

1.3.9 动画钩子代码

1.3.9.1 HTML

```
<transition
  v-on:before-enter="beforeEnter"
  v-on:enter="enter"
  v-on:after-enter="afterEnter"
  v-on:enter-cancelled="enterCancelled"

  v-on:before-leave="beforeLeave"
  v-on:leave="leave"
  v-on:after-leave="afterLeave"
  v-on:leave-cancelled="leaveCancelled"
>
  <!-- ... -->
</transition>
```

1.3.9.2 JavaScript

```
// ...
methods: {
  // -----
  // 进入中
  // -----

  beforeEnter: function (el) {
    // ...
  },
  // 当与 CSS 结合使用时
  // 回调函数 done 是可选的
  enter: function (el, done) {
    // ...
    done()
  },
  afterEnter: function (el) {
    // ...
  },
  enterCancelled: function (el) {
    // ...
  },
}
```

320



转发



微博



Qzone

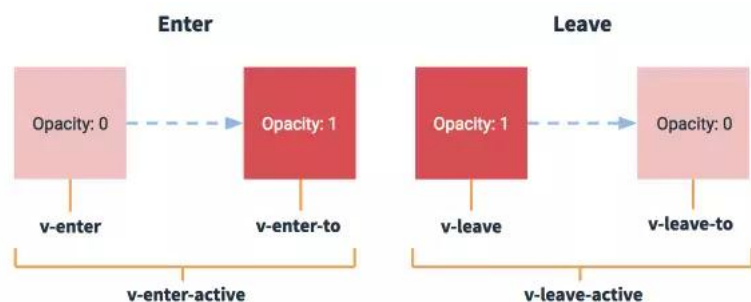


微信

```
// -----
// 离开时
// -----

beforeLeave: function (el) {
  // ...
},
// 当与 CSS 结合使用时
// 回调函数 done 是可选的
leave: function (el, done) {
  // ...
  done()
},
afterLeave: function (el) {
  // ...
},
// leaveCancelled 只用于 v-show 中
leaveCancelled: function (el) {
  // ...
}
}
```

1.3.1 过渡动画生命周期





320



转发



微博



Qzone



微信

1.4. VueX

1.4.1 Vuex是什么?

- 集中管理所有组件的数据
- 可以把它理解为一个仓库
- 将组件中公有的数据都抽到VueX中
- VueX中的数据 所有的组件都可以获取 所有的组件都可以修改

1.4.2 起步

1.4.2.1 下载

```
npm install vuex --save
```

1.4.2.2 创建VueX仓库

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

const store = new Vuex.Store({
  state: {
    // 数据
    count: 0
  },
  mutations: {
    // 方法
    increment (state) {
      state.count++
    }
  }
})

new Vue({
  el:xxx,
  // 挂载到Vue实例上
  store
})
```

1.4.2.3 在组件中获取VueX的数据

- HTML
{{ \$store.state.count }}
- JavaScript
this.\$store.state.count

1.4.2.4 在组件中修改VueX的数据



320



转发



微博



Qzone



微信

- VueX修改数据必须通过 mutations 中的方法修改数据
this.\$store.commit('mutations中的方法', 参数')

1□4□.3□ Vuex - state

- 数据
- 所有组件的都可以使用 获取数据

1□4□.4□ Vuex - mutation

- 方法 修改state中的数据

1□5□、VueCLI

1□5□.1□ 单文件组件

1□5□.1□.1□ 用一个文件能够包含组件的所有内容

- 结构

