

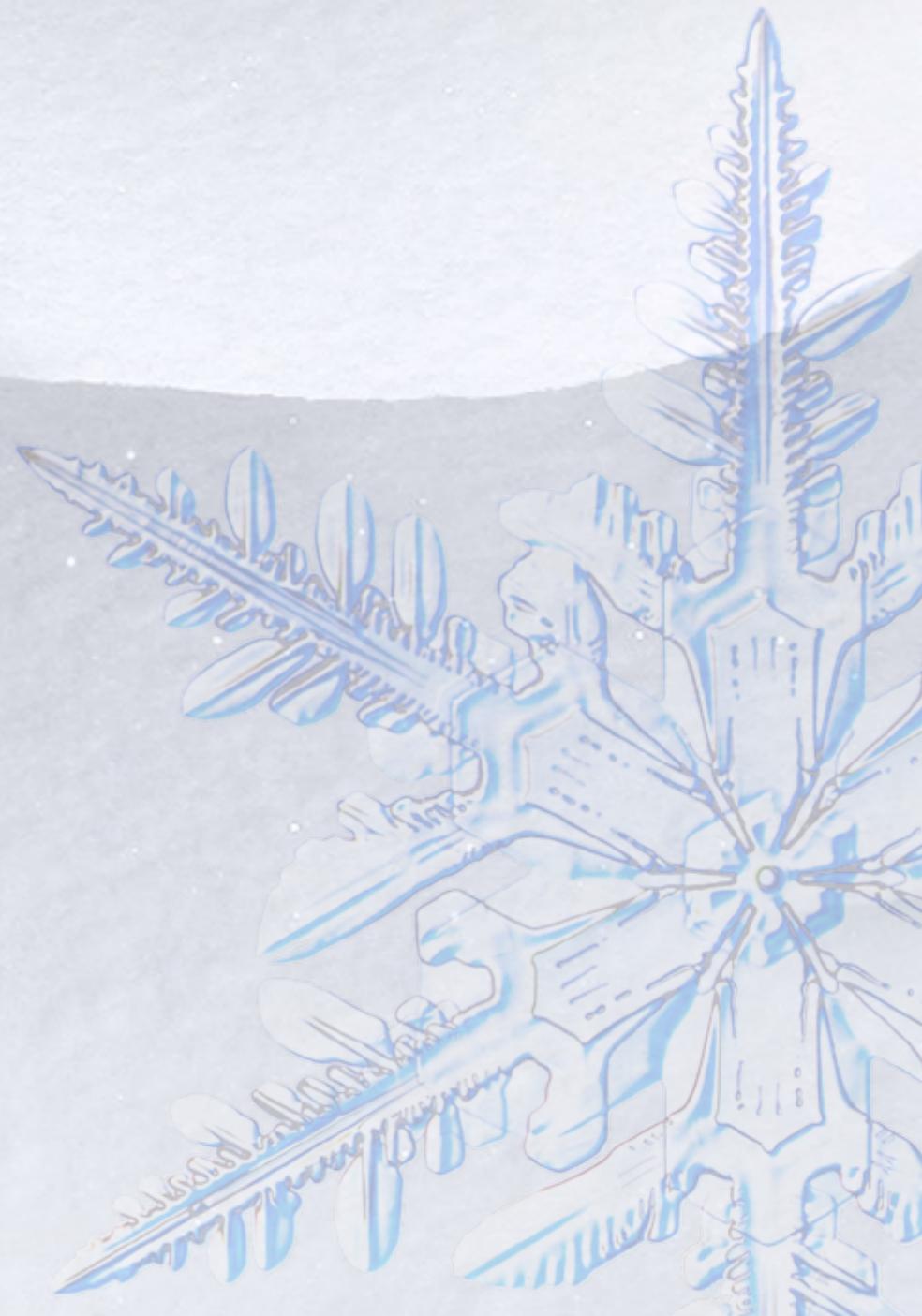
Lecture 7 - 1

JavaScript 事件



目录

- * *Javascript 事件*
- * 事件类型
- * 事件处理模型



事件

❄ 事件和事件处理

- * 使web应用程序的响应性、动态性和交互性更强
- * 通过回调编程

❄ JavaScript事件

- * 允许脚本响应用户与网页上元素的交互
- * 是否可以启动对页面的修改

事件驱动编程

* 事件驱动编程是一种编程范式，其中程序流由事件决定，诸如用户操作(鼠标点击、按键)、传感器输出或来自其他程序/线程的消息。

* 事件驱动编程是图形用户界面和其他应用程序(如JavaScript web应用程序)中使用的主要范式，这些应用程序以执行特定的操作来响应用户输入为中心。

事件驱动编程

- ❖ 在事件驱动的应用程序中，通常有一个主循环监听事件，然后在检测到其中一个事件时触发回调函数。
- ❖ 事件驱动程序可以用任何编程语言编写，尽管使用提供高级抽象的语言(如闭包)更容易完成此任务。

事件处理程序

- ❖ 处理程序接收的输入的回调子例程(在Java和JavaScript中称为
 侦听器)
- ❖ 当事件发生时调用的函数
- ❖ 通常与XHTML元素相关联
- ❖ 必须注册
 - * 也就是说，必须指定关联

JavaScript 的用途

* 事件处理程序可用于处理和验证用户输入、用户操作和浏览器操作：

- * 每次加载页面时应该做的事情
- * 当页面关闭时应该做的事情
- * 当用户单击按钮时应该执行的操作
- * 当用户输入数据时应该验证的内容
- * 还有更多.....

