项目七 自定义指令

本章内容

- 7.1 自定义指令概述
- 7.2 钩子函数
- 7.3 对象字面量

7.1 自定义指令概述

- 7.1.1 自定义全局指令
- 7.1.2 自定义局部指令
- 7.1.3 案例分析

7.1 自定义指令概述

- 除了 Vue 提供的基本指令外(如 v-model 和 v-show), Vue 也允许注册自定义指令。自定义指令是用来操作 DOM 的。自定义指令就是一种有效的补充和扩展，不仅可用于定义任何的 DOM 操作，并且是可复用的。
- 自定义指令分为全局指令和局部指令。

7.1.1 自定义全局指令

- 打开百度首页，搜索输入框就是直接获取焦点的。如图 7-1 所示。该功能其实可以通过注册一个全局指令 v-focus, 该指令的功能是在页面加载时，元素获得焦点。

图 7-1 获取焦点

7.1.1 自定义全局指令

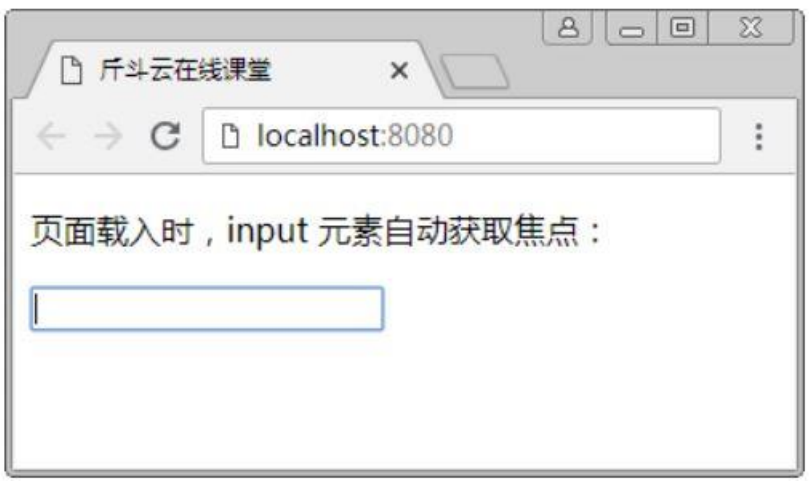
- 自定义全局指令使用 Vue.directive(指令 ID，定义对象)，在这里“指令 ID”就是指令的名字，“定义对象”就是一个对象，包含有该指令的钩子函数。
- 自定义全局指令语法如下，其中钩子函数均为可选

```
Vue. directive(“指令id”,{
        //当该指令第一次绑定到元素上时调用，只调用一次，可以用来执行初始化操作(简言之，指令绑定到元素)
        bind: function(){//常用!!！
        alert(“bind”)
},
        //被绑定有自定义指令的元素插入到DOM中时调用，在这里是插入到了#container中(简言之，元素插入到DOM元素中)
        inserted: function(){
        alert(" inserted ")
},
 //当被绑定的元素所在模板更新时调用
                update: function(){
                alert(“update”)},
  //当被绑定的元素所在模板完成一次更新时调用
componentUpdated: function(){
            alert(" componentUpdated “ )},
  //当指令和元素解绑的时候调用，只执行一次
            unbind: function(){
            alert(" unbind)")}
})
```

例7.1 自定义全局指令

```
<div id="app">
    <p>页面载入时，input 元素自动获取焦点：</p>
    <input v-focus>
  </div>
  <script>
    // 注册一个全局自定义指令 v-focus
    Vue.directive('focus', {
      // 绑定元素插入到 DOM 后执行钩子函数inserted
      inserted: function (el) {
        // 聚焦元素
        el.focus()
      }
    })
    // 创建根实例
    new Vue({
      el: '#app'
    })
  </script>
```

- 需要说明的是这里的 el 指的就是当前指令绑定的 DOM 元素，运行后光标定位在文本框中，直接获取焦点运行结果如下图 所示：



7.1.2 自定义局部指令

- 局部指令在 Vue 实例中使用 directives 选项来注册局部指令，局部指令只能在这个实例中使用。语法如下：

```
new Vue({
        el: '#app',
        directives:   {
              //定义局部指令
    }
})
```

例7.2 自定义局部指令

```
<div id="app">
    <p>页面载入时，input 元素自动获取焦点：</p>
    <input v-focus>
  </div>
  <script>
    // 创建根实例
    new Vue({
      el: '#app',
      directives: {
        // 注册一个局部的指令 v-focus
        focus: {
          // 指令的定义
          inserted: function (el) {  //这里的el指的就是当前指令绑定的DOM元素;
            // 聚焦元素
            el.focus()
          }
        }
      }
    })
  </script>
```

例7.3 定义可拖拽元素

```
<div id="app">
        <p>注意要先给元素加上position定位属性 ,v-drag拖拽是通过更改top和left值来实现的</p>
        <div class="drag" v-drag></div>
</div>
<script>
Vue.directive("drag", function (el) {    //el指的是当前绑定的div
        el.onmousedown = function (e) {
                var strX = e.pageX - this.offsetLeft;
                var strY = e.pageY - this.offsetTop;
                document.onmousemove = function (e) {
                        el.style.left = e.pageX - strX + "px";
                        el.style.top = e.pageY - strY + "px";
                };
                document.onmouseup = function () {
                        document.onmousemove = document.onmouseup = null;
                }
        }
});
var vm = new Vue({
        el: "#app",
        data: {}
})
</script>
```

