# 一.轻松搞定网页设计html+css+javascript—javascrip部分

## 课程介绍

### 1.为什么要学习javascript

javascript是实现网页动态效果的基石，在web开发中扮演重要的角色，被广泛的应用到各个领域：

**1、网页游戏；**

**2、地图搜索(google,baidu)；**

**3、股市信息查询；**

**4、web聊天，等**

**在web开发中，js是必不可少的技术!同时js也是ajax/jquery/extjs等框架的基础。**

### 2.javascript介绍--基本介绍

javascript是一种广泛用于客户端web开发的脚本语言，常用来给HTML网页添加动态功能，比如响应用户的各种操作。

#### 2.1什么是脚本语言？

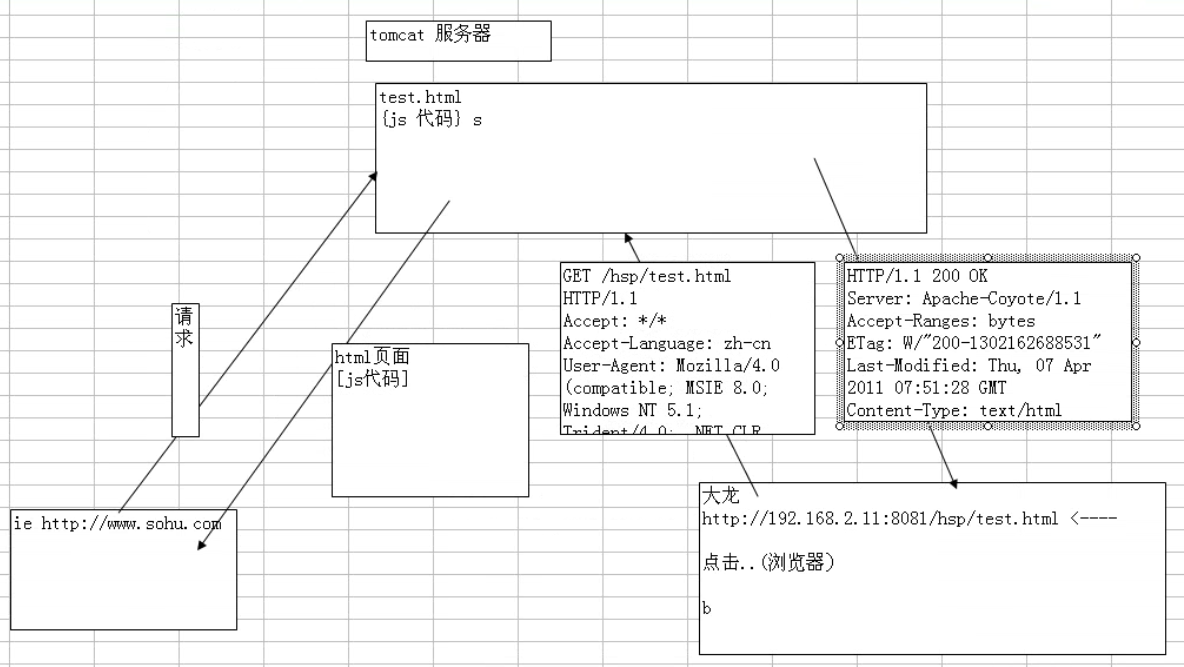
1、脚本语言介于HTML和C，C++，Java，C#等编程语言之间；

2、脚本语言与编程语言有相似地方，其函数与编程语言类似，也有变量。与编程语言之间最大的区别是编程语言的语法和规则更为严格和复杂一些。

3、脚本语言是一种**解释性语言**，例如Python、vbscript，javascript等等，它不象C/C++等可以编译成二进制代码，也可执行的形式存在。

4、脚本语言一般都有相应的脚本引擎来解释执行，javascript的脚本引擎就是浏览器。

5、js脚本语言往往不能独立使用，它和html/jsp/php/asp/asp.net配合使用才有意义。



javascript最主要的设计目的就是让网页中的元素可以被编程也就是让网页动起来，甚至你可以编写js版本的坦克大战游戏；

javascript不能独立运行，它工嵌入到其它编程语言中才能运行(html,asp,asp.net,jsp,php...)；

javascript只能用于b(brower浏览器)/s(server服务器)开发，即我们常说的web开发(比如网站开发，web应用等)；

javascript主要针对客户端编程，也就是说js一般运行在客户端；

javascript也可以对服务器端编程(比较少)；

javascript最新版本是1.8

javascript在客户端(浏览器)执行

### 3.javascript介绍--发展历史

javascript最早是叫livescript，是一种基于对象和事件驱动，并且备安全性的脚本语言，是由netscapce(网景)公司1996开发的，后和sun公司合作，将livescript更名为javascript因为浏览器的差异，我们会看到，在不同的浏览器中使用js编程，会有差异，比如，在给某个控件绑定路应函数时，w3c dom标准是addEventListener()，而ie是使用attachEvent(),还有就是有些方法在ie中可以使用，而在ff中却不能使用，这些都是我们在实际开发中要注意的问题。

**因为js是由浏览器来解释执行的，因此这里有一个问题，不同的类型的浏览器可能对js的支持不一样。**

### 4.js开发工具

如何选择开发工具?

1. 记事本；
2. 2、eclipse或myeclipse

### 5.js语言的特点

javascript语言是简单的；

javascript是基于对象和事件驱动(动态的)；

它可以直接对用户或客户输入做出响应，无需经过web服务程序。它对用户的反映响应，是采用以事件驱动的方式进行的。所谓事件驱动，就是指在主页中执行了某种操作所产生的动作，就称为“事件”。比如按下鼠标，移动窗口、选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

javascript是跨平台的。

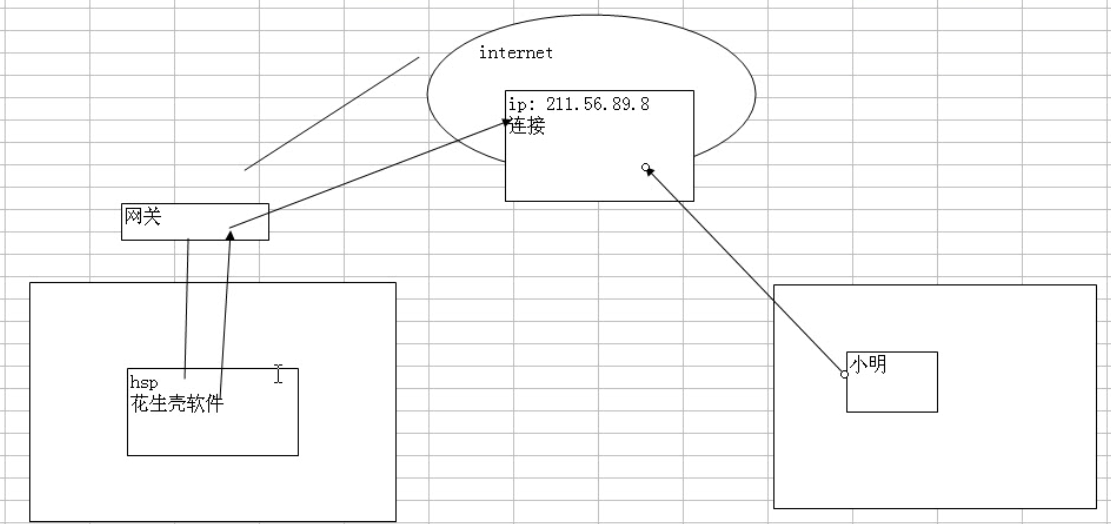
javascript是依赖于浏览器本向，与操作环境无关，只要能运行浏览器的计算机，并支持javascript的浏览器就可以正确执行。

js是脚本编程语言；

js是解释执行的；

js可以对客户端编程，也可以对服务器编程(较少)。

**注:内网映射成外网.实际是自己变成他的代理.**



### 6.javascript和jscript比较

1、jscript是微软开发的，而javascript则是网景和sun合作开发的；

2、在网景开发livescript(javascript前身)是，微软就意识到这个脚本语言会很火，于是找到网景表示希望合作开发livescript，但是网景公司最终选择和微软的对头sun公司合作开发livescript，并更名为javascript。

3、微软一气之下，开发了针对ie的脚本语言，并命名为jscript，在当时的这二种脚本语言中语法是大相径庭，给当时的程序员带来很大的麻烦，后来w3c出面了，w3c统一了标准(ECMAScript)所以现在的这二种脚本语言的语法可以说90%以上都是相同的。

4、在一定程度上，你可以认为jscript和javascript是一样的。

5、在网上搜索javascript帮助手册，完后打开一看是jscript帮助手册，就不足为怪了。

# 二.语言基础

### 1.第一个程序及其注意事项

[js01.html]js第一个案例hello world

<html>

<head>

<title>js示例helloworld</title>

**<!--js代码一般是放在head标签中，但实际也可以在别的位置-->**

**<script language="javascript">**

**window.alert("hello world!");**

**</script>**

</head>

<body>

</body>

</html>

**1、js位置可以随意；**

**2、js必需用<script language="javascript">js代码</script>**

**如果没有使用script包起来，则浏览器会将其视为普通文本。**

**3、在一个html文件中(jsp/php/asp等)可以出现多对script片段。浏览器会按先后顺序依次执行。**

### 2.初步了解js的变量定义和使用

[js02.html]加法运算

<html>

<head>

<title>js示例加法运算</title>

<!--js代码一般是放在head标签中，但实际也可以在别的位置-->

<script language="javascript">

//js中变量的定义(在js中变量用var表示，不论要定义的变量的实际类型)

var num1=456;

var num2=89;

var reault=num1+num2;

window.alert('结果是'+reault);//alert函数,直接写window.alert(num1+num2)可

</script>

</head>

<body>

</body>

</html>

**js为什么需要变量**

不论是使用哪种语言编写程序，变量都是其程序的基本组成单位。js是弱数据类型的语言，在定义变量时，不需要指明该变量的类型(由js引擎来决定)

var name="shunping";//name是字符串

var kk=2;//kk是数

var yy;//yy是undefined

如果name=234;//这时name自动变成数

不声明变量直接使用

x=345;//亦可

**js的变量的类型究竟怎样决定**

**1、js是弱数据类型语言；即在定义变量的时候，统一使用var表示即可，甚至不写也可以。**

**2、js中的变量数据类型是由js引擎决定的**

### 3.js基本语法

#### 3.1基本常识

**javascript中的标识符**

标识符是指javascript中定义的符号，例如：变量名，函数名，数组名等。标识符可以由任意顺序的大小写字母、数字、下划线(\_)和美元符号($)组成，但标识符不能以数字开头，不能是javascript中的保留关键字。

合法的标识符举例：indentifier、username、user\_name、\_userName、$username

非法的标识符举例：int、98.3、Hello World

**javascript严格区分大小写computer和Computer是两个完全不同的符号,标记语言统统不区分大小写,而变成语言是需要的.**

**javascript程序代码的格式**

**每条功能执行语句的最后必需用分号(;)结束，每个词之间用空格、制表符、换行符或大括号、小括号这样的分隔符隔开。**

**语句块使用{}来表示**

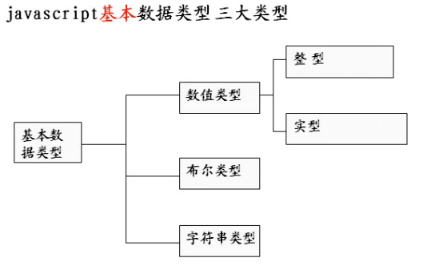
**javascript程序的注释**

/\*...\*/中可以嵌套"//"注释，但不能嵌套"/\*...\*/"。

**调试是比较难的,而且提示的时候不一定正确需要自己去研究**

**Js的数据类型:基本数据类型,复合数据类型,特殊数据类型**

#### 3.2javascript基本数据类型--三大类型



基本数据类型

**数值类型-->整型、实型**

**布尔类型**

**字符串类型**

**通过typeof可以看到变量的具体数据类型是什么**

[js03.html]查看数据类型示例

<html>

<head>

<title>js示例查看数据类型</title>

<script language="javascript">

var v1="abc";

var v2=890;

window.alert("v1是："+typeof v1+"v2是："+typeof v2);

v1=456; **//可以动态的变化,体现了动态语言特征**

window.alert("v1是："+typeof v1+"v2是："+typeof v2);

</script>

</head>

<body>

</body>

</html>

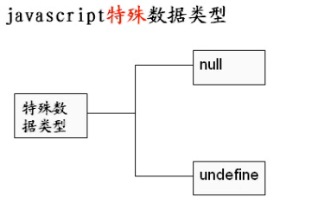
#### 3.3js基本语法--复合数据类型

**复合数据类型：数组；对象。**



#### 3.4js基本语法--特殊数据类型

**javascript特殊数据类型：null；undefine**



null空值: var a=null;

Undefine: var v; window.alert(v);

**//没有给v赋值，就会返回一个undefine,使用的时候就返回undefine**

#### 3.5javascript基本数据类型 三大类型总结

**Number(数字类型)**

1、整形常量(10进制\8进制\16进制)

十六进制以0x或0X开头，例如：0x8a。

八进制必需以0开，例如：0123。

十进制的第一们不能是0(数字0除外)，例如：123。

2、实型常量

12.32、193.98、5E7、4e5等。

特殊数值：NaN、Infinity、isNaN()、isFinite() 不是一个数字/无穷大/不是一个数字的Boolean值/这是一个无穷大么返回Boolean值

NaN:var a="abc"; window alert(parseInt(a));返回提示NaN不是一个数字类型

Infinity: window alert(6/0);返回提示Infinity(无穷大)

编程时一定要尽量避免

isNaN:window alert(isNaN("abc"));不是数返回true，是数返回false

**Boolean(布尔值)--true和false**

js中数字0在布尔值中表示false，非0的数字都为true

var b = 1;

window.alert(b?2:3); //可以使用三元运算符

**String字符串(单引号或者双引号都是可以的)**

"a book of JavaScript"、'abc'、"a"、""

**字符串中的特殊字符，需要以反斜杠(\)后跟一个普通字符来表示**，例如：\r、\n、

\t、\b、\'、\"、\\xxx

js基本语法--定义变量、初始化、赋值

定义变量

什么是定义变量？

var a;这就是定义一个变量，变量名是a

初始化变量

在定义变量的时候，就给值。

var a=45;这就是初始化变量a

给变量赋值

比如你先定义了变量：var tt;

然后，再给值tt=780;-->这就是给变量赋值。

#### 3.6js基本语法--数据类型转换

**自动转换**

案例：

var a=1.2; a=45; a="韩顺平"; window.alert(a);//输出的结果是韩顺平,再次赋值或者加上””就自动变换了.

**强制转换**

如何字符串转成数字？

var a=parseInt("123"); **//parseInt()系统函数直接使用,全部函数**

var b=parseFloat("12.3");

...//其它后面介绍

**parseInt()，parseFloat()是系统函数，可以直接使用。**

#### 3.7js基本语法--运算符

##### 算术运算符 +加、-减、\*乘、/除、%取模

这个听其名而知其意，就是用于计算的，javascript常用的是：

**其实%运算可以得到两个数相除的余数(主要用于整数,小数也是可以的但是没有意义取出来的也是一堆的精度是有问题的)。**

[js04.html]取模示例

<html>

<head>

<title>js示例取模示例</title>

<script language="javascript">

var a=90;

var b=8;

if(a%b==0){

window.alert("能整除");

}else{

window.alert("不能整除");

}

</script>

</head>

<body>

</body>

</html>

##### 算术运算符 ++自加、--自减

这两个运算符比较特殊

var a=90; a++; -->a=a+1

var b=89; b--; -->b=b-1

var a=56;

var b=++a;//b=++a<==>[a=a+1; b=a;] b=a++<==>[b=a;a=a+1;]

window.alert(b); b=57

window.alert(a); a=57

##### 算术运算符 +=左加、-=左减 、/=左除、%=左取模

var a=90;

a+=90;=>a=a+90

var b=89.7;

b+=a;=>b=b+a;

var a=56; var b=90;

a-=34;b%=a;

window.alert(b); b=2

window.alert(a); a=22

##### 关系运算符 ==等于、>大于、<小于、>=大于等于、<=小于等于、!=不等于

数字和字符串比较不会将字符串转成ascll码,但是若是字符串是数据会调用parseInt进行比较的,开发中一定要避免如此的比较

字符传比较的没有了equal()了,任何比较都是==

var a=90; var b=90;

if(a==b){alert("ok1");}

b--;

if(a>b){alert("ok2");}

if(a>=b){alert("ok3");}

[js06.html]运算符的示例

<html>

<head>

<title>js示例关于运算符的使用示例</title>

<script language="javascript">

**var num1=window.prompt("请输入第一个数");**

var num2=window.prompt("请输入第二个数");

num1=parseFloat(num1);

num2=parseFloat(num2);

if(num1>num2){

window.alert("num1>num2");

}else if(num1<num2){

window.alert("num1<num2");

}else{

window.alert(num1=num2);

}

</script>

</head>

<body>

</body>

</html>

#### 3.8介绍window.prompt和document.writeln()

**全局函数,需要查文档,查对象的方法.输入的都是字符串了,若是两个数字要进行相加需要看下边的方法**

[js05.html]示例

<html>

<head>

<title>js示例prompt的使用</title>

<script language="javascript">

var valA=window.prompt("请输入A值：");

var valB=window.prompt("请输入B值：");

document.writeln("a+b的值"+(parseFloat(valA)+parseFloat(valB)));

</script>

</head>

<body>

</body>

</html>

#### 3.9js基本语法--运算符

**逻辑运算符**

用于判断逻辑关系的运算符

**&&与、||或、!非**

var a=90; var b=90;

if(a==b||a>8){window.alert("ok1");}

b--;

if(a>b&&a>45){window.alert("ok2");}

if(!(a<=b)){window.alert("ok3");}

**在逻辑运算中，0、""、false、null、undefined、NaN均表示false,其他的对象或者非空字符串都是真**

&&与 例：

var a=90; var b=9;

if(a>b && a++>90){

window.alert("ok");

}else{

window.alert("no");

}

window.alert("a="+a);

案例说明：

if(逻辑表达式1&&逻辑表达式2){}

如果逻辑表达式1为true则js引擎继续执行逻辑表达式2；如果逻辑表达式1为false则不会执行逻辑表达式2。

||或

if(逻辑表达式1||逻辑表达式2||...){}

只要有一个逻辑表达式为true，整个表达式都为true。

**结论：||将返回第一个不为false那个值(对象亦可)，或者是返回最后一个值(如果全部都是false的话) ,此返回值多数情况下是布尔值，但也可能返回的是一个对象。**

var a=0; var b=""; var c=false;

var d=a||b||c;

window.alert(d);输出最后一个c的值false

!非

if(!逻辑表达式){}

如果逻辑表达式为true则取false。如果逻辑表达式为false则取true。

#### 3.10js基本语法--运算符

位运算和移位运算，其规范和java一致

var a=4>>2;

window.alert(a);结果1。

#### 3.11js基本语法--三大流程控制

##### 3.11.1顺序控制

程序可以顺序的执行。

var a=7;

window.alert("a="+a);

window.alert("hello!");

a++;

window.alert("a="+a);

window.alert("㊣㊣㊣");

a++;

window.alert("a="+a);

##### 3.11.2分支控制

让程序有选择的执行，分支控制有三种：

1、单分支；2、双分支；3、多分支

单分支语法：if(条件表达式){执行语句;}

双分支语法：if(条件表达式){执行语句;}else{执行语句;}

[js07.html]双分支演示

<html>

<head>

<title>js示例双分支控制演示</title>

<script language="javascript">

var age=window.prompt("请输入年龄：");

age=parseFloat(age);

if(age>=18){

window.alert("你年龄大于18、要对自己的行为负责!");

}else{

window.alert("小屁孩，快回家找妈妈!");

}

</script>

</head>

<body>

</body>

</html>

多分支语法：if(条件表达式1){语句;}else if(条件表达式2){语句;}else{语句;}

举例：如果男，男厕所，女，女厕所，否则不上

[js08.html]多分支示例

<html>

<head>

<title>js示例多分支控制演示</title>

<script language="javascript">

var sex=window.prompt("性别");

if(sex=="男"){

window.alert("上男厕所");

}else if(sex=="女"){

window.alert("上女厕所");

}else{

window.alert("不准上厕所");

}

</script>

</head>

<body>

</body>

</html>

##### 3.11.3多分支

switch(条件表达式){

case 常量1: 语句; break;

case 常量2: 语句; break;

...

default: 语句; break;

**注意：表达式可以是js支持的所有数据类型**

举例：a、b、c、d、e、f、g表示星期一、二、三...六、日

[js09.html]多分支示例

<html>

<head>

<title>js示例多分支控制演示</title>

<script language="javascript">

var date=window.prompt("输入abcdefg任意一个字母:");

switch(date){

case "a": window.alert("星期一"); break;

case "b": window.alert("星期二"); break;

case "c": window.alert("星期三"); break;

case "d": window.alert("星期四"); break;

case "e": window.alert("星期五"); break;

case "f": window.alert("星期六"); break;

case "g": window.alert("星期日"); break;

default: window.alert("输入有误!"); break;

}

</script>

</head>

<body>

</body>

</html>

结论：

1、js的switch语句数据类型可以是js支持的任何类型(数组和对象除外)

2、case后面的数据类型可以是任何类型(数组和对象除外),每一个case后边的类型都可以是不同类型的

3、break跳出整个switch

4、如果没有匹配则执行default.default若是卸载最前边是不执行的,java中也是这个样子的.

