# Javascript 基础

console.log()　是在火狐浏览器控制台输出日志 ,IE 浏览器默认不支持 console.log()

alert()　　弹出消息对话框，并且 alert 消息对话框通常用于一些对用户的提示信息

## 1、 JS 与 DOM 的关系

浏览器有渲染 html 代码的功能，把 html 源码在内存里形成一个 DOM 对象，就是文档对象

浏览器内部有一个 JS 的**解释器/执行器/引擎**、如 chrome 用 V8 引擎

我们在 html 里写一个 JS 代码，JS 代码被引擎所执行，而执行的结果就是对 DOM 的操作

而对学习 DOM 操作的结果，就是我们常常看到的特效，比如图片漂浮，文字变色

学习 Javascript 要分清

1. js 语言本身的语法
2. DOM 对象（把 body,div,p 等节点树看成一个对象）
3. BOM 对象（把浏览器的地址栏，历史记录，DOM 看成一个对象）

浏览器是宿主，但 JS 的宿主不限于浏览器，也可以是服务器端 (NodeJS)

## 2、 JS 如何引入

1、 页内的 script 代码

```
<script>
code.....
</script>
```

2、 外部 js 文件

```
<script src="js/jquery.js"></script>
```

注意：外部的.js 文件里面直接写 js 代码

**不要在开头和结尾加标签**

3、 错误的引入方式

```
<script src="xx.js">
var _name ='张三';
alert(_name);
</script>
```

4、 Script 标签写在页面的哪个位置？

页面 head 和 body 都可以写

5、 多段 script 的执行顺序

按引入顺序，逐段执行（思考为什么 N 多页面吧 JS 写在最后？）

a. 防止页面一次出现，等待时间较长，不符合人的阅读习惯

b. 防止 js 语言已经出现，但是变量在 js 之后没有在内存中读取，导致 js 找不到变量发生错误

## 3、 变量声明

### 命名规范

JS 的变量名可以用_,数字,字母,$,组成,且数字不能开头

声明变量用 var 变量名来声明

```
var a= 34;

var b=45;

alert(a+b);

var $ = 'jquery';

alert($);

c=56;

alert(c);
```

**注意**：变量名区分大小写 str 和 Str 不是一个变量

**注意**：不用 var,会污染全局变量

## 4、 JS 变量类型

```
<script type="text/javascript">
//js变量类型
//1 数值类型
var a =23;
var b = 3.14;

//字符串类型
var c ='hello';
var d = 'world';

//布尔型
var e =true;

//null型
var f =null;

//undefined 型
var g = undefined;
/*
——null 是代表对象不存在,在用DOM操作寻找时,没找到，返回null
——如果一个基本型没有定义，可以理解为undefined(未限定，未阐明)

*/
// 数组的写法,"[]"包围,","逗号隔开每一个值
//JS中，数组是索引类型，key是0,1,2,3,4,5.....

var i =['张','王','李','赵'];

//对象(和PHP中的关联数组类似)

var i={'name':'poly', 'age':3};

alert(h[2]); //李
alert(i.age); //3
alert(i['age']); //3




</script>
```

5、 JS 运算符

```
<script>
    //运算符
var a =3;
var b=2;
alert (a%b);//1  求余

if( a>b ) {
    alert(a + '>' +b);
}
//在js中,拼接字符串用 "+"


alert('hello' + 3 + 'world');//helloworld
//左往右加时,碰到第1个非数值型之后，就理解为字符串类型
alert( 3 + 2 + 'hello' + 5 + 'world' );//5hello5world

//逻辑运算符不同
var c = a||b ;//c 在PHP 中为true，在JS中，为a的值3
console.log(c);


a = 0;
b = 9;
c = a||b; //c在PHP中为true,在js中为b的值 9
console.log(c);

//总结,逻辑运算的值,是能确定运算的结果的单元的值


</script>
```

