## 环境篇

开发环境搭建

开发环境搭建指南：<http://www.cnblogs.com/qdrw/articles/6380091.html>

具体步骤：

1、安装nodejs:运行node-v6.11.3-x64.exe

2、安装npm或者cnpm:

1) npm:是集成在node中的,不用另外安装

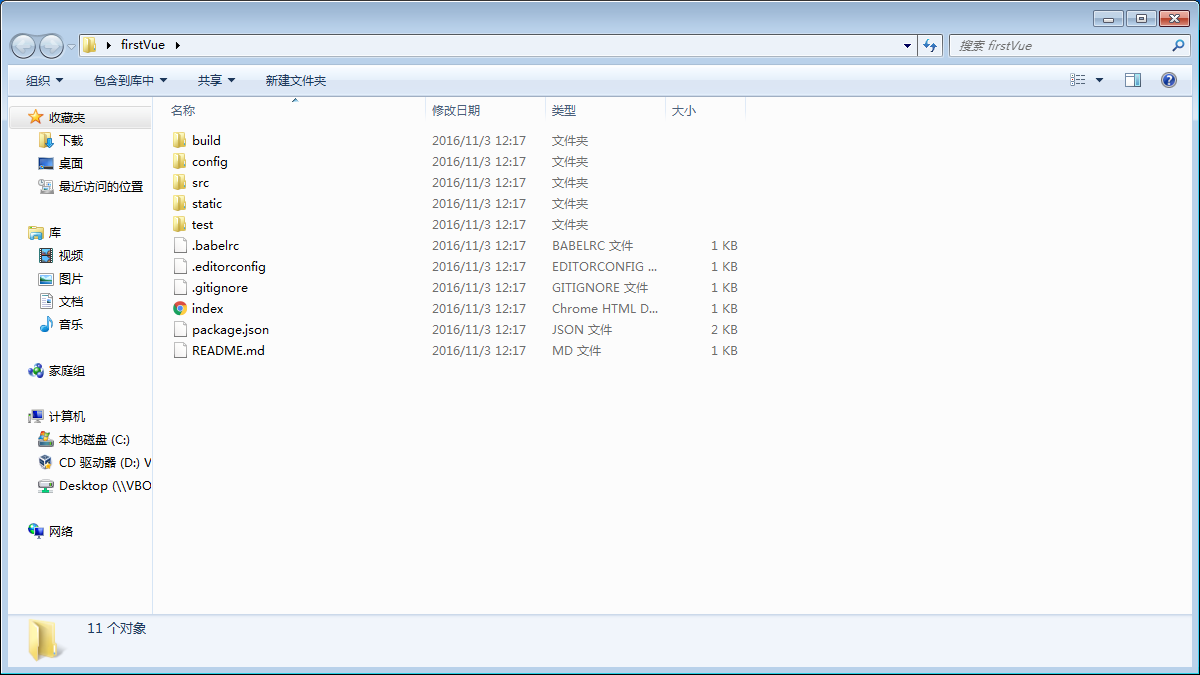
2)cnpm：在命令行中输入 npm install -g cnpm --registry=http://registry.npm.taobao.org 然后等待安装完成

3、安装vue-cli：npm install -g vue-cli

### 项目创建及启动

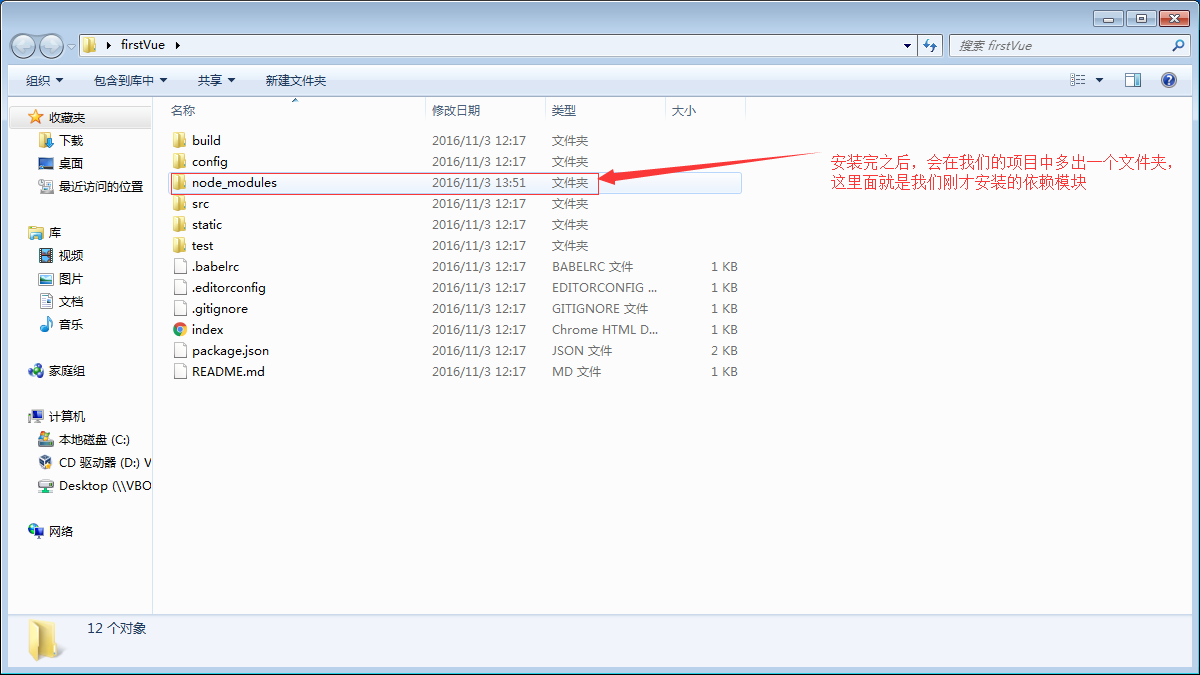
#### 1、 vue init webpack firstVue

这个命令的意思是初始化一个项目，其中webpack是构建工具，也就是整个项目是基于webpack的。其中firstVue是整个项目文件夹的名称，这个文件夹会自动生成在你指定的目录中



#### 2、npm install

安装依赖包，首先cd到项目文件夹（firstVue文件夹），然后运行命令 cnpm install ，等待安装。安装完成之后，会在我们的项目目录firstVue文件夹中多出一个node\_modules文件夹，这里边就是我们项目需要的依赖包资源



#### 3、npm run dev

在项目目录中，运行命令 npm run dev ，会用热加载的方式运行我们的应用，热加载可以让我们在修改完代码后不用手动刷新浏览器就能实时看到修改后的效果。

Webpack内置了小型服务器，比如 webpack 的 hot-dev

## 语法篇

### vue

Vue.js是一个用来构建web UI 的javascript库

#### 1、v-on指令后面可以跟哪些事件？

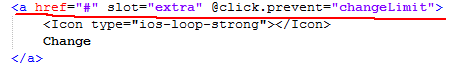
vue中v-on支持的事件很多，比如：click , mouseenter , mouseover , mouseleave等等，建议参考MDN事件类型一览表<https://developer.mozilla.org/zh-CN/docs/Web/Events>

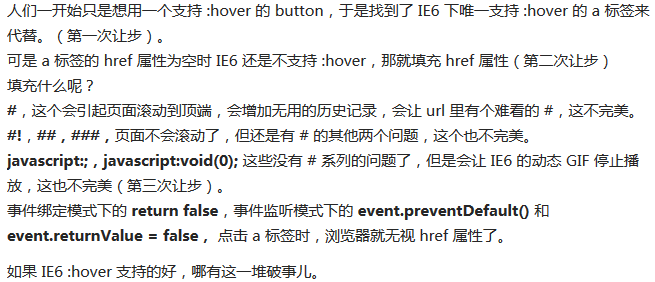
* **.prevent**

为什么需要禁用button的默认行为呢（为了防止在忘记写button的type属性时各个浏览器出现不同的反应，造成兼容性问题）？参考<https://www.cnblogs.com/wisdomoon/p/3330856.html>

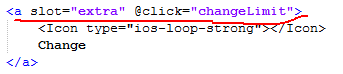
button按钮的type属性有三种：submit、button、reset.如果form内的button按钮有type=submit属性则，按钮会有默认的提交行为，reset属性会让按钮具备重置表单的行为，button属性会让按钮do nothing。如果编写页面时忘了写上button的type属性，那么点击按钮时不同的浏览器会产生不同的后果，所以可以手动禁用button的行为之后再开始加入自己的代码，这样即使忘了写上button的type属性，点击button在所有浏览器中都能有一致的效果。

该修饰符会阻止当前事件的默认行为。同理于调用 event.preventDefault() 方法

**在a标签中，设置属性href为#**会引起页面滚动到顶端，为了避免在低版本浏览器中出现点击滚动的问题，可以禁用a标签的默认事件

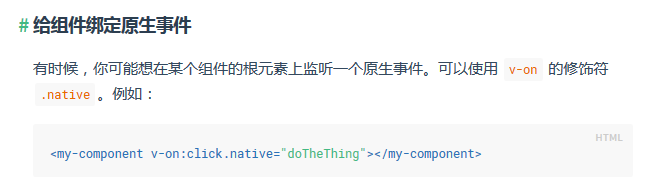


现在的浏览器基本都支持:hover，所以可以不用设置a标签的href，也就不用禁用默认事件了



参考练习\iView\Card

#### 2、vue键盘事件为何要加上.native?



Vue 的官方文档中指出了，Vue 使用的是一套自己的事件传递机制，如 @click 等事件是经过 Vue 封装的。因此，在一些实际上处理 DOM 原生事件的场合才需要添加额外的标识符。封装后的vue组件会阻止click事件，你可以试试只用click不用native,事件是不会触发的。而a标签不会阻止。

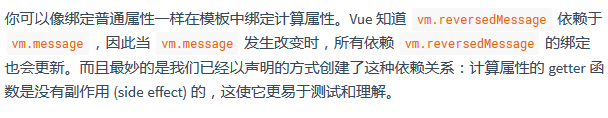
因为你@keyup.enter是写在一个封装好的组件上，如果你写在一个input上就不需要.native



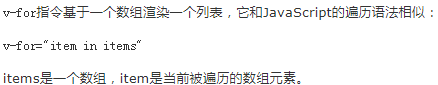
图中两种写法是等价的

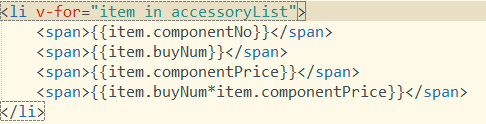
#### 3、计算属性

既指“computed”，又指“computed”中定义的属性名



#### 4、v-for属性是如何渲染的?





v-for指令写在哪个标签中，就重复渲染哪个组件，上图所示代码会渲染出n个<li>

参考华瑞配件商城用户管理营业执照部分

当它们处于同一节点，v-for 的优先级比 v-if 更高



#### 5、VUE是如何加载main.js文件的



#### 6、vue单文件组件

#### 7、slot

插槽(slot)是vue提出来的一个概念，正如名字一样，插槽用于决定将所携带的内容，插入到指定的某个位置，从而使模板分块，具有模块化的特质和更大的重用性。

1、插槽这个概念是相对子组件和父组件存在的，脱离了父组件和子组件谈插槽是没有意义的。因为插槽主要是为了把父组件扔给子组件的内容按照规定显示出来

2、插槽分为具名插槽和匿名插槽

3、子组件模板中必须包含至少一个 <slot> 插口，否则父组件扔给子组件的内容将会被丢弃

4、父组件中有slot属性，子组件中有slot标签





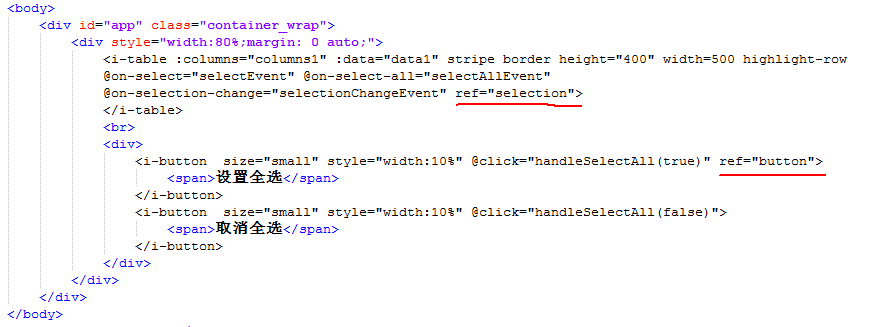
5、最初在 <slot> 标签中的任何内容都被视为**备用内容**。备用内容在子组件的作用域内编译，并且只有在宿主元素为空，且没有要插入的内容时才显示备用内容。

6、根据父组件扔过来的内容分类，会有三种情况：（1）没有slot属性的内容都会被显示到那个默认插槽里；（2）如果父组件扔过来的某一块内容加了slot属性slot=”aaa”，但是子组件模板中找不到一个name为aaa的slot标签，则这块内容会被丢弃；（3）有对应slot属性的内容会被显示到对应的插槽位置，替换掉备用内容

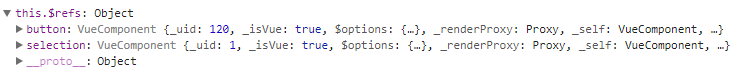
#### 8、ref

ref: 尽管有 prop 和事件，但是有时仍然需要在 JavaScript 中直接访问子组件。为此可以使用 ref 为子组件指定一个引用 ID，类似于jQuery中的id属性