```
<template>
</template>
```

- 逻辑

```
<script>
export default {
}
</script>
```

- 样式

```
<style>
</style>
```

1□5□.1□.2□ 单文件开发的好处

- 更利于编码
- 利于后期维护
- 一个文件包含了所有内容

1□5□.2□ 什么是Vue-CLI



320



转发



微博



Qzone



微信

- 脚手架
- 可以把.vue文件翻译成浏览器可以识别的内容
- 自动刷新浏览器
- 自动压缩代码
- 自动的把高版本的JavaScript翻译成低版本的JavaScript
- 作为代理服务器
- ...

把很多开发中需要用到的功能整合到了一起 让Vue开发人员专注于逻辑代码即可 是用webpack配置出来的

1.5.3 搭建一个脚手架

vue create 项目名 <= 项目名不要有中文!!!不要大些 cd 项目名 npm run serve

1.5.4 Vue-CLI搭建项目的本质

- 创建文件夹
- 下载第三方模块
- 创建项目的基本结构
- 设置各个文件之间的关系
- 创建git仓库

1.5.5 Vue-cli项目结构

- main.js
主要的文件 所有和顶级Vue实例相关的都放到这里
- App.vue
最顶级的组件 仅次于Vue实例 看到的顶级页面结构一般都放在这里
- /components
组件的文件夹
- /assets
静态资源



1.5.6 搭建webpack-vue脚手架(详细版本)

 320

 转发

 微博

 Qzone

 微信

npm init webpack "项目名" cd "项目名" npm instal npm run dev

最后一步选 No, I will handle that myself 自己再npm i 下载速度会快一点

```

? Project name vue_demo
? Project description A Vue.js project
? Author lanhai <1444133084@qq.com>
? Vue build standalone
? Install vue-router? No
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests No
? Setup e2e tests with Nightwatch? No
? Should we run `npm install` for you after the project has been created? (recom
mended) (Use arrow keys)
> Yes, use NPM
Yes, use Yarn
No, I will handle that myself
  
```

单元测试包练习demo时不需要其他默认即可

1.5.7 webpack-vue项目结构

```

|-- build : webpack 相关的配置文件夹(基本不需要修改)
    |-- dev-server.js : 通过 express 启动后台服务器
|-- config: webpack 相关的配置文件夹(基本不需要修改)
    |-- index.js: 指定的后台服务的端口号和静态资源文件夹
|-- node_modules
|-- src : 源码文件夹
    |-- components: vue 组件及其相关资源文件夹
    |-- App.vue: 应用根主组件
    |-- main.js: 应用入口 js
|-- static: 静态资源文件夹
|-- .babelrc: babel 的配置文件
|-- .eslintignore: eslint 检查忽略的配置
|-- .eslintrc.js: eslint 检查的配置
|-- .gitignore: git 版本管制忽略的配置
|-- index.html: 主页面文件
|-- package.json: 应用包配置文件
|-- README.md: 应用描述说明的 readme 文件
  
```

1.5.8 Vue-CLI项目编码

1.5.8.1 编码位置

- 组件的逻辑直接写在xxx.vue中
- 静态资源放在assets文件夹下面 直接使用对应路径引用即可

1.5.8.2 引入css

- 全局引入
main.js => import "路径"
- 局部引入
组件内的script标签 => @import url("路径")

1.5.8.3 注册组件

- 全局组件
引入在main.js中
import 组件 from 地址



320



转发



微博



Qzone



微信

Vue.component('组件名',组件)

任意地方都可以使用 将组件名作为标签名 一次注册 全部使用

- 局部组件

在需要用到这个组件的地方 导入import 组件名 from '地址'

导入后 设置给components 就可以在导入的组件中通过组件名使用该组件

根据使用的位置 决定局部或者全局注册 只在某些地方用 用局部组件

- 组件中的name属性

直接在组件的内部写name:值即可

不能用中文

写了只会 Chrome的Vue插件中就可以看见这个名字 更加利于检索 利于编码

1.5.8.4 整合路由

1.5.8.4.1 下包

npm install vue-router

1.5.8.4.2 导包

import VueRouter from 'vue-router'

1.5.8.4.3 用包

- 创建路由规则

在components文件夹下创建一个组件xxx.vue

import 组件名 from '组件路径'

const routes = [{path:"/xxx/:xx",component:组件名}]

- 创建路由对象