6、 控制结构

```
//for循环
    //for循环
    var arr =['赵','钱','孙','李',''];

    for(var i=0; i<arr.length;i++){
        console.log(arr[i]);

    }

//循环对象
    var obj = {'name':'lisi','age':29,'height':180};

    for (var k in obj) { /循环obj,把键分别赋给k,
        console.log(k);
        console.log(obj.k);  //错误
        console.log(obj[k]); //正确


    }
```

## 7、 对象操作

String 字符串对象

length 属性:长度

concat(String)连接两个或多个字符串

indexOf(string)返回出现字符串的位置

substr(num1,[num2])截取字符串

toLowerCase()转换成小写

toUpperCase()转换成大写

replace(str1,str2)字符串替换

Date 日期对象

getYear()返回年份(2 位或 4 位)

getFullYear()返回年份(4 位)

getMonth()返回月份 0-11

getDate() 返回日期 1-31

getDay() 返回星期数 0-6

getHours() 返回小时数 0-23

getMinutes() 返回分钟数 0-59

getSeconds() 返回秒钟数 0-59

getmilliseconds() 返回毫秒数 0-999

内置对象：

日起对象类：Date（）

var today = new Date();

Date() 函数是所有日期时间对象的一个“抽象模板”，而 today 是通过 Date（）造出来的一个对象。

还有一个对象，是 js 已经创建完毕的，不必再 new 来得到。

Math 对象

Math.random(); 返回 0~1 之间的随机数

Math.ceil(); 对数进行上舍入 //math.ceil(0.6) 返回 1 math.ceil(-5.1) 返回-5

Math.foor(); 对数进行下舍入

Math.round; 把数四舍五入为最接近的整数

sqrt() 开平方根

pow(a,b) 返回数值 a 的,数值 b 的次方

min(a,b) 返回最小值

max(a,b) 返回最大值

数组对象

concat() 返回一个由两个数组合并组成的新数组

join() 返回一个由数组中的所有元素并返回新长度

pop() 删除数组中的最后一个元素并返回该值

push() 像数组中添加新的元素,返回数组的新长度

shift() 删除数组中的第一个元素并返回该值

unshift() 返回一个数组,在数组头部插入了指定的元素

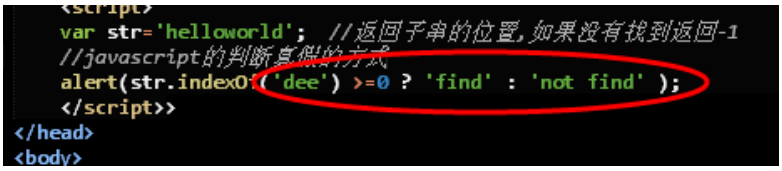
sort()　　返回一个元素被排序了的 array 对象

reverse()　返回一个元素反序的 array 对象

slice()　　返回数组的一个片段

splice()　　从数组中删除元素

## 8、 JS 内置对象




indexOf 是找字符串返回位置

## 9、 浏览器 window 对象

**注：window 对象是浏览器宿主对象,**和 JS 语言无关　　**象**

**window 对象的方法**

window.alert(message)　　类似 js 的 alert

window.conflrm(message)

window.prompt(message[,defaultstr])

window.close()　关闭窗口

window.print()　　打印

**window.setIntval (表达式,毫秒)**

**window.clearIntval(定时器对象)**

**window.setTimeout(表达式,毫秒)**

**window.clearTimeout(定时器对象)**

window 对象的子对象:

**navigator** 浏览器信息对象

appCodeName 内部代码

appName　浏览器名字

appVersion　浏览器版本

platform　　操作系统类型

userAgent 用户代理信息 头信息

cookieEnable 是否支持 cookie

判断浏览器是否支持 cookie

http://www.zixue.it/thread-12911-1-1.html

**location** 地址栏对象

host 主机

port 端口

pathname 路径

protocol 协议

search 查询字符串

assign(url)页面跳转

**history** 历史记录

length: 历史记录的数目

back()

forword()

go()

screen 屏幕对象

height 高度

width 宽度

availHeight 可用高度

availWidth 可用宽度

colorDepth 颜色

**document HTML 文档对象**

即 HTML 代码形成的对象,操作此对象,可动态的改变页面的内容

是我们做 JS 的主战场

## 10、 作用域

**var 的本质:**

var a=3;//声明 a 变量并赋值

b=3;//只是一个单纯的"赋值"

JS 作用域的特点:

首先在函数内部查找变量,找不到则到外层函数查找,逐步找到最外层

即 window 对象,并操作 window 对象的属性.

```
console.log(window.a,window.b);
function t(){
    var a= 'local';
    b = 'global';
}
console.log(window.a,window.b);
//a 的值不变,b 的值为 global, 由于函数内部没有定义 b
//找到外部变量 b 并将 gloal 赋值于 b
```

**思考 :** 为什么可以直接写 alert(),setIntervall()? (setintervall())

## 11、 第一个简单 JS 效果

```
<style>
#test1{
    width:300px;
    height:300px;
```

```
        background:blue;
}
#test2 {
            width:300px;
            height:300px;
            background:red;
        }
        </style>
        </head>
        <body>
            <div id=”test1” onclick=”ch();”></div>
        </body>
        <script>
            function ch(){
            //1 要找到这个 div 对象
            var test1 =document.getElementById(‘test1’);
            //2 要修改它的属性
            test1.id =’test2’;
        }
        </script>
        </html>
```

## 12、 读取和改变对象属性

**小练习：灯泡开关练习**

```
<script>
    /*function push() {
// window 可以省略, 由于js特性,找 document 对象找不到 跳出script 在windows下找 找到对象document
        var test1 =window.document.getElementById('test1');
//此时的做法是只要点击就关灯   本身关灯也无法在开灯
        test1.src='/image/off.jpg';
    } */
    function push2(){

        var test1=document.getElementById('test1');

        if (test1.src.indexOf('on') >=0) {
            test1.src="/image/off.jpg";
        } else{
            test1.src="/image/on.jpg";

        }
    }

</script>

</head>
<body>
<img  id="test1"  src="/image/on.jpg"  alt="灯开" onclick="push2();"/>
</body>
</html>
```

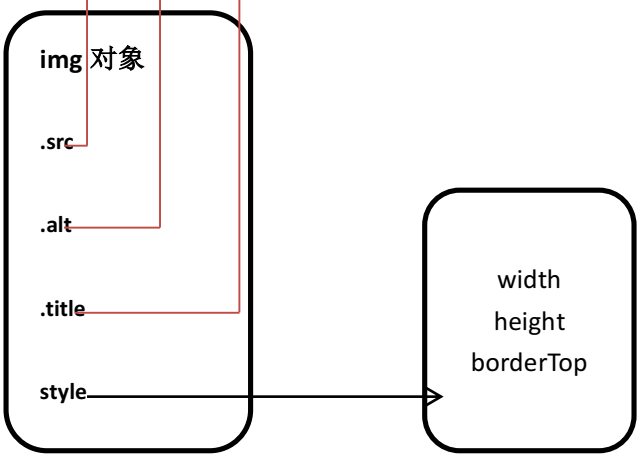
## 13、 找对象

```html
<body>

        <h1> 关键是找对象</h1>
        <div id="test1">
            <p>p1</p>
            <p>p2</p>
            <p>p3</p>
        </div>
        <div class="test2">2333333</div>
        <input type="text" name="username" id="" value="poly" />
</body>

<script >
//用id查询对象唯一返回值是对象
//用标签查询,对象不唯一返回值是集合
//用name查询,对象不唯一返回是集合
```

```
//用id来查询,返回值是"对象"
    console.log(document.getElementById('test1'));

    // 用标签来查询,返回值是? getElementsByTagName 标签名
    //返回的是对象的集合,即使只找到一个对象,也包装成对象集合返回
    ps = document.getElementsByTagName('p');
    ps[1].style.background='green'; //可以看成数组


    //对于表单元素,可以用name来查询,返回值是对象集合
    //alert(document.getElementsByName('username')[0].value);
    //document.getElementsByName('username')[0].value ='lili';

    //按照类名查找,返回集合
    document.getElementsByClassName('test2')[0].style.background ='gray';

    //孩子节点
    //alert(document.getElementById('test1').childNodes.length);
    alert(document.getElementById('test1').children.length); //children非标准属性,但兼容性很好,他不包含空白节点

    document.getElementsByTagName('p')[2].parentNode.style.border='1px solid green';//寻找父节点
```

