

JavaScript

简介

- JavaScript 和 Java 没关系
- ECMAScript 官方叫法
- 语法和 Java 比较像
- 不简单，语法更灵活
- Java 在 JVM 中运行，JS 运行于浏览器（浏览器差异）
- HTML JS (jQuery) 静态网页技术

一、JS 基本语法

- 全面支持 **unicode** 编码（**Java** 也是）
Java 中

```
String name;  
String 姓名; //汉字或其他文字都可以表示，这就是全面支持 unicode 编码
```

- 弱数据类型
相对比于 Java 强数据类型的编程语言

```
String name  
int age  
double salary
```

JS 中：定义变量时不用指定变量数据类型（变量不用指定，数据本身是有类型的）

```
var name = "tarena";  
var age = 10;  
var salary = 10.0;
```

第一个 JS 程序

同 CSS 相似

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
    <script type="text/javascript">  
      alert("hello JS"); //alert 输出  
      document.write("页面"); //写到页面的方法  
    </script>  
  </head>
```

世界因你而不同！

```
<body>
</body>
</html>
```

语法

1. 变量

- Java 中: `String s = "hello";` 变量类型 变量名 = 数据;
- JS 中: `var name = "hello";`

1) 基本数据类型

- 数字类型
JS 中没有整数和浮点数类型之分, 统一叫 `number`
- 字符串类型
"和""效果相同, 都是字符串
- 布尔类型
第一种: `true` `false`
第二种: 非 0 0
第三种: 非 `null` `null` (容易混淆)
 容易混淆的一个例子:

```
if(object){
```


 }
 表示如果有对象为真, 无对象为假
- 特殊类型
 - ✓ **undefined**
定义一个变量如果没有赋值, 变量为 `undefined`
没有定义变量时直接使用, 变量也为 `undefined`
 - ✓ **null**
注意点: `var name = null;` **//null 表示赋值为 null, null 也是一个值, 不同于 undefined**
注意点: `javaScript 函数` **没有提供返回值, 函数的返回值为 null**
 - ✓ **NaN** **not a number**
假如在运算式中用字符串和数字运算, 则显示为 `NaN`

注意: JS 的灵活

- 允许定义多个同名变量

```
var name = "hello";
var name="kitty";
```

```
alert(name);    //会输出 kitty
```

- 定义同名变量，会输出赋值的

```
var name = "hello";  
var name;  
alert(name);    //会输出 hello
```

- JS 默认一行行为 1 条语句（不写分号也可以）

```
var name = "hello"  
alert(name);    //会输出 hello
```

- 允许不使用关键字 **var** （全局变量）

```
name="hello";    //全局变量  
alert(name);  
  
function f(){  
    var a="1001"; //局部变量：用 var 修饰的  
    id=123;       //全局变量 没有用 var 修饰的  
}
```

2) 对象数据类型 ***

function 函数

相当于 Java 中的 method

Java 中定义方法：修饰符 返回值类型 方法名（参数表）异常表{ }

```
public int add(int a , int b) throws Excpetion{  
    return i+j ;  
}
```

第一种定义方法

```
function add(a , b){  
    return i+j ;  
}
```

没返回值类型，没参数类型

提示：学没学好 JS，关键对方法体会够不够深刻

函数使用方式：

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

世界因你而不同！

```
<script type="text/javascript">
    function add(i , j){
        return i + j;
    }

    alert(add(1,2));
</script>
</head>
<body>
</body>
</html>
```

第二种定义方法

```
var fun = new Function("i" , "j" , "return i+j");
alert(fun(3,2));
```

实际编码过程中不会有人用，帮助理解：函数也是可以存到一个变量中的

辨析：

```
function add(i , j){
    return i + j;
}

var fun1 = add(1,2);    //把调用函数的结果传给一个变量
var fun2=add;           //把函数本身赋给一个变量

alert("fun2 " + fun2(1,2));
```

匿名函数 ***

```
var fun = function(i , j){
    return i+j ;
}
alert(fun(1,2));
```

注意：

```
function add(i , j){
    return i+j;
}
alert(add(1,2,3));    //输出 3 多加的参数不会管
alert(add(1));        //输出 NaN (1 + undefined 结果为 NaN)
```

注意：JS 中没有重载

函数参数可为另一函数 ***

```
function f1(){
    alert("test");
}
function f2(f){
    f();
}

f2(f1);
```

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <script>
    function add(i,j){
      return i+j;
    }
    function test(i,j,f){
      return f(i,j);
    }
  </script>
</head>
<body>
  <div class="style2" onclick="alert(test(3,4,add));">
  </div>
</body>
</html>
```

内置对象

函数内部有一个隐含对象

argument[]: 用户在调用函数时所传入的参数

argument[0]: 第 1 个参数

argument[1]: 第 1 个参数

世界因你而不同!

```
function f(i , j){
    alert(arguments[0]);
    alert(arguments[1]);
}
f(1,2);    //输出的是两个参数的值
```

应用：对函数参数的保护措施

```
function add(i , j){
    if(arguments.length != 2){
        alert("error");
        return;
    }

    if( (typeof i != "number") || (typeof j != "number")){
        alert("number error");
        return;
    }
}
//add(1,23,232);
add("a" , "b");
```

创建对象的方式

Object 对象

创建 student 对象？

通过构造函数创建对象

```
var stu = new Object();
stu.name = "suns";
stu.age  = "30";

alert(stu.name+ " " + stu.age);
```

注意：本质上 JS 对象没有方法，只有属性

但是：可以模拟的为对象增加方法，比如 **Document.write()**;

```
function showMethod(){
    alert("show student");
}

var stu = new Object();
stu.name = "suns";
stu.age  = "30";
```

```
stu.show = showMethod;
```

```
stu.show();
```

更常用的方式

匿名函数的使用

```
var stu = new Object();
stu.name = "suns";
stu.age = "30";
stu.show = function(){
    alert("show student");
};

stu.show();
```

在 JS 中写 **get**（）方法

```
var stu = new Object();
stu.name = "suns";
stu.age = "30";
stu.getName = function(){
    return this.name;
};

alert(stu.getName());
```

JSON 对象 ***

JSON 对象：目前在 JS 中常用的创建对象的方式

```
var student = {
    name:"xxx",
    age:30,
    getName:function(){
        return this.name;
    }
}

alert(student["age"]);    //输出对象属性的方法
student.name;
student.age;
student.getName();
```

好处：体现了属性和方法的结构和封装的概念

HTML 对象 ***

每一个 **Html** 标签被 **JS** 认为是一个对象

```
<input type="text" name="name" value="1234" />

var ipu
  ipu.type="text";
  ipu.name="name";
  ipu.value="1234";
```

Array 对象

1. 通过构造方法创建

```
var arr = new Array();
arr[0] = 1;
arr[1] = 2;
```

辨析:

- **Java** 中数组的特点
 1. 长度不可变
 2. 连续内存空间
 3. 一种类型的数组只能存储特定类型的数据
- **JS** 中的数组特点
 1. **JS** 中数组长度可变
 2. 不赋值的元素也会留空间, 为 **undefined**
 3. 可存任意数据

```
var arr = new Array();
arr[0] = 1;
arr[1] = 2;
arr[3] = 3;
arr[4] = "hello";

for(var i=0;i<arr.length;i++){
  alert(arr[i]);
}
```

像 **Java** 中的 **ArrayList**

2. JSON 方式 ***

```
var arr = [1, 2, 3, 4, 5, ];
alert(arr.length);    //IE 数组长度为 6, Firefox 为 5
```


3. 常用 API ***

1) push()

往数组中最后一个位置加入一个元素

```
var arr = [1,2,3];  
arr.push(4);  
alert(arr);    //输出 1, 2, 3, 4
```

2) pop()

删除数组中最后一个元素

```
var arr = [1,2,3];  
arr.pop();  
alert(arr);    //输出 1,2
```

3) shift()

删除数组中第一个元素

```
var arr = [1,2,3];  
arr.shift(0);  
alert(arr);    //输出 2, 3
```

4) unshift()

在原数组第一个位置插入一个元素

5) sort()排序

注意：默认按照 ASCII 码排序

```
var arr = new Array();  
arr = [10,3,5,4,2];  
alert(arr.sort());    //结果是 10,2,3,4,5
```

如果想对数字进行排序

```
var arr = [10,3,5,4,2];  
arr.sort(function(a,b){  
    return a-b;  
});  
  
alert(arr);
```

6) reverse() 反转

7) join("字符")

