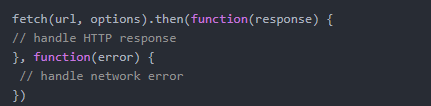
在开发过程中,我们向服务端发送请求,一般会使用三种方式：

1, XMLHttpRequest(XHR)

1. Fetch
2. jQuery实现的AJAX。  
   其中, XMLHttpRequest(XHR)和Fetch是浏览器的原生API，jquery的ajax其实是封装了XHR

jQuery作为一个使用人数最多的库，其AJAX很好的封装了原生AJAX的代码，在兼容性和易用性方面都做了很大的提高，而且jQuery还把jsonp装在了AJAX里面，这样我们就可以开心的跨域了！！！！对比原生AJAX的实现，使用jQuery实现的AJAX就异常简单了．但是，我们仍然逃脱不了一个问题，回调地狱。。。。

Fetch使用说明：

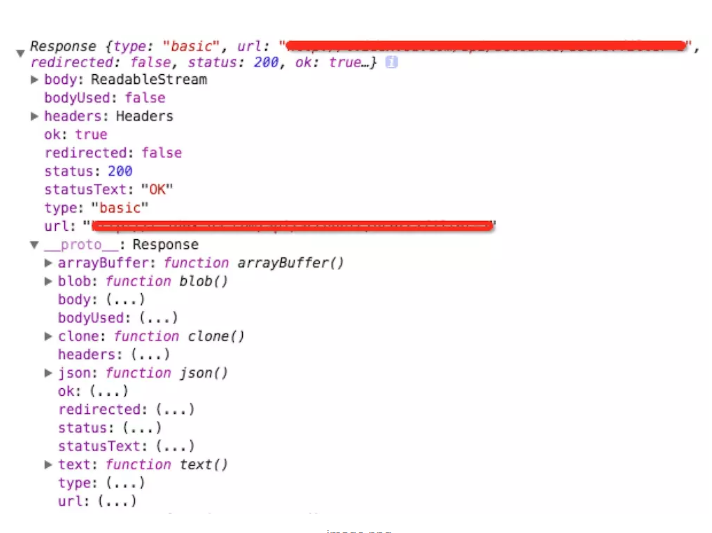


说明：  
a. fetch api返回的是一个promise对象  
b.Options:

* method(String): HTTP请求方法，默认为GET
* body(String): HTTP的请求参数
* headers(Object): HTTP的请求头，默认为{}
* credentials(String): 默认为omit,忽略的意思，也就是不带cookie;还有两个参数，same-origin，意思就是同源请求带cookie；include,表示无论跨域还是同源请求都会带cookie

**问题：**  
使用 fetch 无法取消一个请求。因为 Fetch基于 Promise，而 Promise 无法做到这一点。

c.第一个then函数里面处理的是response的格式，这里的response具体如下：



**object.keys**

Object.keys是es5中新增的方法，用来获取对象自身所有的可枚举的属性名，但不包括原型中的属性，然后返回一个由属性名组成的数组

if (type == 'GET') {

let dataStr = ''; //数据拼接字符串

Object.keys(data).forEach(key => {

dataStr += key + '=' + data[key] + '&';

})

if (dataStr !== '') {

dataStr = dataStr.substr(0, dataStr.lastIndexOf('&'));

url = url + '?' + dataStr;

}

}

Get请求参数会显示在url地址栏，data是参数，遍历这些参数让参数变成字符串形式

Key是每一个参数的属性名，data【key】是每一个参数的属性值拼接上“=”和“&”之后就变成了 f=3&rsv\_bp=0&rsv\_idx=1&tn=baidu& 形式的字符串

