# 第1章 JavaScript 和 HTML DOM 参考手册

JavaScript 参考手册

这些参考手册描述了所有 JavaScript 对象的属性和方法以及示例。

Array

Boolean

Classe

Date

Error

Global

JSON

Math

Number

RegExp

String

运算符

语句

HTML DOM 参考手册

这些参考手册描述了每个 DOM 对象的属性和方法以及实例。

Attributes

Document

Element

Events

Event

HTMLCollection 对象

Location

Navigator

Screen

Style

Window

Web API

该参考文献描述了最常见的 Web API 以及实例。

Console

Geolocation

History

Storage

HTML Element 对象参考手册

这些参考手册描述了每个 HTML 对象的属性和方法以及实例。

a

abbr

address

area

article

aside

audio

b

base

bdo

blockquote

body

br

button

canvas

caption

cite

code

col

colgroup

datalist

dd

del

details

dfn

dialog

div

dl

dt

em

embed

fieldset

figcaption

figure

footer

form

head

header

h1 - h6

hr

html

i

iframe

img

ins

input button

input checkbox

input color

input date

input datetime

input datetime-local

input email

input file

input hidden

input image

input month

input number

input password

input radio

input range

input reset

input search

input submit

input text

input time

input url

input week

kbd

label

legend

li

link

map

mark

menu

menuitem

meta

meter

nav

object

ol

optgroup

option

output

p

param

pre

progress

q

s

samp

script

section

select

small

source

span

strong

style

sub

summary

sup

table

tbody

td

tfoot

th

thead

tr

textarea

time

title

track

u

ul

var

video

其他

CSSStyleDeclaration

JS 类型转换

JavaScript 教程

访问我们的 JavaScript 教程

# 第2章 JavaScript 数组参考手册

Array 对象

Array 对象用于在单个变量中存储多个值：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript 数组</h1>

<p>Array 对象用于在单个变量中存储多个值：</p>

<p id="demo"></p>

<script>

const cars = ["Tesla", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = cars;

</script>

</body>

</html>

数组索引从零开始：数组中的第一个元素是 0，第二个元素是 1，依此类推。

有关数组的教程，请阅读我们的 JavaScript 数组教程。

Array 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 返回创建 Array 对象原型的函数。 |
| length | 设置或返回数组中元素的数量。 |
| prototype | 允许您向数组添加属性和方法。 |

Array 方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| concat() | 连接两个或多个数组，并返回已连接数组的副本。 |
| copyWithin() | 将数组中的数组元素复制到指定位置或从指定位置复制。 |
| entries() | 返回键/值对数组迭代对象。 |
| every() | 检查数组中的每个元素是否通过测试。 |
| fill() | 用静态值填充数组中的元素。 |
| filter() | 使用数组中通过测试的每个元素创建新数组。 |
| find() | 返回数组中第一个通过测试的元素的值。 |
| findIndex() | 返回数组中通过测试的第一个元素的索引。 |
| forEach() | 为每个数组元素调用函数。 |
| from() | 从对象创建数组。 |
| includes() | 检查数组是否包含指定的元素。 |
| indexOf() | 在数组中搜索元素并返回其位置。 |
| isArray() | 检查对象是否为数组。 |
| join() | 将数组的所有元素连接成一个字符串。 |
| keys() | 返回 Array Iteration 对象，包含原始数组的键. |
| lastIndexOf() | 在数组中搜索元素，从末尾开始，并返回其位置。 |
| map() | 使用为每个数组元素调用函数的结果创建新数组。 |
| pop() | 删除数组的最后一个元素，并返回该元素。 |
| push() | 将新元素添加到数组的末尾，并返回新的长度。 |
| reduce() | 将数组的值减为单个值（从左到右）。 |
| reduceRight() | 将数组的值减为单个值（从右到左）。 |
| reverse() | 反转数组中元素的顺序。 |
| shift() | 删除数组的第一个元素，并返回该元素。 |
| slice() | 选择数组的一部分，并返回新数组。 |
| some() | 检查数组中的任何元素是否通过测试。 |
| sort() | 对数组的元素进行排序。 |
| splice() | 从数组中添加/删除元素。 |
| toString() | 将数组转换为字符串，并返回结果。 |
| unshift() | 将新元素添加到数组的开头，并返回新的长度。 |
| valueOf() | 返回数组的原始值。 |

# 第3章 JavaScript Boolean 参考手册

JavaScript Boolean（布尔）

JavaScript 布尔值可以有以下两个值之一：true 或 false。

Boolean() 函数

您可以使用 Boolean() 函数来确定表达式是否为真：

实例

<!DOCTYPE html>

<html>

<body>

<p>显示 Boolean(10 > 9) 的值</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = Boolean(10 > 9);

}

</script>

</body>

</html>

或者更简单：

实例

<!DOCTYPE html>

<html>

<body>

<p>显示 10 > 9 的值</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = 10 > 9;

}

</script>

</body>

</html>

Boolean 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 返回创建 JavaScript Boolean 原型的函数。 |
| prototype | 允许您向 Boolean 原型添加属性和方法。 |

Boolean 方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| toString() | 将布尔值转换为字符串，并返回结果。 |
| valueOf() | 返回布尔值的原始值。 |

Boolean 对象

Boolean 对象表示两个值："true" 或 "false"。

创建 Boolean 对象的语法：

new Boolean(value); //构造函数

Boolean(value); //转换函数

参数

参数 value 由布尔对象存放的值或者要转换成布尔值的值。

返回值

当作为一个构造函数（带有运算符 new）调用时，Boolean() 将把它的参数转换成一个布尔值，并且返回一个包含该值的 Boolean 对象。

如果作为一个函数（不带有运算符 new）调用时，Boolean() 只将把它的参数转换成一个原始的布尔值，并且返回这个值。

注释：如果省略 value 参数，或者设置为 0、-0、null、""、false、undefined 或 NaN，则该对象设置为 false。否则设置为 true（即使 value 参数是字符串 "false"）。

Boolean 对象描述

在 JavaScript 中，布尔值是一种基本的数据类型。Boolean 对象是一个将布尔值打包的布尔对象。Boolean 对象主要用于提供将布尔值转换成字符串的 toString() 方法。

当调用 toString() 方法将布尔值转换成字符串时（通常是由 JavaScript 隐式地调用），JavaScript 会内在地将这个布尔值转换成一个临时的 Boolean 对象，然后调用这个对象的 toString() 方法。

课外书

如需更多信息，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 引用类型

引用类型通常叫做类（class）或对象。本节讲解 ECMAScript 的预定义引用类型。

# 第4章 JavaScript Class 参考手册

JavaScript 类

类是函数的一种，但我们不使用关键字 function 来对其初始化，而是使用关键字 class，并在 constructor() 方法中分配属性：

实例

创建一个 Car 类，然后基于这个 Car 类创建名为 "mycar" 的对象：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 类</h2>

<p>在这个例子中，我们演示了一个简单的类定义以及如何使用它。</p>

<p id="demo"></p>

<script>

class Car {

constructor(brand) {

this.carname = brand;

}

}

mycar = new Car("Tesla");

document.getElementById("demo").innerHTML = mycar.carname;

</script>

</body>

</html>

Class 方法

方法 描述

constructor() 用于创建和初始化在类中创建的对象的特殊方法。

Class 关键字

关键字 描述

extends 扩展类（继承）。

static 为类定义静态方法。

super 引用父类。

# 第5章 JavaScript Date 参考手册

Date 对象

Date 对象用于处理日期和时间。

日期对象是用 new Date() 创建的。

实例化日期有四种方式：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript new Date()</h1>

<p>new Date() 用当前日期和时间创建新的日期对象：</p>

<p id="demo"></p>

<script>

var d = new Date();

document.getElementById("demo").innerHTML = d;

</script>

</body>

</html>

Date 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 返回创建 Date 对象原型的函数。 |
| prototype | 允许您向对象添加属性和方法。 |

Date 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| getDate() | 返回月中的第几天（从 1 到 31）。 |
| getDay() | 返回星期几（0-6）。 |
| getFullYear() | 返回年份。 |
| getHours() | 返回小时（从 0-23）。 |
| getMilliseconds() | 返回毫秒（0-999）。 |
| getMinutes() | 返回分钟（从 0-59）。 |
| getMonth() | 返回月份（从 0-11）。 |
| getSeconds() | 返回秒数（从 0-59）。 |
| getTime() | 返回自 1970 年 1 月 1 日午夜以来与指定日期的毫秒数。 |
| getTimezoneOffset() | 返回 UTC 时间与本地时间之间的时差，以分钟为单位。 |
| getUTCDate() | 根据世界时，返回月份中的第几天（从 1 到 31）。 |
| getUTCDay() | 根据世界时，返回星期几（0-6）。 |
| getUTCFullYear() | 根据世界时，返回年份。 |
| getUTCHours() | 根据世界时，返回小时（0-23）。 |
| getUTCMilliseconds() | 根据世界时，返回毫秒数（0-999）。 |
| getUTCMinutes() | 根据世界时，返回分钟（0-59）。 |
| getUTCMonth() | 根据世界时，返回月份（0-11）。 |
| getUTCSeconds() | 根据世界时，返回秒数（0-59）。 |
| getYear() | 已弃用。请改用 getFullYear() 方法。 |
| now() | 返回自 1970 年 1 月 1 日午夜以来的毫秒数。 |
| parse() | 解析日期字符串并返回自 1970 年 1 月 1 日以来的毫秒数。 |
| setDate() | 设置 Date 对象中月的某一天。 |
| setFullYear() | 设置日期对象的年份 |
| setHours() | 设置日期对象的小时。 |
| setMilliseconds() | 设置日期对象的毫秒数。 |
| setMinutes() | 设置日期对象的分钟数。 |
| setMonth() | 设置日期对象的月份。 |
| setSeconds() | 设置日期对象的秒数。 |
| setTime() | 将日期设置为 1970 年 1 月 1 日之后/之前的指定毫秒数。 |
| setUTCDate() | 根据世界时，设置 Date 对象中月份的一天。 |
| setUTCFullYear() | 根据世界时，设置日期对象的年份。 |
| setUTCHours() | 根据世界时，设置日期对象的小时。 |
| setUTCMilliseconds() | 根据世界时，设置日期对象的毫秒数。 |
| setUTCMinutes() | 根据世界时，设置日期对象的分钟数。 |
| setUTCMonth() | 根据世界时，设置日期对象的月份。 |
| setUTCSeconds() | 根据世界时，设置日期对象的秒数。 |
| setYear() | 已弃用。请改用 setFullYear() 方法。 |
| toDateString() | 将 Date 对象的日期部分转换为可读字符串。 |
| toGMTString() | 已弃用。请改用 toUTCString() 方法。 |
| toISOString() | 使用 ISO 标准将日期作为字符串返回。 |
| toJSON() | 以字符串形式返回日期，格式为 JSON 日期。 |
| toLocaleDateString() | 使用区域设置约定将 Date 对象的日期部分作为字符串返回。 |
| toLocaleTimeString() | 使用区域设置约定将 Date 对象的时间部分作为字符串返回。 |
| toLocaleString() | 使用区域设置约定将 Date 对象转换为字符串。 |
| toString() | 将 Date 对象转换为字符串。 |
| toTimeString() | 将 Date 对象的时间部分转换为字符串。 |
| toUTCString() | 根据世界时，将 Date 对象转换为字符串。 |
| UTC() | 根据 UTC 时间，返回自 1970 年 1 月 1 日午夜以来的日期中的毫秒数。 |
| valueOf() | 返回 Date 对象的原始值。 |

# 第6章 JavaScript Error 参考手册

Error 对象

Error 对象提供发生错误时的错误信息。

实例

在此例中，我们将 "alert" 写为 "adddlert" 以故意产生错误。

返回错误名称和错误描述：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript 错误</h1>

<p>在这个例子中，我们将 “alert” 写成 “adddlert” 来故意产生错误。</p>

<p>Error 对象的 name 属性返回错误的名称，而 message 属性返回错误的描述：</p>

<p id="demo" style="color:red"></p>

<script>

try {

adddlert("Welcome guest!");

}

catch(err) {

document.getElementById("demo").innerHTML =

err.name + "<br>" + err.message;

}

</script>

</body>

</html>

Error 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| name | 设置或返回错误名称。 |
| message | 设置或返回错误消息（字符串）。 |
|  |  |

非标准 Error 对象属性

Mozilla 和 Microsoft 定义了一些非标准的 error 对象属性：

fileName (Mozilla)

lineNumber (Mozilla)

columnNumber (Mozilla)

stack (Mozilla)

description (Microsoft)

number (Microsoft)

请勿在公共网站上使用这些属性。它们并非在所有浏览器中都适用。

# 第7章 JavaScript 全局参考手册

JavaScript 全局属性和函数可用于所有内置 JavaScript 对象。

JavaScript 全局属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| Infinity | 表示正/负无穷大的数值。 |
| NaN | “非数字”（"Not-a-Number"）值。 |
| undefined | 表示变量尚未赋值。 |

JavaScript 全局函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| decodeURI() | 解码 URI。 |
| decodeURIComponent() | 解码 URI 组件。 |
| encodeURI() | 对 URI 进行编码。 |
| encodeURIComponent() | 对 URI 组件进行编码。 |
| escape() | 在 1.5 版中已弃用。请使用 encodeURI() 或 encodeURIComponent() 代替。 |
| eval() | 评估字符串并像脚本代码一样执行它。 |
| isFinite() | 确定值是否是有限的合法数。 |
| isNaN() | 确定值是否是非法数字。 |
| Number() | 将对象的值转换为数字。 |
| parseFloat() | 解析字符串并返回浮点数。 |
| parseInt() | 解析字符串并返回整数。 |
| String() | 将对象的值转换为字符串。 |
| unescape() | 在 1.5 版中已弃用。请使用 decodeURI() 或 decodeURIComponent() 代替。 |

函数还是方法？

调用上面列表中的全局函数而不是全局方法是有意义的，因为函数是全局调用的，而非任何对象。

无论如何，您也可以调用这些函数方法，因为它们是其运行环境的全局对象的方法。在 Web 浏览器中，全局对象是浏览器窗口。那么 isNaN() 其实就是一个窗口方法：window.isNaN()。

全局对象描述

全局对象是预定义的对象，作为 JavaScript 的全局函数和全局属性的占位符。通过使用全局对象，可以访问所有其他所有预定义的对象、函数和属性。全局对象不是任何对象的属性，所以它没有名称。

在顶层 JavaScript 代码中，可以用关键字 this 引用全局对象。但通常不必用这种方式引用全局对象，因为全局对象是作用域链的头，这意味着所有非限定性的变量和函数名都会作为该对象的属性来查询。例如，当JavaScript 代码引用 parseInt() 函数时，它引用的是全局对象的 parseInt 属性。全局对象是作用域链的头，还意味着在顶层 JavaScript 代码中声明的所有变量都将成为全局对象的属性。

全局对象只是一个对象，而不是类。既没有构造函数，也无法实例化一个新的全局对象。

在 JavaScript 代码嵌入一个特殊环境中时，全局对象通常具有环境特定的属性。实际上，ECMAScript 标准没有规定全局对象的类型，JavaScript 的实现或嵌入的 JavaScript 都可以把任意类型的对象作为全局对象，只要该对象定义了这里列出的基本属性和函数。例如，在允许通过 LiveConnect 或相关的技术来脚本化 Java 的 JavaScript 实现中，全局对象被赋予了这里列出的 java 和 Package 属性以及 getClass() 方法。而在客户端 JavaScript 中，全局对象就是 Window 对象，表示允许 JavaScript 代码的 Web 浏览器窗口。

例子

在 JavaScript 核心语言中，全局对象的预定义属性都是不可枚举的，所有可以用 for/in 循环列出所有隐式或显式声明的全局变量，如下所示：

<html>

<body>

<script type="text/javascript">

var variables = "";

for (var name in this)

{

variables += name + "<br />";

}

document.write(variables);

</script>

</body>

</html>

# 第8章 JavaScript JSON 参考手册

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

JSON 是用于存储和传输数据的格式。

JSON 是文本，文本可以在任何地方传输，并可通过任何编程语言读取。

JavaScript 对象可以转换为 JSON，JSON 可以转换回 JavaScript 对象。

这样，我们可以将数据作为 JavaScript 对象使用，而无需复杂的解析或转换。

实例

发送 JSON：

<!DOCTYPE html>

<html>

<body>

<h1>将 JavaScript 对象转换为 JSON 字符串，并将其发送到服务器</h1>

<script>

var myObj = { "name":"Bill", "age":19, "city":"Seattle" };

var myJSON = JSON.stringify(myObj);

window.location = "demo\_json.php?x=" + myJSON;

</script>

</body>

</html>

JSON 方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| parse() | 解析 JSON 字符串并返回 JavaScript 对象。 |
| stringify() | 将 JavaScript 对象转换为 JSON 字符串。 |

有效数据类型

在 JSON 中，值必须是以下数据类型之一：

字符串

数字

对象（包含有效的 JSON 值）

数组

布尔

null

JSON 值不能是以下数据类型之一：

函数

日期

undefined

更多实例

实例

接收 JSON：

<!DOCTYPE html>

<html>

<body>

<h1>将以 JSON 格式编写的字符串转换为 JavaScript 对象</h1>

<p id="demo"></p>

<script>

var myJSON = '{ "name":"Bill", "age":19, "city":"Seattle" }';

var myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.name;

</script>

</body>

</html>

实例

使用 localStorage 将数据存储为 JSON：

<!DOCTYPE html>

<html>

<body>

<h1>从 local storage 存储和检索数据</h1>

<p id="demo"></p>

<script>

var myObj, myJSON, text, obj;

