常用指令;
angular：ng-model,ng-controller，ng-repeat

指令：扩张 html 标签的一个功能及属性
vue:v-model:一般用到表单元素(input)
2.0 自定义组件
组件模板：
1.0 支持片段代码，2.0 不支持片段代码
之前：
<template>组件代码</template>
现在：
必须有一个根元素包裹所有的子元素
组件的定义：
Vue.extend;
2.0 版本：有一些改动，即使可以使用，也不是用；

推出一个简洁的组件定义方式：
直接定义 json:var home={template:"template 的选择其",以及定义组件的所有的包含参数都是可以使用的}-》需要在实例化 Vue 的时候，定义 components:{'组件的名称'：home(声明组件的变量)}====》类似与 Vue.extend

1.0 版本：Vue.component(组件名称,{-->2.0 也可以使用
data(){},
methods:{},
template:
})
生命周期：
1.0 版本：
init:初始化
created：创建
beforeCompile:模板编译之前
complied:编译后
ready:准备==>经常性使用　《===》2.0 的 mounted
beforeDestroy:销毁之前
destroy:销毁
2.0 版本：
beforeCreate:组件刚刚被创建-》自身什么都没有
created:实例已经被创建，属性什么之类的已经与自身绑定
beforeMount:模板编译之前
mounted：模板已经编译完成====代替之前的 ready
beforeUpdate：组件更新之前
updated:组件更新完成
beforeDestroy:销毁之前
destroy:销毁
第二天再加上以后：
常用指令;
angular：ng-model,ng-controller，ng-repeat

指令：扩张 html 标签的一个功能及属性
vue:v-model:一般用到表单元素(input)　双向数据绑定

循环：
angular :ng-repeat
vue: v-for:arr---》数组下表(v,$index) in arr;可以拿到数组的 index===>暂时未解为什么非要这样写，才可以得到下标或者 key
json--->(v,$key,$index) in json;可以获取到 json 的 key 以及下标
object--->(item in obj)-->可以在

解决重复添加重复数据的问题：
track-by="索引"========================>存在重复数据最好加此属性==============可以提高循环的性能

事件：
angular:ng-click

vue:@click 或者 v-on:{click="方法名" 或者 mouseover,mousedown,dbclick,blur 获取焦点事件等}

事件的简写:v-on<==>@

事件的对象:@click="方法名($event)";//vue 固定事件的对象参数必定为$event，如果为其他的，请注意参数顺序
事件的冒泡：阻止冒泡====》event.cancelBubble=true;或者@click.stop="方法名"-》stop 本身就是一个阻止冒泡的方法===》》》推荐使用 vue 自带的
事件的默认行为：@contextmenu="方法名"->鼠标左键不可以使用，使用鼠标右键，但是会带有默认的菜单；使用 event.preventDefault();阻止鼠标默认行为
阻止默认行为的方法：
1.event.preventDefault();
2.@contextmenu.prevent="click 调用的方法名"====>推荐使用

键盘类的事件：
@keydown $event ev.keyCode(按下的键盘值)
@keyup
键盘具体到每一个按键的写法：@keydown/keyup.+(按键的 keycode)

常用键值：keyCode==13;回车事件
vue 简写的回车方法：@keydown.enter="方法名"；/@keydown.13="方法名"；/@keyup.enter="方法名"；/@keyup.13="方法名"；
键盘上下左右键：@keydown/keyup.left/right/up/down/delete 常用的方法

属性与指令的区别：
指令是扩展标签的属性

属性：
v-bind:是用来绑定属性
v-bind 的简写：===》:属性名
特殊属性：class 和 style

:class:几种写法
:class 或者 v-bind:class="[classname1，classname2，classname3]";//classname 是数据
:class="{classname:true,classname2:false}";
:class="{classname:data1,classname2:data2}";//data 是数据，对应的是 boolean
:class="json";//数据中的 json 对象

style 的几种用法：
:style=[c,b];//data==>c:{color:red};==>数据一定是 json 各式的
注意：复合样式的时候，一定要采用驼峰命名规则
:style="a";==>a 是数据，其格式一定是 json 格式的

