

第4章 JavaScript

课程名称：Web程序设计



window对象

~~window.history~~

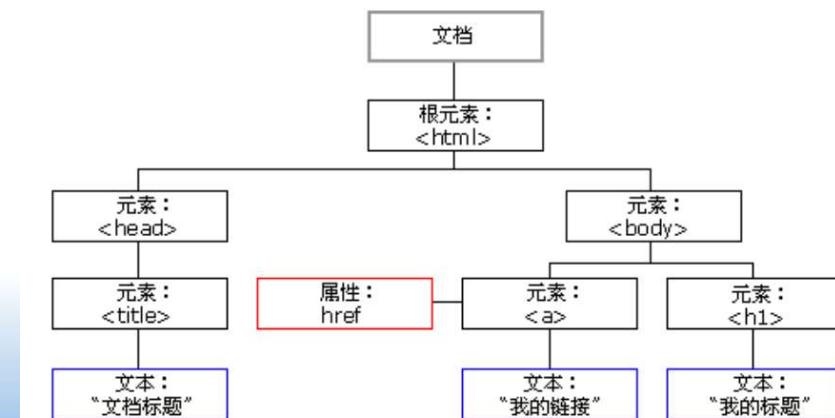


~~window.location~~



document.location

~~window.document~~



~~window.screen~~

~~window.navigator~~

1、JavaScript简介

第4章 JavaScript



1.1 JavaScript 的起源和发展

- ❖ 1994年，JavaScript 语言的前身LiveScript。
- ❖ 1995年，Netscape和Sun推出JavaScript（作者是Brendan Eich）。
- ❖ 1996年，微软发布JScript（与JavaScript兼容）。
- ❖ 1997、1998和1999年，ECMA（European Computer Manufacturers Association）分别发布了ECMAScript第1版（标准ECMA-262）、第2版和第3版。
- ❖ 1998、2005和2009年，DHTML、AJAX和HTML5。
- ❖ 2009年，Ryan Dahl 团队在浏览器之外构建独立的JavaScript运行时Node.js。
- ❖ 2015和2016年，ECMAScript 6和7。
- ❖ 2021年，ECMA-419。



Brendan Eich



1.2 JavaScript的组成

- ❖ ECMAScript
 - JavaScript的语法和基本对象
- ❖ 浏览器对象模型BOM（Browser Object Model）
 - 与浏览器交互的方法和接口
- ❖ 文档对象模型DOM（Document Object Model）
 - 处理网页内容的方法和接口





1.3 JavaScript的特点

- ❖ 解释型脚本语言：Web浏览器中都嵌入了JS引擎，服务器端或搭载了JS引擎的设备都能执行JS代码。
- ❖ 面向对象的语言：预定义和自定义对象，语法与Java类似。
- ❖ 弱类型、动态语言：变量初始化为任意类型，并随时改变其类型。
- ❖ 事件驱动的脚本语言：不需要Web服务器的参与就能对用户的输入做出响应。
- ❖ 跨平台：不依赖于操作系统，可在浏览器中运行。





1.4 JavaScript的作用

- ❖ Web 应用开发：与HTML和CSS一起
 - 操纵HTML元素（改变元素的属性和内容）
 - 动态添加网页内容（元素、属性和文本节点）
 - 校验表单中用户输入的内容
 - 对用户的事件进行响应
 - 为网页增加特效（如动态的菜单、浮动的广告等）
- ❖ 移动应用开发：开发手机或平板电脑上的应用程序，如借助于React Native等框架。
- ❖ Web 游戏：网页中的小游戏
- ❖ 后端 Web 应用开发：JavaScript 运行环境Node.JS



1.5 网页中插入JavaScript脚本的方法

1、行内式

- JavaScript脚本嵌入到HTML标签的事件响应中
- 格式： onEventName = "JavaScript Code"

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title> 行内式插入脚本 </title>
6  </head>
7  <body>
8      <p onClick = "alert('Hello, the WEB world!');">
9          Click Here!
10     </p>
11  </body>
12  </html>
```



行内式插入脚本

文件 | D:/phpstudy_pro/WWW/learn... 星级 用户 :

Sublime Bootstrap中文网 Swiper中文网-轮播... 阅读清单

Click Here!

行内式插入脚本

文件 | D:/phpstudy_pro/WWW/learn... 星级 用户 :

此网页显示

Hello, the WEB world!

确定



1.5 网页中插入JavaScript脚本的方法

2、嵌入式

- 使用<script>标签将JavaScript脚本嵌入HTML文档中
- 格式： <script> JavaScript Code </script>

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title> 嵌入式插入脚本 </title>
6      <script>
7          function msg () { //单行注释：建立函数msg()
8              /*多行注释：函数体，弹出消息警告框。*/
9              alert("Hello, the WEB world!");
10         }
11     </script>
12 </head>
13 <body>
14     <p onClick = "msg()">Click Here!</p> <!-- 调用函数 -->
15 </body>
16 </html>
```



1.5 网页中插入JavaScript脚本的方法

3、链接式

- 通过<script>标记的src属性链接外部JavaScript脚本文件
- 格式： <script src = "scriptFileName">

JavaScript Code

```
</script>
```



3、链接式

- 通过<script>标记的src属性链接外部脚本文件
- 格式： <script src = "scriptFileName">

~~JavaScript Code~~

</script>

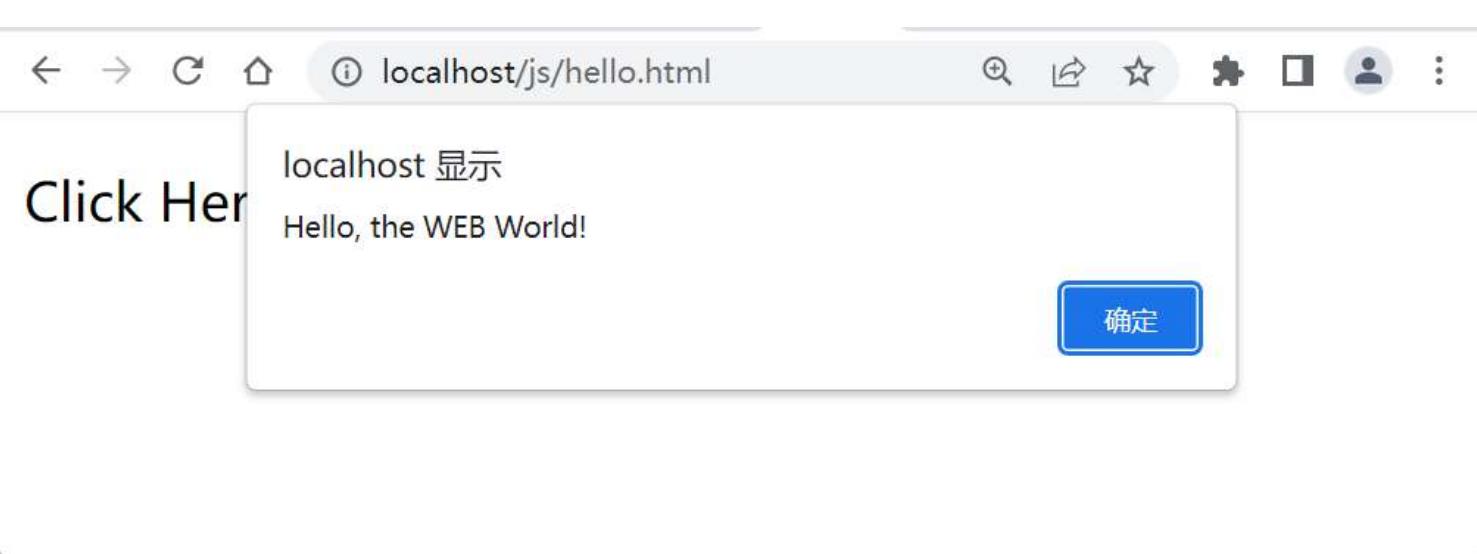
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title> 链接式插入脚本文件 </title>
6      <script src="just4Slides-3.js"></script>
7  </head>
8  <body>
9      <p onClick = "alert('Hello, the WEB world!');">
10         Click Here!
11     </p>
12  </body>
13 </html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>链接式插入脚本文件</title>
    <script src="hello.js"></script>
</head>
<body>
    <p id="clickP">Click Here!</p>
</body>
</html>
```