$refs: 一个对象，持有该组件上已注册过 ref 的所有子组件（this.refs是一个对象，包含了父组件中拥有ref属性的所有子组件实例），属性名称是ref的值，对应的属性值是组件，例如有如下的HTML页面，设置了两个ref：selection和buttion



对应的$ref属性为：



#### 9、v-bind指令可以后跟什么

v-bind:argument="expression"

可以是数字



可以是在data()中定义的值



可以是布尔值



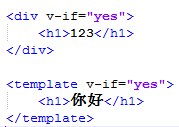
可以是函数名



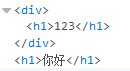


#### 10、v-if指令渲染

1、如果将v-if指令写在div中，条件成立时div也会被渲染出来；如果将v-if指令写在template中，条件成立时只会渲染包裹在template中的元素。对于如下代码：

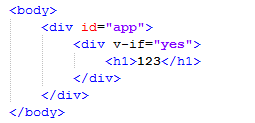


yes为true时渲染结果如下：



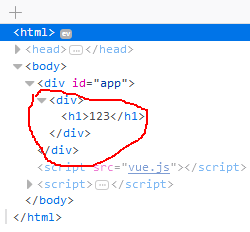
2、v-if根据表达式的真假来删除和插入元素：v-if="*expression*"

注意：v-if会根据表达式的真假来删除和插入元素，而不是隐藏和显示元素。对于如下代码，



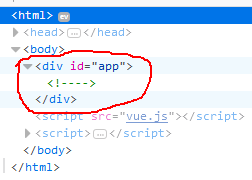
yes为true时渲染结果如下：

app中包含了元素

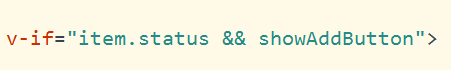


yes为false时渲染结果如下：

app中啥也没有



3、v-if="*expression*"，注意，是根据表达式来插入和删除元素的。



#### 11、el、template、render

优先级不同：render>template>el

模板将会 **替换** 挂载el，render会替换模板

#### 12、template中只能包含一个根元素

先看官网解释<https://vuejs.org/v2/guide/migration.html#Fragment-Instances-removed>



如果模板中有多个根元素，浏览器也会报错并给出提示



#### 13、npm常用命令

**npm install <name>**安装nodejs的依赖包

例如npm install express 就会默认安装express的最新版本，也可以通过在后面加版本号的方式安装指定版本，如npm install express@3.0.6

**npm install <name> -g**将包安装到全局环境中

但是代码中，直接通过require()的方式是没有办法调用全局安装的包的。全局的安装是供命令行使用的，就好像全局安装了vmarket后，就可以在命令行中直接运行vm命令

**npm install <name> --save**安装的同时，将信息写入package.json中

项目路径中如果有package.json文件时，直接使用npm install方法就可以根据dependencies配置安装所有的依赖包

这样代码提交到github时，就不用提交node\_modules这个文件夹了。

**npm init**  会引导你创建一个package.json文件，包括名称、版本、作者这些信息等

**npm remove <name>**移除

**npm update <name>**更新

**npm ls** 列出当前安装的了所有包

**npm root** 查看当前包的安装路径

**npm root -g**查看全局的包的安装路径

**npm help**  帮助，如果要单独查看install命令的帮助，可以使用的npm help install

node –v：查看node的版本

npm –v：查看npm的版本

vue –V：查看vue的版本（注意：V是大写的）

当你在安装包的过程中，有什么东西装不上，或者你感觉是由网络问题导致的，可以用npm config list列出可配置项

#### 14、工程中各个文件

1.index.html文件入口 ；

2.src放置组件和入口文件，是我们要开发的地方 ；

2.1

2.2

3.node\_modules为npm加载的项目依赖的模块 ；

4.config中配置了路径端口值等；（保存一些项目初始化的配置）

5.dist是存放打包之后的东西；

6.build中配置了webpack的基本配置、开发环境配置、生产环境配置等。（保存一些webpack的初始化配置）

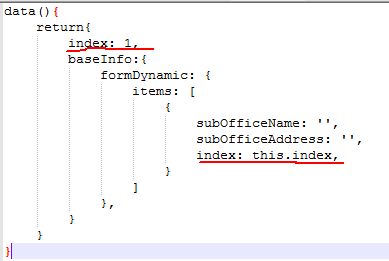
7.package.json中保存了跟npm相关的一些信息

<https://www.cnblogs.com/nangxi/p/7307080.html>

#### 15、data必须是纯粹的对象

**对象必须是纯粹的对象 (含有零个或多个的 key/value 对)**：浏览器 API 创建的原生对象，原型上的属性会被忽略。大概来说，data 应该只能是数据 - 不推荐观察拥有状态行为的对象。

比如下图中定义的data对象就不是纯粹的对象，因为**formDynamic. Items.index要去观察this.index的状态。**



#### 16、传参时如何去除空格

为什么要去空格？用户不小心输入个空格没看到，然后就直接提交肯定有问题啊

解决办法：通过v-model的修饰符.trim来去除空格

<input v-model.trim="blabla…">

#### 17、自定义组件

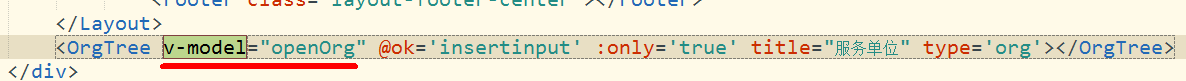
参考海航法务管理系统中律所录入页面使用的OrgTree组件

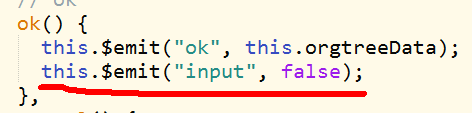
1、使用v-on绑定自定义事件

自定义事件就是除了vue事件系统中规定的事件之外，可以自己定义自己命名的事件，通常用来父子组件之间传递数据

2、



例如，在父组件中调用子组件时，可以直接在子组件OrgTree上使用v-model

然后在子组件里修改openOrg为false

v-model指令其实包含了两个方面：一是绑定了value属性，而是监听了input事件（参考vue的事件系统）

3、为什么子组件模板中的值没有在data中定义，却可以直接在模板中使用



4、自定义v-model

默认情况下，一个组件的 v-model 会使用 value prop 和 input 事件

5、vue的事件系统

#### 18、is属性

参考基础---列表渲染---一个组件的v-fo---todo list 的完整例子和组件---DOM模板解析注意事项

主要是因为像 <ul>、<ol>、<table>、<select> 这样的元素里允许包含的元素有限制，其他元素都会被看做是无效的内容，例如在 <ul>中只能出现<li>，出现别的任意元素都是无效的，可是原生的<li>有时候并不能满足我们的需求，这时候就可以自己封装组件作为<li>，只要给<li>加上is属性就可以了



其中todo-item就是我们自己定义的组件。这样做实现的效果与 <todo-item> 相同，但是可以避开一些潜在的浏览器解析错误

#### 19、在模板中可以绑定哪些属性

1、data函数中返回的普通对象属性

2、计算属性

3、自定义组件中props属性

#### 20、实例生命周期

1、实例初始化

2、beforeCreate：刚进入这个过程时，所有的属性都会先存到 vm.$options 中，在 beforeCreate 之后，完成了属性绑定，即将 $options 里的 data 啦，props啦，methods 啦等等一个个附到 vm 上，然后再触发 created 钩子。所以在 beforeCreate 的时候，通过 this.message 是拿不到值的，在 created 的时候就能通过 this.message 拿到值了,如果 data.message 中的初始值是简单的 string 类型，那么可以直接通过 this.$options.data()["message"] 取到值

3、created：在实例创建完成，属性已绑定之后调用。此时 “el” 属性为 undefined，接下来判断是否有 "el" 选项，如果没有，则需要手动调用 this.$mount(el)挂载到指定节点

4、beforeMount：在挂载到指定 DOM 节点之前调用

5、mounted：在挂载之后调用

6、beforeUpdate：在数据更新前调用

7、updated：在数据更新后调用

8、beforeDestroy：在实例销毁之前调用

9、destroyed：在实例销毁之后调用

#### 21、过渡和动画

transition和animation都是CSS的内容

一、过渡效果的实现方法：

(1)Vue 提供的 transition 组件结合CSS transition

(2) 过渡组件transition提供 [JavaScript 钩子函数](https://cn.vuejs.org/v2/guide/transitions.html#JavaScript-钩子)

二、动画效果的实现方法：

(1)Vue 提供的 transition 封装组件结合CSS animation



(2) 过渡组件transition提供 [JavaScript 钩子函数](https://cn.vuejs.org/v2/guide/transitions.html#JavaScript-钩子)

#### 22、render: h => h(App)

<https://segmentfault.com/q/1010000007130348>





#### 23、组件命名

1、小驼峰式[命名法](https://www.baidu.com/s?wd=%E5%91%BD%E5%90%8D%E6%B3%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uhmYrjPWmHbYnHfznWFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjDkP164rjDdPjckPHc4rjbz)（lower camel case）：  
第一个单字以[小写字母](https://www.baidu.com/s?wd=%E5%B0%8F%E5%86%99%E5%AD%97%E6%AF%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uhmYrjPWmHbYnHfznWFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjDkP164rjDdPjckPHc4rjbz)开始，第二个单字的首字母大写。例如：firstName、lastName。  
2、大驼峰式[命名法](https://www.baidu.com/s?wd=%E5%91%BD%E5%90%8D%E6%B3%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uhmYrjPWmHbYnHfznWFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjDkP164rjDdPjckPHc4rjbz)（upper camel case）：  
每一个单字的首字母都采用大写字母，例如：FirstName、LastName、CamelCase，也被称为 Pascal [命名法](https://www.baidu.com/s?wd=%E5%91%BD%E5%90%8D%E6%B3%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uhmYrjPWmHbYnHfznWFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjDkP164rjDdPjckPHc4rjbz)。

3、短横线分隔式命名法(kebab-case)

#### 24、

v-once: 只渲染一次，不会再改变

Object.freeze():阻止修改现有的属性，也意味着响应系统无法再追踪变化

keep-alive:是对路由进行缓存

### 二、vue-router

#### 1、安装和引入

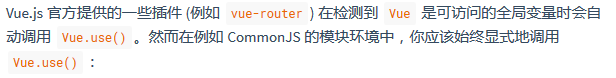
安装：npm install vue-router

引入：

import Router from 'vue-router'

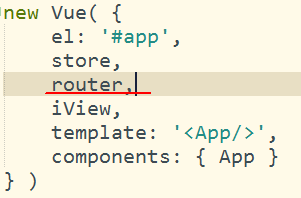
Vue.use(Router) // 要告诉 vue 使用 Router

完成后可以直接在 main.js 中引入并 Vue.use()



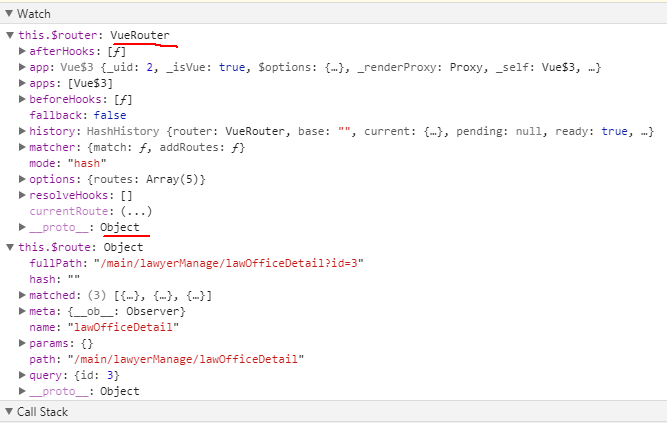
#### 2、将路由引入到Vue实例

如果main.js文件中vue实例中router属性是其他名称，都会报错：Uncaught TypeError: Cannot read property 'matched' of undefined（只是vue实例中router属性的名称必须为router，路由文件中定义路由时名称可以随便取，只要在main.js中导入时写对应的名称即可，例如：export const router123 = new Router({blablabal})）





#### 3、this.$route和this.$router



区别：

1. this.$router

vue实例的路由。this.router.push(location, onComplete?, onAbort?)