```
`const router = new VueRouter({
  routes
})`
```

- 设置给Vue实例

```
`new Vue({
  router
})`
```

1.5.8.4.4 编码位置

- 导入 注册路由 main.js
- router-link router-view app.vue
- 添加组件components
- 静态资源assets

1.5.9 在Vue中使用其他插件

- 可以将插件写到Vue原型中 这样在其他组件中都可以使用这个插件



320



转发



微博



Qzone



微信

- (main.js)Vue.prototype.\$XXX = XXX
- (其他组件) this.\$XXX
- \$XXX 的\$的目的是 区分是Vue自带的属性还是后面自定义在原型中的属性

1.5.9.1 面试问及

- Vue原型用过没
- 为了共享 将插件设置在Vue原型上
- 所有组件都是Vue的实例
- 所有在其他组件中 可以通过 this.\$XXX 获取

1.5.1.0 css预处理与css作用域

1.5.1.0.1 css预处理

1.5.1.0.1.1 使用less

- 下载
 - npm install -D less-loader less
- 使用
 - <style lang='less'> </style> 也可以直接导入less文件

1.5.1.0.1.2 使用sass

- 下载
 - npm install -D sass-loader node-sass
- 使用
 - <style lang='scss'> </style> 也可以至今导入scss文件

1.5.1.0.2 css作用域

- 为style标签 添加 scoped 属性即可
- 生成样式, 和渲染组件时

为元素添加随机属性 样式中添加属性选择器 2者结合 就把css的作用范围 约束

- 为了不影响其他 可以添加这个属性
- 注意
 - 样式的属性是随机生成的 如果要修改样式
 - 直接修改在scoped中的样式 不要抽离成样式文件导入

1.5.1.1 ES6模块化语法

1.5.1.1.1 默认导出导入



320



转发



微博



Qzone



微信

- 导出
export default xxx 只能写一个 如果需要暴露多个 可以在default后面写对象
- 导入
import xxx from "模块"

1.5.1.2 按名字导出导入

- 导出
export const bbb 导出和导入的名字必须一样
- 导入
import {bbb} from "模块"

1.5.1.2 Vue全家桶

- Vue
- axios
- Vue-router
- 饿了么ui

饿了么前端团队开发的PC端的基于Vue的组件 内部封装了很多现成的组件 在VueCLI中使用 elementUI npm i element-ui import ElementUI from 'element-ui' import 'element-ui/lib/theme-chalk/index.css' Vue.use(ElementUI) 有些组件并没有在组件内部使用原生事件 但是有些情况需要一些原生事件 就可以使用.native修饰符来触发

- Vuex

1.6、扩展

1.6.1 link 和 @import 的区别

1.6.1.1 link

- link是html提供引入样式的标签
- link没有兼容性问题
- 可以通过JavaScript来控制link标签 修改样式文件路径
- 会和html一起加载

1.6.1.2 @import

- @import是提供的提供导入样式的方法
- @import有兼容问题 ie5以下不支持
- 不能通过JavaScript来操纵@import
- 等html全部加载完毕后再加载@import

1.6.2 JavaScript数组方法

 320

 转发

 微博

 Qzone

 微信

- arr.filter()

返回所有匹配成功的值 创建一个新数组, 其包含通过所提供函数实现的测试的所有元素 (数组过滤)

```
const oldArr = ["dajsk", "dkjdklas", "kgjftlk", "ksf", "ds", "mfksjjks"];
let res = oldArr.filter((val, index) => val.indexOf("d") !== -1); // 返回所有匹配成功的值
console.log(res); // [ 'dajsk', 'dkjdklas', 'ds' ]
```

- arr.find()

返回匹配的第一个值 返回数组中满足提供的测试函数的第一个元素的值 没有匹配成功返回 undefined

```
const oldArr = ["dajsk", "dkjdklas", "kgjftlk", "ksf", "ds", "mfksjjks"];
let res1 = oldArr.find((val, index) => val.indexOf("d") !== -1); // 返回匹配的的第一个值
console.log(res1); // dajsk
```

- arr.map()

将匹配成功的值做对应的计算后再次返回 创建一个新数组 其结果是该数组中的每个元素都调用一个提供的函数后返回的结果