```
window.onload = function (){
    var objClickP = document.getElementById( elementId: "clickP");
    objClickP.addEventListener( type: "click", msg);
}

function msg(){
    alert("Hello, the WEB World!");
}
```





1.6 JavaScript代码调试工具

- ❖ Firefox: 【工具】→【浏览器工具】→【Web 开发者工具】
- ❖ Chrome: 【右键】→【检查】
- ❖ 页面按F12, 在【控制台】
 - 查看返回消息
 - 输入命令



2、JavaScript基础语法

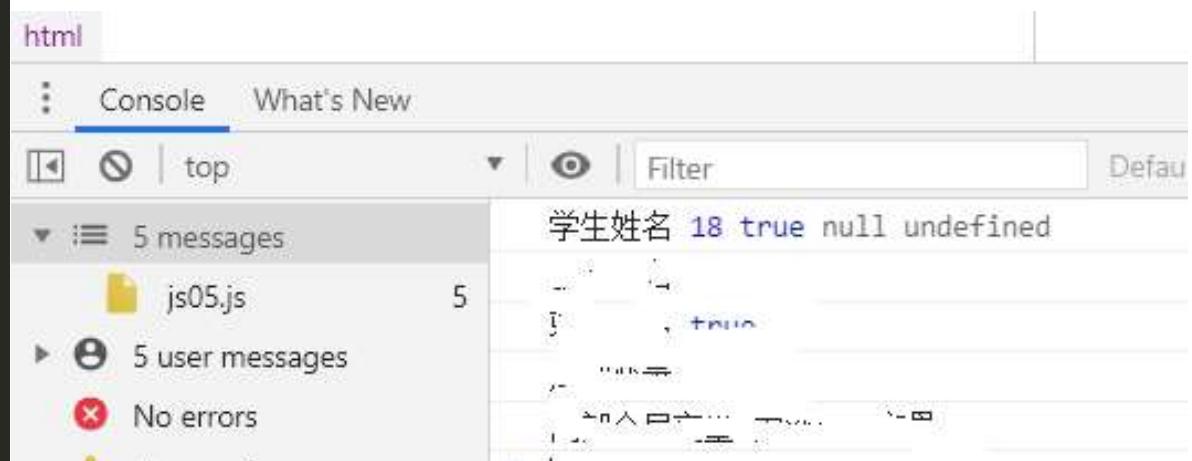
第4章 JavaScript



2.1 常量

- ❖ ECMAScript 6中， 使用关键字**const**定义常量。
- ❖ 类型有String、Number、Boolean、Null和Undefined。

```
1 //定义一个保存姓名的字符常量
2 const NAME = '学生姓名';
3 //定义一个数值型的常量
4 const AGE = 18;
5 //定义一个布尔型的常量
6 const MARRY = true;
7 //定义一个null型的常量
8 const WORK = null;
9 //定一个undefined型的变量
10 const SCORE = undefined;
```



```
12 //在控制台输出
13 console.log(NAME, AGE, MARRY, WORK, SCORE);
```



2.2 变量

- ❖ 用 var 关键字声明变量，并初始化为任何值。
 - `var name = "王晴明";`
 - `var age = 28;`
 - `var male = true;`
- ❖ 变量类型由赋值得类型决定



❖ 变量的命名原则

- 首字符必须是字母、下划线_或者美元符\$
- 其他字符可以是字母、数字、下划线或美元符
- 不能用关键字和保留字
- **大小写敏感**
- 不能有空格、回车符或其他标点字符



JavaScript关键字和保留字

Reserved Words in JavaScript

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	



2.3 作用域

❖ 全局作用域

- <script>标签和文件内
- 页面打开时创建、关闭时销毁
- 全局对象window代表一个浏览器窗口，由浏览器创建，可以直接使用。
- 变量和函数分别作为window对象的属性和方法保存
- 该作用域中定义的变量都是全局变量，页面任意部分都可以访问到。



❖ 函数作用域

- 调用函数时创建、执行完毕后销毁
- 每次调用创建新的函数作用域，相互之间独立。
- 函数内部定义的变量的作用域为该函数
- 函数内部不使用关键字var，可以定义全局变量。



```
34 //先定义一个名为strGlobal_1的全局变量  
35 strGlobal_1 = '全局变量';  
36 //再在函数中定义一个名为strInnerLocal的局部变量  
37 function retInnerLocal(){  
38     //定义局部变量  
39     var strInnerLocal = '局部变量';  
40     //返回局部变量  
41     return strInnerLocal;  
42 }  
43 strGlobal_1 = retInnerLocal();
```

```
45 console.log(strGlobal_1, strInnerLocal);
```

✖ error
⚠ No warnings

✖ Uncaught ReferenceError: strInnerLocal is not defined
at js05.js:44



```
34 //先定义一个名为strGlobal_1的全局变量  
35 strGlobal_1 = '全局变量';  
36 //再在函数中定义一个名为strInnerLocal的局部变量  
37 function retInnerLocal(){  
38     //定义局部变量  
39     var strInnerLocal = '局部变量';  
40     //返回局部变量  
41     return strInnerLocal;  
42 }  
43 strGlobal_1 = retInnerLocal();
```

```
47 console.log(strGlobal_1);
```



No errors

局部变量



```
49 //先定义一个名为strGlobal_2的全局变量
50 strGlobal_2 = '全局变量';
51 //再在函数中定义一个名为strInnerGlobal的全局变量
52 function retInnerGlobal(){
53     //内部定义全局变量
54     strInnerGlobal = '内部全局变量';
55     //返回局部变量
56     return strInnerGlobal;
57 }
58 strGlobal_2 = retInnerGlobal();
```

```
61 console.log(strGlobal_2, strInnerGlobal);
```



No warnings

内部全局变量 内部全局变量



```
var str = "全局变量str";
console.log(str + "-1");

window.onload = function (){
    console.log(str);
    str = "全局变量str被window.onload修改了";
    console.log(str);
}

console.log(str + "-2");
```

全局变量str-1	hello.js:5
全局变量str-2	hello.js:12
全局变量str	hello.js:8
全局变量str被window.onload修改了	hello.js:10



2.4 运算符

❖ 算术运算符

- +、 -、 *、 /、 %、 ++、 --

❖ 比较运算符

- >、 <、 >=、 <=、 ==、 ===、 !=、 !==
- ==等于， 值相等时为**true**。
- ===全等， 值和类型都相等时为**true**。

❖ 逻辑运算符

- &&、 ||、 !



❖ 赋值运算符

- =、 +=、 -=、 *=、 /=、 %=、 &=、 |=、 ^=

❖ 连接运算符

- +

❖ 条件（选择）运算符

- condExp ? exp1 : exp2



2.5 数据类型

- ❖ JavaScript属于弱数据类型定义方式
 - 变量定义时不会声明类型，赋值或使用时由具体的值来决定真正的类型。
- ❖ String、Number、Boolean、Null、Undefined和Object，共6类。



2.5.1 String

- ❖ 由Unicode字符、数字、标点符号等组成，**单引号或双引号**包裹。
- ❖ length属性：字符串长度
- ❖ toUpperCase()、toLowerCase()
- ❖ charAt(index)方法：返回index位置的字符。
- ❖ indexOf(strFind, [startPos])方法：从startPos位置开始查找和定位子串strFind，返回检索位置或-1。
- ❖ search(regExp/strFind)：可以使用正则匹配和字符查询两种方式，返回第一个匹配内容的位置。



- ❖ `substring(start[, end])`方法：返回start到end位置之间的字符串。
- ❖ `replace(strOld/RegExp, strNew)`：查找后使用第二个参数的值进行替换。
- ❖ `split(separator[, maxLen])`方法：根据符号separator将字符串分割成一个数组返回，maxLen为数组的最大长度。
 - Array类型的`join(separator)`方法



❖ 转义字符

- \r 回车
- \n 换行
- \' 单引号
- \t 制表符
- \" 双引号
- \b 空格
- \\ 反斜杠



2.5.2 Number

- ❖ 以浮点形式存储，不区分整数和浮点数。
 - 无法保证精度， $0.1 + 0.2 == 0.3$ 的结果？
 - `var intAge = 28;`
 - `var fltX = 23.45;`
 - `var intBig = 50e5;`
 - `var intSml = 50e-5;`
- ❖ $+1.7976931348623157e+308$ 到 $-1.7976931348623157e+308$ ，超出返回`-Infinity`或`Infinity`。0以上的最小值为`5e-324`。
- ❖ `NAN`表示非法数字（Not A Number）
- ❖ `0b`、`0`和`0x`开头分别表示二、十和十六进制。



2.5.3 Boolean

- ❖ 取值只有两个： true和false。
- ❖ 布尔型数据不能用引号引起起来， 带引号的是字符串。
 - var married = false;



2.5.4 Undefined和Null

- ❖ Undefined类型只有一个值: **undefined**
 - 变量声明未初始化时，为**undefined**。
- ❖ Null类型只有一个值: **null**
 - 不存在的对象的占位符，对象空指针。
- ❖ **null == undefined**， 返回 **true**
- ❖ **null === undefined**， 返回 **false**



2.5.5 Object

- ❖ 引用数据类型
- ❖ 除String、Number、Boolean、Null和Undefined外，其他值都是Object类型。
- ❖ 以花括号界定，以attrName: value的形式定义属性，各属性之间使用逗号隔开。
- ❖ 两个要素都是全局作用域
 - 属性：描述对象特性的一组数据。
 - 方法：操作对象特性的若干动作。



❖ 创建对象有两种方式

new

```
var objInfo = new Object();
objInfo.code = 21210000;
objInfo.data = new Object();
objInfo.data.name = "姓名";
objInfo.data.gender = "男";
objInfo.data.age = "18";
```

JSON格式

```
var objInfo = {
  code:"21210000",
  data:{
    name:"姓名",
    gender:"男",
    age:18
  }
}
```



❖ 访问对象的属性

- objName.attrName, attrName只能是字符串常量
- objName[attrName], attrName可以是字符串常量或变量

```
document.write(objInfo.data.age);
document.write(objInfo["data"]["age"]);
```

❖ 删 除 对 象 的 属性

- delete objName.attrName;

```
console.log(objInfo);
delete objInfo.code;
console.log(objInfo);
```



❖ 遍历对象的属性

- for...in循环语句

