# Javaspcript部分

## Javaspcript基础

### java和javascript区别:

雷锋和雷锋塔的关系（实际上就是没有关系！）

### js特效组成部分:

1. 设置事件
2. 事件触发
3. 属性改变或样式改变

### js标签:

<script>

//js代码!

</script>

### js标签放置位置:

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script>

// 1)定义变量

// 2)定义函数

</script>

</head>

<body>

<!-- html实体标签 -->

</body>

<script>

//特效代码

</script>

</html>

### js放置类型:

1. 1)内联
2. 2)内部
3. 3)外部(推荐)

## js注释

1. 单行注释：//
2. 多行注释:/\* \*/

## 变量

### js定义变量

### js定义变量

js中的变量或函数名命名规则遵守驼峰式写法:dobj.getFullYear();

x=10;

### js输出变量

1. alert(x);
2. document.write(x);

### 字符串连接符:

x+y;

### js的10种变量类型

#### 标准类型

1. 整型

x=10;

1. 浮点型

x=10.5;

1. 字符串

x='hello world!';

1. 布尔型

1)真(true)

2)假(false)

#### 复合类型

1. 数组
2. 数组定义

a.x=new Array(1,2,3);

b.x=[1,2,3];

1. 数组类型

* 一维数组：只有1层的数组。

arr=[1,2,3];

* 二维数组：嵌套2层的数组。

arr=[1,[2]];

* 三维数组：嵌套3层的数组。

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

1. 数组元素赋值

arr=['a','b'];

第一个元素:0=>'a'

第二个元素:1=>'b'

1. 数组元素取值

arr=['a',['b']]

引用方法：arr[1][0] ,表示字母b

arr=['a',['b'，'c'，['d'，'e']]]

引用方法：arr[1][2][1],表示字母e

1. 对象
2. 对象的定义

obj=new Object();

1. 对象组成部分:

属性(特征)，对象的属性是对象静态特征。

obj.username='user123';

obj.age=20;

方法(行为)，对象的函数是对象的动态特征，即对象的动作。

obj.say=function(){

alert('my name is '+this.username);

}

this代表当前对象，这样无论对象名怎么改变，“this”始终表示当前变量。

1. json对象

json一般用于在对象之间交换数据，采用“键:值”对的形式表示数据。现在已经成为事实的对象数据交换桥梁。与PHP和linux bash中的关联数组是一样。

在json中一般只存放静态信息，即只存放对象的属性和值，不定义行为和方法。

1. 定义

obj={

'username':'user1',

'age':'20',

'sex':'nan'

};

注意，最后一个属性后面没有逗号（,）。

1. 实例化方法

constructor 构造方法

instanceof 实例化(推荐)

#### 特殊类型

1. NaN

表示不是一个数字（非数字）。

str='10abc';

num=Number(str); #就是NaN类型

alert(num); #返回值为：“NaN”。

1. Null

表示空值

str=null;

1. undefined

表示未定义。

alert(str);

### js类型测试:

1. js中如何检测数据类型:

x='hello world'; (string)

x=10.5; (number)

x=10 (number)

x=true; (boolean)

x=[1,2,3]; #有问题(object)

x=new Object(); (object)

x={'username':'user1'}; (object)

x=null; (object)

x=undefined; (undefined)

x=NaN; (number)

type = typeof(x)

alert(x)

1. js中如何检测数组类型:

* 方法1：

arr = new Array();

//检测arr是否是Array类的实例

if (arr instanceof Array){

alert(‘array’)

}else{

alert(‘not array’)

}

* 方法2：

arr = new Array();

//检测arr的父类是否是Array

if (x.constructor==Array){

alert(‘array’)

}else{

alert(‘not array’)

}

### 变量类型转换:

#### 字符串转整型

1)parseInt(); #推荐

2)Number();

3)x-0; //算术运算符-会自动将两边的参数强制转换为数值,但效果与Number()类似。

#### 整型转字符串

1)String(); #推荐

2)x+''; //算术运算符+会优先进行字符连接运算，然后才是数学计算，只要参数中有一个为字符串，其它所有参数都自动转换为字符串,效果与String()类似。

#### 所有类型转布尔类型

1)Boolean(x); #推荐

2)!!x; //另外一种写法。

注意：

所有类型转布尔类型为假的如下，其余的都为真。

1)整型(0);

2)浮点型(0.0);

3)字符串('');

4)null;

5)NaN;

6)undefined;

请注意：

1）数组和对象转换为布尔型时，永远为真，没有假的情况出现，哪怕是空数组和空对象；

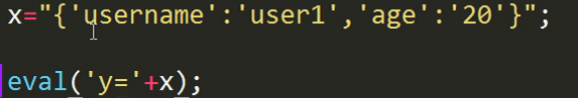
2）字符串“0”转换为布尔型后为真，只有空（null）字符串转换成布尔型后为假。这里说的字符串的情况，而不是数字0的情况，一定注意！

#### json字符串转json对象

1)eval('y='+x);

2)y=eval('('+x+')');

方法一实例截图：

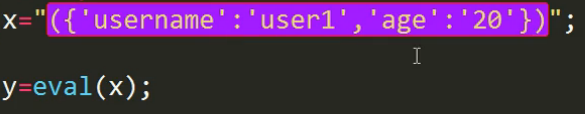


上图中：

x是一个字符串，并不是一个json对象（因为有双引号）。

通过使用eval（）添加了一个字符串“y=”,然后与原有的x进行字符串连接运行,字符变成了“y={‘usetname’:‘user1’,’age’:’20’}”。最后eval（）将这个字符串转换为正常的json对象。

方法2原理：

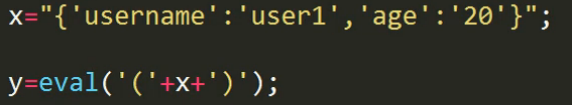


上图中：

x是一个字符串，但加上圆括号以后，就转换成了一个强迫可以执行的单元语句块。

通过使用eval（）将原有的x单元块进行转换和执行，最后赋值给变量y。

方法2实例（推荐）



方法很简单：在原来字串基础上，加上一对原括号就行了！

### Global对象的顶级全局方法

JS是主张基于对象的编程，是基于已有对象（比如浏览器提供的对象）进行编程，这就要求我们对其它对象有深刻的理解，比如浏览器对象。虽然JS主张基于对象的编程，但也同时提供了自己独立存在的对象和方法。

顶级全局方法是全局对象（Global）的方法，但在书写时，Global是不允许写出来的，因此，给人造成了这些方法是不依托对象存在的错觉。

在本节中，我们先学习几个最常用的顶级全局方法，这些方法是以后学习的过程中要经常用到，至于jS自己的对象，将在后续章节中介绍。本节将学习以下全局方法：

#### typeof()

显示数据类型

#### parseInt() #推荐使用

强制变换为整数（Int）类型

#### parseFloat()

强制转换为浮点（Floating）类型

#### eval()

y=’x=10+20’ //这时y的值为“x=10+20”，是一个字符串。

alert(y); //输出结果为“x=10+20”，是一个字符串

alert(X); //变量x不存在，会出错

eval(y); //将字符串转换为表达式

alert(X); //x被定义，并且x的值为“30”，是一个数字

alert(y); //y的值为“30”，是一个数字。

将一个表达式字符串，转换为可执行的表达式。上例中的变量x是转换后产生的。

#### Number()

转换为数字类型，Number()对有些特别的字符串没有办法处理。比如“10abc”,这个字符串，Number()没有办法智能处理，最后会出错。而parseInt()可智能转换为整数10.

#### String()

转换为字符串（String）类型

#### Boolean()

强制转换为布尔型（Boolean）类型

## 运算符

### 一元运算符

i++; 先赋值后运算

++i; 先运算后赋值

i--; 先赋值后运算

--i; 先运算后赋值

### 二元运算符

1)数学运算符

2)比较运算符

3)逻辑运算符

4)赋值运算符

1. 数学运算符

+，-，\*，/，%

1. 赋值运算符

=，+=，-=，\*=，/=，%=

1. 比较运算符

>，<，<=，>=，!= ,==

1. 逻辑运算符

||，&&，!

