

Vue.js

一、Vue.js 简介

Vue.js (读音 /vjuː/, 类似于 **view**) 是一套构建用户界面的**渐进式框架**。与其他重量级框架不同的是, Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层, 它不仅易于上手, 还便于与第三方库或既有项目整合。

MVVM 只关心视图和数据的交互操作

- 作者: 尤雨溪

二、起步

1、下载vue.js 的支持库

```
npm install vue
```

 下载核心库 vue.js vue.min.js

2、Hello world

- JS+JSON 实现 所有功能
- Vue中存在版本差异
 - vue 1.x : 可以对页面中的 body 标签进行 容器指定
 - vue 2.x : 不允许指定 body 标签为 容器, 只能是 body 中的其它标签

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <!--
9     阻止了 浏览对vue 的调试
10    vue 的调试信息显示
11    -->
12   <!-- <script src="../js/vue.min.js"></script> -->
```

```

13     <title>Vue hello world</title>
14 </head>
15 <body>
16     <!--
17         1、在页面中提供一个容器 ， 用于为当前框架展示数据
18         2、通过js 代码实现 vue 初始化 ， 完成容器和vue之间的关联
19     -->
20     <div id="app">
21         <!-- 插值表达式 -->
22         {{hello}}
23     </div>
24
25     <script>
26         // 全局配置项的配置
27         Vue.config.devtools = false;
28         Vue.config.productionTip = false;
29         Vue.config.silent = true;
30
31         // 实例化 new (初始化) vue 对象
32         // new Vue(opt:Object);
33         new Vue({
34             // 通过css的基础选择器语法 完成 元素和Vue对象的关联
35             el: "#app",
36             // 用于存储和页面间的通信数据
37             data: {
38                 hello: "hello Vue 2333"
39             }
40         });
41     </script>
42 </body>
43 </html>

```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6 initial-scale=1.0">
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8     <title>实例挂载</title>
9     <script src="../js/vue.js"></script>
10    <script>

```

```

10     window.onload = function(){
11         // 返回值 new •Vue创建的对象
12         var vm = new Vue({
13             data:{
14                 msg:"挂载显示消息"
15             }
16         });
17
18         // vm ==> viewMode ==> 对于mvvm框架创建的对象，一
    般使用vm进行表示
19         // 挂载 元素对象 （容器）
20         vm.$mount("#app");
21
22         new Vue({
23             data:{
24                 name:"itany"
25             },
26             template:"<h1>这是模版</h1>"
27         }).$mount("#itany")
28
29     };
30
31     </script>
32 </head>
33 <body>
34     <div id="app">
35         {{msg}}
36     </div>
37     <div id="itany">
38         {{name}}
39     </div>
40 </body>
41 </html>

