js五

什么是事件？说白了指的是一件事

如下这些事件都是已存在的，系统处理好的

- 鼠标事件 click点击， mouseover mouseout mouseenter mousemove mouseleave mousewheel

鼠标事件

属性	描述
onclick	当用户点击某个对象时调用的事件句柄。
oncontextmenu	在用户点击鼠标右键打开上下文菜单时触发
ondblclick	当用户双击某个对象时调用的事件句柄。
onmousedown	鼠标按钮被按下。
onmouseenter	当鼠标指针移动到元素上时触发。
onmouseleave	当鼠标指针移出元素时触发
onmousemove	鼠标被移动。
onmouseover	鼠标移到某元素之上。
onmouseout	鼠标从某元素移开。
onmouseup	鼠标按键被松开。

- 键盘事件 keydown(按下时触发) keyup(抬起来时触发) keypress(按下并抬起来时触发)

键盘事件

属性	描述
onkeydown	某个键盘按键被按下。
onkeypress	某个键盘按键被按下并松开。
onkeyup	某个键盘按键被松开。

- 系统事件 resize(浏览器窗口发生改变时触发) scroll(滚动条发生改变) DOMContentLoaded (dom元素加载完了后再触发)

框架/对象（Frame/Object）事件

属性	描述
onabort	图像的加载被中断。（<object>）
onbeforeunload	该事件在即将离开页面（刷新或关闭）时触发
onerror	在加载文档或图像时发生错误。（<object>, <body>和 <frameset>）
onhashchange	该事件在当前 URL 的锚部分发生修改时触发。
onload	一张页面或一幅图像完成加载。
onpageshow	该事件在用户访问页面时触发
onpagehide	该事件在用户离开当前网页跳转到另外一个页面时触发
onresize	窗口或框架被重新调整大小。
onscroll	当文档被滚动时发生的事件。
onunload	用户退出页面。（<body> 和 <frameset>）

- 表单元素 change(表单元素发生改变时触发) input(移动端，表单输入时触发) blur(失去焦点，光标离开表单元素时触发) focus(光标在表单元素时触发)

表单事件

属性	描述
onblur	元素失去焦点时触发
onchange	该事件在表单元素的内容改变时触发（<input>, <keygen>, <select>, 和 <textarea>）
onfocus	元素获取焦点时触发
onfocusin	元素即将获取焦点时触发
onfocusout	元素即将失去焦点时触发
oninput	元素获取用户输入时触发
onreset	表单重置时触发
onsearch	用户向搜索域输入文本时触发（<input="search">）
onselect	用户选取文本时触发（<input> 和 <textarea>）
onsubmit	表单提交时触发

- 拖拽事件

拖动事件

事件	描述
ondrag	该事件在元素正在拖动时触发
ondragend	该事件在用户完成元素的拖动时触发
ondragenter	该事件在拖动的元素进入放置目标时触发
ondragleave	该事件在拖动元素离开放置目标时触发
ondragover	该事件在拖动元素在放置目标上时触发
ondragstart	该事件在用户开始拖动元素时触发
ondrop	该事件在拖动元素放置在目标区域时触发

事件绑定 ：事件（行为）发生时做些具体的事情（给事件绑定行为）

- 事件绑定分为DOM0级事件绑定和DOM2级事件绑定

  > DOM0级事件绑定问题：对同一元素，同一事件绑定多个行为时，后面的行为会覆盖前面的行为，最终只执行最后一次绑定的行为
  > DOM0级事件绑定行为移除 ： oDiv.onclick = null

- 标准浏览器 DOM2级事件绑定 addEventListener
  第一个参数事件类型 第二个参数 事件绑定的行为 第三个参数 事件传播的方式
  oDiv.addEventListener(“click”,fn,false);
  oDiv.addEventListener(“click”,fn,false);//处理了重复绑定

  IE6~8 DOM2级事件绑定 attachEvent
  oDiv.attachEvent(“onclick”,fn);
  oDiv.attachEvent(“onclick”,fn);//执行了两次，没有做重复绑定处理
  标准浏览器绑定的是谁this就是谁，ie浏览器this是window
  ie下事件绑定的顺序是乱的，标准浏览器是按绑定的先后顺序执行
- 标准浏览器 DOM2事件移除 removeEventListener
  oDiv.removeEventListener(“click”,fn,false);

  IE6~8 DOM2级事件移除 detachEvent
  oDiv.detachEvent(“onclick”,fn)

事件对象 e

鼠标/键盘事件对象

属性

属性	描述	DOM
altKey	返回当事件被触发时，"ALT" 是否被按下。	2
button	返回当事件被触发时，哪个鼠标按钮被点击。	2
clientX	返回当事件被触发时，鼠标指针的水平坐标。	2
clientY	返回当事件被触发时，鼠标指针的垂直坐标。	2
ctrlKey	返回当事件被触发时，"CTRL" 键是否被按下。	2
Location	返回按键在设备上的位置	3
charCode	返回onkeypress事件触发键值的字母代码。	2
key	在按下按键时返回按键的标识符。	3
keyCode	返回onkeypress事件触发的键的值的字符代码，或者 onkeydown 或 onkeyup 事件的键的代码。	2
which	返回onkeypress事件触发的键的值的字符代码，或者 onkeydown 或 onkeyup 事件的键的代码。	2
metaKey	返回当事件被触发时，"meta" 键是否被按下。	2
relatedTarget	返回与事件的目标节点相关的节点。	2

screenX	返回当某个事件被触发时，鼠标指针的水平坐标。	2
screenY	返回当某个事件被触发时，鼠标指针的垂直坐标。	2
shiftKey	返回当事件被触发时，"SHIFT" 键是否被按下。	2

