web APIs 第三天

事件进阶

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
事件流	addEventListener(‘事件类型’，回调函数，true/false)	捕获还是冒泡
事件委托	事件对象.target. tagName	得到目标元素
其他事件	load	加载事件，全部资源加载完毕
	DOMContentLoaded	加载事件，HTML文档加载完毕
	scroll	滚动事件
	resize	尺寸事件
元素尺寸与位置	scrollLeft 和 scrollTop	页面/元素被卷去的头部和左侧（滚动），可读写
	offsetLeft 和 offsetTop	元素位置距离定位父级左上距离，只读
	clientWidth 和 clientHeight	元素大小，不包含border、padding等，只读
	offsetWidth 和 offsetHeight	元素大小，包含border、padding等，只读

学习目标

Learning Objectives

1. 学习事件流，事件委托，其他事件等知识
2. 优化多个事件绑定和实现常见网页交互

目录

Contents

- 事件流
- 移除事件监听
- 其他事件
- 元素尺寸与位置
- 综合案例

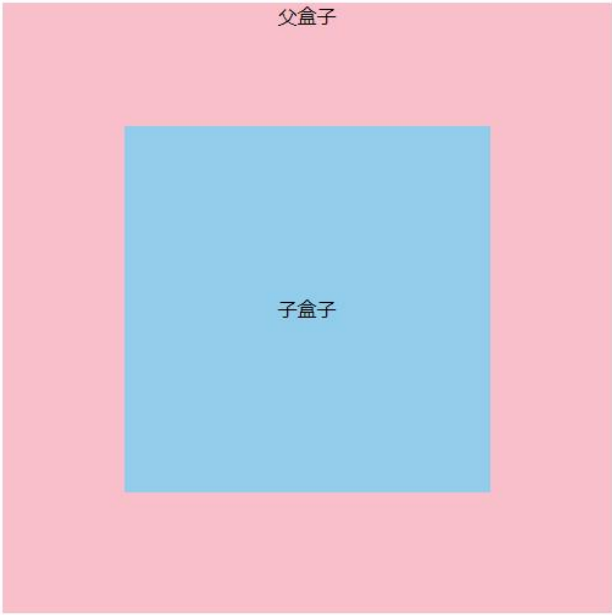
01

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

1.1 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



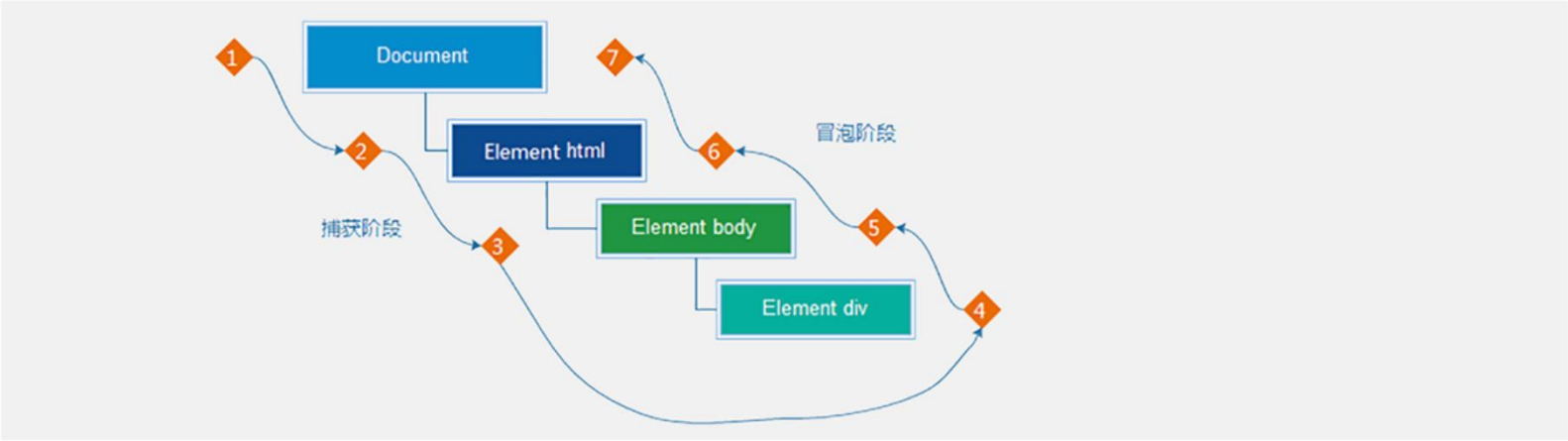

```
<script>
  // 事件流
  const father = document.querySelector('.father')
  const son = document.querySelector('.son')

  // 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>
```

