DAY13 Event事件（下）

2023年5月8日　18:07

浏览器的默认行为

一.浏览器的默认行为

默认行为，就是不用我们注册，它自己就存在的事情

例如：

文本框可以输入内容

浏览器的右键菜单

a标签可以跳转

总结：这些不需要我们注册就能实现的事情，我们叫做默认事件

二.阻止默认行为或者取消浏览器的默认行为

我们有两个方法来阻止默认事件

e.preventDefault()：非IE使用(标准浏览器)

e.returnValue = false ： IE使用

事件流

页面元素接收事件的顺序

当元素触发一个事件的时候，其父元素也会触发相同的事件，父元素的父元素也会触发相同的事件.

1.目标

你是点击在哪个元素身上了，那么这个事件的目标就是这个元素

2.冒泡

就是从事件目标的事件处理函数开始，依次向外，直到 window 的事件处理函数触发

也就是从下向上的执行事件处理函数

3.捕获

就是从 window 的事件处理函数开始，依次向内，直到事件目标的事件处理函数执行

也就是从上向下的执行事件处理函数

addEventListener第三个参数：true捕获 false冒泡(默认)

事件排列的顺序

捕获 - 目标 - 冒泡

```
  <style>
    * {
      padding: 0;
      margin: 0;
    }
    .big {
      width: 400px;
      height: 400px;
      background-color: red;
      margin: 0 auto;
      padding: 100px;
      border-radius: 50%;
    }
    .middle {
      width: 200px;
      height: 200px;
      background-color: green;
      padding: 100px;
      border-radius: 50%;
    }
    .small {
      width: 200px;
      height: 200px;
      background-color: blue;
      border-radius: 50%;
      line-height: 200px;
      text-align: center;
      font-size: 50px;
    }
  </style>

<body>
  <div class="big">
    <div class="middle">
      <div class="small">small</div>
    </div>
  </div>
</body>

  var big = document.querySelector('.big');
  var middle = document.querySelector('.middle');
  var small = document.querySelector('.small');
```

```
small.addEventListener('click', function () {
  console.log('blue');
}, true);
middle.addEventListener('click', function () {
  console.log('green');
});
big.addEventListener('click', function () {
  console.log('red');
}, true);
document.body.addEventListener('click', function () {
  console.log('body');
})
document.documentElement.addEventListener('click', function () {
  console.log('html');
}, true)
document.addEventListener('click', function () {
  console.log('document');
}, true)

// 目标small
// 输出：document -> html -> red ->目标blue -> green ->body
```

事件流下面事件捕获和冒泡的取消(阻止)

1.事件流是页面元素接收事件的顺序

捕获 - 目标 - 冒泡

2.冒泡和捕获的区别

就是在事件的传播中，多个同类型事件处理函数的执行顺序不同

捕获：最不具体的对象(window)先接收事件，向下传播到最具体的对象(目标对象)

冒泡：最具体的对象(目标对象)先接收事件，向上传播到最不具体的对象(window)

3.阻止冒泡和捕获

取消冒泡：让当前操作的具体元素对象（冒泡元素）的事件不会冒泡到父级（外层） - 重点

取消捕获：让当前操作的不具体元素对象的事件不会捕获到具体的元素对象

IE下取消冒泡和捕获：event.cancelBubble = true;

标准取消冒泡和捕获：event.stopPropagation();

注意：事件冒泡和捕获的前提--事件和事件类型相同 父子元素关系。

注意：更加关注事件冒泡，普通事件事件绑定默认产生的，事件捕获需要通过事件绑定进行设置。

事件委托

就是把我要做的事情委托给别人来做.

因为我们的冒泡机制，点击子元素的时候，也会同步触发父元素的相同事件

所以我们就可以把子元素的事件委托给父元素来做

```
// 案例:
// 点击li元素，输出里面对应的内容
// var list = document.querySelectorAll('li');
// list.forEach(function (li) {
//   li.onclick = function () {//有多少个li绑定多少事件，假设100000li，绑定100000次
//     alert(li.innerHTML);
//   }
// });
```

