web APIs 第二天

事件核心

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

描述	属性/方法	效果
事件类型	鼠标事件	click 鼠标点击 mouseenter 鼠标进入 mouseleave 鼠标离开
	焦点事件	focus 获得焦点 blur 失去焦点
	键盘事件	keydown 键盘按下 keyup 键盘抬起
	文本事件 input	当表单value 被修改时触发
	滚动事件 scroll	当元素 或 页面滚动时触发
事件对象	e.key	判断用户按下哪个键

描述	属性/方法	效果
事件流	addEventListener（‘事件类型’，回调函数，true/false)	捕获还是冒泡
事件对象	事件对象.stopPropagation()	阻止事件传播
	事件对象.preventDefault()	阻止默认行为
事件委托	事件对象.target. classList.contains()	判断触发事件的元素是否有指定类名
移除事件监听	removeEventListener（‘事件类型’，回调函数，true/false)	

学习目标

Learning Objectives

1. 掌握事件类型和事件对象，完成常见网页交互
2. 学习事件流，事件委托等知识
3. 优化多个事件绑定和实现综合案例

目录

Contents

- 事件类型
- 事件对象
- 事件流
- 综合案例

事件类型

事件类型

事件类型的大小写敏感的字符串，统一用小写字母

描述	属性/方法	效果
事件类型	鼠标事件	click 鼠标点击 mouseenter 鼠标进入 mouseleave 鼠标离开
	焦点事件	focus 获得焦点 blur 失去焦点
	键盘事件	keydown 键盘按下 keyup 键盘抬起
	文本事件 input	当表单value 被修改时触发
	滚动事件 scroll	当元素 或 页面滚动时触发

事件类型

事件类型的大小写敏感的字符串，统一用小写字母

描述	属性/方法	效果
事件类型	鼠标事件	click 鼠标点击 mouseenter 鼠标进入 mouseleave 鼠标离开
	焦点事件	focus 获得焦点 blur 失去焦点
	键盘事件	keydown 键盘按下 keyup 键盘抬起
	文本事件 input	当表单value 被修改时触发
	滚动事件 scroll	当元素 或 页面滚动时触发

案例

小米搜索框案例

需求：当表单得到焦点，显示下拉菜单，失去焦点隐藏下拉菜单

分析：

①：开始下拉菜单要进行隐藏

②：表单获得焦点 focus，则显示下拉菜单，并且搜索框变色（添加类）

③：表单失去焦点 blur，则隐藏下拉菜单，搜索框复原原来颜色

事件类型

事件类型的大小写敏感的字符串，统一用小写字母

描述	属性/方法	效果
事件类型	鼠标事件	click 鼠标点击 mouseenter 鼠标进入 mouseleave 鼠标离开
	焦点事件	focus 获得焦点 blur 失去焦点
	键盘事件	keydown 键盘按下 keyup 键盘抬起
	文本事件 input	当表单value 被修改时触发
	滚动事件 scroll	当元素 或 页面滚动时触发

事件类型

- keydown / keyup 和 input 事件**区别**

事件	触发时机	得到表单值
keydown	按下键盘时触发	不带最后一次按键值 ab
keyup	弹起键盘时触发	输入内容 abc
input	表单value发生变化时触发	输入内容 abc

- 三者的执行顺序：
  - keydown → input → keyup

案例

统计用户输入字数案例

需求：用户输入文字，可以计算用户输入的字数

功能：

①：统计字数盒子，可以根据文本域获得失去焦点，实现显示和隐藏

②：根据用户输入的字数，写到统计字数盒子里面

案例

统计用户输入字数案例

需求：用户输入文字，可以计算用户输入的字数

分析：

①：文本域获得焦点则显示统计 total 盒子，失去焦点则隐藏统计 total 盒子

②：文本域输入内容，使用input事件，不断取得字符长度（文本域.value.length ）

③：把获得字符长度赋值给 total 字数统计盒子

事件类型

事件类型的大小写敏感的字符串，统一用小写字母

描述	属性/方法	效果
事件类型	鼠标事件	click 鼠标点击 mouseenter 鼠标进入 mouseleave 鼠标离开
	焦点事件	focus 获得焦点 blur 失去焦点
	键盘事件	keydown 键盘按下 keyup 键盘抬起
	文本事件 input	当表单value 被修改时触发
	滚动事件 scroll	当元素 或 页面滚动时触发

页面滚动事件

- 滚动条在滚动的时候持续触发的事件
- 为什么要学?
  - 很多网页需要检测用户把页面滚动到某个区域后做一些处理，比如返回顶部
- 事件名：scroll
- 监听整个页面滚动：

```
// 页面滚动事件
window.addEventListener('scroll', function () {
    // 执行的操作
})
```

  - 给 window 或 document 添加 scroll 事件
- 监听某个元素的内部滚动直接给某个元素加即可

页面滚动事件

- 开发需求：
  - 我们想要页面滚动一段距离，比如100px，就让某些元素显示隐藏，那我们怎么知道，页面滚动了100像素呢？
- 首先解决2个问题：
  1. 页面谁滚动？
  2. 如何知道滚动了多少距离？

页面滚动事件

页面谁滚动?

