一文搞懂BOM浏览器对象模型及应用！

前言

作为一名前端开发者，每个人都是从小白过来的。在我们最开始接触JavaScript的时候，相信大家听到最多的知识点之一就是**DOM对象模型和BOM对象模型**，其中DOM就是我们的文档对象，也就是HTML元素等等，而BOM则是浏览器对象模型，更为准确的说应该是BOM是我们利用JavaScript开发Web应用的核心，它提供给我们可以操作浏览器的一些功能对象。

很多时候如果我们有的需求是针对于浏览器层面的，我们应该首要想到BOM对象，但是很多小伙伴却由于长期不接触而逐渐忘记，今天我们就将BOM捡起来，好好学习一番。

1.DOM和BOM

为了照顾初学的小伙伴，我们简单的介绍下DOM和BOM，以便更好的学习后面的BOM知识点。

首先我们需要知道两个的区别以及基本概念，总结如下。

DOM：

> DOM又称文档对象模型，它是HTML和XML文档的编程接口，它主要描述了一些我们使用JS处理网页内容的方法和接口，它的目标是网页内容。比如说我们使用JS修改页面内容，其实就是操作的DOM文档。

BOM：

> BOM又称浏览器对象模型，它主要用来描述一些与浏览器行为相关的接口和方法，比如我们利用JS调整浏览器窗口大小、标签页跳转等等，这些都是BOM对象。

对于初学者来说，可以理解DOM和BOM有一点点难，因为它们都是对象模型，不是真正存在的东西，它们似乎是一种规范。

为了更好的理解，我这里用最通俗的语言解释一下DOM和BOM。

最简单的理解：

> 大家可以把DOM和BOM想象成两个大箩筐，DOM箩筐里面装的是HTML、XHTML等等元素、属性和相关方法对象，而BOM箩筐内装的是浏览器的相关属性和方法对象。DOM和BOM这两个箩筐中都有一些核心对象或者方法，这就是我们需要学习的。

这样来解释是不是就很好理解了，我们要学会把虚拟的东西实物话，就像上学的时候做物理提时，我们要学会画图，把干瘪瘪的题目实物为图形。

前面既然说BOM对象模型是针对于浏览器的，那么它暴漏了哪些对象供我们使用呢？或者说我们如何操作浏览器呢？接下来我们就分别学习一下BOM暴漏给我们的一些对象。

2.window对象

window对象是BOM的核心对象，它表示的是浏览器的实例。它在我们的JS中则代表全局对象，即Global对象的意思，当然，既然是浏览器实例，我们就只能在网页中的JS内使用window，在普通ECMAscript中，window即Global对象。

既然是全局对象，意味着我们在网页中定义的所有全局变量、方法等都可以在window对象下找到。

我们可以讲window对象打印出来看看。

代码如下：

```
1 console.log(window)
```

输出结果：

2.1 Global作用域

因为window代表的是ECMAscript中的Global对象，大家可以简单的讲window与Global划等号。所以如果使用var声明的全局变量或者方法，都将称为window对象的属性和方法。

代码如下：

```
var name = '小猪课堂';
var say = () => {
  console.log('你好！小猪课堂');
}
console.log(name); // 小猪课堂
console.log(window.name); // 小猪课堂

say(); // 你好！小猪课堂
window.say(); // 你好！小猪课堂
```

上段代码我们使用var声明全局变量和全局方法，然后使用window对象调用它们，这说明我们声明的这些变量和方法全都挂载到了window对象上去了。

需要注意的是：

如果我们使用let或者const声明方法时，该方法不会挂载到window对象上去，比如下列代码会报错：

```
let say = () => {
  console.log('你好！小猪课堂');
}
window.say(); // Uncaught TypeError: window.say is not a function
```

细心的小伙伴应该也发现了，我们声明的全局变量可以不适用window.say的形式调用，可以省略window，因为window是全局作用域，变量或方法找不到的时候自然要去window上找，所以我们挂载到window对象上的方法或变量通常可以直接调用，而不需要window.。

2.2 window窗口关系

每一个窗口都会存在一个window对象，那么这些window对象之间有没有什么关联呢？特别是在有时候我们在某一个网页窗口中，又打开了一个网页窗口，那么它们之间有没有联系呢？

window对象中有三个属性，这三个属性可以用来区别窗口之间的关联：

- window.top：顶层窗口，即最外层窗口
- window.parent：当前窗口的父窗口，如果当前窗口即顶层窗口，则window.top=window.parent。
- self：当前窗口

