# window 对象的方法

* 地址：[window 对象 - window 对象的方法 - 《阮一峰 JavaScript 教程》 - 书栈网 · BookStack](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md)
* [window 对象的方法](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#c13uim)
  + [window.alert()，window.prompt()，window.confirm()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#dujsl7)
  + [window.open(), window.close()，window.stop()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#5j5is)
  + [window.moveTo()，window.moveBy()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#esl6el)
  + [window.resizeTo()，window.resizeBy()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#a2g5kw)
  + [window.scrollTo()，window.scroll()，window.scrollBy()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#4c6z1y)
  + [window.print()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#fej9i6)
  + [window.focus()，window.blur()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#4xgpsu)
  + [window.getSelection()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#6cvx47)
  + [window.getComputedStyle()，window.matchMedia()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#f81ajc)
  + [window.requestAnimationFrame()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#5sks5i)
  + [window.requestIdleCallback()](https://www.bookstack.cn/read/javascript-tutorial/spilt.3.docs-bom-window.md#cbc19z)

**window 对象的方法**

**window.alert()，window.prompt()，window.confirm()**

window.alert()、window.prompt()、window.confirm()都是浏览器与用户互动的全局方法。它们会弹出不同的对话框，要求用户做出回应。注意，这三个方法弹出的对话框，都是浏览器统一规定的式样，无法定制。

**（1）window.alert()**

window.alert()方法弹出的对话框，只有一个“确定”按钮，往往用来通知用户某些信息。

1. window.alert('Hello World');

用户只有点击“确定”按钮，对话框才会消失。对话框弹出期间，浏览器窗口处于冻结状态，如果不点“确定”按钮，用户什么也干不了。

window.alert()方法的参数只能是字符串，没法使用 CSS 样式，但是可以用\n指定换行。

1. alert('本条提示\n分成两行');

**（2）window.prompt()**

window.prompt()方法弹出的对话框，提示文字的下方，还有一个输入框，要求用户输入信息，并有“确定”和“取消”两个按钮。它往往用来获取用户输入的数据。

1. var result = prompt('您的年龄？', 25)

上面代码会跳出一个对话框，文字提示为“您的年龄？”，要求用户在对话框中输入自己的年龄（默认显示25）。用户填入的值，会作为返回值存入变量result。

window.prompt()的返回值有两种情况，可能是字符串（有可能是空字符串），也有可能是null。具体分成三种情况。

1. 用户输入信息，并点击“确定”，则用户输入的信息就是返回值。
2. 用户没有输入信息，直接点击“确定”，则输入框的默认值就是返回值。
3. 用户点击了“取消”（或者按了 ESC 按钮），则返回值是null。

window.prompt()方法的第二个参数是可选的，但是最好总是提供第二个参数，作为输入框的默认值。

**（3）window.confirm()**

window.confirm()方法弹出的对话框，除了提示信息之外，只有“确定”和“取消”两个按钮，往往用来征询用户是否同意。

1. var result = confirm('你最近好吗？');

上面代码弹出一个对话框，上面只有一行文字“你最近好吗？”，用户选择点击“确定”或“取消”。

confirm方法返回一个布尔值，如果用户点击“确定”，返回true；如果用户点击“取消”，则返回false。

1. var okay = confirm('Please confirm this message.');
2. if (okay) {
3. // 用户按下“确定”
4. } else {
5. // 用户按下“取消”
6. }

confirm的一个用途是，用户离开当前页面时，弹出一个对话框，问用户是否真的要离开。

1. window.onunload = function () {
2. return window.confirm('你确定要离开当面页面吗？');
3. }

这三个方法都具有堵塞效应，一旦弹出对话框，整个页面就是暂停执行，等待用户做出反应。

**window.open(), window.close()，window.stop()**

**（1）window.open()**

window.open方法用于新建另一个浏览器窗口，类似于浏览器菜单的新建窗口选项。它会返回新窗口的引用，如果无法新建窗口，则返回null。

1. var popup = window.open('somefile.html');

上面代码会让浏览器弹出一个新建窗口，网址是当前域名下的somefile.html。

open方法一共可以接受三个参数。

1. window.open(url, windowName, [windowFeatures])

