

DOM 事件与事件委托

JS 编程接口

版权声明

本内容版权属杭州饥人谷教育科技有限公司（简称饥人谷）所有。

任何媒体、网站或个人未经本网协议授权不得转载、链接、转贴，
或以其他方式复制、发布和发表。

已获得饥人谷授权的媒体、网站或个人在使用时须注明「资料来源：饥人谷」。

对于违反者，饥人谷将依法追究责任。

联系方式

如果你想要购买本课程
请微信联系 **xiedaimala02** 或 **xiedaimala03**

如果你发现有人盗用本课程
请微信联系 **xiedaimala02** 或 **xiedaimala03**

点击事件

从这个东西开始研究

代码

- **HTML**

```
<div class=爷爷>  
  <div class=爸爸>  
    <div class=儿子>  
      文字  
    </div>  
  </div>  
</div>  
✓ 即 .爷爷>.爸爸>.儿子  
✓ 给三个div分别添加事  
件监听 fnYe / fnBa / fnEr
```

- **提问1：点击了谁**

- ✓ 点击文字，算不算点击儿子？
- ✓ 点击文字，算不算点击爸爸？
- ✓ 点击文字，算不算点击爷爷？
- ✓ 答案：都算

- **提问2：调用顺序**

- ✓ 点击文字，最先调用 fnYe / fnBa / fnEr 中的哪一个函数？
- ✓ 答案：都行。
- ✓ IE 5 认为先调 fnEr，网景认为先调 fnYe，然后掐上了
- ✓ 最后闹到了 W3C

和事佬 W3C

- 2002 年，W3C 发布标准
 - ✓ 文档名为 DOM Level 2 Events Specification
 - ✓ 规定浏览器应该同时支持两种调用顺序
 - ✓ 首先按爷爷=>爸爸=>儿子顺序看有没有函数监听
 - ✓ 然后按儿子=>爸爸=>爷爷顺序看有没有函数监听
 - ✓ 有监听函数就调用，并提供事件信息，没有就跳过