通过指定的字符，构建成一个字符串

```
var arr = [1,2,3];  
var str = arr.join("-");  
alert(arr.join("-"));    //输出 11-2-3
```

String

```
var s = new String("aaaa");  
var s1 = "aaaa";
```

常用 API ***

1. toString() == valueOf()

显示字符串

2. concat()

连接字符串

3. substring(起始位置, 终止位置)

截取字符串 [0, str.length)

```
"sunshuai".substring(1,4);    -- uns
```

4. indexOf()

位置

```
"sunshuai".indexOf("h");    -- 4
```

```
"sunshuai".indexOf("p");    -- -1    -1 表示没有这个字符
```

5. charAt()

第 n 个位置的字符

6. split()

用指定字符切割字符串

```
var str = "test , String , hello";  
var arr = str.split(",");  
alert(arr[0]);
```

7. String.fromCharCode("");

特殊方法：把一个 **ascii** 码转换为一个字符

```
alert(String.fromCharCode("97"));
```

8. length ***

注意：JS 中判断字符串长度用 **length 属性而非 **length()** 方法**

```
var str = "test , String , hello";  
alert(str.length);
```

Date

Java:

```
java.util.Date();  
java.sql.Date();  
java.sql.Timestamp();    -- 用于 JDBC 显示时 分 秒
```

JS:

更像 java.util.Date()
var date = new Date(); 当前日期

1. 创建日期

```
//创建系统日期  
var d = new Date();  
alert(d);  
//创建指定日期（2010 年 8 月 1 日 8: 15）因为有时区问题：小时又加了 8  
var d1 = new Date(Date.UTC(2010,9,1,8,15));  
alert(d1);
```

Date.UTC：将日期格式转换一下

2. 常用 API

1) getYear()

注意：为解决千年虫问题，都从 1900 年开始记年 $1900 - 2011 = 111$

```
//创建系统日期  
var d = new Date();  
alert(d.getYear());    //111
```

2) getMonth()

3) getFullYear()

得到实际的年份

```
//创建系统日期  
var d = new Date();  
alert(d.getFullYear());
```

4) getTime()

距 1970 年的毫秒数

5) getDay()

获得每周的第几天（**星期几**）

6) getDate()

获得一个月中的第几天（**几号**）

7) toLocaleString()

按当前国家显示日期格式显示

```
//创建系统日期  
var d = new Date();  
alert(d.toLocaleString());
```

Math

处理算术运算

常用 API

1) Math.ceil()

向上加 1（非四舍五入）

```
alert(Math.ceil(15.4));    //16
alert(Math.ceil(15.6));    //16
```

2) Math.floor()

截掉小数

```
alert(Math.floor(16.5));    //16
```

类似于 **Oracle** 的 **trunc()**函数

2. 运算符

1) 算术运算

- 加法运算时会做字符串拼接；减法运算时会做算术运算

```
var a = 1;
var b = 4;
var c = "3";
alert(a+c);    //输出 13
alert(b-c);    //输出 1
```

- **parseInt()**方法 **parseFloat()**(字符串)

```
var a = 1;
var c = "3";
alert(a+parseInt(c));    //输出 4
```

2) 比较运算 ***

> < != == >= <=

- JS 中比较两个字符串用 “==”，“===” 仅比较内容

```
var nam="aaa";
var nam2="aaa";
alert(nam==nam2);    //true

var s3=12;
var s4="12";
alert(s3==s4);    //true
```

- “===” 对比的是数据值和类型；“==” 对比数据类型

```
var s3=12;
var s4="12";
alert(s3===s4);    //false
```

3) 逻辑运算

&& || !

4) 赋值运算

=

5) 三目运算符

条件表达式?true:false

6) typeof 运算符

```
var i = "abc";
var j = null;
var x = undefined;
alert(typeof i);
```

```
alert(typeof j);  
alert(typeof x);
```

3. 表达式

```
var a = true;  
var b = false;  
a&&b
```

4. 流程控制

顺序流程控制

条件流程控制 if(){}else if(){} switch(){case :XXX}

循环流程控制 for while do-while

JS 中的 for 循环

```
for(var i=0;i<length;i++){}
```

foreach 循环 ***

作用：

遍历数组

遍历对象属性

1) 遍历数组

```
var arr = [1,2,3,4];  
for(idx in arr){  
    alert(idx + "：" + arr[idx]);  
}
```

idx：表示迭代数组当前的下标

2) 遍历对象属性 ***

```
var student = {  
    name:"xxx",  
    age:20  
};  
for(prop in student){  
    alert(prop + ":" + student[prop]);  
}
```

prop: 表示属性的 **key**

注意：能用 **for** 循环遍历的不是数组就是集合，那么 **JS** 中的对象本质上相当于 **Java** 中的 **Map**

二、事件句柄 ***

事件监听属性

1. 一般性事件监听属性

可加入给任意的 **Html** 标签

- **onclick** 鼠标单击事件
- **ondblclick** 鼠标双击事件
- **onmouseover** 鼠标悬浮事件
- **onmouseout** 鼠标移出事件
- **onmousemove** 鼠标移动事件
- **onmousedown** 鼠标按下事件
- **onmouseup** 鼠标松开事件

举例

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

世界因你而不同！


```

<style type="text/css">
    .style{
        background-color:black;
        color:white;
    }
</style>
</head>
<body>
    <h2 class="style" onclick="alert('onclick');">onclick</h2>
    <h2 class="style" ondblclick="alert('doubleclick');">ondblclick</h2>
    <h2 class="style" onmouseover="alert('mouseover');">onmouseover</h2>
    <h2 class="style" onmouseout="alert('mouseout');">onmouseout</h2>
    <h2 class="style" onmousemove="alert('mousemove');">onmousemove</h2>
    <h2 class="style" onmousedown="alert('mousedown');">onmousedown</h2>
    <h2 class="style" onmouseup="alert('mouseup');">onmouseup</h2>
</body>
</html>

```

2. 页面相关的事件监听属性

只能写在**<body>**标签中

- **onload** 加载页面时
- **onscroll** 滚动事件监听（当滑动滚动按钮时触发）
- **onstop** 点击 **stop** 按钮（红叉）时触发
- **onresize** 当调整浏览器窗口大小事触发
- **onmove** 当移动窗口时触发 **注意：Firefox 不支持，IE 也不支持**

案例：**onload**

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript">
            function show(){
                alert("onload ok");
            }
        </script>
    </head>
    <body onload="show();">
        <br>
        <h1 class="style" >点这里</h1>
    </body>
</html>

```

世界因你而不同！

案例： onscroll

[illegible]

3. 表单事件监听属性

- **onblur** 丢失焦点时触发
- **onfocus** 获得焦点时触发
- **onchange** 当值发生改变时触发

世界因你而不同！

- **onsubmit** 提交表单时触发

案例

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">

      </script>
    </head>
    <body>
      <form onsubmit="alert('onsubmit test');">
        <input onblur="alert('blur test')" type="text"/>
        <input onfocus="alert('focus test')" type="text"/>
        <input onchange="alert('change test')" type="text"/>
        <input type="submit" />
      </form>
    </body>
  </html>
```

4. 事件三要素

三要素：

监听器 事件监听属性（**function()**）

事件 **click**

事件源 发生事件的标签

event 事件对象

- **event.type** 事件类型
- **event.clientX** 事件产生的位置的横坐标 火狐:不 OK; IE:ok
- **event.clientY** 事件产生的位置的纵坐标 火狐: 不 OK; IE: ok
- **event.target** 发生事件的源头 火狐: ok; IE: 不 ok

案例：获得丢失焦点的事件

```
<html>
  <head>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript">
    function listener(event){
        //alert(event);
        //alert(event.type);
        alert(event.clientX + " " + event.clientY);
    }
</script>
</head>
<body>
    <!--获得丢失焦点的事件 -->
    <input type="text" onblur="listener(event);" /> //这样获得事件
</body>
</html>

```

event: 即事件对象

三、DOM ***

document object module 文档对象模型

DOM: 一组 API 方法

功能: 对标签属性的操作, 比如可以实现一些特效, 比如点击图片变大

- 修改标签属性
- 增加删除 html 文本中的标签

1. 修改标签属性

1) 获取 HTML 标签对象 ***

- `getElementsByTagName("name")` array
- `document.getElementById("")` XMLHttpRequest
- `hasChildNodes()` boolean

document.getElementsByTagName()

