

# 初识 javascript

## 1. JavaScript 简介

JavaScript（缩写：JS）是互联网上最流行的脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。

最早是在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网页增加动态功能。

## 2. JavaScript 的特点

- 1). 脚本语言。JavaScript 是一种解释型的脚本语言, C、C++等语言先编译后执行, 而 JavaScript 是在程序的运行过程中逐行进行解释。
- 2). 基于对象。JavaScript 是一种基于对象的脚本语言, 它不仅可以创建对象, 也能使用现有的对象。
- 3). 简单。JavaScript 语言中采用的是弱类型的变量类型, 对使用的数据类型未做出严格的要求, 是基于 Java 基本语句和控制的脚本语言, 其设计简单紧凑。
- 4). 动态性。JavaScript 是一种采用事件驱动的脚本语言, 它不需要经过 Web 服务器就可以对用户的输入做出响应。
- 5). 跨平台性。JavaScript 脚本语言不依赖于操作系统, 仅需要浏览器的支持。

## 3. JavaScript 的作用

HTML 页面是静态的，而 JavaScript 可以弥补 HTML 语言的缺陷，实现 Web 页面客户端的动态效

JavaScript 的作用有以下几点：

### 1). 动态改变网页内容

HTML 页面是静态的，一旦编写，内容是无法改变的。JavaScript 可以弥补这个不足，可以将内容动态地显示在网页中。

### 2). 动态改变网页外观

JavaScript 通过修改网页元素的 CSS 样式，达到动态地改变网页的外观。

### 3). 验证表单数据

可以验证用户在表单中填写的数据。

### 4). 响应事件

JavaScript 是基于事件的语言。例如点击一个按钮弹出一个对话框，就是鼠标点击触发的事件。

JavaScript 可以输出字符、数字、以及 HTML

更多学习资料请关注微信公众号“**前端大学**”后回复“自学资料”  
免费自动下载！

打开微信扫一扫：



## 第一个 js 语句

### 1. 输出字符 “hello word”

代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>自学前端就在 微信公众号： 前端大学 </title>
6 </head>
7 <body>
8   <h1>我的第一个 Web 页面</h1>
9   <p id="demo">我的第一个段落。</p>
10  <script>
11    document.getElementById("demo").innerHTML="段落已修改。";
12  </script>
13 </body>
14 </html>

```

效果：

我的第一个 Web 页面

段落已修改。

## 2. 输出数字

代码如下：

```

<body>
  <h1>我的第一个 Web 页面</h1>
  <p id="demo">我的第一个段落。</p>
  <script>
    document.write(123456789);
  </script>
</body>

```

效果：

123456789

### 3. 输出 HTML

代码如下：

```
<!DOCTYPE html>
<html>
<body>
<button onclick="myFunction()">点我</button>
<script>
function myFunction() {
    document.write(Date());
}
</script>
</body>
</html>
```

效果：



## js 的引入方式

学习了 HTML 之后，我们都知道 HTML 页面是静态的，要实现某些动态效果，就要引入 JavaScript。

JavaScript 在 HTML 的引用共有 4 种：

- 1). 页头引入（head 标签内）；
- 2). 页中引入（body 标签内）；
- 3). 元素事件中引入（标签属性中引入）；
- 4). 引入外部 JS 文件。

#### 1. 页头引入

在 HTML 中引入 JavaScript 需要将代码放在<script>与</script>之间。

页头引入就是将<script>标签放在<head>标签中

举例：在页头引入 JavaScript 代码弹出警告窗口，显示内容“**自学前端就在微信公众号：“前端大学”**”

代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>前端大学</title>
  <script>

    alert("自学前端就在微信公众号： 前端大学 ");

  </script>
</head>
<body >
</body>
</html>
```

效果：



说明：

在<head>标签中引入 JavaScript 代码，代码 alert 的效果是弹出一个窗口。

## 2. 页中引入

页头引入就是将<script>标签放在<body>标签中

举例：在页中引入 JavaScript 代码弹出警告窗口，显示内容“**自学前端就在微信公众号：“前端大学”**”

代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

    <title>前端大学</title>
</head>
<body >
    <script>

        alert("自学前端就在微信公众号： “ 前端大学 ”");

    </script>
</body>
</html>

```

效果：



说明：

和在页头引入是一个效果。

### 3. 元素事件中引入

在元素事件中引入 JavaScript 就是在标签内设置事件属性，调用 JavaScript。

举例：在 html 标签中通过事件响应引入 JavaScript 代码弹出警告窗口，显示内容“自学前端就在微信公众号： “ 前端大学 ””

代码如下：

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>前端大学</title>
</head>
<body >

    <p onclick="alert('自学前端就在微信公众号： “ 前端大学 ”')">点我

</p>
</body>
</html>

```

效果：

说明：

在这段代码中,我们设置了鼠标点击事件,当鼠标点击了这个 p 标签内的内容“点我”时,弹出一个窗口。

#### 4. 引入外部 JS 文件

引入外部文件是常用的引用方法。

语法：

```
<script type="text/javascript" src="URL"></script>
```

这段代码可以放在 head 标签中,也可以放在 body 标签中。

举例：在后缀为 js 的文件中写 JavaScript 代码弹出警告窗口,显示内容“分针学习网是个不错的网站”。通过引用这个 js 文件来执行 JavaScript 代码。

html 部分代码：

```
<body >  
<script type="text/javascript"src="../js/qianduandaxue.js"></script>  
</body>
```

Javascript 代码：

```
alert(‘自学前端就在微信公众号：“前端大学”’);
```

效果：



说明：

在 HTML 中从外部引入 JavaScript,通过调用.js 为后缀的文件来引入。

建议：

从外部引入 JS 文件放在 body 的结尾处,因为网页加载是从上到下加载,如果在开头引入 JS 文件,有些代码不能正确执行。

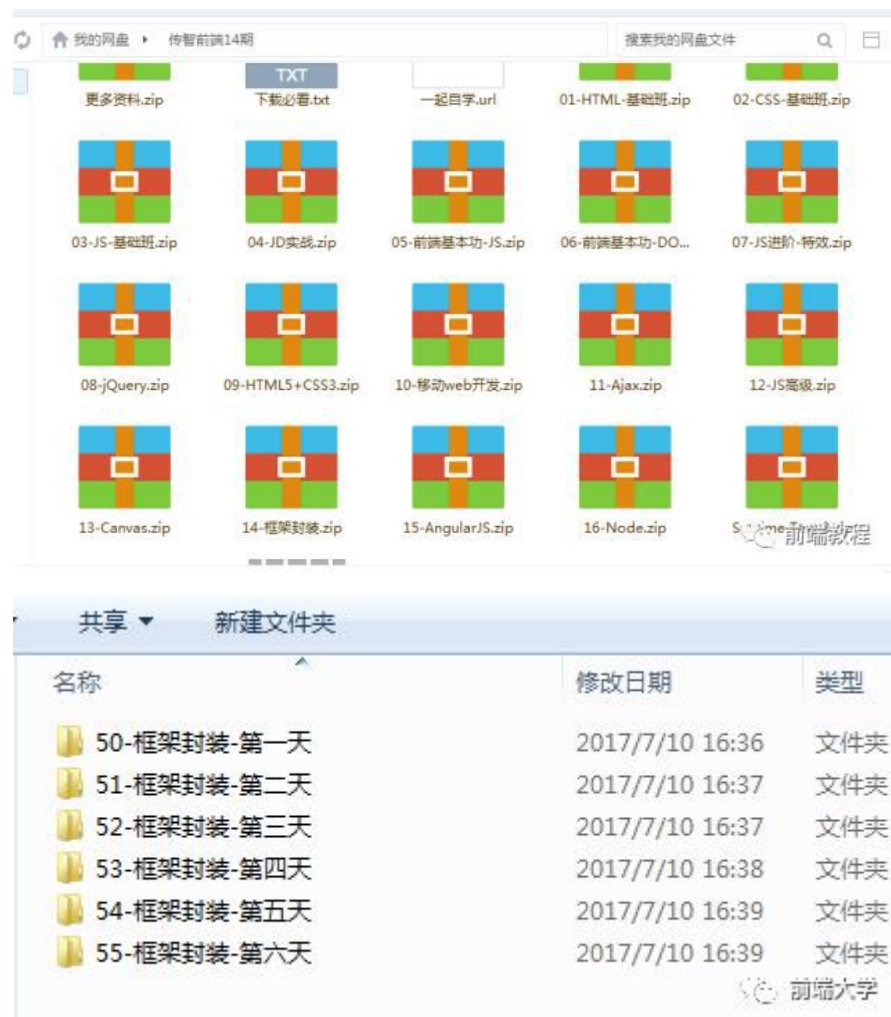
总结：

HTML 中引入 JavaScript 有四种方式，页头引入，页中引入，元素事件引入，外部引入 外部引入 JS 文件要把代码放在 body 标签的末尾，在结束之前。

福利来啦！学前端不易，好的前端视频教程对一个人的学习至关重要！！！请看截图！全套最新“某智”最新 web 前端开发培训全套视频教程+源码+笔记+ppt（打包下载直接解压）（具体的内容请看截图！）

百度云下载链接 <http://pan.baidu.com/s/1gfFxqBh>

提取密码请关注 微信公众号“前端大学” 回复 “某智提取密码” 6 个字自动获取。





工具(T) 帮助(H)		
共享 ▾ 新建文件夹		
名称	修改日期	类型
28-ajax-第一天	2017/7/7 16:20	文件夹
29-ajax-第二天	2017/7/7 16:20	文件夹
30-ajax-第三天	2017/7/7 16:21	文件夹
31-ajax-第四天	2017/7/7 16:21	文件夹
32-ajax-第五天	2017/7/7 16:21	文件夹

。。。。等（咱们是一个自学者组成的联盟！拒绝任何形式的付费！

努力自学！）

某智 2016 年 11 月 web 前端开发培训全套视频教程+源码+笔记+ppt（打包下载直接解压）

包含从 0 基础入门到 javascript 初级入门，再到 javascript 深入学习，再到各个 javascript 框架的学习和综合运用，中间穿插着多个实例实战。。。快来关注下载吧！ 微信搜索公众号 “前端大学”，记住名字只有 “前端大学”

4 个字哦！

咱们的公众号里还有更多直接下载的资料，不定期更新哦！

## 前 端 大 学

分享前端相关技术干货·资讯·高薪职位·教程



长按识别二维码关注前端大学

# js 的输出

JavaScript 没有任何打印或者输出的函数。JavaScript 可以通过不同的方式来输出数据，常用的 3 中输出方式有：

- 1). 使用 `window.alert()` 弹出警告框。
- 2). 使用 `document.write()` 方法将内容写到 HTML 文档中。
- 3). 使用 `console.log()` 写入到浏览器的控制台。

## 1. 使用 弹出警告框

通过 `alert()` 方法输出数据。代码如下：

```
<script>  
    alert("我是警告框，输出 hello,world!");  
</script>
```

效果：



说明：

在 `alert()` 方法中写入内容，网页会弹出警告窗口，窗口中显示写入的内容。

## 2. 直接写入 HTML 文档

`document.write()` 方法来输出想要显示的内容。

代码如下：

```
<script>  
    document.write("这是输入内容，输入到 HTML 中");  
</script>
```

效果：



#### 说明：

这是通过 `document.write()` 直接输出到文档中 注意： 如果在文档已完成加载后执行 `document.write`，整个 HTML 页面将被覆盖。

### 3. 写到控制台

通过 `console.log()` 将数据写入到控制台，数据只会在控制台中显示，并不会显示在网页中。 举例：在控制台输出“我被写入了控制台”。 代码如下：

```
<script>
  console.log('我被写入控制台了');
</script>
```

#### 效果：

按F12可以查看



#### 说明：

通过 `console.log()` 方法写入到控制台中，页面是没有我们输入的这个数据。通常 `console.log()` 被用来检查错误。

#### 总结：

JavaScript 常用的三种输出方式有：使用警告框、直接输出到网页中、写入控制台。

# js 语句

## 1. JavaScript 语句

JavaScript 语句向浏览器发出的命令。语句的作用是告诉浏览器该做什么。

下面的 JavaScript 语句向 id="demo" 的 HTML 元素输出文本 "Hello World":

```
document.getElementById("demo").innerHTML="Hello World";
```

分号的作用:

- 1). 分号用于分隔 JavaScript 语句。
- 2). 通常我们在每条可执行的语句结尾添加分号。
- 3). 使用分号的另一用处是在一行中编写多条语句。

提示:

您也可能看到不带有分号的案例。

在 JavaScript 中, 用分号来结束语句是可选的。

## 2. JavaScript 代码

JavaScript 代码 (或者只有 JavaScript) 是 JavaScript 语句的序列。

浏览器会按照编写顺序来执行每条语句。

本例将操作两个 HTML 元素

代码如下:

```
<h1>My Web Page</h1>
<p id="demo">A Paragraph.</p>
<div id="myDIV">A DIV.</div>
<script>
document.getElementById("demo").innerHTML="Hello World";
document.getElementById("myDIV").innerHTML="How are you?";
</script>
```

效果:

### 3. JavaScript 对大小写敏感

- 1). JavaScript 对大小写是敏感的。
- 2). 当编写 JavaScript 语句时，请留意是否关闭大小写切换键。
- 3). 函数 getElementById 与 getElementbyID 是不同的。
- 4). 同样，变量 myVariable 与 MyVariable 也是不同的。

### 4. 根据 HTML 文档流的执行顺序

- 1). 需要在页面元素渲染前执行的 js 代码应该放在<body>前面的<script>代码块中，
- 2). 而需要在页面元素加载完后的 js 放在</body>元素后面，
- 3). body 标签的 onload 事件注册的函数是在最后执行的。另外请记住，window.onload 就是 body 上注册的 onload 事件，那么如果我们
- 4). body 之前声明了 window.onload = function() {} ;将会不起作用，原因是被 body 中的 onload 覆盖，然而，如果我们把
- 5). window.onload 放在 body 之后的代码块，那么他就会覆盖 body 中注册的 onload 事件，这个相对于\$(document).ready(function()
- 6). {} ) 执行要晚，jQuery 是在 Dom 树加载完之后就运行，也就是标签到位就行，它并不像 window.onload 要求所有资源都被加载完成后
- 7). 想要获取一个元素的高度或宽度，只要元素被加载后就能获取到，但是必须 display 不为 none

# 基本概念

**1. JavaScript 的全部关键字：** 1). **JavaScript 保留关键字** break case catch continue debugger default delete do else false finally for function if in instanceof new null return switch this throw true try typeof var void while with 2). **JavaScript 将来可用的保留关键字** abstract boolean byte char class const double enum export extends final float goto implements import int interface long native package private protected public short static super synchronized throws transient volatile 3). **应避免使用的单词**

那些已经用作 JavaScript 的内置对象或内置函数的名称,如 String 或 parseInt 等。 注意: 像 goto、const、byvalue 等,他们是 JavaScript 的一部分,但是他们在 JavaScript 中没有定义任何特殊功能,只是为了保持相互兼容而已。

## **2. 标识符 1). 标识符格式**

标识符必须是以字母、下划线、美元号等符号开头的,其后可以是零个或若干个字母、数字、下划线、美元号等符号组成的字符串。在 JavaScript 中预定义的运算符如: +、-、\*、/、%不可以用于定义标识符。 2). **Smalltalk 法则** 每个标识符可以有若干个单词左右连接而成,常量标识符应该全部使用大写字母来表示区别,一般标识符应该全部使用小写字母以示区别,特殊常量标识符应该以大写字母开头以示区别,函数的标识符应该以小写字母开头以示区别,不要使用 JavaScript 中预定义保留的关键字。 JavaScript 严格区分大小写字母。

## **3. 常量**

JavaScript 中的用 const 来声明定义常量。const 定义的常量必须初始化,不初始化会报错。常量在第一次定义之后不可再次赋值,如果在严格模式下运行会报错。

举例: 定义一个常量,在之后使用中试图改变常量的值  
代码如下:

```
const b=10;
document.write("定于的常量 b 的值为"+b+"<br>");
b=20;//试图改变常量的值
document.write("改变之后的常量 b 的值为: "+b);
```

效果:

```
定于的常量b的值为10
改变之后的常量b的值为: 10
```

说明:

在定义常量之后便不可再次修改常量的值。

修改常量的值虽然不会报错，但是也修改不成功。 注意:

1. 在 Firefox 和 Chrome 更早期的版本，Safari 5.1.7 和 Opera 12.00，如果使用 const 定义一个变量，这个变量的值仍然可以修改。
2. IE6-10 不支持 const，但是 IE11 支持。

## 4. 变量

变量是用于存储信息的“容器”。 在应用程序中，使用变量来作为值的符号名。

变量的名字又叫做标识符，其需要遵守一定的规则。

格式:

```
var 变量名 = 变量值;
```

用 var 声明的且未赋初值的变量，值会被设定为 undefined。

举例：声明 3 个变量，并输出 代码如下：

```
<script>
  var num=123;
  var str="我爱 前端大学 微信公众号";
  var a;
  document.write(num+"<br>");
  document.write(str+"<br>");
  document.write(a);
</script>
```

效果:



### 1). 变量的作用域

在所有函数之外声明的变量，叫做**全局变量**，因为它可被当前文档中的任何其他代码所访问。

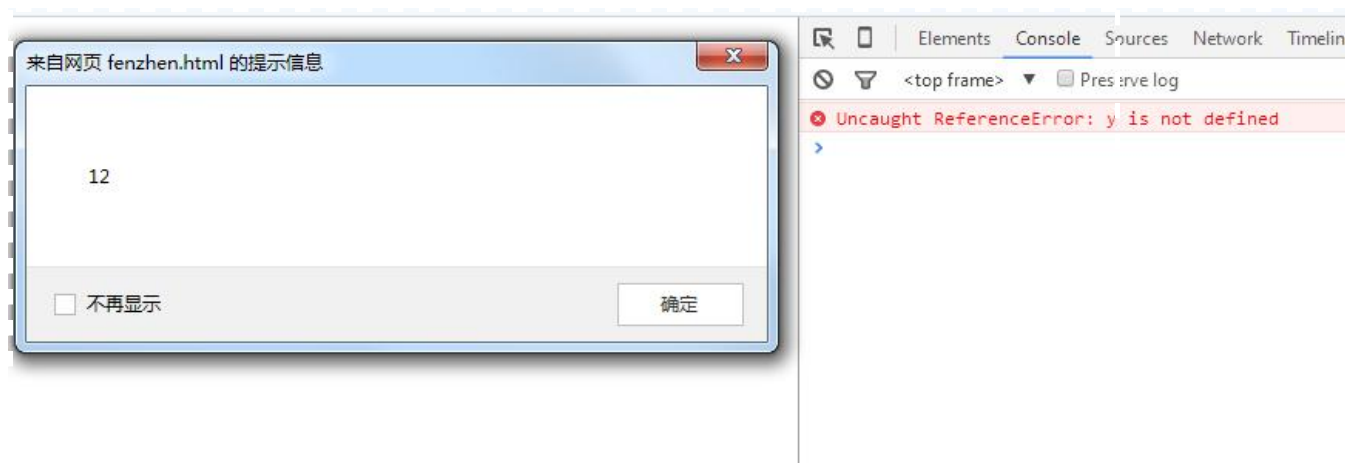
在函数内部声明的变量，叫做**局部变量**，因为它只能在该函数内部访问。

举例：在函数外定义变量 x，然后再函数中使用。在函数内定义变量 y，然后再函数外使用。

代码如下：

```
<script>
  var x=12;
  function myfunc() {
    alert(x);
    var y=110;
  }
  myfunc();
  document.write(y);
</script>
```

效果：



说明：

在函数内声明的变量在函数外使用会出错，引用错误。



注意：

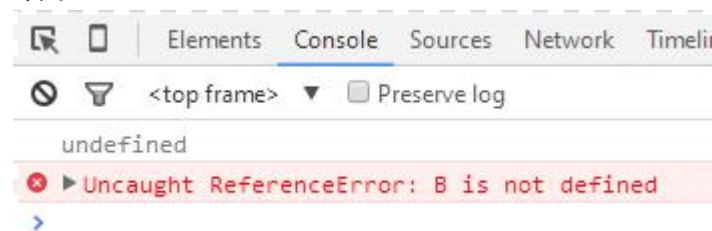
声明变量要在使用变量之前。

变量名对大小写敏感，a 和 A 代表不同的两个变量

举例：

```
<script>
  console.log(a);
  var a=10;
  var b=123;
  console.log(B);
</script>
```

效果：



说明：

如果在使用变量之后声明变量，会返回 undefined。

总结：

变量是用于存储信息的“容器”。

变量名又叫做标识符，其需要遵守一定的命名规则。

变量名对大小写敏感，同一个字母，大小写不同表示不同的变量。

## 5. js 注释 注释

JavaScript 注释可用于提高代码的可读性。 注释不会被执行。 1). 单行注释

单行注释以 // 开头。

举例：

```
<script>
  //弹出一个警告框
  alert("前端大学 微信公众号！")
  var x=1;    //也可以在这里注释
</script>
```

### 2). 多行注释

多行注释以 /\* 开始，以 \*/ 结尾。

举例：

```
<script>
  /*
    定义一个变量 num，并给变量赋值
    在控制台输出变量 num
  */
  var num=123;
  document.write(num);
</script>
```

效果：

123

提示：

注释常用来调试程序，因为注释掉的代码并不会被执行。

总结：

在写代码的时候，注释是很有必要的，提高了代码的可读性。要养成写注释的习惯。

## 课后练习

1. 定义一个函数计算给定周长的圆的面积，比如给定圆周长为 20，计算圆的面积
2. 定义 2 个函数，计算两个 a, b 两个变量的和，其中一个函数，将所求的和加 1，另一个函数一个减 1；并拆分优化这个函数
3. 计算  $1+2+3+***+100$  的值
4. 定义一个 person 的函数，有一个局部变量 name，如果我要在函数外调用和修改 name 变量，要怎么实现呢。
5. 输出以下代码块输出的值是什么，为什么？

```
var i = 0;
```

```
console.log(i, 1);  
function change() {  
    console.log(i, 2);  
    i++;  
}  
console.log(i, 3);  
change();  
console.log(i, 4);  
//代码二  
var i = 0;  
console.log(i, 21);  
function change() {  
    var i = 10;  
    console.log(i, 22);  
    i++;  
}
```

## Js 基本语法

## 数据类型

### 1. 字符串

#### 1). 字符和字符串

什么是字符？字符是指计算机中使用的字母、数字、字和符号，包括：1、2、3、A、B、C、·#¥%. . . . . - \* ( ) -- + 等等。1 个汉字字符存储需要 2 个字节，1 个英文字符存储需要 1 个字节 。

因为就是说我们日常看到的都是字符。

什么是字符串？

字符串(String)是由数字、字母、下划线组成的一串字符，是一个 string 对象。

例子：声明几个字符串

```
var string11="Nice to meet you!";
var string12='He is called Bill ';
var string21="He is called 'Bill'";
var string22='He is called "Bill"';
```

说明：

- 1). 字符串可以用双引号引起来，也可以用单引号引起来声明哦。
- 2). 用双引号引起来的字符串里面有需要用到引号的，就必须用单引号。
- 3). 用单引号引起来的字符串里面有需要用到引号的，就必须用双引号。
- 4). 上面的多种格式声明字符串都是可以的，同学们可以根据自己喜好选择。

2. 转义字符

所有的 ASCII 码都可以用 “\” 加数字（一般是 8 进制数字）来表示。而 C 中定义了一些字母前加“\”来表示常见的那些不能显示的 ASCII 字符,如\0, \t, \n 等，就称为转义字符，因为后面的字符，都不是它本来的 ASCII 字符意思了 。

什么意思呢？也就是说，在加了\后表示的不再是原来字符的意思，转变了意思。同学们可以这样理解他。

转义字符	ASCII 码值（十进制	意义
\a	007	响铃 (BEL)
\b	008	退格 (BS) ， 将当前位置移到前一行
\f	012	换页 (FF)， 将当前位置移到下页开头
\n	010	换行 (LF) ， 将当前位置移到下一行开头
\r	013	回车 (CR) ， 将当前位置移到本行开头
\t	009	水平制表 (HT) （跳到下一个 TAB 位置）
\v	011	垂直制表 (VT)
\\	092	代表一个反斜线字符'\'

