**目录**

[16.4 Math 2](#_Toc1448)

[第17章 浏览器对象模型BOM 4](#_Toc30290)

[17.1 window对象 4](#_Toc544)

[17.2 计时器 6](#_Toc32706)

[17.3 Screen对象 10](#_Toc13415)

[第18章 文档对象模型DOM 12](#_Toc7113)

[18.1 DOM的概念 12](#_Toc27620)

[18.2 获取结点对象 13](#_Toc26178)

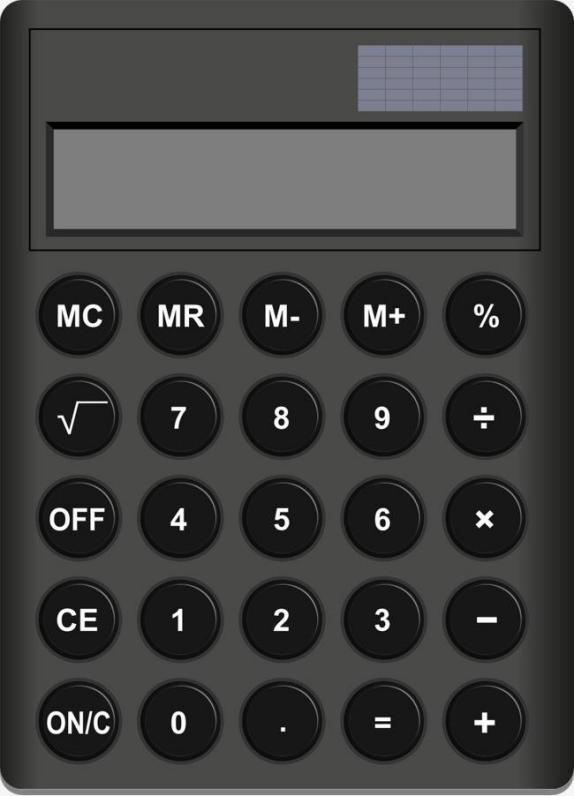
[18.3 结点属性 16](#_Toc32460)

[18.4 结点操作 18](#_Toc1836)

**16.4 Math**

**Math**

**Math对象**提供对数据的**数学计算**。需要注意的是，Math对象是一个**固有**的对象，**无需创建**它，直接把Math作为对象使用就可以调用其所有属性和方法，这是它与其它对象的区别。



Math对象的属性：

|  |  |
| --- | --- |
| **属性** | **说明** |
| E | 返回算术常量e，即自然对数的底数（约等于2.718） |
| LN2 | 返回2的自然对数（约等于0.693） |
| LN10 | 返回10的自然对数（约等于2.302） |
| LOG2E | 返回以2为底e的对数（约等于1.442） |
| LOG10E | 返回以10为底e的对数（约等于0.434） |
| PI | 返回圆周率（约等于3.14159） |
| SQRT1\_2 | 返回2的平方根的倒数（约等于0.707） |
| SQRT2 | 返回2的平方根（约等于1.414） |

Math对象的方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| sin(x) | 返回x的正弦 |
| cos(x) | 返回x的余弦 |
| tan(x) | 返回x的正切 |
| acos(x) | 返回x的反余弦值 |
| asin(x) | 返回x的反正弦值 |
| atan(x) | 返回x的反正切值 |
| **ceil(x)** | **对x进行上取整** |
| **floor(x)** | **对x进行下取整** |
| **abs(x)** | **返回x的绝对数** |
| **exp(x)** | **返回e的x次幂** |
| log(x) | 返回x的自然对数（底为e） |
| **pow(x, y)** | **返回x的y次幂** |
| **max(x, y)** | **返回x和y中的最大值** |
| **min(x, y)** | **返回x和y中的最小值** |
| **round(x)** | **返回x的四舍五入最接近的整数** |
| **sqrt(x)** | **返回x的平方根** |
| **random()** | **返回0-1之间的随机数** |