通过 `document.getElementsByTagName()` 方法获得一个标签对象的数组

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function test(){
        var ps = document.getElementsByTagName("p");
        for(var i=0;i<ps.length;i++){
          alert(i + " " + ps[i]);
        }
      }
    </script>
  </head>
  <body>
    <p>xxXXXXxxxSxx</p>
    <p>xxXXXXSxx</p>
    <input type="submit" onclick="test();" />
  </body>
</html>
```

document.getElementById()

案例 1:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function test(){
        var ps = document.getElementById("p1");
        alert(ps);
      }
    </script>
  </head>
  <body>
    <p id="p1">xxXXXXxxxSxx</p>
    <p>xxXXXXSxx</p>
    <input type="submit" onclick="test();" />
  </body>
</html>
```

案例 2:

通过点击两个按钮，让一张图片变大或变小

```
<html>
  <head>
    <script type="text/javascript">
      function larger(){
        var img = document.getElementById("im");
        img.width=img.width+50;
      }
      function smaller(){
        var img = document.getElementById("im");
        img.width = img.width-50;
      }
    </script>
  </head>
  <body>
    </img>
    <br/>
    <input type="button" value="larger" onclick="larger();"/>
    <input type="button" value="smaller" onclick="smaller();"/>

  </body>
</html>
```

案例 3：获得标签对象的第一种方式 ***

输入框长度如果大于 6，则弹出对话框

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function fun(){
        var nameInput = document.getElementById("name");
        if(nameInput.value.length>=6){
          alert("请输入长度小于 6 的数字");
        }
      }
    </script>
  </head>
```

```

<body>
  <form>
    name:<input type="text" id="name" name="name" onblur="fun();" /><br/>
    pswd:<input type="password" name="passwd" /><br/>

    <input type="submit" value="submit" />
  </form>
</body>
</html>

```

案例 4：获得标签对象的第二种方式 ***

常用的一种方式 **this** 方式

不使用 `getElementById`，另一种方式获取标签对象

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function fun(tag){
        if(tag.value.length>=6){
          alert("请输入长度小于 6 的数字");
        }
      }
    </script>
  </head>
  <body>
    <form>
      name:<input type="text" id="name" name="name" onblur="fun(this);" /><br/>
      pswd:<input type="password" name="passwd" /><br/>

      <input type="submit" value="submit" />
    </form>
  </body>
</html>

```

案例 5：获得标签对象的第三种方式 ***

通过事件模式，`event.target` 获得当前标签对象

```

<html>

```

```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <script type="text/javascript">
    function fun(event){
      if(event.target.value.length>=6){
        alert("请输入长度小于 6 的数字");
      }
    }
  </script>
</head>
<body>
  <form>
    name:<input type="text" id="name" name="name" onblur="fun(event);"/><br/>
    pswd:<input type="password" name="passwd" /><br/>

    <input type="submit" value="submit" />
  </form>
</body>
</html>

```

案例 6：实现注册规则校验（初步）



虽然校验，但是点击 submit 按钮，仍然会提交给 action 属性指定的网站

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function fun(event){
        var span = event.target.nextSibling.nextSibling;    //回车也算一个兄弟节点
        var txt = span.firstChild;

```

世界因你而不同！


```

        if(event.target.value.length>=6){
            txt.nodeValue = "请输入长度小于 6 的字符";
        } else{
            txt.nodeValue = "*";
        }
    }
}
</script>
<style type="text/css">
    .style{
        color:red;
        margin-left:50px;
    }
</style>
</head>
<body>
    <form>
        name:<input type="text" id="name" name="name" onblur="fun(event);"/>
        <span class="style">*</span><br/>
        pswd:<input type="password" name="passwd" />
        <span class="style">*</span><br/>
        <input type="submit" value="submit" />
    </form>
</body>
</html>

```

注意: **nextSibling** 表示兄弟节点

案例 7: 实现注册规则校验 ***

当用户输入的数据格式不符时, 则不向服务器提交数据 (注意: **Firefox** 可以, **IE** 不可以)

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript">
            function fun(event){
                var span = event.target.nextSibling.nextSibling;
                var txt = span.firstChild;
                if(event.target.value.length>=6){
                    txt.nodeValue = "请输入长度小于 6 的字符";
                } else{
                    txt.nodeValue = "*";
                }
            }
        </script>
    </head>
    <body>
        <form>
            name:<input type="text" id="name" name="name" onblur="fun(event);"/>
            <span class="style">*</span><br/>
            pswd:<input type="password" name="passwd" />
            <span class="style">*</span><br/>
            <input type="submit" value="submit" />
        </form>
    </body>
</html>

```

```

        function sub(){
            var nameInput = document.getElementById("name");
            if(nameInput.value.length>=6){
                return false;
            }
            return true;
        }
    </script>
    <style type="text/css">
        .style{
            color:red;
            margin-left:50px;
        }
    </style>
</head>
<body>
    <form action="header.html" onsubmit="return sub();">
        name:<input type="text" id="name" name="name" onblur="fun(event);"/>
        <span class="style">*</span><br/>
        pswd:<input type="password" name="passwd" />
        <span class="style">*</span><br/>
        <input type="submit" value="submit" />
    </form>
</body>
</html>

```

2) DOM 属性 ***

- | | | |
|----------------------------------|-------------------|--------------|
| ● element.parentNode | XMLElement | 父节点 |
| ● element.childNodes | Array | 子节点 |
| ● element.firstChild | XMLElement | 第一个子元素 |
| ● element.lastChild | XMLElement | 最后一个子元素 |
| ● element.nextSibling | XMLElement | 下一个同一级别的兄弟节点 |
| ● element.previousSibling | XMLElement | 上一个同一级别的兄弟节点 |

案例 1:

```

<html>
    <head>
        <script type="text/javascript">

```

```

        function fun(){
            var ul = document.getElementById("u1");
            /*alert(ul.parentNode);
            var lis = ul.childNodes;
            alert(lis[1].firstChild.nodeValue);
            */
            var lis = ul.childNodes;
            alert(lis[3].firstChild.nodeValue);
        }
    </script>
</head>
<body>
    <ul id="u1">
        <li>aaaa</li>
        <li>bbbb</li>

    </ul>
    <input type="button" value="smaller" onclick="fun();"/>
</body>
</html>

```

注意：回车、空格也算一个节点（子标签）

（1）有三个子节点

```

<ul>
    <li><input /></li>
</ul>

```

（2）有一个子节点

```

<ul><li></li></ul>

```

案例：firstChild && childNode[0]

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript">
            function fun(){
                var hh = document.getElementById("hh");
                alert(hh.firstChild.nodeValue);
                alert(hh.childNodes[0].nodeValue);
            }
        </script>
    </head>
    <body id="body">
        <h1 id="hh">wefwef</h1>
    </body>
</html>

```

世界因你而不同！

```
<input type="button" value="smaller" onclick="fun();"/>
</body>
</html>
```

案例：value 和 nodeValue 和 text

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function selectCity(){
        var sel = document.getElementById("contry");
        alert(sel.options[0].firstChild.nodeValue);
        alert(sel.options[0].value);
        alert(sel.options[0].text);
      }
    </script>
  </head>
  <body>
    国家：
    <select id="contry" name="contry" onchange="selectCity();">
      <option value="xz">请选择</option>
      <option value="china">中国</option>
      <option value="usa">美国</option>
      <option value="mg">蒙古</option>
    </select>
    <select name="contry">
      <option>请选择</option>
    </select>
  </body>
</html>
```

2. 增加/删除标签

1) 增加/删除 HTML 标签的方法

- 增加：将<div>放入 id 为 b 的<body>中
var div = document.createElement("div");
var b = document.getElementById("b");
b.appendChild(div);
- 删除：将<body>中的<div>删除
var div = document.getElementById("div1");
b.removeChild(div);
- 创建一个文本
var text = document.createTextNode("文本内容");
- **insertBefore(newNode , targetNode);**
- **replaceChild(newNode , oldNode);**

