知识点列表

编号	名称	描述	级别
1	什么是 javascript	了解脚本语言 JavaScript	*
2	javascript 的一些特点	了解 JS 的一些特性	*
3	javascript 的组成部分	了解 JS 的组成	*
4	标识符	了解 JS 中的常用标识符	*
5	语句	语句以分号结尾	*
6	数据类型	掌握常用数据类型	**
7	什么是 DOM?	初步了解 JS 的 Dom 操作	*
8	W3c dom 模型	熟练掌握 JS 的 Dom 操作	***

注： "*"理解级别 "**"掌握级别 "***"应用级别

目录

1. javascript 基础

1.1. 什么是 javascript *

1) Javascript 是网景公司开发的一种在浏览器端执行的脚本语言
   需要嵌入到 html 当中才能执行
2) 其作用主要包括：
   - ✓ 数据验证(指的是，对表单中的数据进行合法性验证，只有验证通过才能提交)
   - ✓ 操作网页，实现一些动态效果
   - ✓ 访问浏览器，获得浏览器的一些信息。(比如获得浏览器的类型、版本等)
   - ✓ ajax 核心技术之一

1.2. javascript 的一些特点 *

1) javascript 是类 c 的语言
2) javascript 脚本可以保存在 **.js** 文件里，也可以直接写在 html 文件里
3) javascript 基于对象，不是纯粹的面向对象的语言
   比如，没有定义类的语法，也没有继承和多态
4) javascript 是一种弱类型语言，即变量在声明时，不能明确声明其类型
   变量的类型是在运行时确定的，并且可以随时改变

1.3. javascript 的组成部分 *

1) **ECMAScript 规范**
   - ✓ 主要定义了 javascript 的语言基础部分
   - ✓ 各个浏览器都严格遵守该规范，没有兼容性问题
   - ✓ ECMAScript 规范由 ECMA 制订
2) **dom**
   主要定义了如何将 html 转换成一棵符合 dom 规范的树，并且如何对这棵树进行相应的操作。该规范由 w3c 定义，但是，部分浏览器没有严格遵守该规范。写代码时需要考虑兼容性问题。
3) **bom (browser object model)**
   浏览器内置的一些对象，用来操作窗口。

这些对象包括 window、screen、location、navigator、document、XmlHttpRequest 等。虽然该部分没有规范，但是，各个浏览器都支持这些对象

2. 语言基础 **

2.1. 标识符 *

由字母、$、下划线开头，后面可以是字母、$、下划线和数字。比如"A$_"

2.2. 语句 *

语句以" ;"结尾。

2.3. 数据类型 **

1) **基本类型**

a. **number（数字类型）**

- ✓ number 类型代表数字，不管是整数、小数都是 number
- ✓ number 类型有一个对应的包装类 Number
- ✓ number -- > string : toString()方法
- ✓ 最大值 :Number.MAX_VALUE
- ✓ 最小值 :Number.MIN_VALUE
- ✓ 如果超过这个范围，会返回 Infinity,或者-Infinity

b. **string（字符串）**

- ✓ string 属于基本类型，没有 char 类型。
- ✓ string 有一个对应的包装类 String。
- ✓ length 属性　　返回字符串的长度
- ✓ charAt(index)　　返回指定位置的字符
- ✓ substring(from, to)　　返回子字符串
- ✓ indexOf(str)　　指定字符串在原字符串中第一次出现的位置。
- ✓ lastIndexOf(str)　　指定字符串在原字符串中最后一次出现的的位置。
- ✓ match(regexp)　　返回匹配指定正则表达式的字符串，返回的结果是一个数组。
- ✓ search(regexp)　　返回按照正则表达式检索到的字符串位置。
- ✓ toLowerCase/toUpperCase　　返回小写/大写形式
- ✓ replace(regexp,newStr)；　　替换符合正则表达式规定的字符串

c. **boolean（布尔类型 true/false）**

- ✓ boolean 类型使用时，要注意的问题
  - ■ 不为空的字符串，转换成 true。
  - ■ 非零的数字，转换成 true。
  - ■ null、undefined 转换成 false。
- ✓ boolean 类型也有包装类 Boolean。

d. **null（空类型 只有一个值 null）**

- ✓ 在使用 typeof 测试类型时，返回 object。

e. **undefine（未定义类型）**

- ✓ 未定义类型只有一个值 underfine。

2) **引用类型**

a. Array

- ✓ 数组的大小可以改变，并且数组元素的类型可以混合存放。
- ✓ 创建数组的第一种方式使用 Array
- ✓ 创建数组的第二种方式使用[]
- ✓ 数组的常用属性和方法
  - ■ length 属性： 返回数组的长度
  - ■ toString()方法： 返回数组的字符串表示
  - ■ concat()方法, 用于连接两个数组,生成一个新数组
  - ■ join()方法， 用于将数组中的各个元素连接成字符串
  - ■ reverse()方法， 将数组反转。不会生成新数组
  - ■ slice() 用于截取数组的一部分并以数组的形式返回。原数组不会有任何变化。
  - ■ sort()排序， 可通过如下形式来重新定义排序方式：

```
var arr4 = arr3.sort(
        function(a1,a2){
            return -a1.length + a2.length ;}) ;
```

b. Function

- ✓ 使用 function 关键字定义函数。
- ✓ 函数不能有返回类型，但是可以有返回值。
- ✓ 函数内部，可以使用 arguments 对象来获得参数值。
- ✓ 函数没有重载。

c. 变量的作用域

- ✓ js 引擎(浏览器内部负责解释执行 javascript 脚本的模块)在执行 js 脚本时
- ✓ 执行过程分为两个阶段：
  - ■ 第一阶段：
    将所有的变量声明以及函数的定义存放到一个特殊的对象(活动对象)里。
    具体过程：

在解析<script>的代码时，会将变量的声明以及函数的定义存放到全局的活动对象里面。

在遇到某个函数时，会将该函数内部声明的变量及函数的定义存放到局部活动对象里面。

- 第二阶段：

  执行 js 代码，在碰到变量时，会先从该代码所对应的活动对象里查找该变量的定义，如果没有找到，则向上查找对应的活动对象。

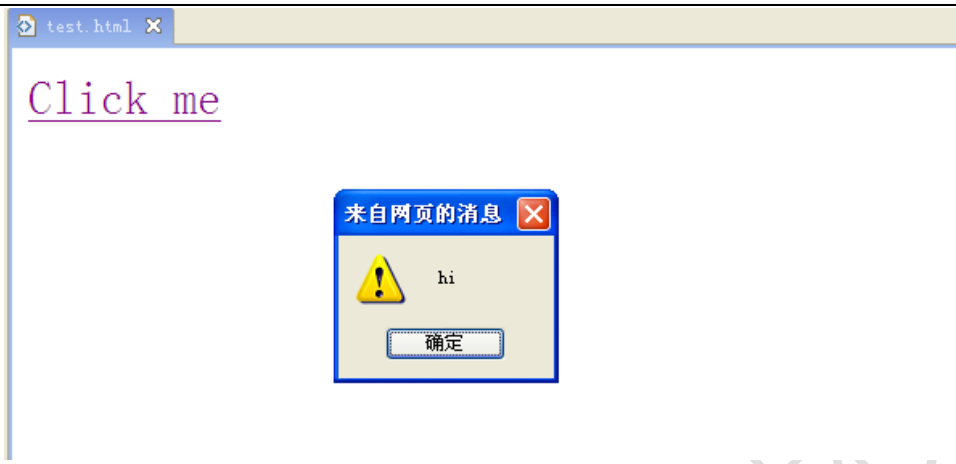
d. Math

- ✓ random()： 随机生成一个 0 到 1 之间的数字。
- ✓ ceil()： 对一个数上舍入。
- ✓ floor()： 对一个数下舍入

2.3.1. 基本数据类型 **

- **演示 1**

```
1  <html>
2    <!--基本数据类型-->
3    <head>
4      <script>
5          function f1(){
6              alert('hi');
7          }
8
9      </script>
10   </head>
11
12   <body style="font-size:30px;">
13
14     <a href="" onclick="f1();">Click me</a>
15
16   </body>
17
18 </html>
19
```