模板：
数据更新模板也更新

(双向数据绑定)
{{dataMsg}}
{{*dataMsg}}==》只绑定一次----------只存在 vue1.0
{{{msg}}}==>可以编译 html 标签----------只存在 vue1.0

过滤器：
uppercase->转大写
lowercase->转小写
debounce 参数（延时时间可有可无）---->延时 =========最好配合键盘事件
currency 货币
capitalize
json

数据配合过滤器：

limitBy n==========限制 n 个
limitBy n m==========>限制 n 个从 m 开始，注意 m 是下标
limitBy n arr.length-m;限制 n 个后面 m 条数据

filterBy n==>过滤得到包含 n 的数据

orderBy m======>排序 根据 m 排序
orderBy -1==========》倒序
orderBy 1==========》升序

vue2.0：版本中以去除内置过滤器 需要自定义
{{数据 | 过滤器 '参数'}}

自定义过滤器：
Vue.filter(过滤器的名字,function(过滤值，参数 1，参数 2，...){});

过滤器一般是：
数据到视图；即 model---- 过滤 --------view

v-model 时用过滤器，控制台会有 two-way 错误，也就是提醒我们使用双向数据过滤；具体写法如下：
双向数据过滤:model-view;view-model

```
Vue.filter(过滤器名称，{
read:function(){},//model-view
write:function(){}//view-model
});
```

自定义指令：---定制自己所需要的指令;必须以 v-开头
Vue.directive(自定义指令名称，function(){});
如果指令名称是 v-开头的时候

自定义元素指令：（用处不大）

```
Vue.elementDirective('zns-el',{
        bind:function(){
            this.el.style.background='red';
        }
    });
```

自定义键盘信息：
Vue.directive('on').keyCodes.ctrl=17;//自定义 ctrl 的 keycode==17
Vue.directive('on').keyCodes.myenter=13;//自定义 myenter 的键，code 值为 13,调用回车的 keyCode

---

交互：
与后台交互必须使用 vue-resource.js

get:
this.$http.get(url,option).then(function(){//回调成功},function(){//回调失败});

post (url,data,{emulateJSON:true})
jsonp
:
(url,data,{jsonp:"cb"//回调的方法名，默认是 callback})

```
//$http 的另外一种写法
this.$http({
    url:'',
    data:{},
    method:"get/post,jsonp",
```

```
    jsonp:"cb"//jsonp 的毁掉函数
});
```

---

vue 的生命周期
生命周期提供的钩子函数：
created--->实例已经被创建
beforeCompile---->编译之前
compiled:----->编译之后
ready------->插入到文档中

销毁
beforeDestroy:------>销毁之前
destroyed：--------------》销毁之后

Vue 销毁的方法：$destroy()--->会执行 beforeDestroy 和 destroyed

网速慢的时候，用户会看到花括号也就是数据模板：
用 v-cloak 解决；
在 style 中：[v-cloak]{display:none}
v-cloak:一般会用到大的段落中--------》等价指令 v-text:自身也会防止闪烁
vue1.0 编译 html:
v-html 和{{{数据}}}；

防止编译过程中闪烁问题：

v-cloak;v-text;v-html;

---

ng:$scope.$watch
计算属性的使用：

```
computed:{//计算属性
                b:function(){//默认使用的是 get
                    return this.a*2;
                }
         }
```

```
computed:{
        c:{
           get:function(){
               return this.a*2;
           },
           set:function(val){
              this.a=val;
           }
        }
}
```

data 和 computed 区别：
computed：里面可以放置业务代码；需要 return；
二者都可以写属性；

---

vue 实例中常用的方法：

```
var vm=new Vue({});
```

vm.$el:实例化的 dom 元素
vm.$data:vm 自身的数据对象
vm.$mount('dom 对象');//手动挂载 vue 程序
vm.$options==========获取自定义的属性和方法
vm.$destroy();==========销毁
vm.$log();//获取当前数据状态

vue 自定义属性和方法：如何访问
vm.$options.自定义属性名称

显示隐藏：
angular：ng-show/ng-hide
vue:v-show/v-hide

数据监听：
vm= new Vue({});
vm.$el/$mount/$options/...

