杰普软件科技有限公司
www.briup.com

Tel: (021)55660810
Fax: (021)55660802
Email:training@briup.com
Msn: training.sh@hotmail.com
Home:http://www.briup.com

地址：上海市闸北区万荣路1188弄F栋6号三层-上海服务外包科技园龙软园区
邮编：200436

电话：021-56657112
传真：021-55661523-8003
电邮：training@briup.com
主页：http://www.briup.com

Briup High-End IT Training

vue

第 2 章

事件、表单、计算属性、监听器

Brighten Your Way And Raise You Up.

◆ 概述

在dom阶段，我们已经讲述了事件机制的特点：

1. 事件三要素
2. 事件绑定
3. 事件流
4. 事件对象
5. 事件代理
6. 事件类型

这些概念在vue中依旧存在，但是vue中的事件机制要更加的简单便捷一些

事件机制

- 在前端开发中，我们需要经常交互。
  - 这个时候，我们就必须监听交互的发生，比如点击、拖拽、键盘事件等等
  - 在Vue中如何监听事件呢？使用v-on指令
- v-on介绍
  - 作用：绑定事件监听器
  - 缩写：@
  - 预期：Function | Inline Statement | Object
  - 参数：event
- 下面，我们就具体来学习v-on的使用。

事件机制

◆这里，我们用一个监听按钮的点击事件，来简单看看v-on的使用

- 下面的代码中，我们使用了v-on:click=“counter--”,可以直接在后面写入表达式
- 另外，我们可以将事件指向一个在methods中定义的函数

```
<div id="app">
    <button v-on:click="counter--">点击按钮1</button>
    <button v-on:click="btnClick">点击按钮2</button>
</div>
methods: {
    btnClick(){
      this.counter++
    }
}
```

◆注：v-on也有对应的语法糖：

- v-on:click可以写成@click

```
<button @click="btnClick">点击按钮2</button>
```

事件机制

- v-on参数问题
- 当通过methods中定义方法，以供@click调用时，需要注意参数问题：
- 情况一：如果该方法不需要额外参数，那么方法后的()可以不添加。
  - 但是注意：如果方法本身中有一个参数，那么会默认将原生事件event参数传递进去

```
<button @click="handleAdd">点击按钮1</button>
handleAdd(event){
    console.log(event);
}
```

- 情况二：如果需要同时传入某个参数，同时需要event时，可以通过$event传入事件。

```
<button @click="handleAddTen(10, $event)">点击按钮2</button>
handleAddTen(count, event){
    console.log(count, event);
}
```

◆ 事件修饰符

在事件处理程序中调用 event.preventDefault() 或 event.stopPropagation() 是非常常见的需求。Vue提供了更好的方式：事件处理函数只有纯粹的数据逻辑，而不是去处理 DOM 事件细节，通过事件修饰符来完成这些细节。

常见修饰符如下

.stop　　停止事件冒泡，相当于调用event.stopPropagation()

```
<button @click.stop="doThis">停止冒泡</button>
```

.prevent　　阻止事件默认行为，相当于调用 event.preventDefault()

```
<form action="baidu">
    <input type="submit" value="提交" @click.prevent="submitClick">
</form>
```

.once　　事件处理函数执行一次后解绑

```
<button @click.once="doThis"></button>
```

.{keyCode | keyAlias}　　只当事件是从特定键触发时才触发回调

```
<input @keyup.enter="doThis">
```

◆ 事件修饰符

按键修饰符

一般与keyup事件类型配合使用

```
.enter、.tab、.delete、.esc、.space、.up、.down、.left、.right
.ctrl、.alt、.shift、.meta
```

鼠标修饰符

```
.left、.right、.middle
```

表单输入绑定

◆ 概述

用v-model指令在表单<input>、<textarea> 及 <select> 等表单元素上创建双向数据绑定。它会根据控件类型自动选取正确的方法来更新元素。v-model 会忽略所有表单元素的 value、checked、selected attribute 的初始值而总是将 Vue 实例的数据作为数据来源。你应该通过 JavaScript 在组件的 data 选项中声明初始值。

表单输入绑定

- 表单控件在实际开发中是非常常见的。特别是对于用户信息的提交，需要大量的表单。
- Vue中使用v-model指令来实现表单元素和数据的双向绑定。

```
<input type="text" v-model="message">
<h2>输入的值是：{{message}}</h2>
data: {
    message: ''
}
```

- 以上代码的解析：
  - 当我们在输入框输入内容时
  - 因为input中的v-model绑定了message，所以会实时将输入的内容传递给message，message发生改变。
  - 当message发生改变时，因为上面我们使用Mustache语法，将message的值插入到DOM中，所以DOM会发生响应的改变。
  - 所以，通过v-model实现了双向的绑定。
- 当然，我们也可以将v-model用于textarea元素

表单输入绑定

- v-model与表单元素结合使用

1. 单行文本框

```
<input v-model="message" placeholder="edit me">
```

2. 多行文本框

```
<textarea v-model="message" placeholder="multiple lines"></textarea>
```

3. 单选按钮

```
<input type="radio"  value="One" v-model="picked">
<input type="radio"  value="Two" v-model="picked">
```

