# 一、JS基础

## 1.1 JS概述

JS是适应动态网页制作的需要而诞生的一种编程语言。是由Netscape公司开发的嵌入到HTML文件中的**基于对象**和**事件驱动**的脚本语言。JS的出现使得网页和用户之间实现了一种实时性的、动态的、交互性的关系。

现在JavaScript、JScript和ECMAScript通称为JavaScript。

标准化后的JS包含三个组成部分：

ECMAScript：定义了基本的语法和基本对象。

DOM（Document Object Model）：文档对象模型，是HTML和XML文档的应用程序编程接口。浏览器中的DOM把整个网页规划成由节点层级构成的文档。用DOM API可以轻松地删除、添加和替换节点。

BOM（Browser Object Model）：浏览器对象模型，描述了与浏览器窗口进行访问和操作的方法和接口。

与Java的关系：

1. JS由浏览器解释执行，Java程序则是编译执行。
2. JS是一种基于对象的脚本语言，Java则是一种面向对象的编程语言。
3. JS是弱类型语言，Java是强类型语言。

JS的特点：

1. 解释性

JS是一种解释性的程序语言，源代码不需要经过编译就可以直接在浏览器中运行。

1. 动态性

JS是一种基于事件驱动的脚本语言，不需要经过Web服务器就可以对用户的输入直接作出响应。

1. 跨平台性

JS依赖于浏览器本身，与操作环境无关。

1. 安全性

JS不予许访问本地的硬盘，不能将数据存入服务器，不允许对网络文档进行修改和删除

1. 基于对象

JS是一种基于对象的语言。

代码执行顺序：**非事件处理代码**在HTML文档内容载入后，按JS在文档中出现的顺序，从上往下依次执行。**事件处理代码**则在HTML文件内容载入完成，并且所有非事件处理代码执行完成后，才根据触发的时间执行对应的事件处理代码。

JS**区分大小写**，语句结尾处的分号是可选的，但是为了使语句不出现歧义，最好带上。

**直接量类型**：整型、浮点型、布尔型、字符型直接量和空值。

## 1.2 变量

JS变量的声明需要使用关键字var。

声明的语法：

方法一：var 变量名；

方法二：var 变量名1，变量名2，…..，变量名n；

方法三：var 变量名1=值1，变量名2=值2，…….，变量名n=值n；

说明：

1. **变量的具体数据类型根据所赋的值的数据类型来确定**。
2. 使用var可以一次声明一个变量，也可以一次声明多个变量。
3. 声明变量时可以不赋值，此时其值默认为undefined。
4. 可以使用var多次声明同一个变量。
5. 变量也可以不事先使用var作声明而直接使用，一般不建议这么使用。

全局变量声明在所有函数之外，作用于整个脚本代码；局部变量是在函数体内使用var声明的变量或者是函数的形参，只在函数内有效。

**所有没有使用var声明的变量不管在哪里使用都属于全局变量。**

## 1.3 数据类型

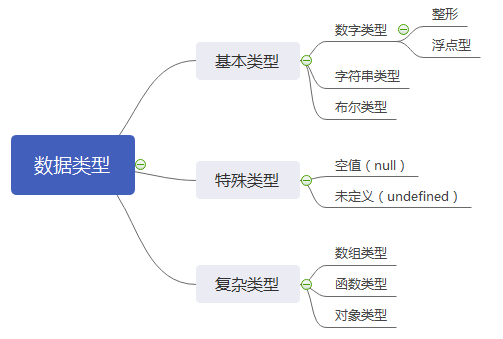


图1.1 数据类型分支图

JS中并不区别整数值和浮点数值，都是由浮点数值表示的。

JS能处理十进制、十六进制（0x或0X开头）、八进制（0开头）的数，但是有些JS不支持八进制，随意最好不用。也可以用指数法表示，实数后面跟随字母e或者E然后再跟数字。

字符串中，由单引号定界的字符串可以含有双引号，由双引号定界的字符串中也可以含有单引号。

表1.1 JS常用的转义字符

|  |  |  |  |
| --- | --- | --- | --- |
| 转义字符 | 描述 | 转义字符 | 描述 |
| \n | 换行符 | \r | 回车符 |
| \t | 水平制表符 | \\ | 反斜杠符 |
| \b | 退格符 | \v | 垂直制表符 |
| \f | 换页符 | \0ddd | 八进制数 |
| \’ | 单引号 | \xnn | 十六进制数 |
| \” | 双引号 | \uhhhh | 4位十六进制数指定的Unicode字符 |

