绑定

- 属性：class,id,value :::v-bind:src="data1"
- 推荐，简写：:src 效果能出来，但会报404
- v-bind:class="data1" :class

```
<strong :class="[color,color2]">测试……</strong>
```

- **测试……** color:false
- color:true;color2:false 这个可以用json 来替换
- v-bind:style="json" :style
- a:{color:red} :style="a"

==========

模版

```
    <script>
        window.onload=function(){
            new Vue({
                el:'#box',
                data:{
                    msg:'welcome vue',
                    msg2:12,
                    msg3:true,
                    arr:['apple','banana','orange','pear'],
                    json:{a:'apple',b:'banana',c:'orange'}
                }
            });
        };
    </script>
</head>
<body  >
        <div class="container"  id="box" >
                <input type="text" v-model="msg">
                <input type="text" v-model="msg">
                  <br>
                {{msg}}
                <br>
                {{msg2}}
                <br>
                {{msg3}}
```

```
                <br>
                {{arr}}
                <br>
                {{json}}
        </div>
</body>
```

- {{msg}} 数据更新模版变化
- {{msg}} 数据一次
- 2.x 对应于：v-once This will never change: {{ msg }}
- {{msg}} 自动转义

> 在网站中动态渲染任意的 HTML 是非常危险的，因为这很容易导致网站受到 XSS 攻击。请只对可信内容使用 HTML 插值，绝对不要对用户提供的内容使用 HTML 插值。

- 2.x 对应于
  v-html

过滤

- {{msg | capitalize }}
- {{msg | filter | filter}} 多个过滤器
- {{msg | currency }} currency：钱 与uppercase v2.x 没有

==========

交互

- $http (ajax)
- get: 获取一个普通文本数据

```
          get:function(){
                      //window.alert("hello");
                      this.$http.get('aa.txt').then(function(res){
                           console.log("true");
                          alert(res.data);
                      },function(res){
                           console.log("false");
                           alert(res.status);
                           console.log(res);
                      })
                  }
```

- post :

```
post:function(){
                    this.$http.post('test.php',{a:1},
{emlateJSON:true}).then(function(res){
                            console.log("true");
                           console.log(res.data);
                           //console.log(res.data.a); //a
                           this.mydata.push(res.data.a)
                       },function(res){
                            console.log("false");
                            alert(res.status);
                            console.log(res);
                       })
                },
```

- jsonp: https://cdn.weather.hao.360.cn/sed_api_weather_info.php?code=101240701&param=pm25&v=1&app=hao360&_jsonp=**jsonp4**&t=2537151
- https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=a&json=1&p=3&sid=1428_21091_20697_22158&req=2&bs=vue%20%24http%20jsonp%20callback%20is%20not%20defined&pbs=vue%20%24http%20jsonp%20callback%20is%20not%20defined&csor=1&cb=jQuery110206764275602433492_1522283284322&_=1522283284394
- 遇到的错误：vue $http jsonp callback is not defined

```
 this.$http.jsonp('https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su',
{wd:this.t1},{jsonp:"cb"}).then((response) => {
                            console.log("true");
                            //把它的值存起来
                            this.history=this.t1;
                            console.log(response.data.s);
                            var arr=response.data.s;
                            for(j = 0; j < arr.length; j++) {
                                this.mydata.push(arr[j]);
                                console.log(arr[j])
                            }
                       }, (response) => {
                            console.log(response.data);
                                alert(response.status);
                                console.log(response);
                       });
```

其它用法：

```
this.$http({
                            url:'url',
                            data:'给后台提交的数据',
                            method:"",
                            jsonp:'cb'  //cnName
```

```
                        });
```