//Storing data:

myObj = { "name":"Bill", "age":19, "city":"Seattle" };

myJSON = JSON.stringify(myObj);

localStorage.setItem("testJSON", myJSON);

//Retrieving data:

text = localStorage.getItem("testJSON");

obj = JSON.parse(text);

document.getElementById("demo").innerHTML = obj.name;

</script>

</body>

</html>

# 第9章 JavaScript Math 参考手册

Math 对象

Math 对象允许您执行数学任务。

Math 不是构造函数。Math 的所有属性/方法都可以通过使用 Math 作为对象来调用，而无需创建它：

var x = Math.PI; // 返回 PI

var y = Math.sqrt(16); // 返回 16 的平方根

Math 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| E | 返回欧拉数（约 2.718）。 |
| LN2 | 返回 2 的自然对数（约 0.693）。 |
| LN10 | 返回 10 的自然对数（约 2.302）。 |
| LOG2E | 返回 E 的以 2 为底的对数（约 1.442）。 |
| LOG10E | 返回 E 的以 10 为底的对数（约 0.434）。 |
| PI | 返回 PI（约 3.14）。 |
| SQRT1\_2 | 返回 1/2 的平方根（约 0.707）。 |
| SQRT2 | 返回 2 的平方根（约 1.414）。 |

Math 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| abs(x) | 返回 x 的绝对值。 |
| acos(x) | 返回 x 的反余弦值，以弧度为单位。 |
| acosh(x) | 返回 x 的双曲反余弦值。 |
| asin(x) | 返回 x 的反正弦值，以弧度为单位。 |
| asinh(x) | 返回 x 的双曲反正弦值。 |
| atan(x) | 返回 x 的反正切值，返回的值是 -PI/2 到 PI/2 之间的弧度值。 |
| atan2(y, x) | 返回其参数商的反正切值。 |
| atanh(x) | 返回 x 的双曲反正切值。 |
| cbrt(x) | 返回 x 的三次方根。 |
| ceil(x) | 返回 x，向上舍入为最接近的整数。 |
| clz32(x) | 返回 x 的 32 位二进制表示中前导零的数量。 |
| cos(x) | 返回 x 的余弦值（x 以弧度为单位）。 |
| cosh(x) | 返回 x 的双曲余弦值。 |
| exp(x) | 返回 Ex 的值。 |
| expm1(x) | 返回 Ex 减去 1 的值。 |
| floor(x) | 返回 x，向下舍入为最接近的整数。 |
| fround(x) | 返回数的最接近的（32 位单精度）浮点表示。 |
| log(x) | 返回 x 的自然对数。 |
| log10(x) | 返回 x 的以 10 为底的对数。 |
| log1p(x) | 返回 1 + x 的自然对数。 |
| log2(x) | 返回 x 的以 2 为底的对数。 |
| max(x, y, z, ..., n) | 返回值最高的数字。 |
| min(x, y, z, ..., n) | 返回值最小的数字。 |
| pow(x, y) | 返回 x 的 y 次幂值。 |
| random() | 返回 0 到 1 之间的随机数。 |
| round(x) | 将 x 舍入为最接近的整数。 |
| sign(x) | 返回数的符号（检查它是正数、负数还是零）。 |
| sin(x) | 返回 x 的正弦值（x 以弧度为单位）。 |
| sinh(x) | 返回 x 的双曲正弦值。 |
| sqrt(x) | 返回 x 的平方根。 |
| tan(x) | 返回角度的正切值。 |
| tanh(x) | 返回数的双曲正切值。 |
| trunc(x) | 返回数字 (x) 的整数部分。 |

# 第10章 JavaScript Number 参考手册

Number 对象

Number 对象是原始数值的包装对象。

创建 Number 对象的语法：

var myNum=new Number(value);

var myNum=Number(value);

参数

参数 value 是要创建的 Number 对象的数值，或是要转换成数字的值。

返回值

当 Number() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 Number 对象。如果不用 new 运算符，把 Number() 作为一个函数来调用，它将把自己的参数转换成一个原始的数值，并且返回这个值（如果转换失败，则返回 NaN）。

Number 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 返回对创建此对象的 Number 函数的引用。 |
| MAX\_VALUE | 可表示的最大的数。 |
| MIN\_VALUE | 可表示的最小的数。 |
| NaN | 非数字值。 |
| NEGATIVE\_INFINITY | 负无穷大，溢出时返回该值。 |
| POSITIVE\_INFINITY | 正无穷大，溢出时返回该值。 |
| prototype | 使您有能力向对象添加属性和方法。 |

Number 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| toString | 把数字转换为字符串，使用指定的基数。 |
| toLocaleString | 把数字转换为字符串，使用本地数字格式顺序。 |
| toFixed | 把数字转换为字符串，结果的小数点后有指定位数的数字。 |
| toExponential | 把对象的值转换为指数计数法。 |
| toPrecision | 把数字格式化为指定的长度。 |

Number 对象描述

在 JavaScript 中，数字是一种基本的数据类型。JavaScript 还支持 Number 对象，该对象是原始数值的包装对象。在必要时，JavaScript 会自动地在原始数据和对象之间转换。在 JavaScript 1.1 中，可以用构造函数 Number() 明确地创建一个 Number 对象，尽管这样做并没有什么必要。

构造函数 Number() 可以不与运算符 new 一起使用，而直接作为转化函数来使用。以这种方式调用 Number() 时，它会把自己的参数转化成一个数字，然后返回转换后的原始数值（或 NaN）。

构造函数通常还用作 5 个有用的数字常量的占位符，这 5 个有用的数字常量分别是可表示的最大数、可表示的最小数、正无穷大、负无穷大和特殊的 NaN 值。 注意，这些值是构造函数 Number() 自身的属性，而不是单独的某个 Number 对象的属性。

比如这样使用属性 MAX\_VALUE 是正确的：

var big = Number.MAX\_VALUE

但是这样是错误的：

var n= new Number(2);

var big = n.MAX\_VALUE

作为比较，我们看一下 toString() 和 Number 对象的其他方法，它们是每个 Number 对象的方法，而不是 Number() 构造函数的方法。前面提到过，在必要时，JavaScript 会自动地把原始数值转化成 Number 对象，调用 Number 方法的既可以是 Number 对象，也可以是原始数字值。

var n = 123;

var binary\_value = n.toString(2);

课外书

如需更多信息，请阅读 JavaScript 高级教程中的相关内容：

ECMAScript 引用类型

引用类型通常叫做类（class）或对象。本节讲解 ECMAScript 的预定义引用类型。

# 第11章 JavaScript RegExp 参考手册

RegExp 对象

RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

直接量语法

/pattern/attributes

创建 RegExp 对象的语法：

new RegExp(pattern, attributes);

参数

参数 pattern 是一个字符串，指定了正则表达式的模式或其他正则表达式。

参数 attributes 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 pattern 是正则表达式，而不是字符串，则必须省略该参数。

返回值

一个新的 RegExp 对象，具有指定的模式和标志。如果参数 pattern 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。

如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 pattern 是正则表达式时，它只返回 pattern，而不再创建一个新的 RegExp 对象。

抛出

SyntaxError - 如果 pattern 不是合法的正则表达式，或 attributes 含有 "g"、"i" 和 "m" 之外的字符，抛出该异常。

TypeError - 如果 pattern 是 RegExp 对象，但没有省略 attributes 参数，抛出该异常。

修饰符

|  |  |
| --- | --- |
| 修饰符 | 描述 |
| i | 执行对大小写不敏感的匹配。 |
| g | 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。 |

方括号

方括号用于查找某个范围内的字符：

|  |  |
| --- | --- |
| 表达式 | 描述 |
| [abc] | 查找方括号之间的任何字符。 |
| [^abc] | 查找任何不在方括号之间的字符。 |
| [0-9] | 查找任何从 0 至 9 的数字。 |
| [a-z] | 查找任何从小写 a 到小写 z 的字符。 |
| [A-Z] | 查找任何从大写 A 到大写 Z 的字符。 |
| [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。 |

元字符

元字符（Metacharacter）是拥有特殊含义的字符：

|  |  |
| --- | --- |
| 元字符 | 描述 |
| . | 查找单个字符，除了换行和行结束符。 |
| \w | 查找单词字符。 |
| \W | 查找非单词字符。 |
| \d | 查找数字。 |
| \D | 查找非数字字符。 |
| \s | 查找空白字符。 |
| \S | 查找非空白字符。 |
| \b | 匹配单词边界。 |
| \B | 匹配非单词边界。 |
| \0 | 查找 NUL 字符。 |
| \n | 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 | 查找以八进制数 xxx 规定的字符。 |
| \xdd | 查找以十六进制数 dd 规定的字符。 |
| \uxxxx | 查找以十六进制数 xxxx 规定的 Unicode 字符。 |

量词

|  |  |
| --- | --- |
| 量词 | 描述 |
| n+ | 匹配任何包含至少一个 n 的字符串。 |
| n\* | 匹配任何包含零个或多个 n 的字符串。 |
| n? | 匹配任何包含零个或一个 n 的字符串。 |
| n{X} | 匹配包含 X 个 n 的序列的字符串。 |
| n{X,Y} | 匹配包含 X 至 Y 个 n 的序列的字符串。 |
| n{X,} | 匹配包含至少 X 个 n 的序列的字符串。 |
| n$ | 匹配任何结尾为 n 的字符串。 |
| ^n | 匹配任何开头为 n 的字符串。 |
| ?=n | 匹配任何其后紧接指定字符串 n 的字符串。 |
| ?!n | 匹配任何其后没有紧接指定字符串 n 的字符串。 |
|  |  |

RegExp 对象属性

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 描述 | FF | IE |
| global | RegExp 对象是否具有标志 g。 | 1 | 4 |
| ignoreCase | RegExp 对象是否具有标志 i。 | 1 | 4 |
| lastIndex | 一个整数，标示开始下一次匹配的字符位置。 | 1 | 4 |
| multiline | RegExp 对象是否具有标志 m。 | 1 | 4 |
| source | 正则表达式的源文本。 | 1 | 4 |

RegExp 对象方法

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 描述 | FF | IE |
| compile | 编译正则表达式。 | 1 | 4 |
| exec | 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 | 检索字符串中指定的值。返回 true 或 false。 | 1 | 4 |

支持正则表达式的 String 对象的方法

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 描述 | FF | IE |
| search | 检索与正则表达式相匹配的值。 | 1 | 4 |
| match | 找到一个或多个正则表达式的匹配。 | 1 | 4 |
| replace | 替换与正则表达式匹配的子串。 | 1 | 4 |
| split | 把字符串分割为字符串数组。 | 1 | 4 |

# 第12章 JavaScript String 参考手册

String 对象

String 对象用于处理文本（字符串）。

创建 String 对象的语法：

new String(s);

String(s);

参数

参数 s 是要存储在 String 对象中或转换成原始字符串的值。

返回值

当 String() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 s 或 s 的字符串表示。

当不用 new 运算符调用 String() 时，它只把 s 转换成原始的字符串，并返回转换后的值。

String 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 对创建该对象的函数的引用 |
| length | 字符串的长度 |
| prototype | 允许您向对象添加属性和方法 |

String 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| anchor() | 创建 HTML 锚。 |
| big() | 用大号字体显示字符串。 |
| blink() | 显示闪动字符串。 |
| bold() | 使用粗体显示字符串。 |
| charAt() | 返回在指定位置的字符。 |
| charCodeAt() | 返回在指定的位置的字符的 Unicode 编码。 |
| concat() | 连接字符串。 |
| fixed() | 以打字机文本显示字符串。 |
| fontcolor() | 使用指定的颜色来显示字符串。 |
| fontsize() | 使用指定的尺寸来显示字符串。 |
| fromCharCode() | 从字符编码创建一个字符串。 |
| indexOf() | 检索字符串。 |
| italics() | 使用斜体显示字符串。 |
| lastIndexOf() | 从后向前搜索字符串。 |
| link() | 将字符串显示为链接。 |
| localeCompare() | 用本地特定的顺序来比较两个字符串。 |
| match() | 找到一个或多个正则表达式的匹配。 |
| replace() | 替换与正则表达式匹配的子串。 |
| search() | 检索与正则表达式相匹配的值。 |
| 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。 |
| small() | 使用小字号来显示字符串。 |
| split() | 把字符串分割为字符串数组。 |
| strike() | 使用删除线来显示字符串。 |
| sub() | 把字符串显示为下标。 |
| substr() | 从起始索引号提取字符串中指定数目的字符。 |
| substring() | 提取字符串中两个指定的索引号之间的字符。 |
| sup() | 把字符串显示为上标。 |
| toLocaleLowerCase() | 把字符串转换为小写。 |
| toLocaleUpperCase() | 把字符串转换为大写。 |
| toLowerCase() | 把字符串转换为小写。 |
| toUpperCase() | 把字符串转换为大写。 |
| toSource() | 代表对象的源代码。 |
| toString() | 返回字符串。 |
| valueOf() | 返回某个字符串对象的原始值。 |

String 对象描述

字符串是 JavaScript 的一种基本的数据类型。

String 对象的 length 属性声明了该字符串中的字符数。

String 类定义了大量操作字符串的方法，例如从字符串中提取字符或子串，或者检索字符或子串。

需要注意的是，JavaScript 的字符串是不可变的（immutable），String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase() 这样的方法，返回的是全新的字符串，而不是修改原始字符串。

在较早的 Netscape 代码基的 JavaScript 实现中（例如 Firefox 实现中），字符串的行为就像只读的字符数组。例如，从字符串 s 中提取第三个字符，可以用 s[2] 代替更加标准的 s.charAt(2)。此外，对字符串应用 for/in 循环时，它将枚举字符串中每个字符的数组下标（但要注意，ECMAScript 标准规定，不能枚举 length 属性）。因为字符串的数组行为不标准，所以应该避免使用它。

# 第13章 JavaScript 运算符参考手册

JavaScript 运算符用于赋值、比较值、执行算术运算等。

JavaScript 算术运算符

算术运算符用于在变量和/或值之间执行算术。

给定 y = 5，下表解释了算术运算符：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运算符 | 描述 | 例子 | y 中的结果 | x 中的结果 |
| + | 加 | x = y + 2 | y = 5 | x = 7 |
| - | 减 | x = y - 2 | y = 5 | x = 3 |
| \* | 乘 | x = y \* 2 | y = 5 | x = 10 |
| / | 除 | x = y / 2 | y = 5 | x = 2.5 |
| % | 模数（除余） | x = y % 2 | y = 5 | x = 1 |
| ++ | 累加 | x = ++y | y = 6 | x = 6 |
| x = y++ | y = 6 | x = 5 |
| -- | 递减 | x = --y | y = 4 | x = 4 |
| x = y-- | y = 4 | x = 5 |

JavaScript 赋值运算符

赋值运算符用于为 JavaScript 变量赋值。

给定 x = 10 且 y = 5，下表解释了赋值运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 例子 | 等同于 | x 中的结果 |
| = | x = y | x = y | x = 5 |
| += | x += y | x = x + y | x = 15 |
| -= | x -= y | x = x - y | x = 5 |
| \*= | x \*= y | x = x \* y | x = 50 |
| /= | x /= y | x = x / y | x = 2 |
| %= | x %= y | x = x % y | x = 0 |

JavaScript 字符串运算符

+ 运算符和 += 运算符还可用于连接（添加）字符串。

给定 text1 = "Good "、text2 = "Morning"，以及 text3 = ""，下表解释了该运算符：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运算符 | 例子 | text1 | text2 | text3 |
| + | text3 = text1 + text2 | "Good " | "Morning" | "Good Morning" |
| += | text1 += text2 | "Good Morning" | "Morning" | "" |

比较运算符

在逻辑语句中使用比较运算符来确定变量或值之间的相等性或差异。

给定 x = 5，下表解释了比较运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 描述 | 比较 | 返回 |
| == | 等于 | x == 8 | false |
| x == 5 | true |
| === | 相等值和相等类型 | x === "5" | false |
| x === 5 | true |
| != | 不相等 | x != 8 | true |
| !== | 不相等的值或不相等的类型 | x !== "5" | true |
| x !== 5 | false |
| > | 大于 | x > 8 | false |
| < | 小于 | x < 8 | true |
| >= | 大于或等于 | x >= 8 | false |
| <= | 小于或等于 | x <= 8 | true |

条件（三元）运算符

条件运算符根据条件将值赋给变量。

语法

variablename = (condition) ? value1:value2

实例

<!DOCTYPE html>

<html>

<body>

<p>输入您的年龄并点击按钮：</p>

<input id="age" value="18">

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var age, voteable;

age = document.getElementById("age").value;

voteable = (age < 18) ? "Too young":"Old enough";

document.getElementById("demo").innerHTML = voteable + " to vote.";

}

</script>

</body>

</html>

例子解释：

如果变量 "age" 的值小于 18，则变量 "voteable" 的值将为 "Too young"，否则，"voteable" 的值将为 "Old enough"。

逻辑运算符

逻辑运算符用于确定变量或值之间的逻辑。

给定 x = 6 且 y = 3，下表解释了逻辑运算符：

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

JavaScript 位运算符

位运算符可处理 32 位数字。该运算中的任何数字操作数都将转换为 32 位数字。结果将转换回 JavaScript 数字。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算符 | 描述 | 例子 | 等同于 | 结果 | 十进制 |
| & | AND | x = 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | x = 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | x = ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | x = 5 ^ 1 | 0101 ^ | 0100 | 4 |
| << | Left shift | x = 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | Right shift | x = 5 >> 1 | 0101 >> 1 | 0010 | 2 |