**第17章 浏览器对象模型BOM**

**17.1 window对象**

**window对象**

**window**对象是**浏览器对象模型BOM(Browser Object Model)**的核心。

window对象方法有：

|  |  |
| --- | --- |
| **方法** | **描述** |
| alert() | 显示带有一段消息和一个确认按钮的警告框 |
| prompt() | 显示可提示用户输入的对话框 |
| confirm() | 显示带有一段消息以及确认按钮和取消按钮的对话框 |
| open() | 打开一个新的浏览器窗口或查找一个已命名的窗口 |
| close() | 关闭浏览器窗口 |
| print() | 打印当前窗口的内容 |
| focus() | 把键盘焦点给与一个窗口 |
| blur() | 把键盘焦点从顶层窗口移开 |
| 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 |
| 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() | 每隔指定的时间执行代码 |
| setTimeout() | 在指定的延迟时间之后来执行代码 |
| clearInterval() | 取消setInterval()的设置 |
| clearTimeout() | 取消setTimeout()的设置 |

**17.2 计时器**

**计时器**

在JS中，可以在设定的时间间隔之后执行代码，而不是在函数被调用后立即执行。

计时器的类型分为2种：

1. **一次性计时器：仅在指定的延迟时间之后触发一次**。

2. **间隔性触发计时器：每隔一定的时间间隔就触发一次**。

计时器的方法有：

|  |  |
| --- | --- |
| **方法** | **描述** |
| **setTimeout()** | **在指定的延迟时间之后来执行代码** |
| **clearTimeout()** | **取消setTimeout()的设置** |
| **setInterval()** | **每隔指定的时间执行代码** |
| **clearInterval()** | **取消setInterval()的设置** |

**setTimeout()**

**setTimeout()**计时器，在**载入后延迟指定时间**后，去执行一次表达式，**仅执行一次**。语法如下：

|  |
| --- |
| setTimeout(代码, 延迟时间); |

其中，第一个参数为要调用的函数或要执行的代码串，第二个参数表示在执行代码前需等待的时间，以**毫秒**为单位（**1s = 1000ms**）。

|  |
| --- |
| **范例：setTimeout()计时器** |
| //网页打开2秒后弹出提示框  setTimeout("alert('Welcome')", 2000); |

clearTimeout()和setTimeout()一起使用，用于**停止计时器**。语法如下：

|  |
| --- |
| clearTimeout(id\_of\_setTimeout); |

**参数id\_of\_setTimeout是setTimeout()返回的ID值**，该值标识要取消的延迟执行代码块。

|  |
| --- |
| **范例：计数器** |
| **counter.html** |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>计数器</title>      <script src="counter.js"></script>  </head>  <body>      <form action="get/post">          <input type="text" id="num">          <input type="button" value="stop" onclick="stopCount();">      </form>  </body>  </html> |
| **counter.js** |
| *var* cnt = 0;        // 计数  *var* counter;        // 计数器  /\*\*   \* 每隔1000毫秒计数加1   \*/  *function* count() {      document.getElementById('num').value = cnt;      cnt++;      counter = setTimeout(count, 1000);  }  /\*\*   \* 停止计数器   \*/  *function* stopCount() {      clearTimeout(counter);  }  setTimeout(count, 1000);        // 启动计数器 |

**setInterval()**

**setInterval()**在执行时，**从载入页面后每隔指定的时间执行代码**。语法如下：

|  |
| --- |
| setInterval(代码, 交互时间); |

其中，第一个参数表示要代用的函数或要执行的代码串，第二个参数表示周期性执行或调用表达式之间的时间间隔，以**毫秒**为单位（**1s = 1000ms**）。

**clearInterval()**方法可**取消由setInterval()设置的交互时间**。语法如下：

|  |
| --- |
| clearInterval(id\_of\_setInterval); |

**参数id\_of\_setInterval是setInterval()返回的ID值**。

|  |
| --- |
| **范例：实时显示当前时间** |
| **current\_time.html** |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>显示当前时间</title>      <script src="current\_time.js"></script>  </head>  <body>      <form action="get/post">          <input type="text" id="time" size="50">      </form>  </body>  </html> |
| **current\_time.js** |
| // function clock() {  //     var date = new Date();  //     document.getElementById("time").value = date;  // }  // setInterval(clock, 1000);  // 箭头函数  setInterval(() *=>* {  *var* date = new *Date*();      document.getElementById("time").value = date;  }, 1000); |