* url：字符串，表示新窗口的网址。如果省略，默认网址就是about:blank。
* windowName：字符串，表示新窗口的名字。如果该名字的窗口已经存在，则占用该窗口，不再新建窗口。如果省略，就默认使用\_blank，表示新建一个没有名字的窗口。另外还有几个预设值，\_self表示当前窗口，\_top表示顶层窗口，\_parent表示上一层窗口。
* windowFeatures：字符串，内容为逗号分隔的键值对（详见下文），表示新窗口的参数，比如有没有提示栏、工具条等等。如果省略，则默认打开一个完整 UI 的新窗口。如果新建的是一个已经存在的窗口，则该参数不起作用，浏览器沿用以前窗口的参数。

下面是一个例子。

1. var popup = window.open(
2. 'somepage.html',
3. 'DefinitionsWindows',
4. 'height=200,width=200,location=no,status=yes,resizable=yes,scrollbars=yes'
5. );

上面代码表示，打开的新窗口高度和宽度都为200像素，没有地址栏，但有状态栏和滚动条，允许用户调整大小。

第三个参数可以设定如下属性。

* left：新窗口距离屏幕最左边的距离（单位像素）。注意，新窗口必须是可见的，不能设置在屏幕以外的位置。
* top：新窗口距离屏幕最顶部的距离（单位像素）。
* height：新窗口内容区域的高度（单位像素），不得小于100。
* width：新窗口内容区域的宽度（单位像素），不得小于100。
* outerHeight：整个浏览器窗口的高度（单位像素），不得小于100。
* outerWidth：整个浏览器窗口的宽度（单位像素），不得小于100。
* menubar：是否显示菜单栏。
* toolbar：是否显示工具栏。
* location：是否显示地址栏。
* personalbar：是否显示用户自己安装的工具栏。
* status：是否显示状态栏。
* dependent：是否依赖父窗口。如果依赖，那么父窗口最小化，该窗口也最小化；父窗口关闭，该窗口也关闭。
* minimizable：是否有最小化按钮，前提是dialog=yes。
* noopener：新窗口将与父窗口切断联系，即新窗口的window.opener属性返回null，父窗口的window.open()方法也返回null。
* resizable：新窗口是否可以调节大小。
* scrollbars：是否允许新窗口出现滚动条。
* dialog：新窗口标题栏是否出现最大化、最小化、恢复原始大小的控件。
* titlebar：新窗口是否显示标题栏。
* alwaysRaised：是否显示在所有窗口的顶部。
* alwaysLowered：是否显示在父窗口的底下。
* close：新窗口是否显示关闭按钮。

对于那些可以打开和关闭的属性，设为yes或1或不设任何值就表示打开，比如status=yes、status=1、status都会得到同样的结果。如果想设为关闭，不用写no，而是直接省略这个属性即可。也就是说，如果在第三个参数中设置了一部分属性，其他没有被设置的yes/no属性都会被设成no，只有titlebar和关闭按钮除外（它们的值默认为yes）。

上面这些属性，属性名与属性值之间用等号连接，属性与属性之间用逗号分隔。

1. 'height=200,width=200,location=no,status=yes,resizable=yes,scrollbars=yes'

另外，open()方法的第二个参数虽然可以指定已经存在的窗口，但是不等于可以任意控制其他窗口。为了防止被不相干的窗口控制，浏览器只有在两个窗口同源，或者目标窗口被当前网页打开的情况下，才允许open方法指向该窗口。

window.open方法返回新窗口的引用。

1. var windowB = window.open('windowB.html', 'WindowB');
2. windowB.window.name // "WindowB"

注意，如果新窗口和父窗口不是同源的（即不在同一个域），它们彼此不能获取对方窗口对象的内部属性。

下面是另一个例子。

1. var w = window.open();
2. console.log('已经打开新窗口');
3. w.location = 'http://example.com';

上面代码先打开一个新窗口，然后在该窗口弹出一个对话框，再将网址导向example.com。

由于open这个方法很容易被滥用，许多浏览器默认都不允许脚本自动新建窗口。只允许在用户点击链接或按钮时，脚本做出反应，弹出新窗口。因此，有必要检查一下打开新窗口是否成功。

1. var popup = window.open();
2. if (popup === null) {
3. // 新建窗口失败
4. }

**（2）window.close()**

window.close方法用于关闭当前窗口，一般只用来关闭window.open方法新建的窗口。

1. popup.close()