## 14、 操作对象的属性

```
<img src="" alt="" title="" style="width:200px;height:300px;"/>
```




普通属性可以通过

对象.标签属性 来访问

CSS 属性通过

对象.style.属性

**例:**

**obj.alt**

**obj.title**

**obj.style.width**

**注意:**

标签属性与 DOM 对象属性的对应关系:

绝大部分两者是相同的,如:

imgobj.src 属性对应<img src=””>中的 src 属性

但也有例外,如:

中,操作 class 属性用 divobj.className

css 属性与 DOM 对象属性的对应关系:

两者通过 obj.style.css 属性名对应

如:

obj.style.width

obj.sytle.background

如果 css 属性带有横线,如 border-top-style,

则把横线去掉后的字母大写

如:

obj.style.borderTopStyle

obj.style.marginLeft

小练习:点击 div,使其宽,高加 10px,底边增宽 1px

```
<style>
    .test1{


        background: #66ccff;
        }

    .test2{

        background: gray;
    }
</style>
<script >
    function t(){
        var div =document.getElementsByTagName('div')[0];
        if (div.className.indexOf('test1') >=0 ) {
            div.className ='test2';
        } else{
            div.className ='test1';

        }
//300像素是字符串 parseInt 提取数字  borderBottomWidth 底边宽度
        div.style.width =parseInt(div.style.width)+5+'px';
        div.style.height =parseInt(div.style.height)+5+'px';
        div.style.borderBottomWidth =parseInt(div.style.borderBottomWidth)+1+'px';

    }
</script>
</head>
<body>
    <div class='test1' onclick="t();"
     style="width:300px; height:300px;
     border-bottom:1px solid black;"
      >
    点击div<br/>
    使其背景色红绿交替<br/>
    宽高增加5px<br/>
    下边框增粗1px<br/>
    </div>
```

## 15、 获取对象在内存中计算后的样式

上节中，obj.style 只能取得”内敛 style”的值

对于<style></style>中的 css 属性值,则无能为力

我们可以用 obj.currentStyle (ie 内核) 和 window.getComputedStyle(一般浏览器内核)来获取.

注意:只有 IE 和 Opera 支持使用 currentStyle 获取 HTMLElement 的计算后的样式,其他浏览器中不支持

标准浏览器中使用 getComputedStyle,IE9 以上也支持 getComputedStyle.

window.getComputedStyle(obj,伪元素)

参数说明

第一个参数为要获取计算后的样式的目标元素

第二个参数为期望的伪元素,如:’after’,’frist-letter’等,一般为 null

考虑兼容性,封装函数

```
function getStyle: function(el,attr){
    return el.currentStyle?el.currentStyle[attr]:getComputedStyle(el,null)[attr];
}
```

注意：这 2 个方法，获取的对象是只读的，要改样式，还得靠 obj.style

## 16、 删除对象

**步骤:**

1、 找到对象

2、 找到他的父对象 parenetObj

parentNode 传回目前节点的父节点。只能应用在有父节点的节点中。

3、 parentObj.removeChild(删除子对象)