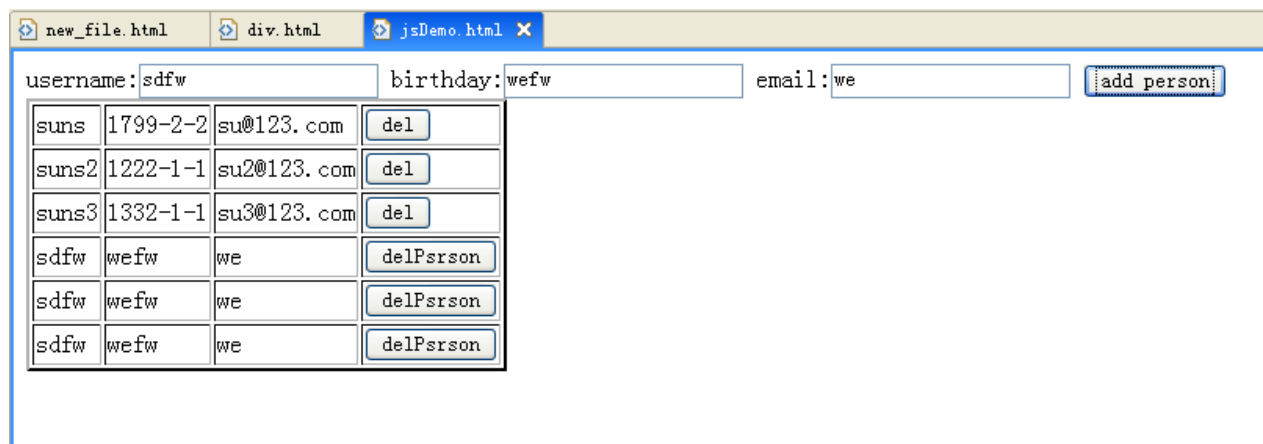
案例 1：增加一个<div>，其中有文本

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function addDiv(){
        var div = document.createElement("div"); //注意有引号
        var b = document.getElementById("b");
        //增加文本
        var txt = document.createTextNode("this is a super boy");
        div.appendChild(txt);
        b.appendChild(div);
      }
    </script>
    <style type="text/css">
      div{
        border:solid 2px red;
        height:30px;
      }
    </style>
  </head>
  <body id="b">
    <input type="button" value="add div" onclick="addDiv();" />
  </body>
</html>
```

案例 2: *****

版本 1:

点击【add person】按钮，在表格中增加数据；点击【del】按钮，在表格中删除数据



suns	1799-2-2	su@123.com	del
suns2	1222-1-1	su2@123.com	del
suns3	1332-1-1	su3@123.com	del
sdfw	wefw	we	delPerson
sdfw	wefw	we	delPerson
sdfw	wefw	we	delPerson

知识点:

- 添加嵌套标签的步骤
 - `document.createElement("td");`
 - `document.createTextNode(name);`
 - `appendChild()`
- 增加一个删除按钮(难点)
- `tag.parentNode.parentNode;`
- `tb.removeChild(tr);`
- `onclick="delPerson(this);"`

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function addPerson(){
        //1 获得三个文本框的值
        var name = document.getElementById("name").value;
        var birth = document.getElementById("birth").value;
        var email = document.getElementById("email").value;

        //2 创建<td>对象
        var nameTd = document.createElement("td");
        var nameTxt = document.createTextNode(name);
        nameTd.appendChild(nameTxt);

        var birthTd = document.createElement("td");
        var birthTxt = document.createTextNode(birth);
        birthTd.appendChild(birthTxt);
```

```

var emailTd = document.createElement("td");
var emailTxt = document.createTextNode(email);
emailTd.appendChild(emailTxt);
    //增加一个删除按钮(难点)
var delTd = document.createElement("td");
var delInput = document.createElement("input");
delInput.value="delPrsrson";
delInput.type="button";
    /*两个注意点:
        * 1 加入了 onclick 属性
        * 2 赋值时不能这样写: =delPerson(this), 这表示传的是调用函数的值
    */
    delInput.onclick=function(){
        delPerson(this); //表示调用 delPerson()方法
    };
delTd.appendChild(delInput);

//3 创建 tr
var tr = document.createElement("tr");
tr.appendChild(nameTd);
tr.appendChild(birthTd);
tr.appendChild(emailTd);
tr.appendChild(delTd);

//4 添加 tr
var tb = document.getElementById("tb");
tb.appendChild(tr);
}

function delPerson(tag){    //注意不要写 delete, 是 JS 中的关键字
    var tr = tag.parentNode.parentNode;
    var tb = document.getElementById("tb");
    tb.removeChild(tr);
}
</script>
</head>
<body id="b">
    username:<input id="name" type="text" />
    birthday:<input id="birth" type="text"/>
    email:<input id="email" type="text"/>
    <input type="button" value="add person" onclick="addPerson();"/>

    <table border=2>
        <tbody id="tb">
            <tr>
                <td>suns</td>

```

```

        <td>1799-2-2</td>
        <td>su@123.com</td>
        <td><input type="button" value="del" onclick="delPerson(this);"/></td>
    </tr>
    <tr>
        <td>suns2</td>
        <td>1222-1-1</td>
        <td>su2@123.com</td>
        <td><input type="button" value="del" onclick="delPerson(this);"/></td>
    </tr>
    <tr>
        <td>suns3</td>
        <td>1332-1-1</td>
        <td>su3@123.com</td>
        <td><input type="button" value="del" onclick="delPerson(this);"/></td>
    </tr>
</tbody>
</table>
</body>
</html>

```

版本 2: style 属性改 CSS 属性

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function addPerson(){
        //获得三个文本框的值
        var name = document.getElementById("name").value;
        var birth = document.getElementById("birth").value;
        var email = document.getElementById("email").value;

        //创建<td>对象
        var nameTd = document.createElement("td");
        var nameTxt = document.createTextNode(name);
        nameTd.appendChild(nameTxt);

        var birthTd = document.createElement("td");
        var birthTxt = document.createTextNode(birth);
        birthTd.appendChild(birthTxt);

        var emailTd = document.createElement("td");

```



```

var emailTxt = document.createTextNode(email);
emailTd.appendChild(emailTxt);
//增加一个删除按钮(难点)
var delTd = document.createElement("td");
var delInput = document.createElement("input");
delInput.value="delPrson";
delInput.type="button";
    /*两个注意点:
        * 1 加入了 onclick 属性
        * 2 赋值时不能这样写: delPerson(this), 这表示传的是调用函数的值
    */
delInput.onclick=function(){
                    delPerson(this);
                };
delTd.appendChild(delInput);

//创建 tr
var tr = document.createElement("tr");
tr.onmouseover=function(){
    gaoLiang(this);
}
tr.onmouseout=function(){
    diLiang(this);
}

tr.appendChild(nameTd);
tr.appendChild(birthTd);
tr.appendChild(emailTd);
tr.appendChild(delTd);

var tb = document.getElementById("tb");
tb.appendChild(tr);
}

function delPerson(tag){    //注意不要写 delete, 是 JS 中的关键字
    var tr = tag.parentNode.parentNode;
    var tb = document.getElementById("tb");
    tb.removeChild(tr);
}

function gaoLiang(tag){
    //注意: 和 tag.style background-color 对比
    tag.style.backgroundColor = "gray";
}

function diLiang(tag){

```

//注意: 和 **tag.style background-color** 对比
tag.style.backgroundColor = "white";

}

</script>

</head>

<body id="b">

username:<input id="name" type="text" />

birthday:<input id="birth" type="text"/>

email:<input id="email" type="text"/>

<input type="button" value="add person" onclick="addPerson();" />

<table border=2>

<tbody id="tb">

<tr onmouseover="gaoLiang(this);" onmouseout="diLiang(this);">

<td>suns</td>

<td>1799-2-2</td>

<td>su@123.com</td>

<td><input type="button" value="del" onclick="delPerson(this);"/></td>

</tr>

<tr onmouseover="gaoLiang(this);" onmouseout="diLiang(this);">

<td>suns2</td>

<td>1222-1-1</td>

<td>su2@123.com</td>

<td><input type="button" value="del" onclick="delPerson(this);"/></td>

</tr>

<tr onmouseover="gaoLiang(this);" onmouseout="diLiang(this);">

<td>suns3</td>

<td>1332-1-1</td>

<td>su3@123.com</td>

<td><input type="button" value="del" onclick="delPerson(this);"/></td>

</tr>

</tbody>

</table>

</body>

</html>

版本 3:

jsDemo.html x my.js my.css header.html My Aptana

username: birthday: email:

<input checked="" type="checkbox"/>	suns	1799-2-2	su@123.com	<input type="button" value="del"/>
<input checked="" type="checkbox"/>	suns2	1222-1-1	su2@123.com	<input type="button" value="del"/>
<input checked="" type="checkbox"/>	suns3	1332-1-1	su3@123.com	<input type="button" value="del"/>
<input type="checkbox"/>	afwe	awef	wefwaefwef	<input type="button" value="delPerson"/>
<input type="checkbox"/>	afwe	awef	wefwaefwef	<input type="button" value="delPerson"/>
<input checked="" type="checkbox"/>	afwe	awef	wefwaefwef	<input type="button" value="delPerson"/>
<input checked="" type="checkbox"/>	afwe	awef	wefwaefwef	<input type="button" value="delPerson"/>

增加了 checkbox

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function addPerson(){
        //1 获得三个文本框的值
        var name = document.getElementById("name").value;
        var birth = document.getElementById("birth").value;
        var email = document.getElementById("email").value;

        //2 创建<td>对象
        var chboxTd = document.createElement("td");           //为 checkbox 添加<td>标签
        var chInput = document.createElement("input");
        chInput.type="checkbox";
        chboxTd.appendChild(chInput);

        var nameTd = document.createElement("td");
        var nameTxt = document.createTextNode(name);
        nameTd.appendChild(nameTxt);

        var birthTd = document.createElement("td");
        var birthTxt = document.createTextNode(birth);
        birthTd.appendChild(birthTxt);

        var emailTd = document.createElement("td");
        var emailTxt = document.createTextNode(email);
        emailTd.appendChild(emailTxt);

        //增加一个删除按钮(难点)
        var delTd = document.createElement("td");
```