**17.3 Screen对象**

**Screen对象**

Screen对象用于获取用户的**屏幕信息**。window.screen对象在编写时**可以不使用**window前缀。语法如下：

|  |
| --- |
| window.screen.属性; |

|  |  |
| --- | --- |
| **属性** | **描述** |
| availHeight | 窗口可以使用的屏幕高度，单位像素 |
| availWidth | 窗口可以使用的屏幕宽度，单位像素 |
| colorDepth | 用户浏览器表示的颜色位数，通常为32位（每像素的位数） |
| pixelDepth | 用户浏览器表示的颜色位数，通常为32位（每像素的位数），IE不支持此属性 |
| height | 屏幕的高度，单位像素 |
| width | 屏幕的宽度，单位像素 |

**screen.availWidth**和**screen.availHeight**属性返回访问者屏幕的宽度和高度，单位为像素，**减去界面特性**，比如**任务栏**等。不同系统的任务栏**默认高度**不一样，及任务栏的位置可在屏幕上下左右任何位置，所以有可能可用宽度和高度不一样。

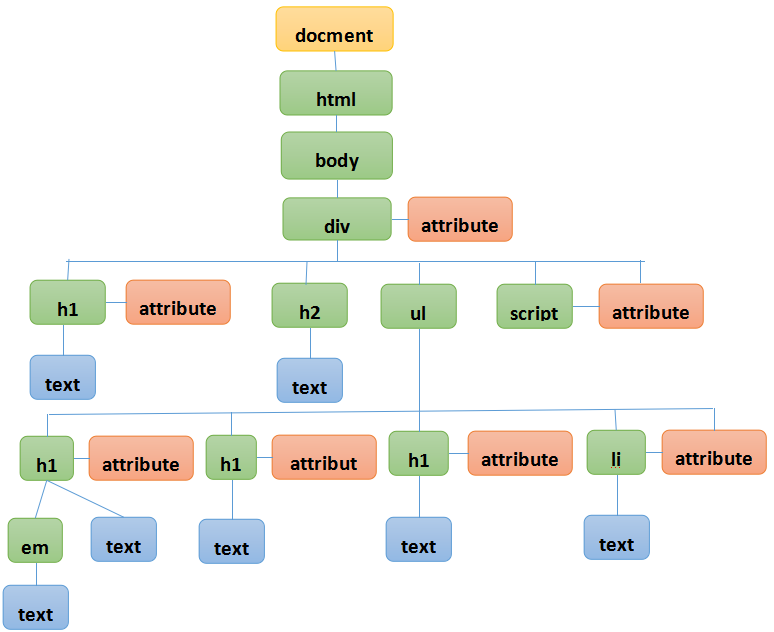
|  |  |
| --- | --- |
| **范例：屏幕信息** | |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>屏幕信息</title>      <script type="text/JavaScript">          console.log("屏幕分辨率：" + screen.width + "\*" + screen.height)          console.log("屏幕可用宽高：" + screen.availWidth + "\*" + screen.availHeight);      </script>  </head>  <body>    </body>  </html> | |
| **运行结果** | 屏幕分辨率：1536\*864  屏幕可用宽高：1536\*824 |

**第18章 文档对象模型DOM**

**18.1 DOM的概念**

**DOM (Document Object Model)**

**文档对象模型DOM**定义访问和处理HTML文档的标准方法，DOM将HTML文档呈现为带有元素、属性和文本的**树结构**（结点树）。



**18.2 获取结点对象**

**getElementById()**

**getElementById()**方法可返回拥有**指定ID的第一个对象的引用**。语法如下：

|  |
| --- |
| document.getElementById(elementID); |

如果没有指定ID的元素则返回null，如果存在多个指定ID的元素则返回第一个。

|  |
| --- |
| **范例：定时变换颜色** |
| **random\_color.html** |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>随机颜色</title>      <script src="random\_color.js"></script>  </head>  <body>      <div id="square" style="width: 100px; height: 100px"></div>  </body>  </html> |
| **random\_color.js** |
| /\*\*   \* 随机生成RGB颜色代码   \* *@returns* rgb颜色   \*/  *function* randomRGB() {  *var* r = Math.floor(Math.random() \* 256);  *var* g = Math.floor(Math.random() \* 256);  *var* b = Math.floor(Math.random() \* 256);      return "rgb(" + r + ", " + g + ", " + b + ")";  }  /\*\*   \* 获取元素结点，设置背景颜色   \*/  *function* changeColor() {  *var* obj = document.getElementById("square");      obj.style.background = randomRGB();  }  // 每隔300ms改变颜色  setInterval(*function* () {      changeColor();  }, 300); |