即 属性名=属性值&属性名=属性值&

如果拼接之后有数据的话从参数开始的地方到末尾截取它

跳转路径的后面拼接“？”后面跟着所有的参数

如：

<http://tieba.baidu.com/f/search/res?qw=object.keys&sm=2&cf=1&ie=utf-8&>

##### 扩展运算符...mapState,意思是把vuex的state数据映射到计算属性里面,映射后的名字为city,所以上面直接用this.city,就不用写this.$store.state.city这个长串了

##### 

mapActions 工具函数会将 store 中的 dispatch 方法映射到组件的 methods 中。和 mapState、mapGetters 也类似，只不过它映射的地方不是计算属性，而是组件的 methods 对象上

Vuex：https://blog.csdn.net/wenjiusui8083/article/details/80221636

1.**this.$router.go(val)** => 在history记录中前进或者后退val步，当val为0时刷新当前页面。

2.**this.$router.push(path)** => 在history栈中添加一条新的记录。

html figure和image的区别如下：  
1、figure元素是一种元素的组合，可带有标题（可选）。figure标签用来表示网页上一块独立的内容，将其从网页上移除后不会对网页上的其他内容产生影响。figure所表示的内容可以是图片、统计图或代码示例。  
figure拥有一个子标签——figcaption标签。  
实例：  
用作文档中插图的图像：  
<figure> <p>黄浦江上的的[卢浦大桥](https://www.baidu.com/s?wd=%E5%8D%A2%E6%B5%A6%E5%A4%A7%E6%A1%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)</p>  
<img src="shanghai\_lupu\_bridge.jpg" width="350" height="234" />  
</figure>  
2、img 元素向网页中嵌入一幅图像。这个标签并不会在网页中插入图像，而是从网页上链接图像。<img> 标签创建的是被引用图像的占位空间。  
用法如下：  
<img src="/i/eg\_tulip.jpg" alt="上海鲜花港 - 郁金香" />  
区别：figure用于对元素进行组合。多用于图片与图片描述组合。而img只是一个图片元素而已。可以嵌套在figure中使用。

**1.29号**

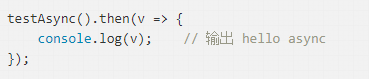
## async 和 await 在干什么

任意一个名称都是有意义的，先从字面意思来理解。async 是“异步”的简写，而 await 可以认为是 async wait 的简写。所以应该很好理解 async 用于申明一个 function 是异步的，而 await 用于等待一个异步方法执行完成。

### async 起什么作用

async 函数返回的是一个 Promise 对象。从[文档](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Statements/async_function" \t "https://segmentfault.com/a/_blank)中也可以得到这个信息。async 函数（包含函数语句、函数表达式、Lambda表达式）会返回一个 Promise 对象，如果在函数中 return 一个直接量，async 会把这个直接量通过 Promise.resolve() 封装成 Promise 对象。

async 函数返回的是一个 Promise 对象，所以在最外层不能用 await 获取其返回值的情况下，我们当然应该用原来的方式：then() 链来处理这个 Promise 对象，就像这样



联想一下 Promise 的特点——无等待，所以在没有 await 的情况下执行 async 函数，它会立即执行，返回一个 Promise 对象，并且，绝不会阻塞后面的语句。这和普通返回 Promise 对象的函数并无二致。

### await

一般来说，都认为 await 是在等待一个 async 函数完成。不过按[语法说明](https://developer.mozilla.org/docs/Web/JavaScript/Reference/Operators/await" \t "https://segmentfault.com/a/_blank)，await 等待的是一个表达式，这个表达式的计算结果是 Promise 对象或者其它值（换句话说，就是没有特殊限定）。

因为 async 函数返回一个 Promise 对象，所以 await 可以用于等待一个 async 函数的返回值——这也可以说是 await 在等 async 函数，但要清楚，它等的实际是一个返回值。注意到 await 不仅仅用于等 Promise 对象，它可以等任意表达式的结果，所以，await 后面实际是可以接普通函数调用或者直接量的

await 等到了它要等的东西，一个 Promise 对象，或者其它值，然后呢？我不得不先说，await 是个运算符，用于组成表达式，await 表达式的运算结果取决于它等的东西。