HTML元素滚动

document.documentElement 返回HTML元素

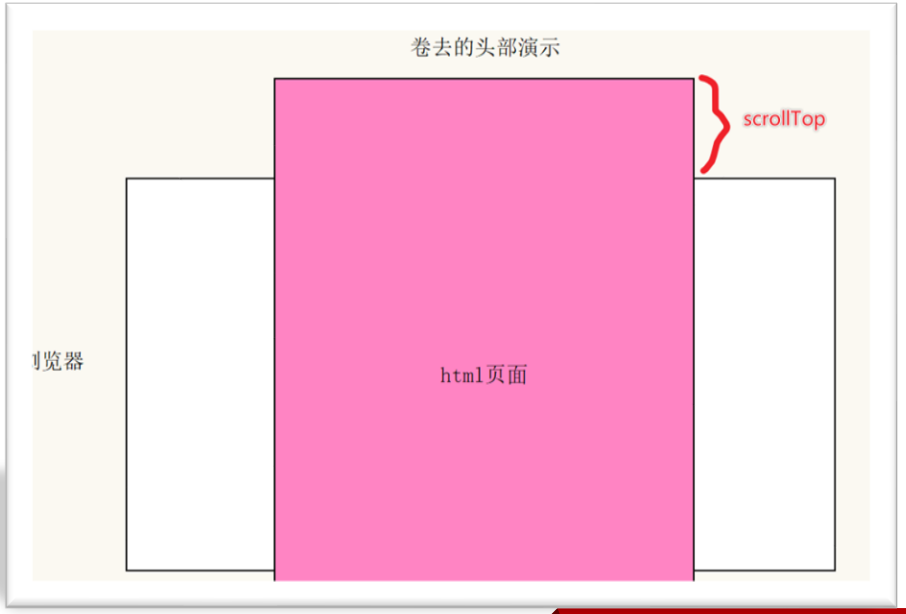
```
<html lang="en">
▶ <head>…</head>
▶ <body>…</body>
</html>
```

```
// 获得页面的滚动距离
window.addEventListener('scroll', function () {
  // document.documentElement 得到html元素
  // scrollTop  得到被卷去的头部
  const n = document.documentElement.scrollTop
  console.log(n)  // 可以得到页面滚动顶部距离
})
```

如何知道滚动了多少距离?

scrollLeft和scrollTop （属性）

- 获取被卷去的左侧和头部
- 获取元素内容往左、往上滚出去看不到的距离
- 这两个值是可读写的



总结

1. 页面滚动是哪个元素滚动？如何得到这个元素？
   - html 元素
   - document.documentElement
2. 被卷去的头部或者左侧用哪个属性？是否可以读取和修改？
   - scrollTop / scrollLeft
   - 可以读取，也可以修改（赋值）

```
// 获得页面的滚动距离
window.addEventListener('scroll', function () {
  // document.documentElement 得到html元素
  // scrollTop  得到被卷去的头部
  const n = document.documentElement.scrollTop
  console.log(n)  // 可以得到页面滚动顶部距离
})
```