上面的例子使用了 4 位无符号的示例。但是 JavaScript 使用 32 位带符号的数字。

因此，在 JavaScript 中，~ 5 不会返回 10，而是返回 -6。

~00000000000000000000000000000101 将返回 11111111111111111111111111111010

typeof 运算符

typeof 运算符返回变量、对象、函数或表达式的类型：

实例

typeof "Bill" // 返回 string

typeof 3.14 // 返回 number

typeof NaN // 返回 number

typeof false // 返回 boolean

typeof [1, 2, 3, 4] // 返回 object

typeof {name:'Bill', age:19} // 返回 object

typeof new Date() // 返回 object

typeof function () {} // 返回 function

typeof myCar // 返回 undefined（如果未声明 myCar）

typeof null // 返回 object

请注意：

NaN 的数据类型是 number

数组的数据类型是 object

日期的数据类型是 object

null 的数据类型是 object

未定义变量的数据类型是 undefined

您不能使用 typeof 定义 JavaScript 对象是否为数组（或日期）。

您无法使用 typeof 来定义 JavaScript 对象是否为数组（或日期）。

delete 运算符

delete 运算符从对象中删除属性：

实例

<!DOCTYPE html>

<html>

<body>

<p>delete 运算符从对象中删除属性。</p>

<p id="demo"></p>

<script>

var person = {

firstname:"Bill",

lastname:"Gates",

age:19,

eyecolor:"blue"

};

delete person.age;

document.getElementById("demo").innerHTML =

person.firstname + " is " + person.age + " years old.";

</script>

</body>

</html>

delete 运算符会同时删除属性的值和属性本身。

删除后，再次将其重新添加前，无法使用该属性。

delete 运算符旨在用于对象属性。它对变量或函数没有影响。

注释：不应在预定义的 JavaScript 对象属性上使用 delete 运算符。这么做可能会使您的应用程序崩溃。

in 运算符

如果指定的属性在指定的对象中，则 in 运算符将返回 true，否则返回 false：

实例

// 数组

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

"Saab" in cars // 返回 false（指定索引号而不是值）

0 in cars // 返回 true

1 in cars // 返回 true

4 in cars // 返回 false（不存在）

"length" in cars // 返回 true（length 是数组属性）

// 对象

var person = {firstName:"Bill", lastName:"Gates", age:19};

"firstName" in person // 返回 true

"age" in person // 返回 true

// 预定义对象

"PI" in Math // 返回 true

"NaN" in Number // 返回 true

"length" in String // 返回 true

instanceof 运算符

如果指定对象是指定对象的实例，则 instanceof 运算符返回 true：

实例

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

cars instanceof Array; // 返回 true

cars instanceof Object; // 返回 true

cars instanceof String; // 返回 false

cars instanceof Number; // 返回 false

void 运算符

void 运算符计算一个表达式并返回 undefined。该运算符通常用于使用 "void(0)" 来获取未定义的原始值（在计算表达式而不使用返回值时很有用）。

实例

<a href="javascript:void(0);">

无用的链接

</a>

<a href="javascript:void(document.body.style.backgroundColor='red');">

单击我将 body 的背景色更改为红色

</a>

# 第14章 JavaScript 语句参考手册

JavaScript 语句

在 HTML 中，JavaScript 语句是 Web 浏览器要“执行”的“指令”。

这条语句告诉浏览器在 id="demo" 的 HTML 元素中写 "Hello Kitty"：

实例

<!DOCTYPE html>

<html>

<body>

<h1>我的网页</h1>

<p id="demo">我的第一个段落。</p>

<script>

document.getElementById("demo").innerHTML = "Hello Kitty.";

</script>

</body>

</html>

JavaScript 语句标识符

JavaScript 语句通常以语句标识符开头，用以标识要执行的 JavaScript 动作。

语句标识符是保留字，不能用作变量名（或其他任何东西）。

下表列出了所有 JavaScript 语句标识符：

|  |  |
| --- | --- |
| 语句 | 描述 |
| break | 退出 switch 或循环。 |
| const | 声明具有恒定值的变量。 |
| class | 声明类。 |
| continue | 如果出现指定条件，则（在循环中）中断一次循环，并继续循环中的下一次迭代。 |
| debugger | 停止执行 JavaScript，并调用调试功能（如果有）。 |
| do ... while | 执行一个语句块，并在条件为 true 时重复该语句块。 |
| for | 循环遍历代码块多次. |
| for ... in | 循环遍历对象的属性。 |
| for ... of | 循环遍历可迭代对象的值。 |
| function | 声明函数。 |
| if ... else ... else if | 标记根据条件执行的语句块。 |
| let | 声明在方括号 {} 作用域内的变量。 |
| return | 停止执行函数并从该函数返回值。 |
| switch | 根据不同情况标记要执行的语句块。 |
| throw | 抛出（生成）错误。 |
| try ... catch ... finally | 标记在 try 块中发生错误时要执行的语句块，并实现错误处理。 |
| var | 声明变量。 |
| while | 标记条件为 true 时要执行的语句块。 |

# 第15章 HTML DOM Attr 对象

HTML DOM 节点

在 HTML DOM （文档对象模型）中，每个部分都是节点：

文档本身是文档节点

所有 HTML 元素是元素节点

所有 HTML 属性是属性节点

HTML 元素内的文本是文本节点

注释是注释节点

Attr 对象

在 HTML DOM 中，Attr 对象表示 HTML 属性。

HTML 属性始终属于 HTML 元素。

NamedNodeMap 对象

在 HTML DOM 中，NamedNodeMap 对象表示元素属性节点的无序集合。

NamedNodeMap 中的节点可通过名称或索引（数字）来访问。

属性和方法

|  |  |
| --- | --- |
| 属性 / 方法 | 描述 |
| attr.isId | 如果属性是 id 类型，则返回 true，否则返回 false。 |
| attr.name | 返回属性的名称。 |
| attr.value | 设置或返回属性的值。 |
| attr.specified | 如果已指定属性，则返回 true，否则返回 false。 |
| nodemap.getNamedItem() | 从 NamedNodeMap 返回指定的属性节点。 |
| nodemap.item() | 返回 NamedNodeMap 中位于指定下标的节点。 |
| nodemap.length | 返回 NamedNodeMap 中的节点数。 |
| nodemap.removeNamedItem() | 移除指定的属性节点。 |
| nodemap.setNamedItem() | 设置指定的属性节点（通过名称）。 |

DOM 4 警告！

在 W3C DOM Core 中，Attr (attribute) 对象从 Node 对象继承所有属性和方法。

在 DOM 4 中，Attr 对象不再从 Node 继承。

为了保证未来的代码安全，您应该避免在属性对象上使用节点对象的属性和方法：

|  |  |
| --- | --- |
| 属性 / 方法 | 避免的理由 |
| attr.appendChild() | 属性没有子节点。 |
| attr.attributes | 属性没有属性。 |
| attr.baseURI | 使用 document.baseURI 代替。 |
| attr.childNodes | 属性没有子节点。 |
| attr.cloneNode() | 使用 attr.value 代替。 |
| attr.firstChild | 属性没有子节点。 |
| attr.hasAttributes() | 属性没有属性。 |
| attr.hasChildNodes | 属性没有子节点。 |
| attr.insertBefore() | 属性没有子节点。 |
| attr.isEqualNode() | 没有意义。 |
| attr.isSameNode() | 没有意义。 |
| attr.isSupported() | 始终为 true。 |
| attr.lastChild | 属性没有子节点。 |
| attr.nextSibling | 属性没有同级节点。 |
| attr.nodeName | 使用 attr.name 代替。 |
| attr.nodeType | 始终为 2 (ATTRIBUTE\_NODE)。 |
| attr.nodeValue | 使用 attr.value 代替。 |
| attr.normalize() | 属性无法被正常化。 |
| attr.ownerDocument | 始终是您的 HTML 文档。 |
| attr.ownerElement | 这是您用来访问该属性的 HTML 元素。 |
| attr.parentNode | 这是您用来访问该属性的 HTML 元素。 |
| attr.previousSibling | 属性没有同级节点。 |
| attr.removeChild | 属性没有子节点。 |
| attr.replaceChild | 属性没有子节点。 |
| attr.textContent | 使用 attr.value 代替。 |

# 第16章 HTML DOM Document 对象

Document 对象

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。

提示：Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问。

Document 对象集合

|  |  |
| --- | --- |
| 集合 | 描述 |
| all[] | 提供对文档中所有 HTML 元素的访问。 |
| anchors[] | 返回对文档中所有 Anchor 对象的引用。 |
| applets | 返回对文档中所有 Applet 对象的引用。 |
| 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。 |
| links[] | 返回对文档中所有 Area 和 Link 对象引用。 |

Document 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| body | 提供对 <body> 元素的直接访问。  对于定义了框架集的文档，该属性引用最外层的 <frameset>。 |
| cookie | 设置或返回与当前文档有关的所有 cookie。 |
| domain | 返回当前文档的域名。 |
| lastModified | 返回文档被最后修改的日期和时间。 |
| referrer | 返回载入当前文档的文档的 URL。 |
| title | 返回当前文档的标题。 |
| URL | 返回当前文档的 URL。 |

Document 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| close() | 关闭用 document.open() 方法打开的输出流，并显示选定的数据。 |
| getElementById() | 返回对拥有指定 id 的第一个对象的引用。 |
| getElementsByName() | 返回带有指定名称的对象集合。 |
| getElementsByTagName() | 返回带有指定标签名的对象集合。 |
| open() | 打开一个流，以收集来自任何 document.write() 或 document.writeln() 方法的输出。 |
| write() | 向文档写 HTML 表达式 或 JavaScript 代码。 |
| writeln() | 等同于 write() 方法，不同的是在每个表达式之后写一个换行符。 |

Document 对象描述

HTMLDocument 接口对 DOM Document 接口进行了扩展，定义 HTML 专用的属性和方法。

很多属性和方法都是 HTMLCollection 对象（实际上是可以用数组或名称索引的只读数组），其中保存了对锚、表单、链接以及其他可脚本元素的引用。

这些集合属性都源自于 0 级 DOM。它们已经被 Document.getElementsByTagName() 所取代，但是仍然常常使用，因为他们很方便。

write() 方法值得注意，在文档载入和解析的时候，它允许一个脚本向文档中插入动态生成的内容。

注意，在 1 级 DOM 中，HTMLDocument 定义了一个名为 getElementById() 的非常有用的方法。在 2 级 DOM 中，该方法已经被转移到了 Document 接口，它现在由 HTMLDocument 继承而不是由它定义了。

# 第17章 HTML DOM Element 对象

HTML DOM 节点

在 HTML DOM （文档对象模型）中，每个部分都是节点：

文档本身是文档节点

所有 HTML 元素是元素节点

所有 HTML 属性是属性节点

HTML 元素内的文本是文本节点

注释是注释节点

Element 对象

在 HTML DOM 中，Element 对象表示 HTML 元素。

Element 对象可以拥有类型为元素节点、文本节点、注释节点的子节点。

NodeList 对象表示节点列表，比如 HTML 元素的子节点集合。

元素也可以拥有属性。属性是属性节点（参见下一节）。

属性和方法

下面的属性和方法可用于所有 HTML 元素上：

|  |  |
| --- | --- |
| 属性 / 方法 | 描述 |
| element.accessKey | 设置或返回元素的快捷键。 |
| addEventListener() | 将事件处理程序附加到元素。 |
| element.appendChild() | 向元素添加新的子节点，作为最后一个子节点。 |
| element.attributes | 返回元素属性的 NamedNodeMap。 |
| element.childNodes | 返回元素子节点的 NodeList。 |
| element.className | 设置或返回元素的 class 属性。 |
| element.clientHeight | 返回元素的可见高度。 |
| element.clientWidth | 返回元素的可见宽度。 |
| element.cloneNode() | 克隆元素。 |
| element.compareDocumentPosition() | 比较两个元素的文档位置。 |
| element.contentEditable | 设置或返回元素的内容是否可编辑。 |
| element.dir | 设置或返回元素的文本方向。 |
| element.firstChild | 返回元素的首个子。 |
| element.getAttribute() | 返回元素节点的指定属性值。 |
| element.getAttributeNode() | 返回指定的属性节点。 |
| element.getElementsByTagName() | 返回拥有指定标签名的所有子元素的集合。 |
| element.getFeature() | 返回实现了指定特性的 API 的某个对象。 |
| element.getUserData() | 返回关联元素上键的对象。 |
| element.hasAttribute() | 如果元素拥有指定属性，则返回true，否则返回 false。 |
| element.hasAttributes() | 如果元素拥有属性，则返回 true，否则返回 false。 |
| element.hasChildNodes() | 如果元素拥有子节点，则返回 true，否则 false。 |
| element.id | 设置或返回元素的 id。 |
| element.innerHTML | 设置或返回元素的内容。 |
| element.insertBefore() | 在指定的已有的子节点之前插入新节点。 |
| element.isContentEditable | 设置或返回元素的内容。 |
| element.isDefaultNamespace() | 如果指定的 namespaceURI 是默认的，则返回 true，否则返回 false。 |
| element.isEqualNode() | 检查两个元素是否相等。 |
| element.isSameNode() | 检查两个元素是否是相同的节点。 |
| element.isSupported() | 如果元素支持指定特性，则返回 true。 |
| element.lang | 设置或返回元素的语言代码。 |
| element.lastChild | 返回元素的最后一个子元素。 |
| element.namespaceURI | 返回元素的 namespace URI。 |
| element.nextSibling | 返回位于相同节点树层级的下一个节点。 |
| element.nodeName | 返回元素的名称。 |
| element.nodeType | 返回元素的节点类型。 |
| element.nodeValue | 设置或返回元素值。 |
| element.normalize() | 合并元素中相邻的文本节点，并移除空的文本节点。 |
| element.offsetHeight | 返回元素的高度。 |
| element.offsetWidth | 返回元素的宽度。 |
| element.offsetLeft | 返回元素的水平偏移位置。 |
| element.offsetParent | 返回元素的偏移容器。 |
| element.offsetTop | 返回元素的垂直偏移位置。 |
| element.ownerDocument | 返回元素的根元素（文档对象）。 |
| element.parentNode | 返回元素的父节点。 |
| element.previousSibling | 返回位于相同节点树层级的前一个元素。 |
| element.removeAttribute() | 从元素中移除指定属性。 |
| element.removeAttributeNode() | 移除指定的属性节点，并返回被移除的节点。 |
| element.removeChild() | 从元素中移除子节点。 |
| element.replaceChild() | 替换元素中的子节点。 |
| element.scrollHeight | 返回元素的整体高度。 |
| element.scrollLeft | 返回元素左边缘与视图之间的距离。 |
| element.scrollTop | 返回元素上边缘与视图之间的距离。 |
| element.scrollWidth | 返回元素的整体宽度。 |
| element.setAttribute() | 把指定属性设置或更改为指定值。 |
| element.setAttributeNode() | 设置或更改指定属性节点。 |
| element.setIdAttribute() |  |
| element.setIdAttributeNode() |  |
| element.setUserData() | 把对象关联到元素上的键。 |
| element.style | 设置或返回元素的 style 属性。 |
| element.tabIndex | 设置或返回元素的 tab 键控制次序。 |
| element.tagName | 返回元素的标签名。 |
| element.textContent | 设置或返回节点及其后代的文本内容。 |
| element.title | 设置或返回元素的 title 属性。 |
| element.toString() | 把元素转换为字符串。 |
| nodelist.item() | 返回 NodeList 中位于指定下标的节点。 |
| nodelist.length | 返回 NodeList 中的节点数。 |