```
for(var objKey in objInfo){  
    var objVal = objInfo[objKey];  
    console.log(objKey + ":" + objVal);  
}
```



2.5.7 typeof运算符

❖ 返回变量的类型

- `typeof(null)`返回`object`
 - `null`表示空对象
- `typeof(NaN)`、`typeof(Infinity)`返回`number`
- `typeof(undefined)`返回`undefined`
 - 未初始化、未声明的变量



2.5.8 强制类型转换

❖ 转换为String

- 调用变量varName所属类型的toString()方法返回转换后结果，不影响原值；null和undefined没有该方法。
- 调用String(varName)函数，将varName转换成String类型。null和undefined分别转换为"null"和"undefined"。
- varName + ""



❖ 转换为Number

- Number(varName)
 - 空字符串、全空格字符串→0
 - 纯数字字符串→数字
 - 其他有非数字内容→NaN
 - true→1
 - false→0
 - null→0
 - undefined→NaN
- parseInt(varName)和parseFloat(varName)
 - 将字符串转换成整数和浮点数



❖ 转换为Boolean

- Boolean(varName)
 - 非0和NaN的Number → true
 - Number 中的0和NaN → false
 - 非空String → true
 - 空String "" → false
 - null → false
 - undefined → false
 - Object → true



2.6 条件语句

❖ if

```
if (condExp){  
    statement;  
}
```

❖ if...else

```
if (condExp){  
    statement1;  
} else {  
    statement2;  
}
```



❖ **if...else if**

```
if (condExp1){  
    statement1;  
}else if (condExp2){  
    statement2;  
}...  
else{  
    statement;  
}
```



❖ **switch**

```
switch (condExp) {  
    case value1:  
        statement1;  
        break;  
    case value2:  
        statement2;  
        break;  
    ...  
    case valueN:  
        statementN;  
        break;  
    [default:  
        statement;]  
}
```



2.7 循环语句

❖ do... while

```
do{  
    statement;  
}while(condExp);
```

❖ while

```
while(condExp){  
    statement;  
};
```



❖ **for**

```
for(exp1; condExp; exp1){  
    statement;  
}
```

❖ **for...in**

```
for(index in arrayName){  
    statement;  
}
```

❖ **break:** 跳出循环

❖ **continue:** 跳过循环的一次迭代



2.8 注释

- ❖ 单行注释以双斜线//开头到行末结束

```
//定义乘法函数，x为乘数、y为被乘数
```

```
//结果返回两个数的积
```

```
function a(x, y) {  
    return(x * y);  
}
```

- ❖ 多行注释以/*开头、以*/结束

```
/*函数名: addUser  
参数 name: 用户姓名  
tel: 用户电话号码
```

```
*/
```

3、对象

第4章 JavaScript



3.1 JavaScript对象概要

- ❖ 自定义对象
 - new和JSON表示
- ❖ this指针
- ❖ 内置对象
 - String
 - Array
 - Function
 - Date
 - Math
 - RegExp



3.2 this指针

- ❖ 调用函数时，解析器会向函数内部传递一个隐含的参数 this， this指向函数执行的上下文对象。根据函数调用方式的不同，this会指向不同的对象。以函数的形式调用时，this永远都是window；以方法的形式调用时，this是调用方法的对象。

```
var name = "全局变量name";
function func(){
    console.log(this.name);
}
var obj = {
    name: "对象obj",
    sayName: func
}
func();                                //全局变量name
obj.sayName();                          //对象obj
```



```
<body>
  <p id="clickP" onmouseover="this.align = 'right';"
     onmouseout="this.align = 'left';"
     onclick=doSomething()>Click Here!</p>
</body>
```

指当前元素<p>

```
function doSomething(){
  this.alert("在这里this指代window对象");
}
```

指window对象



3.3 Array

❖ 属于Object类型

❖ 创建数组

- `var arrayName = new Array(size);`
- `var arrayName = new Array(value1, value2, ...);`
- `var arrayName = [value1, value2, ...];`

❖ 访问数组元素

- `arrayName[index]`

❖ 遍历数据元素

- `for...in循环、arrayName.length`



❖ 数组的常用属性和方法

- `length`: 数组的长度。
- `unshift(newEle1, newEle2, ..., newEleN)`: 向数组开头添加新元素，返回添加元素后数组的长度。
- `push(newEle1, newEle2, ..., newEleN)`: 向数组尾部添加新元素，返回添加元素后数组的长度。
- `shift()`: 删除数组的第一个元素，返回被删除的元素。如果是空数组，返回`undefined`。
- `pop()`: 删除数组的最后一个元素，返回被删除的元素。
- `sort(function(a, b){...})`: 用比较性函数对数组中的元素进行排序（默认按Unicode顺序排序）。
- `reverse()`: 将数组中的元素反向排列。



- `forEach(function(value, index, objArr))`: 遍历数组，对数组的每个元素执行回调函数。value是当前正在遍历的元素，index是当前正在遍历的元素的索引，objArr是正在遍历的数组。

```
var stuNoArry = ["01", "02", "03"];
stuNoArry.forEach(function(value, index, objArr){
    console.log(objArr[index] + " = " + value);
});
```



3.4 JSON格式创建对象和对象数组

```
<script type="text/javascript">
    var person = {                      // 创建对象
        firstName: "Brendan",           // 创建属性
        lastName: "Eich"               // 最后一个属性后不加逗号,
    }
    alert( person.lastName );          // 访问属性
</script>
```

```
<script type="text/javascript">
    var employees = [
        {firstName: "Brendan", lastName: "Eich"},
        {firstName: "Bill", lastName: "Gates"},
        {firstName: "Steve", lastName: "Jobs"}
    ];
    alert( employees[0].lastName );
</script>
```



❖ 对象和JSON字符串之间的转换

- JSON字符串：便于在服务器和浏览器之间交换。
- 对象变量：便于在JavaScript脚本中处理。

- 对象变量obj转换成JSON字符串strJson
 - strJson = JSON.stringify(obj);

- JSON字符串strJson转换成对象变量obj
 - obj = JSON.parse(strJson);



```
<script type="text/javascript">
    var oPerson = {
        firstName:"Brendan",
        lastName:"Eich"
    };
    //将对象转换成JSON字符串
    var strPerson = JSON.stringify(oPerson);
    alert( strPerson );
    //将JSON字符串转换成对象
    var oPersonC = JSON.parse(strPerson);
    alert( oPersonC.lastName );
</script>
```

此网页显示

{"firstName":"Brendan","lastName":"Eich"}

确定

此网页显示

Eich

确定

object

▼ {firstName: 'Brendan', LastName: 'Eich'} ⓘ
 firstName: "Brendan"
 lastName: "Eich"
 ► [[Prototype]]: Object

string

{"firstName":"Brendan","lastName":"Eich"}



```
<script type="text/javascript">
    var oPerson = {
        firstName:"Brendan",
        lastName:"Eich"
    };
    //用for...in语句遍历对象属性
    var strMsg = "";
    for( x in oPerson ){
        strMsg += x + ":" + oPerson[x] + "\r\n";
    }
    alert( strMsg );
</script>
```

此网页显示

firstName:Brendan
lastName:Eich

确定



3.5 函数

❖ 函数定义

- 函数声明

```
function funcName([para1, para2,..., paraN]){\n    statement;\n    [return [expression]];\n}
```

- 函数表达式

```
var funcName = function([para1, para2,..., paraN]){\n    statement;\n    [return [expression]];\n}
```

- return;或者没有return语句时， 函数返回undefined。



❖ 函数调用

- funcName(arg1, arg2, ..., argN);
■ 调用函数时，解析器不检查实参的类型

因此，有可能接收到非法参数，需要时应对参数进行类型检查。

- 调用函数时，解析器也不检查实参的数量

如果实参多余形参，多余实参不会被赋值；如果实参少于形参，没有对应实参的形参是undefined。

- 自运行方式

```
(funcName([para1, para2, ..., paraN]){\n    statement;\n})(arg1, arg2, ..., argN);
```



❖ JavaScript中的函数也是一个对象， 使用typeof检查一个函数对象时， 会返回function。

- 创建函数对象（太费劲儿啦！不用啦！）

```
var funcName = new Function([para1,..., paraN], funcBody);
```

- valueOf()和toString(): 都能返回函数代码。

```
var func = new Function('a', 'b', 'return a + b;');
console.log(func.toString());
console.log(func.valueOf());
console.log(func(1,2));
```



❖ 内置函数

- 浏览器内核自带许多函数，在代码中直接调用即可。
- 浏览器内置函数大致分为常规、数组、日期、数学和字符串五大类别。
 - `var task = setInterval(code, milliseconds):` 在指定的周期（milliseconds毫秒）内反复执行代码块、函数或表达式（code）。返回一个数值型编号task。
 - `clearInterval(task):` 终止编号task对应函数的执行。



3.6 Date

❖ 创建日期对象

- `var objDate = new Date();`
用当前日期和时间作为初始值。
- `var objDate = new Date(strDate);`
使用日期格式的字符串strDate实例化，如"2023-04-22"或
"2023/04/22"。