### 3.12js基本语法--三大流程控制

**循环控制有三种**

##### 3.12.1for循环基本语法：

for(循环初始值;循环条件;步长){

语句;//循环体

}

[js10.html]循环控制示例

<html>

<head>

<title>js示例循环控制示例</title>

**<script language="javascript">**

**for(var i=0;i<10;i++){**

**document.writeln("你好!<br>");**

**}**

**</script>**

</head>

<body>

</body>

</html>

##### 3.12.2while循环基本语法：

while(循环条件){

语句;//循环体

}

**特别说明：while循环是先判断再执行语句。**

例：

var i=0;

while(i<10){

document.writeln("hello!<br>");

i++;

}

##### 3.12.3.do while循环基本语法

do{

语句;//循环体

}while(循环条件);

**特别说明：do while循环是先执行，再判断。**

例：

var i=0;

do{

document.writeln("hello!<br>");

i++;

}while(i<10);

请编写一个程序，可以接收一个整数n，1、计算1+2+...n的值；2、1!+2!+...+n!的值

1、答案

var n=window.prompt("请输入一个整数");

n=parseInt(n);

var res=0;

for(var i=1;i<=n;i++){

res+=i;

}

document.writeln("题1结果是："+res);

2、答案

var res2=0;

var temp=1;

for(var i=1;i<=n;i++){

for(var j=1;j<=i;j++){

temp\*=j;

}

res2+=temp;

temp=1;

}

document.writeln("题2结果是："+res2);

### 3.13在ie中进行js调试

在ie8中通过开发人员工具，就可以进行js的调试工作。

请注意：首先要在IE-工具-internet选项-高级 中取消“禁止脚本调试”

对于低版本的ie浏览器，需要做下面的处理，才能进行js调试

1、安装Microsoft Script Editor，其是office2003中的一个组件，要安装Microsoft Script Editor，首先要有office的安装程序。安装步骤如下：

a、打开word工具-宏-Microsoft脚本编辑器，这时候会需要office的安装文件

b、执行完上一步后，在Microsoft Script Editor中，调试-安装web调试，这个时候还会需要office的安装文件。

c、完成安装后，重起IE和Microsoft Script Editor

2、调试

先决条件：

1、安装了Microsoft Script Editor或Microsoft Script debugger

2、IE中打开了脚本调试。

要在IE中进行JavaScript debugger调试，首先要在IE-工具-internet选项-高级 中取消“禁止脚本调试”，然后重起IE就可以使用Microsoft Script Editor。注意，IE较为不稳定，可能会出现该选项的情况。请关闭IE重来一遍。

如何使用Microsoft Script Editor进行调试：

在JavaScript代码中，在你想要设断点的位置，加上“debugger”，在已处处于debugger状态的Microsoft Script Editor代码窗口中设置断点。在IE中直接使用Microsoft Script Editor，请在IE地址栏中输入“javascript:debugger”回车。Microsoft Script Editor的具体使用方法请参考Microsoft Script Editor的帮助文档。

**使用firebug可以在firefox中调试js**

js代码发生错误，如何定位其错误？

1、可以通过程序员的经验来判断；

2、window.alert()来输出信息；

3、使用firebug的调试用具[cy--5，浏览器对象模型BOM]

**作业：**

[js11.html]图形作业

<html>

<head>

<title>js示例图形示例</title>

<script language="javascript">

var lay=window.prompt("输入图形高度：");

lay=parseInt(lay);

//直角三角形

for(var i=1;i<=lay;i++){

for(var j=1;j<=i;j++){

document.writeln("\*");

}

document.writeln("<br>");

}

document.writeln("<br>");

//金字塔

for(var i=1;i<=lay;i++){

for(var j=1;j<=lay-i;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=2\*i-1;k++){

document.writeln("\*");

}

document.writeln("<br>");

}

document.writeln("<br>");

//空心金字塔

for(var i=1;i<=lay;i++){

for(var j=1;j<=lay-i;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=2\*i-1;k++){

if(i==lay){

document.writeln("\*");

}else if(k==1||k==2\*i-1){

document.writeln("\*");

}else{

document.writeln("&nbsp");

}

}

document.writeln("<br>");

}

document.writeln("<br>");

//实心菱形

for(var i=1;i<=lay-1;i++){

for(var j=1;j<=lay-i;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=(i-1)\*2+1;k++){

document.writeln("\*");

}

document.writeln("<br>");

}

for(var i=1;i<=lay;i++){

for(var j=1;j<=i-1;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=(lay-i)\*2+1;k++){

document.writeln("\*");

}

document.writeln("<br>");

}

document.writeln("<br>");

//空心菱形

for(var i=1;i<=lay-1;i++){

for(var j=1;j<=lay-i;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=(i-1)\*2+1;k++){

if(k==1||k==(i-1)\*2+1){

document.writeln("\*");

}else{

document.writeln("&nbsp");

}

}

document.writeln("<br>");

}

for(var i=1;i<=lay;i++){

for(var j=1;j<=i-1;j++){

document.writeln("&nbsp");

}

for(var k=1;k<=(lay-i)\*2+1;k++){

if(k==1||k==(lay-i)\*2+1){

document.writeln("\*");

}else{

document.writeln("&nbsp");

}

}

document.writeln("<br>");

}

</script>

</head>

<body>

</body>

</html>

# 三.函数

## 3.1函数的基本概念

为完成某一功能的程序指令(语句)的集合，称为函数。

函数分为：自定义函数、系统函数(经常查看js帮助手册)

**js自定义函数基本语法：**

**function 函数名(参数列表){**

**//代码语句**

**return 值;//有无返回值是可以选择的。**

**}**

[js12.html]函数示例

<html>

<head>

<title>js示例函数示例</title>

<script language="javascript" type="text/javascript">//为了浏览器的兼容性必写

<!--

var num1=window.prompt("请输入第一个num：");

var num2=window.prompt("请输入第二个num：");

var operator=window.prompt("请输入运算符：");

num1=parseFloat(num1);

num2=parseFloat(num2);

//如何调用函数

document.write("res="+jiSuan(num1,num2,operator));

//自定义函数

function jiSuan(num1,num2,operator){ **//特别强调：参数名前不要带var**

var res=0;

if(operator=="+"){

res=num1+num2;

}else if(operator=="-"){

res=num1-num2;

}else if(operator=="\*"){

res=num1\*num2;

}else if(operator=="/"){

res=num1/num2;

}else if(operator=="%"){

res=num1%num2;

}

return res;//返回值

}

//-->

</script>

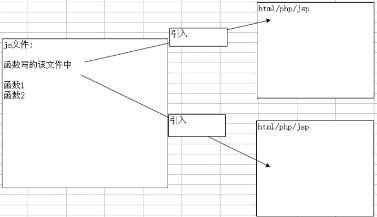
</head>

<body>

</body>

</html>

**如果其它多个html/php/jsp要使用自定义函数，必需引入。**



上面的函数单独提出，写到js文件中，然后在需要的html/php/jsp的地方引入

**js文件引入语法：**

**<script language="javascript" src="js的路径"></script>**

[js12\_1.html]函数引用示例

<html>

<head>

<title>js示例函数示例</title>

**<script language="javascript" src="js12\_1.js"></script>//引用js函数文件(写成两个<><>不要写成</>)**

<script language="javascript" type="text/javascript">//为了浏览器的兼容性必写

<!--

var num1=window.prompt("请输入第一个num：");

var num2=window.prompt("请输入第二个num：");

var operator=window.prompt("请输入运算符：");

num1=parseFloat(num1);

num2=parseFloat(num2);

//如何调用函数

document.write("res="+jiSuan(num1,num2,operator));

//-->

</script>

</head>

<body>

</body>

</html>

**[js12\_1.js]**

//自定义函数

function jiSuan(num1,num2,operator){ //特别强调：参数名前不要带var

var res=0;

if(operator=="+"){

res=num1+num2;

}else if(operator=="-"){

res=num1-num2;

}else if(operator=="\*"){

res=num1\*num2;

}else if(operator=="/"){

res=num1/num2;

}else if(operator=="%"){

res=num1%num2;

}

return res;//返回值

}

js中页面相互调用

为了完成a.html中可以使用到b.html中定义的函数，我们需要补充讲解html中页面之间共用js代码的知识点。

var str="window.alert("ok")";

**eval(str);//eval()系统函数可将字符串作为脚本执行。**

//escape和unescape配合使用，防止乱码产生。

var str1="顺平";

var str2=escape(str1);**//escape()系统函数用于将字符转码传输，防止浏览器显示乱码**

window.alert(str2);

var str3=unescape(str2);**//unescape()系统函数用于将escape转码的字符恢复。两者配合使用方式乱码的出现.**

window.alert(str3);

函数--函数的定义

有的程序员把函数称为方法，希望大家不要被这两个名词搞晕了。

function 函数名(参数列表){

语句;//函数(方法)主体

return 返回值;

}

**1、参数列表：表示函数的输入**

**2、函数主体：表示为了实现某一功能代码块**

**3、函数可以有返回值，也可以没有**

## 3.2函数的几种调用方式

1、函数名(传递给函数的参数1，参数2，...)

2、变量=函数名; (函数名后边不带参数不带括号,就是函数指针变换,带参数就是返回其值,没有返回就是underined)(诡异的用法,不要关心看懂就行)

此时变量就相当于函数的引用(指针)，可以这样调用函数：变量(实际参数..)

-------------------------用法示例-----------------------

alert(gettext()); //正常调用

function gettext(){

a = 1;

b = 2;

return 1+2;

}

abc = gettext

alert(abc()); //诡异的用法

alert(gettext); //将打印出函数体

-------------------------------------------------------------------

**特别说明：**

1、对于有返回值的函数调用，也可以在程序中直接使用返回的结果，

例如：alert("res="+sum(2,3));

2、没有返回值的函数，则返回underined。

**函数--调用过程(起新栈,然后弹出)**

函数--return语句

return 表达式;

abc(5);

//abc是一个函数，它接受数值

function abc(num1){

if(num1>3){

abc(--num1);//递归

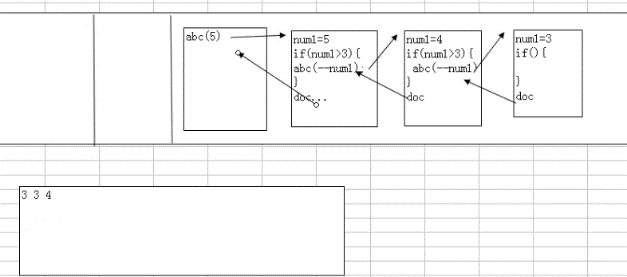
}

document.writeln(num1);

}

结果是：3 3 4

分析图：



## 3.3函数--深入使用

**1、函数的参数列表可以是多个；**

**2、参数列表可以是多个，并且数据类型可以是任意的类型；**

**3、js支持参数个数可变的函数；**

**4、js支持创建动态函数。**

//编写一个函数，可以接受任意多个数，并计算他们的和

abc2(45,90,900);

abc2(4,"hello,world");

function abc2(){

**//在js中有一个arguments,可以访问所有传入值.此时abc2(n1,n2)是可以的,但是用不到,所以js中是不支持函数重载的.**

//window.alert(arguments.length);

//遍历所有的参数

for(var i=0;i<**arguments**.length;i++){

window.alert(arguments[i]);

}

}

使用function类创建函数

创建动态函数的基本语法格式：

var varName=new function(argument1,...,lastArgument);

**说明：所有的参数都必需是字符串型的，最后的参数必需是这个动态函数的功能程序代码。**

例子：

<script language="javascript">

var square=new function("x","y","var sum;sum=x\*x+y\*y;return sum;");

alert(square(3,2));

var alsoDoSquare=doAdd;

alert(alsoDoSquare(3,2));

</script>

注意：闭包和函数的作用域及this还没有讲，必需放在面向对象中讲解。

## 3.4数组

为什么需要数组？

**使用数组，这种数据类型(引用类型/复杂类型/复合类型)，数组的基本概念：用于存放一组数据。**

**特别说明：js的数组不区分数据类型，可以存放任意数据类型。**

<html>

<head>

<title>js示例数组示例</title>

<script language="javascript" type="text/javascript">

<!--

var weights=[3,5,1,3.4,2,50];

var all\_weight=0;

var avg\_weight=0;

for(var i=0;i<weights.length;i++){

all\_weight+=weights[i];

}

avg\_weight=all\_weight/weights.length;

document.writeln("总体重是："+all\_weight+" "+"平均体重是："+**avg\_weight.toFixed(2)**);

//-->

</script>

</head>

<body>

</body>

</html>

### 3.4.1js基本语法--一维数组

js中的数组是用于表示一组数据的集合，它由一对方括号[]包围，数组的每个元素用逗号分隔，数组元素可以是任意类型的数据。

一维数组基本语法：

var 数组名=[元素值1,元素值2,...];

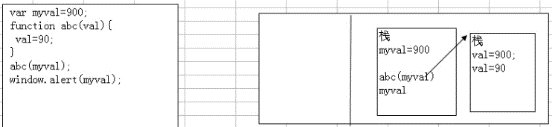
**1、创建数组**

var a=["shunping",123,"1.1",4.5,true];

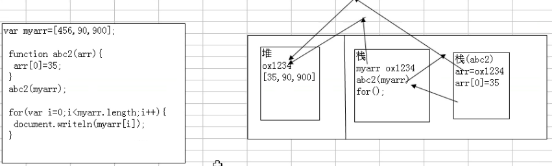
**2、数组在内存中的存在形式**

**3、js中的数组是引用传递(地址传递)**

函数调用传入值：输出结果为900。



函数调用传入引用地址：输出结果为35 90 900



### 3.4.2数组的引用(使用)

数组名[下标]

比如：

var a=[23,"hello",4.5]

你要使用a数组的第三个元素 a[2] 则输出4.5

如果访问a[3] 访问越界输出undefine

结论：不能访问不存在的元素。Undiefind

数组的下标是从0开始编号的。

### 3.4.3js数组允许动态增长

a=[45,true]

a[2]=56;//动态增长

for(var i=0;i<a.length;i++){

window.alert(a[i]); **//window打头的函数可以直接写不要window**

}

输出结果：45 true 56

### 3.4.4js数组的特点

**js数组是可以动态的增长的，这个和java的数组不一样，有点类似java中的ArrayList**

### 3.4.5js数组的几个重要应用

1、使用数组名.length可以得到数组大小；

2、拆分字符串 字符串.split(拆分标准)可以得到一个数组。

3、其它..

例：split和length的使用

var str="hello world abc 顺平";

var arr=str.split(" ",2); //2去前边的几个数据

for(var i=0;i<arr.length;i++){

document.writeln(arr[i]+",");

}

输出结果：hello,world

### 3.4.6遍历一维数组

遍历数组很多种方法，可以使用for循环，while循环遍历数组，其中for循环遍历除了常规用法外，还可以如下使用：**[此方法知道就好，不推荐使用]**

arr=[45,90,0]

arr["gg"]=9000; **//数组的下标可以是字符串,弱语言类似php也可以这样子写**

for(var key in arr){

window.alert(key+"="+arr[key]); **//key可以写成任何其他意义的字符串等**

}

**一维数组作业：**

在运动会上，五个小孩比赛滑轮，他们的划完100米，分别用了10s、12s、5.7s、9s、14s，请编写一个程序，计算他们的平均时间？

[js14.html]一维数组作业题

<html>

<head>

<title>js示例一维数组作业</title>

<script language="javascript" type="text/javascript">

<!--

var boy=prompt("请输入各个孩子的成绩，用空格隔开：");

var boy=boy.split(" ");//将字符串以空格分开取出字符串数组

var all=0;

var avg=0;

for(var i=0;i<boy.length;i++){

boy[i]=parseFloat(boy[i]);//将字符串数组值强转为Float类型

all+=boy[i];

document.writeln(boy[i]+" ");

}

avg=all/boy.length;

document.writeln("平均用时："+avg.toFixed(2));

//-->

</script>

</head>

<body>

</body>

</html>

### 3.4.7一维数组--小结

**1、数组可存放任意类型的数据；**

**2、数组大小不必事先指定，可以动态增长；**

**3、数组名可以理解为指向数组首地址的引用；**

**4、数组元素从0开始编号。**

### 3.4.8js基本语法--多维数组

**多维数组--二维数组**

多维数组我们只介绍二维数组，一维数组的元素还可以是一个数组，就构成二维数组。

#### 3.4.8.1 实例

var arr=[["shunping",123,4.5],["a","b","c"]];

分析

思考：如何访问到"b"，arr[0][2]又是多少？如何遍历二维数组？

arr[1][1]是b

arr[0][2]是4.5

[js15.html]遍历二维数组

<html>

<head>

<title>js示例二维数组遍历</title>

<script language="javascript" type="text/javascript">

<!--

var arr=[["shunping",123,4.5],

["a","b","c"]];

for(var i=0;i<arr.length;i++){

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

document.writeln(arr[i][j]);

}

document.writeln("<br>");

}

//-->

</script>

</head>

<body>

</body>

</html>

#### 3.4.8.2 案例，请用二维数组输出如下图形

0 0 0 0 0 0

0 0 1 0 0 0

0 2 0 3 0 0

0 0 0 0 0 0

[js16.html]遍历输出图形

<html>

<head>

<title>js示例二维数组显示图形</title>

<script language="javascript" type="text/javascript">

<!--

var arr=[[0,0,0,0,0,0],[0,0,1,0,0,0],[0,2,0,3,0,0],[0,0,0,0,0,0]];

for(var i=0;i<arr.length;i++){

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

document.writeln(arr[i][j]+"&nbsp;");

}

document.writeln("<br>");

}

//-->

</script>

</head>

<body>

</body>

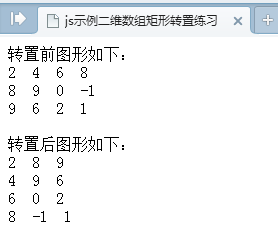
</html>

#### 3.4.8.3 多维数组练习：矩形转置

2 4 6 8

8 9 0 -1

9 6 2 1



[js17.html]矩形转置示例

<html>

<head>

<title>js示例二维数组矩形转置练习</title>

<script language="javascript" type="text/javascript">

<!--

var arr=[[2,4,6,8],[8,9,0,-1],[9,6,2,1]];

var arr\_new=[];//定义一个新数组

document.writeln("转置前图形如下：<br>");

for(var i=0;i<arr.length;i++){

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

document.writeln(arr[i][j]+"&nbsp;");

arr\_new[j]=[0];//初始化arr\_new数组

}

document.writeln("<br>");

}

document.writeln("<br>");

document.writeln("转置后图形如下：<br>");

//对arr数组中的值进行矩形交换到新数组arr\_new中

for(var i=0;i<arr.length;i++){

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

arr\_new[j][i]=arr[i][j];

}

}

//遍历新数组arr\_new输出结果

for(var i=0;i<arr\_new.length;i++){

for(var j=0;j<arr\_new[i].length;j++){

document.writeln(arr\_new[i][j]+"&nbsp;");

}

document.writeln("<br>");

}

//-->

</script>

</head>

<body>

</body>

</html>

### 3.4.9js基本语法--排序

排序的介绍

排序是将一组数据，依指定的顺序进行排列的过程。

排序的分类：

**(交换式排序法、选择式排序法和插入式排序法)**

排序(Sorting)是数据处理中一种很重要的运算，同时也是很常用的运算，一般数据处理工作25%的时间都在进行排序。简单地说，排序就是把一组记录(元素)按照某个域的值的递增(即由小到大)或递减(即由大到小)的次序重新排列的过程。

交换式排序法

交换式排序属于内部排序法，是运用数据值比较后，依判断规则对数据位置进行交换，以达到排序的目的。

交换式排序法又可以为两种：

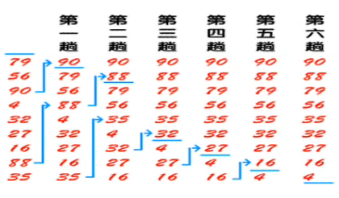
1、冒泡排序法(Bubble sort)；

2、快速排序法(Quick sort)。

#### 3.4.9.1 交换式排序法--冒泡排序法

冒泡排序(Bubble Sorting)的基本思想是：通过对待排序序列从后向前(从下标较大的元素开始)，依次比较相邻元素的排序码，若发现逆序则交换，使排序码较小的元素逐渐从后部向前部(从下标较大的单元移向下标较小的单元)，就象水底下的气泡一样逐渐向上冒。

因为排序的过程中，各元素不断接近自己的位置，如果一趟比较下来没有进行过交换，就说明序列有序，因此要在排序过程中设置一个标志flag判断元素是否进行过交换。从而减少不必要的比较。



[js18.html]冒泡排序示例

<html>

<head>

<title>js示例排序冒泡排序法</title>

<script language="javascript" type="text/javascript">

<!--

var arr=[79,56,90,4,32,27,16,88,35];

var temp=0;

var flag=false;

for(var i=0;i<arr.length-1;i++){

for(var j=0;j<arr.length-1-i;j++){

if(arr[j]<arr[j+1]){

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

flag=true;

}

}

if(flag){

falg=false;

}else{

break;

}

}

document.writeln(arr);

//-->

</script>

</head>

<body>

</body>

</html>

### 3.4.10 js基本语法--查找

查找：常用的查找有两种

1、顺序查找；

2、二分查找。

**二分查找有一个前提，该数组是有序的，如果不是有序的数组，则不能使用二分查找。**

思路：找到数组的中间数(midVal)，和你要查找的数(findVal)进行比较，如果midVal>findVal则说明findVal在数组的左边，就把该数组二分(就只在左边查找)

[js19.html]二分查找法示例

<html>

<head>

<title>js示例二分查找</title>

<script language="javascript" type="text/javascript">

<!--

var arr=[32,52,76,82,104,372];

//测试

binarySearch(arr,52,0,arr.length-1);

function binarySearch(arr,findVal,leftIndex,rightIndex){

//防止无穷递归

if(leftIndex>rightIndex){

//提示找不到

document.writeln("找不到");

return;

}

//找到中间这个值

var midIndex=Math.floor((leftIndex+rightIndex)/2);

var midVal=arr[midIndex];

//比较

if(midVal>findVal){

//在左边找

binarySearch(arr,findVal,leftIndex,midIndex-1);

}else if(midVal<findVal){

//在右边找

binarySearch(arr,findVal,midIndex+1,rightIndex);

}else{

document.writeln("找到下标为："+midIndex);

return;

}

}

//-->

</script>

</head>

<body>

</body>

</html>

# 四.js面向对象特征介绍

javascript是一种面向(基于)对象的动态脚本语言，是一种基于对象(Object)和事件驱动(EventDirven)并具有安全性能的脚本语言。他具有面向对象语言所特有的各种特性，比如封装、继承及多态等。但对于大多数人说，我们只把javascript做为一个函数式语言，只把他用于一些简单的前端数据输入验证以及实现一些简单的页面动态效果等，我们没能完全把握动态语言的各种特性。在很多优秀的Ajax框架中，比如ExtJS、JQuery等，大量使用了javascript的面向对象特性，要使用ext技术，javascript的高级特性，面向对象语言特性是我们必需完全把握的。

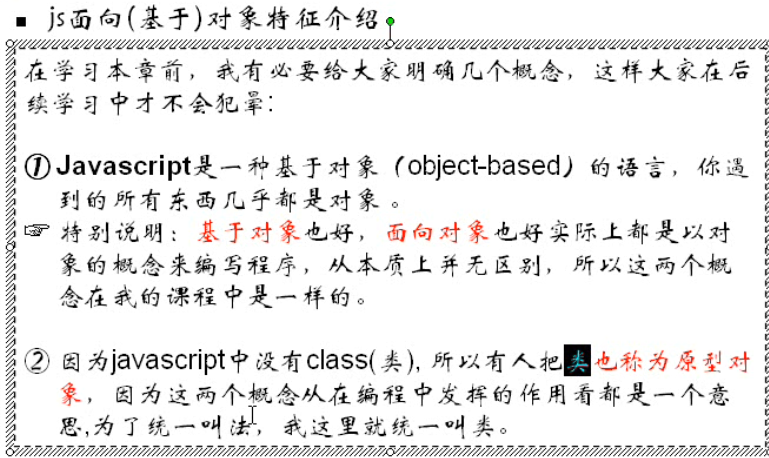
## 4.1js面向(基于)对象特征介绍

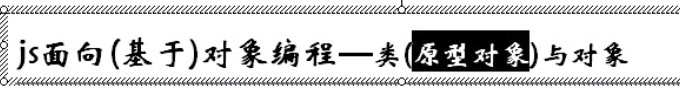
明确概念：

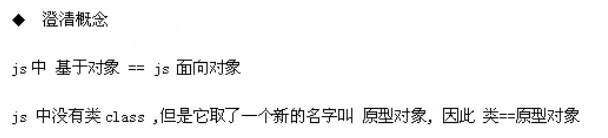
1、javascript是一种基于对象(Object-based)的语言，你遇到的所有东西几乎都是对象。

特别说明：基于对象也好，面向对象也好实际上都是以对象的概念来编写程序，从本质上并无区别，所有这两个概念在我的课程中是一样的。

1. 因为javascript中没有class(类)，所以有人把类也称为原型对象，因为这两个概念从在编程中发挥的作用看都是一个意思，为了统一叫法，我这里就统一叫类。







## 4.2为什么需要面向对象？

一个问题？

张老太养了两只猫猫：一只名字叫小白，今年3岁，白色。还有一只叫小花，今年10岁，花色。请编写一个程序，当用户输入小猫的名字时，就显示该猫的名字，年龄，颜色。如果用户输入的小猫的名错误，则显示张老太没有这只猫猫。

[js20.html]面向对象示例

<html>

<head>

<title>js示例面向对象示例</title>

<script language="javascript" type="text/javascript">

<!--

//这就是猫类，js中使用function定义原型对象(类)

function Cat(){

}

**//Cat();//直接使用则为函数**

**var cat1=new Cat();//这样使用即为类，这时cat1就是一个对象(实例)**

cat1.name="小白";

cat1.age=3;

cat1.color="白色";

**//从上面的代码我们可以看出**

**//1、js中的对象的属性可以动态添加。**

**//2、属性没有限制。**

window.alert(cat1.name);

//-->

</script>

</head>

<body>

</body>

</html>

## 4.3js面向(基于)对象

计算机语言的发展是向接近人的思维方式演变的，这是一个大趋势。js引入的面向对象的思想。js不是纯面向对象的，我们可以认为它是基于面向对象的。

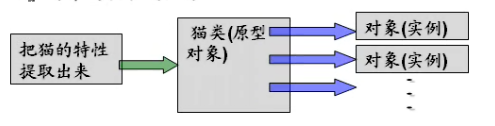
**汇编语言[面向机器]**

**C语言[面向过程]**

**java语言[面向对象]**

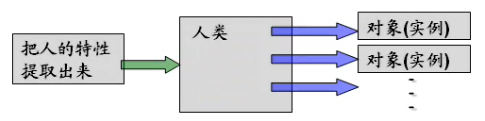
**js[支持面向对象]**

类和对象(实例)的关系



注意：从猫类到猫对象实例，目前有几种说法：1、创建一个对象实例；2、实例化一个对象；3、对象实例化...以后大家听到这些说法，不要迷糊。

当然，上面的猫也可是鱼、狗、人...js支持面向对象编程。



**类(原型对象)和对象(实例)的区别和联系**

**1、类(原型对象)是抽象的、概念的，代表一类事物，比如人、猫..**

**2、对象是具体的，实际，代表一个具体事物**

**3、类(原型对象)是对象实例的模板，对象实例是类的一个个体。**

**类(原型对象)--如何定义类(原型对象)和对象**

**1、工厂方法--使用new Object创建对象并添加相关属性；**

**2、使用构造函数来定义类(原型对象)；**

**3、使用prototype；**

**4、构造函数及原型混合方式；**

**5、动态原型方式。**

### 4.3.1类(原型对象)--如何定义类(原型对象)

基本语法：

function 类名/原型对象(){

}

对象的属性

对象的属性，一般是基本数据类型(数，字符串)，也可是另外的对象。比如我们前面创建猫对象的age就是猫对象的属性。

### 4.3.2创建对象基本语法：

var 对象名=new 类名();