4. 复选按钮

```
<input type="checkbox" value="Jack" v-model="checkedNames">
<input type="checkbox" value="John" v-model="checkedNames">
```

5. 下拉菜单

```
<select v-model="selected">
     <option>A</option>
     <option>B</option>
     <option>C</option>
</select>
```

表单输入绑定

◆ 修饰符

使用修饰符可以增加表单绑定能力

1. lazy

默认情况下，v-model 在每次 input 事件触发后将输入框的值与数据进行同步。可以添加 lazy 修饰符，从而转为在 change 事件之后进行同步

```
<!-- 在“change”时而非“input”时更新 -->
<input v-model.lazy="msg">
```

2. number

如果想自动将用户的输入值转为数值类型，可以给 v-model 添加 number 修饰符

```
<input v-model.number="age" type="number">
```

3. trim

如果要自动过滤用户输入的首尾空白字符，可以给 v-model 添加 trim 修饰符

```
<input v-model.trim="msg">
```

计算属性

computed

- 我们知道，在模板中可以直接通过插值语法显示一些data中的数据。
- 但是在某些情况，我们可能需要对数据进行一些转化或计算后再显示，或者需要将多个数据结合起来进行显示
  - 比如我们有firstName和lastName两个变量，我们需要显示完整的名称。
  - 但是如果多个地方都需要显示完整的名称，我们就需要写多个{{firstName}} {{lastName}}
- 我们可以将上面的代码换成计算属性：
  - OK，我们发现计算属性是写在实例的computed选项中的。

```
<h2>{{fullName}}</h2>
computed: {
    fullName(){
        return this.firstName + ' ' + this.lastName
    }
},
```

计算属性

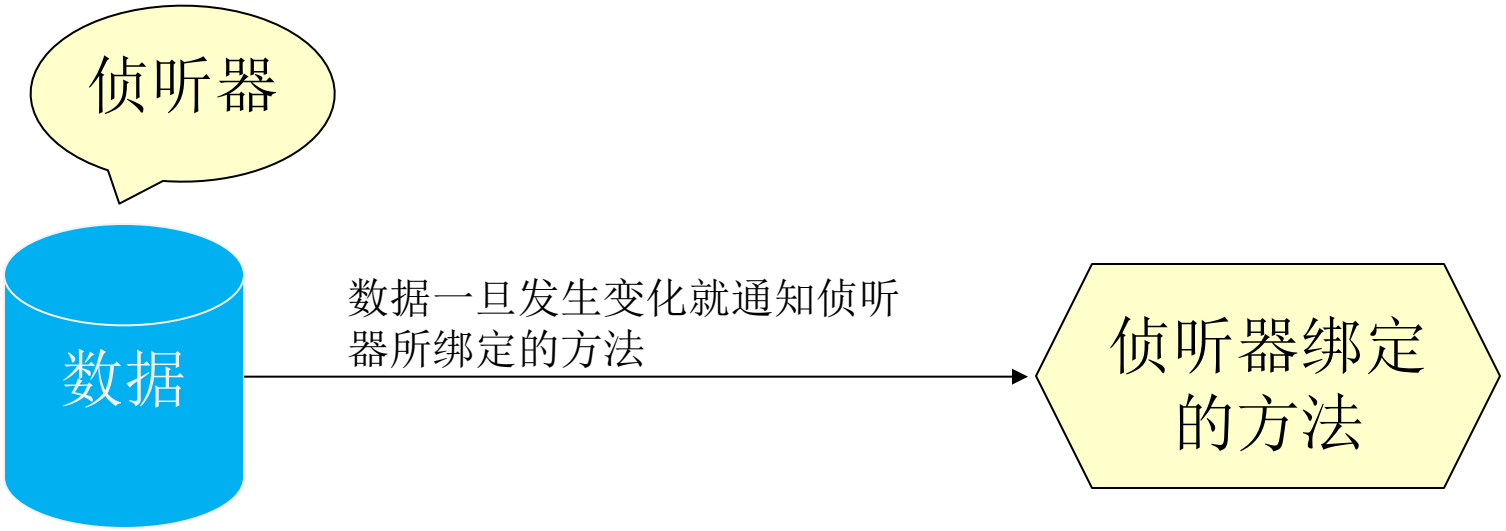
- 计算属性的缓存

- 我们可能会考虑这样的一个问题：
  - methods和computed看起来都可以实现我们的功能，
  - 那么为什么还要多一个计算属性这个东西呢？
  - 原因：计算属性会进行缓存，如果多次使用时，计算属性只会调用一次。

侦听器

◆侦听器的应用场景

- 使用watch来响应数据的变化
- 一般用于异步或者开销较大的操作
- watch中的属性 一定是data中已经存在的数据



侦听器

- 侦听器的基本使用

```
data: {
    testData: 'testData'
},
// 使用侦听器来监听testData的变化
watch: {
// 侦听数据的变化 函数名要和data中定义的属性名保持一致
// 我们给其定义一个形参val 表示的是testData当前更新后最新的值 只要数据发生变化 val就会跟着变化 就会得到最新的值
testData(val){
    //val就是testData更新后拿到的最新的值
    console.log(val);
}}
```

- 侦听器在使用层面上与计算属性类似，但是 有一些场景计算属性是实现不了的 异步或者开销较大的操作的时候