自定义事件的触发dispatchEvent - 简书

笔记本: 程序员
创建时间: 2020/8/20 星期四 15:35
标签: Android
URL: https://www.jianshu.com/p/5f9027722204

自定义事件的触发dispatchEvent

R_X
1 2018.08.28 11:46:32 字数 746 阅读 41,030

一、element.dispatchEvent()

对于标准浏览器，其提供了可供元素触发自定义事件的方法：

`element.dispatchEvent()`。

不过，在使用该方法之前，我们还需要做其他两件事，即创建和初始化。

```
document.createEvent()
event.initEvent()
element.dispatchEvent()
举个例子：

var dom = document.querySelector('#id')
document.addEventListener('alert', function (event) {
  console.log(event)
}, false);

// 创建
var evt = document.createEvent("HTMLEvents");
// 初始化
evt.initEvent("alert", false, false);

// 触发，即弹出文字
dom.dispatchEvent(evt);
```

1、createEvent()

`createEvent()`方法返回新创建的Event对象，支持一个参数，表示事件类型，具体见下表：

参数	事件接口	初始化方法
HTMLEvents	HTMLEvent	initEvent()
MouseEvents	MouseEvent	initMouseEvent()
UIEvents	UIEvent	initUIEvent()

2、initEvent()

`initEvent()`方法用于初始化通过`DocumentEvent`接口创建的Event的值。

支持三个参数：`initEvent(eventName, canBubble, preventDefault)`
分别表示：

- 事件名称
- 是否可以冒泡
- 是否阻止事件的默认操作

3、dispatchEvent()

`dispatchEvent()`就是触发执行了，`dom.dispatchEvent(eventObject)`
参数`eventObject`表示事件对象，是`createEvent()`方法返回的创建的`Event`对象。

二、自定义事件

1、Event

自定义事件的函数有`Event`、`CustomEvent`和`dispatchEvent`

```
// 向 window派发一个resize内置事件
window.dispatchEvent(new Event('resize'))


// 直接自定义事件，使用 Event 构造函数:
var event = new Event('build');
var elem = document.querySelector('#id')
// 监听事件
elem.addEventListener('build', function (e) { ... }, false);
```

```
// 触发事件.
elem.dispatchEvent(event);
```

2、CustomEvent

`CustomEvent` 可以创建一个更高度自定义事件，还可以附带一些数据，具体用法如下：

```
var myEvent = new CustomEvent(eventname, options);
其中 options 可以是：
{
  detail: {
    ...
  },
  bubbles: true,    //是否冒泡
  cancelable: false //是否取消默认事件
}
```

其中 detail 可以存放一些初始化的信息，可以在触发的时候调用。其他属性就是定义该事件是否具有冒泡等等功能。

内置的事件会由浏览器根据某些操作进行触发，自定义的事件就需要人工触发。
dispatchEvent 函数就是用来触发某个事件：

```
element.dispatchEvent(customEvent);
```

上面代码表示，在 `element` 上面触发 `customEvent` 这个事件。
结合起来用就是：

```
// add an appropriate event listener
obj.addEventListener("cat", function(e) { process(e.detail) });

// create and dispatch the event
var event = new CustomEvent("cat", {"detail":{"hazcheeseburger":true}});
obj.dispatchEvent(event);
使用自定义事件需要注意兼容性问题，而使用 jQuery 就简单多了：

// 绑定自定义事件
$(element).on('myCustomEvent', function(){});

```

```
// 触发事件
$(element).trigger('myCustomEvent');
// 此外，你还可以在触发自定义事件时传递更多参数信息：

$( "p" ).on( "myCustomEvent", function( event, myName ) {
  $( this ).text( myName + ", hi there!" );
});
$( "button" ).click(function () {
  $( "p" ).trigger( "myCustomEvent", [ "John" ] );
});
```

3、IE浏览器

由于向下很多版本的浏览器都不支持document.createEvent()方法，因此我们需要另辟蹊径（据说IE有document.createEventObject()和event.fireEvent()方法，但是不支持自定义事件~~）。
IE浏览器有不少自给自足的东西，例如下面要说的这个"propertychange"事件，顾名思意，就是属性改变即触发的事件。
例如文本框value值改变，或是元素id改变，或是绑定的事件改变等等。

```
dom.evtAlert = "2012-04-01";
<br>dom.attachEvent("onpropertychange", function(e) {
    if (e.propertyName == "evtAlert") {
        fn.call(this);
    }
});
```

这个，当我们需要触发自定义事件的时候，只要修改DOM上自定义的evtAlert属性的值即可：

```
dom.evtAlert = Math.random().toString(36).substr(2)
```

此时就会触发dom上绑定的onpropertychange事件，又因为修改的属性名正好是"evtAlert", 于是自定义的fn就会被执行。这就是IE浏览器下事件触发实现的完整机制，应该说讲得还是蛮细的。

三、自定义事件的删除

与触发事件不同，事件删除，各个浏览器都提供了对应的事件删除方法，如`removeEventListener`和`detachEvent`。

不过呢，对于IE浏览器，还要多删除一个事件，就是为了实现触发功能额外增加的`onpropertychange`事件：

```
    dom.detachEvent("onpropertychange", evt);

    var fireEvent = function(element,event){
       if (document.createEventObject){
           // IE浏览器支持fireEvent方法
           var evt = document.createEventObject();
           return element.fireEvent('on'+event,evt)
       }
       else{
           // 其他标准浏览器使用dispatchEvent方法
           var evt = document.createEvent( 'HTMLEvents' );
           evt.initEvent(event, true, true);
           return !element.dispatchEvent(evt);
       }
    };
```