1.1 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：**

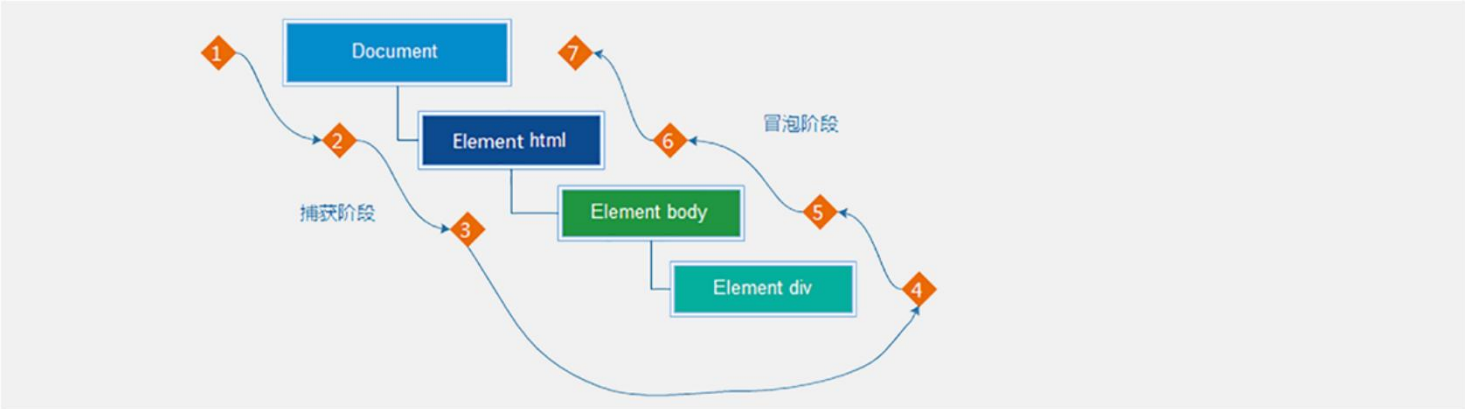
当一个元素的事件被触发时，会从DOM的根元素开始依次调用同名事件（从外到里）

1.2 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)
```

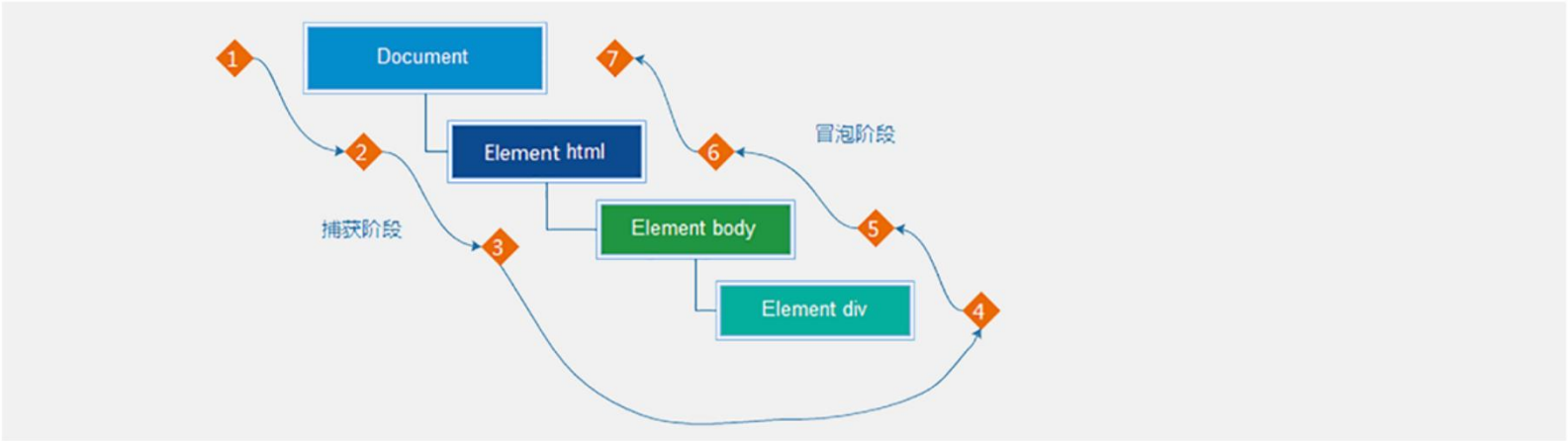
- 说明：
  - addEventListener第三个参数传入 true 代表是捕获阶段触发（很少使用）
  - 若传入false代表冒泡阶段触发，默认就是 false



1.3 事件冒泡

事件冒泡概念：

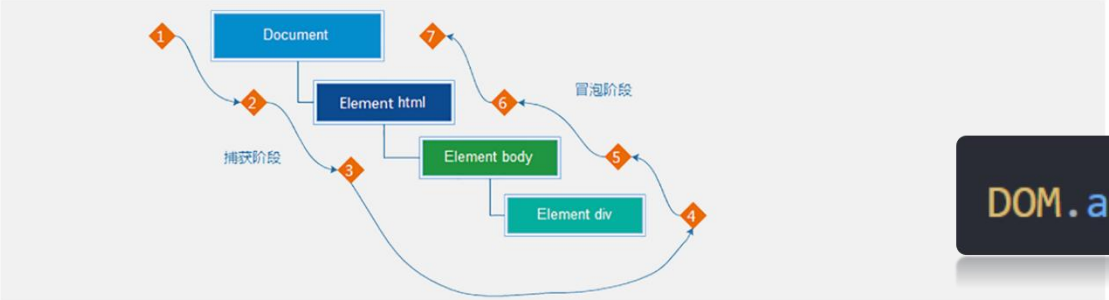
当一个元素的事件被触发时，同样的事件将会在该元素的所有祖先元素中依次被触发。这一过程被称为事件冒泡



- 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)
```

1.4 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)

01

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

1.5 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

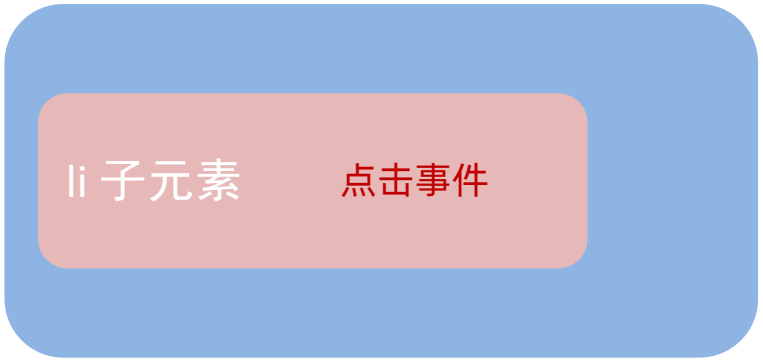
```html
<ul>
    <li>我是第1个小li</li>
    <li>我是第2个小li</li>
    <li>我是第3个小li</li>
    <li>我是第4个小li</li>
    <li>我是第5个小li</li>
</ul>
```

```js
const lis = document.querySelectorAll('ul li')
for (let i = 0; i < lis.length; i++) {
    lis[i].addEventListener('click', function () {
        alert('我被点击了')
    })
}
```

1.5 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
  - **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

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



总结

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
3. 事件委托是委托给了谁？父元素还是子元素？
   - 父元素

```
const ul = document.querySelector('ul')
ul.addEventListener('click', function () {
  alert('我会弹窗')
})
```

1.5 事件委托

- 利用事件委托方式如何得到当前点击的元素呢？
- **实现：**事件对象.target. tagName 可以获得真正触发事件的元素

```
const ul = document.querySelector('ul')
ul.addEventListener('click', function (e) {
  // console.dir(e.target)
  if (e.target.tagName === 'LI') {
    this.style.color = 'pink'
  }
})
```

```
<ul>
    <li>我是第1个小li</li>
    <li>我是第2个小li</li>
    <li>我是第3个小li</li>
    <li>我是第4个小li</li>
    <li>我是第5个小li</li>
</ul>
```