vm.$watch(name,回调函数);//name 被监听的对象 --------------浅度监视

vm.$watch(name,回调函数，{deep:true});//name 被监听的对象 --------------深度监视

---
vue--->过渡(动画)
本质是 css3:transition，animate

通过 Vue.js 的过渡系统，可以在元素从 DOM 中插入或移除时自动应用过渡效果。Vue.js 会在适当的时机为你触发 CSS 过渡或动画，你也可以提供相应的 JavaScript 钩子函数在过渡过程中执行自定义的 DOM 操作。

为了应用过渡效果，需要在目标元素上使用 transition 特性：

transition 特性可以与下面资源一起用：

v-if
v-show
v-for （只为插入和删除触发）
动态组件 （介绍见组件）
在组件的根节点上，并且被 Vue 实例 DOM 方法（如 vm.$appendTo(el)）触发。
当插入或删除带有过渡的元素时，Vue 将：

尝试以 ID "my-transition" 查找 JavaScript 过渡钩子对象——通过 Vue.transition(id, hooks) 或 transitions 选项注册。如果找到了，将在过渡的不同阶段调用相应的钩子。

自动嗅探目标元素是否有 CSS 过渡或动画，并在合适时添加/删除 CSS 类名。

如果没有找到 JavaScript 钩子并且也没有检测到 CSS 过渡/动画，DOM 操作（插入/删除）在下一帧中立即执行。
****
过渡的 CSS 类名

类名的添加和切换取决于 transition 特性的值。比如 transition="fade"，会有三个 CSS 类名：

.fade-transition 始终保留在元素上。

.fade-enter 定义进入过渡的开始状态。只应用一帧然后立即删除。

.fade-leave 定义离开过渡的结束状态。在离开过渡开始时生效，在它结束后删除。
如果 transition 特性没有值，类名默认是 .v-transition, .v-enter 和 .v-leave。

****
自定义过渡类名

我们可以在过渡的 JavaScript 定义中声明自定义的 CSS 过渡类名。这些自定义类名会覆盖默认的类名。当需要和第三方的 CSS 动画库，比如 Animate.css 配合时会非常有用：

Watch me bounce
```
Vue.transition('bounce', {
  enterClass: 'bounceInLeft',
  leaveClass: 'bounceOutRight'
```

})

---

bower**********
bower----------
bower version ----------查看版本
bower info 包名----------查看包的具体版本信息
bower install 包名#版本号----------安装包的具体某个版本
bower uninstall 包名----------卸载包

组件：
组件：----------》一个大对象

全局注册组件
1.注册：用 Vue.extend() 创建一个组件构造器
var myComponent=Vue.extend({'template':,...//选项});
2.把这个构造器用作组件，需要用 Vue.component(tag, constructor) 注册 ：
// 全局注册组件，tag 为 my-component
Vue.component('my-component', MyComponent)

**********组件里面放数据
data 必须是函数形式，返回值必|须是对象（json）形式

局部组件：
第一种写法：
var myComponent=Vue.extend({'template':,...//选项});
new Vue({
el:;
components:{
'组件 tag':myComponent
}
})

第二种写发：

```
new Vue({
            el: '#box',
            components:{
                '组件名称':{//选项
                }
            }
        })
```

第三种写法：

```
  var myComponent=Vue.extend({//选项
     });
          new Vue({
              el: '#box',
              components:{
                  '组件 tag':myComponent
              }
          })
```