7.2 钩子函数

- 一个指令的定义对象可以提供钩子函数，均为可选 bind、inserted、update 、 unbind 、componentUpdated，每个钩子函数的作用在自定义指令中已经介绍过。

7.2.1 钩子函数

- 钩子函数参数（如下面代码中的 inserted 钩子函数的参数 el ）

```
Vue. directive( ‘focus’ , {
        inserted: function (el) {
            el. focus()
    }
})
```

绑定元素插入到DOM后执行钩子函数inserted

7.2.1 钩子函数

- 其实除了el 还有其他的钩子函数参数，指令钩子函数会被传入以下参数
- el：指令所绑定的元素，可以用来直接操作 DOM 。
- binding：一个对象，包含以下属性：

1.name：指令名，不包括 v- 前缀。

2.value：指令的绑定值，例如：v-my-directive="1 + 1" 中，绑定值为 2。

7.2.1 钩子函数

3. oldValue：指令绑定的前一个值，仅在 update 和 componentUpdated 钩子中可用。无论值是否改变都可用。

4. expression：字符串形式的指令表达式。例如 v-my-directive="1 + 1" 中，表达式为 "1 + 1"。

5. arg：传给指令的参数，可选。例如 v-my-directive:foo 中，参数为 "foo"。

6. modifiers：一个包含修饰符的对象。例如：v-my-directive.foo.bar 中，修饰符对象为 { foo: true, bar: true }。

7.2.1 钩子函数

- vnode：Vue 编译生成的虚拟节点。可以对数据进行双向绑定。
- oldVnode：上一个虚拟节点，仅在 update 和 componentUpdated 钩子中可用。

例7.4 用随机的背景色占位

```
<div id="app">
<p>在图片未完成加载前，用随机的背景色占位，图片加载完成后才直接渲染出来。用自定义指令可以非常方便的实现这个功能。 </p>
<div v-imgurl="url"></div>
</div>
<script>
Vue.directive('imgurl', {
        //el指当前绑定的元素img;binding是一个对象
        inserted: function (el, binding) {
                var color = Math.floor(Math.random() * 1000000);//设置随机颜色
                //为img设置背景图片
                el.style.backgroundColor = '#' + color;
                var img = new Image();
                img.src = binding.value;// -->binding.value是指令的绑定值url
                img.onload = function () {
                        el.style.backgroundColor = '';
                        el.style.backgroundImage = "url(" + binding.value + ")";
                }
        }
})
// 创建根实例
new Vue({
        el: '#app',
        data: {
                url: "1.png"
        }
})
</script>
```

例7.5 演示自定义指令参数

```
<style>
    div {
      width: 200px;
      height: 200px;
    }
</style>
<div id="app">
    <div id="hook-arguments-example" v-demo-directive:lightgrey="message"></div>
  </div>
  <script>
    Vue.directive('demoDirective', {
      bind: function (el, binding, vnode) {
        el.style.color = '#fff'
        el.style.backgroundColor = binding.arg
        el.innerHTML =
          '指令名name - ' + binding.name + '<br>' +
          '指令绑定值value - ' + binding.value + '<br>' +
          '指令绑定表达式expression - ' + binding.expression + '<br>' +
          '传入指令的参数argument - ' + binding.arg + '<br>'
      }
    });
    var demo = new Vue({
      el: '#hook-arguments-example',
      data: {
        message: '你好，欢迎加入vue!'
      }
    })
  </script>
```

7.2.2 函数简写

- 在学习的钩子函数中，几乎都会存在el，binding这两个函数。当不需要其他钩子函数时，可以简写函数。

```
Vue. directive(  ‘ runoob’ ，function (el, binding) {
        //设置指令的背景颜色
        el. style. backgroundColor = binding . value. color

})
```

7.3 对象字面量

- Javascript 对象字面量，又称为映射，是键值对的集合。它的语法是用一对大括号包着用逗号分隔的键值对，其中键和值分别用单引号括起来，键和值之间用冒号分隔。
- 在编程语言中，字面量是一种表示值的记法。JavaScript 还支持对象和数组字面量，允许使用一种简洁而可读的记法来创建数组和对象

7.3 对象字面量

例7.6 对象字面量示例

```
<div id="app" v-demo-directive="{ color: 'white', text: 'hello!' }">
</div>
    <script>
        Vue.directive('demoDirective', function (el, binding, vnode) {
            document.write(binding.value.color + " " + binding.value.text);
        })

        var demo = new Vue({
            el: '#app'
        })
    </script>
```

7.3 对象字面量

- binding 是一个对象，binding.value 是对象子面量{ color: 'white', text: 'hello!' }，代码运行后 binding.value.color 获取white，binding.value.text 获取hello!

本章小结

- 本章主要讲了什么是自定义指令，注册全局自定义指令和局部自定义指令。自定义指令中用到的钩子函数 bind、inserted、update、componentUpdated、unbind，钩子函数参数 el、binding 等，指令参数，函数简写及对象字面量。

习题

- 7-1 请说明有那些钩子函数。
- 7-2 简述什么是对象字面量。
- 7-3 编写一个自定义指令的案例。