**getElementsByClassName()**

**getElementByClassName()**方法返回文档中**所有指定类名**的元素集合，作为NodeList对象。NodeList对象代表一个有顺序的结点列表，可以通过索引来访问列表中的结点。使用NodeList的length属性可以确定指定类名的元素个数，并循环各个元素来获取某个元素。语法如下：

|  |
| --- |
| document.getElementsByClassName(className); |

**getElementByName()**

**getElementByName()**方法返回带有**指定名称**的结点对象的集合。语法如下：

|  |
| --- |
| document.getElementsByName(name); |

与getElementById()方法不同的是，通过元素的name属性查询元素，而不是通过id属性。文档中的name属性可能不唯一，所以getElementByName()方法返回的是元素的数组，而不是一个元素。

**getElementsByTagName()**

**getElementsByTagName()**方法返回带有**指定标签名**的结点对象的集合，返回元素的顺序是它们在文档中的顺序。语法如下：

|  |
| --- |
| document.getElementsByTagName(tagName); |

**18.3 结点属性**

**结点属性**

**getAttribute()**方法可以通过元素结点的属性名称获取**属性的值**。语法如下：

|  |
| --- |
| elementNode.getAttribute(name); |

其中，elementNode可以使用getElementById()、getElementsByTagName()等方法获取到元素结点，参数name为需要查询的元素结点的属性名称。

**setAttribute()**方法可以**增加**一个指定名称和值的新属性，或者把一个**现有**的属性设定为指定的值。语法如下：

|  |
| --- |
| elementNode.setAttribute(name, value); |

其中，name表示要设置的属性名，value表示要设置的属性值。

|  |  |
| --- | --- |
| **范例：设置结点属性值** | |
| **getAttribute.html** | |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <title>设置结点属性</title>      <script src="getAttribute.js"></script>  </head>  <body>      <a class="link" href="https://www.baidu.com">百度</a>      <a class="link" href="https://www.bilibili.com">哔哩哔哩</a>  </body>  </html> | |
| **getAttribute.js** | |
| window.onload = *function*() {  *var* links = document.getElementsByClassName("link");      for(*var* i = 0; i < links.length; i++) {          console.log(links[i].getAttribute("href"));          links[i].setAttribute("target", "\_blank");      }  }; | |
| **运行结果** | https://www.baidu.com  https://www.bilibili.com |

**18.4 结点操作**

**创建结点createElement()**

**createElement()**方法可以**创建**结点元素，此方法可返回一个Element对象。语法如下：

|  |
| --- |
| document.createElement(tagName); |

其中参数tagName为字符串值，用来指明创建元素的类型。createElement()方法要与appendChild()或insertBefore()方法联合使用，将元素显示在页面中。

**插入结点appendChild()**

**appendChild()**方法可以在指定结点的最后一个子结点列表之后**添加**一个新的子结点。语法如下：

|  |
| --- |
| elementNode.appendChild(newNode); |

**插入结点insertBefore()**

**insertBefore()**方法可以在已有的子结点前**插入**一个新的子结点。语法如下：

|  |
| --- |
| elementNode.insertBefore(newNode, node); |

**删除结点removeChild()**

**removeChild()**方法可以从子结点列表中**删除**某个结点，如删除成功，此方法可返回被删除的结点，如失败则返回null。语法如下：

|  |
| --- |
| elementNode.removeChild(node); |

**替换结点replaceChild()**

**replaceChild()**方法可以实现子结点的**替换**，返回被替换对象的引用。语法如下：

|  |
| --- |
| elementNode.replaceChild(newNode, oldNode); |

当oldNode被替换时，所有与之相关的属性内容都将被移出。