\'	039	代表一个单引号（撇号）字符
\"	034	代表一个双引号字符
\?	063	代表一个问号
\0	000	空字符(NULL)
\ooo	三位八进制	1 到 3 位八进制数所代表的任意字符
\xhh	二位十六进制	1 到 2 位十六进制所代表的任意字符

**说明：**

同学们可能觉得我们学的又不是 c 语言，为什么要知道转义字符。语言是相通的，这个对于 js 也是一样的，只是我们常用的主要是\"，\\等。所以同学们可以只知道常用的的意义就好。

### 3. 数字

#### 1). 整型数据

例子 2：声明一个整型数据

```
var x2=34;           //不使用小数点来写
```

**说明：**

1). 整型就是不含小数点的数据。

#### 2). 浮点型数据

例子 3：声明一个浮点型数据

```
var x1=34.00;        //使用小数点来写
```

**说明：**

数据的值含有小数点，那么它就是浮点型数据。

### 4. 布尔

布尔型数据就是逻辑类型，它只能有两个值：true 或 false。

例子 4：声明浮点类型的数据。

```
var x=true;
var y=false;
var M=True;
var N=False;
```

**说明：**

- 1). 浮点型数据只有两个值，true 或 false。所以上例中的 M, N 不是布尔类型哦。
- 2). 浮点型数据常用于判断。

## 5. 数组

**定义：**

数组是一组相同数据类型的集合。就像人类是所有人的集合。我们现在讲一下定义一个数组有哪些方式。

例子 5：先声明一个数组，然后再赋值。

```
var people=new Array();  
people[0]="name";  
people[1]="sex";  
people[2]="old";
```

**说明：**

上面例子的意思就是先声明这里有一个数组类型的对象，但是值是什么还没有给。之后 3 句话才给他们赋值。

例子 6：声明的时候就把数组的值给出。

```
var people=new Array("name", "sex", "old");
```

**说明：**

上例子就是在声明的同时赋值。

**总结：**

- 1). 一般都使用第二种方式定义数组，以免出现问题。
- 2). 具体使用什么方法，同学们根据实际情况选择自己最方便的方式，并不是强制的哦。

## 6. 对象

**定义：**

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数……

此外，JavaScript 允许自定义对象。

*对象只是带有属性和方法的特殊数据类型。*

通俗地讲，人类是一个对象，我们的名字，性别就是我们的属性，我们可以跑，跳高这些操作，就是方法。

例子 7：定义一个对象

```
var person={name:"小明", sex:"男", old:20};
```

说明：

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式（name：value）来定义。

属性：

属性的值

```
var person={name:"小明", sex:"男", old:20};
```

说明：

1). 这是例子 7 中的代码，这里小明，男，20 都是属性的值，分别是 name, sex, old 这是三个属性的值。

2). 给属性赋值

例子 8：通过构造函数来给属性赋值

```
function person(firstname, lastname, age, eyecolor)
{
  this.firstname=firstname;
  this.lastname=lastname;
  this.age=age;
  this.eyecolor=eyecolor;
}
```

说明：

1). 这是通过构造函数来给属性赋值，什么是构造函数呢？就是新建对象时一定会调用的函数。就像上学一定会带书一样。不过现在同学们肯定不太容易懂，等学习了后面的函数，同学们就能很好；理解了。

2). 除了通过用构造函数给属性赋值还有像例子 7 一样，直接给属性赋值的方式。所以现阶段的同学，可以直接用例 7 的方式给属性赋值。

## 7. NULL

null 表示"没有对象"，即该处不应该有值。

典型用法：

- 1). 作为函数的参数，表示该函数的参数不是对象。
- 2). 作为对象原型链的终点。

## 8. Undefined

`undefined` 表示"缺少值"，就是此处应该有一个值，但是还没有定义。

典型用法是：

- 1). 变量被声明了，但没有赋值时，就等于 `undefined`。
- 2). 调用函数时，应该提供的参数没有提供，该参数等于 `undefined`。
- 3). 对象没有赋值的属性，该属性的值为 `undefined`。
- 4). 函数没有返回值时，默认返回 `undefined`。

总结：

`NULL` 和 `undefined` 的区别几乎没有，都是无。如果硬要区别：`null` 是没有值，值为空。`undefined` 是没有值，没有被定义，所以主要看有没有被定义来区分二者。

更多学习资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



## 查看数据类型

### 1. typeof



typeof 的作用是返回数据的类型。我们首先告诉同学们这个函数的返回值以及代表的数据类型，这样同学们看到后面的例子才不会迷茫。

返回值	对应的数据类型
undefined	这个值未定义
boolean	这个值是布尔值
string	这个值是字符串
number	这个值是数值
object	这个值是对象或
function	这个值是函数

1). 检查一个变量是否存在, 是否有值. typeof 在两种情况下会返回 "undefined": 一个变量没有被声明的时候, 和一个变量的值是 undefined 的时候.

例子 1:

```
var person={name:"小明", sex:"男", old:20};  
document.write( typeof(person) + " , "+typeof(m));
```

结果为:



说明:

1). 根据返回结果我们可以知道 person 是被定义的, 且是对象。而 m 是没有被定义的。

2). 原始值的类型是什么?

例子 2:

```
var person = {name: "小明", sex: "男", old: 20};  
document.write(typeof(person) + " , " + typeof(m) + " , " + typeof  
("aa") + " , " + typeof (90));
```

结果为：

```
object , undefined , string , number
```

# 运算符

## 1. 什么是运算符

执行变量或值之间的运算的符号。也就是说运算符就是我们在表达式里用于计算的特殊符号，比如加减乘除等等。

## 2. 算术运算符

### 1). 加减乘除求余

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘（星号）	x=y*2	x=10
/	除	x=y/2	x=2. 5
%	求余数（保留整数）	x=y%2	x=1

说明：

（例子中假定 y=5）

### 2). 自增自减

运算符	描述	例子	结果
-----	----	----	----

++	自增	x=++y	x=6
--	自减	x=--y	x=4

**说明：**

(例子中假定 y=5)

### 3. 比较运算符

1). 大于，大于等于，小于，小于等于，等于，不等于等

运算符	描述	例子	结果
>	大于	3>4	false
>=	大于等于	3>=4	false
<	小于	3<4	true
<=	小于等于	3<=4	true
=	等于	3=4	false
!=	不等于等	3!=4	true

**说明：**

返回值只会是 true 或者 false，如果成立则为 true，不然为 false.

### 4. 赋值运算符

1). 简单的赋值运算符由等号(=)实现，只是把等号右边的值赋予等号左边的变量。

复合赋值运算是由乘性运算符、加性运算符或位移运算符加等号(=)实现的。

例子 1：简单的赋值运算

```
var iNum = 10;
```

**说明：**

iNum 的值为 10.

2). 算术运算符与=等的结合

算术运算符与=等的结合就是复合赋值运算，复合赋值运算是由乘性运算符、加性运算符或位移运算符加等号(=)实现的。

### 例子 2：复合赋值运算

```
var iNum = 10;  
iNum = iNum + 10;  
//上面的代码等价于下面  
var iNum = 10;  
iNum += 10;
```

#### 说明：

直接将例子 2 的代码复制到编辑器里面是有问题的哦，因为注释上下的代码表示的是同一个意思，老师只是为了让同学更清楚比较所以放在一起了。

## 5. 逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

运算符	描述	例子	结果
&&	and	x < 10 && y > 1	true
	or	x==5    y==5	false
!	not	!(x==y)	true

#### 说明：

- 1). 表格中我们假定 x=4, y=2;
- 2). and 的要求是左右两边都要成立，or 就是两边至少成立一个，not 就是与结果相反，not 之前为 true 则结果为 false。

## 6. 条件运算符

- 1). 三目运算符

#### 语法：

```
变量=(条件)?value1:value2
```

#### 说明：

条件成立则变量的值被赋为 value1, 不成立则赋值为 value2.

#### 例子 3：

```
var m=(3>2)?3:4;  
document.write(m);
```

结果为：

### 总结：

现在同学们学习的运算符有点多,但是其实结合例子理解记忆,还是很好学习的。  
主要就是结合例子以及现实中的数学比较多。

更多学习资料请关注微信公众号“**前端大学**” 免费自动下载！

打开微信扫一扫：



## 表达式

### 1. 什么是表达式？

简单的说表达式，是由数字、运算符、分组符号（括号）、变量和常量等以能求得数值的有意义排列方法所得的组合。

一个**表达式会产生一个值**, 它可以放在任何需要一个值的地方。

## 2. 表达式、语句

- 1). 在程序设计语言中, 语句指的是执行单元, 通常以行为单位, 表达式指的是可用于计算的式子, 即可能产生一个值的式子。语句可以包含有表达式, 表达式也可以单独形成一个语句。
- 2). 语句可以理解成一个行为. 循环语句和 if 语句就是典型的语句. 一个程序是由一系列语句组成的. JavaScript 中某些需要语句的地方, 你可以使用一个表达式来代替. 这样的语句称之为表达式语句. 但反过来不可以: 你不能在一个需要表达式的地方放一个语句. 比如, 一个 if 语句不能作为一个函数的参数。
- 3). 一个表达式会产生一个值, 它可以放在任何需要一个值的地方, 比如, 作为一个函数调用的参数.。

## 3. 简单表达式

### 1). 算术表达式

在数学课程中算术表达式是指由数字和运算符号组成的式子, 可以简单清晰地记录或描述计算过程和内容。在我们 js 里面也是一样的含义哦。

例子 1: 计算算数表达式  $5+3\times 4$ .

```
var m=5+3*4;  
document.write(m);
```

结果为:



### 2). 逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的有意义的式子称为逻辑表达式。

逻辑表达式的值是一个逻辑值, 即 “true” 或 “false” 。

通俗的讲, 逻辑表达式就是由 6 种关系运算符和 3 种逻辑运算符构成的表达式。

看下面的例子。

例子 2:

```
var m=9>6;  
document.write(m);
```

结果为:



true

说明:

六种关系运算符和三种逻辑运算符: =(等于)、<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、<>(不等于)

### 3). 三目表达式

三目表达式就是三元运算符构成的式子。三元运算符也就是前面学习的条件运算符。

例子 3:

```
var m=(3>2)?3:4;  
document.write(m);
```

结果为:



3

## 4. 复杂表达式

### 1). 简单表达式的组合

复杂表达式是由原始表达式和操作符(operator)组合而成，包括属性访问表达式、对象创建表达式和函数表达式。也就是说简单的表达式组合在一起，就构成了复杂的表达式，同学们可以这样理解哦。

## 2). 表达式的运算优先顺序

在进行表达式的转换过程中，必须了解各种运算的优先顺序，使转换后的表达式能满足数学公式的运算要求。

运算优先顺序为：

A. 括号→函数→乘方→乘、除→加、减→字符连接运算符→关系运算符→逻辑运算符

B. 如果同级的运算是按从左到右次序进行；多层括号由里向外。

例子 4：

```
<script>
  var x;
  var m=(x=(3>2)?9:2)*(3+(5-2)*4);
  alert("m is "+m);
</script>
```

结果为：



说明：

同学们知道运算顺序吗？根据优先级的规则我们知道会先计算  $3 > 2$ ，那么他是成立的，所以  $x$  的值变成了 9，之后式子就变成了  $m = 9 * (3 + 5 - 2) * 4$ ；到这一步同学们就知道结果了，是不是很简单呢。

同学们学习完表达式了，我们快去做练习题，看看自己的有没有掌握这些知识吧。

更多学习资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：





## 练习题

1. js 中有哪几种常见的数据类型。
2. 定义一个有 1, 2, 3, 4, “分针” 等元素组成的数组元。
3. 定义一个 person 的对象，其身高为：172，年龄为 22，名字为：fenzhen;
4. 再控制台输出上面定义的 person 对象的身高，年龄，名字;
5. 把上面的 person 的身高变成 180，年龄变成 25。
6. 再控制台输出下面变量的类型：a=1, b="fenzhen", c=true, person。
7. 计算下面表达式的值。

```
a = 12;
b = 24
c = "fenzhen"
r1 = ( a+b) + (b-a)*a
r2 = b/a*2+b%3;
r3 = a++
r4 = +a;
r5= a--;
r6 = *a;
r7 = (a>b) && (a*12 >b)
r8 = (a>b) || (a*12 >b)
r9 = ! (a>b) && (a*2>=b)?a:b;
```

# 函数

## 函数的定义

### 1. 怎么定义一个函数？

语法：

```
function 函数名(参数)
{
    执行代码
}
```

说明：

- 1). function 是定义函数的关键字，必不可少。
- 2). 函数名是函数的名字。自定义的。函数命名遵循命名规则，具体见下一小节。
- 3). 在调用函数时，您可以向其传递值，这些值被称为参数，参数是可以为空的。
- 4). 在函数定义处的参数我们称为形参，在我们调用函数时传递给函数的实际参数我们称为实参。形参与实参必须以一致的顺序出现。第一个变量值就是第一个被传递的参数的给定的值，以此类推

执行代码是在调用该函数的时候，所做的操作，也称为函数体。

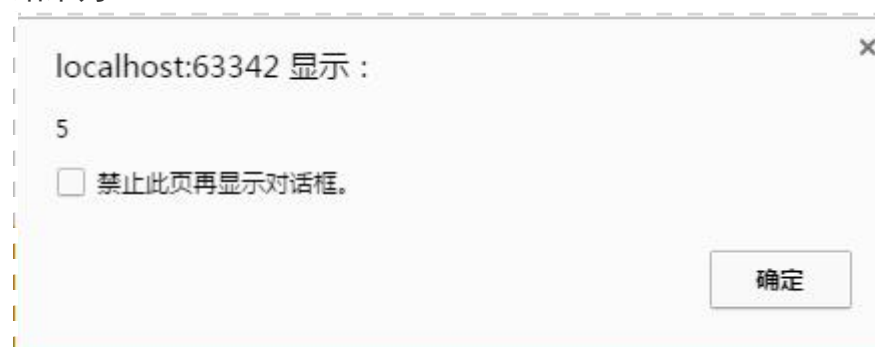
相信有了刚刚的说明，同学们已经大概知道怎么定义一个函数了吧？下面我们来看例子吧。

例子 1：

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
</head>
<body>
<button onclick="add(1, 4)">Try B</button>
</body>
<script>
    function add(a, b) {
        alert(a+b);
    }
</script>
</html>
```

结果为：



总结：按语法定义函数，直接通过函数名调用对应函数。

## 2. 函数名

函数名就是函数的名字。

例子 2：定义一个函数名为 `function_name` 的函数, 没有参数。

```
function function_name()
{
    alert("Hello World!");
}
```

函数命名规则：

- 1). 区分大小写。
- 2). 允许包含字母、数字、下划线、美元符号（\$），但是第一个字符不能是数字。
- 3). 不允许使用其他字符以及关键字和保留字命名。

### 3. 参数（形参）

函数参数就是在调用函数时，你向其传递的值，参数是可以为空的。

例子 3: 定义一个函数，参数为 a, b

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <button onclick="alert(add(1,4))">Try B</button>
</body>
<script>
  function add(a,b) {
    return a+b;
  }
</script>
</html>
```

说明：

1). 在函数定义处的参数我们称为**形参**，在我们调用函数时传递给函数的实际参数我们称为**实参**。x 形参与实参必须以一致的顺序出现。第一个变量值就是第一个被传递的参数的给定的值，以此类推

执行代码是在调用该函数的时候。

2). 参数既可以是变量也可以是常量。常量就是值不会变的量。但是参数为常量就不能发挥函数的灵活性，没有太大的实际意义。

## 函数的调用

函数的调用就是在我们定义一个函数之后，我们应该如何使用这个函数。

语法：

函数名（实参）

例子 1：调用函数名为 myfunction 的函数，实际参数为 name, sex.

```
myfunction(name, sex);
```

例子 2：不同的参数实现同样的效果。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
</head>
<body>
  <button onclick=" myFunction('小明', '男') ">实参</button>
  <button onclick="myFunction(Name, Sex) ">已赋值变量</button>
  <button onclick="myFunction(NAME, SEX) ">已赋值常量</button>
</body>
<script>
  var Name = "小明";
  var Sex = "男";
  const NAME = "小明";
  const SEX = "男";
  function myFunction(name, sex) {
    alert("name is "+name+" , "+"sex is "+sex);
  }
</script>
</html>
```

说明：

- 1). 在我们定义函数时函数的参数称为**形参**，在我们调用函数时传递给函数的实际参数我们称为**实参**，值是唯一确定的，不可更改的量叫**常量**。
- 2). 在函数调用时，形参与实参必须一一对应。
- 3). 实参可以是已经赋值的变量，也可以是赋值一个常量，就像上面的例子。

## 函数的返回值

当我们希望函数返回一个值到调用它的地方的时候，我们怎么办呢？

我们可以通过 **return** 来实现哦。

例子 1：返回变量 name 的值。

```
function myFunction()
{
```

```
var name="小明";  
return name;  
}
```

说明：

1). 例子 1 返回的值是小明。

在使用 return 语句时，函数不会再执行 return 后面的语句，什么意思呢？我们看看后面的例子 2 就知道啦。

例子 2：返回变量 name 的值

```
function myFunction()  
{  
    var name="小明";  
    return name;  
    name="小花";  
}
```

说明：

1). 例子 2 返回的值还是小明。

2). 例子 1 与例子 2 的返回值是一样的，因为函数不会再执行 return 之后的语句。

## 函数的拆分

函数的拆分：是逻辑意义上的拆分。不是说非要将函数拆成多少个，而是为了方便使用而根据实际情况来做具体处理的。

什么情况下会用到函数拆分呢？

比如：

1). 函数体里面的代码很长的时候，可以按照不同的模块或者功能将代码拆分为多个功能较小的函数。

2). 函数里面某一段代码使用的次数特别多，而其他的并不是很多的时候，就可以将他们拆分掉。这样可以提高效率。

例子 1: edit 函数有添加、删除、保存功能。

```
function edit() {  
    实现删除功能的代码  
    实现保存功能的代码  
    实现添加功能的代码  
}
```

说明:

- 1). 由于我们现在正在学习基础知识, 我们这里就不写出具体的实现代码, 而是用文字代表。主要理解函数拆分的意义就好。
- 2). 函数 edit 是由 3 个功能组成的一个大的函数, 代码量会很大。所以我们可以直接将他拆分为删除函数, 保存函数, 添加函数。这样就实现了函数的拆分。

例子 2:

```
function count() {  
    var m=3;  
    var n=6;  
    var a =4;  
    var b=5;  
    var result_ab=a+b;  
    var result_mn=m+n;  
    var result=result_ab+result_mn;  
    return result;  
}
```

说明:

可以看到上例中有很多加法计算, 如果我们能将该函数进行拆分, 将最常用的加法计算作为一个单独的函数, 就会大大提高效率。

代码为:

```
function add(A,B) {  
    return A+B;  
}  
function count() {  
    var m=3;  
    var n=6;  
    var a =4;  
    var b=5;  
    var result_ab=add(a,b);  
    var result_mn=add(m,n);  
    var result=add(result_ab,result_mn);  
    return result;  
}
```

**说明：**

由于举的例子很简单，所以不能明显看出拆分的好处，但是当重复操作变的复杂的时候，代码的缩减量就会大大增加，而且效率也会变高。

**总结：**

- 1). 函数拆分常用在函数体代码太多，函数中某一代码多次使用的情况下。
- 2). 函数拆分是视具体情况而定的，根据具体情况、要求不同，拆分的方式也会不同。

## 变量的作用域

### 1. 什么是变量作用域？

变量的作用域是指**变量有效性的范围**，与变量定义的位置密切相关，作用域是从空间这个角度来描述变量的，也可以理解为**可见性**。在某个范围内变量是可见的，也就是可用的。按照作用域的不同，变量可分为局部变量和全局变量。

在函数中使用 var 关键字进行显式声明的变量是作为**局部变量**，而没有用 var 关键字，使用直接赋值方式声明的是**全局变量**。

**说明：**

当我们使用访问一个没有声明的变量时，JS 会报错。而当我们给一个没有声明的变量赋值时，JS 不会报错，相反它会认为我们是要隐式声明一个全局变量，这一点一定要注意。

### 2. 全局变量

#### 1). 定义全局变量的几种方式

全局变量的定义方式有很多种，下面我们将结合例子给同学们说明一下全局变量的定义方式。



例子 1： 在 js 的 function 外定义一个变量。

```
<script>
var Sex ="男";
function count() {
alert(" sex is "+Sex);
}
</script>
```

说明：

这里 Sex 就是全局变量

例子 2： 不使用 var, 直接给变量赋值，隐式的声明了全局变量 Name。

```
<script>
Name = "小明";
var Sex ="男";
function count() {
alert("name is "+Name+" sex is "+Sex);
}
</script>
```

说明：

这里 Name, Sex 都是全局变量。

例子 3： 使用 window. 变量名定义变量，也是全局变量。

```
Name = "小明";
var Sex ="男";
window. old="19";
function count() {
alert("name is "+Name+" sex is "+Sex+"old is"+old);
}
```

说明：

这里 Name, Sex, old 都是全局变量。

总结：

- 1). 全局变量可以减少变量的个数，减少由于实际参数和形式参数的数据传递带来的时间消耗
- 2). 全局变量保存在静态存储区，程序开始运行时为其分配内存，程序结束释放该内存。与局部变量的动态分配、动态释放相比，生存期比较长，因此过多的全局变量会占用较多的内存单元

- 3). 全局变量破坏了函数的封装性能。函数象一个黑匣子，一般是通过函数参数和返回值进行输入输出，函数内部实现相对独立。但函数中如果使用了全局变量，那么函数体内的语句就可以绕过函数参数和返回值进行存取，这种情况破坏了函数的独立性，使函数对全局变量产生依赖。同时，也降低了该函数的可移植性。
- 4). 全局变量使函数的代码可读性降低。由于多个函数都可能使用全局变量，函数执行时全局变量的值可能随时发生变化，对于程序的查错和调试都非常不利。因此，如果不是万不得已，最好不要使用全局变量。

### 3. 局部变量

**局部变量：**就是声明在函数体中的变量，并且只能在当前函数体内访问。

例子 4：声明两个局部变量，下例中 a, b 就是局部变量。