世界因你而不同!

```

var delInput = document.createElement("input");
delInput.value="delPrsrson";
delInput.type="button";
    /*两个注意点:
        * 1 加入了 onclick 属性
        * 2 赋值时不能这样写: =delPerson(this), 这表示传的是调用函数的值
    */
delInput.onclick=function(){
    delPerson(this); //表示调用 delPerson()方法
};
delTd.appendChild(delInput);

//3 创建 tr
var tr = document.createElement("tr");
tr.appendChild(chboxTd);
tr.appendChild(nameTd);
tr.appendChild(birthTd);
tr.appendChild(emailTd);
tr.appendChild(delTd);

//4 添加 tr
var tb = document.getElementById("tb");
tb.appendChild(tr);
}

function delPerson(tag){    //注意不要写 delete, 是 JS 中的关键字
    var tr = tag.parentNode.parentNode;
    var tb = document.getElementById("tb");
    tb.removeChild(tr);
}

function selAll(){
    var tb = document.getElementById("tb");
    var inputs = tb.getElementsByTagName("input");    //从<tbody>找, 提高效率

    for(var i=0;i<inputs.length;i++){
        if(inputs[i].type=="checkbox"){
            if(inputs[i].checked){
                inputs[i].checked = "true";
            }else{
                inputs[i].checked= "false";
            }
        }
    }
}
}

```

```

function delAll(){
    var tb = document.getElementById("tb");
    var inputs = tb.getElementsByTagName("input");           //从<tbody>找，提高效率

    for(var i=inputs.length-1;i>=0;i--){
        if(inputs[i].type=="checkbox"){
            if(inputs[i].checked){
                tb.removeChild(inputs[i].parentNode.parentNode);
            }
        }
    }
}
</script>
</head>
<body id="b">
    username:<input id="name" type="text" />
    birthday:<input id="birth" type="text"/>
    email:<input id="email" type="text"/>
    <input type="button" value="add person"  onclick="addPerson();" />

    <table border=2>
        <tbody id="tb">
            <tr>
                <td><input type="checkbox"/></td>
                <td>suns</td>
                <td>1799-2-2</td>
                <td>su@123.com</td>
                <td><input type="button" value="del" onclick="delPerson(this);"/></td>
            </tr>
            <tr>
                <td><input type="checkbox"/></td>
                <td>suns2</td>
                <td>1222-1-1</td>
                <td>su2@123.com</td>
                <td><input type="button" value="del" onclick="delPerson(this);"/></td>
            </tr>
            <tr>
                <td><input type="checkbox"/></td>
                <td>suns3</td>
                <td>1332-1-1</td>
                <td>su3@123.com</td>
                <td><input type="button" value="del" onclick="delPerson(this);"/></td>
            </tr>
        </tbody>
    </table>
</hr>

```

```
<center>
    <input type="button" value="selectAll" onclick="selAll();" />
    <input type="button" value="delAll" onclick="delAll();" />
</center>
</body>
</html>
```

2) 级联选项

```
<select>
    <option></option>
</select>
```

SELECT 对象的属性

length	<option>个数
selectedIndex	用户选择下拉列表时选项的序号
options	数组，所有<option>的一个数组

案例 1：获取<select>的值

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
        function selectCity(){
            var sel = document.getElementById("contry");
            alert(sel.options[0].firstChild.nodeValue);
            alert(sel.options[0].text);
            alert(sel.options[0].value);
        }
    </script>
</head>
<body>
    国家：
    <select id="contry" name="contry" onchange="selectCity();">
        <option value="xz">请选择</option>
        <option value="china">中国</option>
        <option value="usa">美国</option>
        <option value="mg">蒙古</option>
    </select>
    <select name="contry">
```

```
        <option>请选择</option>
    </select>
</body>
</html>
```

案例 2：级联选项

第一步：

准备数据

```
var c = {
    china:["北京","上海","天津"],
    usa:["纽约","华盛顿","波特兰"],
    mg:["东京","北海道","大阪"]
};
```

第二步：

通过 **selectedIndex** 获得 option 对象

```
var sel = document.getElementById("contry");
var op = sel.options[sel.selectedIndex];
alert(op.text);
```

第三步：

通过第一级 select 的 value 获得城市数组

```
var c = {
    china:["北京","上海","天津"],
    usa:["纽约","华盛顿","波特兰"],
    mg:["东京","北海道","大阪"]
};

function selectCity(){
    var sel = document.getElementById("contry");
    var op = sel.options[sel.selectedIndex];
    alert(c[op.value]);           //通过国家的 value 获得城市数组
}
```

第四步:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      var c = {
        xz:["请选择"],
        china:["北京","上海","天津"],
        usa:["纽约","华盛顿","波特兰"],
        mg:["东京","北海道","大阪"]
      };

      function selectCity(){
        //从父 select 中取值
        var sel = document.getElementById("contry");
        var op = sel.options[sel.selectedIndex];
        var cityArr = c[op.value];           //通过国家的名字获得城市数组
        //把父 select 中取到的值放入子 select
        var city = document.getElementById("city");
        for(var i=0;i<cityArr.length;i++){
          var ops = new Option();           //1 创建一个 option 对象
          ops.text = cityArr[i];             //2 将城市的值赋值给 option 对象的 text 属性
          city.options[i] = ops;             //3 将对象赋值给 id 为 "city" 的 select 对象属性 options
        }
      }
    </script>
  </head>
  <body>
    国家:
    <select id="contry" name="contry" onchange="selectCity();">
      <option value="xz">请选择</option>
      <option value="china">中国</option>
      <option value="usa">美国</option>
      <option value="mg">蒙古</option>
    </select>
    <select id="city" name="city">
      <option>请选择</option>
    </select>
  </body>
</html>
```


案例 3：当 onmouseover 出现一个框，当 onmouseout 框没了

display:none 不显示
 black 显示块

知识点：

- display 属性：解决某个内容显示不显示
- 显示位置：position top left 属性
- 跟着鼠标走：事件属性 clientX clientY

案例 4：案例 3 规范的写法

test.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="stylesheet" type="text/css" href="my.css"></link>
    <script type="text/javascript" src="my.js"></script>
  </head>
  <body>
    <div>
      <span onmouseover="show(event);"
            onmousemove="show(event);"
            onmouseout="hidden();">core java</span>
      ||<span onmouseover="show(event);"
            onmousemove="show(event);"
            onmouseout="hidden();">hibernate</span>
      ||<span onmouseover="show(event);"
            onmousemove="show(event);"
            onmouseout="hidden();">javascript</span>
    </div>
    <div id="prom" class="prom">
      cccc
    </div>
  </body>
</html>
```

my.css