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.tagName

```
const ul = document.querySelector('ul')
ul.addEventListener('click', function (e) {
  // console.dir(e.target)
  if (e.target.tagName === 'LI') {
    this.style.color = 'pink'
  }
})
```

案例

tab栏切换改造

需求：优化程序，将tab切换案例改为事件委托写法

案例

tab栏切换改造

需求：优化程序，将tab切换案例改为事件委托写法

思路：

1. tab栏顶部高亮效果：

①：给a的父级注册鼠标经过事件，采取事件委托方式

②：注意mouseenter没有冒泡，所以此处使用mouseover

③：如果鼠标经过的是 A，则进行排他思想，删除添加类

④：注意判断的方式 利用 e.target.tagName

案例

tab栏切换改造

需求：优化程序，将tab切换案例改为事件委托写法

思路：

2. tab栏底部跟随显示效果：

⑤： 因为没有索引号了，所以这里我们可以自定义属性，给5个链接添加序号

⑥： 下面大盒子获取索引号的方式 e.target.dataset.id 号， 然后进行排他思想

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()
```

目录

Contents

- 事件流
- 移除事件监听
- 其他事件
- 元素尺寸与位置
- 综合案例

02 移除事件监听

2. 移除事件监听（了解）

移除事件处理函数，也称为解绑事件

移除L2事件监听

- addEventListener方式注册，必须使用：removeEventListener(事件类型, 事件处理函数, [捕获或者冒泡阶段])

注意：匿名函数无法解绑

```
function fn() {
  alert('点击了')
}
// 绑定事件
btn.addEventListener('click', fn)
// 解绑事件
btn.removeEventListener('click', fn)
```

移除L0事件监听

- on事件方式，直接使用null覆盖偶就可以实现事件的解绑

```
// 绑定事件
btn.onclick = function () {
  alert('点击了')
}
// 解绑事件
btn.onclick = null
```

两种注册事件的区别

- 传统on注册（L0）
  - 同一个对象,后面注册的事件会覆盖前面注册(同名事件)
  - 直接使用null覆盖偶就可以实现事件的解绑
  - 只有冒泡阶段，没有捕获阶段
- 事件监听注册（L2）
  - 语法: addEventListener(事件类型, 事件处理函数, 是否使用捕获)
  - 后面注册的事件不会覆盖前面注册的事件(同名事件)
  - 必须使用removeEventListener(事件类型, 事件处理函数, 获取捕获或者冒泡阶段)实现事件解绑
  - 可以通过第三个参数去确定是在冒泡或者捕获阶段执行
  - 匿名函数无法被解绑

目录

Contents

- 事件流
- 移除事件监听
- 其他事件
- 元素尺寸与位置
- 综合案例

03

其他事件

- 页面加载事件
- 页面滚动事件
- 页面尺寸事件

目标：掌握新的事件，做更强交互

3.1 页面加载事件

- 加载外部资源（如图片、外联CSS和JavaScript等）加载完毕时触发的事件
- 为什么要学？
  - 有些时候需要等页面资源全部处理完了做一些事情
  - 老代码喜欢把 script 写在 head 中，这时候直接找 dom 元素找不到
- 事件名：load
- 监听页面所有资源加载完毕：
  - 给 window 添加 load 事件

```
// 页面加载事件
window.addEventListener('load', function () {
    // 执行的操作
})
```

3.1 页面加载事件

- 当初始的 HTML 文档被完全加载和解析完成之后就触发，而无需等待样式表、图像等完全加载
- 事件名：DOMContentLoaded
- 监听页面DOM加载完毕：
  - 给 document 添加 DOMContentLoaded 事件

