#### 指令

V-bind 缩写 ‘:’

<span v-bind:title="message"> message </span>

不加‘:’的引号里面就是字符串，加了会生成一个作用域，引号里面可以是变量，表达式等。

v-if=”true/false” 移除/添加/更新dom元素

v-for of/in

v-on添加一个事件监听器 缩写”@”

v-modle 双向绑定，表单元素都可以用，修饰符，lazy：input不在输入时触发change触发 ，trim，number：input输入转换为数值类型。

new Vue({})创建vue实例

data对象中的所有属性都会加入到vue的响应式系统中，但是只有被创建时的data的属性是响应式的，vm.b=”new”添加新的属性不会触发视图更新,后面要用到一开始初始化就好了。

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

前缀 $，以便与用户定义的属性区分开来，如$data

##### 生命周期钩子

beforeCreate ：new Vue实例化后，数据观测(data observer) 和 event/watcher 事件都没准备好的时候调用（this变量还不能使用，在data下的数据，和methods下的方法，watcher中的事件都不能获得到）

created：实例创建完成后，但还不能对"dom"节点进行操作的时候调用。

beforeMounte：页面挂载之前，页面才开始渲染执行第一次render函数的时候

mounted：页面渲染完，挂载完成可以进行dom操作的时候

beforeUpdate：数据改变但是还没有渲染在页面上的时候

update：数据改变并且页面重新渲染完的时候

beforeDestroy：实例被销毁之前会执行的函数

destroyed：实例被销毁后



##### 模板语法

{{}} 双大括号就是普通的文本，可以是单个表达式

v-html才会解析为真正的html，但是注意动态添加html可能会引起xss攻击，要保证没有重要数据

V-once 当数据改变时，插值处的内容不会再更新

计算属性：computed，getter和setter

侦听属性：watch，在数据变化时执行异步或开销较大的操作时

`\_.debounce` 是一个通过 Lodash 限制操作频率的函数。

##### Class和style

:class=””，引号里面可以是对象也可以是数组，也可以在这里绑定一个返回对象的[计算属性](https://cn.vuejs.org/v2/guide/computed.html)。当在一个自定义组件上使用 class 属性时，这些类将被添加到该组件的根元素上面。这个元素上已经存在的类不会被覆盖。

Style同理

Vue 会尽可能高效地渲染元素，通常会复用已有元素而不是从头开始渲染。所以key值将复用的元素独立开

v-show 不支持 <template> 元素，也不支持 v-else

##### 数组更新检测

##### 事件处理

传入参数$event 可以使用原生事件方法

v-on修饰符

.native 在一个组件的根元素上直接监听一个原生事件

阻止事件冒泡的方法：

1. v-on监听事件加修饰符 .stop
2. 传参 $event , event.stopPropagation();
3. .self 只有自身触发的事件才执行，冒泡到的就不执行

.prevent阻止默认事件发生

event.preventDefault() 阻止默认事件发生

使用修饰符注意点：注意使用的顺序，除了.once可用于组件事件上，其他都用在原生事件

##### 组件

全局注册/局部注册

Vue.component 全局注册

在模块系统内注册局部组件，通过 import/require 使用一个模块系统

prop

父组件向子组件传值 大小写不敏感

对传过来的数据类型验证

在 JavaScript 中对象和数组是通过引用传入的，所以对于一个数组或对象类型的 prop 来说，在子组件中改变这个对象或数组本身将会影响到父组件的状态。

非prop？

##### 自定义事件

$emit(自定义事件名，参数)

组件上使用v-model

<input v-model="searchText">

等价于

<input v-bind:value="searchText" v-on:input="searchText = $event.target.value" >

##### 插槽

添加v-slot指令 缩写#

##### 自定义指令

directives：{

focus：{

inserted：function(el){

el.focus()

}

}

}

inserted

bind

once

el (元素),binding(对象), vnode(虚拟节点)

dinding.value 指令绑定的值

##### 过渡动画

动画部分用<transition>包裹

enter：

进入opacity从0->1 ,v-enter->v-enter-to ， v-enter-active过渡状态，比如设置transition：1s

leave:

离开opacity从1->0 ,v-leave->v-leave-to

mode ：in-out 和out-in

transform：translate

##### 路由

URL对应视图

router->index.js 配置路由文件

main.js 可以引入一些公共的文件

默认hash模式地址有个#

history模式，在配置路由文件 mode：’history’

要注意：直接访问http://yoursite.com/user/id，可能会返回一个404，需要后台配置支持，或者

##### 声明式

<router-link to=”path” exact tag=”element” event></router-link>

点击会默认添加class ，router-link-active，可自己设置，全局配置路由文件linkActiveClass，局部标签添加active-class

全包含匹配=>精确匹配 exact

配置路由文件path:’\*’ ,\*通配符，匹配路径，可以用来找不到页面

重定向 redirect:”path”/{path:”path”}/{name:”name”}/(to)=>{return “path”}

动态绑定path

子路由 和默认子路由

{

path:'/',

component: parent,

children :[

{

path:'',//默认子路由path为空，name写成父组件的

name:parent,

component:child1

},{

path:'child2',

name:parent,

component:child1

},{

path:'/child3',//路径不嵌套但是组件仍然嵌套加个/

name:parent,

component:child1

}

]