### 4.3.3对象--特别说明

#### 1.在js中一切都是对象

function Person(){}

var a=new Person();

window.alert(a.constructor); //a对象实例的构造函数

window.alert(typeof a); //a的类型是什么,是Object

var b="1234";

window.alert(b.constructor); //b对象实例的构造函数

window.alert(typeof b); //b的类型是什么,是String

**思考：如何判断一个对象实例是不是Person类型？两种方法**

if(a instanceof Person){

window.alert("a是Person");

}

if(a.constructor==Person){

window.alert("a是Person");

}

#### 2.对象function特别说明：

**1、在js中一切都是对象；**

**2、类(原型对象)其实也是对象，它实际上是function类的一个实例，通过类我们可以创建自己的对象实例，可以通过下面的代码来说明。**

function Person(){}

var a=new Person();

window.alert(Person.constructor);//Person的构造函数

### 4.3.4对象实例--如何创建对象实例

**var 对象名=new 类名/原型对象名();**

**补讲定义变量带var和不带var的区别：(注意和3.4.1之间的区别)**

var abc=89;//全局变量

function test(){

//在函数里，如果不带var，则表示使用全局的abc变量；

//如果带上var则表示在test()函数中，定义一个新的abc变量。

abc=900; //如果改写成var abc=900;下面的输出结果为89。

}

test();

window.alert(abc); //输出结果为900。

### 4.3.5对象实例--如何访问(使用)对象实例的成员变量

1、对象实例名.属性名;

2、对象实例名["属性名"];

对象实例名["变量名"]的方式**可以实现动态的访问变量**，如：

function Person(){}

var p1=new Person();

p1.name="顺平";

window.alert(p1["name"]+" "+p1.name);

思考题：

加深印象，定义一个人类(Person)(包括名字、年龄)

看下面一段代码：

var a=new Person();

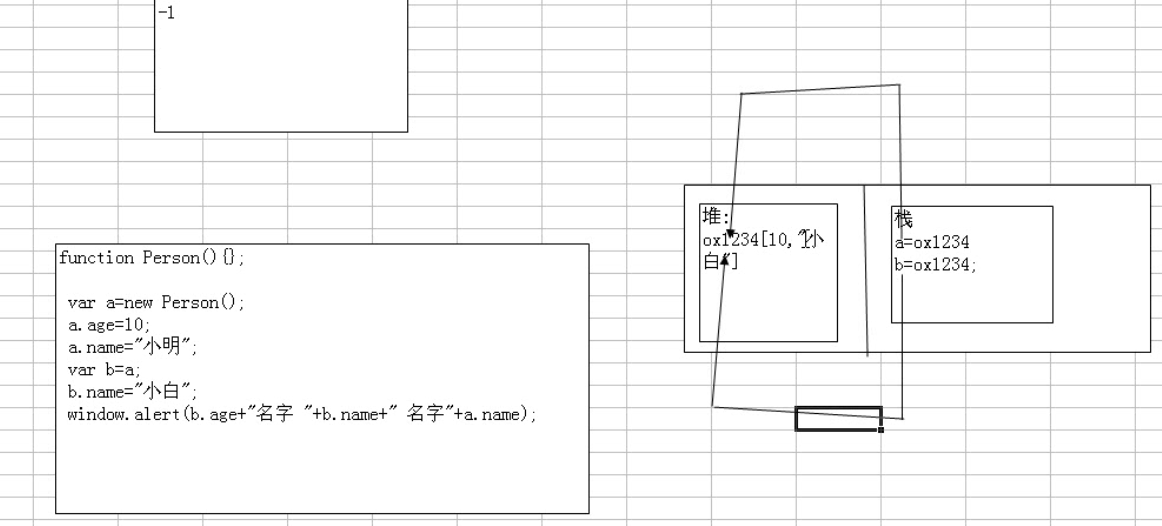
a.age=10;

a.name="小明";

var b=a;

alert(b.age);

请问：b.age究竟是多少？是10

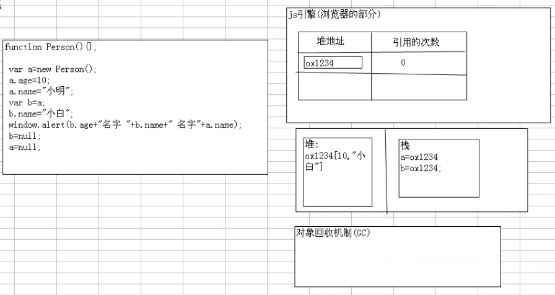


两个都是小白,引用机制.

### 4.3.6对象引用问题的说明：

**js提供了二种方式进行内存回收机制：被动回收机制与主动回收机制。**

**被动回收机制**在于堆地址被引用的次数，当引用次数为0时，对象回收机制GC就将无引用的堆地址空间进行释放。



function Person(){};

var a=new Person();

a.age=10;

a.name="小明";

var b=a;

b.name="小白";

window.alert(b.age+"名字"+b.name+" 名字"+a.name);

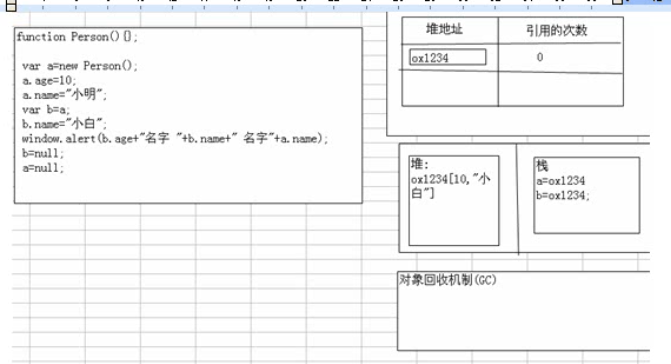
b=null;

window.alert(a.age+"a名字"+a.name);

window.alert("b.age="+b.age);

a=null;

**//第一次b置为null,只是断了b的引用但是对象还在,a置为null,对象应用为0,也被回收了**

****

**主动回收机制**

function Person(){};

var a=new Person();

a.age=10;

a.name="小明";

var b=a;

b.name="小白";

delete a.age;

**//删除a对象的属性,原型对象的属性删掉了,b当然也没法使用了.不能直接删除对象,只能一个个的删除的属性**

window.alert(b.age+"名字"+b.name);

**delete 对象名.属性名; //这样就会释放 对象指向的属性空间**

小思考题

在明白对象是如何在内存中存在后，请大家再看下面的思考题，请问会输出什么信息？

var a=new Person();

a.age=10;

a.name="小明";

var b=a; //连个引用指向了同一个对象

document.wrtie(b.name); //输出小明

b.age=200;

document.wrtie(a.age); //输出200

### 43.7this--问题提出

看一段代码：

function Person(){}

var p1=new Person();

p1.name="顺平";

p1.age=60;

window.alert(p1.name+" "+p1.age);

var p2=new Person();

window.alert(p2.name); //这里会输出什么？(undefine)

在实际编程中，我们可能有这样的需求，当我们创建一个对象后，就希望该对象自动的拥有某些属性[比如：当我们创建一个Person对象后，就希望该对象自动拥有name和age属性，这又怎么办呢？]

使用this来解决

function Person(){

this.name="abc";

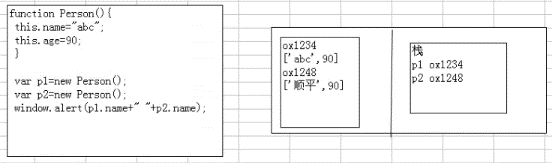
this.age=90;

}

var p1=new Person();

var p2=new Person(); //p1中name改变不影响p2中的name

window.alert(p1.name+" "+p2.name);



**js虚拟机会给每个对象分配this，代表当前对象。坦白的讲，要明白this(我的)不是件容易的事。**

function Person(){

var name="abc"; **//由于name和age在类中通过var来定义，导致name和age为类中的私有变量，所以无法通过引用方式来访问类中私有的成员变量。只能是内部方法进行访问**

var age=900;

this.name2="abc2"; **//this.name2表示name2这个属性是公开的。可以通过实例.属性来引用**

**//这个就是Person类的一个公开方法,类中所有公开的方法都必需通过this来定义**。

this.show=function(){

window.alert(name+" "+age);

}

**//这时Person类中的私有方法,只能在Person类中使用。**

function show2(){

window.alert("show2()"+name+" "+age);

}

}

var p1=new Person();

window.alert(p1.name+" "+p1.age+" "+p1.name2);//出现无法访问的情况，提示undefine

p1.show();

p1.show2();//这里会报错。

**特别说明：**

**1、js中一般在类中不定义私有变量或私有方法。**

**2、类中的变量和方法定义为公开的都必需使用this来定义。**

#### 4.3.7.1 进一步理解this

**记住一句话：哪个对象实例调用this所在的函数，那么this就代表哪个对象实例。**

function test1(){

alert(this.v); //输出90

}

var v=90;

test1(); //等价于window.test1();

function Person(){

this.abc=function(){

window.alert(this.v); //输出hello

}

}

var p=new Person();

p.v="hello";

p.abc();

#### 4.3.7.2 this--注意事项

this不能在类定义的外部使用，只能在类定义的方法中使用。

## 4.4对象--对象(成员)函数的初步介绍

比如：我们希望对象不但有属性，还希望他有行为。(行为在程序中靠函数完成)

在某些情况下，我们需要定义对象函数。比如人对象：除了有一些属性外(成员变量表示的年龄、姓名..)，我们人对象还有一些行为比如：可以说话、跑步..通过学习，我们人还可以做算术题。这时就要用函数才能完成，现在要求对Person对象完善。

1、添加speak函数，输出我是一个好人

2、添加jisuan函数，可以计算从1+..+1000的结果

3、修改jisuan函数，该方法可以接收一个n，计算从1+..+n的结果

4、添加add成员函数，可以计算两个数的和

function Person(name,age){

//这个就是使用传入的实际参数，去初始化属性

this.name=name;

this.age=age;

//输出自己的名字,this.show就是公开的函数,函数名是show

this.show=function(){

document.write("名字="+this.name);

}

this.jisuan=function(n){

var res=0;

for(var i=1;i<=n;i++){

res+=i;

}

return res;

}

}

var p1=new Person("宋江",90);

p1.show;

var p2=new Person("林冲",12);

p2.show;

document.write("<br> res="+p1.jisuan(100));

## 4.5对象--成员函数

给一个对象添加(指定)函数的几种方式

#### 4.5.1方式一语法：

function 类名(){

this.属性;

}

var 对象名=new 类名();

function 函数名(){

//执行代码

}

对象名.属性名=函数名; //这样就相当于把函数赋给对象名.属性名,此时这个属性名就表示一个函数

对象名.属性名();//调用

方式1

function Person(){

this.name="abc";

this.age=900;

}

function show1(){

window.alert("hello"+this.name);

}

var p1=new Person();

p1.abc=show1;

p1.abc();

#### 4.5.2方式二语法：

对象名.属性名=function(参数列表){

//代码

}

对象名.属性名(实际参数);//调用

方式2

function Person(){

this.name="abc";

this.age=900;

}

var p1=new Person();

p1.abc=function show1(){

window.alert("hello"+this.name);

}

p1.abc();

思考题：对第一个案例

1、如果window.alert(p1.abc)会输出什么？

function show1(){

window.alert("hello"+this.name);

}

2、如果window.alert(show1)会输出什么？

function show1(){

window.alert("hello"+this.name);

}

3、对象--成员函数

function Person(){

this.name="abc1";

this.age=900;

}

function show1(){

window.alert("hello"+this.name);

}

var p1=new Person();

p1.abc=show1;

show1();

输出结果：hello 而后边的确实是undiefind,js处理了没有打印出来

function Person(){

this.name="abc1";

this.age=900;

}

var name="北京";

function show1(){

window.alert("hello"+this.name);

}

var p1=new Person();

p1.abc=show1;

show1();

输出结果：hello北京

4、对象--成员函数

给所有对象添加(指定)函数

function Person(){

this.name="abc";

this.age=900;

this.abc=function(v1,v2){

window.alert(this.name+" "+this.age+" "+v1+" "+v2);

}

}

var p1=new Person();

p1.abc(); //输出abc 900 undefine undefine

p1.abc("北京","天津"); //输出abc 900 北京 天津

var p2=new Person();

p2.abc(); //输出abc 900 undefine undefine

p2.abc("南京","东京"); //输出abc 900 南京 东京

-------------------------------------------------------------------------------------------------------------------------

5、对象--成员函数

给所有对象添加(指定)函数

思考题，看看输出什么，从中我们可以得出什么结论？

function Person(){

this.name="abc";

this.age=900;

this.abc=function(v1,v2){

window.alert(this.name+" "+this.age+" "+v1+" "+v2);

}

}

var p1=new Person();

p1.name="中国";

p1.abc();//输出中国 900 undefine undefine

p1.abc("北京","天津");//输出中国 900 北京 天津

var p2=new Person();

p2.abc("南京","东京");//输出中国 900 南京 东京

对象--成员函数

再看使用另外一种方法来给类增加方法

#### 4.5.3方式三

前面的几种方法有一个问题：那就是每个对象独占函数代码，如果对象很多则会影响效率。

js设计者，给我们提供另一个方法，原型法：这样多个对象可以共享函数。

function Dog(){}

var dog1=new Dog();

dog1.shout=function(){

window.alert("小狗");

}

dog1.shout();

var dog2=new Dog(){}

dog2.shout();//这里报错

function Dog(){}

var dog1=new Dog();

//使用prototype[类]去绑定一个函数给shout

Dog.prototype.shout=function(){

window.alert("小狗");

}

dog1.shout();

var dog2=new Dog();

dog2.shout();//这里ok

-------------------------------------------------------------------------------

## 4.6 补讲==号的作用

1、当==的两边都是字符串的时候，则比较内容是否相等；

2、如==的两边是数字、则比较数的大小是否相等；

3、如==的两边是对象或是对象函数则比较地址是否相等。

## 4.7对象Object类

Object类是所有javascript类的基类，提供了一种创建自定义对象的简单方式，不需要程序员再定义构造函数。

**主要属性：**

constructor--对象的构造函数

prototype--获得类的prototype对象，static性质。

**主要方法：**

hasOwnProperty(property)--是否属于本类定义的属性

isPrototypeOf(object)--是否是指定类的prototype

propertyIsEnumerable(property)--是否可例举的属性

toString()--返回对象对应的字符串

valueOf()--返回对象对应的原始类型值

<script language="javascript">

function getAttributeValue(attr){

alert(person[attr]);

}

var person=new Object();

person.name="zs";

person.age=18;

getAttributeValue("name");

getAttributeValue("age");

</script>

[js21.html]Object类的示例

<html>

<head>

<title>js示例Object类</title>

<script language="javascript" type="text/javascript">

<!--

//初步体验Object类

var p1=new Object();//通过Object直接创建对象

p1.name="sp";

window.alert(p1.constructor);

//-->

</script>

</head>

<body>

</body>

</html>

对象--成员函数(方法)

加深对类和对象的认识

如何给类添加方法(如何给某类型的所有对象添加方法)

思考题：

var i=new Number(10);//等价于var i=10;

//我们可以给类添加方法

Number.prototype.add=function(a){//所有Number类都有add方法

return this+a;

}

window.alert(i.add(10).add(30));//结果为50

var b=90;

window.alert(b.add(40));//结果为130

上面的写法我们成为对类的修改和扩展。

请思考给js的Array对象扩展一个find(val)方法，当一个Array对象调用该方法的时候，如果能找到val则返回其下标，否则返回-1

//体验一下Array

var arr1=new Array(3);

arr1[0]="George";

arr1[1]="John";

arr1[2]="Thomas";

//遍历该数组

for(var i=0;i<arr1.length;i++){

document.write(arr1[i]+"&nbsp;");

}

document.write("<br>");

//使用Array提供的方法，颠倒数据

arr1.reverse();

for(var i=0;i<arr1.length;i++){

document.write(arr1[i]+"&nbsp;");

}

//现在我们一起看看如何给所有Array对象添加一个find(val)方法

Array.prototype.find=function(val){

//遍历数组this

for(var i=0;i<this.length;i++){

if(val==this[i]){

return i;

}

}

return -1;

}

document.write("John下标为："+arr1.find("John"));

**什么是闭包？(此知识点在讲封装时再讲。)**

闭包(closure)

function a(){

var i=0;

function b(){

alert(++i);

}

return b;

}

var c=a();

c();

闭包的作用就是在a执行完并返回后，闭包使得javascript的垃圾回收机制GC不会收回a所占用的资源，因为a的内部函数b的执行需要依赖a中的变量。

javascript闭包就是在另一个作用域中保存了一份它从上一级函数或作用域取得的变量(键值对)，而这些键值对是不会随上一级函数的执行完成而销毁。这样在执行完var c=a()后，变量c实际上是指向了函数b,b中用到了变量i，再执行c()后就会弹出一个窗口显示i的值(第一次为1)。这段代码其实就创建了一个闭包，为什么？因为函数a外的变量c引用了函数a内的函数b就是说：当函数a的内部函数b被函数a外的一个变量引用的时候，就创建了一个我们通常所谓的“闭包”。

**成员函数--细节说明**

1、函数的参数列表可以是多个

案例在Person类中编写一个成员方法(面向对象方式)，从键盘输入三个数，返回最大的那个数。

参数列表可以是多个，并且数据类型可以是任意的类型，整数、小数、类

function 函数名(参数列表){

语句;//函数主体

return 返回值;

}

2、函数可以没有返回值，但最多只能有一个返回值。

返回类型可以是任意的数据类型(整数、小数、字符串)也可没有返回值

function 函数(参数列表){

语句;//函数主体

return 返回值;

}

**js中的函数还有以下两点特别说明：**

1、js支持参数个数可变的函数；

2、js支持创建动态函数，这个用的比较少，了解即可。

3、js中不支持函数的重载，具体案例

function test(a,b){

window.alert("hello");

}

function test(a){

window.alert(a);

}

function test(a,b){

window.alert(a+" "+b);

}

window.test(3,"hello");

结论：js在调用函数的时候，是根据函数名来调用，如果有多个函数名相同，则以最后一个出现的函数来执行。

请看下面一段代码(可变参数)

function abc(a,b){

return a+b;

}

function abc(a,b,c){

return a+b+c;

}

window.alert(abc(1,2));//报错

window.alert(abc(1,2,3));//输出6

function abc(){

var s=0;

for(var i=0;i<arguments.length;i++){

s+=arguments[i];

}

return s;

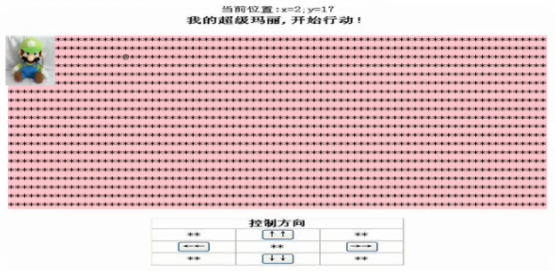
}

window.alert(abc(1,2));//输出3

window.alert(abc(7,8,9));//输出24

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

面向对象编程的综合案例



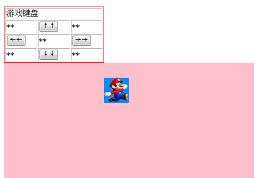
Mario游戏分析

1、如何通过按钮控制图片的位置

2、设计相关的对象

要求：1、Mario碰到边界给一个提示

2、Mario可以去找另外一个物体



[js22.html]Mario示例 **注意：在IE8以下浏览器测试成功**

<html>

<head>

<title>js示例玛里奥综合示例</title>

<meta name="keywords" content="keyword1,keyword2,keyword3">

<meta name="description" content="this is my page">

<meta name="content-type" content="text/html; charset=">

<link rel="stylesheet" type="text/css" href="./js22css.css">

<script language="javascript" type="text/javascript">

<!--

//设计Mario类

function Mario(){

this.x=0;

this.y=0;

//移动0上1右2下3左

this.move=function(direct){

switch(direct){

case 0:

//这里为了改变img的left和top,我们需要得到img元素(对象)

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

var top=mymario.style.top;

top=parseInt(top.substr(0,top.length-2));

mymario.style.top=(top-10)+"px";

if(top<0||top>400){

alert("mario越界了!")

}

break;

case 1:

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

var left=mymario.style.left;

left=parseInt(left.substr(0,left.length-2));

mymario.style.left=(left+10)+"px";

if(left<0||left>500){

window.alert("mario越界了!")

}

break;

case 2:

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

var top=mymario.style.top;

top=parseInt(top.substr(0,top.length-2));

mymario.style.top=(top+10)+"px";

if(top<0||top>400){

window.alert("mario越界了!")

}

break;

case 3:

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

var left=mymario.style.left;

left=parseInt(left.substr(0,left.length-2));

mymario.style.left=(left-10)+"px";

if(left<0||left>500){

window.alert("mario越界了!")

}

break;

}

}

}

//创建Mario对象

var mario=new Mario();

//全局函数

/\* function marioMove(direct){

switch(direct){

case 0:

mario.move(direct);

break;

case 1:

mario.move(direct);

break;

case 2:

mario.move(direct);

break;

case 3:

mario.move(direct);

break;

}

}\*/

//-->

</script>

</head>

<body>

<table border="1px" class="controlcenter">

<tr><td colspan="3">游戏键盘</td></tr>

<tr><td>\*\*</td><td><input type="button" value="↑↑" onclick="mario.move(0)"/></td><td>\*\*</td></tr>

<tr><td><input type="button" value="←←" onclick="mario.move(3)"/></td><td>\*\*</td><td><input type="button" value="→→" onclick="mario.move(1)"/></td></tr>

<tr><td>\*\*</td><td><input type="button" value="↓↓" onclick="mario.move(2)"/></td><td>\*\*</td></tr>

</table>

<div class="gamediv">

<img id='mymario' style="width:50px;left=100px;top=50px;position:absolute;"src="mario.jpg" />

</div>

</body>

</html>

[js22css.css]选择器文件

.gamediv{

width: 500px;

height: 400px;

background-color: pink;

position: absolute;

}

/\*表格样式\*/

.controlcenter{

width: 200px;

height: 100px;

border: 1px solid red;

}

问题：

1、怎么直接获取CSS的内容？

IE中使用obj.currentStyle方法；

FF中使用getComputedStyle方法。

js面向(基于)对象编程--构造方法(函数)

构造函数(方法)介绍

什么是构造函数呢？在回答这个问题之前，我们来看一个需求：前面我们在创建人类

的对象时，是先把一个对象创建好后，再给他的年龄和姓名属性赋值，如果现在我要求，

在创建人类的对象时，就直接指定这个对象的年龄和姓名，该怎么做？

构造函数(方法)是一种特殊的方法，它的主要作用是完成对对象实例的初始化。它有几个特点：

1、构造函数(方法)名和类名相同。

2、在创建一个对象实例时，系统会自动的调用该类的构造方法完成对新对象的初始化。

**构造函数基本用法：**

function 类名(参数列表){

属性=参数值;//不带this为私有，带有this为共有。

}

案例：在创建一个Person实例的时候，就指定一个成员方法，该方法有三个参数，两个数和运算符号(+,-,\*,/)根据用户的输入可以得到相应的结果。

在给一个对象初始化属性值的时候，也可以指定函数属性

[js23.html]构造函数传方法示例

<html>

<head>

<title>js示例构造函数可传方法</title>

<script language="javascript" type="text/javascript">

<!--

function jiSuan(num1,num2,oper){

if(oper=="+"){

return num1+num2;

}else if(oper=="-"){

return num1-num2;

}else if(oper=="\*"){

return num1\*num2;

}else{

return num1/num2;

}

}

function Person(name,age,fun){

this.name=name;

this.age=age;

this.myfun=fun;

}

var p1=new Person("aa",9,jiSuan);

window.alert(p1.name);

window.alert(p1.myfun(89,90,"+"));

//-->

</script>

</head>

<body>

</body>

</html>

**构造方法(函数)小结**

1、构造方法名和类名相同；

2、主要作用是完成对新对象实例的初始化；

3、在创建对象实例时，系统自动调用该对象的构造方法。

类定义的完善：

在介绍了构造函数(方法)后，我们对象的定义就可以完善一步：

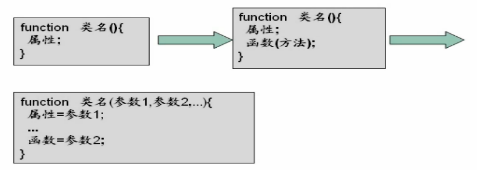
function 类名(){ function 类名(){ function 类名(参数1,参数2,..){

属性; → 属性; → 属性=参数1;

} 函数(方法); ...

