Day 11---事件

一、事件和事件流(了解)(102)

JavaScript是一种基于对象(Object)和事件驱动(Event Driven)并具有安全性能的脚本语言。JavaScript和HTML之间的交互就是通过一系列的事件来实现的。

1.1 什么是事件

JavaScript和HTML之间的交互是通过事件实现的，事件就是文档或浏览器窗口中发生的一些特点的交互瞬间，可以使用侦听器或处理程序来预订事件，以便事件发生时执行相应的代码.

事件，就是用户或者是浏览器执行的某种动作。

比如：click、load等都是事件的名字。

1.2 事件流

事件流描述的是从页面中接收事件的顺序。

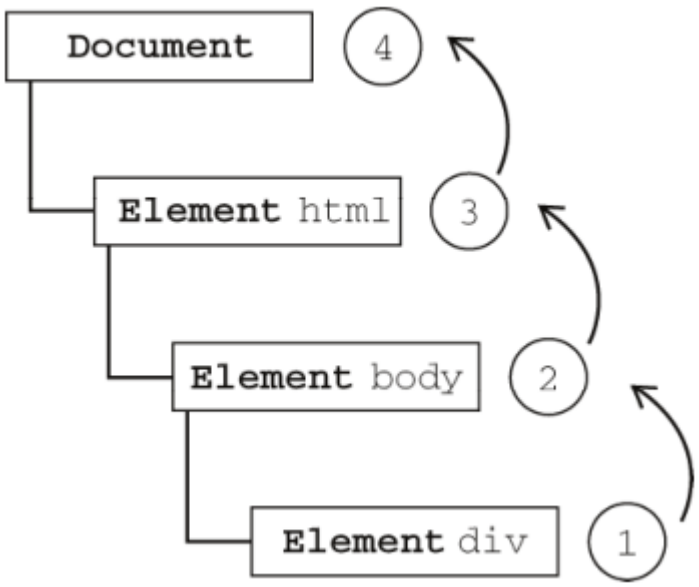
有两种事件流：冒泡流和捕获流

1.2.1 冒泡流

IE 的事件流叫做事件冒泡（event bubbling），即事件开始时由最具体的元素（文档中嵌套层次最深的那个节点）接收，然后逐级向上传播到较为不具体的节点（文档）。

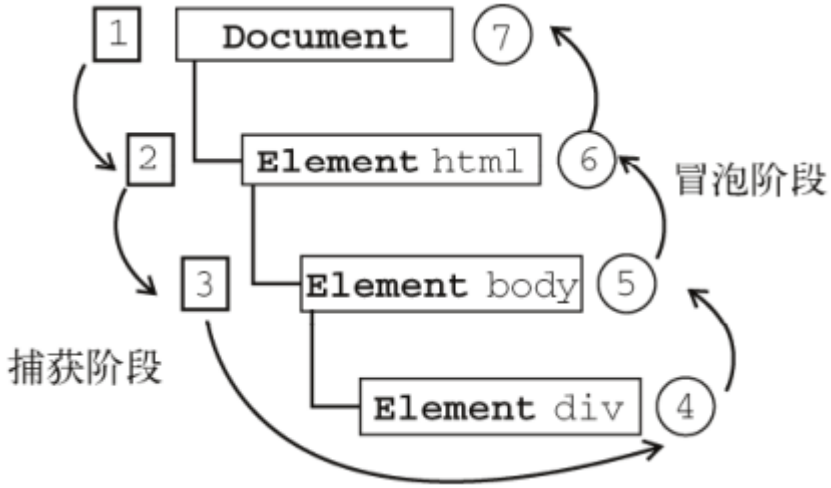
```
<!DOCTYPE html>
<html>
  <head>
    <title>Event Bubbling Example</title>
  </head>
  <body>
     <div id="myDiv">Click Me</div>
  </body>
</html>
```

现在点击div元素，则事件的传播顺序是冒泡的方式从底层向上传播。



1.2.2 捕获流

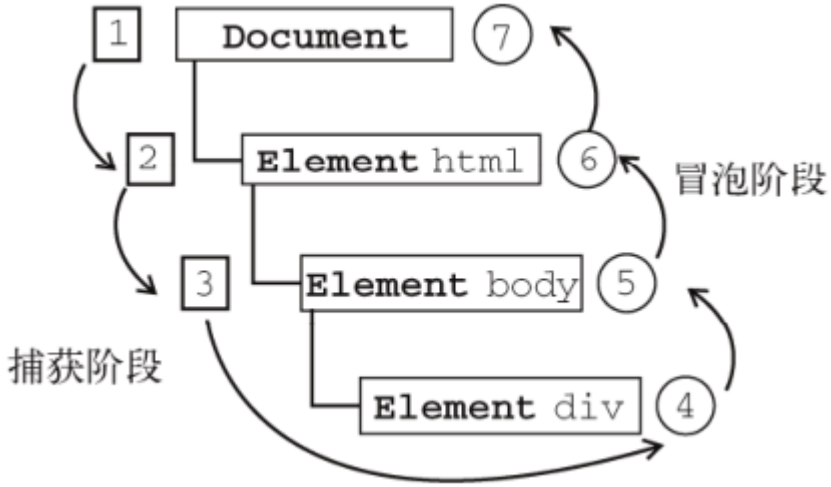
Netscape Communicator团队提出的另一种事件流叫做事件捕获（event capturing）。事件捕获的思想是不太具体的节点应该更早接收到事件，而最具体的节点应该最后接收到事件。