表1.2 处理字符串内容的方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| charAt(位置) | 返回String对象指定位置处的字符 |
| indexOf(查找的子串[,index]) | 返回从位置index之后首次出现查找的子串的位置。如果index省略，则从字符串第一个字符开始查找。如果没有找到要查找的子串，返回-1 |
| lastIndexOf(查找的子串[,index]) | 返回要查找的子串在String对象中最后一次出现的位置。查找是在字符串的index位置从后往前查找，如果省略index参数，则从字符串的最后一个字符开始查找。如果没有找到要查找的子串，返回-1 |
| match(正则表达式) | 在一个字符串中寻找与正则表达式匹配的子串。如果找不到匹配的子串，将返回null |
| replace(正则表达式，新字符串) | 使用新字符串替换匹配正则表达式的字符串后作为新字符串返回 |
| search(正则表达式) | 搜索与参数指定的正则表达式的匹配。如果找不到，返回-1 |
| split(正则表达式) | 根据参数指定的正则表达式将字符串分割成字符串数组 |
| slice(索引值i[,索引值j]) | 提取并返回字符串的索引值i到索引值j-1之间的子串。如果省略索引值j，则返回字符串从i位置到结尾的所有子串。i和j可以是负数 |
| substring(索引值i[,索引值j]) | 提取并返回字符串的索引值i到索引值j-1之间的子串。如果省略索引值j，则返回字符串从i位置到结尾的所有子串。i和j不可以是负数 |
| toLowerCase() | 将字符串中的字母全部转换为小写后作为新字符串返回 |
| toUpperCase() | 将字符串中的字母全部转换为大写后作为新字符串返回 |
| toString() | 返回字符串对象的原始字符串值。这是针对字符串对象的方法 |
| valueOf() | 返回字符串对象的原始字符串值。这是针对字符串对象的方法 |

表1.3 处理字符串显示的常用方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| fontcolor(颜色) | 使用参数所指定的颜色设置字符串的字体颜色 |
| fontsize(大小) | 使用参数所指定的大小设置字符串的字体大小。参数取值为1-7，默认值为3.参数可取正、负值，他们都是相对3来说的。 |
| bold() | 设置字符串加粗显示 |
| italics() | 设置字符串的字体格式为斜体 |
| big() | 设置字符串的字体为大字体 |
| small() | 设置字符串的字体为小字体 |
| strike() | 设置字符串显示删除线 |
| sub() | 设置字符串以下标显示 |
| sup() | 设置字符串以上标显示 |

注：当字符串调用方法时，JS就会将字符串通过调用new String(字符串)的方法将字符串转换为对象，所以其实是字符串对象在调用方法。

布尔类型也可以进行算术运算，此时true将转换为1，false转换为0。

null是JS的关键字，表示没有对象，用于定义空的或不存在的引用，是一个对象类型。null参与算术运算时其值会自动转换为0。

undefined是全局对象的一个特殊属性，表示一个未声明的变量，或已声明但没有赋值的变量。

## 1.4 表达式和运算符

===（严格等于）：左右两边操作数的值相等且数据类型相同时返回true

！===（严格不等于）：左右两边操作数的值不相等或者数据类型不相同时返回true

其它关系运算符在使用时，字符串数据都会在进行关系比较前转换为数字类型。

JS中的假值：false、null、undefined、0、-0、NaN和” ”。

逻辑与和或表达式的值既可以是布尔值，也可以是非布尔值。逻辑非运算的结果总是布尔值。

其它的部分和C语言一样。

this运算符：表示当前对象。

new运算符：用于创建对象。

## 1.5 语句

for….in语句：

语法：for(变量 in 对象){循环体；}

**说明**：对于数组，变量的值等于所遍历到的元素所对应的索引；对于对象，变量的值等于所遍历到的属性名。

其余和C语言一样。

注释：单行注释用//，多行注释用/\*注释内容\*/。

# 二、在网页中嵌入脚本

有三种方式：在HTML标签的事件属性中直接添加脚本代码、使用<script>标签在网页中直接插入脚本代码、使用<script>标签链接外部脚本文件。

1. 直接添加脚本

例：<input type=”button” onClick=”Javascript:alert(‘欢迎’);” value=”点击试试”/>

1. 插入脚本

例：

<head>

…

<script type=”text/javascript”>

function showMsg(){

alert(“欢迎”);

}

</script>

</head>

<body>

…

<input type=”button” onClick=”showMsg()” value=”点击试试”/>

…

</body>

1. 链接脚本文件

例：<script type=”text/javascript” src=”脚本文件”></script>

注：虽然此时<script>是个空标签，但是不可以用缩写形式。

# 三、脚本函数

函数分为内置函数和用户自定义函数。

自定义函数需要使用关键字function。定义要求同C语言，函数的虚参在内存中没有分配空间。

函数调用有三种方式：（调用函数时若实参数少于虚参数则没有实参的值将指定为undefined）

1. 一般调用

例：

…

<body>

<script>

document.write(函数名())

..

</script>

…

</body>

1. 在事件响应中调用

见上一章第二种情况的调用

1. 在链接中调用函数

格式：href=”javascript:函数名”。

表3.1 常用内置函数

|  |  |  |
| --- | --- | --- |
| 函数 | 说明 | 添加说明 |
| parseInt() | 将字符型参数转化为整型 | 遇到不合法字符立马停止解析，返回已经解析的值 |
| parseFloat() | 将字符型参数转化为浮点型 |
| isFinite() | 判断一个数值是否是有限的 |
| isNaN() | 判断参数是否为非数字 |
| encodeURL() | 将字符串转化为有效的URL |
| decodeURL() | 对encodeURL编码的文本进行解码 |