} 函数=参数2;

}



面向对象编程进一步认识

**创建对象的又一种形式：**

1、对于比较简单的对象，我们也可以这样来创建

var dog={name:'小明'};

window.alert(dog.name);

var dog={name:'小明',sayHello:function(a,b){window.alert("结果="(a+b));}};

window.alert(dog.name);

window.alert(dog["name"]);//一样可以访问

dog.sayHello(45,90);

面向对象编程进一步认识

有时，我会看到这样一种调用方法

函数名.call(对象实例);

//这样调用该函数的this就指向了对象实例

var dog={name:'hello'};

function test(){

window.alert(this.name);

}

test();

window.test();

**test.call(dog);**//相当于去调用test(),同时this执行dog对象。

new test();

**面向对象编程进一步认识--for...in**

var dog={name:'小时',sayHello:function(a,b){window.alert("结果="+(a+b));}};

//循环列出dog对象的所有属性和方法 对象名['属性名']

for(var key in dog){

window.alert(dog[key]);

}

//循环列出window对象的所有属性和方法

for(var key in window){

document.writeln(key+":"+window[key]+"<br>")0;

}

**js面向(基于)对象编程--三大特征**

**抽象**

在讲解面向对象编程的三大特征前，我们先了解什么叫抽象，在定义一个类时候，实际上就是把一类事物的共有的属性和行为提取出来，形成一个物理模型(模板)。这种研究问题的方法称为抽象。

**封装--什么是封装**

封装就是把抽象出来的属性和对象属性的操作封装在一起，属性被保护在内部，程序的其它部分只有通过授权的操作(函数)，才能对属性进行操作。

**封装--访问控制修饰符**

电视机的开关，对音量，颜色，频道的控制是公开的，谁都可以操作，但是对机箱后盖，主机板的操作却不是公开的，一般由专业维修人员来玩。

那么在js中如何实现这种类似的控制呢？

[js24.html]封装示例

<html>

<head>

<title>js示例封装示例</title>

<script language="javascript" type="text/javascript">

<!--

function Person(name,agei,sal){

this.name=name;//公开的属性

var age=agei;//私有属性

var salary=sal;//私有属性

//在类中如何定义公开的方法(特权方法)，私有方法(内部方法)

//如果我们希望操作私有的属性，则可以通过公开的方法去实现

this.show=function(){

window.alert(age+" "+salary);

}

//私有方法,可以访问类中的对象属性(不论公开的还是私有的)

function show(){

window.alert(age+" "+salary);

}

}

var p1=new Person('sp',20,50000);

window.alert(p1.name);

p1.show();

//-->

</script>

</head>

<body>

</body>

</html>

封装

**js提供有以下几种控制方法和属性的访问权限**

1、公开级咖哩 ：对外公开

2、私有级别：类本身可以访问，不对外公开

function Person(pname,page){

this.name=pname;//公开属性(变量)

var age=page;//私有属性(变量)

//私有方法

function fun1(){

window.alert("hello");

}

//公开函数(特权函数)

this.fun2=function(){

window.alert(this.name+" "+age);

fun1();

}

}

请思考输出什么结果，为什么？

var p1=new Person("顺平",90);

p1.fun2();//正常输出

window.alert(p1.name);//正常输出

//p1.fun1();//输出什么，错误

window.alert(p1.age);//错误,undefined

**封装**

**通过prototype给所有对象添加方法，但是这种方法不能访问私有变量和私有方法。**

function Person(pname,page){

this.name=pname;//公开属性(变量)

var age=page;//私有属性(变量)

//私有方法

function fun1(){

window.alert("hello");

}

//公开函数(特权函数)

this.fun2=function(){

//可以访问所有成员

window.alert(this.name+" "+age);

fun1();

}

}

var p1=new Person("顺平",90);

//给所有对象添加公共方法

**//不能调用私有变量和方法**

Person.prototype.fun3=function(){

return "fun3()hello";

}

//输出"fun3()hello"

window.alert(p1.fun3());

**特别注意：类名.prototype.函数名=function(){}这个也称为后置绑定。**

**继承--为什么有？**

1、为什么需要继承

[js25\_1.html]继承示例(冗余)

<html>

<head>

<title>js示例继承示例(冗余)</title>

<script language="javascript" type="text/javascript">

<!--

function MidStu(name,age){

this.name=name;

this.age=age;

this.show=function(){

window.alert(this.name+" "+this.age);

}

//计算学费

this.payFee=function(money){

window.alert("应交"+parseFloat(money\*0.8));

}

}

function Pupil(name,age){

this.name=name;

this.age=age;

this.show=function(){

window.alert(this.name+" "+this.age);

}

//计算学费

this.payFee=function(money){

window.alert("应交"+parseFloat(money\*0.5));

}

}

var midStu=new MidStu("顺平",20);

midStu.show();

midStu.payFee(40);

var pupil=new Pupil("sp",30);

pupil.show();

pupil.payFee(40);

//-->

</script>

</head>

<body>

</body>

</html>

**//怎么解决代码冗余-->继承**

**//把学生类抽象出来(即把中学生和小学生的共性拿出来)**

[js25\_2.html]继承示例(继承优化冗余)

<html>

<head>

<title>js示例继承示例(继承优化冗余)</title>

<script language="javascript" type="text/javascript">

<!--

function Stu(name,age){

this.name=name;

this.age=age;

this.show=function(){

window.alert(this.name+" "+this.age);

}

}

function MidStu(name,age){

this.stu=Stu;

this.stu(name,age);//js中实际上通过对象冒充来实现继承(此语句必不可少,因为js是动态语言，如果不执行它，则不能实现继承效果)

//计算学费

this.payFee=function(money){

window.alert("应交"+parseFloat(money\*0.8));

}

}

function Pupil(name,age){

this.stu=Stu;

this.stu(name,age);

//计算学费

this.payFee=function(money){

window.alert("应交"+parseFloat(money\*0.5));

}

}

var midStu=new MidStu("顺平",20);

midStu.show();

midStu.payFee(40);

var pupil=new Pupil("sp",30);

pupil.show();

pupil.payFee(40);

//-->

</script>

</head>

<body>

</body>

</html>

**继承--解决之道**

继承可以解决代码利用，让我们的编程更加靠近人类思维，当多个类存在相同的属性(变量)和方法时，可以从这些类中抽象出父类(比如刚才的Stu)，在父类中定义这些相同的属性和方法，所有的不需要重新定义这些属性和方法，js中没有extends关键字，它可以通过如下方式实现继承。

**1、对象冒充；2、call及apply**

**继承--小结**

1、js通过前面的方式可以实现多重继承；

2、Object类是js所有类的基类。

**方法重载(overloading)及覆盖(overriding)**

简单的说：方法重载就是类的同一种功能的实现方式，到底采用哪种方式，取决于调用者给出的参数。

js不支持重载操作。因为javascript中的方法本身就是一个带可变参数的，通过在方法体内检测方法的参数情况，来实现重载的效果。[调用最后那个方法]

覆盖，也称为改写，就是指子类中定义的方法替换掉父类的方法。(案例说明)

函数重载案例：

function test1(){

window.alert("test1");

}

function test1(var1){

window.alert("test2");

}

test1();//显示？undefined

函数覆盖案例：

function Stu(name,age){

this.name=name;

this.age=age;

this.show=function(){

window.alert(this.name+" "+this.age);

}

}

function MidStu(name,age){

this.stu=Stu;

this.stu(name,age);//这句话必需有才能继承

this.payFee=function(sal){//这个是MidStu自己的函数

window.alert("应付学费="+sal\*0.8);

}

this.show=function(){

window.alert("我是midstu的show");

}

}

var midstu=new MidStu("顺平",20);

midstu.show();

**多态--概念**

所谓多态，就是指一个引用(类型)在不同情况下的多种状态，在java中多态是指通过指向父类的引用，来调用在不同子类中实现的方法。

js实际上是无态的，是一种动态语言，一个变量的类型是在运行的过程中由js引擎来决定的，所以说js天生就支持多态。

多态的案例：

function Person(){

this.test1=function(){

window.alert("Person test1");

}

}

function Cat(){

this.test1=function(){

window.alert("Cat test1()");

}

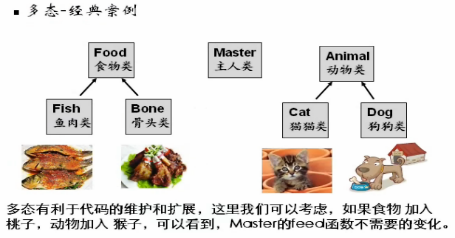
}

var v=new Person();

v.test1();

v=new Cat();

v.test1();



[js26.html] 多态演示

<html>

<head>

<title>js示例多态演示</title>

<script language="javascript" type="text/javascript">

<!--

//主人类

function Master(){

//给动物喂食

this.feed=function(animal,food){

window.alert(animal.constructor);

document.writeln("主人给"+animal.name+" 喂"+food.name+"<br>");

}

}

//食物类

function Food(name){

this.name=name;

}

//鱼类继承食物类

function Fish(name){

this.food=Food;

this.food(name);

}

//骨头类继承食物类

function Bone(name){

this.food=Food;

this.food(name);

}

//桃子类继承食物类

function Peach(name){

this.food=Food;

this.food(name);

}

//动物类

function Animal(name){

this.name=name;

}

//狗类继承动物类

function Dog(name){

this.animal=Animal;

this.animal(name);

}

//猫类继承动物类

function Cat(name){

this.animal=Animal;

this.animal(name);

}

//猴子类继承动物类

function Monkey(name){

this.animal=Animal;

this.animal(name);

}

var cat=new Cat("小猫咪");

var dog=new Dog("小狗");

var fish=new Fish("小鱼");

var bone=new Bone("骨头");

var monkey=new Monkey("小猴子");

var peach=new Peach("桃子");

var master=new Master();

master.feed(cat,fish);

master.feed(dog,bone);

master.feed(monkey,peach);

//-->

</script>

</head>

<body>

</body>

</html>

**补讲闭包**

解释：

1、闭包和垃圾回收机制GC是相关联的；

2、闭包实际上是在涉及到一个对象的成员属性，何时被GC处理的问题；

3、怎样才能对对象的属性形成一个闭包

[js27.html]闭包演示

<html>

<head>

<title>js示例闭包closure演示</title>

<script language="javascript" type="text/javascript">

<!--

function a(){

this.i=0;

function b(){

alert(++i);

}

return b;

}

//闭包是和GC垃圾回收机制相关的。

a();//此时a调用使用了函数中的i后，GC扫描i无任何函数指向时，就会回收释放空间。

var c=a();//而a调用后将b赋给c了，GC扫描知道c得到的返回值b中包含了i，此时GC会判断i虽然没有使用，但仍有变量指向，GC暂时不能回收并释放空间，并将i先保留。

c();

//-->

</script>

</head>

<body>

</body>

</html>

**js的内部类**

javascript中本身有提供一些可以直接使用的类，这种类就是内部类。

主要有：Object、Array、Math、Boolean、String、Number、Date、RegExp

js内部类--概述

前面我们讲的对象都是自定义类(就是自己写的)，js的设计者为了编程方便，也提供了事先定义好的类(有属性和方法了)，这些类在我们做web开发的时候，经常用到。我们称为js内部类。

按照使用方式，我们把js内部类分为两类(动态类、静态类)

js内部类：

**1、动态类**

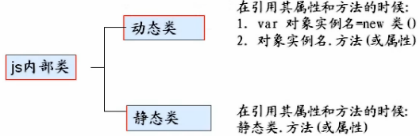
在引用其属性和方法的时候：

1、var 对象实例名=new 类名();

2、对象实例名.方法(或属性);

**2、静态类**

在引用其属性和方法的时候：静态类.方法(或属性);



[js28.html]内部类之静态、动态类示例

<html>

<head>

<title>js示例内部类示例</title>

<script language="javascript" type="text/javascript">

<!--

//静态类--Math的用法

window.alert(Math.abs(-12));

//动态类--Date的用法

//显示当前的日期

var nowdate=new Date();

window.alert(nowdate.toLocaleString());

//-->

</script>

</head>

<body>

</body>

</html>

**Object类**

Object类是所有javascript类的基类，提供了一种创建自定义对象的简单方式，不需要程序再定义构造函数。

**主要属性：**

constructor--对象的构造函数

prototype--获得类的prototype对象，static性质

**主要方法：**

hasOwnProperty(propety)--是否属性于本类定义的属性

isPrototypeOf(object)--是否是指定类的prototype

propertyIsEnumerable(property)--是否可例举的属性

toString()--返回对象对应的字符串

valueOf()--返回对象对应的原始类型值

案例：

<script language="javascript">

function getAttributeValue(attr){

alert(person[attr]);

}

var person=new Object();

person.name="zs";

person.age=18;

getAttributeValue("name");

getAttributeValue("age");

</script>

**Math[对象的作用：用于执行常见的数学任务]**

Math是静态类，提供了常用的数学函数和常数，介绍几个最常用的函数，其它的请大家参考javascript帮助文档

1、abs(x) 返回数的绝对值；

2、ceil(x) 对一个数进行上舍入

3、floor(x) 对一个数进行下舍入

4、max(x,y) 求x,y中较大的数

5、min(x,y) 求x,y中较小的数

6、round(x) 对x进行四舍五入

7、random() 一个大于0小于1的16位小数位的数字

案例：随机产生1至100的整数

alert(Math.round(Math.random()\*100));

**Date类[对象的作用：用于处理日期和时间]**

Date类是动态类，是提供了日期和时间的操作，介绍几个最常用的函数，其它的请大家参考javascript帮助文档。

1、Date() 返回当前日期和时间

2、getDate() 从Date对象返回一个月中的某一天

3、getDay() 从Date对象返回一周中的某一天

4、getMonth() 从Date对象返回月份

5、getYear() 从Date对象返回年

6、getHours() 从Date对象返回小时数

7、getMinutes() 从Date对象返回分钟

8、getSeconds() 从Date对象返回秒数

案例：

var date=new Date();

alert(date.Date().toLocaleString());//返回当前时间

alert(date.getYear()+"年"+date.getMonth()+"月");//返回当前年月

Date类

思考题：当用户浏览网页的时候，根据当前的时间，给出问候语：

6:00-9:00早上好

9:01-11:30上午好

11:31-14:30中午好

14:31-17:30下午好

17:31-18:40傍晚好

18:41-00:00晚上好

00:01-5:59凌晨好

[js29.html]内部类Date类示例

<html>

<head>

<title>js示例内部类Date类示例</title>

<script language="javascript" type="text/javascript">

<!--

var date=new Date();

function showHello(date){

//得到当前小时数

var hour=date.getHours();

//得到分钟

var minute=date.getMinutes();

//转成秒数

var mysecond=hour\*3600+minute\*60;

if(mysecond>6\*3600&&mysecond<(9\*3600+1\*60)){

alert("早上好");

}else if(mysecond>=(9\*3600+1\*60)&&mysecond<(11\*3600+31\*60)){

alert("上午好");

}else if(mysecond>=(11\*3600+31\*60)&&mysecond<(14\*3600+31\*60)){

alert("中午好");

}else if(mysecond>=(14\*3600+31\*60)&&mysecond<(17\*3600+31\*60)){

alert("下午好");

}else if(mysecond>=(17\*3600+31\*60)&&mysecond<(18\*3600+41\*60)){

alert("傍晚好");

}else if((mysecond>=(18\*3600+41\*60)&&mysecond<(23\*3600+60\*60-1))||mysecond<60){

alert("晚上好");

}else if(mysecond>=(0\*3600+1\*60)&&mysecond<(6\*3600+60\*60)){

alert("凌晨好");

}

}

showHello(date);

//-->

</script>

</head>

<body>

</body>

</html>

**String类[对象的作用：用于处理已有的字符块]**

String是动态类，提供了对字符串的各种操作，介绍几个常用的函数，其它的请参考javascript帮助文档。

1、indexOf() 返回某个字符串值在该字符串中首次出现的位置

2、split() 把字符串分割为字符串数组

3、substr() 提取取从start下标开始的指定数目的字符

4、substring() 提取字符串中介于两个指定下标之间的子串

5、charAt() 返回指定位置的字符

6、length 属性，可以得到字符串的长度

7、toString() js中所有内部对象的成员方法，作用是将对象中的数据转成某个格式的字符串，我们在实际运用中在详细介绍

8、match()/replace()/search()用的很多，但是涉及到正则表达式，这三个函数放在正则表达式章节中介绍

案例：

[js30.html]String类示例

<html>

<head>

<title>js示例内部类String类示例</title>

<script language="javascript" type="text/javascript">

<!--

var str="abcd12345"

alert(str.length);//length获得字符串长度

var str2="abc def oop";

alert(str2.split(" "));//split分割字符得到子字符串

var str3="abcdef";

alert(str3.substr(1,3));//输出bcd

alert(str3.substring(1,3));//输出bc

var str4="uvwxyz";

alert("charAt"+str4.charAt(3));//输出x

var str5="ab 123 56 ab";

alert("indexOf"+str5.indexOf("ab",1));//输出10

//-->

</script>

</head>

<body>

</body>

</html>

**Array类[对象的作用：使用单独的变量名存储一系列的值]**

Array是动态类，提供了对数组的操作，使用Array类可以轻松的创建数组，并对数组进行删除、排序和合并等操作。

通过Array对象创建数组

1、var myArr1=new Array();

可动态的添加元素：myArr1[0]=3;myArr1[1]="hello";

2、var myArr2=new Array(12,34,3.56,"白骨精");

myArr2[0]="牛魔王";myArr2[1]="红孩儿";myArr2[2]="铁扇公主";

案例：

[js31.html]Array类示例

<head>

<title>js示例内部类Array类示例</title>

<script language="javascript" type="text/javascript">

<!--

var myarr=new Array();

//动态添加数据

myarr[0]="sp";

myarr[1]=90;

alert(myarr.length+" "+myarr);//输出2 sp,90

myarr.pop();//出栈，删除数组中最后一个

alert(myarr.length+" "+myarr);//输出1 sp

myarr.push("abcd");

alert(myarr.length+" "+myarr);//输出2 sp,abcd

//-->

</script>

</head>

<body>

</body>

</html>

**Array提供了对数组的操作，使用Array对象可以轻松的创建数组，并对数组进行删除、排序和合并等操作。**

1、concat() 连接两个或更多的数组，并返回结果

2、sort() 对数组的元素进行排序

3、toString() 把数组转换为字符串，并返回结果

4、pop() 删除并返回数组的最后一个元素

5、push() 向数组的末尾添加一个或更多元素，并返回新的长度

6、length 属性，可以得到数组的长度

**Boolean类[对象的作用：用于将非逻辑值转换为逻辑值true或false]**

**Boolean动态类**，是一个把布尔值打包的布尔对象，这个对象在实际编程中，用的不太多，了解即可。

1、toString() 把逻辑值转换为字符串，并返回结果

2、valueOf() 返回Boolean对象的原始值

**Number类[对象的作用：原始数值的包装对象]**

Number类是该对象是原始数值的包装类。常用的函数有：

1、toString() 把一个Number对象转换为一个字符串，并返回结果

2、toFixed() 把数字转换为字符串，结果的小数点后有指定位数的数字

案例：

[js32.html]Number类示例

<html>

<head>

<title>js示例内部类Number类示例</title>

<script language="javascript" type="text/javascript">

<!--

var aa=890.567;

a=aa.toString();

alert(a);//转成字符串输出890.567

b=aa.toFixed(2);

alert(b);//四舍五入取小数点后2位输出890.57

var c=10;

alert(c.toString(2));//输出二进制1010

alert(c.toString(8));//输出八进制12

alert(c.toString(16));//输出十六进制a

//-->

</script>

</head>

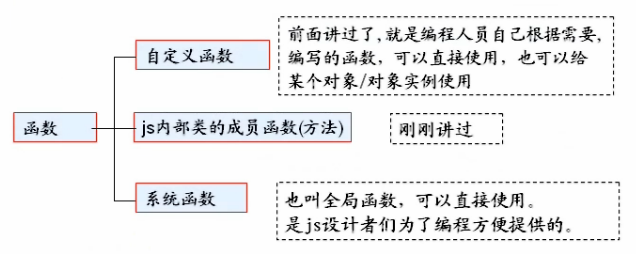
<body>

</body>

</html>

**js系统函数--概述**

前面我们讲的面向对象编程时，给大家介绍了函数(自定义函数和js内部类的成员函数)，现在我们再讲js最后一类函数--系统函数



函数

1、自定义函数

前面讲过了，就是编程人员自己根据需要，编写的函数，可以直接使用，也可以给某个对象/对象实例使用。

2、js内部类的成员函数(方法)

3、系统函数

也叫全局函数，可以直接使用，是js设计者们为了编程方便提供的。



**常用系统函数**

decodeURI() 解码某个编码的URI

encodeURI() 把字符串编码为URI

eval() 计算js字符串，并把它作为脚本代码来执行

isNaN() 检查某个值是否是数字

parseFloat() 解析一个字符串并返回一个浮点数

parseInt() 解析一个字符串返回一个整数

**encodeURI和decodeURI**

URL:Uniform Resource Locator,统一资源定位符；

URI:Universal Resource Identifier,通用资源标识符。

URI含义更广泛，URL是URI的一个部分，通常指网址在一定程度上，你可以认为URI=URL

这两个函数往往是成对使用，主要是用于屏蔽URL中的一些特殊符，比如空格

**常用系统函数**

另外大家还应当掌握以下几个系统函数：

1、eval() 可以把一个字符串当做脚本来运行，用处很大。

2、parseInt() 把字符串转成整数

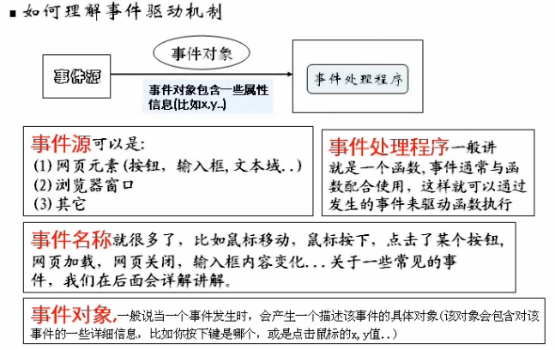
3、parseFloat() 把字符串转成小数

4、isNaN() 判断某个值是不是数(该函数不是特别完美)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**js事件驱动机制--概述**

**js是采用事件驱动(event-driven)响应用户操作的**。比如通过鼠标或者按键在浏览器窗口或者网页元素(按钮，文本框...)上执行的操作，我们称之为事件(Event)。由鼠标或热键引发的一连串程序的动作，称之为事件驱动(Event-Driver)。对事件进行处理程序或函数，我们称之为事件处理程序(Event Handler)。



事件的分类

**鼠标事件：**

当用户在页面上用鼠标点击页面元素时，对应的dom节点会角发鼠标事件，主要有click、dblclick、mousedown、mouseout、mouseover、mouseup、mousemove等。

**键盘事件：**

当用户用键盘输入信息时，会触发键盘操作事件。主要包括keydown、keypress、keyup三个。

**HTML事件：**

在html节点加载变更等相关的事件，比如window的onload、unload、abort、error，文本框select、change等等。

**其它事件：**

页面中一些特殊对象运行过程中产生的事件，比如xmlhttprequest对象的相关事件。

**快速入门案例**

[js33.html]js事件驱动机制快速入门案例

<html>

<head>

<title>js事件驱动机制快速入门案例</title>

<script language="javascript" type="text/javascript">

<!--

function show\_coords(event){

x=event.clientX;

y=event.clientY;

alert(event+"--X 坐标:"+x+",Y 坐标:"+y);

}

//-->

</script>

</head>

<body onmousedown="show\_coords(event)">

<p>Click in the document.An alert box will alert the x and y coordinates of the mouse pointer.</p>

</body>

</html>

**如何理解事件驱动机制**

为了让大家进一步理解js的事件驱动机制，我们来看两个案例：

案例：显示一个按钮，点击按钮后弹出对话框显示当前时间

[js34.html]