```
document.addEventListener('DOMContentLoaded', function () {
    // 执行的操作
})
```

总结

1. 页面加载事件有哪两个？有什么区别？
   - load 事件
     - 监听整个页面资源给 window 加
   - DOMContentLoaded
     - 给 document 加
     - 无需等待样式表、图像等完全加载

03

其他事件

- 页面加载事件
- 元素滚动事件
- 页面尺寸事件

目标：掌握新的事件，做更强交互

3.2 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， 比如固定导航栏，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

3.2 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？

3.2 页面滚动事件

页面谁滚动？

HTML元素滚动

document.documentElement 返回HTML元素

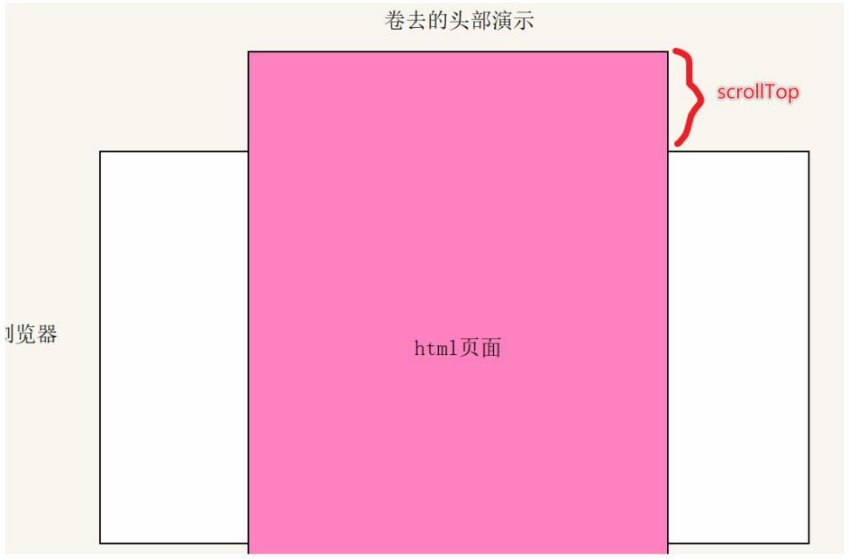
```
<html lang="en">
▶ <head>…</head>
▶ <body>…</body>
</html>
```

```
// 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

页面滚动显示隐藏侧边栏

需求：当页面滚动大于等于300像素的距离时候，就显示侧边栏，否则隐藏侧边栏

案例

页面滚动显示隐藏侧边栏

需求：当页面滚动大于300像素的距离时候，就显示侧边栏，否则隐藏侧边栏

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，就让侧边栏显示，否则隐藏

③：显示和隐藏配合css过渡，利用opacity加渐变效果

案例

返回顶部

需求：点击返回按钮，页面会返回顶部

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;
}
```

03

其他事件

- 页面加载事件
- 元素滚动事件
- 页面尺寸事件

目标：掌握新的事件，做更强交互

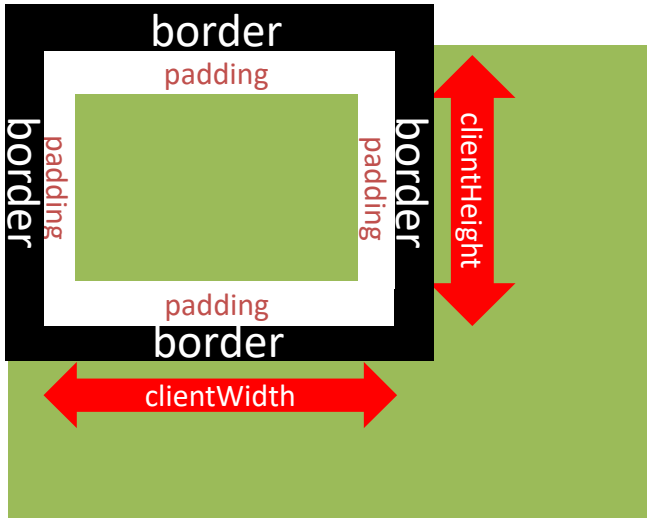
3.3 页面尺寸事件

- 会在窗口尺寸改变的时候触发事件：
  - resize

```
window.addEventListener('resize', function () {
    // 执行的代码
})
```

3.3 页面尺寸事件-获取元素宽高

- 获取宽高：
  - 获取元素的可见部分宽高（不包含border，margin，滚动条等）
  - clientWidth和clientHeight



案例

Rem基准值

需求：分析 flexible.js 源码