- 动态插入javascript
- document.write("

vue 生存周期

钩子函数： created -> 实例已经创建 beforeCompile -> 编译之前 v2.x 废除 使用 created 钩子函数替代 compiled 替换 -> 编译之后， 使用 mounted 钩子函数替代。 ready -> 插入到文档

destroyed -> 销毁

如 mounted、updated 和 destroyed

- v-cloak 防止闪烁，比较大的段落 {{}}
- v-text="msg" == {{msg}} -->回忆一下 v-html
- 代码

```
    <script>
     window.onload=function(){
         //创建实例
       var c=new Vue({
              el:"#box",
              data:{
                  msg:"xie",
                  a:1,
                  //b:2
              },
              computed:{
                   b:function(){
                       //业务逻辑
                       return this.a;
                   },
                   c:{
                       get:function() {

                       },
                       set:function(res){
                               this.a=res;
                       }
                   }
              },
              methods:{
                  get:function(){
                     window.alert("hello");

                  }
              },
```

```
            created: function() {
                console.log("编译之前");
            },
            mounted:function(){
                console.log('编译之后');

                this.$nextTick(function () {
                 // 代码保证 this.$el 在 document 中
                 console.log("ready");
             })
            },
            //2.x 已经废除  以上替代
            ready:function(){
                console.log('ready^');

            },
            destroyed:function(){
                console.log("销毁")
            },
            aa:11
    });
      document.onclick=function(){
         // c.$destroy() ;//显示 销毁
         c.a=110;
      }

      //小元素:
      console.log(c.$el);
      c.$el.style.background = 'green';
      //data
      console.log(c.$data);// data 数据对象
      //自定义属性
      //值
      console.log(c.$options.aa);
      //方法
       console.log(c.$options.show());
      //c.$destroy() ;//显示 销毁

      //现在数据的状态
      console.log(c.$log());//请使用 Vue Devtools 感受最佳debug体验。

    }
  </script>
</head>
<body>
  <div class="container" id="box" v-cloak>
     <input type="button" v-bind:value="msg" @click="get()">

     <span v-text="msg"></span>
     <div>a=>{{a}} <br/>b=>{{b}}
      </div>
```

- ng : $scopt.$watch
- 计算属性的使用： computed

```
computed:{
                    b:function(){
                        //业务逻辑  注意return
                        return this.a;
                    },
                    c:{
                        get:function() {

                        },
                        set:function(res){
                                this.a=res;
                        }
                    }
                },


                document.onclick=function(){
                    // c.$destroy() ;//显示 销毁
                    c.a=110;
                    }
```

自带小元素（vue实例简单方法）

- vm.$el -> 就是元素
- vm.$data -> 就是data
- 手动挂载：vm.$mount('#box'); //手动挂载
  //小元素： console.log(c.$el); c.$el.style.background = 'green'; //data console.log(c.$data);// data 数据对象 //自定义属性 //值 console.log(c.$options.aa); //方法 console.log(c.$options.show()); //c.$destroy() ;//显示 销毁

  ```
  //现在数据的状态
  console.log(c.$log());//请使用 Vue Devtools 感受最佳debug体验。
  ```

循环

- v-for="(index ,value) in data"
- 重复数据 track-by="index"
- ---2.x

过滤器

- limitBy

```
computed: {
                    // | limitBy 2    2.x代替
                     filteredItems: function () {
                         return this.arr.slice(0, 1)
                     }
                }
      });
```

- filterBy

```
<p v-for="user in users | filterBy searchQuery in 'name'">{{ user.name }}</p>
在 computed 属性中使用js内置方法 .filter method:
<p v-for="user in filteredUsers">{{ user.name }}</p>
computed: {
  filteredUsers: function () {
    var self = this
    return self.users.filter(function (user) {
      return user.name.indexOf(self.searchQuery) !== -1
    })
  }
}
```

- orderBy

```
 orderedUsers: function () {
                         return _.orderBy(this.arr, -1) //'name'  1:倒序  -1：顺
序
                     }
```

- 自定义过滤器

```
        //自定义过滤器
          Vue.filter('toDou',function(input){
              alert(input);
          })
```