1. this.route

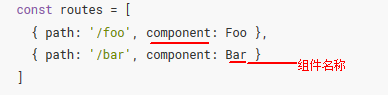
当前组件匹配到的路由，是一个对象

#### 4、多视图路由如何定义

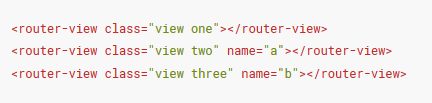
涉及到三个概念：路由、视图、组件

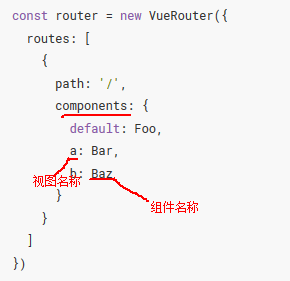
一个路由指向一个页面，但一个页面可以有一个或者多个视图，比如一个布局，有 sidebar（侧导航） 和 main（主内容） 两个视图，一个视图用代码表示就是一个<router-view>，一个视图使用一个组件渲染，因此多个视图的页面需要多个组件。

对于单个视图的页面，可以定义如下路由来指向该页面：



而对于多个视图的页面，就需要多个组件。确保正确使用 components 配置（带上 s）：





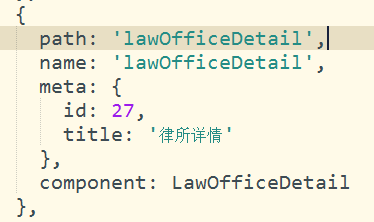
#### 5、如何复用组件

复用组件，也就是传不同的参数，不同的路由都渲染相同的一个组件，例如，我们有一个 User 组件，对于所有 ID 各不相同的用户，都要使用这个组件来渲染。

编程式导航都是通过router.push()方法来实现的。

方法一：通过query实现

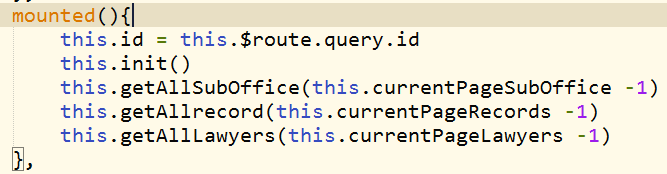
路由定义如下



律所查询页面导航方式如下：

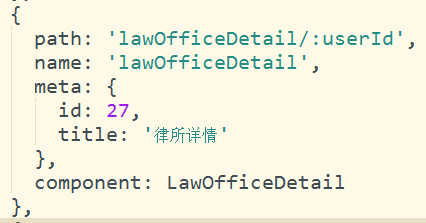


律所详情页面获取参数方式如下：

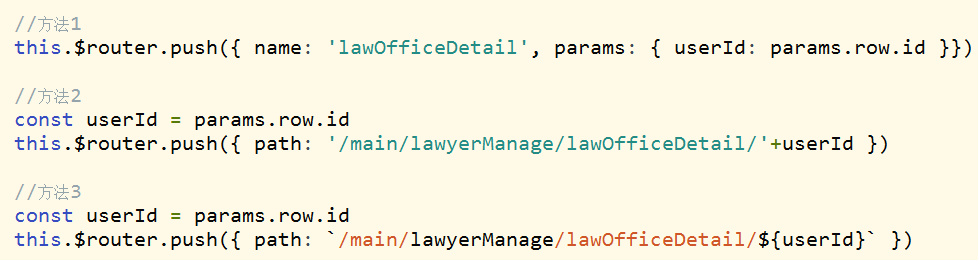


方法二：通过params实现

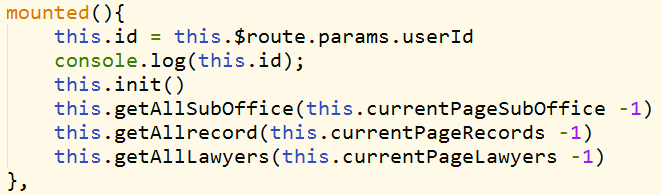
路由定义方式如下：



律所查询页面导航方式如下：



律所详情页面获取参数方式如下：

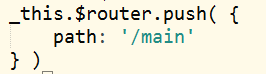


#### 6、重定向和别名

重定向：当用户访问 /a时，URL 将会被替换成 /b，然后匹配路由为 /b



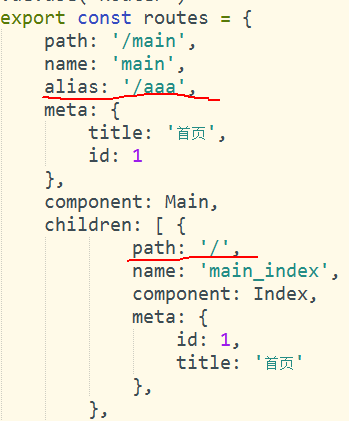
执行以下代码



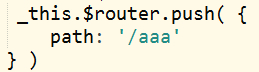
之后，浏览器路径自动跳转到



别名：/a 的别名是 /b，意味着，当用户访问 /b 时，URL 会保持为 /b，但是路由匹配则为 /a，就像用户访问 /a 一样。（\*\*\*\*\*\*用户访问的是别名）



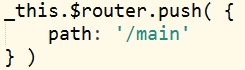
执行完以下代码

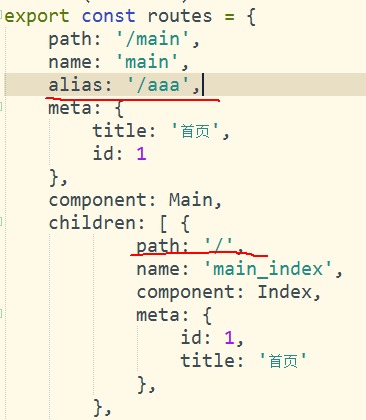


之后，浏览器路径自动跳转到



但显示的效果和用户访问“/main”是完全一样的





#### 7、导航守卫

一个路由导航的过程可以想象成从彭阳回西安的过程；每一个收费站都是一个守卫

导航：路由要去到某个组件（页面），导航的过程就是根据路由显示对应页面的过程

守卫：相当于沿途的检查站

to:去哪里

from：从哪里来

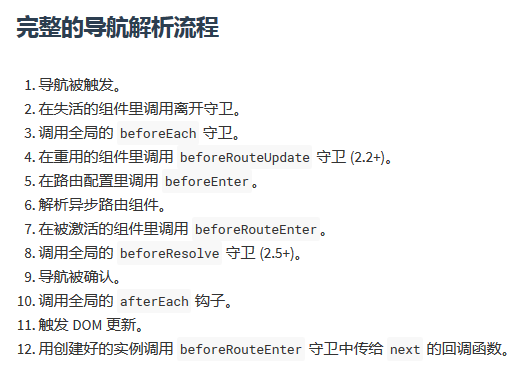
next（）：检查结果，

要么放行，next(),进入下一个检查站（守卫或者钩子）；

要么禁止通过；next(false);回到原处

要么让其改道，去别的页面next({name:’###’})

全局后置钩子afterEach，不需要next()参数，因为此时导航已经到达了目标组件，即已经被确认，所以不需要守卫告诉它下一步去哪里

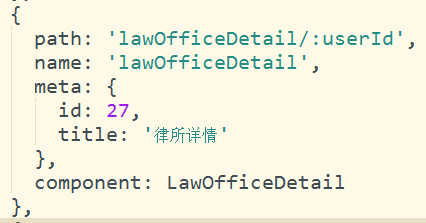


第九步执行完之后，新组建才会生成

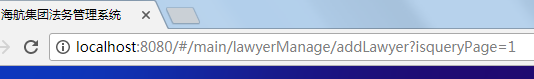
#### 8、params和query的区别

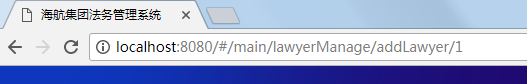
二者本质上都是用来给即将进入的路由传递参数，进而实现组件复用或者参数传递，不同点在于：

1、用params传参时需要修改路由定义文件，用query传递参数不需要修改路由定义



2、用query传递参数时浏览器地址栏的url会有参数拼接



params传递参数时浏览器地址栏的url是反斜杠后跟参数

### 三、vuex

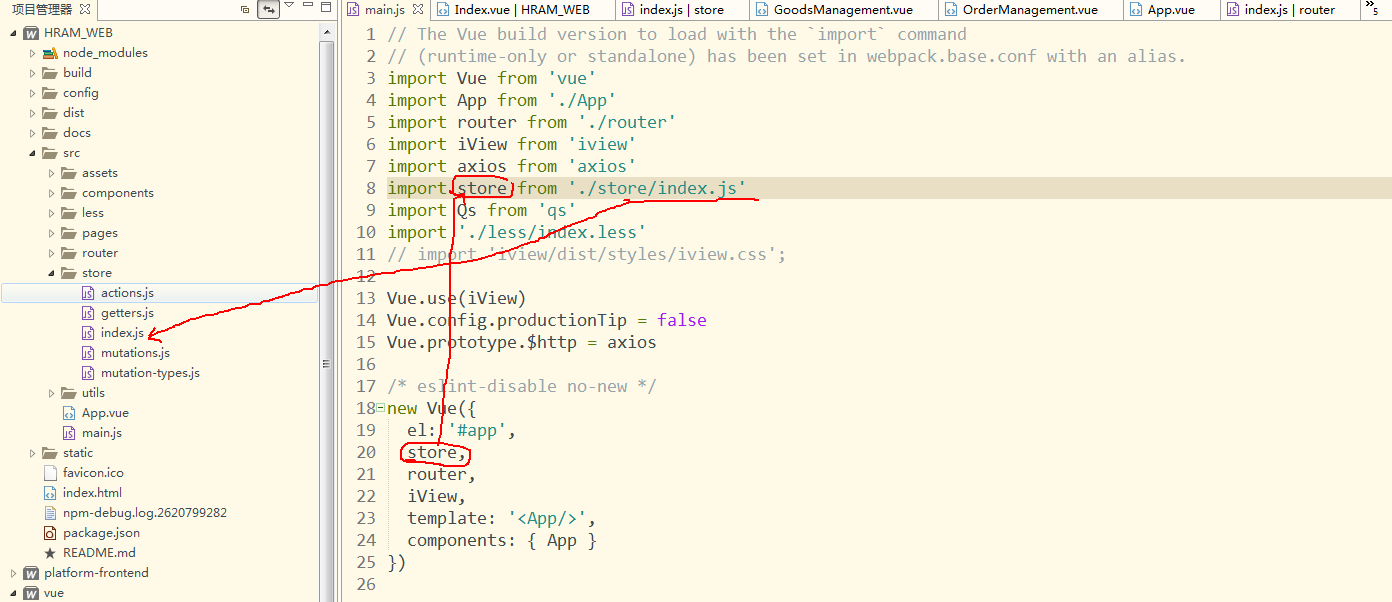
Vue的项目中，如果项目简单， 父子组件之间的数据传递可以使用 props 或者 $emit 等方式 进行传递

但是如果是大中型项目中，很多时候都需要在不相关的平行组件之间传递数据，并且很多数据需要多个组件循环使用。这时候再使用上面的方法会让项目代码变得冗长，并且不利于组件的复用，提高了耦合度。

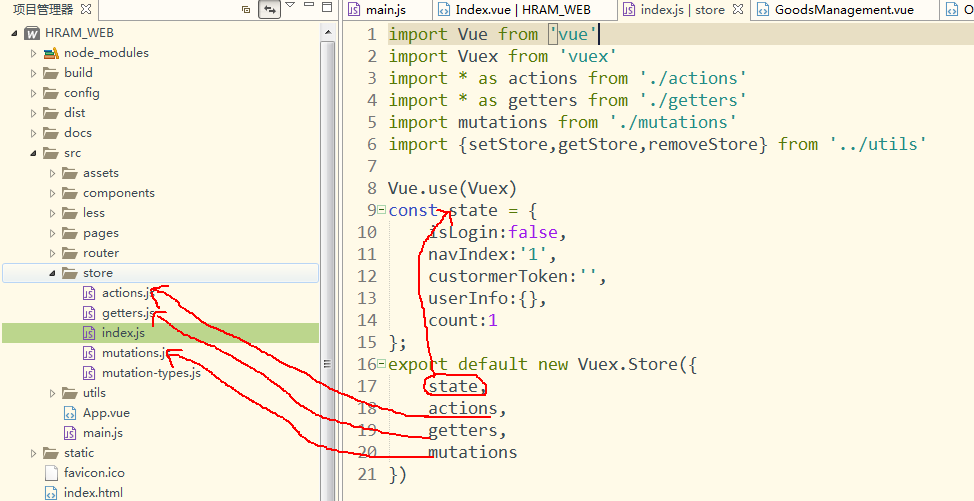
链接：<https://www.cnblogs.com/first-time/p/6815036.html>

#### 1、store的定义和使用

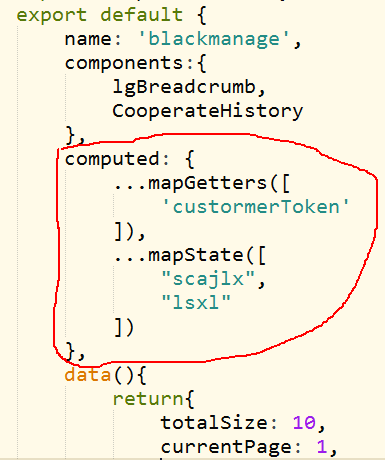
1.1、在根实例中关联store，其中store定义在store/index.js中