```
const oldArrMap = [3, 4, 7, 1, 8, 5];
let res2 = oldArrMap.map((val, index) => {
  // 将匹配成功的值做对应的计算后再次返回
  if (val > 5) {
    val = val * 2;
  }
  return val;
});
console.log(res2); // [ 3, 4, 14, 1, 16, 5 ]
```

因为这些方法都是返回的新数组 并没有覆盖原来的数组所以可以继续链式调用数组的方法继续过滤

- arr.forEach()

遍历数组 方法对数组的每个元素执行一次提供的函数

```
const oldArrForEach = [3, 6, 8, 2, 8, 0];
let num = 0;
oldArrForEach.forEach((val, index) => {
  num += val;
});
console.log(num); // 27
oldArrForEach.forEach((val, index) => {
  if (index === 2) return console.log(val); // 8
});
```

1.6.3 解决跨域方案



320



转发



微博



Qzone



微信

- CORS
服务器设置允许访问 响应一个响应头
存在低版本不识别这个响应头
- JSONP
浏览器script的src支持跨域访问 发送一个callback去服务器
服务器接收callback 返回一个函数的调用携带数据
浏览器接收到返回值当作js执行 执行代码

注意 需要在浏览器声明callback去的函数 需要在script请求前声明 兼容性强 只能发送get请求

- flash
现已不用 苹果不支持flash
- sliverlight
- WebSocket
- postmessage
- iframe
- ...

1□6□.4□ 接口的请求方法和restful api

网络请求设计方法时 考虑到数据的操作主要：增删改查 方法的命名可以体现这个操作 一般常用的就是get和post

1. GET (SELECT) : 从服务器取出资源 (一项或多项) .
2. POST (CREATE) : 在服务器新建一个资源.
3. PUT (UPDATE) : 在服务器更新资源 (客户端提供改变后的完整资源) .
4. PATCH (UPDATE) : 在服务器更新资源 (客户端提供改变的属性) .
5. DELETE (DELETE) : 从服务器删除资源.
6. HEAD: 获取资源的元数据.
7. OPTIONS: 获取信息, 关于资源的哪些属性是客户端可以改变的.

1□6□.5□ Cookie、Session、token、localStorage、sessionStorage

1□6□.5□.1□ Cookie

- 用来解决http 是无状态 的
- 什么是无状态?

每次请求 浏览器和服务端交互完毕后 彼此并没有留下什么 继续请求 也无法判断你是谁 如登陆功能 为了能够保存一些信息 服务器返回响应报文时 会偷偷的带一个响应头 作用是在浏览器中偷偷保存一些信息set-cookie 浏览器接收到这个响应头后 会在本地保存这个响应头 第二次请求时 浏览器就会自动带上这个信息去服务器 服务器接收到这个信息 就知道你是谁了 ajax跨域请求 默认不携带cookie 需要设置 跨域cookie在浏览器中无法看到 需要抓包

