第一节 propsData Option 全局扩展的数据传递

propsData 不是和属性有关，他在用全局扩展使进行传递数据。

我们用propsData三步骤解决传值：

1.在全局扩展里加入props进行接收。 propsData{a:1}

2.传递时用propsData进行传递 。 Props['a']

3.用插值的形式写入模板 {{a}}

完整代码：

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10 | Var header-a=Vue.extend({  Template:`<p>{{message}}-{{a}}</p>`,  Data:function(){  Return{  Message:’Hello，I am header!‘  }  },  Props['a']  });  New header-a.({propsData{a:1}})  Var header-a=Vue.extend({  Template:`<p>{{message}}-{{a}}</p>`,  Data:function(){  Return{  Message:’Hello，I am header!‘  }  },  Props['a']  });  New header-a({propsData{a:1}}).$mount('header'); |

第二节 computed Option 计算选项

computed的作用：主要是对源数据进行改造输出。

改造输出：包括格式的编辑，大小写转换，顺序重排，添加符号….

1.格式化输出结果：

读出价格的例子：原始数据是price：100，输出给用户的样子是￥100元

主要JS代码：

Computed：{

newPrice:function(){

Return: this.price='￥'+'this.price'+'元';

}

}

2.用computed属性反转数组

例如我们得到一个新闻列表数组，按时间进行排序

主要JS代码：

Computed:{

reverseNews:function(){

Return this.newsList.reverse();

}

}

第三节 Methoids Option 方法选项

一、使用方法和人JavaScript的使用发放一样，分为两部：

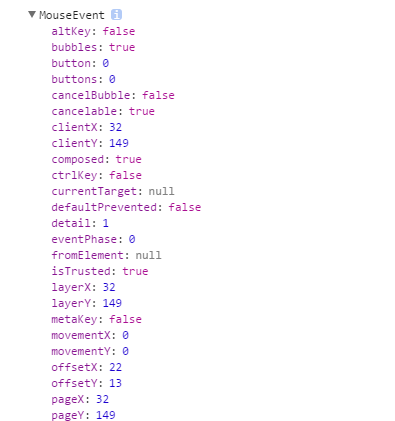
1、在methods的方法中进行声明，比如我们再给add方法加上一个num参数，就要写出add：function(num){}

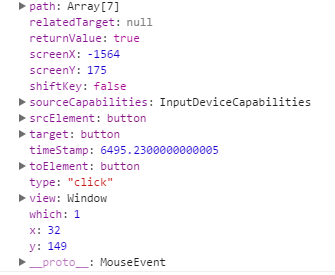
2、调用方法时直接传递，比如我们要传递2这个参数，在button上直接可以写<button @click="add(2)"></button>

二methods中的$event参数

传递的event参数都是关于你点击鼠标事件的一些属性

传递：<button @click="add(2,event)"></button>





三、native 给组件绑定构造器里的原生事件

在实际开发中经常需要把偶个按钮封装成组件，然后反复使用，如何让组件调用构造器里的方法，而不是组件里的方法

现在把add按钮组装成组件：

声明btn对象：

var btn:={

Template:`<button>组件ADD</button>`

}

在构造器里声明：

Components：{

“btn”:btn

}

用.native修饰器来调用构造器里的add方法

<p><btn @click.native="add(3)"></btn></p>

四、作用域外部调用构造器的方法

这种不经常使用，如果你出现了这种情况，说明你的代码组件不够好

<button onclick="app.add(4)"></button>

第四节 watch 选项 监控数据

一、一个监控变化的案例

模拟温度变化情况，根据温度变化提示建议穿衣

二、用实力属相写watch监控

有些时候我们会用实例属性的形式来写watch监控，吧watch写在构造器的外部，这样写的好处降低了我们程序的耦合度，是程序变得灵活

App.$(watch('xxx'),function(){});

第五节 Mixins 混入选项操作

Mixins一般有两种用途：

1.在你已经写好了后，需要年增加方法或者临时的活动时使用的方法，这是用混入会减少原代码的污染

2.很多地方都会用到的公用的方法，用混入的方法可以减少代码量，实现代码重用

一、Mixins的进本用法

我们现在有个数字点击递增的程序，假设已经完成了，这时我们希望每次数据变化时都能够在控制台打印出提示：“数据发生变化”.