1.2、store的定义



1.3、读取store中的值

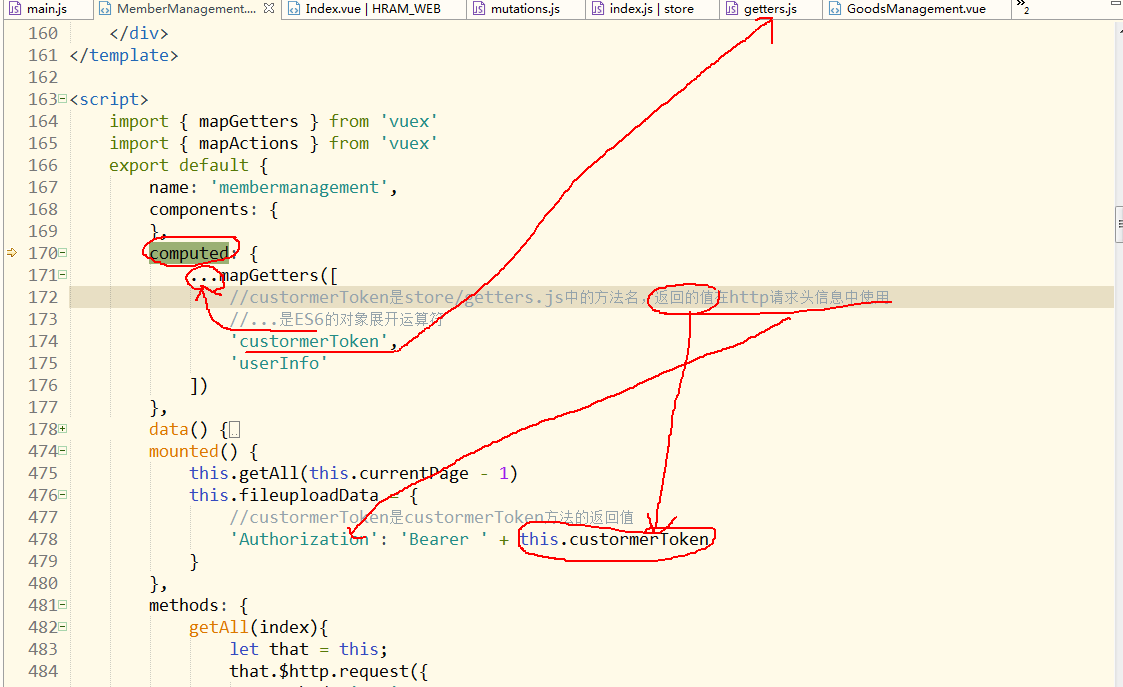


值的改变需要实时响应到各个组件，而computed正好可以监视值的改变并且做出响应，所以读取store中的值都是在computed中

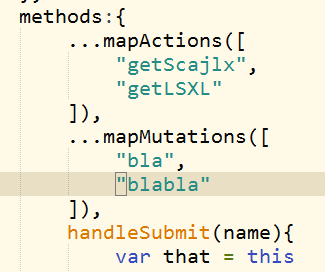
(1) **使用对象展开运算符，在computed中引入mapState 工具函数，将 store 中的 state 映射到局部计算属性中**

(2) 有时候，我们需要对state的数据进行筛选，过滤。这些操作都是在组件的计算属性进行的。如果多个组件需要用到筛选后的数据，那我们就必须到处重复写该计算属性函数；或者将其提取到一个公共的工具函数中，并将公共函数多处导入 - 两者都不太理想。如果把数据筛选完在传到计算属性里就不用那么麻烦了，getters就是干这个的

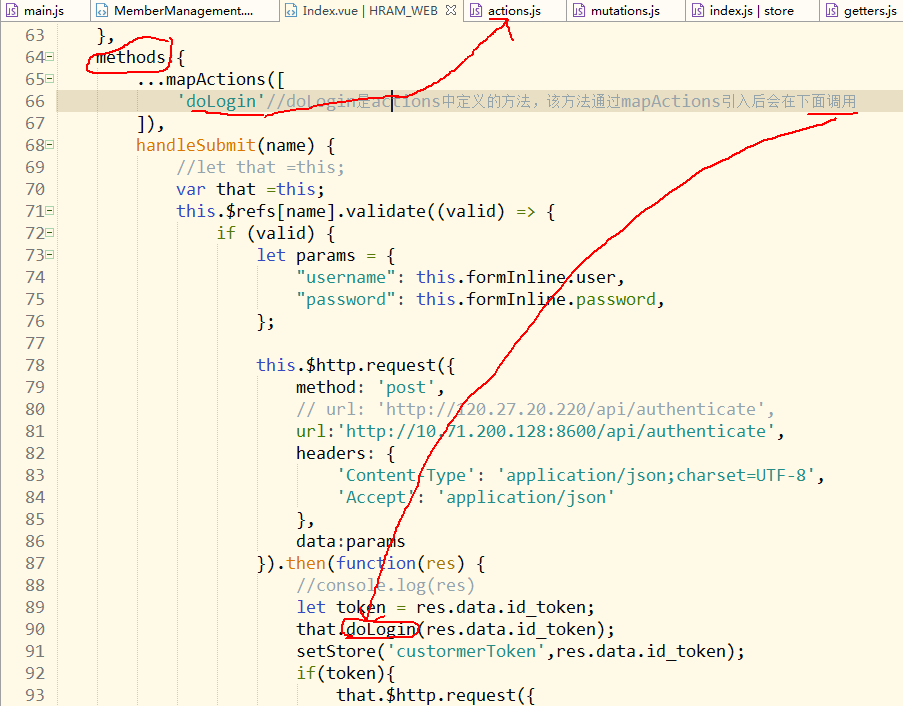
**使用对象展开运算符，在computed中引入getters，通过getters对象中的方法读取store中的值。**getters下的函数接收接收state作为第一个参数



1.4、在组件中引入修改值的方法

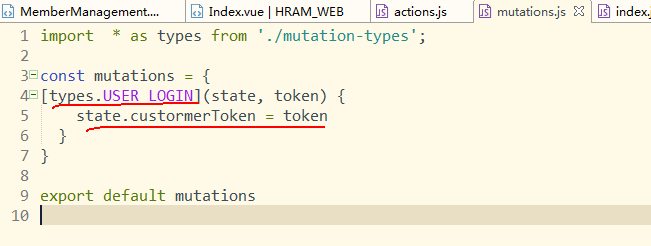


方法需要放在methods中，



1.5、actions会通过mutations，让mutations帮他提交数据的变更



1.6、mutations最终改变state中的custormerToken

#### 2. 为什么要使用对象扩展运算符



项目中在把store的getters混入到实例的computed属性中时都会用到扩展运算符，当计算属性名称和状态子树名称对应相同时，我们可以向 mapState 工具函数传入一个字符串数组（意思是：computed中的属性名称和store中定义的状态名称相同时）。

从源码分析得出<http://www.imooc.com/article/14741，mapGetters>函数返回的是一个对象，而实例的计算属性本身已经是一个对象{ }，所以此处需要用到扩展运算符遍历mapGetters的返回对象的属性，将其拷贝到当前对象之中



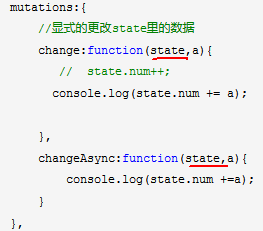
#### 3、必须通过mutation来改变state



#### 4、异步回调和同步回调的区别

#### 5、Mutation和action的区别

**Mutation：**

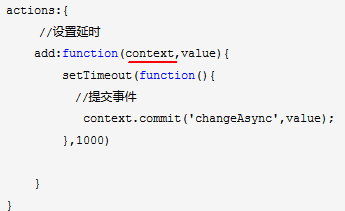
(1) 传入state；例如：

(2) 提交：可以在组件中使用 this.$store.commit('xxx') 提交 mutation，或者使用 mapMutations 辅助函数将组件中的 methods 映射为 store.commit 调用

(3) mapMutations

(4)有事件类型

**Action：**

1. 传入context；例如：，context是与 store 实例具有相同方法和属性的 context 对象

(2)分发：在组件中使用 this.$store.dispatch('xxx') 分发 action，或者使用 mapActions 辅助函数将组件的 methods 映射为 store.dispatch 调用

(3)mapActions

(4)没有事件类型

#### 6、mutation事件类型的用法

在mutation-types文件中定义类型，通过中括号来使用定义好的类型



#### 7、严格模式

状态变更且不是由 mutation 函数引起的，将会抛出错误

#### 8、华瑞项目中doLogin不是异步操作为何还要引入action？

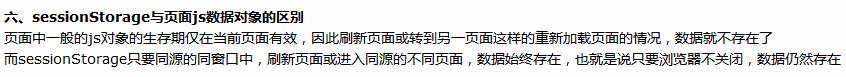


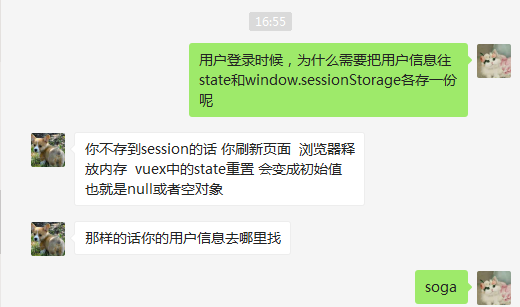
#### 9、刷新页面时vuex中的state会重置

问题：用户登录时候，为什么需要把用户信息往state和window.sessionStorage各存一份呢？为什么不存一份或者存到localStorage呢

答案：state里的数据在刷新之后会还原，万一用户刷新了一下页面岂不是要重新登录？localStorage 和 sessionStorage 都行 , 看你想保存多久；localStorage可以一直存储，sessionStorage当用户关闭浏览器时就会被清除

刷新页面不会清除sessionStorage和localStorage





### 四、vue-resource(废弃):

1、vue1.0中的$http服务 需要引入一个叫vue-resource.js的文件，因为vue.js中没有$http服务。

2、vue更新到2.0之后，作者就宣告不再对vue-resource更新，而是推荐使用axios

3、Vue 原本有一个官方推荐的 ajax 插件 vue-resource，但是自从 Vue 更新到 2.0 之后，官方就不再更新 vue-resource

### 五、axios

#### 1、介绍

基于 Promise 的 HTTP 请求客户端，可同时在浏览器和 node.js 中使用

功能特性：



目前主流的 Vue 项目，都选择 axios 来完成 ajax 请求(ajax技术的核心是xmlhttprequest)，而大型项目都会使用 Vuex 来管理数据

安装其他插件的时候，可以直接在 main.js 中引入并 Vue.use()，但是 axios并不是vue插件，所以不能 使用Vue.use()，只能在每个需要发送请求的组件中即时引入。为了解决这个问题，有两种开发思路：

1. 是在main.js 中引入 axios 之后，修改原型链

import axios from 'axios'

Vue.prototype.$http = axios

2、是结合 Vuex，封装一个 aciton。

完成三次握手，主机A与主机B才开始传送数据，所以有时候会看到有的请求发送出去之后看不到发送的参数，这时候就有再去看浏览器的提示，chrome浏览器中Provisional headers are shown提示，一般来说，如果看到这个提示，说明这个请求并没有发送出去，后台没有开服务。

完整的请求还应当包括 .then 和 .catch

.then(function(res){

console.log(res)

})

.catch(function(err){

console.log(err)

})

在发送请求后，使用then方法来处理响应结果，then方法有两个参数，第一个参数是响应成功时的回调函数，第二个参数是响应失败时的回调函数。

当请求成功时（包括响应失败和响应成功），会执行 .then，否则执行 .catch

#### 2、配置

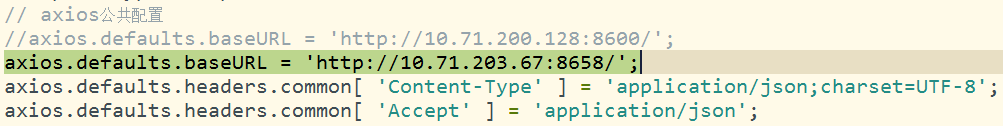
使用qs库对数据进行编码

<https://zhuanlan.zhihu.com/p/26284614>



* baseURL的配置

前期这个地址只要配成和自己联调接口的后台人员的IP地址就可以；后期整个项目开发完成后，会把所有功能部署到同一台服务器，该地址需要改成服务器的地址



配置了baseURL以后，代码中接口的url只需要按如下方式来写就可以了



而不需要再写成如下方式（省略了前面的服务器地址部分）



XML对象和http请求的区别

ajax请求和普通http请求的区别

用axios发送的是ajax请求还是普通http请求

为什么axios请求接口会发起两次请求

使用配置发送，使用别名发送

算不算跨域请求

### 六、iView

#### 1、定制主题

定制主题，就是根据项目要求，修改页面样式，比如修改body的背景色（并不是只能修改iView组件的样式），参考iView官网介绍



#### 2、Table

\_highlight: true只是高亮显示，

\_checked: true 可以默认选中当前项，必须配合columns 数据的 type: 'selection'，选中的数据并不会高亮显示

列宽width没有单位，否则会报错

通过给 columns 数据设置一项，指定 type: 'index'，可以自动显示一个从 1 开始的索引列。

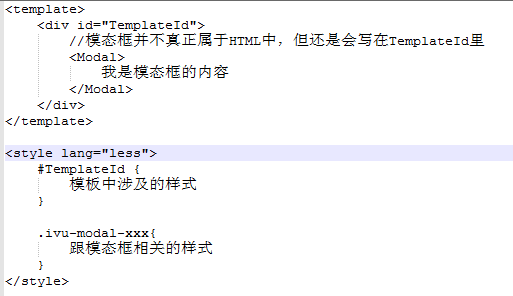
#### 3、Modal

通过使用 v-model 实现双向绑定来显示 / 隐藏对话框。

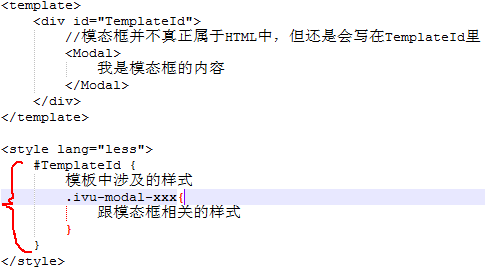
具名slot，就是有名字的slot。

* 给Modal添加样式时需要注意

正确写法：



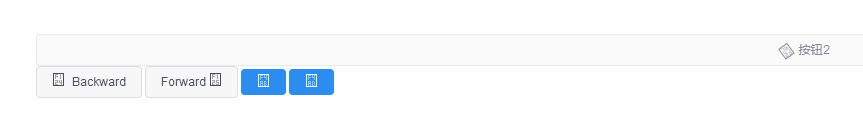
错误写法如下：



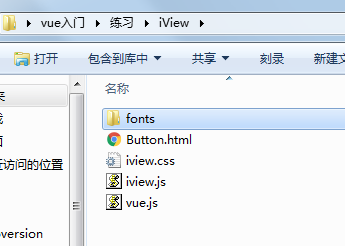
如果按这种错误写法在组件内修改Modal的样式，完全不会起作用，就会出现“我明明修改了，可就是不管用”这种事情，这时候模态框的样式依然是由源文件中的样式决定，例如，宽度始终是520px。