案例 页面滚动显示隐藏返回顶部按钮

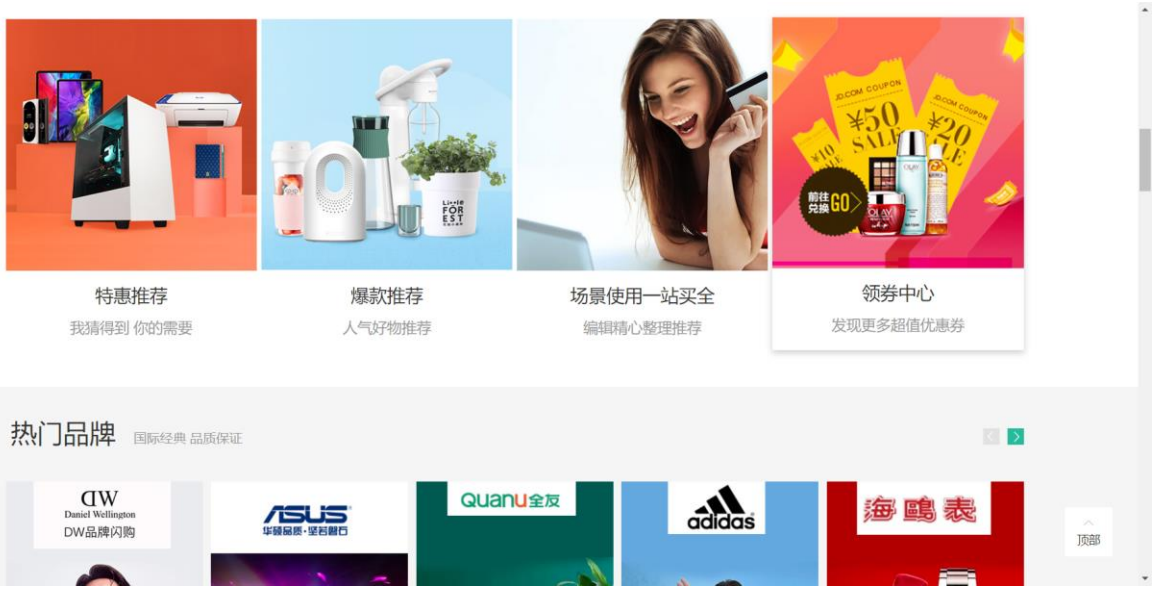
需求：当页面滚动大于300像素的距离时候，就显示返回按钮，否则隐藏

分析：

①：需要用到页面滚动事件 scroll

②：html元素卷去的头部 scrollTop，如果大于等于300，就让返回按钮显示，否则隐藏

③：显示和隐藏 通过 style.display 来控制



案例 返回顶部

需求：点击返回按钮，页面会返回顶部

分析：

①：按钮注册点击事件

②：利用HTML元素的 scrollTop 赋值为 0 可以让页面返回顶部

③：小技巧：如果想要页面滑动效果滚动到页面顶部可以使用css属性

```
/* 页面滑动 */
html {
  /* 让滚动条丝滑的滚动 */
  scroll-behavior: smooth;
}
```

目录

Contents

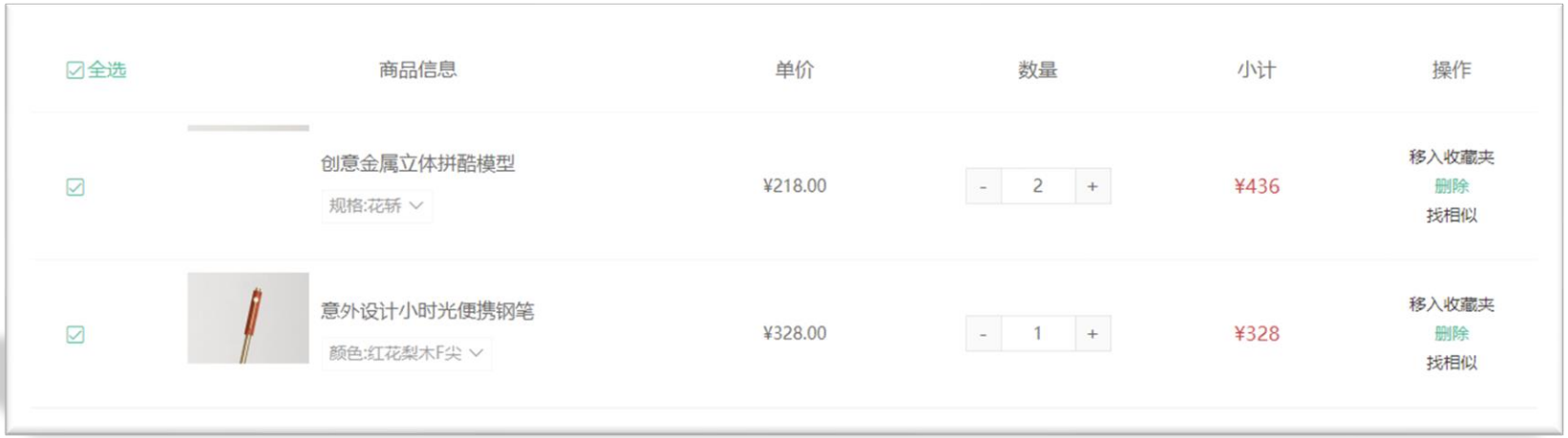
- 事件类型
- 事件对象
- 事件流
- 综合案例

事件对象

- 获取事件对象
- 事件对象常用属性

事件对象（重要）

- 事件对象是什么
  - 也是个对象，这个对象里有事件触发时的相关信息，包含属性和方法
  - 例如：鼠标点击事件中，事件对象就存了鼠标点在哪个位置等信息
- 使用场景
  - 可以判断用户按下哪个键，比如按下回车键可以发布新闻
  - 可以判断鼠标点击了哪个元素，从而做相应的操作



事件对象

- 语法：
  - 注册事件中，回调函数的第一个参数就是事件对象
  - 一般命名为event、ev、e

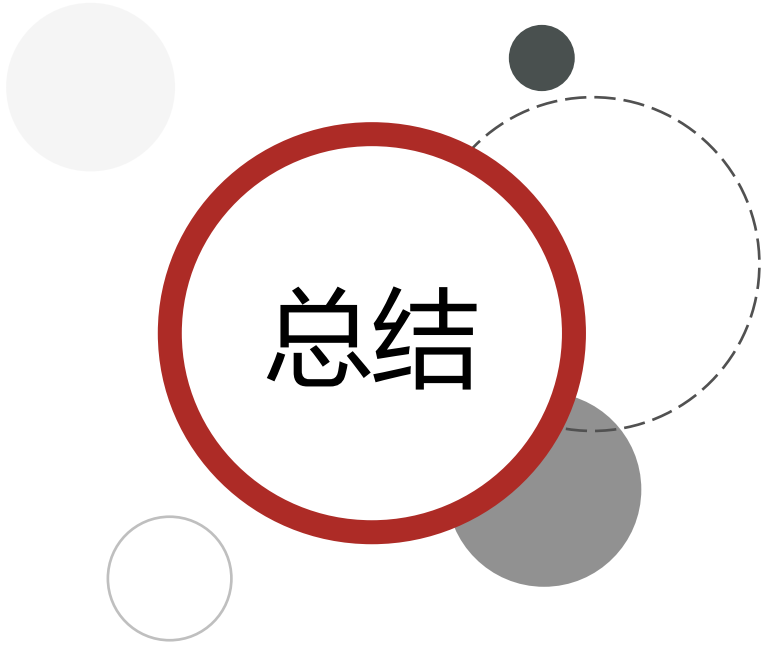
```
元素.addEventListener('click', function (e) {

}
```