1.3 DOM事件流

“DOM2级事件”规定的事件流包括三个阶段：事件捕获阶段、处于目标阶段和事件冒泡阶段。

首先发生的是事件捕获，为截获事件提供了机会。然后是实际的目标接收到事件。最后一个阶段是冒泡阶段，可以在这个阶段对事件做出响应。



注意：IE9、Opera、Firefox、Chrome 和 Safari 都支持 DOM 事件流；IE8 及更早版本不支持 DOM 事件流。

二、事件处理程序

事件处理程序，就是响应事件的函数。事件处理程序的名字是以“on”开头的。
例如：事件click --->事件处理程序 onclick

提示：事件处理程序一般都是小写字母。

2.1 HTML事件处理程序(103)

每一个HTML标签所支持的每种事件，都可以使用与事件处理程序同名的属性来指定。这个属性的属性值是可以执行的js代码。

可以把代码直接写到onclick属性的值中:

```
<!-- 当点击button按钮时候，就会执行onclick属性值中的JavaScript的代码 -->
<button onclick="alert('hello world')">点我可以给整个世界问好</button>
```

更建议把代码封装在一个函数中，然后在onclick的属性值中调用封装的函数：

```
<!--注意：在onclick的属性中调用方法的时候，一定要添加( ),表示调用方法-->
<button onclick="showMsg();">点我可以给整个世界问好</button>
    <script type="text/javascript">
        function showMsg () {
            alert("hello world");
        }
    </script>
</body>
```

> 在HTML中指定事件处理程序有两个缺点:
>
> 1. 存在时差问题，比如函数定义在文档尾部，在未解析函数之前就点击了按钮，则会出现错误
> 2. HTML代码和JavaScript代码紧密耦合，如果要更换事件的处理程序，要同时改动html代码和js代码
>
> 这正是许多开发人员摒弃 HTML 事件处理程序，转而使用 JavaScript 指定事件处理程序的原因所在。

2.2 DOM0 级别事件处理程序(104)

为了遵循HTML 与JavaScript代码层次分离的原则，我们可以在JavaScript 中处理事件。这种处理方式也称为脚本模型。

所谓的DOM0级事件处理程序就是将一个函数赋值给事件处理程序。这时事件处理程序可以看成是元素对象的方法，事件处理程序就是在元素的作用域中运行。(this就指代这个元素对象)

> 优点：写法简单，跨浏览器

1. 匿名函数的写法。

```
<body>
    <button id="btn">点我可以给整个世界问好</button>
    <script type="text/javascript">
        // 匿名函数的写法
        document.getElementById("btn").onclick = function () {
            alert("世界你好");
        }
    </script>
</body>
```

2. 通过方法名给事件属性赋值。

```
<body>
    <button id="btn">点我可以给整个世界问好</button>
    <script type="text/javascript">
        //通过方法名赋值.  注意此处不能添加括号：因为方法的调用是在将来点击的时候，而不是现在
        document.getElementById("btn").onclick = showMsg;
        function showMsg () {
            alert("世界你再好");
        }
    </script>
</body>
```

2.3 删除事件处理程序

如果想让事件以后不再执行，则可以删除事件处理程序。

ocument.getElementById(id).onclick = null; 使用这种方法可以删除HTML事件和DOM0事件。

```
<body>
    <button id="btn" onclick="alert('你好世界')">点我可以给整个世界问好</button>
    <script type="text/javascript">
        document.getElementById("btn").onclick = null;
    </script>
</body>
```

三、事件类型(105)

JavaScript 可以处理的事件类型有很多，常见的有UI事件、焦点事件、鼠标事件、滚轮事件、文本事件、键盘事件。还有专门为触摸设备和移动设备准备的触摸事件、手势事件、屏幕方向改变等。