```
var objDate = new Date("Apr 21 2023");
var objDate = new Date("2023-4-22");
var objDate = new Date("2023/4/23");
console.log(objDate);
```



❖ 日期对象的常用方法

- `getFullYear()`: 返回4位数字的完整年份值。
- `getMonth()`: 返回月份值，0~11，需要加1。
- `getDate()`: 返回月中的日期，1~31。
- `setFullYear(year, month, day)`: 设置年月日，年选项，月、日可选。
- `setMonth(month, day)`: 设置月日，月必选，日可选。
- `setDate(day)`: 设置日。



❖ 日期对象的常用方法

- `getHours()`: 返回0~23的数字，表示24小时制。
- `getMinutes()`: 返回0~59的数字，表示分钟。
- `getSeconds()`: 返回0~59的数字，表示秒数。
- `setHours(hour, min, sec)`: 设置时分秒，时必选，分、秒可选。
- `setMinutes(min, sec)`: 设置分秒，分必选，秒可选。
- `setSeconds(sec)`: 设置秒。
- `getTime()`: 返回自 1970 年 1 月 1 日午夜以来与指定日期的毫秒数。
- `setTime(millisec)`: 将日期设置为 1970 年 1 月 1 日之后/之前的指定毫秒数。



❖ 日期对象的常用方法

- `getDay()`: 返回一周中的某一天，0~6，0是星期日。
- `getTime()`: 返回从1970年1月1日开始到现在的毫秒数。
- `toString()`: 将日期对象转换为字符串。
- `toTimeString()`: 将日期对象的时间部分转换为字符串。
- `toDateString()` : 将日期对象的日期部分转换为字符串。
- `toLocaleTimeString()`: 根据本地时间格式，将日期对象的时间部分转换为字符串。



3.7 Math

- ❖ 内置工具类，封装了与数学运算相关的属性和方法。无需创建，直接使用。
- ❖ Math对象的常用属性和方法

- PI
- E
- abs(x)、exp(x)、pow(x, y)
- log(x)、log2(x)、log10(x)
- cos(x)、sin(x)、acos(x)
- ceil(x)、floor(x)、round(x)、trunc(x)
- sign(x)、sqrt()
- min(x, y, z, ..., n)、max(x, y, z, ..., n)
- random()

```
console.log(Math.PI);
console.log(Math.E);
console.log(Math.abs(-Math.PI));
console.log(Math.exp(2));
console.log(Math.cos(Math.PI));
```



3.8 RegExp

- ❖ 定义一些字符串的规则，来检查一个字符串是否符合规则，
获取将字符串中符合规则的内容提取出来。
- ❖ 使用typeof检查正则对象，会返回object。
- ❖ 创建正则表达式对象
 - var objRE = new RegExp(pattern, modifiers);
 - var strRE = /pattern/modifiers;

匹配模式modifiers

 - i忽略大小写
 - g全局匹配模式
 - m执行多行匹配

```
var reg = new RegExp("ab", "i");
var str = "Abc";
var result = reg.test(str);
console.log(result); //true

var reg = /a/i;
```



❖ 对字符串使用正则表达式

- String类
 - search(regExp)
 - replace(regExp, strNew)
- RegExp类
 - test(str): 搜索str, 返回true或者false。
 - exec(str): 搜索str, 返回找到的文本, 未找到返回null。



pattern元素	含义
	或者
[]	包含其中的任意内容，或关系
[^]	[^] 不包含其中的内容
{n}	前面内容出现n次。多内容时用()括起来。
{m,}	出现至少m次
{m, n}	出现m至n次
+	出现至少1次，{1,}
*	出现0次或多次，{0,}
?	出现0或1次，{0,1}
^	表示开头
\$	表示结尾



pattern元素	含义
\	转义字符
\w	任意字母、数字、_，相当于[A-zA-Z_]
\W	除了字母、数字、_，相当于[^A-zA-Z_]
\d	任意的数字，相当于[0-9]
\D	除了任意的数字，相当于[^0-9]
\s	空格
\S	除了空格
\b	单词边界
\B	除了单词边界



❖ 正则检查手机号

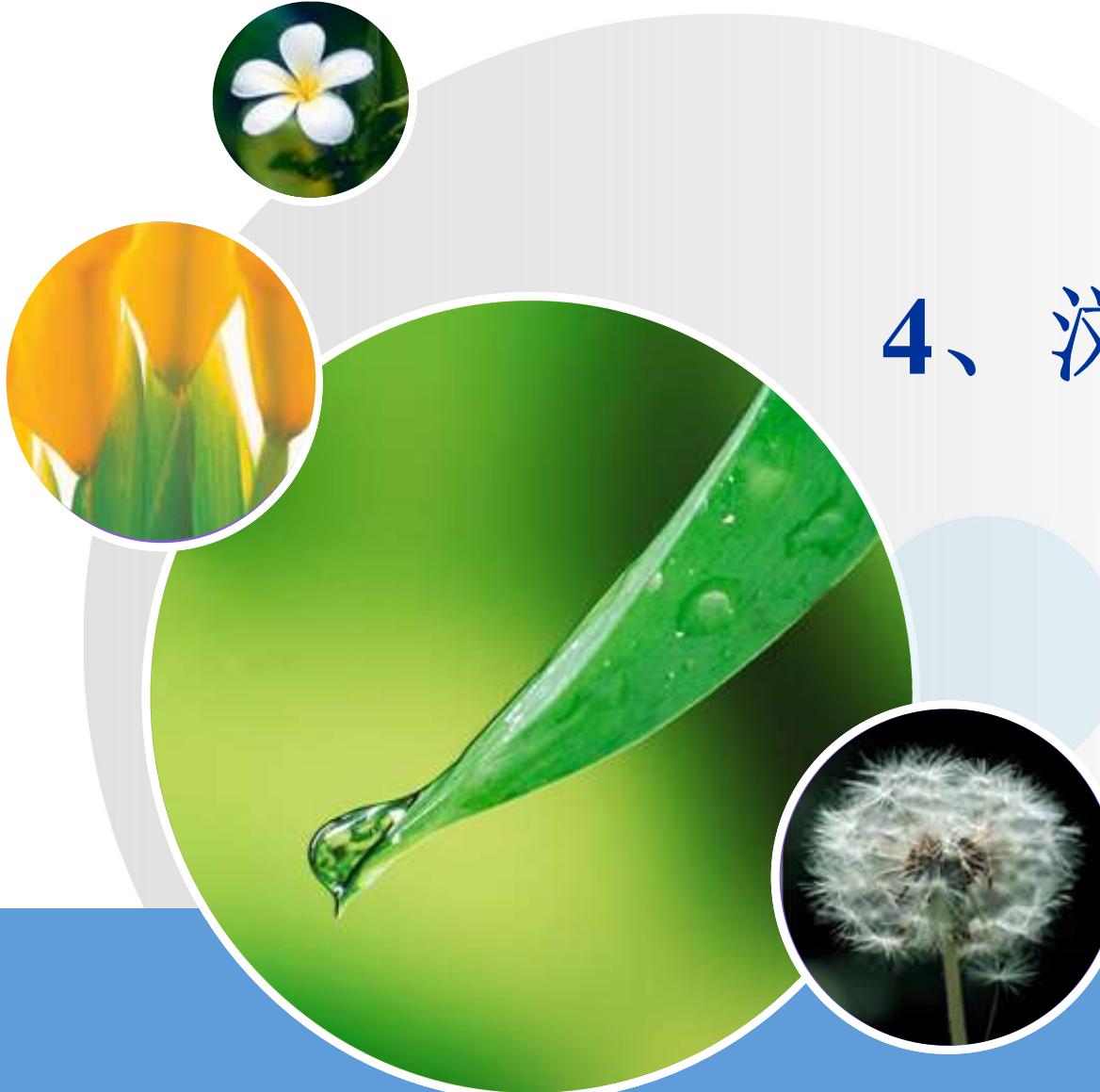
```
var phoneStr = "15131494601";
var phoneReg = /^1[3-9][0-9]{9}$/;

console.log(phoneReg.test(phoneStr));
console.log(/^1[3-9][0-9]{9}$/.test(phoneStr));

console.log(phoneReg.exec(phoneStr));
console.log(/^1[3-9][0-9]{9}$/.exec(phoneStr));

console.log(phoneStr.search(phoneReg));
console.log(phoneStr.search(/^1[3-9][0-9]{9}$/));