#### 4、浏览器不显示iView图标的问题

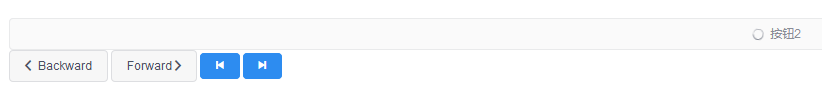
在html页面中使用了iView的图标，但是发现无法正常显示。



主要是因为fonts文件夹不在项目根目录下，只需要把fonts拷贝到项目中即可，如下：



加入fonts文件后显示正常，如下图：



#### 5、栅格系统

24栅格系统就是将栅格所在的区域进行24等分

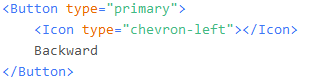
#### 6、Button

* Size属性既可以改变按钮的高度，又可以改变按钮的宽度，当用style属性设定了宽度后，size只能控制高度
* 给Button添加图标的方法：

1. 设置icon属性



1. 直接在Button内使用Icon组件



* 给Button添加文字的方法：

1)直接写



2)写在span 里面



#### 7、Input

Size属性只影响高度，宽度由style 属性设定

#### 8、Radio

* value属性

只在单独使用时有效。可以使用 v-model 双向绑定数据

* lable属性

只在组合使用时有效。指定当前选项的 value 值，组合会自动判断当前选择的项目

* 不需要数组

因为单选按钮组每次只能选择一个选项，无需使用数组

* 和开关不同

Radio单独用于切换到选中状态，如果想切换到不选中状态，只能再点击同组的其他Radio

#### 9、Checkbox

* value属性

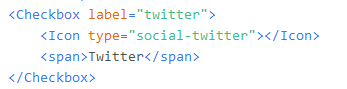
只在单独使用时有效。可以使用 v-model 双向绑定数据

* lable属性

只在组合使用时有效。指定当前选项的 value 值，组合会自动判断当前选择的项目

* 给Checkbox添加图标的方法：

只能在Checkbox内使用Icon组件

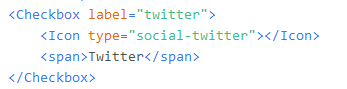


* 给Checkbox添加文字的方法：

1)直接写



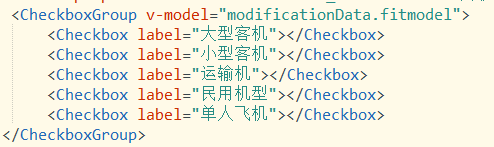
2)写在span 里面



* 数组

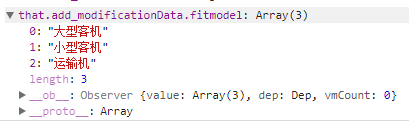


使用CheckboxGroup配合数组来生成组合。每一组CheckboxGroup都对应一个数组，用来存储被勾选的选项。数组名等于v-model绑定的名字，数组中的值是被选中的Checkbox的label。例如：对于如下“适用机型”这个CheckboxGroup





当勾选前三项之后，对应的数组如下：



如果后台需要以字符串的形式来发送“适用机型”这个参数，可以调用数组的join()方法

split()方法把字符串转换为数组

join()方法把数组转换为字符串

注：splice(index,howmany) 方法可删除从 index 处开始的零个或多个元素

slice(start,end) 方法可从已有的数组中返回选定的元素

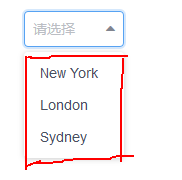
pop()删除并返回数组的最后一个元素

通过直接给CheckboxGroup的v-model绑定的名字赋值一个数组，来一次选中若干个选项：



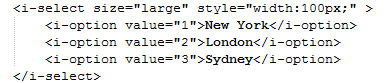
#### 10、Select

* 什么是Select组件的选项？



直观的看，选项就是图中红色圈里的内容；

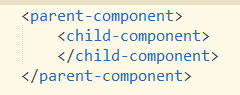
从代码上看，选项就是下图中的Option



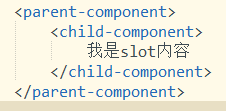
* 选项显示内容的逻辑是什么？

优先显示 *slot内容*，如果没有定义 slot，则显示label的值，如果没有设置 label，则显示value的值

组件设计初衷就是要配合使用的，最常见的就是形成父子组件的关系：父组件在它的模板中使用了子组件



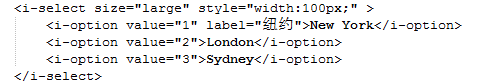
但更多的时候，父组件在使用子组件的时候会给子组件传入数据，也就是*slot内容*



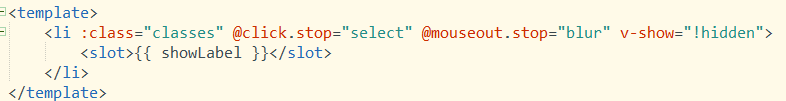
在子组件的模板中，必须要有

除非子组件模板包含至少一个 <slot> 插口，否则父组件的内容将会被**丢弃;**当子组件模板只有一个没有属性的插槽时，父组件传入的整个内容片段将插入到插槽所在的 DOM 位置，并替换掉插槽标签本身

**再回头来看Select选项显示内容的逻辑…**



去查看iView中的Select的源码，就会发现，option的模板中确实是定义了插槽的，且该插槽没有属性



所以对于父组件Select传入进来内容“New York”就会插入到插槽所在的DOM位置，因为Select选项会优先显示Slot内容，所以”New York”就会被显示出来。

* 验证Select时校验失败问题

今天在用iview自带的表单验证select标签的时候，一直验证不通过，代码如下：

<FormItem label="所属任务分组" prop="groupId">

<Select class="width-400px" v-model="dailyTaskItem.groupId" placeholder="请选择任务分组">

<Option v-for="item in taskGroupList" :value="item.id" :key="item.id">{{item.name}}</Option>

</Select>

</FormItem>

taskGroupList数据结构如下：

taskGroupList: [

{

groupId: 123,

name: **'分组001'**

},

{

groupId: 125,

name: **'分组002'**

},

{

groupId: 153,

name: **'分组003'**

}

],

后来查看了文档，iview内部用async-validator: <https://github.com/yiminghe/async-validator>

iview默认校验数据类型为String，而我的select用的value是number类型的,所以修改如下(校验添加type):

validateFirstStep: {

groupId: [

{ type: **'number'**, required: **true**, message: **'请选择任务分组'**, trigger: **'change'** }

],

},

#### 11、DatePicker

* 设置属性 confirm，选择日期后，选择器不会主动关闭，需用户确认后才可关闭
* Value属性：日期可以是 JavaScript 的 Date，例如 **new Date()**，也可以是标准的日期格式

注意：value 使用 v-model 时，值是 Date 类型，可以配合 @on-change 使用

#### 12、TimePicker

* 设置属性 confirm，选择器会有清空和确定按钮
* 时间，可以是 JavaScript 的 Date，例如 **new Date()**，也可以是标准的时间格式

注意：value 使用 v-model 时，值是 Date 类型，可以配合 @on-change 使用

#### 13、Cascader级联选择器

* value 为当前选择的数据的 value 值的数组，比如 ['beijing', 'gugong'] ，按照级联顺序依次排序，使用 v-model 进行双向绑定
* 设置属性 change-on-select 点任何一级都可以选择

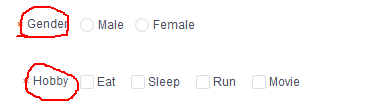


设置了该属性后，用户可以只选择江苏，或者只选择“江苏/苏州”，不用非得选出狮子林或者拙政园

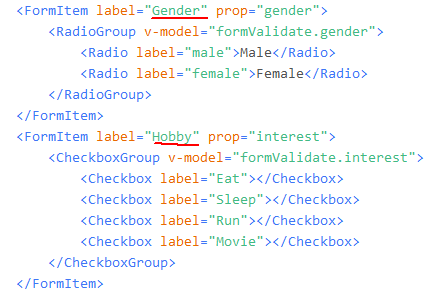
#### 14、Form表单

* 在 Form 内，每个表单域由 FormItem 组成

给 FormItem 设置属性 label 可以显示表单域的标签



对应的代码是：



* Form的label-width和 FormItem 的label-width

给FormItem设置了label后，必须要设置label-width。可以在每个FormItem上设置label-width，也可以在Form上设置一次label-width；所有的 FormItem 都会继承 Form 组件的 label-width 的值

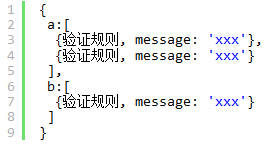
* 表单验证

1) 给 Form 设置属性 rules，rules是一个对象，同时给需要验证的 FormItem 设置属性 prop 指向对应字段即可; prop对应表单域 model 里的字段，如果只是验证而不考虑提交，model中可以不写某个字段

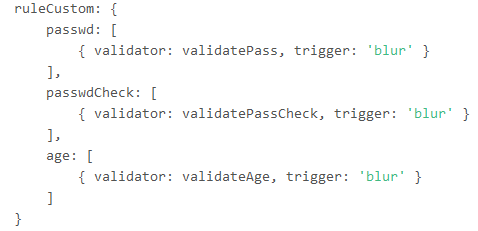
2)在data()的return中定义rules：对象，一个属性字段就是一个rule，rule是一个数组

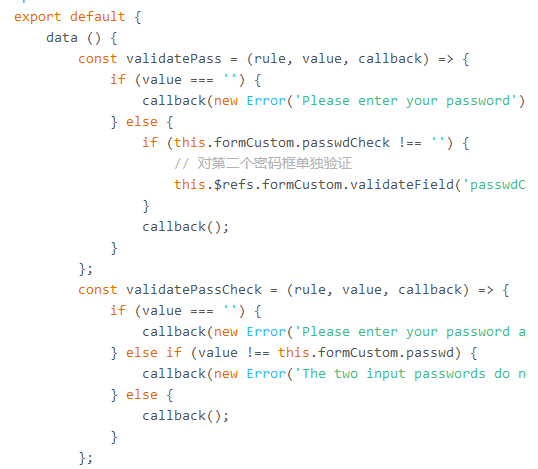
3)同时给需要验证的 FormItem 设置属性 prop 指向对应字段即可

4)一个表单域可以设置多条验证规则，格式为：



5) 可以完全自定义验证规则来完成更复杂的验证





* 整个表单验证validate

对整个表单进行校验，参数为检验完的回调，会返回一个 Boolean 表示成功与失败，支持 Promise

* model:表单数据对象

model属性的值和ref属性的值是完全不同的概念，即使有的代码里二者属性值内容完全一样，但只的是不同的东西。Model属性的值指的是组件data（）中定义的数据对象；Form没有value属性，也不可以使用 v-model 实现双向绑定

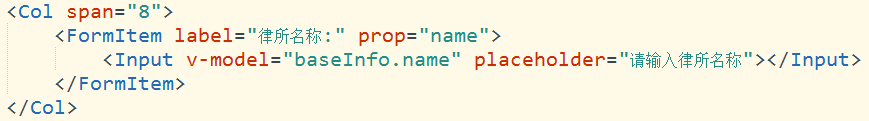
* 重置整个表单

首先要给Form设置ref属性和model属性，其次要给FormItem设置prop属性，然后给重置按钮绑定单击函数：this.$refs[name].resetFields();

* prop

动态设置 FormItem 的 prop 属性时，会依据上层的 Form 组件的 model 来获取。因为prop对应表单域 model 里的字段，所以设置prop属性时不能写model的值





