目录

[1. 概述 6](#_Toc507320780)

[1.1. 基础知识 6](#_Toc507320781)

[1.1.1. WEB万维网（缩写www） 6](#_Toc507320782)

[1.1.2. 超文本 6](#_Toc507320783)

[1.1.3. 超媒体 6](#_Toc507320784)

[1.1.4. HTTP 6](#_Toc507320785)

[1.1.5. HTTPS 6](#_Toc507320786)

[1.1.6. FTP 6](#_Toc507320787)

[1.1.7. URL 7](#_Toc507320788)

[1.1.8. MIME 7](#_Toc507320789)

[1.1.9. 超链接 7](#_Toc507320790)

[1.1.10. 网站（website） 7](#_Toc507320791)

[1.1.11. 首页 7](#_Toc507320792)

[1.2. 开发技术与工具 8](#_Toc507320793)

[1.2.1. 系统的架构模式 8](#_Toc507320794)

[1.2.2. 浏览器 8](#_Toc507320795)

[1.2.3. 服务器 8](#_Toc507320796)

[1.2.4. 客户端应用技术 8](#_Toc507320797)

[1.2.5. 服务端技术 9](#_Toc507320798)

[1.3. Web技术发展概述 9](#_Toc507320799)

[1.3.1. Web技术发展的三个阶段 9](#_Toc507320800)

[1.3.2. Web2.0应用技术 9](#_Toc507320801)

[1.4. 学习方法 9](#_Toc507320802)

[2. HTML 10](#_Toc507320803)

[2.1. HTML概述 10](#_Toc507320804)

[2.1.1. HTML历史 10](#_Toc507320805)

[2.1.2. HTML概念 10](#_Toc507320806)

[2.1.3. HTML 标签 10](#_Toc507320807)

[2.1.4. HTML 元素 10](#_Toc507320808)

[2.1.5. HTML的特点 10](#_Toc507320809)

[2.1.6. 结构标记 11](#_Toc507320810)

[2.2. HTML基础知识 11](#_Toc507320811)

[2.2.1. 特殊标记 11](#_Toc507320812)

[2.2.2. 标题字标记 12](#_Toc507320813)

[2.2.3. 段落标记 12](#_Toc507320814)

[2.2.4. 字体标记 12](#_Toc507320815)

[2.2.5. 字体修饰标记 12](#_Toc507320816)

[2.2.6. 列表 12](#_Toc507320817)

[2.2.7. 相对路径与绝对路径 12](#_Toc507320818)

[2.2.8. 超链接 13](#_Toc507320819)

[2.2.9. 图片 14](#_Toc507320820)

[2.3. HTML表格 14](#_Toc507320821)

[2.3.1. 表格标记 14](#_Toc507320822)

[2.3.2. 表格的常用属性 14](#_Toc507320823)

[2.3.3. 表格的对齐方式 14](#_Toc507320824)

[2.3.4. 表格合并行合并列 15](#_Toc507320825)

[2.3.5. 表格的结构标记 15](#_Toc507320826)

[2.4. HTML表单 15](#_Toc507320827)

[2.4.1. 表单标记 15](#_Toc507320828)

[2.4.2. 表单元素input 16](#_Toc507320829)

[2.4.3. 下拉列表 17](#_Toc507320830)

[2.4.4. 文本区域标记 17](#_Toc507320831)

[2.5. 框架 18](#_Toc507320832)

[2.5.1. iframe 18](#_Toc507320833)

[2.5.2. frameset 18](#_Toc507320834)

[2.6. 其他知识 18](#_Toc507320835)

[2.6.1. 块级元素 18](#_Toc507320836)

[2.6.2. 内联元素 19](#_Toc507320837)

[2.6.3. div与span 19](#_Toc507320838)

[2.6.4. marquee 19](#_Toc507320839)

[2.7. 实验 19](#_Toc507320840)

[2.7.1. HTML网页标签练习 19](#_Toc507320841)

[2.7.2. 简单个性化网站设计 19](#_Toc507320842)

[2.7.3. 制作百度搜索界面 19](#_Toc507320843)

[2.7.4. 制作百度图库搜索界面 19](#_Toc507320844)

[3. CSS 21](#_Toc507320845)

[3.1. CSS语法 21](#_Toc507320846)

[3.1.1. CSS 21](#_Toc507320847)

[3.1.2. CSS声明 21](#_Toc507320848)

[3.1.3. CSS的使用方法 21](#_Toc507320849)

[3.2. CSS选择器 22](#_Toc507320850)

[3.2.1. 常用选择器 22](#_Toc507320851)

[3.2.2. 组合选择器 23](#_Toc507320852)

[3.2.3. 属性选择器 23](#_Toc507320853)

[3.2.4. 伪选择器 24](#_Toc507320854)

[3.2.5. 选择器优先级 25](#_Toc507320855)

[3.3. CSS属性 26](#_Toc507320856)

[3.3.1. 字体 26](#_Toc507320857)

[3.3.2. 文本 26](#_Toc507320858)

[3.3.3. 边框 26](#_Toc507320859)

[3.3.4. 背景 27](#_Toc507320860)

[3.3.5. 表格 27](#_Toc507320861)

[3.3.6. 列表 28](#_Toc507320862)

[3.4. CSS布局 28](#_Toc507320863)

[3.4.1. 盒子模型 28](#_Toc507320864)

[3.4.2. 定位 30](#_Toc507320865)

[3.4.3. 浮动 30](#_Toc507320866)

[3.4.4. display 31](#_Toc507320867)

[3.4.5. 布局居中 32](#_Toc507320868)

[3.5. CSS3 33](#_Toc507320869)

[3.5.1. 圆角 33](#_Toc507320870)

[3.5.2. 渐变/过渡效果 34](#_Toc507320871)

[3.5.3. 动画 35](#_Toc507320872)

[3.5.4. 2D&3D转换 36](#_Toc507320873)

[3.6. 实验 37](#_Toc507320874)

[3.6.1. CSS选择器以及样式练习 37](#_Toc507320875)

[3.6.2. 装饰个人网站 37](#_Toc507320876)

[3.6.3. 美化百度图库搜索 37](#_Toc507320877)

[4. JavaScript 38](#_Toc507320878)

[4.1. JavaScript语法基础 38](#_Toc507320879)

[4.1.1. JavaScript的特点 38](#_Toc507320880)

[4.1.2. JavaScript的功能 39](#_Toc507320881)

[4.1.3. JavaScript在HTML中的使用 39](#_Toc507320882)

[4.1.4. JavaScript语法 39](#_Toc507320883)

[4.1.5. JavaScript变量以及变量提升 40](#_Toc507320884)

[4.1.6. JavaScript数据类型 41](#_Toc507320885)

[4.1.7. JavaScript运算符及优先级 43](#_Toc507320886)

[4.1.8. JavaScript语句结构 43](#_Toc507320887)

[4.2. JavaScript常用对象及方法 43](#_Toc507320888)

[4.2.1. String 43](#_Toc507320889)

[4.2.2. Array 44](#_Toc507320890)

[4.2.3. Date 45](#_Toc507320891)

[4.2.4. Math 45](#_Toc507320892)

[4.2.5. RegExp 46](#_Toc507320893)

[4.3. JavaScript函数 48](#_Toc507320894)

[4.3.1. 函数的定义 48](#_Toc507320895)

[4.3.2. 函数参数与返回值 48](#_Toc507320896)

[4.3.3. 函数的调用 49](#_Toc507320897)

[4.3.4. 内置函数 49](#_Toc507320898)

[4.4. JavaScript事件 49](#_Toc507320899)

[4.4.1. JavaScript事件处理程序的调用 50](#_Toc507320900)

[4.4.2. JavaScript鼠标事件 50](#_Toc507320901)

[4.4.3. JavaScript键盘事件 50](#_Toc507320902)

[4.4.4. JavaScript页面事件 50](#_Toc507320903)

[4.4.5. JavaScript表单/编辑事件 51](#_Toc507320904)

[4.4.6. JavaScript事件模型 51](#_Toc507320905)

[4.4.7. JavaScript事件对象 51](#_Toc507320906)

[4.4.8. JavaScript事件监听 52](#_Toc507320907)

[4.5. JavaScript BOM & DOM 52](#_Toc507320908)

[4.5.1. window对象属性 53](#_Toc507320909)

[4.5.2. window对象方法 53](#_Toc507320910)

[4.5.3. document对象属性 54](#_Toc507320911)

[4.5.4. document对象方法 54](#_Toc507320912)

[4.5.5. element属性与方法 55](#_Toc507320913)

[4.5.6. attribute属性与方法 56](#_Toc507320914)

[4.5.7. DOM节点操作 56](#_Toc507320915)

[4.6. 实验 57](#_Toc507320916)

[4.6.1. 实现九九乘法表 57](#_Toc507320917)

[4.6.2. 逐行打印九九乘法表 57](#_Toc507320918)

[4.6.3. 实现开关灯 57](#_Toc507320919)

[4.6.4. 实现随机彩灯 57](#_Toc507320920)

[4.6.5. 网页背景色切换 57](#_Toc507320921)

[4.6.6. 实现网页计算器 57](#_Toc507320922)

[4.6.7. 金额大小写转换 58](#_Toc507320923)

[4.6.8. 制作个人相册 58](#_Toc507320924)

[5. Ajax 59](#_Toc507320925)

[5.1. Ajax传统编程 59](#_Toc507320926)

[5.1.1. Ajax概述 59](#_Toc507320927)

[5.1.2. Ajax优点 59](#_Toc507320928)

[5.1.3. Ajax的缺点 60](#_Toc507320929)

[5.1.4. XMLHttpRequest对象 60](#_Toc507320930)

[5.1.5. Ajax开发步骤 61](#_Toc507320931)

[5.2. Ajax与XML 62](#_Toc507320932)

[5.2.1. XML语法与作用 62](#_Toc507320933)

[5.2.2. 读取XML节点与属性 62](#_Toc507320934)

[5.3. 实验 63](#_Toc507320935)

[5.3.1. 异步/同步请求文本数据 63](#_Toc507320936)

[5.3.2. 异步/同步请求XML数据 63](#_Toc507320937)

[5.3.3. CD唱片清单 63](#_Toc507320938)

[5.3.4. 多级联动地址下拉列表 63](#_Toc507320939)

[6. HTML5 64](#_Toc507320940)

[6.1. HTML5新特性 64](#_Toc507320941)

[6.1.1. HTML5自定义元素 64](#_Toc507320942)

[6.1.2. HTML5语义元素 64](#_Toc507320943)

[6.1.3. HTML5废弃元素 65](#_Toc507320944)

[6.1.4. HTML5表单属性 65](#_Toc507320945)

[6.1.5. HTML5表单元素 66](#_Toc507320946)

[6.1.6. HTML5表单input控件 66](#_Toc507320947)

[6.2. Canvas 67](#_Toc507320948)

[6.2.1. Canvas绘图步骤 67](#_Toc507320949)

[6.2.2. context属性 67](#_Toc507320950)

[6.2.3. 绘制图形 68](#_Toc507320951)

[6.2.4. 绘制图像 69](#_Toc507320952)

[6.2.5. SVG 69](#_Toc507320953)

[6.3. 多媒体元素 70](#_Toc507320954)

[6.3.1. 多媒体标签的使用 70](#_Toc507320955)

[6.3.2. 常见属性 71](#_Toc507320956)

[6.3.3. 常用方法 72](#_Toc507320957)

[6.3.4. 常见事件 72](#_Toc507320958)

[6.4. 其他 73](#_Toc507320959)

[6.4.1. HTML5 location 73](#_Toc507320960)

[6.4.2. 本地存储 73](#_Toc507320961)

[6.4.3. 应用程序缓存（Application Cache） 74](#_Toc507320962)

[6.4.4. Web Workers 74](#_Toc507320963)

[6.5. 实验 74](#_Toc507320964)

[6.5.1. 制作一个新元素构成的表单 74](#_Toc507320965)

[6.5.2. 绘制一幅五星红旗 74](#_Toc507320966)

[6.5.3. 制作一个在线播放器 74](#_Toc507320967)

[7. jQuery 75](#_Toc507320968)

[7.1. jQuery基础 75](#_Toc507320969)

[7.1.1. 概述 75](#_Toc507320970)

[7.1.2. 选择器 75](#_Toc507320971)

[7.1.3. jQuery与HTML 79](#_Toc507320972)

[7.2. jQuery事件与动画 82](#_Toc507320973)

[7.2.1. jQuery事件 82](#_Toc507320974)

[7.2.2. jQuery动画 86](#_Toc507320975)

[7.3. JQuery与Ajax 88](#_Toc507320976)

[7.3.1. JSON 88](#_Toc507320977)

[7.3.2. 常用方法 88](#_Toc507320978)

[7.4. 实验 90](#_Toc507320979)

[7.4.1. 实现多种按钮点击事件 90](#_Toc507320980)

[7.4.2. jQuery动画 90](#_Toc507320981)

[7.4.3. jQuery与HTML 90](#_Toc507320982)

[8. 其他 91](#_Toc507320983)

[8.1. Bootstrap 91](#_Toc507320984)

[8.2. Easy UI 91](#_Toc507320985)

[8.3. node.js、vue.js 91](#_Toc507320986)

[8.4. 实验 91](#_Toc507320987)

[8.4.1. 企业门户网站 91](#_Toc507320988)

[8.4.2. 旅游网站前台 91](#_Toc507320989)

[8.4.3. 在线音乐播放器 91](#_Toc507320990)

1. 概述
   1. 基础知识
      1. WEB万维网（缩写www）

WWW是环球信息网的缩写，（亦作“Web”、“WWW”、“’W3’”，英文全称为“**World Wide Web**”），中文名字为“万维网”，"环球网"等，常简称为Web。分为Web客户端和Web服务器程序。

* + 1. 超文本

超文本是用**超链接**的方法，将各种不同空间的文字信息组织在一起的**网状文本**。超文本更是一种用户界面范式，用以显示文本及与文本之间相关的内容。

* + 1. 超媒体

“超媒体”是超级媒体的缩写。超媒体是一种**采用非线性网状结构**对块状多媒体信息（包括文本、图像、视频等）进行组织和管理的技术。超媒体在本质上和超文本是一样的，只不过超文本技术在诞生的初期管理的对象是纯文本，所以叫做超文本。

* + 1. HTTP

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。

* + 1. HTTPS

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。

* + 1. FTP

FTP 是File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于Internet上的控制文件的双向传输。

* + 1. URL

URL：Uniform Resource Locator，统一资源定位符；URL是Internet上用来描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上。采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。

URL的格式由下列三部分组成：协议（或称为服务方式）；存有该资源的主机IP地址（有时也包括端口号）；主机资源的具体地址。如目录和文件名等。

第一部分和第二部分之间用”：//”符号隔开，第二部分和第三部分用”/”符号隔开。第一部分和第二部分是不可缺少的，第三部分有时可以省略。

* + 1. MIME

MIME：Multipurpose Internet Mail Extensions;

Web中通过MIME来指定文档的类型；

格式：**（顶层的媒体类型有7种）**

类型/子类型

text/html(plain)、image/gif(jpeg)、video/mpeg(quicktime)

audio、application、multipart、message

* + 1. 超链接

超级链接简单来讲，就是指按内容链接。

超级链接在本质上属于一个网页的一部分，它是一种允许我们同其他网页或站点之间进行连接的元素。各个网页链接在一起后，才能真正构成一个网站。

* + 1. 网站（website）

网站（Website）开始是指在因特网上，根据一定的规则，使用HTML等制作的用于展示特定内容的相关网页的集合。网页，是网站中的任何一个页面，通常是HTML格式（文件扩展名为html、htm、asp、aspx、php或jsp等）。

* + 1. 首页

首页，又称主页或起始页，是用户打开浏览器时默认打开的一个或多个网页。首页也可以指一个网站的入口网页，即打开网站后看到的第一个页面，大多数作为首页的文件名是index、default或main加上扩展名。

* 1. 开发技术与工具
     1. 系统的架构模式

C/S：客户端/服务器结构，client/server，服务器通常采用高性能的工作站，并采用大型数据库系统。客户端需要安装专用的客户端软件。这种结构可以充分利用服务器和客户端的硬件环境优势，将任务合理分配到客户端和服务器，降低系统通讯开销。在2000年以前，这种结构占据网络程序开发领域的主流。

B/S：浏览器/服务器结构，browers/server，客户端不需要开发任何用户界面，通过web浏览器向服务器发送请求。

* + 1. 浏览器

1993年初，第一个图形用户界面的Browser，Mosaic诞生。

1994.12，Netscape面世；

1995年，IE推出；

浏览器：Internet Explorer、Firefox、Chrome、Opera、Safari

* + 1. 服务器

Apache：59%，支持多种平台，开放源码，速度快，可靠性高；

IIS：31%，支持Windows平台；

Web Server：Internet Information Server、Apache WebSphere、AOL Server、Jigsaw、俄罗斯Nginx

* + 1. 客户端应用技术

在进行Web应用开发时，离不开客户端技术的支持。目前，比较常用的客户端技术包括HTML语言、CSS样式、Flash和客户端脚本技术。

HTML语言是客户端技术的基础，主要用于显示网页信息，它不需要编译，由浏览器解释执行。

CSS就是一种叫做样式表（style sheet）的技术，也有人称之为层叠样式表（Cascading Style Sheet）。在制作网页时采用CSS样式，可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。

客户端脚本技术是指嵌入到Web页面中的程序代码，这些程序代码是一种解释性的语言，浏览器可以对客户端脚本进行解释。

* + 1. 服务端技术

在开发动态网站时，离不开服务器端技术，目前，比较常用的服务器端技术主要有CGI、ASP、PHP、ASP.NET和JSP。

CGI是最早用来创建动态网页的一种技术，它可以使浏览器与服务器之间产生互动关系。CGI的全称是Common Gateway interface，即通用网关接口。

ASP（Active Server Page）是一种使用很广泛的开发动态网站的技术。

PHP来自于Personal Home Page一词，但现在的PHP已经不再表示名词的缩写，而是一种开发动态网页技术的名称。

Java Server Pages简称JSP。JSP是以Java为基础开发的，所以它沿用Java强大的API功能。

* 1. Web技术发展概述
     1. Web技术发展的三个阶段

静态网页、动态网页、web2.0

* + 1. Web2.0应用技术

Blog、wiki、RSS、SNS社交网络

* 1. 学习方法

（1）通过网络资源w3school

（2）网络视频、麦子学院

（3）小组合作

（4）敢动、愿动、多动

1. HTML
   1. HTML概述
      1. HTML历史

1989 Tim Berners-Lee

1991 HTML

1993 HTML+

1995 HTML2.0

1997 HTML3.2

1999 HTML4.0.1

2000 XHTML

2002 HTML5 2003 XHTML5

2008 W3C发布HTML5草案

* + 1. HTML概念

HTML 是用来描述网页的一种语言。

HTML 指的是超文本标记语言 (Hyper Text Markup Language)