事件对象

事件对象-常见属性

属性名	类型	说明
altKey	boolean	事件发生时，是否按下alt按键
ctrlKey	boolean	事件发生时，是否按下ctrl按键
shiftKey	boolean	事件发生时，是否按下shift按键
offsetX	number	事件发生时，鼠标相对于事件源的x坐标
offsetY	number	事件发生时，鼠标相对于事件源的y坐标
target	object	事件源对象
pageX	number	事件发生时，鼠标相对于网页的x坐标
pageY	number	事件发生时，鼠标相对于网页的y坐标
clientX	number	事件发生时，鼠标相对于视口的x坐标
clientY	number	事件发生时，鼠标相对于视口的y坐标
key	string	如果是键盘相关事件，则事件对象中包含该属性，表示键盘事件发生时，按下的是什么键。'Enter' 回车键



1. 事件对象是什么？
   - 也是个对象，这个对象里有事件触发时的相关信息
2. 事件对象在哪里？
   - 在事件绑定的回调函数的第一个参数就是事件对象

```
元素.addEventListener('click', function (e) {

}
```

3. 想要得到用户按下了键盘哪个键怎么来实现？
   - 事件对象.key 属性

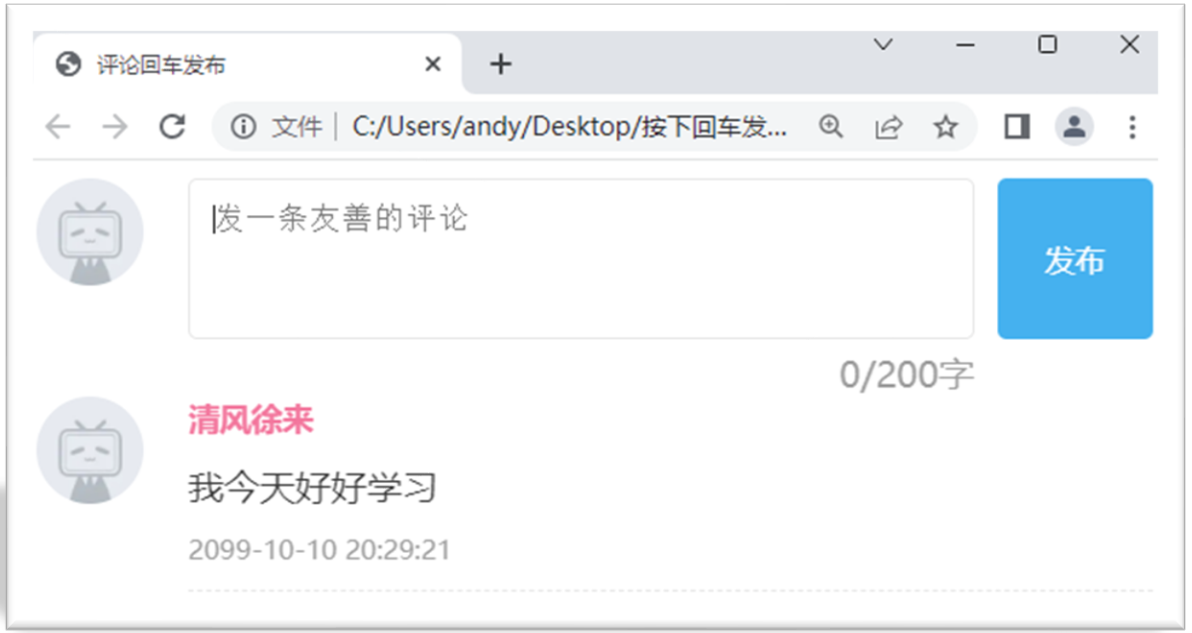
案例 评论回车发布

需求：按下回车键，可以发布评论

功能：

①：按下回车，可以显示评论信息，并且评论内容显示到对应位置

②：输入完毕，文本域清空内容，并且字数复原为 0