**练习**：删除最后一个 li

<ul>

<li>春</li>

<li>夏</li>

<li>秋</li>

<li>冬</li>

</ul>

```
    //由于函数是在点击后才执行，也面已经加载完毕可以找到
    function del(){
        var lis = document.getElementsByTagName('li');
        var lastli= lis[lis.length-1] ;
        lastli.parentNode.removeChild(lastli);
    }
    </script>
</head>
<body>

    <input type="button" value="删除最后一个" onclick="del();">
    <ul>

        <li>春</li>
        <li>夏</li>
        <li>秋</li>
        <li>冬</li>
    </ul>
</body>
```

## 17、 创建对象

1、 创建对象

2、 找到父对象 parentObj

3、 parentObj.addChild(对象);

练习：添加 1 个 li

<ul>

<li>春</li>

<li>夏</li>

<li>秋</li>

<li>冬</li>

```
    <script>
        function  add(){

            var li= document.createElement('li');
            var tex = document.createTextNode('冬');
            li.appendChild(tex);
            document.getElementsByTagName('ul')[0].appendChild(li);

        }
    </script>
```

添加属性 函数 setAttribute 用法 obj.setAttribute(“href”,”#”);

练习 2

```
</head>
<body>
    <input type="button" value="增加" onclick=" add() ;">
    <ul>
        <li>春</li>
        <li>夏</li>
        <li>秋</li>

    </ul>

    <div>
        <p>
            <ul>
                <li><a href=""></a></li>
            </ul>
        </p>

        <p>
            <ul>
                <li><a href=""></a></li>
            </ul>
        </p>

        <p>
            <ul>
                <li><a href=""></a></li>
            </ul>
        </p>
    </div>

</body>
</html>
```

在最后一个 a 标签内添加内容和连接 点击一次添加一个

```
function add(){
    var txt= document.createTextNode('ddd');    //创建文本
    var a  = document.createElement('a');       //创建a标签
    a.appendChild(txt);                         //文本父对象为a标签
    var li =document.getElementsByTagName('li');    // 查找a标签的父对象 li

    //明确 li   的位置,将新的a对象传给并为a对象的属性赋值
    var lilast =li[li.length -1].appendChild(a).setAttribute("href","http://www.baidu.com");
    var ul =document.getElementsByTagName('ul'); //  查找li标签的父对象ul
    var ullast = ul[ul.length -1].lilast;       //明确ul位置将子对象lilast传给ul
    var p =document.getElementsByTagName('p')[document.getElementsByTagName('p').length - 1].ullast;//明确ul的父对象p并传参
    document.getElementsByTagName('div')[0].p;  //创建这个对象给windows

}
```

效果

---

增加

- 春
- 夏
- 秋
-
-
- ddddddddddddddddddd

## 18、 暴力操作节点

**innerHTML:**代表节点内的内容，能读能写

不是一个 w3c 规定的标准对象属性，但是-------各浏览器支持的很好

```
<script>
    function add3 () {
        var ul =document.getElementsByTagName('ul')[0];
        ul.innerHTML ='<li>春</li> <li>夏</li><li>秋</li>';
        }

    function add1() {
        var ul =document.getElementsByTagName('ul')[0];
        // alert(ul.innerHTML)     测试 innerHTML是否具有读取功能
        ul.innerHTML += '<li>冬</li>'

    }
</script>
```