<html>

<head>

<title>js事件驱动机制示例II</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

alert(new Date().toLocaleString());

}

//-->

</script>

</head>

<body>

<input type="button" onclick="test()" value="点击查看当前时间">

</body>

</html>

案例：通过点击按钮改变DIV的颜色

[js35\_1.html]

<html>

<head>

<title>js事件驱动机制操作示例III</title>

<script language="javascript" type="text/javascript">

<!--

function test(eventObj){

//获取div1

var div1=document.getElementById('div1');

if(eventObj.value=="黑色"){

div1.style.backgroundColor="black";

}else if(eventObj.value=="红色"){

div1.style.backgroundColor="red";

}

}

//-->

</script>

</head>

<body>

<!--如何通过修改style来改变style-->

<div id="div1" style="width:400px;height:300px;background-color:red">div1</div>

<input type="button" value="黑色" onclick="test(this)" />

<input type="button" value="红色" onclick="test(this)" />

</body>

</html>

**javascript访问修改CSS样式表**

javascript访问修改样式表，可以方便的动态修改页面：

1、访问元素中style属性的CSS样式

这个可以直接使用style对象方便的访问，例如：

<div id="mdiv" style="background-color:blue;">...</div>

访问CSS的方法是：

<script type="text/javascript">

//获得元素

var oDiv=document.getElementById("mdiv");

//访问元素的style对象，再访问对象中的属性，也可以修改属性的值，直接为他赋值

alert(oDiv.style.backgroundColor);

</script>

2、访问外部定义的CSS样式(类定义的CSS样式)

这个没法使用上面的方法去访问，因为CSS数据不是存储在style属性中，它是存储在类中的。

**访问方法：先取得定义类的样式表的引用，用document.styleSheets集合实现这个目的，这个集合包含HTML页面中所有的样式表，DOM为每个样式表定义一个cssRules的集合，这个集合中包含定义在样式表中的所用CSS规则(注意：Mozilla和Safasi中是cssRules,而IE中是rules)。**

例如：

<style>

/\*第一条规则\*/

div.ss{

background-color:red;

width:101px;

}

/\*第二条规则\*/

a.btn2{

background:url(imag/2-AccessCtl.jpg);

}

</style>

访问CSS：

var ocssRules=document.styleSheets[0].cssRules||document.styleSheets[0].rules;

//访问第一条规则

alert(ocssRules[0].style.backgroundColor);

//设置值

ocssRules[0].style.width="992px";

//访问第二条规则

alert(ocssRules[1].style.background);

//设置值

ocssRules[0].style.background="url(imag/3-back.jpg);";

以上是我用到的时候一些简单的应用，具体深入的使用可以参看《javascript高级程序设计》中DOM技术部分

**案例：通过点击按钮改变DIV的颜色(使用CSS样式表修改)**

[js35\_2.html]**特别说明：此方式在IE8上测试通过，高版本浏览器未能通过。**

<html>

<head>

<title>js事件驱动机制操作外部CSS案例</title>

<link href="js35\_2.css" rel="stylesheet" type="text/css" />

<script language="javascript" type="text/javascript">

<!--

function test(eventObj){

//获取js35\_2.css中所有class选择器都获取

var ocssRules=document.styleSheets[0].rules;//IE浏览器下获取

//var ocssRules=document.styleSheets[0].cssRules;FF浏览器下获取

//从ocssRules取出你希望的class

var style1=ocssRules[0];

if(eventObj.value=="黑色"){

style1.style.backgroundColor="black";

}else if(eventObj.value=="红色"){

style1.style.backgroundColor="red";

}

}

//-->

</script>

</head>

<body>

<div id="div1" class="style1">div1</div>

<input type="button" value="黑色" onclick="test(this)" />

<input type="button" value="红色" onclick="test(this)" />

</body>

</html>

[js35\_2.css]样式表选择器

.style1{

width:600px;

height:400px;

background-color:black;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**案例：通过点击按钮改变DIV的颜色(使用CSS样式表修改)**

[js35\_3.html] **特别说明：此方式在各种版本浏览器通过测试。**

<html>

<head>

<title>js事件驱动机制操作示例III</title>

<link href="js35\_2.css" rel="stylesheet" type="text/css" />

<script language="javascript" type="text/javascript">

<!--

function test(eventObj){

//获取div1

var div1=document.getElementById('div1');

if(eventObj.value=="黑色"){

div1.style.backgroundColor="black";

}else if(eventObj.value=="红色"){

div1.style.backgroundColor="red";

}

}

//-->

</script>

</head>

<body>

<!--如何通过修改style来改变style-->

<div id="div1" class="style1">div1</div>

<input type="button" value="黑色" onclick="test(this)" />

<input type="button" value="红色" onclick="test(this)" />

</body>

</html>

[js35\_2.css]样式表选择器

.style1{

width:600px;

height:400px;

background-color:black;

}

**如何理解事件驱动机制不同浏览器兼容**

<script language="javascript">

<!--

if(window.XMLHttpRequest){//Mozilla,Safari,IE7,IE8

if(!window.ActiveXObject){//Mozilla,Safari

alert("Mozilla,Safari");

}else{

alert("IE");

}

}else{

alert("IE6");

}

//-->

</script>

**一个事件可以被多个函数监听**

**事件写在前面的会被先调用**

[js36.html]单事件多监听示例

<html>

<head>

<title>js事件驱动机制单事件多监听</title>

<script language="javascript" type="text/javascript">

<!--

function test1(e){

alert("test1事件");

}

function test2(e){

alert("test2事件");

}

//-->

</script>

</head>

<body>

<input type="button" value="点击" onclick="test1(this),test2(this)" />

</body>

</html>



js事件驱动机制--常用的事件

**属性 当以下情况发生时，出现此事件**

onabort 图像加载被中断

onblur 元素失去焦点

onchange 用户改变域的内容

onclick 鼠标点击某个对象

ondblclick 鼠标双击某个对象

onerror 当加载文档或图像时发生某个错误

onfocus 元素获得焦点

onkeydown 某个键盘的键被按下

onkeypress 某个键盘的键被按下或按住

onkeyup 某个键盘的键被松开

onload 某个页面或图像被完成加载

onmousedown 某个鼠标按键被按下

onmousemove 鼠标被移动

onmouseout 鼠标从某元素移开

onmouseover 鼠标被移到某元素之上

onmouseup 某个鼠标按键被松开

onreset 重置按钮被点击

onresize 窗口或框架被调整尺寸

onselect 文本被选定

onsubmit 提交按钮被点击

onunload 用户退出页面

**window有三个事件**

onload 页面打开时

onunload 关闭页面后

unbeforeunload 关闭页面时

**注意：onunload/onbeforeunload仅在IE6中测试成功**

[js37.html]onfocus/onload/onunload/onbeforeunload示例

<html>

<head>

<title>js事件驱动机制单onfocus焦点</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

alert("选中了输入框");

}

function test1(){

alert("页面打开了");

//把鼠标定位到test输入框

document.getElementById('text1').onfocus();

}

function test2(){

alert("页面关闭了");

}

//-->

</script>

</head>

<body onload="test1()" onunload="test2()" onbeforeunload="test2()">

<input type="text" onfocus="test()" id="text1" />

</body>

</html>

js常用事件实例

请使用js完成如下功能：

1、防止用户通过点击鼠标右键菜单copy网页内容

2、当用户试图选中网页文字copy时，给出提示(版权所有，禁止拷贝)

oncontextmenu,onselectstart

[js38.html]oncontextmenu,onselectstart示例

<html>

<head>

<title>js事件驱动机制单oncontextmenu,onselectstart事件</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

alert("版权所有，禁止拷贝");

return false;

}

//-->

</script>

</head>

<body oncontextmenu="return test()" onselectstart="return false">

<p>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

</p>

<p>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

测试内容，测试内容，测试内容，测试内容，测试内容<br>

</p>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DOM入门案例



[js39.html] 乌龟抓鸡示例

<html>

<head>

<title>js示例--乌龟抓鸡</title>

<script language="javascript" type="text/javascript">

<!--

//响应用户点击按钮，或者是按下键盘

function move(obj){

//得到乌龟图片的高度和宽度

var wugui\_height=67;

var wugui\_width=94;

//得到鸡图片的高度和宽度

var cock\_height=73;

var cock\_width=76;

//得到乌龟所在的div对象(这里就体现了dom特色)

var wugui=document.getElementById("wugui");

//window.event可以判断用户是通过点击鼠标还是通过键盘来控制乌龟

if(window.event){

//获得用户按下键的code值

key=obj.keyCode;

}

//得到该乌龟当前的top和left

var wugui\_top=wugui.style.top;

var wugui\_left=wugui.style.left;

//处理象素px后缀，进行数字运算

wugui\_top=parseInt(wugui\_top.substring(0,wugui\_top.indexOf("p")));

wugui\_left=parseInt(wugui\_left.substring(0,wugui\_left.indexOf("p")));

//判断用户点击哪个按

if(obj.value=="向下走"||key==40){

wugui.style.top=(wugui\_top+10)+"px";

wugui\_top=wugui\_top+10;

}else if(obj.value=="向右走"||key==39){

wugui.style.left=(wugui\_left+10)+"px";

wugui\_left=wugui\_left+10;

}else if(obj.value=="向左走"||key==37){

wugui.style.left=(wugui\_left-10)+"px";

wugui\_left=wugui\_left-10;

}else if(obj.value=="向上走"||key==38){

wugui.style.top=(wugui\_top-10)+"px";

wugui\_top=wugui\_top-10;

}

//判断乌龟是否碰到鸡

//得到公鸡的left和top

var cock=document.getElementById("cock");

//得到该公鸡当前的top和left

var cock\_top=cock.style.top;

var cock\_left=cock.style.left;

cock\_top=parseInt(cock\_top.substring(0,cock\_top.indexOf("p")));

cock\_left=parseInt(cock\_left.substring(0,cock\_left.indexOf("p")));

y=Math.abs(cock\_top-wugui\_top);

x=Math.abs(cock\_left-wugui\_left);

//判断条件

xx=0;

yy=0;

//判断纵向

if(wugui\_top<cock\_top){

if(y<wugui\_height){

yy=1;

}

}else{

if(y<cock\_height){

yy=1;

}

}

//判断横向

if(wugui\_left<cock\_left){

if(x<wugui\_width){

xx=1;

}

}else{

if(x<cock\_width){

xx=1;

}

}

if(xx==1&&yy==1){

alert("乌龟好猛啊，顶!");

wugui.style.top=120+"px";

wugui.style.left=100+"px";

}

}

//-->

</script>

</head>

<body onkeydown="return move(event)">

<table border="1px">

<tr>

<td></td>

<td><input type="button" value="向上走" onclick="move(this)" /></td>

<td></td>

</tr>

<tr>

<td><input type="button" value="向左走" onclick="move(this)" /></td>

<td></td>

<td><input type="button" value="向右走" onclick="move(this)" /></td>

</tr>

<tr>

<td></td>

<td><input type="button" value="向下走" onclick="move(this)" /></td>

<td></td>

</tr>

</table>

<!--把乌龟放在div中-->

<div id="wugui" style="position:absolute;left:100px;top:120px;">

<img src="1.jpg" border="1px" alt="" />

</div>

<!--鸡放在div中-->

<div id="cock" style="position:absolute;left:200px;top:200px;">

<img src="2.jpg" border="1px" alt="" />

</div>

</body>

</html>

**为什么学dom编程？**

1、通过dom编程，我们可以写出各种网页游戏；

2、dom编程，也是我们学习ajax技术的基础，所以我们必需掌握好dom编程。

dom编程简介

DOM=Document Object Model(文档对象模型)，根据W3C DOM规范，**DOM是HTML与XML的应用编程接口(API)**，DOM将整个页面映射为一个由层次节点组成的文件。

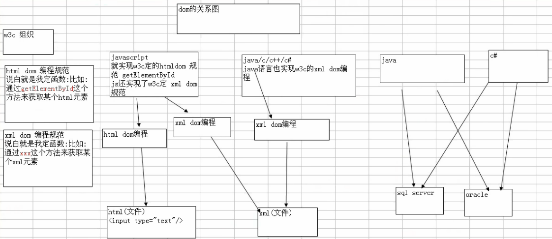
js把浏览器，网页文档和网页文档中的html元素都用相应的内置对象来表示，这些**对象与对象间的层次关系称为dom**，**针对网页(html,jsp,php,aspx,net)的dom就是html dom**。我们这讲的就是html dom

关于使用js对xml文件的dom编程，我们这暂时不讲。到讲ajax的时候在给大家讲解。

**dom分为html dom与xml dom**

可以借助数据库编程来讲清dom编程分类

**DOM原理图：**



**dom编程简介--dom对象**

前面说过，js把浏览器，网页文档和网页文档中的html元素都用相应的内置对象(看文档)来表示，这些对象与对象间的层次关系称为dom，针对网页(html,jsp,php,aspx,net)的dom就是html dom。我们这讲的就是html dom

这里有几个特别重要的概念要给大家说清林：

1、上面说的内置对象就是dom对象，准确的说是html dom对象。

2、因为我目前讲的是html dom编程，所以我提到的dom编程，dom对象都是说的html dom编程和html dom对象，请大家不要犯迷糊。

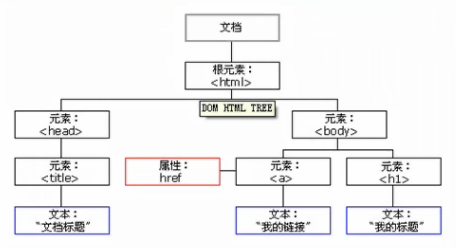
3、dom编程时，会把html文档看做是一颗dom树。

4、dom编程的核心就是各个dom对象。

**dom编程简介--html dom层次图**

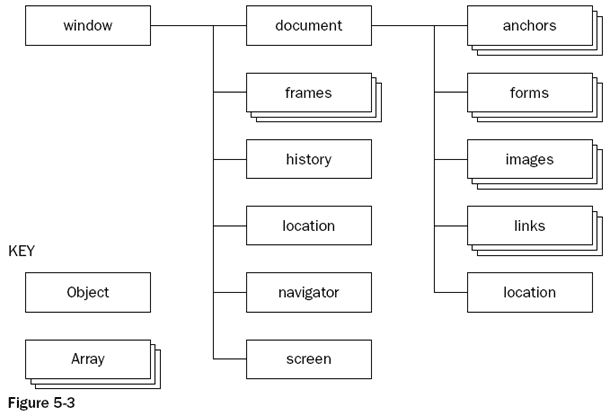
html dom定义了访问和操作html文档的标准方法。

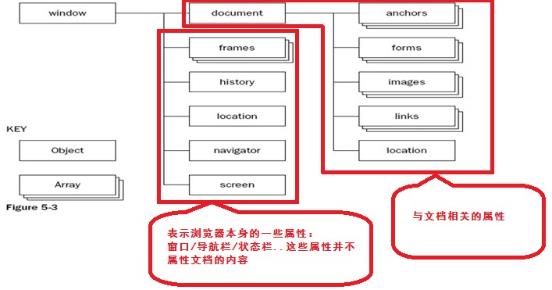
html dom把html文档呈现为带有元素、属性和文本的树结构(节点树)



**bom简介**

BOM(The Browser Object Model)--浏览器对象模型



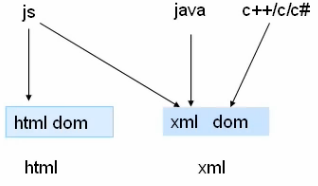


通过使用BOM，可移动窗口、更改状态栏文本。bom是w3c组织提出的，他建议所有的浏览器都应当遵循这样的设计规范，可以说bom和dom关系密切，相互影响，bom为纲，dom为目，我们可以简单的理解dom就是bom的具体实现。

**dom简介**

1、dom(document object model)文档对象模型，根据w3c dom规定，dom是html和xml的应用程序编程接口(api)，dom将html或xml映射成由层次节点组成的模型，根据规范的严格层度，分为1,2,3级。

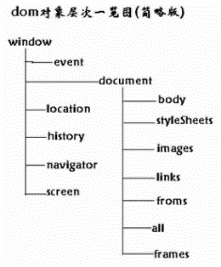
2、dom的核心是提供了一套访问结构化文档的api，核心是对节点各种操作。



**dom对象简介**

js把浏览器，网页文档和网页文档中的html元素和相应的内置对象来表示，这些内置对象就是dom对象，编程人员通过访问dom对象，就可以实现对浏览器本身、网页文档、网页文档中元素的操作，从而控制浏览器和网页元素的行为和外观。

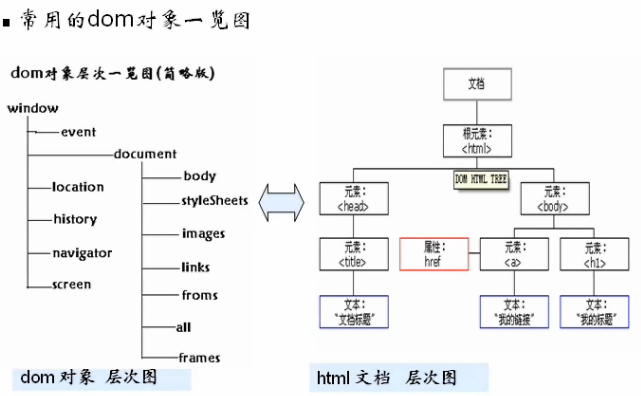
**常用的dom对象一览图**



这里，只列举了常用的dom对象。从上图可以看出：

dom对象本身也有层次，比如document对象是window对象的一个成员属性，而body对象又是document对象的一个成员属性。

而在前面说过，dom编程其实就是把网页文档看做是一棵树(html domt tree)(倒的)，然后用相应的dom对象去访问网页文档，从而达到控制浏览器和网页的行为和外观。



**常用的dom的每个Node节点属性和方法**

**特别提示：**

html dom和xml dom都遵循相同dom规范的，所以他们有很多相同的方法和属性，因此我们也可以去查看xml dom的node节点提供的方法和属性。

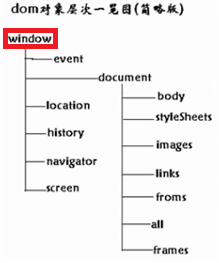
DOM概述

Document Object Model，文档对象模型，根据w3c dom规范，dom是html

|  |  |  |
| --- | --- | --- |
| **属性名称** | **类型** | **说明** |
| nodeName | String | 节点名称 |
| nodeValue | String | 节点值 |
| nodeType | Number | 节点类型 |
| parentNode | Node | 父节点 |
| firstChild | Node | 第一个子节点 |
| lastChild | Node | 最后一个子节点 |
| childNodes | NodeList | 所有子节点 |
| previousSibling | Node | 前一个节点 |
| nextSibling | Node | 后一个节点 |
| ownerDocument | Document | 获得该节点所属的文档对象 |
| attributes | Map | 代表一个节点的属性对象 |

|  |  |  |
| --- | --- | --- |
| **方法名称** | **返回值** | **说明** |
| hasChildNodes() | Boolean | 当前节点是否有子节点 |
| appendChild(node) | Node | 往当前节点上添加子节点 |
| removeChild(node) | Node | 删除子节点 |
| replaceChild(oldNode,newNode) | Node | 替换子节点 |
| insertBefore(newNode,refNode) | Node | 在指定节点的前面插入新节点 |

**window对象**



window对象表示一个浏览器窗口或一个框架。在客户端javascript中，window对象是全局对象，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来用。

在使用window对象的方法和属性的时候，可以不带window，比如：window.alert("韩顺平!");，可以直接写成alert("韩顺平");

**window对象**

讲解window对象常用的函数和属性，其它的参考帮助文档，还有很多可能需要在做项目的时候，才会接触。

1、alert() 显示消息和一个确认按钮的警告框

2、confirm() 显示消息以及确认按钮和取消按钮的对话框

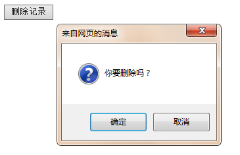
3、complile()

4、setInterval() 按照指定的周期(毫秒计)来循环调用函数或计算表达式

5、clearInterval() 取消由setInterval()设置的定时器

6、setTimeout() 指定的毫秒数后调用函数或计算表达式

7、clearTimeout() 取消由setTimeout()设置的定时器



[dom01-window.html]confirm()方法使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

var res=window.confirm("你要删除吗？");

if(res){

window.alert("点击确定，返回true。删除!");

}else{

window.alert("点击取消，返回false。未删除!");

}

}

//-->

</script>

</head>

<body>

<input type="button" value="删除记录" onclick="test()" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例：需求每隔5秒钟弹出hello world!

[dom02-window.html]setInterval请时循环方法示例1

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function sayHello(){

window.alert("hello!");

}

setInterval("sayHello()",5000);

function showTime(){

//在元素间的普通文本就是通过 [对象名.innerText]

document.getElementById("mytime").innerText=new Date().toLocaleString();

}

setInterval("showTime()",1000);

//-->

</script>

</head>

<body>

<span id="mytime"></span>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**setInterval()及clearInterval()方法的使用**

[dom03-window.html]setInterval()及clearInterval()方法的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

var n=1;

var count=0;

//让静态图片动起来

function runChild(){

count++;

if(count>13){

//停止定时器

clearInterval(myTimer);

}

//得到img对象

var myimg=document.getElementById("myimg");

myimg.src=(n++)+".jpg";

if(n>13){//因为有13张图片

n=1;

}

}

var myTimer=setInterval("runChild()",200);

//-->

</script>

</head>

<body>

<img src="1.jpg" id="myimg" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**setTimeout()方法的使用**

[dom04-window.html]setTimeout()方法的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

//延时5秒弹出hello对话框

function sayHello(){

alert("Hello!");

}

setTimeout("sayHello()",5000);

//-->

</script>

</head>

<body>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**setTimeout()及clearTimeout()方法的使用**

[dom05-window.html]setTimeout()及clearTimeout()方法的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

var n=1;

var count=0;

//让静态图片动起来

function runChild(){

count++;

if(count>13){

//停止定时器

clearInterval(myTimer);//停止setInterval()定时器

alert("休息5秒!");

var myTimer2=setTimeout("reRun()",5000);//开启setTimeout()定时器

count=0;

n=1;

}

//得到img对象

var myimg=document.getElementById("myimg");

myimg.src=(n++)+".jpg";

if(n>13){//因为有13张图片

n=1;

}

}

var myTimer=setInterval("runChild()",200);

function reRun(){

myTimer=setInterval("runChild()",200);

}

function stopMove(){

var stop=window.confirm("确定终止图片滚动？");

if(stop){

clearInterval(myTimer);//终止setInterval定时器

clearTimeout(myTimer2);//终止setTimeout定时器

}

}

//-->

</script>

</head>

<body>

<img src="1.jpg" id="myimg" /><br>

<input type="button" value="终止图片滚动" onclick="stopMove()" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**window对象**

moveTo() 把窗口的左上角移动到一个指定的坐标

moveBy() 可相对窗口的当前坐标把它移动指定的像素

resizeBy() 按照指定的像素调整窗口的大小

resizeTo() 把窗口的大小调整到指定的宽度和高度

open() 打开一个新的浏览器窗口或查找一个已命名的窗口

close() 关闭窗口

closed 返回窗口是否已被关闭

status 设置窗口状态栏的文本

opener 返回对创建此窗口的窗口的引用

onload 页面装载

onunload 关闭窗口

**moveTo()及moveBy()使用 [IE下测试通过]**

[dom06-window.html]moveTo()及moveBy()使用演示

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

window.moveTo(100,100);//需在窗口化下才可看到演示

var move=window.confirm("是否再次移动？");

if(move){

window.moveBy(100,100);

}

}

//-->

</script>

</head>

<body>

我是一个窗口

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**resizeBy()及resizeTo()的使用 [IE下测试通过]**

[dom07-window.html]resizeBy()及resizeTo()的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

resizeBy(300,200);

}

function test2(){

resizeTo(400,500);

}

//-->

</script>

</head>

<body>

<input type="button" value="按像素调整窗口大小" onclick="test()" />

<input type="button" value="指定窗口为宽400高500" onclick="test2()" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**open()的使用**

[dom08-window.html]open()的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

window.open("test.html","\_blank","width=300,height=300,nenubar=no,location=no,titlebar=no");

}

//-->

</script>

</head>

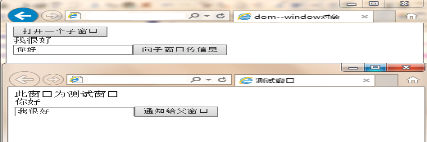
<body>

<input type="button" value="打开一个新窗口" onclick="test()" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**opener父窗口和子窗口通信 [IE中测试通过]**