# 第18章 HTML DOM 事件

HTML DOM 事件

HTML DOM 事件允许 JavaScript 在 HTML 文档中的元素上注册不同的事件处理程序。

事件通常与函数结合使用，在事件发生之前函数不会被执行（例如当用户单击按钮时）。

如需有关事件的教程，请学习我们的 JavaScript 事件教程。

|  |  |  |
| --- | --- | --- |
| 事件 | 描述 | 属于 |
| abort | 媒体加载中止时发生该事件。 | UiEvent  Event |
| afterprint | 当页面开始打印时，或者关闭打印对话框时，发生此事件。 | Event |
| animationend | CSS 动画完成时发生此事件。 | AnimationEvent |
| animationiteration | 重复 CSS 动画时发生此事件。 | AnimationEvent |
| animationstart | CSS 动画开始时发生此事件。 | AnimationEvent |
| beforeprint | 即将打印页面时发生此事件。 | Event |
| beforeunload | 在文档即将被卸载之前发生此事件。 | UiEvent  Event |
| blur | 当元素失去焦点时发生此事件。 | FocusEvent |
| canplay | 当浏览器可以开始播放媒体时，发生此事件。 | Event |
| canplaythrough | 当浏览器可以在不停止缓冲的情况下播放媒体时发生此事件。 | Event |
| change | 当form元素的内容、选择的内容或选中的状态发生改变时，发生此事件 | Event |
| click | 当用户单击元素时发生此事件。 | MouseEvent |
| contextmenu | 当用户右键单击某个元素以打开上下文菜单时，发生此事件。 | MouseEvent |
| copy | 当用户复制元素的内容时发生此事件。 | ClipboardEvent |
| cut | 当用户剪切元素的内容时发生此事件。 | ClipboardEvent |
| dblclick | 当用户双击元素时发生此事件。 | MouseEvent |
| drag | 拖动元素时发生此事件。 | DragEvent |
| dragend | 当用户完成拖动元素后，发生此事件。 | DragEvent |
| dragenter | 当拖动的元素进入放置目标时，发生此事件。 | DragEvent |
| dragleave | 当拖动的元素离开放置目标时，发生此事件。 | DragEvent |
| dragover | 当拖动的元素位于放置目标之上时，发生此事件。 | DragEvent |
| dragstart | 当用户开始拖动元素时发生此事件。 | DragEvent |
| drop | 当将拖动的元素放置在放置目标上时，发生此事件。 | DragEvent |
| durationchange | 媒体时长改变时发生此事件。 | Event |
| ended | 在媒体播放到尽头时发生此事件。 | Event |
| error | 当加载外部文件时发生错误后，发生此事件。 | ProgressEvent  UiEvent  Event |
| focus | 在元素获得焦点时发生此事件。 | FocusEvent |
| focusin | 在元素即将获得焦点时发生此事件。 | FocusEvent |
| focusout | 在元素即将失去焦点时发生此事件。 | FocusEvent |
| fullscreenchange | 当元素以全屏模式显示时，发生此事件。 | Event |
| fullscreenerror | 当元素无法在全屏模式下显示时，发生此事件。 | Event |
| hashchange | 当 URL 的锚部分发生改变时，发生此事件。 | HashChangeEvent |
| input | 当元素获得用户输入时，发生此事件。 | InputEvent  Event |
| invalid | 当元素无效时，发生此事件。 | Event |
| keydown | 当用户正在按下键时，发生此事件。 | KeyboardEvent |
| keypress | 当用户按下键时，发生此事件。 | KeyboardEvent |
| keyup | 当用户松开键时，发生此事件。 | KeyboardEvent |
| load | 在对象已加载时，发生此事件。 | UiEvent  Event |
| loadeddata | 媒体数据加载后，发生此事件。 | Event |
| loadedmetadata | 加载元数据（比如尺寸和持续时间）时，发生此事件。 | Event |
| loadstart | 当浏览器开始查找指定的媒体时，发生此事件。 | ProgressEvent |
| message | 在通过此事件源接收消息时，发生此事件。 | Event |
| mousedown | 当用户在元素上按下鼠标按钮时，发生此事件。 | MouseEvent |
| mouseenter | 当指针移动到元素上时，发生此事件。 | MouseEvent |
| mouseleave | 当指针从元素上移出时，发生此事件。 | MouseEvent |
| mousemove | 当指针在元素上方移动时，发生此事件。 | MouseEvent |
| mouseout | 当用户将鼠标指针移出元素或其中的子元素时，发生此事件。 | MouseEvent |
| mouseover | 当指针移动到元素或其中的子元素上时，发生此事件。 | MouseEvent |
| mouseup | 当用户在元素上释放鼠标按钮时，发生此事件。 | MouseEvent |
| mousewheel | 不推荐使用。请改用 wheel 事件。 | WheelEvent |
| offline | 当浏览器开始脱机工作时，发生此事件。 | Event |
| online | 当浏览器开始在线工作时，发生此事件。 | Event |
| open | 当打开与事件源的连接时，发生此事件。 | Event |
| pagehide | 当用户离开某张网页进行导航时，发生此事件。 | PageTransitionEvent |
| pageshow | 在用户导航到某张网页时，发生此事件。 | PageTransitionEvent |
| paste | 当用户将一些内容粘贴到元素中时，发生此事件。 | ClipboardEvent |
| pause | 当媒体被用户暂停或以编程方式暂停时，发生此事件。 | Event |
| play | 当媒体已启动或不再暂停时，发生此事件。 | Event |
| playing | 在媒体被暂停或停止以缓冲后播放时，发生此事件。 | Event |
| popstate | 窗口的历史记录改变时，发生此事件。 | PopStateEvent |
| progress | 当浏览器正处于获得媒体数据的过程中时，发生此事件。 | Event |
| ratechange | 媒体播放速度改变时发生此事件。 | Event |
| reset | 重置表单时发生此事件。 | Event |
| resize | 调整文档视图的大小时发生此事件。 | UiEvent  Event |
| scroll | 滚动元素的滚动条时发生此事件。 | UiEvent  Event |
| search | 当用户在搜索字段中输入内容时，发生此事件。 | Event |
| seeked | 当用户完成移动/跳到媒体中的新位置时，发生该事件。 | Event |
| seeking | 当用户开始移动/跳到媒体中的新位置时，发生该事件。 | Event |
| select | 用户选择文本后（对于<input>和<textarea>）发生此事件 | UiEvent  Event |
| show | 当 <menu> 元素显示为上下文菜单时，发生此事件。 | Event |
| stalled | 当浏览器尝试获取媒体数据但数据不可用时，发生此事件。 | Event |
| storage | Web 存储区域更新时发生此事件。 | StorageEvent |
| submit | 在提交表单时发生此事件。 | Event |
| suspend | 当浏览器有意不获取媒体数据时，发生此事件。 | Event |
| timeupdate | 当播放位置更改时发生此事件。 | Event |
| toggle | 当用户打开或关闭 <details> 元素时，发生此事件。 | Event |
| touchcancel | 在触摸被中断时，发生此事件。 | TouchEvent |
| touchend | 当手指从触摸屏上移开时，发生此事件。 | TouchEvent |
| touchmove | 当手指在屏幕上拖动时，发生此事件。 | TouchEvent |
| touchstart | 当手指放在触摸屏上时，发生此事件。 | TouchEvent |
| transitionend | CSS 转换完成时，发生此事件。 | TransitionEvent |
| unload | 页面卸载后（对于 <body>），发生此事件。 | UiEvent  Event |
| volumechange | 当媒体的音量已更改时，发生此事件。 | Event |
| waiting | 当媒体已暂停但预期会恢复时，发生此事件。 | Event |
| wheel | 当鼠标滚轮在元素向上或向下滚动时，发生此事件。 | WheelEvent |
|  |  |  |

HTML DOM 事件属性和方法

|  |  |  |
| --- | --- | --- |
| 属性/方法 | 描述 | 属于 |
| altKey | 返回触发鼠标事件时是否按下了 "ALT" 键。 | MouseEvent |
| altKey | 返回触发按键事件时是否按下了 "ALT" 键。 | KeyboardEvent  TouchEvent |
| animationName | 返回动画的名称。 | AnimationEvent |
| bubbles | 返回特定事件是否为冒泡事件。 | Event |
| button | 返回触发鼠标事件时按下的鼠标按钮。 | MouseEvent |
| buttons | 返回触发鼠标事件时按下的鼠标按钮。 | MouseEvent |
| cancelBubble | 设置或返回事件是否应该向上层级进行传播。 |  |
| cancelable | 返回事件是否可以阻止其默认操作。 | Event |
| changeTouches | 返回在上一触摸与该触摸之间其状态已更改的所有触摸对象的列表 | TouchEvent |
| charCode | 返回触发 onkeypress 事件的键的 Unicode 字符代码。 | KeyboardEvent |
| clientX | 返回触发鼠标事件时，鼠标指针相对于当前窗口的水平坐标。 | MouseEvent  TouchEvent |
| clientY | 返回触发鼠标事件时，鼠标指针相对于当前窗口的垂直坐标。 | MouseEvent  TouchEvent |
| clipboardData | 返回对象，其中包含受剪贴板操作影响的数据。 | ClipboardData |
| code | 返回触发事件的键的代码。 | KeyboardEvent |
| composed | 指示该事件是否可以从 Shadow DOM 传递到一般的 DOM。 | Event |
| composedPath() | 返回事件的路径。 |  |
| createEvent() | 创建新事件。 | Event |
| ctrlKey | 返回触发鼠标事件时是否按下了 "CTRL" 键。 | MouseEvent |
| ctrlKey | 返回按键鼠标事件时是否按下了 "CTRL" 键。 | KeyboardEvent  TouchEvent |
| currentTarget | 返回其事件侦听器触发事件的元素。 | Event |
| data | 返回插入的字符。 | InputEvent |
| dataTransfer | 返回一个对象，其中包含被拖放或插入/删除的数据。 | DragEvent  InputEvent |
| defaultPrevented | 返回是否为事件调用 preventDefault() 方法。 | Event |
| deltaX | 返回鼠标滚轮的水平滚动量（x 轴）。 | WheelEvent |
| deltaY | 返回鼠标滚轮的垂直滚动量（y 轴）。 | WheelEvent |
| deltaZ | 返回鼠标滚轮的 Z 轴滚动量。 | WheelEvent |
| deltaMode | 返回数字，代表增量值（像素、线或页面）的度量单位。 | WheelEvent |
| detail | 返回数字，指示鼠标被单击了多少次。 | UiEvent |
| elapsedTime | 返回动画已运行的秒数。 | AnimationEvent |
| elapsedTime | 返回过渡已运行的秒数。 |  |
| eventPhase | 返回当前正在评估事件流处于哪个阶段。 | Event |
| getModifierState() | 返回包含目标范围的数组，此范围将受到插入/删除的影响。 | MouseEvent |
| getTargetRanges() | 返回包含目标范围的数组，此范围将受到插入/删除的影响。 | InputEvent |
| inputType | 返回更改的类型（即 "inserting" 或 "deleting"）。 | InputEvent |
| isComposing | 返回事件的状态是否正在构成。 | InputEvent  KeyboardEvent |
| isTrusted | 返回事件是否受信任。 | Event |
| key | 返回事件表示的键的键值。 | KeyboardEvent |
| key | 返回更改后的存储项的键。 | StorageEvent |
| keyCode | 返回触发 onkeypress、onkeydown 或 onkeyup 事件的键的 Unicode 字符代码。 | KeyboardEvent |
| location | 返回键盘或设备上按键的位置。 | KeyboardEvent |
| lengthComputable | 返回进度的长度是否可计算。 | ProgressEvent |
| loaded | 返回已加载的工作量。 | ProgressEvent |
| metaKey | 返回事件触发时是否按下了 "META" 键。 | MouseEvent |
| metaKey | 返回按键事件触发时是否按下了 "META" 键。 | KeyboardEvent  TouchEvent |
| MovementX | 返回相对于上一 mousemove 事件的位置的鼠标指针的水平坐标 | MouseEvent |
| MovementY | 返回相对于上一 mousemove 事件的位置的鼠标指针的垂直坐标 | MouseEvent |
| newURL | 返回更改 hash 后的文档 URL。 | HasChangeEvent |
| newValue | 返回更改后的存储项目的新值。 | StorageEvent |
| offsetX | 返回鼠标指针相对于目标元素边缘位置的水平坐标。 | MouseEvent |
| offsetY | 返回鼠标指针相对于目标元素边缘位置的垂直坐标。 | MouseEvent |
| oldURL | 返回更改 hash 前的文档 URL。 | HasChangeEvent |
| oldValue | 返回更改后的存储项目的旧值。 | StorageEvent |
| onemptied | 当发生不良情况且媒体文件突然不可用时，发生此事件。 |  |
| pageX | 返回触发鼠标事件时鼠标指针相对于文档的水平坐标。 | MouseEvent |
| pageY | 返回触发鼠标事件时鼠标指针相对于文档的垂直坐标。 | MouseEvent |
| persisted | 返回网页是否被浏览器缓存。 | PageTransitionEvent |
| preventDefault() | 如果可以取消事件，则将其取消，不执行属于该事件的默认操作。 | Event |
| propertyName | 返回与动画或过渡相关联的 CSS 属性的名称。 | AnimationEvent  TransitionEvent |
| pseudoElement | 返回动画或过渡的伪元素的名称。 | AnimationEvent  TransitionEvent |
| region |  | MouseEvent |
| relatedTarget | 返回与触发鼠标事件的元素相关的元素。 | MouseEvent |
| relatedTarget | 返回与触发事件的元素相关的元素。 | FocusEvent |
| repeat | 返回是否重复按住某个键。 | KeyboardEvent |
| screenX | 返回窗口/鼠标指针相对于屏幕的水平坐标。 | MouseEvent |
| screenY | 返回窗口/鼠标指针相对于屏幕的垂直坐标。 | MouseEvent |
| shiftKey | 返回事件触发时是否按下了 "SHIFT" 键。 | MouseEvent |
| shiftKey | 返回按键事件触发时是否按下了 "SHIFT" 键。 | KeyboardEvent  TouchEvent |
| state | 返回包含历史记录条目副本的对象。 | PopStateEvent |
| stopImme...() | 防止同一事件的其他侦听器被调用。 | Event |
| stopPropagation() | 防止事件在事件流中进一步传播。 | Event |
| storageArea | 返回代表受影响的存储对象的对象。 | StorageEvent |
| target | 返回触发事件的元素。 | Event |
| targetTouches | 返回包含仍与触摸面接触的所有触摸点的Touch对象的TouchList列表 | TouchEvent |
| timeStamp | 返回创建事件的时间（相对于纪元的毫秒数）。 | Event |
| total | 返回将要加载的工作总量。 | ProgressEvent |
| touches | 返回当前与表面接触的所有 touch 对象的列表。 | TouchEvent |
| type | 返回事件名称。 | Event |
| url | 返回已更改项目的所在文档的 URL。 | StorageEvent |
| which | 返回触发鼠标事件时按下的鼠标按钮。 | MouseEvent |
| which | 返回触发 onkeypress 事件的键的 Unicode 字符码，或触发 onkeydown 或 onkeyup 事件的键的 Unicode 键码 | KeyboardEvent |
| view | 返回对发生事件的 Window 对象的引用。 | UiEvent |

# 第19章 事件对象

事件对象

当 HTML 中发生事件时，该事件属于某个事件对象，例如，鼠标单击事件属于 MouseEvent 对象。

如需更多有关事件的知识，请阅读我们的 JavaScript 事件教程。

Event 对象

所有事件对象均基于 Event 对象，并继承其所有属性和方法。

|  |  |
| --- | --- |
| Event 对象 | 描述 |
| Event | 所有事件对象的父对象。 |

其他事件对象

这些是最常见的事件对象：

|  |  |
| --- | --- |
| Event 对象 | 描述 |
| AnimationEvent | 针对 CSS 动画 |
| ClipboardEvent | 针对剪贴板的修改 |
| DragEvent | 针对拖放交互 |
| FocusEvent | 针对与焦点有关的事件 |
| HashChangeEvent | 针对 URL 锚点部分的更改 |
| InputEvent | 针对用户输入 |
| KeyboardEvent | 针对键盘交互 |
| MouseEvent | 针对鼠标交互 |
| PageTransitionEvent | 针对导航到网页或离开网页 |
| PopStateEvent | 针对历史记录条目中的更改 |
| ProgressEvent | 针对加载外部资源的进度 |
| StorageEvent | 针对窗口的存储区域中的更改 |
| TouchEvent | 针对触摸交互 |
| TransitionEvent | 针对 CSS 过渡 |
| UiEvent | 针对用户界面交互 |
| WheelEvent | 针对鼠标滚轮交互 |

# 第20章 DOM HTMLCollection

HTMLCollection 对象

HTMLCollection 对象是 HTML 元素的类似数组的列表。

诸如 getElementsByTagName() 之类的方法会返回 HTMLCollection。

属性和方法

可以在 HTMLCollection 对象上使用以下属性和方法

|  |  |
| --- | --- |
| 属性 / 方法 | 描述 |
| item() | 返回 HTMLCollection 中指定索引处的元素。 |
| length | 返回 HTMLCollection 中的元素数。 |
| namedItem() | 返回 HTMLCollection 中有指定 ID 或名称的元素。 |

实例

实例

获取 HTMLCollection：

<!DOCTYPE html>

<html>

<body>

<h1>HTMLCollection 对象</h1>

<p>使用 getElementsByTagName() 方法来返回 HTMLCollection：</p>

<script>

var x = document.getElementsByTagName("P");

document.write(x);

</script>

</body>

</html>

实例

输出文档中 <p> 元素的数量：

<!DOCTYPE html>

<html>

<body>

<h1>HTMLCollection 对象</h1>

<p>使用 getElementsByTagName() 方法来返回 HTMLCollection。</p>

<p>此页面上 P 元素的数量：</p>

<script>

var x = document.getElementsByTagName("P");

document.write(x.length);

</script>

</body>

</html>

实例

循环遍历 HTMLCollection 中的每个元素

<!DOCTYPE html>

<html>

<body>

<h1>HTMLCollection 对象</h1>

<p>循环遍历此页面上的每个 HTML 元素，并输出其标记名称：</p>

<script>

var x, i, l;

x = document.getElementsByTagName("\*");

l = x.length;

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

document.write(x[i].tagName + "<br>");

}

</script>

</body>

</html>

# 第21章 Location 对象

Location 对象

Location 对象包含有关当前 URL 的信息。

Location 对象是 Window 对象的一个部分，可通过 window.location 属性来访问。

例子

<html>

<head>

<script type="text/javascript">

function currLocation()

{

alert(window.location)

}

function newLocation()

{

window.location="/index.html"

}

</script>

</head>

<body>

<input type="button" onclick="currLocation()" value="显示当前的 URL">

<input type="button" onclick="newLocation()" value="改变 URL">

</body>

</html>

Location 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| hash | 设置或返回从井号 (#) 开始的 URL（锚）。 |
| host | 设置或返回主机名和当前 URL 的端口号。 |
| hostname | 设置或返回当前 URL 的主机名。 |
| href | 设置或返回完整的 URL。 |
| pathname | 设置或返回当前 URL 的路径部分。 |
| port | 设置或返回当前 URL 的端口号。 |
| protocol | 设置或返回当前 URL 的协议。 |
| search | 设置或返回从问号 (?) 开始的 URL（查询部分）。 |

Location 对象方法

|  |  |
| --- | --- |
| 属性 | 描述 |
| assign() | 加载新的文档。 |
| reload() | 重新加载当前文档。 |
| replace() | 用新的文档替换当前文档。 |

Location 对象描述

Location 对象存储在 Window 对象的 Location 属性中，表示那个窗口中当前显示的文档的 Web 地址。它的 href 属性存放的是文档的完整 URL，其他属性则分别描述了 URL 的各个部分。这些属性与 Anchor 对象（或 Area 对象）的 URL 属性非常相似。当一个 Location 对象被转换成字符串，href 属性的值被返回。这意味着你可以使用表达式 location 来替代 location.href。