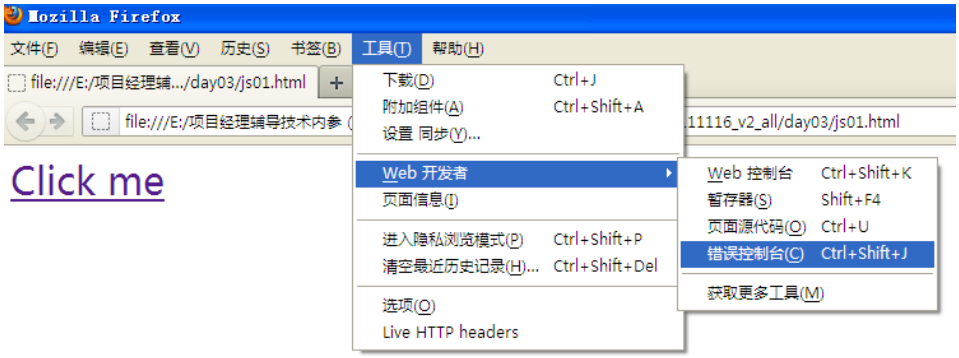


- **演示 2**

```
5       function f1(){
6           /* 是一种弱类型语言，即变量在声明时
7             不能明确声明其类型*/
8           var i = 100;
9
10          /* 变量的类型是在运行
11             时确定的，并且，可以随时改变*/
12          i = 'abc';
13
14          /* 可以使用typeof获知变量的类型*/
15          alert(typeof i);
16      }
```

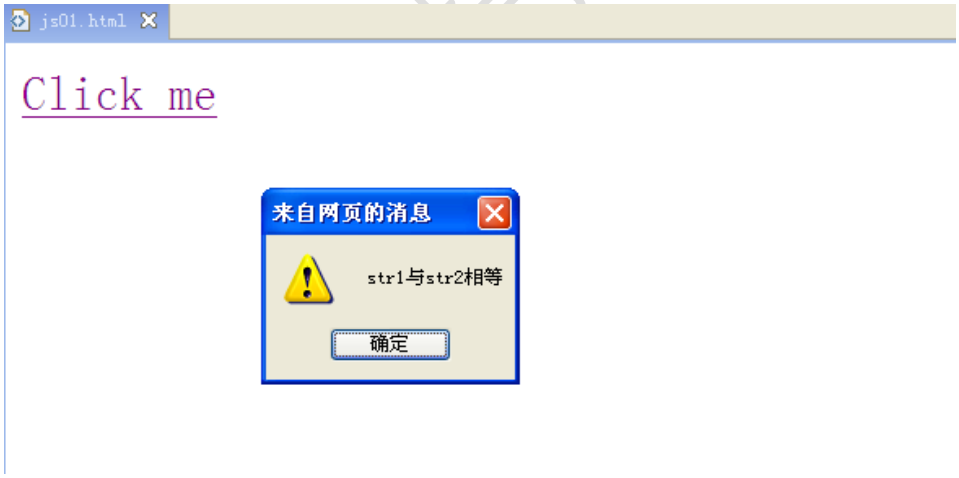
- **演示 3**

JS 调试错误：工具 -> Web 开发者 -> 错误控制台



- **演示 4**

```
20
21          var str1 = "abc";
22          var str2 = "abc";
23          if(str1 == str2){
24              alert("str1与str2相等");
25          }
```



- **演示 5**

```
27              var i2 = 100;
28              var s2 = '100';
29              /* "=="的作用是比较值 */
30              if(i2 == s2){
31                  alert("i2与s2相等");
32              }
```

- **演示 6**

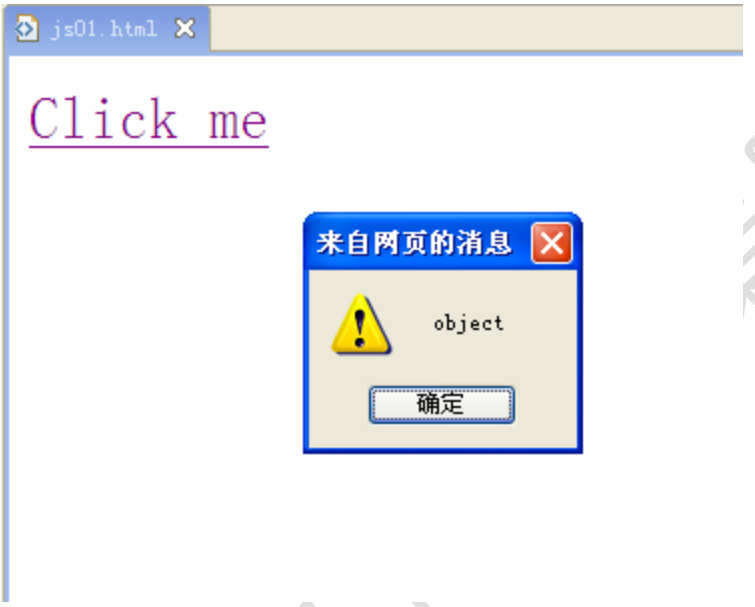
```
28              var s2 = '100';
29              /* "=="的作用是比较值 */
30              if(i2 == s2){
31                  //alert("i2与s2相等");
32              }
33
34              /* "==="的作用是先比较类型，只有类
35                 型一致，才比较值 */
36              if(i2 === s2){
37                  alert("i2与s2相等");
38              }else{
39                  alert("i2与s2不相等");
40              }
```

- **演示 7**

```
6      function f1(){
7
8          var flag = false;
9          alert(typeof flag);
```

- **演示 8**

```
 6⊟        function f1(){
 7
 8             var s1 = null;
 9⊟            /* null类型，在使用typeof测试时，会
10                输出object类型 */
11            alert(typeof s1);
12
```



- **演示 9**

```
 6⊟        function f1(){
 7
 8             var s3;
 9⊟            /* 一个变量声明过了，但没人赋值
10                类型是undefined */
11            alert(typeof s3);
12        }
```