}

##### 滚动行为

滚动条前进后退滚动条位置不变

hash定位到元素标签类似于锚点 元素id就是hash值，#id

scrollBehavior:function(to,from,savedPosition){} //savedPosition 离开时滚动条位置

注：这个功能只在支持 history.pushState 的浏览器中可用。

也可以返回一个 Promise 来得出预期的位置描述

动态路由 path：“/path/:id?”//?表示动态部分可能有可能没有

对组件注入：$router 路由实例

$route 当前激活的路由信息对象每个组件都有。就是当前所在组件的路由信息，是个对象

$route.params.id 得到的是动态路由的动态地址部分

:to表示动态的路由形式

:to=”{path:’path’,query:{queryInfo:’queryInfo’}}” => path?queryInfo=queryInfo

path：路径

params对象：动态路由参数

query对象：URL查询参数，就是地址？后面的那一坨可以传数据

meta ：{}路由下标，加一些标识。$route.matched对应meta信息

##### 编程式导航

浏览器的历史记录，前进和后退

$router实例的方法

back后退

forword前进

go()前进和后退的步数正负整数 负数后退，正数前进 0刷新当前页

push，导航到不同url，历史记录栈添加一个url记录，可以是字符串也可以是对象（但如果提供了 path，params 会被忽略）

replace，导航到不同URL，替换历史记录栈的当前记录

##### 命名路由

<router-view> 可以添加class设置样式，class会添加到组件的根节点

可以在界面中拥有多个单独命名的视图，<router-view> 加个name，没加name的默认为default。components：{name1：组件，name2：组件...}

<transition>

<router-view/>

</transition>

做路由的过渡动画

props?

##### 导航守卫

用来通过跳转或取消的方式守卫导航,参数或查询的改变并不会触发进入/离开的导航守卫

导航钩子函数

切换不同导航的时候执行的函数。

全局的

进入到导航之前

router.beforEach(to,from,next){

next()//有这一步才会进入到导航跳到路由页面

}

也可next(‘path’)，指定进入导航路径

可以用来做重定向功能，比如判断是否登录，没有登陆就跳转到登陆页面

afterEach

局部的

针对某一个具体的路由 beforeEnter

组件里的

beforeRouteEnter，是在beforeCreate之前执行所以this是undefined，可以使用回调函数传入vm，通过vm访问组件实例

beforeRouteUpdate是否更新，比如带有动态参数的路径/foo/1跳转到/foo/2由于foo是一样的，会渲染同样的 Foo 组件，因此组件实例会被复用。this也可用

beforeRouteLeave 离开时 禁止用户在还未保存修改前突然离开

如果目的地和当前路由相同，只有参数发生了改变 (比如从一个用户资料到另一个 /users/1 -> /users/2)，你需要使用 [beforeRouteUpdate](https://router.vuejs.org/zh/guide/essentials/dynamic-matching.html" \l "%E5%93%8D%E5%BA%94%E8%B7%AF%E7%94%B1%E5%8F%82%E6%95%B0%E7%9A%84%E5%8F%98%E5%8C%96) 来响应这个变化。

##### 路由懒加载

代码分割+异步组件

const Foo = () => import('./Foo.vue')

const router = new VueRouter({

routes: [

{ path: '/foo', component: Foo }

]})

##### 自定义指令

directive

##### 过滤器

用在 {{}}和v-bind

##### 插件

做成插件可以不用每次用都调用

1. 放在原型上

Vue.prototype.$名称= “”

1. 定义一个对象，对象里是方法传参Vue，然后Vue.use(对象)

##### 异步组件

require分割

()=>import(‘URL’)

##### 打包

npm run build

apache环境

www文件是根目录

config=>index.js 路径配置

##### Vuex

new Vuex.store()

mutation里面必须同步的

state 公共数据　　mapState把state里的数据映射出来

getter相当于store里的计算属性 mapGetters

组件通过dispatch（方法名，传参）方法调用store里的actions

actions：{

方法名( ctx,传参){

ctx.commit(方法名，传参)

}

}

actions通过commit方法触发mutations， mapMutations

提交mutations　改变state数据

用之前要import {mapState} from vuex 引用

...mapState([‘数据名’])，扩展并映射，里面可以是数组也可以是对象

module就是对复杂的文件进行拆分

区别：Action 可以包含任意异步操作。可以不用，但是最后都要调用mutations。mapMutations

#### axios

get (url,{params：{}})

post(url ,{数据})

配置固定的请求地址

creat({

baseURL:””,

timeout:time,

header:请求头,

responseType：返回数据格式(例如json),

params：{},

transformRequest[function(data){//post，put，patchl}],

transformRespons[function(){}],

cancelToken:source.token //取消请求

})

var cancelToken=axios.cancelToken

var source=cancelToken.source()

source.cancel()

判断是axios自己取消if(axios.isCancel(erro)){}

queryString.stringify 数据转换成字符串形式，queryString是vue提供的，import引用即可

并发请求

axios.all([axios1,axios2])并发请求，发送的是一个数组，返回的也是一个数组

axios.spread(val1,val2),可接受将返回的数组值一一对应，val1是axios1的返回，val2是axios2的返回

全局引用

在组件内this.$http.get(url)

##### 响应式原理

Object.defineProperty() 将data对象的属性转为setter/getter，它们让vue能够追踪依赖，当属性被访问或修改的时候通知变更。每个组件对应一个watcher实例，它会在 组件渲染过程中接触过的数据属性记录成依赖，当依赖项的setter触发就通知watcher重新渲染关联组件

set

Object.assign() 用原对象与要混合进去的对象的属性一起创建一个新的对象。才能触发更新

发布者-订阅者模式

异步更新队列

在队列里所有的watcher ，在下一个的事件循环“tick”中，Vue 刷新队列并执行实际 (已去重的) 工作，更新完DOM执行nextTick()

拦截器