

全选文本框案例1

需求:用户点击全选,则下面复选框全部选择,取消全选则全部取消

分析:

①:全选复选框点击,可以得到当前按钮的 checked

②:把下面所有的小复选框状态checked,改为和全选复选框一致

🖳 全选	商品	商家	价格
~	小米手机	小米	¥ 1999
~	小米净水器	小米	¥ 4999
~	小米电视	小米	¥ 5999



全选文本框案例2

需求:用户点击全选,则下面复选框全部选择,取消全选则全部取消,文字对应变化

分析:

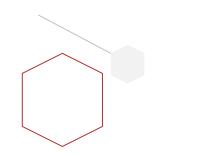
①:遍历下面的所有的checkbox,添加点击事件

②:检查小复选框选中的个数,是不是等于小复选框总的个数,

③: 把结果给 全选按钮

④: 利用css 复选框选择器 input:checked

🖔 全选	商品	商家	价格
~	小米手机	小米	¥ 1999
~	小米净水器	小米	¥ 4999
✓	小米电视	小米	¥ 5999



Web APIs 第三天

Dom事件进阶







- 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





- ◆ 事件流
- ◆ 事件委托
- ◆ 其他事件
- ◆ 元素尺寸与位置
- ◆ 综合案例





事件流

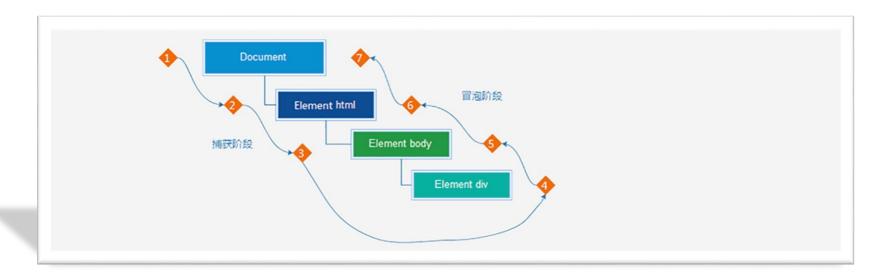
- 事件流与两个阶段说明
- 事件捕获
- 事件冒泡
- 阻止冒泡
- 解绑事件



1.1 事件流和两个阶段说明

目标: 能够说出事件流经过的2个阶段

● 事件流指的是事件完整执行过程中的流动路径



- 说明:假设页面里有个div,当触发事件时,会经历两个阶段,分别是捕获阶段、冒泡阶段
- 简单来说:捕获阶段是 从父到子 冒泡阶段是从子到父
- 实际工作都是使用事件冒泡为主





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
- 解绑事件



1.2 事件捕获

目标: 简单了解事件捕获执行过程

● 事件捕获概念:

从DOM的根元素开始去执行对应的事件 (从外到里)

- 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
 - ➤ 若是用 LO 事件监听,则只有冒泡阶段,没有捕获





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
- 解绑事件



1.3 事件冒泡

目标: 能够说出事件冒泡的执行过程

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
- L2事件监听第三个参数是 false,或者默认都是冒泡

```
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')
})
son.addEventListener('click', function () {
    alert('我是儿子')
})
```





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
- 解绑事件



1.4 阻止冒泡

目标: 能够写出阻止冒泡的代码

● **问题:** 因为默认就有冒泡模式的存在,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()
})
```



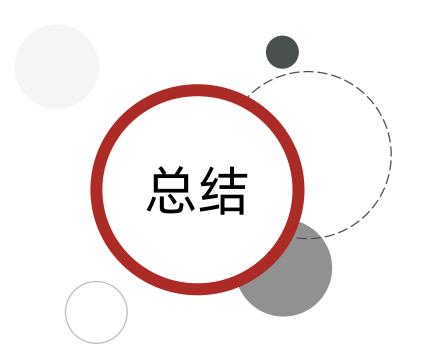
1.4 阻止冒泡

我们某些情况下需要阻止默认行为的发生,比如 阻止 链接的跳转,表单域跳转

● 语法:

e.preventDefault()





1. 阻止冒泡如何做?

事件对象.stopPropagation()

2. 阻止元素默认行为如何做?

e.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
- 解绑事件



1.5 解绑事件

on事件方式,直接使用null覆盖偶就可以实现事件的解绑

语法:

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



1.5 解绑事件

addEventListener方式,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])
例如:

注意: 匿名函数无法被解绑



鼠标经过事件的区别

- 鼠标经过事件:
 - ➤ mouseover 和 mouseout 会有冒泡效果
 - ➤ mouseenter 和 mouseleave 没有冒泡效果 (推荐)



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(同一个事件)
 - ▶ 直接使用null覆盖偶就可以实现事件的解绑
 - ▶ 都是冒泡阶段执行的
- 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(同一个事件)
 - 可以通过第三个参数去确定是在冒泡或者捕获阶段执行
 - ▶ 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)
 - ▶ 匿名函数无法被解绑





- ◆ 事件流
- ◆ 事件委托
- ◆ 其他事件
- ◆ 元素尺寸与位置
- ◆ 综合案例



目标: 能够说出事件委托的好处

委托:













- 1. 如果同时给多个元素注册事件, 我们怎么做的?
 - ➤ for循环注册事件
- 2. 有没有一种技巧 注册一次事件就能完成以上效果呢?