* 或：多个条件中只要有一个为真，则整体为真，否则为假。
* 与：多个条件中必须所有都为真，整体才为真，否则为假。
* 非：原值取反，真变成假，假变成真。

1. 逻辑运算符开关功能

||

* 逻辑或“两边任一条件为真，则为真
* 逻辑假开关：左边为假才执行右边

&&

* 逻辑与：两边都为真，则为真
* 逻辑真开关：左边为真才执行右边

### 三元运算符

score=(i>=60)?'yes':'no';

alert(score);

注意：三元运算符会改变原始变量score的值，但if语句不会。

## js语法:

### if语句

#### 单分支

if(){

}else{

}

#### 多分支

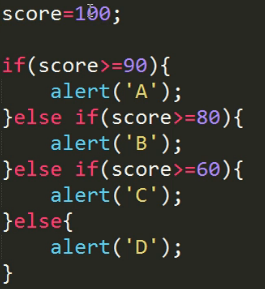
if(){

}else if(){

}else if(){

}else{

}



在if语句中，从上向下执行，只要有一个条件成立，剩下的条件不会再判断，直接跳到分支语句之后的第一条语句开始执行，即分支语句中，依据“从上往下、只选其一”的原则执行！也就是说分支结构中，永远只执行众多分支中的一个！！！

### switch分支判断

switch(){

case 1:

//代码

break;

case 2:

//代码

break;

default:

//代码

break;

}

1. jS中如果在分支中不加break语句，就会依次执行第一个匹配分支（含）后面的所有分支，进行多次匹配，加上break语句以后，只会众多分支中选择一个匹配的执行，匹配分支执行完毕后，向后跳到整个switch语句后面的第一条语句继续执行。
2. Switch语句主要用于“==”某几个确定的选项时使用，不能判断“>、<、>=、<=、!=”等情况。当判断条件只“==”时，强烈建议使用switch分支结构。

### while循环

i=0;

while(i<3){

document.write('<h1>+i+</h1>');

i++;

}

#可以遍历数组元素

### for循环

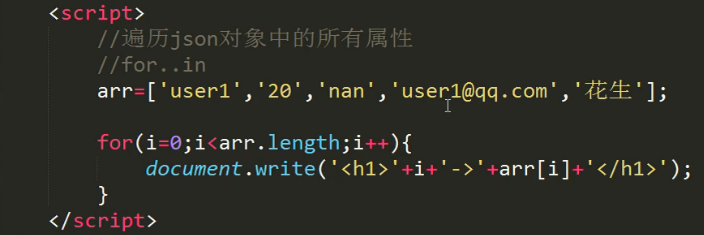
#### for..语法

for(i=0;i<3;i++){

document.write('<h1>+i+</h1>');

}

#可以遍历数组元素



个人体会：感觉arr.length并不是获取数组所占的空间大小，而是获得数组或对象的索个数。如果数组里有5个元素，那么arr.length=5。

#### for..in语法

for(i in obj){

alert(i+'->'+obj[i]);

}

#1)遍历json对象

#2)遍历数组元素





1. for..in语句用于不确定成员数量的情况下。强烈推荐使用这种方法遍历数组和对象。
2. 当数组和对象里的索引值是一个变量时，不能用obj.i的形式引用，而应该用obj[i]的形式。
3. 可以把数组也理解成json对象，只不过默认是以数字为索引的，这些默认的索引由系统自动生成且保存在某个地方，不显示出来而已！

### with语法

with(document){

write('111');

write('222');

write('333');

}

document.write是完整的语法，当调用同一个对象的方法时，可使用with语句，简化代码，如上图，前面都是以document开头的语句，可以将document写在括号中，这样以后的语句就不用指定对象名了，相当于临时指定了一个默认对象。

## 函数

首先要弄明白函数和方法的区别：方法目标是定指定某个对象内部行为特征，它本质上就是一个函数，但这个函数只属于这个对象，要调用时，必须在前面指定对象名。

而函数是独立于对象存在的，函数的目标是执行一系列的语句，以减少代码冗余。

### 自定义函数

function list(参数列表){

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

document.write('<h1>'+i+'->'+array[i]+'</h1>');

}

document.write('<hr>');

}

1. 函数中的参数列表其实就是一个变量（或者说是占位符），当用户调用函数并在括号中指明了参数时，相当于用户输入的括号里的实际内容将替换函数的参数列表。
2. 参数可以有多个，中间用逗号隔开。

### 匿名函数

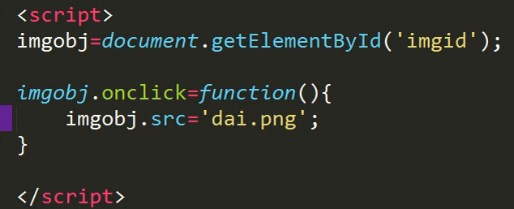
匿名函数就是没有名字的函数。

imgobj.onclick=function(){

imgobj.src='dai.png';

}

#函数的作用: 降低代码的冗余度，提高代码的重用性!



匿名函数用于只需要执行一系列的行为，而不需要指定函数名的情况。被广泛应用于改变web控件的行为和属性。

## 变量作用域

变量作用域:

1.全局变量

2.局部变量

#只有在函数内部带var的变量为局部变量，其余全为全局变量.



如上图所示：程序最终报错，提示x变量未定义。

只有在函数内部使用了关键字var定义的变量才是局部变量，即使在函数内部定义，前面没有关键字var定义的变量也是全局变量！因此，在定义变量时就遵循下面的规则：

1. 在函数外面，不管是否使用的var关键字定义变量，都是全局变量，因此在函数外面定义变量时，统一不要使用var关键字；
2. 在函数里面，如果要定义全局变量不要使用var关键字；
3. 在函数里面，如果要定义局部变量才使用var关键字；

依据上面的定义规则，只要看一点，前面有关键字var定义的变量就是局部变量，没有var关键字的变量就是全局变量。

## 消息框

### alert()：警告框

<script>

//警告框

x=10;

alert(x);

</script>

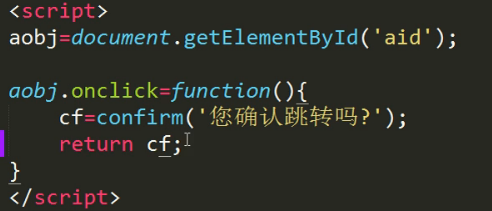
### confirm()：确认框

示例：控制控件的默认行为。

第一种写濡



第二种写法：

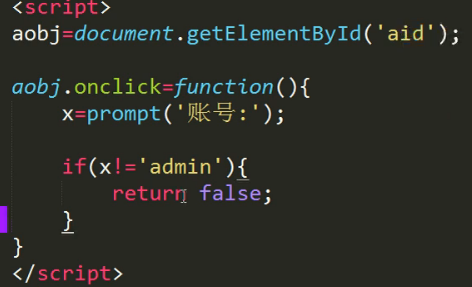


注意：确认框returm的用法。可以用于控制页面是否跳转。这里return的含义为：

1. 如果返回的结果为真，则执行默认的跳转动作；
2. 如果返回的结果为假，则不执行前面的跳转动作。

### prompt()：提示框

示例：用于临时认证。



## js内置对象

虽然js是主张基于对象的编程，但js自身也拥有自己独立的内置对象，主要有6类。

### 数学对象

1. 对象

js内部本身就存在一个名为Math的数学对象，因此不用实例化，可以直接使用。

1. 属性:

Math.PI：圆周率

1. 方法:

Math.ceil()：不管小数是多少，都进位为整数，即10.1也为11。

Math.floor()：不管小数是多少，都扔掉小数数，即10.99也为10。

Math.round()：四舍五入

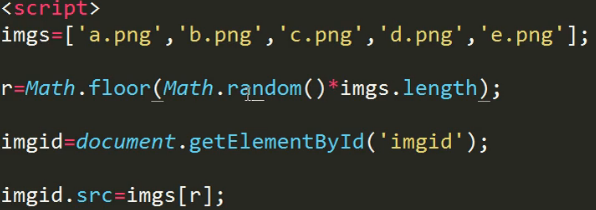
Math.max()：获取多个数中的最大数。

Math.min()：获取多个数中的最小数。

Math.random()：产生0—1之间的随机小数。

1. 示例

#随机数实例—每次刷新都从5幅图片中随机更换一幅图片。



扩展联想：如果把图片换成用户名是否可以做成投奖程序？

### 日期对象

1. 创建:

时间对象需要手式生成，命令如下：

dobj=new Date();

1. 属性

无

1. 方法:

dobj.getFullYear()：获取时间对象中的年

month=dobj.getMonth()+1：获取时间对象中的月，注意月是以0还始，是js中的一个小缺陷吧。

day=dobj.getDate()：获取时间对象中的日

hour=dobj.getHours()：获取时间对象中的时

minute=dobj.getMinutes()：获取时间对象中的分

second=dobj.getSeconds()：获取时间对象中的秒

week=dobj.getDay()：获取时间对象中的周

1. 秒表实例

|  |  |
| --- | --- |
|  |  |

#### 定时器

定时器有主要作用是指定时间周期内，反复执行某个功能。比如上例中的每隔1秒获取1次系统时间。

1. 创建

sobj=setInterval(func,time);

1. 属性

无

1. 方法

clearInterval(sobj)：清除定时器

1. 示例

|  |  |
| --- | --- |
| 图像旋转并防止定时器叠加事例 | 倒计时跳转事例 |
|  |  |

### 超时器

超时器有主要作用是指定时间周期内，执行一次某个功能。

1. 创建

tobj=setTimeout(func,time);

1. 属性

无

1. 方法

clearTimeout(tobj)：清除

1. 示例



### 字符串

1. 属性:

str.length;字符串的长度，实际上就是所有字符的个数（包含空格）。

1. 方法:

indexOf()：指定字符第一次出现的位置。参数：（要查找的字符）

lastIndexOf()：指定字符最后一次出现的位置。参数：（要查找的字符）

substr(start,length)：从一个字符串中截取部分字符。参数：（开始位置，要截取的字符个数）

slice(start,end)：从一个字符串中截取部分字符。参数：（开始位置，结束位置但不包含结束位置）

split(reg)：从一个字符串中分割成多个字符串，保存到数组中。参数：（分隔符）

toUpperCase()：所有字符转换为大写。

toLowerCase()：所有字符转换为小写。

search(reg)：查找某一个字符或单词在字符串的位置。参数：（要查找的字符或单词）。

match(reg)：查找某一个字符或单词，只要找到第一个后就停止查找。参数：（要查找的字符或单词）。如果要匹配所有找到的结果，需要使用正则表达式。如：str.match(/linux/g)。正则会在下节介绍。

replace(reg,"str")：替换某一个字符或单词，只要找到第一个后就停止替换。参数：（要替换的字符，替换成什么）。如果要替换所有的，需要使用正则表达式。如：str.match(/linux/g)。

### 正则对象

1. 正则表达式:



只有这4个方法支持正则表达式

1. 元字符

. 查找单个字符。匹配任意一个字符，包含空格，不包含换行和结束符。可以把任意多个意识的每一个字符当成数组元素进行拆分，即把单词或子拆分成单个字符。

\w 查找单词字符，匹配字母、数字和下划线中的一个字母。

\W 查找非单词字符，匹配非字母、数字和下划线中的任意一个字符。

\d 查找数字。

\D 查找非数字字符。

\s 查找空白字符。

\S 查找非空白字符。

\b 匹配单词边界，以空格或tab为分界。可以放在左右两边来检查是否存在边界

\B 匹配非单词边界。可以放在左边或右边来检查是否没有边界。

1. 中括号

[abc] 查找方括号之间的任何字符。

[^abc] 查找任何不在方括号之间的字符。

[0-9] 查找任何从 0 至 9 的数字。

[a-z] 查找任何从小写 a 到小写 z 的字符。

[A-Z] 查找任何从大写 A 到大写 Z 的字符。

(red|blue|green) 查找任何指定的选项。圆括号表示一个单元，|表示或运算。

1. 修饰符

i 执行对大小写不敏感的匹配。

g 执行全局匹配。

m 执行多行匹配。

1. 量词

n+ 匹配任何包含至少一个 n 的字符串。

n\* 匹配任何包含零个或多个 n 的字符串。

n? 匹配任何包含零个或一个 n 的字符串。

n{X} 匹配包含 X 个 n 的序列的字符串。

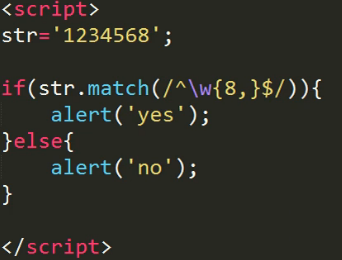
n{X,Y} 匹配包含 X 或 Y 个 n 的序列的字符串。

n{X,} 匹配包含至少 X 个 n 的序列的字符串。

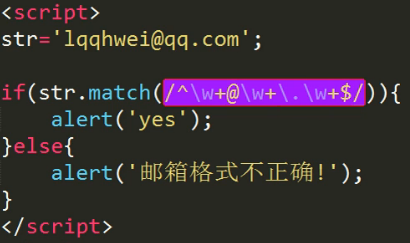
n$ 匹配任何结尾为 n 的字符串。

^n 匹配任何开头为 n 的字符串。

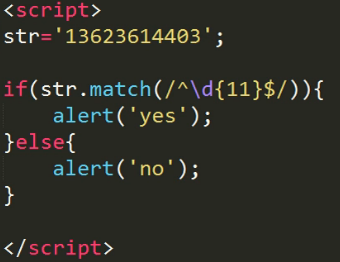
1. 正则表达式的实际例子
2. 检查密码长度



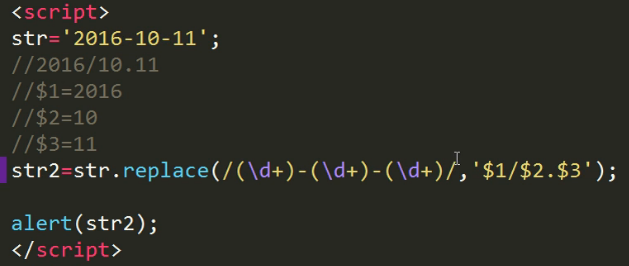
1. 检查邮箱格式



1. 检查手机格式



1. 高级替换



当使用圆括号时，$1、$2、$3已经系统由自动赋值了，这些变量称为内部变量，内部变量可以直接使用，1对圆括号对应1个内部变量，编号从1开始！

### 数组对象

1. 属性:

Length：数组的长度，实际上是数据的元素的个数。

1. 方法:

join("/")：将数组用指定的分隔符连接成字符串‘

pop()：把数组里面最后一个元素弹出，也就是删除最后一个元素。返回新数组的长度。

push()：向数组最后压入一个元素，也就是在最后增加一个元素。返回新数组的长度。

shift()：把数组里面第一个元素弹出并且返回，也就是删除第一个元素并返回。

unshift()：向数组最前面压入一个元素，也就是在最前面增加一个元素。

reverse()：反序排列数组所有元素。

concat()：连接两个数组，生成一个大数组。

slice(start,end)：取出数组中指定位置的元素。参数（开始位置、结束位置但不包含）。

splice()：取出数组中指定位置的元素。参数（开始位置，多少个，新添加元素列表），同时改变原数组。同时可以向原数组中添加新元素，即截取的同时可以添加元素。

sort(show)：排序。参数（调用的排序函数名），改变原数组。只能按

show(a,b){

return a-b; //正序

//return b-a; //倒序

}

注意：上面红色字体的方法具备改变原数组的能力！

1. 检测数组类型（复习）

if(arr instanceof Array){

alert('yes');

}else{

alert('no');