我们可以把这三个属性打印出来看看。

代码如下：

```
console.log(window.top);
```

```
console.log(window.parent);
console.log(window.self);
console.log(window.top == window.parent); // true
console.log(window.top == window.self); // true
```

打印结果：

```
▶ Window {window: Window, self: Window, document: document, name: '', location: Location, …}
▶ Window {window: Window, self: Window, document: document, name: '', location: Location, …}
▶ Window {window: Window, self: Window, document: document, name: '', location: Location, …}
true
true
```

我们可以看到window的parent、top、self三个属性是相等的，因为我们这里只有一个窗口。你可是试试在当前也变打开新的窗口，然后再看看这三个属性的只，比如使用a标签打开。

2.3 窗口位置

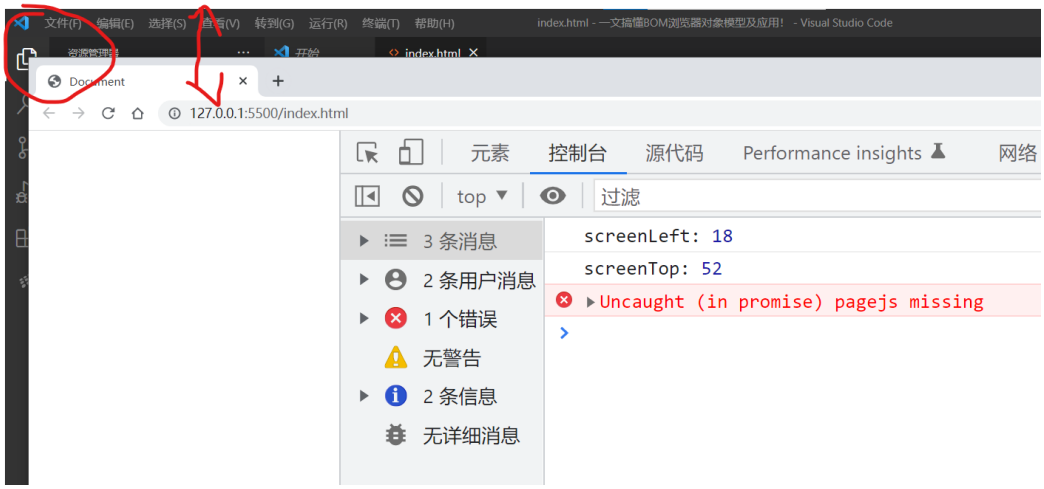
window对象中提供了两个属性来获取当前窗口距离屏幕左侧和顶部的距离，单位为像素。

- screenLeft：窗口距离屏幕左侧的距离
- screenTop：窗口距离屏幕顶部的距离

示例代码：

```
console.log('screenLeft:', window.screenLeft);
console.log('screenTop:', window.screenTop);
```

打印结果：



window对象内部还提供了两个方法将浏览器窗口移动到指定位置，当然，这两个方法根据浏览器的不同可能会被禁掉。

- moveTo：将窗口移动到指定坐标
- moveBy：将窗口向指定方向移动指定像素值

示例代码：

```
window.moveTo(200, 200); // 把窗口移动到坐标位置(200, 200)
window.moveBy(-50, 0); // 将窗口像做移动50像素
```