如果它等到的不是一个 Promise 对象，那 await 表达式的运算结果就是它等到的东西。

如果它等到的是一个 Promise 对象，await 就忙起来了，它会阻塞后面的代码，等着 Promise 对象 resolve，然后得到 resolve 的值，作为 await 表达式的运算结果。

# Vue基础语法之@click、时间修饰符@click.stop与@click.prevent、按键修饰符（如@keyup.enter）

1、绑定监听@click:（以监听click为例，其他如keyup，用法类似）  
  v-on:click="fun"  
  @click="fun"  
  @click="fun(参数)"

2、@click.stop与@click.prevent

@click.stop 阻止事件冒泡

@click.prevent 阻止事件的默认行为，

<a href="http://www.baidu.com" @click.prevent="test4">百度一下</a>   //阻止a标签跳转，仅执行函数test4

<form  action="/xxx"   @submit.prevent="test5">   //阻止表单提交，仅执行函数test5

         <input type="submit" value="注册">  
</form>

3、按键修饰符

@keyup.enter

//按下enter时，执行方法test7

<input type="text" @keyup.enter="test7">

methods: {

      test7 (event) {  
        console.log(event.keyCode)  
        alert(event.target.value)  
      }

}

**@input**

@input是用来定义输入的，是接收其他组件传过来的数据的。相当于指令的值绑定，无论是单向的(@)还是双向的(=)。都是将父作用域的值“输入”到子作用域中，然后子作用域进行相关处理。

# **Transition**

Vue1.0中transition做为标签的行内属性被vue支持。但在Vue2.0中。Vue放弃了旧属性的支持并提供了transition组件，transition做为标签被使用。

###### **使用transition完成任何元素进入/离开的过渡组件需要满足下列条件**

* 条件渲染（v-if）
* 条件展示（v-show）
* 动态组件
* 组件根节点
* <transition>中只有name属性，不可添加其他标签属性
* <transition>中只能有一个子元素并且该子元素需要有v-show或者v-if来控制是否显示

##### 过渡CSS类名

###### <transition>中的name属性用于 替换 vue钩子函数中的类名v-

* v-enter: 定义进入过渡的开始状态。在元素被插入时生效，在下一个帧移除。
* v-enter-active: 定义进入过渡的结束状态。在元素被插入时生效，在transition/animation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生效，在下一个帧移除。
* v-leave-active: 定义离开过渡的结束状态。在离开过渡被触发时生效，在transition/animation完成之后移除。

1.31

二级路由

只需要在路由中加上一个childern字段即可



# [Vue.nextTick 的原理和用途](https://segmentfault.com/a/1190000012861862)

* 用法：

在下次 DOM 更新循环结束之后执行延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。





祥见网址：<https://segmentfault.com/a/1190000012861862>

**minxin使用及介绍**

vue中提供了一种混合机制--mixins，用来更高效的实现组件内容的复用。最开始我一度认为这个和组件好像没啥区别。。后来发现错了。下面我们来看看mixins和普通情况下引入组件有什么区别？

组件在引用之后相当于在父组件内开辟了一块单独的空间，来根据父组件props过来的值进行相应的操作，单本质上两者还是泾渭分明，相对独立。

而mixins则是在引入组件之后，则是将组件内部的内容如data等方法、method等属性与父组件相应内容进行合并。相当于在引入后，父组件的各种属性方法都被扩充了。

单纯组件引用：

父组件 + 子组件 >>> 父组件 + 子组件

mixins：

父组件 + 子组件 >>> new父组件

作用：多个组件可以共享数据和方法，在使用mixin的组件中引入后，mixin中的方法和属性也就并入到该组件中，可以直接使用。钩子函数会两个都被调用，mixin中的钩子首先执行。

下面给大家介绍vue mixin的用法，具体介绍如下所示：

~定义一个 js 文件(mixin.js)

export default {

data() {

return {

name: 'mixin'

}

},

created() {

console.log('mixin...', this.name);

},

mounted() {},

methods: {}

}

~~在vue文件中使用mixin

import '@/mixin'; // 引入mixin文件

export default {

mixins: [mixin]