不过 Anchor 对象表示的是文档中的超链接，Location 对象表示的却是浏览器当前显示的文档的 URL（或位置）。但是 Location 对象所能做的远远不止这些，它还能控制浏览器显示的文档的位置。如果把一个含有 URL 的字符串赋予 Location 对象或它的 href 属性，浏览器就会把新的 URL 所指的文档装载进来，并显示出来。

除了设置 location 或 location.href 用完整的 URL 替换当前的 URL 之外，还可以修改部分 URL，只需要给 Location 对象的其他属性赋值即可。这样做就会创建新的 URL，其中的一部分与原来的 URL 不同，浏览器会将它装载并显示出来。例如，假设设置了Location对象的 hash 属性，那么浏览器就会转移到当前文档中的一个指定的位置。同样，如果设置了 search 属性，那么浏览器就会重新装载附加了新的查询字符串的 URL。

除了 URL 属性外，Location 对象的 reload() 方法可以重新装载当前文档，replace() 可以装载一个新文档而无须为它创建一个新的历史记录，也就是说，在浏览器的历史列表中，新文档将替换当前文档。

# 第22章 Navigator 对象

Navigator 对象

Navigator 对象包含有关浏览器的信息。

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

Navigator 对象集合

|  |  |
| --- | --- |
| 集合 | 描述 |
| plugins[] | 返回对文档中所有嵌入式对象的引用。  该集合是一个 Plugin 对象的数组，其中的元素代表浏览器已经安装的插件。Plug-in 对象提供的是有关插件的信息，其中包括它所支持的 MIME 类型的列表。  虽然 plugins[] 数组是由 IE 4 定义的，但是在 IE 4 中它却总是空的，因为 IE 4 不支持插件和 Plugin 对象。 |

Navigator 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| appCodeName | 返回浏览器的代码名。 |
| appMinorVersion | 返回浏览器的次级版本。 |
| appName | 返回浏览器的名称。 |
| appVersion | 返回浏览器的平台和版本信息。 |
| browserLanguage | 返回当前浏览器的语言。 |
| cookieEnabled | 返回指明浏览器中是否启用 cookie 的布尔值。 |
| cpuClass | 返回浏览器系统的 CPU 等级。 |
| onLine | 返回指明系统是否处于脱机模式的布尔值。 |
| platform | 返回运行浏览器的操作系统平台。 |
| systemLanguage | 返回 OS 使用的默认语言。 |
| userAgent | 返回由客户机发送服务器的 user-agent 头部的值。 |
| userLanguage | 返回 OS 的自然语言设置。 |

Navigator 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| javaEnabled() | 规定浏览器是否启用 Java。 |
| taintEnabled() | 规定浏览器是否启用数据污点 (data tainting)。 |

Navigator 对象描述

Navigator 对象包含的属性描述了正在使用的浏览器。可以使用这些属性进行平台专用的配置。

虽然这个对象的名称显而易见的是 Netscape 的 Navigator 浏览器，但其他实现了 JavaScript 的浏览器也支持这个对象。

Navigator 对象的实例是唯一的，可以用 Window 对象的 navigator 属性来引用它。

# 第23章 Screen 对象

Screen 对象

Screen 对象包含有关客户端显示屏幕的信息。

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

Screen 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| availHeight | 返回显示屏幕的高度 (除 Windows 任务栏之外)。 |
| availWidth | 返回显示屏幕的宽度 (除 Windows 任务栏之外)。 |
| bufferDepth | 设置或返回调色板的比特深度。 |
| colorDepth | 返回目标设备或缓冲器上的调色板的比特深度。 |
| deviceXDPI | 返回显示屏幕的每英寸水平点数。 |
| deviceYDPI | 返回显示屏幕的每英寸垂直点数。 |
| fontSmoothingEnabled | 返回用户是否在显示控制面板中启用了字体平滑。 |
| height | 返回显示屏幕的高度。 |
| logicalXDPI | 返回显示屏幕每英寸的水平方向的常规点数。 |
| logicalYDPI | 返回显示屏幕每英寸的垂直方向的常规点数。 |
| pixelDepth | 返回显示屏幕的颜色分辨率（比特每像素）。 |
| updateInterval | 设置或返回屏幕的刷新率。 |
| width | 返回显示器屏幕的宽度。 |

Screen 对象描述

每个 Window 对象的 screen 属性都引用一个 Screen 对象。Screen 对象中存放着有关显示浏览器屏幕的信息。JavaScript 程序将利用这些信息来优化它们的输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。

# 第24章 HTML DOM Style 对象

Style 对象

Style 对象代表一个单独的样式声明。可从应用样式的文档或元素访问 Style 对象。

使用 Style 对象属性的语法：

document.getElementById("id").style.property="值"

Style 对象的属性：

背景

边框和边距

布局

列表

杂项

定位

打印

滚动条

表格

文本

规范

Background 属性

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

Border 和 Margin 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| border | 在一行设置四个边框的所有属性 |
| borderBottom | 在一行设置底边框的所有属性 |
| borderBottomColor | 设置底边框的颜色 |
| borderBottomStyle | 设置底边框的样式 |
| borderBottomWidth | 设置底边框的宽度 |
| borderColor | 设置所有四个边框的颜色 (可设置四种颜色) |
| borderLeft | 在一行设置左边框的所有属性 |
| borderLeftColor | 设置左边框的颜色 |
| borderLeftStyle | 设置左边框的样式 |
| borderLeftWidth | 设置左边框的宽度 |
| borderRight | 在一行设置右边框的所有属性 |
| borderRightColor | 设置右边框的颜色 |
| borderRightStyle | 设置右边框的样式 |
| borderRightWidth | 设置右边框的宽度 |
| borderStyle | 设置所有四个边框的样式 (可设置四种样式) |
| borderTop | 在一行设置顶边框的所有属性 |
| borderTopColor | 设置顶边框的颜色 |
| borderTopStyle | 设置顶边框的样式 |
| borderTopWidth | 设置顶边框的宽度 |
| borderWidth | 设置所有四条边框的宽度 (可设置四种宽度) |
| margin | 设置元素的边距 (可设置四个值) |
| marginBottom | 设置元素的底边距 |
| marginLeft | 设置元素的左边距 |
| marginRight | 设置元素的右边据 |
| marginTop | 设置元素的顶边距 |
| outline | 在一行设置所有的outline属性 |
| outlineColor | 设置围绕元素的轮廓颜色 |
| outlineStyle | 设置围绕元素的轮廓样式 |
| outlineWidth | 设置围绕元素的轮廓宽度 |
| padding | 设置元素的填充 (可设置四个值) |
| paddingBottom | 设置元素的下填充 |
| paddingLeft | 设置元素的左填充 |
| paddingRight | 设置元素的右填充 |
| paddingTop | 设置元素的顶填充 |

Layout 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| clear | 设置在元素的哪边不允许其他的浮动元素 |
| clip | 设置元素的形状 |
| content | 设置元信息 |
| counterIncrement | 设置其后是正数的计数器名称的列表。其中整数指示每当元素出现时计数器的增量。默认是1。 |
| counterReset | 设置其后是正数的计数器名称的列表。其中整数指示每当元素出现时计数器被设置的值。默认是0。 |
| cssFloat | 设置图像或文本将出现（浮动）在另一元素中的何处。 |
| cursor | 设置显示的指针类型 |
| direction | 设置元素的文本方向 |
| display | 设置元素如何被显示 |
| height | 设置元素的高度 |
| markerOffset | 设置marker box的principal box距离其最近的边框边缘的距离 |
| marks | 设置是否cross marks或crop marks应仅仅被呈现于page box边缘之外 |
| maxHeight | 设置元素的最大高度 |
| maxWidth | 设置元素的最大宽度 |
| minHeight | 设置元素的最小高度 |
| minWidth | 设置元素的最小宽度 |
| overflow | 规定如何处理不适合元素盒的内容 |
| verticalAlign | 设置对元素中的内容进行垂直排列 |
| visibility | 设置元素是否可见 |
| width | 设置元素的宽度 |

List 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| listStyle | 在一行设置列表的所有属性 |
| listStyleImage | 把图像设置为列表项标记 |
| listStylePosition | 改变列表项标记的位置 |
| listStyleType | 设置列表项标记的类型 |

Positioning 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| bottom | 设置元素的底边缘距离父元素底边缘的之上或之下的距离 |
| left | 置元素的左边缘距离父元素左边缘的左边或右边的距离 |
| position | 把元素放置在static, relative, absolute 或 fixed 的位置 |
| right | 置元素的右边缘距离父元素右边缘的左边或右边的距离 |
| top | 设置元素的顶边缘距离父元素顶边缘的之上或之下的距离 |
| zIndex | 设置元素的堆叠次序 |

Printing 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| orphans | 设置段落留到页面底部的最小行数 |
| page | 设置显示某元素时使用的页面类型 |
| pageBreakAfter | 设置某元素之后的分页行为 |
| pageBreakBefore | 设置某元素之前的分页行为 |
| pageBreakInside | 设置某元素内部的分页行为 |
| size | 设置页面的方向和尺寸 |
| widows | 设置段落必须留到页面顶部的最小行数 |

Scrollbar 属性 (IE-only)

|  |  |
| --- | --- |
| 属性 | 描述 |
| scrollbar3dLightColor | 设置箭头和滚动条左侧和顶边的颜色 |
| scrollbarArrowColor | 设置滚动条上的箭头颜色 |
| scrollbarBaseColor | 设置滚动条的底色 |
| scrollbarDarkShadowColor | 设置箭头和滚动条右侧和底边的颜色 |
| scrollbarFaceColor | 设置滚动条的表色 |
| scrollbarHighlightColor | 设置箭头和滚动条左侧和顶边的颜色，以及滚动条的背景 |
| scrollbarShadowColor | 设置箭头和滚动条右侧和底边的颜色 |
| scrollbarTrackColor | 设置滚动条的背景色 |

Table 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| borderCollapse | 设置表格边框是否合并为单边框，或者像在标准的HTML中那样分离。 |
| borderSpacing | 设置分隔单元格边框的距离 |
| captionSide | 设置表格标题的位置 |
| emptyCells | 设置是否显示表格中的空单元格 |
| tableLayout | 设置用来显示表格单元格、行以及列的算法 |
|  |  |
|  |  |

Text 属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| color | 设置文本的颜色 |
| font | 在一行设置所有的字体属性 |
| fontFamily | 设置元素的字体系列。 |
| fontSize | 设置元素的字体大小。 |
| fontSizeAdjust | 设置/调整文本的尺寸 |
| fontStretch | 设置如何紧缩或伸展字体 |
| fontStyle | 设置元素的字体样式 |
| fontVariant | 用小型大写字母字体来显示文本 |
| fontWeight | 设置字体的粗细 |
| letterSpacing | 设置字符间距 |
| lineHeight | 设置行间距 |
| quotes | 设置在文本中使用哪种引号 |
| textAlign | 排列文本 |
| textDecoration | 设置文本的修饰 |
| textIndent | 缩紧首行的文本 |
| textShadow | 设置文本的阴影效果 |
| textTransform | 对文本设置大写效果 |
| unicodeBidi |  |
| whiteSpace | 设置如何设置文本中的折行和空白符 |
| wordSpacing | 设置文本中的词间距 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| dir | 设置或返回文本的方向 |
| lang | 设置或返回元素的语言代码 |
| title | 设置或返回元素的咨询性的标题 |

它是一组样式属性及其值的文本表示。这个文本格式化为一个 CSS 样式表，去掉了包围属性和值的元素选择器的花括号。

将这一属性设置为非法的值将会抛出一个代码为 SYNTAX\_ERR 的 DOMException 异常。当 CSS2Properties 对象是只读的时候，试图设置这一属性将会抛出一个代码为 NO\_MODIFICATION\_ALLOWED\_ERR 的 DOMException 异常。

关于 CSS2Properties 对象

CSS2Properties 对象表示一组 CSS 样式属性及其值。它为 CSS 规范定义的每一个 CSS 属性都定义一个 JavaScript 属性。

一个 HTMLElement 的 style 属性是一个可读可写的 CSS2Properties 对象，就好像 CSSRule 对象的 style 属性一样。不过，Window.getComputedStyle() 的返回值是一个 CSS2Properties 对象，其属性是只读的。

访问 Style 对象

可以从文档的头部或特定的 HTML 元素访问 Style 对象。

从文档的头部访问 style 对象：

实例

var x = document.getElementsByTagName("STYLE");

访问指定元素的样式对象：

实例

var x = document.getElementById("myH1").style;

创建 Style 对象

您可以使用 document.createElement() 方法创建 <style> 元素：

实例

var x = document.createElement("STYLE");

您还可以设置已有元素的样式属性：

实例

document.getElementById("myH1").style.color = "red";

相关页面

HTML 教程：HTML CSS

CSS 教程：CSS 教程

HTML 参考手册：HTML <style> 标签

CSS 参考手册：CSS 属性

# 第25章 Window 对象

Window 对象

Window 对象表示浏览器中打开的窗口。

如果文档包含框架（frame 或 iframe 标签），浏览器会为 HTML 文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象。

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

Window 对象集合

|  |  |
| --- | --- |
| 集合 | 描述 |
| frames[] | 返回窗口中所有命名的框架。  该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架或 <iframe>。属性 frames.length 存放数组 frames[] 中含有的元素个数。注意，frames[] 数组中引用的框架可能还包括框架，它们自己也具有 frames[] 数组。 |

Window 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| closed | 返回窗口是否已被关闭。 |
| defaultStatus | 设置或返回窗口状态栏中的默认文本。 |
| document | 对 Document 对象的只读引用。请参阅 Document 对象。 |
| history | 对 History 对象的只读引用。请参数 History 对象。 |
| innerheight | 返回窗口的文档显示区的高度。 |
| innerwidth | 返回窗口的文档显示区的宽度。 |
| length | 设置或返回窗口中的框架数量。 |
| location | 用于窗口或框架的 Location 对象。请参阅 Location 对象。 |
| name | 设置或返回窗口的名称。 |
| Navigator | 对 Navigator 对象的只读引用。请参数 Navigator 对象。 |
| opener | 返回对创建此窗口的窗口的引用。 |
| outerheight | 返回窗口的外部高度。 |
| outerwidth | 返回窗口的外部宽度。 |
| pageXOffset | 设置或返回当前页面相对于窗口显示区左上角的 X 位置。 |
| pageYOffset | 设置或返回当前页面相对于窗口显示区左上角的 Y 位置。 |
| parent | 返回父窗口。 |
| Screen | 对 Screen 对象的只读引用。请参数 Screen 对象。 |
| self | 返回对当前窗口的引用。等价于 Window 属性。 |
| status | 设置窗口状态栏的文本。 |
| top | 返回最顶层的先辈窗口。 |
| window | window 属性等价于 self 属性，它包含了对窗口自身的引用。 |
| screenLeft  screenTop  screenX  screenY | 只读整数。声明了窗口的左上角在屏幕上的的 x 坐标和 y 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY。 |

Window 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| alert() | 显示带有一段消息和一个确认按钮的警告框。 |
| blur() | 把键盘焦点从顶层窗口移开。 |
| clearInterval() | 取消由 setInterval() 设置的 timeout。 |
| clearTimeout() | 取消由 setTimeout() 方法设置的 timeout。 |
| close() | 关闭浏览器窗口。 |
| confirm() | 显示带有一段消息以及确认按钮和取消按钮的对话框。 |
| createPopup() | 创建一个 pop-up 窗口。 |
| focus() | 把键盘焦点给予一个窗口。 |
| moveBy() | 可相对窗口的当前坐标把它移动指定的像素。 |
| moveTo() | 把窗口的左上角移动到一个指定的坐标。 |
| open() | 打开一个新的浏览器窗口或查找一个已命名的窗口。 |
| print() | 打印当前窗口的内容。 |
| prompt() | 显示可提示用户输入的对话框。 |
| resizeBy() | 按照指定的像素调整窗口的大小。 |
| resizeTo() | 把窗口的大小调整到指定的宽度和高度。 |
| scrollBy() | 按照指定的像素值来滚动内容。 |
| scrollTo() | 把内容滚动到指定的坐标。 |
| setInterval() | 按照指定的周期（以毫秒计）来调用函数或计算表达式。 |
| setTimeout() | 在指定的毫秒数后调用函数或计算表达式。 |

Window 对象描述

Window 对象表示一个浏览器窗口或一个框架。在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用。例如，可以只写 document，而不必写 window.document。

同样，可以把当前窗口对象的方法当作函数来使用，如只写 alert()，而不必写 Window.alert()。

除了上面列出的属性和方法，Window 对象还实现了核心 JavaScript 所定义的所有全局属性和方法。

Window 对象的 window 属性和 self 属性引用的都是它自己。当你想明确地引用当前窗口，而不仅仅是隐式地引用它时，可以使用这两个属性。除了这两个属性之外，parent 属性、top 属性以及 frame[] 数组都引用了与当前 Window 对象相关的其他 Window 对象。

要引用窗口中的一个框架，可以使用如下语法：

frame[i] //当前窗口的框架

self.frame[i] //当前窗口的框架

w.frame[i] //窗口 w 的框架

要引用一个框架的父窗口（或父框架），可以使用下面的语法：

parent //当前窗口的父窗口

self.parent //当前窗口的父窗口

w.parent //窗口 w 的父窗口

要从顶层窗口含有的任何一个框架中引用它，可以使用如下语法：

top //当前框架的顶层窗口

self.top //当前框架的顶层窗口

f.top //框架 f 的顶层窗口

新的顶层浏览器窗口由方法 Window.open() 创建。当调用该方法时，应把 open() 调用的返回值存储在一个变量中，然后使用那个变量来引用新窗口。新窗口的 opener 属性反过来引用了打开它的那个窗口。