}

### 事件对象:

#### 常用事件

1. 鼠标事件

onclick 鼠标点击某个对象

ondblclick 鼠标双击某个对象

onmousedown 某个鼠标按键被按下

onmouseup 某个鼠标按键被松开

onmousemove 鼠标被移动

onmouseenter 鼠标被移到某元素之上

onmouseleave 鼠标从某元素移开

1. 键盘事件

onkeydown 某个键盘的键被按下

onkeyup 某个键盘的键被松开

1. 表单事件

onblur 元素失去焦点

onfocus 元素获得焦点

onselect 文本被选定

onchange 用户改变域的内容

onsubmit 提交按钮被点击

onreset 重置按钮被点击

1. 窗口事件

onload 某个页面或图像被完成加载

onresize 窗口或框架被调整尺寸

1. 其他事件

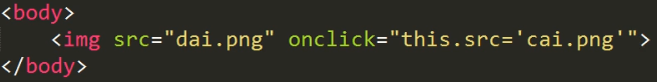
onerror 当加载文档或图像时发生某个错误

oncontextmenu 右键打开菜单

#### 事件与控件的绑定方法

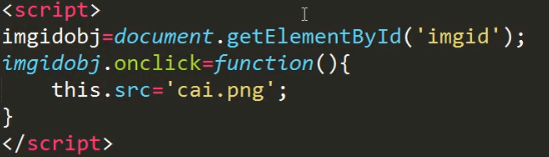
1. 内联绑定

在HTML标签内绑定，这种方法也称内联绑定。



1. 外部绑定。（推荐）

如在js脚本采用onclick方法绑定单击事件。



#### 事例

1. 鼠标事件
2. 获取鼠标左中右键并改变图像背景颜色



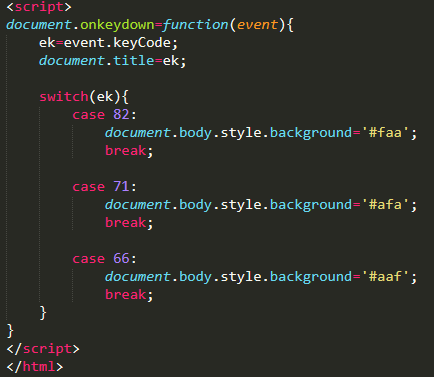
event对象作为参数传递给了匿名函数，event.button属性可以获取鼠标的左中右键，分别为0、1、2，通过switch分支语句，可以分别为鼠标的三个键设置不同的功能！

1. 图像跟随鼠标移动

|  |  |
| --- | --- |
| 定位图像的初始位置： | 图像跟随鼠标移动： |
|  |  |

event对象作为参数传递给了匿名函数，event.clientX属性可以获取鼠标当前的X坐标，event.clientY属性可以获取鼠标当前的Y坐标，后面一定要加上“px”单位。

1. 键盘事件
2. 获取键盘码并改变背景颜色



event对象作为参数传递给了匿名函数，event.keyCode属性可以获取当前被按下的键盘代码，从而感知是哪一个键被按下。

1. 用键盘控制图像移动

|  |  |
| --- | --- |
| 获取屏幕的宽度和高度 | 实现键盘控制图片移动 |
|  |  |

document.documentEliment.clientWidth可以获取浏览器正文区的宽度，document.documentEliment.clientHeight可以获取浏览器正文区的高度。

1. 弹力球效果

document.documentEliment.clientWidth可以获取浏览器正文区的宽度，document.documentEliment.clientHeight可以获取浏览器正文区的高度。结合定时器自动刷新功能可以制作出弹力球效果。

|  |  |
| --- | --- |
|  |  |

#### 表单事件:

onblur 元素失去焦点

onfocus 元素获得焦点

onselect 文本被选定

onchange 用户改变域的内容

onsubmit 提交按钮被点击

onreset 重置按钮被点击

#### 窗口事件

onload 某个页面或图像被完成加载

页面在加载过程中，首先是加载全部标签，其次是完成图片等静态资源下载，所有静态资源下载完毕后，才能称为页面加载完毕，这时才会触发onload事件。

onresize 窗口或框架被调整尺寸

onscroll 窗口滚动

三个高度：

可视区域的高

document.documentElement.clientHeight

屏幕的总高度

document.documentElement.scrollHeight

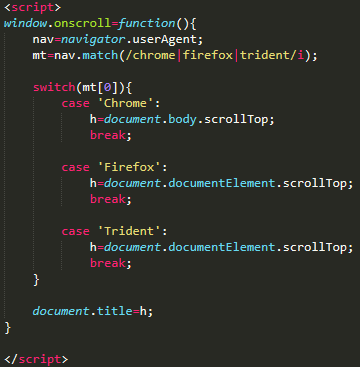
已滚动的高

document.documentElement.scrollTop #firefox，ie

document.body.scrollTop #chrome

事例：

1. 判断浏览器的类型



1. 回到顶部



#### 其他事件:

1.onerror 当加载文档或图像时发生某个错误。

2.oncontextmenu 右键打开菜单

事例：自定义鼠标右键

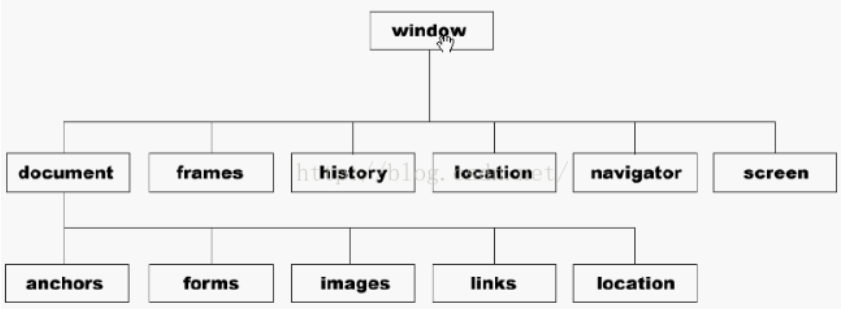
 

## BOM基础

### 什么是BOM？

浏览器对象模型（Browser Object Model），提供了独立于内容而与浏览器窗口进行交互的对象，由于BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是窗口（window），我们平时上网时打开的每一个标签都是一个独立的窗口（window）。

它由一系列相关的对象构成，并且每个对象都提供了很多方法与属性。但BOM缺乏标准，JavaScript语法标准化组织是ECMA，DOM的标准化组织是W3C，BOM最初是Netscape浏览器标准的一部分。



### BOM对象

1. window对象
2. navigator浏览器特性
3. screen屏幕对象
4. history对象
5. location地址栏
6. document对象（就是DOM对象）

#### window对象

1. 属性:

frames //窗口数组,逐渐淘汰中，只在服务器端后台少量使用。实际上就是框架。

top //顶级窗口

1. 方法:

alert()：前面已经学过了。

confirm()：前面已经学过了。

prompt()：前面已经学过了。

setInterval()：前面已经学过了。

clearInterval()：前面已经学过了。

setTimeout()：前面已经学过了。

clearTimeout()：前面已经学过了。

除了全局对象（Global）的方法引用可以不用在方法名前面写对象名以外，还有一个也不用写对象名，就是windows对象

#### navigator浏览器特性

1. 属性:

userAgent //更精确的版本，只学这个属性就可以了，其它的以后再学其它的。

hash //获取url里的锚点#abc

host //主机名

href //当前的url地址

pathname //index.html 页面的名称

protocol //协议 http://

search //参数 ?id=10&name=user1

1. 方法:

reload(); //加载当前页面，刷新当前页面

#### screen屏幕对象

screen对象提供获取访问者屏幕信息的方法。这些信息可用来确定显示哪个图像或页面有多大。无论是否使用screen对象，都需要创建一个好的基于CSS的设计以便能优雅地处理各种大小的屏幕。

1. 属性：

availHeight //屏幕的可用高度、，即减去了操作系统任务栏这样的控件所使用的空间

availWidth //屏幕的可用宽度，即减去了操作系统任务栏这样的控件所使用的空间

colorDepth //颜色深度（即屏幕可以显示多少种颜色）

height //屏幕的总的高度，即分辨率的高度

width //屏幕的总的宽度，即分辨率的宽度

1. 方法

无

只需要了解这么多。

#### history对象

提供了通过访问者浏览历史向前和向后移动的方法。处于安全考虑，JavaScript不能访问浏览器所访问网站的URI。具体说，可以使用back()、forward()和go()方法。back()和forward()分别向后和向前移动一个页面。go()方法移动到参数指定的索引值。