- 浏览器记录了事件相关的信息
- 标准浏览器：形参e(arguments[0]) IE :window.event
- e.clientX 到可视窗左边的距离
- e.clientY 到可视窗口上边的距离
- e.pageX || (e.clientY+(document.documentElement.scrollLeft || document.body.scrollLeft)) 到文档左边的距离
- e.pageY || (e.clientY+(document.documentElement.scrollTop||document.body.scrollTop)) 到文档上边的距离
- e.type 事件类型
- e.target||e.srcElement 事件源
- e.preventDefault() 标准下 e.returnValue = false; IE下 阻止默认行为
- e.stopPropagation() 标准下 e.cancelBubble = true; IE下 阻止事件冒泡

promise

Promise是构造函数 new 去运行
参数是一个回调函数,回调函数有两参数，resolve(执行成功的回调) reject(执行失败的回调)
then 是Promise类原型上的方法 ，通过实例调用 then有两个参数，第一个表示成功回调 ，第二个参数表示失败回调 then方法返回的是新的promise实例，所以可以继续调用then方法(类似于jquery的链式写法)
promise有三种状态 第一种是pending(等待) 第二种是fulfilled （ 成功状态 ） 第三种状态 rejected （ 失败 ）

```
let p =  new Promise(function(resolve,reject){
   window.setTimeout(function(){//属于是等待状态，要写成异步的
    //resolve("a");
    reject();
  },1000)
});
p.then(function(data){
   console.log(data);
  },function(){
// console.log("error1");
throw new Error("error1");//只有手动抛出异常状态才会变成失败的状态，其他都是成功状态  （前提是没有再返回promise对象）
}).then(function(){
        console.log("b");
    },function(){
        console.log("error2")
    })
```

阻止默认行为

var oA = document.querySelector(“a”);
oA.onclick = function(e){
alert(1);
//return false; //阻止默认行为
e.preventDefault();
}
oA.addEventListener(“click”,function(e){
alert(1);
e.preventDefault(); //DOM2级事件绑定只能通过e.preventDefault()来阻止
},false)

事件传播

- 事件传播：冒泡和捕获

- 事件冒泡：当前元素事件被触发后，其祖先元素的相同事件也会被触发 由内往外
- 事件捕获：当前元素事件被触发后，其祖先元素的相同事件也会被触发 由外往里 (标准浏览器下DOM2级事件绑定的方式才有) 基本上不用

事件委托

1.若遍历元素比较多时，性能会比较低
2.后期再添加新元素时 ,之前元素所处理的逻辑会失效
利用事件冒泡的机制，由于事件会传播，触发当前元素事件时 ，最终会触发祖先元素，所以可以给祖先元素绑定事件，也就是说把事件的绑定委托在祖先元素上 ->事件委托
事件委托是高性能解决事件绑定的方案,解决了动态绑定问题

订阅发布模式

- 让代码具有可维护性，可复用性，可移植性
- 不再单纯的专注于代码本身，站在宏观角度思想代码，想着如何规划和管理代码
- 低耦合，高内聚
  - 低耦合：每个模块之间的代码没有关联性
  - 高内聚：每个模块内部是由关联性很强代码组成，都是用来实现单一个功能，得遵守单一职责原则
- 如何将每个单独的功能在需要执行的地方执行了？->订阅发布模式

订阅发布模式 - >订阅 和发布

- 订阅 ->做计划
- 发布 ->执行计划
- 取消订阅->取消计划

```
function fn1(){alert(1)}
function fn2(){alert(2)}
    document.addEventListener("click",fn1,false)
    document.addEventListener("click",fn2,false)
    document.removeEventListener("click",fn2,false)
```

- 鼠标点击时这件事发生时 事件类型
- 做计划 对事件添加绑定的行为 fn1,fn2
- 取消计划 对事件解除绑定的行为 fn2
- 做计划 用户点击时

```
function on(ele,type,fn){
有个事件池，里面放着跟事件相关的行为
```

```
如何定义一个跟事件类型相关的事件池（数组） ele["my"+type]
        if(!ele["my"+type]){
            ele["my"+type] = [];
        }
        var a =  ele["my"+type];
        for(var i = 0;i<a.length;i++){//处理重复绑定
            if(a[i]===fn){
             return;
            }
        }
       a.push(fn);//把行为放入事件池中
    }
```

```
function off(ele,type,fn){
1.先获取事件池
        var a = ele["my"+type];
遍历事件池，查看是否有这个行为（计划）
        if(a&&a.length>0){
            for(var i = 0;i<a.length;i++){
                if(a[i]===fn){
                    a[i] = null;//防止数组发生塌陷，假移除
                }
            }
        }
    }
```

```
function fire(type){//发布计划  把事件池中的行为执行
        var a = this["my"+type];
        if(a&&a.length>0){
            for(var i = 0;i<a.length;i++){
                if(typeof a[i] ==="function"){
                    a[i].call(this);//把事件的行为执行，并改变绑定行为中
的this关键字
                }else{
                    a.splice(i,1);
                    i--;
                }
            }
        }
    }
```