```

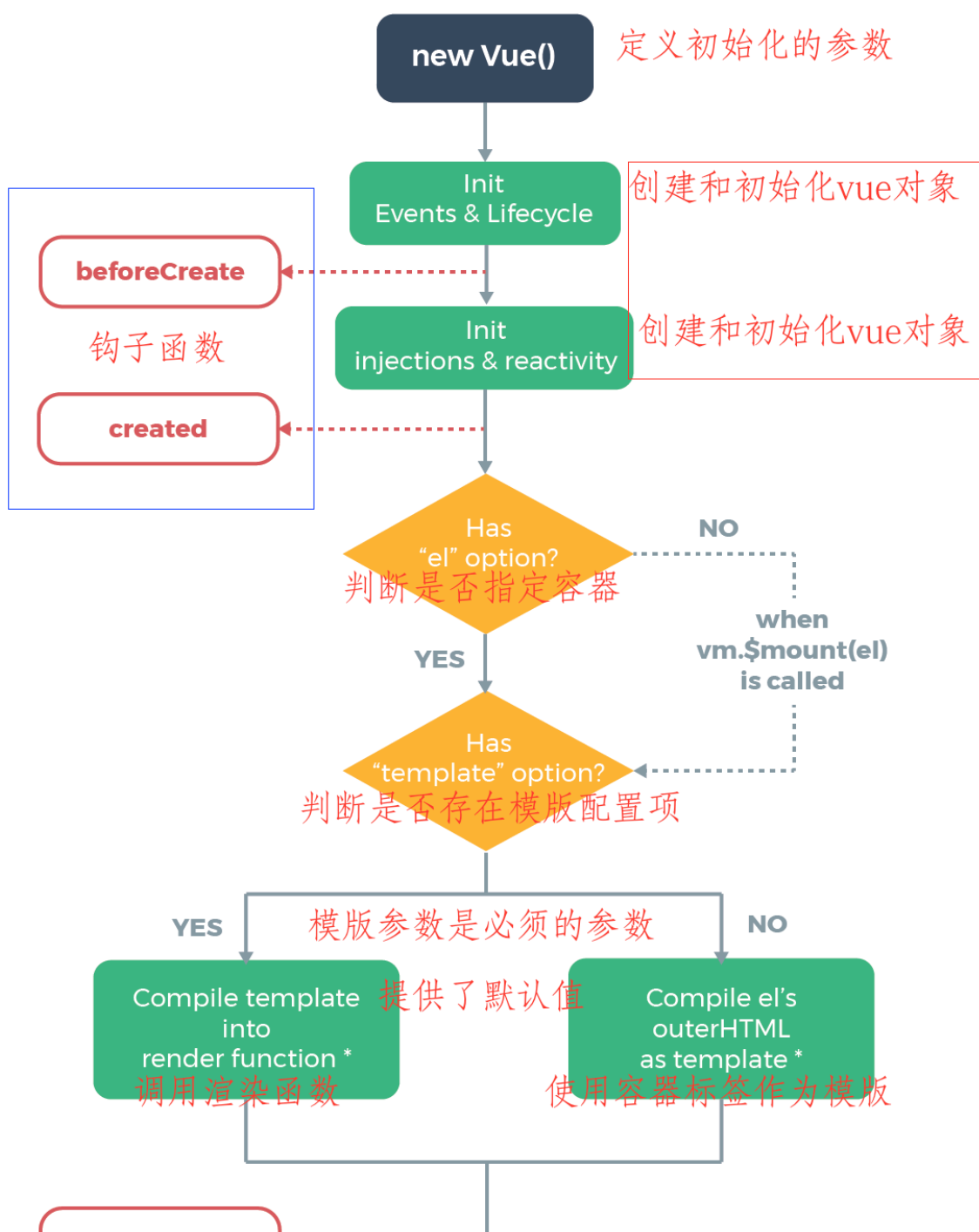
3、全局配置项

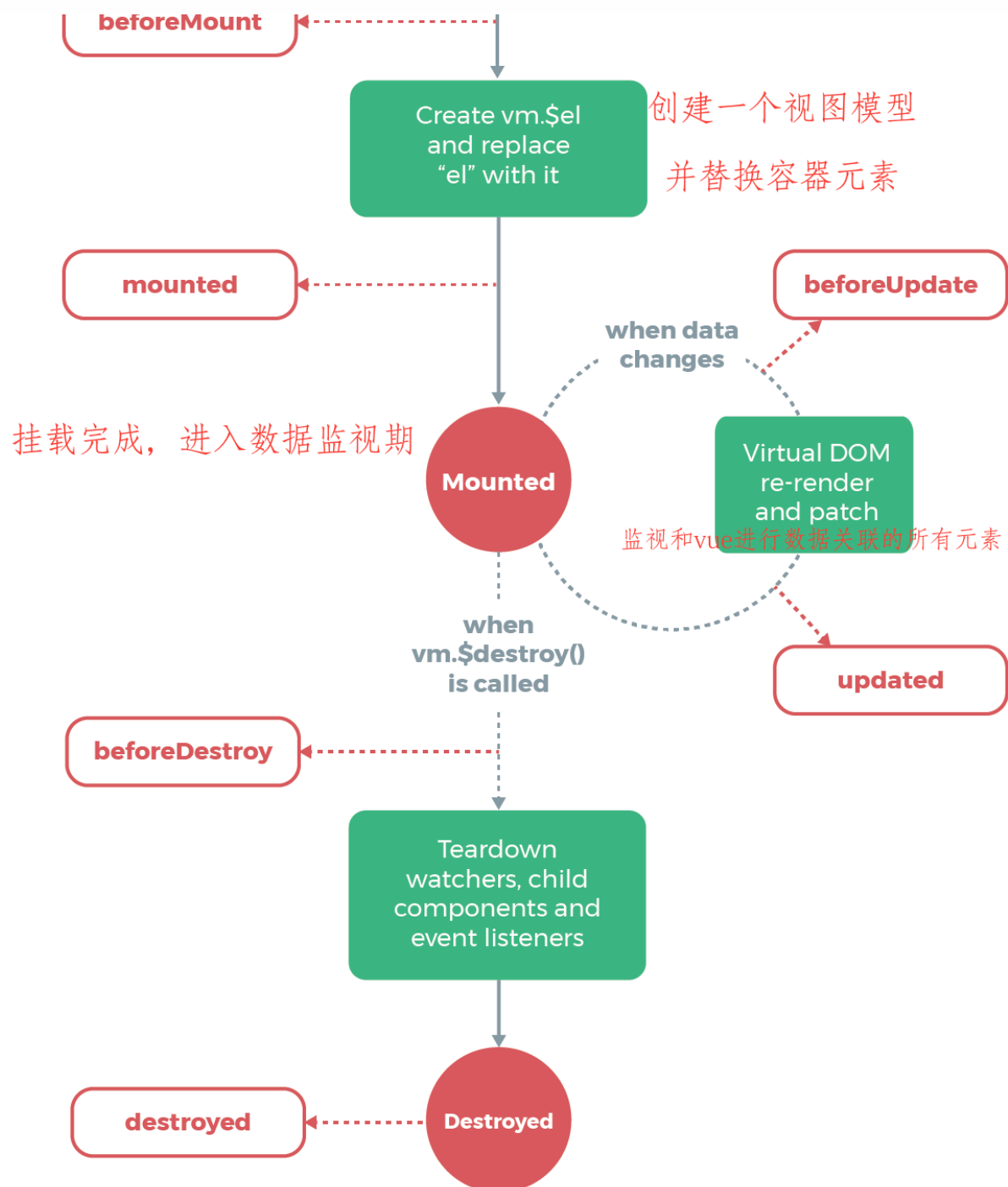
- 对整个Vue 的运行环境进行配置
 - silent 取消所有的信息和日志警告
 - optionMergeStrategies 自定义合并策略的选项。
 - devtools 设置开发工具是否可用
 - errorHandler 定义vue 页面出现错误时的 处理函数

- warnHandler 定义vue 页面出现警告时的 处理函数
- ignoredElements 须使 Vue 忽略在 Vue 之外的自定义元素
- keyCodes 设置键盘映射表 ==> 后续课程进行讲解
- performance 对浏览器的开发工具进行性能检测的
- productionTip 生产环境的提示消息

4、生命周期

- 创建到消亡的过程
 - Vue 对象的 创建到消亡的过程





* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

三、模版语法

1、插值

- 语法 `{{ }}`

1.1 普通文本的插入

```
1 | <div>{{msg}}</div>
```

1.2 HTML 标签的插入

- 直接使用 {{}} 进行带有标签的字符串写入时，类似于 JQUERY text()
- 插值表达式 不可以写入 html标签 （不能解析HTML标签） ==> 指令
- 指令后 可以直接写 简单的 JS 表达式