2.4 窗口位置

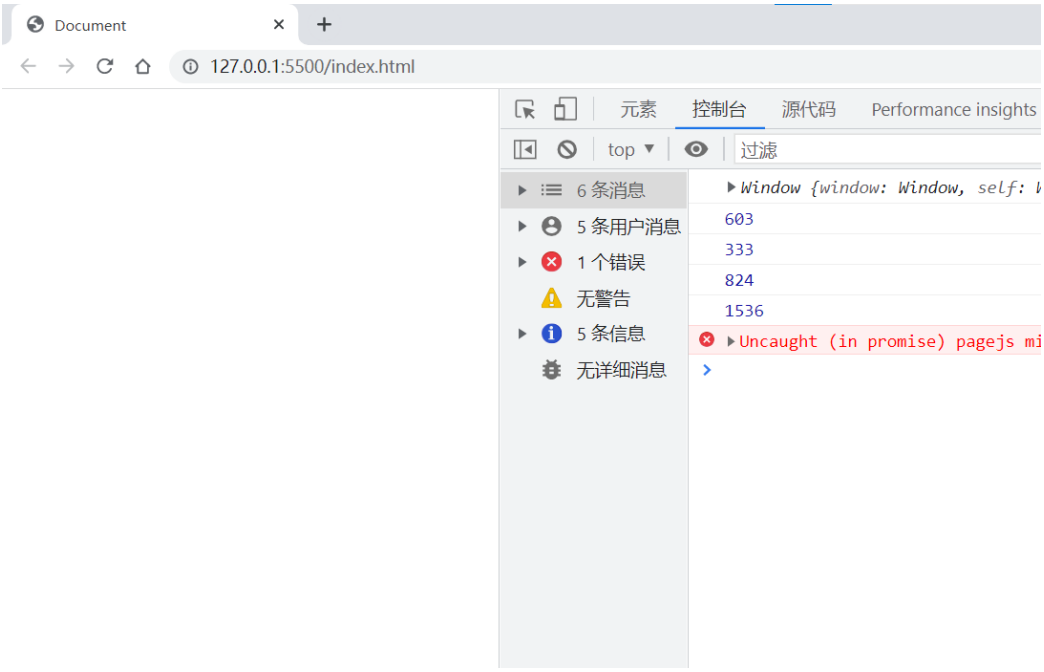
window虽然提供了属性获取浏览器窗口的大小，但是在不同的浏览器中获取浏览器窗口大小都不是太容易，目前常见的浏览器通常提供一下4个属性获取浏览器窗口大小：

- innerWidth
- innerHeight
- outerWidth
- outerHeight

示例代码：

```
console.log(window.innerHeight);
console.log(window.innerWidth);
console.log(window.outerHeight);
console.log(window.outerWidth);
```

输出结果：



可以看到在这里innerHeight和innerWidth代表的是浏览器可视窗口的大小，outerHeight和outerWidth表示的是浏览器窗口的大小。

但是在移动设备上，可视窗口的大小近乎等于浏览器窗口的大小。

在我们的实际开发中通常使用如下方法获取可视窗口的大小：

```
document.documentElement.clientHeight
document.documentElement.clientWidth
```

2.5 视口位置

前面我们讲解的都是浏览器窗口的位置，其实window对象中还提供了方法用来控制可视窗口的位置，所谓视口就是可视区域，比如我们一个网页出现了横向或者纵向滚动条，那么它必然有些内容看不见。

这个时候我们可以手动滚动滚动条来让内容出现，我们也可以使用window提供的方法来让可视窗口变换位置。

- scrollBy：让视口像指定方向滚动多少像素值
- scrollTo：让视口滚动到指定坐标

示例代码：

```
window.scrollBy(0, 100); // 让视口向下滚动100px
window.scrollTo(0, 0); // 让视口滚动到页面左上角
```

初次之前scrollTo方法还可以接收其它参数，比如控制滚动的时候是否平滑，感兴趣的小伙伴可以自行下来更加仔细的学习。

2.6 导航与打开新窗口

window对象提供了一个open()方法，它用于导航到指定URL，当然也可以是打开新窗口。open方法接收三个参数。

- window.open(url, target, str, bool)
  - url：需要打开的url地址
  - target：目标窗口，假如已经有目标窗口存在，则会在对应的窗口中打开该url
  - str：一个特性字符串，主要是用来配置新窗口或者新标签页的浏览器特性，比如工具栏、窗台栏等等
  - bool：一个布尔值，用来控制是否需要打开新的窗口。

我们通常只需要传前面三个参数即可，因为我们常见的需求就是打开一个新的标签页等等。

示例代码：

```
window.open('www.baidu.com', 'topFram') // 在目标窗口topFram中打开url
// 可以使用a标签实现
<a href="www.baidu.com" target="topFram"></a>
```

2.7 window其它属性

前面我们介绍了几个window对象下的属性，其实window对象里面还有特别多的属性，而且我们经常用到，这里举例几个，大家看到这几个应该就能举一反三了。

- setTimeout、setInterval
- alert()

- parseInt()

.....