```
function test() {  
    a = 30;  
    var b = 20;  
}
```

**说明：**

- 1). 当局部变量与全局变量同名的时候，全局变量会被局部变量覆盖。也就是说函数在使用该变量的时候会以局部变量覆盖全局变量，也就是只有局部变量会起效果。在此定义域中赋值等操作时都不会使用到全局变量。
- 2). 在 main 函数或其他函数里定义了局部变量，同时同名的全局变量同时也存在，当调用一个使用了该变量的函数（该变量在此函数中无定义）时，函数会使用全局变量。而不是 main 或其他函数中定义的同名变量

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



# 内部函数与闭包

## 1. 内部函数

内部函数又叫 js 内置函数，是浏览器内核自带的，不用引入任何函数库就可以直接使用的函数。

下面会讲一下所有的内部函数，但是同学们不用担心这么多的函数，怎么记住。这里完全不需要死记硬背哦，你只用*浏览几次，大约知道他的功能*，之后要用的时候再去详细研究他的使用与功能。

javascript 内置函数一共可分为五类：

### 1). 常规函数

- (1) alert 函数：显示一个警告对话框，包括一个 OK 按钮。
- (2) confirm 函数：显示一个确认对话框，包括 OK、Cancel 按钮。
- (3) escape 函数：将字符转换成 Unicode 码。
- (4) eval 函数：计算表达式的结果。
- (5) isNaN 函数：测试是(true)否(false)不是一个数字。
- (6) parseFloat 函数：将字符串转换成浮点数字形式。
- (7) parseInt 函数：将字符串转换成整数数字形式(可指定几进制)。
- (8) prompt 函数：显示一个输入对话框，提示等待用户输入。

### 2). 数组函数

(1) join 函数：转换并连接数组中的所有元素为一个字符串。(2) length 函数：返回数组的长度。(3) reverse 函数：将数组元素顺序颠倒。(4) sort 函数：将数组元素重新排序。

### 3). 日期函数

(1) getDate 函数：返回日期的“日”部分，值为 1~31 (2) getDay 函数：返回星期几，值为 0~6，其中 0 表示星期日，1 表示星期一，...，6 表示星期六 (3) getHours 函数：返回日期的“小时”部分，值为 0~23。(4) getMinutes 函数：返回日期的“分钟”部分，值为 0~59。见上例。(5) getMonth 函数：返回日期的“月”部分，值为 0~11。其中 0 表示 1 月，2 表示 3 月，...，11 表示 12 月。见前面的例子。(6) getSeconds 函数：返回日期的“秒”部分，值为 0~59。见前面的例子。(7) getTime 函数：返回系统时间。(8) getTimezoneOffset 函数：返回此地区的时差(当地时间与 GMT 格林威治标准时间的地区时差)，单位为分钟。(9) getYear 函数：返回日期的“年”部分。返回值以 1900 年为基数，例如 1999 年为 99。(10) parse 函数：返回从 1970 年 1 月 1 日零时整算起的毫秒数(当地时间)。(11) setDate 函数：设定日期的“日”部分，值为 0~31。(12) setHours 函数：设定日期的“小时”部分，值为 0~23。(13) setMinutes 函数：设定日期的“分钟”部分，值为 0~59。(14) setMonth 函数：设定日期的“月”部分，值为 0~11。其中 0 表示 1 月，...，11 表示 12 月。(15) setSeconds 函数：设定日期的“秒”部分，值为 0~59。(16) setTime 函数：设定时间。时间数值为 1970 年 1 月 1 日零时整算起的毫秒数。(17) setYear 函数：设定日期的“年”部分。(18) toGMTString 函数：转换日期成为字符串，为 GMT 格林威治标准时间。(19) setLocaleString 函数：转换日期成为字符串，为当地时间。(20) UTC 函数：返回从 1970 年 1 月 1 日零时整算起的毫秒数，以 GMT 格林威治标准时间计算。

### 4). 数学函数

(1) abs 函数：即 Math.abs(以下同)，返回一个数字的绝对值。(2) acos 函数：返回一个数字的反余弦值，结果为  $0 \sim \pi$  弧度(radians)。(3) asin 函数：返回一个数字的正弦值，结果为  $-\pi/2 \sim \pi/2$  弧度。(4) atan 函数：返回一个数字的正切值，结果为  $-\pi/2 \sim \pi/2$  弧度。(5) atan2 函数：返回一个

坐标的极坐标角度值。(6)ceil 函数: 返回一个数字的最小整数值(大于或等于)。(7)cos 函数: 返回一个数字的余弦值, 结果为-1~1。(8)exp 函数: 返回 e(自然对数)的乘方值。(9)floor 函数: 返回一个数字的最大整数值(小于或等于)。(10)log 函数: 自然对数函数, 返回一个数字的自然对数(e)值。(11)max 函数: 返回两个数的最大值。(12)min 函数: 返回两个数的最小值。(13)pow 函数: 返回一个数字的乘方值。(14)random 函数: 返回一个 0~1 的随机数值。(15)round 函数: 返回一个数字的四舍五入值, 类型是整数。(16)sin 函数: 返回一个数字的正弦值, 结果为-1~1。(17)sqrt 函数: 返回一个数字的平方根值。(18)tan 函数: 返回一个数字的正切值。

## 5). 字符串函数

(1)anchor 函数: 产生一个链接点(anchor)以作超级链接用。anchor 函数设定<A NAME=...>的链接点的名称, 另一个函数 link 设定<A HREF=...>的 URL 地址。

(2)big 函数: 将字体加到一号, 与<BIG>...</BIG>标签结果相同。(3)blink 函数: 使字符串闪烁, 与<BLINK>...</BLINK>标签结果相同。(4)bold 函数: 使字体加粗, 与<B>...</B>标签结果相同。(5)charAt 函数: 返回字符串中指定的某个字符。(6)fixed 函数: 将字体设定为固定宽度字体, 与<TT>...</TT>标签结果相同。(7)fontcolor 函数: 设定字体颜色, 与<FONT COLOR=color>标签结果相同。(8)fontsize 函数: 设定字体大小, 与<FONT SIZE=n>标签结果相同。(9)indexOf 函数: 返回字符串中第一个查找到的下标 index, 从左边开始查找。(10)italics 函数: 使字体成为斜体字, 与<I>...</I>标签结果相同。(11)lastIndexOf 函数: 返回字符串中第一个查找到的下标 index, 从右边开始查找。(12)length 函数: 返回字符串的长度。(不用带括号)(13)link 函数: 产生一个超级链接, 相当于设定<A HREF=...>的 URL 地址。(14)small 函数: 将字体减小一号, 与<SMALL>...</SMALL>标签结果相同。(15)strike 函数: 在文本的中间加一条横线, 与<STRIKE>...</STRIKE>标签结果相同。(16)sub 函数: 显示字符串为下标字(subscript)。(17)substring 函数: 返回字符串中指定的几个字符。(18)sup 函数: 显示字符串为上标字(superscript)。(19)toLowerCase 函数: 将字符串转换为小写。(20)toUpperCase 函数: 将字符串转换为大写。

下面我们学习的知识就很重点啦。同学们，我们知道局部变量在外部是不可见的，访问不到，那么我们要访问局部变量应该怎么办呢？可以使用闭包哦，学习下面的内容你就知道了。

## 2. 闭包

闭包就是能够读取其他函数内部变量的函数。

例子 1：什么东西是闭包？例子中函数 f2 就是一个闭包。

```
function f1() {  
    var n=999;  
    nAdd=function() {n+=1}  
    function f2() {  
        alert(n);  
    }  
    return f2;  
}
```

说明：

这里的函数 f1 内部有一个局部变量，在函数 f1 之外是不可见的，但是通过 f2 函数，我们可以弹出局部变量的值，这也是闭包最大的用处。

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



# 函数作为参数与回调函数

## 1. 函数作为参数

如题我们可以知道是将一个函数作为另一个函数的参数来使用，那么使用的方式有哪些呢？

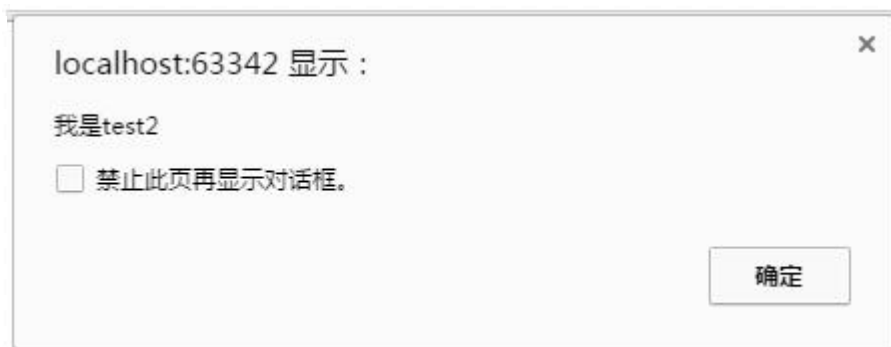
例子 1：直接将函数名作为参数，直接调用。

```
function test1(Func)
{
    Func();
}
function test2()
{
    alert("我是 test2");
}
test1(test2);
```

说明：

这里 test1 有一个参数，这个参数是一个函数。再定义一个函数 test2，这里直接将他作为 test1 的参数。

结果为：



例子 2：定义一个调用以带参数的函数为参数的函数。

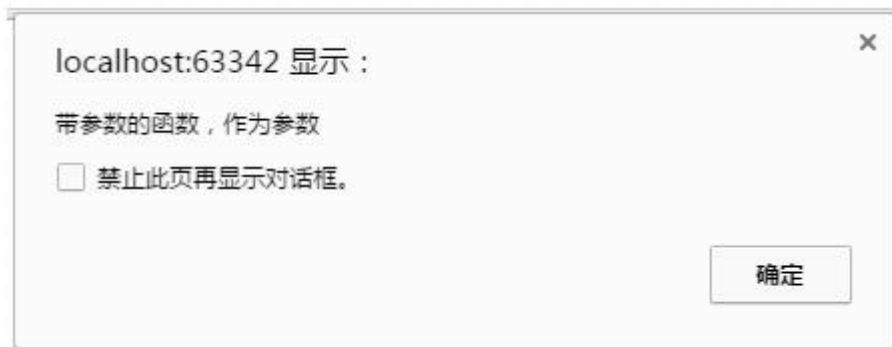
```
function test1(Func)
{
    Func(mydata);
}
function test2(data)
{
    alert(data);
}
```

```
test1(test2("带参数的函数，作为参数"));
```

说明：

这和例子 1 的区别就是作为参数的函数自身也带有参数，所以 test1 在使用 test2 的时候，给了他参数。

结果为：



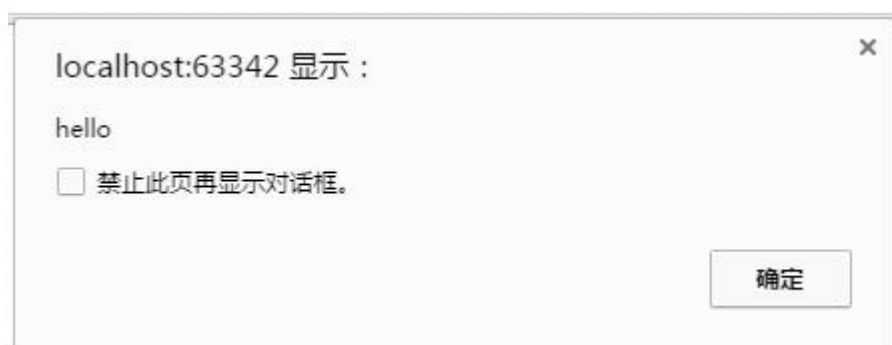
例子 3： 作为参数的函数不再单独定义

```
function test1(Func)
{
    Func("hello");
}
test1(function(data)
{
    alert(data);
}));
```

说明：

作为参数的函数不再单独定义，而是在使用的时候，自己定义。这种比较适合作为参数的函数体比较简单的时候。

结果为：



## 2. 回调函数



我们知道 js 代码是*顺序执行的*，但是有时候我们需要等到一个操作结束之后再进行下一个操作，这时候就需要用到回调函数。

什么是回调函数？在认识 js 回调函数之前，首先要了解“**函数也是一种数据类型**”，它也可以像变量一样使用。就像前面学习的函数作为参数。

**回调函数**：简单通俗点就是当有 a 和 b 两个函数，当 a 作为参数传给 b，并在 b 中执行，这时 a 就是一个回调(callback)函数，如果 a 是一个匿名函数，则为匿名回调函数。

回调函数是一个作为变量传递给另外一个函数的函数，它在主体函数执行完之后执行。

之后等我们学习更加深入，就会更加清楚回调函数，现在的我们不清楚也没关系，知道回调函数的概念就行啦。

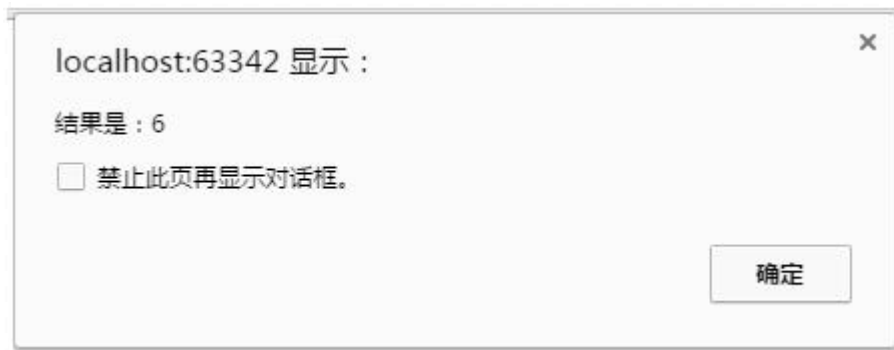
## 递归函数

递归就是函数自己调用自己

例子 1：如下形式的函数，自己调用自己。这里实现的是一个简单的阶乘函数。

```
function f(x) {  
    if (x === 1) {  
        return 1;  
    } else {  
        return x * f(x - 1);  
    }  
}  
alert("结果是: "+f(3));
```

结果为：



说明：

- 1). 我们可以看到在 f 函数的内部又调用了 f 函数，但是参数是不一样的。参数会一直减小，直到为 1 就不再返回函数，而是返回数字 1 了。
- 2). 试想一下，一直返回的都是函数自身，我们这个函数会有结束的一天吗？所以我们需要知道使用递归函数一定要注意，处理不当就会进入死循环。

## 练习题

定义一个函数计算给定周长的圆的面积，比如给定圆周长为 20，计算圆的面积。

定义 2 个函数，计算两个 a, b 两个变量的和，其中一个函数，将所求的和加 1，另一个函数一个减 1；并拆分优化这个函数。

计算  $1+2+3+***+100$  的值。

定义一个 person 的函数，有一个局部变量 name，如果我要在函数外调用和修改 name 变量，要怎么实现呢。

输出以下代码块输出的值是什么，为什么？

```
var i = 0;
console.log(i, 1);
function change() {
  console.log(i, 2);
  i++;
}
console.log(i, 3);
change();
```

```
console.log(i, 4);  
//代码二  
var i = 0;  
console.log(i, 21);  
function change() {  
    var i = 10;  
    console.log(i, 22);  
    i++;  
}  
console.log(i, 23);  
change();  
console.log(i, 24);
```

## 函数的作用

问：什么是函数？

答：函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

说明：什么是事件驱动呢？通俗讲就是你点什么按钮（即产生什么事件），电脑执行什么操作（即调用什么函数），下面我们看例子吧。

例子 1： 点击按钮，弹出 Hello World!。

```
<!DOCTYPE html>  
<html>  
<head>  
<script>  
function myFunction()  
{  
    alert("Hello World!");  
}  
</script>  
</head>  
<body>  
<button onclick="alert('Hello World!');">按钮 A</button>  
<button onclick="alert('Hello World!');">按钮 B</button>  
</body>
```

```
</html>
```

例子 2: 点击按钮, 弹出 Hello World!。

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>
<body>
<button onclick="myFunction()">按钮 A</button>
<button onclick="myFunction()">按钮 B</button>
</body>
</html>
```

#### 说明:

例子 1 与例子 2 实现的效果是一样的, 都是点击按钮后, 弹出 Hello World!

例子 1 的代码与例子 2 的区别在于: 例子 1 使用的函数内部的代码, 例子 2 使用的函数名字。当函数内部代码很多的时候, 例子 1 的代码数量将远远多于例子 2。对于说明中的函数名, 函数内部代码等概念不懂的同学不要慌, 后面我们会讲解具体的概念。这里只是为了直观表现函数的作用而举的例子。现在不用懂具体代码的意义。

#### 总结:

函数是用来帮助我们封装、调用代码的最方便的工具。

函数可以重复调用一段代码, 减少代码量, 方便使用, 也会利用实现模块化。

更多学习全套视频资料请关注微信公众号“**前端大学**” 免费自动下载！

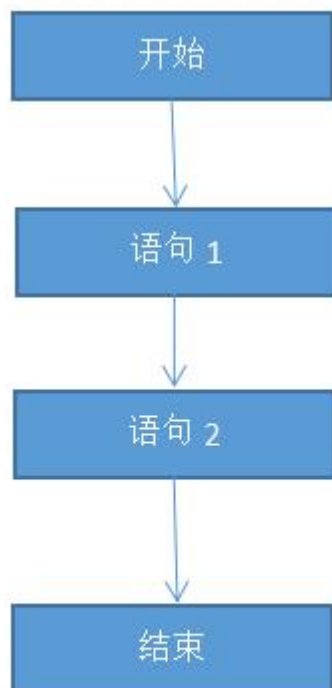
打开微信扫一扫：



## 流程控制

## 顺序结构

顺序结构是 JavaScript 中最基本的结构，说白了就是按照从上到下、从左到右的顺序执行。



顺序结构流程图

一般而言，在 JavaScript 中，程序总体是按照顺序结构执行的，但是在顺序结构可以包含选择结构和循环结构。

举例：实现内容的替换

代码如下：

```
<body>
  <h1>My Web Page</h1>
  <p id="demo">A Paragraph.</p>
  <div id="myDIV">A DIV.</div>
  <script>
    document.getElementById("demo").innerHTML="Hello World";
    document.getElementById("myDIV").innerHTML="How are you?";
  </script>
</body>
```

效果：



上面的例子首先执行的是<h1>标签，显示的是"My Web Page"字样，接着执行的是<p>标签里的语句显示的是"A Paragraph. "；第三步是执行<div>标签里的语句显示"myDIV"；第四步执行<script>标签里的语句，首先把"A Paragraph. "替换成"Hello World",然后把"myDIV"替换成"How are you?"。整个流程是一句跟着一句执行的这就是顺序结构。

## 选择结构

### 1. if 语句

只有当指定条件为 true 时，该语句才会执行代码。

语法：

```
if (condition)
{
    当条件为 true 时执行的代码
}
```

注意：

请使用小写的 if。使用大写字母（IF）会生成 JavaScript 错误！

举例：实现当时间小于 20:00 是，生成问候 "Good day"

代码如下：

```
<body>
<p>如果时间早于 20:00，会获得问候 "Good day"。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x="";
    var time=new Date().getHours();
    if (time<20) {
        x="Good day";
    }
    document.getElementById("demo").innerHTML=x;
}
</script>
</body>
```

效果：

如果时间早于 20:00，会获得问候 "Good day"。

[点击这里](#)

Good day

**注意：**

在这个语法中，没有 `..else..`。您已经告诉浏览器只有在指定条件为 `true` 时才执行代码。

## 2. If...else 语句

使用 `if...else` 语句在条件为 `true` 时执行代码，在条件为 `false` 时执行其他代码

语法：

```
if (condition)
{
    当条件为 true 时执行的代码
}
else
{
    当条件不为 true 时执行的代码
}
```

举例：实现当时间小于 20:00 时，生成问候 "Good day"，否则生成问候 "Good evening"。

代码如下：

```
<body>
<p>点击这个按钮，获得基于时间的问候。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x="";
    var time=new Date().getHours();
    if (time<20) {
        x="Good day";
    }
    else{
        x="Good evening";
    }
}
```



```
document.getElementById("demo").innerHTML=x;
}
</script>
</body>
```

效果:



### 3. If...else if...else 语句

使用 if...else if...else 语句来选择多个代码块之一来执行。

语法:

```
if (condition1)
{
    当条件 1 为 true 时执行的代码
}
else if (condition2)
{
    当条件 2 为 true 时执行的代码
}
else
{
    当条件 1 和 条件 2 都不为 true 时执行的代码
}
```

举例：实现如果时间小于 10:00，则生成问候 "Good morning"，如果时间大于 10:00 小于 20:00，则生成问候 "Good day"，否则生成问候 "Good evening"

代码如下：

```
<body>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
    document.write("<b>早上好</b>");
}
else if (time>=10 && time<16)
{
```

```

    document.write("<b>今天好</b>");
}
else
{
    document.write("<b>晚上好!</b>");
}
</script>
<p>
这个例子演示了 if..else if...else 语句。
</p>
</body>

```

效果：

早上好

这个例子演示了 if..else if...else 语句。

#### 4. 多重选择语句（switch）

switch 语句用于基于不同的条件来执行不同的动作。

语法：

```

switch(表达式) {
    case 值 1:
        执行代码块 1
        break;
    case 值 2:
        执行代码块 2
        break;
    ...
    case 值 n:
        执行代码块 n
        break;
    default:
        与 case 值 1 、 case 值 2...case 值 n 不同时执行的代码
}

```

说明：

switch 必须赋初始值，值与每个 case 值匹配。

满足条件则执行该 case 后的所有语句，并用 break 语句来阻止运行下一个 case。

如所有 case 值都不匹配，执行 **default** 后的语句。

举例：通过 switch 来判断该输出星期几代码如下：

```
<script>
  var week=3; //设置一个代表星期的变量
  switch(week) {
    case 1:
      document.write("今天是星期一")
      break;
    case 2:
      document.write("今天是星期二")
      break;
    case 3:
      document.write("今天是星期三")
      break;
    case 4:
      document.write("今天是星期四")
      break;
    case 5:
      document.write("今天是星期五")
      break;
    case 6:
      document.write("今天是星期六")
      break;
    default:
      document.write("今天是星期天")
  }
</script>
```

效果：

今天是星期三

**说明：**根据 week 的值来执行不同的代码，例子中的 week 值为 3，所以执行 case 3 后面的代码。 **提示：**case 中的条件可以是数字也可以是字符串，但必须是确定的值。

**总结：**switch 语句用于基于不同的条件来执行不同的动作。但条件必须是确定的值。

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



## 循环结构

### 1. for 循环

for 循环语句 for 循环语句是在创建循环时常用到的语句。for 循环的循环结构：

```
for (初始化变量; 判断条件; 更新迭代) {  
    循环语句  
}
```

说明：

- 1). 初始化变量在整个循环的最开始只执行一次。
- 2). for 循环是先执行初始化变量的代码，然后判断条件是否返回 true，如果是则执行循环语句，如果判断条件为 FALSE 则不执行循环。
- 3). 执行完循环后，执行更新迭代的代码，然后再判断循环条件是否为 true，如果是就再次执行循环语句，直到循环条件返回值 false。
- 4). for 循环的 3 个条件都是可以省略的，如果没有退出循环的判断条件，就必须使用 break 语句退出循环，否则就是死循环。

例子 1：用 for 循环来输出 0 到 100 的累加值。代码如下：

```
<script>  
    for (var i = 0, sum=0; i<=100; i++) {  
        sum+=i;  
    }  
    document.write("0 到 100 的累加值是: "sum);
```

```
</script>
```

结果：

0到100的累加值是：5050

说明：

初始化变量  $i=0$ ,  $sum=0$ , 判断  $i$  是否小于等于 100, 如果是, 执行语句  $sum=sum+i$ , 然后执行迭代更新语句  $i++$ , 再次根据循环条件判断是否继续执行。

for 循环中初始化的变量可以在 for 循环之外使用。 例子 2: 通过两个 for 循环来输出九九乘法表代码如下：

```
<script>
    for (var i = 1; i<10; i++) {
        for (var j=1; j<=i; j++) {
            document.write(i+" * "+j+" = "+i*j+"&nbsp;&nbsp;&nbsp;");
        }
        document.write("<br/>");
    }
</script>
```

结果： 0

```
1 * 1 = 1
2 * 1 = 2  2 * 2 = 4
3 * 1 = 3  3 * 2 = 6  3 * 3 = 9
4 * 1 = 4  4 * 2 = 8  4 * 3 = 12  4 * 4 = 16
5 * 1 = 5  5 * 2 = 10  5 * 3 = 15  5 * 4 = 20  5 * 5 = 25
6 * 1 = 6  6 * 2 = 12  6 * 3 = 18  6 * 4 = 24  6 * 5 = 30  6 * 6 = 36
7 * 1 = 7  7 * 2 = 14  7 * 3 = 21  7 * 4 = 28  7 * 5 = 35  7 * 6 = 42  7 * 7 = 49
8 * 1 = 8  8 * 2 = 16  8 * 3 = 24  8 * 4 = 32  8 * 5 = 40  8 * 6 = 48  8 * 7 = 56  8 * 8 = 64
9 * 1 = 9  9 * 2 = 18  9 * 3 = 27  9 * 4 = 36  9 * 5 = 45  9 * 6 = 54  9 * 7 = 63  9 * 8 = 72  9 * 9 = 81
```

在一个 for 循环中嵌套一个 for 循环。 总结：for 循环是最常用的循环，循环次数可以为零。

## 2. while 循环

While 循环会在指定条件为真时循环执行代码块。

只要指定条件为正，循环就可以一直执行代码。

语法：

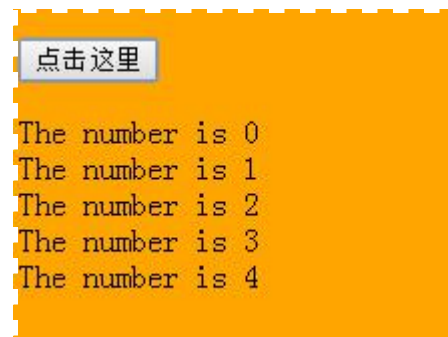
```
while (条件)
{
    需要执行的代码
}
```

举例：输出数字 1~5

代码如下：

```
function myFunction()
{
var x="", i=0;
while (i<5)
{
x=x + "The number is " + i + "<br>";
i++;
}
document.getElementById("demo").innerHTML+=x;
}
</script>
```