# 四、事件处理

JS的一个基本特征就是事件驱动。所谓**事件驱动**就是当用户执行了某种操作后会因此而引发一系列程序的执行。

**事件处理**是指程序对事件作出的响应。事件处理中涉及的程序称为**事件处理程序**。

在web页面中产生事件的对象称为**事件目标**。

事件处理涉及的工作包括事件处理程序的**定义**及其**注册**和**调用**。

**事件对象**是指与特定事件相关且包含该事件详细信息的对象。

表4.1 JS常用事件

|  |  |  |
| --- | --- | --- |
| 事件 | | 描述 |
| 鼠标事件 | click | 用户单击鼠标时触发 |
| dblclick | 用户双击鼠标时触发 |
| mousedown | 用户按下鼠标时触发 |
| mouseup | 用户按下鼠标后松开时触发 |
| mouseover | 用户将鼠标移动到某对象范围的上方时触发 |
| mousemove | 用户移动鼠标时触发 |
| mouseout | 当用户鼠标离开某对象范围时触发 |
| 键盘事件 | keypress | 当用户键盘上的某个字符键被按下并且释放时触发 |
| keydown | 当用户键盘上的某个按键被按下时触发 |
| keyup | 当用户键盘上的某个按键被按下后松开时触发 |
| 窗口事件 | abort | 图形尚未完全加载前用户就单击了一个超链接或单击停止按钮时触发 |
| error | 加载文件或图像发生错误时触发 |
| load | 页面内容加载完成时触发 |
| resize | 浏览器的窗口大小被改变时触发 |
| unload | 当前页面关闭或退出时触发 |
| 表单事件 | blur | 当前表单失去焦点时触发 |
| click | 用户单击复选框、单选按钮或button、submit和reset按钮时触发 |
| change | 表单元素的内容发生改变并且元素失去焦点时触发 |
| focus | 表单元素获得焦点时触发 |
| reset | 用户单击表单上的reset按钮时触发 |
| select | 用户选择了一个input或textarea表单域中的文本时触发 |
| submit | 用户单击submit按钮提交表单时触发 |

## 4.1注册

注册事件处理程序有三种方式：

1. 设置HTML标签的事件属性为事件处理程序；
2. 设置事件目标的事件属性为事件处理函数；
3. 使用事件目标调用addEventListener()方法。

1和2两种注册方式中的事件属性名的组成形式是：”on” + 事件明，例onclick、onfocus。

补充：事件属性中的脚本代码不能包含函数声明，但可以是函数调用或一系列使用分号分隔的脚本代码。根据Web标准的JS和HTML应分离的原则，通过设置HTML标签的事件属性的方式注册事件处理程序并不是很好，应尽量避免使用。

## 4.2 调用

this关键字的使用：this具体指向的对象取决于this在程序中的位置以及程序的执行方式。出现在事件处理函数的**实参中**，则指向事件目标。若出现在事件处理**函数体中**，当事件处理函数的绑定是通过HTML属性时，则指向window对象；如果事件处理函数的绑定是通过事件目标的属性时，则指向事件目标。

调用事件处理程序时，JS会把事件对象event作为参数传给事件处理程序。

表4.2 event对象的常用属性和方法

|  |  |
| --- | --- |
| 属性/方法 | 说明 |
| altKey | 用于判断键盘事件发生时“Alt”键是否被按下 |
| button | 用于判断鼠标事件发生时哪个鼠标键被点击了。在遵循W3C标准的浏览器中，鼠标左、中、右键分别用0、1和2表示，其它浏览器中中用1、4、2表示。 |
| clientX | 用于获取鼠标事件发生时相对于**可视窗口**左上角的鼠标指针的水平坐标 |
| clientY | 用于获取鼠标事件发生时相对于**可视窗口**左上角的鼠标指针的垂直坐标 |
| ctrlKey | 用于判断键盘事件发生时“Ctrl”键是否被按下 |
| relatedTarget | 用于获取鼠标事件发生时与事件目标相关的节点 |
| screenX | 用于获取鼠标事件发生时相对于**文档窗口**左上角的鼠标指针的水平坐标 |
| screenY | 用于获取鼠标事件发生时相对于**文档窗口**左上角的鼠标指针的垂直坐标 |
| shiftKey | 用于判断键盘事件发生时“Shift”键是否被按下 |
| offsetX | 用于获取鼠标事件发生时相对于**事件目标**左上角的鼠标指针的水平偏移，在Chrome、Opera和Safari浏览器中，左上角为外边框的位置，在Firefox和IE中，左上角为内边框位置 |
| offsetY | 用于获取鼠标事件发生时相对于**事件目标**左上角的鼠标指针的垂直偏移 |
| srcElement | 用于在IE8及以下版本的IE浏览器中获取事件目标 |
| target | 在W3C标准浏览器中获取事件目标 |
| type | 获取事件类型 |
| return Value | 取值为true或false。用于在IE8及以下版本的IE浏览器中决定是否执行与事件关联的默认动作。当值为false时，不执行默认动作。 |
| preventDefault() | 在W3C标准的浏览器中，通知浏览器取消事件的默认操作 |

对于使用对象属性或HTML标签属性注册的事件处理程序也可以通过它返回false值来取消事件的浏览器默认操作。

# 五、正则表达式模式匹配

## 5.1 正则表达式定义

正则表达式总是以斜杠（/）开头和结尾。**定义语法**为：/字符串序列/[正则表达式修饰符]

说明：斜杠中间为正则表达式，修饰符用于描述匹配方式。

常用转义字符：换页符（\f）、换行符（\n）、回车符（\r）、制表符（\t）、垂直制表符（\v）。

常用元字符：和其它语言的一样。

修饰符：不区分大小写（i）、全局匹配（返回所有的匹配）（g）、多行匹配（m）

## 5.2 RegExp对象

创建方式：

1. 使用定义正则表达式方式创建RegExp对象

例： var pattern = /java\d{3}/ig（修饰符可以组合使用）

注：同一个正则表达式，每一次运行时都会创建新的RegExp对象。

1. 使用RegExp()构造函数创建RegExp对象

格式：RegExp(“正则表达式主体部分”，”修饰符（可选）”)