HTML 不是一种编程语言，而是一种标记语言 (markup language)

标记语言是一套标记标签(markup tag)

HTML 使用标记标签来描述网页

* + 1. HTML 标签

HTML 标签是由尖括号包围的关键词，比如 <html>

HTML 标签通常是成对出现的，比如 <b> 和 </b>

标签对中的第一个标签是开始标签，第二个标签是结束标签

开始和结束标签也被称为开放标签和闭合标签

**<标签>内容</标签>**

* + 1. HTML 元素

一个 HTML 元素包含了开始标签与结束标签，如<p>这是一个元素（包含标签）</p>

* + 1. HTML的特点

HTML文档由浏览器解释执行

不编译、不报错

HTML中不支持 空格、回车、制表符，它们都会被解析成一个空白字符

HTML不区分大小写，万维网联盟（W3C）在 HTML 4 中推荐使用小写，而在未来 (X)HTML 版本中强制使用小写。

* + 1. 结构标记

HTML的结构标记包括html、head、body

·<html>……</html>：表示HTML文件开始和结束，其中包含<head>和<body>标记。HTML文档中所有的内容都应该在这两个标记之间。

·<head>……</head>：HTML文件的头部标记，主要包括页面的一些基本描述语句，包括定义网页标题的title。

·body>……</body>：HTML文件的主体标记，绝大多数HTML内容都放置在这个区域中。

* 1. HTML基础知识
     1. 特殊标记
        1. 水平线

<hr/>标签在HTML页面中创建水平线。hr元素可用于分隔内容。

* + - 1. 换行

不产生一个新段落的情况下进行换行（新行），使用 <br/> 标签。

* + - 1. 空格

在网页中添加空格，请使用&nbsp;标签。

* + - 1. 注释

浏览器会忽略注释，也不会显示它们。开始括号之后（左边的括号）需要紧跟一个叹号。

<!-- This is a comment -->

* + - 1. 特殊符号

" &quot; & &amp; < &lt; > &gt;

© &copy; ® &reg;

* + 1. 标题字标记

HTML 标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

浏览器会自动地在标题的前后添加空行。

* + 1. 段落标记

HTML 段落是通过 <p> 标签定义的。

* + 1. 字体标记

<font> 标签规定文本的字体、字体尺寸、字体颜色。

<font face="宋体" size="3" color="green">这是文本!</font>

* + 1. 字体修饰标记

<b> 定义粗体文本。 <big> 定义大号字。

<em> 定义着重文字。 <i> 定义斜体字。

<small> 定义小号字。 <strong> 定义加重语气。

<sub> 定义下标字。 <sup> 定义上标字。

<ins> 定义插入字。 <del> 定义删除字。

<s> 不赞成使用。使用 <del> 代替。

<strike> 不赞成使用。使用 <del> 代替。

<u> 不赞成使用。使用样式（style）代替。

* + 1. 列表

无序列表始于 <ul> 标签。每个列表项始于 <li>。

有序列表始于 <ol> 标签。每个列表项始于 <li> 标签。

自定义列表以 <dl> 标签开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。

* + 1. 相对路径与绝对路径
       1. 绝对路径

绝对路径指的是文件或目录在硬盘上的真实路径。使用绝对路径定位链接目标文件比较清晰，但是有缺点，第一个缺点需要输入更多的内容，第二个缺点如果链接指向文件被移动了，就需要重新设置所有的相关链接。

【例】现在以代码的形式为大家展示利用绝对路径创建一个超链接。代码如下所示。

<body>

<a href="C:\test\02\test.html">使用绝对路径链接一个文件</a>

</body>

* + - 1. 相对路径

相对路径最适合网站的内部链接。只要是属于同一网站之下的，即使不在同一个目录下，相对链接也非常适合。文件相对地址是书写内部链接的理想形式。只要是处于站点文件夹内，相对地址可以自由地在文件之间构建链接。这种地址形式利用的是构建链接的两个文件之间的相对关系，不受站点文件夹所处服务器位置的影响。因此这种书写形式省略了绝对地址中的相同部分。这样做的有点是：站点文件夹所在的服务器地址发生改变时，文件夹的所有内部链接不会出现问题。

相对链接的使用方法为：

如果链接到同一目录下的文件，则只需输入要链接文件名+文件后缀名。要链接到下一级目录中的文件，需要先输入目录名，然后加“/”，再输入文件名+文件后缀名。如链接到上一级目录中的文件，则先输入“../”，再输入文件名+文件后缀名。

【例】现在以代码的形式为大家展示利用相对路径创建一个超链接。代码如下所示。

<body>

<a href="../test1.html"> 使用相对路径链接一个文件</a>

</body>

* + 1. 超链接

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：<a href="url">Link text</a>

通过使用 href 属性 - 创建指向另一个文档的链接

通过使用 name 属性 - 创建文档内的书签。

使用 Target 属性，你可以定义被链接的文档在何处显示。

* + - 1. 锚

<a name="label">锚（显示在页面上的文本）</a>

将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到这个锚

* + 1. 图片

在 HTML 中，图像由 <img> 标签定义。<img src="url" />

<img> 是空标签。要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

常用属性：alt、height、width

背景色bgcolor

背景图片background

* 1. HTML表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。

* + 1. 表格标记

<table> 定义表格

<caption> 定义表格标题。

<th> 定义表格的表头。

<tr> 定义表格的行。

<td> 定义表格单元。

* + 1. 表格的常用属性

表格的边框属性：border，默认值为0；

表格的边框色：bordercolor

设置表格的宽度和高度：<table width=value height=value>

表格的背景颜色：bgcolor

表格的背景图片：background

* + 1. 表格的对齐方式

在水平方向上，可以通过设定行属性align的值，来改变本行的水平对齐方式，分别为左对齐、居中对齐和右对齐。具体语法如下：

<tr align=value>

value：表格的对齐方式可以取值为left、center和right。

在垂直方向上，可以通过设定行属性valign的值，来改变本行的垂直对齐方式，分别为居上、居中和居下。具体语法如下：

<tr valign=value>

value：表格的对齐方式可以取值为top、middle和bottom。

* + 1. 表格合并行合并列

colspan 属性规定单元格可横跨的列数。

rowspan 属性规定单元格可横跨的行数。

* + 1. 表格的结构标记

<thead>、<tbody>、<tfoot>

* 1. HTML表单

表单的主要功能是收集信息，具体说是收集浏览者的信息。HTML表单是HTML页面与浏览器实现交互的重要手段。

* + 1. 表单标记

HTML 表单用于搜集不同类型的用户输入，使用标记<form>定义。

<**form** **action**="action\_page.php" **method**="GET" **target**="\_blank" **ectype**="application/x-www-form-urlencoded">

</**form**>

HTML 表单元素主要包含<input> 元素、<textarea>元素、<select>元素、<button>元素。

* + - 1. 表单属性

name 属性规定表单的名称。

action 属性规定当提交表单时，向何处发送表单数据。

method 方法规定如何发送表单数据（form-data）（表单数据会被发送到在 action 属性中规定的页面中）。Get：默认。将表单数据（form-data）以名称/值对的形式附加到URL中：URL?name=value&name= value。Post：以 HTTP post 事务的形式发送表单数据（form-data）。

target 属性规定一个名称或一个关键词，指示在何处打开 action URL，即在何处显示提交表单后接收到的响应。

enctype 属性规定在将表单数据发送到服务器之前如何对其进行编码。

application/x-www-form-urlencoded 默认。在发送前对所有字符进行编码（将空格转换为 "+" 符号，特殊字符转换为 ASCII HEX 值）。

multipart/form-data 不对字符编码。**当使用有文件上传控件的表单时，该值是必需的。**

text/plain 将空格转换为 "+" 符号，但不编码特殊字符。

* + - 1. 提交方式的区别

1、GET提交的数据将现实在地址栏，不安全；POST 不显示，更加安全；

2、GET提交的数据长度有限制，POST没有限制；

3、GET收集数据之后主动推送给服务器，POST则是收集数据之后通知服务器来接收。因此，POST方式在服务器异常的时候将提交失败；

4、GET由于主动推送信息，速度较快；POST较慢；

5、一般情况下（不特别指明提交方式），默认的提交方式都为GET。

* + 1. 表单元素input

文字域text

密码域password

单选框radio

复选框checkbox

普通按钮button

提交按钮submit

重置按钮reset

图像域image

隐藏域hidden

文件域file

* + - 1. input通用属性

type：用于指定添加的是哪种类型的输入字段，共有17个可选值。

disabled：用于指定输入字段不可用，即字段变成灰色。其属性值可以为空值，也可以指定为disabled

checked：用于指定输入字段是否处于被选中状态，用于type属性值为radio和checkbox的情况下。其属性值可以为空值，也可以指定为checked

width：用于指定输入字段的宽度，用于type属性值为image的情况下。

height：用于指定输入字段的高度，用于type属性值为image的情况下。

maxlength：用于指定输入字段可输入文字的个数，用于type属性值为text和password的情况下，默认没有字数限制

readonly：用于指定输入字段是否为只读。其属性值可以为空值，也可以指定为readonly

size：用于指定输入字段的宽度，当type属性为text和password时，以文字个数为单位，当type属性为其他值时，以像素为单位

src：用于指定图片的来源，只有当type属性为image时有效

alt：用于指定当图片无法显示时，显示的文字，只有当type属性为image时有效

name：用于指定输入字段的名称

value：用于指定输入字段默认数据值，当type属性为checkbox和radio时，不可省略此属性，为其他值时，可以省略。当type属性为button、reset和submit时，指定的是按钮上的显示文字；当type属性为checkbox和radio时，指定的是数据项选定时的值。

* + 1. 下拉列表

<select> 元素用来创建下拉列表。

<select> 元素中的 <option> 标签定义了列表中的可用选项。

multiple属性为 true 时，可选择多个选项。

size=”number”：规定下拉列表中可见选项的数目。

* + 1. 文本区域标记

在HTML中还有一种特殊定义的文本样式，称为文字域或文本域。它与文字字段的区别在于可以添加多行文字，从而可以输入更多的文本。这类控件在一些留言板中最为常见。具体语法如下：

<textarea name=”textname” value=”text\_value” **rows**=rows\_value **cols**=cols\_value>

* 1. 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。每份HTML文档称为一个框架，并且每个框架都独立于其他的框架。

* + 1. iframe

iframe 用于在网页内显示网页。

<iframe src="URL"></iframe>

height 和 width 属性用于规定 iframe 的高度和宽度。

<iframe src="demo\_iframe.htm" width="200" height="200"></iframe>

* + 1. frameset

框架结构标签（<frameset>）定义如何将窗口分割为框架，每个 frameset 定义了一系列行或列。

rows/cols 的值规定了每行或每列占据屏幕的面积。

frame 标签定义了放置在每个框架中的 HTML 文档。

<frameset rows="50%,50%">

<frame src="/example/html/frame\_a.html">

<frameset cols="25%,75%">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

</frameset>

* 1. 其他知识
     1. 块级元素

大多数 HTML 元素被定义为块级元素或内联元素。

编者注：“块级元素”译为 block level element，“内联元素”译为 inline element。

块级元素在浏览器显示时，通常会以新行来开始（和结束）。

例子：<h1>, <p>, <ul>, <table>

* + 1. 内联元素

HTML 内联元素

内联元素在显示时通常不会以新行开始。

例子：<b>, <td>, <a>, <img>

* + 1. div与span

HTML <div> 元素是块级元素，它是可用于组合其他 HTML 元素的容器。<div> 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

HTML <span> 元素是内联元素，可用作文本的容器。<span> 元素也没有特定的含义。

* + 1. marquee

页面文字的自动滚动效果，可由JavaScript来实现，而html标签 - <marquee></marquee>可以实现多种滚动效果，无需js控制。

使用marquee标记不仅可以移动文字，也可以移动图片，表格等.

语法：<marquee>...</marquee>； 说明：在标记之间添加要进行滚动的内容。

* 1. 实验
     1. HTML网页标签练习
     2. 简单个性化网站设计
* 使用HTML基本标签设计个人主页；
* 使用HTML书签链接设计好友简介子页面；
* 使用多级列表标签设计兴趣爱好子页面；
* 使用表格、图片标记设计个人信息子页面；
* 使用表单标签设计一个注册界面；
* 使用段落及对齐属性设计版权声明子页面。
  + 1. 制作百度搜索界面
    2. 制作百度图库搜索界面

使用HTML技术实现百度图库搜索功能，效果如图1所示。

网页中包含百度logo，并且图片logo.jpg与当前网页在同一目录下。

网页中存在一个表单，表单中包括输入框、提交按钮、隐藏域3个控件，要求提交方式为get。

输入框要求name属性值为word，隐藏域的name属性值为tn，取值为baiduimage，提交按钮的文字显示为“百度一下”。

百度图库的提交网址为：http://image.baidu.com/search/index

使用JavaScript验证输入内容是否为空，若为空则给出警告提示，且不能提交搜索。



1. CSS
   1. CSS语法
      1. CSS

* CSS 指层叠样式表 (Cascading Style Sheets)
* 样式定义如何显示 HTML 元素
* 样式通常存储在样式表中
* 把样式添加到HTML中，是为了解决内容与表现分离的问题
* 外部样式表可以极大提高工作效率
* 外部样式表通常存储在 CSS 文件中
* 多个样式定义可层叠为一
  + - 1. 层叠样式表
* 样式通常存储在样式表中
* 多种样式叠加
* 样式的继承和覆盖
  + - 1. 优点
* 将内容与表现分离，加快网页显示速度
* 易维护，提高工作效率
* 易扩展
* 可移植
  + 1. CSS声明

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明:



每条声明由一个属性和一个值组成。

属性和值被**冒号**分开。

**声明总是以分号(;)结束，声明组以大括号({})括起来。**

**CSS注释以 "/\*" 开始, 以 "\*/" 结束。**

* + 1. CSS的使用方法

插入样式表的方法有三种:

* 外部样式表(External style sheet)
* 内部样式表(Internal style sheet)
* 内联样式(Inline style)

**内联样式）Inline style > （内部样式）Internal style sheet >（外部样式）External style sheet > 浏览器默认样式**

* + - 1. 内联样式

在相关的标签内使用样式（style）属性。Style 属性可以包含任何 CSS 属性。

<p style="color:sienna;margin-left:20px">这是一个段落。</p>

* + - 1. 内部样式

当单个文档需要特殊的样式时，就应该使用内部样式表。使用 <style> 标签在文档头部定义内部样式表。

<style>

hr {color:sienna;}

p {margin-left:20px;}

body {background-image:url("images/back40.gif");}

</style>

* + - 1. 外部样式

外部样式表可以在任何文本编辑器中进行编辑。文件不能包含任何的 html 标签。样式表应该以 .css 扩展名进行保存。

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

* 1. CSS选择器
     1. 常用选择器
* 通配符选择器

通配符\*代表所有网页元素。

* 类型（标记）选择器 E{……}

CSS选择器可以直接选用HTML已经存在的标记作为选择器，选中网页中所有该标签标记的元素。

* id选择器 #myid

HTML元素以id属性来设置id选择器,CSS 中 id 选择器以 "#" 来定义。

id是唯一的。

* class选择器 .classname

class 选择器用于描述一组元素的样式，class 选择器有别于id选择器，class可以在多个元素中使用。

class 选择器在HTML中以class属性表示, 在 CSS 中，类选择器以一个点"."号显示。

**派生选择器：**

标记.类名 {样式属性1:样式值1;样式属性2:样式值2}

* 组合声明

E,F{……}

选中所有的E和F元素。

* + 1. 组合选择器

组合选择符说明了两个选择器直接的关系。

在 CSS3 中包含了四种组合方式:

* 后代选取器(以空格分隔 E F)
* 子元素选择器(以大于号分隔E>F）
* 相邻兄弟选择器（以加号分隔E+F）
* 普通兄弟选择器（以破折号分隔E~F）
  + 1. 属性选择器

CSS属性选择器可以选中具有特定属性的HTML元素样式。

|  |  |  |  |
| --- | --- | --- | --- |
| **选择器** | **示例** | **说明** | **版本** |
| [*attribute*] | [target] | 选择所有带有target属性元素 | 2 |
| [*attribute*=*value*] | [target=-blank] | 选择所有使用target="-blank"的元素 | 2 |
| [*attribute*~=*value*] | [title~=flower] | 选择标题属性包含单词"flower"的所有元素 | 2 |
| [*attribute*|=*language*] | [lang|=en] | 选择一个lang属性的起始值="EN"的所有元素 | 2 |
| [*attribute*^=*value*] | a[src^="https"] | 选择每一个src属性的值以"https"**开头**的元素 | 3 |
| [*attribute*$=*value*] | a[src$=".pdf"] | 选择每一个src属性的值以".pdf"**结尾**的元素 | 3 |
| [*attribute*\*=*value*] | a[src\*="runoob"] | 选择每一个src属性的值**包含子字符串**"runoob"的元素 | 3 |

* + 1. 伪选择器
       1. 伪类选择器
* 链接伪类

|  |  |  |  |
| --- | --- | --- | --- |
| :link | a:link | 选择所有未访问链接 | 1 |
| :visited | a:visited | 选择所有访问过的链接 | 1 |
| :hover | a:hover | 选择鼠标在链接上面时 | 1 |
| :active | a:active | 选择活动链接 | 1 |

* 操作伪类

|  |  |  |  |
| --- | --- | --- | --- |
| :focus | input:focus | 选择具有焦点的输入元素 | 2 |
| [:enabled](http://www.runoob.com/cssref/sel-enabled.html) | input:enabled | 选择每一个已启用的输入元素 | 3 |
| [:disabled](http://www.runoob.com/cssref/sel-disabled.html) | input:disabled | 选择每一个禁用的输入元素 | 3 |
| [:checked](http://www.runoob.com/cssref/sel-checked.html) | input:checked | 选择每个选中的输入元素 | 3 |
| [:empty](http://www.runoob.com/cssref/sel-empty.html) | p:empty | 选择每个没有任何子级的p元素（包括文本节点） | 3 |