* 可以使用许多不同的方法让JavaScript处理事件：

- * HTML事件属性可以直接执行JavaScript代码
- * HTML事件属性可以调用JavaScript函数
- * 可以将自己的事件处理函数分配给HTML元素
- * 可以阻止事件的发送或处理
- * 还有更多.....

语法

* element.addEventListener(event, function, useCapture);

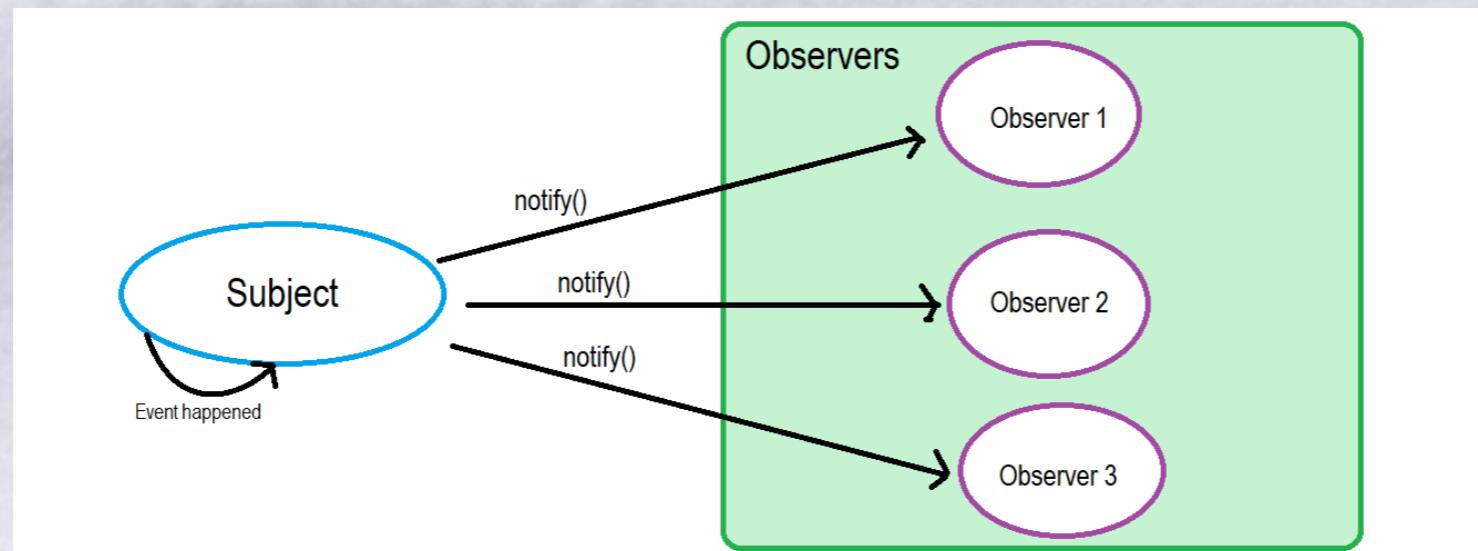
- * 第一个参数是事件类型 (如"click" 或"mousedown").
- * 第二个参数是我们想在事件发生时调用的函数.
- * 第三个参数是一个布尔值, 指定是使用事件冒泡还是使用事件捕获, 是可选参数.