1. 属性

无

1. 方法

back()：返回上一个页面

forward()：前进下一个页面

go(1或-1)：可代替back()和forwad(),-1是back()、1是forwad()。

#### location地址栏

location主要作用是改变地址栏的URL，实现页面跳转。包括任何有关查询字符串、使用的协议以及其他相关组件的信息。

1. 属性

hash //获取url里的锚点#abc

host //主机名

href //当前的url地址

pathname //index.html 页面的名称

protocol //协议 http://

search //参数 ?id=10&name=user1

#注: 一般情况下我们改变地址栏中的地址时，直接使用location='http://www.baidu.com'，而不用location.href='http://www.baidu.com';

1. 方法

reload()：刷新当前页面

1. 事例

重定向到我的网站：

location.assign("http://www.braingia.org");

调用assign()方法与设置href属性基本一样：

location.href = "http://www.braingia.org";

还可以改变location对象的其他属性，比如端口、查询字符串或路径。为了将路径设置为blog，可以则么做：

location.pathname = "blog";

为了将查询字符串设置为?name=Steve，可以这样：

location.search = "?name=Steve";

可以调用reload()方法重新载入页面：

location.reload();

调用location.reload()，浏览器会从缓存那里载入页面，而不是从服务器再次请求页面。然而，如果给这个方法传递一个布尔值true，那么浏览器将直接从服务器重载页面：

location.reload(true);

注意：使用reload()方法，尽量在脚本内重载页面，不要通过事件触发的函数调用，后者很可能会促使一个循环条件。

## DOM(document object model)基础和Ajax简介

### 获取的DOM基础

1. 直接可以获取的dom基础对象

document

document.documentElement：实际上就是<HTML>标签对

document.head

document.title

document.body

1. 获取元素对象的方法

document.getElementById() #返回一个元素对象(dom对象)，id属性只能处理一个元素，不便于批量处理。

document.getElementsByTagName() #返回一个集合对象，通过标签名得到一组对象集合。如所有的<h1>标签。

document.getElementsByClassName() #返回一个集合对象，通过类名得到一组对象集合。

document.getElementsByName() #返回一个节点列表(nodelist)，通过标签名的name属性的值得到一组对象集合，并不是所有标签都有name属性，不常用，几乎不用。

1. 元素对象标准属性

下面4个属性是任何一个标签都有的。

obj.id：

obj.className;

obj.style;

obj.title;

1. 元素对象非标准属性:(undefined)

obj.username;

obj.age;

obj.sex;

1. 统一的方法来获取标准或非标准属性

获取元素属性的标准方法。

获取属性

fidobj.getAttribute('age');

设置属性

fidobj.setAttribute('age','20');

1. 元素对象js属性

js在获取一个标签是，除了标签自身的标准属性和非标准属性，js也会赋予元素一些js定义的专用属性。

obj.tagName:获取标签标签名（只包含标签名且不带<>）。

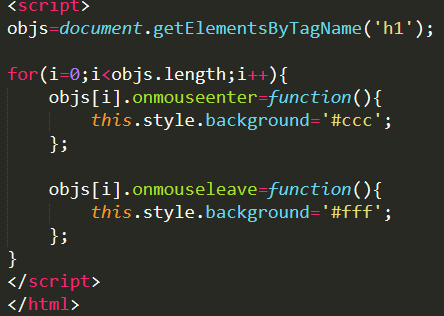
obj.innerHTML:获取标签标签内部的值，即标签内部的所有文本部分（不包含标签）,但标签内部嵌套的标签会展示出来。

obj.textContent: :获取标签标签内部的所有文本部分，标签内部嵌套的标签会忽略。

obj.outerHTML：获取标签标签外部的值，即展示的整条HTML代码（包含标签）。

### DOM元素对象的6个实例:

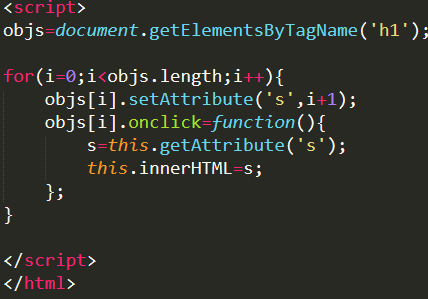
1. 移入移出特效

1. 循环点击换色

1. 点击换行号

1. 点击标题切换内容

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>index</title>

<style>

\*{

font-family: 微软雅黑;

}

h1{

cursor:pointer;

background: #aaf;

}

</style>

</head>

<body>

<h1>linux技术</h1>

<p>linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!linux is very much!</p>

<h1>php技术</h1>

<p>php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!php is very much!</p>

<h1>js技术</h1>

<p>js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!js is very much!</p>

</body>

<script>

h1s=document.getElementsByTagName('h1');

ps=document.getElementsByTagName('p');

for(i=0;i<h1s.length;i++){

h1s[i].setAttribute('i',0);

h1s[i].setAttribute('s',i);

ps[i].setAttribute('id','p'+i);

h1s[i].onclick=function(){

s=this.getAttribute('s');

pobj=document.getElementById('p'+s);

i=parseInt(this.getAttribute('i'));

if(i%2==0){

pobj.style.display='none';

}else{

pobj.style.display='block';

}

this.setAttribute('i',i+1);

};

}

</script>

</html>

1. 多选、反选和全不选

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>index</title>

<style>

\*{

font-family: 微软雅黑;

}

</style>

</head>

<body>

<h2>请选择以下选项:</h2>

<form action="">

<p>

<label><input type="checkbox" class='chk'> linux</label>

</p>

<p>

<label><input type="checkbox" class='chk'> php</label>

</p>

<p>

<label><input type="checkbox" class='chk'> javascript</label>

</p>

<p>

<input type="button" value="全选" id='btn1'>

<input type="button" value="全不选" id='btn2'>

<input type="button" value="反选" id='btn3'>

</p>

</form>

</body>

<script>

//全选、全不选和反选

btn1obj=document.getElementById('btn1');

btn2obj=document.getElementById('btn2');

btn3obj=document.getElementById('btn3');

chks=document.getElementsByClassName('chk');

btn1obj.onclick=function(){

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

chks[i].checked=true;

}

}

btn2obj.onclick=function(){

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

chks[i].checked=false;

}

}

btn3obj.onclick=function(){

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

chks[i].checked=!chks[i].checked;

}

}

</script>

</html>

1. 选水果

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>index</title>

<style>

\*{

font-family: 微软雅黑;

}

select{

height:150px;

width:100px;

}

</style>

</head>

<body>

<h2>请选择你喜欢的水果:</h2>

<form action="">

<select name="" id="s1" size='2'>

<option value="">苹果</option>

<option value="">南瓜</option>

<option value="">冬瓜</option>

<option value="">面包</option>

<option value="">北瓜</option>

</select>

<input type="button" value=">>" id='btn'>

<select name="" id="s2" size='2'></select>

</form>

</body>

<script>

s1obj=document.getElementById('s1');

s2obj=document.getElementById('s2');

btnobj=document.getElementById('btn');

btnobj.onclick=function(){

idx=s1obj.selectedIndex;

opts=s1obj.options;

opt=opts[idx];

s2obj.add(opt);

}

</script>

</html>

### Ajax无刷新技术(了解即可，以后深入学习)

1. 生成xhr对象

xhr=new XMLHttpRequest()

1. 准备连接

xhr.open('get','del.php?id='+id,true)

1. 发送请求

xhr.send()

1. 获取php响应数据

xhr.onreadystatechange=function(){

if(xhr.readyState==4 && xhr.status==200){

s=xhr.responseText;

1. 根据响应数据走前端特效

if(s=='1'){

trobj=document.getElementById('tr'+id);

trobj.style.display='none';

}

# Jquery部分

### Jquery简介

1. jquery产品介绍

jQuery 是一个 JavaScript 库。包含进HTML文档中直接使用。

jQuery 极大地简化了 JavaScript 编程。

jQuery 很容易学习。

1. jquery软件下载

下载地址：[www.jquery.com](http://www.jquery.com)

下载时分为压缩版的非压缩版，区别在于：

1. 压缩版去掉了空格、换行等无用字符，无高亮显示，根本无法阅读，整个文档只有一行，但容量很小，下载速度更快，导致打开网页更快一些。；
2. 非压缩版格式比较友好，有高亮显示，便于学习，但容量较大，用户使用时下载速度更慢，导致打开网页更慢一些。
3. 在学习时使用非压缩版，正式发布时使用压缩版
4. jquery安装使用
5. 建议使用方法

jQuery 库是一个 JavaScript 文件，您可以使用 HTML的 <script> 标签引用它：

<head>

。。。

<script src="jquery-x.xx.xx.min.js"></script>

</head>

lamp将下载的文件放在网页的同一目录下，直接使用外部文件包含进行HTML就可以使用jQuery了，一般放在<head>标签的最后面.