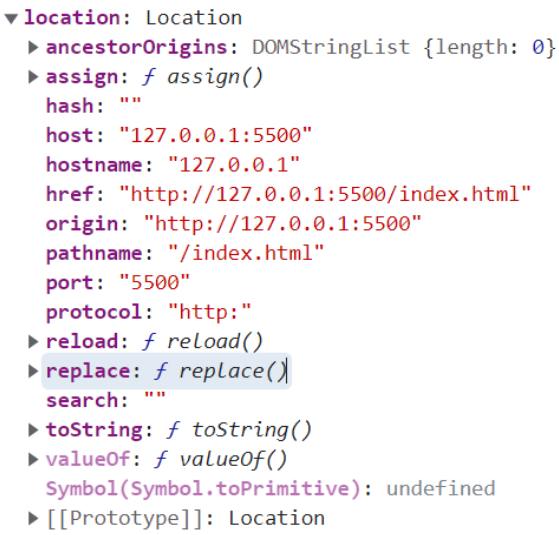
具体还有哪些属性大家可以打印window对象出来看看

3.location对象

location对象是BOM对象中非常有用的，可以说是最重要的特性之一，它主要提供了当前窗口中加载的文档信息，以及一些导航功能。location对象既是window的属性，也是document的属性。所以我们可以通过两种方式来获取它：

- window.location
- document.location

我们可以在window对象中找到它，如下图所示：



可以看到location对象中包含了很多有用的信息，大家可以根据需求自行获取。

3.1 查询字符串

在location中有一个属性很重要，它就是search属性，它的值就是url中包括？在内后后面所哟内容。

比如我们查看下面打印结果：

利用search我们就可以不用分割url便获得了url的查询字段，再借助字符串分割，我们就能很方便的获取url参数了。

3.2 操作url

location属性中有一个assign方法，该方法接收一个url地址，它会让我们的网页立即启动导航到新的url，而且还会再浏览器历史记录中增加一条记录。

示例代码：

```
location.assign('http://www.baidu.com');
```

上段代码将会立即讲我们的页面跳转至指定url页面。

当然，细心的小伙伴可能发现location对象中的href属性便是当前url地址，我们是否可以更改此地址来达到跳转的目的呢?

实际上，上述代码的实现效果与下段代码的实现效果一致：

```
location.href = 'http://www.baidu.com'
或
location = 'http://www.baidu.com'
```

4.navigator对象

BOM中提供了一个navigator对象，它主要用来获取浏览器的一些基础信息，比如版本、名称等信息。

navigator提供了特别多的属性，大家可以有需要的时候去查看每个属性代表的意义即可：

属性/方法	说　明
activeVrDisplays	返回数组，包含 ispresenting 属性为 true 的 VRDisplay 实例
appCodeName	即使在非 Mozilla 浏览器中也会返回"Mozilla"
appName	浏览器全名
appVersion	浏览器版本。通常与实际的浏览器版本不一致
battery	返回暴露 Battery Status API 的 BatteryManager 对象
buildId	浏览器的构建编号
connection	返回暴露 Network Information API 的 NetworkInformation 对象
cookieEnabled	返回布尔值，表示是否启用了 cookie
credentials	返回暴露 Credentials Management API 的 CredentialsContainer 对象
deviceMemory	返回单位为 GB 的设备内存容量
doNotTrack	返回用户的“不跟踪”（do-not-track）设置
geolocation	返回暴露 Geolocation API 的 Geolocation 对象
getVRDisplays()	返回数组，包含可用的每个 VRDisplay 实例
getUserMedia()	返回与可用媒体设备硬件关联的流
hardwareConcurrency	返回设备的处理器核心数量
javaEnabled	返回布尔值，表示浏览器是否启用了 Java
language	返回浏览器的主语言
languages	返回浏览器偏好的语言数组

（续）

属性/方法	说　明
locks	返回暴露 Web Locks API 的 LockManager 对象
mediaCapabilities	返回暴露 Media Capabilities API 的 MediaCapabilities 对象
mediaDevices	返回可用的媒体设备
maxTouchPoints	返回设备触摸屏支持的最大触点数
mimeTypes	返回浏览器中注册的 MIME 类型数组
onLine	返回布尔值，表示浏览器是否联网
oscpu	返回浏览器运行设备的操作系统和（或）CPU
permissions	返回暴露 Permissions API 的 Permissions 对象
platform	返回浏览器运行的系统平台
plugins	返回浏览器安装的插件数组。在 IE 中，这个数组包含页面中所有<embed>元素
product	返回产品名称（通常是"Gecko"）
productSub	返回产品的额外信息（通常是 Gecko 的版本）
registerProtocolHandler()	将一个网站注册为特定协议的处理程序
requestMediaKeySystemAccess()	返回一个期约，解决为 MediaKeySystemAccess 对象
sendBeacon()	异步传输一些小数据
serviceWorker	返回用来与 ServiceWorker 实例交互的 ServiceWorkerContainer
share()	返回当前平台的原生共享机制
storage	返回暴露 Storage API 的 StorageManager 对象
userAgent	返回浏览器的用户代理字符串
vendor	返回浏览器的厂商名称
vendorSub	返回浏览器厂商的更多信息
vibrate()	触发设备振动
webdriver	返回浏览器当前是否被自动化程序控制