模板：
1.template:'直接写一个标签'
2.利用 script;

```
script type="x-template" id="tem">
    <h3 @click="change">{{msg}}</h3>
    <ul>
        <li>232</li>
```

```
        <li>22222222222</li>
        <li>333333333333</li>
    </ul>
</script>
```

组件属性 template:id;
第三种形式

```
<template id="tem">
    <h2>我是第二种形式的模板</h2>
    <h3 @click="change">{{msg}}</h3>
    <ul>
        <li>232</li>
        <li>22222222222</li>
        <li>333333333333</li>
    </ul>
</template>
```

组件属性 template:id;

动态组件：
:is：判断是谁
<component :is="组件名称"></component>

父子组件：

```
'aa':{//付组件
                template:'<h2>我是父类组件</h2><bb></bb>',//调用子组件
                components:{//定义子组件
                    'bb':{
                        template:'<h2>我是子组件</h2>'
                    }
                }
            }
```

**********vue 默认情况下,子组件没有办法访问父组件的数据
**********组件间的数据传递：
1.子组件获取父组件的数据 data:
**********（注意':'-》冒号是属性绑定）
在调用子组件：
<子组件名称 :属性名称='父类的某个数据'></子组件名称>
子组件内部：

```
第一种写法：
props:['属性 1','属性 2'];
第二种写法：
props:{
    '属性 1':属性 1 绑定之的类型,
    '属性 2':数值类型
}
```

2.父级获取子级的数据：
****
子组件把数据发送到父组件中：
vm.$emit('事件名称', 传递的数值); //推荐使用
父组件中：在子组件的标签上使用@或者 v-on:事件名称=‘方法名称’//此方法名称需要在父组件中定义；事件名称于子组件中传递的事件名称保持一致

总结：
子组件向父组件传递数据：
vm.$emit('事件名称', 数据); -----等价----------vm.$dispatch('事件名称', 数据);
父组件向下广播数据：
vm.$broadcast('事件名称',数据)；广播后的数据，放在 events:{}

在 vue2.0 中$dispatch，$broadcast 已经废弃

---

slot：占位

多个 slot 对应相应的表签：每个标签加上 slot="slot 名称"，slot 标签加上 name 属性

---

vue-单页面应用
vue-resource-----交互$http
vue-router:----------路由(根据对应的 url，映射到相应的页面)

vue-router：
view---------->a 标签使用 v-link="{path:'跳的 url'}"

展示：router-view

```
js:
//准备一个 root 组件
    var App=Vue.extend();

    //2.home.news 的组件
    var Home=Vue.extend({
        template:'<h3>我是主页</h3>'
    });

    var News=Vue.extend({
        template:'<h3>我是新闻页</h3>'
    });
    //3.准备路由
    var router=new VueRouter();
    //4.关联，类似与 angular 的 config
    router.map({
        'home':{
            component:Home
        },
        'news':{
            component:News
        }
    });
    //5.启动路由
    router.start(App,'#box');
```

跳转：
router.redirect();

路由之间的嵌套：（多层路由）

使用 subRoutes;

```
router.map({
        'home':{//一级路由
            component:Home,
            subRoutes:{
                '二级路由名称':{//选项信息}
            }
        },
        'news':{
            component:News
        }
    });
```

eg:
主页：home
    登陆：home/login
    注册:home/register

新闻：news

路由的其他信息：
路由中传参数：路由地址/:id====>传参数 id 的值
  $route.params =》获取当前路由传递的参数；获取到的是 obj；需要用 json 过滤器解析成{"id":value}'

  $route.path============>获取当前路由的路径

  {{$route.query | json}}==============>当前地址栏的数据 类似 get 请求后坠的数据

---

vue-loader:基于 webpack
    .vue 文件：==================>叫做组件
        放置的是 vue 组件代码包含三部分
        1.
        <template>
            html 代码
        </template>
        2.
        <style>
            css
        </style>
        3.
        <script>
            js
        </script>