```
    *我是第1个小li
    *我是第2个小li
    *我是第3个小li
    *我是第4个小li
    *我是第5个小li
    **
```

```
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')
    })
}</pre>
```



目标: 能够说出事件委托的好处

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。

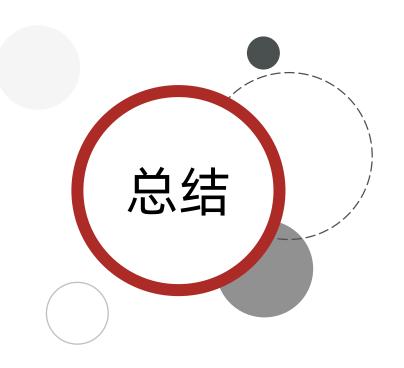
> 实现:事件对象.target.tagName 可以获得真正触发事件的元素

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

li 子元素 点击事件

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





- 1. 事件委托的好处是什么?
 - ▶ 减少注册次数,提高了程序性能
- 2. 事件委托是委托给了谁? 父元素还是子元素?
 - ▶ 父元素
- 3. 如何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'
    }
})
```





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







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: 给a的父级 注册点击事件,采取事件委托方式

②: 如果点击的是A,则进行排他思想,删除添加类

③: 注意判断的方式 利用 e.target.tagName

④: 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

⑤: 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







- ◆ 事件流
- ◆ 事件委托
- ◆ 其他事件
- ◆ 元素尺寸与位置
- ◆ 综合案例





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



3.1 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
})
```

● 注意:不光可以监听整个页面资源加载完毕,也可以针对某个资源绑定load事件

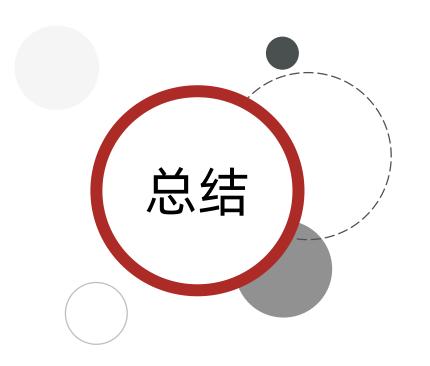


3.1 页面加载事件

- 当初始的 HTML 文档被完全加载和解析完成之后,DOMContentLoaded 事件被触发,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
})
```





- 1. 页面加载事件有哪两个? 如何添加?
 - ➤ 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





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

- 使用场景:
- 》 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?
- 就可以使用scroll 来检测滚动的距离~~~

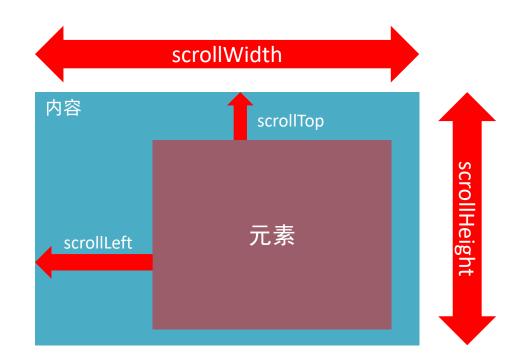


3.2 页面滚动事件-获取位置

- scrollLeft和scrollTop (属性)
 - > 获取被卷去的大小

 - ▶ 这两个值是可读写的
- 尽量在scroll事件里面获取被卷去的距离

```
div.addEventListener('scroll', function () {
    console.log(this.scrollTop)
})
```