案例

评论回车发布

需求：按下回车键盘，可以发布信息

分析：

①：通过键盘事件中的事件对象判断回车键，则显示评论模块，把内容填入

②：清空文本域就是修改文本域值为空字符串，同时把字数通过 innerText 复原为 0

③：点击发布按钮也可以发布评论，此处 回车可以采用 btn.click() 方法更简单

目录

Contents

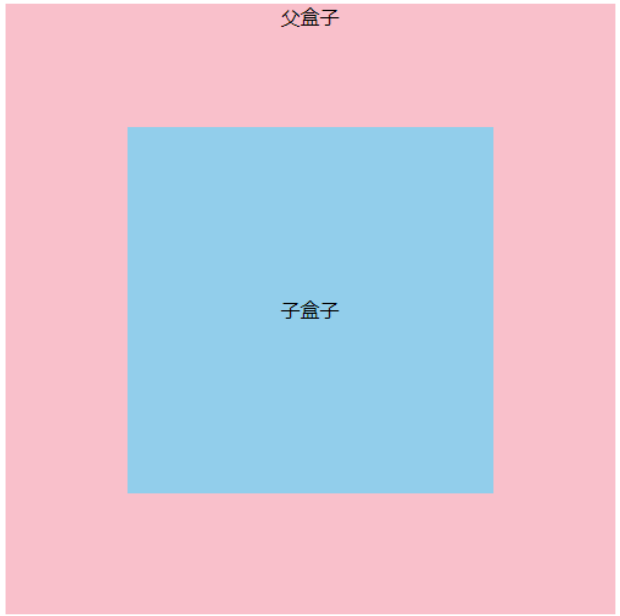
- 事件类型
- 事件对象
- 事件流
- 综合案例

事件流

- 事件流与两个阶段说明
- 事件捕获
- 事件冒泡
- 阻止冒泡
- 事件委托

3.1 事件流

- 为什么要学习事件流?
  - 可以帮我们解决一些疑惑，比如点击子盒子会会弹出2次的问题
  - 可以帮我们优化代码，事件委托



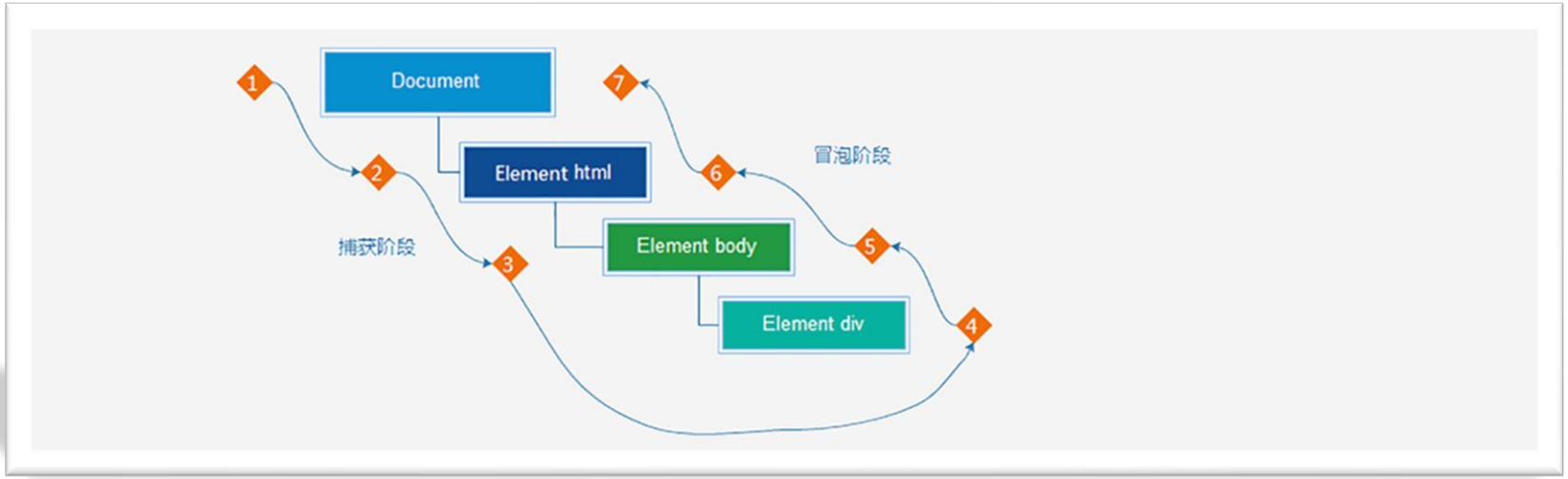

```
<script>
  // 事件流
  const father = document.querySelector('.father')
  const son = document.querySelector('.son')

  // 点击父盒子
  father.addEventListener('click', function () {
    alert('我是爸爸')
  })
  // 点击子盒子
  son.addEventListener('click', function () {
    alert('我是儿子')
  })
</script>
```