- **演示 10**

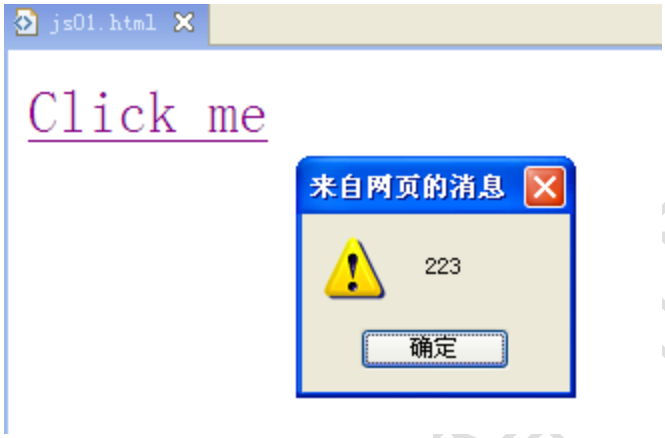
```
/* boolean类型使用时，要注意的问题
   a，不为空的字符串，转换成true。
   b，非零的数字，转换成true。
   c，null、undefined转换成false */
function f2(){
    //var flag = 'hello';
    //var flag = '';
    //var flag = 10;
    //var flag = null;
    var flag;
    if(flag){
        alert(true);
    }else{
        alert(false);
    }
}
```

- **演示 11**

```
7
8       /* 数据类型转换
9          string --> number
10         number -->string */
11      function f3(){
12          //var s1 = '150';
13          var s1 = '150.03';
14          //var s2 = parseInt(s1) + 100;
15          var s2 = parseFloat(s1) + 100;
16          alert(s2);
17
```

- **演示 12**

```
8       /* 数据类型转换
9          string --> number
10         number -->string */
11      function f3(){
12
13          var s3 = '123a';
14          var n3 = parseInt(s3);
15          alert(n3 + 100);
16
```

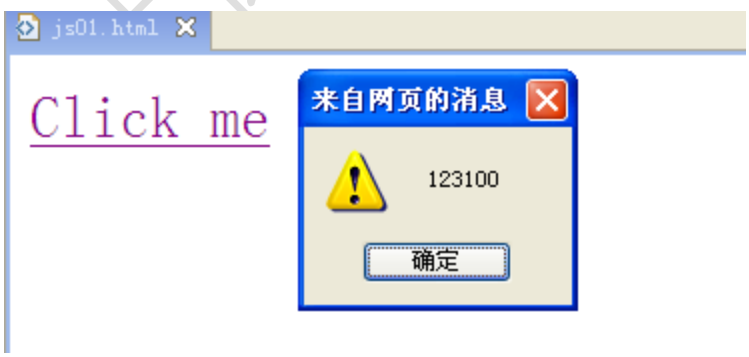


- **演示 13**

```
/* 数据类型转换
   string --> number
   number -->string */
function f3(){

    /* isNaN(): 如果一个值是非数字，返回true */
    var s4 = '123a';
    if(isNaN(s4)){
        alert('非数字');
    }else{
        alert('是一个数字');
    }
```

- **演示 14**

```
 8        /* 数据类型转换
 9           string --> number
10           number -->string */
11        function f3(){
12
13            var n1 = 123;
14            /* js在执行里，将number类型自动
15               转换成其对应的包装类型Number
16               然后调用其对应的方法。*/
17            var n1Str = n1.toString();
18            alert(n1Str + 100);
19        }
20    </script>
```



【案例 1】基本数据类型 **

```
<html>
  <!--基本数据类型-->
  <head>
    <script>
        /* 5种基本数据类型的使用 */
        function f1(){

            //alert('hi') ;

            /* 是一种弱类型语言，即变量在声明时
               不能明确声明其类型*/
            var i = 100 ;

            /* 变量的类型是在运行
             时确定的，并且，可以随时改变*/
            i = 'abc' ;

            /* 可以使用typeof获知变量的类型*/
            //alert(typeof i) ;

            var str1 = "abc" ;
            var str2 = "abc" ;
            if(str1 == str2){
                //alert("str1与str2相等") ;
            }

            var i2 = 100 ;
            var s2 = '100' ;
            /* "=="的作用是比较值 */
            if(i2 == s2){
                //alert("i2与s2相等") ;
            }

            /* "==="的作用是先比较类型，只有类
               型一致，才比较值 */
            if(i2 === s2){
```

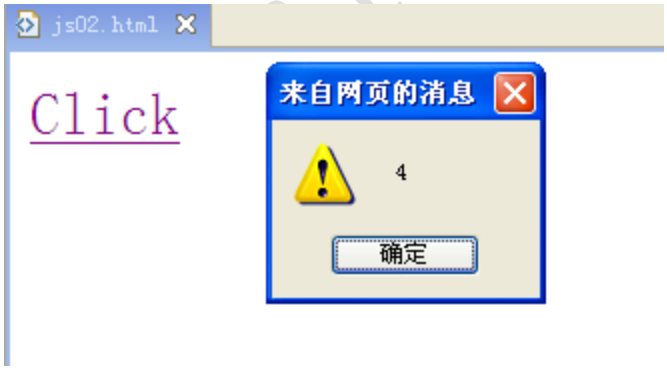
```
            //alert("i2与s2相等") ;
        }else{
            //alert("i2与s2不相等") ;
        }

        var flag = false ;
        //alert(typeof flag) ;

        var s1 = null ;
        /* null类型，在使用typeof测试时，会
           输出object类型 */
        //alert(typeof s1) ;

        var s3 ;
        /* 一个变量声明过了，但没人赋值
           类型是undefined */
        alert(typeof s3) ;
    }

    /* boolean类型使用时，要注意的问题
       a, 不为空的字符串，转换成true。
       b, 非零的数字，转换成true。
       c, null、undefined转换成false */
    function f2(){
        //var flag = 'hello' ;
        //var flag = '' ;
        //var flag = 10 ;
        //var flag = null ;
        var flag ;
        if(flag){
            alert(true) ;
        }else{
            alert(false) ;
        }
    }

    /* 数据类型转换
       string --> number
```

```
            number -->string */
        function f3(){
            //var s1 = '150' ;
            var s1 = '150.03' ;
            //var s2 = parseInt(s1) + 100 ;
            var s2 = parseFloat(s1) + 100 ;
            //alert(s2) ;

            var s3 = '123a' ;
            var n3 = parseInt(s3) ;
            //alert(n3 + 100) ;

            /* isNaN()：如果一个值是非数字，返回true */
            var s4 = '123a' ;
            if(isNaN(s4)){
                //alert('非数字') ;
            }else{
                //alert('是一个数字') ;
            }

            var n1 = 123 ;
            /* js在执行里，将number类型自动
               转换成其对应的包装类型Number
               然后调用其对应的方法。*/
            var n1Str = n1.toString() ;
            alert(n1Str + 100) ;
        }
    </script>
</head>

<body style="font-size :30px ;">

    <a href="" onclick="f3() ;">Click me</a>

</body>

</html>
```