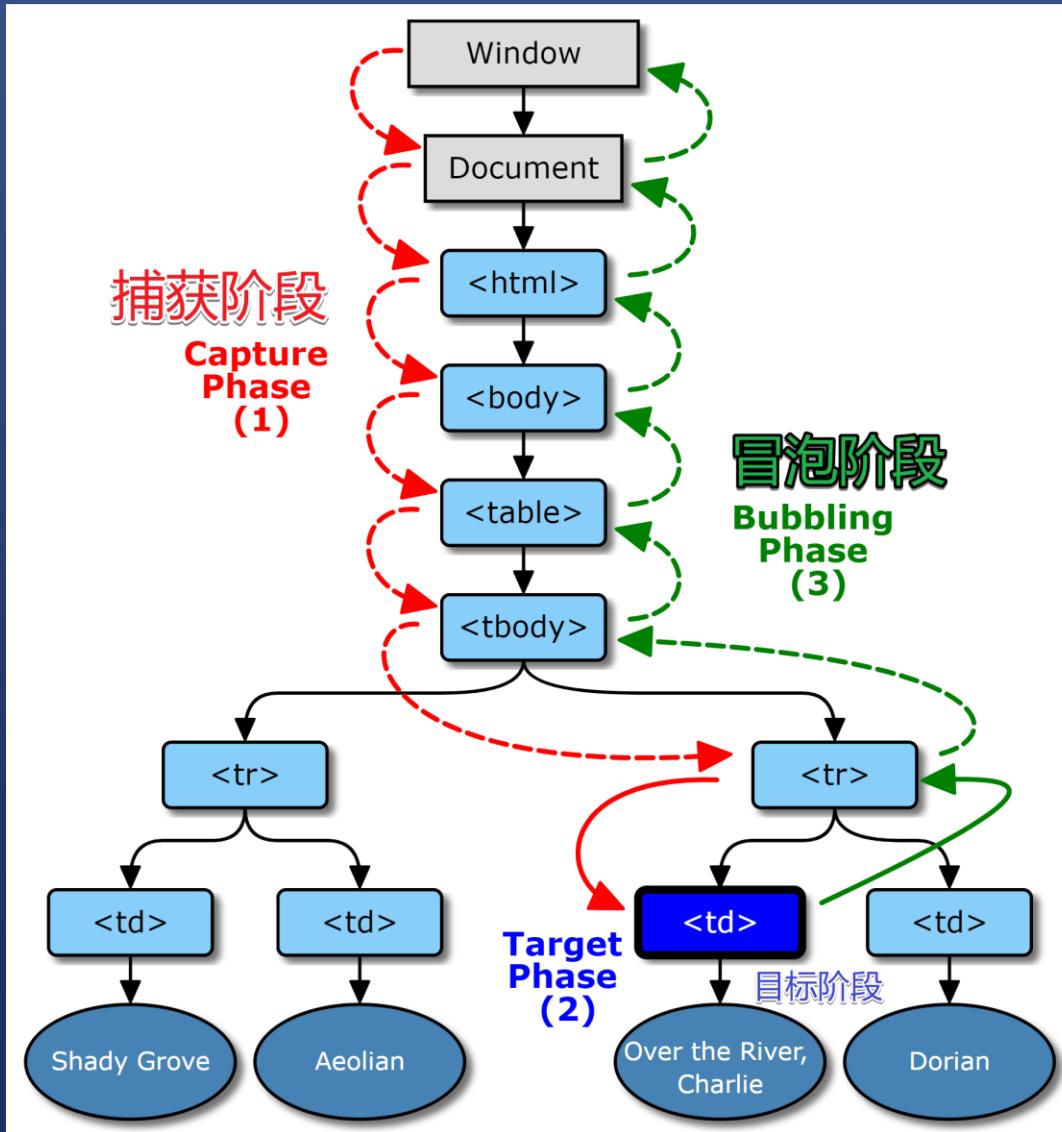
• 术语

- ✓ 从外向内找监听函数，叫事件捕获
- ✓ 从内向外找监听函数，叫事件冒泡

疑问：那岂不是 fnYe / fnBa / fnEr 都调用两次？非也！

开发者自己选择把 fnYe 放在捕获阶段还是放在冒泡阶段

示意图



addEventListener

- 事件绑定 API

- ✓ IE 5*: baba.attachEvent('onclick', fn) // 冒泡
- ✓ 网景: baba.addEventListener('click',fn) // 捕获
- ✓ W3C: baba.addEventListener('click', fn, bool)

- 如果 bool 不传或为 falsy

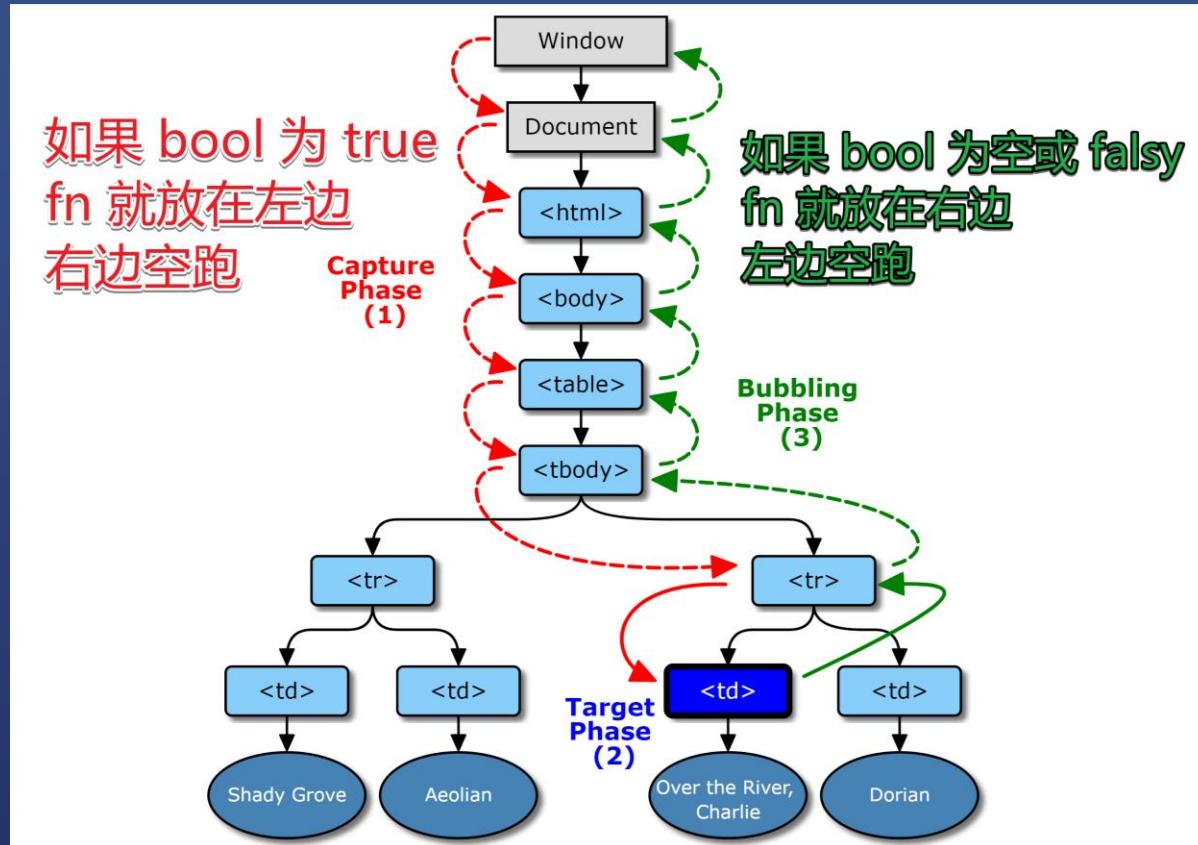
- ✓ 就让 fn 走冒泡，即当浏览器在冒泡阶段发现 baba 有 fn 监听函数，就会调用 fn，并提供事件信息