注：在这种方法中，主体部分的所有转义字符前面需要再添加“\”作为其前缀。

表5.1 RegExp对象常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| $1~$9 | 分别存储对应正则表达式中圆括号表达式所匹配的子字符串 |
| global | 用于判断正则表达式是否带有修饰符g，有返回true，无返回false |
| ignoreCase | 用于判断正则表达式是否带有修饰符i，有返回true，无返回false |
| multiline | 用于判断正则表达式是否带有修饰符m，有返回true，无返回false |
| lastIndex | 当正则表达式带有修饰符g时，该属性存储继续匹配的起始位置 |
| source | 表示正则表达式文本 |

RegExp对象常用的方法有exec(string)和test(string)。

**exec(string)**：对参数指定的字符串执行一个正则表达式匹配。没有找到匹配返回null，找到则返回一个**数组**，数组的第一个元素是第一次匹配的字符串，第二个元素是第二次匹配的字符串，以此类推。**数组的index属性包含了发生匹配的字符开始位置**。

**test(string)**：对参数指定的字符串执行一个正则表达式匹配，有则返回true，无则返回false。

## 5.3 用于匹配的String方法

除了使用RegExp对象进行正则表达式匹配为还可以用字符串的方法进行匹配。

具有匹配功能的方法：

match(pattern)：在一个字符串中寻找参数指定的正则表达式的匹配。没有匹配返回null，有匹配返回一个由匹配结果组成的数组。**如果正则表达没有设置修饰符g，则只检索第一个匹配**。**此时返回一个数组，第一个元素就是匹配的字符串，其余的元素则是正则表达式中用圆括号括起来的子表达式**。

replace(pattern,newStr)：将匹配第一个参数指定的正则表达式pattern的子串替换为第二个参数指定的子串。正则表达式设置修饰符g则符合条件的全部替换，否则只替换第一个。

注：若果replace()的第一个参数是字符串而不是正则表达式，则replace()将直接搜索这个字符串。

search()：搜索参数指定的正则表达式的匹配。没找到则返回-1，找到则返回第一个匹配的子串的起始位置。

注：该方法不支持全局检索。

split(pattern)：将调用它的字符串拆分成一个子串组成的数组。

# 六、JS内置对象

JS对象指既可以保存一组不同类型的数据（属性），又可以包含有关处理这些数据的函数（方法）的特殊数据类型。

常用的JS内置对象有：Array对象、String对象、Date对象、Math对象、正则表达式RegExp对象。

## Array对象：

创建方法：

方式一：var 数组对象名=new Array();

方式二：var 数组对象名=new Array(数组元素个数);

方式三：var 数组对象名=new Array(元素1,元素2,……,元素n);

方式四：var 数组对象名=[元素（可选）];

注：与其他语言不同的是，JS中使用数组名可以获取整个数组的值。

常用属性：length，获取数组长度。

表6.1 数组对象常用方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| concat(数组1,…,数组n) | 用于将一个或多个数组合并到数组对象中。参数可以是具体的值，也可以是数组对象。 |
| join(分隔符) | 将数组内各个元素以分隔符连接成一个字符串，参数省略时默认为“逗号” |
| push(元素1,,…,元素n) | 向数组的末尾添加一个或多个元素，并返回新的长度。至少有一个参数。 |
| reverse() | 颠倒数组中元素的顺序 |
| slice(start,end) | 返回包含从数组对象中截取的第start—end-1之间的元素的数组。end省略时表示直到最后的元素。 |
| sort() | 按字典顺序对数组元素重新排序 |
| toString() | 把数组转换为字符串并返回。效果等于不带参数的join() |

## String对象：

字符串变量具有和字符串对象一样的作用，同样可以存储和处理文本。字符串对象和字符串具有相同的属性和方法。对字符串进行处理时，JS首先将其转换为一个伪对象，因而可以使用它访问属性和方法。因为创建字符串对象有可能拖慢执行速度并可能产生其他副作用，所以在实际项目中尽量不要对字符串创建对象。

常用属性：length

常用方法：见表1.2和表1.3

## Math对象：

常用属性：E（自然对数的底）、PI

常用的方法：

表6.2 Math对象常用的方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| abs(x) | 取绝对值 |
| acos(x) | 反余弦，x必须是一个数字 |
| asin(x) | 反正弦，x必须是-1.0~1.0之间的数 |
| atan(x) | 反正切，x必须是-1.0~1.0之间的数 |
| ceil(x) | 向上取整 |
| exp(x) | e指数 |
| floor(x) | 向下取整 |
| log(x) | 取自然对数 |
| max(x1,x2) | 最大值 |
| min(x1,x2) | 最小值 |
| pow(x1,x2) | x1的x2次方 |
| random() | 0~1.0之间的随机数 |
| round(x) | 四舍五入 |
| sqrt(x) | 平方根 |
| sin(x) | 正弦，x以弧度表示，下同 |
| cos(s) | 余弦 |
| tan(x) | 正切 |

## Date对象：

创建：

var date = new Date([日期参数])

1. 省略不写：获取系统当前日期和时间
2. 日期字符串：“月 日，公元年 时：分：秒”或“月/日/公元年 时：分：秒”或“月 日 ，公元年”或“月/日/公元年”
3. 一律以数值表示：“公元年，月，日，时，分，秒”或“公元年，月，日”