[dom09-window.html]opener的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

var newWindow;

function test(){

newWindow=window.open("test.html","\_blank");

}

function test2(){

newWindow.$("myspan2").innerText=$("info2").value;//向子窗口发送信息

}

//将document.getElementById封装成函数

function $(id){

return document.getElementById(id);

}

//-->

</script>

</head>

<body>

<input type="button" value="打开一个子窗口" onclick="test()" /><br>

<span id="myspan">消息</span><br>

<input type="text" id="info2" /><input type="button" value="向子窗口传信息" onclick="test2()" />

</body>

</html>

[test.html](dom09-window.html子窗口)

<html>

<head>

<title>测试窗口</title>

<script language="javascript" type="text/javascript">

<!--

function notify(){

opener.$("myspan").innerText=$("info").value;//向父窗口发送信息

}

function $(id){

return document.getElementById(id);

}//-->

</script>

</head>

<body>

此窗口为测试窗口<br>

<span id="myspan2">子消息</span><br>

<input id="info" type="text" value="" /><input type="button" value="通知给父窗口" onclick="notify();">

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**window对象**

在掌握window对象一些常用方法后，我们来看几个案例

一个js版本的用户登录系统，当用户输入admin密码是123456就跳到第二页面，4秒后，自动跳转到第三个页面。

[dom10-window.html]login页面

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

var name=$("uname");

var pwd=$("pwd");

if(name.value=="admin"&&pwd.value=="123456"){//用户名和密码判断

return true;

}else{

alert("用户名或密码错误，请重新输入!");

name.value="";

pwd.value="";

return false

}

}

//将document.getElementById封装成函数

function $(id){

return document.getElementById(id);

}

//-->

</script>

</head>

<body>

<form action="ok.html">

用户名：<input type="text" id="uname" /><br>

密&nbsp&nbsp码：<input type="password" id="pwd" /><br>

<input type="submit" value="登&nbsp&nbsp录" onclick="return test()" /><br>

</form>

</body>

</html>

[ok.html]读秒页面

<html>

<head>

<title>登录等待窗口</title>

<script language="javascript" type="text/javascript">

<!--

//setTimeout("javascript:window.open("manage.html")",5000);

function changeSec(){

$("time").innerText=$("time").innerText-1;

if($("time").innerText==0){

clearInterval(timer);//清空计时器

open("manage.html","\_self");

}

}

var timer=setInterval("changeSec()",1000);

function $(id){

return document.getElementById(id);

}

//-->

</script>

</head>

<body>

登录成功<br>

<span id="time">5</span>秒后跳转至管理页面!<br>

</body>

</html>

[manage.html]管理页面

<html>

<head>

<title>登录管理窗口</title>

<script language="javascript" type="text/javascript">

<!--

//-->

</script>

</head>

<body>

登录管理窗口<br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**status的使用(可以控制状态栏信息) [IE中测试通过]**

[dom11-window.html]status的使用示例

<html>

<head>

<title>dom--window对象</title>

<script language="javascript" type="text/javascript">

<!--

var space\_num=0;

var dir=1;

function myScroll(){

var space\_my="";

space\_num=space\_num+1\*dir;

if(space\_num>50||space\_num<0){

dir=dir\*-1;

}

for(var i=0;i<space\_num;i++){

space\_my=space\_my+" ";

}

window.status=space\_my+"世界你好!";

}

function startIt(){

setInterval("myScroll()",100);

}

//-->

</script>

</head>

<body onload="startIt()">

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**history对象**



history对象：Contains information about the URLs visited by the client，即：该对象包含客户端访问过的URL信息。

从dom层次图看，history是window对象的成员属性，但实际history是由js runtime engine自动创建的，所以你也可以认为history就是一个js对象

history对象常用的函数和属性

1、back() 加载history列表中的前一个URL

2、forward() 加载history列表中的下一个URL(需已访问过的URL才可以使用)

3、go() 加载history列表中的某个具体页面

4、length 返回浏览器历史列表中的URL数量

**back()的使用**

[test.html]

<a href="dom01-history.html">goto dom01-history</a>

[dom01-history.html]back()的使用示例

<html>

<head>

<title>history使用示例</title>

<script language="javascript" type="text/javascript">

<!--

function goback(){

history.back();//等价于history.go(-1);

}

//-->

</script>

</head>

<body>

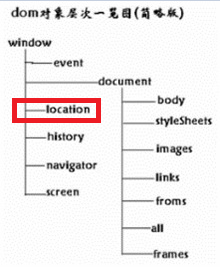
<a href="#" onclick="goback()">返回test页面</a>//不只是input可以有onclick事件

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**location对象**



location对象：Contains information about the current URL,即：该对象包含客户端当前URL信息

从dom层次图看，location是window对象的成员属性，但实际location也是由js runtime engine自动创建的，所以你也可以认为location就是一个js对象。

**location属性和方法**

assign() 加载新的文档

reload() 重新加载当前文档

replace() 用新的文档替换当前文档

hash 设置或返回#开始的URL

host 设置或返回主机名和当前URL的端口号

hostname 设置或返回当前URL的主机号

href 设置或返回完整的URL

pathname 设置或返回当前URL的路径部分

port 设置或返回当前URL的端口号

protocol 设置或返回当前URL的协议

search 设置或返回从问号(?)开始的URL(查询部分)

[dom01-location.html]location的方法和属性使用示例

<html>

<head>

<title>location使用示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

location.reload();//等同刷新页面

document.writeln("当前页面地址："+location.href+"<br>当前主机名："+location.hostname+"<br>当前端口号："+location.port);

}

//-->

</script>

</head>

<body>

<input type="button" value="刷新页面" onclick="test()" /><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**navigator对象**



navigator对象：Contains information about the browser，即：该对象包含当前浏览器的各信息。

从dom层次图看，navigator是window对象的成员属性，但实际navigator也是由js runtime engine自动创建的，所以你也可以认为navigator就是一个js对象。

**navigator对象常用的函数和属性**

1、javaEnabled() 返回该浏览器是否支持java

2、appName 浏览器的名称

3、appVersion 浏览器的版本

4、browserLanguage 浏览器当前使用的语言

5、cookieEnabled 返回浏览器是否启用cookie

6、platform 返回浏览器所在操作系统名称

7、systemLanguage 系统默认语言

**navigator对象的使用**

[dom01-navigator.html]navigator获得浏览器信息的使用示例

<html>

<head>

<title>navigator使用示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

document.writeln("目前使用的浏览器是："+navigator.appName+"<br>使用者系统为："+navigator.platform+"<br>使用的语言是："+navigator.systemLanguage);

}

//-->

</script>

</head>

<body>

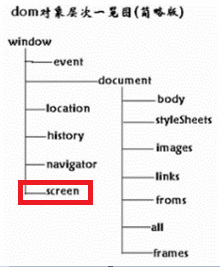
<input type="button" value="获得浏览器信息" onclick="test()" /><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**screen对象**



screen对象：Contains information about the client's screen and rendering capabilities，即：该对象包含有关客户机显示屏幕的信息。

从dom层次图看，screen是window对象的成员属性，但实际screen也是由js runtime engine自动创建的，所以你也可以认为screen就是一个js对象。

**screen对象常用的属性**

1、height 返回显示屏幕的高度(按像素)

2、width 返回显示屏幕的宽度(按像素)

3、availHeight 返回显示屏幕可用高度(去掉windows任务栏)

4、availWidth 返回显示屏幕可以宽度

[dom01-screen.html]screen的属性使用

<html>

<head>

<title>screen使用示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

var w=screen.width;

var h=screen.height;

var aw=screen.availWidth;

var ah=screen.availHeight;

document.getElementById("myspan").innerText="当前分辨率为："+w+"\*"+h+"--||--"+"当前可用区域"+aw+"\*"+ah;

if(w==1024&&h==768){

alert("当前分辨率为1024\*768");

}else{

alert("当前分辨率为："+w+"\*"+h);

}

}

//-->

</script>

</head>

<body>

<span id="myspan"></span>

<br>

<input type="button" value="获取当前屏幕分辨率" onclick="test()" /><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**event对象**



event对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态，事件通常与函数结合使用。

从dom对象层次图看，event对象是window对象的属性，**其实在其它的dom对象都存在event事件(一种event对象就代表一类事件)，理解这一点非常重要!**

事件驱动编程

事件驱动编程

所谓事件驱动，简单地说就是你点什么按钮(即产生什么事件)，电脑执行什么操作用什么函数)。当然事件不仅限于用户的操作，当对象处于某种状态时，可以获得消息通知，然后对这个消息感兴趣的程序就可以执行。

事件驱动编程的几个核心对象

事件源：谁发出事件通知，发出消息就是事件主体。

事件名称：发出什么样的通知的名称，比如鼠标到我头上了，我被别人点了一下。

事件响应函数：谁对这个事件感兴趣，当这个事件发生时要执行什么样的操作。

事件对象：一般来说，当事件发生时，会产生一个描述该事件的具体对象，包括具体的参数起发给响应函数，好让他们通过事件对象来了解事件更加详细的信息。

**特别注意：**前面已详细讲解了event事件，这里就不在详细阐述了。

**如何绑定事件监听**

1、直接在某个html控件上指定

2、getElementById("")获取控件后，再绑定

3、通过addEventListener()或者是attachEvent()来绑定

4、一个事件可以有多个事件监听者(函数)

W3C DOM标准：

[Object].addEventListener("name\_of\_event",fnHandler,bCapture);

[Object].removeEventListener("name\_of\_event",fnHandler,bCapture);

IE中独有的事件监听方法：

[Object].attachEvent("name\_of\_event\_handler",fnHandler);

[Object].detachEvent("name\_of\_event\_handler",fnHandler);

通过getElementById("")获取控件后绑定事件

[dom01-event.html]

<html>

<head>

<title>event示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

alert("hello!");

}

//-->

</script>

</head>

<body>

<input id="but" type="button" value="测试" /><br>

<script language="javascript" type="text/javascript">

document.getElementById("but").onclick=test;

</script>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**在IE下通过attachEvent绑定和detachEvent解除绑定**

[dom02-event.html]

<html>

<head>

<title>event示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

alert("投票成功!");

//解除事件绑定 detachEvent高版本IE无法使用

document.getElementById("but").detachEvent("onclick",test);

}

//-->

</script>

</head>

<body>

<input id="but" type="button" value="投票" /><br>

<script language="javascript" type="text/javascript">

//attachEvent高版本IE无法使用

document.getElementById("but").attachEvent("onclick",test);

</script>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**如何获取事件对象**

dom会将事件对象通过参数传递给响应函数(事件监听函数)，而ie则可以直接通过window.event来获取事件对象

非IE浏览器

DOM中获取事件对象：

DOM中要求event是传给事件响应函数的第一个参数

oDiv.onclick=function(){

var oEvent=arguments[0];

//oEvent即为事件对象

}

或者

oDiv.onclick=function(oEvent){

//参数oEvent即为事件对象

}

IE浏览器

IE中的Event对象-window.event

oDiv.onclick=function(){

var oEvent=window.event;

//oEvent就是事件对象

}

**event对象**

1、onclick 用户点击某个html元素触发的事件

2、ondbclick 用户双击某个html元素触发的事件

3、onfous 元素获得焦点

4、onkeydow 某个键盘按键被按下

5、onkeypress 某个键盘按键被按下并松开

6、onkeyup 某个键盘按键被松开

7、onmousedown 鼠标按钮被铵下

8、onmouseover 鼠标移到某元素上

9、onmouseout 鼠标从某元素移开

10、onmouseup 鼠标按钮被松开

11、onmousemove 鼠标在某个元素移动

**event对象**

1、onresize 窗口或框架被重新调整大小

2、onselect 文本被选择

3、onsubmit 提交按钮被按下

这里**特别强调一点，并不是所有的html元素都有相同的event事件(对象)**，这个请大家在做项目的时候特别注意，比如提交按钮有onsubmit事件，但是输入框就没有的，具体参考js帮助文档。

**event对象**

1、clientX 返回当事件被触发时，鼠标指针的横坐标(针对窗口)

2、clientY 返回当事件被触发时，鼠标指针的纵坐标(针对窗口)

3、screenX 返回当事件被触发时，鼠标指针的横坐标(针对屏幕)

4、screenY 返回当事件被触发时，鼠标指针的纵坐标(针对屏幕)

5、altKey 返回事件被触发时，“ALT”是否被按下

6、shiftKey 返回当事件被触发时，“shift”是否被按下

7、keyCode 返回被按下键的unicode字符码

**案例：**

要求用户在一个输入框中输入一个数，如果不是数字，则给出提示。

[dom03-event.html]

<html>

<head>

<title>event示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(event){

//用户每按下一个键，就去判断是不是一个数

if(event.keyCode<48||event.keyCode>57){

alert("Sorry，输入的不是数!");

return=false;

}

}

//-->

</script>

</head>

<body>

请输入一个六位数：<input id="txt" type="text" onkeypress="return test(event)"/><br>

<input type="button" value="提交" onsubmit="test()" />

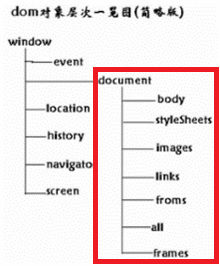
</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--document对象**

document对象



Document对象代表整个html文档，可用来访问页面中的所有元素，是最复杂的一个dom对象，可以说是学习好dom编程的关键所在。

Document对象是window对象的一个成员属性，通过window.document来访问，当然也可以直接使用document

**document对象常用的函数和属性，其它的参考帮助文档。**

1、write() 向文档写文本或html表达式或javascript代码

2、writeln() 等同于write()方法，不同的是在每个表达式之后写一个换行符

3、open() 打开一个新文档，并删除当前文档内容

4、close() 关闭open()文档流

5、getElementById() 通过html控件的id得到该控件。如果有相同的id则只取第一个

6、getElementsByName() 通过html控件的名字返回对象集合

7、getElementsByTagName() 通过html的标签名返回对象集合

8、createElement() 创建一个指定标签名的元素[比如：动态创建超链接]

**getElementById()的使用**

[dom1-document.html]getElementById()的使用示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

var myhref=document.getElementById("a1");//相同id标签只找第一个

alert(myhref.innerText);

}

//-->

</script>

</head>

<body>

<a id="a1" href="http://www.sohu.com">连接到sohu</a><br>

<a id="a1" href="http://www.sina.com.cn">连接到sina</a><br>

<a id="a1" href="http://www.163.com">连接到163</a><br>

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**通过元素名来获取集合getElementsByName()的使用**

[dom2-document.html]getElementsByName()的使用示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

//id不能唯一，但是name可以重复

var hobbies=document.getElementsByName("hobby");

//alert(hobbies.length);

for(var i=0;i<hobbies.length;i++){

//如果判断是否选择

if(hobbies[i].checked){

alert("你的爱好是："+hobbies[i].value) ;

}

}

}

//-->

</script>

</head>

<body>

请选择你的爱好

<input type="checkbox" name="hobby" value="足球" />足球

<input type="checkbox" name="hobby" value="旅游" />旅游

<input type="checkbox" name="hobby" value="音乐" />音乐

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**通过标签名来获取对象(元素)getElementsByTagName()的使用**

[dom3-document.html]getElementsByTagName()的使用示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

//通过标签名来获取对象(元素)

var myObjs=document.getElementsByTagName("input");

for(var i=0;i<myObjs.length;i++){

alert(myObjs[i].value);

}

}

//-->

</script>

</head>

<body>

请选择你的爱好

<input type="checkbox" name="hobby" value="足球" />足球

<input type="checkbox" name="hobby" value="旅游" />旅游

<input type="checkbox" name="hobby" value="音乐" />音乐

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**document对象**

**动态的创建元素(节点)/添加元素(节点)/删除元素(节点)**

[dom4-document.html]动态创建、添加、删除元素节点示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function test1(){

//创建元素

var myElement=document.createElement("a");//createElement(??)??写希望创建的html元素的标签名

//给新元素添加必要的信息

myElement.id="id1";

myElement.href="http://www.sina.com.cn";

myElement.innerText="连接到sina";

//给新元素指定位置

myElement.style.left="400px";

myElement.style.top="300px";

myElement.style.position="absolute";

//添加到document.body上

document.body.appendChild(myElement);

}

function test2(){

var myButton=document.createElement("input");

myButton.type="button";

myButton.value="这是动态建立的按钮";

document.getElementById("div1").appendChild(myButton);

}

function test3(){

//删除一个元素(删除一个元素前提：必需获得父元素)

//这是第一种删除方法(不灵活)

document.body.removeChild(document.getElementById("id1"));

**//第二种方法(推荐此方法)**

**//document.getElementById("id1").parentNode.removeChild(document.getElementById("id1"));**

}//-->

</script>

</head>

<body>

<input type="button" onclick="test1()" value="动态创建超链接" />

<input type="button" onclick="test2()" value="动态创建按钮" />

<input type="button" onclick="test3()" value="删除超链接" />

<div id="div1" style="width:200px;height:400px;border:1px solid red;">div1</div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--document对象**

**常用的dom的每个Node节点属性和方法(加强篇)**

**特别提示：**

html dom和xml dom都遵循相同dom规范的，所以他们有很多相同的方法和属性，因此我们也可以去查看xml dom的node节点提供的方法和属性。

DOM概述

Document Object Model，文档对象模型，根据w3c dom规范，dom是html

|  |  |  |
| --- | --- | --- |
| **属性名称** | **类型** | **说明** |
| nodeName | String | 节点名称 |
| nodeValue | String | 节点值 |
| nodeType | Number | 节点类型 |
| parentNode | Node | 父节点 |
| firstChild | Node | 第一个子节点 |
| lastChild | Node | 最后一个子节点 |
| childNodes | NodeList | 所有子节点 |
| previousSibling | Node | 前一个节点 |
| nextSibling | Node | 后一个节点 |
| ownerDocument | Document | 获得该节点所属的文档对象 |
| attributes | Map | 代表一个节点的属性对象 |

|  |  |  |
| --- | --- | --- |
| **方法名称** | **返回值** | **说明** |
| hasChildNodes() | Boolean | 当前节点是否有子节点 |
| appendChild(node) | Node | 往当前节点上添加子节点 |
| removeChild(node) | Node | 删除子节点 |
| replaceChild(oldNode,newNode) | Node | 替换子节点 |
| insertBefore(newNode,refNode) | Node | 在指定节点的前面插入新节点 |

**node属性方法的使用 [前、后节点功能在IE8以上浏览器未通过测试]**

[dom5-document.html]node属性和方法的使用示例

<html>

<head>

<title>document示例Node节点--乌龟抓鸡</title>

<script language="javascript" type="text/javascript">

<!--

function test(){

var wuguiDiv=$('wugui');

alert(wuguiDiv.nodeName+" "+wuguiDiv.nodeType+" "+wuguiDiv.nodeValue);

alert(wuguiDiv.childNodes.length+" "+wuguiDiv.nextSibling.nodeValue+" "+wuguiDiv.previousSibling.nodeValue+" "+wuguiDiv.parentNode);

}

function $(id){

return document.getElementById(id);

}

//-->

</script>

</head>

<body onkeydown="return move(event)">

<input type="button" value="Node测试" onclick="test()" /><br>

<table border="1px">

<tr>

<td></td>

<td><input type="button" value="向上走" onclick="move(this)" /></td>

<td></td>

</tr>

<tr>

<td><input type="button" value="向左走" onclick="move(this)" /></td>

<td></td>

<td><input type="button" value="向右走" onclick="move(this)" /></td>

</tr>

<tr>

<td></td>

<td><input type="button" value="向下走" onclick="move(this)" /></td>

<td></td>

</tr>

</table>

<!--把乌龟放在div中-->

<div id="wugui" style="position:absolute;left:100px;top:120px;">

<img src="1.jpg" border="1px" alt="" />

</div>

<!--鸡放在div中-->

<div id="cock" style="position:absolute;left:200px;top:200px;">

<img src="2.jpg" border="1px" alt="" />

</div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--document对象**

**document对象的属性**

1、alinkColor 当前文档超链接的颜色

2、linkColor 当前文档超链接的颜色

3、alinkColor 当前文档访问过的超链接颜色

4、bgColor 当前文档背景色

5、fgColor 文档前景色

6、title ` 当前文档的标题

7、URL 当前文档URL

8、domain 当前文件访问的域名

**document属性的使用[IE中测试通过]**

[dom6-document.html]document属性的使用

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

document.fgColor="white";//前景色

document.bgColor="black";//背景色

alert(document.title);//获得title信息

alert(document.URL);//获得URL地址信息

//-->

</script>

</head>

<body>

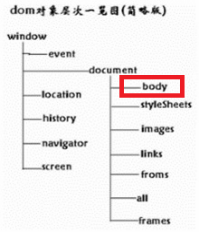
hello,world!你好，中国!

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**body对象**



body对象代表文档的主体(HTML body)。

body对象是document对象的一个成员属性，通过document.body来访问。

使用body对象，要求文档的主体创建后才能使用，也就是说不能在文档的body体还没有创建就去访问body。

**body对象常用的函数和属性**

1、appendChild() 添加元素

2、removeChild() 删除元素

3、getElementsByTagName() 通过html元素名称，得到对象数组

4、bgColor 文档背景色

5、backgorund 文档背景图

6、innerText 某个元素间的文本

7、innerHTML 某个元素间的html代码

8、onload 文档加载时触发

9、onunload 文档关闭时触发

10、onbeforeunload 文档关闭前触发

**innerText()与innerHTML()区别**

[dom7-document-body]innerText()与innerHTML()区别示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function test1(){

//innerText作为文本解析

document.getElementById("myspan").innerText="<a href='http://www.sohu.com'>转到sohu</a><br>";

}

function test2(){

//innerHTML作为html解析

document.getElementById("myspan").innerHTML="<a href='http://www.sohu.com'>转到sohu</a><br>";

}

//-->

</script>

</head>

<body>

<span id="myspan"></span><br>

<input type="button" value="测试innerText" onclick="test1()" />

<input type="button" value="测试innerHTML" onclick="test2()" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**body对象**

1、onselectstart 用户选中文档body体的内容时触发

2、onscroll 用户拉动滚动条时触发

**特别说明：**onscroll在使用时需要遵循html4.0的标准，否则不能执行，也可以使用window.onscroll=函数名来代替body对象的onscroll事件

**onscroll与onselectstart的使用[IE8以上浏览器未通过测试]**

[dom8-document-body.html]onscroll与onselectstart的使用示例

<html>

<head>

<title>document示例</title>

<script language="javascript" type="text/javascript">

<!--

function window\_onscroll(){

myHref.style.top=document.body.scrollTop+50;

myHref.style.left=document.body.scrollLeft;

}

function document\_onselectstart(){

alert("121");

return false;//返回false,可以禁止用户选网页中的文本；当用户选文本时会触发onselectstart事件，返回false就不会选中

//也可在body中加入onselectstart="return false"同样达到这个效果

}

//-->

</script>

<script language="javascript" for=document event=onselectstart>

<!--

//这样等于与在body上写入onselectstart="return false"

return document\_onselectstart();

//-->

</script>

</head>

<body onscroll="return window\_onscroll()">

<TEXTAREA id="Textareal" name="Textareal" rows="500" cols="500">

asdfzxcvqwerzasdvzxcfawdsf

</TEXTAREA>

<a id="myHref" href="http://www.sohu.com" style="left:0px;position:absolute;top:50px;word-break:keep-all">

<img src="sohu.jpg" />

</a>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

body对象

案例：

一个小太阳，当它碰到边框后，就自动的转变方向，很多流氓网站就有这种广告。

知识点：

1、document.body.clientWidth //获得客户端浏览器的宽度

2、document.body.clientHeight //获得客户端浏览器的高度

3、offsetWidth //获得当前对象的宽度

4、offsetHeight //获得当前对象的高度

综合示例

[dom9-domcument--body]示例

<html>

<head>

<title>document示例 </title>

<script language="javascript" type="text/javascript">

<!--

//定两个方向

var directX=1;//X轴的方向

var directY=1;//Y轴的方向

var sunX=0;//太阳的坐标

var sunY=0;

var speed=1;

function sunMove(){

sunX+=directX\*speed;

sunY+=directY\*speed;

var sundiv=document.getElementById("sundiv");

//修改div的left top

sundiv.style.top=sunY+"px";

sundiv.style.left=sunX+"px";

//判断什么时候转变方向

//判断X方向转向 (offsetWidth可以返回当前对象的实际宽度)

if(sunX+sundiv.offsetWidth>=document.body.clientWidth||sunX<=0){

directX=-directX;