* + - 1. 伪元素选择器

伪元素选择器是针对元素位置的选择器，可以在元素的不同位置进行样式设置。

|  |  |  |
| --- | --- | --- |
| [:first-letter](http://www.runoob.com/cssref/sel-firstletter.html) | p:first-letter | 选择每个<p> 元素的第一个字母 |
| [:first-line](http://www.runoob.com/cssref/sel-firstline.html) | p:first-line | 选择每个<p> 元素的第一行 |
| [:first-child](http://www.runoob.com/cssref/sel-firstchild.html) | p:first-child | 选择器匹配属于任意元素的第一个子元素的 <p> 元素 |
| [:before](http://www.runoob.com/cssref/sel-before.html) | p:before | 在每个<p>元素之前插入内容 |
| [:after](http://www.runoob.com/cssref/sel-after.html) | p:after | 在每个<p>元素之后插入内容 |

* + - 1. 结构伪类选择器

|  |  |  |  |
| --- | --- | --- | --- |
| [:first-child](http://www.runoob.com/cssref/sel-firstchild.html) | p:first-child | 指定只有当<p>元素是其父级的第一个子级的样式。 | 2 |
| [:last-child](http://www.runoob.com/cssref/sel-last-child.html) | p:last-child | 选择每个p元素是其父级的最后一个子级。 | 3 |
| [:only-child](http://www.runoob.com/cssref/sel-only-child.html) | p:only-child | 选择每个p元素是其父级的唯一子元素 | 3 |
| [:first-of-type](http://www.runoob.com/cssref/sel-first-of-type.html) | p:first-of-type | 选择每个p元素是其父级的第一个p元素 | 3 |
| [:last-of-type](http://www.runoob.com/cssref/sel-last-of-type.html) | p:last-of-type | 选择每个p元素是其父级的最后一个p元素 | 3 |
| [:only-of-type](http://www.runoob.com/cssref/sel-only-of-type.html) | p:only-of-type | 选择每个p元素是其父级的唯一p元素 | 3 |
| [:nth-child(*n*)](http://www.runoob.com/cssref/sel-nth-child.html) | p:nth-child(2) | 选择每个p元素是其父级的第二个子元素 | 3 |
| [:nth-last-child(*n*)](http://www.runoob.com/cssref/sel-nth-last-child.html) | p:nth-last-child(2) | 选择每个p元素的是其父级的第二个子元素，从最后一个子项计数 | 3 |
| [:nth-of-type(*n*)](http://www.runoob.com/cssref/sel-nth-of-type.html) | p:nth-of-type(2) | 选择每个p元素是其父级的第二个p元素 | 3 |
| [:nth-last-of-type(*n*)](http://www.runoob.com/cssref/sel-nth-last-of-type.html) | p:nth-last-of-type(2) | 选择每个p元素的是其父级的第二个p元素，从最后一个子项计数 | 3 |

* + 1. 选择器优先级

当两个规则都作用到了同一个html元素上时，如果定义的属性有冲突，那么应该用谁的值的，CSS有一套优先级的定义。

* !important 会覆盖页面内任何位置定义的元素样式。
* 作为style属性写在元素内的样式
* id选择器
* 类选择器
* 标签选择器
* 通配符选择器
* 浏览器自定义或继承

总结：!important > 行内样式>ID选择器 > 类选择器 > 标签 > 通配符 > 继承 > 浏览器默认属性

CSS优先级：是由四个级别和各级别的出现次数决定的。

四个级别分别为：行内选择符、ID选择符、类别选择符、元素选择符。

优先级的算法：

* 每个规则对应一个初始"四位数"：0、0、0、0
* 若是 行内选择符，则加1、0、0、0
* 若是 ID选择符，则加0、1、0、0
* 若是类选择符/属性选择符/伪类选择符，则分别加0、0、1、0
* 若是 元素选择符/伪元素选择符，则分别加0、0、0、1

算法：将每条规则中，选择符对应的数相加后得到的”四位数“，从左到右进行比较，大的优先级越高。

* 1. CSS属性
     1. 字体

|  |  |  |
| --- | --- | --- |
| **属性** | **说明** | **CSS** |
| [font](http://www.runoob.com/cssref/pr-font-font.html) | 在一个声明中设置所有字体属性 | 1 |
| [font-family](http://www.runoob.com/cssref/pr-font-font-family.html) | 规定文本的字体系列 | 1 |
| [font-size](http://www.runoob.com/cssref/pr-font-font-size.html) | 规定文本的字体尺寸 | 1 |
| [font-style](http://www.runoob.com/cssref/pr-font-font-style.html) | 规定文本的字体样式 | 1 |
| [font-variant](http://www.runoob.com/cssref/pr-font-font-variant.html) | 规定文本的字体样式 | 1 |
| [font-weight](http://www.runoob.com/cssref/pr-font-weight.html) | 规定字体的粗细 | 1 |
| [font-stretch](http://www.runoob.com/cssref/css3-pr-font-stretch.html) | 收缩或拉伸当前的字体系列 | 3 |
| color | 用于设置字体颜色 |  |
| [opacity](http://www.runoob.com/cssref/css3-pr-opacity.html) | 设置一个元素的透明度级别 | 3 |

* + 1. 文本

|  |  |  |
| --- | --- | --- |
| **属性** | **说明** | **CSS** |
| [color](http://www.runoob.com/cssref/pr-text-color.html) | 设置文本的颜色 | 1 |
| [direction](http://www.runoob.com/cssref/pr-text-direction.html) | 规定文本的方向 | 2 |
| [letter-spacing](http://www.runoob.com/cssref/pr-text-letter-spacing.html) | 设置字符间距 | 1 |
| [line-height](http://www.runoob.com/cssref/pr-dim-line-height.html) | 设置行高 | 1 |
| [text-align](http://www.runoob.com/cssref/pr-text-text-align.html) | 规定文本的水平对齐方式 | 1 |
| [text-decoration](http://www.runoob.com/cssref/pr-text-text-decoration.html) | 规定添加到文本的装饰效果 | 1 |
| [text-indent](http://www.runoob.com/cssref/pr-text-text-indent.html) | 规定文本块首行的缩进 | 1 |
| [text-transform](http://www.runoob.com/cssref/pr-text-text-transform.html) | 控制文本的大小写 | 1 |
| [text-overflow](http://www.runoob.com/cssref/css3-pr-text-overflow.html) | 指定当文本溢出包含的元素，应该发生什么 | 3 |
| [text-shadow](http://www.runoob.com/cssref/css3-pr-text-shadow.html) | 为文本添加阴影 | 3 |
| [text-wrap](http://www.runoob.com/cssref/css3-pr-text-wrap.html) | 指定文本换行规则 | 3 |

text-shadow: h-shadow v-shadow blur（模糊距离） color;

注意： text-shadow属性连接一个或更多的阴影文本。属性是阴影，指定的每2或3个长度值和一个可选的颜色值用逗号分隔开来。已失时效的长度为0。

* + 1. 边框

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [border](http://www.runoob.com/cssref/pr-border.html) | 复合属性。设置对象边框的特性。 | 1 |
| [border-bottom-color](http://www.runoob.com/cssref/pr-border-bottom-color.html) | 设置或检索对象的底部边框颜色。 | 1 |
| [border-bottom-style](http://www.runoob.com/cssref/pr-border-bottom-style.html) | 设置或检索对象的底部边框样式。 | 1 |
| [border-bottom-width](http://www.runoob.com/cssref/pr-border-bottom-width.html) | 设置或检索对象的底部边框宽度。 | 1 |
| [border-style](http://www.runoob.com/cssref/pr-border-style.html) | 设置或检索对象的边框样式。 | 1 |
| [border-image](http://www.runoob.com/cssref/css3-pr-border-image.html) | 设置或检索对象的边框样式使用图像来填充。 | 3 |
| [border-radius](http://www.runoob.com/cssref/css3-pr-border-radius.html) | 设置或检索对象使用圆角边框。 | 3 |
| [box-shadow](http://www.runoob.com/cssref/css3-pr-box-shadow.html) | 向方框添加一个或多个阴影。 | 3 |

* + 1. 背景

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [background](http://www.runoob.com/cssref/css3-pr-background.html) | 复合属性。设置对象的背景特性。 | 1 |
| [background-attachment](http://www.runoob.com/cssref/pr-background-attachment.html) | 设置或检索背景图像是随对象内容滚动还是固定的。必须先指定background-image属性。 | 1 |
| [background-color](http://www.runoob.com/cssref/pr-background-color.html) | 设置或检索对象的背景颜色。 | 1 |
| [background-image](http://www.runoob.com/cssref/pr-background-image.html) | 设置或检索对象的背景图像。 | 1 |
| [background-position](http://www.runoob.com/cssref/pr-background-position.html) | 设置或检索对象的背景图像位置。必须先指定background-image属性。 | 1 |
| [background-repeat](http://www.runoob.com/cssref/pr-background-repeat.html) | 设置或检索对象的背景图像如何铺排填充。必须先指定background-image属性。 | 1 |
| [background-clip](http://www.runoob.com/cssref/css3-pr-background-clip.html) | 指定对象的背景图像向外裁剪的区域。 | 3 |
| [background-size](http://www.runoob.com/cssref/css3-pr-background-size.html) | 检索或设置对象的背景图像的尺寸大小。 | 3 |

* + 1. 表格

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [border-collapse](http://www.runoob.com/cssref/pr-border-collapse.html) | 规定是否合并表格边框 | 2 |
| [border-spacing](http://www.runoob.com/cssref/pr-border-spacing.html) | 规定相邻单元格边框之间的距离 | 2 |
| [caption-side](http://www.runoob.com/cssref/pr-tab-caption-side.html) | 规定表格标题的位置 | 2 |
| [table-layout](http://www.runoob.com/cssref/pr-tab-table-layout.html) | 设置用于表格的布局算法 | 2 |

* + 1. 列表

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [list-style](http://www.runoob.com/cssref/pr-list-style.html) | 在一个声明中设置所有的列表属性 | 1 |
| [list-style-image](http://www.runoob.com/cssref/pr-list-style-image.html) | 将图象设置为列表项标记 | 1 |
| [list-style-position](http://www.runoob.com/cssref/pr-list-style-position.html) | 设置列表项标记的放置位置 | 1 |
| [list-style-type](http://www.runoob.com/cssref/pr-list-style-type.html) | 设置列表项标记的类型 | 1 |

* 1. CSS布局
     1. 盒子模型

CSS盒模型本质上是一个盒子，封装周围的HTML元素，它包括：边距，边框，填充，和实际内容。

盒模型允许我们在其它元素和周围元素边框之间的空间放置元素。

下面的图片说明了盒子模型(Box Model)：



不同部分的说明：

**Margin(外边距)**- 清除边框外的区域，外边距是透明的。

**Border(边框)**- 围绕在内边距和内容外的边框。

**Padding(内边距)**- 清除内容周围的区域，内边距是透明的。

**Content(内容)**- 盒子的内容，显示文本和图像。

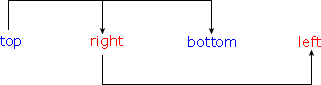
* + - 1. 值复制

如果缺少左外边距的值，则使用右外边距的值。

如果缺少下外边距的值，则使用上外边距的值。

如果缺少右外边距的值，则使用上外边距的值。

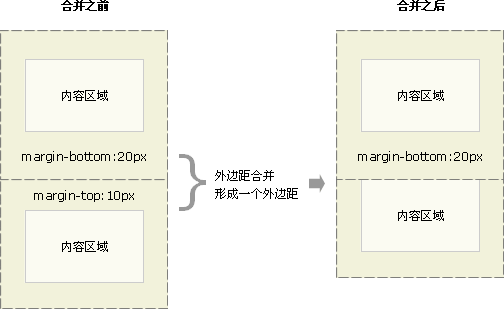
下图提供了更直观的方法来了解这一点：

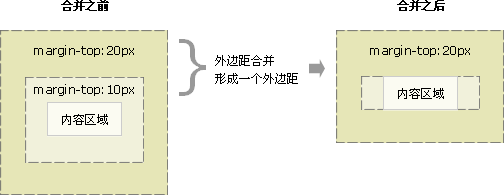


换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到。

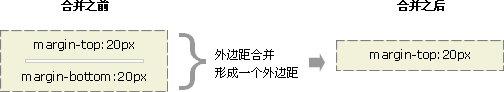
* + - 1. 外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

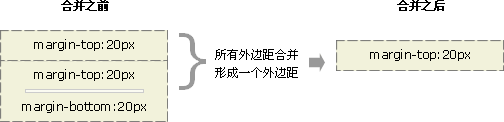




假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：



如果这个外边距遇到另一个元素的外边距，它还会发生合并：



这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

* + 1. 定位

CSS 定位 (Positioning) 属性允许你对元素进行定位。可以选择 4 种不同类型的定位，这会影响元素框生成的方式。

static：元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。

relative：元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

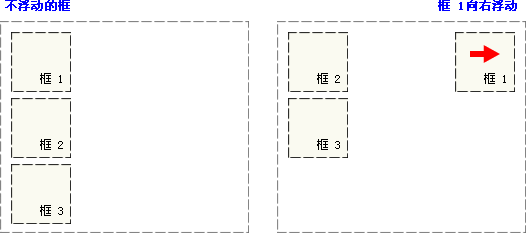
absolute：元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。

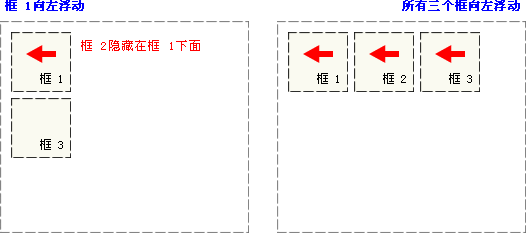
fixed：元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。

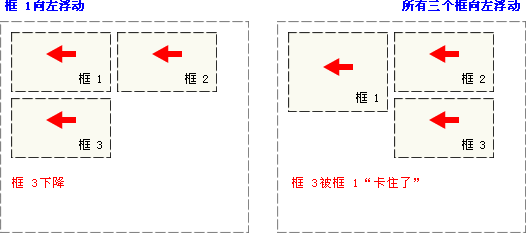
* + 1. 浮动

通过 float 属性实现元素的浮动。

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。







clear属性指定段落的左侧或右侧不允许浮动的元素。

z-index 属性指定一个元素的堆叠顺序。number，设置元素的堆叠顺序。**数字越大，则堆叠在越上面。**

拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。

注意： z-index 进行定位元素(position:absolute, position:relative, or position:fixed)。

* + 1. display

display 属性规定元素应该生成的框的类型。

* none：此元素不会被显示。
* block：此元素将显示为块级元素，此元素前后会带有换行符。
* inline：默认。此元素会被显示为内联元素。
* inline-block：行内块元素。（CSS2.1 新增的值）

display:block就是将元素显示为块级元素。block元素的特点是：

* 总是在新行上开始；
* 高度、行高以及顶和底边距都可控制；
* 宽度缺省是它的容器的100%，除非设定一个宽度

display:inline就是将元素显示为行内元素。inline元素的特点是：

* 和其他元素都在一行上；
* 高、行高及顶和底边距不可改变；
* 宽度就是它的文字或图片的宽度，不可改变。

inline和block可以控制一个元素的行宽高等特性，需要切换的情况如下：

让一个inline元素从新行开始；

让块元素和其他元素保持在一行上；

控制inline元素的宽度（对导航条特别有用）；

控制inline元素的高度；

无须设定宽度即可为一个块元素设定与文字同宽的背景色。

display:inline-block将对象呈递为内联对象，但是对象的内容作为块对象呈递。旁边的内联对象会被呈递在同一行内，允许空格。

* + 1. 布局居中
       1. 参考线

元素的顶线：元素所占区域的顶部。

元素的中线：元素所占区域的中间线。

元素的基线：字母X的底边。

元素的底线：元素所占区域的底部边界。

行的顶线：实际高度的上边界。

行的中线：基线往上1/2字母x的高度，大约位于x交叉点。

行的基线：行内默认文本x的底边。

行的底线：实际高度的下边界。

* + - 1. 单行内容垂直居中
* 设置行高和元素高度，且两者相等
* 设置over-flow:hidden
  + - 1. 多行内容垂直居中
* 不固定容器高度
* 只设置padding-top和padding-bottom大小，且两者相等
  + - 1. inline-block元素垂直居中
* 设置容器行高
* 设置inline元素垂直居中
  + - 1. div元素内容垂直居中
* 嵌套两层div
* 外层div显示方式为table
* 内层div显示为table-cell且垂直居中
* 嵌套两层div
* 外层div设置高度height
* 内层div设置相对定位，top为容器50%，高度height的1/2
* margin-top为-1/4\*height，overflow设置为auto
* 嵌套两层div
* 外层div设置高度height
* 内层div之前增加一个浮动元素float:left;高度为容器的1/2
* 浮动元素margin-bottom为-1/2\*内层div高度
* 内层div设置clear:both，设置固定高度height
  1. CSS3
     1. 圆角

通过CSS3，够创建圆角边框，向矩形添加阴影，使用图片来绘制边框。这将到以下边框属性：

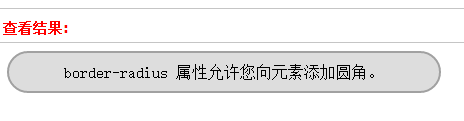
border-radius

box-shadow

border-image

border-radius:25px;

-moz-border-radius:25px; /\* 老的 Firefox \*/



向 div 元素添加方框阴影：

div

{

box-shadow: 10px 10px 5px #888888;

-moz-box-shadow: 10px 10px 5px #888888; /\* 老的 Firefox \*/

}

将图片规定为包围 div 元素的边框：

div

{

-webkit-border-image:url(border.png) 30 30 round; /\* Safari 5 \*/

-o-border-image:url(border.png) 30 30 round; /\* Opera \*/

border-image:url(border.png) 30 30 round;

}

* + 1. 渐变/过渡效果

CSS3 过渡（transition）是元素从一种样式逐渐改变为另一种的效果。要实现这一点，必须规定两项内容：

* 规定您希望把效果添加到哪个 CSS 属性上
* 规定效果的时长
* 如需向多个样式添加过渡效果，请添加多个属性，由逗号隔开。

div

{

width:100px;

height:100px;

background:yellow;

transition:width 2s, height 2s;

-moz-transition:width 2s, height 2s, -moz-transform 2s; /\* Firefox 4 \*/

-webkit-transition:width 2s, height 2s, -webkit-transform 2s; /\* Safari and Chrome \*/

-o-transition:width 2s, height 2s, -o-transform 2s; /\* Opera \*/

}

div:hover

{

width:200px;

height:200px;

transform:rotate(180deg);

-moz-transform:rotate(180deg); /\* Firefox 4 \*/

-webkit-transform:rotate(180deg); /\* Safari and Chrome \*/

-o-transform:rotate(180deg); /\* Opera \*/

}

transition 属性是一个简写属性，用于设置四个过渡属性：

transition-property

transition-duration

transition-timing-function

transition-delay

transition: property duration timing-function delay;