注意：在以前HTML中，使用jquery，需要在<script> 标签中使用 type="text/javascript"属性，如：<script type="text/javascript">。

在 HTML5 中，不必那样做了。JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！

1. 替代方案（不推荐，只是为了看懂别人的使用方法）

如果你不希望下载并存放jQuery，那么也可以通过网络来引用jquery。如果你的站点用户是国内的，建议使用百度、又拍云、新浪等国内CDN地址，如果你站点用户是国外的可以使用谷歌和微软。请使用以下代码之一：

* Staticfile CDN:

<head> <script src="https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js"> </script> </head>

* 百度 CDN:

<head> <script src="https://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js"> </script> </head>

* 又拍云 CDN:

<head> <script src="https://upcdn.b0.upaiyun.com/libs/jquery/jquery-2.0.2.min.js"> </script> </head>

* 新浪 CDN:

<head> <script src="https://lib.sinaapp.com/js/jquery/2.0.2/jquery-2.0.2.min.js"> </script> </head>

不大推荐使用国外的CDN来获取版本（如google CDN），因为Google产品在中国很不稳定。



1. 预备知识（硬背）

三个学习的好途径：

<http://www.w3school.com.cn/jquery/index.asp>

1. DOM 对象转成 jQuery 对象

对于已经是一个 DOM 对象，只需要用 $() 把DOM对象包装起来，就可以获得一个 jQuery 对象了

$(DOM 对象)

如：

var v = document.getElementById("v"); //DOM对象

var $v = $(v); //jQuery 对象

注： var是定义变量

转换后，就可以任意使用 jQuery 的方法。

1. jQuery 对象转成 DOM 对象

jQuery对象转换成 DOM 对象有两种方法：[index] 和 .get(index)；

* jQuery 对象是一个数据对象:

可以通过 index] 的方法，来得到相应的 DOM 对象。如：

var $v = $("#v"); //jQuery 对象

var v = $v[0]; //DOM 对象

alert(v.checked); //检测这个checkbox是否被选中

* jQuery 本身提供的对象

通过get(index)方法得到相应的DOM对象。如：

var $v = $("#v"); //jQuery 对象

var v = $v.get(0); //DOM对象 ( $v.get()[0] 也可以 )

alert(v.checked); //检测这个 checkbox 是否被选中

**lamp**需要再强调的是： DOM对象才能使用DOM中的方法，jQuery对象是不可以使用DOM中的方法。同样，jQuery对象才能使用jQuery中的方法，DOM对象是不可以使用jQuery中的方法。

1. jquery中的$(this)和js中的this的关系:

其中是同一个东西，无非$()把this转成了jquery对象。

注意：不能用$(this)只能调用jquery的属性和方法，否则不符合jquery的语法；this只能使用js原生属性和方法，否则不符合js的语法.



### 核心函数

**jQuery 是为事件处理特别设计的。**jQuery 事件处理方法是 jQuery 中的核心函数。事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。术语由事件“触发”（或“激发”）经常会被使用。通常会把 jQuery 代码放到 <head>部分的事件处理方法中。

#### each();

each()方法能使DOM循环结构简洁，不容易出错，它封装了十分强大的遍历功能，使用也很方便，它可以遍历一维数组、多维数组、DOM,JSON等等。

事例：

1. each处理一维数组

var arr1 = [ "aaa", "bbb", "ccc" ];

$.each(arr1, function(i,val){

alert(i);

alert(val);

});

alert(i)将输出0，1，2

alert(val)将输出aaa，bbb，ccc

1. each处理二维数组

var arr2 = [['a', 'aa', 'aaa'], ['b', 'bb', 'bbb'], ['c', 'cc', 'ccc']]

$.each(arr, function(i, item){

alert(i);

alert(item);

});

arr2为一个二维数组，item相当于取这二维数组中的每一个数组。

item[0]相对于取每一个一维数组里的第一个值

alert(i)将输出为0，1，2，因为这二维数组含有3个数组元素

alert(item)将输出为  ['a', 'aa', 'aaa']，['b', 'bb', 'bbb']，['c', 'cc', 'ccc']

对此二位数组的处理稍作变更之后

var arr = [['a', 'aa', 'aaa'], ['b', 'bb', 'bbb'], ['c', 'cc', 'ccc']]

$.each(arr, function(i, item){

$.each(item,function(j,val){

alert(j);

alert(val);

});

});

alert(j)将输出为0，1，2，0，1，2，0，1，2

alert(val)将输出为a，aa，aaa，b，bb，bbb，c，cc，ccc

1. each处理json数据

这个each就有更厉害了，能循环每一个属性

var obj = { one:1, two:2, three:3};

each(obj, function(key, val) {

alert(key);

alert(val);

});

这里alert(key)将输出one two three

alert(val)将输出one，1，two，2，three,3

这边为何key不是数字而是属性呢，因为json格式内是一组无序的属性-值，既然无序，又何来数字呢。

而这个val等同于obj[key]

1. ecah处理dom元素

此处以一个input表单元素作为例子。如果你dom中有一段这样的代码

<input name="aaa" type="hidden" value="111" />

<input name="bbb" type="hidden" value="222" />

<input name="ccc" type="hidden" value="333" />

<input name="ddd" type="hidden"  value="444"/>

然后你使用each如下

$.each($("input:hidden"), function(i,val){

alert(val);

alert(i);

alert(val.name);

alert(val.value);

});

那么，alert(val)将输出[object HTMLInputElement]，因为它是一个表单元素。

alert(i)将输出为0，1，2，3

alert(val.name);将输出aaa,bbb,ccc,ddd，如果使用this.name将输出同样的结果

alert(val.value);  将输出111,222,333,444，如果使用this.value将输出同样的结果

如果将以上面一段代码改变成如下的形式

$("input:hidden").each(function(i,val){

alert(i);

alert(val.name);

alert(val.value);

});

可以看到，输出的结果是一样的，至于两种写法究竟区别在哪，我也还不知。此改变运用到上面几段数组的操作也会输出同样的结果。

这样，几个例子的实际结果已经得到答案。接着再继续往下研究，总不能知其然不知其所以然。

从以上的例子中可知jQuery和jQuery对象都实现了该方法，对于jQuery对象，只是把each方法简单的进行了委托：把jQuery对象作为第一个参数传递给jQuery的each方法。

看下jQuery中的each实现（网络摘抄）

function (object, callback, args) {

//该方法有三个参数:进行操作的对象obj，进行操作的函数fn，函数的参数args

var name, i = 0,length = object.length;

if (args) {

if (length == undefined) {

for (name in object) {

if (callback.apply(object[name], args) === false) {

break;

}

}

} else {

for (; i < length;) {

if (callback.apply(object[i++], args) === false) {

break;

}

}

}

} else {

if (length == undefined) {

for (name in object) {

if (callback.call(object[name], name, object[name]) === false) {

break;

}

}

} else {

for (var value = object[0]; i < length && callback.call(value, i, value) !== false; value = object[++i]) {}

/\*object[0]取得jQuery对象中的第一个DOM元素，通过for循环，

得到遍历整个jQuery对象中对应的每个DOM元素，通过 callback.call( value,i,value);