效果：



## 程序的继续与终止

### 1. break 语句

在之前的 switch 语句中，break 语句用于跳出 switchu()。break 语句用于跳出循环。

举例：用 break 语句来跳出 for 循环。代码如下：

```
<script>
  for (var i = 1; i<8; i++) {
    document.write(i+"<br/>");
    if(i==3){
      break;
    }
  }
}
```

```
</script>
```

效果：



说明：

执行循环，输出 i 的值，判断 i 的值是否等于 3，当 i 的值等于 3 的时候，执行语句 break，跳出循环，之后的循环也不执行。

## 2. continue 语句

continue 用于跳过循环中的一个迭代。continue 语句跳出循环后，会继续执行该循环之后的代码（如果有的话） 举例：通过 continue 来跳过当前循环，执行下一个循环。代码如下：

```
<script>
    for (var i = 1; i<=5; i++) {
        if(i==3){
            continue;
        }
        document.write(i+"<br/>");
    }
</script>
```

效果：



说明：

执行循环，输出 i 的值，判断 i 的值是否等于 3，当 i 的值等于 3 的时候，执行语句 continue。

如效果图所示，continue 只是跳过了当前循环，之后的循环继续执行。 **总结：**根据情况的不同，选择使用 break 还是 continue。

break 是跳出整个循环，不管当前循环之后还有没有代码都不再执行。

continue 是跳出当前循环，执行下一个循环。

## 练习题

1. 定义一个函数识别一个整数是奇数还是偶数（用两种方法）
2. 定义一个函数循环输出 1~20 的奇数（用两种方法）

## 字符串对象

### 定义字符串

1. 定义：

JavaScript 字符串用于存储和处理文本。

2. 字符串可以存储一系列字符，如 "Fen Zhen"。换句话说就是字符串是由多个字符组成的。

字符串可以是插入到引号中的任何字符。你可以使用单引号或双引号如：

```
var carname = "Fen Zhen";  
var carname = 'Fen Zhen';
```

3. 你可以使用索引位置来访问字符串中的每个字符如：

```
var character = carname[7];
```

4. 字符串的索引从 0 开始, 这意味着第一个字符索引值为 [0], 第二个为 [1], 以此类推。

你可以在字符串中使用引号，字符串中的引号不要与字符串的引号相同如：

```
var answer = "It's alright";  
var answer = "He is called 'fenzhen'";
```



```
var answer = 'He is called "fenzhen"';
```

5. 你也可以在字符串添加转义字符来使用引号如:

```
<p id="demo"></p>
<script>
  var x = 'It\'s alright';
  var y = "He is called \"Johnny\"";
  document.getElementById("demo").innerHTML = x + "<br>" + y;
</script>
```

效果:

```
It's alright
He is called "Johnny"
```

## 6. length 属性

定义:

length 属性可返回字符串中的字符数目

语法:

```
stringObject.length
```

举例: 实现返回字符串的长度

代码如下:

```
<script type="text/javascript">
  var txt="Hello World!"
  document.write(txt.length)
</script>
```

效果:

```
12
```

更多学习全套视频资料请关注微信公众号“**前端大学**” 免费自动下载！

打开微信扫一扫：



## 字符串操作

### 获取字符

#### 1. charAt() 方法

功能：返回字符串的第几个字符格式：

```
string.charAt(index)
```

注意：

index 必须是整数。0、1、2……。字符串的一个字符的下标是 0，第二个字符的下标是 1。之后的以次类推 举例：实现返回字符串的最后一个字符代码如下：

```
<p>字符串为：HELLO WORLD</p>
<p id="demo">单击按钮显示字符串的最后一个字符。</p>
<button onclick="myFunction()">点我</button>
<script>
  function myFunction() {
    var str="HELLO WORLD";
    var n=str.charAt(str.length-1);
    document.getElementById("demo").innerHTML='最后一个字符是：'+n;
  }
}
```

```
</script>
```

效果：

字符串为：HELLO WORLD

单击按钮显示字符串的最后一个字符。

点我

点击前的效果

字符串为：HELLO WORLD

最后一个字符是：D

点我

点击后的效果

## 2. 直接索引字符 在 javascript 中，字符串可以被当做数组来处理，所以我

们可以用数组下标的方式来访问单个字符。代码如下：

```
<script type="text/javascript">
  var str="hello world";
  console.log(str[0]); //输出 h
</script>
```

效果：



## 3. 两种方式的不同

1. 第一个区别是超出范围的返回值不同 使用 `string[index]` 的方式，对于超出 `index` 范围的，会返回 `undefined`。而使用 `charAt(index)` 的方式，对于超出范围的会返回一个空的字符串。2. 第二个区别，是兼容性问题 `string[index]` 的方式在 IE6~8 下会返回 `undefined`，也就是 IE6~8 不兼容此方法。而 `charAt(index)` 经测试，在 IE6~8 下也能够正常返回值。

总结

如果你不需要考虑 IE6~8 的话，就可以随便用了，至于性能，都是 JavaScript 的方法，差别微乎其微。如果还是苦逼的要考虑 IE6~8 的话，还是使用 `charAt()` 比较好，安全又放心。

## 查找字符串

### 1. Match() 方法

**功能：** `match()` 方法可在字符串内检索指定的值格式：

```
string.match(regex)
```

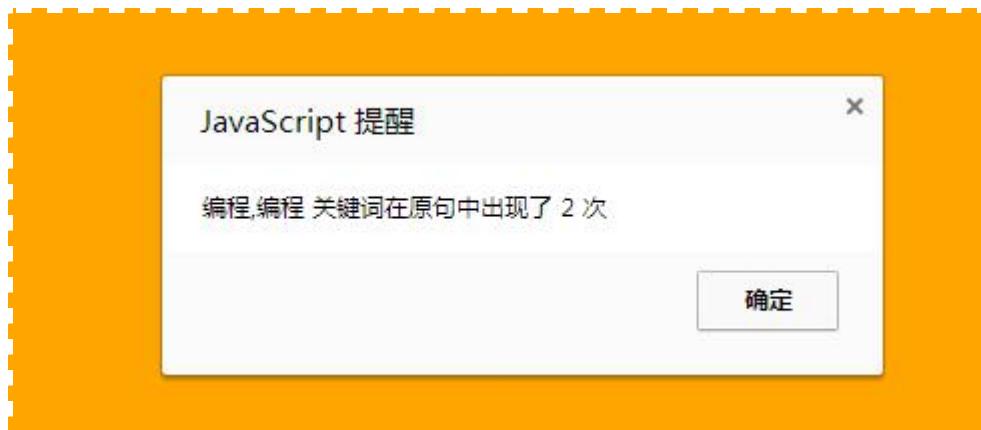
**说明：**

`match()` 方法将检索字符串 String Object，以找到一个或多个与 `regex` 匹配的文本。这个方法的行为在很大程度上有赖于 `regex` 是否具有标志。如果 `regex` 没有标志，那么 `match()` 方法就只能在 `stringObject` 中执行一次匹配。如果没有找到任何匹配的文本，`match()` 将返回 `null`。否则，它将返回一个数组，其中存放了与它找到的匹配文本有关的信息。`match` 也是在目标字符串对象中寻找与关键词匹配与否的一个方法，它的强大功能在于通过关键词的规则创建可以实现复杂搜寻功能，非常灵活。不建立规则前提下，`match` 可当作 `search` 来使用，语法也一样，不同的是，它返回的是关键词自身（若匹配）和 `null`（若不匹配）——这没有关系，如果只是为了检测匹配。显然地，这不是它存在于 javascript 世界中的理由，它定有不同于 `search` 的特色，即通过规则创建完成实现通盘匹配。

举例：实现输出查找的字符，出现几次输出几次字符，并输出总计次数代码如下：

```
<script>
var Str = "请问编程入门网是一个编程技能学习的网站吗？";
var ShowStr = Str.match(/编程/gi);
var Result = ShowStr + " 关键词在原句中出现了 " + ShowStr.length + " 次";
alert(Result);
</script>
```

效果：



## 2. Search() 方法

功能：

search() 方法用于检索字符串中指定的字符串格式：

```
string.search(searchvalue)
```

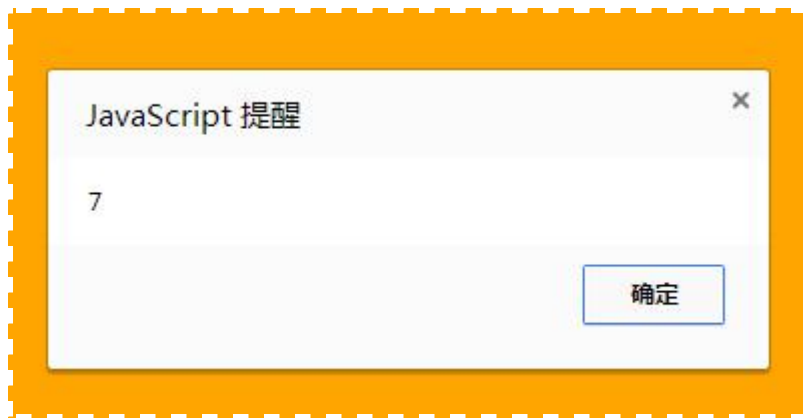
说明：

search 在一个字符串对象（string object）中查找关键词字符串（规范表达式，regular expression），若匹配（即在目标字符串中成功找到关键词）则返回关键词在目标字符串中第一次出现的位置序列，反之，如果不匹配，就返回-1。

举例：实现返回所查找的字符在字符串中第一次出现的位置代码如下：

```
<script>
window.onload = function() {
var MyStr = "前端大学是学习编程网站，一个学习编程的乐园！";
var Re = "编程";
var ShowStr = MyStr.search(Re);
alert(ShowStr);
}
</script>
```

效果：



我们看得出来，search 方法只关心有无匹配，一旦找到匹配，就提供返回值，并且立刻中断查找的执行。上例中的目标字符串（MyStr）有两个“编程”，后两个 search 并不关心，因为条件已经满足，它退出了自身的匹配查找工作。据此原理，当我们只需要验证有无匹配的操作时，用 search 既简单又高效。上例中，第二行为创建规则表达式“/编程/gi”，双反斜杠内的“编程”为搜寻关键词，其后的参数“gi”表示全部匹配（同等于“g”，若只用“i”，只匹配一次）。这样的规则用于 match 方法，将返回有规则的数组

### 3. indexOf() 方法

功能：

indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的位置。

如果没有找到匹配的字符串则返回 -1 格式：

```
string.indexOf(searchvalue, start)
```

说明：

1). 其中 star 是可选参数，规定在字符串中开始检索的位置。

它的合法取值是 0 到 stringObject.length - 1。如省略该参数，则将从字符串的首字符开始检索。

2). indexOf 是区分大小写的举例：实现在字符串的第六个字符开始查找所差字符在字符串中首次出现的位置代码如下：

```
<script>
function myFunction() {
    var str="Hello welcome world, welcome to the universe.";
    var n=str.indexOf("welcome", "7");
    document.getElementById("demo").innerHTML=n;
}
</script>
```

效果：

单击按钮来定位指定文本首次出现的位置。

点我

点击前

21

点我

点击后

## 替换字符串

### 1. `replace()`

功能：

用于在字符串中用一些字符替换另一些字符

该方法不会改变原始字符串。

格式：

```
string.replace(searchvalue, newvalue)
```

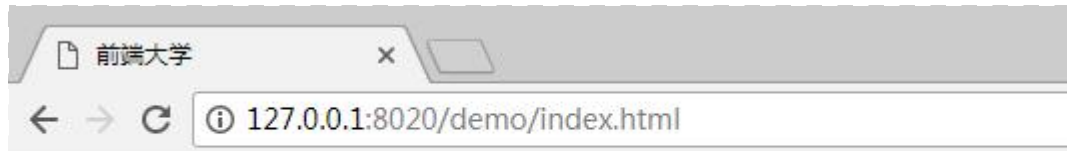
说明：

- 1). `searchvalue`: 规定子字符串或要替换的模式 的 `RegExp` 对象。
- 2). 如果该值是一个字符串，则将它作为要检索的直接量文本模式，而不是首先被转换为 `RegExp` 对象。
- 3). `newvalue` : 一个字符串值。规定了替换文本或生成替换文本的函数。 举例：实现局部字符串的替换代码如下：

```
<body>
<p>单击按钮将段落中“Microsoft”替换成“www.qianduandaxue.com”：</p>
<p id="demo">Visit Microsoft!</p>
<button onclick="myFunction()">点我</button>
<script>
function myFunction() {
```

```
var str=document.getElementById("demo").innerHTML;  
var n=str.replace("Microsoft","www.qianduandaxue.com");  
document.getElementById("demo").innerHTML=n;  
}  
</script>  
</body>
```

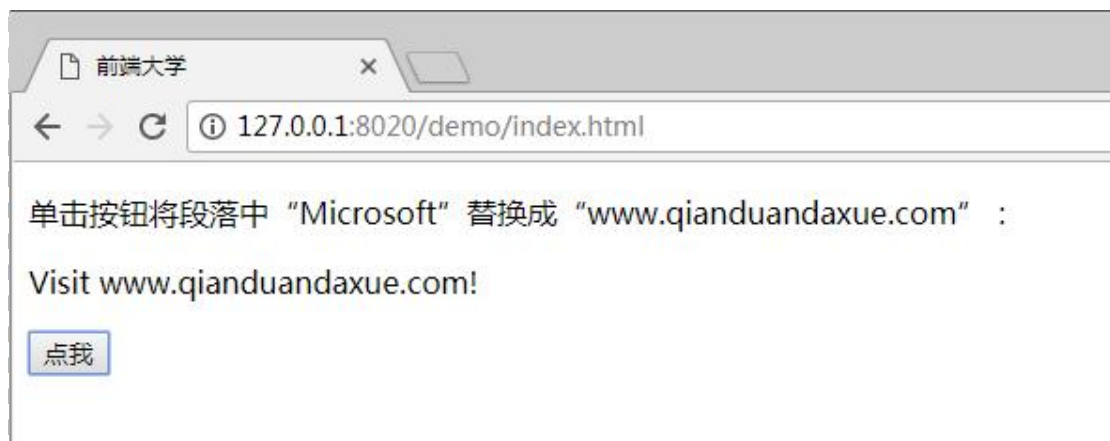
效果：



单击按钮将段落中“Microsoft”替换成“www.qianduandaxue.com”：

Visit Microsoft!

点我



## 2. 英文字母大小写的替换

### 1). toLowerCase()

功能：用于把字符串转换为小写

格式：

```
string.toLowerCase()
```

### 2). toUpperCase()

功能：用于把字符串转换为大写

格式：

```
string.toUpperCase()
```



举例：将字符串中的所有字符都换成小写或将字符串中所有字符都换成大写

```
<script>
var txt="Hello World!";
document.write(txt.toLowerCase() + "<br>");
document.write(txt.toUpperCase());
</script>
```

效果：

```
hello world!
HELLO WORLD!
```

## 字符串连接

### 1. concat() 连接字符串

功能：用于连接两个或多个字符串格式：

```
string2.concat.(string2)
```

提示：

该方法没有改变原有字符串，但是会返回连接两个或多个字符串新字符串

举例：实现 Hello 和 world! 两段字符串的连接

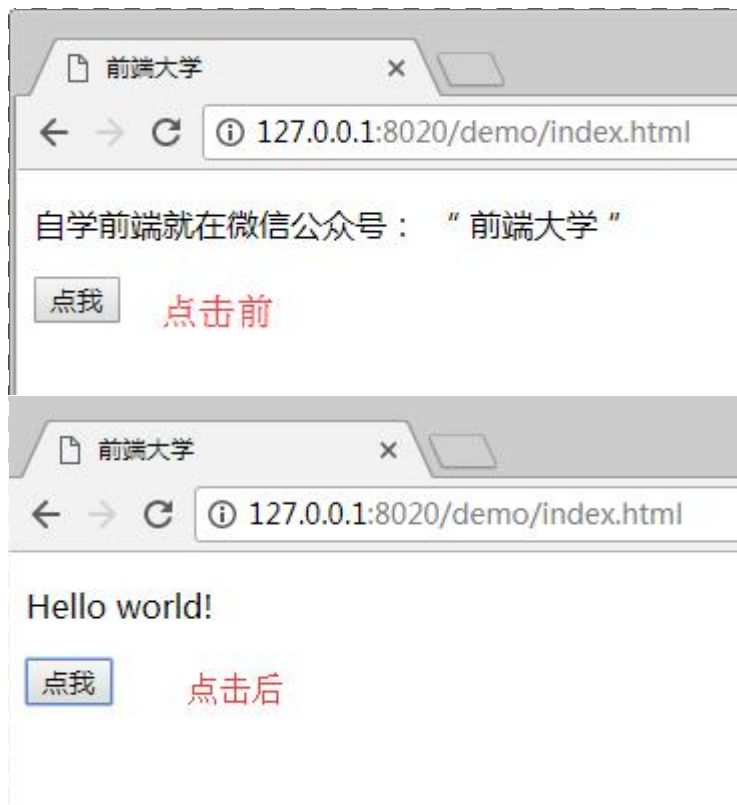
代码如下：

```
<body>

  <p id="demo">自学前端就在微信公众号： “ 前端大学 ” </p>

  <button onclick="myFunction()">点我</button>
  <script>
    function myFunction() {
      var txt1 = "Hello ";
      var txt2 = "world!";
      var n=txt1.concat(txt2);
      document.getElementById("demo").innerHTML=n;
    }
  </script>
</body>
```

效果：



## 2. “+” 连接字符串

这种方法是最便捷快速的, 如果只连接 100 个以下的字符串建议用这种方法最方便

举例：实现两个字符串的连接

代码如下：

```
<p id="demo">自学前端就在微信公众号： “ 前端大学 ” </p>
<button onclick="myFunction()">点我</button>
<script>
  function myFunction() {
    var txt1 = "Hello ";
    var txt2 = "world!";
    var n=txt1+txt2;
    document.getElementById("demo").innerHTML=n;
  }
</script>
```

效果：



## 截取字符

### 1. 定义

`substring()` 方法用于提取字符串中介于两个指定下标之间的字符

语法：

```
stringObject.substring(start, stop)
```

参数	描述
start	必需。一个非负的整数，规定要提取的子串的第一个字符在 <code>stringObject</code> 中的位置。
stop	可选。一个非负的整数，比要提取的子串的最后一个字符在 <code>stringObject</code> 中的位置多 1。如果省略该参数，那么返回的子串会一

直到字符串的结尾。
-----------

## 2. 返回值

一个新的字符串，该字符串值包含 *stringObject* 的一个子字符串，其内容是从 *start* 处到 *stop-1* 处的所有字符，其长度为 *stop* 减 *start*。

说明：

`substring()` 方法返回的子串包括 *start* 处的字符，但不包括 *stop* 处的字符。如果参数 *start* 与 *stop* 相等，那么该方法返回的就是一个空串（即长度为 0 的字符串）。如果 *start* 比 *stop* 大，那么该方法在提取子串之前会先交换这两个参数。

**重要事项：**


与 `slice()` 和 `substr` 方法不同的是，`substring()` 不接受负的参数

举例：使用 `substring()` 从字符串中提取一些字符

代码如下：

```
<script type="text/javascript">
  var str="Hello world!"
  document.write(str.substring(3))
</script>
```

效果：



lo world!

举例 2：带两个参数的字符截取

代码如下：

```
<script type="text/javascript">
  var str="Hello world!"
  document.write(str.substring(3,7))
</script>
```

效果：



lo w

## 练习题

1. 写一个函数，将两个字符串链接，并且截取链接后的字符串的最后 2 个字符；
2. 把字符串"Fenzhen"转化为小写，把“haohaoxuexi”转化为大写；
3. 写一个函数，获取字符串制定索引的值，
4. 写一个函数，把给定字符串中的“分”替换为“fen”
5. 写一个函数，查找给定字符串中“分”有几个，并且每一个的索引值是多少

## 数组对象

## 数值对象

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



# JS 数据类型

js 中的数据类型总体来说分为两种，他们分别是：

## 1. 值类型（基本类型）

数值型(Number), 字符类型(String), 布尔值型(Boolean), null 和 underfined

## 2. 引用类型（类）

函数，对象，数组等

**值类型理解：**变量之间的互相赋值，是指开辟一块新的内存空间，将变量值赋给新变量保存到新开辟的内存里面；之后两个变量的值变动互不影响；

例如：

```
var a=10;//开辟一块内存空间保存变量 a 的值“10”；  
var b=a;//给变量 b 开辟一块新的内存空间，将 a 的值“10”赋值一份保存到新的内存里；  
//a 和 b 的值以后无论如何变化，都不会影响到对方的值；
```

**引用类型理解：**变量之间的互相赋值，只是指针的交换，而并非将对象（普通对象，函数对象，数组对象）复制一份给新的变量，对象依然还是只有一个，只是多了一个指引；

例如：

```
var a={x:1,y:2};//需要开辟内存空间保存对象，变量 a 的值是一个地址，这个地址指向保存对象的空间；  
var b=a;//将 a 的指引地址赋值给 b，而并非复制一给对象且新开一块内存空间来保存；  
//这个时候通过 a 来修改对象的属性，则通过 b 来查看属性时对象属性已经发生改变；
```

js 解释器有自己的内存管理机制，当不再有任何一个引用指向一个对象时，解释器就会认为此对象没用了，然后在动回收此对象所占用的内存资源；

## 3. 整数与浮点数

### 1). 整数

整数包括正整数, 负整数和 0. 如正整数:1、2、3. . . . . 负整数: -1、-2、-3. . . .