常用方法：getDate（获取对象的日），getDay（获取星期，取值0~6，星期日对应0），getMonth（获取月份，取值0~11，一月对应0），getFullYear（获取年），getHours（获取时），getMinutes（获取分），getSeconds（获取秒），getTime（返回自1970/1/1 00:00:00以来的毫秒数），Date.parse(日期字符串)（作用同前一个，静态方法）

# 七、BOM对象

BOM提供了独立于内容的、可以与浏览器窗口进行交互对象结构。主要用于管理窗口与窗口之间的通讯。

JS中有两种对象模型：BOM（Browser Object Model）和DOM（Document Object Model）。

BOM包含的对象：window、screen、navigator、history、location、frames、document。

## window对象

如果网页文档中包含frame或iframe标签，则浏览器会为每个框架创建一个window对象，并将它们存放在frames集合中。

**window对象的所有属性和方法都是全局性的。而且JS中的所有全局变量都是window对象的属性；所有全局函数都是window对象的方法。**

表7.1 window对象的常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| defaultStatus | 设置或返回窗口状态栏，主要针对IE，FF和谷歌没有状态栏 |
| status | 设置窗口状态栏的信息，主要针对同上 |
| document | 引用document对象 |
| history | 引用history对象 |
| location | 引用location对象 |
| navigator | 引用navigator对象 |
| screen | 引用screen对象 |
| name | 设置或返回窗口的名称 |
| opener | 返回创建当前窗口的窗口 |
| self | 返回当前窗口，等价于window对象 |
| top | 返回最顶层窗口 |
| parent | 返回当前窗口的父窗口 |

表7.2 window对象的常用方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| back() | 回到历史记录中的上一网页，相当于浏览器中的‘后退’按钮 |
| forward() | 加载历史清单中的下一网址，相当于浏览器中的‘前进’按钮 |
| blur() | 使窗口失去焦点 |
| focus() | 使窗口获得焦点 |
| close() | 关闭窗口 |
| home() | 进入客户端在浏览器上设置的主页 |
| print() | 打印当前窗口的内容，相当于浏览器中的打印功能 |
| alert(警告信息字符) | 显示警告对话框，用以提示用户注意某些事项 |
| confirm(确认信息字符串) | 显示确认对话框，有“确认”和“取消”两个按钮，单击确认按钮返回true，单击取消按钮返回false |
| prompt(提示字符串，[默认值]) | 显示提示输入信息对话框，返回用户输入信息 |
| open(URL，窗口名称，[窗口规格]) | 打开新窗口 |
| setTimeout(执行程序，毫秒) | 在指定的毫秒数后调用函数或计算表达式 |
| setInterval(执行程序，毫秒) | 按照指定的周期（以毫秒计）来调用函数或计算表达式 |
| clearTimeout(定时器对象) | 取消setTimeout设置的定时器 |
| clearInterval(定时器对象) | 取消setInterval设置的定时器 |

window常用的别名：

opener：表示打开当前窗口的父窗口。

parent：表示当前窗口的上一级窗口。

top：表示最顶层的窗口。

self：表示当前活动的窗口。

window对象的应用：

1. 创建警告对话框。
2. 创建确认对话框。
3. 创建信息提示对话框。
4. 打开指定窗口。

基本语法：open(URL,窗口名称[,规格参数])

**URL**可以是完整的网址，表示在指定窗口中打开该网址页面；也可以是相对路径表示的文件名称，表示在指定窗口中打开该文件；此外，也可以是一个空字符串。

**窗口名称**：可以是任意符合规范的名字，也可以使用\_blank、\_self、\_parent和\_top这些关键字作为窗口名称。作用分别是：新开一个窗口显示文档、在当前窗口显示文档、在当前窗口的父窗口显示文档和在顶层窗口中显示文档。名称可以是一个空字符串，相当于\_blank。

**规格参数**：由许多用逗号隔开的参数字符串组成，用于指定新窗口的外观和属性。

表7.3 规格参数

|  |  |  |
| --- | --- | --- |
| 规格参数 | 用法 | 默认值 |
| directories=yes|no|1|0 | 是否显示连接工具栏 | no |
| fullscreen=yes|no|1|0 | 是否以全屏显示 |
| loaction=yes|no|1|0 | 是否显示网址栏 |
| menubar=yes|no|1|0 | 是否显示菜单栏，如果打开窗口的父窗口不显示菜单栏，打开窗口将不显示 |
| resizable=yes|no|1|0 | 是否可以改变窗口尺寸 |
| scrollbars=yes|no|1|0 | 如果设置网页内容超过窗口大小，是否显示滚动条 |
| status=yes|no|1|0 | 是否显示状态栏 |
| titlebar=yes|no|1|0 | 是否显示标题栏 |
| toolbar=yes|no|1|0 | 是否显示工具栏 |
| height=number | 设置窗口的高度 |  |
| width=number | 设置窗口的宽度 |  |
| left=number | 设置窗口左上角相对于显示器左上角的X坐标 | 像素单位 |
| top=number | 设置窗口左上角相对于显示器左上角的Y坐标 |

1. 定时器

## Navigator对象

navigator对象包含有关浏览器的信息，包含的属性描述了正在使用的浏览器。

表7.4 navigator对象的常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| apppCodeName | 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 | 返回浏览器是否启用cookie，启用则返回true，否则返回false |
| 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 |