2.3.2. 字符串的常用属性和方法 **

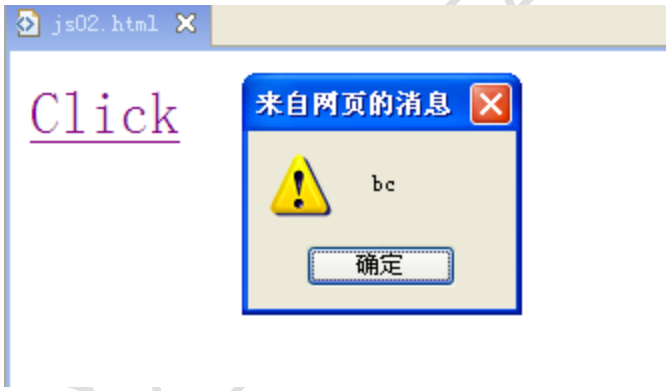
- **演示 1**

```
<html>
  <!--字符串的常用属性和方法
  (实际上，调用的是string类型
  对应的包装类String的属性和方法)-->
  <head>
    <script>
        function f1(){
            var str = "abcd";
            alert(str.length);
            var str1 = str.substring(1,3);
            //alert(str1);
        }
    </script>
  </head>

  <body style="font-size:30px;">
    <a href="" onclick="f1();">Click</a>
  </body>

</html>
```



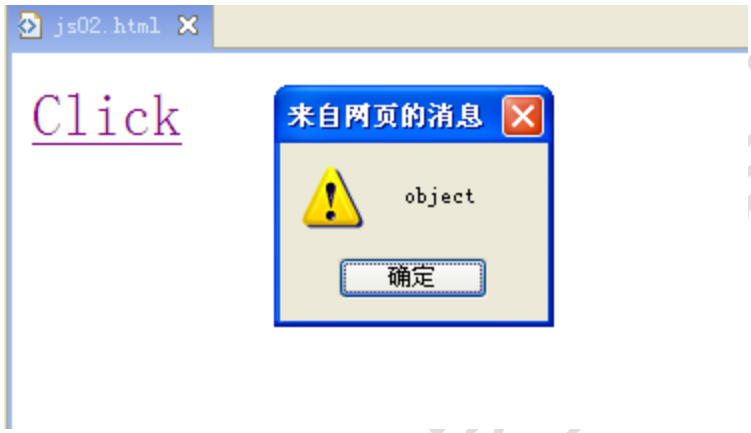
- **演示 2**

```html
<html>
  <!--字符串的常用属性和方法
  (实际上，调用的是string类型
  对应的包装类String的属性和方法)-->
  <head>
    <script>
        function f1(){
            var str = "abcd";
            //alert(str.length);
            var str1 = str.substring(1,3);
            alert(str1);
        }
    </script>
  </head>

  <body style="font-size:30px;">
    <a href="" onclick="f1();">Click</a>
  </body>

</html>
```



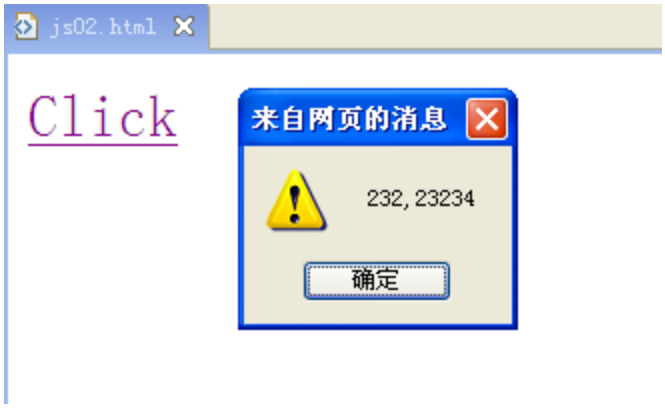
- **演示 3**

```
 8          /* 字符串中，关于正则表达式的几个方法 */
 9          function f2(){
10              var str = "asdf232asfasf23234";
11              /* js在执行时，会将/[0-9]+/转换
12                 成一个RegExp对象。"g"表示搜索
13                 整个字符串。"i"表示忽略大小写。*/
14              var reg = /[0-9]+/g;
15              alert(typeof reg);
16
```



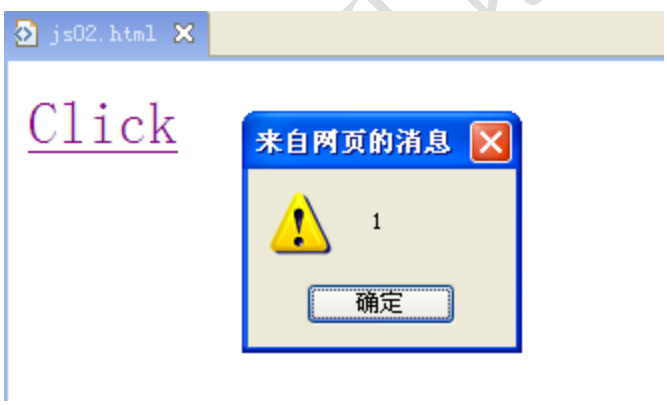
- **演示 4**

```
 8          /* 字符串中，关于正则表达式的几个方法 */
 9          function f2(){
10              var str = "asdf232asfasf23234";
11              /* js在执行时，会将/[0-9]+/转换
12                 成一个RegExp对象。"g"表示搜索
13                 整个字符串。"i"表示忽略大小写。*/
14              var reg = /[0-9]+/g;
15              //alert(typeof reg);
16
17              var arr = str.match(reg);
18              alert(arr);
```



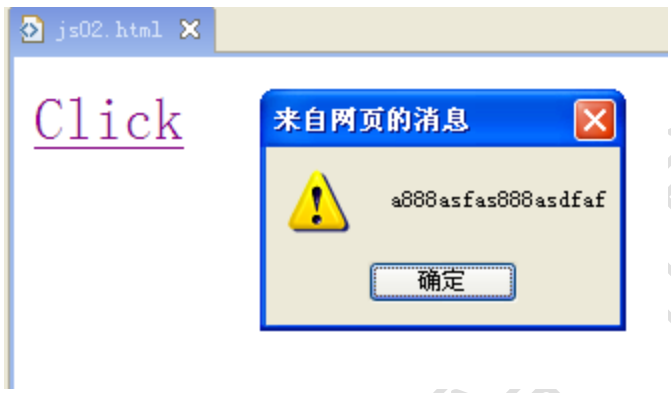
- **演示 5**

```
8          /* 字符串中，关于正则表达式的几个方法 */
9          function f2(){
10
11             var str2 = "a12234asfas23asdfaf";
12             var reg2 = /[0-9]+/;
13             var index = str2.search(reg2);
14             alert(index);
15
```



- **演示 6**

```
8          /* 字符串中，关于正则表达式的几个方法 */
9          function f2(){
10
11             var str3 = "a1444asfas44asdfaf";
12             var reg3 = /[0-9]+/g;
13             var str4 = str3.replace(reg3,'888');
14             alert(str4);
15
16         }
17     </script>
```