## 2). 浮点数

浮点数是表示小数的一种方法。所谓浮点就是小数点的位置不固定，与此相反有定点数，即小数点的位置固定。整数可以看做是一种特殊的定点数，即小数点在末尾一般的浮点数有点象科学计数法，包括符号位、指数部分和尾数部分。

浮点数是指小数点位置可以浮动的数据，通常以下式表示： $E$  次方  $N = M \cdot R$  其中  $N$  为浮点数， $M$  为尾数， $E$ （为阶码也就是多少次方的意思）， $R$  为阶的基数， $R$  一般为 2 进制 (01)，8... (01234567)，16... (0123456789abcdef)

简单说浮点数是指能够精确到小数点以后的数值类型

# Math 对象

**功能：** Math 对象用于执行数学任务

**语法：**

```
var pi_value=Math.PI;  
var sqrt_value=Math.sqrt(15);
```

**说明：**

Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数

Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

**属性：**

属性	描述
E	返回算术常量 e，即自然对数的底数（约等于 2.718）。
LN2	返回 2 的自然对数（约等于 0.693）。
LN10	返回 10 的自然对数（约等于 2.302）。

LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。
LOG10E	返回以 10 为底的 e 的对数（约等于 0.434）。
PI	返回圆周率（约等于 3.14159）。
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。
SQRT2	返回 2 的平方根（约等于 1.414）。

## 数值运算

### 1. 取最大值和最小值

**功能：** `min()` 方法可返回指定的数字中带有最小值的数字。

**语法：**

```
Math.min(n1, n2, n3, ..., nX)
```

**参数值：**

参数	描述
<code>n1, n2, n3, ..., nX</code>	可选。一个或多个值。在 ECMAScript v3 之前，该方法只有两个参数。

**返回值：**

类型	描述
Number	参数中最小的值。如果没有参数，则返回 Infinity。如果有某个参数为 NaN，或是不能转换成数字的非数字值，则返回 NaN。

**举例：** 实现返回数组中的最小值

**代码如下：**

```
<script type="text/javascript">
    document.write(Math.min(5, 7) + "<br />")
    document.write(Math.min(-3, 5) + "<br />")
    document.write(Math.min(-3, -5) + "<br />")
</script>
```



```
document.write(Math.min(7.25, 7.30))  
</script>
```

输出结果:

```
5  
-3  
-5  
7.25
```

## 2. 数值取整

1). `Math.ceil()` 执行向上舍入, 即它总是将数值向上舍入为最接近的整数; 2). `Math.floor()` 执行向下舍入, 即它总是将数值向下舍入为最接近的整数; 3). `Math.round()` 执行标准舍入, 即它总是将数值四舍五入为最接近的整数。

举例 1: 实现向上和向下取整

```
alert(Math.ceil(25.9)); //取整后为 26  
alert(Math.ceil(25.5)); //取整后为 26  
alert(Math.ceil(25.1)); //取整后为 26  
alert(Math.round(25.9)); //取整后为 26  
alert(Math.round(25.5)); //取整后为 26  
alert(Math.round(25.1)); //取整后为 25  
alert(Math.floor(25.9)); //取整后为 25  
alert(Math.floor(25.5)); //取整后为 25  
alert(Math.floor(25.1)); //取整后为 25
```

分析:

对于所有介于 25 和 26 (不包括 26) 之间的数值, `Math.ceil()` 始终返回 26, 因为它执行的是向上舍入。`Math.round()` 方法只在数值大于等于 25.5 时返回 26; 否则返回 25。最后, `Math.floor()` 对所有介于 25 和 26 (不包括 26) 之间的数值都返回 25。

举例 2: 标准取整

```
Math.round(12.2) // 返回 12  
Math.round(12.7) // 返回 13  
Math.round(12.0) // 返回 12
```

分析:

`round` 执行的是运算是四舍五入方法。

### 3. 绝对值 Math.abs()

`Math.abs(-1);`

其中-1 为要求绝对值的数值。

**注意：** 在求取一个数值的绝对值时可以使用 `abs` 函数，由于此函数属于 `Math` 类，所有在前边要加上 `Math.`。

举例：

```
<script language="javascript">
    document.write("0 的绝对值为: ", Math.abs(0), "<br>");
    document.write("1 的绝对值为: ", Math.abs(1), "<br>");
    document.write("-1 的绝对值为: ", Math.abs(-1), "<br>");
</script>
```

输出结果：

```
0的绝对值为: 0
1的绝对值为: 1
-1的绝对值为: 1
```

### 4. 随机数的生成 random()

**功能：** 可返回介于  $0 \sim 1$  之间的一个随机数。

**语法：**

`Math.random()`

举例：返回  $0 \sim 1$  之间的一个随机数

```
<script type="text/javascript">
    document.write(Math.random())
</script>
```

返回值：

```
0.9499262114364027
```

**注意：**

每次返回的结果都不一样，这里的结果只是参考

### 5. 返回数的平方根 sqrt()

**功能：** 可返回一个数的平方根

**语法：**

```
Math.sqrt(x)
```

注意：

其中参数“x”是必须的。若参数小于 0，则返回 NaN。

举例：返回几个数的平方根

```
var a=Math.sqrt(0);  
var b=Math.sqrt(1);  
var c=Math.sqrt(9);  
var d=Math.sqrt(0.64);  
var e=Math.sqrt(-9);
```

返回值为：

```
0  
1  
3  
0.8  
NaN
```

## 6. 幂运算 pow()

功能： 可返回 x 的 y 次幂的值。

语法：

```
Math.pow(x, y)
```

参数	描述
X	必需。底数。必须是数字。
Y	必需。幂数。必须是数字。

说明：

- 1). 如果结果是虚数或负数，则该方法将返回 NaN。
- 2). 如果由于指数过大而引起浮点溢出，则该方法将返回 Infinity。

举例： 把 pow() 运用到不同的数字组合上

```
<script type="text/javascript">  
  document.write(Math.pow(0,0) + "<br />")  
  document.write(Math.pow(0,1) + "<br />")  
  document.write(Math.pow(1,1) + "<br />")  
  document.write(Math.pow(1,10) + "<br />")  
  document.write(Math.pow(2,3) + "<br />")
```

```
document.write(Math.pow(-2,3) + "<br />")
document.write(Math.pow(2,4) + "<br />")
document.write(Math.pow(-2,4) + "<br />")
</script>
```

返回结果：

```
1
0
1
1
8
-8
16
16
```

## 字符串转化为数值

在 js 读取文本框或者其他表单数据的时候获得的值是字符串类型的, 比如两个文本框 a 和 b, 假设获得 a 的 value 值为 11, b 的 value 值为 9 , 那么 a.value 要小于 b.value, 由于他们都是字符串形式的.

方法主要有三种

转换函数、强制类型转换、利用 js 变量弱类型转换。

### 1. 转换函数

js 提供了 parseInt() 和 parseFloat() 两个转换函数。前者把值转换成整数, 后者把值转换成浮点数。仅仅有对 String 类型调用这些方法, 这两个函数才干正确执行; 对其它类型返回的都是 NaN(Not a Number)。

举例:

```
parseInt("1234blue"); //returns 1234
parseInt("0xA"); //returns 10
```

```
parseInt("22.5"); //returns 22  
parseInt("blue"); //returns NaN
```

parseInt() 方法还有基模式，能够把二进制、八进制、十六进制或其它不论什么进制的字符串转换成整数。基是由 parseInt() 方法的第二个参数指定的。

举例：

```
parseInt("AF", 16); //returns 175  
parseInt("10", 2); //returns 2  
parseInt("10", 8); //returns 8  
parseInt("10", 10); //returns 10
```

假设十进制数包括前导 0，那么最好采用基数 10，这样才不会意外地得到八进制的值。

举例：

```
parseInt("010"); //returns 8  
parseInt("010", 8); //returns 8  
parseInt("010", 10); //returns 10
```

parseFloat() 方法与 parseInt() 方法的处理方式相似。使用 parseFloat() 方法的还有一不同之处在于，字符串必须以十进制形式表示浮点数，parseFloat() 没有基模式。

举例：使用 parseFloat() 方法的演示

```
parseFloat("1234blue"); //returns 1234.0  
parseFloat("0xA"); //returns NaN  
parseFloat("22.5"); //returns 22.5  
parseFloat("22.34.5"); //returns 22.34  
parseFloat("0908"); //returns 908  
parseFloat("blue"); //returns NaN
```

## 2. 强制类型转换

还可使用强制类型转换（type casting）处理转换值的类型。使用强制类型转换能够访问特定的值，即使它是还有一种类型的。ECMAScript 中可用的 3 种强制类型转换例如以下：Boolean(value)——把给定的值转换成 Boolean 型；

Number(value)——把给定的值转换成数字（能够是整数或浮点数）；

String(value)——把给定的值转换成字符串。用这三个函数之中的一个转换值，将创建一个新值，存放由原始值直接转换成的值。这会造成意想不到的后果。当要转换的值是至少有一个字符的字符串、非 0 数字或对象（下一节将讨论这一

点) 时, Boolean() 函数将返回 true。假设该值是空字符串、数字 0、undefined 或 null, 它将返回 false。

能够用以下的代码段测试 Boolean 型的强制类型转换。

举例:

```
Boolean(""); //false - empty string
Boolean("hi"); //true - non-empty string
Boolean(100); //true - non-zero number
Boolean(null); //false - null
Boolean(0); //false - zero
Boolean(new Object()); //true - object
```

Number() 的强制类型转换与 parseInt() 和 parseFloat() 方法的处理方式相似, 仅仅是它转换的是整个值, 而不是部分值。

举例:

```
Number(false) //输出 0
Number(true) //输出 1
Number(undefined) //输出 NaN
Number(null) //输出 0
Number("5.5 ") //输出 5.5
Number("56 ") //输出 56
Number("5.6.7 ") //输出 NaN
Number(new Object()) //输出 NaN
Number(100) //输出 100
```

最后一种强制类型转换方法 String() 是最简单的。

举例:

```
var s1 = String(null); //"null"
var oNull = null;
var s2 = oNull.toString(); //won't work, causes an error
```

### 3. 利用 js 变量弱类型转换

举例:

```
<script>
  var str= '012.345 ';
  var x = str-0;
  x = x*1;
</script>
```

说明:

上例利用了 js 的弱类型的特点，仅仅进行了算术运算，实现了字符串到数字的类型转换，只是这种方法还是不推荐的

## 练习题

1. 写一个函数，给定原定于圆周长，计算该圆的面积是多少
2. 写一个函数，保留 3 位有效小数点。
3. 写一个函数，求给定 4 个数的最大值和最小值。
4. 写一个函数，对于给定的 a, b, c 求解  $ax^2+bx+c=0$  ( $x^2$  表示的 x 的平方)
5. 写一个生成 4 个小写字母的验证码的函数

## 数组简介

在程序语言中数组的重要性不言而喻，JavaScript 中数组也是最常使用的对象之一，数组是值的有序集合，由于弱类型的原因，JavaScript 中数组十分灵活、强大。之前我们都是用一个变量来存储一个字符，那么如果我们要存储多个字符的时候是否要定义很多个变量？这时我们可以定义一个数组来存储多个字符，使用起来会方便很多同时方便对变量的查找（遍历）。比如存储四个变量下面两种定义方法是一样的效果：

方法一：定义多个单变量

```
var str1="HTML"  
var str1="CSS"  
var str1="JAVASCRIPT"  
var str1="PHP"
```

方法二：定义数组

```
var ayy= new array ("HTML", "CSS", "JAVASCRIPT", "PHP") ;
```

## 1. 定义

数组对象用来在单独的变量名中存储一系列的值。

我们使用关键词 `new` 来创建数组对象。下面的代码定义了一个名为 `arr` 的数组对象：

```
var arr = new Array();
```

## 2. 数组的赋值

有两种向数组赋值的方法（你可以添加任意多的值，就像你可以定义你需要的任意多的变量一样）。

```
var mycars=new Array();  
mycars[0]="Saab";  
mycars[1]="Volvo";  
mycars[2]="BMW";
```

也可以使用一个整数自变量来控制数组的容量：

```
var mycars=new Array(3);  
mycars[0]="Saab";  
mycars[1]="Volvo";  
mycars[2]="BMW";  
var mycars=new Array("Saab", "Volvo", "BMW");
```

注意：

上的数字 0, 1, 2 等叫做数组的索引，它是用来被遍历时使用的

或者用数组字面量（直接量）方便的创建数组。一个数组字面量是在一对方括号中包裹着一个或多个用 逗号 隔开的表达式。并且数组中的每个值都有索引。从 0 开始。

```
var empty = [];  
var cars=["Saab", "Volvo", "BMW"];
```

## 3. 数组元素的获取

在 JavaScript 种获取数组某一项的值都是通过数组元素的下标来获取

举例：创建一个数组，然后获取数组中的某个字符。

代码如下：

```
<script type="text/javascript">  
    //创建数组
```



```
var arr=new Array("厦门","福州","漳州","龙岩","泉州");  
document.write(arr[4]);  
</script>
```

效果：

泉州

## 数组长度

定义：length 属性可设置或返回数组中元素的数目

语法：

```
arrayObject.length
```

说明：

- 1). 数组的 length 属性总是比数组中定义的最后一个元素的下标大 1。对于那些具有连续元素，而且以元素 0 开始的常规数组而言，属性 length 声明了数组中的元素的个数。
- 2). 数组的 length 属性在用构造函数 Array() 创建数组时被初始化。给数组添加新元素时，如果必要，将更新 length 的值。
- 3). 设置 length 属性可改变数组的大小。如果设置的值比其当前值小，数组将被截断，其尾部的元素将丢失。如果设置的值比它的当前值大，数组将增大，新的元素被添加到数组的尾部，它们的值为 undefined。

举例：实现如何使用 length 属性返回并设置数组的长度

代码如下：

```
<script type="text/javascript">  
var arr = new Array(3)  
arr[0] = "John"  
arr[1] = "Andy"
```

```
arr[2] = "Wendy"
document.write("Original length: " + arr.length)
document.write("<br />")
arr.length=5
document.write("New length: " + arr.length)
</script>
```

效果:

```
Original length: 3
New length: 5
```

## 数组元素的插入和删除

### 1. unshift()

功能：在数组开头添加元素，并返回该数组

格式：

数组对象.**unshift**(element1, element2, ..., elementn);

注意：

unshift() 方法不创建新的创建，而是直接修改原有的数组 举例：实现在数组的头部添加 f-z.cn 元素代码如下：

```
<script type="text/javascript">
var arr = new Array()
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
document.write(arr + "<br />")
document.write(arr.unshift("f-z.cn") + "<br />")
document.write(arr)
</script>
```

效果：

```
George, John, Thomas  
4  
f-z. cn, George, John, Thomas
```

## 2. push()

**功能：**向数组的末尾追加一个或多个元素，并且返回新的长度。

**格式：**

数组对象.**push**(element 1, element 2, ..., element n)

**注意：**

push() 方法是在数组的末尾添加元素，而不是在中间插入元素。

举例：实现在数组的尾部添加 f-z. cn 元素代码如下：

```
<script type="text/javascript">  
var arr = new Array()  
arr[0] = "George"  
arr[1] = "John"  
arr[2] = "Thomas"  
document.write(arr + "<br />")  
document.write(arr.push("f-z. cn") + "<br />")  
document.write(arr)  
</script>
```

**效果：**

```
George, John, Thomas  
4  
George, John, Thomas, f-z. cn
```

## 3. shift()

**功能：**把数组的第一个元素从其中删除，并返回第一个元素的值

**格式：**

arrayObject.**shift**()

**注意：**

如果数组是空的，那么 shift() 方法将不进行任何操作，返回 undefined 值。

请注意，该方法不创建新数组，而是直接修改原有的 arrayObject

举例：实现删除数组的头元素代码如下：

```
<script type="text/javascript">  
var arr = new Array(3)
```

```
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
document.write(arr + "<br />")
document.write(arr.shift() + "<br />")
document.write(arr)
</script>
```

效果:

```
George, John, Thomas
George
John, Thomas
```

#### 4. pop()

功能: 删除并返回数组中的最后一个元素

格式:

```
arrayObject.pop()
```

注意:

pop() 方法将删除 arrayObject 的最后一个元素, 把数组长度减 1, 并且返回它删除的元素的值。如果数组已经为空, 则 pop() 不改变数组, 并返回 undefined 值。举例: 实现删除数组的最后一个元素代码如下:

```
<script type="text/javascript">
var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
document.write(arr + "<br />")
document.write(arr.pop() + "<br />")
document.write(arr)
</script>
```

效果:

```
George, John, Thomas
Thomas
George, John
```

# 数组元素的截取

## 1. slice()

**功能：**可从已有的数组中返回选定的元素

**格式：**

```
arrayObject.slice(begin, end)
```

**说明：**

**begin：**必需参数。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。

**end：**规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素

该方法并不会修改数组，而是返回一个子数组 **小技巧：**

- 1). 您可使用负值从数组的尾部选取元素
- 2). 如果 end 未被规定，那么 slice() 方法会选取从 start 到数组结尾的所有元素 举例：实现选择数组中前四个元素

代码如下：

```
<script type="text/javascript">
    //创建数组的同时对元素赋值
    var arr=new
Array("Html", "Css", "JavaScript", "jQuery", "bootstrap");
    document.write("所选的元素为： "+"<br/>" +arr.slice(0,4));
</script>
```

**效果：**

```
所选的元素为：
Html,Css,JavaScript,jQuery
```

# 数组的排序与反转

## 1. sort()

**功能：** 用于对数组的元素进行排序

**语法：**

```
arrayObject.sort(sortby)
```

**返回值：** 对数组的引用。请注意，数组在原数组上进行排序，不生成副本。

**说明：**

如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序，说得更精确点，是按照字符编码的顺序进行排序。要实现这一点，首先应把数组的元素都转换成字符串（如有必要），以便进行比较。

如果想按照其他标准进行排序，就需要提供比较函数，该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，其返回值如下：

若 a 小于 b，在排序后的数组中 a 应该出现在 b 之前，则返回一个小于 0 的值。

若 a 等于 b，则返回 0。

若 a 大于 b，则返回一个大于 0 的值。

**举例：** 创建一个数组，并按字母顺序进行排序

**代码如下：**

```
<script type="text/javascript">
  var arr = new Array(6)
  arr[0] = "George"
  arr[1] = "John"
  arr[2] = "Thomas"
```

```
arr[3] = "James"  
arr[4] = "Adrew"  
arr[5] = "Martin"  
document.write(arr + "<br />")  
document.write(arr.sort())  
</script>
```

效果:

```
George, John, Thomas, James, Adrew, Martin  
Adrew, George, James, John, Martin, Thomas
```

## 2. reverse() 方法

**功能:** 用于颠倒数组中元素的顺序。

**注意:**

它不是创建新的数组，而是改变原来的数组。

**语法:**

```
arrayObject.reverse()
```

**举例:** 创建一个数组，然后颠倒其元素的顺序

**代码如下:**

```
<script type="text/javascript">  
  var arr = new Array(3)  
  arr[0] = "George"  
  arr[1] = "John"  
  arr[2] = "Thomas"  
  document.write(arr + "<br />")  
  document.write(arr.reverse())  
</script>
```

效果:

```
George, John, Thomas  
Thomas, John, George
```

# 数组的链接与转换

## 1. concat()

**功能：**连接两个或多个数组

**格式：**

```
arrX.concat(arrX1, arrX2, . . . . . , arrXn)
```

**说明：**

- 1). 返回的是一个新的数组。该数组是通过把所有 arrX1~arrXn 参数添加到 arrX 中生成的。
- 2). arr 可以使任意数组
- 3). 该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本 举例：  
实现多个数组的连接

代码如下：

```
<script type="text/javascript">
var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
var arr2 = new Array(3)
arr2[0] = "James"
arr2[1] = "Adrew"
arr2[2] = "f-z. cn"
arr.concat(arr2);           //连接两个数组
var arr3=new Array();       //数组与元素连接
arr3=arr.concat(arr2);
document.write(arr3.concat(4,5));
</script>
```

**效果：**

```
George, John, Thomas, James, Adrew, f-z. cn, 4, 5
```

## 2. join()

**功能：**把数组中的所有元素放入一个字符串

元素是通过指定的分隔符进行分隔的



格式:

```
arr.join("指定的分隔符")
```

注意:

“指定的分隔符”是可选项，没有指定的话默认是逗号 举例：实现数组元素的连接成字符串

代码如下:

```
<script type="text/javascript">
var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "fenzhen"
document.write(arr.join()+"<br/>") //没有指定分隔符
document.write(arr.join("$"))      //有指定分隔符
</script>
```

效果:

```
George, John, Thomas, fenzhen
George$John$Thomas$fenzhen
```

### 3. toString()

功能: 可把一个逻辑值转换为字符串, 并返回结果 格式:

```
arr.toString()
```

代码如下:

```
<script type="text/javascript">
var arr=new Array("html", "css", "javascript", "jQuery", "Ajax",
"fenzhen");
document.write(arr.toString());
</script>
```

效果:

```
html, css, javascript, jQuery, Ajax, fenzhen
```

### 4. join() 与 toString() 的区别

`join()` 方法将数组中所有的元素转化为字符串，并将这些字符串有逗号隔开合并成一个字符串作为方法的结果返回。如果调用时给定参数 `string`，就将 `string` 作为在结果字符串中分开有各个数组元素形成的字符串的分隔符。

`toString()` 方法返回一个包含数组中所有元素，且元素之间以逗号隔开的字符串，该方法在将数值作为字符串使用时强制调用，且无须显示声明此方法的调用。

## 练习题

1. 写一个函数，给定一个数组，求他的最大值，最小值, 第二大的值，最大的偶数，最小的基数，求出所有元素的和
2. 写一个函数，求一个数组第一个元素和最后一个元素的和
3. 写一个函数，不用 `shift` 的方法，获取数组的第一个元素和最后一个元素。
4. 把字符串 "1212+10+20+30"，变成表达式 `1212-10-2-30`，并且计算其结果
5. 用两种方式将 `[10, 20, 30]` 变成字符串 `'10+20+30'`
6. 把一个数组翻转后连接自身
7. 在数组 `【100, 300, 400】` 中的 100 与 300 之间插入 200；在末尾插入 500

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



## 时间对象

## 时间的使用场景

生活中我们经常需要知道时间, 感觉时间是一个很小的细节是吗? 但是用到时间的地方可是很多的哦。

### 1. 显示文章发布时间

例子 1: 一篇百度新闻报道

## 财经一习谈 | 这五年，习近平的扶贫足迹

侯雪静、董峻、胡璐/新华网  
2017-05-22 10:50

字号

从黄土高坡到雪域高原，从西北边陲到云贵高原，他风雪兼程，几乎走遍全国14个集中连片特困地区。他对贫困群众念兹在兹、心有牵挂。

近五年来，30多次国内各地考察，几乎每次都提到扶贫，超过25个重要场合对扶贫开发工作作出重要指示。

党的十八大以来，以习近平同志为核心的党中央把脱贫攻坚作为关乎党和国家政治方向、根本制度和发展道路的大事，扶贫开发成为实现全面小康社会的底线目标。

**【习语】全面建成小康社会，最艰巨最繁重的任务在农村、特别是在贫困地区。没有农村的小康，特别是没有贫困地区的小康，就没有全面建成小康社会。**



说明：

我们可以看到红色方框里面用到了时间，对于新闻而言，时间是必不可少的元素哦。

### 2. 在线时钟

例子 2：百度上面的在线时钟，因为只是图片，所以同学们看不到这个时间是会动态改变的，就像我们手表一样，他每一秒都在改变。

北京时间 - 国家授时中心标准时间



### 3. 在线日历

例子 3：网络上常用的在线日历。



说明：这个我们下几节会讲到实现这个日历需要用到的知识哦，我们再多看几个例子吧。

### 4. 倒计时

例子 4：一个很好看的倒计时卡片。

推荐	个人
2017年05月 <b>30</b> 00:00:00 五月初五	<b>端午节 还剩7天4小时29分17秒</b> 农历五月初五是端午节，又称端阳节，是中国十分盛行的隆重节日。
2017年06月 <b>01</b> 00:00:00 五月初七	<b>六一儿童节 还剩9天4小时29分17秒</b> 每年六月一日是国际儿童节，又称儿童节、六一儿童节。六一儿童节是全世界少年儿童的节日。
2017年06月 <b>07</b> 08:00:00 五月十三	<b>2017年高考 还剩15天12小时29分17秒</b> 2017年全国高等教育入学考试。

## 5. 定时器

例子 5:



同学们看了之后是不是发现用到时间的地方很多？到处都用到它？是不是很想学习怎么实现呢？下面我们一起来学习时间对象吧。

## 基本概念

### 1. 时区

时区 (Time Zone) 是地球上的区域使用同一个时间定义。

### 2. 时间格式

时间格式有很多种，我们可以根据实际情况以及个人喜好选择哦，下面老师举几个时间格式的例子。

1). 2016-12-12 12:00:00

2). 2016/12/12 12:00:00

3). thu may 14 2015 12:00:00

**说明：**

时间什么格式不重要，它只是同学们觉得好看或者一种习惯写法而已，不是必须要按照某种格式来哦。

### 3. 时间戳

时间戳，一个能表示一份数据在某个特定时间之前已经存在的、完整的、可验证的数据，通常是一个字符序列，唯一地标识某一刻的时间。

是不是感觉很虚无？时间戳不是什么玄乎的东西，就是文件属性里的创建、修改、访问时间。

1). 1970 到现在的毫秒数.

The screenshot shows a web-based tool for converting between Unix timestamps and Beijing time. At the top, it displays the current Unix timestamp as 1495453617, with buttons for '开始' (Start), '停止' (Stop), and '刷新' (Refresh). Below this, there are three main conversion sections. The first section allows converting a Unix timestamp (1495453544) to Beijing time (2017/5/22 \*9:45:44). The second section allows converting a Beijing time string to a Unix timestamp. The third section provides a detailed input for Beijing time (year, month, day, hour, minute, second) and a button to convert it to a Unix timestamp.

**说明：**

1). 我们可以看到 1970 到现在的毫秒数吧。

2). 1 秒=1000 毫秒

This screenshot shows the tool with the Beijing time set to 2017年4月5日5时5分5秒. The '转换Unix时间戳' button has been clicked, resulting in a Unix timestamp of 1491339905000 milliseconds, as shown in the output field on the right.