该方法只对顶层窗口有效，iframe框架之中的窗口使用该方法无效。

**（3）window.stop()**

window.stop()方法完全等同于单击浏览器的停止按钮，会停止加载图像、视频等正在或等待加载的对象。

1. window.stop()

**window.moveTo()，window.moveBy()**

window.moveTo()方法用于移动浏览器窗口到指定位置。它接受两个参数，分别是窗口左上角距离屏幕左上角的水平距离和垂直距离，单位为像素。

1. window.moveTo(100, 200)

上面代码将窗口移动到屏幕(100, 200)的位置。

window.moveBy()方法将窗口移动到一个相对位置。它接受两个参数，分别是窗口左上角向右移动的水平距离和向下移动的垂直距离，单位为像素。

1. window.moveBy(25, 50)

上面代码将窗口向右移动25像素、向下移动50像素。

为了防止有人滥用这两个方法，随意移动用户的窗口，目前只有一种情况，浏览器允许用脚本移动窗口：该窗口是用window.open()方法新建的，并且窗口里只有它一个 Tab 页。除此以外的情况，使用上面两个方法都是无效的。

**window.resizeTo()，window.resizeBy()**

window.resizeTo()方法用于缩放窗口到指定大小。

它接受两个参数，第一个是缩放后的窗口宽度（outerWidth属性，包含滚动条、标题栏等等），第二个是缩放后的窗口高度（outerHeight属性）。

1. window.resizeTo(
2. window.screen.availWidth / 2,
3. window.screen.availHeight / 2
4. )

上面代码将当前窗口缩放到，屏幕可用区域的一半宽度和高度。

window.resizeBy()方法用于缩放窗口。它与window.resizeTo()的区别是，它按照相对的量缩放，window.resizeTo()需要给出缩放后的绝对大小。

它接受两个参数，第一个是水平缩放的量，第二个是垂直缩放的量，单位都是像素。

1. window.resizeBy(-200, -200)

上面的代码将当前窗口的宽度和高度，都缩小200像素。

**window.scrollTo()，window.scroll()，window.scrollBy()**

window.scrollTo方法用于将文档滚动到指定位置。它接受两个参数，表示滚动后位于窗口左上角的页面坐标。

1. window.scrollTo(x-coord, y-coord)

它也可以接受一个配置对象作为参数。

1. window.scrollTo(options)

配置对象options有三个属性。

* top：滚动后页面左上角的垂直坐标，即 y 坐标。
* left：滚动后页面左上角的水平坐标，即 x 坐标。
* behavior：字符串，表示滚动的方式，有三个可能值（smooth、instant、auto），默认值为auto。

1. window.scrollTo({
2. top: 1000,
3. behavior: 'smooth'
4. });

window.scroll()方法是window.scrollTo()方法的别名。

window.scrollBy()方法用于将网页滚动指定距离（单位像素）。它接受两个参数：水平向右滚动的像素，垂直向下滚动的像素。

1. window.scrollBy(0, window.innerHeight)

上面代码用于将网页向下滚动一屏。

如果不是要滚动整个文档，而是要滚动某个元素，可以使用下面三个属性和方法。

* Element.scrollTop
* Element.scrollLeft
* Element.scrollIntoView()

**window.print()**

window.print方法会跳出打印对话框，与用户点击菜单里面的“打印”命令效果相同。

常见的打印按钮代码如下。

1. document.getElementById('printLink').onclick = function () {
2. window.print();
3. }

非桌面设备（比如手机）可能没有打印功能，这时可以这样判断。

1. if (typeof window.print === 'function') {
2. // 支持打印功能
3. }

**window.focus()，window.blur()**

window.focus()方法会激活窗口，使其获得焦点，出现在其他窗口的前面。

1. var popup = window.open('popup.html', 'Popup Window');
2. if ((popup !== null) && !popup.closed) {
3. popup.focus();
4. }

上面代码先检查popup窗口是否依然存在，确认后激活该窗口。

window.blur()方法将焦点从窗口移除。

当前窗口获得焦点时，会触发focus事件；当前窗口失去焦点时，会触发blur事件。

**window.getSelection()**

window.getSelection方法返回一个Selection对象，表示用户现在选中的文本。

1. var selObj = window.getSelection();

使用Selection对象的toString方法可以得到选中的文本。

1. var selectedText = selObj.toString();