【案例 2】字符串的常用属性和方法 **

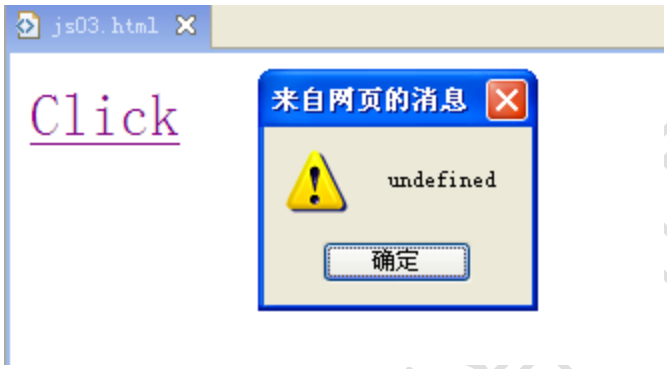
```
<html>
  <!--字符串的常用属性和方法
  (实际上，调用的是string类型
  对应的包装类String的属性和方法)-->
  <head>
    <script>
        function f1(){
            var str = "abcd" ;
            //alert(str.length) ;
            var str1 = str.substring(1,3) ;
            alert(str1) ;
        }

        /* 字符串中，关于正则表达式的几个方法 */
        function f2(){
```

```
            var str = "asdf232asfasf23234" ;
            /* js在执行时，会将/[0-9]+/转换
                成一个RegExp对象。"g"表示搜索
                整个字符串。"i"表示忽略大小写。*/
            var reg = /[0-9]+/g ;
            //alert(typeof reg) ;
            var arr = str.match(reg) ;
            //alert(arr) ;

            var str2 = "a12234asfas23asdfaf" ;
            var reg2 = /[0-9]+/ ;
            var index = str2.search(reg2) ;
            //alert(index) ;

            var str3 = "a1444asfas44asdfaf" ;
            var reg3 = /[0-9]+/g ;
            var str4 = str3.replace(reg3,'888') ;
            //alert(str4) ;

        }
    </script>
</head>

<body style="font-size :30px ;">
    <a href="" onclick="f2() ;">Click</a>
</body>

</html>
```

2.3.3. 数组 **

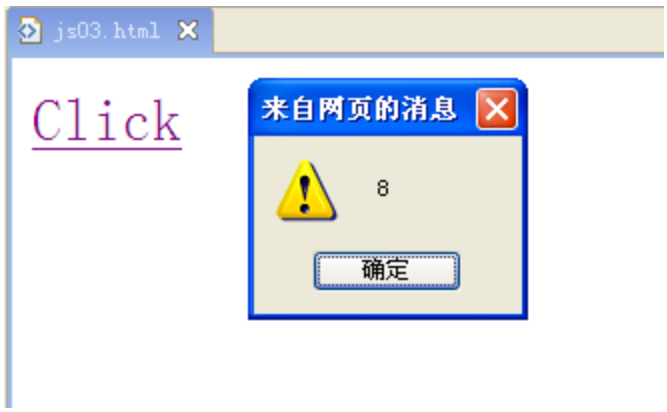
- **演示 1**

```
<script>
    /* 创建数组 */
    function f1(){
        /* 创建数组的第一种方式
           使用Array。*/
        var arr1 = new Array();
        arr1[0] = 1;
        arr1[7] = 12;
        alert(arr1[2]);
        //alert(arr1.length);
```



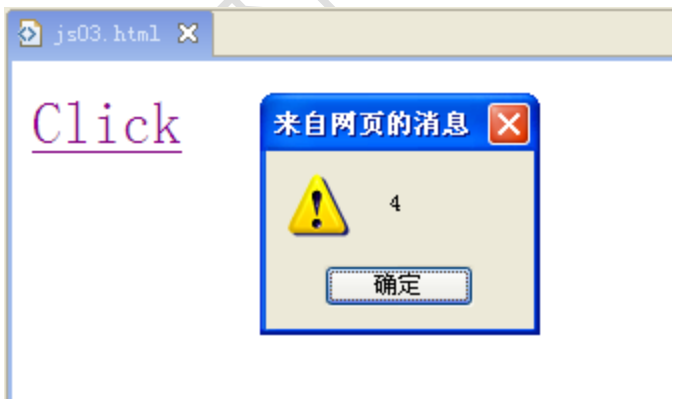
- **演示 2**

```
<script>
    /* 创建数组 */
    function f1(){
        /* 创建数组的第一种方式
           使用Array。*/
        var arr1 = new Array();
        arr1[0] = 1;
        arr1[7] = 12;
        //alert(arr1[2]);
        alert(arr1.length);
```



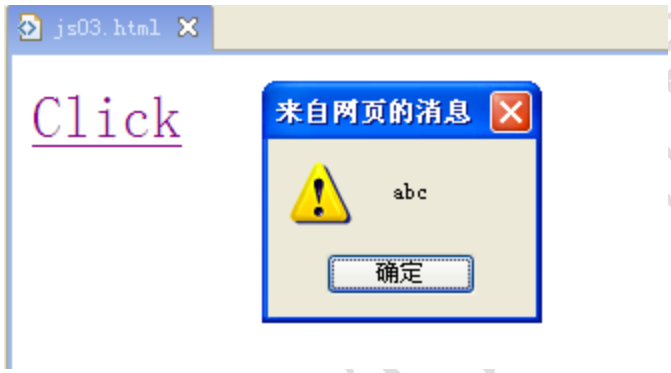
- **演示 3**

```
4    <script>
5        /* 创建数组 */
6        function f1(){
7            /* 创建数组的第二种方式
8             * 使用[]。
9             */
10           var arr2 = [];
11           arr2[3] = 12;
12           alert(arr2.length);
13           var arr3 = [1,2,'abc'];
14           arr3[4] = 12;
15           //alert(arr3[2]);
16       }
```



- **演示 4**

```
<script>
    /* 创建数组 */
    function f1(){
        /* 创建数组的第二种方式
         * 使用[]。
         */
        var arr2 = [];
        arr2[3] = 12;
        //alert(arr2.length);
        var arr3 = [1,2,'abc'];
        arr3[4] = 12;
        alert(arr3[2]);
    }
```

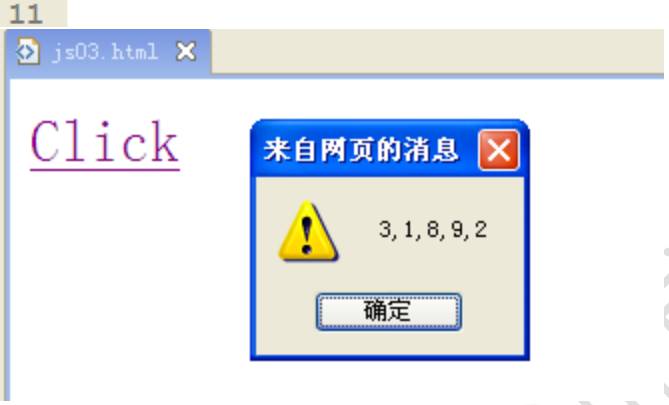


- **演示 5**

```
<script>

    /* 二维数组 */
    function f2(){
        var arr1 = new Array;
        arr1[0] = [3,1,11];
        arr1[1] = [18,22];
        arr1[2] = [19];
        for(i=0;i<arr1.length;i++)
            for(j=0;j<arr1[i].length;j++){
                alert(arr1[i][j]);
            }
    }
```