如果写成prop=”baseInfo.name”就错了，因为baseInfo中没有baseInfo对象，只有name属性

#### 15、Message全局提示

* 轻量级的信息反馈组件，在顶部居中显示，并自动消失。有多种不同的提示状态可选择
* 使用方式：实例调用，不是以标签的方式来使用



#### 16、Notice通知提醒

* 在界面右上角显示可关闭的全局通知
* 使用方式：实例调用，不是以标签的方式来使用



#### 17、Modal模态框

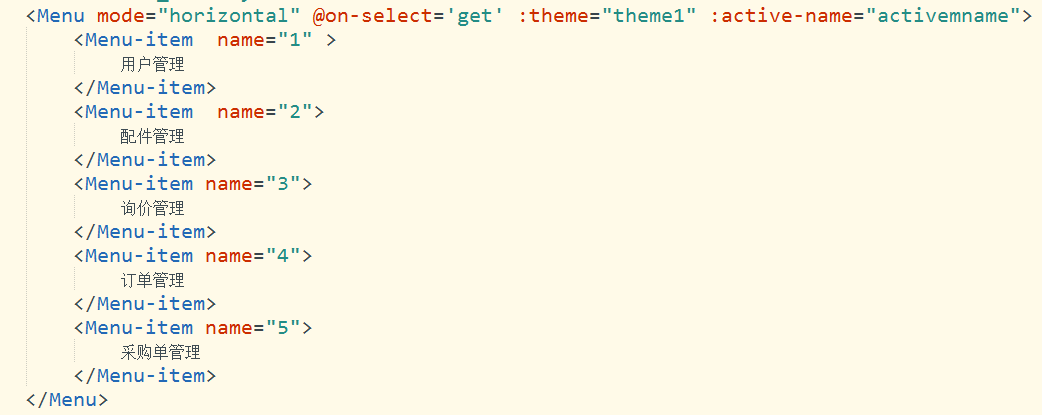
* 在浮层中显示的对话框
* 使用方式1：普通组件使用<Modal><Modal>
* 使用方式2：实例调用



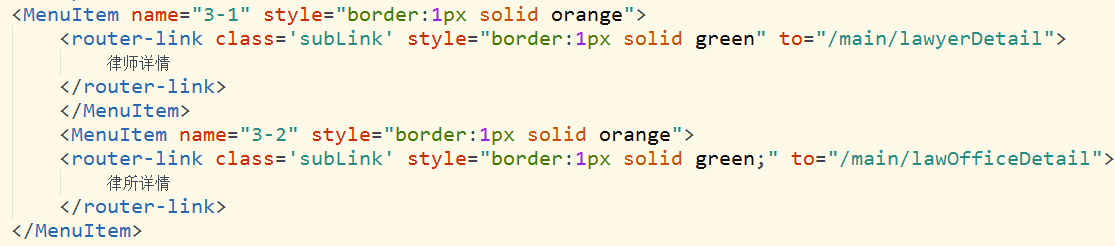
#### 18、Menu导航菜单

* 有两种方式可以实现菜单项的导航功能：

1. 通过Menu的on-select事件



2、通过在MenuItem中嵌入<router-link>



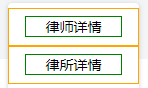
这种借助<router-link实现的导航功能有一个bug：浏览器渲染出来的a标签（router-link）和li(MenuItem)之间存在间距，用户只有在点击到绿色范围内时才能完成跳转，当点击到黄色和绿色之间的区域时没有响应，很容易给用户造成点击没有回应的错觉。这个bug不好解决，如果去掉MenuItem的padding，会让所有的菜单项紧紧挨在一起，视觉上不好看，保留padding又会造成点击的盲区，可以通过设置样式来解决这个问题，请参考《CSS设计指南》P188页的菜单一节

解决的思路是：

第一步：把a标签，即绿色元素设置为块级元素，因为块级元素会扩展填充父元素

第二步：把橙色原素的内边距取消掉。因为第一步设置完后，a标签虽然会扩展，但遇到橙色父元素的内边距就没法再扩展了，只有消除了橙色父元素的内边距，a标签才能继续扩展

第三步：给a标签设置内边距和字体大小。经过前两步，a标签已经完全扩展填满了橙色父元素，字体也变得很大，而且上下两个a标签的字体都快挨到一起了，这样还不够美观。所以要给绿色a标签设置内边距（注意，设置的是内边距，绝对不是外边距，因为设置了外边距的话，橙色和绿色之间又会 点击的盲区）和字体，让字体显示正常



#### 18、Page分页

current，当前页码，也就是会高亮显示的页码，默认值为1，如果将默认值改为0，会出现下图所示页面



并不是将current设置为多少，就会高亮显示多少，还要看总共能返回多少条数据以及每页显示多少条，例如总共100条返回，page-size设置为10，那么最多需要10页就可以显示完所有数据，此时如果把current设置为11，也不会显示出来

给用户显示时，第一页是1，但在与后台交互时程序员更倾向于用0来表示第一页，因此常常会看到这种写法。这种写法只是为了让代码看起来更加符合编码习惯，用不是必须的，也可以跟后台约定传1时查询第一页的参数

#### 11、总结

* 配合数组的组件：

1) Checkbox多选框：用于一组可选项多项选择的Checkbox组件。(Checkbox还可单独用于标记切换某种状态，这时候不需要数组)

2) Select选择器：设置了属性multiple的Select组件。多选模式下，Select接受数组类型的数据，所以返回的也是数组

3)Cascader级联选择器：value 为当前选择的数据的 value 值的数组，比如 ['beijing', 'gugong'] ，按照级联顺序依次排序，使用 v-model 进行双向绑定

* 有loading 属性的组件

1) Select选择器

2) Cascader级联选择器

3)Modal模态框

* 可以通过实例调用的组件

1)Message全局提示

2)Notice通知提醒

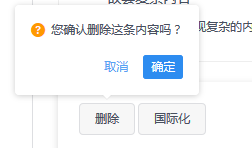
3)Modal模态框

* 可以弹出确认对话框的组件

1. Modal模态框



2)Poptip气泡提示



与 Modal 不同的是，它会出现在就近元素，相对轻量，且无遮罩层

#### 12、校验

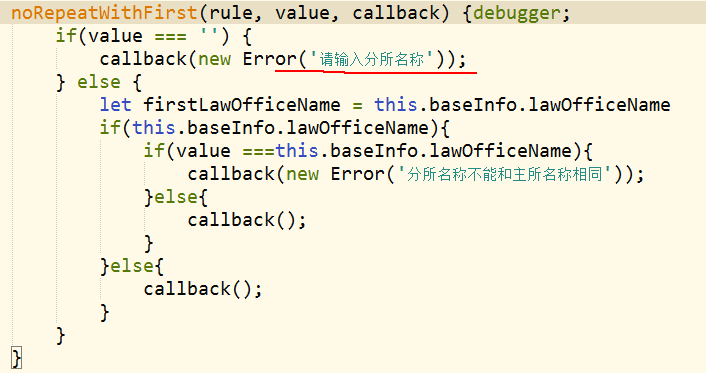
1、输入框的trigger是blur

选择框的trigger是change

2、动态添加的输入项如果有required校验，但没有设置message，则提示信息自动使用prop属性名：



3、validator中如果又重写了非空验证，则这时候新的提示信息会覆盖原来的message



### 七、Less

#### 1、定义变量用@

定义好的变量在使用时仍然要加@



#### 2、嵌套规则

保留HTML中的代码结构，嵌套默认后代选择器，如果需要子代选择器，则在子代前面加>

#### 3、编译Less文件

lessc style.less style.css

使用以上命令来编译style.less文件生成 style.css 文件

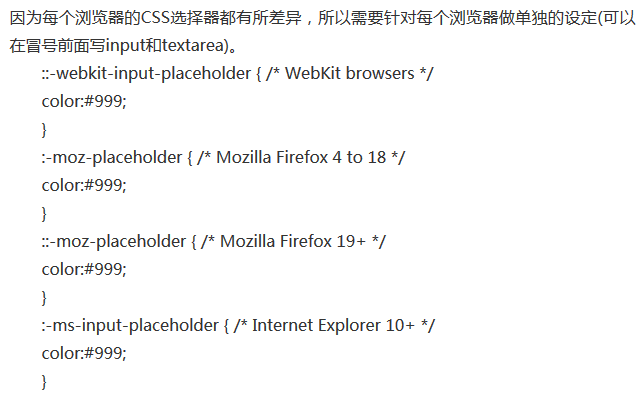
#### 4、display:inline-block;是什么意思？

表示行内块状元素。html的块级元素具有新元素另起一行的特性，使用inline-block可以使块级元素处在同一行内，在容器总宽度允许的情况下而不换行

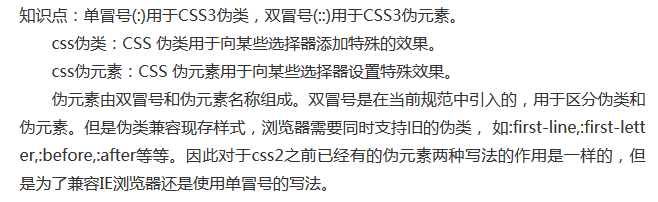
div是块级元素，把div块级元素设置成了[内联元素](https://www.baidu.com/s?wd=%E5%86%85%E8%81%94%E5%85%83%E7%B4%A0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PhmkPH63PjbsPWfsmhR30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rHfLnH61P1TsnWnLrHR1rf)，[内联元素](https://www.baidu.com/s?wd=%E5%86%85%E8%81%94%E5%85%83%E7%B4%A0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9PhmkPH63PjbsPWfsmhR30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rHfLnH61P1TsnWnLrHR1rf)设置高度是没效果的，所以你的红色背景当然看不到

#### 5、设置placeholder字体颜色

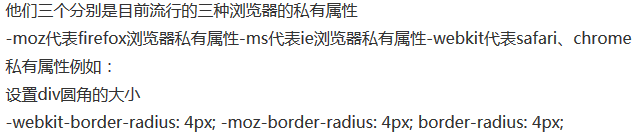
因为目前很多浏览器对CSS3还不是很支持，而且每个浏览器的CSS选择器都有所差异，所以需要针对每个浏览器做单独的设定



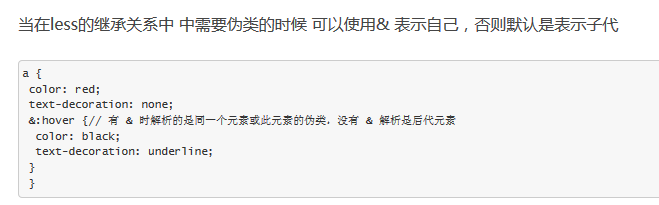
#### 冒号



#### 7、css3中-moz、-ms、-webkit分别代表的意思是什么呢？



#### 8、less中&代表的什么意思？



#### 9、CSS语言和CSS框架

静态CSS语言：css

动态CSS语言：LESS和SASS

框架：包括样式框架和布局框架：<http://www.iteye.com/news/21373>

### 7、CSS

#### 1、import

@import url(“global.css”);

@import url(global.css);

@import “global.css”;

以上3种方式都有效。当使用url (url)的方式时，包住路径的引号可有可无；当直接使用url时，包住路径的引号必须保留

#### 2、为文档添加样式的三种方法

1)行内样式

style=”width:100%;”

2)嵌入样式

<head>

<style>

.container\_wrap {

width:100%;

margin: 50px 0 0 0;

}

</style>

</head>

3)外部样式：链接样式和导入样式

<link rel="stylesheet" type="text/css" href="iview.css">

@import ”mystyle.css”

#### 3、在vue中如何引入样式文件

通过webpack搭建的项目，如何引入样式文件

1）在index.html模板html文件中引入，这种方式引入的原样编译在生成的html文件中，如果想要通过link引入外部的样式文件，建议使用这种方式：

[?](http://www.jb51.net/article/121550.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <!DOCTYPE html>  <html>   <head>    <meta charset="utf-8">    <title>2imis</title>    <link rel="stylesheet" href="./static/reset.css" rel="external nofollow" >   </head>   <body>    <div id="app"></div>    <!-- built files will be auto injected -->   </body>  </html> |

2）在入口js文件main.js中引入，一些公共的样式文件，可以在这里引入。这里会对scss文件进行解析，将对应的css代码插入生成html文件的style标签中，成为内联样式。

[?](http://www.jb51.net/article/121550.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import Vue from 'vue'  import App from './App'  import router from './router'  import ElementUi from 'element-ui'  import '@/common/scss/theme-blue.scss'  Vue.config.productionTip = false  Vue.use(ElementUi);  /\* eslint-disable no-new \*/  new Vue({    router,    render: h => h(App)  }).$mount('#app'); |

3）在对应的模板.vue文件中引入

<style lang="less">

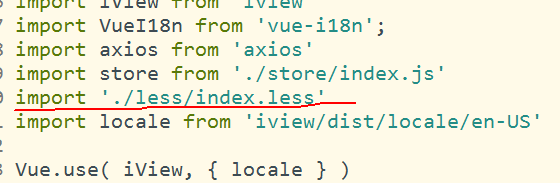
@import url("../../less/common.less");

</style>

总结：

1、在index.html文件中引入

2、在main.js文件中import



3、在App.vue文件中import

优先级：App.vue样式最高，main.js次之，index.html最低

#### 4、居中

方式一、DOM在浏览器中的居中通过设置样式margin:auto;来实现

方式二、字体在容器中的居中通过text-align: center; 来实现

text-align属性只有4个值，left,right，center，和justify，控制着文本在水平方向的对其方式。其中，center值也可以用来在较大的元素中居中较小的固定宽度的元素或者图片

#### 5、:first-child 和 last-child

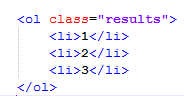
e: first-child代表一组e元素中的第一个，有如下规则：

ol.results li:first-child{

color:blue;

}

应用给以下标记：



#### 6、float

要浮动哪个元素就给哪个元素添加float类

#### 7、水平居中和垂直居中

<https://www.cnblogs.com/cainiao-Shun666/p/6622728.html>

text-align属性是设置给父元素的，可以实现父元素中的文本、图片的对其

文本的垂直居中：把line-height和元素的高度设置成一样的值，就能使文本垂直居中

#### 8、布局

传统布局： [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float)属性。

Flex 布局：display: flex;