3.1 事件流

事件流指的是事件完整执行过程中的流动路径

当触发事件时，会经历两个阶段，分别是捕获阶段、冒泡阶段



- **事件捕获概念：**

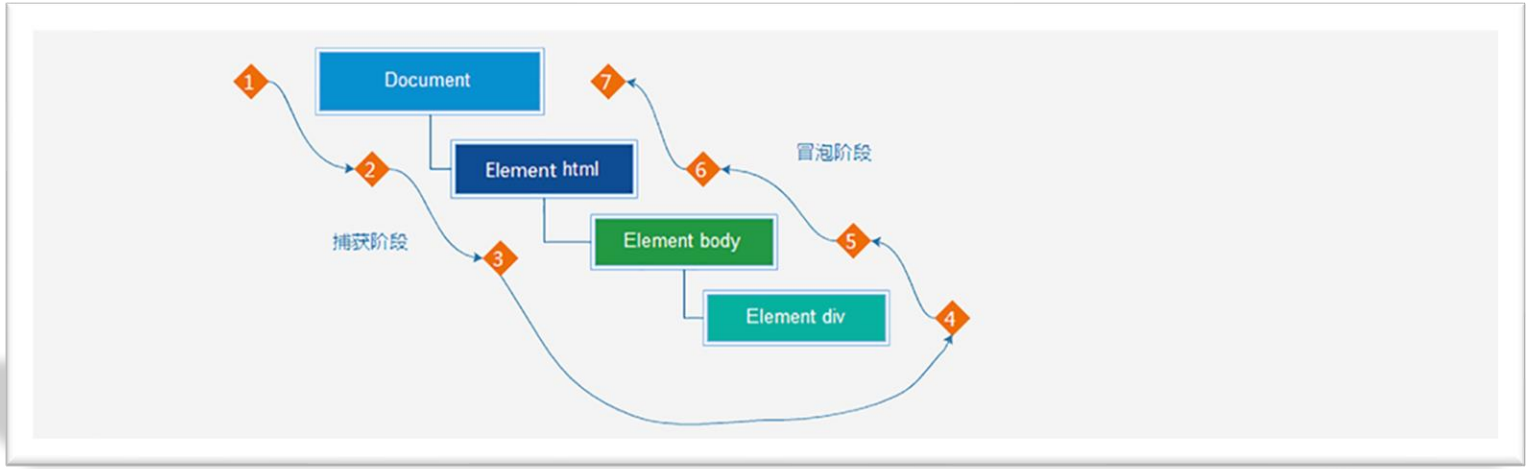
当一个元素的事件被触发时，会从DOM的根元素开始依次调用同名事件 (从外到里)

3.2 事件捕获

- 事件捕获需要写对应代码才能看到效果
- 代码：

```
DOM.addEventListener(事件类型，事件处理函数，是否使用捕获机制)
```

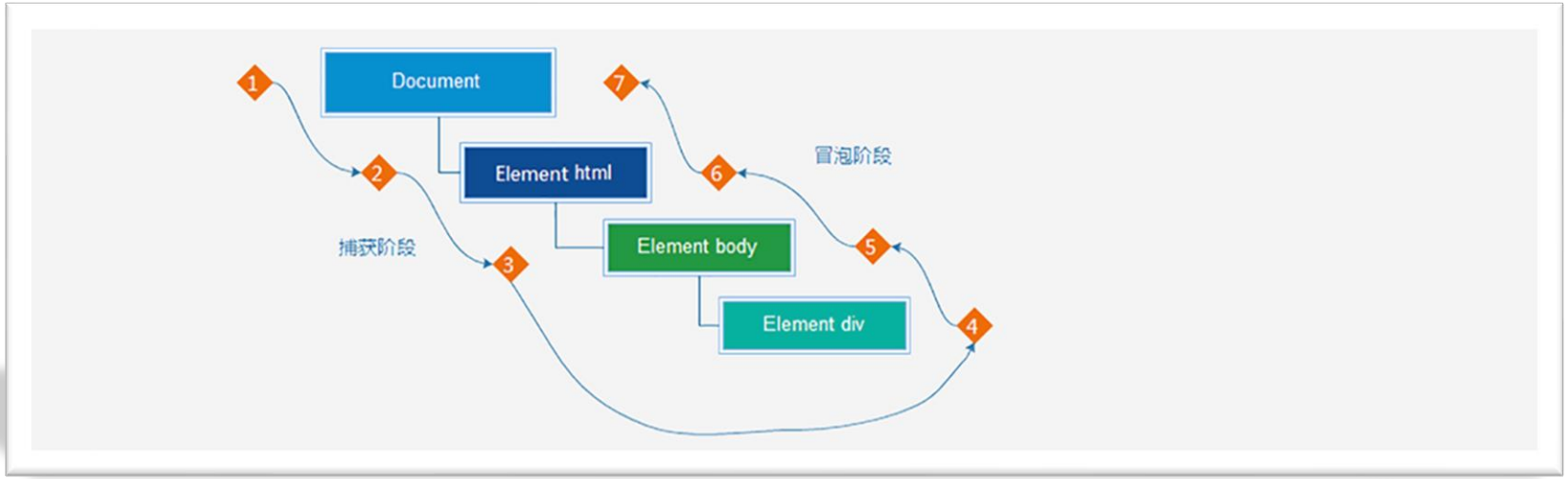
- 说明：
  - addEventListener第三个参数传入 true 代表是捕获阶段触发（很少使用）
  - 若传入false代表冒泡阶段触发，默认就是 false



3.3 事件冒泡

事件冒泡概念:

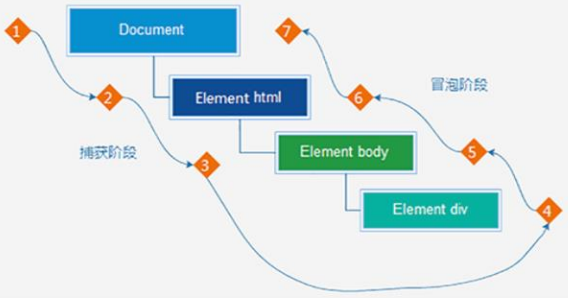
当一个元素的事件被触发时，同样的事件将会在该元素的所有祖先元素中依次被触发。这一过程被称为事件冒泡



- 简单理解：当一个元素触发事件后，会依次向上调用所有父级元素的 同名事件
- 事件冒泡是默认存在的，或者第三个参数传入 false 都是冒泡
- 实际工作都是使用事件冒泡为主