**window.getComputedStyle()，window.matchMedia()**

window.getComputedStyle()方法接受一个元素节点作为参数，返回一个包含该元素的最终样式信息的对象，详见《CSS 操作》一章。

window.matchMedia()方法用来检查 CSS 的mediaQuery语句，详见《CSS 操作》一章。

**window.requestAnimationFrame()**

window.requestAnimationFrame()方法跟setTimeout类似，都是推迟某个函数的执行。不同之处在于，setTimeout必须指定推迟的时间，window.requestAnimationFrame()则是推迟到浏览器下一次重流时执行，执行完才会进行下一次重绘。重绘通常是 16ms 执行一次，不过浏览器会自动调节这个速率，比如网页切换到后台 Tab 页时，requestAnimationFrame()会暂停执行。

如果某个函数会改变网页的布局，一般就放在window.requestAnimationFrame()里面执行，这样可以节省系统资源，使得网页效果更加平滑。因为慢速设备会用较慢的速率重流和重绘，而速度更快的设备会有更快的速率。

该方法接受一个回调函数作为参数。

1. window.requestAnimationFrame(callback)

上面代码中，callback是一个回调函数。callback执行时，它的参数就是系统传入的一个高精度时间戳（performance.now()的返回值），单位是毫秒，表示距离网页加载的时间。

window.requestAnimationFrame()的返回值是一个整数，这个整数可以传入window.cancelAnimationFrame()，用来取消回调函数的执行。

下面是一个window.requestAnimationFrame()执行网页动画的例子。

1. var element = document.getElementById('animate');
2. element.style.position = 'absolute';
3. var start = null;
4. function step(timestamp) {
5. if (!start) start = timestamp;
6. var progress = timestamp - start;
7. // 元素不断向左移，最大不超过200像素
8. element.style.left = Math.min(progress / 10, 200) + 'px';
9. // 如果距离第一次执行不超过 2000 毫秒，
10. // 就继续执行动画
11. if (progress < 2000) {
12. window.requestAnimationFrame(step);
13. }
14. }
15. window.requestAnimationFrame(step);

上面代码定义了一个网页动画，持续时间是2秒，会让元素向右移动。

**window.requestIdleCallback()**

window.requestIdleCallback()跟setTimeout类似，也是将某个函数推迟执行，但是它保证将回调函数推迟到系统资源空闲时执行。也就是说，如果某个任务不是很关键，就可以使用window.requestIdleCallback()将其推迟执行，以保证网页性能。

它跟window.requestAnimationFrame()的区别在于，后者指定回调函数在下一次浏览器重排时执行，问题在于下一次重排时，系统资源未必空闲，不一定能保证在16毫秒之内完成；window.requestIdleCallback()可以保证回调函数在系统资源空闲时执行。

该方法接受一个回调函数和一个配置对象作为参数。配置对象可以指定一个推迟执行的最长时间，如果过了这个时间，回调函数不管系统资源有无空虚，都会执行。

1. window.requestIdleCallback(callback[, options])

callback参数是一个回调函数。该回调函数执行时，系统会传入一个IdleDeadline对象作为参数。IdleDeadline对象有一个didTimeout属性（布尔值，表示是否为超时调用）和一个timeRemaining()方法（返回该空闲时段剩余的毫秒数）。

options参数是一个配置对象，目前只有timeout一个属性，用来指定回调函数推迟执行的最大毫秒数。该参数可选。

window.requestIdleCallback()方法返回一个整数。该整数可以传入window.cancelIdleCallback()取消回调函数。

下面是一个例子。

1. requestIdleCallback(myNonEssentialWork);
2. function myNonEssentialWork(deadline) {
3. while (deadline.timeRemaining() > 0) {
4. doWorkIfNeeded();
5. }
6. }

上面代码中，requestIdleCallback()用来执行非关键任务myNonEssentialWork。该任务先确认本次空闲时段有剩余时间，然后才真正开始执行任务。

下面是指定timeout的例子。

1. requestIdleCallback(processPendingAnalyticsEvents, { timeout: 2000 });

上面代码指定，processPendingAnalyticsEvents必须在未来2秒之内执行。

如果由于超时导致回调函数执行，则deadline.timeRemaining()返回0，deadline.didTimeout返回true。

如果多次执行window.requestIdleCallback()，指定多个回调函数，那么这些回调函数将排成一个队列，按照先进先出的顺序执行。