3.2 页面滚动事件-获取位置

● 开发中,我们经常检测页面滚动的距离,比如页面滚动100像素,就可以显示一个元素,或者固定一个元素

```
// 页面滚动事件
window.addEventListener('scroll', function () {
   // document.documentElement 是html元素获取方式
   const n = document.documentElement.scrollTop
   console.log(n)
})
```

注意事项

document.documentElement HTML 文档返回对象为HTML元素

```
<html lang="en">

lang="en">

head>...</head>

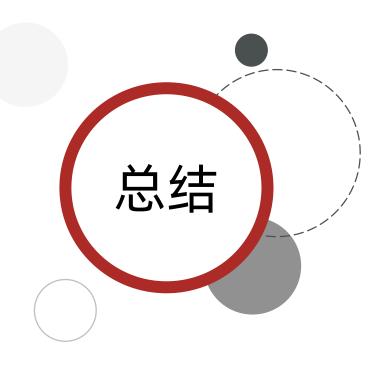
long="en">

head>...</head>

head>...</head>

head>...</head>
```





- 1. 被卷去的头部或者左侧用那个属性?是否可以读取和修改?
 - scrollTop / scrollLeft
 - ▶ 可以读取,也可以修改(赋值)
- 2. 检测页面滚动的头部距离(被卷去的头部)用那个属性?
 - document.documentElement.scrollTop

```
// 页面滚动事件
window.addEventListener('scroll', function () {
    // document.documentElement 是html元素获取方式
    const n = document.documentElement.scrollTop
    console.log(n)
})
```





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





国 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

②:检测页面被卷去的头部,如果大于300,就让侧边栏显示

③:显示和隐藏配合css过渡,利用opacity加渐变效果



3.2 页面滚动事件-滚动到指定的坐标

- scrollTo() 方法可把内容滚动到指定的坐标
- 语法: 元素.scrollTo(x, y)
- 例如:

// 让页面滚动到 y 轴1000像素的位置 window.scrollTo(0, 1000)





返回顶部

需求:点击返回按钮,页面会返回顶部







返回顶部

需求:点击返回按钮,页面会返回顶部

分析:

①:绑定点击事件

②:利用scrollTop让页面返回顶部





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
})
```

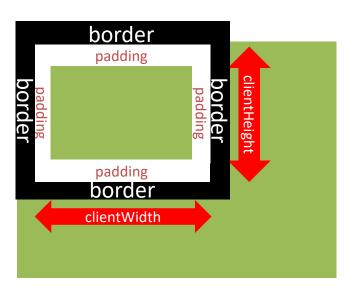
● 检测屏幕宽度:

```
window.addEventListener('resize', function () {
    let w = document.documentElement.clientWidth
    console.log(w)
})
```



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(不包含边框, margin, 滚动条等)
 - clientWidth和clientHeight







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)
```





- ◆ 事件流
- ◆ 事件委托
- ◆ 其他事件
- ◆ 元素尺寸与位置
- ◆ 综合案例





04 元素尺寸于位置



4. 元素尺寸于位置

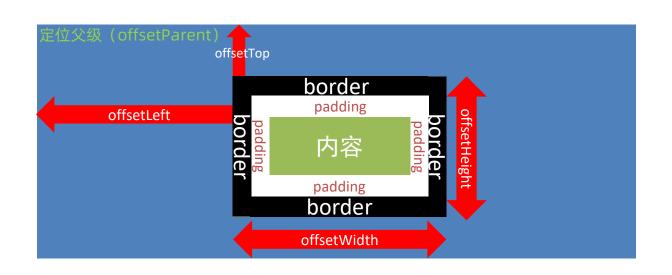
- 使用场景:
- ▶ 前面案例滚动多少距离,都是我们自己算的,最好是页面滚动到某个元素,就可以做某些事。
- ▶ 简单说,就是通过js的方式,得到元素在页面中的位置
- > 这样我们可以做,页面滚动到这个位置,就可以做某些操作,省去计算了