console.log(phoneStr.replace(phoneReg, "legal phone"));
console.log(phoneStr.replace(/^1[3-9][0-9]{9}$/ , "legal phone"));
```



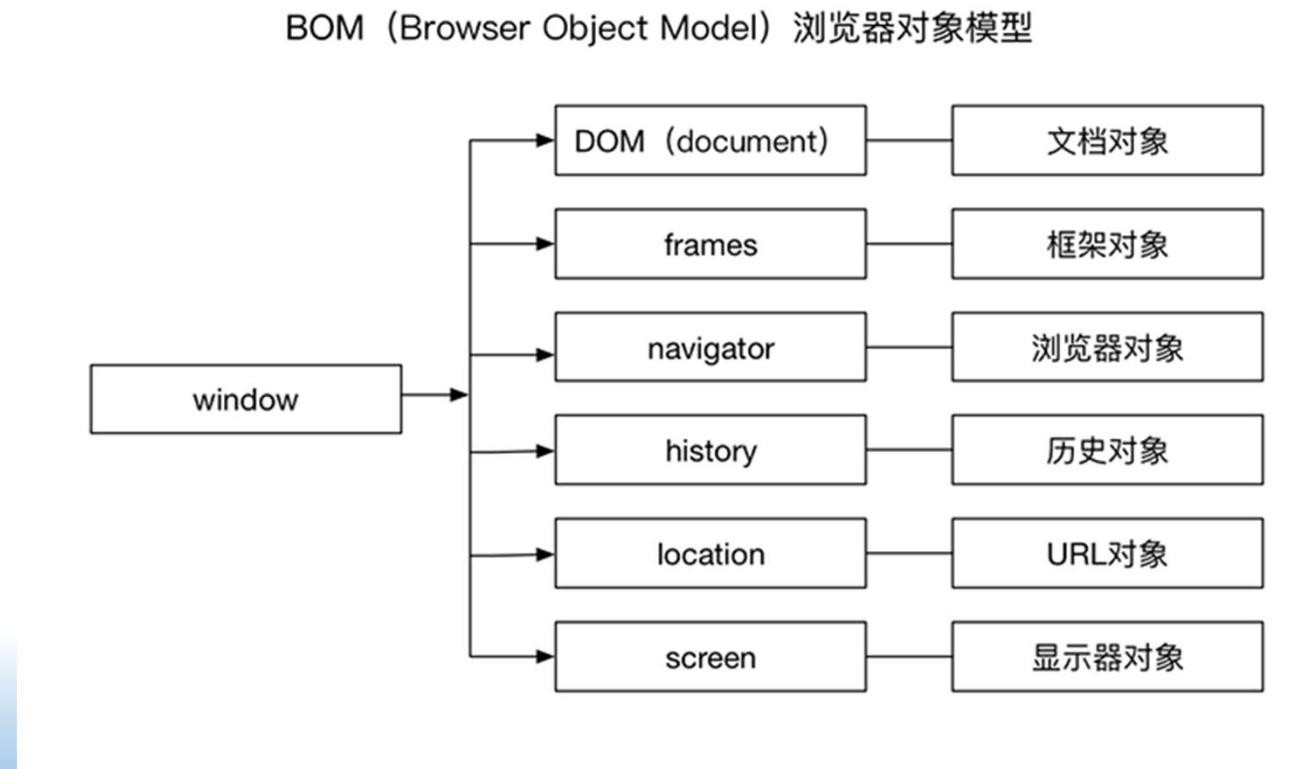
4、浏览器对象模型 **BOM**

第4章 JavaScript



4.1 浏览器对象模型BOM (Browser Object Model)

- ❖ JavaScript运行在浏览器中时，浏览器对象模型BOM（Browser Object Model）提供了对浏览器进行操作的API。





4.2 window对象

- ❖ window对象提供了操作浏览器窗口的API。如：
 - 调整窗口的大小和位置
 - 打开新窗口
 - 系统提示框
 - 状态栏控制
 - 定时操作



window的常用属性

- ❖ **document**: 当前窗口中显示的文档。
- ❖ **frames**: 当前窗口中所有frame对象的集合。
- ❖ **location**: 当前窗口的URL。
- ❖ **name**: 当前窗口的名称。
- ❖ **status**: 当前窗口的状态栏信息。defaultStatus，缺省状态。
- ❖ **top**: 最顶层的浏览器窗口对象。
- ❖ **parent**: 包含当前窗口的父窗口对象。
- ❖ **opener**: 打开/创建当前窗口的窗口对象。
- ❖ **closed**: 当前窗口是否已关闭。
- ❖ **self**: 当前窗口。
- ❖ **screen**: 用户屏幕对象。
- ❖ **navigator**: 浏览器对象。



window的常用方法

❖ 调整窗口的位置或大小

- `window.moveBy(dx, dy);`
 - 相对窗口当前位置移动(dx, dy)。
- `window.moveTo(x, y);`
 - 窗口左上角移动到坐标(x, y)位置。
- `window.resizeBy(dw, dh);`
 - 相对窗口当前大小调整(dw, dh)。
- `window.resizeTo(w, h);`
 - 窗口大小调整为(w, h)。



❖ window打开和关闭

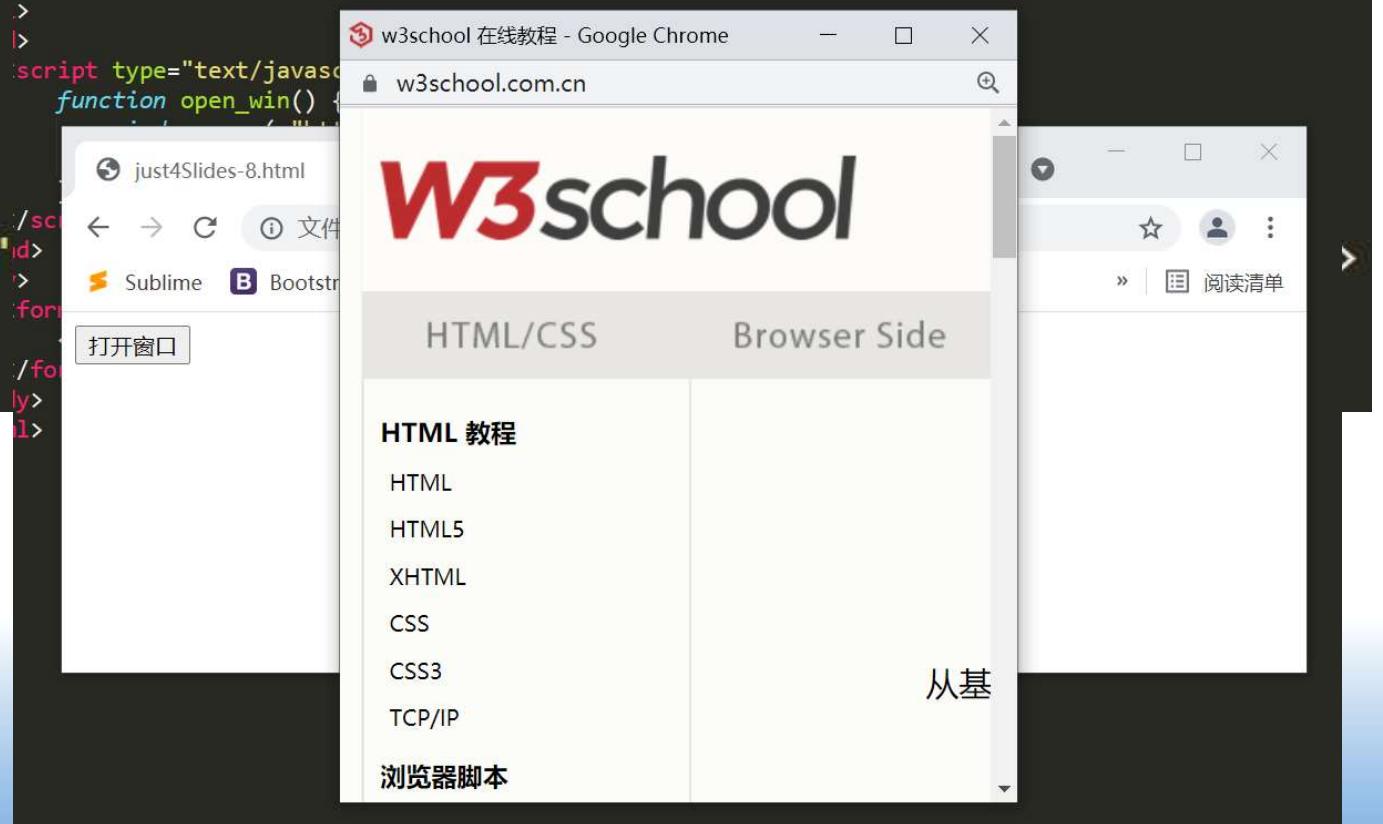
- `window.open([URL] [, name] [, features]);`
 - 打开一个新的或已知的名为name的窗口，窗口属性features可以包含height、width、left、top、menubar、scrollbar、status、toolbar和resizable等，地址为URL。
- `window.close();`
 - 关闭一个已打开的浏览器窗口。通常是顶层的用`window.open()`方法打开的。
 - 其他的窗口，直接调用`self.close()`或者`close()`关闭自身既可。