}

详见网址：<https://blog.csdn.net/qq_36838191/article/details/81004590>

**Minxin**

混入minxins：分发vue组件中可复用功能的灵活方式。混入对象可以包含任意组件选项。组件使用混入对象时，所有混入对象的选项将混入该组件本身的选项。

选项合并：组件选项和对象选项同名时，将以恰当方式混合。

数据对象：和组件数据冲突时以组件数据优先

同名钩子函数：混合为一个数组，都将被调用，混入对象钩子先调用。

值为对象选项：methods，components，将被混合为同一个对象，两对象键名冲突时，取组件对象的键值对。

混入方式：全局混入，组件混入

Mixins一般有两种用途：

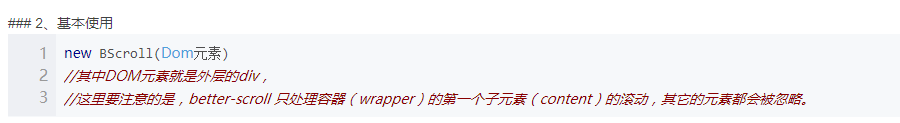
1、在你已经写好了构造器后，需要增加方法或者临时的活动时使用的方法，这时用混入会减少源代码的污染。

2、很多地方都会用到的公用方法，用混入的方法可以减少代码量，实现代码重用。

具体用法详见：<https://blog.csdn.net/sinat_17775997/article/details/76570822>

**better-scroll**

### **1、使用better-scroll的基本条件**

* 必须包含两个大的div，外层和内层div
* 外层div设置可视的大小(宽或者高)-有限制宽或高
* 内层div，包裹整个可以滚动的部分
* 内层div高度一定大于外层div的宽或高，才能滚动
* 

**基本使用案列参考：<https://blog.csdn.net/qq_34664239/article/details/80683976>**

**include**

为什么要使用？

使用include 指令可以方便的把在多个页面中要重复显示的内容抽取出来，大大的减少代码的从重复量，方便我们对重复内容的维护。

Include指令分为：静态指令和动态指令

**使用静态include指令引入页面**

说明：在jsp中插入一个或多个包含文件或代码的文件（jsp,html,inc,txt），把文件插入后与原来的jsp文件合并成一个新的jsp页面

语法：<%@include file=”从页面文件地址”%>

**使用动态include指令引入页面**

说明：在jsp中插入一个或多个包含文件或代码的文件（jsp,html,inc,txt），从页面内容和主页面容然具有很强的独立性

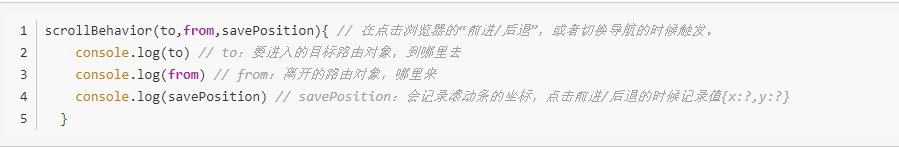
详见网址：<https://www.cnblogs.com/Marydon20170307/p/7338600.html>

# vue路由中设置滚动行为(scrollBehavior)

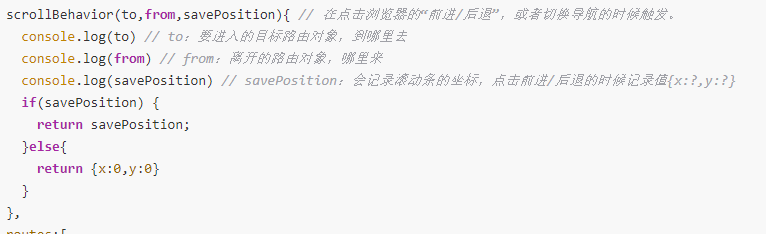
在文档页面([http://localhost:8080/document](http://localhost:8080/document" \t "https://blog.csdn.net/wandoumm/article/details/_blank))拉动滚动条，然后刷新浏览器会发现滚动条依然在原来的位置，这是浏览器的默认行为，会记录浏览器滚动条默认位置。