4. 元素尺寸于位置-尺寸

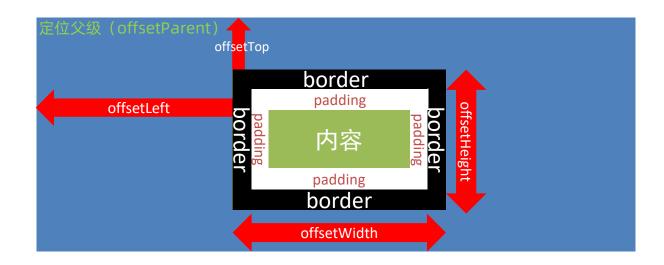
- 获取宽高:
 - ➤ 获取元素的自身宽高、包含元素自身设置的宽高、padding、border
 - ➤ offsetWidth和offsetHeight
 - ▶ 获取出来的是数值,方便计算
 - ▶ 注意: 获取的是可视宽高, 如果盒子是隐藏的, 获取的结果是0
- 获取位置:
 - ▶ 获取元素距离自己定位父级元素的左、上距离
 - ➤ offsetLeft和offsetTop 注意是只读属性



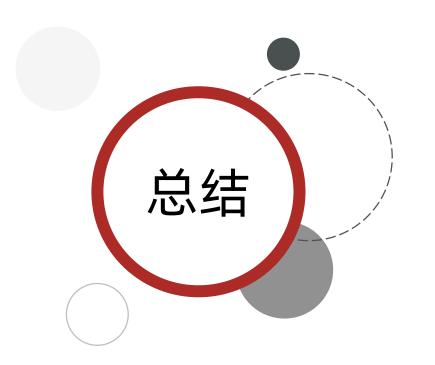


4. 元素尺寸于位置-尺寸

- 获取位置:
 - 1. offsetLeft和offsetTop 注意是只读属性
 - ▶ 获取元素距离自己定位父级元素的左、上距离







- 1. offsetWidth和offsetHeight是得到元素什么的宽高?
 - ➤ 内容 + padding + border
- 2. offsetTop和offsetLeft 得到位置以谁为准?
 - ▶ 带有定位的父级
 - ▶ 如果都没有则以 文档左上角 为准



国 案例

仿京东固定导航栏案例

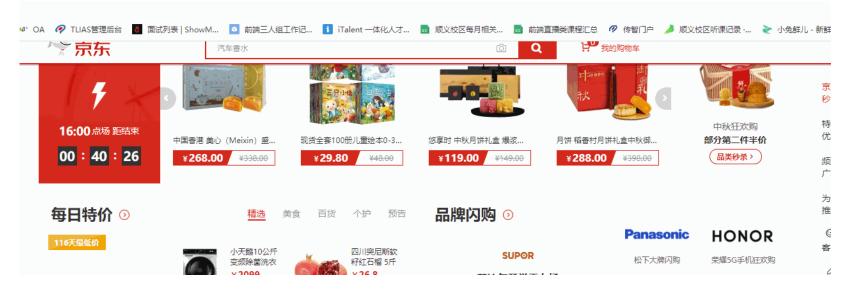
需求: 当页面滚动到秒杀模块, 导航栏自动滑入, 否则滑出

分析:

①:用到页面滚动事件

②:检测页面滚动大于等于 秒杀模块的位置 则滑入,否则滑出

③: 主要移动的是秒杀模块的顶部位置





1 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 left 值 = 点击链接的offsetLeft

④:添加过渡效果

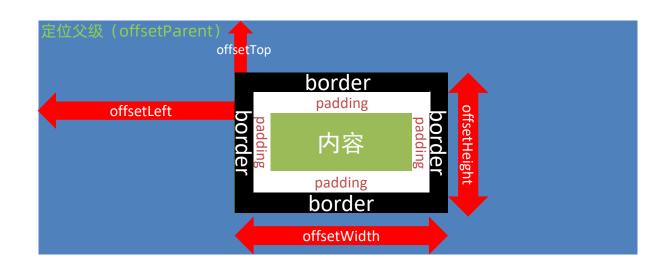




4. 元素尺寸于位置-尺寸

- 获取位置:
 - element.getBoundingClientRect()

方法返回元素的大小及其相对于视口的位置





总结:

属性	作用	说明
scrollLeft和scrollTop	被卷去的头部和左侧	配合页面滚动来用,可读写
clientWidth 和 clientHeight	获得元素宽度和高度	不包含border,margin,滚动条 用于js 获取元素大小,只读属性
offsetWidth和offsetHeight	获得元素宽度和高度	包含border、padding,滚动条等,只读
offsetLeft和offsetTop	获取元素距离自己定位父 级元素的左、上距离	获取元素位置的时候使用,只读属性