1.3 js 脚本

- 可以执行 简单的 js 表达式

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <script>
9     window.onload = ()=>{
10       new Vue({
11         el:"#app",
12         data:{
13           msg:"文本消息",
14           html:"<h4>标签文本</h4>"
15         }
16       });
17     }
18   </script>
19   <title>插值表达式</title>
20 </head>
21 <body>
22   <div id="app">
23     <h4>{{msg}}</h4>
24     <div>
25       {{html}}
26     </div>
27     <h4>js 表达式</h4>
28     <div>{{ 1+1 }}</div>
29     <div>{{ 10/3 }}</div>
30     <div>{{ 10%3 }}</div>
31     <div>{{ 1==1 ? "true":"false" }}</div>
32     <!-- ++ ..... 不能执行 -->
33     <!-- <div>{{ ++num }}</div> -->
34   </div>
35 </body>
36 </html>

```

2、指令

- Vue中所指的指令是 以 `v-` 开头的元素属性 ,这些特殊的属性称之为 Vue的指令

- 指令参数 在相关指令后 以 : 的方式对该指令进行参数传递, v-on:click
- 指令修饰符 修饰符定义在 指令的后面 . 方式表示, 用于做限制和判断

2.1 v-text

- 功能和 {{}} 一样 做文本写入操作
- 依赖于一个标签, 值是写在 标签中的

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <script>
9     window.onload = ()=>{
10
11       new Vue({
12         el:"#app",
13         data:{
14           msg:"文本消息"
15         }
16       });
17     }
18   </script>
19   <title>指令</title>
20 </head>
21 <body>
22   <div id="app">
23     <h4>{{msg}}</h4>
24     <h4 v-text="msg"></h4>
25   </div>
26 </body>
27 </html>
```

2.2 v-html

- 将输入html 字符串以解析的方式写入到页面


```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <script>
9     window.onload = ()=>{
10       new Vue({
11         el:"#app",
12         data:{
13           html:"<h4>标签文本</h4>",
14         }
15       });
16     }
17   </script>
18   <title>指令</title>
19 </head>
20 <body>
21   <div id="app">
22     <div>{{html}}</div>
23     <div v-html="html"></div>
24   </div>
25 </body>
26 </html>

```

2.3 v-on

- 事件绑定
- v-on 使用需要去配合参数进行使用，参数就是元素的事件，就是原始的HTML的事件名
- v-on 监听HTML 的原始DOM事件，指定的方法定义方式和原生的方法定义方式一样
- 绑定的事件一定要是在创建的Vue对象中所指定的事件，事件方法是定义在 methods 中
- 提供 简写方式 `v-on:事件` ==> `@事件`

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <script>
9     window.onload = ()=>{
10       new Vue({
11         el:"#app",
12         data:{
13           msg:"文本消息",
14         },
15         methods:{
16           show:function(){
17             this.msg = "aaaa";
18           }
19         }
20       });
21     }
22   </script>
23   <title>指令</title>
24 </head>
25 <body>
26   <div id="app">
27     <h4>{{msg}}</h4>
28
29     <input type="button" value="按钮" v-
on:click="show()">
30   </div>
31 </body>
32 </html>

```

● 增减修饰符

- 定义键盘拦截
- 阻止冒泡、默认事件

```

1 <!DOCTYPE html>
2 <html lang="en">

```

```
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <style>
8     .div1{
9       width: 300px;
10      height: 300px;
11      border: 1px solid black;
12    }
13    .div2{
14      width: 200px;
15      height: 200px;
16      border: 1px solid black;
17    }
18    .div3{
19      width: 100px;
20      height: 100px;
21      border: 1px solid black;
22    }
23
24  </style>
25  <script src="../../js/vue.js"></script>
26  <title>Document</title>
27  <!--
28    1、阻止默认行为
29    2、阻止冒泡
30
31    .stop - 调用 event.stopPropagation()。
32    .prevent - 调用 event.preventDefault()。 阻止默认事件
33    .self - 只当事件是从侦听器绑定的元素本身触发时才触发回调。
34  -->
35  <script>
36    window.onload = function(){
37      new Vue({
38        el:"#app",
39        methods:{
40          fun1(){
41            console.log("被点了");
42          },
43          fun2(){
```

```

44         console.log("fun2");
45     },
46     fun3(){
47         console.log("fun3");
48     },
49     fun4(){
50         console.log("fun4");
51     },
52     fun5(){
53         console.log("fun5");
54     }
55 }
56 });
57 }
58
59 </script>
60 </head>
61 <body>
62     <!-- a 标签的 点击事件 默认行为 就是 跳转页面 -->
63     <a href="http://www.baidu.com">百度</a>
64     <div id="app">
65         <a href="http://www.baidu.com" v-on:click.prevent >
66 百度</a>
67         <a href="http://www.baidu.com" v-
68 on:click.prevent="fun1()" >百度</a>
69         <hr>
70         <!-- 事件冒泡 -->
71         <div class="div1" v-on:click.self="fun5()">
72             <div class="div2" v-on:click="fun4()">
73                 <div class="div3" v-on:click="fun3()">
74                     <span v-on:click.stop="fun2()">测试
75 </span>
76                 </div>
77             </div>
78         </div>
79     </div>
80 </body>
81 </html>

```

2.4 v-bind

- 元素属性绑定, class style 普通属性该如何处理

- 普通属性
 - class
 - style
- `v-bind` 的简写方式 `:`属性

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6   initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <style>
9     .aa{
10       width: 100px;
11       height: 100px;
12       background-color: rebeccapurple;
13     }
14     .bb{
15       color: white;
16     }
17     .cc{
18       color: blue;
19     }
20     .fs{
21       font-size: 20px;
22     }
23   </style>
24   <script src="../js/vue.js"></script>
25   <title>v-bind 的使用</title>
26   <script>
27     window.onload = function(){
28       new Vue({
29         el:"#app",
30         data:{
31           w:"100px",
32           h:"100px",
33           imgurl:"../img/生命周期.png",
34           statu:true,
35           // aa:"aa bb",
36           divStyle:"aa",
37           colorStyle:"bb",
```

```

37         flag:true,
38         clazz:{aa:true,bb:true,cc:false},
39         style1:{color:'red'},
40         style2:{fontSize:'30px'}
41     },
42     methods:{
43         change(){
44             this.w = "200px";
45             this.h = "200px";
46             this.imgurl =
"http://www.baidu.com/img/bd_logo1.png";
47         },
48         changeColor(){
49             // this.colorStyle = "cc";
50             this.colorStyle = this.colorStyle
== "bb"? "cc":"bb";
51         },
52         changeFlag(){
53             this.flag = !this.flag;
54         }
55     }
56 });
57 }
58
59 </script>
60 </head>
61 <body>
62     <div id="app">
63         <h1>普通属性的绑定</h1>
64         <!--
65             指令后面的 "" 中指定是 vue 实例中 data 中的变量名
66             -->
67         <button type="button" @click="change()">改变大
小</button>
68         
69         <hr>
70         <!--
71             vue 中对于互斥属性 一般可以直接使用 boolean 进行状态
的选择
72             vue 对于 复选单选, 下拉列表, 存在一种特殊的 选中方式?
73             .....