上述属性中，最常用的是userAgent和cookieEnabled。

navigator对象的方法：

javaEnabled()：规定浏览器是否启用Java。

preference()：用于取得浏览器的偏好设置。

## Location对象

location对象包含了浏览器当前显示的文档的URL信息。

**注意：document对象也有一个location属性，也包含了当前文档的URL信息。**

表7.5 location对象常用属性

|  |  |
| --- | --- |
| 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） |

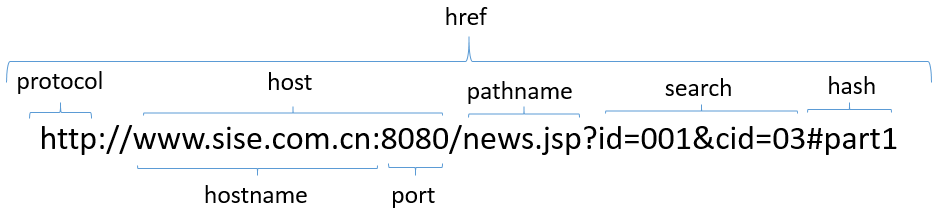


图7.1 URL的结构示意图

location对象方法：

assign()：加载新的文档。

reload()：重新加载当前文档。

replace()：用新的文档替换当前文档，且无须为它创建一个新的历史记录。

## history对象

history对象包含用户访问过的URL。history对象是window对象的一个对象类型的属性。

处于隐私方面的原因，history对象不再允许脚本访问已经访问过的URL。

historty对象属性：length，返回浏览器历史列表中的URL数量。

方法：

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(number)：加载history列表中的某个具体页面。参数number是要访问的URL在history的URL列表中的**相对位置**，可取正数或负数。在当前页面前面的URL的位置为负数（如在前一个页面的位置为-1），反之则为正数。

## screen对象

常用属性：

availHeight：返回显示屏幕的可用高度，单位为像素（下同），不包括任务栏。

availWidth：返回显示屏幕的可用宽度，不包括任务栏。

height：返回显示屏幕的高度。

width：返回显示屏幕的宽度。

colorDepth：返回当前颜色设置所用的位数 -1：黑色；8:256色；16：增强色；24/32：真彩色。

# 八、DOM对象

DOM提供了一组独立于语言和平台的应用程序编程接口，描述了如何访问和操纵XML和HTML文档的结构和内容。

在DOM中，一个HTML文档是一个树状结构，其中的每一块内容称为一个节点。

经常操作的节点的类型主要有document节点、元素节点（包括根元素节点）、属性节点和文本节点。

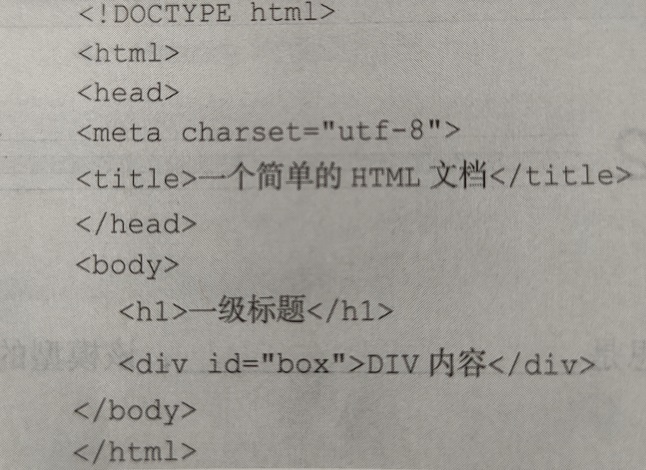


图8.1 简单的HTML文档

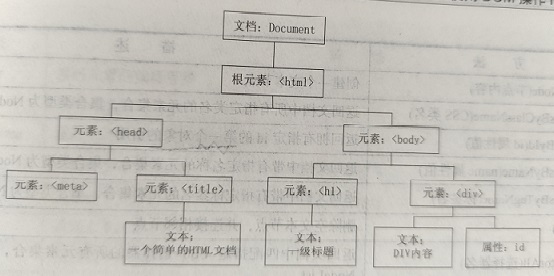


图8.2 8.1中文档的DOM模型树结构

## document对象

document是节点树中的顶层节点，代表的是整个HTML文档，它是操作文档其它内容的入口。

表8.1 document常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| anchors | 返回文档中的所有书签锚点，通过数组下标引用每一个锚点 |
| body | 代表body元素 |
| forms | 返回文档中的所有表单，通过数组下标引用每一个表单 |
| images | 返回文档中的所有图片，通过数组下标引用每一个图片 |
| lastModified | 用于获取文档的最后修改时间和日期 |
| links | 返回文档的所有链接，通过数组下标引用每一个链接 |
| location | 用于跳转到指定的URL |
| title | 用于设置或获取文档标题 |
| URL | 返回当前文档完整的URL |

表8.2 document对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| createAttribute(节点名) | 创建一个属性节点 |
| createElement(节点名) | 创建一个元素节点 |
| createTextNode(节点内容) | 创建一个文本节点 |
| getElementsByClassName(CSS类名) | 返回文档中所有指定类名的元素集合 |
| getElementById(id属性值) | 返回拥有指定id的**第一个对象**的引用 |
| getElementsByName(name属性值) | 返回文档中带有指定名称的元素集合 |
| getElementByTagName(标签名) | 返回文档中带有指定标签名的元素集合 |
| normalize() | 删除空文本节点，并连接相邻节点 |
| querySelectorAll(选择器名) | 返回文档中匹配指定CSS选择器的所有元素集合 |
| write(字符串) | 向文档写指定的字符串 |

