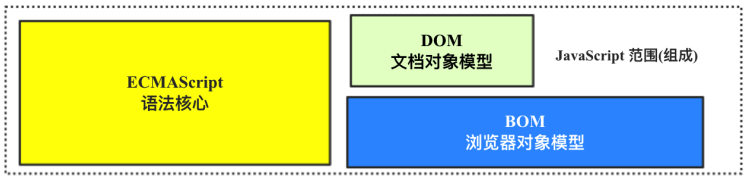
BOM

> 本文介绍BOM相关的知识点，介绍的重点在于BOM核心Window对象的成员细节。

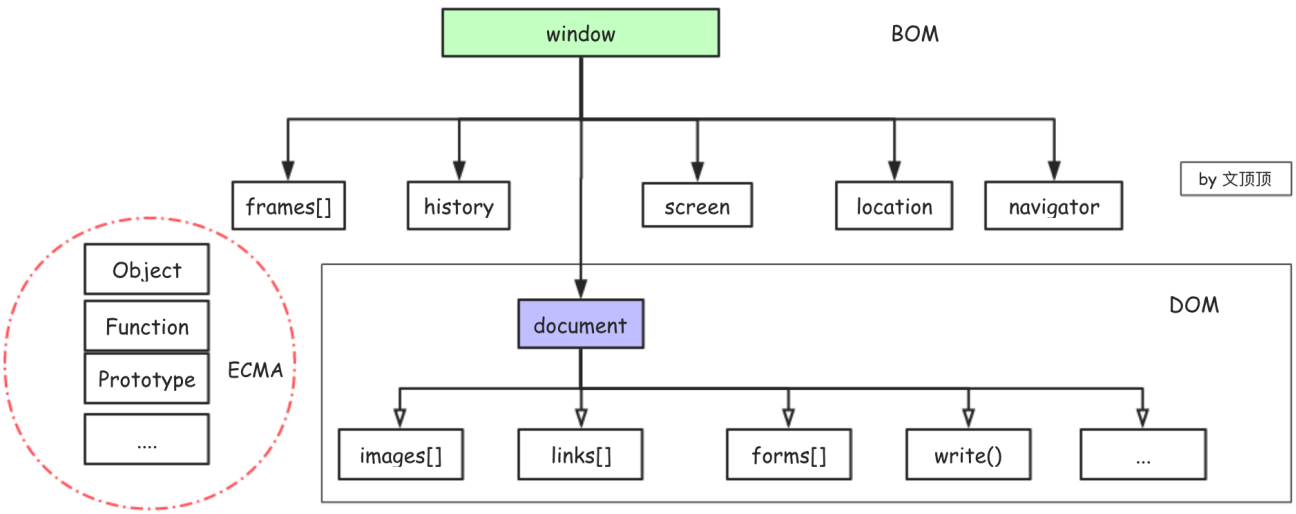
BOM简单介绍



我们已经知道JavaScript的范围包括 `ECMAScript(语言核心) + DOM(文档对象模型) + BOM(浏览器对象模型)` 。

BOM 是 Browser Object Model（浏览器对象模型）的缩写，它提供了独立于内容 而与浏览器窗口进行交互的对象。由于BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是window，它表示流浏览器的一个实例。BOM由一系列相关的对象构成，并且每个对象都提供了很多方法与属性，window作为最顶层的对象，BOM中所有的对象都是通过它延伸出来的。

标准化 JavaScript语法的标准化组织是ECMA，DOM的标准化组织是W3C, 而BOM因为缺乏标准，BOM最初其实是Netscape浏览器标准的一部分，而这也正是各种浏览器不兼容的根源所在。此外需要指出的是，W3C为了把浏览器中JavaScript最基本的部分标准化，已经将BOM的主要方面纳入了HTML5的规范中。



window和全局作用域

在浏览器中，window对象拥有着双重的角色，它既是通过JavaScript访问浏览器窗口的接口，也是ECMAScript规定中的Global全局对象。因此，所有在全局作用域中声明的变量、函数都会自动成为window对象的属性和方法。