- 如果 bool 为 true

- ✓ 就让 fn 走捕获，即当浏览器在捕获阶段发现 baba 有 fn 监听函数，就会调用 fn，并提供事件信息

*2020 之后，**永远不要学习关于 IE 5、6、7、8、9、10、11 的知识，用到再搜**

你可以选择把 fn 放在哪边



代码示例

不安全 | js.jirengu.com/zirozujedu/1/edit?html,js,output

文件 库 Share HTML CSS JavaScript Console Output

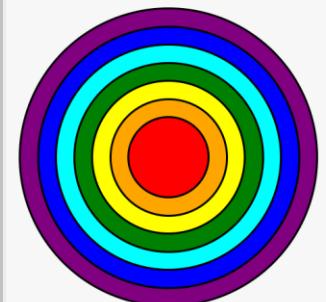
HTML ▾

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>JS Bin</title>
6 </head>
7 <body>
8   <div class="level1 x">
9     <div class="level2 x">
10    <div class="level3 x">
11      <div class="level4 x">
12        <div class="level5 x">
13          <div class="level6 x">
14            <div class="level7 x">
15
16            </div>
17          </div>
18        </div>
19      </div>
20    </div>
21  </div>
22 </body>
23 </html>
```

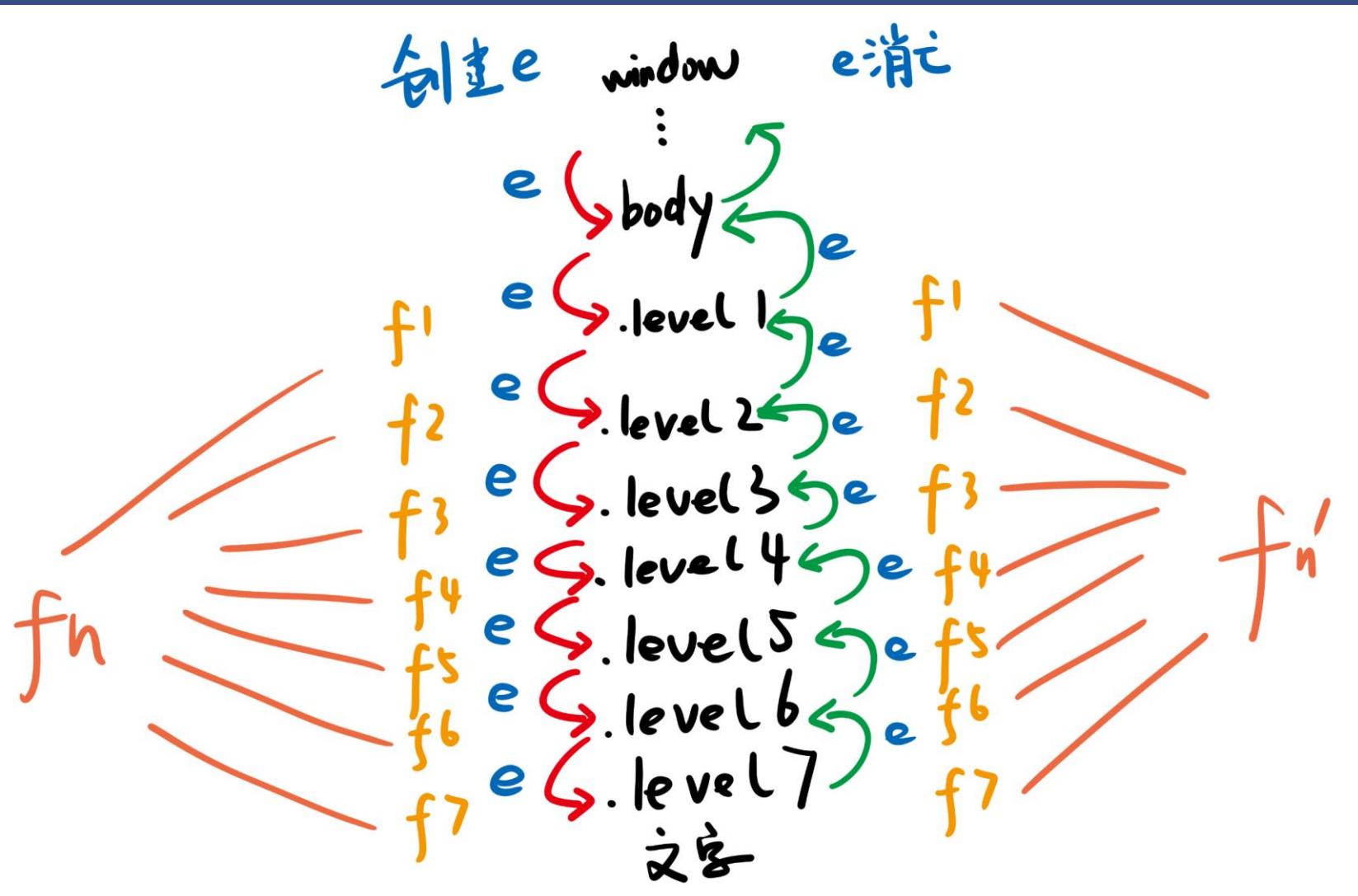
JavaScript ▾