```

```

74      -->
75      <input type="checkbox" :checked="statu">
76      <hr>
77      <!-- class的绑定 -->
78      <!-- <div class="aa bb"></div> -->
79      <!-- 1、变量的定义方式 -->
80      <!-- <div :class="aa"> -->
81
82      <!-- 2、数组的定义方式 -->
83      <div :class="[divStyle,colorStyle]">
84          div1
85      </div>
86      <input type="button" value="变变变"
@click="changeColor()">
87
88      <!--
89          3、JSON 定义方式      (常用)
90          key 使用 样式名
91          value  boolean
92          可以处理互斥样式
93          success error
94      -->
95      <div :class="{aa:true,bb:false,cc:true}">
96          div2
97      </div>
98      <div class="fs" :class="{aa:true,bb:flag,cc:!flag}"
@click="changeFlag()">
99          div3
100      </div>
101      <!--
102          4、变量json 的方式进行定义
103      -->
104      <div :class="clazz">
105          div4
106      </div>
107      <!-- 对于 style 的绑定 -->
108      <div style="color:red; font-size:30px" >div5</div>
109      <!--
110          key 样式名称    ==> 遵循 原生 JS 的Style操作方式
xxx-yyy-zzz ==> xxxYyyZzz
111          value 是样式的值
112      -->

```

```
113     <div :style="{color:'red',fontSize:'30px'}">
114         div6
115     </div>
116     <!--
117         可以取数组对象
118     -->
119     <div :style="[ {color:'red'}, {fontSize:'30px'} ]">
120         div7
121     </div>
122     <div :style="[style1,style2]">
123         div8
124     </div>
125 </div>
126 </body>
127 </html>
```

2.5 v-model

- 双向数据绑定，经常用在表单属性中
- .lazy 修饰符 用于定义input 框的 光标移开事件
- v-model 双向数据绑定对于表单元素的影响


```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <script>
9     window.onload = ()=>{
10       new Vue({
11         el:"#app",
12         data:{
13           str:"aaa"
14         }
15       });
16     }
17   </script>
18   <title>指令</title>
19 </head>
20 <body>
21   <div id="app">
22     <input type="text" v-model="str">
23     <input type="text" v-model="str">
24     <h4>{{str}}</h4>
25   </div>
26 </body>
27 </html>

```

2.6 v-pre

- 完整显示 == 让当前标签中的 {{}} 不去做解析，作为文本在页面中显示

2.7 v-cloak

- 解决页面闪烁问题
- 该属性单独使用无效，需要配合自定义CSS样式
- 需要定义 css 样式 `[v-cloak]{display: none; }`

2.8 v-once

- 只绑定一次，在Vue对象创建时 指定该属性的元素只会渲染一次，后需要Vue数据发生变化不会再影响到该标签

2.9 v-if

- 判断，根据boolean类型判断，用于决定绑定了v-if的元素是否显示
- 根据结果 选择 创建元素还是不创建元素

2.10 v-show

- 判断，根据boolean类型判断，用于决定绑定了v-show的元素是否显示
- 会直接创建DOM 元素，但根据 display: none; 方式进行元素的隐藏

2.11 v-else 和 v-else-if

- v-else 和 v-else-if 不可以单独使用，必须配合 v-if 使用

2.12 v-for

- 对绑定 v-for 的元素做循环操作

四、自定义指令

- 全局自定义：在所有的Vue 实例中都可以使用
- 局部自定义：在指定的Vue 实例中才可以使用

五、自定义过滤器

- Vue 1.x 版本中 集成了大量的内置过滤器，字符串转换、时间转换.....
- Vue 2.x 删除了所有内置过滤器 ==> 自定义内置过滤器 ==> 使用三方过滤器
- 全局自定义：在所有的Vue 实例中都可以使用
- 局部自定义：在指定的Vue 实例中才可以使用

六、计算属性

- 使用缓存的方式处理过滤器的功能
- 结果过程 类似于过滤器
- 过滤器每次调用都会执行一次运算过程，计算的结果不会缓存
- 计算属性在其依赖属性未发生变换时，运算过程只会执行一次，计算的记过会被缓存，直到依赖属性发生变换才会再次计算
- 计算属性 默认 只提供 单向数据绑定操作 ==> 只能取值 不能赋值

```

2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <title>计算属性</title>
9 </head>
10 <body>
11   <div id="app">
12     <input type="text" v-model="num">
13     <hr>
14     <h1>{{num}} > 10 : {{ num | number}}</h1>
15     <h1>{{num}} > 10 : {{ num | number}}</h1>
16     <h1>{{num}} > 10 : {{ num | number}}</h1>
17     <h1>{{num}} > 10 : {{ num | number}}</h1>
18     <hr>
19     <h1>{{num}} > 10 : {{ result }}</h1>
20     <h1>{{num}} > 10 : {{ result }}</h1>
21     <h1>{{num}} > 10 : {{ result }}</h1>
22     <h1>{{num}} > 10 : {{ result }}</h1>
23     <hr>
24     <input type="text" v-model="result">
25     <hr>
26     <input type="text" v-model="num2">
27     <input type="text" v-model="addNum">
28     <input type="text" v-model="num3"
@change="setNum2()">
29     <h1>{{num2}}:{{addNum}}</h1>
30
31   </div>
32   <script>
33     Vue.filter("number",function(data){
34       console.log("过滤器:",Math.random());
35       return data>10
36     });
37
38     new Vue({
39       el:"#app",
40       // 普通属性值
41       data:{