事件类型	事件	事件处理程序	说明
UI 事件	load	onload	1、当页面完全加载后在 window 上触发； 2、当图像加载完毕后在<img>元素上触发
	unload	onunload	与 onload 的相对，当页面完全卸载在 window 上触发 只有 IE 支持
	scroll	onscroll	页面滚动的时候在 window 上触发
	resize	onresize	页面进行缩放的时候在 window 上触发
焦点事件	focus	onfocus	在元素获得焦点时触发。该事件不会冒泡。
	blur	onblur	在元素失去焦点的时候触发。该事件不会冒泡
鼠标事件	click	onclick	用户按下鼠标
	dblclick	ondblclick	用户双击鼠标
	mouseover	onmouseover	当鼠标从元素外部进入元素内部时触发
	mouseout	onmouseout	当鼠标离开元素时触发
	mousedown	onmousedown	当鼠标在元素内部按下时触发
	mouseup	ommouseup	当鼠标在元素内部抬起时触发
键盘事件	keyup		抬起键盘上的任意键　e.keyCode
	keydown		按下键盘上的任意键
	keypress		按下键盘上的字符键键

重点掌握的事件有：

1.load
2.scroll
3.resize
4.click
5.mouseover
6.mouseout

3.1 UI事件

3.1.1 load事件

> load事件会在页面或图像加载完成立即执行。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
        //方法1：给window的onload属性设置函数。建议使用这种
        window.onload = function () {    // 函数在页面执行完毕之后自动执行。
            alert("页面加载完毕")
        };
    </script>
</head >
<!--方法2： HTML事件处理： 给body 的onload属性直接设置 JavaScript代码， 也可以调用函数-->
<body onload="alert('你好')">:
</body>
</html>
```

3.1.2 onunload事件

> - onunload事件在用户退出页面时发生，当页面完全卸载后在window上面触发，或当框架集卸载后在框架集上触发。
> - 只要用户从一个页面切换到另一个页面就会触发该事件
> - 仅IE支持。所以实际较少使用。

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
        window.onunload = function () { //像这种事件注册性质的代码一般写在head中比较好。
            alert("你要抛弃我了么？");
        }
    </script>
    <body>
        <a href="http://www.baidu.com">百度一下吧</a>
    </body>
</head >
```

3.1.3 onresize事件

> 当浏览器窗口被调整到一个新的高度或宽度时，就会触发 resize 事件。这个事件在 window （窗口）上面触发。所以也可以在body元素中使用 onresize 属性来指定事件处理程序

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
         window.onresize = function () {
            alert("窗口发生了变化")
        }
    </script>
</head>
```

> 当窗口发生变化的时候，可以获取浏览器新的可视宽和高

```
<script type="text/javascript">
    window.onresize = function () {
    //获取兼容模式下的可视宽和高
    var width = document.documentElement.clientWidth || window.innerWidth ||
document.body.clientWidth;
    var height = document.documentElement.clientHeight || window.innerHeight
                                                        || document.body.clientHeight;
    alert("窗口发生了变化\n" + "宽：" + width + "\n高：" + height);
  }
</script>
```

3.1.4 onscroll事件

> onscroll事件，当滚动条滚动式触发。事件在window上面触发。由于滚动过程中，会重复多次调用，所以，处理逻辑和代码不能过于复杂，或者会影响用户滚动效果。

```
<script type="text/javascript">
     window.onscroll = function () {
        console.log("开始滚动...");
        //跨浏览器获得滚动的距离
        console.log(document.documentElement.scrollTop | document.body.scrollTop);
    }
</script>
```

3.2 焦点事件(106)

> 焦点事件会在页面元素获得或失去焦点时触发。利用这些事件并与 document.hasFocus() 方法及 document.activeElement 属性配合，可以知晓用户在页面上的行踪。主要有4常用焦点事件。

3.2.1 onfucs(获得焦点)事件

> 当元素获获得焦点时触发。这个事件可以发生在任何的元素上。而且这个事件不会冒泡(也就是不会再往上层传递)

```
<body>
    <div>
        <input type="text" name="user">
    </div>
    <p>上面的文本框获得焦点后会我会变成红色</p>
    <script type="text/javascript">
        var p1 = document.getElementsByTagName("p")[0];
        var textInput = document.getElementsByTagName("input")[0];
        textInput.onfocus = function () {
            p1.style.color = "red";
        }
       var div1 = document.getElementsByTagName("div")[0];
       //当div中input获取焦点后，并会冒泡到上层div，所以这个函数不会执行。
        div1.onfocus = function () {
            div1.style.backgroundColor = "#000";
        }
    </script>
</body>
```

3.2.2 onblur(失去焦点)事件

> 当元素失去焦点是触发。和onfucs对应。这个事件也不冒泡

```
<body>
    <div>
        <input type="text" name="user">
    </div>
    <p>上面的文本框获得焦点后会我会变成红色,失去焦点会变成蓝色</p>
    <script type="text/javascript">
        var p1 = document.getElementsByTagName("p")[0];
        var textInput = document.getElementsByTagName("input")[0];
        textInput.onfocus = function () {
            p1.style.color = "red";
        }
       //失去焦点事件
        textInput.onblur = function () {
            p1.style.color = "blue";
        }
    </script>
</body>
```

3.3.3 onfocusin和onfocusout

> onfocusin是onfoucs的冒泡版本，onfocusout是onblur的冒泡版本。
>
> 这两个方法在chrome和firfox上面均不支持。