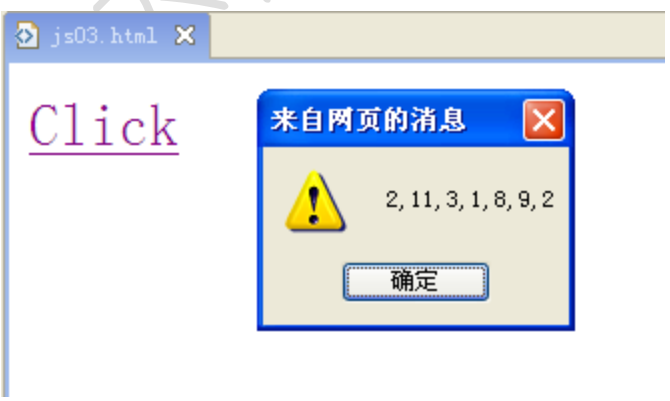
- **演示 6**

```
4    <script>
5
6        /* 数组的常用方法 */
7        function f3(){
8            var arr1 = [3,1,8,9,2];
9            var str = arr1.toString();
10           alert(arr1);
11
```



- **演示 7**

```
6        /* 数组的常用方法 */
7        function f3(){
8            var arr1 = [3,1,8,9,2];
9            var str = arr1.toString();
10           //alert(arr1);
11
12           var arr2 = [2,11];
13           var arr3 = arr2.concat(arr1);
14           alert(arr3);
15
```

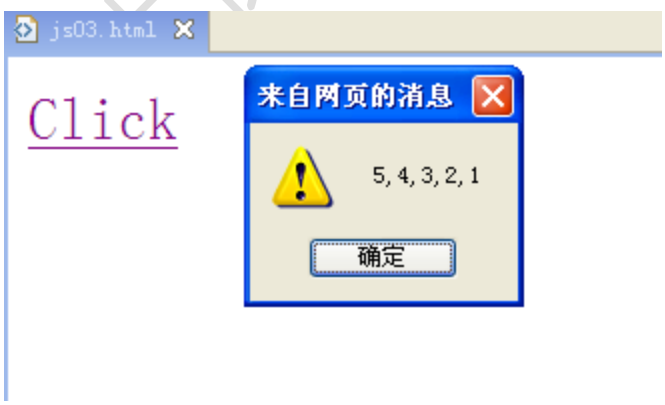


- **演示 8**

```
6          /* 数组的常用方法 */
7          function f3(){
8              var arr4 = ["hello","haha","888"];
9              var str4 = arr4.join('|');
10             alert(str4);
11
```

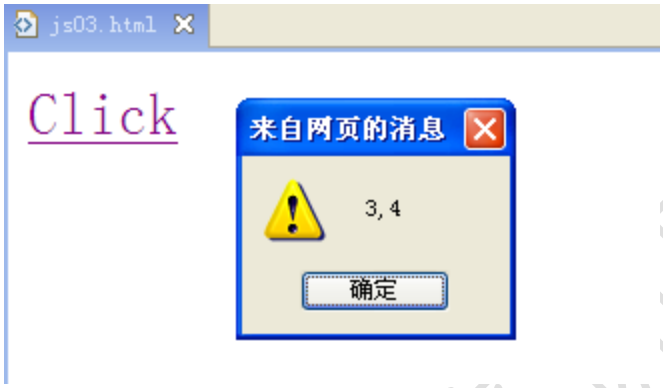
- **演示 9**

```
6          /* 数组的常用方法 */
7          function f3(){
8
9              var arr5 = [1,2,3,4,5];
10             arr5.reverse();
11             alert(arr5);
12
```



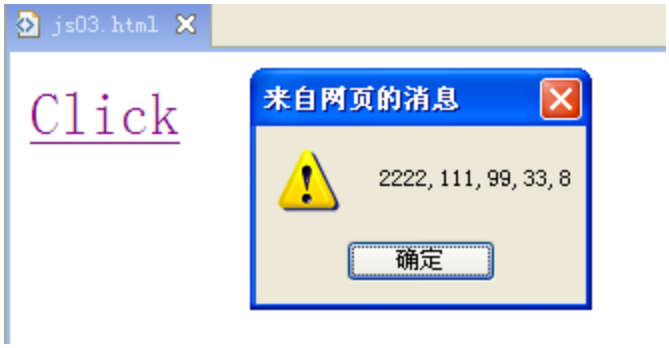
- **演示 10**

```
<script>

    /* 数组的常用方法 */
    function f3(){

        var arr6 = [1,2,3,4,5];
        var arr7 = arr6.slice(2,4);
        alert(arr7);

```



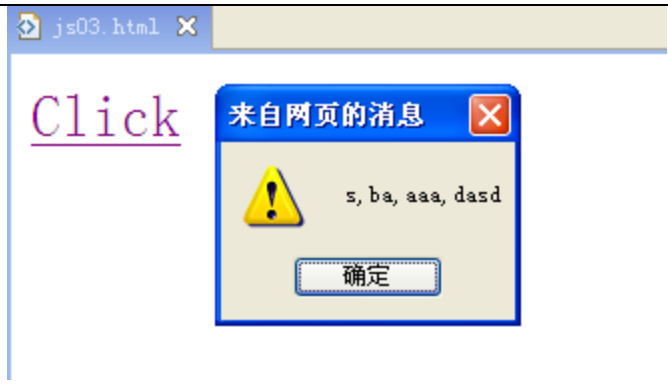
- **演示 11**

```
    /* 数组的常用方法 */
    function f3(){

        //var arr8 = [3,1,8,9,2];
        var arr8 = [33,111,8,99,2222];
        /* sort()方法默认情况下，按照
           字典顺序来排序。*/
        /* javascript当中，函数其实就是
           一个对象，对象是可以作为参数
           进行传递的。*/
        arr8.sort(function(t1,t2){
            return t2 - t1;
        });
        alert(arr8);
```



- **演示 12**

```
<html>
  <!--数组-->
  <head>
    <script>

        /* 数组的常用方法 */
        function f3(){

            var arr9 = ['ba','aaa','s','dasd'];
            /* 按字符串长度从小到大排序 */
            arr9.sort(function(a1,a2){
                return a1.length - a2.length;
            });
            alert(arr9);
        }
    </script>
  </head>
  <body style="font-size:30px;">
    <a href="" onclick="f3();">Click</a>
  </body>
</html>
```



【案例 3】数组 **

```
<html>
  <!--数组-->
  <head>
    <script>
        /* 创建数组 */
        function f1(){
            /* 创建数组的第一种方式
               使用Array。*/
            var arr1 = new Array() ;
            arr1[0] = 1 ;
            arr1[7] = 12 ;
            //alert(arr1[2]) ;
            //alert(arr1.length) ;

            /* 创建数组的第二种方式
             * 使用[]。
             */
            var arr2 = [] ;
            arr2[3] = 12 ;
            //alert(arr2.length) ;
            var arr3 = [1,2,'abc'] ;
            arr3[4] = 12 ;
            alert(arr3[2]) ;
        }
```