```

```

42         num:6,
43         num2:1,
44         num3:0
45     },
46     // 计算属性值
47     computed:{
48         // name:Function
49         // 1、计算属性所对应的function 不能使用 => 函数
50         //      使用=> 函数 不能保证 计算属性函数中的this是
    当前的 Vue 实例
51         // 2、计算属性的函数一定要有返回值
52         result:function(){ //get 方法
53             // 增加一些其他的判断获取逻辑流程
54             console.log("计算属性: ",Math.random());
55             return this.num > 10;
56         },
57         // getter 缺少 setter
58         // get set 方法 用于表示对一个参数的取值和赋值的
    两种操作
59         // addNum:function(){
60         //     return parseInt(this.num2)*2;
61         // }
62         addNum:{
63             get:function(){
64                 return parseInt(this.num2)*2;
65             },
66             // set 必须和 依赖属性进行关联
67             // set 和自己进行关联 会造成死循环
68             set:function(value){
69                 // console.log(value);
70                 // this.addNum = parseInt(value)/2;
71                 this.num2 = parseInt(value)/2;
72             }
73         },
74     },
75     methods:{
76         setNum2(){
77             this.num2 = this.num3/2;
78         }
79     }
80 });
81 </script>

```

```
82 </body>
83 </html>
```

七、过渡效果

- 提供简单的方式实现CSS动画效果
- 会和Vue实例中的数据做关联

1、基础应用

- 生命周期
 - 开始 过渡 结束
 - 开始前 开始 开始后 过渡前 过渡 过渡后 结束前 结束 结束后
 - 对于DOM元素需要添加动画，将该元素放置在 `<transition`
`>DOM</transition>`
- transition 只用于定义单元素动画
- transition-group 定义多元素动画 一定要在子元素上 绑定一个 key 属性
 - key 用于区分动画的加载元素

八、Vue 对象的实例方法

- 堆栈空间的数据存储
- 对于vue 而言，所有的实例属性都具有两种表现形式（watch除外）
 - vm.\$名称 对象调用
 - Vue.名称 全局调用
- vm.\$set 或者 Vue.set ：为新增减的属性完成 getter 和 setter 方法的定义，并完成赋值

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8     <script src="../js/vue.js"></script>
9     <title>实例属性</title>
10 </head>
```

```
10 <body>
11   <div id="app">
12     <input type="text" v-model="msg">
13     <span>基本类型: {{msg}}</span>
14     <input type="button" value="设置消息"
15     @click="setMsg()">
16     <input type="button" value="设置新消息"
17     @click="setNewMsg()">
18     <br>
19     <input type="text" v-model="user.name">
20     <span>引用类型: {{user}}</span>
21     <span></span>
22     <input type="button" value="设置用户年龄"
23     @click="setUserAge()">
24   </div>
25
26   <hr>
27   <input type="text" v-model="num">
28 </div>
29
30 <hr>
31 <input type="button" value="设置用户性别"
32 onclick="setUserSex()">
33 <input type="button" value="设置新消息"
34 onclick="setData()">
35 <script>
36
37   // let arr1 = new Array();
38   // let arr2 = new Array();
39   // let arr3 = arr2;
40   // arr2.push(1);
41   // console.log(arr3);
42   // console.log(arr1===arr2);
43   // console.log(arr2===arr3);
44
45   let vm = new Vue({
46     el: "#app",
47     data: {
48       msg: "基本类型的数据",
49       num: 10,
50       user: {
51         name: "tom"
52       }
53     }
54   })
```

```

47     },
48     methods: {
49         setMsg() {
50             this.msg = "新消息";
51         },
52         setNewMsg() {
53             // this.newMsg = "这是一条新的消息";
54             // this.$set(this.$data, "newMsg", 1);
55         },
56         setUserAge() {
57             // 想办法给age 提供 getter setter 方法
58             // 属性 set 实例属性
59             // vm.$set(target, 属性名称, 属性值)
60             // Vue.set()
61             // this.user.age = 23;
62             // Vue.set(this.user, "age", 23);
63             // vm.$set(this.user, "age", 23);
64             this.$set(this.user, "age", 23);
65         }
66     },
67     // 数据监视器
68     watch: {
69         // 监视的是已经具有 getter 和 setter 方法的属性
70         // key 取值 为 data 中定义的属性名称（一层）
71         // 监视一层
72         msg: function(newValue, oldValue) {
73             console.log(newValue, ":", oldValue);
74         },
75         num: function(newValue, oldValue) {
76             if (isNaN(newValue)) {
77                 this.num = oldValue;
78             }
79         },
80         // 实现深度监视
81         user: {
82             // handler 该监视器被调用的回调函数
83             // newValue 和 oldValue 取得 是栈中的值
84             handler: (newValue, oldValue) => {
85                 console.log(newValue === oldValue);
86                 console.log(newValue, ":", oldValue);
87             },

```

的变换

```

88         // deep 设置监视模式    默认 false
89         deep:true
90     }
91 }
92 });
93
94
95     function setUserSex(){
96         vm.$set(vm.user,"sex","男");
97     }
98
99     function setData(){
100         // vm.$set(vm)
101         // vm.$set(vm,"newMsg","消息");
102         // 指定的vue 对象所 定义过的数据
103         console.log(vm.$data);
104
105     }
106 </script>
107 </body>
108 </html>

```

九、组件定义

- 组件的全局和局部定义方式，和过滤、指令的全局和局部定义方式基本一样
- Vue 的组件就是自定义标签 `<组件名称>`
- 全局组件
 - 先定义构造器，在创建组件
 - 直接创建组件
- 局部组件
 - 先定义构造器，在创建组件
 - 直接创建组件
- 模板加载 引用
- 组件中 data 的数据定义方式


```

1 Vue.component("hello",{
2   template:"#wbs",
3   data:function(){
4     return {
5       arr:["a","b","c"]
6     }
7   }
8 });

```

- 原因：组件在项目运行过程中，可能会被创建多次，所以data要求是函数
 - 为了解决 js 中 多变量 可以实现 引用数据类型的数据共享
 - 为了隔离组件与组件间的数据独立，互不影响，通过函数的方式，让组件在每次创建时都返回一个新的对象
 - 堆栈的数据存储原理
- 动态组件
- 组件缓存

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <script src="../js/vue.js"></script>
9   <title>动态组件</title>
10 </head>
11 <body>
12   <div id="app">
13     <input type="button" :value="title"
14     @click="changePage()">
15     <!-- <login></login> -->
16     <!--
17       在组件上定义 的样式 或属性，会直接传递到 该组件的根元素
18     -->
19     <!-- <regist style="display:none"></regist> -->
20     <!--
21       根据用户的需求 选择性的渲染相关组件
22       占位符