## 19、 联动菜单

所用知识点：事件 +DOM 操作

三级菜单不是正规写法,只是尝试不建议用

```
<script>
    var area1=[
        ["福州","厦门","泉州"],
        ["郑州","洛阳"]
    ];
    var area2=[
        ["鼓楼区","台江区"],
        ["湖里区","思明区"],
        ["鲤城区","丰泽区"]
    ];
    var area3=[
        ["中原区","二七区","惠济区"],
        ["涧西区","洛龙区","西工区"]
    ];

    function lg(){

        var opt1='';
        var sel= document.getElementById('prov');
        if((sel.value-1) <0){
            document.getElementById('city').innerHTML = opt1;
            return;
        }

        for(var i=0,sh=area1[sel.value-1].length; i<sh;i++){
            opt1=opt1+'<option value="'+i+'">'+area1[sel.value-1][i]+'</option>';
        }
        document.getElementById('city').innerHTML = opt1;
    }
```

```
function lg2(){
    var opt2='';
    var sel = document.getElementById('prov');
    var sel2= document.getElementById('city');

    if (sel.value ==0){
        return;
    }else if(sel.value ==1){
        for(var i=0,sh2=area2[sel2.value].length;i<sh2;i++){
            opt2=opt2+'<option value="'+i+'">'+area2[sel2.value][i]+'</option>';
            }
        }else if(sel.value ==2){
            for(var i=0,sh2=area3[sel2.value].length;i<sh2;i++){
            opt2=opt2+'<option value="'+i+'">'+area3[sel2.value][i]+'</option>';
            }

        }



    document.getElementById('areas').innerHTML = opt2;

}
```

```
</script>
<body>
    <select name="" id="prov" onchange="lg()">
        <option value="0">请选择</option>
        <option value="1">福建</option>
        <option value="2">河南</option>
    </select>
    <select id="city" onchange="lg2()"></select>
    <select id="areas"></select>
</body>
</html>
```

## 20、 定时器

**windows.setTimeout**(‘语句’,’毫秒’);指定毫秒后执行一次语句

注：定时器不属于 js 知识，他是 windows 对象提供的功能

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无标题</title>
    <script>
    function bang(){
        document.getElementsByTagName('img')[0].src ='/image/warn.jpg'
    }
    window.setTimeout('bang()',3000);
    </script>
</head>
<body>
    <h1>定时器</h1><br />
    <img src="/image/zadan.jpg" alt=""/>
</body>
```

定时器

定时器


**windows.setInterval** (‘语句’,’毫秒’); //每经过执行语句

小练习:倒计时炸弹

思路:每 1 秒,修改剩余时间,剩余时间为 0 时,修改爆炸后的图片

```
    <script>
    function bang(){
        var inp =document.getElementsByName('time')[0];
        var times =parseInt(inp.value)-1;
        inp.value=times;
        if(times ==0){
            document.getElementsByTagName('img')[0].src ='/image/warn.jpg';
        }

    }
    setInterval("bang()",1000);

    </script>
</head>
<body>
    <h1>定时器</h1><br />
    <input type="button" name="time" value="5" /> <br />
    <img src="/image/zadan.jpg" alt=""/>
```

出现负值

**清除定时器:**

- clearInterval()
- clearTimeout()

```
    <script>
    function bang(){
        var inp =document.getElementsByName('time')[0];
        var times =parseInt(inp.value)-1;
        inp.value=times;
        if(times ==0){
            document.getElementsByTagName('img')[0].src ='/image/warn.jpg';
            clearInterval(clock);
        }

    }
    var clock =setInterval("bang()",1000);
    if (clock ==0){

    }

    </script>
```

**setTimeout 实现倒计时**

setTimeout 可以反复调用自身，达到类似 setInterval 的效果

例：setTimeout 版的定时炸弹

```
    <script>
        var clcok =null;

        function bang(){
            var inp =document.getElementsByName('time')[0];
            var times =parseInt(inp.value)-1;
            inp.value=times;
            if(times ==0){
                document.getElementsByTagName('img')[0].src ='/image/warn.jpg';
                clearTimeout(clock);
            }else{
                setTimeout('bang()',1000);
            }

        }

        clock =setTimeout('bang()',1000);