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [transition](http://www.w3school.com.cn/cssref/pr_transition.asp) | 简写属性，用于在一个属性中设置四个过渡属性。 | 3 |
| [transition-property](http://www.w3school.com.cn/cssref/pr_transition-property.asp" \o "CSS3 transition-property 属性) | 规定应用过渡的 CSS 属性的名称。 | 3 |
| [transition-duration](http://www.w3school.com.cn/cssref/pr_transition-duration.asp" \o "CSS3 transition-duration 属性) | 定义过渡效果花费的时间。默认是 0。 | 3 |
| [transition-timing-function](http://www.w3school.com.cn/cssref/pr_transition-timing-function.asp" \o "CSS3 transition-timing-function 属性) | 规定过渡效果的时间曲线。默认是 "ease"。 | 3 |
| [transition-delay](http://www.w3school.com.cn/cssref/pr_transition-delay.asp) | 规定过渡效果何时开始。默认是 0。 | 3 |

* + 1. 动画

通过 CSS3，我们能够创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。

@keyframes：定义动画

animation：将动画绑定到元素

在 @keyframes 中创建动画时，请把它捆绑到某个选择器，否则不会产生动画效果。通过规定至少以下两项 CSS3 动画属性，即可将动画绑定到选择器：

* 规定动画的名称
* 规定动画的时长

动画是使元素从一种样式逐渐变化为另一种样式的效果。可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%。

div

{

……

**animation:myfirst 5s;**

**-moz-animation:myfirst 5s; /\* Firefox \*/**

-webkit-animation:myfirst 5s; /\* Safari and Chrome \*/

-o-animation:myfirst 5s; /\* Opera \*/

}

@keyframes myfirst

{

0% {background:red; left:0px; top:0px;}

25% {background:yellow; left:200px; top:0px;}

50% {background:blue; left:200px; top:200px;}

75% {background:green; left:0px; top:200px;}

100% {background:red; left:0px; top:0px;}

}

@-moz-keyframes myfirst /\* Firefox \*/

{

……

}

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **CSS** |
| [@keyframes](http://www.w3school.com.cn/cssref/pr_keyframes.asp) | 规定动画。 | 3 |
| [animation](http://www.w3school.com.cn/cssref/pr_animation.asp) | 所有动画属性的简写属性，除了 animation-play-state 属性。 | 3 |
| [animation-name](http://www.w3school.com.cn/cssref/pr_animation-name.asp) | 规定 @keyframes 动画的名称。 | 3 |
| [animation-duration](http://www.w3school.com.cn/cssref/pr_animation-duration.asp) | 规定动画完成一个周期所花费的秒或毫秒。默认是 0。 | 3 |
| [animation-timing-function](http://www.w3school.com.cn/cssref/pr_animation-timing-function.asp) | 规定动画的速度曲线。默认是 "ease"。 | 3 |
| [animation-delay](http://www.w3school.com.cn/cssref/pr_animation-delay.asp) | 规定动画何时开始。默认是 0。 | 3 |
| [animation-iteration-count](http://www.w3school.com.cn/cssref/pr_animation-iteration-count.asp) | 规定动画被播放的次数。默认是 1。 | 3 |
| [animation-direction](http://www.w3school.com.cn/cssref/pr_animation-direction.asp" \o "CSS3 animation-direction 属性) | 规定动画是否在下一周期逆向地播放。默认是 "normal"。 | 3 |
| [animation-play-state](http://www.w3school.com.cn/cssref/pr_animation-play-state.asp) | 规定动画是否正在运行或暂停。默认是 "running"。 | 3 |
| [animation-fill-mode](http://www.w3school.com.cn/cssref/pr_animation-fill-mode.asp) | 规定对象动画时间之外的状态。 | 3 |

* + 1. 2D&3D转换

通过 CSS3 转换transform，我们能够对元素进行移动、缩放、转动、拉长或拉伸。

* + - 1. 2D 转换方法

translate()：元素从其当前位置移动到给定的 left（x 坐标） 和 top（y 坐标） 位置。

rotate()：元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。

scale()：元素的尺寸会增加或减少，根据给定的宽度（X 轴）和高度（Y 轴）参数。

skew()：元素翻转给定的角度，根据给定的水平线（X 轴）和垂直线（Y 轴）参数

matrix()：把所有 2D 转换方法组合在一起。

* + - 1. 3D 转换

CSS3 允许您使用 3D 转换来对元素进行格式化。以下常用 3D 转换方法：

rotateX()：元素围绕其 X 轴以给定的度数进行旋转。

rotateY()：元素围绕其 Y 轴以给定的度数进行旋转。

matrix3d()：把所有 3D 转换方法组合在一起。

* 1. 实验
     1. CSS选择器以及样式练习
     2. 装饰个人网站
     3. 美化百度图库搜索

1. JavaScript
   1. JavaScript语法基础

JavaScript 是世界上最流行的编程语言。这门语言可用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。

JavaScript 是一种轻量级的脚本语言，由浏览器解释执行。

* + 1. JavaScript的特点
* 解释性

JavaScript不同于一些编译性的程序语言，例如C、C++等，它是一种解释性的程序语言，它的源代码不需要经过编译，而直接在浏览器中运行时被解释。

* 基于对象

JavaScript是一种基于对象的语言。这意味着它能运用自己已经创建的对象。因此，许多功能可以来自于脚本环境中对象的方法与脚本的相互作用。

* 事件驱动

JavaScript可以直接对用户或客户输入做出响应，无须经过Web服务程序。它对用户的响应，是以事件驱动的方式进行的。

* 跨平台

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

* 安全性

JavaScript是一种安全性语言，它不允许访问本地的硬盘，并不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互。

* 弱变量

JavaScript语言中的变量是弱类型的，数据类型无须作出严格的要求。

* 动态联编

在程序运行时才能确定将要调用的函数。

* 开发商

JavaScript是网景公司开发的，Java是sun公司开发的。

* + 1. JavaScript的功能
* 写入HTML

使用document.write()方法将内容写到 HTML 文档中。

* 获得网页元素并改变元素属性
* 事件响应

onClick=fun();

* 表单验证

onsubmit = return fun();

* 提示（警告）

使用window.alert()弹出警告框。

使用console.log()写入到浏览器的控制台。

* + 1. JavaScript在HTML中的使用
* 在页面中直接嵌入JavaScript

在HTML文档中可以使用<script>…</script>标记将JavaScript脚本嵌入到其中。在HTML文档中可以使用多个<script>标记，每个<script>标记中可以包含多个JavaScript的代码集合。

|  |  |
| --- | --- |
| 属性 | 说明 |
| language | 设置所使用的脚本语言及版本 |
| src | 设置一个外部脚本文件的路径位置 |
| type | 设置所使用的脚本语言，此属性已代替language属性 |
| defer | 此属性表示当HTML文档加载完毕后再执行脚本语言 |

* 链接外部JavaScript

可以将这些脚本代码放置在一个单独的文件中（保存文件的扩展名为.js），然后在需要使用该代码的Web页面中链接该JavaScript文件即可。

在Web页面中链接外部JavaScript文件的语法格式如下：

<script language="javascript" src="javascript.js"></script>

* + 1. JavaScript语法
* 可以在 HTML 文档中放入不限数量的脚本。
* 脚本可位于 HTML 的 <body> 或 <head> 部分中，或者同时存在于两个部分中。
* 通常把函数放入 <head> 部分中，或者放在页面底部。把它们安置到同一处位置，不会干扰页面的内容。
* JavaScript程序按照在HTML文件中出现的顺序逐行执行。
* 函数体内的代码，不会被立即执行，只有当所在的函数被其他程序调用时，该代码才会被执行。
* JavaScript对字母大小写是敏感（严格区分字母大小写）的.
* 分号用于分隔 JavaScript 语句。
* JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。
* JavaScript注释

单行注释以 // 开头。多行注释以 /\* 开始，以 \*/ 结尾。

* + 1. JavaScript变量以及变量提升

在 JavaScript 中创建变量通常称为“声明”变量。且使用 var 关键词来声明变量。

* 变量必须以字母开头
* 变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）
* 变量名称对大小写敏感（y 和 Y 是不同的变量）
* 不能使用JavaScript中的关键字。
* 变量声明之后，如果不赋值，该变量是空的（它没有值）undefined。
  + - 1. 局部变量与全局变量

变量的作用域（scope）是指某变量在程序中的有效范围，也就是程序中定义这个变量的区域。在JavaScript中变量根据作用域可以分为两种：全局变量和局部变量。

* 全局变量是定义在所有函数之外，作用于整个脚本代码的变量；
* 局部变量是定义在函数体内，只作用于函数体的变量，函数的参数也是局部性的，只在函数内部起作用。
  + - 1. 变量提升

JavaScript 中，函数及变量的声明都将被提升到函数的最顶部。

JavaScript 中，变量可以在使用后声明，也就是变量可以先使用再声明。

变量提升：函数声明和变量声明总是会被解释器悄悄地被"提升"到方法体的最顶部。

**JavaScript 只有声明的变量会提升，初始化的不会。**

var x = 5; // 初始化 x

elem = document.getElementById("demo"); // 查找元素

elem.innerHTML = x + " " + y; // 显示 x 和 y

var y = 7; // 初始化 y

* + 1. JavaScript数据类型
       1. 数值
* 整型数据
* 十六进制数据0x或者0X
* 八进制数据 以0开头
* 浮点型数据
  + - 1. 字符串

字符串（string）是由Unicode字符、数字、标点符号等组成的序列，字符串型数据是包含在单引号或双引号中的，由单引号定界的字符串中可以含有双引号，由双引号定界的字符串中也可以含有单引号。

单引号内出现双引号或双引号内出现单引号时，不需要进行转义。但是，双引号内出现双引号或单引号内出现单引号，则必须进行转义。

* + - 1. 布尔

布尔数据类型只有两个值，这两个合法的值分别由直接量“true”和“false”表示。

**在必要时会将true转换成1，将false转换成0。**

* + - 1. 特殊类型
* 转义字符

以反斜杠开头的不可显示的特殊字符通常称为控制字符，也被称为转义字符。

|  |  |
| --- | --- |
| **转义字符** | **描述** |
| \b | 退格 |
| \n | 回车换行 |
| \t | Tab符号 |
| \f | 换页 |
| \' | 单引号 |
| \v | 跳格（Tab，水平） |
| \r | 换行 |
| \\ | 反斜杠 |
| \OOO | 八进制整数，范围000~777 |
| \" | 双引号 |
| \uhhhh | 十六进制编码的Unicode字符 |

* undefined

未定义类型的变量是undefined，表示变量还没有赋值（如var a;），或者赋予一个不存在的属性值（如var a=String.notProperty;）。

* NaN

此外，JavaScript中有一种特殊类型的数字常量NaN，即“非数字”。当在程序中由于某种原因发生计算错误后，将产生一个没有意义的数字，此时JavaScript返回的数字值就是NaN。

* null

JavaScript中的关键字null是一个特殊的值，它表示为空值，用于定义空的或不存在的引用。如果试图引用一个没有定义的变量，则返回一个null值。

由此可见，null与undefined的区别是，null表示一个变量被赋予了一个空值，而undefined则表示该变量尚未被赋值。

* 对象

JavaScript 对象是拥有属性和方法的数据。

var car = {type:"Fiat", model:500, color:"white"};

var person = {

firstName: "John",

lastName : "Doe",

id : 5566,

fullName : function()

{

return this.firstName + " " + this.lastName;

}

};

访问对象属性：

person.lastName;

person["lastName"];

访问对象方法：

name = person.fullName();

* + 1. JavaScript运算符及优先级

|  |  |  |
| --- | --- | --- |
| === | 绝对等于。根据表面值和数据类型同时进行判断 | "17"===17 /返回值为false |
| typeof | 返回它的操作数当前所容纳的数据的类型 | 判断一个变量是否已被定义 |
| new | 通过new运算符来创建一个新对象 | new constructor[(arguments)] |

|  |  |  |
| --- | --- | --- |
| **优先** | **结合** | **运算符** |
| 最高 | 向左 | .、[ ]、( ) |
| 由高到低依次排列 | 向右 | ++、--、-、!、delete、new、typeof、void |
| 向左 | \*、/、% |
| 向左 | +、- |
| 向左 | <<、>>、>>> |
| 向左 | <、<=、>、>=、in、instanceof |
| 向左 | ==、!=、===、!=== |
| 向左 | & |
| 向左 | ^ |
| 向左 | | |
| 向左 | && |
| 向左 | || |
| 向右 | ?: |
| 向右 | = |
| 向右 | \*=、/=、%=、+=、-=、<<=、>>=、>>>=、&=、^=、|= |
| 最低 | 向左 | , |

* + 1. JavaScript语句结构
* 赋值语句
* 条件控制语句
* 循环语句
* 异常捕获语句try catch
* 异常抛出语句throw
  1. JavaScript常用对象及方法
     1. String

|  |  |
| --- | --- |
| charAt() | 返回在指定位置的字符。 |
| split() | 把字符串分割为字符串数组。 |
| toUpperCase() | 把字符串转换为大写。 |
| toLowerCase() | 把字符串转换为小写。 |
| lastIndexOf() | 从后向前搜索字符串。 |
| substr() | 从起始索引号提取字符串中指定数目的字符。 |
| valueOf() | 返回某个字符串对象的原始值。 |
| charCodeAt() | 返回在指定的位置的字符的 Unicode 编码。 |
| toString() | 返回字符串。 |
| search() | 检索与正则表达式相匹配的值。 |
| indexOf() | 检索字符串。 |
| concat() | 连接字符串。 |
| slice() | 提取字符串的片断，并在新的字符串中返回被提取的部分。 |
| substring() | 提取字符串中两个指定的索引号之间的字符。 |
| replace() | 替换与正则表达式匹配的子串。 |
| match() | 找到一个或多个正则表达式的匹配。 |

* + 1. Array

创建 Array 对象的语法：

var array = [];

var array = new Array();

var array = new Array(size);

var array = [element0, element1, ..., elementn];

var array = new Array(element0, element1, ..., elementn);

|  |  |
| --- | --- |
| **方法** | **描述** |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_array.asp) | 连接两个或更多的数组，并返回结果。 |
| [join()](http://www.w3school.com.cn/jsref/jsref_join.asp) | 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。 |
| [pop()](http://www.w3school.com.cn/jsref/jsref_pop.asp) | 删除并返回数组的最后一个元素 |
| [push()](http://www.w3school.com.cn/jsref/jsref_push.asp) | 向数组的末尾添加一个或更多元素，并返回新的长度。 |
| [reverse()](http://www.w3school.com.cn/jsref/jsref_reverse.asp) | 颠倒数组中元素的顺序。 |
| [shift()](http://www.w3school.com.cn/jsref/jsref_shift.asp) | 删除并返回数组的第一个元素 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_array.asp) | 从某个已有的数组返回选定的元素 |
| [sort()](http://www.w3school.com.cn/jsref/jsref_sort.asp) | 对数组的元素进行排序 |
| [splice()](http://www.w3school.com.cn/jsref/jsref_splice.asp) | 删除元素，并向数组添加新元素。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_array.asp) | 把数组转换为字符串，并返回结果。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString_array.asp) | 把数组转换为本地数组，并返回结果。 |
| [unshift()](http://www.w3school.com.cn/jsref/jsref_unshift.asp) | 向数组的开头添加一个或更多元素，并返回新的长度。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_array.asp) | 返回数组对象的原始值 |

* + 1. Date

Date 对象用于处理日期和时间。

创建 Date 对象的语法：var myDate=new Date();

var myDate=new Date(2018,1,1);

|  |  |
| --- | --- |
| **方法** | **描述** |
| [getDate()](http://www.w3school.com.cn/jsref/jsref_getDate.asp) | 从 Date 对象返回一个月中的某一天 (1 ~ 31)。 |
| [getDay()](http://www.w3school.com.cn/jsref/jsref_getDay.asp) | 从 Date 对象返回一周中的某一天 (0 ~ 6)。 |
| [getMonth()](http://www.w3school.com.cn/jsref/jsref_getMonth.asp) | 从 Date 对象返回月份 (0 ~ 11)。 |
| [getFullYear()](http://www.w3school.com.cn/jsref/jsref_getFullYear.asp) | 从 Date 对象以四位数字返回年份。 |
| [getHours()](http://www.w3school.com.cn/jsref/jsref_getHours.asp) | 返回 Date 对象的小时 (0 ~ 23)。 |
| [getMinutes()](http://www.w3school.com.cn/jsref/jsref_getMinutes.asp) | 返回 Date 对象的分钟 (0 ~ 59)。 |
| [getSeconds()](http://www.w3school.com.cn/jsref/jsref_getSeconds.asp) | 返回 Date 对象的秒数 (0 ~ 59)。 |
| [getMilliseconds()](http://www.w3school.com.cn/jsref/jsref_getMilliseconds.asp) | 返回 Date 对象的毫秒(0 ~ 999)。 |
| [getTime()](http://www.w3school.com.cn/jsref/jsref_getTime.asp) | 返回 1970 年 1 月 1 日至今的毫秒数。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString.asp) | 根据本地时间格式，把 Date 对象转换为字符串。 |
| [toLocaleTimeString()](http://www.w3school.com.cn/jsref/jsref_toLocaleTimeString.asp) | 根据本地时间格式，把 Date 对象的时间部分转换为字符串。 |
| [toLocaleDateString()](http://www.w3school.com.cn/jsref/jsref_toLocaleDateString.asp) | 根据本地时间格式，把 Date 对象的日期部分转换为字符串。 |