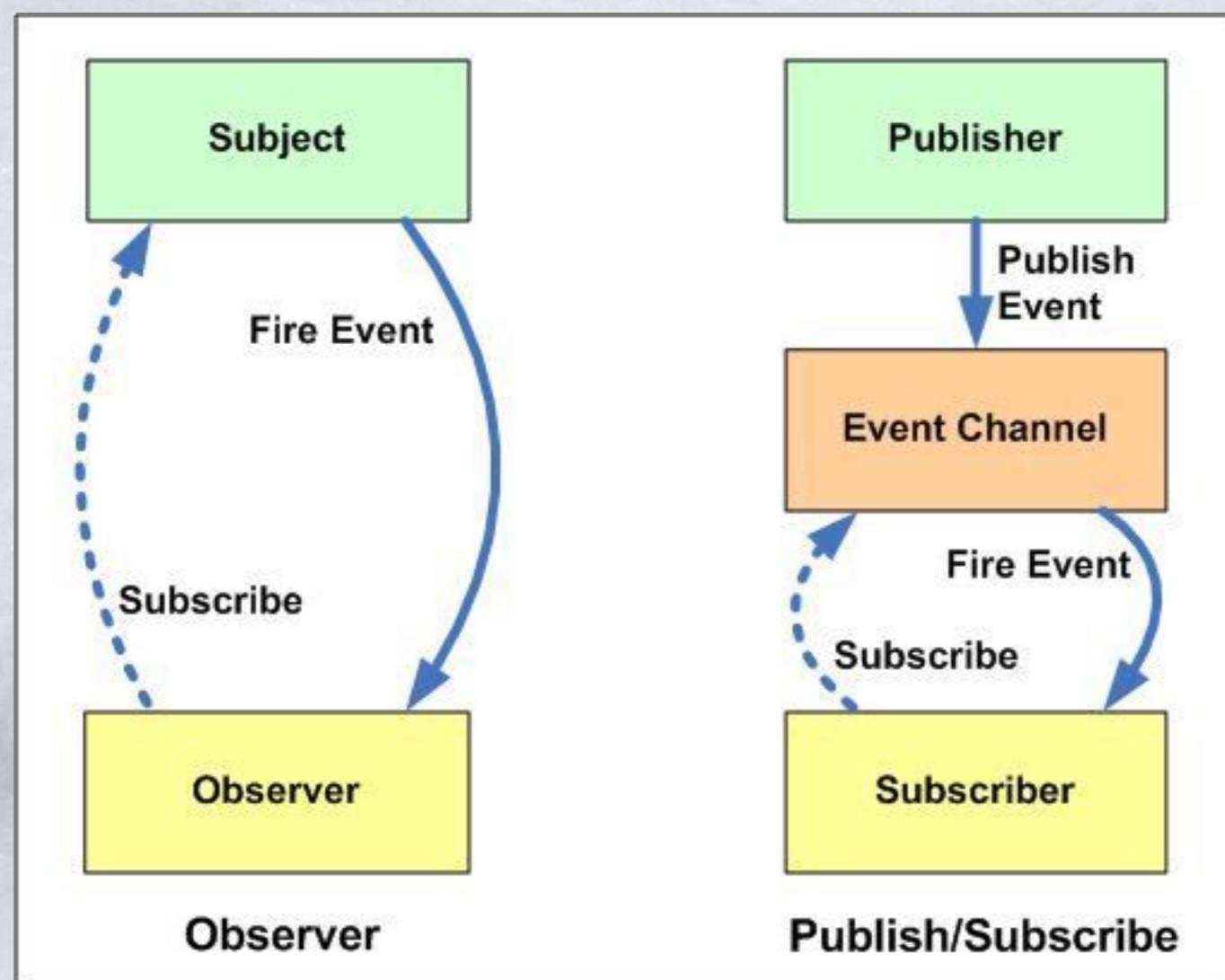
* removeEventListener()方法删除已由addEventListener()方法注册的事件处理程序