```
<head>
    <script type="text/javascript">
        function open_win() {
            window.open( "http://www.w3school.com.cn", "_blank",
                "width=400, height=400")
        }
    </script>
</head>
<body>
    <form>
        <input type = "button" value = "打开窗口" onclick = "open_win()">
    </form>
</body>
```



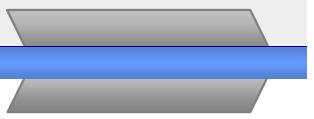
```
<head>
  <script type="text/javascript">
    function open_win() {
      window.open( "http://www.w3school.com.cn", "_blank",
      "width=400, height=400")
    }
  </script>
</head>
<body>
  <form>
    <input type = "button" value="打开窗口" onclick="open_win()">
  </form>
</body>
```



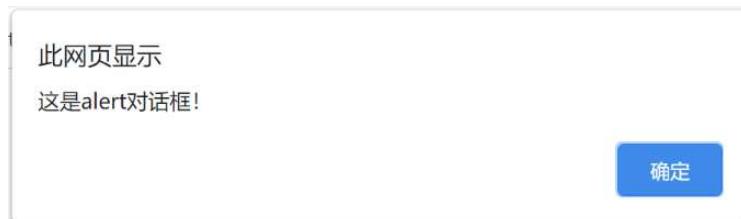


❖ 系统对话框

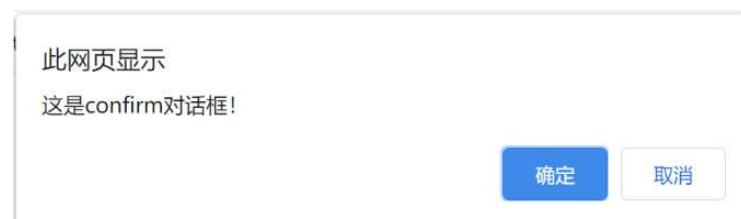
- `window.alert([msg]);`
 - 显示带有“确定”按钮的警告对话框，`msg`是显示给用户的信息。
 - `msg`必须是纯文本，不支持HTML格式，如需换行使用转义字符`\n`。
- `window.confirm([msg]);`
 - 显示带有“确定”和“取消”按钮的确认对话框。
 - “确定”返回`true`；“取消”返回`false`。
- `window.prompt([msg] [, default]);`
 - 显示带有“确定”和“取消”按钮的带有文本输入的对话框。`default`为文本输入框中的默认文本（可为空）。
 - “确定”返回用户输入的字符串，若未输入任何信息，返回`undefined`；“取消”返回`null`。



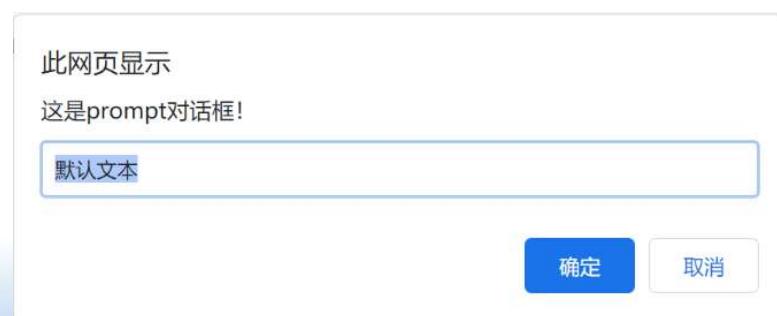
- `window.alert("这是alert对话框! ");`



- `window.confirm("这是confirm对话框! ");`



- `window.prompt "这是prompt对话框! ", "默认文本");`





❖ 定时器

- `setTimeout(code, milliseconds);`
 - 在milliseconds毫秒后，执行代码code。只执行一次。
- `setInterval(code, milliseconds);`
 - 每经过milliseconds毫秒，执行一次代码code。循环执行。
○ 返回值可作为`clearInterval(task)`方法的参数，使循环执行终止。
- `clearInterval(task);`
 - 终止`setInterval`函数开启的循环执行。



4.3 location对象

- ❖ location对象提供与window相关的URL信息
 - 通过location可进行网页的跳转
 - location.href = URL;
 - location = URL;

```
<body>
    <form>
        <input type = "button" value = "网页跳转"
               onclick = "location = 'http://www.w3school.com.cn'">
    </form>
</body>
```



location对象常用属性

- href: 当前载入页面的完成URL。
- protocol: URL中的协议信息，如http。
- host: URL中服务器信息，如www.baidu.com。
- hostname: 通常与host相同，有时会省略www。
- port: URL中的端口信息，如8080。
- pathname: URL中主机名后的部分，如/p/index.html。
- search: 执行GET请求时URL中?后的查询字符串，如?name=value。
- hash: 如果URL包含#，返回#后的部分，如#a1。



location对象常用方法

- `location.assign(URL);`
 - 同`location.href`, 新地址将加到浏览器的历史栈。
- `location.replace(URL);`
 - 同`assign`, 新地址不加到历史栈, 不能通过`back`和`forward`访问。
- `location.reload(true | false);`
 - 重新载入页面, 默认`false`时从浏览器缓存中, `true`时从服务器端重载。
- `location.reload(URL);`
 - 打开新的页面。



4.4 document对象

- ❖ document是表示HTML文档的对象DOM
 - document.write()方法可在当前页面中输出字符串或HTML代码片段。

```
<body>
    <script type="text/javascript">
        document.write("<h1>", "WEB世界欢迎您！", "</h1>");
    </script>
</body>
```





document对象常用属性

- body: 访问<body>, 如果定义了框架, 返回最外层<frameset>。
- cookie: 当前文档的cookie。
- title: 文档标题, 等价于HTML的title标签。
- domain: 当前文档的域名。
- URL: 当前文档的URL。
- referrer: 载入当前文档的文档的URL。
- lastModified: 当前文档最后修改的日期和时间。



document对象常用方法

- `document.write();`
 - 向页面写入文本、HTML表达式和JavaScript代码。
- `document.writeln();`
 - 写入后加换行。
- `document.createElement(tagName);`
 - 创建tagName标签类型的元素，返回。
- `document.getElementById(id);`
 - 获取指定id的对象。
- `document.getElementsByName(name);`
 - 获取指定名字name的对象集合。
- `document.getElementsByTagName(tagName);`
 - 获取指定标签名tagName的对象集合。

5、文档对象模型 DOM

第4章 JavaScript



5.1 文档对象模型DOM (Document Object Module)

- ❖ DOM提供了对HTML页面元素进行操作的标准API。
- ❖ JavaScript通过浏览器内置的document对象访问DOM。
- ❖ 每个HTML元素都有一个DOM节点（node）与其对应，每个节点都有一系列属性和方法。例如，
 - 通过HTML元素对应节点的innerHTML属性可以获取或更改该元素的内容。
 - 属性nodeName和nodeValue是每个元素型节点都有的通用属性，表示该节点对应的HTML元素的标签名和元素的值。

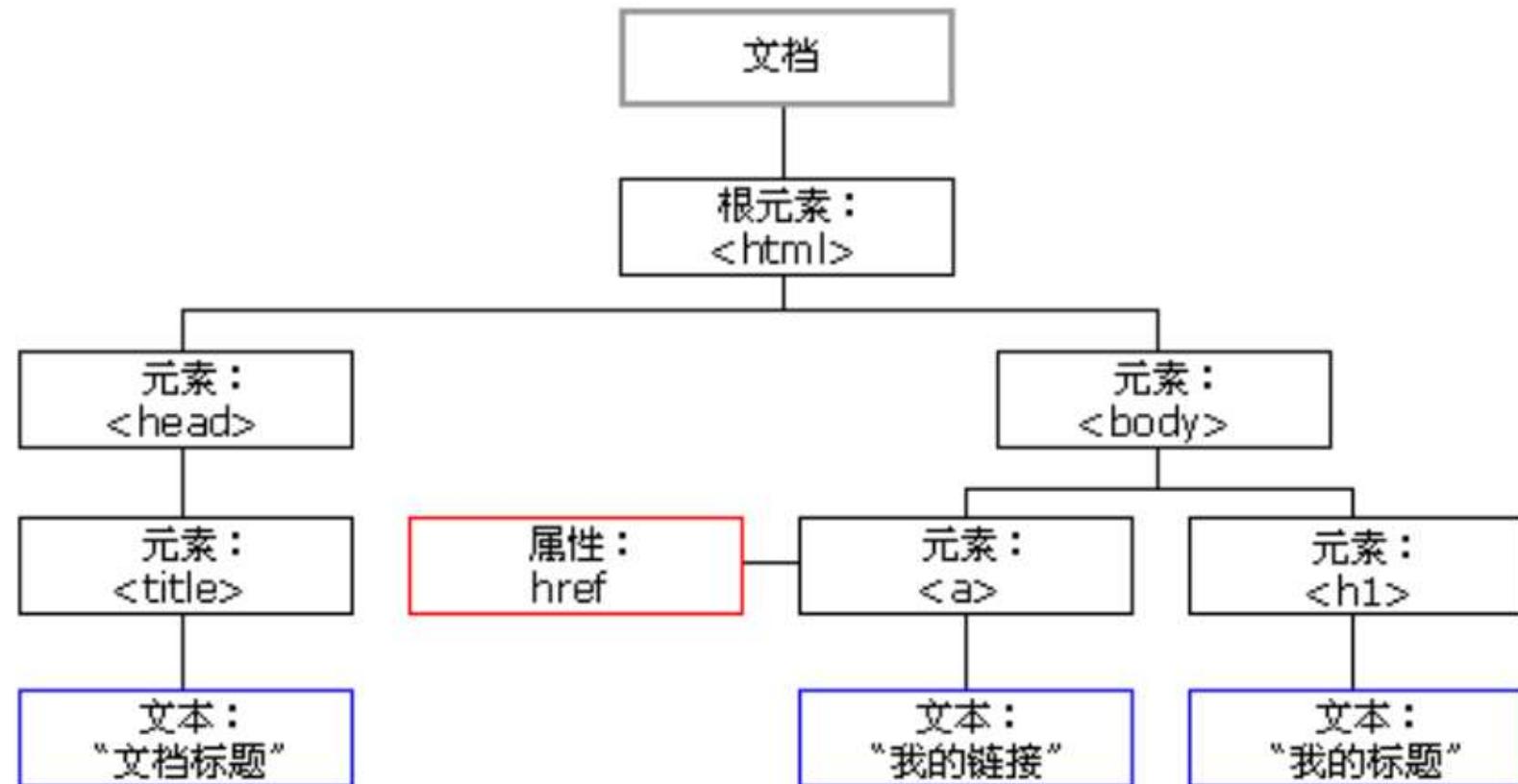


HTML文档

```
<html>
  <head>
    <title>文档标题</title>
  </head>
  <body>
    <a href="我的URL"> 我的链接 </a>
    <h1> 我的标题 </h1>
  </body>
</html>
```



HTML文档对应的DOM



- ❖ 注意：文本是元素型节点的子节点，而不是元素型节点的值。



5.2 访问节点

1、document.getElementById(id);

- 返回在HTML页面对应的document中具有指定id的元素型节点