一般来说，Window 对象的方法都是对浏览器窗口或框架进行某种操作。而 alert() 方法、confirm() 方法和 prompt 方法则不同，它们通过简单的对话框与用户进行交互。

# 第26章 Console 对象

Console 对象

Console 对象提供对浏览器调试控制台的访问。

Console 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| assert() | 如果 assertion 为 false，则将错误消息写入控制台。 |
| clear() | 清空控制台。 |
| count() | 记录这个对 count() 的特定调用已被调用的次数。 |
| error() | 将错误消息输出到控制台。 |
| group() | 在控制台中创建新的分组。 |
| groupCollapsed() | 在控制台中创建新的行内组。但是新组是折叠创建的。用户需要使用按钮将其展开。 |
| groupEnd() | 退出控制台中的当前分组。 |
| info() | 将信息性消息输出到控制台。 |
| log() | 将消息输出到控制台。 |
| table() | 将表格式的数据显示为表。 |
| time() | 启动计时器（可跟踪操作需要多长时间）。 |
| timeEnd() | 停止以前由 console.time() 启动的计时器。 |
| trace() | 将堆栈跟踪输出到控制台。 |
| warn() | 将警告消息输出到控制台。 |

# 第27章 Geolocation 对象

Geolocation 对象

Geolocation 对象允许用户向 Web 应用程序提供其位置。出于隐私原因，会要求用户允许报告位置信息。

注意：此特性仅在安全上下文（HTTPS）中可用。

注意：对于 iPhone 等装有 GPS 的设备，地理位置最为准确。

Geolocation API 是通过 navigator.geolocation 对象发布的。

Geolocation 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| coordinates | 返回设备在地球上的位置和海拔。 |
| position | 返回在给定时间相关设备的位置。 |
| positionError | 返回使用地理定位设备时发生错误的原因。 |
| positionOptions | 描述包含选项属性的对象，该对象作为 Geolocation.getCurrentPosition() 和 Geolocation.watchPosition() 的参数传递。 |

Geolocation 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| clearWatch() | 注销先前使用 Geolocation.watchPosition() 安装的位置/错误监视处理程序。 |
| getCurrentPosition() | 返回设备的当前位置。 |
| watchPosition() | 返回监听 ID 值，然后将其传递给 Geolocation.clearWatch() 方法即可用于注销处理程序。 |

# 第28章 History 对象

History 对象

History 对象包含用户（在浏览器窗口中）访问过的 URL。

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

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

History 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| length | 返回浏览器历史列表中的 URL 数量。 |

History 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| back() | 加载 history 列表中的前一个 URL。 |
| forward() | 加载 history 列表中的下一个 URL。 |
| go() | 加载 history 列表中的某个具体页面。 |

History 对象描述

History 对象最初设计来表示窗口的浏览历史。但出于隐私方面的原因，History 对象不再允许脚本访问已经访问过的实际 URL。唯一保持使用的功能只有 back()、forward() 和 go() 方法。

例子

下面一行代码执行的操作与单击后退按钮执行的操作一样：

history.back()

下面一行代码执行的操作与单击两次后退按钮执行的操作一样：

history.go(-2)

# 第29章 Storage 对象

Storage 对象

Web Storage API 的 Storage 对象提供对特定域的会话存储或本地存储的访问。这使您可以读取、添加、修改和删除存储的数据项。

Storage 对象属性和方法

|  |  |
| --- | --- |
| 属性/方法 | 描述 |
| key(n) | 返回存储中第 n 个键的名称。 |
| length | 返回存储在 Storage 对象中的数据项的数量。 |
| getItem(keyname) | 返回指定键名的值。 |
| setItem(keyname, value) | 将键添加到存储中，或更新键的值（若已存在）。 |
| removeItem(keyname) | 从存储中删除键。 |
| clear() | 清空存储中的所有键。 |

Web Storage API 相关页面

|  |  |
| --- | --- |
| 属性 | 描述 |
| window.localStorage | 允许将键/值对保存在Web浏览器中。存储没有到期日期的数据。 |
| window.sessionStorage | 允许将键/值对保存在 Web 浏览器中。为会话存储数据。 |

# 第30章 HTML DOM Anchor 对象

Anchor 对象

Anchor 对象表示 HTML 超链接。

在 HTML 文档中 <a> 标签每出现一次，就会创建 Anchor 对象。

锚可用于创建指向另一个文档的链接（通过 href 属性），或者创建文档内的书签（通过 name 属性）。

您可以通过搜索 Document 对象中的 anchors[] 数组来访问锚，或者使用 document.getElementById()。

Anchor 对象的属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| accessKey | 设置或返回访问一个链接的快捷键。 |
| charset | 设置或返回被链接资源的字符集。 |
| coords | 设置或返回逗号分隔列表，包含了图像映射中链接的坐标。 |
| href | 设置或返回被链接资源的 URL。 |
| hreflang | 设置或返回被链接资源的语言代码。 |
| id | 设置或返回一个链接的 id。 |
| innerHTML | 设置或返回一个链接的内容。 |
| name | 设置或返回一个链接的名称。 |
| rel | 设置或返回当前文档与目标 URL 之间的关系。 |
| rev | 设置或返回目标 URL 与之间当前文档的关系。 |
| shape | 设置或返回图像映射中某个链接的形状。 |
| tabIndex | 设置或返回某个链接的 Tab 键控制次序。 |
| target | 设置或返回在何处打开链接。 |
| type | 设置或返回被链接资源的 MIME 类型。 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| dir | 设置或返回文本的方向。 |
| lang | 设置或返回元素的语言代码。 |

Anchor 对象的方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| blur() | 把焦点从链接上移开。 |
| focus() | 给链接应用焦点。 |
|  |  |

# 第31章 HTML DOM Abbreviation 对象

Abbreviation 对象

Abbreviation 对象代表 HTML <abbr> 元素。

访问 Abbreviation 对象

您可使用 getElementById() 来访问 <abbr> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 ABBR 元素的演示</h3>

<p>The <abbr id="myAbbr" title="World Health Organization">WHO</abbr> was founded in 1948.</p>

<p>单击按钮可获取“WHO”的缩写。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myAbbr").title;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Abbreviation 对象

您可使用 document.createElement() 方法来创建 <abbr> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮来创建包含缩写 "WHO" 的 ABBR 元素。</p>

<button onclick="myFunction()">试一试</button>

<p id="myP"></p>

<script>

function myFunction() {

var x = document.createElement("ABBR");

var t = document.createTextNode("WHO");

x.setAttribute("title", "World Health Organization");

x.appendChild(t);

document.getElementById("myP").appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Abbreviation 对象支持标准属性和事件。

# 第32章 HTML DOM Address 对象

Address 对象

Address 对象代表 HTML <address> 元素。

访问 Address 对象

您可使用 getElementById() 来访问 <address> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 ADDRESS 元素的演示</h3>

<address id="myAdr">

Example.com<br>

Box 564, Disneyland<br>

USA

</address>

<p>单击按钮将地址的文本颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myAdr");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Address 对象

您可使用 document.createElement() 方法来创建 <address> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮来创建带有一些文本的 ADDRESS 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("ADDRESS");

var t = document.createTextNode("Box 564, Disneyland, USA");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

# 第33章 HTML DOM Area 对象

Area 对象

Area 对象代表图像映射的一个区域（图像映射指的是带有可点击区域的图像）

在 HTML 文档中 <area> 标签每出现一次，就会创建一个 Area 对象。

Area 对象的属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| accessKey | 设置或返回访问某个区域的快捷键。 |
| alt | 设置或返回当浏览器无法显示某个区域时的替换文字。 |
| coords | 设置或返回图像映射中可点击区域的坐标。 |
| hash | 设置或返回某个区域中 URL 的锚部分。 |
| host | 设置或返回某个区域中 URL 的主机名和端口。 |
| href | 设置或返回图像映射中链接的 URL。 |
| id | 设置或返回某个区域的 id。 |
| noHref | 设置或返回某个区域是否应是活动的还是非活动的。 |
| pathname | 设置或返回某个区域中的 URL 的路径名。 |
| protocol | 设置或返回某个区域中的 URL 的协议。 |
| search | 设置或返回某个区域中 URL 的查询字符串部分。 |
| shape | 设置或返回图像映射中某个区域的形状。 |
| tabIndex | 设置或返回某个区域的 tab 键控制次序。 |
| target | 设置或返回在何处打开区域中的 link-URL。 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| dir | 设置或返回文本的方向。 |
| lang | 设置或返回元素的语言代码。 |
| title | 设置或返回元素的 title。 |

# 第34章 HTML DOM Article 对象

Article 对象

Article 对象代表 HTML <article> 元素。

注释：Internet Explorer 8 以及更早的版本不支持 <article> 元素。

访问 Article 对象

您可使用 getElementById() 来访问 <article> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 ARTICLE 元素的演示</h3>

<article id="myArticle">

<h1>Article Heading</h1>

<p>Some article text..</p>

</article>

<p>点击按钮将文章内容的文字颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myArticle");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Article 对象

您可使用 document.createElement() 方法来创建 <article> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮可创建包含 H1 和 P 元素的 ARTICLE 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("ARTICLE");

x.setAttribute("id", "myArticle");

document.body.appendChild(x);

var heading = document.createElement("H1");

var txt1 = document.createTextNode("Heading in Article");

heading.appendChild(txt1);

document.getElementById("myArticle").appendChild(heading);

var para = document.createElement("P");

var txt2 = document.createTextNode("Paragraph in article.");

para.appendChild(txt2);

document.getElementById("myArticle").appendChild(para);

}

</script>

</body>

</html>

标准属性和事件

Article 对象支持标准属性和事件。

# 第35章 HTML DOM Aside 对象

Aside 对象

Aside 对象代表 HTML <aside> 元素。

注释：Internet Explorer 8 以及更早的版本不支持 <aside> 元素。

访问 Aside 对象

您可使用 getElementById() 来访问 <aside> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 ASIDE 元素的演示</h3>

<p>"My family and I visited The Epcot center this summer."</p>

<aside id="myAside">

<h4>Epcot Center</h4>

<p>The Epcot Center is a theme park in Disney World, Florida.</p>

</aside>

<p>Click the button to set the text color of the aside content to red.</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myAside");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Aside 对象

您可使用 document.createElement() 方法来创建 <aside> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮可创建包含 H4 和 P 元素的 ASIDE 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("ASIDE");

x.setAttribute("id", "myAside");

document.body.appendChild(x);

var heading = document.createElement("H4");

var txt1 = document.createTextNode("Epcot Center");

heading.appendChild(txt1);

document.getElementById("myAside").appendChild(heading);

var para = document.createElement("P");

var txt2 = document.createTextNode("The Epcot Center is a theme park in Disney World.");

para.appendChild(txt2);

document.getElementById("myAside").appendChild(para);

}

</script>

</body>

</html>

标准属性和事件

Aside 对象支持标准属性和事件。

# 第36章 HTML DOM Audio 对象

Audio 对象

Audio 对象是 HTML5 中的新对象。

Audio 对象表示 HTML <audio> 元素。

访问 Audio 对象

您可以通过使用 getElementById() 来访问 <audio> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 AUDIO 元素</h3>

<audio id="myAudio" controls>

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

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

Your browser does not support the audio element.

</audio>

<p>请点击按钮来获得以秒计的音频时长。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myAudio").duration;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Audio 对象

您可以通过使用 document.createElement() 方法来创建 <audio> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 AUDIO 元素</h3>

<p>请点击按钮来创建 AUDIO 元素，该元素将播放 .ogg 文件格式的声音。</p>

<button onclick="myFunction()">试一下</button>

<p id="demo"></p>

<script>

function myFunction()

{

var x = document.createElement("AUDIO");

x.setAttribute("src", "/i/horse.mp3");

x.setAttribute("controls", "controls");

document.body.appendChild(x);

document.getElementById("demo").innerHTML = "<b>注释：</b>IE 和 Safari 不支持 .ogg 文件格式。这只是一个例子。如需使其在所有浏览器中运行，您应该在 audio 元素中使用 source 元素。";

}

</script>

</body>

</html>

Audio 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| audioTracks | 返回表示可用音频轨道的 AudioTrackList 对象。 |
| autoplay | 设置或返回是否在就绪（加载完成）后随即播放音频。 |
| buffered | 返回表示音频已缓冲部分的 TimeRanges 对象。 |
| controller | 返回表示音频当前媒体控制器的 MediaController 对象。 |
| controls | 设置或返回音频是否应该显示控件（比如播放/暂停等）。 |
| crossOrigin | 设置或返回音频的 CORS 设置。 |
| currentSrc | 返回当前音频的 URL。 |
| currentTime | 设置或返回音频中的当前播放位置（以秒计）。 |
| defaultMuted | 设置或返回音频默认是否静音。 |
| defaultPlaybackRate | 设置或返回音频的默认播放速度。 |
| duration | 返回音频的长度（以秒计）。 |
| ended | 返回音频的播放是否已结束。 |
| error | 返回表示音频错误状态的 MediaError 对象。 |
| loop | 设置或返回音频是否应在结束时再次播放。 |
| mediaGroup | 设置或返回音频所属媒介组合的名称。 |
| muted | 设置或返回是否关闭声音。 |
| networkState | 返回音频的当前网络状态。 |
| paused | 设置或返回音频是否暂停。 |
| playbackRate | 设置或返回音频播放的速度。 |
| played | 返回表示音频已播放部分的 TimeRanges 对象。 |
| preload | 设置或返回音频的 preload 属性的值。 |
| readyState | 返回音频当前的就绪状态。 |
| seekable | 返回表示音频可寻址部分的 TimeRanges 对象。 |
| seeking | 返回用户当前是否正在音频中进行查找。 |
| src | 设置或返回音频的 src 属性的值。 |
| textTracks | 返回表示可用文本轨道的 TextTrackList 对象。 |
| volume | 设置或返回音频的音量。 |

Audio 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| addTextTrack() | 向音频添加新的文本轨道。 |
| canPlayType() | 检查浏览器是否能够播放指定的音频类型。 |
| fastSeek() | 在音频播放器中指定播放时间。 |
| getStartDate() | 返回新的 Date 对象，表示当前时间线偏移量。 |
| load() | 重新加载音频元素。 |
| play() | 开始播放音频。 |
| pause() | 暂停当前播放的音频。 |

标准属性和事件

Audio 对象支持标准属性和事件。

# 第37章 HTML DOM Bold 对象

Bold 对象

Bold 对象代表 HTML <b> 元素。

访问 Bold 对象

您可使用 getElementById() 来访问 <b> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 B 元素的演示</h3>

<p>单击按钮将 <b id="myB">此粗体文本</b> 的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myB");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Bold 对象

您可使用 document.createElement() 方法来创建 <b> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击按钮来创建带有一些文本的 B 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("B");

var t = document.createTextNode("Some bold text");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Bold 对象支持标准属性和事件。

# 第38章 HTML DOM Base 对象

Base 对象

Base 对象代表 HTML 的 base 元素。

在 HTML 文档中 <base> 每出现一次，Base 对象就会被创建。

Base 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| href | 设置或返回针对页面中所有链接的基准 URL。 |
| id | 设置或返回 <base> 元素的 id。 |
| target | 设置或返回针对页面中所有链接的默认目标框架。 |

# 第39章 HTML DOM Bdo 对象

Bdo 对象

Bdo 对象代表 HTML <bdo> 元素。

访问 Bdo 对象

您可使用 getElementById() 来访问 <bdo> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 BDO 元素的演示</h3>

<p><bdo id="myBdo" dir="rtl">这一段将从右到左。</bdo></p>

<p>单击按钮可获取 bdo 元素内文本的文本方向。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myBdo");

if (x.dir === "rtl") {

document.getElementById("demo").innerHTML = "Text direction is right to left.";

} else {

document.getElementById("demo").innerHTML = "Text direction is left to right.";

}

}

</script>

</body>

</html>

创建 Bdo 对象

您可使用 document.createElement() 方法来创建 <bdo> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建一个 BDO 元素，其中文本从右到左。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("BDO");

var t = document.createTextNode("This text will go right-to-left.");

x.setAttribute("dir", "rtl");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Bdo 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| dir | 设置或返回 <bdo> 元素的 dir 属性的值 |
|  |  |

标准属性和事件

Bdo 对象支持标准属性和事件。

# 第40章 HTML DOM Blockquote 对象

Blockquote 对象

Blockquote 对象表示 HTML <blockquote> 元素。

访问 Blockquote 对象

您可以通过使用 getElementById() 来访问 <blockquote> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 BLOCKQUOTE 元素</h3>

<p>Here is a quote from WWF's website:</p>

<blockquote id="myBlockquote" cite="http://www.worldwildlife.org/who/index.html">

For 50 years, WWF has been protecting the future of nature.