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
document.getElementById("myDiv").addEventListener("click",
myFunction, true);
element.removeEventListener("mousemove", myFunction);
```

观察者模式

- ✿ 观察者模式是一种软件设计模式，在这种模式中，一个称为主题的对象维护一个名为观察者的依赖项列表，通常通过调用它们的一个方法自动通知其任何状态变化
- ✿ 主要用于实现分布式事件处理系统
- ✿ 观察者模式也是我们熟悉的模型-视图-控制器(MVC)体系结构模式中的关键部分。观察者模式在许多编程库和系统中实现，包括几乎所有的GUI工具包。
- ✿ 发布者/订阅者模式的子集





特征

好处:

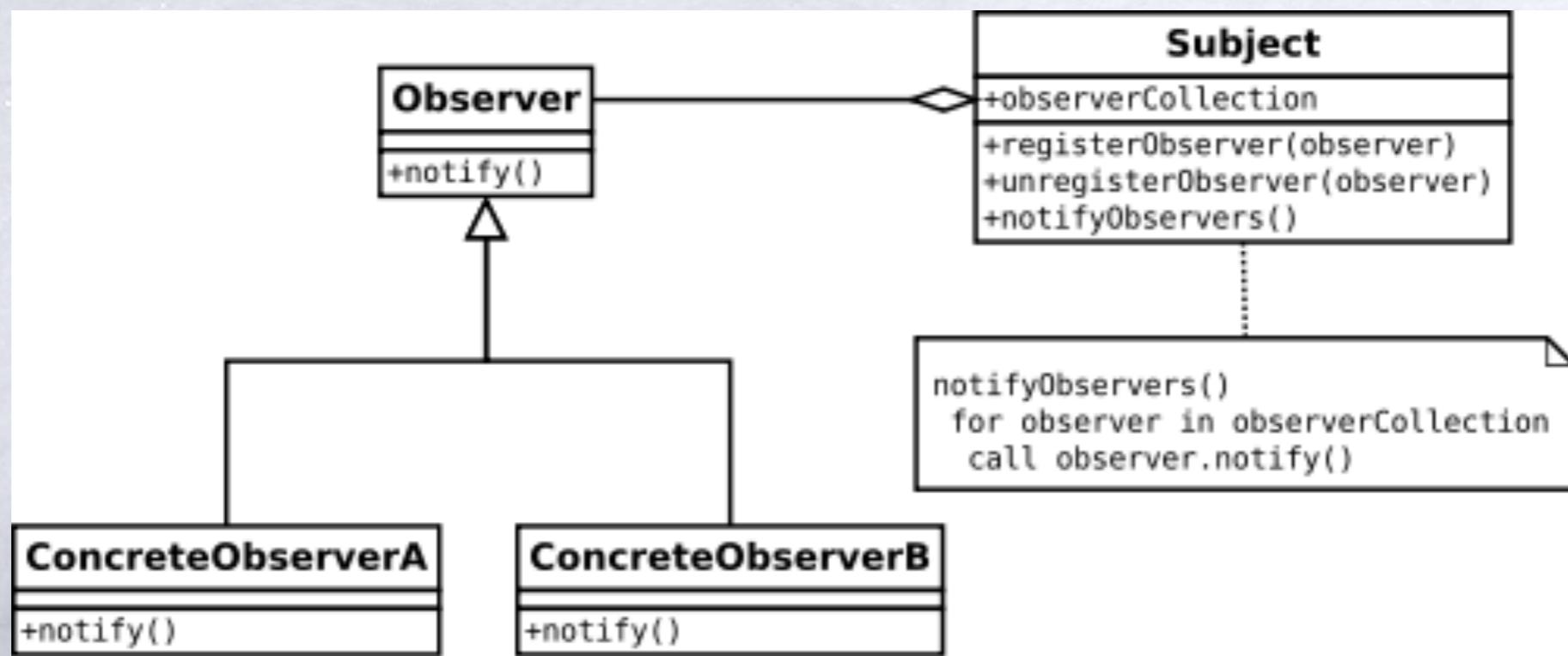
- * 主题与观察者之间的抽象耦合
- * 支持广播通信

注意:

- * 观察者模式会导致内存泄漏，即所谓的失效监听器问题，

观察者模式

* 事件使得一个主题可能有多个观察者队列

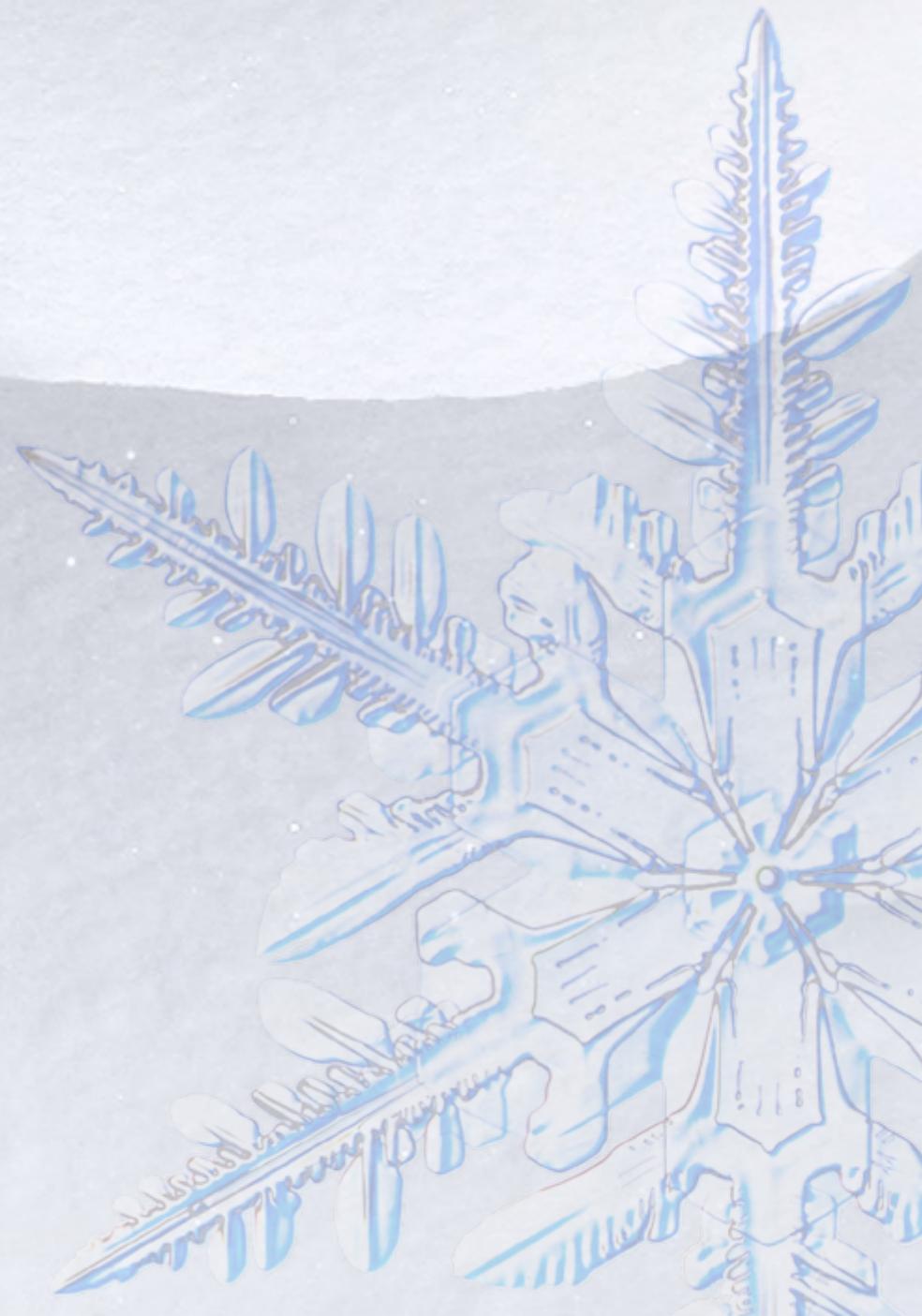


用jQuery方式附加事件处理程序

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
    hiddenBox.show();
});
```