```

```

22         v-bind:is ==> 用于指定组件
23         = 实例的变量
24         每次切换时。都会重新渲染组件
25         默认组件不缓存,一旦切换原组件直接销毁,每次显示时都会重
新创建组件
26         在Vue中如何让动态组件不被销毁
27         -->
28         <!-- <component :is="page"></component> -->
29         <!--
30             第一调用该组件时,会创建,一旦创建完成,该组件不会被销
毁,存放在内存中
31             后续调用直接从内存中读取
32             <keep-alive> ==> 保持存活
33             <component :is="page"></component>
34             </keep-alive>
35         -->
36         <keep-alive>
37             <component :is="page"></component>
38         </keep-alive>
39
40     </div>
41
42     <!--
43         Vue 1.x template 没有特别要求
44         Vue 2.x template 有且仅有一个根元素
45     -->
46     <template id="login">
47         <div>
48             <h1>登录</h1>
49             <label for="name">登录名: </label>
50             <input type="text" name="" id="name">
51             <br>
52             <label for="pwd">密码</label>
53             <input type="password" name="" id="pwd">
54             <br>
55             <input type="button" value="登录">
56         </div>
57     </template>
58
59     <template id="regist">
60         <div id="aa">
61             <h1>注册</h1>

```

```

62         <label for="name">登录名: </label>
63         <input type="text" name="" id="name">
64         <br>
65         <label for="pwd">密码</label>
66         <input type="password" name="" id="pwd">
67         <br>
68         <input type="button" value="注册">
69     </div>
70 </template>
71 <script>
72     Vue.component("login",{
73         template:"#login",
74         mounted(){
75             console.log("重现创建登录组件");
76         }
77     });
78     Vue.component("regist",{
79         template:"#regist",
80         mounted(){
81             console.log("重现创建注册组件");
82         }
83     });
84
85
86
87     new Vue({
88         el:"#app",
89         data:{
90             page:"login",
91             title:"去注册"
92         },
93         methods:{
94             changePage(){
95                 this.page =
96                 this.page=="login"? "regist": "login";
97                 this.title = this.page=="login"? "去注
98                 册": "去登录"
99             }
100         }
101     });
102 </script>

```

```
102 </body>
103 </html>
```

- 组件分发

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <script src="../js/vue.js"></script>
8   <title>动态组件</title>
9 </head>
10 <body>
11   <div id="app">
12     <hello>
13       <!-- <div>
14         <h1>2017年10月25号</h1>
15         <h2>=====</h2>
16       </div>
17       <div>
18         <h2>=====</h2>
19         <h1>tom</h1>
20       </div> -->
21       <div slot="s2">
22         <h1>2017年10月25号</h1>
23         <h2>=====</h2>
24       </div>
25       <div slot="s1">
26         <h2>=====</h2>
27         <h1>tom</h1>
28       </div>
29     </hello>
30   </div>
31
32   <!--
33     Vue 1.x template 没有特别要求
34     Vue 2.x template 有且仅有一个根元素
35   -->
36   <template id="hello">
```

```

37         <div>
38             <!-- 占位符 -->
39             <!-- <slot></slot>
40             <h1>hello</h1>
41             <slot></slot> -->
42             <slot name="s1"></slot>
43             <h1>hello</h1>
44             <slot name="s2"></slot>
45         </div>
46     </template>
47     <script>
48         Vue.component("hello",{
49             template:"#hello"
50         });
51
52
53         new Vue({
54             el:"#app"
55         });
56
57     </script>
58 </body>
59 </html>

```

十、组件的参数传递

1、父组件向子组件传递参数

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6  initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <script src="../js/vue.js"></script>
9      <title>父组件向子组件传递参数</title>
10 </head>
11 <body>
12     <div id="app">
13         <h1>父组件取值</h1>
14         <p>msg:{{msg}}</p>

```

```

14     <p>user:{{user}}</p>
15     <hr>
16     <!--
17         父组件向子组件的传值方式
18         1、依赖于指令 v-bind:自定义属性名 ==> 用在子组件标签
    上
19         2、组件中的属性 props ==> 定义在子组件实例中
20
21         vue 的官方不建议使用
22         ==> vue 2.x 概念 : 单向数据操作
23         ==> 父组件和子组件之间完成数据独立
24         ==> 子组件数据操作 不应该影响父组件的原始数据
25         ==> 因为子组件存在部分数据是由父组件提供, 这些
    数据应该由父组件维护
26         在标签上是不能绑定对象的
27
28         实际开发过程中经常使用
29
30         1、vue 是将绑定对象的地址 传递给了 子组件
31         2、通过地址 获取对应的参数
32         ? 子组件从父组件获取的对象 和父组件 中定义的对象是否是同
    一个
33         -->
34         <itany v-bind:info="msg" :user="user"></itany>
35     </div>
36
37     <template id="itany">
38         <div>
39             <h1>子组件取值</h1>
40             <p>msg:{{info}}</p>
41             <input type="text" v-model="info">
42             <p>user:{{user}}</p>
43             <input type="text" v-model="user.name">
44
45         </div>
46     </template>
47
48     <script>
49         new Vue({
50             el: "#app",
51             data: {
52                 msg: "父组件定义的数据",

```

```

53         user:{
54             name:"tom",
55             age:23
56         }
57     },
58     components:{
59         itany:{
60             template:"#itany",
61             data:function(){
62                 return {
63
64                 }
65             },
66             computed:{
67
68             },
69             // 用于 从父组件中 获取传递的参数
70             props:{
71                 info:"",
72                 user:{}
73             }
74         },
75     }
76 });
77 </script>
78 </body>
79 </html>

```

- 单向数据流：见后续笔记

2、子组件向父组件传递参数

- 事件发送 emit()

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <script src="../js/vue.js"></script>

```