1□6□.5□.2□ Session

Session 是将用户数据存储在服务端中 通过sessionId来验证查找服务器中的用户信息 sessionId一般是存放在浏览器的cookie中的

所以Session需要配合浏览器的cookie或者浏览器的其他存储技术一起使用



320



转发



微博



Qzone



微信

1.6.5.3 token

和cookie差不多 也可以记录登陆状态 服务器生成的 通过用户浏览器版本、用户信息...生成的一个密钥

浏览器不会自动保存 可以接口本地存储来保存token 浏览器不会自动携带发送 每次请求接口时可以通过headers携带存储的token headers{ Authorization : token }

1.6.5.4 localStorage

可以把数据存储在本地(浏览器) 只要用户不删除 则会一直保存 每个域名都是独立的保存数据 不同域名不能互相访问 长久保存数据可以存储到 localStorage 可以存储5M数据

- 保存数据 localStorage.setItem(key,value)
- 获取数据 localStorage.getItem(key) => 如果没有这个数据 则返回 null
- 删除一个数据 localStorage.removeItem(key)
- 清空所有数据 localStorage.clear()

1.6.5.5 sessionStorage

短暂存储数据 可以多页面传值 相当于localStorage会更安全 浏览器关闭后就不会保存了 可以存储5M数据

- 保存数据 sessionStorage.setItem(key,value)
- 获取数据 sessionStorage.getItem(key) => 如果没有这个数据 则返回 null
- 删除一个数据 sessionStorage.removeItem(key)
- 清空所有数据 sessionStorage.clear()

1.7、Xmind笔记

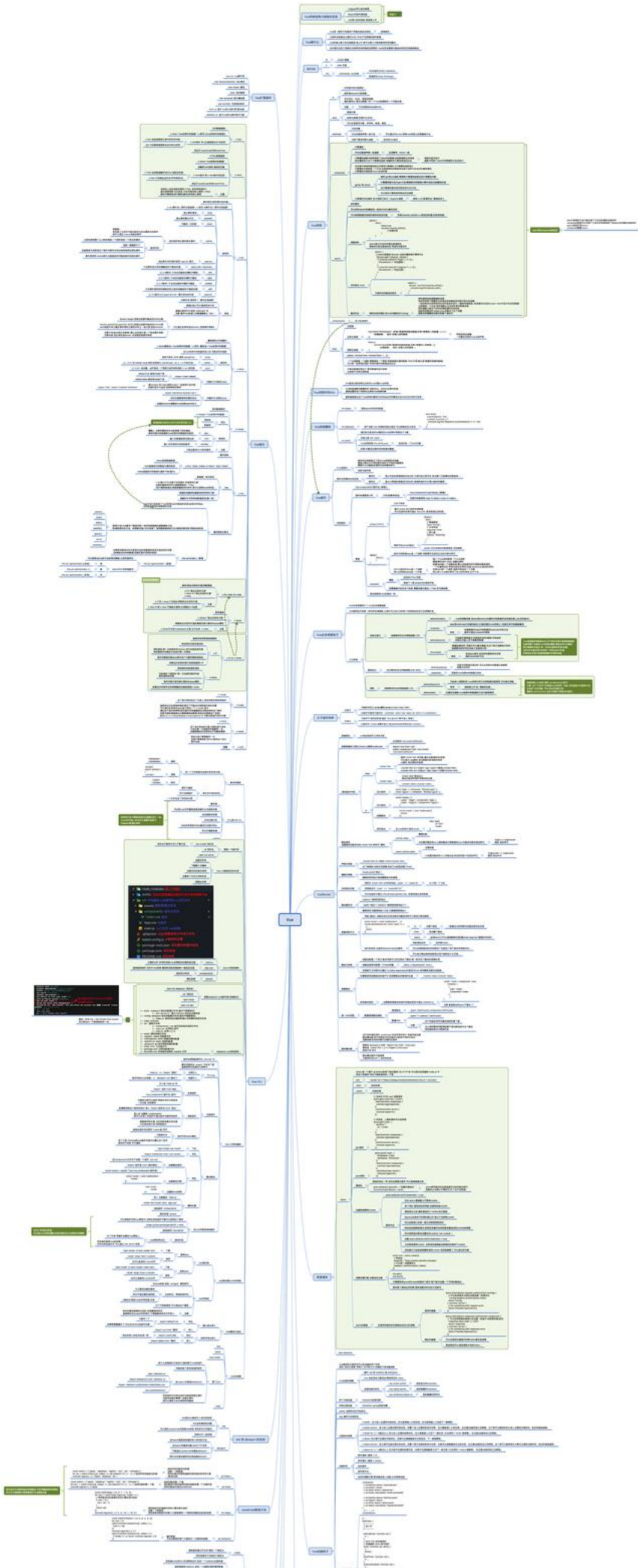
320

转发

微博

Qzone

微信



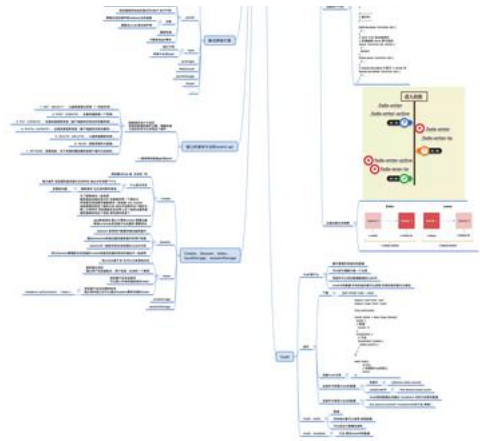
320

转发

微博

Qzone

微信



很多都是自己的话去解释和理解 可能会存在错误 或者有错别字 欢迎指出谢谢

若有感兴趣的小伙伴，需要VUE学习文档思维导图原图的，关注我，私信回复获取：VUE学习文档思维导图

作者：蓝海00
转载链接：<https://www.jianshu.com/p/125ce0c89603>

☆ 收藏 ⓘ 举报

320 条评论

写下您的评论...

评论

- Echa攻城狮 14天前

转发了

回复

0 0 ⓘ
- Morefree60976538 13天前

react可读性差?? 可读性差的代码不是看谁写?

回复 · 3条回复

0 0 ⓘ
- 科幻精彩瞬间 12天前

不错，我用angular

回复 · 1条回复

1 0 ⓘ
- 暴躁的Zz 14天前

vue大法好

回复 · 1条回复

1 0 ⓘ
- 黑色白兰地 12天前

看框架，vue或许有优势，但看生态，react领先太多了。

回复 · 1条回复

0 0 ⓘ

查看更多评论

相关推荐


