day12

HTML DOM

使用点操作符 访问和设置元素的 属性

普通设置属性

```
var img = document.createElement('img');
img.setAttribute('src', '1.jpg');
console.log(img);
```

HTMLdom设置属性

```
var img = new Image();
img.src = '2.jpg';
console.log(img);
```

核心DOM和HTML DOM的区别

- 核心DOM
  - 核心DOM(标准DOM) 适用于节点操作
  - 可以创建 修改 删除 查询 节点
- HTML DOM
  - HTML DOM 适用于属性的访问和修改

css属性-1

通过element.style.cssText

- 获得元素的行内样式文本
- 添加一个字体样式

```
var box = document.querySelector('#box');
// 获得元素的行内样式文本
console.log(box.style.cssText);
// 添加一个字体样式
box.style.cssText += 'font-size:2em;';
```

2.style行内样式的修改

```
console.log(box.style); // 对象 类数组对象
for (var i = 0; i < box.style.length; i++) {
    console.log(box.style[i]); // 属性名
    console.log(box.style[box.style[i]]); // 属性值
}
console.log(box.style.fontSize);  // 获得单个属性 有多个单词 使用驼峰命名法
console.log(box.style.backgroundColor);
console.log(box.style['background-color']);

box.style.backgroundColor = 'yellow';
```

css属性-2

获取元素的宽高，返回获得元素实际大小

- offsetWidth/offsetHeight
- 标准盒模型 content+padding+border
- 怪异盒模型（IE盒模型） content

获得元素的定位位置(偏移量)

- offsetTop/offsetLeft
- 注意
  - 属性受margin影响
  - 受父元素边框和内填充影响，如果自己设置了absolute，那么就不受父元素的padding影响
  - 如果父元素定位了 根据父元素的左上角开始计算位置
  - 如果父元素没有定位 根据页面左上角开始计算位置

滚动距离

- 获得滚动条距离页面顶部的距离 document.documentElement.scrollTop
- window.onscroll 滚动条 滚动事件
- window.onresize 修改窗口大小 事件

获得计算后样式

注意：获得计算后样式，有兼容问题

DOM标准方法

- getComputedStyle(elm)['styleName']; 例子

```
console.log(getComputedStyle(box)['width']);
```

- elm.currentStyle['styleName']; 例子

```
console.log(box.currentStyle['width']);
```

注意：低版本ie（ie9）以下不兼容

文档碎片

- document.createDocumentFragment();
- 文档碎片(用于创建DOM结构)
- 优化DOM操作
- 设计的意义是 减少DOM操作的频次
- 常见DOM操作 创建 查找 修改 删除 替换

```
var f = document.createDocumentFragment();
for (var i = 0; i < 50000; i++) {
    var tempLi = document.createElement('li');
    tempLi.innerHTML = '内容' + i;
    f.appendChild(tempLi);
}
list.appendChild(f);
```

DOM0级事件

DOM0级事件处理(事件绑定)

- 将一个函数 赋值给一个事件处理的属性
- DOM0级没有兼容问题 所有浏览器都支持
- DOM0级事件只能绑定 一个事件处理函数

```
btn.onclick = function() {
    alert(1);
}

btn.onclick = function() {
    alert(2);
}
```

代码上述情况：方法2会覆盖方法1，最后输出2

- 移除事件 将事件属性赋值为null

```
btn.onclick = null;
```

这段代码如果放到.onclick函数内部，当鼠标点击一次后就把方法清空了

DOM2级事件

DOM2级事件(事件监听)

- 注意
  - DOM2级可以为一个元素的相同事件 添加多个事件处理函数
  - 多个事件处理函数的执行顺序是按照添加顺序执行的
  - dom2级事件名没有on
- elm.addEventListener(eventType,fn,boolean);
  - 参数
    - eventType 事件类型 string
    - fn 事件触发时要执行的函数 function
    - boolean 冒泡(false)/捕获(true)

```
btn.addEventListener('click', function() {
    alert(123);
});
```

- elm.removeEventListener(eventType,fnName);
  - 参数
    - eventType 事件类型 string
    - fnName 函数 function

```
btn.addEventListener('click', function() {
    alert(456);
    btn.removeEventListener('click', arguments.callee);
    console.log(arguments.callee);
});
```

arguments.callee 用来指定当前执行的函数

IE事件处理

IE添加事件 elm.attachEvent(eventType,fn)

- 参数
  - eventType 事件类型 string
  - fn 事件触发需要执行的函数 function

IE移除事件 elm.detachEvent(eventType,fn);

- 参数
  - eventType 事件类型 string
  - fn 事件触发需要执行的函数 function

事件处理兼容

兼容注意

- 兼容原则 渐进真增强/优雅降级
- 减少 冗(rong)余代码
- 兼容事件 不要用 querySelector

兼容代码

```
function addEvent(elm, type, fn) {
    if (elm.addEventListener) { // 判断是否支持该函数
        elm.addEventListener(type, fn);
    } else if (elm.attachEvent) {
        elm.attachEvent('on' + type, fn);
    }
}

function removeEvent(elm, type, fn) {
    if (elm.removeEventListener) {
        elm.removeEventListener(type, fn);
    } else if (elm.detachEvent) {
        elm.detachEvent('on' + type, fn);
    }
}
```

惰性函数

定义

- 惰性函数 是一种函数的高阶用法
- 函数的作用是需要执行一次才能确定

```
function addEvent(elm, type, fn) {
    if (elm.addEventListener) {
        addEvent = function(elm, type, fn) {
            elm.addEventListener(type, fn);
        }
    } else if (elm.attachEvent) {
        addEvent = function(elm, type, fn) {
            elm.attachEvent('on' + type, fn);
        }
    }
    addEvent(elm, type, fn);
}
```

惰性函数应用

上述简便算法

```
// 终极版本
var addEvent = (function() {
    if (document.addEventListener) {
        return function(elm, type, fn) {
            elm.addEventListener(type, fn);
        }
    } else if (document.attac (parameter) type: any
        return function(elm, type, fn) {
            elm.attachEvent('on' + type, fn);
        }
    }
})();
```