全局作用域

> ✧ 定义在全局环境下的变量和函数都会成为 `window` 对象的成员(属性和方法)
> ✧ 编码的时候应该尽可能少的使用全局变量，以避免污染全局环境
> ✧ 没有用var声明的变量会成为全局变量，即 `window` 对象的属性
> ✧ 在编码时 `window` 前缀可以被省略，如 `window.console.log()` 通常写成 `console.log()` ;

```
/*01-全局作用域示例*/
console.log(window.age);        //undefined
var age = 12;
console.log(age, window.age);   //12 12  age自动成为window的属性

console.log(window.arr);        //undefined
var arr = [1,2,3,"T"];
console.log(arr == window.arr); //true
arr.push("V");
console.log(window.arr);        //[1,2,3,"T","V"];

function sum(a,b) {
  return a + b;
}
console.log(sum(1, 2));         //3
console.log(window.sum(1, 3));  //4

/*02-没有使用var声明的变量默认成为全局变量*/
address = "北京市海淀区";
console.log(window.address);   //"北京市海淀区";

function f() {
  var a = "我是a";
      b = "我是b";
}

f();                            /*调用函数f，执行函数体中的代码*/
//console.log(a);               //报错 无法访问变量a 因为a是f函数中的局部变量
console.log(b);                 //"我是b"  b默认成为全局变量
console.log(window.a);          //undefined
console.log(window.b);          //"我是b"

/*03-省略window前缀*/
console.log(window.console == console);  //true
```

因为全局作用域中声明的所有变量都会自动成为window的属性，而实际的开发中代码量可能是巨大的，且项目可能是由很N多人一起维护的，因此应该尽可能的减少全局变量的数量，以防止污染全局环境。

关于这个问题的解决方案可以有两种，一种是根据具体的业务和功能将部分代码 封装到匿名函数(闭包)中 保持独立性，一种是把很多变量和函数 封装到特定的对象中 处理。当然，更成熟的方案是使用 模块化 的方式来组织项目和代码结构，其实模块化的开发方式也是使用匿名函数封装的一种变形，在这里我们暂不作具体的展开。

注意 `delete` 用于删除对象的属性，需注意虽然使用 `delete` 关键字可以删除直接定义在 `window` 上面的属性，但却无法直接删除用 `var` 声明的全局变量(在严格模式下不能禁止使用 `delete` 来删除遍历，错误信息为 Uncaught SyntaxError: Delete of an unqualified identifier in strict mode.)。

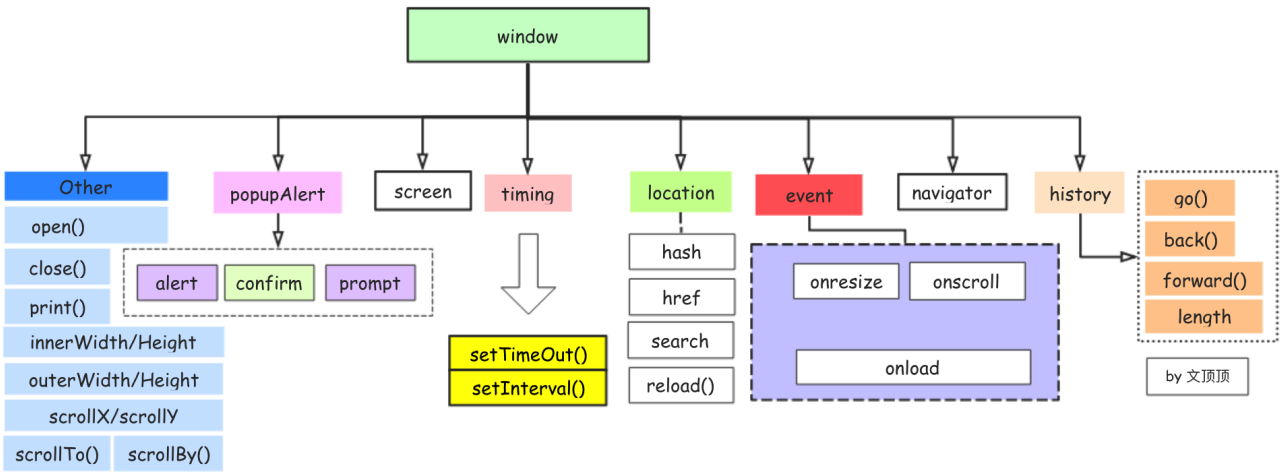
```
/*测试1：使用delete删除对象的属性*/
 var obj = {id:"1001"};
 console.log(delete obj.id);           //true
 console.log(obj.id);                  //undefined

 /*测试2：使用delete删除全局变量的属性*/
 var className = 'H5-1904';
 console.log(delete className);        //false
 console.log(className);               //"H5-1904"
 console.log(delete window.className); //false
 console.log(window.className);        //"H5-1904"

 /*测试3：使用delete删除直接定义在window对象上面的属性*/
 window.test = "测试的属性";
 console.log(delete window.test);      //true
 console.log(window.test);             //undefined
```

