事件高级

目录 Contents

1. 注册事件（绑定事件）

1.1 注册事件概述

给元素添加事件，称为注册事件或者绑定事件。

注册事件有两种方式：传统方式和方法监听注册方式

传统注册方式

- 利用 on 开头的事件 onclick
- <button onclick="alert('hi~')"></button>
- btn.onclick = function() {}
- 特点： 注册事件的**唯一性**
- 同一个元素同一个事件只能设置一个处理函数，最后注册的处理函数将会覆盖前面注册的处理函数

方法监听注册方式

- w3c 标准 推荐方式
- addEventListener() 它是一个方法
- IE9 之前的 IE 不支持此方法，可使用 attachEvent() 代替
- 特点：同一个元素同一个事件可以注册多个监听器
- 按注册顺序依次执行

1. 注册事件（绑定事件）

1.2 addEventListener 事件监听方式

```
eventTarget.addEventListener(type, listener[, useCapture])
```

`eventTarget.addEventListener()`方法将指定的监听器注册到 `eventTarget`（目标对象）上，当该对象触发指定的事件时，就会执行事件处理函数。

该方法接收三个参数：

- type：事件类型字符串，比如 click 、mouseover ，注意这里不要带 on
- listener：事件处理函数，事件发生时，会调用该监听函数
- useCapture：可选参数，是一个布尔值，默认是 false。学完 DOM 事件流后，我们再进一步学习

1. 注册事件（绑定事件）

1.3 attachEvent 事件监听方式

```
eventTarget.attachEvent(eventNameWithOn, callback)
```

`eventTarget.attachEvent()`方法将指定的监听器注册到 `eventTarget`（目标对象） 上，当该对象触发指定的事件时，指定的回调函数就会被执行。

该方法接收两个参数：

- eventNameWithOn：事件类型字符串，比如 onclick 、onmouseover ，这里要带 on
- callback： 事件处理函数，当目标触发事件时回调函数被调用

注意：IE8 及早期版本支持

1. 注册事件（绑定事件）

1.4 注册事件兼容性解决方案

```
function addEventListener(element, eventName, fn) {
      // 判断当前浏览器是否支持 addEventListener 方法
      if (element.addEventListener) {
        element.addEventListener(eventName, fn);  // 第三个参数 默认是false
      } else if (element.attachEvent) {
        element.attachEvent('on' + eventName, fn);
      } else {
        // 相当于 element.onclick = fn;
        element['on' + eventName] = fn;
}
```

兼容性处理的原则： 首先照顾大多数浏览器，再处理特殊浏览器

目录 Contents

2. 删除事件（解绑事件）

2.1 删除事件的方式

1. 传统注册方式

```
eventTarget.onclick = null;
```

2. 方法监听注册方式

① `eventTarget.removeEventListener(type, listener[, useCapture]);`

② `eventTarget.detachEvent(eventNameWithOn, callback);`

2. 删除事件（解绑事件）

2.2 删除事件兼容性解决方案

```
function removeEventListener(element, eventName, fn) {
    // 判断当前浏览器是否支持 removeEventListener 方法
    if (element.removeEventListener) {
      element.removeEventListener(eventName, fn);  // 第三个参数 默认是false
    } else if (element.detachEvent) {
      element.detachEvent('on' + eventName, fn);
    } else {
      element['on' + eventName] = null;
}
```