* + 1. Math

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()。

通过把 Math 作为对象使用就可以调用其所有属性和方法。

|  |  |
| --- | --- |
| **方法** | **描述** |
| [abs(x)](http://www.w3school.com.cn/jsref/jsref_abs.asp) | 返回数的绝对值。 |
| [acos(x)](http://www.w3school.com.cn/jsref/jsref_acos.asp) | 返回数的反余弦值。 |
| [asin(x)](http://www.w3school.com.cn/jsref/jsref_asin.asp) | 返回数的反正弦值。 |
| [atan(x)](http://www.w3school.com.cn/jsref/jsref_atan.asp) | 以介于-PI/2与PI/2弧度之间的数值来返回x的反正切值。 |
| [atan2(y,x)](http://www.w3school.com.cn/jsref/jsref_atan2.asp) | 返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。 |
| [ceil(x)](http://www.w3school.com.cn/jsref/jsref_ceil.asp) | 对数进行上舍入。 |
| [cos(x)](http://www.w3school.com.cn/jsref/jsref_cos.asp) | 返回数的余弦。 |
| [exp(x)](http://www.w3school.com.cn/jsref/jsref_exp.asp) | 返回 e 的指数。 |
| [floor(x)](http://www.w3school.com.cn/jsref/jsref_floor.asp) | 对数进行下舍入。 |
| [log(x)](http://www.w3school.com.cn/jsref/jsref_log.asp) | 返回数的自然对数（底为e）。 |
| [max(x,y)](http://www.w3school.com.cn/jsref/jsref_max.asp) | 返回 x 和 y 中的最高值。 |
| [min(x,y)](http://www.w3school.com.cn/jsref/jsref_min.asp) | 返回 x 和 y 中的最低值。 |
| [pow(x,y)](http://www.w3school.com.cn/jsref/jsref_pow.asp) | 返回 x 的 y 次幂。 |
| [random()](http://www.w3school.com.cn/jsref/jsref_random.asp) | 返回 0 ~ 1 之间的随机数。 |
| [round(x)](http://www.w3school.com.cn/jsref/jsref_round.asp) | 把数四舍五入为最接近的整数。 |
| [sin(x)](http://www.w3school.com.cn/jsref/jsref_sin.asp) | 返回数的正弦。 |
| [sqrt(x)](http://www.w3school.com.cn/jsref/jsref_sqrt.asp) | 返回数的平方根。 |
| [tan(x)](http://www.w3school.com.cn/jsref/jsref_tan.asp) | 返回角的正切。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_math.asp) | 返回 Math 对象的原始值。 |

* + 1. RegExp

RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

* + - 1. 修饰符

|  |  |
| --- | --- |
| **修饰符** | **描述** |
| [i](http://www.w3school.com.cn/jsref/jsref_regexp_i.asp) | 执行对大小写不敏感的匹配。 |
| [g](http://www.w3school.com.cn/jsref/jsref_regexp_g.asp) | 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。 |
| m | 执行多行匹配。 |

* + - 1. 方括号

方括号用于查找某个范围内的字符：

|  |  |
| --- | --- |
| **表达式** | **描述** |
| [[abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset.asp) | 查找方括号之间的任何字符。 |
| [[^abc]](http://www.w3school.com.cn/jsref/jsref_regexp_charset_not.asp) | 查找任何不在方括号之间的字符。 |
| [0-9] | 查找任何从 0 至 9 的数字。 |
| [a-z] | 查找任何从小写 a 到小写 z 的字符。 |
| [A-Z] | 查找任何从大写 A 到大写 Z 的字符。 |
| [A-z] | 查找任何从大写 A 到小写 z 的字符。 |
| [adgk] | 查找给定集合内的任何字符。 |
| [^adgk] | 查找给定集合外的任何字符。 |
| (red|blue|green) | 查找任何指定的选项。 |

* + - 1. 元字符

元字符（Metacharacter）是拥有特殊含义的字符：

|  |  |
| --- | --- |
| **元字符** | **描述** |
| [.](http://www.w3school.com.cn/jsref/jsref_regexp_dot.asp) | 查找单个字符，除了换行和行结束符。 |
| \w | 查找单词字符。 |
| [\W](http://www.w3school.com.cn/jsref/jsref_regexp_wordchar_non.asp) | 查找非单词字符。 |
| [\d](http://www.w3school.com.cn/jsref/jsref_regexp_digit.asp) | 查找数字。 |
| [\D](http://www.w3school.com.cn/jsref/jsref_regexp_digit_non.asp) | 查找非数字字符。 |
| [\s](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace.asp) | 查找空白字符。 |
| [\S](http://www.w3school.com.cn/jsref/jsref_regexp_whitespace_non.asp) | 查找非空白字符。 |
| [\b](http://www.w3school.com.cn/jsref/jsref_regexp_begin.asp) | 匹配单词边界。 |
| [\B](http://www.w3school.com.cn/jsref/jsref_regexp_begin_not.asp) | 匹配非单词边界。 |
| \0 | 查找 NUL 字符。 |
| [\n](http://www.w3school.com.cn/jsref/jsref_regexp_newline.asp) | 查找换行符。 |
| \f | 查找换页符。 |
| \r | 查找回车符。 |
| \t | 查找制表符。 |
| \v | 查找垂直制表符。 |
| [\xxx](http://www.w3school.com.cn/jsref/jsref_regexp_octal.asp) | 查找以八进制数 xxx 规定的字符。 |
| [\xdd](http://www.w3school.com.cn/jsref/jsref_regexp_hex.asp) | 查找以十六进制数 dd 规定的字符。 |
| [\uxxxx](http://www.w3school.com.cn/jsref/jsref_regexp_unicode_hex.asp) | 查找以十六进制数 xxxx 规定的 Unicode 字符。 |

* + - 1. 量词

|  |  |
| --- | --- |
| **量词** | **描述** |
| [n+](http://www.w3school.com.cn/jsref/jsref_regexp_onemore.asp) | 匹配任何包含至少一个 n 的字符串。 |
| [n\*](http://www.w3school.com.cn/jsref/jsref_regexp_zeromore.asp) | 匹配任何包含零个或多个 n 的字符串。 |
| [n?](http://www.w3school.com.cn/jsref/jsref_regexp_zeroone.asp) | 匹配任何包含零个或一个 n 的字符串。 |
| [n{X}](http://www.w3school.com.cn/jsref/jsref_regexp_nx.asp) | 匹配包含 X 个 n 的序列的字符串。 |
| [n{X,Y}](http://www.w3school.com.cn/jsref/jsref_regexp_nxy.asp) | 匹配包含 X 至 Y 个 n 的序列的字符串。 |
| [n{X,}](http://www.w3school.com.cn/jsref/jsref_regexp_nxcomma.asp) | 匹配包含至少 X 个 n 的序列的字符串。 |
| [n$](http://www.w3school.com.cn/jsref/jsref_regexp_ndollar.asp) | 匹配任何结尾为 n 的字符串。 |
| [^n](http://www.w3school.com.cn/jsref/jsref_regexp_ncaret.asp) | 匹配任何开头为 n 的字符串。 |
| [?=n](http://www.w3school.com.cn/jsref/jsref_regexp_nfollow.asp) | 匹配任何其后紧接指定字符串 n 的字符串。 |
| [?!n](http://www.w3school.com.cn/jsref/jsref_regexp_nfollow_not.asp) | 匹配任何其后没有紧接指定字符串 n 的字符串。 |

* + - 1. RegExp 对象属性

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **描述** | **FF** | **IE** |
| [global](http://www.w3school.com.cn/jsref/jsref_regexp_global.asp) | RegExp 对象是否具有标志 g。 | 1 | 4 |
| [ignoreCase](http://www.w3school.com.cn/jsref/jsref_regexp_ignorecase.asp) | RegExp 对象是否具有标志 i。 | 1 | 4 |
| [lastIndex](http://www.w3school.com.cn/jsref/jsref_lastindex_regexp.asp) | 一个整数，标示开始下一次匹配的字符位置。 | 1 | 4 |
| [multiline](http://www.w3school.com.cn/jsref/jsref_multiline_regexp.asp) | RegExp 对象是否具有标志 m。 | 1 | 4 |
| [source](http://www.w3school.com.cn/jsref/jsref_source_regexp.asp) | 正则表达式的源文本。 | 1 | 4 |

* + - 1. RegExp 对象方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| [compile](http://www.w3school.com.cn/jsref/jsref_regexp_compile.asp) | 编译正则表达式。 | 1 | 4 |
| [exec](http://www.w3school.com.cn/jsref/jsref_exec_regexp.asp) | 检索字符串中指定的值。返回找到的值，并确定其位置。 | 1 | 4 |
| [test](http://www.w3school.com.cn/jsref/jsref_test_regexp.asp) | 检索字符串中指定的值。返回 true 或 false。 | 1 | 4 |

* + - 1. 支持正则表达式的 String 对象的方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| [search](http://www.w3school.com.cn/jsref/jsref_search.asp) | 检索与正则表达式相匹配的值。 | 1 | 4 |
| [match](http://www.w3school.com.cn/jsref/jsref_match.asp) | 找到一个或多个正则表达式的匹配。 | 1 | 4 |
| [replace](http://www.w3school.com.cn/jsref/jsref_replace.asp) | 替换与正则表达式匹配的子串。 | 1 | 4 |
| [split](http://www.w3school.com.cn/jsref/jsref_split.asp) | 把字符串分割为字符串数组。 | 1 | 4 |

* 1. JavaScript函数

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。

* + 1. 函数的定义

在JavaScript中，函数的定义是由关键字function、函数名加一组参数以及置于大括号中需要执行的一段代码定义的。定义函数的基本语法如下：

function functionName([parameter 1, parameter 2,……]){

statements;

[return expression;]

}

* functionName：必选，用于指定函数名。在同一个页面中，函数名必须是唯一的，并且区分大小写。
* parameter：可选，用于指定参数列表。当使用多个参数时，参数间使用逗号进行分隔。一个函数最多可以有255个参数。
* statements：必选，是函数体，用于实现函数功能的语句。
* expression：可选，用于返回函数值。expression为任意的表达式、变量或常量。
  + 1. 函数参数与返回值

参数列表中不需要添加var关键字。参数之间也必须用逗号“,”分隔。

返回值不需要提前声明，如果存在返回值在函数body中添加return关键字。

* + 1. 函数的调用

函数的定义语句通常被放在HTML文件的<head>段中，而函数的调用语句通常被放在<body>段中。

* 在body中调用函数
* 在事件响应中调用函数
* 通过链接调用函数

在<a>标签中的href标记中使用“javascript:关键字”格式来调用函数，当用户单击这个链接时，相关函数将被执行。

* 超链接与事件结合

能够有效隐藏超链接的URL。

* + 1. 内置函数

全局属性和函数可用于所有内建的 JavaScript 对象。

|  |  |
| --- | --- |
| **函数** | **描述** |
| [decodeURI()](http://www.w3school.com.cn/jsref/jsref_decodeURI.asp) | 解码某个编码的 URI。 |
| [encodeURI()](http://www.w3school.com.cn/jsref/jsref_encodeuri.asp) | 把字符串编码为 URI。 |
| [escape()](http://www.w3school.com.cn/jsref/jsref_escape.asp) | 对字符串进行编码。 |
| [eval()](http://www.w3school.com.cn/jsref/jsref_eval.asp) | 计算 JavaScript 字符串，作为脚本代码执行。 |
| [isFinite()](http://www.w3school.com.cn/jsref/jsref_isFinite.asp) | 检查某个值是否为有穷大的数。 |
| [isNaN()](http://www.w3school.com.cn/jsref/jsref_isNaN.asp) | 检查某个值是否是数字。 |
| [Number()](http://www.w3school.com.cn/jsref/jsref_number.asp) | 把对象的值转换为数字。 |
| [parseFloat()](http://www.w3school.com.cn/jsref/jsref_parseFloat.asp) | 解析一个字符串并返回一个浮点数。 |
| [parseInt()](http://www.w3school.com.cn/jsref/jsref_parseInt.asp) | 解析一个字符串并返回一个整数。 |
| [String()](http://www.w3school.com.cn/jsref/jsref_string.asp) | 把对象的值转换为字符串。 |
| [unescape()](http://www.w3school.com.cn/jsref/jsref_unescape.asp) | 对由 escape() 编码的字符串进行解码。 |

* 1. JavaScript事件

HTML DOM 使 JavaScript 有能力对 HTML 事件做出反应。事件处理是一段JavaScript代码，总是与页面中的特定部分以及一定的事件相关联。当与页面特定部分关联的事件发生时，事件处理器就会被调用。

* + 1. JavaScript事件处理程序的调用
* 在JavaScript中调用

在JavaScript中调用事件处理程序，首先需要：

1. 获得要处理对象的引用
2. 然后将要执行的处理函数赋值给对应的事件。

<input id="save" name="bt\_save" type="button" value="保存">

<script language="javascript">

var b\_save=document.getElementById("save");

b\_save.onclick=function(){

alert("单击了保存按钮");

}

</script>

* 在HTML中使用

在HTML中分配事件处理程序，只需要在HTML标记中添加相应的事件，并在其中指定要执行的代码或是函数名即可。

<input name="bt\_save" type="button" value="保存" onclick="clickFunction();">

function clickFunction(){

alert("单击了保存按钮");

}

* + 1. JavaScript鼠标事件

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| onclick | 当用户点击某个对象时调用的事件句柄。 |
| ondblclick | 当用户双击某个对象时调用的事件句柄。 |
| onmousedown | 鼠标按钮被按下。 |
| onmousemove | 鼠标被移动。 |
| onmouseout | 鼠标从某元素移开。 |
| onmouseover | 鼠标移到某元素之上。 |

* + 1. JavaScript键盘事件

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| onkeydown | 某个键盘按键被按下。 |
| onkeypress | 某个键盘按键被按下并松开。 |
| onkeyup | 某个键盘按键被松开。 |

* + 1. JavaScript页面事件

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| onload | 一张页面或一幅图像完成加载。 |
| onunload | 用户退出页面。 |
| onabort | 图像的加载被中断。 |

* + 1. JavaScript表单/编辑事件

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| onsubmit | 确认按钮被点击。 |
| onblur | 元素失去焦点。 |
| onchange | 域的内容被改变。 |
| onselect | 文本被选中。 |
| onfocus | 元素获得焦点。 |

* + 1. JavaScript事件模型

DOM（文档对象模型）结构是一个树型结构，当一个HTML元素产生一个事件时，该事件会在元素结点与根结点之间的路径传播，路径所经过的结点都会收到该事件，这个传播过程可称为DOM事件流。

* 捕获型事件(Capturing)：Netscape Navigator的实现，它与冒泡型刚好相反，由DOM树最顶层元素一直到最精确的元素。
* 冒泡型事件(Bubbling)：从DOM树型结构上理解，就是事件由叶子结点沿祖先结点一直向上传递直到根结点；从浏览器界面视图HTML元素排列层次上理解就是事件由具有从属关系的最确定的目标元素一直传递到最不确定的目标元素。
  + 1. JavaScript事件对象

在**IE浏览器**中事件对象是**window对象的一个属性event**，并且event对象只能在事件发生时候被访问，所有事件处理完后，该对象就消失了。而标准的DOM中规定event必须作为唯一的参数传给事件处理函数。故为了实现兼容性，通常采用下面的方法：

function someHandle(event) {

if(window.event)

event=window.event；

}

在IE中，触发事件的**对象**包含在event的**srcElement**属性中，而在标准的DOM浏览器中，对象包含在**target**属性中。为了处理两种浏览器兼容性：

function handle(oEvent){

if(window.event)

oEvent = window.event;//处理兼容性，获得事件对象

var oTarget;

if(oEvent.srcElement) //处理兼容性，获取事件目标

oTarget = oEvent.srcElement;

else

oTarget = oEvent.target;

alert(oTarget.tagName);//弹出目标的标记名称

}

window.onload = function(){

var oImg = document.getElementsByTagName(“img”)[0];

oImg.onclick = handle;

}

* + 1. JavaScript事件监听

实现DOM标准的浏览器与IE浏览器中注册元素事件监听器方式有所不同，它通过元素的addEventListener方法注册，该方法既支持注册冒泡型事件处理，又支持捕获型事件处理。

element.addEventListener(‘click’, observer, useCapture);

addEventListener方法接受三个参数。

第一个参数是事件名称，事件名称是没‘on’开头的；

第二个参数observer是回调处理函数；

第三个参数注明该处理回调函数是在事件传递过程中的捕获阶段被调用还是冒泡阶段被调用，默认true为捕获阶段。

移除已注册的事件监听器调用element的removeEventListener即可，参数不变。

element.removeEventListener(‘click’, observer, useCapture);

* 1. JavaScript BOM & DOM

浏览器对象模型 (BOM) 使 JavaScript 有能力与浏览器“对话”。

浏览器对象模型（Browser Object Model）尚无正式标准。

由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

所有浏览器都支持 window 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。全局变量是 window 对象的属性。全局函数是 window 对象的方法。

* + 1. window对象属性
       1. window尺寸

var w=window.innerWidth

|| document.documentElement.clientWidth

|| document.body.clientWidth;

var h=window.innerHeight

|| document.documentElement.clientHeight

|| document.body.clientHeight;

* + - 1. Window Screen

window.screen 对象在编写时可以不使用 window 这个前缀。

screen.availWidth - 可用的屏幕宽度

screen.availHeight - 可用的屏幕高度

* + - 1. Window Location

window.location 对象在编写时可不使用 window 这个前缀。

location.hostname 返回 web 主机的域名

location.pathname 返回当前页面的路径和文件名

location.port 返回 web 主机的端口 （80 或 443）

location.protocol 返回所使用的web协议（http://、https://）

location.href 属性返回当前页面的 URL。

location.assign(“URL”) 方法加载新的文档。

* + - 1. window.history

window.history 对象包含浏览器的历史。

history.back() - 与在浏览器点击后退按钮相同

history.forward() - 与在浏览器中点击按钮向前相同

* + 1. window对象方法

window.open() - 打开新窗口

window.close() - 关闭当前窗口

window.moveTo() - 移动当前窗口

window.resizeTo() - 调整当前窗口的尺寸

|  |  |
| --- | --- |
| alert() | 显示带有一段消息和一个确认按钮的警告框。 |
| confirm() | 显示带有一段消息以及确认按钮和取消按钮的对话框。 |
| prompt() | 显示可提示用户输入的对话框。 |

* + - 1. JavaScript 计时事件

通过使用 JavaScript，我们有能力作到在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。我们称之为计时事件。

在 JavaScritp 中使用计时事件是很容易的，两个关键方法是:

|  |  |
| --- | --- |
| setInterval() | 按照指定的周期（以毫秒计）来调用函数。 |
| setTimeout() | 在指定的毫秒数后调用函数或计算表达式。 |
| clearInterval() | 取消由 setInterval() 设置的 timeout。 |
| clearTimeout() | 取消由 setTimeout() 方法设置的 timeout。 |

* + 1. document对象属性

每个载入浏览器的 HTML 文档都会成为 Document 对象。

Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。

Document 对象集合

|  |  |
| --- | --- |
| **集合** | **描述** |
| forms[] | 返回对文档中所有 Form 对象引用。 |
| images[] | 返回对文档中所有 Image 对象引用。 |
| body | 提供对 <body> 元素的直接访问。 |
| alinkColor | 用来获取或设置当链接获得焦点时显示的颜色 |
| linkColor | 获取或设置页面中未单击的链接的颜色 |
| vlinkColor | 获取或设置页面中单击过的链接的颜色。 |
| bgColor | 获取或设置页面的背景颜色。 |
| fgColor | 获取或设置页面的前景颜色，即为页面中文字的颜色。 |

