Vue.js详解

Lesson 8

Vue列表渲染

授课老师：爱米

Vue.js列表渲染

- 1.v-for
- 2.template v-for
- 3.数组变动检测
- 4.对象v-for
- 5.值域v-for
- 6.显示过滤/排序的结果

v-for

- 使用 v-for 指令基于一个数组渲染一个列表。

```
<ul id="demo"><li v-for="item in items">{{ item.message }}</li></ul>
```

```
var demo = new Vue({
  el: '#demo',
  data: {
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
});
```

v-for

在v-for块内能访问父组件作用域内的属性以及数组索引$index：

```
<ul id="demo">
  <li v-for="item in items">
    {{parentMessage}}-{{$index}}-{{item.message}}
  </li>
</ul>
```

```
var demo = new Vue({
  el: '#demo',
  data: {
    parentMessage: 'Parent',
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
});
```

v-for

- 从1.0.17开始可以使用of分隔符，更接近JavaScript遍历器语法：

```
<div v-for="item of items"></div>
```

template v-for

- 类似于 template v-if，也可以将 v-for 用在 <template> 标签上，以渲染一个包含多个元素的块。

```
<ul>
  <template v-for="item in items">
    <li>{{ item.msg }}</li>
    <li class="divider"></li>
  </template>
</ul>
```

数组变动检测

- 1.变异方法
- 2.替换数组
- 3.track-by
- 4.track-by $index
- 5.问题

变异方法

- 1.push()/pop()
- 2.shift()/unshift()
- 3.splice()
- 4.sort()/reverse()

替换数组

- 不会修改原始数组而是返回一个新数组的非变异方法，直接用新数组替换旧数组：

```
demo.items = example1.items.filter(function (item) {
  return item.message.match(/Foo/);
})
```

- filter()/concat()/slice()

track-by

- 用全新对象替换数组，使用track-by特性给Vue.js提示，以尽可能地复用已有实例。

```
{items: [
  { _uid: '88f869d', ... }, { _uid: '7496c10', ... }
]}
```

```
<div v-for="item in items" track-by="_uid">
  <!-- content -->
</div>
```

track-by $index

- 使用track-by="$index"强制让v-for进入原位更新模式。
- 片断不会被移动，而是简单地以对应索引的新值刷新。
- DOM节点不再映射数组元素顺序的改变，不能同步临时状态以及组件的私有状态。

问题

- 因为 JavaScript 的限制，Vue.js 不能检测到下面数组变化：
- 1.直接用索引设置元素，如：vm.items[0] = {};
- 2.修改数据的长度，如：vm.items.length = 0。

问题

- 第一个问题，Vue.js扩展了观察数组，为它添加了一个$set()方法：

```
demo.items.$set(0, { childMsg: 'Changed!'})
```

- 第二个问题，需用一个空数组替换items。

问题

- 除了$set()，Vue.js也为观察数组添加了$remove()方法，用于从目标数组中查找并删除元素，在内部自动调用splice()。

```
var index = this.items.indexOf(item);
if (index !== -1) {
  this.items.splice(index, 1);
}
```

```
this.items.$remove(item);
```

对象v-for

- 也可以使用v-for遍历对象。除了$index之外，作用域内还可以访问另外一个特殊变量$key。

```
<ul id="repeat-object" class="demo">
  <li v-for="value in object">{{ $key }} : {{ value }}</li>
</ul>
```

```
new Vue({
  el: '#repeat-object',
  data: {
    object: {
      FirstName: 'John',
      LastName: 'Doe',
      Age: 30
    }
  }
});
```

值域v-for

- v-for也可以接收一个整数，此时它将重复模板数次。

```
<div>
  <span v-for="n in 10">{{ n }} </span>
</div>
```

显示过滤/排序的结果

- 想要显示过滤/排序过的数组，同时不实际修改或重置原始数据。有两个办法：

- 1.创建一个计算属性，返回过滤/排序过的数组；
  2.使用内置的过滤器 filterBy 和 orderBy。

- 计算属性有更好的控制力，也更灵活，因为它是全功能 JavaScript。但是通常过滤器更方便。

THANKS
FOR WATCH