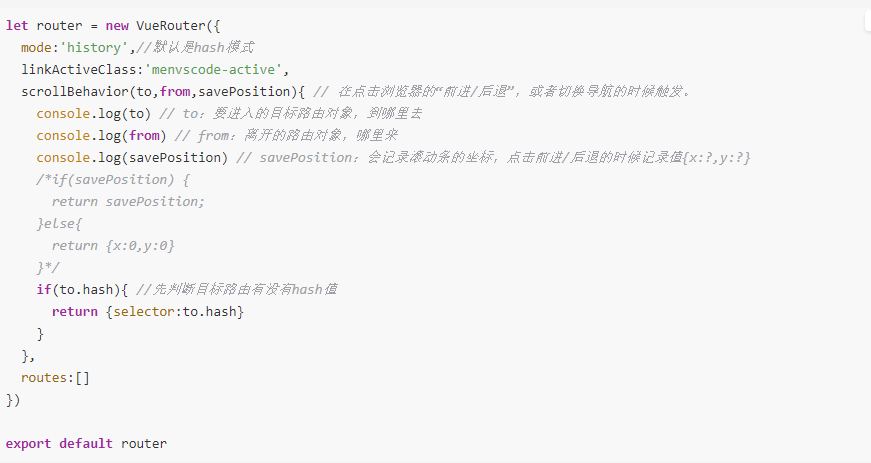
        但是点击浏览器“前进/后退”按钮，会发现当初那个页面的滚动条从0开始了，没有记录上一次滚动条的位置。现在要求点击浏览器“前进/后退”按钮，页面滚动条要记录上一次的位置，这时需要设置它的的滚动行为。

        这时候需要在路由配置中设置 scrollBehavior(to,from,savePosition)函数，**函数有三个参数。scrollBehavior() 函数在点击浏览器的“前进/后退”，或者切换导航的时候触发。**



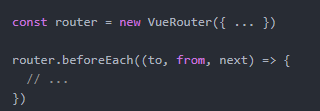
  实现滚动行为的代码





# 详见网址：https://blog.csdn.net/wandoumm/article/details/80270865

### **如何设置一个全局守卫**



当一个导航触发时，全局前置守卫按照创建顺序调用。守卫是异步解析执行，此时导航在所有守卫 resolve 完之前一直处于 **等待中**。

每个守卫方法接收三个参数：

to: Route: 即将要进入的目标 [路由对象](https://link.jianshu.com?t=https://router.vuejs.org/zh-cn/api/route-object.html" \t "_blank)

from: Route: 当前导航正要离开的路由

next: Function: 一定要调用该方法来 **resolve** 这个钩子。执行效果依赖 next 方法的调用参数。

next(): 进行管道中的下一个钩子。如果全部钩子执行完了，则导航的状态就是 **confirmed** （确认的）。

next(false): 中断当前的导航。如果浏览器的 URL 改变了（可能是用户手动或者浏览器后退按钮），那么 URL 地址会重置到 from 路由对应的地址。

next('/') **或者** next({ path: '/' }): 跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。你可以向 next 传递任意位置对象，且允许设置诸如 replace: true、name: 'home' 之类的选项以及任何用在 [router-link 的 to prop](https://link.jianshu.com?t=https://router.vuejs.org/zh-cn/api/router-link.html" \t "_blank) 或 [router.push](https://link.jianshu.com?t=https://router.vuejs.org/zh-cn/api/router-instance.html%23%E6%96%B9%E6%B3%95" \t "_blank) 中的选项。

next(error): (2.4.0+) 如果传入 next 的参数是一个 Error 实例，则导航会被终止且该错误会被传递给 [router.onError()](https://link.jianshu.com?t=https://router.vuejs.org/zh-cn/api/router-instance.html%23%E6%96%B9%E6%B3%95" \t "_blank) 注册过的回调。

**确保要调用** next **方法，否则钩子就不会被 resolved。**