我们使用navigator属性时，通常就是为了确定浏览器的类型。

4.1 检测插件

我们经常上一些网站的时候会提示我们安装某些插件等等，它们是怎么知道我们没有安装某个插件呢?

这其实就要归功于navigator对象中plugins属性，plugins属性是一个数组，数组中的每一项都包含一下属性：

- name：插件名称
- description：插件介绍

- filename：插件文件名
- length：由当前插件处理的MIME类型数量

示例代码：

```
window.navigator.plugins
```

5.screen对象

screen对象也是BOM对象中的属性之一，可以通过window.screen来调用。它主要用来存储客户端各种信息，由于用的比较少，我们可以给出它的每个属性代表什么，供大家参考。

属　性	说　明
availHeight	屏幕像素高度减去系统组件高度（只读）
availLeft	没有被系统组件占用的屏幕的最左侧像素（只读）
availTop	没有被系统组件占用的屏幕的最顶端像素（只读）
availWidth	屏幕像素宽度减去系统组件宽度（只读）
colorDepth	表示屏幕颜色的位数；多数系统是 32（只读）
height	屏幕像素高度
left	当前屏幕左边的像素距离
pixelDepth	屏幕的位深（只读）
top	当前屏幕顶端的像素距离
width	屏幕像素宽度
orientation	返回 Screen Orientation API 中屏幕的朝向

我们也可以打印出来看看。

代码如下：

```
console.log(screen)
```

打印结果：

```
▼Screen {availWidth: 1536, availHeight: 824, width: 1536, height: 864, colorDepth: 24, …}
  availHeight: 824
  availLeft: 0
  availTop: 0
  availWidth: 1536
  colorDepth: 24
  height: 864
  isExtended: false
  onchange: null
▶ orientation: ScreenOrientation {angle: 0, type: 'landscape-primary', onchange: null}
  pixelDepth: 24
  width: 1536
▶ [[Prototype]]: Screen
```

6.history对象

history对象主要用来表示当前窗口的历史导航记录，但是由于安全考虑，我们时没法直接看到访问过的url，我们可以打印出来看看：

```
▼History {length: 4, scrollRestoration: 'auto', state: null}
    length: 4
    scrollRestoration: "auto"
    state: null
  ▶ [[Prototype]]: History
```

6.1 导航

history对象中有一个go方法，它可以控制浏览器的前进后退，它的参数为一个数字，代表的时浏览器前进或者后退多少步。

示例代码：

```
// 后退一页
history.go(-1)
// 前进一页
history.go(1)
// 前进两页
history.go(2)
```

6.2 历史状态管理

在以前,我们每次变化url是则代表重新刷新页面,比如浏览器前进后退则代表刷新整个页面,这种用户体验是非常查的.

为了解决这类问题, 出现了hashChange事件,它会在页面url散列变化时被触发, 这个时候我们可以执行一些我们自己的操作. 对于hashChange事件需要比较复杂,需要大家自己下来补充这一块的知识.

而我们的history对象中有一个pushState方法, 该方法可以让开发者改变浏览器URL的时候不会加载新的页面. 该方法接收三个参数.

示例代码:

```
history.pushState(stateObject, "My title", "baz.html");
```

pushState的具体使用方法和规则可以参考MDN：pushState

这里不做过多介绍, 总之大叫需要知道pushState的作用即可:

> pushState可以让我们的url改变时不刷新页面.

目前很多单页面应用的路由就是使用的这一原理, 后续我们也会推出文章专门讲解pushState.

总结

到这里我们介绍了BOM对象中的常见属性和方法, 大家有没有发现, 我们介绍的所有属性和方法都是在window对象下, 所以,大家到这里是否明白了BOM对象模型是什么呢?