注意，设为 Flex 布局以后，子元素的float、clear和vertical-align属性将失效，这时候如果想实现子元素的浮动，就不能用float属性，可以通过justify-content实现

#### 9、雪碧图

用画图板可以测量图片的尺寸

### 九、javascript

#### 1、JavaScript历史

**背景**：一开始仅仅作为输入验证器出现

1995年之前，为了完成简单的表单验证，必须把表单数据发送到服务器端，在服务器端用Perl语言做校验。当时的拨号上网速度很慢，完成一次简单的表单验证可能需要几十秒，与此同时，网页的复杂性越来越高，试想一下，你填了一堆数据，经过几十秒的等待，数据发送到了服务器，服务器一检查，发现你少填了一个输入项，然后又经过几十秒的等待，服务器把校验结果发给了你。这个过程是不是很考验你的耐心。如果不能改变输入验证的方式，客户端与服务器端交互的时间会非常漫长。

人们迫切需要一种技术，帮助人们在客户端就完成输入验证操作，验证通过后把正确的数据发送到服务器。当时走在技术革新最前沿的Netscape公司，决定开发一种客户端语言，用来处理这种简单的验证，并在自家的浏览器Netscape Navigator2中使用。

起初这种脚本语言被命名为LiveScript，该语言将同时在浏览器和服务器中使用（它在服务器上的名字叫做LIveWire）。为了赶在规定的发布日期前完成LiveScript的开发，Netscape公司决定和Sun公司合伙搞。在Netscape Navigator2正式发布前夕，Netscape为了搭上媒体热炒Java的顺风车，临时把LiveScript改名为JavaScript。

JavaScript 1.0大获成功，各个浏览器厂商都在自己的浏览器中加入了JavaScript实现，微软也不例外。随着JavaScript版本的增加，JavaScript的标准化问题就被提上了议事日程。

**ECMA-262：** ECMA-262 自然是ECMA第262号标准,这个标准就是ECMAScript（ECMA制定了很多标准，其中第262号是跟JavaScript相关的）

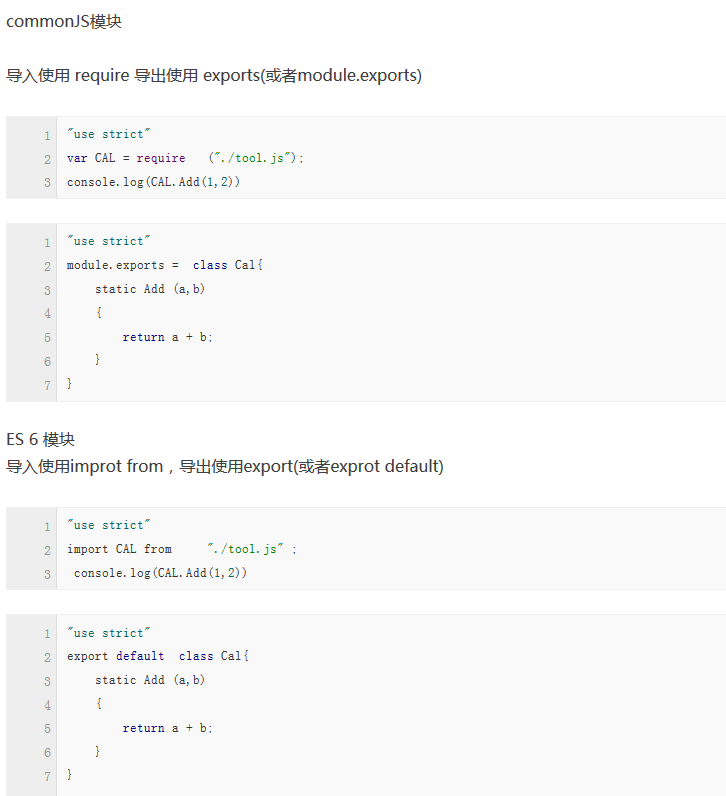
**ECMAScript：**由欧洲计算机制造商向协会制定的JavaScript脚本语言标准

欧洲计算机制造商向协会负责JavaScript的标准化工作。由该委员会制定的JavaScript标准统称为ECMAScript。ECMA-262只是ECMAScript

**ECMAScript 6：**由于ECMAScript6是在2015年发布的，所以也叫ECMAScript2015，简称ES2015

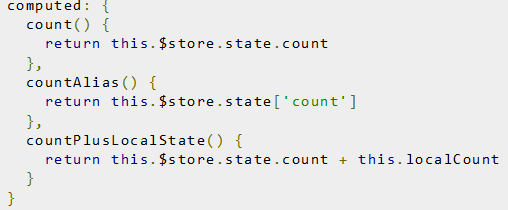
**现状**：已经发展称为一门强大的编程语言

#### 2、ES6模块与commonJS 模块的转换

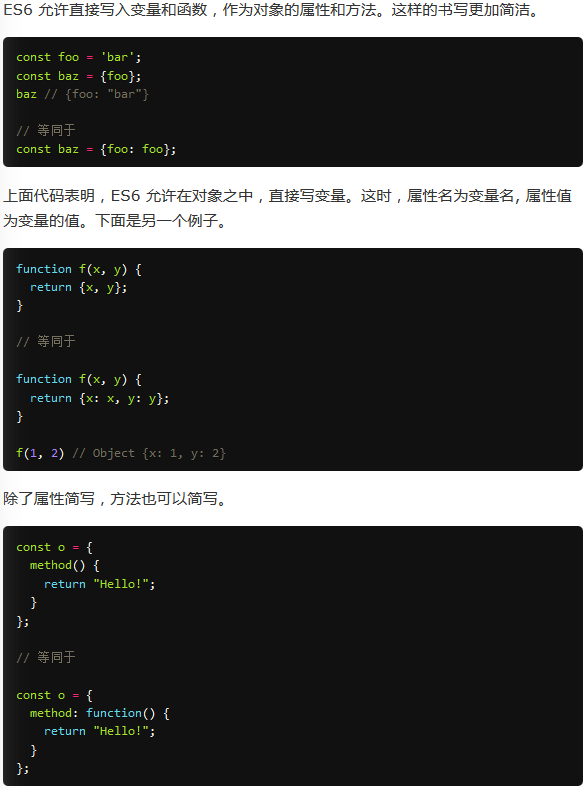


#### 3、ES6对象的简写

下图中的对象中省略了属性名，并且属性的值为函数

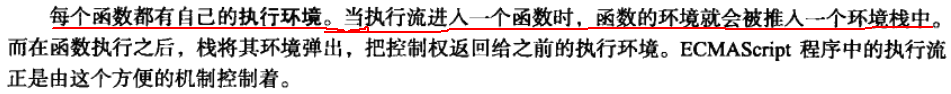


这是ES6的语法：对象的扩展-属性的简洁表示法



#### 4、ES6的let命令

ES5 只有全局作用域和函数作用域，没有块级作用域，但是在ES6中用let声明的变量，只在let命令所在的代码块内有效，let实际上为 JavaScript 新增了块级作用域。



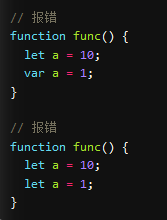
let声明的变量只在块级作用域内有效，不会提升，存在暂时性死区

#### 5、ES6的const命令

const命令声明的常量也是不提升，同样存在暂时性死区，只能在声明的位置后面使用。

#### 6、ES6——不允许重复声明

包含两种情况：1、使用不同的命令重复声明同一个变量；2、使用同一个命令重复声明同一个变量；



#### 7、深拷贝和浅拷贝

var a={

x:1,

y:2

};

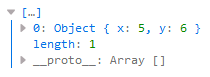
var b=[ ];

b.push(a);

a.x=5;

a.y=6;

console.log(b); 打印结果是：

而不是x：1，y:2

这是因为在 JavaScript 中对象和数组是引用类型，指向同一个内存空间，即数组b中压入的是对象a的地址，而不是a的内容，当a的内容变化后，b的结果也会改变

对于字符串类型，浅复制是对值的复制，对于对象来说，**浅复制是对对象地址的复制**，并没 有开辟新的栈，也就是复制的结果是两个对象指向同一个地址，修改其中一个对象的属性，则另一个对象的属性也会改变，***而深复制则是开辟新的栈***，两个对象对应两个不同的地址，修改一个对象的属性，不会改变另一个对象的属性

#### 8、ES6的箭头函数

var sum = (num1, num2) => { return num1 + num2; }

操作符(var、let、const) 函数名 等号= (参数1、参数2...) 箭头=> {函数体}

箭头函数根本没有自己的this，导致内部的this就是外层代码块的this，也就是说箭头函数是和父级上下文绑定在一起的，this 不会是如你所预期的 Vue 实例

#### 9、如何定义公共函数

参考海航法务系统中getarea函数的定义

步骤：

Step1：在一个公共的.js文件中定义好公共函数getarea，可以用箭头函数来定义

Step2：在需要调用该公共函数的组件里导入函数

import {getarea} from '../utils/area.js'

Step3：在需要调用该公共函数的地方直接调用函数

注意，import进来的变量和函数都相当于全局变量，可以直接使用，所以不要再把getarea放到methods中

#### 10、session、cookie、sessionStorage、localStorage和页面js数据对象的区别

<https://www.cnblogs.com/zr123/p/8086525.html>

sessionStorage

Web storage e

localStorage

浏览器端存储

cookie

分类

session

服务器端存储

**Cookie和session的关系:**

1. 浏览器客户端发起请求

每次请求一个新的页面的时候cookie都会被发送到服务器，所以服务器能够知道客户端cookie中保存了哪些值。

服务器收到请求后首先检查这个客户端的请求里是否已包含了一个session标识（称为session id），session标识本质就是一个存储了session id的cookie

如果客户端请求不包含session id，则说明以前该客户没有访问过该服务器，服务器就会为客户端创建一个session（对象）并且生成一个与此session相关联的session id；这个session id将被在本次响应中返回给客户端保存

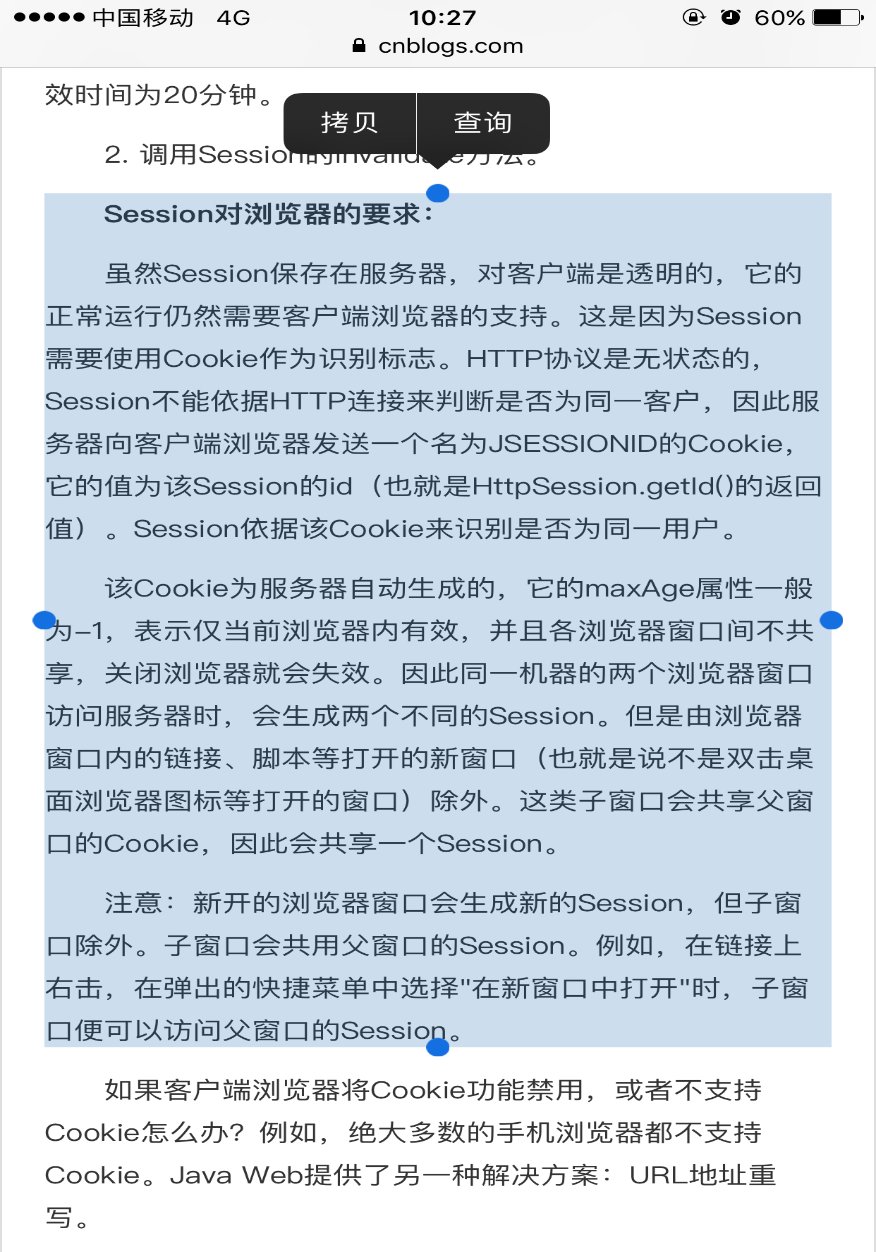
如果已包含则说明以前该客户访问过该服务器并且已经为此客户端创建过session，服务器就按照session id把存储在服务器上的session检索出来使用