## 元素对象

一个元素节点就是一个元素对象，代表一个HTML元素。

表8.3 元素对象常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| attributes | 返回元素的属性列表，列表类型为NamedNodeMap |
| childNodes | 返回元素的子节点列表，列表类型为NodeList |
| className | 设置或返回元素的class属性 |
| clientHeight | 在页面上返回内容的可视高度，包括内边距，但不包括边框、外边框和滚动条 |
| clientWidth | 在页面上返回内容的可视宽度，包括同上 |
| contentEditable | 设置或返回元素的内容是否可编辑 |
| firstChild | 返回元素的第一个子节点 |
| id | 设置或返回元素的id |
| innerHTML | 设置或返回元素的内容 |
| lastChid | 返回元素的最后一个子节点 |
| nextSibling | 返回该元素紧跟着的下一个兄弟节点 |
| nodeName | 返回元素的标签名（大写） |
| nodeType | 返回元素的节点类型 |
| offsetHeight | 返回元素的高度，包括边框和内边距，但不包括外边距 |
| offsetWidth | 返回元素的宽度，包括同上 |
| offsetLeft | 返回元素相对于文档或偏移容器（当元素被定位时）的水平偏移位置 |
| offsetTop | 返回元素相对于文档或偏移容器（当元素被定位时）的垂直偏移位置 |
| offsetParent | 返回元素的偏移容器 |
| parentNode | 返回元素的父节点 |
| previousSibling | 返回该元素紧跟着的前一个兄弟节点 |
| scrollHeight | 返回整个元素的高度，包括带滚动条的隐藏的地方 |
| scrollWidth | 返回整个元素的宽度，包括带滚动条的隐藏的地方 |
| scrollLeft | 返回水平滚动条的水平位置 |
| scrollTop | 返回水平滚动条的垂直位置 |
| style | 设置或返回元素的样式属性 |
| tagName | 返回元素的标签名（大写），作用和nodeName完全一样 |
| title | 设置或返回元素的title属性 |

表8.4 元素对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| appendChild(子节点) | 在元素的子节点列表后面添加一个新的子节点 |
| focus() | 设置元素获取焦点 |
| getAttribute(属性名) | 返回元素指定属性的值 |
| getElementsByTagName(标签名) | 返回元素的指定标签名的所有子节点列表 |
| getElemensByClassName(CSS类名) | 返回元素的指定类名的所有子节点列表 |
| hasAttributes() | 判断元素是否存在属性，存在返回true，反之false |
| hasChildNodes() | 判断元素是否存在子节点，是返回true，否false |
| hasfocus() | 判断元素是否获得焦点，是返回true，否false |
| insertBefore(节点1，节点2) | 在元素的指定子节点（节点2）前插入一个新的子节点 |
| querySelectorAll(选择器名) | 返回文档中匹配指定CSS选择器的所有元素集合 |
| removeAttribute(属性名) | 删除元素的指定属性 |
| removeChild(子节点) | 删除元素的指定子节点 |
| replaceChild(新节点，旧节点) | 使用新的节点替换元素指定的子节点 |
| setAttribute(属性名，属性值) | 设置元素指定的属性值 |

## 属性对象

一个属性节点就是一个属性对象，代表HTML元素的一个属性。

元素的所有属性存放在表示无序的集合NamedNodeMap中，可以通过名称或索引来访问。

属性/方法：

name：使用属性对象来引用，返回元素属性的名称。

value：使用属性对象来引用，设置或返回元素属性的值。

item()：为NamedNodeMap对象的方法，返回该集合中指定下标的节点。

length：为NamedNodeMap对象的属性，返回该集合中的节点数。

## 使用DOM访问文档元素

文档元素的获取可采用一下五种方式：

1. 用指定的id属性：调用document.getElementById(id属性值)。
2. 用指定的name属性：调用document.getElemensByName(name属性值)。
3. 用指定的标签名字：调用document|元素对象.getElementsByTagName(标签名)。
4. 用指定的CSS类：调用document|元素对象.getElementsByClassName(类名)。
5. 匹配指定的CSS选择器：调用document|元素对象.querySelectorAll(选择器)。

## 使用DOM创建、插入、修改和删除节点

创建节点：创建元素节点调用document.createElement(“节点名”)；创建文本节点调用document.createTextNode(“节点名”)；创建属性节点调用document.createAttribute(“节点名”)。

插入节点：element.appendChild(子节点)为在元素子节点列表的后面附加子节点。element.insertBefore(节点1，节点2)为在某个子节点前面添加子节点。

修改节点：element.replaceChild(新节点，旧节点)。

删除节点：element.removeChild(子节点)。

## 表单及表单元素对象

表8.5 form对象常用属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| action | 设置或返回表单的action属性 |
| elements | 包含表单中所有表单元素的数组，使用索引引用其中的元素 |
| length | 返回表单中的表单元素数目 |
| 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的数据所显示的Frame或窗口 |
| onreset | 在重置表单元素之前调用事件处理方法 |
| onsubmit | 在提交表单之前调用事件处理方法 |

form对象常用方法：

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。

获取表单的方式：

1. 引用document的forms属性：document.forms[索引值]，索引值从0开始。
2. 直接引用表单的name属性：document.formName。
3. 通过表单的ID：document.getElementById()。
4. 通过表单的name属性：document.getElementsByName()。
5. 通过表单标签：document.getElementsByTagName()。
6. 通过选择器：document.querySelectorAll()。