```
div{
  cursor:pointer;
```

```

}
.prom{
    width:70px;
    height:60px;
    background-color:black;
    color:white;
    position:absolute;
    display:none;
}

```

my.js

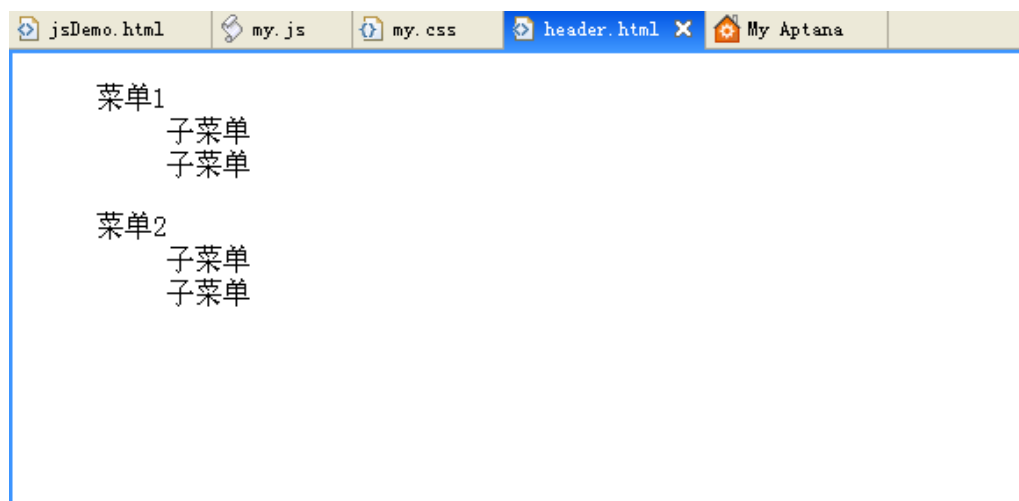
```

function show(event){
    var div = document.getElementById("prom");
    div.style.display = "block";
    div.style.left = event.clientX;
    div.style.top = event.clientY;
}

function hidden(){
    var div = document.getElementById("prom");
    div.style.display = "none";
}

```

案例 5：级联菜单



如图：点击【菜单】，则子菜单消失，出现

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function change(tgg){
        if(tgg.nextSibling.nextSibling.style.display=="none"){

```

世界因你而不同！

```

        tgg.nextSibling.nextSibling.style.display="block";
    }else{
        tgg.nextSibling.nextSibling.style.display="none";
    }
}
</script>
<style type="text/css">
    ul,li{
        list-style-type:none;
        cursor:pointer;
    }
</style>
</head>
<body>
    <ul>
        <li onclick="change(this);">菜单 1</li>
        <ul>
            <li>子菜单</li>
            <li>子菜单</li>
        </ul>
    </ul>
    <ul>
        <li onclick="change(this);">菜单 2</li>
        <ul>
            <li>子菜单</li>
            <li>子菜单</li>
        </ul>
    </ul>
</body>
</html>

```

四、BOM 浏览器内置对象 ***

BOM: browser object module 浏览器对象模型
浏览器对象模型 比如: **document**

1. 常用的浏览器内置对象

● window 一个浏览器窗口

注意: **window** 对象的方法可以不使用“对象.方法”的形式, 可以直接写方法

- ✓ **alert();** 输出
- ✓ **confirm();** 对话框, 返回值为 **boolean** 类型
- ✓ **prompt();**
- ✓ **setTimeout();** 在某个特定的时间间隔后, 执行指定的功能

```
i = setTimeout(function(){
    alert("xxx");
}, 5000); //5 秒
```

- ✓ **clearTimeout(i);** 取消所设置的 **timeout** 操作, 需要传入参数 **i** (**i** 代表要取消的 **timeout**)
- ✓ **setInterval();** 周期性执行某个功能
- ✓ **clearInterval();**

- ✓ **open();** 打开新的浏览器页面
- ✓ **close();** 关闭浏览器页面 **IE** 可以; **firefox** 不可以

● document 一张页面

- ✓ **write();**
- ✓ **getElementById();**
- ✓ **getElementsByTagName;**
- ✓ **createElement();**
- ✓ **createTextNode();**
- ✓
- ✓ 属性
 - ◆ **images[]** 所有图片对象
 - ◆ **forms[]** 所有表单对象<form>
 - **submit()**方法
 - **elements[]**属性
 - ◆ **anchor[]** 所有超级链接对象<a>

● location 位置, 代表地址栏

2. window 对象的常用方法

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
```

```

function testConfirm(){
    var result = confirm("man or woman?");
}

function testPrompt(){
    var n = prompt("aaa");
    alert(n);
}

var i;
function testTimeout(){
    i = setTimeout(function()    //返回值 i 表示 setTimeout()的一个编号
        {alert("aaa");
        },3000);
}

function testclearTimeout(){
    clearTimeout(i);            //要传入我们要取消的 timeout 操作的编号
}

function testInterval(){
    setInterval(function(){
        alert("interval");
    },3000);
}

```

</script>

</head>

<body>

<!--confirm()测试-->

<input type="button" value="textConfirm" onclick="testConfirm();" />

跳转了

<!--prompt()测试 IE 可以，Firefox 不行-->

<input type="button" value="testP" onclick="testPrompt();" />

<!--setTimeout()测试-->

<input type="button" value="testT" onclick="testTimeout();" />

<!--clearTimeout()测试-->

<input type="button" value="testTclear" onclick="testclearTimeout();" />

<!--setInterval()测试-->

<input type="button" value="testInterval" onclick="testInterval();" />

<!--close()测试-->

<input type="button" value="testclose" onclick="window.close();" />

<!--open()测试-->

<input type="button"

value="testOpen"

onclick="window.open('gaoliang.html','new','height=100,top=200,left=200');" />

</body>

</html>

confirm(): 对话框，返回值为 **boolean** 类型

注意： 一个实用的小技巧 `跳转`

3. document 对象的属性

form 对象

form 对象的子标签 `elements[]`

`submit();` 当你调用 `submit()` 方法

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript">
      function testForm(){
        //alert(document.forms[0].method);          //获得<form>提交的方法
        alert(document.forms[0].elements[0].value);
      }
      function send(){
        //提交的另一种方法，注意，此处按钮不是 submit 类型，是 button 类型
        document.forms[0].submit();
      }
    </script>
  </head>
  <body>
    <form method="post" action="gaoliang.html">
      <input type="button" value="textForm" onclick="send();"/>
    </form>
  </body>
</html>
```

4. Location 的常用方法

location.href="url" 当前页面跳转到一个新页面

案例： 用按钮实现超级链接

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

世界因你而不同！

```
<script type="text/javascript">
    function hrf(){
        location.href = "gaoliang.html";
    }

</script>
</head>
<body>
    <input type="button" value="textForm" onclick="href();" />
</body>
</html>
```

五、JQuery

1. 简介

JQuery: JS 框架

框架的作用：软件开发过程中的半成品，简化程序员的开发

JS 框架：

- **Prototype**
- **JQuery**
- **ExtJS**

JQuery 作用：

- 屏蔽浏览器差异 因为 JS 的缺点（存在浏览器差异），应用而生 JQuery
- 简化开发
- 简化 **Ajax** 开发

JQuery 下载：

www.jquery.com download 1.6

2. 第一个 JQuery

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testJQuery(){
        $("#div1").html("xxx")                ##表示 id 选择器
        .css("width",50)
        .css("height",100)
        .css("border","solid 1px red");
      }
    </script>
  </head>
  <body>
    <div id="div1">
    </div>
    <input type="button" value="click" onclick="testJQuery();" />
  </body>
</html>
```

知识点:

- **<script type="text/javascript" src="jquery-1.4.js"></script>** 引入 JQuery 框架
- **\$("#id");** 通过 id 获取"标签对象" **非常注意: 此对象不是<div>对象而是 JQuery 对象**
- **.html("xxx")** 向标签对象中放入文本信息
- **.css("width",50)** 加入 css 样式的方法

1) 此对象非彼对象

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testJQuery(){
        alert($("#div1"));                //这样获得的是 JQuery 中的对象
        alert(document.getElementById("div1")); //这样获得的才是 JS 中的<div>对象
      }
    </script>
  </head>
  <body>
    <div id="div1">
    </div>
```



```
        <input type="button" value="click" onclick="testJQuery();" />
    </body>
</html>
```

注意: JQuery 中的对象实际上封装了 JS 中的<div>对象

```
//伪代码
var jqueryDiv={
    Object:javascript div
}
jqueryDiv.html(){
    javascript.div 做操作
}
```

2) JQuery 对象获得 JS 对象

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testObject(){
        alert(document.getElementById("div1"));
        alert($("#div1").get(0));
      }
    </script>
  </head>
  <body>
    <div id="div1">
    </div>
    <input type="button" value="click" onclick="testObject();" />
  </body>
</html>
```

3) JS 对象获得 JQuery 对象

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testObject(){
        var div = document.getElementById("div1");
        alert(div);
        alert($(div));
      }
    </script>
  </head>
  <body>
    <div id="div1">
    </div>
  </body>
</html>
```

```

    }
    </script>
</head>
<body>
    <div id="div1">
    </div>
    <input type="button" value="click" onclick="testObject();" />
</body>
</html>

