DAY12 Event事件（上）

2023年5月6日 19:01

事件绑定

一.普通事件(DOM 0级事件)

1.普通事件：一个元素对象上面只能绑定一个事件处理函数，如果多个下面覆盖上面。

普通事件也叫DOM 0级事件

弊端：普通事件不适合项目协作，会有冲突的

```
// document.onclick = function () {//被下面覆盖
//   console.log('张三的事件');
// };
// document.onclick = function () {//会触发
//   console.log('李四的事件');
// };
```

2.取消普通事件：普通事件无法触发,将null赋值给事件

document.onclick = null;

二.事件绑定(DOM 2级事件)

1.事件绑定：一个元素对象上面可以绑定多个事件处理函数，按照顺序执行。

事件绑定也叫DOM 2级事件

非常适合项目协作

事件绑定的语法：元素对象.addEventListener(事件类型,事件处理函数,是否捕获);

事件类型：不能带on，比如click

事件处理函数：函数表达式或者函数名称

是否捕获：可选的参数，默认是冒泡(false) 捕获true

```
// document.addEventListener('click', function () {
//   console.log('张三的事件');
// });
// document.addEventListener('click', function () {
//   console.log('李四的事件');
// });
```

上面绑定的事件无法取消，因为匿名函数无法比较

```
document.removeEventListener('click', function () {//无效
  console.log('张三的事件');
});

document.removeEventListener('click', function () {//无效
```

```
  console.log('李四的事件');
});

// function zhangsan() {
//   console.log('张三的事件');
// }
// function lisi() {
//   console.log('李四的事件');
// }
// document.addEventListener('click', zhangsan);
// document.addEventListener('click', lisi);
```

2.取消事件绑定：让绑定的事情不再触发

语法格式：元素对象.removeEventListener(事件类型,事件处理函数,是否捕获)

注意：如果绑定的事件要取消，必须将事件处理函数写成有名称的函数(将函数单独列出来)，因为匿名函数无法比较

```
// console.log(function () { } == function () { });//false   比较地址
// console.log([] == []);
// console.log({} == {});
```

总结：

所有的普通事件可以改写成事件绑定，但是反之不行

项目开发一般采用事件绑定，防止冲突，测试可以自由使用普通事件或者事件绑定

事件对象概念

一.事件对象的概述

1.在触发DOM上的某个事件(元素)时，会产生一个事件对象event，这个对象包含着所有事件相关的信息。

2.该事件的本质就是一个函数,而该函数的形参接收一个event对象(事件处理函数的第一个参数就是事件对象)

3.注意：事件通常与函数结合使用，函数不会在事件发生前被执行.

```
// document.onclick = function (e) {//e:事件对象
//   console.log(e);
// }
```

二.一个事件的组成

触发谁的事件：事件源

触发什么事件：事件类型

触发以后做什么：事件处理函数

```
// function fn(e) {//事件处理函数
//   console.log(e);
```

```
// }
// document.onclick = fn;    fn事件处理函数    document:事件源    click:事件类型
// fn();    fn是普通函数
// document.addEventListener('click', function (e) {//事件处理函数
//    console.log(e);
// })
// document.addEventListener('click', fn);//fn事件处理函数
```

获取事件对象

1.标准浏览器：事件对象是事件处理函数的第一个参数

```
// document.onclick = function (e) {
//    console.log(e);
// }
```

2.非标准浏览器：事件对象是自带的，window下面的event子对象，标准浏览器也支持event。

```
// document.onclick = function (e) {
//    console.log(window.event);
//    console.log(e);
// }
```

最终的写法 - 推荐的写法

标准浏览器优先，非标准浏览器不支持第一个参数，非标准支持event

兼容的写法

```
// document.onclick = function (e) {
//    e = e || event; //标准的写法
//    console.log(e);
// };
```

事件类型

1.浏览器事件

load
scroll
resize

2.鼠标事件

click： 鼠标单击事件
dblclick： 双击事件
contextmenu： 右键单击事件
mousedown： 鼠标左键按下事件
mouseup： 鼠标左键抬起事件
mousemove： 鼠标移动
mouseover： 鼠标移入事件
mouseout： 鼠标移出事件

3.键盘事件

keyup： 键盘抬起事件

keydown： 键盘按下事件

keypress： 键盘按下再抬起事件

注意：键盘事件针对document或者表单等，因为键盘是固定的，不可以对某个盒子进行键盘操作。

4.表单事件

focus: 得到焦点,光标停止表单内

blur: 失去焦点,光标离开表单内

change: 内容改变，失去焦点触发

input: 内容改变即触发

submit: 提交事件

注意：submit事件针对form表单，不是当前的按钮

事件对象下面的属性

1.button属性：获取鼠标按键编码(分辨鼠标的左中右键) - 了解

```
// document.onmousedown = function (e) {
//    e = e || event;//获取事件对象
//    console.log(e.button);//使用事件对象下面的button属性
//    // 0,1,2分别表示鼠标的左中右三个键
// }
```

2.which属性：获取鼠标的按键编码以及键盘的按键编码 - 重点（keyCode也可以获取键盘的键码)

键盘上面的没个按键都对应一个编码(unicode编码)

A-Z 65-90

空格键:32

回车键:13

```
// document.onmousedown = function (e) {
//    e = e || event;//获取事件对象
//    console.log(e.which);// 1,2,3分别表示鼠标的左中右三个键
// }
// document.onkeydown = function (e) {
//    e = e || event;//获取事件对象
//    console.log(e.which);
// }
```

3.组合按键最主要的就是 alt / shift / ctrl 三个按键

事件对象里面也为我们提供了三个属性

altKey： alt 键按下得到 true，否则得到 false

shiftKey： shift 键按下得到 true，否则得到 false

ctrlKey： ctrl 键按下得到 true，否则得到 false

```
// 例子ctrl+enter键，组合
// document.onkeydown = function (e) {
//   e = e || event;//获取事件对象
//   if (e.ctrlKey && e.which === 13) {
//     console.log('ctrl+enter触发');
//   }
// }
// shift + alt + A
// document.onkeydown = function (e) {
//   e = e || event;//获取事件对象
//   if (e.shiftKey && e.altKey && e.which === 65) {
//     console.log('shift + alt + A触发');
//   }
// }
```

事件对象其他属性介绍

1.type:获取当前操作的事件类型 获取的事件类型不显示on

```
// document.onclick = function (e) {
//   e = e || event;
//   console.log(e.type);//click
// };

// document.addEventListener('click', function (e) {
//   e = e || event;
//   console.log(e.type);//click
// });
```

2.鼠标事件中获取鼠标的位置属性

2.1.clientX, clientY：鼠标相对于可视区的位置。

```
// document.onmousemove = function (e) {
//   e = e || event;
//   console.log(e.clientX, e.clientY);
// }
```

2.2.pageX, pageY：鼠标相对于文档的位置。

```
// document.onmousemove = function (e) {
//   e = e || event;
//   console.log('可视区' + e.clientX, e.clientY);
//   console.log('文档' + e.pageX, e.pageY);
// };
```

2.3.offsetX, offsetY：鼠标相对于操作元素(鼠标位置)到元素边缘(左上)的位置。

注意：offset位置可以通过鼠标的位置减去盒子的定位求到。

```
// var box = document.querySelector('.box');
```

```
// box.onmousemove = function (e) {
//   e = e || event;
//   console.log(e.offsetX, e.offsetY);
// }
```

2.4.screenX, screenY : 鼠标相对于屏幕的位置。

```
// document.onmousemove = function (e) {
//   e = e || event;
//   console.log(e.screenX, e.screenY);
// }
```