```

8     <title>子组件向父组件传递参数</title>
9 </head>
10 <body>
11     <div id="app">
12         <!--
13             事件能不能绑定?
14             绑定的不是原生 DOM 的事件
15             w3c 对特定的 标签 定义并提供接口事件
16
17             自定义子组件上的 vue 事件 需要指定明确的触发时机
18
19             1、在子组件标签上以自定义事件名称的方式 绑定 父组件的 方
20             法
21             注意：因为参数不定 方法不能带括号
22             2、在子组件中可以通过 特定时机 使用 vm.$emit() 触发自定
23             义方法
24         -->
25         <itany :msg="msg" @aa="setInfo"></itany>
26         <hr>
27         <!--
28             移动端 点击事件是 tap
29         -->
30         <h1>父组件数据</h1>
31         <p>msg:{{msg}}</p>
32         <p>info:{{info}}</p>
33     </div>
34
35     <template id="itany">
36         <div>
37             <h1>子组件数据</h1>
38             <p>msg:{{msg}}</p>
39             <p>info:{{info}}</p>
40             <input type="button" value="发送数据"
41             @click="sendInfo()">
42             <input type="text" v-model="info">
43         </div>
44     </template>
45
46     <script>
47         function show(){
48             alert(1);
49         }

```



```

47     new Vue({
48         el: "#app",
49         data: {
50             msg: "父组件的消息",
51             info: "",
52             aaa: 1
53         },
54         methods: {
55             setInfo(data, a) {
56                 // alert(data + ":" + a);
57                 this.info = data;
58                 // alert(1);
59             }
60         },
61         components: {
62             itany: {
63                 template: "#itany",
64                 data: function() {
65                     return {
66                         info: "子组件消息",
67                         test: "sss"
68                     }
69                 },
70                 props: {
71                     msg: ""
72                 },
73                 methods: {
74                     sendInfo() {
75                         // alert(1);
76                         // 触发 自定义的 vue 事件
77                         // 发送==> 用于触发自定义事件的
78                         // this.$emit(name, args...);
79                         //     name 自定义的事件名
80                         //     args...    不定长数组    可以传
81                         // 递任意个数的参数
82                     }
83                 },
84                 watch: {
85                     info: function(newValue, oldValue) {
86                         // console.log(1);

```

```

87         this.$emit("aa",newValue);
88     }
89 },
90     mounted(){
91         this.$emit("aa",this.info);
92     }
93 }
94 }
95 });
96 </script>
97 </body>
98 </html>

```

3、非父子组件间的参数传递

- 中央数据总线 — 中间替代组件
- 使用一个空的 没有任何作用的 Vue 实例 作为一个临时的数据存储区
- 通过这个数据存储区进行 数据的传递和交换

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6  initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <script src="../js/vue.js"></script>
9      <title>非父子组件数据的传递</title>
10 </head>
11 <body>
12     <div id="app">
13         <hello></hello>
14         <world></world>
15     </div>
16     <template id="a">
17         <div>
18             <h1>hello 组件</h1>
19             <p>{{msg}}</p>
20             <input type="button" value="向WORLD组件发送msg"
21 @click='emitPrint()'>
22         </div>
23     </template>

```

```

22     <template id="b">
23         <div>
24             <h1>world 组件</h1>
25             <p>{{msg}}</p>
26             <input type="button" value="触发Event的自定义方法"
@click='emitPrint()'>
27         </div>
28     </template>
29     <script>
30         // 1、获取new Vue() 的实例对象
31         let Event = new Vue(); // 使用空的Vue 实例作为数据仓库
中央数据总线
32         // Event 的命名 取决于 中央数据总线使用 数据操作 技术
(自定义事件)
33         // Event.$on(name,fun);
34         Event.$on("print",function(){
35             alert(1);
36         });
37
38
39         Vue.component("hello",{
40             template:"#a",
41             data:function(){
42                 return {
43                     msg:"hello 组件消息"
44                 }
45             },
46             methods:{
47                 emitPrint(){
48                     // Event.$emit("print");
49                     Event.$emit("hello-msg",this.msg);
50                 }
51             },
52             mounted(){
53                 // console.log(Event);
54                 // vue 实例中 $on 绑定自定义事件
55                 // Event.$emit("print");
56                 // Event.$on("hello-msg",()=>{
57                     //     this.msg
58                 // });
59             }
60         });

```

```

61
62     Vue.component("world",{
63         template:"#b",
64         data:function(){
65             return {
66                 msg:""
67             }
68         },
69         methods:{
70             emitPrint(){
71                 Event.$emit("print");
72             }
73         },
74         mounted(){
75             // console.log(Event);
76             Event.$on("hello-msg",(msg)=>{
77                 this.msg = msg;
78             });
79         }
80     })
81
82     new Vue({
83         el:"#app"
84     });
85
86
87     </script>
88 </body>
89 </html>