</blockquote>

<p>请点击按钮来获得引用的 URL。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myBlockquote").cite;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Blockquote 对象

您可以通过使用 document.createElement() 方法来创建 <blockquote> 元素：

var x = document.createElement("BLOCKQUOTE");

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 BLOCKQUOTE 元素</h3>

<p>请点击按钮来创建 BLOCKQUOTE 元素，此引用的来源是 worldwildlife.org。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("BLOCKQUOTE");

var t = document.createTextNode("For 50 years, WWF has been protecting the future of nature.");

x.setAttribute("cite", "http://www.worldwildlife.org/who/index.html");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Blockquote 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| cite | 设置或返回引用的 cite 属性的值。 |

标准属性和事件

Blockquote 对象支持标准属性和事件。

# 第41章 HTML DOM Body 对象

Body 对象

Body 对象代表文档的主体 (HTML body) 。

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| dir | 设置或返回文本的方向。 |
| id | 设置或返回 body 的 id。 |
| lang | 设置或返回元素的语言代码。 |
| title | 设置或返回元素的咨询性的标题。 |

# 第42章 HTML DOM BR 对象

BR 对象

BR 对象代表 HTML <br> 元素。

访问 BR 对象

您可使用 getElementById() 来访问 <br> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 BR 元素的演示</h3>

<div>This is a div <br id="myBR"> element with some <br> line breaks.</div>

<p>单击按钮将文档中的第一个 BR 元素设置为 display="none"。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myBR");

x.style.display = "none";

}

</script>

</body>

</html>

创建 BR 对象

您可使用 document.createElement() 方法来创建 <br> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击按钮来创建一些 BR 元素。</p>

<button onclick="myFunction()">试一试</button>

<p>当您单击该按钮时，会在 div 中的每个 span 元素之前插入一个 br 元素，从而在下面的文本中进行适当的换行。</p>

<div id="myDIV">

<span>Hello,</span>

<span>My name is Bob.</span>

<span>I live in Bobville.</span>

<span>I am 45 years old.</span>

</div>

<script>

function myFunction() {

// 获取 id="myDIV" 的 div 元素

var div = document.getElementById("myDIV");

// 获取 div 内的所有 span 元素

var spans = div.getElementsByTagName("SPAN");

// 创建一个循环，它将在 div 中的每个 span 元素之前插入一个 br 元素，从第二个 span 元素开始

var i;

for (i = 1; i < spans.length; i++) {

var newElem = document.createElement("BR");

div.insertBefore(newElem, spans[i]);

}

}

</script>

</body>

</html>

BR 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| clear | HTML5 不支持。请改用 style.clear。  设置或返回浮动对象周围的文本流。 |

# 第43章 HTML DOM Button 对象

Button 对象

Button 对象代表一个按钮。

在 HTML 文档中 <button> 标签每出现一次，Button 对象就会被创建。

Button 对象的属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| accessKey | 设置或返回访问某个按钮的快捷键。 |
| disabled | 设置或返回是否禁用按钮。 |
| form | 返回对包含按钮的表单的引用。 |
| id | 设置或返回按钮的 id。 |
| name | 设置或返回按钮的名称。 |
| tabIndex | 设置或返回按钮的 Tab 键控制次序。 |
| type | 返回按钮的表单类型。 |
| value | 设置或返回显示在按钮上的文本。 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| dir | 设置或返回文本的方向。 |
| lang | 设置或返回元素的语言代码。 |
| title | 设置或返回元素的 title 属性 |

# 第44章 HTML DOM Canvas 对象

Canvas 对象

Canvas 对象表示一个 HTML 画布元素 - <canvas>。它没有自己的行为，但是定义了一个 API 支持脚本化客户端绘图操作。

你可以直接在该对象上指定宽度和高度，但是，其大多数功能都可以通过 CanvasRenderingContext2D 对象获得。 这是通过 Canvas 对象的 getContext() 方法并且把直接量字符串 "2d" 作为唯一的参数传递给它而获得的。

<canvas> 标记在 Safari 1.3 中引入，在制作此参考页时，它在 Firefox 1.5 和 Opera 9 中也得到了支持。在 IE 中，<canvas> 标记及其 API 可以使用位于 excanvas.sourceforge.net 的 ExplorerCanvas 开源项目来模拟。

提示：如果希望学习如何使用 <canvas> 来绘制图形，可以访问 Mozilla 提供的 Canvas 教程（英文）以及相应的 中文 Canvas 教程。

Canvas 对象的属性

height 属性

画布的高度。和一幅图像一样，这个属性可以指定为一个整数像素值或者是窗口高度的百分比。当这个值改变的时候，在该画布上已经完成的任何绘图都会擦除掉。默认值是 300。

width 属性

画布的宽度。和一幅图像一样，这个属性可以指定为一个整数像素值或者是窗口宽度的百分比。当这个值改变的时候，在该画布上已经完成的任何绘图都会擦除掉。默认值是 300。

Canvas 对象的方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| getContext() | 返回一个用于在画布上绘图的环境。 |

# 第45章 HTML DOM Caption 对象

Caption 对象

Caption 对象代表 HTML <caption> 元素。

访问 Caption 对象

您可使用 getElementById() 来访问 <caption> 元素：

实例

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

}

</style>

</head>

<body>

<h3>如何访问 CAPTION 元素的演示</h3>

<table>

<caption id="myCaption">Monthly savings</caption>

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

<td>February</td>

<td>$50</td>

</tr>

</table>

<p>单击按钮将标题的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myCaption");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Caption 对象

您可使用 document.createElement() 方法来创建 <caption> 元素：

实例

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

margin-top: 20px;

}

</style>

</head>

<body>

<p>单击该按钮以创建 CAPTION 元素。</p>

<button onclick="myFunction()">试一试</button>

<table id="myTable">

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

<td>February</td>

<td>$50</td>

</tr>

</table>

<script>

function myFunction() {

var x = document.createElement("CAPTION");

var t = document.createTextNode("Monthly savings");

x.appendChild(t);

var table = document.getElementById("myTable")

table.insertBefore(x, table.childNodes[0]);

}

</script>

</body>

</html>

提示：您还可以使用 Table 对象的 createCaption 方法创建 <caption> 元素。

Caption 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | HTML5 不支持。请改用 style.textAlign 或 style.captionSide。  设置或返回标题的对齐方式。 |

标准属性和事件

Caption 对象支持标准属性和事件。

# 第46章 HTML DOM Cite 对象

Cite 对象

Cite 对象代表 HTML <cite> 元素。

访问 Cite 对象

您可使用 getElementById() 来访问 <cite> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 CITE 元素的演示</h3>

<img src="/i/photo/tulip.jpg" alt="The Tulip" width="300" height="300">

<p><cite id="myCite">《郁金香》</cite> W3School 团队拍摄于 2019 年。</p>

<p>单击按钮将引文的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myCite");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Cite 对象

您可使用 document.createElement() 方法来创建 <cite> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 CITE 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<img src="/i/photo/tulip.jpg" alt="The Tulip" width="300" height="300">

<script>

function myFunction() {

var x = document.createElement("CITE");

var t = document.createTextNode("《郁金香》");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Cite 对象支持标准属性和事件。

# 第47章 HTML DOM Code 对象

Code 对象

Code 对象代表 HTML <code> 元素。

访问 Code 对象

您可使用 getElementById() 来访问 <code> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 CODE 元素的演示</h3>

<code id="myCode">A piece of computer code</code>

<p>单击按钮将代码元素的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myCode");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Code 对象

您可使用 document.createElement() 方法来创建 <code> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 CODE 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<script>

function myFunction() {

var x = document.createElement("CODE");

var t = document.createTextNode("A piece of computer code");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Code 对象支持标准属性和事件。

# 第48章 HTML DOM Column 对象

Column 对象

Column 对象表示 HTML <col> 元素。

访问 Column 对象

您可以通过使用 getElementById() 来访问 <col> 元素：

<!DOCTYPE html>

<html>

<head>

<style>

table,th,td

{

border:1px solid black;

}

</style>

</head>

<body>

<h3>关于如何访问 COL 元素的一个演示</h3>

<table>

<colgroup>

<col id="myCol" span="2" style="background-color:red">

<col style="background-color:yellow">

</colgroup>

<tr>

<th>ISBN</th>

<th>Title</th>

<th>Price</th>

</tr>

<tr>

<td>3476896</td>

<td>My first HTML</td>

<td>$53</td>

</tr>

<tr>

<td>5869207</td>

<td>My first CSS</td>

<td>$49</td>

</tr>

</table>

<p>请点击按钮，来获得 COL 元素应该横跨的列数。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myCol").span;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Column 对象

您可以通过使用 document.createElement() 方法来创建 <col> 元素：

var x = document.createElement("COL");

Column 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| span | 设置或返回列的 span 属性的值。 |

标准属性和事件

Column 对象支持标准属性和事件。

# 第49章 HTML DOM ColumnGroup 对象

ColumnGroup 对象

ColumnGroup 对象表示 HTML <colgroup> 元素。

访问 ColumnGroup 对象

您可以通过使用 getElementById() 来访问 <colgroup> 元素：

<!DOCTYPE html>

<html>

<head>

<style>

table,th,td

{

border:1px solid black;

}

</style>

</head>

<body>

<h3>关于如何访问 COLGROUP 元素的一个演示</h3>

<table>

<colgroup id="myColgroup" span="2" style="background:red"></colgroup>

<tr>

<th>ISBN</th>

<th>Title</th>

<th>Price</th>

</tr>

<tr>

<td>3476896</td>

<td>My first HTML</td>

<td>$53</td>

</tr>

<tr>

<td>5869207</td>

<td>My first CSS</td>

<td>$49</td>

</tr>

</table>

<p>请点击按钮，来获得 COLGROUP 元素应该横跨的列数。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myColgroup").span;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 ColumnGroup 对象

您可以通过使用 document.createElement() 方法来创建 <colgroup> 元素：

var x = document.createElement("COLGROUP");

ColumnGroup 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| span | 设置或返回列的 span 属性的值。 |

标准属性和事件

ColumnGroup 对象支持标准属性和事件。

# 第50章 HTML DOM Datalist 对象

Datalist 对象

Datalist 对象是 HTML5 中的新对象。

Datalist 对象表示 HTML <datalist> 元素。

访问 Datalist 对象

您可以通过使用 getElementById() 来访问 <datalist> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 Datalist 元素</h3>

<form>

<input list="myDatalist">

<datalist id="myDatalist">

<option value="Internet Explorer">

<option value="Firefox">

<option value="Chrome">

<option value="Opera">

<option value="Safari">

</datalist>

</form>

<p>请点击按钮来获得 datalist 中 option 元素的数目。</p>

<button onclick="myFunction()">试一下</button>

<p id="demo"></p>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 datalist 元素。</p>

<p><b>注释：</b>input 元素的 list 属性用于绑定 datalist 元素。</p>

<script>

function myFunction() {

var x = document.getElementById("myDatalist").options.length;

document.getElementById("demo").innerHTML = "Found " + x + " options in the list.";

}

</script>

</body>

</html>

创建 Datalist 对象

您可以通过使用 document.createElement() 方法来创建 <datalist> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 Datalist 元素</h3>

<p>请点击按钮来创建 INPUT 字段、DATALIST 元素以及 OPTION 元素。</p>

<form id="myForm"></form>

<button onclick="myFunction()">试一下</button>

<p><b>注释：</b>Internet Explorer 9 以及更早的版本不支持 datalist 元素。</p>

<script>

function myFunction() {

var x = document.createElement("INPUT");

x.setAttribute("list", "browsers");

document.getElementById("myForm").appendChild(x);

var y = document.createElement("DATALIST");

y.setAttribute("id", "browsers");

document.getElementById("myForm").appendChild(y);

var z = document.createElement("OPTION");

z.setAttribute("value", "Chrome");

document.getElementById("browsers").appendChild(z);

}

</script>

</body>

</html>

Datalist 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| options | 返回 datalist 中所有选项的集合。 |

标准属性和事件

Datalist 对象支持标准属性和事件。

# 第51章 HTML DOM DD 对象

DD 对象

DD 对象代表 HTML <dd> 元素。

访问 DD 对象

您可使用 getElementById() 来访问 <dd> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 DD 元素的演示</h3>

<dl>

<dt>Coffee</dt>

<dd id="myDD">Black hot drink</dd>

</dl>

<p>点击按钮来获取 dd 元素的 HTML 内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myDD").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 DD 对象

您可使用 document.createElement() 方法来创建 <dd> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 DD 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<dl id="myDL">

<dt>Coffee</dt>

</dl>

<script>

function myFunction() {

var x = document.createElement("DD");

var t = document.createTextNode("Black hot drink");

x.appendChild(t);

var y = document.getElementById("myDL");

y.appendChild(x);

}

</script>

</body>

</html>

# 第52章 HTML DOM Del 对象

Del 对象

Del 对象表示 HTML <del> 元素。

访问 Del 对象

您可以通过使用 getElementById() 来访问 <del> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 DEL 元素</h3>

<p><del id="myDel" cite="why\_deleted.htm">这段文本已被删除。</del></p>

<p>点击按钮来获得页面的 URL，此页面解释了文本被删除的原因。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myDel").cite;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Del 对象

您可以通过使用 document.createElement() 方法来创建 <del> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 DEL 元素</h3>

<p>点击按钮来创建 DEL 元素。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("DEL");

var t = document.createTextNode("Some deleted text");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Del 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| cite | 设置或者返回被删除文本的 cite 属性值。 |
| dateTime | 设置或者返回被删除文本的 datetime 属性值。 |

标准属性和事件

Del 对象支持标准属性和事件。

# 第53章 HTML DOM Details 对象

Details 对象

Details 对象表示 HTML <details> 元素。

访问 Details 对象

您可以通过使用 getElementById() 来访问 <details> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 DETAILS 元素</h3>

<details id="myDetails">

Some additional details...

</details>

<p>点击按钮来显示额外的细节。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myDetails");

x.open=true;

}

</script>

</body>

</html>

创建 Details 对象

您可以通过使用 document.createElement() 方法来创建 <details> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 DETAILS 元素</h3>

<p>点击按钮来创建 DETAILS 元素。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("DETAILS");

var t = document.createTextNode("Some additional details...");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Details 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| open | 设置或者返回 details 对于用户是否是可见的。 |

标准属性和事件

Details 对象支持标准属性和事件。

# 第54章 HTML DOM DFN 对象

DFN 对象

DFN 对象代表 HTML <dfn> 元素。

访问 DFN 对象

您可使用 getElementById() 来访问 <dfn> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 DFN 元素的演示</h3>

<dfn id="myDFN">Definition term</dfn>

<p>单击按钮将定义项的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myDFN");

x.style.color = "red";

}

</script>

</body>

</html>

创建 DFN 对象

您可使用 document.createElement() 方法来创建 <dfn> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮来创建带有一些文本的 DFN 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<script>

function myFunction() {

var x = document.createElement("DFN");

var t = document.createTextNode("A definition term");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

DFN 对象支持标准属性和事件。

# 第55章 HTML DOM Dialog 对象

Dialog 对象

Dialog 对象是 HTML5 中的新对象。

Dialog 对象表示 HTML <dialog> 元素。

注释：目前只有 Chrome Canary 和 Safari 6 支持 <dialog> 元素。

访问 Dialog 对象

您可以通过使用 getElementById() 来访问 <dialog> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 Dialog 元素</h3>

<p>点击按钮来打开对话窗口。</p>

<button onclick="myFunction()">试一下</button>

<p><b>注释：</b>目前只有 Chrome Canary 和 Safari 6 支持 <dialog> 元素。</p>

<dialog id="myDialog">This is an open dialog window</dialog>

<script>

function myFunction()

{

var x = document.getElementById("myDialog");

x.open = true;

}

</script>

</body>

</html>

创建 Dialog 对象

您可以通过使用 document.createElement() 方法来创建 <dialog> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 Dialog 元素</h3>

<p>点击按钮来创建含有文本的 DIALOG 元素。</p>

<button onclick="myFunction()">试一下</button>

<p><b>注释：</b>目前只有 Chrome Canary 和 Safari 6 支持 <dialog> 元素。</p>

<script>

function myFunction()

{

var x = document.createElement("DIALOG");

var t = document.createTextNode("This is an open dialog window");

x.setAttribute("open", "open");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Dialog 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| open | 设置或返回对话是否是打开的。 |
| returnValue | 设置或返回对话的返回值。 |

Dialog 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| close() | 关闭对话。 |
| show() | 显示对话。 |
| showModal() | 显示对话，并使其成为 top-most 型对话。 |

标准属性和事件

Dialog 对象支持标准属性和事件。

# 第56章 HTML DOM Div 对象

Div 对象

Div 对象代表 HTML <div> 元素。

访问 Div 对象

您可使用 getElementById() 来访问 <div> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 DIV 元素的演示</h3>

<div id="myDIV">This is a div element.</div>

<p>点击按钮获取 div 元素的内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myDIV").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Div 对象

您可使用 document.createElement() 方法来创建 <div> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建一个带有粉红色背景颜色和一些文本的 DIV 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("DIV");

var t = document.createTextNode("This is a div element.");

x.setAttribute("style", "background-color: pink;");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Div 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | 标准属性和事件  Div 对象支持标准属性和事件。 |

HTML5 中不支持。请改用 style.textAlign。

设置或返回 <div> 元素的 align 属性值。

# 第57章 HTML DOM DList 对象

DList 对象

DList 对象代表 HTML <dl> 元素。

访问 DList 对象

您可使用 getElementById() 来访问 <dl> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 DL 元素的演示</h3>

<dl id="myDL">

<dt>Coffee</dt>

<dd>Black hot drink</dd>

<dt>Milk</dt>

<dd>White cold drink</dd>

</dl>

<p>单击按钮可获取 dl 元素的 HTML 内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myDL").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 DList 对象

您可使用 document.createElement() 方法来创建 <dl> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮可创建 DL、DT 和 DD 元素：描述列表，包含术语和描述。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("DL");

x.setAttribute("id", "myDL");

document.body.appendChild(x);

var y = document.createElement("DT");

var txt1 = document.createTextNode("Coffee");

y.appendChild(txt1);

y.setAttribute("id", "myDT");

document.getElementById("myDL").appendChild(y);

var z = document.createElement("DD");

var txt2 = document.createTextNode("Black hot drink");

z.appendChild(txt2);

document.getElementById("myDL").appendChild(z);

}

</script>

</body>

</html>

标准属性和事件

DList 对象支持标准属性和事件。

# 第58章 HTML DOM DT 对象

DT 对象

DT 对象代表 HTML <dt> 元素。

访问 DT 对象

您可使用 getElementById() 来访问 <dt> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 DT 元素的演示</h3>