**web Storage和Cookie的区别**

cookie是不可或缺的，cookie的作用是与服务器进行交互，作为http规范的一部分而存在的，而web Storage仅仅是为了在本地“存储”数据而生

Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的，cookie的大小是受限的，并且每次请求一个新的页面的时候cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可跨域调用。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **session** | **cookie** | **sessionStorage** | **localStorage** | **页面js数据对象** |
| **数据类型** | 对象 | 字符串 | 对象 | 对象 | 所有JavaScript数据类型 |
| **在浏览器端查看方式** | **/** | document.cookie | window.sessionStorage | window.localStorage | 只能在函数运行时查看 |
| **有效期** | 默认有效时间是30min，可以在配置文件中自行设置 | 不设置时间：浏览器会话期  设置时间：过期日期之前都有效 | 仅在当前浏览器窗口关闭之前有效 | 始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据 | 程序运行中有效 |
| **存储大小** | 没有限制 | 不能超过4K | 5M或更大 | 5M或更大 | **/** |
| **刷新页面** | **/** | 不受影响 | 不受影响 | 不受影响 | 数据消失 |
| **操作方法** | ？ | 无 | setItem,getItem,removeItem,clear等 | setItem,getItem,removeItem,clear等 | **/** |
| **作用域** | **/** | 在所有同源窗口中都是共享 | 在所有同源窗口中都是共享 | 在所有同源窗口中都是共享 | **/** |
| **有无路径** | 不能区分路径，同一个用户在访问一个网站期间，所有的session在任何一个地方都可以访问到 | 路径与域一起构成cookie的作用范围 | **/** | **/** | **/** |



#### 11、json字符串转成json对象

原生方法：

var obj = eval('(' + str + ')');

**或者**

var obj = JSON.parse(str); //由JSON字符串转换为JSON对象

jQuery方法：

var obj = str.parseJSON();

#### 12、callee、this、caller

参考《JavaScript高级程序设计第三版》5.5.4函数的内部属性

在函数内部有两个特殊的对象：arguments和this，其中arguments有一个属性叫callee，该属性的值为一个指针，指向拥有这个arguments对象的函数即

1、callee：指向这个函数本身 函数名==arguments.callee

2、this：（函数被调用之后才有this）函数的执行环境，**【this】指向于此函数的调用者（拥有者）**当函数在全局作用域中调用时，this引用的是全局对象window；当把这个函数赋给对象o并在o上调用该函数，this引用的就是对象o

3、caller：调用当前函数的函数（的引用），如果是在全局环境中调用当前函数，则caller的值为null

4、常识：回调函数会改变this的指向

在律所详情页面，查询律师信息时，必须先把this的值保存在另外的that中



5、回调函数没有自己的this

6、不要在选项属性或回调上使用[箭头函数](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Functions/Arrow_functions)

7、https://www.cnblogs.com/gavinyyb/p/6286750.html

Javascript 支持将函数作为参数传递，回调函数变量指向的函数对象都未与任何上下文绑定，所有未与明确上下文绑定的变量都是自由变量，浏览器器中所有自由变量的上下文都是 window 对象。

#### 12、在JavaScript中Shim和Polyfill有什么区别



### 十、H5

#### 1、template元素的渲染

不会渲染template本身，只会渲染包裹在template里面的内容，对于如下代码：渲染结果为

跟template一样不会被渲染的还有<router-view> 组件，只会渲染路径匹配到的视图组件

#### 2、HTML <style> scoped 属性

scoped 属性是 HTML5 中的新属性。scoped 属性是一个布尔属性。

如果使用该属性，则样式仅仅应用到 style 元素的父元素及其子元素。

#### 3、body的默认宽度和高度是多少

前提是要有文档类型声明<!DOCTYPE html>，如果没有文档类型声明，即使用%给body或者body的子元素设置高度，也会起作用

body的默认宽度等于整个网页背景的宽度，body在没有设置具体值时会自动默认百分比来定义宽度。因此[电脑显示器](https://www.baidu.com/s?wd=%E7%94%B5%E8%84%91%E6%98%BE%E7%A4%BA%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4uWRzmWK-rHbsnHIBPyRk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3nHbknjm1n1bdn1c4nj6vn0)尺寸不同body具体值也会不同

body没有默认的高度

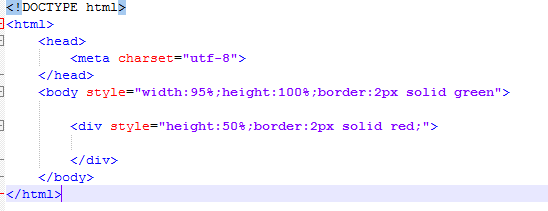
建议再加强一下html 元素的知识.  
首先任何元素都是没有高度的, 高度均是有内容撑开的.  
行内元素的宽度也是由内容撑开, 块元素宽度默认是100% 即为父元素宽度的 100%

给html和body的height设置为100%是没有效果的，height=100%是继承父元素的高度，  
html和body的本身就是根元素，父元素

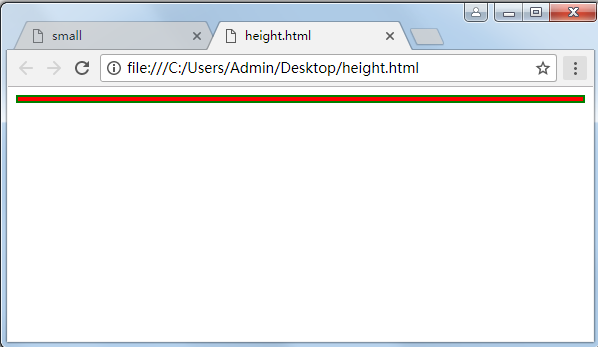
但给html和body的width设置为100%是有效果的

以下讨论用%设置子绝对定位的子元素高度会有什么效果

结论一：先不给子元素设置宽度。如果父元素没有高度（例如body元素就没有高度），用%给子元素设置高度后，根本不起作用，撑不开父元素body（用px给子元素设置高度是可以撑开的）

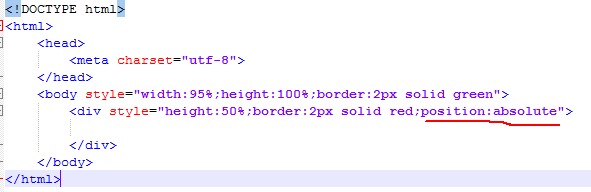


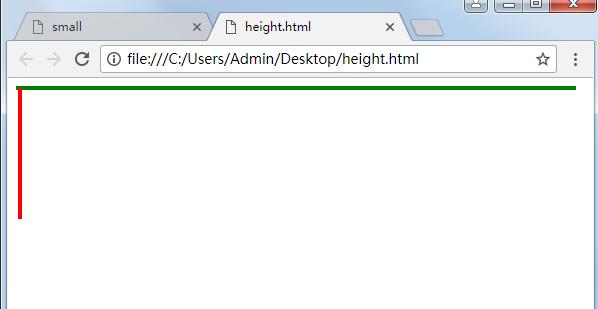
效果如下图:



运行后发现body包围着子元素，子元素有默认的宽度；修改子元素div的height，不起任何作用。

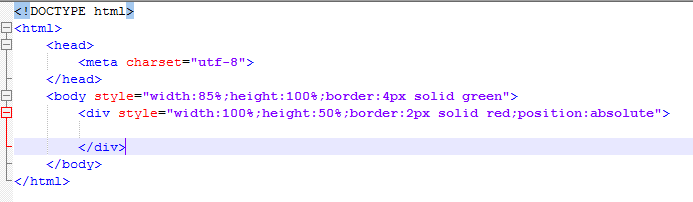
结论二：，如果子元素被绝对定位后用%设置高度就起作用了

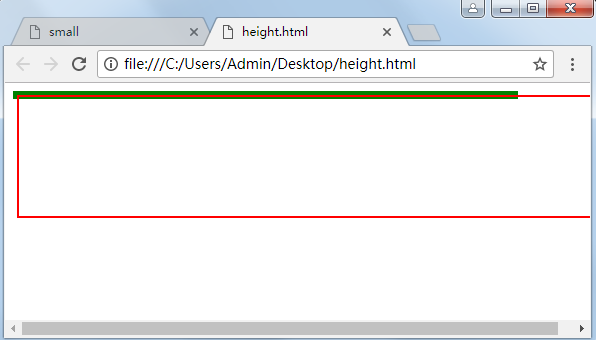




但这时候子元素的宽度也没有了，本来子元素默认是有宽度的，宽度为父元素body的100%，为什么绝对定位后竟然没有宽度了呢？这是因为你用absolute定位后元素就已经脱离文档流，他已经没有父元素了，有的只是一个定位上下文（父元素可不等于定位上下文），所以原来默认的占父元素100%的宽度就没有了。你让元素脱离了文档流，又要文档把它算进里面去计算宽度，这要求有点精分。

结论三：元素脱离文档流后，原来在文档里很多样式，包括默认的宽度对该元素都没有约束了，该元素只能以默认的定位上下文即body作为参考，（元素浮动后和文档就没有关系了，只能给它独立设定宽度、高度、边距等），也可以理解为此时元素不再受制于父元素，（或者说该元素已经没有任何父元素了）只受制于定位上下文（在该例中就是body）





从图中可以看出，此时绝对定位的div元素高度是body的50%，宽度是body的100%，而且该div元素并没有被包围在绿色的边框中，这是因为本来body中只有一个边框为红色的div，现在该div被绝对定位后脱离了文档流，body里面啥都没有了，没有任何内容来撑开body。Div元素只是相对于body来定位，而不是说该div元素是body的子元素

结论三：绝对行为的元素不设置top left right bottom会有什么影响

结论：<https://segmentfault.com/a/1190000003109367>



补充：当top设置为0px时，该元素的containing block的上边缘应该是其父元素创建的containing block的上边缘；和left设置为0px时，该元素的containing block的左边缘应该是其父元素创建的containing block的左边缘；此时该元素会覆盖其他子元素

#### 4、websocket

websocket也是html5的新增加内容之一,号称是**下一代客户端/服务器异步通信办法，可以实现消息推送**

**在控制台打印’WebSocket’ in window，如果打印结果为true，说明该浏览器支持WebSocket**



#### 5、canvas

### 十一、开发环境

#### 1、SVN

SVN安装及下载：<https://jingyan.baidu.com/article/358570f6638aa4ce4724fcf7.html>

如果想更方便的提交或者更新代码，可以在HBuilder中配置SVN

HBuilder配置SVN：<http://blog.csdn.net/Sam_Zhang_/article/details/51604912>

1、



1. callback
2. 是其父元素创建的containing block的左边缘；此时该元素会覆盖其他子元素

### 十二、打包上线

npm run build 打包上线之后如何在本地查看效果：

<https://www.cnblogs.com/qiu-Ann/p/7477593.html>

运行完npm run build之后会生成dist目录

### 十二、webpack

webpack支持三种模块化语法：CommonJS，AMD，ES6

npm脚本

webpack天生不支持.css文件类型，要处理.css类型的文件，需要依赖loader

### 问题

Pushlet和Websocket的区别

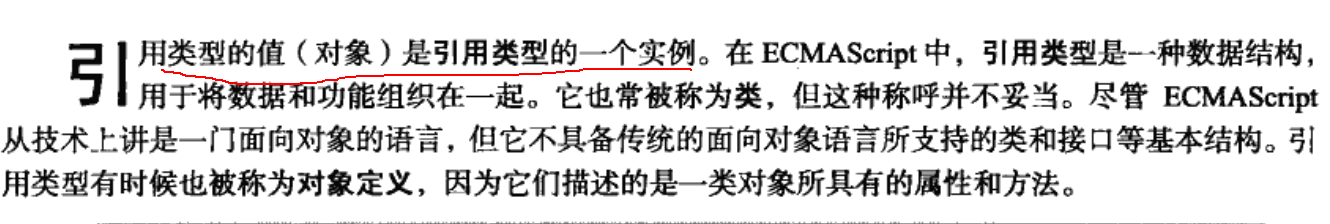
打包工具和构建工具：程序的构建，包括编译、运行、打包和文档生成等，

routes中定义的路由对象有show 属性，难道还可以自定义属性吗

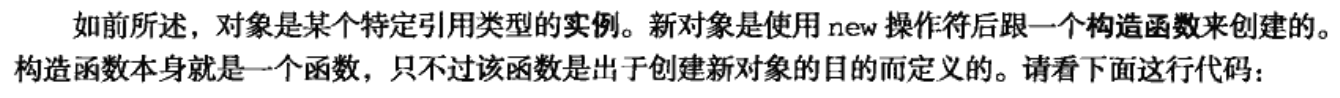
webpack是前端的构建工具；gradle也是一款构建工具，Angular-CLI是webpack内核的。



ES5中没有类的概念，只有引用类型，第五章介绍了每种引用类型。



对象是引用类型的实例，Object,Function,Array,Date都可以通过new操作符产生实例



构造函数本身就是个函数，只不过这个函数时出于创建新对象的目的而定义的。ES5中的构造函数还不是constrator,而是Object( ),Function( ),Array( ),Date( )等