目录 Contents

3. DOM 事件流

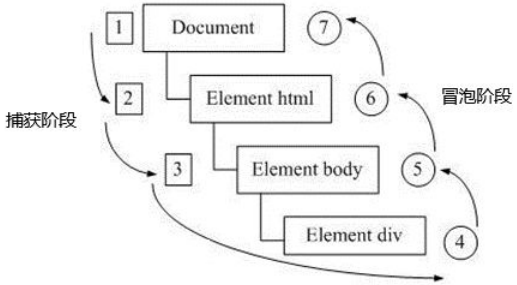
事件流描述的是从页面中接收事件的顺序。

事件发生时会在元素节点之间按照特定的顺序传播，这个传播过程即 DOM 事件流。

比如我们给一个div 注册了点击事件：

DOM 事件流分为3个阶段：

1. 捕获阶段
2. 当前目标阶段
3. 冒泡阶段

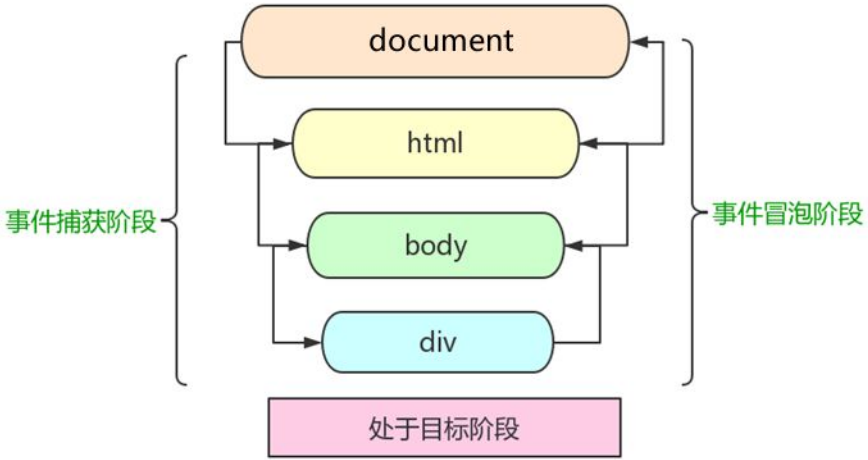


DOM事件流

- 事件冒泡： IE 最早提出，事件开始时由最具体的元素接收，然后逐级向上传播到到 DOM 最顶层节点的过程。
- 事件捕获： 网景最早提出，由 DOM 最顶层节点开始，然后逐级向下传播到到最具体的元素接收的过程。

3. DOM 事件流

我们向水里面扔一块石头，首先它会有一个下降的过程，这个过程就可以理解为从最顶层向事件发生的最具体元素（目标点）的捕获过程；之后会产生泡泡，会在最低点（ 最具体元素）之后漂浮到水面上，这个过程相当于事件冒泡。



3. DOM 事件流

事件发生时会在元素节点之间按照特定的顺序传播，这个传播过程即 DOM 事件流。

注意

1. JS 代码中只能执行捕获或者冒泡其中的一个阶段。
2. onclick 和 attachEvent 只能得到冒泡阶段。
3. addEventListener(type, listener[, useCapture])第三个参数如果是 true，表示在事件捕获阶段调用事件处理程序；如果是 false（不写默认就是false），表示在事件冒泡阶段调用事件处理程序。
4. 实际开发中我们很少使用事件捕获，我们更关注事件冒泡。
5. 有些事件是没有冒泡的，比如 onblur、onfocus、onmouseenter、onmouseleave
6. 事件冒泡有时候会带来麻烦，有时候又会帮助很巧妙的做某些事件，我们后面讲解。

目录 Contents

4. 事件对象

4.1 什么是事件对象

```
eventTarget.onclick = function(event) {}
eventTarget.addEventListener('click', function(event) {})
// 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
```

官方解释：event 对象代表事件的状态，比如键盘按键的状态、鼠标的位置、鼠标按钮的状态。

简单理解：事件发生后，跟事件相关的一系列信息数据的集合都放到这个对象里面，这个对象就是事件对象 event，它有很多属性和方法。

比如：

1. 谁绑定了这个事件。

2. 鼠标触发事件的话，会得到鼠标的相关信息，如鼠标位置。

3. 键盘触发事件的话，会得到键盘的相关信息，如按了哪个键。

4. 事件对象

4.2 事件对象的使用语法

```
eventTarget.onclick = function(event) {
    // 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
}
eventTarget.addEventListener('click', function(event) {
  // 这个 event 就是事件对象，我们还喜欢的写成 e 或者 evt
})
```