总结

1. 事件流是什么？分为哪几个阶段？
   - 事件流指的是事件完整执行过程中的流动路径
   - 分为：捕获阶段、冒泡阶段
2. 怎么理解事件捕获阶段？语法怎么写？
   - 从DOM的根元素开始去执行对应的事件 (从外到里、父到子)
   - addEventListener第三个参数传入 true
   - 实际开发中，事件捕获用的很少，了解即可
3. 怎么理解事件冒泡？
   - 当一个元素触发事件后，会依次向上调用所有父级元素的同名事件
   - 子到父



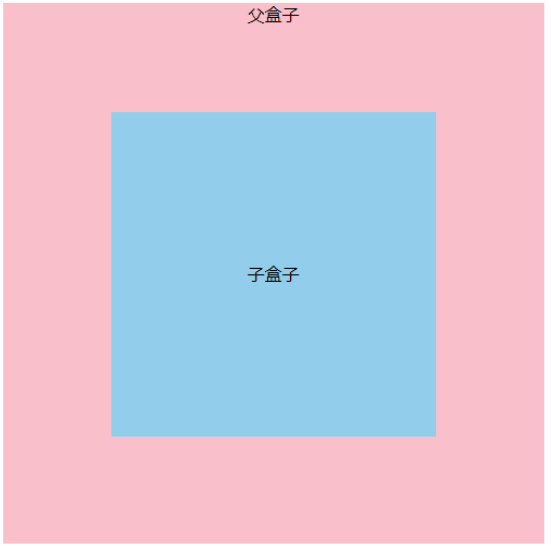

```
DOM.addEventListener(事件类型，事件处理函数，是否使用捕获机制)
```

阻止冒泡

- **问题：**因为默认就有冒泡阶段的存在，所以容易导致事件影响到父级元素
- **需求：**若想把事件就限制在当前元素内，就需要阻止事件冒泡
- **前提：**阻止事件冒泡需要拿到事件对象
- **语法：**

```
事件对象.stopPropagation()
```

- **注意：**此方法可以阻断事件流动传播，不光在冒泡阶段有效，捕获阶段也有效




```
const father = document.querySelector('.father')
const son = document.querySelector('.son')
document.addEventListener('click', function () {
  alert('我是爷爷')
})
fa.addEventListener('click', function () {
  alert('我是爸爸')
})
// 需要事件对象
son.addEventListener('click', function (e) {
  alert('我是儿子')
  // 阻止冒泡
  e.stopPropagation()
})
```

鼠标经过/离开事件的区别

- 鼠标经过/离开事件：
  - mouseover 和 mouseout 会有冒泡
  - mouseenter 和 mouseleave 没有冒泡 (常用)

阻止默认行为

- 阻止元素发生默认的行为
- 例如：
  - 当点击提交按钮时阻止对表单的提交
  - 阻止链接的跳转等等
- **语法：**

```
事件对象.preventDefault()
```

事件流

- 事件流与两个阶段说明
- 事件捕获
- 事件冒泡
- 阻止冒泡
- 事件委托

事件委托

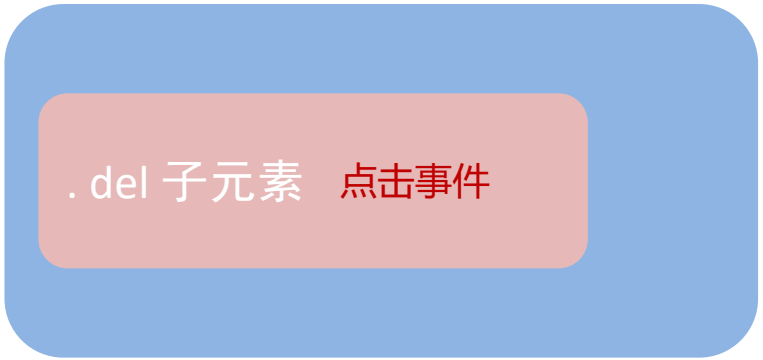
- 事件委托(Event Delegation)：是JavaScript中注册事件的常用技巧，也称为事件委派、事件代理
- 简单理解：原本需要注册在子元素的事件委托给父元素，让父元素担当事件监听的职务
- 为什么要用事件委托呢?
  - 如果同时给多个元素注册事件，还需要利用循环多次注册事件

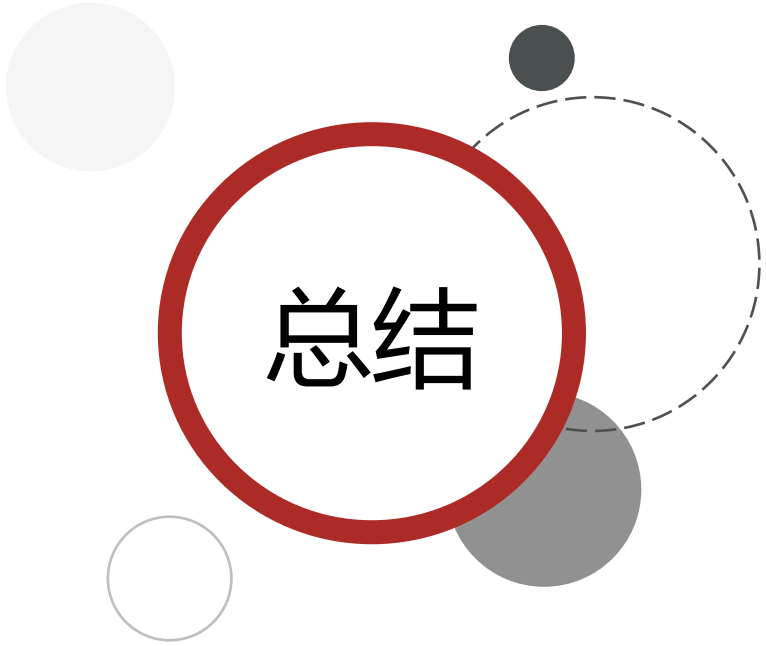

```
const delBtns = document.querySelectorAll('.del')
delBtns.forEach(function (del) {
  del.addEventListener('click', function () {
    alert('被点击了')
  })
})
```