```

注意：只有 **JQuery** 对象，才可以调用 **JQuery** 框架中的方法，比如 **.html()**、**.css()**

3. 获得 JQuery 对象的方式 ***

- **id 选择器** **<element id="aaa">** **\$("#aaa");**
- **类型选择器** **<element class="style1">** **\$(".style1");**
- **标签选择器** **<p>xxx</p>** **\$("p");**
- **派生选择器** 参看案例 2 **\$("ul li")**
- **直接派生** 参看案例 3 **\$("ul>li");**

案例 1：标签选择器

```

<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
        function testTag(){
            var p = $("p");
            alert(p.get(0));
            alert(p.get(1));
        }
    </script>
    </head>
    <body>
        <p>aa</p>
        <p>bb</p>
        <input type="button" value="test" onclick="testTag();" />
    </body>
</html>

```

注意：通过标签选择器获得的对象，返回值仍然是一个 **JQuery** 对象，**JQuery** 对象中有两个 **<p>** 对象

案例 2：派生选择器

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testTag(){
        var jqueryli = $("#ul li");      //能操作所有<ul>下的<li>
        jqueryli.css("color", "red");
      }
    </script>
  </head>
  <body>
    <ul>
      <li>xxx</li>
      <ol>
        <li>xxx</li>
      </ol>
    </ul>
    <input type="button" value="test" onclick="testTag();"/>
  </body>
</html>
```

案例 3：直接派生选择器 ***

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testTag(){
        var jqueryli = $("ul li");
        jqueryli.css("color", "red");
      }
      function testTag2(){
        var jqueryli = $("ul>li");
        jqueryli.css("color", "red");
      }
    </script>
  </head>
  <body>
    <ul>
```

```

        <li>xxx</li>
      </ol>
    </ul>
    <input type="button" value="test" onclick="testTag2();" />
  </body>
</html>

```

4. 和表单相关的属性

<input type="text" value="xxxx" /> **\$(":text");** //加冒号即可

:checkbox

:submit

:radio

:button

:text

:reset

:file

:hidden

:input

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testInput(){
        alert($(":input").get(1).value);    //输出 tttt222, 所有<input>中的第 1 个
        //alert($(":text").get(0).value);    //输出 tttt, 所有 text 中的第 0 个
        //alert($(":password").get(0).value);
      }
    </script>
  </head>
  <body>
    <input type="text" value="tttt" />
    <input type="text" value="ttt222" />
    <input type="password" value="ppppword" />
    <input type="button" value="test" onclick="testInput();" />
  </body>
</html>

```

5. 过滤属性

1) 基本过滤属性

:first	\$(":text:first").css("width",400);	将第一个文本框 CSS 改变
:last		
:not(selector)	selector 为 id 选择器	
:even	奇数（从 0 开始，看到的效果是第 1 个）	
:odd	偶数	
:eq(index)	等于标签 index 的	
:gt(index)		
:lt(index)		

案例 1：将第一个文本框 CSS 改变

:first

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function testText(){
        $( ":text:first" ).css("width",400);
      }
    </script>
  </head>
  <body>
    <input type="text" value="tttt" />
    <input type="text" value="tttt2222" />
    <input type="button" value="test" onclick="testText();"/>
  </body>
</html>
```

案例 2：将奇数行列表颜色变红

:odd

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```

<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    function testText(){
        $("ul li:odd").css("background-color","red");
    }
</script>
</head>
<body>
    <ul>
        <li>----1----</li>
        <li>----2----</li>
        <li>----3----</li>
        <li>----4----</li>
        <li>----5----</li>
        <li>----6----</li>
        <li>----7----</li>
        <li>----8----</li>
        <li>----9----</li>
    </ul>
    <input type="button" value="test" onclick="testText();"/>
</body>
</html>

```

案例 3：将 index 为 2 的背景变红

:eq(index)

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function testText(){
                $("ul li:eq(1)").css("background-color","red");
            }
        </script>
    </head>
    <body>
        <ul>
            <li>----1----</li>
            <li>----2----</li>
            <li>----3----</li>
            <li>----4----</li>
            <li>----5----</li>
        </ul>
    </body>
</html>

```

```

        <li>----6----</li>
        <li>----7----</li>
        <li>----8----</li>
        <li>----9----</li>
    </ul>
    <input type="button" value="test" onclick="testText();" />
</body>
</html>

```

案例 4：除了 selector 不操作，其余都操作

:not(selector)

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function testText(){
                $("ul li:not(#li4)").css("background-color","red");
            }
        </script>
    </head>
    <body>
        <ul>
            <li>----1----</li>
            <li>----2----</li>
            <li>----3----</li>
            <li id="li4">----4----</li>
            <li>----5----</li>
            <li>----6----</li>
            <li>----7----</li>
            <li>----8----</li>
            <li>----9----</li>
        </ul>
        <input type="button" value="test" onclick="testText();" />
    </body>
</html>

```

2) 内容过滤

:contains(text)	包含指定内容
:empty	找到没有标签体的标签（独标签，比如<input type="button"/>）
:has(selector)	是否包含
:parent	过滤是父标签的标签

案例 1: :contains(text)

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test(){
        $("div:contains('abc')").css("color","red");;
      }
    </script>
  </head>
  <body>
    <div>abcdewef</div>
    <div>efeeeee</div>
    <input type="button" value="test" onclick="test();"/>
  </body>
</html>
```

案例 2: :has()

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test(){
        $("div:has('span')").css("color","red");
      }
    </script>
  </head>
```



```
<body>
  <div>abcdewef</div>
  <div>
    <span>xxx</span>
  </div>
  <input type="button" value="test" onclick="test();"/>
</body>
</html>
```

案例 3: :parent

: parent 如果为父标签

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test() {
        $("div:parent").css("width","50").css("height","100").css("background-color","red");
      }
    </script>
    <style type="text/css">
      .div1 {
        border:solid 1px red;
        width:50px;
        height:50px;
      }
    </style>
  </head>
  <body>
    <div class="div1"></div>
    <div>
      <span>aaa</span>
    </div>

    <input type="button" value="test" onclick="test();"/>
  </body>
</html>
```

3) 和显示相关的过滤属性

:visible

:hidden

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test(){
        $("div:visible").css("color","red");
      }
    </script>
  </head>
  <body>
    <div style="display:none">abcdewef</div>
    <div>
      <span>xxx</span>
    </div>
    <input type="button" value="test" onclick="test();"/>
  </body>
</html>
```

4) 和表单点选相关的过滤属性

:checked \$(":checkbox:**checked**"); 对被选择的 checkbox 选择项过滤

:selected \$("select option:**selected**"); 对下拉列表中选择 option 过滤

```
$("select option:selected").get(0).value
```

5) 和标签自身属性相关的过滤属性

[attribute]

[attribute=value]

对有 id 属性的<div>改为 red, id 为"div2"的背景颜色改为 yellow

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
```

世界因你而不同!

```

        function test(){
            $("div[id]").css("color","red");
            $("div[id='div2']").css("background-color","yellow");
        }
    </script>
</head>
<body>
    <div style="display:block">aaa</div>
    <div id="div2">bbb</div>

    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

6) 子标签相关的过滤属性

:nth-child(index|odd|even|epr)

Index: 从 1 开始 odd: 偶数 even: 奇数 **epr:表达式**

:nth-child(xn+y) 举例: **:nth-child(2n+1);** **:nth-child(n+3);**

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    function test(){
        $("ul li:nth-child(4)").css("color","red");
        $("ul li:nth-child(2n+1)").css("color","red");
    }
</script>
</head>
<body>
    <ul>
        <li>11</li>
        <li>12</li>
        <li>121</li>
        <li>123</li>
        <li>32324</li>
    </ul>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

6. JQuery 中操作 CSS 属性

第一种方式

```
<html>

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test(){
        //第一种方式
        var s = {
          border:"solid 1px red",
          backgroundColor:"#FFCCDD"
        };
        $("div").css(s);
      }
    </script>
  </head>
  <body>
    <div>asdfwef</div>
    <input type="button" value="test" onclick="test();"/>
  </body>
</html>
```

第二种方式

```
<html>

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function test(){

        //第二种方式
        $("span").addClass("style01");      //必须用类选择器，将.class 样式放入进来
        //$("span").removeClass("style");    //取消

      }
    </script>
    <style type="text/css">
```

```

        .style{
            border:solid 1px red;
        }
        .style01{
            border:solid 1px black;
        }
    </style>
</head>
<body>
    <div>aaa</div>
    <span class="style">bbb</span>
    <br/>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

7. 修改对象一般属性

- **`$("img").attr("width",100);`**
- **`$(":text").attr("value","xxx");`**
- **`$(":text").removeAttr("value","");`**
- **`$(":text").attr({property:value,property:value});`**
- **`$(":text").attr("value")`**