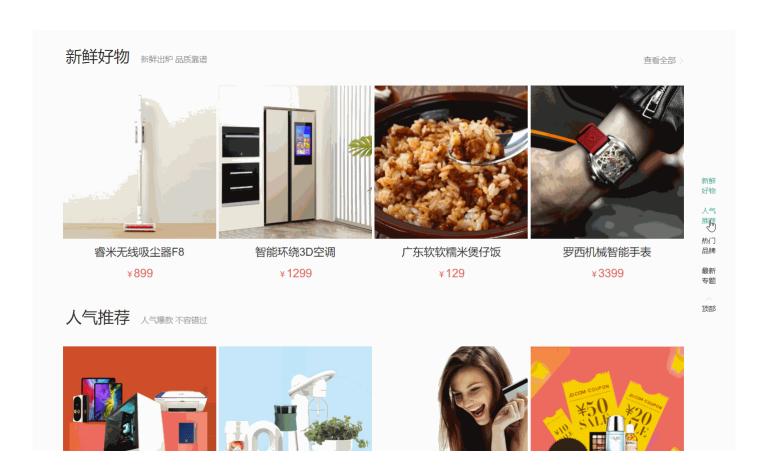


- ◆ 事件流
- ◆ 事件委托
- ◆ 其他事件
- ◆ 元素尺寸与位置
- ◆ 综合案例





需求:点击不同的模块,页面可以自动跳转不同的位置





国 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

模块分析:

①:页面滚动到对应位置,导航显示,否则隐藏模块

②:点击导航对应小模块,页面会跳到对应大模块位置

③:页面滚动到对应位置,电梯导航对应模块自动发生变化



国 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

模块分析:

①:显示隐藏电梯盒子和返回顶部已经完成,可以放到自执行函数里面,防止变量污染

②: 电梯模块单独放到自执行函数里面

③:点击每个模块,页面自动滚动到对应模块,使用事件委托方法更加简单

④:页面滚动到对应位置,电梯导航对应模块自动发生变化





需求:点击不同的模块,页面可以自动跳转不同的位置

模块分析:

①:显示隐藏电梯盒子和返回顶部已经完成,可以放到自执行函数里面,防止变量污染



1 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

模块分析2:点击每个模块,页面自动滚动到对应模块,使用事件委托方法更加简单

①:点击小模块,当前添加 active这个类

②: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





需求:点击不同的模块,页面可以自动跳转不同的位置

模块分析2:点击每个模块,页面自动滚动到对应模块,使用事件委托方法更加简单

点击小盒子li,页面跳转到对应大盒子位置

核心思想: 把对应大盒子的offfsetTop 给document.documentElement.scrollTop

①: 我们发现小盒子li 的自定义属性里面值 跟 大盒子 后面一致

②: 利用模板字符串 把点击的 自定义属性值 给 大盒子, 就找到对应的大盒子了,

③: 然后拿到这个大盒子的 offsetTop 值 给 document.documentElement.scrollTop 即可





需求:点击不同的模块,页面可以自动跳转不同的位置

模块分析3:页面滚动到大盒子位置,电梯导航小盒子对应模块自动处于选中状态

①: 当页面滚动了, 先移除所有小li 里面a 的状态 active 防止有多个 active





需求:点击不同的模块,页面可以自动跳转不同的位置

模块分析3:页面滚动到大盒子位置,电梯导航小盒子对应模块自动处于选中状态

①: 当页面滚动了, 先移除所有小li的状态

②: 因为页面滚动需要不断获取大盒子的位置, 所以需要把所有的大盒子都获取过来

③: 开始进行滚动判断

- 如果页面滚动大于 新鲜好物大盒子的offsetTop 并且小于 人气推荐盒子的offsetTop 就把对应的小盒子先出来添加类

- 依次类推

- 最后一个, 如果大于等于最新专题模块, 就选出最后一个对应小盒子(更精确)

新鲜 好物

人气 推荐

热门品牌

最新 专题

顶部





- 1. 整理今天笔记
- 2. 练习电梯导航模块、全选反选都要写3遍
- 3. 做测试题: PC: https://ks.wjx.top/vj/wmkituU.aspx
- 4. 作业



今天多一份拼搏,明日多一份欢笑



传智教育旗下高端IT教育品牌