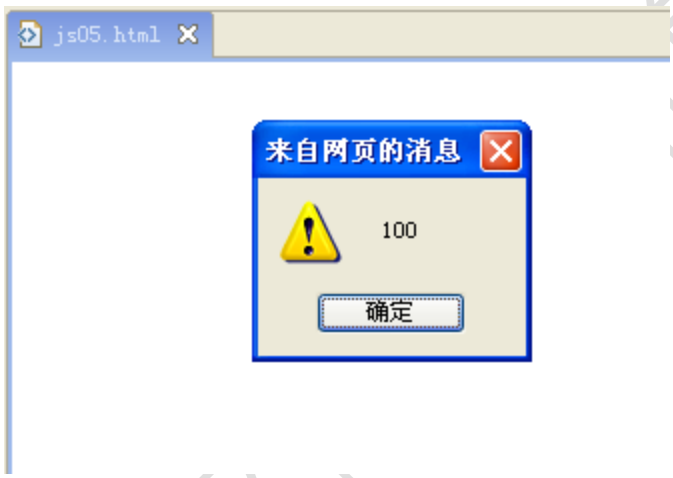
```
/* 二维数组 */
function f2(){
    var arr1 = new Array ;
    arr1[0] = [3,1,11] ;
    arr1[1] = [18,22] ;
    arr1[2] = [19] ;
    for(i=0 ;i<arr1.length ;i++)
        for(j=0 ;j<arr1[i].length ;j++){
            alert(arr1[i][j]) ;
        }
}

/* 数组的常用方法 */
function f3(){
    var arr1 = [3,1,8,9,2] ;
    var str = arr1.toString() ;
    //alert(arr1) ;

    var arr2 = [2,11] ;
    var arr3 = arr2.concat(arr1) ;
    //alert(arr2) ;

    var arr4 = ["hello","haha","888"] ;
    var str4 = arr4.join('|') ;
    //alert(str4) ;

    var arr5 = [1,2,3,4,5] ;
    arr5.reverse() ;
    //alert(arr5) ;

    var arr6 = [1,2,3,4,5] ;
    var arr7 = arr6.slice(2,4) ;
    //alert(arr7) ;

    //var arr8 = [3,1,8,9,2] ;
    var arr8 = [33,111,8,99,2222] ;
    /* sort()方法默认情况下，按照
       字典顺序来排序。*/
```

```
                /* javascript当中，函数其实就是
                    一个对象，对象是可以作为参数
                    进行传递的。*/
                arr8.sort(function(t1,t2){
                    return t2 - t1 ;
                }) ;
                //alert(arr8) ;

                var arr9 = ['ba','aaa','s','dasd'] ;
                /* 按字符串长度从小到大排序 */
                arr9.sort(function(a1,a2){
                    return a1.length - a2.length ;
                }) ;
                alert(arr9) ;
            }
        </script>
    </head>
    <body style="font-size :30px ;">
        <a href="" onclick="f3() ;">Click</a>
    </body>
</html>
```

2.3.4. 函数

- **演示 1　全局变量和局部变量**

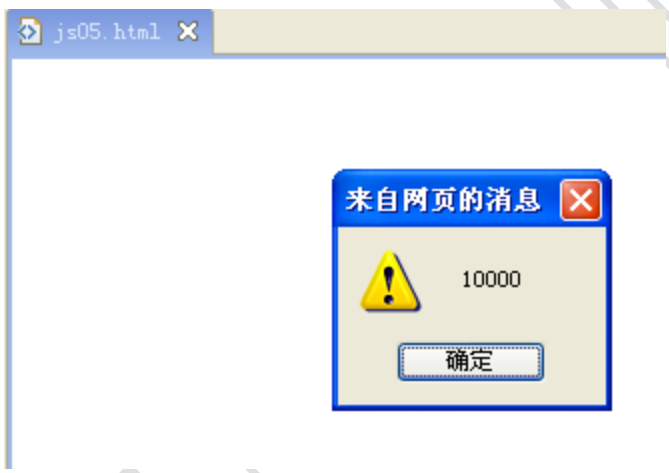
```
<html>
    <head>
        <script>
            var i = 100;
            function f1(){
                alert(i);
            }
            f1();
        </script>
    </head>
    <body style="font-size:30px;">
    </body>
</html>
```



- **演示 2　全局变量和局部变量**

```html
<html>
    <head>
        <script>
            var i = 100;
            function f1(){
                i = 10000;
                alert(i);
            }
            f1();
        </script>
    </head>
    <body style="font-size:30px;">
    </body>
</html>
```

加载页面时执行



【案例 4】函数 *

```html
<html>
    <!--函数-->
    <head>
        <script>
        /* 传参 */
        function f1(info){
```

```
            alert(info) ;
        }

        /* 在一个函数内部，可以通过arguments
           对象来访问参数值。*/
        function add(a1,a2){
            return arguments[0] + arguments[1] ;
        }

        /* 没有函数的重载，如果有两个函数重名，
           则后一个会覆盖前一个。*/
        function add(a1){
            return arguments[0] + arguments[1] + 1 ;
        }

        function f2(){
            var i = add(1,2) ;
            alert(i) ;
        }

        function f3(){
            alert('hello') ;
        }

        f3() ;  //会执行f3中内容

        </script>
    </head>

    <body style="font-size :30px ;">
        <input type="button"
        value="函数的基本使用"
        onclick="f1('hello') ;"/>
        <br/>
        <input type="button"
        value="函数的基本使用"
        onclick="f2() ;"/>
    </body>
```

```
</html>
```

3. DOM ***

3.1. 什么是 DOM? *

Document Object Model(文档对象模型),定义了如何将一个结构化的文档(比如 xml, html)转换成一棵树，并且也定义了如何操作这棵树的方法或者属性。这样做的目的，是为了方便对结构化文档的操作。

3.2. w3c dom 模型 ***

w3c 定义了 dom 模型，用来将 html 文档转换成内存当中的一棵树，并且也定义了相应的操作树的属性和方法。

1) **树的结构（了解）**

Node
- Document
  - HTMLDocument
    - HTMLBodyElement
- Element
  - HTMLElement
    - HTMLFormElement
      - HTMLInputElement
      - HTMLSelectElement
        - HTMLOptionElement
    - HTMLDivElement
    - HTMLTableElement
      - HTMLTableCaptionElement
      - HTMLTableRowElement
      - HTMLTableCellElement

2) **dom 操作（重点）**

a. 查找方式（常用方式）

```
//document 是 HTMLDocument 的实例。
var obj = document.getElementById(id)；
```

b. 创建节点

c. 添加节点

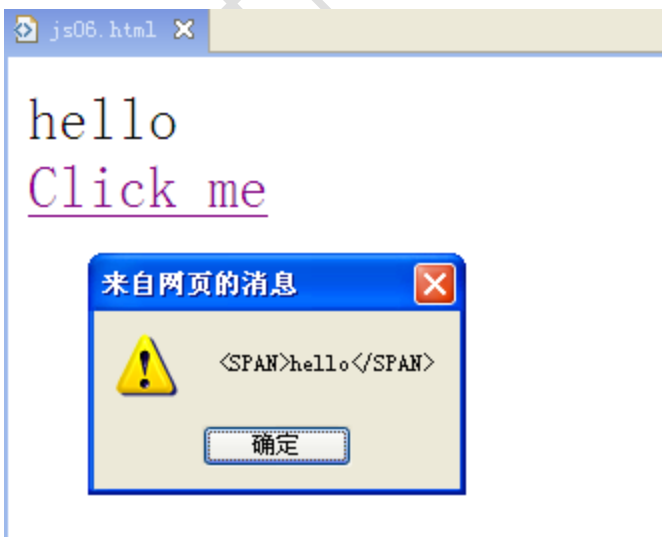
d. 删除节点

e. 样式操作

- **演示 1**

Dom 查找的第一种方式：根据 id 查找

```
<html>
  <!--dom操作-->
  <head>
    <script>
        function f1(){
            var obj = document.getElementById('d1');
            /* innerHTML属性：可以读或者写一个
               节点的html内容。 */
            alert(obj.innerHTML);
        }

    </script>
  </head>
  <body style="font-size:30px;">
    <div id="d1">
        <span>hello</span>
    </div>
    <a href="" onclick="f1();">Click me</a>
</body>
</html>
```