```
// 声明一个计算字号的函数
const setFontSize = function () {
  // 获取 html元素
  const html = document.documentElement
  // 获取 html元素的宽度
  const clientWidth = html.clientWidth
  // html 根字号设置： 当前页面宽度（html元素） / 10 划分为10等份
  html.style.fontSize = clientWidth / 10 + 'px'
}
// 页面加载先调用执行一次
setFontSize()
// 如果页面尺寸发生变化，则重新执行函数，重新计算
window.addEventListener('resize', setFontSize)
```

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
事件委托	事件对象.target. tagName	得到目标元素
其他事件	load	加载事件，全部资源加载完毕
	DOMContentLoaded	加载事件，HTML文档加载完毕
	scroll	滚动事件
	resize	尺寸事件
元素尺寸与位置	scrollLeft 和 scrollTop	页面/元素被卷去的头部和左侧（滚动），可读写
	offsetLeft 和 offsetTop	元素位置距离定位父级左上距离，只读
	clientWidth 和 clientHeight	元素大小，不包含border、padding等，只读
	offsetWidth 和 offsetHeight	元素大小，包含border、padding等，只读

目录

Contents

- 事件流
- 事件委托
- 其他事件
- 元素尺寸与位置
- 综合案例

04 元素尺寸与位置

4. 元素尺寸与位置

获得元素的尺寸大小和页面中的位置

- 使用场景：
  - 可以通过js的方式，得到元素在页面中的位置
  - 可以通过js的方式，得到元素的实际大小

中华文明源远流长、博大精深，是中华民族生生不息、发展壮大的丰厚滋养，也为人类文明进步作出了重大贡献。习近平总书记在党的二十大报告中作出“推进文化自信自强，铸就社会主义文化新辉煌”的重大部署，提出“增强中华文明传播力影响力”的任务要求，为新时代新征程提升国家文化软实力、加强国际传播能力建设、推动中华文化更好走向世界，指明了前进方向，提供了根本遵循。必须进一步增强使命感责任感，深入贯彻党的二十大战略部署，对外宣介展示好习近平新时代中国特色社会主义思想是中华文化和中国精神的时代精华，讲好中华文明故事，推动文明交流互鉴，展现可信、可爱、可敬的中国形象，努力为全面建设社会主义现代化国家、全面推进中华民族伟大复兴营造有利外部舆论环境，为推动构建人类命运共同体作出新的贡献。

4. 元素尺寸与位置

offset家族

大小

- offsetWidth 和 offsetHeight
- 获取元素的自身宽高、包含元素自身设置的宽高、padding、border
- 返回的是数字不带单位，并且是只读属性

位置

- offsetLeft 和 offsetTop
- 获取元素距离自己定位父级元素的左、上距离，跟绝对定位类似
- 如果父级都没有定位则以浏览器文档为准
- 返回的是数字不带单位，并且是只读属性

总结

1. offsetWidth和offsetHeight是包含哪些影响盒子大小的样式？
   - 自身宽高+ padding + border
2. offsetTop和offsetLeft 得到位置以谁为准？
   - 带有定位的父级（跟绝对定位类似）
   - 如果都没有则以浏览器文档为准

注意：offset家族返回不带单位的数字，而且都是只读的

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