}

//上面的if语句等价于下面的if

/\*if(sunX+sundiv.offsetWidth>=document.body.clientWidth){

divectX=-1;

}else if(sunX<=0){

divectX=1;

}\*/

//判断Y方向转向

if(sunY+sundiv.offsetHeight>=document.body.clientHeight||sunY<=0){

directY=-directY;

}

}

setInterval("sunMove()",10);

//-->

</script>

</head>

<body style="background-image: url('./ship.jpg');background-repeat:no-repeat;">

<div id="sundiv" style="position:absolute">

<img src="sun.gif" />

</div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--style对象**

**style对象**



style对象和document对象下的集合对象styleSheets有关系，styleSheets是文档中所有style对象的集合，这里讲解的重点是style对象，styleSheets不是重点。

**style对象定义：**

Represents the current settings of all possible inline styles for agiven element。即表示当前元素的样式设置。

[dom10-document-style.html]自由拖动DIV

<html>

<head>

<title>document示例--可拖动的DIV</title>

<script language="javascript" type="text/javascript">

<!--

var x=0,y=0,x1=0,y1=0;

var moveable=false;

var index=20000;

//开始拖动

function startDrag(obj,evt){

e=evt?evt:window.event;

if(!window.captureEvents){

obj.setCapture();

} else{

window.captureEvents(Event.MOUSEMOVE|Event.MOUSEUP);

}

var win=obj.parentNode;//取得父窗体(父窗体就是该div得上一级div)

win.style.zIndex=++index;//设置父窗体的z轴值

x=e.clientX;//取得当前鼠标的X坐标

y=e.clientY;//取得当前鼠标的Y坐标

x1=parseInt(win.style.left);//将父窗体的距浏览器左边界的距离转换为NUMBER

y1=parseInt(win.style.top);//将父窗体的距浏览器上边界的距离转换为NUMBER

moveable=true;

}

function drag(obj,evt){

e=evt?evt:window.event;

if(moveable){

var win=obj.parentNode;

win.style.left=x1+e.clientX-x;

win.style.top=y1+e.clientY-y;

}

}

//停止拖动

function stopDrag(obj){

if(moveable){

if(!window.captureEvents){

obj.releaseCapture();

}else{

window.releaseEvents(Event.MOUSEMOVE|Event.MOUSEUP);

}

moveable=false;

}

}

//-->

</script>

</head>

<body>

<div id="l1" style="position:absolute;width:200px;height:200px;background-color:#99CCFF;z-index:200;top:100px;left:154px;">

<div id="title" onMouseDown="startDrag(this,event)" onMouseMove="drag(this,event)" onMouseUp="stopDrag(this)" style="width:200px;height:20px;background-color:#330033;top:0px;left:0px;z-index:200;position:absolute;font-size:9pt;color:#FFFFFF;

padding-top:5px;padding-left:5px;cursor:hand;">浮动窗口

</div>

实例

</div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**style对象**

style对象不是针对某一个html元素，而是对所有的html元素而言的，也就是说，我们可以通过document.getElementById("id").style.property="值",来控制网页文档的任何一个元素(对象)的样式，当然这个很重要的。

js通过style对象把css结合起来的，所以html dom学习时要有css的基础。

style对象

使用style对象属性的语法：

document.getElementById("id号").style.property="值"，这里我们需要参考帮助文档，看看究竟有哪些property

**style对象常用的函数和属性**

**background属性**

|  |  |
| --- | --- |
| **属性** | **描述** |
| backaround | 在一行中设置所有的背景属性 |
| backgroundAttachment | 设置背景图像是否固定或随页面滚动 |
| backgroundColor | 设置元素的背景颜色 |
| backgroundImage | 设置元素的背景图像 |
| backgroundPosition | 设置背景图像的起如位置 |
| backgroundPositionX | 设置backgroundPosition属性的X坐标 |
| backgroundPositionY | 设置backgroundPosition属性的Y坐标 |
| backgroundRepeat | 设置是否及如何重复背景图像 |

[dom11-document.style.html]background属性的使用示例

<html>

<head>

<title>document示例--坦克转向</title>

<script language="javascript" type="text/javascript">

<!--

function change(obj){

if(obj.value=="上"){

tank.style.backgroundPositionY="0px";

}else if(obj.value=="右"){

tank.style.backgroundPositionY="120px";

}else if(obj.value=="下"){

tank.style.backgroundPositionY="80px";

}else if(obj.value=="左"){

tank.style.backgroundPositionY="40px";

}

}

//-->

</script>

</head>

<body>

<div id="tank" style="background-position-y:0px;background-image:url('./itank.jpg');width:40px;height:40px"></div><br>

<input type="button" value="上" onclick="change(this)" />

<input type="button" value="右" onclick="change(this)" />

<input type="button" value="下" onclick="change(this)" />

<input type="button" value="左" onclick="change(this)" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom详解--style对象**

web坦克大战1.0版

[dom12-document-style.html]

<html>

<head>

<title>document示例--坦克转向</title>

</head>

<body onkeydown="dosomething(event)">

<div id="filed" style="width:500px;height:400px; background-color:black;position:absolute;">

<div id="mytank" style="background-position-y:0px;background-image:url('./itank.jpg');width:40px;height:40px;position:absolute;"></div><br>

</div>

<script language="javascript" type="text/javascript">

<!--

//用面向对象的方法开发web版本坦克大战1.0版(可以通过asdw键来控制坦克的走向)

function MyTank(x,y,direct){

this.x=x;//坦克的横坐标

this.y=y;//坦克的纵坐标

this.direct=direct;//方向

this.speed=1;//速度

//初始化

mytank.style.left=this.x+"px";

mytank.style.top=this.y+"px";

mytank.style.backgroundPositionY="0px";

//event表示按键事件

this.move=function move(event){

//a表示向左 3、s表示向下 2、d表示向右 1、w表示向上 0

switch(event.keyCode){

case 65://向左

this.x-=this.speed;

this.direct=3;

mytank.style.backgroundPositionY="40px";

break;

case 83://向下

this.y+=this.speed;

this.direct=2;

mytank.style.backgroundPositionY="80px";

break;

case 68://向右

this.x+=this.speed;

this.direct=1;

mytank.style.backgroundPositionY="120px";

break;

case 87://向上

this.y-=this.speed;

this.direct=0;

mytank.style.backgroundPositionY="0px";

break;

}

//统一改变位置

mytank.style.left=this.x+"px";

mytank.style.top=this.y+"px";

}

}

//判断用户的操作

function dosomething(event){ if(event.keyCode==65||event.keyCode==83||event.keyCode==68||event.keyCode==87){

hero.move(event);

}

}

//创建坦克

var hero=new MyTank(150,360,0);

//-->

</script>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**style对象**

1、backgroundRepeat 设置是否及如何重复背景图像

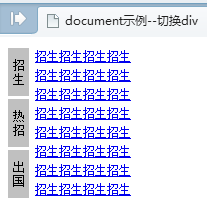
2、border 在一行设置四个边框的所有属性

3、display 设置元素显示方式

4、width 设置元素的宽度

5、height 设置元素的高度

6、visibility 属性设置元素是否可见



div切换示例

[dom13-document-style.html]

<html>

<head>

<title>document示例--切换div</title>

<link rel="stylesheet" type="text/css" href="./dom13css.css">

<script language="javascript" type="text/javascript">

<!--

function change(val,obj){

/\* if(val=="zs"){

zs.style.visibility="visible";//visibility只是隐藏

rz.style.visibility="hidden";

cg.style.visibility="hidden";

}else if(val=="rz"){

zs.style.visibility="hidden";

rz.style.visibility="visible";

cg.style.visibility="hidden";

}else if(val=="cg"){

zs.style.visibility="hidden";

rz.style.visibility="hidden";

cg.style.visibility="visible";

}\*/

switch(val){

case "zs":

obj.style.backgroundColor="#FFC12D";

zs.style.display="block";

rz.style.display="none";

cg.style.display="none";

break;

case "rz":

obj.style.backgroundColor="#FFC12D";

zs.style.display="none";

rz.style.display="block";

cg.style.display="none";

break;

case "cg":

obj.style.backgroundColor="#FFC12D";

zs.style.display="none";

rz.style.display="none";

cg.style.display="block";

break;

}

}

function change2(obj){

obj.style.backgroundColor="silver";

}

//-->

</script>

</head>

<body>

<div class="div1">

<!--导航DIV-->

<div class="navi">

<ul>

<li id="zsli" onmouseover="change('zs',this)" onmouseout="change2(this)">招生</li>

<li id="rzli" onmouseover="change('rz',this)" onmouseout="change2(this)">热招</li>

<li id="cgli" onmouseover="change('cg',this)" onmouseout="change2(this)">出国</li>

</ul>

</div>

<!--超链接DIV3个-->

<div id="zs" class="zs">

<ul>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

<li><a href='#'>招生招生招生招生</a></li>

</ul>

</div>

<div id="rz" class="rz" >

<ul>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

<li><a href='#'>热招热招热招热招</a></li>

</ul>

</div>

<div id="cg" class="cg">

<ul>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

<li><a href='#'>出国出国出国出国</a></li>

</ul>

</div>

</div>

</body>

</html>

---------------------------------------------------------------------------------------------------------------------------------

[dom13css.css]样式表选择器

body{

font-size:12px;

}

.div1{

width:126px;

height:156px;

}

.navi{

width:21px;

height:156px;

float:left;

}

.navi ul{

padding:0px;

margin-left:0px;

margin-top:0px;

float:left;

}

.navi ul li{

list-style-type:none;

width:21px;

height:38px;

margin-top:3px;

text-align:center;

padding-top:10px;

background-color:silver;

float:left;

}

.zs,.rz,.cg{

width:100px;

height:156px;

margin-left:4px;

float:left;

}

.zs ul,.rz ul,.cg ul{

padding:0px;

margin-top:2px;

margin-left:0px;

float:left;

}

.zs ul li,.rz ul li,.cg ul li{

list-style-type:none;

margin-top:0px;

margin-left:2px;

line-height:19px;

float:left;

}

.rz,.cg{

/\*visibility:hidden;\*/

display:none;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**style对象**

**Positioning属性**

**属性 描述**

botton 设置元素的底边缘距离父元素底边缘的之上或之下的距离

left 置元素的左边缘距离父元素左边缘的左边或右边的距离

position 把元素放置在static,relative,absolute或fixed的位置

right 置元素的右边缘距父元素右边缘的左边或右边的距离

top 设置元素的顶边缘距离父元素顶边缘的之上或之下的距离

zIndex 设置元素的堆叠次序

**style对象**

**Table属性**

**属性 描述**

borderCollapse 设置表格边框是否合并为单边框，或者像在标准的HTML中那样分离

borderSpacing 设置分隔单元格边框的距离

captionSide 设置表格标题的位置

emptyCells 设置是否显示表格中的空单元格

tableLayout 设置用来显示表格单元格、行以及列的算法

**style对象**

**Text属性**

**属性 描述**

color 设置文本的颜色

font 在一行设置所有字体属性

fontFamily 设置元素的字体系列

fontSize 设置元素的字体大小

fontSizeAdjust 设置/调整文本的尺寸

fontStretch 设置如何紧缩或伸展字体

fontStyle 设置元素的字体样式

fontVariant 用小型大写字母字体来显示文本

fontWeight 设置字体的粗细

letterSpacing 设置字符间距

lineHeight 设置行间距

guotes 设置在文本中使用哪种引号

textAlign 排列文本

textDecoration 设置文本的修饰

textlndent 缩紧首行的文本

textShadow 设置文本的阴影效果

textTransform 对文本设置大写效果

unicodeBidi

whiteSpace 设置如何设置文本中的折行和空白符

wordSpacing 设置文本中的词间距

**style对象**

现在我们对style对象有了基本的了解，可以做几个实例

案例：完成功能，鼠标点击+号显示家庭成员，点击-号则收回+

[dom14-document-style.html]树状展开和关闭

<html>

<head>

<title>document示例--树状展开和关闭</title>

<script language="javascript" type="text/javascript">

<!--

function test1\_onclick(){

if(test1.innerText=="-"){

myList.style.display="none";

test1.innerText="+";

}else{

myList.style.display="block";

test1.innerText="-";

//怎样体现body的all这个对象的作用

//因为你可以通赤body的all来访问body中所有的控件(通过id)

//所以代码可以写成

//document.body.all.test1.innerText="-";

//document.body.all("text1").innerText="-";

//document.body.all.item("text1").innerText="-";

}

}

//-->

</script>

</head>

<body>

<p>

<FONT face="宋体">

<span id=test1 style="border-left-color:#ff0099;border-bottom-color:#ff0099;cursor:hand;border-top-style:ridge;border-right-color:#ff0099;border-bottom-style:ridge" language=javascript onclick="return test1\_onclick()">-</span>

myFamily

</FONT>

</p>

<ul id=myList>

<li><font face="宋体">爸爸</font></li>

<li><font face="宋体">妈妈</font></li>

<li><font face="宋体">弟弟</font></li>

</ul>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**style对象**

使用js实现购物车的功能(简易功能)

[dom15-document-style.html]

<html>

<head>

<title>document示例--购物车简易功能</title>

<script language="javascript" type="text/javascript">

<!--

function gouwu(obj){

var fruits=document.getElementsByName("fruit");

var totalPrice=0;

//遍历所有的checkbox，计算新的总价

for(var i=0;i<fruits.length;i++){

//判断是否选中

if(fruits[i].checked){

totalPrice+=parseFloat(fruits[i].value);

}

}

myspan.innerText=totalPrice;

}

//-->

</script>

</head>

<body>

<h1>我的购物车</h1>

<input type="checkbox" name="fruit" onclick="gouwu(this)" value="10"/> 苹果 10元<br>

<input type="checkbox" name="fruit" onclick="gouwu(this)" value="20"/> 香蕉 20元<br>

<input type="checkbox" name="fruit" onclick="gouwu(this)" value="30"/> 西瓜 30元<br>

<input type="checkbox" name="fruit" onclick="gouwu(this)" value="40"/> 栗子 40元<br>

<input type="checkbox" name="fruit" onclick="gouwu(this)" value="50"/> 哈密瓜 50元<br>

总价格是：<span id="myspan">0</span>元

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[dom16-document-style.html]

<html>

<head>

<title>document示例--课程选择</title>

<select id="myCourse" onchange="getCourse();">

<option value="" selected>--请选择一门课程--</option>

</select>

<textarea id="myares" cols=30 rows=10></textarea>

<script language="javascript" type="text/javascript">

<!--

var last\_select\_num=3;//加入从数据库中查询，发现用户上次选择了3门课程

//动态添加第一门课程java

var myOpteion=document.createElement("OPTION");

myOpteion.value="java";

myOpteion.text="java";

myCourse.add(myOpteion);

//动态添加第一门课程oracle

var myOpteion=document.createElement("OPTION");

myOpteion.value="oracle";

myOpteion.text="oracle";

myCourse.add(myOpteion);

//动态添加第一门课程j2ee

var myOpteion=document.createElement("OPTION");

myOpteion.value="j2ee";

myOpteion.text="j2ee";

myCourse.add(myOpteion);

function getCourse(){

//注意这两种方式的结果，有什么区别

myares.value+="你选中了:"+myCourse.value+"\r\n";

// myares.value="你选中了:"+myCourse.options[myCourse.selectedIndex].innerText;

}

//-->

</script>

</head>

<body>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[dom17-document-style.html]

<html>

<head>

<title>document示例--全选功能</title>

<script language="javascript" type="text/javascript">

<!--

function selectCheck(obj){

var fruits=document.getElementsByName("fruit");

if(obj.innerText=="全选"){

for(i=0;i<fruits.length;i++){

fruits[i].checked=true;

}

}else if(obj.innerText=="取消"){

for(i=0;i<fruits.length;i++){

fruits[i].checked=false;

}

}

}

//响应复选框

function selectCheck2(){

var fruits=document.getElementsByName("fruit");

if(myselect.checked){

for(i=0;i<fruits.length;i++){

fruits[i].checked=true;

}

}else{

for(i=0;i<fruits.length;i++){

fruits[i].checked=false;

}

}

}

//-->

</script>

</head>

<body>

<input type="checkbox" name="fruit"/> 苹果<br>

<input type="checkbox" name="fruit"/> 香蕉<br>

<input type="checkbox" name="fruit"/> 西瓜<br>

<input type="checkbox" name="fruit"/> 栗子<br>

<input type="checkbox" name="fruit"/> 哈密瓜<br>

<a href="#" onclick="selectCheck(this)">全选</a>

<a href="#" onclick="selectCheck(this)">取消</a><br>

<input type="checkbox" id="myselect" onclick="selectCheck2()"/> 是否全选

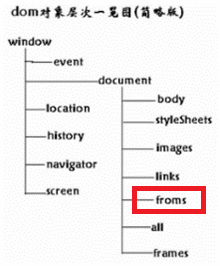
</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--forms对象(集合)和form对象**

**forms对象(集合)**



**forms对象(集合)定义：**

Retrieves a collection,in source order,of all form objects in the document，即按照表单在文档中的顺序得到form对象，forms对象集合中包括了当前文档的所有form对象。

讲forms对象(集合)的目的其实是为了讲解form对象，form对象会单独讲解。

**forms常用的函数和属性，其它的参考帮助文档**

1、length 设置或得到集合大小

2、item() 通过序列号得到form元素

3、namedItem() 通过名字得到form元素

**forms函数和属性的使用**

[dom18-document-forms]

<!DOCTYPE html>

<html>

<head>

<title>dom18-doucment-forms.html</title>

<meta name="keywords" content="keyword1,keyword2,keyword3">

<meta name="description" content="this is my page">

<meta name="content-type" content="text/html; charset=">

<!--<link rel="stylesheet" type="text/css" href="./styles.css">-->

<script type="text/javascript" language="javascript">

<!--

function test(){

//如何取得所有的表单

var allforms=document.forms;

//可以通过allforms去访问指定表单

//alert(allforms[0].username.value); 与下面的语句等价

alert(allforms.item(0).username.value);

}

//-->

</script>

</head>

<body>

<h1>个人信息</h1>

<form action="">

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

P:<input type="password" name="pwd" /><br>

<input type="submit" value="提交" />

</form>

<h1>兴趣爱好</h1>

<form action="">

爱好1:<input type="text" name="hobby1"/><br>

爱好2:<input type="text" name="hobby2" /><br>

<input type="submit" value="提交" />

</form>

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

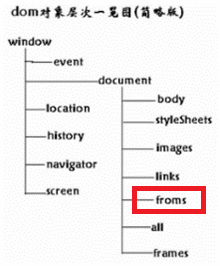
</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

dom对象详解--forms对象(集合)和form对象

**form对象**



form对象代表一个HTML表单在HTML文档中<form>每出现一次，form对象就会被创建。

从dom对象层次图看，document.forms对象是当前文档所有form对象的集合。

[dom19-document-forms.html]用户注册案例

<!DOCTYPE html>

<html>

<head>

<title>dom19-document-forms.html</title>

<meta name="keywords" content="keyword1,keyword2,keyword3">

<meta name="description" content="this is my page">

<meta name="content-type" content="text/html; charset=">

<link rel="stylesheet" type="text/css" href="./dom19css.css">

<script type="text/javascript" language="javascript">

<!--

function checkinfo(){

span1.innerText="";

span2.innerText="";

span3.innerText="";

//获取表单的用户名

if(document.forms[0].username.value.length<4||document.forms[0].username.value.length>6){

span1.innerText="\*用户名长度为4-6之间";

return false;

}

//获取密码并判断

if(document.forms[0].pwd1.value.length<=3){

span2.innerText="\*密码不能为空，不能少于3位";

return false;

}

if(document.forms[0].pwd1.value!=document.forms[0].pwd2.value){

span3.innerText="\*两次密码要一样";

return false;

}

}

//-->

</script>

</head>

<body>

<!--<form action="" onsubmit="函数名">也可通过此方法验证-->

<form action="">

<h1>用户注册</h1>

用&nbsp户&nbsp名：<input type="text" name="username" /><span id="span1"></span><br>

密&nbsp&nbsp&nbsp&nbsp码：<input type="password" name="pwd1" /><span id="span2"></span><br>

确认密码：<input type="password" name="pwd2" /><span id="span3"></span><br>

年&nbsp&nbsp&nbsp&nbsp龄：<input type="text" name="age" /><br>

电子邮件：<input type="text" name="email" /><br>

电话号码：<input type="text" name="phone" /><br>

<input type="submit" value="提交注册内容" onclick="return checkinfo()" />

<input type="reset" value="重置" />

</form>

</body>

</html>

---------------------------------------------------------------------------------------------------------------------------------

[dom19css.css]样式表

#span1,#span2,#span3{

font-size:12px;

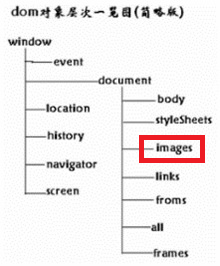
color:red;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

dom对象详解--images对象(集合)/img对象

**images对象(集合)**



images对象(集合)定义：

Retrieves a collection,in source order,of img objects in the document，即按照img在文档中的顺序得到img对象，images对象集合中包括了当前文档的所有img对象。

讲images对象(集合)的目的其实是为了讲解img对象。

**images常用的函数和属性**

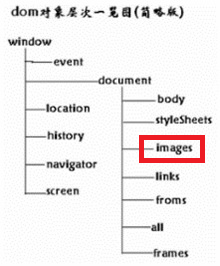
1、length 设置或得到集合大小

2、item() 通过序列号得到img元素

3、namedItem() 通过名字得到img元素

dom对象详解--images对象(集合)/img对象

img对象



img对象代表一个图片在HTML文档中<img>每出现一次，就会创建一个img对象。

从dom对象层次图看，document.images对象是当前文档所有img对象的集合。

**img对象常用的属性**

1、id 设置或得到该图片id

2、name 设置或得到该图片name

3、src 设置或得到该图片src

4、width 设置或得到该图片width

5、height 设置或得到该图片height

6、style 设置或得到该图片style，通过style对象的各个属性，可以控制图片更 多的样式。

**img对象**

**image对象的事件句柄**

**事件句柄 描述**

onabort 当用户放弃图像的装载时调用的事件句柄

onerror 在装载图像的过程中发生错误时调用的事件句柄

onload 当图像装载完毕时调用的事件句柄

**案例**

[dom20-document-images.html]

<!DOCTYPE html>

<html>

<head>

<title>domcument-images</title>

<script type="text/javascript" language="javascript">

<!--

function test1(){

myid.innerText="no ok";

}

function test2(){

myid.innerText="ok";

}

//-->

</script>

</head>

<body>

<!--文件名正确显示ok否则no ok-->

<img src="sun.gif" onerror="test1()" onload="test2()" />

<span id="myid"></span>

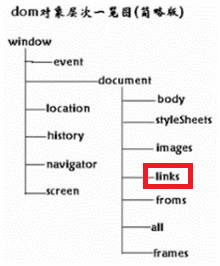
</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**dom对象详解--links对象(集合)/link对象**

**links对象(集合)**



links对象(集合)定义：

Retrieves a collection of all a objects that specify the HREF property and all area objects in the document检索所有指定的HREF属性的对象的集合和文档中的所有区域中的对象

讲links对象(集合)，就自然的引出了link对象。

links对象(集合)

**links常用的函数和属性**

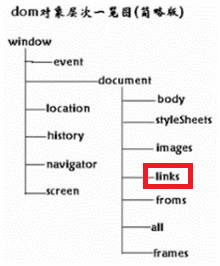
1、length 设置或得到集合大小

2、item() 通过序列号得到超链接对象

3、namedItem() 通过名字得到超链接对象

dom对象详解--links对象(集合)/link对象

link对象



link对象代表一个超链接在HTML文档中<a>每出现一次，就会创建一个link对象。

从dom对象层次图看document.links对象是当前文档所有link对象的集合。

**link常用的属性**

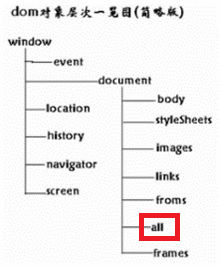
1、id 设置或得到该超链接id

2、name 设置或得到该超链接name

3、href 设置或得到该超链接href

**dom对象详解--all对象(集合)**

**all对象(集合)**



Returns a reference to the collection of elements contained by the object

返回参考要素所包含的对象的集合

all对象(集合)，能得到当前文档的所有元素，一般在遍历文档的时候使用，在实际开发中用的不是很多。

案例

[dom21-document-images.html]

<html>

<head>

<title>domcument-images</title>

<script type="text/javascript" language="javascript">

<!--

function test1(){

myid.innerText="no ok";