<dl>

<dt id="myDT">Coffee</dt>

<dd>Black hot drink</dd>

</dl>

<p>单击该按钮来获取 dt 元素的 HTML 内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myDT").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 DT 对象

您可使用 document.createElement() 方法来创建 <dt> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮可创建 DL、DT 和 DD 元素：描述列表，包含术语和描述。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("DL");

x.setAttribute("id", "myDL");

document.body.appendChild(x);

var y = document.createElement("DT");

var txt1 = document.createTextNode("Coffee");

y.appendChild(txt1);

y.setAttribute("id", "myDT");

document.getElementById("myDL").appendChild(y);

var z = document.createElement("DD");

var txt2 = document.createTextNode("Black hot drink");

z.appendChild(txt2);

document.getElementById("myDL").appendChild(z);

}

</script>

</body>

</html>

标准属性和事件

DT 对象支持标准属性和事件。

# 第59章 HTML DOM Emphasized 对象

Emphasized 对象

Emphasized 对象代表 HTML <em> 元素。

访问 Emphasized 对象

您可使用 getElementById() 来访问 <em> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 EM 元素的演示</h3>

<p>点击按钮将 <em id="myEm">这个强调文本</em> 的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myEm");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Emphasized 对象

您可使用 document.createElement() 方法来创建 <em> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 EM 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("EM");

var t = document.createTextNode("Some emphasized text");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Emphasized 对象支持标准属性和事件。

# 第60章 HTML DOM Embed 对象

Embed 对象

Embed 对象是 HTML5 中的新对象。

Embed 对象表示 HTML <embed> 元素。

访问 Embed 对象

您可以通过使用 getElementById() 来访问 <embed> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 Embed 元素</h3>

<embed id="myEmbed" src="/i/helloworld.swf">

<p>点击按钮来获得内嵌 flash 文件的 URL。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myEmbed").src;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Embed 对象

您可以通过使用 document.createElement() 方法来创建 <embed> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 Embed 元素</h3>

<p>点击按钮来创建含有内嵌 flash 动画的 EMBED 元素。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("EMBED");

x.setAttribute("src", "/i/helloworld.swf");

document.body.appendChild(x);

}

</script>

</body>

</html>

Embed 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| height | 设置或返回 embed 元素中 height 属性的值。 |
| src | 设置或返回 embed 元素中 src 属性的值。 |
| type | 设置或返回 embed 元素中 type 属性的值。 |
| width | 设置或返回 embed 元素中 width 属性的值。 |

标准属性和事件

Embed 对象支持标准属性和事件。

# 第61章 HTML DOM Fieldset 对象

Fieldset 对象

Fieldset 对象表示 HTML <fieldset> 元素。

访问 Fieldset 对象

您可以通过使用 getElementById() 来访问 <fieldset> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 FIELDSET 元素</h3>

<fieldset id="myFieldset" name="personalia">

Name: <input type="text" name="username"><br>

Email: <input type="text" name="usermail"><br>

</fieldset>

<p>点击按钮来禁用 fieldset。</p>

<button onclick="myFunction()">试一下</button>

<p><b>注释：</b>Internet Explorer 和 Safari 不完全支持 Fieldset 对象的 disabled 属性。</p>

<script>

function myFunction()

{

var x = document.getElementById("myFieldset");

x.disabled = true;

}

</script>

</body>

</html>

提示：您也可以通过遍历表单的 elements 集合来访问 Fieldset 对象。

创建 Fieldset 对象

您可以通过使用 document.createElement() 方法来创建 <fieldset> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 FIELDSET 元素</h3>

<p>点击按钮来创建包含文本的 FIELDSET 元素。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("FIELDSET");

var t = document.createTextNode("Insert content here...");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Fieldset 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| disabled | 设置或返回 fieldset 是否被禁用。 |
| form | 返回对包含该 fieldset 的表单的引用。 |
| name | 设置或返回 fieldset 的 name 属性值。 |
| type | 返回 fieldset 的表单元素类型。 |

标准属性和事件

Fieldset 对象支持标准属性和事件。

# 第62章 HTML DOM Figcaption 对象

Figcaption 对象

Figcaption 对象代表 HTML <figcaption> 元素。

访问 Figcaption 对象

您可使用 getElementById() 来访问 <figcaption> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 FIGCAPTION 元素的演示</h3>

<figure>

<img src="/i/photo/tulip.jpg" alt="The Tulip" width="300" height="300">

<figcaption id="myFigCap">Fig.1 - 郁金香是百合科郁金香属的一个栽培种，又称洋荷花、旱荷花。</figcaption>

</figure>

<p>单击按钮将标题的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myFigCap");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Figcaption 对象

您可使用 document.createElement() 方法来创建 <figcaption> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 FIGCAPTION 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<figure id="myFigure">

<img src="/i/photo/tulip.jpg" alt="The Tulip" width="300" height="300">

</figure>

<script>

function myFunction() {

var x = document.createElement("FIGCAPTION");

var t = document.createTextNode("Fig.1 - 郁金香是百合科郁金香属的一个栽培种，又称洋荷花、旱荷花。");

x.appendChild(t);

var y = document.getElementById("myFigure");

y.appendChild(x);

}

</script>

</body>

</html>

标准属性和事件

Figcaption 对象支持标准属性和事件。

# 第63章 HTML DOM Figure 对象

Figure 对象

Figure 对象代表 HTML <figure> 元素。

访问 Figure 对象

您可使用 getElementById() 来访问 <figure> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 Figure 元素的演示</h3>

<p>郁金香（学名：Tulipa gesneriana）是百合科郁金香属的一个栽培种，又称洋荷花、旱荷花。</p>

<figure id="myFigure">

<img src="/i/photo/tulip.jpg" alt="The Tulip" width="300" height="300">

</figure>

<p>单击按钮以获取 Figure 元素的 id。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myFigure").id;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Figure 对象

您可使用 document.createElement() 方法来创建 <figure> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮来创建一个包含 IMG 元素的 Figure 元素。</p>

<button onclick="myFunction()">试一试</button><br><br>

<script>

function myFunction() {

var x = document.createElement("FIGURE");

x.setAttribute("id", "myFigure");

document.body.appendChild(x);

var y = document.createElement("IMG");

y.setAttribute("src", "/i/photo/tulip.jpg");

y.setAttribute("width", "300");

y.setAttribute("width", "300");

y.setAttribute("alt", "The Tulip");

x.appendChild(y);

}

</script>

</body>

</html>

# 第64章 HTML DOM Footer 对象

Footer 对象

Footer 对象代表 HTML <footer> 元素。

注释：Internet Explorer 8 以及更早的版本不支持 <footer> 元素。

访问 Footer 对象

您可使用 getElementById() 来访问 <footer> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 FOOTER 元素的演示</h3>

<footer id="myFooter">

<p>Some text in footer.</p>

</footer>

<p>单击按钮获取 footer 元素的内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myFooter").innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Footer 对象

您可使用 document.createElement() 方法来创建 <footer> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建一个包含带有一些文本的 P 元素的 FOOTER 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("FOOTER");

x.setAttribute("id", "myFooter");

document.body.appendChild(x);

var y = document.createElement("P");

var t = document.createTextNode("This is a p element in a footer element.");

y.appendChild(t);

document.getElementById("myFooter").appendChild(y);

}

</script>

</body>

</html>

# 第65章 HTML DOM Form 对象

Form 对象

Form 对象代表一个 HTML 表单。

在 HTML 文档中 <form> 每出现一次，Form 对象就会被创建。

Form 对象集合

|  |  |
| --- | --- |
| 集合 | 描述 |
| elements[] | 包含表单中所有元素的数组。 |

Form 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| acceptCharset | 服务器可接受的字符集。 |
| action | 设置或返回表单的 action 属性。 |
| enctype | 设置或返回表单用来编码内容的 MIME 类型。 |
| id | 设置或返回表单的 id。 |
| length | 返回表单中的元素数目。 |
| method | 设置或返回将数据发送到服务器的 HTTP 方法。 |
| name | 设置或返回表单的名称。 |
| target | 设置或返回表单提交结果的 Frame 或 Window 名。 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| dir | 设置或返回文本的方向。 |
| lang | 设置或返回元素的语言代码。 |
| title | 设置或返回元素的 title 属性。 |

Form 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| reset() | 把表单的所有输入元素重置为它们的默认值。 |
| submit() | 提交表单。 |

Form 对象事件句柄

|  |  |
| --- | --- |
| 事件句柄 | 描述 |
| onreset | 在重置表单元素之前调用。 |
| onsubmit | 在提交表单之前调用。 |

# 第66章 HTML DOM Head 对象

Head 对象

Head 对象代表 HTML <head> 元素。

访问 Head 对象

您可使用 getElementById() 来访问 <head> 元素：

var x = document.getElementsByTagName("HEAD")[0];

创建 Head 对象

您可使用 document.createElement() 方法来创建 <head> 元素：

var x = document.createElement("HEAD");

# 第67章 HTML DOM Header 对象

Header 对象

Header 对象代表 HTML <header> 元素。

注释：Internet Explorer 8 以及更早的版本不支持 <header> 元素。

访问 Header 对象

您可使用 getElementById() 来访问 <header> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 HEADER 元素的演示</h3>

<article>

<header id="myHeader">

<h3>Internet Explorer 9</h3>

</header>

<p>Windows Internet Explorer 9 (abbreviated as IE9) was released to

the public on March 14, 2011 at 21:00 PDT.....</p>

</article>

<p>单击按钮将标题元素的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myHeader");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Header 对象

您可使用 document.createElement() 方法来创建 <header> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建一个 HEADER 元素，其中包含一个带有一些文本的 h3 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("HEADER");

x.setAttribute("id", "myHeader");

document.body.appendChild(x);

var y = document.createElement("H3");

var t = document.createTextNode("This is a h3 element in a header element");

y.appendChild(t);

document.getElementById("myHeader").appendChild(y);

}

</script>

</body>

</html>

# 第68章 HTML DOM Heading 对象

Heading 对象

Heading 对象代表 HTML 标题元素：<h1> 到 <h6>。

访问 Heading 对象

您可使用 getElementById() 来访问标题元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 H2 元素的演示</h3>

<h2 id="myHeading">This is a h2 element.</h2>

<p>单击按钮将 h2 元素的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myHeading");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Heading 对象

您可使用 document.createElement() 方法来创建标题元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 H1 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("H1");

var t = document.createTextNode("Welcome to My Homepage");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Heading 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | HTML5 中不支持。请改用 style.textAlign。 |

# 第69章 HTML DOM HR 对象

HR 对象

HR 对象代表 HTML <hr> 元素。

访问 HR 对象

您可使用 getElementById() 来访问 <hr> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 HR 元素的演示</h3>

<hr id="myHR">

<p>单击按钮隐藏 HR 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myHR");

x.style.display = "none";

}

</script>

</body>

</html>

创建 HR 对象

您可使用 document.createElement() 方法来创建 <hr> 元素：

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建 HR 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("HR");

document.body.appendChild(x);

}

</script>

</body>

</html>

HR 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | HTML5 中不支持。请改用 style.textAlign。  设置或返回水平线的对齐方式。 |
| color | HTML5 中不支持。请改用 style.color。  设置或返回水平线的颜色。 |
| noshade | HTML5 中不支持。  设置或返回水平线是否应以一种纯色呈现（无阴影）。 |
| size | HTML5 中不支持。请改用 style.height。  设置或返回水平线的高度。 |
| width | HTML5 中不支持。请改用 style.width。  设置或返回水平线的宽度。 |

# 第70章 HTML DOM HTML 对象

HTML 对象

HTML 对象代表 HTML <html> 元素。

访问 HTML 对象

您可使用 getElementById() 来访问 <html> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 HTML 元素的演示</h3>

<p>单击该按钮可获取 html 元素的 HTML 内容。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementsByTagName("HTML")[0].innerHTML;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

提示：您还可以使用 document.documentElement 属性访问 <html> 元素。

标准属性和事件

HTML 对象支持标准属性和事件。

# 第71章 HTML DOM Italic 对象

Italic 对象

Italic 对象代表 HTML <i> 元素。

访问 Italic 对象

您可使用 getElementById() 来访问 <i> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 I 元素的演示</h3>

<p>单击按钮将 <i id="myItalic">此斜体文本</i> 的颜色设置为红色。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.getElementById("myItalic");

x.style.color = "red";

}

</script>

</body>

</html>

创建 Italic 对象

您可使用 document.createElement() 方法来创建 <i> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以创建带有一些文本的 I 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("I");

var t = document.createTextNode("Some italic text");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

# 第72章 HTML DOM IFrame 对象

IFrame 对象

IFrame 对象代表 HTML <iframe> 元素。

访问 IFrame 对象

您可使用 getElementById() 来访问 <iframe> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h3>如何访问 IFRAME 元素的演示</h3>

<iframe id="myFrame" src="/index.html"></iframe>

<p>单击按钮以获取 iframe 的 URL。</p>

<button onclick="myFunction()">试一试</button>

<p id="demo"></p>

<script>

function myFunction() {

var x = document.getElementById("myFrame").src;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

提示：您还可以使用 window.frames 属性访问 <iframe> 元素。

创建 IFrame 对象

您可使用 document.createElement() 方法来创建 <iframe> 元素：

实例

<!DOCTYPE html>

<html>

<body>

<p>单击该按钮以在框架中创建带有 w3school.com.cn 的 IFRAME 元素。</p>

<button onclick="myFunction()">试一试</button>

<script>

function myFunction() {

var x = document.createElement("IFRAME");

x.setAttribute("src", "https://www.w3school.com.cn");

document.body.appendChild(x);

}

</script>

</body>

</html>

IFrame 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | HTML5 中不支持。请改用 style.cssFloat。 |
| contentDocument | 返回 iframe 生成的 document 对象。 |
| frameBorder | HTML5 中不支持。请改用 style.border。  设置或返回 iframe 中 frameborder 属性的值。 |
| height | 设置或返回 iframe 中 height 属性的值。 |
| longDesc | Not supported in HTML5.  设置或返回 iframe 中 longdesc 属性的值。 |
| marginHeight | HTML5 中不支持。请改用 style.margin。  设置或返回 iframe 中 marginheight 属性的值。 |
| marginWidth | HTML5 中不支持。请改用 style.margin。  设置或返回 iframe 中 marginwidth 属性的值。 |
| name | 设置或返回 iframe 中 name 属性的值。 |
| sandbox | 设置或返回 iframe 中 sandbox 属性的值。 |
| scrolling | Not supported in HTML5.  设置或返回 iframe 中 scrolling 属性的值。 |
| seamless | 设置或返回 iframe 是否应该看起来像是包含文档的一部分（无边框或滚动条）。 |
| src | 设置或返回 iframe 中 src 属性的值。 |
| srcdoc | 设置或返回 iframe 中 srcdoc 属性的值。 |
| width | 设置或返回 iframe 中 width 属性的值。 |
| contentWindow | 返回 iframe 生成的 window 对象。 |

# 第73章 HTML DOM Image 对象

Image 对象

Image 对象代表嵌入的图像。

<img> 标签每出现一次，一个 Image 对象就会被创建。

Image 对象的属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| align | 设置或返回与内联内容的对齐方式。 |
| alt | 设置或返回无法显示图像时的替代文本。 |
| border | 设置或返回图像周围的边框。 |
| complete | 返回浏览器是否已完成对图像的加载。 |
| height | 设置或返回图像的高度。 |
| hspace | 设置或返回图像左侧和右侧的空白。 |
| id | 设置或返回图像的 id。 |
| isMap | 返回图像是否是服务器端的图像映射。 |
| longDesc | 设置或返回指向包含图像描述的文档的 URL。 |
| lowsrc | 设置或返回指向图像的低分辨率版本的 URL。 |
| name | 设置或返回图像的名称。 |
| src | 设置或返回图像的 URL。 |
| useMap | 设置或返回客户端图像映射的 usemap 属性的值。 |
| vspace | 设置或返回图像的顶部和底部的空白。 |
| width | 设置或返回图像的宽度。 |

标准属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| className | 设置或返回元素的 class 属性。 |
| title | 设置或返回元素的 title。 |

Image 对象的事件句柄

|  |  |
| --- | --- |
| 事件句柄 | 描述 |
| onabort | 当用户放弃图像的装载时调用的事件句柄。 |
| onerror | 在装载图像的过程中发生错误时调用的事件句柄。 |
| onload | 当图像装载完毕时调用的事件句柄。 |

# 第74章 HTML DOM Ins 对象

Ins 对象

Ins 对象表示 HTML <ins> 元素。

访问 Ins 对象

您可以通过使用 getElementById() 来访问 <ins> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何访问 Ins 元素</h3>

<p><ins id="myIns" cite="why\_inserted.htm">这是一段插入的文本。</ins></p>

<p>点击按钮来获得页面的 URL，该页面解释了文本被删除的原因。</p>

<p id="demo"></p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.getElementById("myIns").cite;

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

创建 Ins 对象

您可以通过使用 document.createElement() 方法来创建 <ins> 元素：

<!DOCTYPE html>

<html>

<body>

<h3>演示如何创建 Ins 元素</h3>

<p>点击按钮来创建 INS 元素。</p>

<button onclick="myFunction()">试一下</button>

<script>

function myFunction()

{

var x = document.createElement("INS");

var t = document.createTextNode("被插入的文本");

x.appendChild(t);

document.body.appendChild(x);

}

</script>

</body>

</html>

Ins 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| cite | 设置或返回被插入文本的 cite 属性的值。 |
| dateTime | 设置或返回被插入文本的 datetime 属性的值。 |