事件委托

- 事件委托是利用事件流的特征解决一些开发需求的知识技巧
  - **优点：**
    - 减少注册次数，可以提高程序性能
    - 动态生成的元素也能触发事件
  - **原理：**事件委托其实是利用事件冒泡的特点
    - 给父元素注册事件，当我们触发子元素的时候，会冒泡到父元素身上，从而触发父元素的事件

tbody.addEventListener('click', function(){})　执行父级点击事件





1. 什么是事件委托?
   - 常用技巧，也称为事件委派、事件代理
   - 原本需要注册在子元素的事件委托给父元素，让父元素担当事件监听的职务
2. 事件委托的好处是什么?
   - 减少注册次数，提高了程序性能
   - 动态生成的元素也能触发事件
3. 事件委托是委托给了谁？父元素还是子元素?
   - 父元素

```
tbody.addEventListener('click', function () {
  alert('我会弹窗')
})
```

事件委托

- 利用事件委托方式如何得到当前触发事件的元素呢?
- **实现：**事件对象.target.classList.contains() 可以判断真正触发事件的元素是否包含指定类名

```
// 需求：  点击每个删除按钮，都会弹出询问框确认是否删除

// 给tbody注册点击事件
const tbody = document.querySelector('tbody')

tbody.addEventListener('click', function (e) {
  // alert('我会弹窗')

  // 3. 利用事件对象.target 得到目标元素
  // console.log(e.target)
  // e.target.style.color = 'red'

  if (e.target.classList.contains('del')) {
    confirm('确认删除吗？')
  }
})
```

总结

1. 如何利用事件委托方式找到真正触发的元素？

   - 事件对象.target.classList.contains()

```
// 需求：  点击每个删除按钮，都会弹出询问框确认是否删除

// 给tbody注册点击事件
const tbody = document.querySelector('tbody')

tbody.addEventListener('click', function (e) {
  // alert('我会弹窗')

  // 3. 利用事件对象.target 得到目标元素
  // console.log(e.target)
  // e.target.style.color = 'red'

  if (e.target.classList.contains('del')) {
    confirm('确认删除吗？')
  }
})
```

案例

事件委托版 - 价格筛选案例

需求：优化程序，将价格筛选案例改为事件委托写法

0-100元 | 100-300元 | 300元以上 | 全部区间


称心如意手摇咖啡磨豆机咖啡豆研磨机
¥289.00


日式黑陶功夫茶组双侧把茶具礼盒装
¥288.00


竹制干泡茶盘正方形沥水茶台品茶盘
¥109.00


古法温酒汝瓷酒具套装白酒杯莲花温酒器
¥488.00

案例

事件委托版 - 价格筛选案例

需求：优化程序，将价格筛选案例改为事件委托写法

思路：点击筛选效果

①：把点击事件委托注册给父元素

②：判断触发事件的元素是否为导航a链接

③：因为没有循环遍历了，没有下标了，所以这里可以使用自定义属性，给5个链接添加序号

④：根据序号判断需要哪个区间的数据，筛选得到对应的数据并渲染展示

目录

Contents

- 事件类型
- 事件对象
- 事件流
- 综合案例

排他思想

- 是一种思路，目的是突出显示某个元素
- 比如，有多个元素，当鼠标经过时，只有当前元素会添加高亮样式，其余的元素移除样式

口诀：注意顺序

①：排除其他人

②：保留我自己

案例 Tab栏切换

需求：鼠标点击不同的选项卡，底部可以显示 不同的内容

业务:

①：顶部点击谁，谁会高亮显示

②：底部跟随切换不同内容

步骤

Tab栏切换

业务1分析：鼠标点击谁，谁会高亮显示

①：给a的父级注册鼠标点击事件（事件委托）

②：如果鼠标点击的是 A

- 判断的方式 利用 e.target.classList.contains()

③：先移除其余元素身上的 active 类，而给当前元素添加 active 类（排他思想）

- e.target 是当前元素

步骤

Tab栏切换

业务2分析：底部跟随切换不同内容

①：先把5个item获取过来得到伪数组

②：底部其余 item盒子移除 active 类（隐藏）

③：然后根据下标选取对应盒子，然后添加 active类（显示）

- 给5个链接添加下标 （使用自定义属性）
- 获取下标（ `e.target.dataset.index` ）

传智教育旗下高端IT教育品牌