    </script>
```

## 21、 DOM 中的事件

- onclick 元素点击时
- onfocus 元素获得焦点时
- onblur 元素失去焦点时

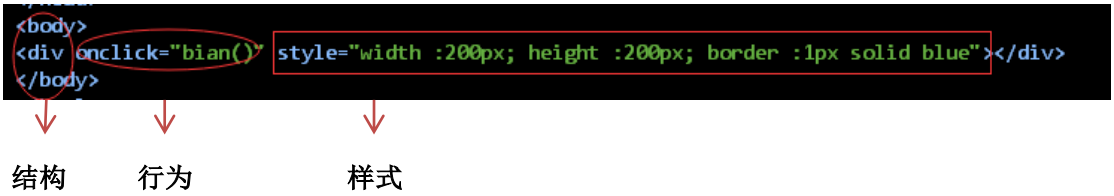
```
    <script >
        function t1(){
        document.getElementsByName('username')[0].style.border='3px solid #66CCFF';
        };
        function t2(){
            document.getElementsByName('username')[0].style.border='';
        }
    </script>
</head>
<body>
    <p>
            用户名<input type="text" name ="username" onfocus="t1();"onblur="t2();" />
    </p>
    <p>
            Email:<input type="text" name ="email" />
    </p>
```

- onmouseover 鼠标经过时
- onsubmit 表单提交时
- onload 页面加载完毕时 运行

注意：onsubmit=”return t2()”;t2()函数才能拦截提交效果

```
    <style>
    div{
        width: 200px;
        height: 200px;
        background: gray;
    }
</style>
<script >
    function t1(){
        alert('谁？');
    }
    function t2(){
        alert('请填写完整');
        return false;
    }
</script>
</head>

<body>
    <h1>鼠标经过事件</h1>
    <div onmouseover="t1();"></div>
    <form action="/js/24.html" onsubmit="return t2();">
        <p><input type="text" name="username" id=""/></p>
        <p><input type="text" name="email" id=""/></p>
        <p><input type="submit" value="提交" /></p>

    </form>
</body>
```

## 22、 结构样式行为分离




结构行为样式分离

```
<style>
    div{
        width :300px;
        height :100px;
        border :1px solid blue;
    }
</style>
</head>

<body>
    <div onclick="bian();" src="xx" >
        <p>
            <ul>
                <li>好</li>
            </ul>
        </p>

    </div>
</body>
<script >
/*  function bian(){
        document.getElementsByTagName('div')[0].style.border="1px solid red";
    }
    document.getElementsByTagName('div')[0].onclick=bian;*/
    document.getElementsByTagName('div')[0].onclick =function (){
        document.getElementsByTagName('div')[0].style.border="1px solid red";
    }

</script>
</html>
```

通过属性绑定方法

this 指的就是 div 对象

```
</body>
<script >
/*      document.getElementsByTagName('div')[0].onclick =function (){
        document.getElementsByTagName('div')[0].style.border="1px
        solid red";}*/
        document.getElementsByTagName('div')[0].onclick=function(){
            this.style.border="1px solid red;"
        }
```

## 事件对象 Event 对象

**事件对象:**事件发生的瞬间, 发生位置, 时间, 鼠标按键, 触发的节点等信息, 被打包成 1 个对象.

此对象, 系统自动传递给时间函数的第 1 个参数.

**小练习:**

```
<style>
    img {
            display:  block;
            width:   200px;
            height:  125px;
            position:  relative;
            left :0px;
            top:0px;
    }
</style>
<body>
    <pre>
        1:事件对象包含事件相关的信息，如鼠标，时间，触发的DOM对象等
        2：事件对象被系统传递给事件函数的第1个参数
        3：事件对象的属性在IE/W3C略有不同
        4：一个重要的事件属性：target，scrElement(IE下),代表事件发生的所在DOM对象
    </pre>
    <img src="/image/meinv.jpg" alt="" />
    <!--display  为属性  block为块-->
</body>
    <script>
    var img = document.getElementsByTagName('img')[0];
    img.onmouseover= function(ev){
        this.style.left = parseInt(ev.pageX) +10 +'px';
        //pageX()    偏移量
        this.style.top  = parseInt(ev.pageY) + 10 +'px';
        //pageY()   偏移量
    }
    </script>
```

**事件委托**

当有比较多元素需要绑定某事件时，可以把事件绑定在他们的父元素上

把事件处理委托给了他的上级

重要属性 target