事件委托	事件对象.target. tagName	得到目标元素
其他事件	load	加载事件，全部资源加载完毕
	DOMContentLoaded	加载事件，HTML文档加载完毕
	scroll	滚动事件
	resize	尺寸事件
元素尺寸与位置	scrollLeft 和 scrollTop	页面/元素被卷去的头部和左侧（滚动），可读写
	offsetLeft 和 offsetTop	元素位置距离定位父级左上距离，只读
	clientWidth 和 clientHeight	元素大小，不包含border、padding等，只读
	offsetWidth 和 offsetHeight	元素大小，包含border、padding等，只读

案例 吸附顶部导航栏案例

需求：当页面滚动到头部模块，导航栏自动滑入，否则滑出

分析：

①：用到页面滚动事件

②：检测页面滚动大于等于头部模块的位置，顶部导航栏则滑入显示，否则滑出隐藏

案例

实现bilibili 点击小滑块移动效果

需求：当点击链接，下面红色滑块跟着移动

分析：

①：用到事件委托

②：点击链接得到当前元素的 offsetLeft 值

③：修改 line 颜色块的 transform 值 = 点击链接的offsetLeft

④： 添加过渡效果

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
事件委托	事件对象.target. tagName	得到目标元素
其他事件	load	加载事件，全部资源加载完毕
	DOMContentLoaded	加载事件，HTML文档加载完毕
	scroll	滚动事件
	resize	尺寸事件
元素尺寸与位置	scrollLeft 和 scrollTop	页面/元素被卷去的头部和左侧（滚动），可读写
	offsetLeft 和 offsetTop	元素位置距离定位父级左上距离，只读
	clientWidth 和 clientHeight	元素大小，不包含border、padding等，只读
	offsetWidth 和 offsetHeight	元素大小，包含border、padding等，只读

目录

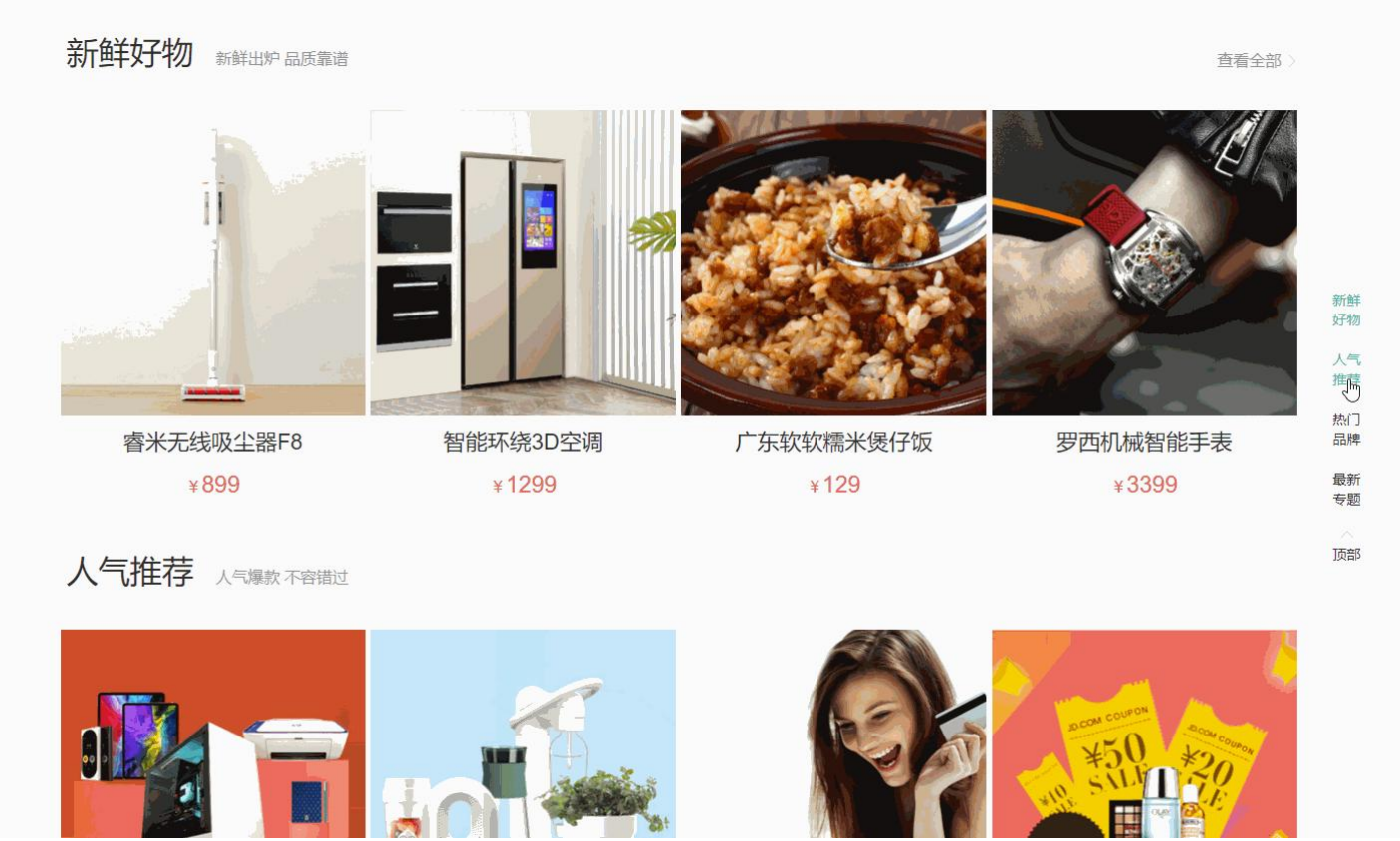
Contents

- 事件流
- 移除事件监听
- 其他事件
- 元素尺寸与位置
- 综合案例

案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置



案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置

业务分析：

1：点击导航对应小模块，页面会滚动到对应大模块位置

1.1 小模块高亮

1.2 页面滑动到对应大模块

2：页面滚动到大模块，电梯导航对应小模块会自动高亮

2.1 移除原先的小模块高亮

2.2 判断滑动大模块位置给小模块添加高亮

案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置

业务1分析：

1：点击导航对应小模块，页面会滚动到对应大模块位置

1.1 小模块高亮

1.2 页面滑动到对应大模块

1.1 小模块高亮效果

循环给所有小模块注册点击事件（此处不用事件委托，业务2后期需要）

(1)：点击小模块，当前添加 active这个类

(2)：移除以前的active类

案例

电梯导航

1.1 小模块高亮效果

要解决处理初次获取不到active 报错的问题

解决方案：

①： 不能直接获取这个类，然后移除，这样会报错

②： 先获取这个类，然后加个判断

- 如果有这个类，就移除
- 如果么有这个类，返回为 null， 就不执行移除，就不报错了
- 利用逻辑与中断即可