```

4、单向数据流

- 父组件 的数据 更新 可以 实时 传递给子组件，会影响子组件的数据
- 子组件 从父组件中继承数据，不能实时的将修改结果传递给父组件

实际开发过程中 存在 一种情况 需要 打破单向数据流

- 如何打破单向数据流 （单向）
- js 使用引用类型 打破单向数据流

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>

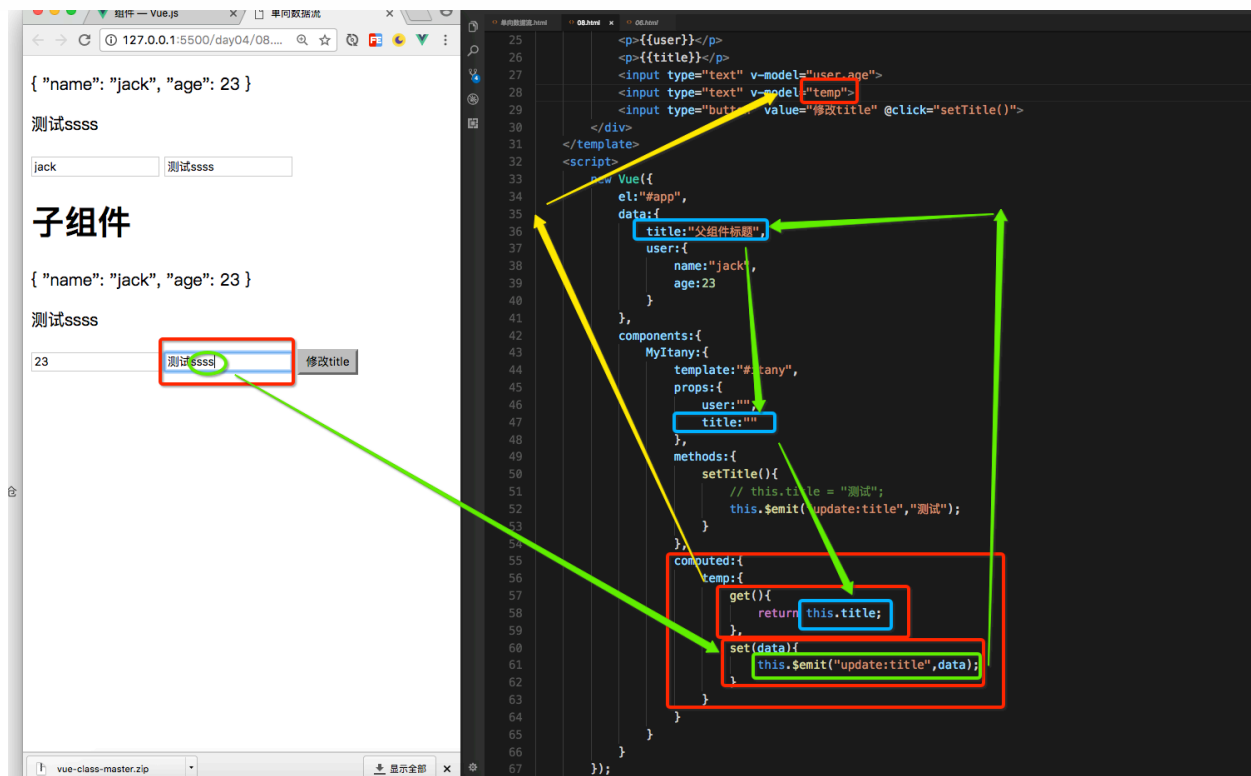
```

```

4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <script src="../js/vue.js"></script>
8     <title>单向数据流</title>
9 </head>
10 <body>
11     <div id="app">
12         <p>{{user}}</p>
13         <p>{{title}}</p>
14         <input type="text" v-model="user.name">
15         <input type="text" v-model="title">
16         <!--
17             Vue2.3 ~ .sync 会隐式的 为 指定的属性创建一个
update:属性名 方法
18                                     update:title
19             -->
20         <my-itany :user="user" :title.sync="title"></my-
itany>
21     </div>
22     <template id="itany">
23         <div>
24             <h1>子组件</h1>
25             <p>{{user}}</p>
26             <p>{{title}}</p>
27             <input type="text" v-model="user.age">
28             <input type="text" v-model="temp">
29             <input type="button" value="修改title"
@click="setTitle()">
30         </div>
31     </template>
32     <script>
33         new Vue({
34             el:"#app",
35             data:{
36                 title:"父组件标题",
37                 user:{
38                     name:"jack",
39                     age:23
40                 }
41             },

```

```
42         components:{
43             MyItany:{
44                 template:"#itany",
45                 props:{
46                     user:"",
47                     title:""
48                 },
49                 methods:{
50                     setTitle(){
51                         // this.title = "测试";
52                         this.$emit("update:title","测试");
53                     }
54                 },
55                 computed:{
56                     temp:{
57                         get(){
58                             return this.title;
59                         },
60                         set(data){
61                             this.$emit("update:title",data);
62                         }
63                     }
64                 }
65             }
66         }
67     });
68 </script>
69 </body>
70 </html>
```



十一、路由

- 三方组件 `vue-router` , Vue 官方提供和维护的
- 在页面中实现组件的切换
- `npm install vue-router`

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <link rel="stylesheet" href="../css/animate.css">
9   <style>
10     .a{
11       position: absolute;
12     }
13   </style>
14   <script src="../js/vue.js"></script>
15   <!-- 在单页面中 完成 多组件间的切换 -->
16   <script src="../js/vue-router.js"></script>
17   <title>路由的实现</title>
18 </head>
19 <body>
```

```

19     <div id="app">
20         <div>
21             <!--
22                 <router-link></router-link>    ==> html  <a>
    标签
23                     to    指向该标签所跳转的路由
24             -->
25             <router-link to="/">首页</router-link>
26             <router-link to="/news">新闻</router-link>
27         </div>
28         <hr>
29         <!-- 4、加载对应地址的 组件 -->
30         <!--
31             <router-view></router-view>
32             占位符    路由的占位符
33             <router-view></router-view> 默认会去请求一个路径
    /
34         -->
35         <transition enter-active-class="animated
    bounceInLeft" leave-active-class="animated bounceOutRight">
36             <router-view class="a"></router-view>
37         </transition>
38
39     </div>
40
41     <template id="news">
42         <div>
43             <h1>新闻</h1>
44         </div>
45     </template>
46     <template id="home">
47         <div>
48             <h1>首页</h1>
49         </div>
50     </template>
51     <script>
52         // 1、定义组件
53         // Vue.component("news",{
54         //     template:"#news"
55         // });
56         // Vue.component("home",{
57         //     template:"#home"

```



```

58         // });
59         var Home = {
60             name: "home", //让用户定义该组件的名称
61             template: "#home"
62         };
63         var News = {
64             template: "#news"
65         };
66
67         // Vue.component("home", Home)
68
69         // 2、配置路由
70         // class VueRouter
71         //     opt    地址和组件间的关系
72         // new VueRouter(opt:Object)
73         // new VueRouter({
74         //     // 指定 路由路径和 组件间的关系
75         //     routes:[
76         //         {},
77         //         {}
78         //     ]
79         // })
80
81         // 2.1 定义路由关系对象
82         const routes = [
83             {path: "/", component: Home},
84             // {path: "/home", component: Home},
85             {path: "/news", component: News},
86         ];
87         // 2.2 传入配置项，实例路由和组件
88         const router = new VueRouter({
89             // routes: routes
90             routes
91         });
92
93         // 3、在容器中 注入路由 注册路由
94         new Vue({
95             el: "#app",
96             // router: router
97             router
98         });
99     </script>

```

```
100 </body>
```

```
101 </html>
```

十二、模块化开发

1、vue Ajax 模块化

2、单文件组件 .vue （html css js）

十二、vuex 中央数据总线（仓库）