目录

- ❄ Javascript 事件
- ❄ 事件类型
- ❄ 事件处理模型



可能发生的不同事件

* 在 Web 中，事件在浏览器窗口中被触发并且通常被绑定到窗口内部的特定部分 — 可能是一个元素、一系列元素、被加载到这个窗口的 HTML 代码或者是整个浏览器窗口。举几个可能发生的不同事件：

- * 用户在某个元素上点击鼠标或悬停光标。
- * 用户在键盘中按下某个按键。
- * 用户调整浏览器的大小或者关闭浏览器窗口。
- * 一个网页停止加载。
- * 提交表单。
- * 播放、暂停、关闭视频。
- * 发生错误。



DOM 2 事件类型

* 用户界面 (UI) 事件:

- * DOMFocusIn, DOMFocusOut, DOMActivate

* 鼠标事件:

- * click, mousedown, mouseup, mouseover, mousemove, mouseout

* 键盘事件: (not in DOM 2, but will in DOM 3)

* 变动事件:

- * DOMSubtreeModified, DOMNodeInserted, ...

* HTML事件:

- * load, unload, abort, error, select, change, submit, reset, focus, blur, resize, scroll

HTML5新事件

❄️ **audio, video**

- * canplay, playing, suspend,.....

❄️ **drag/drop**

❄️ **history**

❄️ **new form events**

- * invalid

❄️ **offline,online.....**

❄️ **message**

触屏和移动设备事件

- ❄ 功能强大的移动设备，特别是带有触摸屏的移动设备被广泛采用，要求创建新的事件类别。
- ❄ 在许多情况下，触屏事件映射到传统事件类型，如单击和滚动。但并非所有触屏UI的交互都模仿鼠标，也不是所有的触摸都可以被视为鼠标事件。
 - * `gesturestart`, `gestureend`

事件类型的使用

问题:事件是棘手的，有跨浏览器的不兼容性方面的原因:

- * 模糊W3C事件规范;
- * IE不符合网络标准;
- * 等等

事件对象

有时候在事件处理函数内部，您可能会看到一个固定指定名称的参数，例如event, evt或简单的e。这被称为事件对象，它被自动传递给事件处理函数，以提供额外的功能和信息。事件对象具有以下属性/方法：

```
function name(event) {  
    // an event handler function ...  
}
```

JS

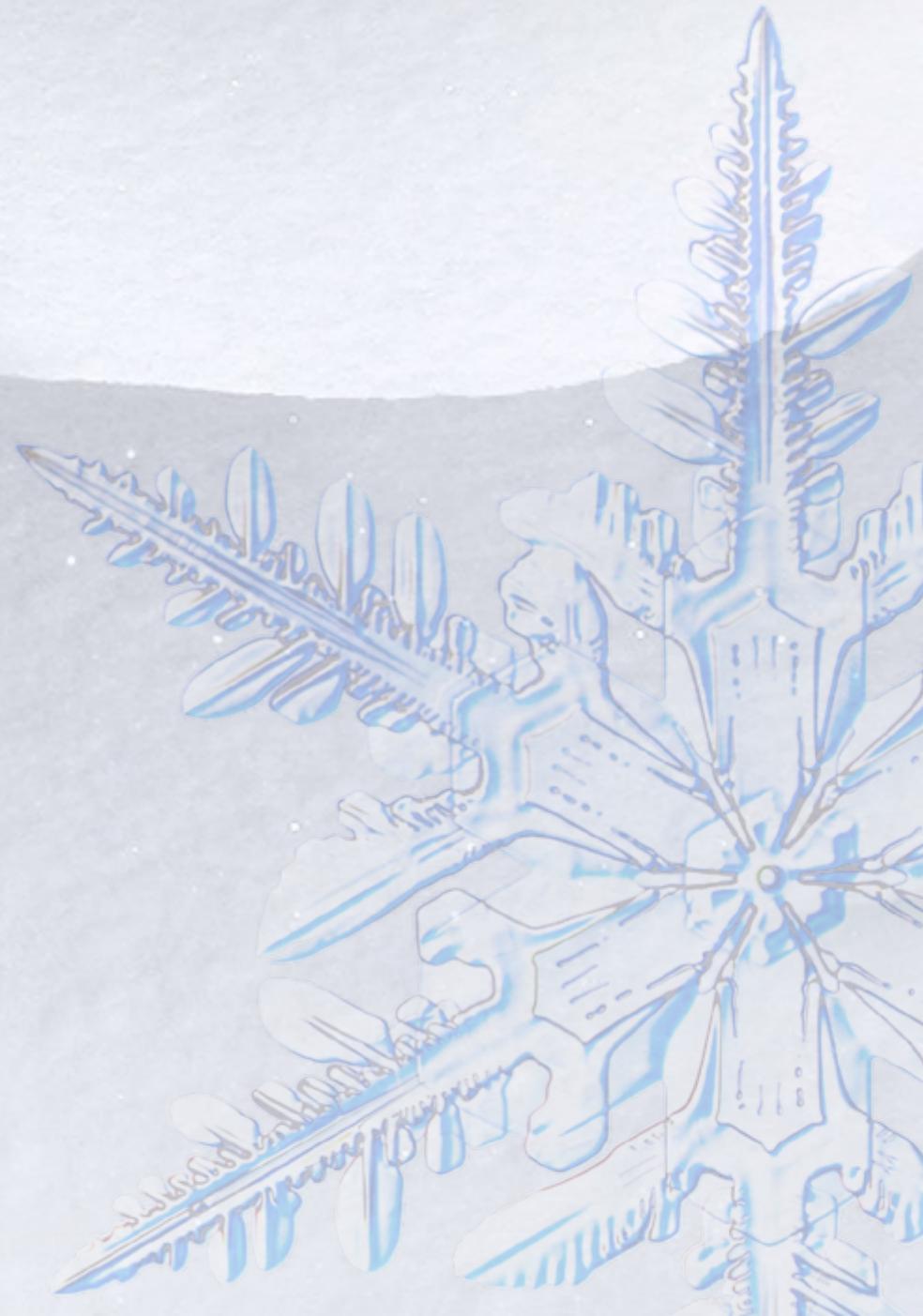
method/property name	description
type	Returns the name of the event, such as "click" or "mousedown"
currentTarget	Returns the element whose event listeners triggered the event
preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur
stopPropagation()	Prevents further propagation of an event during event flow