2020年了,再不会webpack敲得代码就不香了(近万字实战)

Echa攻城狮 · 107评论 · [相关](#)

 320


 转发


 微博


 Qzone

 微信

	<div>5 个提升你 JavaScript 编码水平的实例</div> <div>前端达人 · 24评论 · 相关</div> <div>×</div>
	<div>ES6 完全使用手册附加案例实战讲解</div> <div>Echa攻城狮 · 49评论 · 相关</div> <div>×</div>
	<div>Vue 中 强制组件重新渲染的正确方法</div> <div>前端小智 · 18评论 · 相关</div> <div>×</div>
	<div>「前端进阶」高性能渲染十万条数据(虚拟列表)</div> <div>Echa攻城狮 · 47评论 · 相关</div> <div>×</div>
	<div>Vue合理配置WebSocket并实现群聊</div> <div>Echa攻城狮 · 78评论 · 相关</div> <div>×</div>
	<div>带你彻底掌握 Vue 3.0 的响应式系统</div> <div>源码时代 · 19评论 · 相关</div> <div>×</div>
	<div>详细讲解高频使用的 Git 命令</div> <div>Echa攻城狮 · 3评论 · 相关</div> <div>×</div>
	<div>构建大型 Vue.js 项目的10条建议</div> <div>Echa攻城狮 · 59评论 · 相关</div> <div>×</div>
	<div>推荐 8 个漂亮的 vue.js 进度条组件</div> <div>Echa攻城狮 · 11评论 · 相关</div> <div>×</div>
	<div>手把手教你前端的各种文件上传攻略和大文件断点续传</div> <div>Echa攻城狮 · 73评论 · 相关</div> <div>×</div>
	<div>JavaScript编码面试题：手动实现数组的map, filter, reduce方法</div> <div>灵魂诗人猪猪侠 · 7评论 · 相关</div> <div>×</div>
	<div>12 种使用 Vue 的最佳做法</div> <div>前端小智 · 18评论 · 相关</div> <div>×</div>

 320

 转发

 微博

 Qzone

 微信

	<div>佳田...实现HTML页面生成图片</div> <div>分布式定时任务调度框架实践</div> <div>Echa攻城狮 · 33评论 · 相关</div>	×
	<div>Typescript, 我永远不会回到JavaScript了</div> <div>闻数起舞 · 58评论 · 相关</div>	×
	<div>Vue源码中9个可借鉴的基础方法</div> <div>web秀 · 31评论 · 相关</div>	×
	<div>《SpringMVC 进阶版》</div> <div>JAVA高级程序员 · 16评论 · 相关</div>	×
	<div>精通Java必须掌握的10款开源工具，你知道你个？附java高清教程</div> <div>太白讲编程 · 0评论 · 相关</div>	×
	<div>Nginx 转发匹配规则，后端程序员不得不会！</div> <div>追逐仰望星空 · 108评论 · 相关</div>	×