二、调用顺序

从调用顺序来说，都是混入的先执行，然后构造器里的在执行，需要注意的是，这并不是方法的覆盖，而是被执行了两遍。

全局混入先执行 在执行混入方法 最后执行构造器里的混入

第六节：Extendsd Option 扩展选项

通过外部增加对象的形式，对构造器进行扩展。

实例



二、delimiters选项

delimiters是改变我们插值的符号，vue默认的插值是双大括号{{}}，但我们会有需求更改这个插值

 第一节 propsData Option 全局扩展的数据传递

propsData 不是和属性有关，他在用全局扩展使进行传递数据。

我们用propsData三步骤解决传值：

1.在全局扩展里加入props进行接收。 propsData{a:1}

2.传递时用propsData进行传递 。 Props['a']

3.用插值的形式写入模板 {{a}}

完整代码：

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10 | Var header-a=Vue.extend({  Template:`<p>{{message}}-{{a}}</p>`,  Data:function(){  Return{  Message:’Hello，I am header!‘  }  },  Props['a']  });  New header-a.({propsData{a:1}})  Var header-a=Vue.extend({  Template:`<p>{{message}}-{{a}}</p>`,  Data:function(){  Return{  Message:’Hello，I am header!‘  }  },  Props['a']  });  New header-a({propsData{a:1}}).$mount('header'); |

第二节 computed Option 计算选项

computed的作用：主要是对源数据进行改造输出。

改造输出：包括格式的编辑，大小写转换，顺序重排，添加符号….

1.格式化输出结果：

读出价格的例子：原始数据是price：100，输出给用户的样子是￥100元

主要JS代码：

Computed：{

newPrice:function(){

Return: this.price='￥'+'this.price'+'元';

}

}

2.用computed属性反转数组

例如我们得到一个新闻列表数组，按时间进行排序

主要JS代码：

Computed:{

reverseNews:function(){

Return this.newsList.reverse();

}

}

第三节 Methoids Option 方法选项

一、使用方法和人JavaScript的使用发放一样，分为两部：

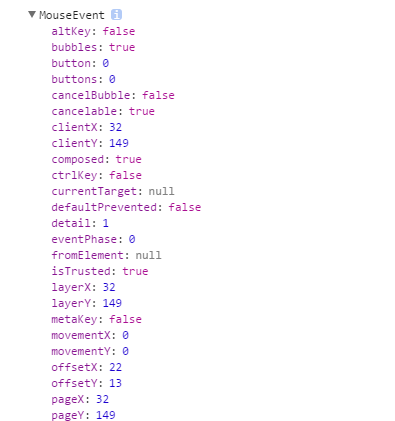
1、在methods的方法中进行声明，比如我们再给add方法加上一个num参数，就要写出add：function(num){}

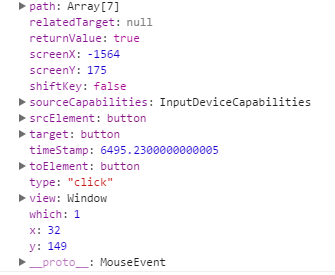
2、调用方法时直接传递，比如我们要传递2这个参数，在button上直接可以写<button @click="add(2)"></button>

二methods中的$event参数

传递的event参数都是关于你点击鼠标事件的一些属性

传递：<button @click="add(2,event)"></button>





三、native 给组件绑定构造器里的原生事件

在实际开发中经常需要把偶个按钮封装成组件，然后反复使用，如何让组件调用构造器里的方法，而不是组件里的方法

现在把add按钮组装成组件：

声明btn对象：

var btn:={

Template:`<button>组件ADD</button>`

}

在构造器里声明：

Components：{

“btn”:btn

}

用.native修饰器来调用构造器里的add方法

<p><btn @click.native="add(3)"></btn></p>

四、作用域外部调用构造器的方法

这种不经常使用，如果你出现了这种情况，说明你的代码组件不够好

<button onclick="app.add(4)"></button>

第四节 watch 选项 监控数据

一、一个监控变化的案例

模拟温度变化情况，根据温度变化提示建议穿衣

二、用实力属相写watch监控

有些时候我们会用实例属性的形式来写watch监控，吧watch写在构造器的外部，这样写的好处降低了我们程序的耦合度，是程序变得灵活

App.$(watch('xxx'),function(){});

第五节 Mixins 混入选项操作

Mixins一般有两种用途：

1.在你已经写好了后，需要年增加方法或者临时的活动时使用的方法，这是用混入会减少原代码的污染

2.很多地方都会用到的公用的方法，用混入的方法可以减少代码量，实现代码重用

一、Mixins的进本用法

我们现在有个数字点击递增的程序，假设已经完成了，这时我们希望每次数据变化时都能够在控制台打印出提示：“数据发生变化”.



二、调用顺序

从调用顺序来说，都是混入的先执行，然后构造器里的在执行，需要注意的是，这并不是方法的覆盖，而是被执行了两遍。

全局混入先执行 在执行混入方法 最后执行构造器里的混入

第六节：Extendsd Option 扩展选项

通过外部增加对象的形式，对构造器进行扩展。

实例

1. delimiters选项

delimiters是改变我们插值的符号，vue默认的插值是双大括号{{}}，但我们会需求更改这个插值delimiters：['${','}']