****

webpack 的目录结构：
webpack：每个模块分开写，最后打包在一起

简单目录：
    |-index.html
    |-main.js----------------------入口文件
    |App.vue   vue 文件
    |-package.json   工程文件（项目依赖，名称，配置）
        npm init --yes(会自动生成)package.json 文件==================注意里面不可以有任何的注释
    |-webpack.config.js webpack 配置文件
****
package.json:
"scripts": {
    "dev": "webpack --inline --hot --port 端口号（如果端口占用的时候 用）"//--inline 实时刷新，hot 热载
  },
##########
webpack 准备工作：
npm 换成 cnpm 也可以

下载 webpack------npm install webpack --save-dev
                  npm install webpack-dev-server --save-dev

App.vue 变成正常的代码 ===============使用 vue-loader

vue-html-loader:加载 html,
css-loader:css
vue-style-loader:css
vue-hot-reload-api:js@1.3.2

babel:
babel-loader
babel-core,
babel-plugin-transform-runtime
babel-preset-es2015
babel-runtime

最最核心部分：
vue:
vue
最后运行项目
    npm run dev
            ------package.json
                  ----'script':{
                    'dev':'webpack-dev-server --inline --hot --port 端口号',
                    "build":"webpack -p"//打包并压缩，这样就不用 npm run dev;直接打开 index.html 就可以了
                  }

运行时的一个错误信息：
    EADDRINUSE============>端口号被占用

es6：模块化开发
    导出模块： export default{}

    引入模块：import 使用名称 form 地址

****
vue-router 配合 vue-loader 使用：
    1.下载 vue-router 模块
        npm install vue-router@0.7.13 --save-dev (配合 vue1.0 使用；因为我没有配置 cnpm 所以我就用 npm)
    2.在入口文件引入 vur-router 模块
    3.Vue.use(VueRouter);//使用任何常见都要使用 use();即安装 Vue.js 插件
    4.配置路由
        var router=new VueRouter();
        router.map({//配置路由});
    5.开启路由
        router.start(主模块，'包裹的元素');
        注意一下这个错误：
        //bundle.js:9631 [Vue warn]: Attribute "id" is ignored on component because the component is a fragment instance: http://vuejs.org/guide/components.html#Fragment-Instance
        如何屏蔽这个错误：
        在被包裹的主模块中，加上包裹的元素就 ok，原因====》因为主模块中是一个虚拟的标签，要告诉主模块中所有的标签对应的是谁；即：主模块中是一个虚构的一个元素，需要有个整体的父集，然后这个父集对应的是被包裹的元素；

        ---注意：未用路由的时候，是把组件写到 body,或者父集元素里面
                用路由：==在 index.html 中需要有一个最外层的元素，来包裹里面的组件；除此之外，主模块里面也需要一个根元素来包裹里面所有的标签。注意此标签要和 index.html 中的标签保持一直

上线是 npm run build===>build 实际上是执行了 webpack -p

****
脚手架：vue-cli===>vue 脚手架
    脚手架的作用----帮我们提供好基本的项目路径
    vue-cli 集成了很多项目模板：(如下)
        simple
        webpack 非常重要
            Eslint 检测代码的规范
        webpack-simple

        2，3 对比：webpack 比 webpack-simple 对了一个代码检查和单元测试
        browserify
        browserify-simple
    vue-cli：基本使用流程
    1.npm install vue-cli -g   安装 vue 命令环境
        验证是否安装成功   vue --version 查看是否有 vue 的版本号
    2.生成项目模板
        vue init 模板名称 本地文件夹的名称
    3.进入生成目录里面
        cd 目录名称
        npm install 包名
    4.npm run dev

注意：vue2.0:v-bind:是用来绑定属性

---

vue 调试工具
vue-devtools---->调试工具
谷歌浏览器如何安装调试工具：
github:搜索 vue-devtools-》然后打开：---》找到 Get it on the Chrome Web Store.然后点击获取