将callback的this对象指向value对象，并且传递两个参数,i表示索引值，value表示DOM元素；

其中callback是类似于 function(index, elem) { ... } 的方法。

所以就得到 $("...").each(function(index, elem){ ... });

\*/

}

}

return object;

}

#### size()

size() 方法在 jQuery 版本 1.8 中[被废弃](javascript:NewWindow('/try/deprecated.htm'))。 请使用 [length](http://www.runoob.com/jquery/prop-length.html) 属性代替。

size() 方法返回被 jQuery 选择器匹配的元素的数量。

#### length

length 属性包含 jQuery 对象中元素的数目

#### get();

$.get() 方法使用 HTTP GET 请求从服务器加载数据。

1. 语法

$.get(URL,data,function(data,status,xhr),dataType)

|  |  |
| --- | --- |
| 参数 | 描述 |
| *URL* | 必需。规定您需要请求的 URL。 |
| *data* | 可选。规定连同请求发送到服务器的数据。 |
| *function(data,status,xhr)* | 可选。规定当请求成功时运行的函数。  额外的参数：  *data* - 包含来自请求的结果数据  *status* - 包含请求的状态（"success"、"notmodified"、"error"、"timeout"、"parsererror"）  *xhr* - 包含 XMLHttpRequest 对象 |
| *dataType* | 可选。规定预期的服务器响应的数据类型。  默认地，jQuery 会智能判断。  可能的类型：  "xml" - 一个 XML 文档  "html" - HTML 作为纯文本  "text" - 纯文本字符串  "script" - 以 JavaScript 运行响应，并以纯文本返回  "json" - 以 JSON 运行响应，并以 JavaScript 对象返回  "jsonp" - 使用 JSONP 加载一个 JSON 块，将添加一个 "?callback=?" 到 URL 来规定回调 |

1. 实例：
2. 请求 "test.php"，但是忽略返回结果：

$.get("test.php");

1. 请求 "test.php" 并连同请求发送一些额外的数据（忽略返回结果）：

$.get("test.php", { name:"Donald", town:"Ducktown" });

1. 请求 "test.php" 并传递数据数组到服务器（忽略返回结果）：

$.get("test.php", { 'colors[]' : ["Red","Green","Blue"] });

1. 请求 "test.php" 并提醒请求的结果：

$.get("test.php", function(data){

alert("Data: " + data);

});

#### get(index)

#### index()

#当前元素在该类元素中排第几?也就是索引集团

如果参数是一组DOM元素或者jQuery对象，那么返回值就是传递的元素相对于原先集合的位置。

如果参数是一个选择器，那么返回值就是原先元素相对于选择器匹配元素中的位置。如果找不到匹配的元素，则返回-1。

#### data()

$.data() 函数用于在指定的元素上存取数据，返回设置值。

1. 定义和用法

**lamp**提示：

* 这是一个底层方法，.data() 方法更方便使用。
* 通过data()函数存取的数据都是临时的，一旦页面刷新，之前存放的数据都将被移除。
* 该方法目前并不提供在XML文档上跨平台设置，Internet Explorer不允许在XML文档中通过自定义属性附加数据。

1. 语法
2. 用法一

*$*.data( element, key, value )

**lamp**注意：

通过data()函数存取的数据都是临时数据，一旦页面刷新，之前存放的数据都将被移除

undefined 是一个没有被识别数据值。调用 jQuery.data( el, "name", undefined ) 将返回对应的 "name" 数据，等价于 jQuery.data(el, "name" )。

我们可以在一个元素上设置不同的值，并获取这些值：

jQuery.data(document.body, 'foo', 52);

jQuery.data(document.body, 'bar', 'test');

1. 用法二

*$*.data( element, key )

我们可以在一个元素上设置不同的值，并获取这些值：

alert(jQuery.data( document.body, 'foo' ));

alert(jQuery.data( document.body ));

|  |  |
| --- | --- |
| 参数 | 描述 |
| element | Element类型 要存储数据的DOM对象 |
| key | 可选。String类型 指定的键名字符串。 |
| value | 可选。 Object类型 需要存储的任意类型的数据。 |

1. 设置

data({'s':10});

1. 获取

i=data('s');

### 选择器

选择器的作用就是为了选中指定的元素对象，只有选择的对象，才能进行下一步操作，jQuery 选择器允许您对 HTML 元素组或单个元素进行操作。jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。它基于已经存在的 [CSS 选择器](http://www.runoob.com/cssref/css-selectors.html)，除此之外，它还有一些自定义的选择器。jQuery 中所有选择器都以美元符号开头：$()。在jquery中提供了非常多的选择器，常用的部分选择器如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 类别 | 选择器 | 实例 | 选取 |
|  | 基础 | [\*](http://www.w3school.com.cn/jquery/selector_all.asp) | $("\*") | 所有元素，基本不能单独使用，一般与前导符一起使用。 |
|  | [#id](http://www.w3school.com.cn/jquery/selector_id.asp) | $("#lastname") | id="lastname" 的元素 |
|  | [.class](http://www.w3school.com.cn/jquery/selector_class.asp) | $(".intro") | 所有 class="intro" 的元素 |
|  | [element](http://www.w3school.com.cn/jquery/selector_element.asp) | $("p") | 所有 <p> 元素 |
|  | .class.class | $(".intro.demo") | 所有 class="intro" 且 class="demo" 的元素 |
|  | 层级 | a b | $(".div1 h2") | 获得div1元素内部所有h2的后代元素。（父子,爷孙…） |
|  | a>b | $(".div1>h2") | 获得div1元素内部所有名为h2子元素。（只找父子关系，不找孙子及以后） |
|  | a+b | $(".div1+h2") | 获得div1元素后面的第一个兄弟h2。（兄弟） |
|  | a~b | $(".div1~h2") | 获得div1元素后面的所有的兄弟h2。（兄弟 |
|  | 基本 | :first | $("p:first") | 第一个 <p> 元素 |
|  | :last | $("p:last") | 最后一个 <p> 元素 |
|  | :even | $("tr:even") | 所有偶数 <tr> 元素 |
|  | :odd | $("tr:odd") | 所有奇数 <tr> 元素 |
|  | :eq(index) | $("ul li:eq(3)") | 列表中的第四个元素（index 从 0 开始） |
|  | [:gt(no)](http://www.w3school.com.cn/jquery/selector_gt.asp) | $("ul li:gt(3)") | 列出 index 大于 3 的元素 |
|  | :lt(no) | $("ul li:lt(3)") | 列出 index 小于 3 的元素 |
|  | :not(selector) | $("input:not(:empty)") | 所有不为空的 input 元素 |
|  | 属性 | [[attribute]](http://www.w3school.com.cn/jquery/selector_attribute.asp) | $("[href]") | 所有带有 href 属性的元素 |
|  | [[attribute=value]](http://www.w3school.com.cn/jquery/selector_attribute_equal_value.asp) | $("[href='#']") | 所有 href 属性的值等于 "#" 的元素 |
|  | [attribute!=value] | $("[href!='#']") | 所有 href 属性的值不等于 "#" 的元素 |
|  | [[attribute$=value]](http://www.w3school.com.cn/jquery/selector_attribute_end_value.asp) | $("[href$='.jpg']") | 所有 href 属性的值包含以 ".jpg" 结尾的元素 |
|  | [attr\*=val] | $("[href\*='.jpg']") | 所有 href 属性的值包含".jpg" 的元素 |
|  | 子元素 | :nth-child | $(" div h1:nth-child(2)") | 每一个div里面的第2个h1，请注意，数字从1开始，不是从0开始。 |
|  | :first-child | $("div h1:first-child") | 每一个div里面的第一个h1 |
|  | :last-child | $(" div h1:last-child") | 每一个div里面的最后一个h1 |
|  | :only-child | $(" div h1:only-child") | 找到子元素只有一个h1的div |
|  | 表单 | [:input](http://www.w3school.com.cn/jquery/selector_input.asp) | $(":input") | 所有 <input> 元素 |
|  | [:text](http://www.w3school.com.cn/jquery/selector_input_text.asp) | $(":text") | 所有 type="text" 的 <input> 元素 |
|  | [:password](http://www.w3school.com.cn/jquery/selector_input_password.asp) | $(":password") | 所有 type="password" 的 <input> 元素 |
|  | [:radio](http://www.w3school.com.cn/jquery/selector_input_radio.asp) | $(":radio") | 所有 type="radio" 的 <input> 元素 |
|  | [:checkbox](http://www.w3school.com.cn/jquery/selector_input_checkbox.asp) | $(":checkbox") | 所有 type="checkbox" 的 <input> 元素 |
|  | [:submit](http://www.w3school.com.cn/jquery/selector_input_submit.asp) | $(":submit") | 所有 type="submit" 的 <input> 元素 |
|  | [:button](http://www.w3school.com.cn/jquery/selector_input_button.asp) | $(":button") | 所有 type="button" 的 <input> 元素 |
|  | [:image](http://www.w3school.com.cn/jquery/selector_input_image.asp) | $(":image") | 所有 type="image" 的 <input> 元素 |
|  | [:file](http://www.w3school.com.cn/jquery/selector_input_file.asp) | $(":file") | 所有 type="file" 的 <input> 元素 |
|  | :hidden | $(":hidden") | 所有 type="hidden" 的 <input> 元素 |
|  | 表单  属性 | [:checked](http://www.w3school.com.cn/jquery/selector_input_checked.asp) | $(":checked") | 所有被选中的 input 元素 |
|  | [:selected](http://www.w3school.com.cn/jquery/selector_input_selected.asp) | $(":selected") | 所有被选取的 input 元素 |
|  | 其它 | [:header](http://www.w3school.com.cn/jquery/selector_header.asp) | $(":header") | 所有标题元素 <h1> - <h6> |
|  | [:animated](http://www.w3school.com.cn/jquery/selector_animated.asp) |  | 所有动画元素 |
|  | [:contains(text)](http://www.w3school.com.cn/jquery/selector_contains.asp) | $(":contains('W3School')") | 包含指定字符串的所有元素 |
|  | [:empty](http://www.w3school.com.cn/jquery/selector_empty.asp) | $(":empty") | 无子（元素）节点的所有元素 |
|  | :hidden | $("p:hidden") | 所有隐藏的 <p> 元素 |
|  | [:visible](http://www.w3school.com.cn/jquery/selector_visible.asp) | $("table:visible") | 所有可见的表格 |
|  | s1,s2,s3 | $("th,td,.intro") | 所有带有匹配选择的元素 |
|  | [:enabled](http://www.w3school.com.cn/jquery/selector_input_enabled.asp) | $(":enabled") | 所有激活的 input 元素 |
|  | [:disabled](http://www.w3school.com.cn/jquery/selector_input_disabled.asp) | $(":disabled") | 所有禁用的 input 元素 |