鼠标事件

Event	Description	DOM
<u>onclick</u>	The event occurs when the user clicks on an element	2
<u>ondblclick</u>	The event occurs when the user double-clicks on an element	2
<u>onmousedown</u>	The event occurs when the user presses a mouse button over an element	2
<u>onmouseenter</u>	The event occurs when the pointer is moved onto an element	2
<u>onmouseleave</u>	The event occurs when the pointer is moved out of an element	2
<u>onmousemove</u>	The event occurs when the pointer is moving while it is over an element	2
<u>onmouseover</u>	The event occurs when the pointer is moved onto an element, or onto one of its children	2
<u>onmouseout</u>	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
<u>onmouseup</u>	The event occurs when a user releases a mouse button over an element	2

目录

- ❄ Javascript 事件
- ❄ 事件类型
- ❄ 事件处理模型



DOM 0

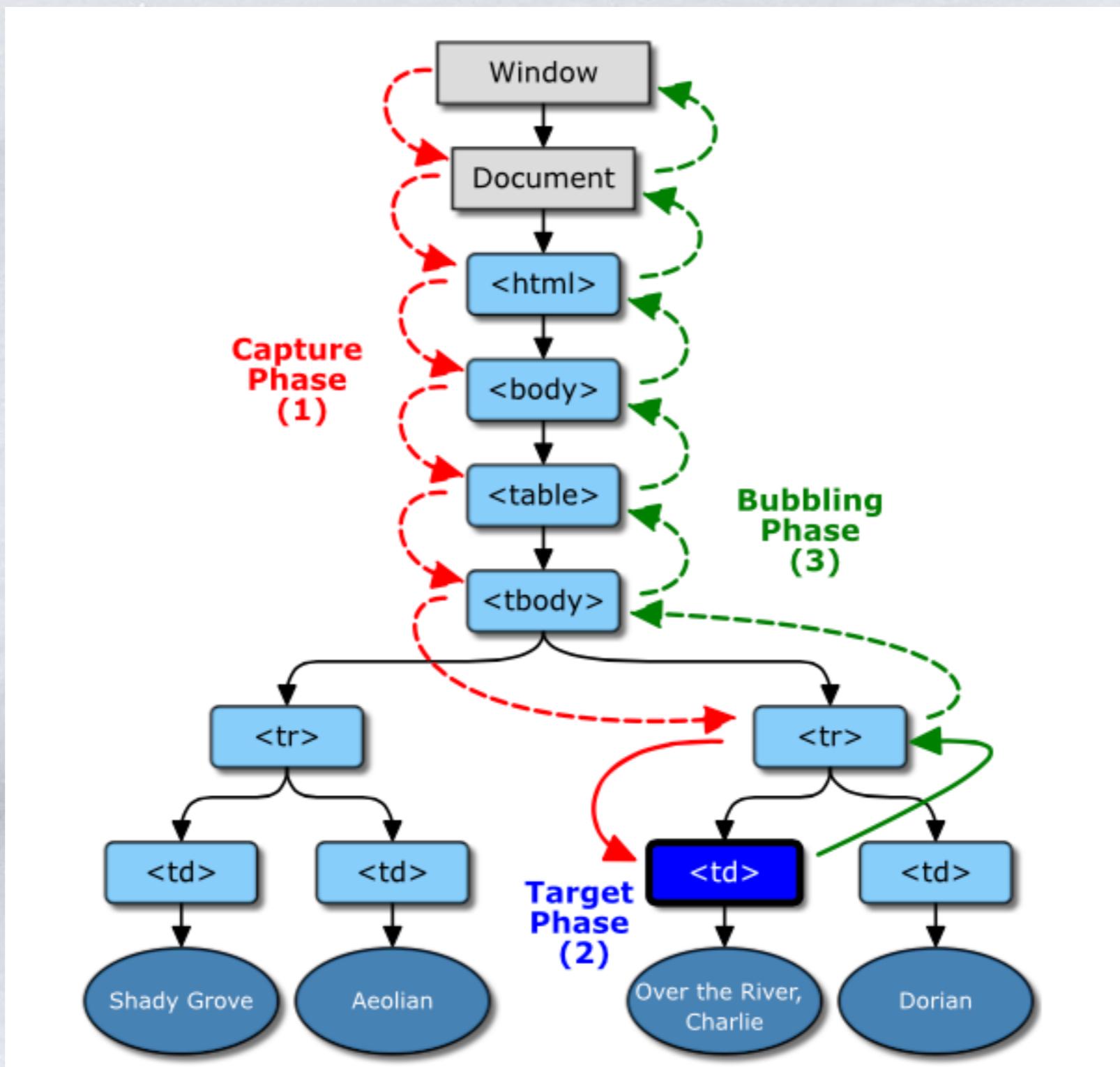
* 此事件处理模型是由Netscape Navigator引入的，到2005年仍然是跨浏览器最多的模型。有两种模型类型：