北京时间	2017	年	4	月	5	日	5	时	5	分	5	秒	转换Unix时间戳	1491339905	秒	▼
------	------	---	---	---	---	---	---	---	---	---	---	---	-----------	------------	---	---

说明：

对比两个图，我们可以看到。上面以毫秒为单位的值是下面以秒为单位的值的1000 倍，所以我们可以知道 1 秒=1000 毫秒。

## 4. 时间对象的定义

1). new Date()

怎么创建一个日期对象？看下面的例子

例子 1：创建一个日期对象：

```
var objDate=new Date([arguments list]);
```

说明：

objDate 是我们自定义的日期实例名称，后面是创建一个 Date 对象，Date 后面的括号里面放参数。

参数的形式主要有以下 3 种：

例子 2：

```
new Date("month dd yyyy hh:mm:ss");
```

说明：

后面的 hh:mm:ss 可选（不选的话就是默认的开始时间），而且前三项的顺序可以随意，甚至各字段后面可以加逗号。

例子 3：

```
new Date(yyyy, mth, dd, hh, mm, ss);
```

说明：

除了前两个字段（年、月字段）外，其余的都是可选的（不选的话就默认为开始的），不过，此处顺序最好别随意变换。

例子 4：

```
new Date(ms);
```

说明：

参数表示的是需要创建的时间和 GMT 时间 1970 年 1 月 1 日之间相差的毫秒数。



### 总结：

各种函数的含义如下

month: 用英文表示月份名称，从 January 到 December

mth: 用整数表示月份，从（1月）到 12（12月）

dd: 表示一个月中的第几天，从 1 到 31

yyyy: 四位数表示的年份

hh: 小时数，从 0（午夜）到 23（晚 11 点）

mm: 分钟数，从 0 到 59 的整数

ss: 秒数，从 0 到 59 的整数

ms: 毫秒数，为大于等于 0 的整数。

### 注意：

- 1). *js 的时间是调用用户系统的时间*，但是用户的电脑时间有可能不正确，所以可能会有问题哦。如果要保证正确，记得先将系统时间调为正确的。
- 2). 如果需要标准时间，需要用服务器的时间（以后介绍，想了解的同学也可以自己了解）

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



# 将时间对象转为字符串

## 1. toString()

toString() 函数用于将当前对象以字符串的形式返回，toString() 函数的返回值为 String 类型。也就是返回当前对象的字符串形式。

JavaScript 的许多内置对象都重写了该函数，以实现更适合自身的功能需要。

类型	行为描述
Array	将 Array 的每个元素转换为字符串，并将它们依次连接起来，两个元素之间用英文逗号作为分隔符进行拼接。
Boolean	如果布尔值是 true，则返回"true"。否则返回"false"。
Date	返回日期的文本表示。
Error	返回一个包含相关错误信息的字符串。
Function	返回如下格式的字符串，其中 functionname 是一个函数的名称，此函数的 toString 方法被调用： "function functionname() { [native code] }"
Number	返回数值的字符串表示。还可返回以指定进制表示的字符串
String	返回 String 对象的值。
Object（默认）	返回"[object ObjectName]"，其中 ObjectName 是对象类型的名称。

例子 1：

```
<script>
    //数组
    var array = ["CodePlayer", true, 12, -5];
    document.writeln( array.toString() );
    document.write("<br>"); //换行
    // 日期
```

```

var date = new Date(2013, 7, 18, 23, 11, 59, 230);
document.writeln( date.toString());
document.write("<br>");
// 日期 2
var date2 = new Date(1099, 7, 18, 23, 11, 59, 230);
document.writeln( date2.toString());
document.write("<br>");
// 数字
var num = 15.26540;
document.writeln( num.toString());
// 布尔
var bool = true;
document.writeln( bool.toString() );
document.write("<br>");
// Object
var obj = {name: "张三", age: 18};
document.writeln( obj.toString() );
</script>

```

结果为：

```

CodePlayer,true,12,-5
Sun Aug 18 2013 23:11:59 GMT+0800
Fri Aug 18 1099 23:11:59 GMT+0800
15.2654 true
[object Object]

```

说明：

`document.write("<br>");` 是为了结果好看而加入的换行，同学们可以不用管它哦。

根据上例，结合上面的表格说明，同学们应该能理解 `toString()` 函数来，下面我们学习新的函数吧。

## 2. toUTCString ()

`toUTCString()` 方法可根据世界时 (UTC) 把 `Date` 对象转换为字符串，并返回结果。

语法：

```
dateObject.toUTCString()
```

说明：

前面是任意的时间对象名，具体可以看例子哦。

例子 2： 我们将使用 toUTCString() 来把今天的日期转换为（根据 UTC）字符串。

```
<script type="text/javascript">
    var d = new Date()
    document.write (d.toUTCString())
</script>
```

结果为：



Tue, 23 May 2017 10:33:48 GMT

说明：

这里的结果会因为运行程序的时间而不同哦，所以同学们不要觉得错误了哦。

### 3. toLocalString ()

toLocalString () 方法把数组转换为本地字符串。


语法：

```
arrayObject.toLocaleString()
```

例子 3：

```
<script type="text/javascript">
    var arr = new Array(3);
    arr[0] = "George";
    arr[1] = "John";
    arr[2] = "Thomas";
    document.write(arr.toLocaleString())
</script>
```

结果为：



George,John,Thomas

同学们看到这个,是不是感觉这三个的结果相差不大?不知道具体区别在哪里?  
那么我们现在来具体区分一下他们的区别吧。

#### 4. 三者的区别

1). `toString()` 函数用于将当前对象以字符串的形式返回,它的返回值为 `String` 类型。

2). `toUTCString()` 根据世界时 (UTC) 把 `Date` 对象转换为字符串

3). `toLocaleString()` 方法把数组转换为本地字符串,首先调用每个数组元素的 `toLocaleString()` 方法,然后使用地区特定的分隔符把生成的字符串连接起来,形成一个字符串。例如,同样是 3 月 21 日,在美国, `(new Date).toLocaleString()` 可能会返回 "03/21/08 01:02:03",而在欧洲,返回值则可能是 "21/03/08 01:02:03",因为欧洲的惯例是将日期放在月份前面。

**注意:**

`toLocaleString` 只是用来显示结果给用户;最好不要在脚本中用来做基本计算,因为返回的结果是随机器不同而不同的。

## 获取时间

### 1. 获取年月日

### 1). getFullYear()

getFullYear() 方法可返回一个表示年份的 4 位数字。

例子 1:

```
<script type="text/javascript">
  var d = new Date();
  document.write(d.getFullYear());
</script>
```

结果为:



说明:

1). 该函数的返回值是一个四位数，表示包括世纪值在内的完整年份，而不是两位数的缩写形式。

2). 该方法总是结合一个 Date 对象来使用。

### 2). getMonth()

getMonth() 方法可返回表示月份的数字。

例子 2:

```
<script type="text/javascript">
  var d=new Date();
  document.write(d.getMonth());
</script>
```

结果为:



说明:

1). 它返回的月份字段，是使用本地时间，所以肯定有些同学的结果和老师的不一样哦。而且返回值是 0（一月）到 11（十二月）之间的一个整数。

2). 该方法也总是结合一个 Date 对象来使用。

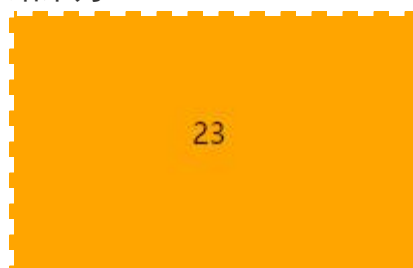
### 3). getDate()

getDate() 方法可返回月份的某一天，返回对象所指的月份中的某一天，使用本地时间。返回值是 1 ~ 31 之间的一个整数。

例子 3：返回当前为多少号。

```
<script type="text/javascript">
    var d = new Date()
    document.write(d.getDate())
</script>
```

结果为：



注意：

- 1). 月份是用 0 开始的哦，0 就代表 1 月，所以根本不会有 12，同学们记得注意哦。
- 2). 返回的结果由同学们运行该程序的日期有关哦，只要是当前的号数就是正确的哦。

## 2. 获取时分秒

### 1). getFullYear()

返回 Date 对象中用本地时间表示的年份值。

例子 4：

```
<script type="text/javascript">
    var d = new Date();
    var n = d.getFullYear();
    document.write(n);
</script>
```

结果为：



说明：

- 1). `getFullYear` 方法以绝对数字的形式返回年份值。
- 2). `getFullYear()` 和 `getYear()` 区别就是前者不会出现浏览器兼容性问题，而后者会因浏览器不同出现不同的结果。

## 2). `getMonth()`

`getMonth()` 方法可返回表示月份的数字。

例子 5：返回当前月份

```
<script type="text/javascript">
    var d=new Date();
    document.write(d.getMonth());
</script>
```

结果为：



## 3. 获取星期

### 1). `getDay()`

`getDay()` 方法可返回表示星期的某一天的数字。返回值是 0（周日）到 6（周六）之间的一个整数。

例子 6：返回当前星期几的数字表示。

```
<script type="text/javascript">
    var d=new Date();
    document.write(d.getDay());
</script>
```

结果为：



2

**说明：**

因为今天星期二，所以返回 2。

**注意：**

星期日的返回值是 0 哦！

## 2). 用 switch

例子 8：

```
<script>
//创建 Date 对象的一个实例
var today = new Date();
//取出 today 对象中的星期值
var week = today.getDay(); //返回值 0-6, 0 代表星期天
//输出中文的星期
switch (week) {
    case 1:
        str = "星期一";
        break;
    case 2:
        str = "星期二";
        break;
    case 3:
        str = "星期三";
        break;
    case 4:
        str = "星期四";
        break;
    case 5:
        str = "星期五";
        break;
    case 6:
        str = "星期六";
        break;
    default:
        str = "星期天";
}
```

```
}  
document.write("今天是: " + str);  
</script>
```

结果为:



今天是：星期二

#### 4. 用数组

例子 8:

```
<script type="text/javascript">  
var d = new Date();  
var weekday = new Array(7);  
weekday[0] = "星期日";  
weekday[1] = "星期一";  
weekday[2] = "星期二";  
weekday[3] = "星期三";  
weekday[4] = "星期四";  
weekday[5] = "星期五";  
weekday[6] = "星期六";  
document.write("今天是" + weekday[d.getDay()])  
</script>
```

结果为:



今天是：星期二

总结:

- 1). 时间函数很多，同学们可能会觉得很难记，可以不用记住，在实际运用中，多用几次自然就熟悉了。
- 2). 一定要特别注意：时间里面的月份是用 0 开始。

## 设置时间

### 1. 设置时间对象

#### 1). 用时间字符串

例子 1:

```
<script type="text/javascript">
  var s = '2017-04-18 09:16:15';
  s = s.replace(/-/g, "/");
  var date = new Date(s);
  document.write(date);
</script>
```

结果为:



Tue Apr 18 2017 09:16:15 GMT+0800 (中国标准时间)

说明:

- 1). `/-/g` 是正则表达式，表示将所有短横线-替换为斜杠/，其中 `g` 表示全局替换。
- 2). 对于上面代码有不懂的同学可以不用管它，对于现阶段的我们基本用不上，随着我们深入了解 js，就又会很容易理解了，所以不用着急哦。

## 2). 用时间戳

例子 2:

```
<script type="text/javascript">
    function getLocalTime(nS) {
        return new Date(parseInt(nS) * 1000).toLocaleString().substr(0, 17)
    }
    document.write(getLocalTime(1293072805));
</script>
```

结果为:



2010/12/23 上午10:5

## 3). 获取当前时间

例子 3:

```
<script language="javascript" type="text/javascript">
    <!--
    //获得当前时间, 刻度为一千分之一秒
    var initializationTime = (new Date()).getTime();
    function showLeftTime() {
        var now = new Date();
        var year = now.getFullYear();
        var month = now.getMonth();
        var day = now.getDate();
        var hours = now.getHours();
        var minutes = now.getMinutes();
        var seconds = now.getSeconds();
        document.all.show.innerHTML = "" + year + "年" + month + "月" + day
        + "日 " + hours + ":" + minutes + ":" + seconds + "";
        //一秒刷新一次显示时间
        var timeID = setTimeout(showLeftTime, 1000);
    }
    //-->
</script>
```

结果为:



2017年4月23日 19:54:17

说明：

月份是从 0 开始的，所以这里表示的时间应该是 2017 年 5 月 23 日哦！

## 2. 设置年月日

### 1). setFullYear (year, month, day)

setFullYear () 方法用于设置年份。

例子 4：

```
<script type="text/javascript">
    var d = new Date();
    d.setFullYear(1992);
    document.write(d);
</script>
```

结果为：



Sat May 23 1992 19:58:52 GMT+0800 (中国标准时间)

说明：

- 1). 参数 year 是必须有的，表示年份的四位整数。用本地时间表示。
- 2). 参数 month 是可选的，表示月份的数值，介于 0 ~ 11 之间。用本地时间表示。
- 3). 参数 day 是可选的，表示月中某一天的数值，介于 1 ~ 31 之间。用本地时间表示。

## 2). `setMonth(month, day)`

`setMonth()` 方法用于设置月份。

例子 5:

```
<script type="text/javascript">
    var d = new Date();
    d.setMonth(0);
    document.write(d);
</script>
```

结果为:



Mon Jan 23 2017 20:03:43 GMT+0800 (中国标准时间)

说明:

- 1). 参数 `month` 是必须有的, 是一个表示月份的数值, 该值介于 0 (一月) ~ 11 (十二月) 之间。
- 2). 参数 `day` 是可选的, 是一个表示月的某一天的数值, 该值介于 1 ~ 31 之间 (以本地时间计)。

在 ECMAScript 标准化之前, 不支持该参数。

## 3). `setDate(day)`

`setDate()` 方法用于设置一个月的某一天。

例子 6:

```
<script type="text/javascript">
    var d = new Date();
    d.setDate(15);
    document.write(d);
</script>
```

结果为:

Mon May 15 2017 20:07:09 GMT+0800 (中国标准时间)

说明：

参数是必须有的，表示一个月中的一天的一个数值（1 ~ 31）。

### 3. 设置时分秒

#### 1). `setHours(hour, min, sec, millisec)`

`setHours()` 方法用于设置指定的时间的小时字段。

例子 7：

```
<script type="text/javascript">
  var d = new Date();
  d.setHours(15);
  document.write(d);
</script>
```

结果为：

Tue May 23 2017 15:12:51 GMT+0800 (中国标准时间)

说明：

- 1). 参数 `hour` 是必须有的，它表示小时的数值，介于 0（午夜） ~ 23（晚上 11 点） 之间，以本地时间为准。
- 2). 参数 `min` 是可选的，它表示分钟的数值，介于 0 ~ 59 之间。在 ECMAScript 标准化之前，不支持该参数，以本地时间为准。

3). 参数 `sec` 是可选的, 它表示秒的数值, 介于  $0 \sim 59$  之间。在 ECMAScript 标准化之前, 不支持该参数, 以本地时间为准。

4). 参数 `millisec` 是可选的, 它表示毫秒的数值, 介于  $0 \sim 999$  之间。在 ECMAScript 标准化之前, 不支持该参数, 以本地时间为准。

## 2). `setMinutes(min, sec, millisec)`

`setMinutes()` 方法用于设置指定时间的分钟字段, 同样可用于设置秒数与毫秒数。

例子 8:

```
<script type="text/javascript">
  var d = new Date();
  d.setMinutes(17);
  document.write(d);
</script>
```

结果为:



Tue May 23 2017 20:17:46 GMT+0800 (中国标准时间)

说明:

1). 参数 `min` 是必须有的, 它表示分钟的数值, 介于  $0 \sim 59$  之间, 以本地时间为准。

2). 参数 `sec` 是可选的, 它表示秒的数值, 介于  $0 \sim 59$  之间。在 ECMAScript 标准化之前, 不支持该参数。

3. 参数 `millisec` 是可选的, 它表示毫秒的数值, 介于  $0 \sim 999$  之间。在 ECMAScript 标准化之前, 不支持该参数。

## 3). `setSeconds(sec, millisec)`

`setSeconds()` 方法用于设置指定时间的秒的字段

例子 9:



```
<script type="text/javascript">
  var d = new Date();
  d.setSeconds(17);
  document.write(d);
</script>
```

结果为：

Tue May 23 2017 20:20:17 GMT+0800 (中国标准时间)

说明：

- 1). 参数 `sec` 是必须有的，它表示秒的数值，该值是介于 0 ~ 59 之间的整数。
- 2). 参数 `millisec` 是可循环的，它表示毫秒的数值，介于 0 ~ 999 之间。在 ECMAScript 标准化之前，不支持该参数。

好啦，同学们时间对象已经学习完了，我们就来看看下一节的习题吧。

## 练习题

用两种获取当前时间 3 天后的时间，并打印成 “年-月-日 时：分：秒”

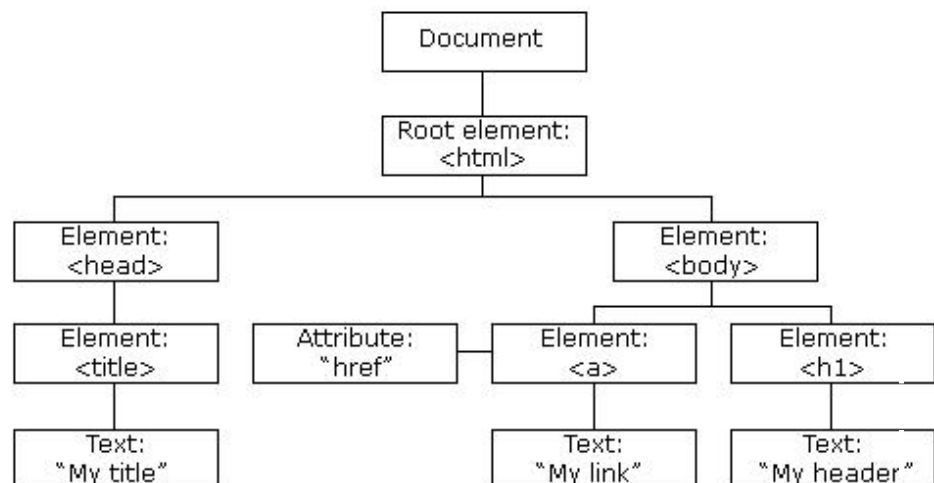
- 1). 时间戳
- 2). 时间戳 `etDate(day)` 的方式
2. 写一个获取今天是星期的函数
3. 把当前时间转化为国际标准时间，美国时间；
4. 中国采用哪个时区，比国际时间快还是慢，相差多久。

5. 写一个函数，计算给定时间与当前时间的间隔，比如显示“1 秒前，1 分钟前，1 小时前，1 天前，1 月前，超过 1 年显示具体的年月日，比如“2016-10-12”

## DOM 对象

### Dom 对象简介

#### 1. HTML DOM 树：



#### 2. 什么是 DOM？

DOM 是 W3C（万维网联盟）的标准。DOM 定义了访问 HTML 和 XML 文档的标准：“W3C 文档对象模型（DOM）是中立于平台和语言的接口，它允许程序和脚本动态地访问和更新文档的内容、结构和样式。”

W3C DOM 标准被分为 3 个不同的部分：

- 1). 核心 DOM—针对任何结构化文档的标准模型
- 2). XML DOM—针对 XML 文档的标准模型
- 3). HTML DOM—针对 HTML 文档的标准模型

### 3. HTML DOM 是：

- 1). HTML 的标准对象模型
- 2). HTML 的标准编程接口
- 3). W3C 标准 HTML DOM 定义了所有 HTML 元素的对象和属性，以及访问它们的方法。换句话说，HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准

### 4. 对 DOM 节点的理解

我们先给个 html 例子，我们对例子一一解析就会很清楚了举例：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>DOM 节点</title>
</head>
<body>
  <p>
    <a href="#">前端大学</a>
  </p>
  <span>
    前端大学
  </span>
</body>
</html>
```

解析：

#### 1). 根节点

上面例子中的根节点就是<html>

#### 2). 父节点

一个节点的上面一个节点就是它的父节点。如本例中<p>的父节点是<body>，同理<span >的父节点也是<body>

#### 3). 子节点

跟父节点相反，一个节点的下一个节点就是它的子节点。<body> 的子节点是<p>、<span>，它的父节点是<html>

#### 4). 兄弟节点

兄弟节点有相同父节点且在同一个层。本例中的兄弟节点是<p>跟<span>

# 获取 dom 对象

获取 DOM 中指定元素本章的主要知识点：

- 1). 熟悉获取 DOM 中指定元素的两种方法：`getElementById()` 和 `getElementsByName()`。
- 2). 熟悉 `getElementById()` 和 `getElementsByName()` 的区别。在 CSS 入门教程的学习中，我们知道：如果要对某个元素进行样式操作，就必须先通过一种方式来选中该元素（也就是 CSS 选择器），然后才能进行相关样式的操作。在 JavaScript 中，如果要对节点进行创建、删除等操作，同样也要通过一种方式来选中该节点。只有你获取了该元素节点，才能进行各种操作，很容易理解吧。在 JavaScript 中，可以通过以下 2 种方式来选中指定元素：

- 1). `getElementById()`；
- 2). `getElementsByName()`；

## 1. `getElementById()`

在 JavaScript 中，如果想通过 id 来选中元素，我们可以使用 document 对象的 `getElementById()` 方法。

`getElementById()` 方法类似于 CSS 中的 id 选择器。

语法：

```
document.getElementById("元素 id");
```

举例：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
</head>
<body>
  <div id="fenzhen">前端大学 JavaScript 入门教程</div>
  <script type="text/javascript">
    var e = document.getElementById("fenzhen");
    e.style.color = "red";
```

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

在浏览器预览效果如下：



**分析：**

这里使用 `document.getElementById()` 的方法获取 id 为 `fenzhen` 的 `div` 元素，然后把这个 DOM 对象赋值给变量 `e`，然后使用 DOM 对象的 `style` 对象来设置 `div` 元素颜色为红色。我们在“JavaScript 操作 CSS 样式”这一节会详细给大家介绍这种方法。

## 2. `getElementsByName()`

在 JavaScript 中，如果想通过 `name` 来选中元素，我们可以使用 `document` 对象的 `getElementsByName()` 方法。

**语法：**

```
document.getElementsByName("表单元素 name 值");
```

**说明：**

`getElementsByName()` 方法都是用于获取表单元素。

与 `getElementById()` 方法不同的是，使用该方法的返回值是一个数组，而不是一个元素。因此，我们想要获取具体的某一个表单元素，就必须通过数组下标来获取。

**注意：**

是 `getElementsByName()` 而不是 `getElementByName()`。因为是数组嘛，当然是复数。

**举例：**

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
</head>
<body>
  <input name="web" id="radio1" type="radio" value="HTML"/>
  <input name="web" id="radio2" type="radio" value="CSS"/>
  <input name="web" id="radio3" type="radio" value="JavaScript"/>
  <input name="web" id="radio4" type="radio" value="jQuery"/>
  <script type="text/javascript">
    alert(document.getElementsByName("web")[0].value);
  </script>
</body>
</html>
```

在浏览器预览效果如下：



分析：

`getElementById()` 方法在实际开发中比较少用，大家了解一下即可。

其实可以这样说，`getElementById()` 和 `getElementsByName()` 这两种方法是“JavaScript 选择器”。

除了 `getElementById()` 和 `getElementsByName()` 这两种方法，JavaScript 还提供另外一种 `getElementsByTagName()` 方法，这个方法类似于 CSS 中的元素选择器。但是 `getElementsByTagName()` 方法有个很大缺点，它会把整个页面中相同的元素都选中。用起来也比较麻烦，实际开发中很少用到。

## 获取相关节点

## 1. 访问父节点

### A. parentNode()

功能：获取指定节点的父节点

语法：

```
elementNode.parentNode
```

注意：父节点只能有一个 举例：实现获取<p>节点的父节点代码如下：

```
<div id="text">
  <p id="con"> parentNode 获取指点节点的父节点</p>
</div>
<script type="text/javascript">
  var mynode= document.getElementById("con");
  document.write(mynode.parentNode.nodeName);
</script>
```

效果：

```
parentNode 获取指点节点的父节点
DIV
```

## 2. 访问兄弟节点

### 1). nextSibling

功能：可返回某个节点之后紧跟的节点（处于同一树层级中）

语法：

```
nodeObject.nextSibling
```

说明：

- 1). 如果无此节点，则该属性返回 null
- 2). 元素中的空格被视作文本，而文本被视作文本节点 举行：

实现获取兄弟节点的 ID 代码如下：

```
<body style="background-color:orange">
  <p>列表示例：</p>
  <ul id="myList">
    <li id="item1">fenzhen</li><li id="item2">waite</li>
  </ul>
  <p id="demo">点击按钮来获得首个项目的下一个同胞的 id。</p>
  <button onclick="myFunction()">获取</button>
  <script>
```