换一种思路，利用冒泡机制，点击子元素的时候，也会同步触发父元素的相同事件

事件对象下面的属性：target

当你触发点击事件的时候，你点击在哪个元素上，target 就是哪个元素

```
// var ul = document.querySelector('ul');
// ul.onclick = function (e) {
//   e = e || event;
//   // console.log(e.target);//获取当前点击的目标元素(前提某个添加事件的元素内部的)
//   if (e.target.nodeName === 'LI') {//限定点击的子元素是li，才输出
//     alert(e.target.innerHTML)
//   }
// }
```

事件委托梳理与优缺点

事件委托的概念

利用冒泡的原理，将子元素的事件委托给父元素。

事件委托有关的属性

target：获取当前操作的目标元素(nodeName进行判断)

srcElement：IE浏览器获取目标元素的属性

为什么要用事件委托

子元素一般都是渲染(异步)进来的，一开始无法获取，但是父元素一定存在。

新创建的元素也可以拥有事件，新创建的元素也在某个父元素内部，委托父元素。

事件委托的优缺点

优点：减少事件绑定次数，提升性能，因为事件是绑定在父元素上，新加入的子元素同样拥有事件。

缺点：所有事件都用事件代理，可能会出现事件误判。即本不该被触发的事件被绑定上了事件。

例如：你给document添加点击事件，里面的子元素只要有点击事件都会触发document上面的事件，本不该触发了触发了。

H5拖拽与拖放的API

1.拖拽元素增加属性draggable="true"

2.在拖动目标上触发事件(拖拽的元素)

dragstart - 用户开始拖动元素时触发

drag - 元素正在拖动时触发，拖拽的过程(类似于move)

dragend - 用户完成元素拖动后触发

3.释放目标时触发的事件(目标元素)

ondragenter - 当被鼠标拖动的对象进入其容器范围内时触发此事件

ondragover - 当被拖动的对象在另一对象容器范围内拖动时触发(保证ondrop事件正常触发)

ondragleave - 当被鼠标拖动的对象离开其容器范围内时触发此事件

ondrop - 在一个拖动过程中，释放鼠标键时触发此事件(解除ondragover的默认行为才有效)

```
del.ondragover = function () {
```

```
    console.log('进入中...');
    return false;//ondrop事件的触发必须先解除默认行为
  }
```

封装事件库

一.封装事件库

移动端 tap 事件

click事件在移动端会有延迟(300ms)，主要原因是苹果手机在设计时，考虑到用户在浏览需要放大缩小。

touch事件是针对触屏手机上的触摸事件

tap事件在移动端，代替click作为点击事件，tap事件不是原生的，所以需要独立封装

二.移动端事件类型

touchstart：触摸开始事件

touchend： 触摸结束事件

touchmove： 触摸移动事件

三.代码演示

这就是移动端的单击事件-也叫tap事件

targetTouches : 手指(数组) targetTouches[0] 一根手指

clientX : 位置

changedTouches :松开时的位置坐标

单击操作的特点

1.单击只有一根手指

2.判断手指开始触摸和手指松开的时间差异不能大于指定的值(100ms)

3.保证没有滑动操作，如果有抖动必须保证抖动的距离在指定的范围内。

```
<style>
  .box {
    width: 300px;
    height: 300px;
    background-color: red;
  }
</style>
```

```
<body>
  <div class="box"></div>
</body>

  var box = document.querySelector('.box');
  var startTime, startX, startY;
  box.addEventListener('touchstart', function (e) {
    e = e || event;
    // 判断是否只有一根手指进行操作
    // 说明不止一个手指
    if (e.targetTouches.length > 1) return;
    // 获取当前时间
    // startTime = new Date();
    startTime = Date.now();
    // 获取当前手指开始点击的坐标
    startX = e.targetTouches[0].clientX;
    startY = e.targetTouches[0].clientY;
  });
  box.addEventListener('touchend', function (e) {
    e = e || event;
    // 判断是否只有一根手指进行操作
    // 说明不止一个手指
    if (e.targetTouches.length > 1) return;
    // 松开手指时的时间不能大于指定的时间值
    if (new Date() - startTime > 100) return;
    // 松开手指的新的坐标
    var endX = e.changedTouches[0].clientX;
    var endY = e.changedTouches[0].clientY;
    if (Math.abs(endX - startX) < 5 && Math.abs(endY - startY) < 5)
{
      console.log('这就是移动端的单击事件-也叫tap事件');
    }
  });
```