这个 event 是个形参，系统帮我们设定为事件对象，不需要传递实参过去。

当我们注册事件时， event 对象就会被系统自动创建，并依次传递给事件监听器（事件处理函数）。

4. 事件对象

4.3 事件对象的兼容性方案

事件对象本身的获取存在兼容问题：

1. 标准浏览器中是浏览器给方法传递的参数，只需要定义形参 e 就可以获取到。

2. 在 IE6~8 中，浏览器不会给方法传递参数，如果需要的话，需要到 window.event 中获取查找。

解决:

```
e = e || window.event;
```

4. 事件对象

4.4 事件对象的常见属性和方法

e.target 和 this 的区别：

this 是事件绑定的元素， 这个函数的调用者（绑定这个事件的元素）

e.target 是事件触发的元素。

4. 事件对象

4.4 事件对象的常见属性和方法

事件对象属性方法	说明
e.target	返回触发事件的对象　　标准
e.srcElement	返回触发事件的对象　　非标准 ie6-8使用
e.type	返回事件的类型　比如 click mouseover 不带on
e.cancelBubble	该属性阻止冒泡 非标准 ie6-8使用
e.returnValue	该属性 阻止默认事件（默认行为） 非标准 ie6-8使用 比如不让链接跳转
e.preventDefault()	该方法 阻止默认事件（默认行为） 标准 比如不让链接跳转
e.stopPropagation()	阻止冒泡 标准

目录 Contents

5. 阻止事件冒泡

5.1 阻止事件冒泡的两种方式

事件冒泡：开始时由最具体的元素接收，然后逐级向上传播到到 DOM 最顶层节点。

事件冒泡本身的特性，会带来的坏处，也会带来的好处，需要我们灵活掌握。

阻止事件冒泡

- 标准写法：利用事件对象里面的 `stopPropagation()`方法

```
e.stopPropagation()
```

- 非标准写法：IE 6-8 利用事件对象 cancelBubble 属性

```
e.cancelBubble = true;
```

5. 阻止事件冒泡

5.2 阻止事件冒泡的兼容性解决方案

```
if(e && e.stopPropagation){
    e.stopPropagation();
}else{
    window.event.cancelBubble = true;
}
```

目录 Contents

6. 事件委托（代理、委派）

事件冒泡本身的特性，会带来的坏处，也会**带来的好处**，需要我们灵活掌握。生活中有如下场景：

咱们班有100个学生， 快递员有100个快递， 如果一个个的送花费时间较长。同时每个学生领取的时候，也需要排队领取，也花费时间较长，何如?

解决方案： 快递员把100个快递，**委托**给班主任，班主任把这些快递放到办公室，同学们下课自行领取即可。

优势： 快递员省事，委托给班主任就可以走了。 同学们领取也方便，因为相信班主任。

6. 事件委托（代理、委派）

事件冒泡本身的特性，会带来的坏处，也会**带来的好处**，需要我们灵活掌握。程序中也有如此场景：

```
<ul>
    <li>知否知否，应该有弹框在手</li>
    <li>知否知否，应该有弹框在手</li>
    <li>知否知否，应该有弹框在手</li>
    <li>知否知否，应该有弹框在手</li>
    <li>知否知否，应该有弹框在手</li>
</ul>
```

点击每个 li 都会弹出对话框，以前需要给每个 li 注册事件，是非常辛苦的，而且访问 DOM 的次数越多，这就会延长整个页面的交互就绪时间。

6. 事件委托（代理、委派）

事件委托

事件委托也称为事件代理， 在 jQuery 里面称为事件委派。

事件委托的原理

不是每个子节点单独设置事件监听器，而是事件监听器设置在其父节点上，然后利用冒泡原理影响设置每个子节点。

以上案例：给 ul 注册点击事件，然后利用事件对象的 target 来找到当前点击的 li，因为点击 li，事件会冒泡到 ul 上，ul 有注册事件，就会触发事件监听器。

事件委托的作用

我们只操作了一次 DOM ， 提高了程序的性能。

目录 Contents

7. 常用的鼠标事件

7.1 常用的鼠标事件

鼠标事件	触发条件
onclick	鼠标点击左键触发
onmouseover	鼠标经过触发
onmouseout	鼠标离开触发
onfocus	获得鼠标焦点触发
onblur	失去鼠标焦点触发
onmousemove	鼠标移动触发
onmouseup	鼠标弹起触发
onmousedown	鼠标按下触发

7. 常用的鼠标事件

7.1 常用的鼠标事件

1.禁止鼠标右键菜单

contextmenu主要控制应该何时显示上下文菜单，主要用于程序员取消默认的上下文菜单

```
document.addEventListener('contextmenu', function(e) {
e.preventDefault();
})
```

2.禁止鼠标选中（selectstart 开始选中）

```
document.addEventListener('selectstart', function(e) {
 e.preventDefault();
})
```