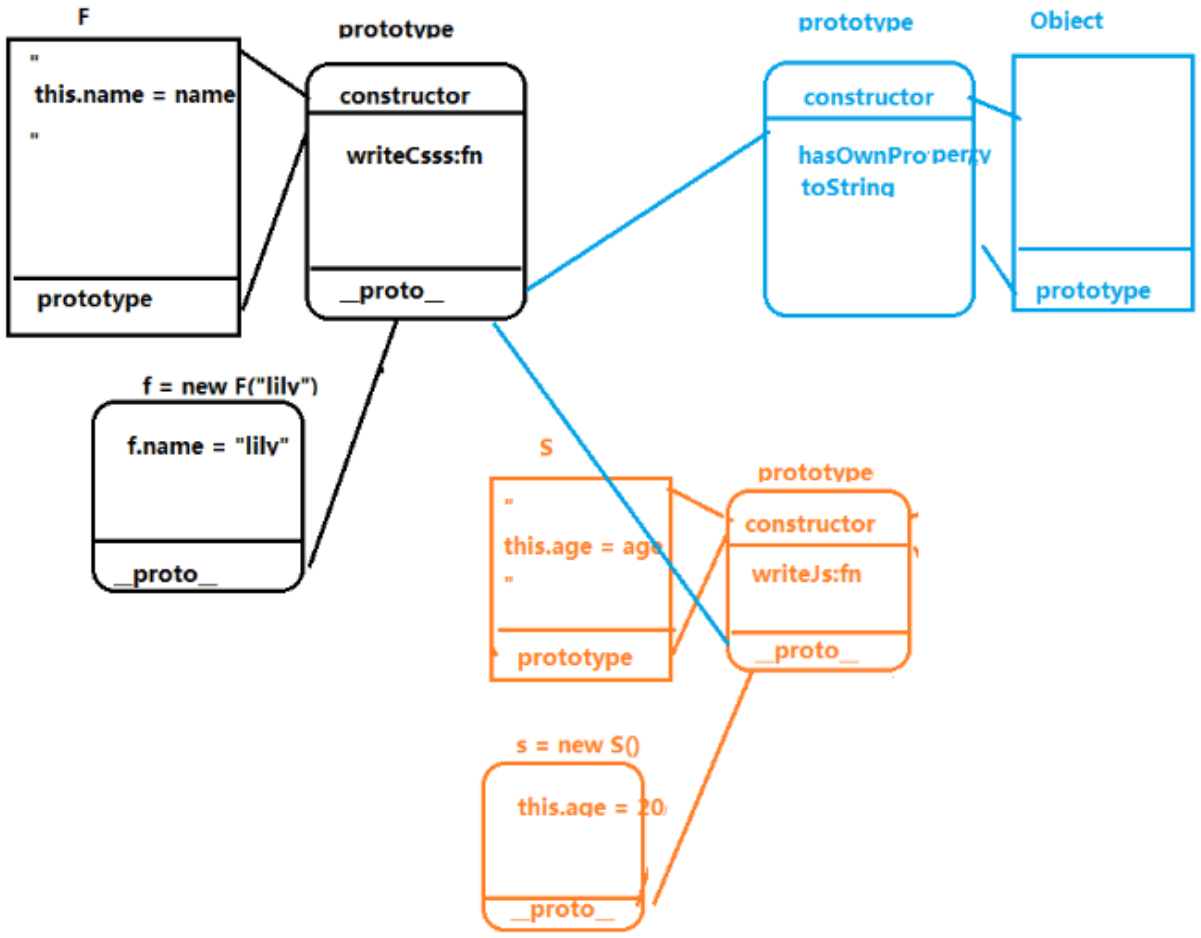
- 做计划 on
- 取消计划 off
- 执行计划 fire

```
function fn1(){
        alert("给红包")
    }
function fn2(){
        alert("喝喜酒")
    }
function fn3(){
        alert("抢新娘")
    }
```

```
var  oDiv = document.getElementById("div1");
on(oDiv,"marry",fn1);
on(oDiv,"marry",fn2);
on(oDiv,"marry",fn3);
off(oDiv,"marry",fn3);
```

```
window.setTimeout(function(){
发布计划
fire.call(oDiv,"marry");
},2000);
```

继承


F
"
this.name = name
"
prototype
prototype
constructor
writeCsss:fn
__proto__
prototype
constructor
hasOwnProperty
toString
Object
prototype
f = new F("lily")
f.name = "lily"
__proto__
S
"
this.age = age
"
prototype
prototype
constructor
writeJs:fn
__proto__
s = new S()
this.age = 20
__proto__