- * 内联模型：事件处理程序作为元素的属性添加。
- * 传统模型：可以通过脚本添加/删除事件处理程序。与内联模型一样，每个事件只能注册一个事件处理程序。通过将处理程序名称分配给元素对象的事件属性来添加事件。要删除事件处理程序，只需将属性设置为null。

```
<p>Hey <a href="http://  
www.example.com"  
onclick="triggerAlert('Joe'); return  
false;">Joe</a>!</p>  
  
<script>  
    function triggerAlert(name) {  
        window.alert("Hey " + name);  
    }  
</script>
```

```
<script>  
    var triggerAlert = function () {  
        window.alert("Hey Joe");  
  
        // Assign an event handler  
        document.onclick = triggerAlert;  
        // Assign another event handler  
        window.onload = triggerAlert;  
        // Remove the event handler that was just assigned  
        window.onload = null;  
    }  
</script>
```

DOM 事件流



事件处理阶段

* 在DOM兼容浏览器中，事件流分为3个阶段：

- * 捕获阶段：事件从Document节点自上而下向目标节点传播的阶段；
- * 目标阶段：真正的目标节点正在处理事件的阶段；
- * 冒泡阶段：事件从目标节点自下而上向Document节点传播的阶段。

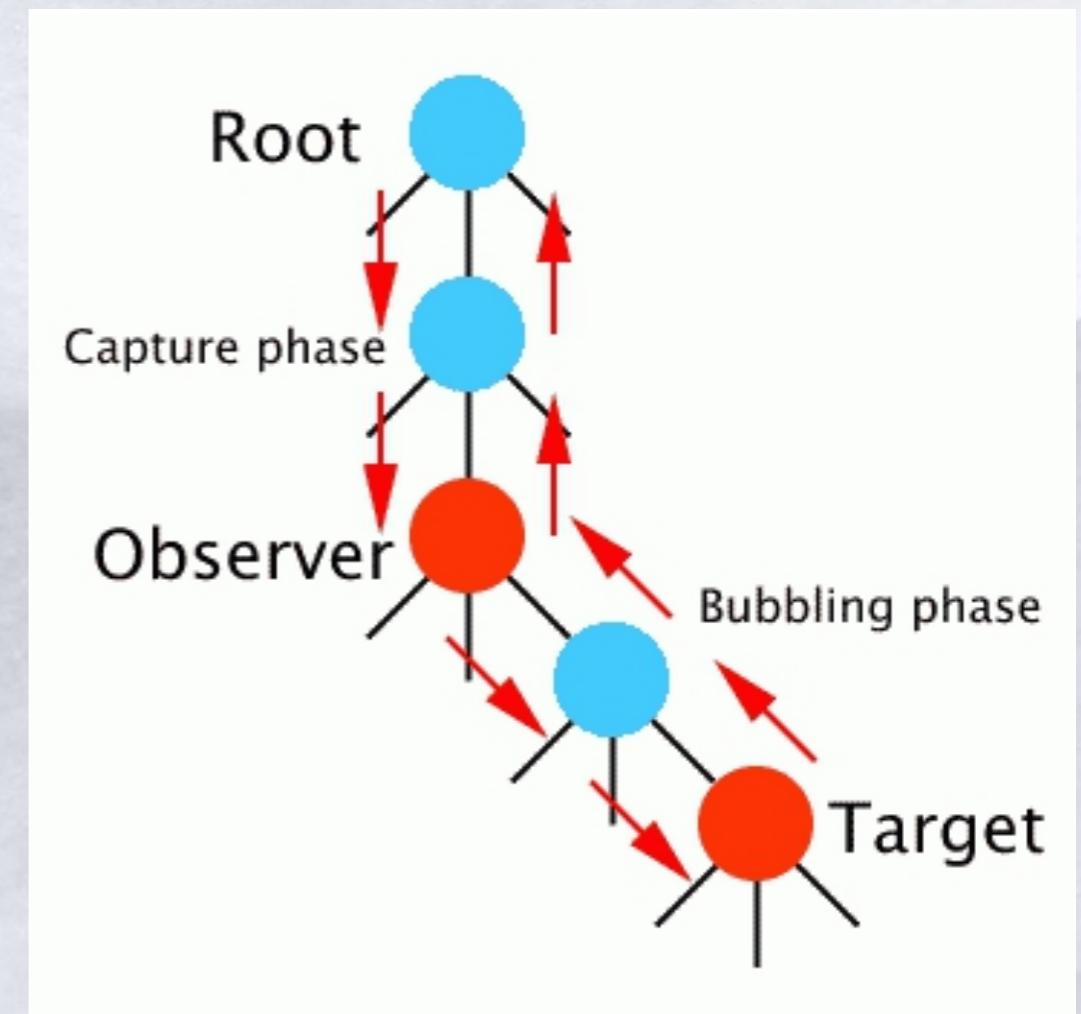
* 在现代浏览器中，默认情况下，所有事件处理程序都在冒泡阶段进行注册。

事件流

- * 每个事件都有一个目标节点，可以通过事件访问该目标

```
element.onclick = handler(e);
function handler(e){
  if(!e) var e = window.event;
  // e refers to the event
  // see detail of event
  var original = e.eventTarget;
}
```

- * 每个事件都起源于浏览器，并传递给DOM
- * 职责链模式



DOM 2

Name	Description	Argument type	Argument name
addEventListener	Allows the registration of event listeners on the event target.	DOMString	type
		EventListener	listener
		boolean	useCapture
removeEventListener	Allows the removal of event listeners from the event target.	DOMString	type
		EventListener	listener
		boolean	useCapture
dispatchEvent	Allows sending the event to the subscribed event	Event	evt

浏览器兼容性

* 表中的数字指定了完全支持这些方法的第一个浏览器版本

Method	Chrome	IE	Firefox	Safari	Opera
addEventListener()	1.0	9.0	1.0	1.0	7.0
removeEventListener()	1.0	9.0	1.0	1.0	7.0

阻止默认行为

❄ 停止事件的传播

- * event.stopPropagation()
- * cancelBubble=true (IE某些版本)

❄ 禁止默认行为

- * event.preventDefault()

*

```
video.onclick = function(e) {  