```
// 如果有active这个类，我就移除，没有这个类，则不需要移除
const old = document.querySelector('.xtx-elevator-list .active')  // 如果没有则返回为 null
console.log(old)
// document.querySelector('.xtx-elevator-list .active').classList.remove('active')
// 利用逻辑与中断
old && old.classList.remove('active')
this.classList.add('active')
```

案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置

业务1分析：

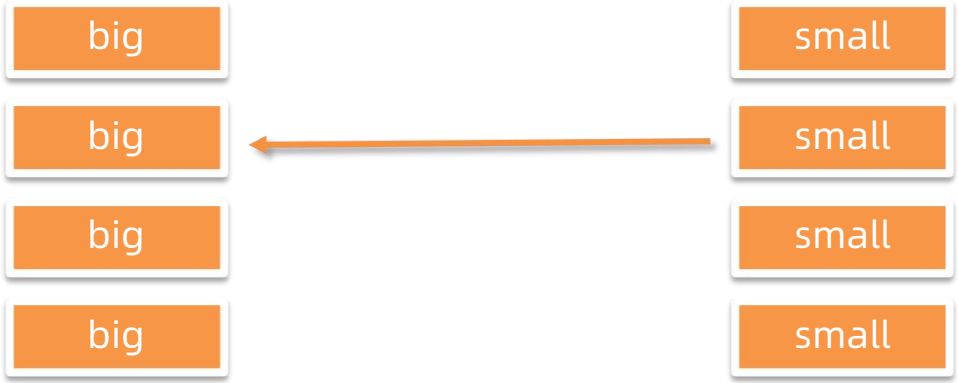
1：点击导航对应小模块，页面会滚动到对应大模块位置

1.1 小模块高亮

1.2 页面滑动到对应大模块

1.2 页面滑动到对应大模块

核心思想： document.documentElement.scrollTop 滑动到对应大盒子的offsetTop



案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置

业务分析：

1：点击导航对应小模块，页面会滚动到对应大模块位置

1.1 小模块高亮

1.2 页面滑动到对应大模块

2：页面滚动到大模块，电梯导航对应小模块会自动高亮

2.1 移除原先的小模块高亮

2.2 判断滑动大模块位置给小盒子添加高亮

案例

电梯导航

需求：点击不同的模块，页面可以自动跳转不同的位置

业务2分析：

2：页面滚动到大模块，电梯导航对应小模块会自动高亮

2.1 移除原先的小模块高亮

2.2 判断滑动大模块位置给小模块添加高亮

(1). 页面滚动先遍历所有的大模块

(2). 判断当前页面的滚动距离是否大于等于大模块的offsetTop，然后排他思想,选取对应小模快高亮

黑马程序员

www.itheima.com

传智教育旗下高端IT教育品牌