* + 1. document对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [getElementById()](http://www.w3school.com.cn/jsref/met_doc_getelementbyid.asp) | 返回对拥有指定 id 的第一个对象的引用。 |
| [getElementsByName()](http://www.w3school.com.cn/jsref/met_doc_getelementsbyname.asp) | 返回带有指定名称的对象集合。 |
| [getElementsByTagName()](http://www.w3school.com.cn/jsref/met_doc_getelementsbytagname.asp) | 返回带有指定标签名的对象集合。 |
| [write()](http://www.w3school.com.cn/jsref/met_doc_write.asp) | 向文档写 HTML 表达式 或 JavaScript 代码。 |
| [writeln()](http://www.w3school.com.cn/jsref/met_doc_writeln.asp) | 等同于 write() 方法，不同的是在每个表达式之后写一个换行符。 |

* 使用HTML 文本

document.getElementById("demo").innerHTML = "new Text";

* 改变值（属性）

document.getElementById("demo").value[property] = "value";

* 改变CSS样式

document.getElementById("demo").style.css = "new value";

* + 1. element属性与方法

|  |  |
| --- | --- |
| element.id | 设置或返回元素的 id。 |
| element.innerHTML | 设置或返回元素的内容。 |
| element.style | 设置或返回元素的 style 属性。 |
| element.parentNode | 返回元素的父节点。 |
| element.childNodes | 返回元素子节点的 NodeList。 |
| element.nodeValue | 设置或返回元素值。 |
| element.getAttribute() | 返回指定属性名的属性值。 |
| **element.setAttribute()** | 添加指定的属性，并为其赋值。 |
| element.removeAttribute() | 从元素中移除指定属性。 |
| element.insertBefore() | 在指定的已有的节点前插入新节点。 |
| element.removeChild() | 从元素中移除子节点。 |
| element.replaceChild() | 替换元素中的子节点。 |
| element.cloneNode() | 克隆元素。 |

* + 1. attribute属性与方法

|  |  |
| --- | --- |
| attr.name | 返回属性的名称。 |
| attr.value | 设置或返回属性的值。 |

* + 1. DOM节点操作
       1. 创建节点

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

如需向某个新添加的元素添加文本，您必须首先创建文本节点。

* 创建文本节点
* 创建元素节点
* 将文本节点添加给元素节点
* 获得网页中已经存在的某个节点（也可以是document.body）
* 向该节点添加新的元素节点
  + - 1. 创建文档碎片

使用循环语句通过使用appendChild()方法，将节点添加到页面中。由于 appendChild()方法在每一次添加新的节点时都会刷新页面，这会使浏览器显得十分缓慢。这里可以通过使用createDocumentFragment()方法来解决这个问题。createDocumentFragment()方法用来创建文件碎片节点。

var cdf=document.createDocumentFragment();//创建文件碎片

for(var i=0;i<aText.length;i++){ //循环创建新节点

var imgNode = document.createElement("img");

imgNode.setAttribute("src", "img/off.jpg"); cdf.appendChild(imgNode);

}

document.body.appendChild(cdf);

* + - 1. 删除节点

如需删除 HTML 元素，您必须首先获得该元素的父元素。

使用方法：parent.removeChild(child);

* + - 1. 节点的其他操作

cloneNode() 方法创建节点的拷贝，并返回该副本。

node.cloneNode(deep)

replaceChild() 方法用新节点替换某个子节点。

node.replaceChild(newnode,oldnode)

* 1. 实验
     1. 实现九九乘法表
     2. 逐行打印九九乘法表
     3. 实现开关灯

使用HTML以及JavaScript制作一个简易的开关灯示例。（所有的图片存放在网页当前目录的img文件夹下）

a) 页面中显示一张“灯灭了”的图(turn\_off.jpg)、一段文字（“灯灭了”）、一个按钮（按钮上显示“开灯”）；

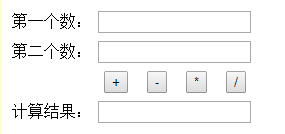
b) 点击按钮之后，图片换成“灯亮了”的图(turn\_on.jpg)，文字改变为“灯亮了”，按钮上的文字变为“关灯”；

c) 再次点击按钮之后，图片换回“灯灭了”的图(turn\_off.jpg)，文字改变为“灯灭了”，按钮上的文字变为“开灯”。

d) 当灯亮的时候，文字的颜色为红色；灯灭的时候，文字颜色为黑色。

* + 1. 实现随机彩灯
    2. 网页背景色切换
* 自动切换
* 下拉列表切换
  + 1. 实现网页计算器

使用HTML以及JavaScript代码制作一个简易的计算器，如图。



获得前两个文本框的值，点击按钮可以进行运算，运算结果显示在第三个文本框中；

进行数据检验（判断用户输入是否为空、是否为数字）；

判断除数是否为0。

请根据提示代码（提示代码见答题纸），完善上述程序。

* + 1. 金额大小写转换
    2. 制作个人相册

1. Ajax
   1. Ajax传统编程

在传统的Web应用模式中，页面中用户的每一次操作都将触发一次返回Web服务器的HTTP请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个HTML页面给客户端。

而在Ajax应用中，页面中用户的操作将通过Ajax引擎与服务器端进行通信，然后将返回结果提交给客户端页面的Ajax引擎，再由Ajax引擎来决定将这些数据插入到页面的指定位置。

* + 1. Ajax概述

AJAX（ Asynchronous JavaScript and XML）是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

AJAX 基于因特网标准，并使用以下技术组合：

* XMLHttpRequest 对象（与服务器异步交互数据）
* JavaScript/DOM（显示/取回信息）
* CSS（设置数据的样式）
* XML（常用作数据传输的格式）
  + 1. Ajax优点
* 无刷新更新数据

AJAX最大优点就是能在不刷新整个页面的前提下与服务器通信维护数据。

* 异步与服务器通信。

AJAX使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。

* 前端和后端负载平衡。

AJAX可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。

* 基于标准被广泛支持。

AJAX基于标准化的并被广泛支持的技术，不需要下载浏览器插件或者小程序，但需要客户允许JavaScript在浏览器上执行。

* 界面与应用分离。

Ajax使WEB中的界面与应用分离（也可以说是数据与呈现分离），有利于分工合作、减少非技术人员对页面的修改造成的WEB应用程序错误、提高效率、也更加适用于现在的发布系统。

* + 1. Ajax的缺点
* AJAX干掉了Back和History功能，即对浏览器机制的破坏。

在动态更新页面的情况下，用户无法回到前一个页面状态，因为浏览器仅能记忆历史记录中的静态页面。

* AJAX的安全问题。

Ajax技术就如同对企业数据建立了一个直接通道。这使得开发者在不经意间会暴露比以前更多的数据和服务器逻辑。

* 对搜索引擎支持较弱。

对搜索引擎的支持比较弱。如果使用不当，AJAX会增大网络数据的流量，从而降低整个系统的性能。

* 破坏程序的异常处理机制。

像Ajax.dll，Ajaxpro.dll这些Ajax框架是会破坏程序的异常机制的，给我们的调试带来了很大的困难。

* 违背URL和资源定位的初衷。
* AJAX不能很好支持移动设备。
* 客户端过肥，太多客户端代码造成开发上的成本。

编写复杂、容易出错 ；冗余代码比较多（层层包含js文件是AJAX的通病，再加上以往的很多服务端代码现在放到了客户端）；破坏了Web的原有标准。

* + 1. XMLHttpRequest对象

Ajax使用的技术中，最核心的技术就是XMLHttpRequest，它是一个具有应用程序接口的JavaScript对象，能够使用超文本传输协议（HTTP）连接一个服务器。

* + - 1. 初始化

IE浏览器把XMLHttpRequest实例化为一个ActiveX对象。具体方法如下：

var http\_request = new ActiveXObject("Msxml2.XMLHTTP");

或者

var http\_request = new ActiveXObject("Microsoft.XMLHTTP");

非IE浏览器（例如，Firefox、Opera、Mozilla、Safari）把XMLHttpRequest对象实例化为一个本地JavaScript对象。具体方法如下：

var http\_request = new XMLHttpRequest();

* + - 1. 属性

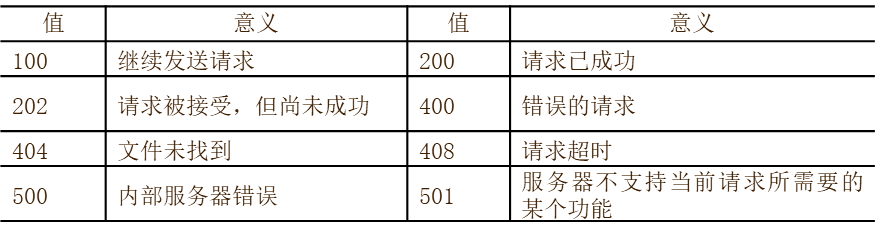
onreadystatechange：指定状态改变时所触发的事件处理器的属性。

readyState：用于获取请求状态的属性readyState。

responseText：用于获取服务器的文本（字符串）响应。

responseXML：用于获取服务器的XML文本响应。

status：用于返回服务器的HTTP状态码。



* + - 1. 方法

open()方法用于设置进行异步请求目标的URL、请求方法以及其他参数信息，具体语法如下：

open("method","URL"[,asyncFlag[,"userName"[, "password"]]])

|  |  |
| --- | --- |
| 参数名称 | 参数描述 |
| method | 用于指定请求的类型，一般为GET或POST |
| URL | 用于指定请求地址，可以使用绝对地址或者相对地址，并且可以传递查询字符串 |
| asyncFlag | 为可选参数，用于指定请求方式，异步请求为true，同步请求为false，默认情况下为true |
| userName | 为可选参数，用于指定请求用户名，没有时可省略 |
| password | 为可选参数，用于指定请求密码，没有时可省略 |

send()方法用于向服务器发送请求。如果请求声明为异步，该方法将立即返回，否则将等到接收到响应为止。send()方法的语法格式如下：

send(content)

* + 1. Ajax开发步骤
* 初始化
* 编写回调函数
* 创建请求
* 发送请求
* 初始化
* 创建请求
* 发送请求
* 处理返回信息
  1. Ajax与XML

在网页数据交互中，纯文本的方式很难体现出数据与数据之间的关系。

在学习数据结构的时候也存在类似的问题，因此如何获得比较复杂的数据结构中的数据是关键。

* + 1. XML语法与作用

XML 指可扩展标记语言（eXtensible Markup Language）。宗旨是传输数据，而非显示数据。没有预定义标签，需自行定义。

作用：

* 把数据从 HTML 分离
* 简化数据共享（纯文本）
* 简化数据传输

语法：

* 必须有根元素
* 必须有一个关闭标签
* 对大小写敏感
* 必须正确嵌套
* 特殊字符必须替换
* 空格会被保留
* 属性值必须加引号
  + 1. 读取XML节点与属性

DOM（Document Object Model 文档对象模型）定义了访问和操作文档的标准方法。

getElementsByTagName("TagName")[0].childNodes[0].nodeValue

getElementsByTagName("TagName")[0].getAttributeNode("name").nodeValue

* 1. Ajax与JSON
     1. JSON
        1. 概念

JSON: JavaScript Object Notation(JavaScript 对象表示法)

JSON 是存储和交换文本信息的语法。类似 XML。

JSON 比 XML 更小、更快，更易解析。

* + - 1. 语法规则
* 数据在名称：值
* 数据由逗号分隔
* 大括号{}保存对象
* 中括号[]保存数组
  + - 1. 编写示例

var sites = [

{ "name":"runoob" , "url":"www.runoob.com" },

{ "name":"google" , "url":"www.google.com" },

{ "name":"微博" , "url":"www.weibo.com" }

];

* + - 1. JSON转换为JavaScript对象
    1. Ajax读取JSON

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

* 使用 XML

读取 XML 文档

使用 XML DOM 来循环遍历文档

读取值并存储在变量中

* 使用 JSON

读取 JSON 字符串

用 eval() 处理 JSON 字符串

可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

{

"name": "张三",

"sex": "男",

"age":18,

"price":2000

}

//var person = JSON.parse(xmlhttp.responseText);

var person = eval("("+xmlhttp.responseText+")");

name.innerHTML=person.name;

sex.innerHTML=person.sex;

age.innerHTML=person.age;

* 1. 实验
     1. 异步/同步请求文本数据
     2. 异步/同步请求XML数据
     3. CD唱片清单
     4. 多级联动地址下拉列表

1. HTML5

HTML5 是下一代 HTML 标准。HTML5 是 W3C 与 WHATWG 合作的结果,WHATWG 指 Web Hypertext Application Technology Working Group。WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

HTML5 中的一些有趣的新特性：

* 用于绘画的 canvas 元素
* 用于媒介回放的 video 和 audio 元素
* 对本地离线存储的更好的支持
* 新的特殊内容元素，比如 article、footer、header、nav、section
* 新的表单控件，比如 calendar、date、time、email、url、search
  1. HTML5新特性

<!doctype> 声明必须位于 HTML5 文档中的第一行,使用非常简单:

<!DOCTYPE html>

HTML5 定了 8 个新的 HTML语义（semantic）元素。所有这些元素都是块级元素。为了能让旧版本的浏览器正确显示这些元素，你可以设置 CSS 的display属性值为block:

header, section, footer, aside, nav, main, article, figure {

display: block;

}

* + 1. HTML5自定义元素

可以为 HTML 添加新的元素。eg. <hero></hero>

* + 1. HTML5语义元素

HTML5提供了新的元素来创建更好的页面结构：

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<article>](http://www.runoob.com/tags/tag-article.html) | 定义页面独立的内容区域。 |
| [<aside>](http://www.runoob.com/tags/tag-aside.html) | 定义页面的侧边栏内容。 |
| <footer> | 定义 section 或 document 的页脚。 |
| <header> | 定义了文档的头部区域 |
| <nav> | 定义导航链接的部分。 |
| <time> | 定义日期或时间。 |

* + 1. HTML5废弃元素

以下的 HTML 4.01 元素在HTML5中已经被删除:

<acronym> <applet> <basefont> <big>

**<center> <dir> <frame> <frameset>**

**<font>**

<noframes>

<strike> <tt>

* + 1. HTML5表单属性

HTML5 的 <form> 和 <input>标签添加了几个新属性。

* <form>新属性：

autocomplete：规定 form 或 input 域应该拥有自动完成功能。

novalidate：规定在提交表单时不应该验证 form 或 input 域。

* <input>新属性：

autocomplete

autofocus：是一个 boolean 属性。规定在页面加载时，域自动地获得焦点。

form：规定输入域所属的一个或多个表单。

formaction：用于描述表单提交的URL地址。该属性会覆盖<form> 元素中的action属性。

formenctype：描述了表单提交到服务器的数据编码 (只对form表单中 method="post" 表单)。该属性覆盖 form 元素的 enctype 属性。

formmethod：属性定义了表单提交的方式。该属性覆盖了 <form> 元素的的method 属性。

formnovalidate：一个 boolean 属性。该属性描述了 <input> 元素在表单提交时无需被验证。该属性会覆盖 <form> 元素的novalidate属性.

formtarget：指定一个名称或一个关键字来指明表单提交数据接收后的展示。该属性覆盖 <form>元素的target属性。

height 与 width：规定用于 image 类型的 <input> 标签的图像高度和宽度。

list：属性规定输入域的 datalist。datalist 是输入域的选项列表。

min 与 max：min、max 和 step 属性用于为包含数字或日期的 input 类型规定限定（约束）。

multiple：一个 boolean 属性。规定<input> 元素中可选择多个值。

pattern (regexp)：描述了一个正则表达式用于验证 <input> 元素的值。

placeholder：提供一种提示（hint），描述输入域所期待的值。简短的提示在用户输入值前会显示在输入域上。

required：一个 boolean 属性。规定必须在提交之前填写输入域（不能为空）。

step：为输入域规定合法的数字间隔。

* + 1. HTML5表单元素

HTML5 有以下新的表单元素:

* <datalist>

<datalist> 元素规定输入域的选项列表。

<datalist>属性规定form或input域应该拥有自动完成功能。当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项。

* <keygen>

<keygen> 元素的作用是提供一种验证用户的可靠方法。

* <output>

<output> 标签定义不同类型的输出，比如脚本的输出。

* + 1. HTML5表单input控件

HTML5 拥有多个新的表单输入类型。这些新特性提供了更好的输入控制和验证。

* color：用在input字段主要用于选取颜色。
* date：允许你从期选择器选择一个日期。
* datetime：允许你选择一个日期（UTC 时间）。chrome不支持
* datetime-local：允许你选择一个日期和时间 (无时区)。
* email：用于应该包含 e-mail 地址的输入域。
* month：允许你选择一个月份。
* number：用于应该包含数值的输入域。
* range：用于应该包含一定范围内数字值的输入域。
* search：用于搜索域，比如站点搜索或 Google 搜索。
* tel：定义输入电话号码字段。
* time：允许你选择一个时间。
* url：用于应该包含 URL 地址的输入域。
* week：允许你选择周和年。
  1. Canvas

Canvas元素专门用来绘制图形。在页面上放置一个canvas元素，就相当于在页面上放置了一块“画布”，可以在其中进行图形的描绘。

可放到HTML页面中的最基本的canvas元素代码如下所示。

<canvas width="200" height="200"> </canvas>

假如要为其增加一个边框，可以用标准CSS边框属性来设置，代码如下。

<canvas id="djx" style="border: 1px solid;" width="200" height="200"> </canvas>

* + 1. Canvas绘图步骤
* 创建画布

通过引用特定的canvas id值来获取对canvas对象的访问权。

* 获得2D画笔

调用canvas对象的getContext方法，同时通过传入“2d”来获取一个二维上下文，这也是到目前为止唯一可用的上下文。

var canvas = document.getElementById('djx');

var context = canvas.getContext('2d');

* 开启路径

context.beginPath();

* 绘制路径

context.moveTo(70, 140);

context.lineTo(140, 70);

在结束canvas操作的时候，通过调用

* 关闭路径并绘制

context.closePath();

context.stroke()方法完成对角线的绘制。

context.stroke();

* + 1. context属性
* strokeStyle

图形边框的样式。在该属性中填入边框的颜色值。

* fillStyle属性

填充的样式，在该属性中填入填充的颜色值。

* lineWidth 属性

该属性设置当前绘线的粗细，属性值必须为正数。默认值是1.0。

* lineCap 属性

该属性决定了线段端点显示的样子。它可以为下面的三种值之一：butt，round 和 square，默认是 butt。

* lineJoin 属性

该属性值决定了图形中两线段连接处所显示的样子。它可以是以下三种值之一：round, bevel 和 miter。默认是 miter。

* shadowOffsetX：阴影的横向位移量
* shadowOffsetY：阴影的纵向位移量
* shadowBlur：阴影的模糊范围
* shadowColor：阴影的颜色
  + 1. 绘制图形
* fill() 填充当前绘图（路径）。
* stroke() 绘制已定义的路径。
* fillText() 绘制"被填充的"文本。
* strokeText() 绘制文本（无填充）。
* 画线条lineTo(x,y)
* 画矩形context.strokeRect(x,y,width,height);
* 填充矩形context.fillRect(x,y,width,height);
* 画圆形context.arc(x,y,radius, startAngle, endAngle,anticlockwise)
* 线性渐变