    e.stopPropagation();  
    video.play();  
};
```

重写传统模型中使用的示例

```
<script>
var heyJoe = function () {
    window.alert("Hey Joe!");
}

// Add an event handler
document.addEventListener( "click", heyJoe, true ); // capture
phase

// Add another event handler
window.addEventListener( "load", heyJoe, false ); // bubbling
phase

// Remove the event handler just added
window.removeEventListener( "load", heyJoe, false );
</script>
```

Microsoft-specific 模型

❄️微软直到Internet Explorer 8才遵循W3C模型，因为它自己的模型是在W3C标准批准之前创建的。Internet Explorer 9遵循DOM3事件，Internet Explorer 11删除了对微软特定模型的支持。

Name	Description	Argument type	Argument name
attachEvent	Similar to W3C's addEventListener method.	String	sEvent
		Pointer	fpNotify
detachEvent	Similar to W3C's removeEventListener method.	String	sEvent
		Pointer	fpNotify
fireEvent	Similar to W3C's dispatchEvent method.	String	sEvent
		Event	oEventObject

IE曾经的特定操作

- ❄ 为了防止事件冒泡，开发人员必须设置事件的cancelBubble属性。
- ❄ 为了防止事件的默认动作被调用，开发人员必须设置事件的“returnValue”属性。
- ❄ 避免使用！！！

跨浏览器的解决方式示例

```
var x = document.getElementById("myBtn");
if (x.addEventListener) {
    // For all major browsers, except IE 8 and earlier
    x.addEventListener("click", myFunction);
} else if (x.attachEvent) { // For IE 8 and earlier versions
    x.attachEvent("onclick", myFunction);
}
```

事件处理程序的绑定

内联

- * ``

传统:

- * `element.onclick = myFunction;`

DOM 2:

- * `element.addEventListener("click", myFunction);`

IE: (evil enough!)

- * `element.attachEvent('onclick', myFunction);`

JQuery, Prototype...and so on:

- * `jQuery.on()`
- * `Event.observe('target', 'click', myFunction);`

推荐阅读

- ❄ JavaScript: The Good Parts, by Douglas Crockford.
- ❄ Programming Javascript Applications.
- ❄ JAVASCRIPT设计模式, Ross Harmes & Dustin Diaz. 人民邮电出版社.
- ❄ <http://w3techs.com>
- ❄ <http://kangax.github.io/es5-compat-table/es6/>
- ❄ <https://babeljs.io/docs/learn-es2015/>
- ❄ <https://github.com/lukehoban/es6features>

Thanks!!!