- **演示 2 ***

改变页面显示内容

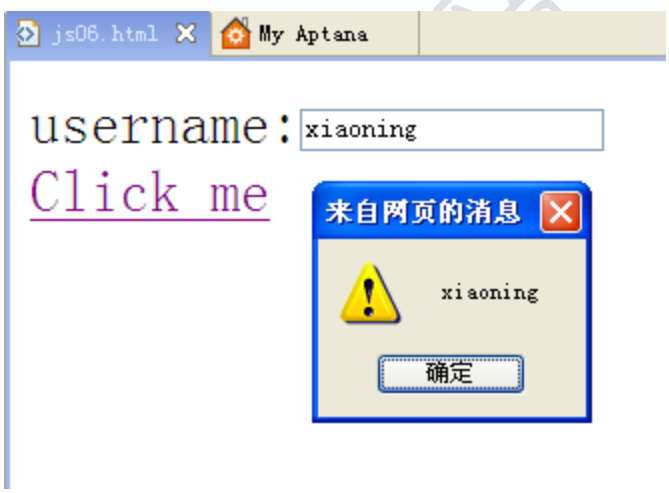
```
<html>
  <!--dom操作-->
  <head>
    <script>
        function f1(){
            var obj = document.getElementById('d1');
            /* innerHTML属性：可以读或者写一个
               节点的html内容。 */
            //alert(obj.innerHTML);
            obj.innerHTML = 'hello kitty';
        }

    </script>
  </head>
  <body style="font-size:30px;">
    <div id="d1">
        <span>hello</span>
    </div>
    <a href="javascript:;" onclick="f1();">Click me</a>
</body>
</html>
```

✓ ” Javascript:; “ 表示一个空白的 JS 语句

- **演示 3**

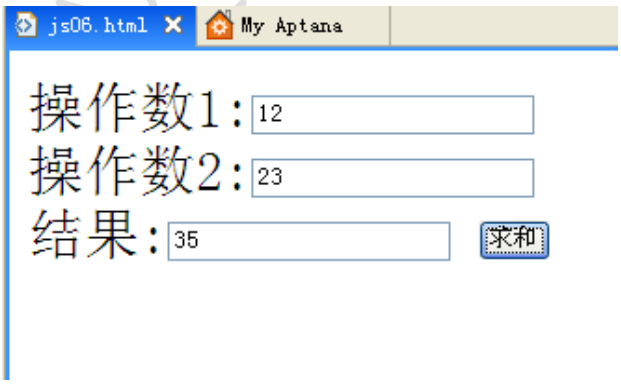
获得用户文本框输入的数据

```
<html>
  <!--dom操作-->
  <head>
    <script>
        function f1(){
            var obj2 =
                document.getElementById('username');
            //value属性：可以读或者写一个节点的value值。
            alert(obj2.value);
        }

    </script>
  </head>
  <body style="font-size:30px;">

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

    <a href="javascript:;" onclick="f1();">
    Click me</a>
</body>
```



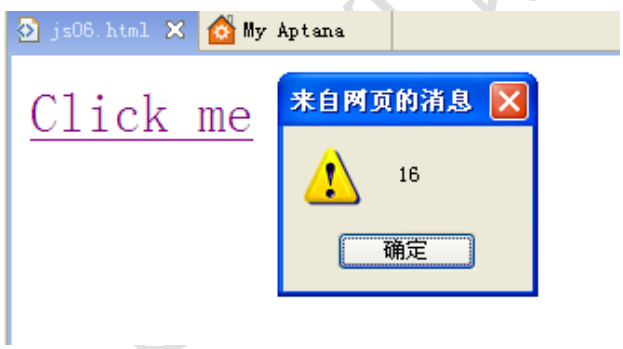
- **演示 4**

```html
<html>
  <!--dom操作-->
  <head>
    <script>
        function sum(){
            var txtObj1 =
            document.getElementById('op1');
            var txtObj2 =
            document.getElementById('op2');
            var v1 = parseInt(txtObj1.value);
            var v2 = parseInt(txtObj2.value);
            var v3 = v1 + v2;
            var txtObj3 =
            document.getElementById('op3');
            txtObj3.value = v3;
        }
    </script>
  </head>
  <body style="font-size:30px;">

    操作数1:<input name="op1" id="op1"/>
    <br/>
    操作数2:<input name="op2" id="op2"/>
    <br/>
    结果:<input name="op3" id="op3"/>
    <input type="button" value="求和"
    onclick="sum();"/>

</body>
</html>
```



- **演示 5**

```
<html>
  <!--dom操作-->
  <head>
    <script>
        function f2(){
                var op1 = Math.random();
                //alert(op1);
                //var op2 = Math.ceil(13.98);
                var op2 = Math.floor(13.98);
                //alert(op2);
                var op3 =
                    Math.floor(Math.random()*34);
                alert(op3);
        }
    </script>
  </head>
  <body style="font-size:30px;">

    <a href="javascript:;"
    onclick="f2();">Click me</a>
  </body>
</html>
```



【案例 5】dom 操作 ***

```
<html>
  <!--dom操作-->
  <head>
    <script>
        function f1(){
```

```
            var obj = document.getElementById('d1') ;
            /* innerHTML属性：可以读或者写一个
               节点的html内容。*/
            //alert(obj.innerHTML) ;
            //obj.innerHTML = 'hello kitty' ;

            var obj2 =
            document.getElementById('username') ;
            /* value属性：可以读或者写一个节点的
               value值。*/
            //alert(obj2.value) ;
            obj2.value = 'abc' ;
        }

        function sum(){
            var txtObj1 =
            document.getElementById('op1') ;
            var txtObj2 =
            document.getElementById('op2') ;
            var v1 = parseInt(txtObj1.value) ;
            var v2 = parseInt(txtObj2.value) ;
            var v3 = v1 + v2 ;
            var txtObj3 =
            document.getElementById('op3') ;
            txtObj3.value = v3 ;
        }

        function f2(){
            var op1 = Math.random() ;
            //alert(op1) ;
            //var op2 = Math.ceil(13.98) ;
            var op2 = Math.floor(13.98) ;
            //alert(op2) ;
            var op3 = Math.floor(Math.random()*34) ;
            alert(op3) ;
        }
    </script>
</head>
```

```
  <body style="font-size :30px ;">
    <div id="d1">
        <span>hello</span>
    </div>

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

    操作数1 :<input name="op1" id="op1"/>

    操作数2 :<input name="op2" id="op2"/>

    结果 :<input name="op3" id="op3"/>

    <input type="button" value="求和"
           onclick="sum() ;"/><br/>

    <a href="javascript : ;" onclick="f2() ;">Click me</a>
</body>
</html>
```