```
<body>
    <h1 id="myHeader" onclick="getValue()"> 这是标题 </h1>
    <p> 点击标题，会提示出它的值。 </p>
    <script type="text/javascript">
        function getValue(){
            var oHeader = document.getElementById("myHeader");
            alert(oHeader.innerHTML);
        }
    </script>
</body>
```



1、document.getElementById(id);

- 返回在HTML页面对应的document中具有指定id的元素型节点

```
<body>
    <h1 id="myHeader" onclick="getValue()"> 这是标题 </h1>
    <p> 点击标题，会提示出它的值。 </p>
    <script type="text/javascript">
        function getValue(){
            var oHeader = document.getElementById("myHeader");
            alert(oHeader.innerHTML);
        }
    </script>
</body>
```





2、document.getElementsByTagName(tagName);

- 返回document对象（或元素型节点对象）的子元素中所有标签名为tagName的元素型节点对象数组

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了?"><br>
    <script type="text/javascript">
        function getElements(){
            var inputs = document.getElementsByTagName("input");
            var strInputs = "";
            for ( i = 0; i < inputs.length; i++){
                strInputs += inputs[i].value + "\n";
            }
            alert(strInputs);
        }
    </script>
</body>
```



2、document.getElementsByTagName(tagName);

- 返回document对象（或元素型节点对象）的子元素中所有标签名为tagName的元素型节点对象数组

The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the following HTML and JavaScript:

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="显示" style="width: 100px; height: 30px; border: 1px solid black; border-radius: 5px; background-color: #f0f0f0; font-size: 14px; font-weight: bold; padding: 5px; margin-bottom: 10px;"/>
    <script type="text/javascript">
        function getElements(){
            var inputs = document.getElementsByTagName("input");
            var strInputs = "";
            for (var i = 0; i < inputs.length; i++){
                strInputs += inputs[i].value + "\n";
            }
            alert(strInputs);
        }
    </script>
</body>
```

The browser window shows the URL `localhost/temp.html`. An alert box is displayed with the following text:
localhost 显示
早上好呀!
早上好!
吃了吗?
都说什么了?
都说什么了?
都说什么了?
都说什么了?
都说什么了?



3、document.getElementsByName(name);

- 返回document对象（或元素型节点对象）的子元素中所有名字属性为name的元素型节点对象数组

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了？"><br>
    <script type="text/javascript">
        function getElements(){
            var inputs = document.getElementsByName("myInput");
            var strInputs = "";
            for ( i = 0; i < inputs.length; i++){
                strInputs += inputs[i].value + "\n";
            }
            alert(strInputs);
        }
    </script>
</body>
```



❖ 获取<html>和<body>元素的节点

- var objHtml = document.documentElement;
- var objBody = document.body;

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了？"><br>
    <p id="dom"></p>
    <script type="text/javascript">
        var objHtml = document.documentElement;
        var objBody = document.body;
        var domString = "";
        domString += objHtml.nodeName + ":" + objHtml.innerHTML + "\n";
        domString += objBody.nodeName + ":" + objBody.innerHTML + "\n";
        console.log(domString);
    </script>
</body>
```



❖ 获取<html>和<body>元素的节点

- var objHtml = document.documentElement;
- var objBody = document.body;

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了?"><br>
    <p id="dom"></p>
    <script type="text/javascript">
        var objHtml = document.documentElement;
        var objBody = document.body;
        var domString = "";
        domString += objHtml.nodeName + ":" + objHtml.innerHTML;
        domString += objBody.nodeName + ":" + objBody.innerHTML;
        console.log(domString);
    </script>
</body>
```

```
HTML:<head>
    <meta charset="UTF-8">
</head>
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了?"><br>
    <p id="dom"></p>
    <script type="text/javascript">
        var objHtml = document.documentElement;
        var objBody = document.body;
        var domString = "";
        domString += objHtml.nodeName + ":" + objHtml.innerHTML + "\n";
        domString += objBody.nodeName + ":" + objBody.innerHTML + "\n";
        console.log(domString);
    </script></body>
BODY:
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="都说什么了?"><br>
    <p id="dom"></p>
    <script type="text/javascript">
        var objHtml = document.documentElement;
        var objBody = document.body;
        var domString = "";
        domString += objHtml.nodeName + ":" + objHtml.innerHTML + "\n";
        domString += objBody.nodeName + ":" + objBody.innerHTML + "\n";
        console.log(domString);
    </script>
    .
    .
    .
temp.html:18
```



❖ childNodes属性获取子节点数组

- var objChildren = objNode.childNodes;
- objNode可以是document或元素型节点

```
<body>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input name="myInput" type="text" size="20"><br>
    <input type="button" onclick="getElements()" value="Get Elements">
    <p id="dom"></p>
    <script type="text/javascript">
        var objBody = document.body;
        var objChildren = objBody.childNodes;
        var domString = "";
        for (var i = 0; i < objChildren.length; i++){
            domString += objChildren[i].nodeName + ":" + objChildren[i].innerHTML + "\n";
        }
        console.log(domString);
    </script>
</body>
```

```
#text:undefined
INPUT:
BR:
#text:undefined
P:
#text:undefined
SCRIPT:
    var objBody = document.body;
    var objChildren = objBody.childNodes;
    var domString = "";
    for (var i = 0; i < objChildren.length; i++){
        domString += objChildren[i].nodeName + ":" + objChildren[i].innerHTML +
    "\n";
    }
    console.log(domString);
```



5.3 访问元素属性

- ❖ 获取元素属性值

- `var attrValue = objNode.getAttributeName();`

- ❖ 设置元素属性值

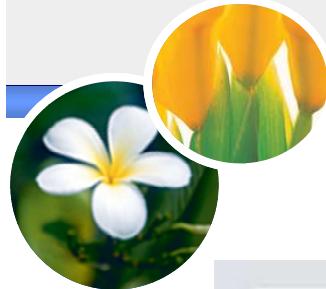
- `objNode.setAttributeName(attrValue);`

- ❖ 删 除 元 素 属性

- `objNode.removeAttributeName();`



```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title> 设置元素属性 </title>
5  </head>
6  <body>
7      <div>
8          
9          <ul>
10             <li onmouseover = "changePic(1)"> 小动物 </li>
11             <li onmouseover = "changePic(2)"> 圣诞节 </li>
12             <li onmouseover = "changePic(3)"> 卡通画 </li>
13         </ul>
14     </div>
15     <script type="text/javascript">
16         function changePic(n){
17             var myImg = document.getElementById("picbox");
18             myImg.src = "../img/img-"+n+".jpg"; //更新元素的属性值
19         }
20     </script>
21 </body>
22 </html>
```



设置元素属性

文件 | D:/phpstudy_pro/WWW/le...

Sublime Bootstrap中文网 阅读清单

- 小动物
- 圣诞节
- 卡通画



❖ 访问节点元素属性的函数

- `objNode.getAttribute(attrName);`
 - 返回`objNode`元素节点`attrName`属性的值，或`null`。
- `objNode.setAttribute(attrName, attrValue);`
 - 将`objNode`元素节点的`attrName`属性赋值为`attrValue`。
`attrName`不存在时添加新属性。
- `objNode.hasAttribute(attrName);`
 - 检测`objNode`元素节点是否包含`attrName`属性，返回`true`或`false`。
 - 获取元素属性值时，先检测是否存在，避免返回`null`。
- `objNode.removeAttribute(attrName);`
 - 删除`objNode`元素节点的`attrName`属性，无返回值。

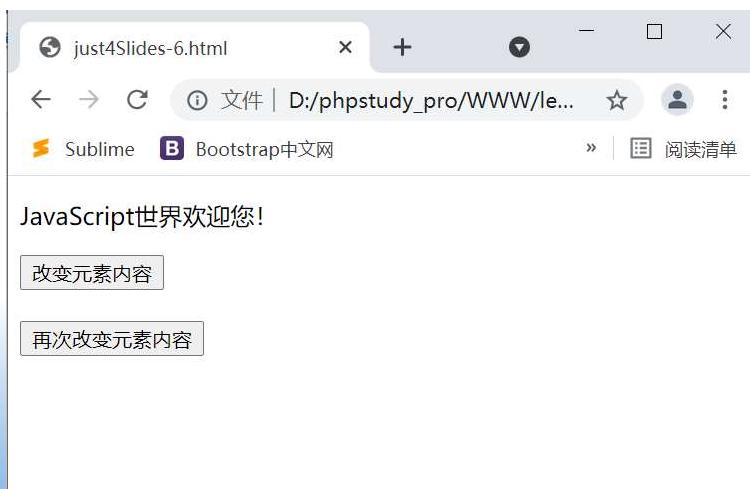
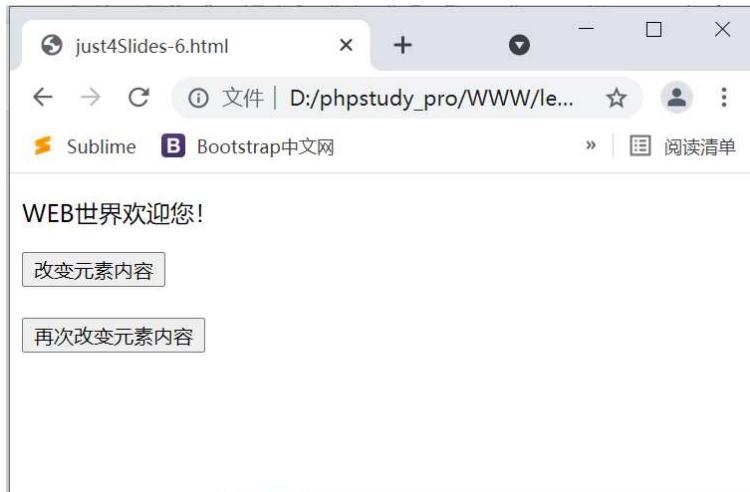