```
function myFunction()
{
    var x=document.getElementById("demo");
    x.innerHTML=document.getElementById("item1").nextSibling.id;
}
</script>
</body>
```

效果:

列表示例:

- fenzhen
- waite

点击按钮来获得首个项目的下一个同胞的 id。

获取

点击前

列表示例:

- fenzhen
- waite

item2

获取

点击后

注意:

返回某个节如果上面两个<li>标签有空格则会出现“undefined”

## 2). previousSibling

功能: 可返回某个节点之前紧跟的节点（处于同一树层级中）

语法:

```
nodeObject.previousSibling
```

说明:

- 1). 如果无此节点, 则该属性返回 null。
- 2). 两个属性获取的是节点。
- 3). Internet Explorer 会忽略节点间生成的空白文本节点（例如, 换行符号）, 而其它浏览器不会忽略 代码如下:

```
<body>
<p>列表示例: </p>
```



```

<ul id="myList"><li id="item1">fenzhen</li><li
id="item2">waite</li></ul>
<p id="demo">请点击按钮来获得第二个列表项的前一个同胞的 id。</p>
<button onclick="myFunction()">获取</button>
<script>
function myFunction()
{
var itm=document.getElementById("item2");
var x=document.getElementById("demo");
x.innerHTML=itm.previousSibling.id;
}
</script>
</body>

```

效果:

列表示例:

- fenzhen
- waite

请点击按钮来获得第二个列表项的前一个同胞的 id。

获取

点击前

列表示例:

- fenzhen
- waite

item1

获取

点击后

### 3. childNodes

**功能:** 访问选定元素节点下的所有子节点的列表, 返回的值可以看作是一个数组, 它具有 length 属性

**语法:**

```
elementNode.childNodes
```

**注意:**

如果选定的节点没有子节点, 则该属性返回不包含节点的 NodeList

举例：实现获取<body>的子节点的相关信息代码如下：

```
<body style="background-color:orange">
  <p id="demo">请点击按钮来获得 body 元素子节点的相关信息。</p>
  <button onclick="myFunction()">获取</button>
  <script>
    function myFunction()
    {
      var txt="";
      var c=document.body.childNodes;
      for (i=0; i<c.length; i++)
      {
        txt=txt + c[i].nodeName + "<br>";
      };
      var x=document.getElementById("demo");
      x.innerHTML=txt;
    }
  </script>
</body>
```

效果：



# 操作 DOM 属性、HTML 和文本

## 1. js 获取的 html 内容包括

- 1). 文本内容
- 2). 属性值

可以采用 js 的 dom 方法, 比如 `document.getElementById()`

但是此种方法不简洁, 使用起来不方便, 违反了短小精悍的原则, 推荐采用 jquery 的方法, jquery 即 js 库, 封装了 js 的一些方法, 直接用就好了。

**注意:**

- 1). 使用 jquery 需要包含 jquery.js 文档 `<script type="text/JavaScript" src="jquery.js"></script>`

至于 jquery.js, 网上老多了, 请自行下载。

- 2). `<script></script>` 里面必须包含 `$(document).ready()` 方法。

这里介绍 jquery 的 3 个方法

- 1). `text()` text 即设置或者返回文本内容
- 2). `html()` 设置或者返回 html 内容
- 3). `val()` 设置或者返回表单字段的值

**举例:**

```
<head>
  [html] view plain copy
  <meta charset="UTF-8"/>
  <script src="jquery.js"></script>
  <script>
    $(document).ready(function() {
      $("#btn1").click(function() {
        $("#test1").text("<b>Hello world!<b>");
      });
      $("#btn2").click(function() {
        $("#test2").html("<b>Hello world!</b>");
      });
      $("#btn3").click(function() {
        $("#test3").val("Dolly Duck");
      });
    });
  </script>
```

```

</head>
<body>
<p id="test1">这是段落。</p>
<p id="test2">这是另一个段落。</p>
<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p>
<button id="btn1">设置文本</button>
<button id="btn2">设置 HTML</button>
<button id="btn3">设置值</button>
</body>

```

在点击按钮的效果：

这是段落。

这是另一个段落。

Input field:

点击了 3 个 button 后的图如下，可以看出 html() 和 text() 的区别，html() 会识别出内容中的标签，而 text() 则是是什么就是什么，一切保持原汁原味。

**<b>Hello world!<b>**

Hello world!

Input field:  csdn.net/

## 2. 通过 attribute 获取和设置属性值

获取属性的值：

```
getAttribute(属性名) oText.getAttribute('value')
```

设置属性的值：

```
setAttribute(属性名, 要设置的值) oText.setAttribute('value', 'hello')
```

删除：

```
removeAttribute(属性名) oText.removeAttribute('value')
```

举例：实现获取连接的“target”属性值

代码如下：

```
请阅读 <a href="www.f-z.cn" target="_blank">前端大学</a>，  
<p id="demo">请点击按钮来显示上面这个链接的 target 属性值。</p>  
<button onclick="myFunction()">试一下</button>  
<script>  
    function myFunction()  
    {  
        var a=document.getElementsByTagName("a")[0];  
  
document.getElementById("demo").innerHTML=a.getAttribute("target");  
    }  
</script>
```

效果：



# 操作 DOM 对象的 CSS

## 1. 获取 css 样式

我们可以通过 `document.write()` 来获取元素的 css。

举例：获取 css 的值

代码如下：

```
<head>
  <title></title>
  <meta charset="utf-8">
  <script type="text/javascript">
    function change() {
      var e = document.getElementById("fenzhen");
      document.write("字体的颜色为：" + e.style.color + "<br/>");
      document.write("字体的背景颜色为："
+ e.style.backgroundColor + "<br/>");
      document.write("字体的外边距为：" + e.style.margin + "<br/>");
    }
  </script>
</head>
<body>
  <h1 id="fenzhen"
style="color:blue;background-color:gray;margin:20px">绿叶学习网</h1>
  <input type="button" value="改变样式" onclick="change()" />
</body>
```

效果：



字体的颜色为：blue  
字体的背景颜色为：gray  
字体的外边距为：20px

点击后

## 2. 设置 css 的值

举例：设置字体大小及边框

代码如下：

```
<script type="text/javascript">  
    function change() {  
        var e = document.getElementById("fenzhen");  
        e.style.color = "blue";  
        e.style.backgroundColor = "gray";  
        e.style.margin="20px";  
    }  
</script>
```

效果：

分针网

改变样式

点击前

分针网

改变样式

点击后

# 创建 DOM 节点

## 1. createElement()

**功能：**创建元素节点。此方法可返回一个 Element 对象

**语法：**


```
document.createElement(tagName)
```

**注意：**

tagName：字符串值，这个字符串用来指明创建元素的类型 举例：实现创建按钮代码如下：

```
<script type="text/javascript">
    var body = document.body;
    var input = document.createElement("input");
    input.type = "button";
    input.value = "这是一个新创建的按钮";
    body.appendChild(input);
</script>
```

**效果：**



## 2. createTextNode()

**功能：**创建新的文本节点，返回新创建的 Text 节点

**语法：**

```
document.createTextNode(data)
```

**提示：**

data：字符串值，可规定此节点的文本 举例：创建文本节点代码如下：

```
<body>
    <p id="demo">单击按钮创建文本节点。</p>
    <button onclick="myFunction()">点我</button>
    <script>
        function myFunction() {
            var t=document.createTextNode("Hello World");
            document.body.appendChild(t);
        };
    </script>
```



```
</body>
```

效果：



### 3. 创建节点的 HTML

我们可以使用 `innerHTML` 来创建节点的 HTML, 并且将创建的 HTML 插入到节点中。

举例：实现创建 `p` 标签跟 `span` 标签, 并且插入到 `div` 标签中

代码如下：

```
<body style="background-color:orange">
  <div id="qianduan"></div>
  <script type="text/javascript">
    var e = document.getElementById("qianduan");
    e.innerHTML="<span style='color:red;font-weight:bold;'>前端大
学 微信公众号</span><p
style='color:blue;font-weight:bold;'>JavaScript 入门教程</p>";
  </script>
</body>
```

效果：



# 节点的操作

## 1. removeChild()

功能：从子节点列表中删除某个节点

语法：

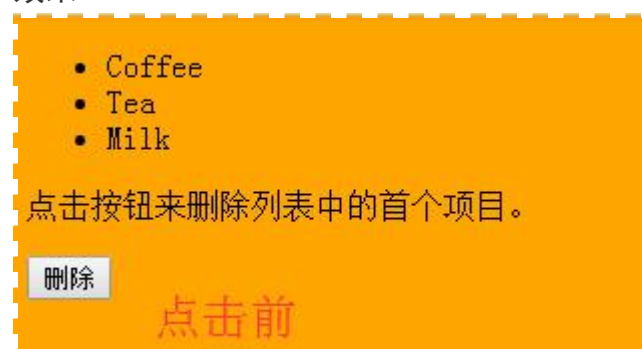
```
nodeObject.removeChild(node)
```

说明：

- 1). 如删除成功，此方法可返回被删除的节点，如失败，则返回 NULL。
- 2). node : 必需，指定需要删除的节点 举例：实现删除节点代码如下：

```
<body style="background-color:orange">
  <ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>
  <p id="demo">点击按钮来删除列表中的首个项目。</p>
  <button onclick="myFunction()">删除</button>
  <script>
    function myFunction()
    {
      var list=document.getElementById("myList");
      list.removeChild(list.childNodes[0]);
    }
  </script>
</body>
```

效果：



- Tea
- Milk

点击按钮来删除列表中的首个项目。

删除

点击后

## 2. replaceChild

**功能：** 实现子节点(对象)的替换。返回被替换对象的引用

**语法：**

```
node.replaceChild (newnode,oldnew )
```

**说明：**

1). newnode : 必需，用于替换 oldnew 的对象。

2). oldnew : 必需，被 newnode 替换的对象 举例：实现替换代码如下：

```
<p>单击按钮将段落中“Microsoft”替换成“fenzhen”：</p>
<p id="demo">Visit Microsoft!</p>
<button onclick="myFunction()">替换</button>
<script>
function myFunction() {
    var str=document.getElementById("demo").innerHTML;
    var n=str.replace("Microsoft","fenzhen");
    document.getElementById("demo").innerHTML=n;
}
</script>
```

**效果：**

单击按钮将段落中“Microsoft”替换成“fenzhen”：

Visit Microsoft!

替换

点击前



### 3. 复制节点

`cloneNode()` 方法创建指定节点的副本。

`cloneNode()` 方法有一个参数 (`true` 或 `false`)。该参数指示被克隆的节点是否包括原节点的所有属性和子节点。

举例：实现复制 `ul` 的所有节点

代码如下：

```
<head>
  <title></title>
  <meta name="twitter:" content="" charset="utf-8">
  <script type="text/javascript">
    function add() {
      var e = document.getElementById("list");
      document.body.appendChild(e.cloneNode(1));
    }
  </script>
</head>
<body style="background-color:orange">
  <ul id="list">
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
    <li>jQuery</li>
    <li>ASP.NET</li>
  </ul>
  <input type="button" value="添加" onclick="add()" />
</body>
```

效果：



## 练习题

1. 创建个 script 标签，并且插入 head 标签的末尾，引入自己的一个外部 js 文件，js 文件的代码如下：

```
console.log("引入了外部 js")
```

2. 对如下代码，做以下操作

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>dom 操作</title>
</head>
<body>
  <h1 id="ID1" style="color:red">标题 1</h1>
  <h1 id="ID2" style="font-size:100px;">fenzhen</h1>
```

```

<input type="button" value="提示" />
</body>
</html>
```

- 1). 输出 h1 标签的 html，html 标签的字体颜色，字号大小
- 2). 把 h2 的内容改成 h1 的内容
- 3). 在 body 标签里面添加一个 div 标签，div 标签的内容为 '

fenzhen</p>'
- 4). 把上面添加的 div 的字体颜色变成红色，字号 15px
- 5). 把 img 的图片变成分针的 logo
- 6). 把 input 的值变成：“危险提示”

## Window 对象和 document 对象

### Window 对象

#### Window 对象简介

Window 对象是 **JavaScript** 层级中的顶层对象。

Window 对象表示一个浏览器窗口或一个框架，它在<body>或<frameset>出现时被自动创建。

Window 对象是全局对象，它的属性可作为全局变量来使用，它的方法可当作函数来使用，也就是说，引用 Window 对象的属性和方法时，可以省略对象名。如：使用 document 属性不必写作 window.document，使用 alert() 方法不必写作 window.alert()。

Window 对象的集合:window.frames[]

**功能：**frames[]是窗口中所有命名的框架组成的数组。这个数组的每个元素都是一个 Window 对象，对应于窗口中的一个框架。

**语法：**

```
window.frames
```

浏览器窗口中每个<frameset>和<iframe>定义的框架都是 frames[] 数组中的元素。

frames.length 属性存放了 frames[] 数组的长度。

**注意：**

frames[] 数组的元素内部还可以有框架，所以 frames[i] 可以有自己的 frames[] 数组。

**Window 对象的属性**

**功能：**是对当前窗口自身的引用。它和 window 属性是等价的。

**语法：**

```
window.self
```

**注意：**

window、self、window.self 是等价的。

## Window 窗口的打开与关闭

### 1. open()

**功能：**打开一个新窗口。

**语法：**

```
window.open(URL, 窗口名称, 参数);
```

**注意：**

1). **URL：**指的是打开窗口的地址，如果 URL 为空字符串，则浏览器打开一个空白窗口，并且可以使用 document.write() 方法动态输出 HTML 文档。

2). **窗口名称：**指的是 window 对象的名称，可以是 a 标签或 form 标签中 target 属性值。如果指定的名称是一个已经存在的窗口名称，则返回对该窗口的引用，而不会再新打开一个窗口。

3). **参数**：对打开的窗口进行属性设置。

参数及其用法：

- 1). top 窗口顶部距离屏幕顶部的距离
  - 2). left 窗口左边距离屏幕左边的距离
  - 3). width 窗口的宽度（浏览器默认单位为 px）
  - 4). height 窗口的高度（浏览器默认单位为 px）
- 举例 1. 实现打开一个新窗口，并且在新窗口加载分针学习网首页代码如下：

```
window.open("http://www.f-z.cn", "", "");
```

举例 2：打开一个指定位置的窗口代码如下：

```
window.open("http://www.f-z.cn ", "", "top=200, left=200");
```

举例 3：打开一个指定大小的窗口代码如下：

```
window.open("http://www.f-z.cn ", "", "width=200, height=200");
```

**举例：** 实现打开新窗口分针官网首页宽为 500px，高为 400px 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <meta charset="utf-8">
  <script type="text/javascript">
    function openWindow() {
      window.open("http://www.f-z.cn ", "", "width=500, height=400");
    }
  </script>
</head>
<body>
  <input id="btn" type="button" value="打开窗口"
  onclick="openWindow()" />
</body>
</html>
```

效果：







## 2. close()

功能：关闭窗口

### 1). 关闭当前窗口

格式 1:

```
window.close();
```

格式 2:

```
close();
```

格式 3:

```
this.close();
```

### 2). 关闭子窗口格式:

```
窗口名.close();
```

注意:

- 1). 使用 window.open() 方法动态创建的窗口时，我们可以将窗口以变量形式保存，然后再使用 close() 方法关闭动态创建的窗口
- 2). 所谓的“关闭子窗口”就是关闭之前使用 window.open() 方法动态创建的子窗口。

举例：实习关闭子窗口。

代码如下:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
```

```

<head>
  <title></title>
  <script type="text/javascript">
    var newWindow =
window.open("http://www.f-z.cn", "", "width=500,height=400");
    function closeWindow()
    {
      newWindow.close();
    }
  </script>
</head>
<body>
  <input type="button" value="关闭窗口" onclick="closeWindow()" />
</body>
</html>

```

效果： 当打开网页的时候会自动打开分针网的官网；点击关闭窗口可以关闭打开的窗口。

## 返回文档的宽度与高度

### 1. innerheight

**功能：**返回窗口的文档显示区的高度。

### 2. innerwidth

**功能：**返回窗口的文档显示区的宽度。

**注意：**使用 outwidth 和 outheight 属性获取加上工具条与滚动条窗口的宽度与高度。

**语法：**

#### 1). 获取高度和宽度

```

window.innerWidth
window.innerHeight

```

## 2). 设置高度和宽度:

```
window.innerWidth=pixels  
window.innerHeight=pixels
```

举例：实现获取文档的宽度和高度

代码如下：

```
<script>  
function myFunction() {  
    var w=window.innerWidth;  
    var h=window.innerHeight;  
    x=document.getElementById("demo");  
    x.innerHTML="Width: " + w + " Height: " + h;  
}  
</script>
```

效果：



# location 对象的属性及方法

## 1. Location 对象

Location 对象包含有关当前 URL 的信息。

Location 对象是 window 对象的一部分, 可通过 window.Location 属性对其进行访问。

**注意：** 没有应用于 Location 对象的公开标准，不过所有浏览器都支持该对象。

## 2. Location 对象属性

属性	描述
----	----

hash	返回一个 URL 的锚部分
host	返回一个 URL 的主机名和端口
hostname	返回 URL 的主机名
href	返回完整的 URL
pathname	返回的 URL 路径名。
port	返回一个 URL 服务器使用的端口号
protocol	返回一个 URL 协议
search	返回一个 URL 的查询部分

### 3. Location 对象方法

#### 功能：

reload() 方法用于刷新当前文档。

reload() 方法类似于你浏览器上的刷新页面按钮。

如果把该方法的参数设置为 true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档。这与用户在单击浏览器的刷新按钮时按住 Shift 键的效果是完全一样。

#### 语法：

```
location.reload(forceGet)
```

#### 说明：

参数	类型	描述
forceGet	Boolean	可选。如果把该方法的参数设置为 true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档。

举例：实现页面的重新加载

代码如下：

```
<head>
<meta charset="utf-8">
<title>分针网</title>
<script>
    function reloadPage() {
```

```
    location.reload()
  }
</script>
</head>
<body>
  <input type="text">
<input type="button" value="重新加载页面" onclick="reloadPage()">
</body>
```

效果：



解析：

第一步是原来的页面；

第二步是在输入框的中输入内容“www. f-z. cn”；

第三步是点击“重新加载页面”后得到的原来的页面。

## document 对象

## document 对象简介

document 对象是文档的根节点，window.document 属性就指向这个对象。也就是说，只要浏览器开始载入 HTML 文档，这个对象就开始存在了，可以直接调用。

document.childNodes 属性返回该对象的所有子节点。对于 HTML 文档来说，document 对象一般有两个子节点。

第一个子节点是 document.doctype，表示文档类型节点（DocumentType）。对于 HTML5 文档来说，该节点就代表：

```
<!DOCTYPE html>
```

第二个子节点是 document.documentElement，表示元素节点（Element），代表：

```
<html lang="en">
```

document 对象的属性：

属性	描述
document.title	设置文档标题等价于 HTML 的<title>标签
document.bgColor	设置页面背景色
document.linkColor	未点击过的链接颜色
document.alinkColor	激活链接(焦点在此链接上)的颜色
document.fgColor	设置前景色(文本颜色)
document.vlinkColor	已点击过的链接颜色
document.URL	设置 URL 属性从而在同一窗口打开另一网页
document.fileCreatedDate	文件建立日期，只读属性
document.fileModifiedDate	文件修改日期，只读属性
document.fileSize	文件大小，只读属性
document.cookie	设置和读出 cookie
document.charset	设置字符集 简体中文:gb2312

# 常用的内置函数

本章节继续讲解一些 JavaScript 内置的全局函数。 1. **eval()** 函数

**功能：**eval() 函数计算 JavaScript 字符串，并把它作为脚本代码来执行。

如果参数是一个表达式，eval() 函数将执行表达式。

如果参数是 Javascript 语句，eval() 将执行 Javascript 语句。

**语法：**

```
eval(string)
```

**注意：**

string 是必需的，是要计算的字符串，其中含有要计算的 JavaScript 表达式或要执行的语句。

举例：通过 eval() 函数来执行一些代码

代码如下：

```
<script>
    eval("var x=10;var y=20;document.write(x*y)");
    document.write("<br>" + eval("2+2"));
    document.write("<br>" + eval(x+17));
    document.write("<br>" + eval());
</script>
```

效果：



200  
4  
27  
undefined

**说明：**

如果 eval() 函数没有参数的话，返回 undefined。

## 2. 编解码函数

用于将特殊字符串进行编码以及对边忙过的字符串进行解码。根据遵循的编码规则可以分为两种。

### 1). 遵循 ISO8859-1 规则的编解码

**encodeURIComponent() 函数：**把字符串作为 URI 进行编码。

**decodeURI() 函数：**对 encodeURIComponent() 函数编码过的 URI 进行解码。

语法:

```
encodeURIComponent(uri)
```

说明:

uri 是必需的, 表示一个字符串, 含有 URI 或其他要编码的文本。

语法:

```
decodeURI(uri)
```

说明:

uri 是必需的, 表示一个字符串, 含有要解码的 URI 或其他要解码的文本。

举例: 用 encodeURIComponent() 函数来编码字符串 "Hello world!", 然后用 decodeURI() 函数来解码。

代码如下:

```
<script>
  var str="Hello world!";
  document.write(str+"<br>");
  var ustr=encodeURIComponent(str);
  document.write(ustr+"<br>");
  document.write(decodeURI(ustr)+"<br>");
</script>
```

效果:

```
Hello world!
Hello%20world!
Hello world!
```

## 2). 遵循 Unicode 规则的编解码

**escape() 函数:** 可对字符串进行编码, 这样就可以在所有的计算机上读取该字符串。

注意:

- 1). 该方法不会对 ASCII 字母和数字进行编码, 也不会对下面这些 ASCII 标点符号进行编码: \* @ - \_ + . / 。其他所有的字符都会被转义序列替换。
- 2). 双字节字符被替换成十六进制的转义序列, 中文范围: %u4e00--%u9fa5 。

**unescape() 函数:** 可对通过 escape() 编码的字符串进行解码。

语法:

```
escape(string)
```

说明:

string 是必需参数, 表示要被转义或编码的字符串。



语法：

```
unescape(string)
```

说明：

string 是必需参数，表示要解码的字符串。

举例：用 escape() 函数来编码字符串 "Hello world!"，然后用 unescape() 函数来解码。

代码如下：

```
<script>
var str="Hello world!";
document.write(str+"<br>");
var ustr=escape(str);
document.write(ustr+"<br>");
document.write(unescape(ustr)+"<br>");
</script>
```

效果：

```
Hello world!
Hello%20world%21
Hello world!
```

总结：

eval() 函数比较少用到。

编码和解码函数在 AJAX 技术中很有用。

更多学习全套视频资料请关注微信公众号“**前端大学**”  
免费自动  
下载！

打开微信扫一扫：



# 定时器

## 1. 执行一次的定时器

### 1). `setTimeout()`

功能：用于在指定的毫秒数后调用函数或计算表达式

语法：

```
setTimeout (code, millisec)
```

注意：

- 1). `code` 必需。要调用的函数后要执行的 JavaScript 代码串
- 2). `millisec` 必需。在执行代码前需等待的毫秒数
- 3). `setTimeout()` 只执行 `code` 一次 举例：实现五秒后出现提示框代码如下：

```
<head>
  <meta charset="utf-8">
  <script type="text/javascript">
    function timedMsg()
    {
      var t=setTimeout("alert(' 分钟： www.f-z.cn ')", 5000)
    }
  </script>
</head>
<body>
  <form>
    <input type="button" value="点我，五秒后会出现提示框哦"
      onClick="timedMsg()" />
  </form>
</body>
```