7. 常用的鼠标事件

7.2 鼠标事件对象

event对象代表事件的状态，跟事件相关的一系列信息的集合。现阶段我们主要是用鼠标事件对象 MouseEvent 和键盘事件对象 KeyboardEvent。

鼠标事件对象	说明
e.clientX	返回鼠标相对于浏览器窗口可视区的 X 坐标
e.clientY	返回鼠标相对于浏览器窗口可视区的 Y 坐标
e.pageX	返回鼠标相对于文档页面的 X 坐标 IE9+ 支持
e.pageY	返回鼠标相对于文档页面的 Y 坐标 IE9+ 支持
e.screenX	返回鼠标相对于电脑屏幕的 X 坐标
e.screenY	返回鼠标相对于电脑屏幕的 Y 坐标

7. 常用的鼠标事件

案例： 跟随鼠标的天使

这个天使图片一直跟随鼠标移动

7. 常用的鼠标事件

案例分析

① 鼠标不断的移动，使用鼠标移动事件： mousemove
② 在页面中移动，给document注册事件
③ 图片要移动距离，而且不占位置，我们使用绝对定位即可
④ 核心原理： 每次鼠标移动，我们都会获得最新的鼠标坐标， 把这个x和y坐标做为图片的top和left 值就可以移动图片

7. 常用的鼠标事件

实现代码

```
var pic = document.querySelector('img');
document.addEventListener('mousemove', function(e) {
var x = e.pageX;
var y = e.pageY;
pic.style.top = y - 40 + 'px';
pic.style.left = x - 50 + 'px';
})
```

目录 Contents

8. 常用的键盘事件

8.1 常用键盘事件

事件除了使用鼠标触发，还可以使用键盘触发。

键盘事件	触发条件
onkeyup	某个键盘按键被松开时触发
onkeydown	某个键盘按键被按下时触发
onkeypress	某个键盘按键被按下时 触发 但是它不识别功能键 比如 ctrl shift 箭头等

注意：

1. 如果使用addEventListener 不需要加 on

2. onkeypress 和前面2个的区别是，它不识别功能键，比如左右箭头，shift 等。

3. 三个事件的执行顺序是： keydown -- keypress --- keyup

8. 常用的键盘事件

8.2 键盘事件对象

键盘事件对象 属性	说明
keyCode	返回该键的ASCII 值

注意： onkeydown 和 onkeyup 不区分字母大小写，onkeypress 区分字母大小写。

在我们实际开发中，我们更多的使用keydown和keyup， 它能识别所有的键（包括功能键）

Keypress 不识别功能键，但是keyCode属性能区分大小写，返回不同的ASCII值

8. 常用的键盘事件

8.3 ASCII 表

ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

高四位 / 低四位		ASCII控制字符 0000 0						ASCII控制字符 0001 1						ASCII打印字符 0010 2		ASCII打印字符 0011 3		ASCII打印字符 0100 4		ASCII打印字符 0101 5		ASCII打印字符 0110 6		ASCII打印字符 0111 7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Backspace 代码：DEL

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。 2013/08/08

8. 常用的键盘事件

案例： 模拟京东按键输入内容

当我们按下 s 键， 光标就定位到搜索框

8. 常用的键盘事件

案例分析

① 核心思路： 检测用户是否按下了s 键，如果按下s 键，就把光标定位到搜索框里面
② 使用键盘事件对象里面的keyCode 判断用户按下的是否是s键
③ 搜索框获得焦点： 使用 js 里面的 focus() 方法

8. 常用的键盘事件

实现代码

```
var search = document.querySelector('input');
document.addEventListener('keyup', function(e) {
// console.log(e.keyCode);
if (e.keyCode === 83) {
search.focus();
}
})
```

8. 常用的键盘事件

案例： 模拟京东快递单号查询

要求：当我们在文本框中输入内容时，文本框上面自动显示大字号的内容。

8. 常用的键盘事件

案例分析

① 快递单号输入内容时， 上面的大号字体盒子（con）显示(这里面的文字

② 同时把快递单号里面的值（value）获取过来赋值给 con盒子（innerText）做为内容

③ 如果快递单号里面内容为空，则隐藏大号字体盒子(con)盒子

④ 注意： keydown 和 keypress 在文本框里面的特点： 他们两个事件触发的时候，文字还没有落入文本框中。

⑤ keyup事件触发的时候， 文字已经落入文本框里面了

⑥ 当我们失去焦点，就隐藏这个con盒子

⑦ 当我们获得焦点，并且文本框内容不为空，就显示这个con盒子

传智播客旗下高端IT教育品牌