5.4 访问和设置元素内容

- ❖ innerHTML：元素开始和结束标签之间的内容。
- ❖ nodeValue：节点的值，如文本节点。

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <p id = "welcome"> WEB世界欢迎您！ </p>
5      <input type = "button" value = "改变元素内容"
6          onClick = "document.getElementById('welcome').childNodes[0].nodeValue = 'JavaScript世界欢迎您！';">
7      <br/><br/>
8      <input type = "button" value = "再次改变元素内容"
9          onClick = "document.getElementById('welcome').innerHTML = '<h1>我们大家都欢迎您！</h1>';" />
10     </body>
11     </html>
```

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4     <p id = "welcome"> WEB世界欢迎您！ </p>
5     <input type = "button" value = "改变元素内容"
6         onClick = "document.getElementById('welcome').childNodes[0].nodeValue = 'JavaScript世界欢迎您！';">
7     <br/><br/>
8     <input type = "button" value = "再次改变元素内容"
9         onClick = "document.getElementById('welcome').innerHTML = '<h1>我们大家都欢迎您！</h1>';" />
10 </body>
11 </html>
```





❖ 访问元素CSS样式

- **style:** 元素节点的CSS样式属性。

```
var objNode = document.getElementById('test');
objNode.style.color = 'red';
objNode.style.fontFamily = '楷体';
```



4-object.html > ...

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <script src="4-object.js"></script>
5      </head>
6      <body>
7          <p id="test">p元素test</p>
8      </body>
9  </html>
```

4-object.js > ...

```
1  /*4-object.js*/
2  var objNode = document.getElementById('test');
3  console.log(objNode);
4  objNode.style.color = 'red';
5  objNode.style.fontFamily = '楷体';
6  console.log(objNode.style.cssText);
```



The screenshot shows a browser developer tools console window. The title bar indicates the URL is `localhost:3000/4-object...`. The toolbar includes icons for back, forward, search, and refresh, along with a user profile icon and a red-bordered "更新" (Update) button. The main area has tabs for "元素" (Elements), "控制台" (Console), and "源代码" (Source). The "控制台" tab is selected. Below it is a toolbar with icons for play/pause, stop, top, eye, and filter, followed by a message "过滤" (Filter) and "5" errors. A dropdown menu says "默认级别" (Default Level) and shows "49 个问题" (49 issues) with "48" errors and "1" warning. The console log lists one error: "null" at "4-object.js:3" followed by a detailed stack trace.

```
null
4-object.js:3
✖ > Uncaught TypeError: Cannot read properties of null (reading 'style')
    at 4-object.js:4:9
```



4-object.html > ...

```
1 <!DOCTYPE html>
2 <html>
3     <head></head>
4     <body>
5         <p id="test">p元素test</p>
6         <script src="4-object.js"></script>
7     </body>
8 </html>
```

↑ ↓ ⌂ ⌂

① localhost:3000/4-object... 🔎 ⚡ ☆ 🧩 🔍 🚙 更新 ⋮

p元素 test

| 元素 控制台 源代码 » 5 36 | ⋮

76 条已隐藏 

默认级别 ▾ | 37 个问题: ✘ 36 = 1

4-object.js:3

```
<p id="test" style="color: red; font-family: 楷体;">p元素  
test</p>
```

color: red; font-family: 楷体;

4-object.js:6



5.5 创建节点

- ❖ `document.createElement(tagName);`
 - 创建tagName类型元素节点
- ❖ `document.createTextNode(txt);`
 - 创建内容为txt的文本节点
- ❖ `objNode.appendChild(childNode);`
 - 在当前节点的子节点列表的最后添加childNode
- ❖ `objNode.replaceChild(newNode, oldNode);`
 - oldNode子节点替换为newNode
- ❖ `objNode.insertBefore(newNode, nodeChild);`
 - 在nodeChild子节点前增加新节点newNode



- ❖ `objNode.cloneNode(false | true);`
 - 复制并创建新节点
 - `true`包含全部后代节点，`false`只包含属性和值。
- ❖ `objNode.removeChild(nodeChild);`
 - 删除子节点`nodeChild`后返回，未找到时返回`null`。



```
59      <!--追加节点-->
60      <ul id="ul">
61          <li> AA </li>
62          <li> BB </li>
63          <li> CC </li>
64      </ul>
65      <script type="text/javascript">
66          //获取页面中指定id的元素对象并保存在变量中
67          var objUl = document.getElementById("ul");
68          //创建一个li元素并保存到变量中
69          var objLi = document.createElement("li");
70          //创建元素显示的文本节点并保存到变量中
71          var objTxt = document.createTextNode("DD");
72          //将文本节点追加到新元素汇总
73          objLi.append(objTxt);
74          //向列表元素追加新增加的元素
75          objUl.appendChild(objLi);
76      </script>
77  </body>
78  </html>
```



- AA
- BB
- CC
- DD

```
59      <!--追加节点-->
60      <ul id="ul">
61          <li> AA </li>
62          <li> BB </li>
63          <li> CC </li>
64      </ul>
65      <script type="text/javascript">
66          //获取页面中指定id的元素对象并保存在变量中
67          var objUl = document.getElementById("ul");
68          //创建一个li元素并保存到变量中
69          var objLi = document.createElement("li");
70          //创建元素显示的文本节点并保存到变量中
71          var objTxt = document.createTextNode("DD");
72          //将文本节点追加到新元素汇总
73          objLi.append(objTxt);
74          //向列表元素追加新增加的元素
75          objUl.appendChild(objLi);
76      </script>
77  </body>
78  </html>
```



```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <div id="links">
5          <p><a href = "1.html"> first </a></p>
6          <p><a href = "2.html"> second </a></p>
7          <p><a href = "3.html"> third </a></p>
8      </div>
9      <script>
10         n = document.createElement("h3");
11         n.innerHTML = "Hello!";
12         q = document.getElementById("links");
13         q.replaceChild(n, q.getElementsByTagName("p")[0]);
14     </script>
15  </body>
16  </html>
```



just4Slides-6.html

← → ⌂ ⌂ 文件 | D:/phpstudy_pro/WWW/le... ☆ ⌂ Sublime ⌂ Bootstrap中文网 ⌂ 阅读清单

Hello!

second

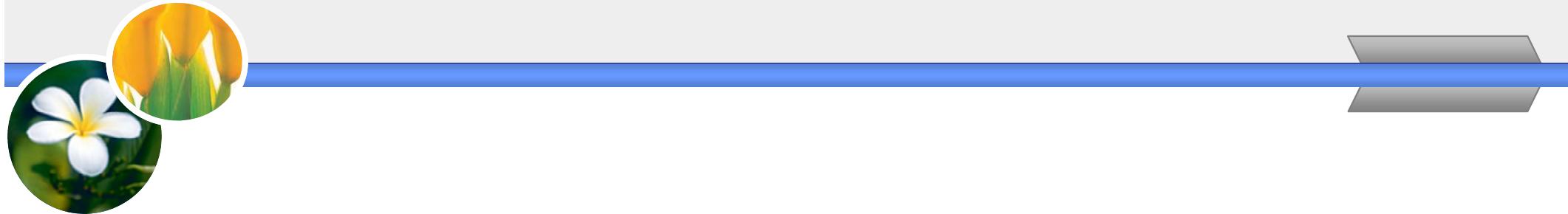
third

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <div id="links">
5          <p><a href = "1.html"> first </a></p>
6          <p><a href = "2.html"> second </a></p>
7          <p><a href = "3.html"> third </a></p>
8      </div>
9      <script>
10         n = document.createElement("h3");
11         n.innerHTML = "Hello!";
12         q = document.getElementById("links");
13         q.replaceChild(n, q.getElementsByTagName("p")[0]);
14     </script>
15  </body>
16  </html>
```



例：控制表单元素

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title> 检查表单 </title>
5      <script type="text/javascript">
6          function checkValue() {
7              if ( document.getElementById("username").value == "" ){
8                  alert( "用户名不能为空！" );
9                  return( false );
10             }
11             return( true );
12         }
13     </script>
14 </head>
15 <body>
16     <form method = "post" action = "" onsubmit = "return(checkValue())">
17         username: <input type = "text" id = "username"/>
18         password: <input type = "text" id = "password"/>
19         <input type = "submit" value = "登 录" />
20     </form>
21 </body>
22 </html>
```



just4Slides-6.html

← → C ⌂ 文件 | D:/phpstudy_pro/WWW/learnTest/js/just4Slides-6.html

Sublime Bootstrap中文网 Swiper中文网-轮播... Download jQuery...

username: password: 登录

just4Slides-6.html

← → C ⌂ 文件 | D:/phpstudy_pro/WWW/learnTest/js/just4Slides-6.html

Sublime Boot...

username:

此网页显示
用户名不能为空!

确定