1) 案例：利用 **attr** 删改查 ***

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function test(){
                $(":text").attr("value","hello jquery");           //改
                alert($(":text").attr("value"));                //查   获得属性值
                $(":text").removeAttr("value","");                //删   清空
            }
        </script>
    </head>
    <body>
        <input type="text" />
        <input type="button" value="test" onclick="test();"/>
    </body>
</html>

```

8. 创建删除一个 JQuery

1) 插入节点:

- **append();** 放置在父标签内部最后位置（追加） 父亲加孩子
- **prepend();** 放置在父标签最前的位置（插入）
- **after();** 针对同级标签
- **before();** 针对同级标签
- **appendTo();** 用于父子标签 孩子加父亲
- **prependTo();**

2) 删除节点:

- **remove()** 全删除
- **remove("div")** 删除指定的标签
- **empty()** 清空节点中的文本内容

3) 节点的导航:

- **children()** 只考虑子元素，不考虑其他后代元素
- **next()** 下一个兄弟
- **prev()** 上一个兄弟
- **siblings()** 兄弟们
- **parent()** 父元素

4) 设置和获取 html 的值

- **html("xxx")||\$("div").html()** 加入 html 注意：辨析 **html()**和 **text()** 案例 5
- **\$("div").text()** 指加入文本内容
- **\$(XX).val()** 设置和修改文本框和下拉框的值
比 **\$(".text").attr("value")** 功能弱

5) innerHtml 属性 ***

```
var div = document.getElementById("div");  
div.innerHTML = "<strong>xiaoheizi</strong>"; 相当于 html("xxx")方法
```

案例 1: append()

```
$("html");  
$("<div>xxxx</div>");  
$("body").append($("<div>xxxx</div>"));
```

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
    <script type="text/javascript" src="jquery-1.4.3.js"></script>  
    <script type="text/javascript">  
      function test(){
```

```

        var d = $("<div>aaa</div>");
        $("body").append(d);    //相当于 d.appendTo($("body"));
    }
</script>
</head>
<body>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

案例 2: **remove()** 删除全部

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function test(){
                $("body").remove();
            }
        </script>
    </head>
    <body>
        <input type="button" value="test" onclick="test();"/>
    </body>
</html>

```

案例 3: **empty()**

清空节点中的文本内容

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function test(){
                $("div").empty();    //注意调用对象，清空 div 中的内容
            }
        </script>
    </head>
    <body>
        <div>
            <input type="button" value="test" onclick="test();"/>
        </div>
    </body>
</html>

```

```

    </script>
</head>
<body>
    <div>afwefw</div>
    <div>dwwe</div>
    <span>sdfwe</span>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

案例 4: each()方法 ***

注意：不好理解，取出<body>中的子节点，再循环取出所有子节点

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            function test() {
                var childs = $("body").children();
                childs.each(function(i){    //i 代表每次循环的次数    each()函数是拿出所有 JS 对象
                    //alert(this);        //this 代表拿到的 JS 对象
                    alert($(this).attr("value"));
                    });
            }
        </script>
    </head>
    <body>
        <div>afwefw</div>
        <div>dwwe</div>
        <span>sdfwe</span>
        <input type="button" value="test" onclick="test();"/>
    </body>
</html>

```

案例 5: html()和 text()的区别

.html() 设置和获取 html()方法的作用

```
<html>
```

世界因你而不同！


```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    function test(){
        $("div").html("<strong>nzbbsn</strong>");
        $("span").text("<strong>seeewe</strong>");

    }
</script>
</head>
<body>
    <div></div>
    <span></span>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

案例 6: `$(:text).val()`和`$(":text").attr("value")`

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    function test(){
        alert($(":button").val("kkee"));
        alert($(":button").attr("value"));
        alert($(":text").val());
        alert($(":text").attr("value"));
    }
</script>
</head>
<body>
    <input type="text" value="aaa"/>
    <input type="button" value="test" onclick="test();"/>
</body>
</html>

```

9. 事件

bind() 绑定

1) 事件监听的方式

第一种方式:

```
<input type="button" value="test" onclick="test();"/>
```

第二种方式:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function fun(){
        var ip = document.getElementById("but");
        ip.onclick = test;
      }

      function test(){
        alert("xxxx");
      }
    </script>
  </head>
  <body onload="fun();">
    <input type="button" id="but" value="test" />
  </body>
</html>
```

第三种方式: **JQuery 方式 *****

常用

世界因你而不同!

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    //按这样写，用法是在 onload 事件触发时调用此函数中的内容
    $(function() {
        /*$(":button").bind("click",function(){           //bind()绑定
            alert("xxxx");
        });*/
        //简化写法
        $(":button").click(function(){
            alert("xxxx");
        });
    })
</script>
</head>
<body>
    <input type="button" id="but" value="test" />
</body>
</html>
```

2) 合成事件

事件合成函数: `hover()`

hover(enter , leave)

相当于 **onmouseover()+onmouseout()**

模拟光标悬停事件

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script type="text/javascript" src="jquery-1.4.3.js"></script>
<script type="text/javascript">
    //按这样写，用法是在 onload 事件触发时调用此函数中的内容
    $(function(){
        /*$("div").mouseover(function(){
            alert("xxxx");
        });
        $("div").mouseout(function(){
            alert("yyyy");
        });*/
    });
</script>
</html>
```

世界因你而不同！

```

        $("div").hover(function(){
            alert("xxxx");
        },function(){
            alert("yyyy");
        });
    })
</script>
</head>
<body>
    <div style="border:solid 1px red;width:100px;height:100px">ccc</div>
    <input type="button" id="but" value="test" />
</body>
</html>

```

事件合成函数: **toggle()**

toggle(fun1,fun2,fun3,...)

鼠标连续单击事件

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="jquery-1.4.3.js"></script>
        <script type="text/javascript">
            $(function(){
                $(":button").toggle(function(){
                    $("#div").hide("slow");
                },function(){
                    $("div").show("fast");
                },function(){
                    $("#div").css("color","red");
                },function(){
                    $("div").css("color","green");
                });
            })
        </script>
    </head>
    <body>
        <div style="border:solid 1px black;width:100px;height:100px">cccc</div>
        <input type="button" value="test" />
    </body>
</html>

```

事件对象的属性

event.type

event.target

event.pageX/pageY 相当于 **clientX/clientY**

自动触发事件 **trigger("click")**

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      //按这样写，用法是在 onload 事件触发时调用此函数中的内容
      $(function(){
        $("div").click(function(){
          alert("xxxx");
          $("#div").trigger("click");        //我触发一次单击事件后，自动触发事件
        });
      })
    </script>
  </head>
  <body>
    <div style="border:solid 1px red;width:100px;height:100px">ccc</div>
  </body>
</html>
```

10. 动画

- **show()** 显示一个表单元素 相当于 **display:block**
 - **hide()** 隐藏一个表单元素 相当于 **display:none**
- 参数:
- slow**
 - fast**
 - nomal**
- **fadeIn()** 透明度增加至消失
 - **fadeOut()** 透明度减小至出现

世界因你而不同!

- **slideUp()** 缩小至消失
- **slideDown()** 增大至还原

案例 1: **show() hide()**

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      $(function(){
        $(":button").toggle(function(){
              $("div").hide("slow");
        },function(){
              $("div").show("fast");
        });
      })
    </script>
  </head>
  <body>
    <div style="border:solid 1px black;width:100px;height:100px">cccc</div>
    <input type="button" value="test" />
  </body>
</html>
```

案例 2: 操作图片

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script type="text/javascript" src="jquery-1.4.3.js"></script>
    <script type="text/javascript">
      function fun1(){
        $("img").fadeIn();      //透明度增加至消失
      }
      function fun2(){
        $("img").fadeOut();      //透明度减小至出现
      }
      function fun3(){
        $("img").slideUp();      //缩小至消失
      }
    </script>
  </head>
  <body>
    <img alt="A placeholder image for the jQuery effects demo." data-bbox="100 920 200 980"/>
  </body>
</html>
```

```
    }  
    function fun4(){  
        $("img").slideDown(); //增大至还原  
    }  
</script>  
</head>  
<body>  
    <input type="button" value="fadeIn" onclick="fun1();"/>  
    <input type="button" value="fadeOut" onclick="fun2();"/>  
    <input type="button" value="slidUp" onclick="fun3();"/>  
    <input type="button" value="slidDown" onclick="fun4();"/>  
    </img>  
</body>  
</html>
```