### 筛选器

jquery中的对象选择是由选择器和筛选器组合而成，目的是为了更方便的选择对象。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 类别 | 选择器 | 实例 | 选取 |
|  | 过滤 | eq() | $("h1").eq(0) | 查找索引等于0的h1元素 |
|  | first() | $("h1").first() | 查找第1个h1元素 |
|  | last() | $("h1").last() | 查找最后1个h1元素 |
|  | not() | $("h1").not($("h1").last（）) | 查找除了最后一个所有h1元素，注意括号m内一定要先有对象，再有方法 |
|  | slice() | $("h1").slice(0,3) | 截取指定范围的h1元素,不包括最后一个 |
|  | 查找 | children() | $("div1").children() | 查找div1的所有子元素，不包含孙子（含）以后 |
|  | find() | $(" div1").find("h1") | 查找div1的所有后代元素，包含孙子（含）… |
|  | next() | $("h1").next() | 下一个兄弟h1元素 |
|  | nextAll() | $("h1").nextAll() | 后面所有的兄弟h1元素 |
|  | parent() | $("h1").parent() | 父元素 |
|  | prev() | $("h1").prev() | 前一个兄弟h1元素 |
|  | prevAll() | $("h1").prevAll() | 前面所有的兄弟h1元素 |
|  | siblings() | $("h1").siblings() | 前后所有的兄弟h1元素 |
|  | 串联 | add() | $("h1").add("p") | 所有的h1和p元素，组合选择器 |
|  | andSelf() | $("h1").next().andSelf() | 找到其它元素后连接自己 |

### 属性操作方法

选择了对象以后，就要对对象进行操作，本节介绍常见的对象属性操作方法。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 类别 | 选择器 | 实例 | 选取 |
|  | 属性 | attr() | $("h1").attr({"age"，"20}) | 获取、修改、添加属及属性的值，需要键值对 |
|  | removeAttr() | $("h1").removeAttr("age") | 删除属性 |
|  | 类 | addClass() | $("div1").addClass() | 添加某个类 |
|  | removeClass() | $("div1").removeClass() | 从某个类删除 |
|  | toggleClass() | $("h1").toggleClass() | 不属于就添加，属于就删除 |
|  | HTML | html() | $("h1").html() | 标签内部所有子标签和文本 |
|  | 文本 | text() | $("h1").text() | 标签内部所有文本，不包含标签 |
|  | 值 | val() | $("h1").val() | 只有表单元素才有val,代表value。 |

### 文本处理

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 类别 | 选择器 | 实例 | 选取 |
|  | 内部插入 | append() | $("div2").append($(his)) | 向div2追加对象 |
|  | appendTo() | $("div1").appendTo("div2") | 追加到指定对象 |
|  | prepend() | $("div2").prepend($(his)) | 向div2前面追加 |
|  | prependTo() | $("div1").prependTo($(his)) | 追加到对象前面 |
|  | 外部插入 | after() | $("div2").after($(this)) | 向div2后面插入 |
|  | insertAfter() | $($(this)).insertAfter("div2") | 插入div2后面 |
|  | before() | $("div2").before($(this)) | 向div2前面插入 |
|  | insertBefore() | $($(this)).insertBefore("div2") | 插入div2前面 |
|  | 包围 | wrap() | $("p").wrap("<h2>") | 把所有的p标签单独放入不同h2标签对中 |
|  | wrapAll() | $("p").wrapAll("<h2>") | 把所有的p标签放入一个h2对标签中 |
|  | wrapInner() | $("p").wrapInner("<i>") | 在所有的p标签中单独放入一对i标签中 |
|  | 复制 | clone() | $($(this)).clone().insertAfter($(this)) | 自我复制后放在最后面,但复制品不能再被复制，即不复制事件 |
|  | clone(true) | $($(this)).clone(true).insertAfter($(this)) | 自我复制且复制事件 |

### Css

#### css

css();

#### 位置

offset();

position();

scrollTop();

scrollLeft();

#### 尺寸

height();

width();

innerHeight();

innerWidth();

outerHeight();

outerWidth();

### 事件

#### 页面载入

ready(fn);

#当文档准备完毕时

#当文档加载完毕时(onload)

#### 事件处理

bind();

#$('.close').on('click',function(){

//代码

});

unbind();

#$('.close').off('click');

one();

#### 事件委派

live();

#$(document).on('click','.close',function(){

//代码

});

die();

#$(document).off('click','.close');

#### 事件切换

hover();

#### 事件类型

obj.onclick=function(){};

obj.click(function(){});

### 效果

#### 基本

show(speed,[fn]);

hide(speed,[fn]);

toggle(speed,[fn]);

#### 滑动

slideDown(speed,[fn]);

slideUp(speed,[fn]);

slideToggle(speed,[fn]);

#### 淡入淡出

fadeIn(speed,[fn]);

fadeOut(speed,[fn]);

fadeTo(speed,opacity,[fn]);

#### 自定义

animate({},600,function(){});

### Ajax

1)ajax请求

2)ajax工具

[实例1: 无刷新删除]

[实例2: 无刷新登录]

### 工具

#### $.inArray();

#### $.merge();

#### $.unique();

#### $.parseJSON();

#### $.isArray();

#### $.isFunction();

#### $.isEmptyObject();

#### $.trim();

### jQuery扩展实例

#### 表单验证

#### 城市级联

#### 拖动图片

#### 轮播图