```
<style>
    table {
        width:  300px;
        height: 300px;
        border: 0;
        border-collapse: collapse;
        background: #66ccff;
    }
    td {
        border:1px solid gray;

    }

</style>
</head>
<body>
    <h1>事件委托</h1>
        <table>
        </table>

</body>
```

```
<script>
    /*var tds=document.getElementsByTagName('td');
    var i=0;
    while (i <tds.length) {
        tds[i].onclick =function(){
            this.style.background ='black';
        }
        i++;
    }*/
    /****这种方法消耗资源
    我们的目标是每次点击其中一个单元格
    都可以使之变色***/

    var num =0;
    document.getElementsByTagName('table')[0].onclick = function(ev){
        //ev.target 可以找到单元格对象
        //num++%2? 'black':'white'  计数器
        ev.target.style.background = num++ %2 ?'black' :'white';
    }
</script>
```

## 23、 JS 使用正则

**声明**

```
var patt=/*\d{11}$/;
```

**使用**

```
//判断 String 是事符合正则要求
patt.test(String);
//找出字符串中的符合正则的子串
patt.exec(String);
```

^ ：匹配输入的开始位置。

\：将下一个字符标记为特殊字符或字面值。

* ：匹配前一个字符零次或几次。

+ ：匹配前一个字符一次或多次。

(pattern) 与模式匹配并记住匹配。

x|y：匹配 x 或 y。

[a-z] ：表示某个范围内的字符。与指定区间内的任何字符匹配。

\w ：与任何单词字符匹配，包括下划线。

{n,m} 最少匹配 n 次且最多匹配 m 次

$ ：匹配输入的结尾。

例子:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无标题</title>
</head>
<body>
    <pre>
        1:用户名只能是字母及数字6-11位
        2:email为正确的email
    </pre>
    <form action="">
        <p> 用户名:<input type="text" name="username" /></p>
        <p>  Email:<input type="text" name="email" /></p>
        <p> <input type="submit" value="提交" /></p>
    </form>

</body>
```

```
</body>
<script>
    document.getElementsByTagName('form')[0].onsubmit =function(){
        //正则检测
        //判断用户名  第一位开始只能大小写和数字 共6-11位
        var  patt = /^[a-zA-Z0-9]{6,11}$/;
        if(!patt.test(document.getElementsByName('username')[0].value)){
            alert('用户名由6-11是字母及数字组成');
            return false;
        }
        //判断email xx@mp.weixin.qq.com  mp 一段  后面的一段
        // 域名不包括下划线
        var patt = /^\w+@[a-zA-Z0-9\-]+(\.[a-zA-Z0-9\-]+)+$/;
        if(!patt.test(document.getElementsByName('email')[0].value)){
            alert('请输入正确的eamil');
            return false;
        }

    }
</script>
```

正则检测 patt.exec(String)

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无标题</title>
<script>
    function find(){
        var lis =document.getElementsByTagName('li');
        var i=0;
        patt=/\w+@[a-zA-Z0-9\-]+(\.[a-zA-Z0-9\-]+)+/;

        while(i<lis.length){
                if(patt.exec(lis[i].innerHTML)){
                    lis[i].style.background ='yellow';
                }
            i +=1;
        }
    }
    </script>
</head>
<body>
    <input type="button" value="标注有邮箱的人" onclick="find()">
    <ul>
        <li>张飞</li>
        <li>刘备&lt;liubei@zixue.it&gt;</li>
        <li>关羽</li>
        <li>赵云&lt;zhaoyun@qq.com&gt;</li>
    </ul>
</body>
```