```
1 const level1 = document.querySelector('.level1')
2 const level2 = document.querySelector('.level2')
3 const level3 = document.querySelector('.level3')
4 const level4 = document.querySelector('.level4')
5 const level5 = document.querySelector('.level5')
6 const level6 = document.querySelector('.level6')
7 const level7 = document.querySelector('.level7')
8
9 let n = 1
10
11 level1.addEventListener('click', (e)=>{
12   const t = e.currentTarget
13   setTimeout(()=>{
14     t.classList.remove('x')
15   },n*1000)
16   n+=1
17 })
18 level2.addEventListener('click', (e)=>{
19   const t = e.currentTarget
20   setTimeout(()=>{
21     t.classList.remove('x')
22   },n*1000)
23   n+=1
24 })
25 level3.addEventListener('click', (e)=>{
26   const t = e.currentTarget
```

Output 736px



代码图解



小结

- **两个疑问**
 - ✓ 儿子被点击了，算不算点击老子？
 - ✓ 那么先调用老子的函数还是先调用儿子的函数？
- **捕获与冒泡**
 - ✓ 捕获说先调用爸爸的监听函数
 - ✓ 冒泡说先调用儿子的监听函数
- **W3C 事件模型**
 - ✓ 先捕获（先爸爸=>儿子）再冒泡（再儿子=>爸爸）
 - ✓ 注意 e 对象被传给所有监听函数
 - ✓ 事件结束后，e 对象就不存在了

target v.s. currentTarget

- **区别**

- ✓ e.target - 用户操作的元素
- ✓ e.currentTarget - 程序员监听的元素
- ✓ this 是 e.currentTarget，我个人不推荐使用它

- **举例**

- ✓ div > span{文字}，用户点击文字
- ✓ e.target 就是 span
- ✓ e.currentTarget 就是 div

一个特例

- **背景**

- ✓ 只有一个 div 被监听 (不考虑父子同时被监听)
- ✓ fn 分别在捕获阶段和冒泡阶段监听 click 事件
- ✓ 用户点击的元素就是开发者监听的

- **代码**

- ✓ `div.addEventListener('click', f1)`
- ✓ `div.addEventListener('click', f2, true)`
- ✓ 请问, f1 先执行还是 f2 先执行?
- ✓ 如果把两行调换位置后, 请问哪个先执行?
- ✓ 错误答案: f2 先执行
- ✓ 正确答案: 谁先监听谁先执行
- ✓ 总结: 这是一个特例

取消冒泡

- 捕获不可取消，但冒泡可以
 - ✓ e.stopPropagation() 可中断冒泡，浏览器不再向上走
 - ✓ 通俗来说：有人打我，我自己解决，别告诉我老子
 - ✓ 一般用于封装某些独立的组件

不可取消冒泡

- **有些事件不可取消冒泡**

- ✓ MDN 搜索 scroll event，看到 Bubbles 和 Cancelable
- ✓ Bubbles 的意思是该事件是否冒泡
- ✓ Cancelable 的意思是开发者是否可以取消冒泡
- ✓ 推荐看 MDN 英文版，中文版内容不全

The `scroll` event fires when the document view or an element has been scrolled.

Bubbles

Yes

Cancelable

No

Interface

`Event`

Event handler property

`onscroll`

插曲：如何阻止滚动

- scroll 事件不可取消冒泡
 - ✓ 阻止 scroll 默认动作没用，因先有滚动才有滚动事件
 - ✓ 要阻止滚动，可阻止 wheel 和 touchstart 的默认动作
 - ✓ 注意你需要找准滚动条所在的元素，[示例](#)
 - ✓ 但是滚动条还能用，可用 CSS 让滚动条 width: 0
- CSS 也行
 - ✓ 使用 overflow: hidden 可以直接取消滚动条
 - ✓ 但此时 JS 依然可以修改 scrollTop

小结

- **target 和 currentTarget**
 - ✓ 一个是用户点击的，一个是开发者监听的
- **取消冒泡**
 - ✓ `e.stopPropagation()`
- **事件的特性**
 - ✓ `Bubbles` 表示是否冒泡
 - ✓ `Cancelable` 表示是否支持开发者取消冒泡
 - ✓ 如 `scroll` 不支持取消冒泡
- **如何禁用滚动**
 - ✓ 取消特定元素的 `wheel` 和 `touchstart` 的默认动作

自定义事件

- 浏览器自带事件
 - ✓ 一共100多种事件，列表在 MDN 上
- 提问
 - ✓ 开发者能不能在自带事件之外，自定义一个事件
 - ✓ 答案：可以，见示例

The screenshot shows a browser-based code editor interface with three main panels: HTML, JavaScript, and Output.

- HTML Panel:**

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>JS Bin</title>
6 </head>
7 <body>
8   <div id="div1">
9     <button id="button1">
10    点击触发 frank 事件
11  </button>
12 </div>
13 </body>
14 </html>
```
- JavaScript Panel:**

```
1 button1.addEventListener('click', ()=>{
2   const event = new CustomEvent("frank",
3     {"detail":{name:'frank', age: 18}})
4   button1.dispatchEvent(event)
5
6 button1.addEventListener('frank', (e)=>{
7   console.log('frank')
8   console.log(e)
9 })
```
- Output Panel:**

```
点击触发 frank 事件
```