```
<body>
    <div>
        <input type="text" name="user">
    </div>
    <p>上面的文本框获得焦点后会我会变成红色,失去焦点会变成蓝色</p>
    <script type="text/javascript">
        var p1 = document.getElementsByTagName("p")[0];
        var textInput = document.getElementsByTagName("input")[0];
        //input获取焦点
        textInput.onfocusin = function () {
            // alert("获取焦点")
            p1.style.color = "red";
        }
        //input失去焦点
        textInput.onfocusout = function () {
            p1.style.color = "blue";
        }

        var div1 = document.getElementsByTagName("div")[0];
        //给div1设置设置获取焦点事件
        div1.onfocusin = function () {  //当div中input获取焦点后，并会冒泡到上层div，所以这个函数不会执行。
            div1.style.backgroundColor = "#000";
        }
        // 给div2设置失去焦点事件
        div1.onfocusout = function () {
            div1.style.backgroundColor = "#f00";
        }
    </script>
</body>
```

3.3 鼠标事件(107)

3.3.1 onclick事件

> 鼠标单击事件，一般是鼠标左键。单按下鼠标左键或按下回车后触发

3.3.2 ondbclick事件

> 鼠标双击事件

```
<body>
    <div>点我们啊</div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        div1.ondblclick = function () {
            alert("你双击了我，我很开心");
        }
    </script>
</body>
```

3.3.3 onmousedown事件

> 当用户按下任意鼠标按钮时触发。不能通过键盘触发

```
<body>
    <div>点我们啊</div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        div1.onmousedown = function () {
            alert("你按下鼠标键，你想干吗");
        }
    </script>
</body>
```

3.3.4 onmouseup事件

> 用户释放鼠标按钮时触发

```
<body>
    <div>点我啊</div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        div1.onmouseup = function () {
            alert("你松开了鼠标键，你想干吗");
        }
    </script>
</body>
```

3.3.5 onmouseover和onmouseout事件

> onmouseover当鼠标移动到一个元素的上方是触发。
>
> onmouseout当鼠标从一个元素的上方移走的时候触发

```
<body>
    <div>鼠标放上来我要变色</div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        //当鼠标移动到div的上方
        div1.onmouseover = function () {
            div1.style.backgroundColor = "blue";
        }
        //当鼠标从div的上方移走
        div1.onmouseout = function () {
            div1.style.backgroundColor = "white";
        }
    </script>
</body>
```

3.3.6 onmouseenter和onmouseleave事件

> - onmouseenter：这个事件不冒泡。效果同onmouseover
> - onmouseleave ：这个事件不冒泡，效果同onmouseout

3.3.7 onmousemove事件

> 当鼠标在元素内部移动时触发，这个事件会重复触发

3.4 键盘事件

> 有三个键盘事件：
>
> keydown：当用户按下键盘上的任意键时触发，而且如果按住不放的话，会重复触发此事件。
>
> keypress：当用户按下键盘上的字符键时触发，而且如果按住不放的话，会重复触发此事件。
>
> keyup：当用户释放键盘上的键时触发。

```
<body>
    <div><input type="text" value="鼠标放上来我要变色"></input></div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        /*div1.onkeydown = function () {
            console.log("你按下了任意键");
        }*/
        div1.onkeypress = function () {
            console.log("你按下了字符键");
        }
        div1.onkeyup = function () {
            console.log('你松开了键盘');
        }
    </script>
</body>
```

> 获取按下的具体键：

```
<body>
    <div>
        <input type="text" value="鼠标放上来我要变色"></input>
    </div>
    <script type="text/javascript">
        var div1 = document.getElementsByTagName("div")[0];
        //在键盘上每个键都有一个keyCode,可以通过如下方法获取每个键对应的keyCode
        div1.onkeypress = function(e) {
            alert(e.keyCode);
        }
        // 判断是否按下alt键、shift键、ctrl键
        /*div1.onkeydown = function (e) {
            if(e.altKey){
                alert("alt");
            }else if(e.shiftKey){
                alert("shift");
            }else if(e.ctrlKey){
                alert("ctrl")
            }
        }*/
        //判断是否同时按下了alt和ctrl键
        div1.onkeydown= function(e){
            if(e.altKey&& e.ctrlKey)
                {alert("alt和ctrl同时按下");
            }
        }
    </script>
</body>
```

3.5 load和ready事件的区别

> ready，表示文档结构已经加载完成（不包含图片等非文字媒体文件）；

> load，表示页面包含图片等文件在内的所有元素都加载完成。

```
```