表单元素对象

表单元素可分为三大类：输入元素（<input>标签）、选择元素（<select>标签）、文本域元素（<textarea>标签）。

**输入元素**：文本框（text）、密码框（password）、隐藏域（hidden）、文件域（file）、单选按钮（radio）、复选框（checkbox）、普通按钮（button）、提交按钮（submit）、重置按钮（reset）。

**选择元素**：多项选择列表或下拉菜单（select）、选项（option）。

表单元素的常用属性：

公共属性：

disabled：设置或返回是否禁用表单元素。注意，hidden元素没有此属性。

id：设置或返回表单元素的id属性。

name：设置或返回表单元素的name属性。注意，option元素没有此属性。

type：对输入元素可设置或返回type属性；对选择和文本域两类元素则只能返回type属性。

value：设置或返回表单元素的value属性。注意，select元素没有此属性。

私有属性：

text和password：

defaultValue：设置或返回文本框或密码框的默认值。

maxLength：设置或返回框中最多可输入的字符数。

readOnly：设置或返回框是否是只读的。

size：设置或返回框的尺寸（长度）。

textarea：

defaultValue：设置或返回文本域元素的默认值。

rows：设置或返回文本域元素的高度。

cols：设置或返回文本域元素的宽度。

radio和checkbox：

checked:设置或返回框的选中状态。

defaultChecked：返回框的默认选中状态。

select：

length：返回选择列表中的选项数目。

multiple：设置或返回是否选择多个项目。

selectedInedx：设置或返回选择列表中被选项的索引号。注意，若允许多重选择，则仅返回第一个被选选项的索引号。

size：设置或返回选择列表中的可见行数。

option：

defaultSelected:返回selected属性的默认值。

selected：：设置或返回selected属性的值。

text：设置或返回某个选项的纯文本值。

常用的事件属性：

公共事件属性：

onblur：当表单元素失去焦点时调用事件处理函数。

onfocus：当表单元素获得焦点时调用事件处理函数。

私有：

text、password、textarea：

onSelect：当选择了一个input或textarea中的文本时调用事件处理函数。

onChange：表单元素的内容发生改变并且元素失去焦点时调用事件处理函数。

radio、checkbox、button、submit、reset：

onClick：单击复选框、单选按钮或button、submit和reset按钮时调用事件处理函数。

常用的方法：

公共方法：

blur()：从表单元素上移开焦点。

focus()：在表单元素上设置焦点。

私有：

text、password：

select()：选取文本框或密码框中的内容。

radio、checkbox、button、submit、reset：

click()：在表单元素上模拟一次鼠标点击。

select：

add()：向选择列表添加一个选项。

remove()：从选择列表中删除一个选项。

获取表单元素的方式：

1. 引用表单对象的elements属性：document.formName.elements[索引值]。
2. 直接引用表单元素的name属性：document.formName.name。
3. 通过表单元素的ID：调用document.getElementById()方法。
4. 通过表单元素的name属性：调用document.getElementsByName()方法。
5. 通过表单标签：调用document.getElementsByTagName()方法。
6. 通过选择器：调用document.querySelectorAll()方法。

# JSON

JavaScript对象表示法（JavaScript Object Notation）。

是轻量级的文本数据交换格式；

独立于语言；

具有自我描述性，更易理解。

转换为JS对象：使用内建的 eval() 函数。

和XML的相似和差异：

相似：

JSON 是纯文本；

JSON 具有“自我描述性”（人类可读）；

JSON 具有层级结构（值中存在值）；

JSON 可通过 JavaScript 进行解析；

JSON 数据可使用 AJAX 进行传输。

差异：

没有结束标签；

更短；

读写的速度更快；

能够使用内建的 JavaScript eval() 方法进行解析；

使用数组；

不使用保留字。

XML 必须使用 XML 解析器进行解析。而 JSON 可通过标准的 JavaScript 函数进行解析。

**语法**：

JSON 语法是 JavaScript 对象表示法语法的子集。

* 数据在名称/值对中
* 数据由逗号分隔
* 花括号保存对象
* 方括号保存数组

JSON 数据的书写格式是：名称/值对。

名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值：

"firstName" : "John"

JSON**值的类型**：

* 数字（整数或浮点数）
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在方括号中）
* 对象（在花括号中）
* null

JSON 对象在花括号中书写：

对象可以包含多个名称/值对：{ "firstName":"John" , "lastName":"Doe" }

JSON 文件的**文件类型**是 ".json"

JSON 文本的MIME类型是 "application/json"

eval()使用时需要把文本包含在括号中避免语法错误，例如：eval(“(” + text + “)”)。

eval() 函数可编译并执行任何 JavaScript 代码。这隐藏了一个潜在的安全问题。使用 JSON 解析器将 JSON 转换为 JavaScript 对象是更安全的做法。JSON 解析器只能识别 JSON 文本，而不会编译脚本。

语法：JSON.parse(JSON数据)。返回JS数组。

JSON.stringify() 把 JavaScript 对象转换为字符串。

也可以对 JavaScript 数组进行字符串化；

会把任何日期对象转换为字符串；

从 JS 对象删除任何函数，包括键和值；