## 事件深入:

简写的:

@click="" 推荐

### 事件对象:

@click="show($event)"

### 事件冒泡:

阻止冒泡:

a). ev.cancelBubble=true;

b). @click.stop 推荐

### 默认行为(默认事件):

阻止默认行为:

a). ev.preventDefault();

b). @contextmenu.prevent 推荐

### 键盘:

@keydown $event ev.keyCode

@keyup

常用键:

回车

a). @keyup.13

b). @keyup.enter

上、下、左、右

@keyup/keydown.left

@keyup/keydown.right

@keyup/keydown.up

@keyup/keydown.down

.....

## 模板:

{{msg}} 数据更新模板变化

{{\*msg}} 数据只绑定一次

{{{msg}}} HTML转意输出

## 交互:

$http （ajax）

如果vue想做交互

引入: vue-resouce

### get:

获取一个普通文本数据:

this.$http.get('aa.txt').then(function(res){

alert(res.data);

},function(res){

alert(res.status);

});

给服务发送数据:√

this.$http.get('get.php',{

a:1,

b:2

}).then(function(res){

alert(res.data);

},function(res){

alert(res.status);

});

### post:

this.$http.post('post.php',{

a:1,

b:20

},{

emulateJSON:true

}).then(function(res){

alert(res.data);

},function(res){

alert(res.status);

});

### jsonp:

https://sug.so.360.cn/suggest?callback=suggest\_so&word=a

https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=a&cb=jshow

this.$http.jsonp('https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su',{

wd:'a'

},{

jsonp:'cb' //callback名字，默认名字就是"callback"

}).then(function(res){

alert(res.data.s);

},function(res){

alert(res.status);

});

### [v-cloak](https://cn.vuejs.org/v2/api/#v-cloak)

**不需要表达式**

**用法**：

这个指令保持在元素上直到关联实例结束编译。和 CSS 规则如 [v-cloak] { display: none } 一起用时，这个指令可以隐藏未编译的 Mustache 标签直到实例准备完毕。

**示例**：

|  |
| --- |
| [v-cloak] {  display: none; } |

|  |
| --- |
| <div v-cloak>  {{ message }} </div> |

不会显示，直到编译结束。

防闪烁---指令：

<span>{{msg}}</span> -> v-text

{{{msg}}}(2.0废除) -> v-html

## vue生命周期:

### 钩子函数:

created -> 实例已经创建 √

beforeCompile -> 编译之前

compiled -> 编译之后

ready -> 插入到文档中 √

beforeDestroy -> 销毁之前

destroyed -> 销毁之后

## 计算属性

computed:{

b:function(){ //默认调用get

return 值

}

}

--------------------------

computed:{

b:{

get:获取值，并返回值

set:设置值

}

}

computed:里面可以放置一些业务逻辑代码，一定记得return

## vue实例简单方法：

vm.$el ---> 元素

vm.$data -----> data

vm.$mount ----->手动挂载vue程序

vm.$mount('#box');

vm.$options -> 获取自定义属性

var vm=new Vue({

aa:11, //自定义属性,

show:function(){

alert(1);

},

data:{

a:1

}

}).$mount('#box');

vm.$options.show();

vm.$destroy -> 销毁对象

vm.$log(); -> 查看现在数据的状态

## 循环：

v-for="value in data"

会有重复数据？

track-by='索引' 提高循环性能

track-by='$index/uid'

## 过滤器:

### vue提供过滤器:

capitalize uppercase currency....

debounce 配合事件，延迟执行

### 数据配合使用过滤器:

limitBy 限制几个

limitBy 参数(取几个)

limitBy 取几个 从哪开始

filterBy 过滤数据

filterBy ‘谁’

orderBy 排序

orderBy 谁 1/-1

1 -> 正序

2 -> 倒序

自定义过滤器: model ->过滤 -> view

Vue.filter(name,function(input){

});

时间转化器

过滤html标记

### 双向过滤器:\*

Vue.filter('filterHtml',{

read:function(input){ //model-view

return input.replace(/<[^<+]>/g,'');

},

write:function(val){ //view -> model

return val;

}

});

数据 -> 视图

model -> view

view -> model

## 监听数据变化:

vm.$el/$mount/$options/....

vm.$watch(name,fnCb); //浅度

vm.$watch(name,fnCb,{deep:true}); //深度监视

## vue组件:

组件: 一个大对象

### 定义一个组件:

#### 1. 全局组件

var Aaa=Vue.extend({

template:'<h3>我是标题3</h3>'

});

Vue.component('aaa',Aaa);

\*组件里面放数据:

data必须是函数的形式，函数必须返回一个对象(json)

var Aaa=Vue.extend({

      template:'<h3>{{msg}}</h3>',

      data(){

        return {

          msg:'ddddd'

        }

      }

    });

#### 2. 局部组件

放到某个组件内部

var vm=new Vue({

el:'#box',

data:{

bSign:true

},

components:{ //局部组件

aaa:Aaa

}

});

--------------------------------------

另一种编写方式:

Vue.component('my-aaa',{

template:'<strong>好</strong>'

});

var vm=new Vue({

el:'#box',

components:{

'my-aaa':{

template:'<h2>标题2</h2>'

}

}

});

### 动态组件:

<component :is="组件名称"></component>

### **组件数据传递**

#### 1. 子组件就想获取父组件data

在调用子组件：

<bbb :m="数据"></bbb>

子组件之内:

props:['m','myMsg']

props:{

'm':String,

'myMsg':Number

}

#### 2. 父级获取子级数据

\*子组件把自己的数据，发送到父级

vm.$emit(事件名,数据);

v-on: @

--------------------------------------------

vm.$dispatch(事件名,数据) 子级向父级发送数据

vm.$broadcast(事件名,数据) 父级向子级广播数据

配合: event:{}

在vue2.0里面已经，报废了