事件委托

- **场景一**

- ✓ 你要给 100 个按钮添加点击事件，咋办？
- ✓ 答：监听这 100 个按钮的祖先，等冒泡的时候判断 target 是不是这 100 个按钮中的一个

- **场景二**

- ✓ 你要监听目前不存在的元素的点击事件，咋办？
- ✓ 答：监听祖先，等点击的时候看看是不是我想要监听的元素即可

- **优点**

- ✓ 省监听数（内存）
- ✓ 可以监听动态元素

封装事件委托

- **要求**

- ✓ 写出这样一个函数 `on('click', '#testDiv', 'li', fn)`
- ✓ 当用户点击 `#testDiv` 里的 `li` 元素时，调用 `fn` 函数
- ✓ 要求用到事件委托

- **答案一**

- ✓ 判断 `target` 是否匹配 '`li`'

- **答案二**

- ✓ 递归判断 `target` / `target`的爸爸 / `target`的爷爷

- **整合进 jQuery**

- ✓ 有兴趣可以自己实现 `$('#xxx').on('click', 'li', fn)`

JS 支持事件吗

- 答

- ✓ 支持，也不支持。本节课讲的 DOM 事件不属于 JS 的功能，术语浏览器提供的 DOM 的功能
- ✓ JS 只是调用了 DOM 提供的 addEventListener 而已

- 追问

- ✓ 如何当 JS 支持事件？请手写一个事件系统。
- ✓ 目前大家的水平还写不出来，可以先思考一段时间。

再见

希望你对 DOM 事件，有一个完整的了解