|  |  |  |
| --- | --- | --- |
| jspang（Vue） | | |
| directive  自定义指令 | <p v-cheng='color'>{{num}}</p> <button @click='add'>加一</button>  Vue.directive('cheng',function(el,binding){  **console**.log(el);  **console**.log(binding);  el.**style**.**color**=binding.**value**; })  Vue.directive('cheng',{bind,inserted,update,componentUpdated}) | jspang  后面接一个对象，分别是五个周期函数 |
| extend扩展实力构造器 | var authorExtend=Vue.extend({  **template**:'<p><a :href="authorURL">{{authorName}}</a></p>',  data:function(){  return {  **authorName**:'Lion',  **authorURL**:'http://baidu.com'  }  } }) new authorExtend().$mount('aaa') new authorExtend().$mount('#aaa') |  |
| Vue.set全局操作 | function change(){  Vue.**set**(app,'count',6) // 改编数据的对象（app/mydata） 属性 改变的值 } |  |
| 生命周期（钩子函数） | beforeCreate:function(){},初始化之后 created:function(){},创建完成 beforeMount:function(){},挂载之前 mounted:function(){},被创建 beforeUpdate:function(){},数据更新前 updated:function(){},被更新后 activated:function(){},activated deactivated:function(){},deactivated beforeDestroy:function(){},销毁之前 destroyed:function(){}销毁之后 |  |
| Template  三种方式使用template | <div id="app">  <div id="tem\_1">  <p>11</p>  </div> </div> <div id="app2">  </div> <div id="app3">  <script type="x-template" id="demo3">  <div>  <p>33</p>  <p>33</p>  </div>  </script> </div>  var app=new Vue({  **el**:'#app',  **data**:{  **message**:'hello Vue!'  },  **template**:'#tem\_1' }); var app2 = new Vue({  **el**:'#app2',  template:`<div><p>222</p><p>222</p><p>222</p></div>` }) var app3=new Vue({  **el**:'#app3',  **template**:'#demo3' }) | template模板标签里面只能放一个标签，所以一般都是放一个大盒子 |
| 组件component | //全局组件 Vue.component('Lion',{  **template**:'<div style="color: red"><div>全剧组件1</div><div>全剧组件1</div></div>' })  //局部组建 var app2 = new Vue({  **el**:'#app2',  **components**:{  **'liona'**:{  **template**:'<div style="color: green;">局部组建</div>'  },  **'ynn2'**:{  **template**:'<div style="color: green;">局部组建2</div>'  }  } }) | 局部：components  全局：  component |
| component的props属性，绑定 | <lion v-bind:mess="mess"></lion>  var app = new Vue({  **el**:'#app',  **data**:{  **mess**:'传值'  },  **components**:{  **'lion'**:{  **template**:'<div>我是12345，你是 {{ mess }} </div>',  **props**:['mess']  }  } }) |  |
| 子父组件 | var ynn={  **template**:'<div style="color: green">我是子组件</div>' } var app = new Vue({  **el**:'#app',  **components**:{  **'lion'**:{  **template**:'<div style="color: red">我是父组件<ynn></ynn></div> ',  **components**:{  **'ynn'**:ynn  }  }  } }) |  |
| component标签 | <component v-bind:is='who'>wos</component>  <button @click='change'>切换</button>  var linoA={  **template**:'<div>我是lionA</div> ' }; var linoB={  **template**:'<div>我是lionB</div> ' } var app= new Vue({  **el**:'#app',  **data**:{  **who**:'linoA'  },  **components**:{  **'lionA'**:linoA,  **'lionB'**:linoB  },  **methods**:{  change:function(){  if(this.**who**=='lionA'){  this.**who**='lionB'  }else if(this.**who**=='lionB'){  this.**who**='lionA'  }  }  } }) | 绑定的属性必须为is  v-bind:is=“”  表示绑定组件 |
| 选项  propsData | var order = Vue.extend({  **template**:'<div><p>{{mess}}</p><p>{{aaa}}</p></div>',  data:function(){  return {  **mess**:'我是一个扩展'  }  },  **props**:['aaa'] }) new order({**propsData**:{**aaa**:123456}}).$mount('#app') |  |
| 选项  computed | <p>{{newmess}}</p> <p>{{**mess**}}</p>  var app=new Vue({  **el**:'#app',  **data**:{  **mess**:100  },  **computed**:{  newmess:function(){  return '￥'+this.**mess**+'元'  }  } }) |  |
| 选项  methods | var order= {  **template**:'<button>组件加2</button>' } var app=new Vue({  **el**:'#app',  **data**:{  **mess**:1  },   **methods**:{  add:function(num,a){  **console**.log(a);  if(!num==''){  this.**mess**+=num  }else{  this.**mess**++  }  }  },  **components**:{  **order**:order  }, }) |  |
| 选项  watch监听 | var arr=['第一种','第二种','第三种']; var app=new Vue({  **el**:"#app",  **data**:{  **wendu**:10,  **zuo**:arr[1]  },  **methods**:{  add:function(){  this.**wendu**+=5  },  reduce:function(){  this.**wendu**-=5  }  },  **watch**:{  wendu:function(newV,oldV){  if(newV>=40){  this.**zuo** =arr[2]  }else if(newV>=20&&newV<40){  this.**zuo** =arr[1]  }else{  this.**zuo** =arr[0]  }  }  } })  eg:app.$**watch**('xxx',**function**(){}) | 局部：watch  全局：$watch |
| 选项:  混入mixins | var addlog={  **data**:{  **aaa**:20  },  **methods**:{  addadd:function(){  this.aaa++  }  } } var app=new Vue({  **el**:'#app',  **data**:{  **num**:1,  },  **methods**:{  add:function(){  this.**num**++  }  },  **mixins**:[addlog] })  全局混入：Vue.mixin（） | 局部：mixins  全局：mixin |
| 引用JQ | <button @click='change()'>change</button> <button @click='changeagaine()'>changeagaine</button> <button onclick='app.onadd()'>onadd</button>  var app=new Vue({  **el**:'#app',  **data**:{  **mess**:123  },  **methods**:{  change:function(){  $('#app p').html('改变了')  },  changeagaine:function(){  $('#app p').html('又改变了')  },  onadd:function(){  $('#app p').html('外部改变')  }  } }) |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 实例方法 | 1. $mount()挂在 2. $destory()卸载 3. $forceUpdate()更新 4. $nextTick()数据修改 |  |
| 实例事件 | <button onclick='reducebox()'>reduce</button>  app.$on('reduce',function(){  this.**num**-- }) function reducebox(){  app.$emit('reduce') } | $on |
|  | <button onclick="oncebox()">once</button>  app.$once('**once**',function(){  this.**num**=this.**num**\*10 }) function oncebox(){  app.$emit('**once**') } | $once |
|  | <button onclick="offbox()">off</button>  function offbox(){  app.$off('reduce') } | $off |
| 内置组件 | <div id="app">  <lion>  <span slot='name'>{{**dialogdata**.**name**}}</span>  <span slot='url'>{{**dialogdata**.**url**}}</span>  <span slot='skill'>{{**dialogdata**.**skill**}}</span>  </lion> </div> <template id='demo'>  <div>  <p>名称：<slot name='name'></slot></p>  <p>地址：<slot name='url'></slot></p>  <p>标签：<slot name='skill'></slot></p>  </div> </template>  var lion={  **template**:'#demo' } var app=new Vue({  **el**:'#app',  **data**:{  **dialogdata**:{  **name**:'百度',  **url**:'www.baidu.com',  **skill**:'web前端'  }  },  **components**:{  **lion**:lion  } }) |  |