}

function test2(){

myid.innerText="ok";

}

function abc(){

var myall=document.all;

for(var i=0;i<myall.length;i++){

alert(myall[i].nodeType+" "+myall[i].nodeName);

}

}

//-->

</script>

</head>

<body>

<!--文件名正确显示ok否则no ok-->

<img src="sun.gif" onerror="test1()" onload="test2()" />

<span id="myid"></span><br>

<input type="button" value="测试" onclick="abc()"/>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

dom对象详解--其它html dom对象

**其它html dom对象**

img对象对于images对象集合，link对象对应links对象集合，form对象对应forms对象集合，frame对象对应frames对象集合...实际上html元素还有很多，比如button/select/radio/text/option/table...这些也是html dom对象，它们的使用和image对象，link对象非常相似，相信大家可以举一反三。

dom对象详解--其它html dom对象

table对象

鉴于table对象在web编程中的重要性，此处重点讲解table对象。

table对象代表一个html表格。

在HTML文档中<table>标签每出现一次，一个table对象就会被创建。

**table对象常用的属性**

1、id 设置或得到该表格的id

2、name 设置或得到该表格的name

3、border 设置或得到该表格的的边框

4、width 设置或得到该表格的的宽度

5、height 设置或得到该表格的的高度

6、cellpadding 设置或返回单元格内容和单元格边框之间的空白量

7、cellspacing 设置或返回在表格中的单元格之间的空白量

8、style 设置或得到该表格的style对象

**table对象常用的属性(对象集合)**

1、rows[] 表格中所有行的一个数组(针对表格)

2、cells[] 表格中所有单元格的一个数组(针对一行)

**rows与cells示例**

[dom22-document.table.html]

<html>

<head>

<title>domcument-table</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

//取出所有行

/\* var myrows=mytab.rows;

alert(myrows.length);

alert(myrows[1].cells[0].innerText+""+myrows[1].cells[1].innerText+""+myrows[1].cells[2].innerText);

\*/

//mytab.rows表示所有行

for(var i=0;i<mytab.rows.length;i++){

//取出一行

var eachRow=mytab.rows[i];

//对该行遍历

//eachRow.cells表示一行的所有列

for(var j=0;j<eachRow.cells.length;j++){

alert(eachRow.cells[j].innerText);

}

}

}

//-->

</script>

</head>

<body>

<h3>水浒英雄排行榜</h3>

<table id="mytab" border="1px">

<tr><td>排名</td><td>姓名</td><td>外号</td></tr>

<tr><td>1</td><td>宋江</td><td>及时雨</td></tr>

<tr><td>2</td><td>卢俊义</td><td>玉麒麟</td></tr>

</table>

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**table对象常用属性**

**1、deleteRow 删除行**

**2、insertRow 添加行**

**3、insertCell 添加列**

案例

[dom23-document-table.html]

<html>

<head>

<title>domcument-table</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

//取出所有行

/\* var myrows=mytab.rows;

alert(myrows.length);

alert(myrows[1].cells[0].innerText+" "+myrows[1].cells[1].innerText+" "+myrows[1].cells[2].innerText);

\*/

//mytab.rows表示所有行

for(var i=0;i<mytab.rows.length;i++){

//取出一行

var eachRow=mytab.rows[i];

//对该行遍历

//eachRow.cells表示一行的所有列

for(var j=0;j<eachRow.cells.length;j++){

alert(eachRow.cells[j].innerText);

}

}

}

//演示删除一行

function test2(){

mytab.deleteRow(1);

}

//演示插入一行

function test3(){

var tableRow=mytab.insertRow(3);

tableRow.insertCell(0).innerText=pm.value;

tableRow.insertCell(1).innerText=xm.value;

tableRow.insertCell(2).innerText=wh.value;

}

//-->

</script>

</head>

<body>

<h3>水浒英雄排行榜</h3>

<table id="mytab" border="1px">

<tr><td>排名</td><td>姓名</td><td>外号</td></tr>

<tr><td>1</td><td>宋江</td><td>及时雨</td></tr>

<tr><td>2</td><td>卢俊义</td><td>玉麒麟</td></tr>

</table>

<input type="button" onclick="test()" value="tesing" /><br>

排名<input id="pm" type="text" value=""/>姓名<input id="xm" type="text" />外号<input id="wh" type="text" /><input type="button" value="添加一行" onclick="test3()"/><br>

<input type="button" onclick="test2()" value="删除一行" />

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[dom24-document-table.html]

<html>

<head>

<title>domcument-table</title>

<script type="text/javascript" language="javascript">

<!--

function addHero(){

//遍历整个表格看看有没有重复的编号

for(var i=0;i<mytab.rows.length;i++){

var eachRow=mytab.rows[i];

if(eachRow.cells[0].innerText==pm.value){

alert("编号"+pm.value+"重复，请修改编号");

return;

}

}

var tableRow=mytab.insertRow(mytab.rows.length);

tableRow.insertCell(0).innerText=pm.value;

tableRow.insertCell(1).innerText=xm.value;

tableRow.insertCell(2).innerText=wh.value;

}

//-->

</script>

</head>

<body>

<h3>水浒英雄排行榜</h3>

<table id="mytab" border="1px">

<tr><td>排名</td><td>姓名</td><td>外号</td></tr>

<tr><td>1</td><td>宋江</td><td>及时雨</td></tr>

<tr><td>2</td><td>卢俊义</td><td>玉麒麟</td></tr>

</table>

<h3><font color="red">添加新英雄</font></h3>

排名<input id="pm" type="text" /><br>

姓名<input id="xm" type="text" /><br>

外号<input id="wh" type="text" /><br>

<input type="button" value="添加" onclick="addHero()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**为什么需要正则表达式？**

**正则表达式(RegExp)**

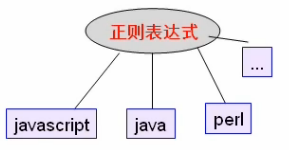
如何按一定规则快速查找到需要找寻的内容，js的设计者们给我们提供了一个叫正则表达式(RegExp对象)，专门用于处理类似问题。

RegExp对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

正则表达式--基本概念

在计算机科学中，是指一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串。在很多文本编辑器或其他工具里，正则表达式通常被用来检索或替换那些符合某个模式的文本内容。许多程序设计语言都支持利用正则表达式进行字符串操作。

**特别强调：**正则表达式不是js的专利产品，实际上很多程序设计语言都支持正则表达式进行字符串操作!



一个正则表达式，就是用某种模式去匹配一类字符串的一个公式。很多人因为它们看上去比较古怪而且复杂所以不敢去使用，不过，经过一点点练习之后，就觉得这些复杂的表达式其实写起来还是相当简单的，而且，一旦你弄懂它们，你就能把数小时辛苦而且易错的文本处理工作缩短在几分钟(甚至几秒钟)内完成。

快速入门案例

给你一个字符串(或一篇文章)，请你找出所有四个数字连在一起的子串？

[01.html]入门案例

<html>

<head>

<title>dom示例--RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

//得到用户的内容

var con=content.innerText;

var myReg=/(\d){4}/gi;//这是一个正则表达式对象，可以检索4个连续的数

while(res=myReg.exec(con)){ //res是检索出来的结果，但是一个结果就对应一个数组，该数组的res[0]就是找到的那个文本

alert("找到"+res[0]);

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

小结：

创建一个RegExp对象有两种方法：

var regExp=/正则表达式/gi; (隐式创建)

var regExp=new RegExp(正则表达式,"gi"); (显示创建)

**特别说明：**g表示正则表达式执行全局匹配；i表示正则表达式不区分大小写的匹配;m表示多行模式

**正则表达式一览表**

|  |  |  |  |
| --- | --- | --- | --- |
| **表达式** | **描述** | **表达式** | **描述** |
| ^ | 匹配行首 | $ | 匹配行尾 |
| . | 匹配任何单个字符 | [] | 匹配集合中的字符 |
| [^] | 匹配不在集合中的字符 | ? | 匹配前一个样式0或1次 |
| \* | 匹配前一个样式0或更多次 | + | 匹配前一个样式1或更多次 |
| | | 匹配前一个或后一个样式 | & | 匹配前一个样式0或更多次 |
| # | 匹配前一个样式1或更多次 | {} | Group char to form one pattem |
| () | Group and remember | \ | 下一个字符直接用原意(只对a-z之外的字符有效) |
| < | 匹配词的开始 | > | 匹配词的结尾 |
| \X## | 匹配ASCII码为##(16进制)的字符 | \N## | 匹配ASCII码为##(10进制)的字符 |
| \O## | 匹配ASCII码为##(8进制)的字符 | \a | 匹配\a |
| \b | 匹配\b | \f | 匹配\f |
| \r | 匹配0x13(回车) | \t | 匹配0x09(跳格) |
| \v | 匹配\v | \e | 匹配Esc(0x05) |
| \n | 匹配0x10(换行) | \s | 匹配空格(回车、换行、制表符、空格) |
| \S | 匹配非空格(!\S) | \w | 整字匹配 |
| \W | 非整字匹配 | \d | 匹配数字 |
| \D | 匹配非数字 | \U | 匹配大写字母 |
| \L | 匹配小写字母 | \C | 后面开始大小写敏感 |
| \c | 后面开始大小写不敏感 | \# | 匹配前面定义的组(#:1-9,a-f) |

正则表达式对象

RegExp对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

创建RegExp对象有以下几种方法：

1、隐式创建RegExp对象

它的创建形式是/pattern/[flag]这种形式；

说明：pattern是必需的，flag是可选择的。

pattern就是按照正则表达式的模式进行编写，flag是可以选择的，它主要有以下几种标识

符：g全局标志、i忽略大小写、m用作多行标志。

2、显示的创建RegExp对象

它的创建形式是new RegExp("pattern"[,"flag"]);

说明：pattern是必需的，flag是可选择的。

pattern就是按照正则表达式的模式进行编写，flag是可以选择的，它主要有以下几种标识符：g全局标志、i忽略大小写、m用作多行标志。

**RegExp对象常用的方法**

1、exec() 检索字符串中指定的值，并返回值(找不到返null)

2、test() 检索字符串中指定的值，返回true或false

test()示例

[02.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.innerText;

var myreg=/abc/gi;

if(myreg.test(con)){

alert("有abc");

}else{

alert("没有abc");

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

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

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**string对象与正则表达式有关的方法**

常用的方法

1、match()

语法格式为：match(reExp)

与RegExp对象的exec方法类似，它使用正则表达式模式对字符串执行搜索，并返回一个包含搜索结果的数组。

2、search()

语法格式为：search(reExp)

它返回使用正则表达式搜索时，第一个匹配的字符串在整个被搜索的字符串中的位置

3、replace()

语法格式为：replace(reExp,replaceText)

使用正则表达式模式对字符串执行搜索，并对搜索到的内容用指定的字符替换，返回值为包含替换后的内容的字符串对象。

4、split()

语法格式为：split(regExp) 或 split("按什么字符分割")

把字符串分割为字符串数组

search()/match()/replace()的使用示例

[03.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

//search()返回查找字符串出现的位置

var str="Visit W3School!";

alert(str.search(/W3School/));

//match()返回检索指定的值

function test(){

var con=content.innerText;

var myreg=/abc/gi;

res=con.match(myreg);

for(var i=0;i<res.length;i++){

alert(i+" "+res[0]);

}

}

//replace()查找替换

function test2(){

var con1=content.innerText;

var newcon=con1.replace(/sel.value/gi,rec.value);

content2.innerText=newcon;

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

查找内容<input type="text" id="sel" /><br>

替换内容<input type="text" id="rec" /><br>

<input type="button" value="替换" onclick="test2()"/>

<textarea id="content2" rows="10" cols="20"></textarea><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**RegExp对象的属性**

RegExp对象属性分为静态属性和实例属性，所谓静态属性是指：可以通过RegExp就可以访问。所谓实例属性是指：通过RegExp对象实例才能访问使用。

**RegExp有哪些静态属性**

1、index 是当前表达式模式首次匹配内容的开始位置，从0开始计数，其初始值为-1，每次成功匹配时，index属性都会随之改变

2、lastindex 是当前表达式模式首次匹配内容中最后一个字符的下一个位置，从0开始计数

3、input 返回当前所作用的字符串

4、leftContext 是当前表达式模式最后一个匹配字符串左边的所有内容

5、rightContext 是当前表达式模式最后一个匹配字符串右边的所有内容

静态属性index/leftContext/rightContext的使用示例

[04.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

//静态属性index/leftContext/rightContext

function test(){

var con=content.innerText;

var myReg=/(\d){4}/gi;

while(res=myReg.exec(con)){

alert("index="+RegExp.index);//index是取检索内容匹配位置的值

alert("lefttext="+RegExp.leftContext);//leftContext是取检索内容匹配位置左边的内容

alert("righttext="+RegExp.rightContext);//rightContext是取检索内容匹配位置右边的内容

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

</body>

</html>

**特别注意：**RegExp静态属性的使用方式为：**RegExp.属性名**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**RegExp对象的实例属性**

1、global 返回创建RegExp对象实例时指定的global标志(g)的状态(true,false)

2、ignoreCase 返回创建RegExp对象实例时指定的ignreCase标志(i)的状态(true,false)

3、multiLine 返回创建RegExp对象实例时指定的multiLine标志(m)的状态(true,false)

4、source 返回创建RegExp对象实例时指定的表达式文本字符串

**正则表达式的语法--子表达式、捕获、反向引用**

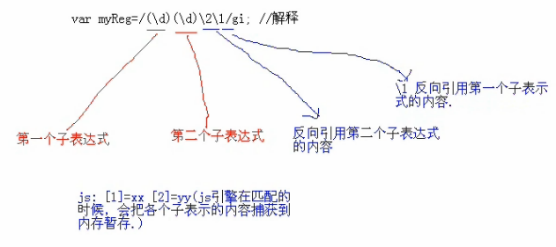
子表达式、捕获、反向引用

如果你认为正则表达式只能检索固定的字符串或是连续的几个数字，那你就太小看正则表达了。

案例：

给你一个字符串(或一篇文章)，请你找出所有四个数字连在一起的子串，并且这四个数字

要满足1、第一位与第四位相同；2、第二位与第三位相同，比如1221，5775，...



[05.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.innerText;

var myReg=/(\d)(\d)\2\1/gi;//()包起来为子表达式，\2反向引用第二个子表达式，\1反向引用第一个子表达式

while(res=myReg.exec(con)){

alert(res[0]);

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例:查找aabbccdd格式的数字

[06.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.innerText;

var myReg=/(\d)\1(\d)\2(\d)\3(\d)\4/gi;

while(res=myReg.exec(con)){

alert(res[0]);

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例：找出abbba规则的数字

[07.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.innerText;

var myReg=/(\d)(\d)\2\2\1/gi;

while(res=myReg.exec(con)){

alert(res[0]);

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例：12321-444999111前5个任意数字后9个数要按规则来写

[08.html]

<html>

<head>

<title>RegExp</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.innerText;

var myReg=/(\d){5}\-(\d)\2\2(\d)\3\3(\d)\4\4/gi;

while(res=myReg.exec(con)){

alert(res[0]);

}

}

//-->

</script>

</head>

<body>

<textarea id="content" rows="10" cols="20"></textarea><br>

<input type="button" value="测试" onclick="test()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**正则表达式的语法--元字符详解**

**元字符**

如果要想灵活的运用正则表达式，必需了解其中各种元字符的功能，元字符从功能上大致分为：

1、限定符；2、选择匹配符；3、分组组后和反向引用符；4、特殊字符；5、字符匹配符；6、定位符

**元字符--限定符**

用于指定其前面的字符和组合项连续出现多少次

{n}说明：n表示出现的次数。比如a{3},1{4},(\d){2}

但是这里要注意一点，1{3}去匹配1111111的话，会是到什么结果呢？111 111

**元字符--限定符**

{n,m}说明：n表示至少出现的n次最多m次。比如a{3,4},1{4,5},(\d){2,5}

我们看看1{3,4}去匹配1111111的话，会得到什么结果呢？1111 111

js在默认匹配中使用的是贪婪匹配的原则，即尽可能匹配多的字符串。

**元字符--限定符**

+说明：+表示出现1次到任意多次。比如/a+/gi,/1+/gi,/(\d)+/gi

看看/1+/gi去匹配1111111的话会得到什么结果呢？1111111

**元字符--限定符**

\*说明：\*表示出现0次到任意多次。比如/a\*/gi,/1\*/gi,/(\d)\*/gi

看看/a1\*/gi去匹配a111的话会得到什么结果呢？a111

**元字符--限定符**

?说明：?表示出现0次到任意多次。比如/a?/gi,/1?/gi,/(\d)?/gi

看看/a1?/gi去匹配a111的话会得到什么结果呢？a1

**元字符--字符匹配符**

[a-z]说明：[a-z]表示可以匹配a-z中任意一个字符。比如：/[a-z]/gi,/[a-z]{2}/gi

看看/[A-Z]/gi去匹配allc8会得到什么结果？a l l c

**元字符--字符匹配符**

[A-Z]表示可以匹配A-Z中任意一个字符

[0-9]表示可以匹配0-9中任意一个字符

**元字符--字符匹配符**

[^a-z]说明：[^a-z]表示可以匹配不是a-z中的任意一个字符。

比如/[^a-z]/gi,/[^a-z]{2}/gi

看看/[a-z]/gi去匹配allc8会得到什么结果？a l l c

用/[^a-z]{2}/gi又会得到什么结果？得不到任何结果

用/[^a-z]/gi又会得到什么结果？ 8

**元字符--字符匹配符**

[^A-Z]表示可以匹配不是A-Z中任意一个字符

[^0-9]表示可以匹配不是0-9中任意一个字符

**元字符--字符匹配符**

[abcd]表示可以匹配abcd中的任意一个字符。

[^abcd]表示可以匹配不是abcd中的任意一个字符。

当然上面的abcd你可以根据实际情况修改，以适应你的需求。

**元字符--字符匹配符**

\d 表示可以匹配0-9的任意一个数字，相当于[0-9]

\D 表示可以匹配不是0-9中的任意一个数字，相当于[^0-9]

\w 匹配任意英文字符、数字和下划五，相当于[a-zA-Z0-9\_]

\W 相当于[^a-zA-Z0-9\_]是\w刚好相反

\s 匹配任何空白字符(空格，制表符等)

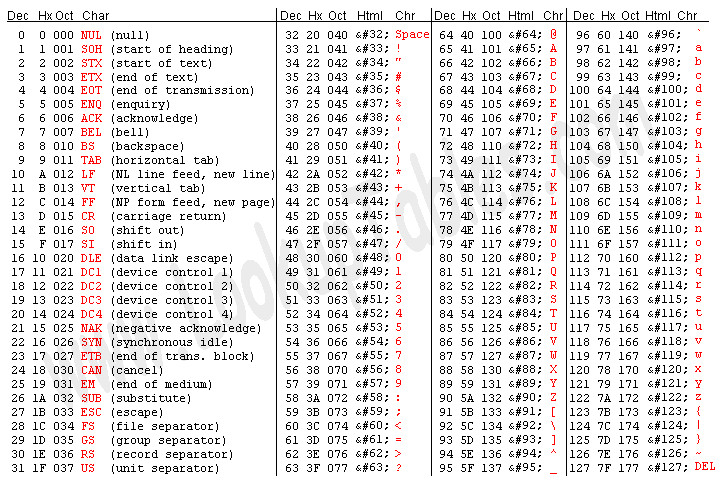
\S 匹配任何非空白字符，和\s刚好相反

. 匹配出\n之外的所有字符，如果要匹配本身则需要使用\.

思考题：写一个正则表达式可以匹配任意三个连续的字符 /([\d\D])\1{2}/gi

**元字符--特殊字符匹配**

在字符中，可能你会遇到比较特殊的字符，针对这种情况，正则表达式通过\xn的方式来匹配\xn的x是固定的，n是一个十六进制的数，比如\x21就是去匹配ascii码表中十六进制是21的字符，查表可知就是!



**元字符--定位符**

定位符用于规定要匹配的字符串出现的位置，比如在字符串的开始还是在结束的位置，这个也是相当有用的，必须掌握。

**^符号说明：匹配目标字符串的开始位置**

比如/^han/gi去匹配"hanshunping han han"，看看会得到什么结果？han

**在开始处出现匹配的字符串才会得到值**

**$符号说明：匹配目标字符串的结束位置。**

比如/han$/gi去匹配"hanshunping han han"，看看会得到什么结果？han

**在结尾处出现匹配的字符串才会得到值**

**元字符--定位符**

\b符号说明：匹配目标字符串的边界。

这个字符串的边界，有点不好理解，举例说明：

比如/han\b/gi去匹配"hanshunping sphan nnhan"，我们会匹配到"hanshunping sphan nnhan"，所以这里说的字符串的边界指的是子串间有空格，或者是目标字符串的结束位置，**特别注意：**没有目标字符串的开始位置

\B符号说明：匹配目标字符串的非边界。

这个和\b含义刚刚相反，不再赘述。看案例：

比如/han\B/gi去匹配"hanshunping sphan nnhan"，会匹配到"hanshunping sphan nnhan"

**元字符--转义符号\**

\符号说明：在我们使用正则表达式去检索某些特殊字符的时候，需要用到转义符号，否则检索不到结果，甚至会报错。

案例：

用/$/gi 去匹配"abc$("会怎样？ 返回空

用/(/gi 去匹配"abc$("会怎样？ 报错

**元字符--转义符号\**

**需要用到转义符号的字符有以下：**

**. \* + ( ) $ / \ ? [ ] ^ { }**

**元字符--选择匹配符**

有时候，在匹配某个字符串的时候是选择性的，即：既可以匹配这个，又可以匹配那个，这时需要用到选择匹配符号|

选择匹配符号还是比较好理解的，案例说明：

用/(han|韩)/gi去匹配"hanshunping 韩顺平"，会得到什么结果？ han 韩

案例：给你一个字符串，请你验证该字符串是否是一个电子邮件？

[09.html]

<html>

<head>

<title>RegExp--验证邮件格式</title>

<script type="text/javascript" language="javascript">

<!--

function test(){

var con=content.value;

alert(con);

var myReg=/^[a-zA-Z0-9\_-]+@([a-zA-Z0-9]+\.)+([a-zA-Z]){2,3}$/;

if(myReg.test(con)){

alert("格式正确!");

}else{

alert("格式错误!");

}

}

//-->

</script>

</head>

<body>

<input id="content" type="text" />

<input type="button" value="验证邮件格式" onclick="test()"/><br>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

案例：给你一个字符串，请你验证该字符串是否是一个身份证？

[10.html]二代身份证验证

<!DOCTYPE html>

<html>

<head>

<title>RegExp--二代身份证验证.html</title>

<meta name="keywords" content="keyword1,keyword2,keyword3">

<meta name="description" content="this is my page">

<meta name="content-type" content="text/html; charset=">

<!--<link rel="stylesheet" type="text/css" href="./styles.css">-->

<script type="text/javascript" language="javascript">

<!--

function idCard(){

area={11:"北京",12:"天津",13:"河北",14:"山西",15:"内蒙古",21:"辽宁",22:"吉林",23:"黑龙江",31:"上海",32:"江苏",33:"浙江",34:"安徽",35:"福建",36:"江西",37:"山东",41:"河南",42:"湖北",43:"湖南",44:"广东",45:"广西",46:"海南",50:"重庆",51:"四川",52:"贵州",53:"云南",54:"西藏",61:"陕西",62:"甘肃",63:"青海",64:"宁夏",65:"新疆",71:"台湾",81:"香港",82:"澳门",91:"国外"}

var idInfo=idNumber.value;

var idReg=/^[1-9]\d{5}[1-2](9|0)\d\d[0-1]\d[0-3]\d{4}(\d|x)$/;

//判断输入的ID长度和格式是否正确

if(!idReg.test(idInfo)||idInfo==""){

alert("不能为空或输入有误!请核实后重新输入!");

}else{

//通过地区码判断ID号是否非法

var myArea=idInfo.substring(0,2);//只判断了开始的2位地区码省份，有兴趣的朋友可以添加市及县的判断

if(area[parseInt(myArea)]==null){

alert("非法二代身份证!");

}

//取出性别

var mySex=idInfo.substring(16,17);

//取出生日

var myDate=idInfo.substring(6,14);

alert("身份证号码："+idInfo+" 籍贯所在地："+area[parseInt(myArea)]+" 性别："+(mySex%2?"男":"女")+" 生日："+myDate);

}

}

//-->

</script>

</head>

<body>

<input id="idNumber" type="text" />

<input type="button" value="二代身份证验证" onclick="idCard()"/>

</body>

</html>