//创建线性渐变范围 起点x,y 终点x,y

var ling = context.createLinearGradient(250, 50, 250, 100);

ling.addColorStop(0, "red");

ling.addColorStop(1, "white");

// 使用线性渐变对象为画笔的填充属性赋值

context.fillStyle = ling;

context.fillRect(200, 10, 100, 200);

* 放射性渐变

var rad1 = ctx.createRadialGradient(50,150,50,50,50,20);

rad1.addColorStop(0,"red");

rad1.addColorStop(1,"black");

ctx.fillStyle = rad1;

ctx.fillRect(0,0,500,300);

ctx.strokeStyle = "white";

ctx.arc(50,50,20,0,Math.PI\*2,true);

ctx.arc(50,150,50,0,Math.PI\*2,true);

ctx.stroke();

* 贝塞尔曲线

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)

该方法使用六个参数。绘制贝塞尔曲线的时候，需要两个控制点，cp1x为第一个控制点的横坐标，cp1y为第一个控制点的纵坐标；cp2x为第二个控制点的横坐标，cp2y为第二个控制点的纵坐标；x为贝塞尔曲线的重点横坐标，y为贝塞尔曲线的终点纵坐标。

* 透明度

RGBa（255,255,255,0）

通过设置 globalAlpha属性或者使用一个半透明颜色作为轮廓或填充的样式来绘制透明或半透明的图形。globalAlpha属性定义代码如下所示：

globalAlpha = transparency value

这个属性影响到 canvas 里所有图形的透明度，其有效的值范围是 0.0 （完全透明）到 1.0（完全不透明），默认是 1.0。

* + 1. 绘制图像
* 在画布上定位图像：

context.drawImage(img,x,y);

* 在画布上定位图像，并规定图像的宽度和高度：

context.drawImage(img,x,y,width,height);

* 剪切图像，并在画布上定位被剪切的部分：

context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);

* + 1. SVG

SVG 指可伸缩矢量图形 (Scalable Vector Graphics)

SVG 用于定义用于网络的基于矢量的图形

SVG 使用 XML 格式定义图形

SVG 图像在放大或改变尺寸的情况下其图形质量不会有损失

SVG 是万维网联盟的标准

**优点：**

SVG 图像可通过文本编辑器来创建和修改

SVG 图像可被搜索、索引、脚本化或压缩

SVG 是可伸缩的

SVG 图像可在任何的分辨率下被高质量地打印

SVG 可在图像质量不下降的情况下被放大

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">

<polygon points="100,10 40,180 190,60 10,60 160,180" style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;">

</svg>

|  |  |
| --- | --- |
| **Canvas** | **SVG** |
| * 依赖分辨率 * 不支持事件处理器 * 弱的文本渲染能力 * 能够以 .png 或 .jpg 格式保存结果图像 * 最适合图像密集型的游戏，其中的许多对象会被频繁重绘 | * 不依赖分辨率 * 支持事件处理器 * 最适合带有大型渲染区域的应用程序（比如谷歌地图） * 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快） * 不适合游戏应用 |

SVG 有一些预定义的形状元素，可被开发者使用和操作：

* 矩形 <rect>
* 圆形 <circle>
* 椭圆 <ellipse>
* 线 <line>
* 折线 <polyline>
* 多边形 <polygon>
* 路径 <path>
  1. 多媒体元素
     1. 多媒体标签的使用

HTML5 提供了播放音频视频文件的标准。

* HTML5 规定了在网页上嵌入音频元素的标准，即使用 <audio> 元素。

<audio controls>

<source src="horse.ogg" type="audio/ogg">

<source src="horse.mp3" type="audio/mpeg">

浏览器不支持 audio 元素时显示的文本。

</audio>

上述可以简化为：

<audio src="horse.mp3" type="audio/mpeg" controls></audio>

* HTML5 规定了一种通过 video 元素来包含视频的标准方法。

<video width="320" height="240" controls>

<source src="movie.mp4" type="video/mp4">

<source src="movie.ogg" type="video/ogg">

您的浏览器不支持Video标签。

</video>

* + 1. 常见属性

src属性是指播放媒体的URL地址；

type属性表示媒体类型，其属性值为播放文件的MIME类型；

autoplay属性用于指定媒体是否在页面加载后自动播放；

perload属性用于指定视频或音频数据是否预加载。

poster（video元素独有属性）当视频不可用时，可以使用该元素向用户展示一幅替代用的图片。poster=”URL”

loop属性用于指定是否循环播放视频或音频。

controls属性指定是否为视频或音频添加浏览器自带的播放用的控制条。控制条中具有播放、暂停等按钮。

wdith属性和height属性（后两个video元素独有属性）用于指定视频的宽度与高度（以像素为单位）。

error属性：在读取、使用媒体数据的过程中只要出现错误，该属性将返回一个MediaError对象。

currentSrc属性来读取播放中的媒体数据的URL地址，该属性为只读属性。

buffered属性返回一个实现TimeRanges接口的对象，以确认浏览器是否已缓存媒体数据。

readyState属性返回媒体当前播放位置的就绪状态。readyState属性为只读属性。

seeking属性返回一个布尔值，表示浏览器是否正在请求某一特定播放位置的数据，true表示浏览器正在请求数据，false表示浏览器已停止请求。

currentTime属性用于读取媒体的当前播放位置，也可以通过修改currentTime属性来修改当前播放位置。

startTime属性用来读取媒体播放的开始时间，通常为“0”。

duration属性来读取媒体文件总的播放时间。

played属性返回一个TimeRanges对象，从该对象中可以读取媒体文件的已播放部分的时间段。

paused属性返回一个布尔值，表示是否暂停播放，true表示媒体暂停播放，false表示媒体正在播放。

ended属性返回一个布尔值，表示是否播放完毕，true表示媒体播放完毕，false表示还没有播放完毕。

defaultPlaybackRate属性用来读取或修改媒体默认的播放速率。

playbackRate属性用于读取或修改媒体当前的播放速率。

volume属性用于读取或修改媒体的播放音量，范围为0到1，0为静音，1为最大音量。

muted属性用于读取或修改媒体的静音状态，该值为布尔值，true表示处于静音状态，false表示处于非静音状态。

* + 1. 常用方法

media.play()播放视频，并将media.paused的值强行设为false。

media.pause()暂停视频，并将media.paused的值强行设为ture。

media.load()重新载入视频，并将media.playbackRate的值强行设为media.defaultPlaybackRate的值，且强行将media.error的值设为null。

* + 1. 常见事件

timeupdate事件：当前播放位置（currentTime属性）改变，可能是播放过程中的自然改变，也可能是被人为地改变，或由于播放不能连续而发生的跳变；

ended事件：播放由于媒介结束而停止；

play事件：即将开始播放，当执行了play方法时触发，或数据下载后元素被设为autoplay（自动播放）属性。

pause事件：暂停播放，当执行了pause方法时触发。

durationchange事件：媒介时长（duration属性）改变；

timeupdate事件：当前播放位置（currentTime属性）改变，可能是播放过程中的自然改变，也可能是被人为地改变，或由于播放不能连续而发生的跳变；

volumechange事件：音量（volume属性）改变或静音（muted属性）。

loadedmetadata事件：浏览器获取完媒介资源的时长和字节。

loadeddata事件：浏览器已加载当前播放位置的媒介数据。

* 1. 其他
     1. HTML5 location

使用 getCurrentPosition() 方法来获得用户的位置。

* + 1. 本地存储

HTML5 提供了两种在客户端存储数据的新方法：

localStorage - 没有时间限制的数据存储

sessionStorage - 针对一个 session 的数据存储。

<script>

// Check browser support

if (typeof(Storage) !== "undefined") {

// Store

**localStorage.setItem("lastname", "Gates");**

// Retrieve

document.getElementById("result").innerHTML = **localStorage.getItem("lastname");**

} else {

document.getElementById("result").innerHTML = "抱歉！您的浏览器不支持 Web Storage ...";

}

</script>

* + 1. 应用程序缓存（Application Cache）

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。

应用程序缓存为应用带来三个优势：

* 离线浏览 - 用户可在应用离线时使用它们
* 速度 - 已缓存资源加载得更快
* 减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

<!DOCTYPE HTML>

<html manifest="demo.appcache">

...

</html>

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

manifest 文件的建议的文件扩展名是：".appcache"。请注意，manifest 文件需要配置正确的 MIME-type，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

* + 1. Web Workers

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成。

web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行。**web worker 是运行在后台的 JavaScript，不会影响页面的性能。**

* 1. 实验
     1. 制作一个新元素构成的表单
     2. 绘制一幅五星红旗
     3. 制作一个在线播放器

1. jQuery

过去只有Flash才能实现的动画效果，jQuery也可以做到，而且丝毫不逊色于Flash，让开发人员感受到了Web 2.0时代的魅力。

* 1. jQuery基础
     1. 概述

jQuery是一个简洁快速的JavaScript脚本库，它能让你在网页上简单的操作文档、处理事件、运行动画效果或者添加异步交互。jQuery的设计会改变你写JavaScript代码的方式，提高我们的编程效率。

* + - 1. JQuery下载与配置

可以从它的官方网站（http://jQuery.com）中下载到。

在需要应用jQuery的页面中使用下面的语句，将其引用到文件中。

<script language="javascript" src="JS/jQuery-1.6.1.min.js"></script>

或者

<script src="JS/jQuery-1.6.1.min.js" type="text/javascript"></script>

* + - 1. jQuery特点
* 代码精致小巧
* 强大的功能函数
* 跨浏览器
* 链式的语法风格
* 插件丰富
  + - 1. JQuery的函数工厂

jQuery的工厂函数“$”。无论我们使用哪种类型的选择符都需要从一个“$”符号和一对“()”开始。在“()”中通常使用字符串参数，参数中可以包含任何CSS选择符表达式。

* + 1. 选择器
       1. JQuery基本选择器

元素选择器：$("div")：用于获取文档中全部的<div>。

ID选择器：$("#idname")：用于获取文档中ID属性值为idname的一个元素。

类名选择器：$(".classname")：用于获取文档中使用CSS类名为classname的所有元素。

通配符选择器$("\*")

* + - 1. JQuery复合选择器
* $(" selector1,selector2,selectorN");
* $("ancestor descendant");
* $("parent > child");
* $("prev + next");
* $("prev ~ siblings");
  + - 1. JQuery过滤选择器

简单过滤器是指以冒号开头，通常用于实现简单过滤效果的过滤器。

|  |  |  |
| --- | --- | --- |
| 过滤器 | 说明 | 示例 |
| :first | 匹配找到的第一个元素，它是与选择器结合使用的 | $("tr:first")//匹配表格的第一行 |
| :last | 匹配找到的最后一个元素，它是与选择器结合使用的 | $("tr:last")//匹配表格的最后一行 |
| :even | 匹配所有索引值为偶数的元素，索引值从0开始计数 | $("tr:even")//匹配索引值为偶数的行 |
| :odd | 匹配所有索引值为奇数的元素，索引从0开始计数 | $("tr:odd")//匹配索引值为奇数的行 |
| :eq(index) | 匹配一个给定索引值的元素 | $("tr:eq(1)")//匹配第二个div元素 |
| :gt(index) | 匹配所有大于给定索引值的元素 | $("tr:gt(0)")//匹配第二个及以上的div元素 |
| :lt(index) | 匹配所有小于给定索引值的元素 | $("tr:lt(2)")//匹配第二个及以下的div元素 |
| :header | 匹配如 h1, h2, h3……之类的标题元素 | $(":header")//匹配全部的标题元素 |
| :not(selector) | 去除所有与给定选择器匹配的元素 | $("input:not(:checked)") //匹配没有被选中的input元素 |
| :animated | 匹配所有正在执行动画效果的元素 | $(":animated ") //匹配所有正在执行的动画 |

内容过滤器就是通过DOM元素包含的文本内容以及是否含有匹配的元素进行筛选。

|  |  |  |
| --- | --- | --- |
| 过滤器 | 说明 | 示例 |
| contains(text) | 匹配包含给定文本的元素 | $("li:contains('DOM')")//匹配含有“DOM”文本内容的li元素 |
| :empty | 匹配所有不包含子元素或者文本的空元素 | $("td:empty")//匹配不包含子元素或者文本的单元格 |
| :has(selector) | 匹配含有选择器所匹配元素的元素 | $("td:has(p)")//匹配表格的单元格中含有<p>标记的单元格 |
| :parent | 匹配含有子元素或者文本的元素 | $("td: parent")//匹配不为空的单元格，即在该单元格中还包括子元素或者文本 |

元素的可见状态有两种，分别是隐藏状态和显示状态。可见性过滤器就是利用元素的可见状态匹配元素的。因此，可见性过滤器也有两种，一种是匹配所有可见元素的:visible过滤器，另一种是匹配所有不可见元素的:hidden过滤器。

* + - 1. JQuery表单选择器

表单对象的属性过滤器通过表单元素的状态属性（例如选中、不可用等状态）匹配元素，包括:checked过滤器、:disabled过滤器、:enabled过滤器和:selected过滤器4种。

|  |  |  |
| --- | --- | --- |
| 过滤器 | 说明 | 示例 |
| :checked | 匹配所有选中的被选中元素 | $("input:checked") //匹配checked属性为checked的input元素 |
| :disabled | 匹配所有不可用元素 | $("input:disabled") //匹配disabled属性为disabled的input元素 |
| :enabled | 匹配所有可用的元素 | $("input:enabled ") //匹配enabled属性为enabled的input元素 |
| :selected | 匹配所有选中的option元素 | $("select option:selected") //匹配select元素中被选中的option |

表单选择器是匹配经常在表单内出现的元素。但是匹配的元素不一定在表单中。

|  |  |  |
| --- | --- | --- |
| 选择器 | 说明 | 示例 |
| :input | 匹配所有的input元素 | $(":input") //匹配所有的input元素  $("form :input") //匹配<form>标记中的所有input元素，需要注意，在form和:之间有一个空格 |
| :button | 匹配所有的普通按钮，即type="button"的input元素 | $(":button") //匹配所有的普通按钮 |
| :checkbox | 匹配所有的复选框 | $(": checkbox") //匹配所有的复选框 |
| :file | 匹配所有的文件域 | $(": file") //匹配所有的文件域 |
| :hidden | 匹配所有的不可见元素，或者type为hidden的元素 | $(": hidden") //匹配所有的隐藏域 |
| :image | 匹配所有的图像域 | $(": image") //匹配所有的图像域 |
| :password | 匹配所有的密码域 | $(": password") //匹配所有的密码域 |
| :radio | 匹配所有的单选按钮 | $(": radio") //匹配所有的单选按钮 |
| :reset | 匹配所有的重置按钮，即type=" reset "的input元素 | $(": reset ") //匹配所有的重置按钮 |
| :submit | 匹配所有的提交按钮，即type=" submit "的input元素 | $(": submit") //匹配所有的提交按钮 |
| :text | 匹配所有的单行文本框 | $(":text") //匹配所有的单行文本框 |

* + - 1. JQuery索引器

JQuery有两个不同的索引器eq()和get()。索引器是从0开始计数的。

eq()方法返回的是一个jQuery包装集，所以它只能调用jQuery的方法，而get()方法返回的是一个DOM对象，所以它只能用DOM对象的方法。

$(“#test”).get(0)等效于$(“#test”)[0]。

（1）通过元素的id获得引用document.getElementById

（2）通过元素的class获得引用document.getElementsByClass

（3）通过元素的标签名获得引用document.getElementsBTagName

（4）通过元素的name获得引用document.getElementsByName

（5）使用JQuery的id选择器获得元素的对象引用$("#id")

（6）使用JQuery的class或者其他组合选择器获得元素的对象引用$(".class").eq(i)

（7）使用JQuery的过滤选择器获得元素的对象引用$(":text").eq(i)

* + 1. jQuery与HTML

由于元素内容又可分为文本内容和HTML内容，那么，对元素内容的操作也可以分为对文本内容操作和对HTML内容进行操作。

* + - 1. 修改网页文本值

$("div").text([val]);

$("div").html([val]);

* + - 1. 修改元素值

|  |  |  |
| --- | --- | --- |
| 方法 | 说明 | 示例 |
| val() | 用于获取第一个匹配元素的当前值，返回值可能是一个字符串，也可能是一个数组。例如当select元素有两个选中值时，返回结果就是一个数组 | $("#username").val();//获取id为username的元素的值 |
| val(val) | 用于设置所有匹配元素的值 | $("input:text").val("新值")//为全部文本框设置值 |
| val(Val\_array) | 用于为check、select和radio等元素设置值，参数为字符串数组 | $("select").val(['列表项1','列表项2']);//为下拉列表框设置多选值 |

* + - 1. 修改网页样式

addClass(class) 为所有匹配的元素添加指定的CSS类名

$("div").addClass("blue line"); //为全部div元素添加blue和line两个CSS类

removeClass(class) 从所有匹配的元素中删除全部或者指定的CSS类

$("div").addClass("line"); //删除全部div元素中添加的lineCSS类

toggleClass(class) 如果存在（不存在）就删除（添加）一个CSS类

$("div").toggleClass("yellow"); //当匹配的div元素中存在yellow CSS类，则删除该类，否则添加该CSS类

toggleClass(class,switch) 如果switch参数为true则加上对应的CSS类，否则就删除，通常switch参数为一个布尔型的变量

$("img").toggleClass("show",true); //为img元素添加CSS类show

$("img").toggleClass("show",false); //为img元素删除CSS类show

css(name) 返回第一个匹配元素的样式属性

$("div").css("color"); //获取第一个匹配的div元素的color属性值

css(name,value) 为所有匹配元素的指定样式设置值

$("img").css("border","1px solid #000000"); //为全部img元素设置边框样式

css(properties) 以{属性：值，属性：值，……}的形式为所有匹配的元素设置样式属性

$("tr").css({

"background-color":"#0A65F3",//设置背景颜色

"font-size":"14px", //设置字体大小

"color":"#FFFFFF" //设置字体颜色

});