效果：

点我，五秒后会出现提示框哦

第一步



## 2). clearTimeout()

功能：可取消由 setTimeout() 方法设置的 timeout

语法：

```
clearTimeout(id_of_settimeout)
```

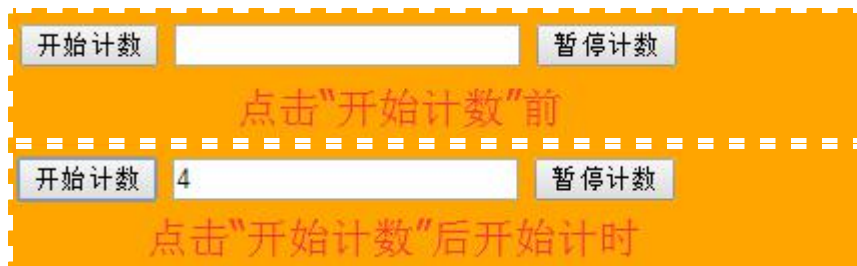
注意：

- 1). id\_of\_settimeout 由 setTimeout() 返回的 ID 值
- 2). 该值标识要取消的延迟执行代码块 举例：实现开始/暂停计数。

代码如下：

```
<head>
  <meta charset="utf-8">
  <script type="text/javascript">
var c=0
var t
function timedCount() {
  document.getElementById('txt').value=c
  c=c+1
  t=setTimeout("timedCount()",1000)
}
function stopCount() {
  clearTimeout(t)
}
  </script>
</head>
<body>
  <form>
    <input type="button" value="开始计数" onClick="timedCount()">
    <input type="text" id="txt">
    <input type="button" value="暂停计数" onClick="stopCount()">
  </form>
</body>
```

效果：



解析：

此后如果没有点击“暂停计数”则一直计数，点击“暂停计数”后仍然可以再点击“开始计数”继续计数。

## 2. 重复执行的定时器

### 1). `setInterval()`

功能：可按照指定的周期（以毫秒计）来调用函数或计算表达式。

语法：

```
setInterval (code, millisec)
```

注意：

- 1). `ode` 必需。要调用的函数或要执行的代码串。
- 2). `millisec` 必需。周期性执行或调用 `code` 之间的时间间隔，以毫秒计。
- 3). `setInterval()` 方法会不停地调用函数，直到 `clearInterval()` 被调用或窗口被关闭。
- 4). 由 `setInterval()` 返回的 ID 值可用作 `clearInterval()` 方法的参数

### `clearInterval()`

功能：可取消由 `setInterval()` 设置的 `timeout`。

语法：

```
clearInterval(id_of_setinterval)
```

注意：

`id_of_setinterval` 由 `setInterval()` 返回的 ID 值。 举例：实现停在获取系统的时间

代码如下：

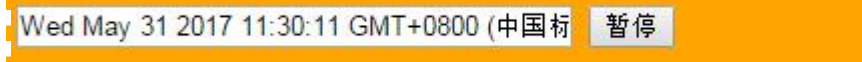
```
<head>
  <meta charset="utf-8">
  <input type="text" id="clock" size="35" />
  <script language=javascript>
    var int=self.setInterval("clock()",50)
```

```

function clock() {
    var t=new Date()
    document.getElementById("clock").value=t
}
</script>
</head>
<body>
    <button onclick="int=window.clearInterval(int)"> 暂停</button>
</body>

```

效果：



解析：

获取了系统的时间，点击暂停后计时会停止

## 弹出框

### 1. alert()

功能：消息警告框格式：

```
alert ( ) ;
```

注意：

alert 方法有一个参数，即希望对用户显示的文本字符串。该字符串不是 HTML 格式。该消息框提供了一个“确定”按钮让用户关闭该消息框，并且该消息框是模式对话框，也就是说，用户必须先关闭该消息框然后才能继续进行操作。 举例：实现进入网站弹出“分针是一个不错的网站！请按“确定”继续”的提示框。

代码如下：

```

<head>
    <meta charset="utf-8">
    <script type="text/javascript">
function display_alert() {
    alert("分针是一个不错的网站！请按“确定”继续")
}
</script>

```

```
</head>
<body style="background-color:orange">
<input type="button" onclick="display_alert()" value="点我" />
</body>
```

效果：



## 2. confirm()

功能：确认消息框格式：

**confirm**(提示信息)

注意：

- 1). message: 要在 window 上弹出的对话框中显示的纯文本，而非 HTML 内容
- 2). confirm 方法的返回值为 true 或 false
- 3). 该消息框也是模式对话框：用户必须在响应该对话框（单击一个按钮）将其关闭后，才能进行下一步操作。举例：实现点击“确定”则继续，点击“取消”则弹出警告框。

代码如下：

```
<script type="text/javascript">
var cfm= window.confirm("单击“确定”继续。单击“取消”停止。");
if (cfm) {
window.alert("欢迎访问我们分针的 Web 页！")
} else window.alert("下次见啦！");
</script>
```

效果：



### 解析：

图一是一进入网页弹出的，图二是点击“确定”弹出的提示框，图三是点击“取消”弹出的提示框

### 3. prompt()

功能：提示消息框格式：

```
prompt("message");
```

### 注意：

提示消息框提供了一个文本字段，用户可以在此字段输入一个答案来响应您的提示

该消息框有一个“确定”按钮和一个“取消”按钮。默认文本为 "<undefined>"。

与 alert() 和 confirm() 方法类似，prompt 方法也将显示一个模式消息框。

用户在继续操作之前必须先关闭该消息框 举例：实现进入网页时弹出可输入文本的提示框。

代码如下：

```
<head>  
<meta charset="utf-8">
```

```
<script type="text/javascript">
function disp_prompt()
{
    var name=prompt("Please enter your name","")
    if (name!=null && name!="")
    {
        document.write("Hello " + name + " !")
    }
}
</script>
</head>
<body style="background-color:orange">
<input type="button" onclick="disp_prompt()" value="Display a prompt
box" />
</body>
```

效果:



## 文档操作

### 1. 标题操作



定义：

title 属性可返回当前文档的标题（ HTML title 元素中的文本）。

语法：

```
document.title
```

举例：实现获取网页的标题

```
The title of the document is:
<script type="text/javascript">
document.write(document.title)
</script>
```

效果：

```
The title of the document is: 分针网
```

## 2. 获取 URL

定义：

URL 属性可返回当前文档的 URL。

语法：

```
document.URL
```

说明：

一般情况下，该属性的值与包含文档的 Window 的 location.href 属性相同。不过，在 URL 重定向发生的时候，这个 URL 属性保存了文档的实际 URL，而 location.href 保存了请求的 URL

举例：实现获取当前的 URL

代码如下：

```
The URL of this document is:
<script type="text/javascript">
document.write(document.URL)
</script>
```

效果：

```
The URL of this document is: file:///C:/Users/Administrator/Desktop/try/html/%E6%96%B0%E5%BB%BA%E6%96%87%E6%91%AC%E6%96%87%E6%A1%A3.h
```

## 练习题

1. 写一个函数，当页面加载的时候，每隔 1 秒打印当前的时间
2. url:  
`http://www.f-z.cn/web/platform.php?id=%E5%88%86%E9%92%88&name=!%E5%8A%AA%E5%8A%9B%E5%81%9A%E5%A5%BD%E6%95%99%E7%A8%8B` 打印该 url 的 host, port, protocol, path 以及 name 和 id 的值
3. 弹出框提示“是否要前往分针网”，如果用户点击确定，让当前页面 5 秒后跳转到“`http://www.f-z.cn`”，如果取消的弹出提示“您选择留在当前页面”
4. 用户访问当前页面的时候，弹出对话框，让用户输入名字，然后用户点击确定后，打印用户输入的名字

更多学习全套视频资料请关注微信公众号“**前端大学**”免费自动下载！

打开微信扫一扫：



## Js 事件

### 什么是事件

1. 什么是事件？

JavaScript 创建动态页面。事件是可以被 JavaScript 侦测到的行为。  
网页中的每个元素都可以产生某些可以触发 JavaScript 函数的事件。  
比方说, 我们可以在用户点击某按钮时产生一个 `onClick` 事件来触发某个函数。  
事件可以是浏览器行为, 也可以是用户行为。

## 2. 事件的分类

JavaScript 的事件有很多, 可大致分为 5 大类。

- 1). 鼠标事件
- 2). 键盘事件
- 3). 表单事件
- 4). 定时器

在本课程中只讲解一些常用的事件。

**注意:** 事件通常与函数一起使用, 当事件发生时执行函数。

# 鼠标事件

鼠标事件有:

在本章节中只讲解几个常用的鼠标事件。

属性	描述
<code>onclick</code>	当用户点击某个对象时调用的事件句柄
<code>oncontextmenu</code>	在用户点击鼠标右键打开上下文菜单是触发
<code>ondblclick</code>	当用户双击某个对象时调用事件句柄
<code>onmousedown</code>	鼠标按钮被按下
<code>onmouseenter</code>	当鼠标指针移动到元素上时触发
<code>onmouseleave</code>	鼠标指针移出元素是触发
<code>onmousemove</code>	鼠标被移动
<code>onmouseover</code>	鼠标移到某元素之上

onmouseout	鼠标从某元素移开
onmouseup	鼠标按键被松开

下面我们就来对常用的三个事件进行详细的讲解：

## 1. 鼠标点击事件

常用的鼠标点击事件有 onclick、onmousedown 和 onmouseup 这三个，鼠标双击事件比较少用到。三个常用的鼠标点击事件的执行顺序是

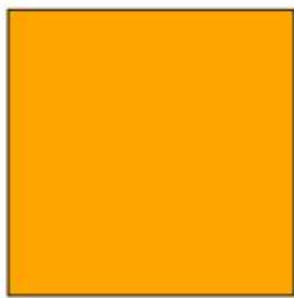
onmousedown→onmouseup→onclick。如果只是在元素上按了鼠标，但是不松开按键，onclick 和 onmouseup 都不会被触发。

举例：为一个 div 添加鼠标点击事件、鼠标按键按下事件、鼠标按键松开事件。

代码如下：

```
<div class="box" id="box"></div>
<script>
var box=document.getElementById('box');
box.onmousedown=mousedown;
box.onclick=click;
box.onmouseup=mouseup;
function mousedown() {
    this.innerHTML=this.innerHTML+"触发鼠标按下事件<br>";
}
function click() {
    this.innerHTML=this.innerHTML+"触发鼠标点击事件<br><br>";
}
function mouseup() {
    this.innerHTML=this.innerHTML+"触发鼠标松开事件<br>";
}
</script>
```

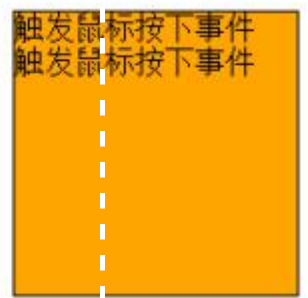
效果：



初始状态



鼠标点击并松开



鼠标按下，不松开

说明：

从图中可以看出鼠标点击时的执行顺序，以及如果不松开鼠标按键是否会触发事件。

## 2. 鼠标移入移出事件

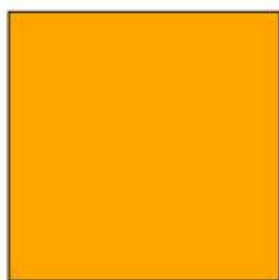
鼠标的移入移出事件分别是 `mouseover` 和 `mouseout`。这两个通常是连在一起使用的。

举例：通过鼠标移入、移出事件向元素中添加内容。

代码如下：

```
<div class="box" id="box"></div>
<script>
var box=document.getElementById('box');
box.onmouseover=mouseover;
box.onmouseout=mouseout;
function mouseover() {
    this.innerHTML=this.innerHTML+"触发鼠标移入事件<br>";
}
function mouseout() {
    this.innerHTML=this.innerHTML+"触发鼠标移出事件<br><br>";
}
</script>
```

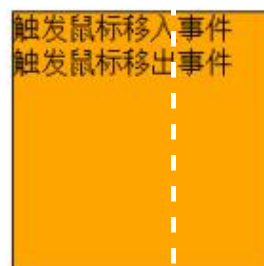
效果：



初始状态



鼠标移入 div 但不离开



鼠标移入 div，再移出 div

说明：

鼠标移入移出事件通常都是一起使用的。可以通过该事件实现放大镜效果。

总结：

- 1). 鼠标事件有 10 个，常用的就是鼠标点击事件和鼠标移入移出事件。
- 2). 鼠标事件中的某些事件通常都是一起使用的。

## 键盘事件

JavaScript 中的键盘事件只有三个。

属性	描述
onkeydown	某个键盘按键被按下
onkeypress	某个键盘按键被按下并松开
onkeyup	某个键盘事件松开

这三个键盘事件的执行顺序是： onkeydown → onkeypress → onkeyup。

### 1. onkeydown 事件

当键盘按键被按下时触发。

举例：为一个输入框设定 onkeydown 事件。

代码如下：

```

<div class="box">
<p>在输入框中按下按键触发</p>
<input type="text" id="input" onkeydown="keydown()">
</div>
<script>
function keydown() {
    alert("你在输入框中按下了一个键");
}
</script>

```

效果：



初始状态



在输入框中按下任意键

说明：

在输入框中按下任意键，都会调用函数，函数内执行弹出警告框的代码。

## 2. onkeypress 事件

onkeypress 事件只在按下键盘的任一“字符键”（如 A~Z、数字键）时触发，单独按下“功能键”（如 F1~F12、Ctrl 键、Shift 键、Alt 键等）不会触发。

举例：通过 onkeypress 事件弹出警告框。

代码如下：

```

<div class="box">
<p>在输入框中按下按键触发</p>
<input type="text" id="input" onkeypress="keypress()">
</div>
<script>
function keypress() {
    alert("触发 onkeypress 事件");
}
</script>

```

效果：



初始状态



按下字符按键

说明：

如果按下功能键，则没有任何效果。

### 3. onkeyup 事件

在 JavaScript 中，onkeyup 事件是在键盘的某个键被按下之后松开的一瞬间触发的事件。

举例：通过 onkeyup 事件改变元素的内容。

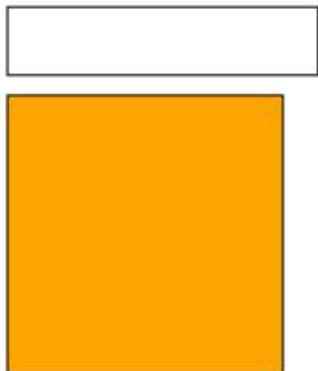
代码如下：

```
<div class="box">
<p>在输入框中输入内容：</p>
<input type="text" id="input" onkeyup="keyup()">
<div id="box"></div>
</div>
<script>
var box=document.getElementById('box');
var input=document.getElementById('input');
function keyup() {
    box.innerHTML=input.value;
}
</script>
```

效果：

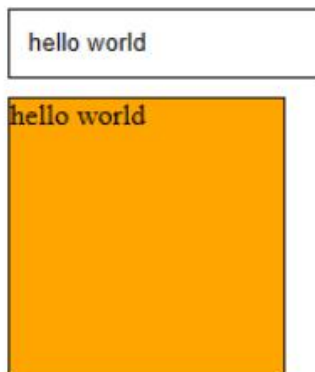


在输入框中输入：



初始状态

在输入框中输入：



输入普通文本

在输入框中输入：



输入含标签的文本

说明：

- 1). onkeyup 会在键盘按键松开的时候触发。
- 2). 用 onkeyup 提取 value 的话,可以提取到已经输入的内容,如果是 onkeydown 和 onkeypress 都只会提取到按下按键之前的内容。

总结：

JavaScript 的键盘事件只有 3 个。根据不同的情况使用。

更多学习全套视频资料请关注微信公众号“**前端大学**”  
免费自动  
下载！

打开微信扫一扫：



# 表单事件

表单事件有：

属性	描述
onblur	元素失去焦点是触发
onchange	该事件在表单元素的内容是触发（<input>,<keygen>,<select>和<textarea>）
onfocus	元素获取焦点是触发
onfocusin	元素即将获取焦点是触发
onfocusout	元素即将失去焦点是触发
oninput	元素获取用户输入时触发
onreset	表单重置是触发
onsearch	用户向搜索域输入文本是触发（<input="search">）
onselect	用户选取文本时触发（<input>和<textarea>）
onsubmit	表单提交时触发

常用的表单事件有 onchange 事件、onsubmit 事件、onblur 事件和 onfocus 事件。

## 1. onchange 事件：

- 1). onchange 事件会在域的内容改变时发生。
- 2). onchange 事件也可用于单选框与复选框改变后触发的事件。

onchange 事件的触发步骤：

- 1). 当 input 的获得焦点的时候，系统储存当前值
- 2). 当 input 失去焦点的时候，判断当前值是否和之前的储存值相同，如果不同则触发 onchange 事件。（非 IE 浏览器可以通过回车来判断）

举例：当输入框中的内容改变时，将输入的小写字母转换成大写字母。

代码如下：

```
<form>输入需要转换的字符串<input type="text" id="input"
onchange="func()"></form>
<p>当输入框失去焦点的时候，函数被触发，小写字母转换成大写字母</p>
<script>
function func() {
    var x=document.getElementById("input");
    console.log(x.value);
    x.value=x.value.toUpperCase();
    console.log(x.value);
}
</script>
```

效果：



说明：如果只是输入内容，并没有按回车键或者离开输入框，onchange 不会触发。

## 2. onsubmit 事件

- 1). onsubmit 属性只在 <form> 表单中使用。
- 2). onsubmit 事件是当表单提交时进行相关 js 操作的一个事件。
- 3). onsubmit 事件会在表单中的确认按钮被点击时发生。

当该事件触发的函数中返回 false 时，表单就不会被提交。

语法：

```
onsubmit="return 函数名()"
```

举例：事件触发的函数返回 false，看页面会不会跳转。

代码如下：

```
<form action="http://www.f-z.cn" onsubmit="return func()">
  <input type="text" >
  <input type="submit" value="提交">
</form>
<script>
function func() {
  alert("触发 onsubmit 事件");
  return false;
}
</script>
```

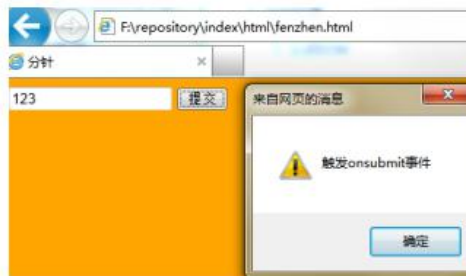
效果：

说明：

如果需要页面跳转的话，只需要将返回值改为 true 就可以了，当返回值是 false 时，表单不会被提交。

### 3. onfocus 事件和 onblur 事件

onfocus 事件在对象获得焦点时发生。onblur 事件刚好相反，在对象失去焦点时发生。



点击提交按钮



页面并没有跳转

举例：当输入框获得焦点的时候改变输入框的颜色，当输入框失去焦点的时候验证内容是否为空，并把输入框的颜色改成白色。

代码如下：

```
<form>
  <input id="input" type="text" onfocus="func(this)"
onblur="check(this)">
</form>
<script>
function func(x) {
  x.style.backgroundColor ="yellow";
}
function check(x) {
  x.style.backgroundColor ="#fff"
  if(x.value=="") {
    alert("内容不能为空");
  }
}
</script>
```

效果：



初始状态



点击输入框



输入内容为空，并且输入框失去焦点



输入框内容不为空

### 说明：

onblur 经常用于 Javascript 验证代码，一般用于表单输入框。

onfocus 通常用于 <input>，<select>，和<a>。

### 总结：

表单的事件有很多，常用的有 onchange 事件、onsubmit 事件、onblur 事件和 onfocus 事件。

更多学习全套视频资料请关注微信公众号“**前端大学**” 免费自动下载！

打开微信扫一扫：



## 浏览器事件

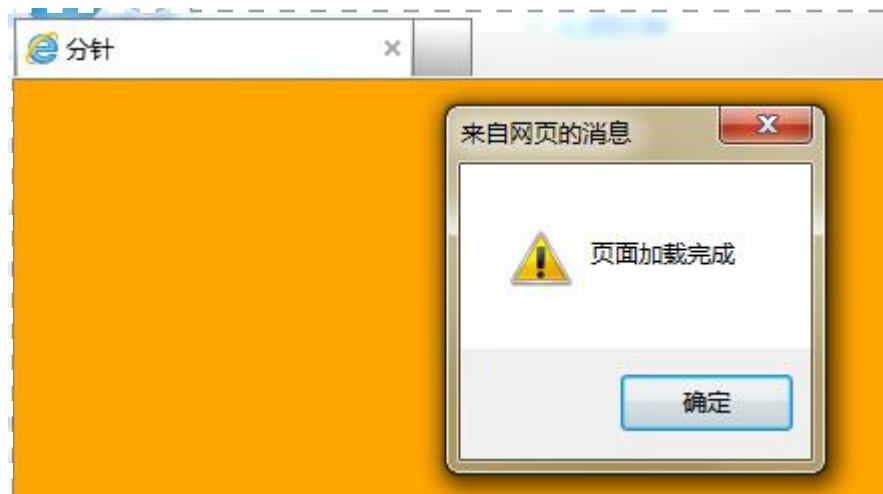
### 1. onload 事件

- 1). onload 事件会在页面或图像加载完成后立即发生。
- 2). onload 通常用于 <body> 元素，在页面完全载入后(包括图片、css 文件等等。)执行脚本代码。 举例：页面加载完成后弹出警告框。

代码如下：

```
<body onload="func()">
<script>
function func() {
    alert("页面加载完成");
}
</script>
</body>
```

效果：



说明：当 body 标签中的内容加载完成后调用函数名为 func 的函数。

## 2. onresize 事件

定义：

onresize 事件会在窗口或框架被调整大小时发生。

语法：

```
window.onresize=function() {SomeJavaScriptCode} ;
```

参数	描述
SomeJavaScriptCode	必需。规定该事件发生时执行的 JavaScript。

举例： 实现当浏览器被重置大小时执行 Javascript 代码：

代码如下：

```
<head>
<meta charset="utf-8">
<title>分针网</title>
<script>
function myFunction() {
    var w=window.outerWidth;
    var h=window.outerHeight;
    var txt="窗口大小: 宽度=" + w + ", 高度=" + h;
    document.getElementById("demo").innerHTML=txt;
}
</script>
</head>
<body onresize="myFunction()">
<p>尝试调整浏览器的窗口</p>
```



```
<p id="demo">&nbsp;</p>  
<p>注意:该例子在 IE8 或更早版本下可能不工作, IE8 或更早的版本不支持  
window 对象的 outerWidth/outerHeight 属性</p>  
</body>
```

效果:

尝试调整浏览器的窗口

窗口大小: 宽度=603, 高度=222

注意:该例子在IE8 或更早版本下可能不工作, IE8 或更早的版本不支持window对象的outerWidth/outerHeight属性

尝试调整浏览器的窗口

窗口大小: 宽度=683, 高度=232

注意:该例子在IE8 或更早版本下可能不工作, IE8 或更早的版本不支持window对象的outerWidth/outerHeight属性

解释:

(此教程中的图片进行了缩放) 当按住鼠标该表浏览器窗口大小的时候 js 代码宽度(w)跟高度(h)会自动改变.

福利来啦! 学前端不易, 好的前端视频教程对一个人的学习至关重要!!!

请看截图! 全套最新 “某智” 最新 web 前端开发培训全套视频教程+

源码+笔记+ppt (打包下载直接解压) (具体的内容请看截图!)

百度云下载链接 <http://pan.baidu.com/s/1gfFxqBh>

提取密码请关注 微信公众号 “**前端大学**” 回复 “**某智提取密码**” 6个字自动获取。



工具(I) 帮助(H)		
共享 ▾ 新建文件夹		
名称	修改日期	类型
28-ajax-第一天	2017/7/7 16:20	文件夹
29-ajax-第二天	2017/7/7 16:20	文件夹
30-ajax-第三天	2017/7/7 16:21	文件夹
31-ajax-第四天	2017/7/7 16:21	文件夹
32-ajax-第五天	2017/7/7 16:21	文件夹

。。。。等（咱们是一个自学者组成的联盟！拒绝任何形式的付费！

努力自学！）

某智 2016 年 11 月 web 前端开发培训全套视频教程+源码+笔记+ppt（打包下载直接解压）

包含从 0 基础入门到 javascript 初级入门，再到 javascript 深入学习，再到各个 javascript 框架的学习和综合运用，中间穿插着多个实例实战。。。快来关注下载吧！ 微信搜索公众号 “前端大学”，记住名字只有 “前端大学”

4 个字哦！

咱们的公众号里还有更多直接下载的资料，不定期更新哦！

## 前 端 大 学

分享前端相关技术干货·资讯·高薪职位·教程



长按识别二维码关注前端大学