* + - 1. 修改元素属性

attr(name) 获取匹配的第一个元素的属性值（无值时返回undefined）

$("img").attr('src'); //获取页面中第一个img元素的src属性的值

attr(key,value) 为所有匹配的元素设置一个属性值（value是设置的值）

$("img").attr("title","草莓正在生长"); //为图片添加一标题属性，属性值为“草莓正在生长”

attr(key,fn) 为所有匹配的元素设置一个函数返回的属性值（fn代表函数）

$("#fn").attr("value", function() {

return this.name ; //返回元素的名称

}); //将元素的名称作为其value属性值

attr(properties) 为所有匹配元素以集合（{名:值,名:值}）形式同时设置多个属性

//为图片同时添加两个属性，分别是src和title

$("img").attr({src:"test.gif",title:"图片示例"});

removeAttr(name) 为所有匹配元素删除一个属性

$("img"). removeAttr("title"); //移除所有图片的title属性

* + - 1. 操作网页节点
* 创建节点

//方法一

var $p=$("<p></p>");

$p.html("<span style='color:#FF0000'>添加的内容</span>");

$("body").append($p);

//方法二

var $txtP=$("<p><span style='color:#FF0000'>添加的内容</span></p>");

$("body").append($txtP);

//方法三

$("body").append("<p><span style='color:#FF0000'>内容</span></p>");

//弹出新添加的段落节点p的文本内容

alert($(“p”).text());

* 插入节点

append(content) 为所有匹配的元素的内部追加内容

$("#B").append("<p>A</p>"); //向id为B的元素中追加一个段落

appendTo(content)将所有匹配元素添加到另一个元素的元素集合中

$("#B").appendTo("#A");

//将id为B的元素追加到id为A的元素后面，也就是将B元素移动到A元素的后面

prepend(content) 为所有匹配的元素的内部前置内容

$("#B").prepend("<p>A</p>"); //向id为B的元素内容前添加一个段落

prependTo(content)将所有匹配元素前置到另一个元素的元素集合中

$("#B").prependTo("#A"); //将id为B的元素添加到id为A的元素前面，也就是将B元素移动到A元素的前面

* 在元素外部插入

在元素外部插入就是将要添加的内容添加到元素之前或元素之后。jQuery提供了如表19-7所示的在元素外部插入的方法。

after(content) 在每个匹配的元素之后插入内容

$("#B").after("<p>A</p>"); //向id为B的元素的后面添加一个段落

insertAfter(content) 将所有匹配的元素插入到另一个指定元素的元素集合的后面

$("<p>test</p>").insertAfter("#B"); //将要添加的段落插入到id为B的元素的后面

before(content) 在每个匹配的元素之前插入内容

$("#B").prepend("<p>A</p>"); //向id为B的元素内容前添加段落

insertBefore(content) 把所有匹配的元素插入到另一个指定元素的元素集合的前面

$("#B").prependTo("#A"); //将id为B的元素添加到id为A的元素前面，也就是将B元素移动到A元素的前面

* 删除、复制与替换节点

jQuery提供了两种删除节点的方法，分别是empty()和remove([expr])方法，其中，empty()方法用于删除匹配的元素集合中所有的子节点，并不删除该元素；remove([expr])方法用于从DOM中删除所有匹配的元素。

jQuery提供了clone()方法用于复制节点，该方法有两种形式，一种是不带参数，用于克隆匹配的DOM元素并且选中这些克隆的副本；另一种是带有一个布尔型的参数，当参数为true时，表示克隆匹配的元素以及其所有的事件处理并且选中这些克隆的副本，当参数为false时，表示不复制元素的事件处理。

jQuery提供了两个替换节点的方法，分别是replaceAll(selector)和replaceWith(content)。其中，replaceAll(selector)方法用于使用匹配的元素替换掉所有selector匹配到的元素；replaceWith(content)方法用于将所有匹配的元素替换成指定的HTML或DOM元素。这两种方法的功能相同，只是两者的表现形式不同。

* 1. jQuery事件与动画
     1. jQuery事件
        1. 页面加载事件

$(document).ready()方法是事件模块中最重要的一个函数，它极大地提高了Web响应速度。 $(document)是获取整个文档对象，从这个方法名称来理解，就是获取文档就绪的时候。方法的书写格式为：

$(document).ready(function() {

//在这里写代码

});

可以简写成：

$().ready(function() {

//在这里写代码

});

当$()不带参数时，默认的参数就是document，所以$()是$(document)的简写形式。

* + - 1. 常见事件

blur(fn) 在每一个匹配元素的blur事件中绑定一个处理函数，在元素失去焦点时触发，既可以是鼠标行为也可以是使用Tab键离开的行为

change(fn) 在每一个匹配元素的change事件中绑定一个处理函数，在元素的值改变并失去焦点时触发

click(fn) 在每一个匹配元素的click事件中绑定一个处理函数，在元素上单击时触发

dblclick(fn) 在每一个匹配元素的dblclick事件中绑定一个处理函数，在某个元素上双击触发

error(fn) 在每一个匹配元素的error事件中绑定一个处理函数，当JavaSprict发生错误时，会触发error()事件

focus(fn) 在每一个匹配元素的focus事件中绑定一个处理函数，当匹配的元素获得焦点时触，通过鼠标点击或者Tab键触发

keydown(fn) 在每一个匹配元素的keydown事件中绑定一个处理函数，当键盘按下时触发

keyup(fn) 在每一个匹配元素的keyup事件中绑定一个处理函数，会在按键释放时触发

keypress(fn) 在每一个匹配元素的keypress事件中绑定一个处理函数，敲击按键时触发（即按下并抬起同一个按键）

load(fn) 在每一个匹配元素的load事件中绑定一个处理函数，匹配的元素内容完全加载完毕后触发

mousedown(fn) 在每一个匹配元素的mousedown事件中绑定一个处理函数，鼠标在元素上点击后触发

mousemove(fn) 在每一个匹配元素的mousemove事件中绑定一个处理函数，鼠标在元素上移动时触发

mouseout(fn) 在每一个匹配元素的mouseout事件中绑定一个处理函数，鼠标从元素上离开时触发

mouseover(fn) 在每一个匹配元素的mouseover事件中绑定一个处理函数，鼠标移入对象时触发

mouseup(fn) 在每一个匹配元素的mouseup事件中绑定一个处理函数，鼠标点击对象释放时

resize(fn) 在每一个匹配元素的resize事件中绑定一个处理函数，当文档窗口改变大小时触发

scroll(fn) 在每一个匹配元素的scroll事件中绑定一个处理函数，当滚动条发生变化时触发

select(fn) 在每一个匹配元素的select事件中绑定一个处理函数，当用户在文本框（包括input和textarea）选中某段文本时触发

submit(fn) 在每一个匹配元素的submit事件中绑定一个处理函数，表单提交时触发

unload(fn) 在每一个匹配元素的unload事件中绑定一个处理函数，在元素卸载时触发该事件

* + - 1. 事件绑定与移除

在jQuery中，为元素绑定事件可以使用bind()方法，该方法的语法结构如下：

bind(type,[data],fn)

type：事件类型。

data：可选参数，作为event.data属性值传递给事件对象的额外数据对象。大多数的情况下不使用该参数。

fn：绑定的事件处理程序。

例如，为普通按钮绑定一个单击事件，用于在单击该按钮时，弹出提示对话框，可以使用下面的代码：

$("input:button").bind("click",function(){alert('您单击了按钮');});

在jQuery中，为元素移除绑定事件可以使用unbind()方法，该方法的语法结构如下：

unbind([type],[data])

type：可选参数，用于指定事件类型。

data：可选参数，用于指定要从每个匹配元素的事件中反绑定的事件处理函数。

例如，要移除为普通按钮绑定的单击事件，可以使用下面的代码：

$("input:button").unbind("click");

在jQuery中，为元素绑定一次性事件处理可以使用one()方法，该方法的语法结构如下：

one(type,[data],fn)

type：用于指定事件类型。

data：可选参数，作为event.data属性值传递给事件对象的额外数据对象。

fn：绑定到每个匹配元素的事件上面的处理函数。

例如，要实现只有当用户第一次单击匹配的div元素时，弹出提示对话框显示div元素的内容，可以使用下面的代码：

$("div").one("click", function(){

alert( $(this).text() );//在显示div元素的内容

});

* + - 1. 模拟用户操作

在jQuery中一般常用triggerHandler()方法和trigger()方法来模拟用户的操作触发事件。

triggerHandler()方法不会导致浏览器同名的默认行为被执行，而trigger()方法会导致浏览器同名的默认行为的执行。

$(document).ready(function() {

$("input:button").bind("click",function(event,msg1,msg2){

alert(msg1+msg2); //弹出提示对话框

}).trigger("click",["欢迎访问","科技"]); //页面加载触发单击事件

});

模仿悬停事件是指模仿鼠标移动到一个对象上面又从该对象上面移出的事件，可以通过jQuery提供的hover(over,out)方法实现。hover()方法的语法结构如下：

hover(over,out)

over：用于指定当鼠标在移动到匹配元素上时触发的函数。

out：用于指定当鼠标在移出匹配元素上时触发的函数。

例如，模拟隐藏超级链接地址，也可以使用下面的代码实现：

$("a.main").hover(function(){

window.status="http://www.mrbccd.com";

return true;//设定状态栏文本

},function(){

window.status="完成";

return true; //设定状态栏文本

});

模拟鼠标连续单击事件实际上是为每次单击鼠标时设置一个不同的函数。

toggle()方法会在第一次单击匹配的元素时，触发指定的第一个函数，下次单击这个元素时会触发指定的第二个函数，按此规律直到最后一个函数。随后的单击会按照原来的顺序循环触发指定的函数。toggle()方法的语法格式如下：

toggle(odd,even)

odd：用于指定奇数次单击按钮时触发的函数。

even：用于指定当偶数次单击按钮时触发的函数。

例如，要实现单击页面上的工具图片（id为tool的img元素），显示工具提示，再将单击时，隐藏工具提示可以使用下面的代码。

$("#tool").toggle(

function(){$("#tip").css("display","");},

function(){$("#tip").css("display","none");}

);

* + 1. jQuery动画
       1. 显示隐藏
* 使用hide()方法可以隐藏匹配的元素

$("img").hide();

另一种是带参数的形式，用于以优雅的动画隐藏所有匹配的元素，并在隐藏完成后可选地触发一个回调函数，其语法格式如下：

hide(speed,[callback])

speed：用于指定动画的时长。可以是数字，也就是元素经过多少毫秒（1000毫秒=1秒）后完全隐藏。

callback：可选参数，用于指定隐藏完成后要触发的回调函数。

例如，要在300毫秒内隐藏页面中的id为ad的元素，可以使用下面的代码：

$("#ad").hide(300);

* 使用show()方法可以显示匹配的元素

show()方法有两种语法格式，一种是不带参数的形式，用于实现不带任何效果的显示匹配元素，其语法格式如下：

show()

例如，要隐藏页面中的全部图片，可以使用下面的代码：

$("img").show();

另一种是带参数的形式，用于以优雅的动画隐藏所有匹配的元素，并在隐藏完成后可选择地触发一个回调函数，其语法格式如下：

show(speed,[callback])

speed：用于指定动画的时长。可以是数字，也就是元素经过多少毫秒（1000毫秒=1秒）后完全显示。

callback：可选参数，用于指定隐藏完成后要触发的回调函数。

例如，要在300毫秒内显示页面中的id为ad的元素，可以使用下面的代码：

$("#ad").show(300);

* 使用toggle()方法可以实现切换元素的可见状态
  + - 1. 淡入淡出

如果在显示或隐藏元素时不需要改变元素的高度和宽度，只单独改变元素的透明度的时候，就需要使用淡入淡出的动画效果了。

fadeIn(speed,[callback]) 通过增大不透明度实现匹配元素淡入的效果

$("img").fadeIn(300); //淡入效果

fadeOut(speed,[callback]) 通过减小不透明度实现匹配元素淡出的效果

$("img").fadeOut(300); //淡出效果

fadeTo(speed,opacity,[callback]) 将匹配元素的不透明度以渐进的方式调整到指定的参数

$("img").fadeTo(300,0.15); //在0.3秒内将图片淡入淡出至15%不透明

* + - 1. 滑入滑出

slideDown()方法（用于滑动显示匹配的元素）

slideUp()方法（用于滑动隐藏匹配的元素）

slideToggle()方法（用于通过高度的变化动态切换元素的可见性）来实现滑动效果。

slideDown(speed,[callback])

* + - 1. 自定义动画
* 使用animate()方法创建自定义动画

animate()方法的操作更加自由，可以随意控制元素的属性，实现更加绚丽的动画效果。animate()方法的基本语法格式如下：

animate(params,speed,callback)

params：表示一个包含属性和值的映射，可以同时包含多个属性，例如{left:"200px",top:"100px"}。

speed：表示动画运行的速度，参数规则同其他动画效果的speed一致，它是一个可选参数。

callback：表示一个回调函数，当动画效果运行完毕后执行该回调函数，它也是一个可选参数。

示例：

$(document).ready(function(){

$("#fish").animate({left:300},1000)

.animate({top:200},1000)

.animate({left:0},200)

.animate({top:0},200);

});

* 使用stop()方法停止动画

stop()方法也属于自定义动画函数，它会停止匹配元素正在运行的动画，并立即执行动画队列中的下一个动画。stop()方法的语法格式如下：

stop(clearQueue,gotoEnd)

clearQueue：表示**是否清空尚未执行完的动画队列**（值为true时表示清空动画队列）。

gotoEnd：表示是**否让正在执行的动画直接到达动画结束时的状态**（值为true时表示直接到达动画结束时状态）。

例如，需要停止某个正在执行的动画效果，清空动画序列并直接到达动画结束时的状态，只需在$(document).ready()方法中加入下面这句代码即可：

$("#btn\_stop").click(function(){

$("#fish").stop("true","true");//停止动画效果

});

* 1. JQuery与Ajax
     1. JSON

JSON由两个数据结构组成，一种是对象（“名称/值”形式的映射），另一种是数组（值的有序列表）。JSON没有变量或其他控制，只用于数据传输。

* 对象

在JSON中，可以使用下面的语法格式来定义对象。

{"属性1":属性值1,"属性2":属性值2……"属性n":属性值n}

* 数组

在JSON中，可以使用下面的语法格式来定义对象。

{"数组名":[

对象1,对象2……,对象n

]}

* + 1. 常用方法
       1. $.load()

load()方法通过AJAX请求从服务器加载数据，并把返回的数据放置到指定的元素中。它的语法格式如下：

.load( url [, data] [, complete(responseText, textStatus, XMLHttpRequest)] )

* + - 1. $.get()

get()方法

$.get()方法用于通过GET方式来进行异步请求，它的语法格式如下。

$.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType] )

* + - 1. $.post()

post()方法

$.post()方法用于通过POST方式进行异步请求，它的语法格式如下：

$.post( url [, data] [, success(data, textStatus, jqXHR)] [, dataType] )

* + - 1. $.ajax()

使用$.ajax()方法用户可以根据功能需求自定义Ajax操作，$.ajax()方法的语法格式如下。

$.ajax( url [, settings] )

type 用于指定请求方式，可以设置为GET或者POST，默认值为GET

data 用于指定发送到服务器的数据。如果数据不是字符串，将自动转换为请求字符串格式。在发送GET请求时，该数据将附加在URL的后面。设置processData参数值为false，可以禁止自动转换。该设置参数的值必须为key/value格式。如果为数组，jQuery将自动为不同值对应同一个名称。例如{foo:["bar1", "bar2"]}将转换为'&foo=bar1&foo=bar2'

dataType 用于指定服务器返回数据的类型。如果不指定，jQuery将自动根据HTTP包的MIME信息返回responseXML或responseText，并作为回调函数参数传递，可用值如下：

text：返回纯文本字符串。

xml：返回XML文档，可用jQuery进行处理。

html：返回纯文本HTML信息（包含的<script>元素会在插入DOM后执行）。

script：返回纯文本JavaScript代码。不会自动缓存结果，除非设置了cache参数。

json：返回JSON格式的数据。

jsonp：JSONP格式。使用JSONP形式调用函数时，如果存在代码“url?callback=?”，那么jQuery将自动替换?为正确的函数名，以执行回调函数。

async 设置发送请求的方式，默认是true，为异步请求方式，同步请求方式可以设置成false

beforeSend(jqXHR, settings) 用于设置一个发送请求前可以修改XMLHttpRequest对象的函数，例如，添加自定义HTTP头等

complete(jqXHR, textStatus) 用于设置一个请求完成后的回调函数，无论请求成功或失败，该函数均被调用

error(jqXHR, textStatus, errorThrown) 用于设置请求失败时调用的函数

success(data, textStatus, jqXHR) 用于设置请求成功时调用的函数

global 用于设置是否触发全局AJAX事件。设置为true，触发全局AJAX事件，设置为false则不触发全局AJAX事件，默认值为true

timeout 用于设置请求超时的时间（单位为毫秒）。此设置将覆盖全局设置

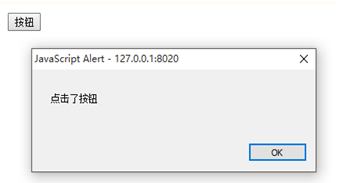
cache 用于设置是否从浏览器缓存中加载请求信息，设置为true将会从浏览器缓存中加载请求信息。默认值为true，当dataType的值为script和jsonp时值为false

dataFilter(data,type) 用于指定将Ajax返回的原始数据的进行预处理的函数。提供data和type两个参数：data是Ajax返回的原始数据，type是调用$.ajax()时提供的dataType参数。函数返回的值将由jQuery进一步处理

contentType 用于设置发送信息数据至服务器时内容编码类型，默认值为application/x-www-form-urlencoded，该默认值适用于大多数应用场合

ifModified 用于设置是否仅在服务器数据改变时获取新数据。使用HTTP包的Last-Modified头信息判断，默认值为false

* 1. 实验
     1. 实现多种按钮点击事件



(1) 在HTML中为按钮添加点击事件，并将对应的JavaScript编写出来。

(2) 在JavaScript代码中获得按钮对象，并为按钮动态添加点击事件和事件处理函数。

(3) 为按钮添加事件监听器，用于监听点击事件，并编写对应的回调函数。（提示：为元素添加事件监听器的方法为addEventListener）

(4) 假设网页已经引入JQuery脚本库，请使用JQuery技术实现按钮点击事件和对应的事件函数。

* + 1. jQuery动画

1. 实现显示和隐藏动画效果
2. 实现淡入淡出动画效果
3. 实现滑入滑出动画效果
4. 实现一个重复播放的动画效果
   * 1. jQuery与HTML
5. 使用jQuery实现开关灯
6. 使用jQuery实现计算器
7. 其他
   1. Bootstrap
   2. Easy UI
   3. node.js、vue.js
   4. 实验
      1. 企业门户网站
      2. 旅游网站前台
      3. 在线音乐播放器