window核心成员详解

接下来，我将分门别类的讲解浏览器窗口的大小、滚动、导航、打开、弹窗、位置操作、历史记录以及事件处理等内容，所以这些功能都通过 `window` 的核心成员来提供和实现。



window的 location 对象

`location` 对象是window中最有用最重要的对象之一，它提供了与当前窗口中加载的文档有关的信息，而且它非常特殊，它既是 `window` 的属性也是 `document` 的属性。

```
location 对象的主要属性

hash     设置/返回从井号 (#) 开始的 URL（锚点）==>哈希值。
href     设置/返回完整的 URL。
search   设置/返回从问号 (?) 开始的 URL（参数部分）。
host     服务器名称和端口号
hostname 服务器名称
```

```
pathname URL中的目录和文件名
port     返回指定的端口号，如果没有指定则返回空字符串
protocol 返回网络协议

location 对象的主要方法
assign()             加载新的页面
reload()             重新加载(刷新)
replace(newurl)      使用新的页面来替换当前页面
```

url 的组成 http://www.baidu.com:10086/api/reg.php?username=zs&password=123&age=18#123

协议 ： http

域名 ： baidu.com

端口 ： 10086 (默认:80)

路径 ： /api/

参数 ： username=zs&password=123&age=18

哈希 ： #123

窗口的主要属性

跨浏览器确定窗口的大小可以使用window的 innerWidth 、 innerHeight 、 outerWidth 和 outerHeight 四个属性，它们分别对应的是页面视图容器的宽高和浏览器窗口本身的尺寸。需要注意的是，这几个属性存在兼容性问题，在IE8-中需要通过DOM来获取大小信息。

> **页面视图容器**也称为浏览器的视口(viewport),相比窗口本身来说它不包括工具栏和滚动条。

```
/*01-获取页面可视区域的大小(宽 | 高)*/
  var w =window.innerWidth
      || document.documentElement.clientWidth
      || document.body.clientWidth;

  var h =window.innerHeight
      || document.documentElement.clientHeight
      || document.body.clientHeight;

  /*兼容IE8-
  * 如果是标准模式则使用document.documentElement.clientWidth,
  * 如果是混杂模式则使用document.body.clientWidth
  * */
  console.log(w);                  //877   可视区域的宽度
  console.log(h);                  //410   可视区域的高度

  /*02-获取window窗口本身的大小*/
  console.log(window.outerWidth); //877   窗口的宽度
  console.log(window.outerHeight);//778   窗口的高度

  /*03-获取滚动条位置相关的信息*/
  console.log(window.scrollX);    //水平滚动条滚动过的距离（只读）
  console.log(window.scrollY);    //垂直滚动条滚动过的距离（只读）

  /*04-操作滚动的相关方法*/
```

```
window.scrollTo(100,100);        //指定滚动位置
window.scrollBy(-100,100);       //设置基于当前位置的滚动距离
```

窗口的打开和关闭

语法 `window.open([ URL ],[ name ],[ features ],[ replace ])`

作用 `open()` 方法用于打开一个新的浏览器窗口或查找一个已命名的窗口。

```
/* 作用：打开新的窗口
 * 语法：window.open(URL,name,features,replace)
 * 参数：
 *   URL    打开指定页面的URL，没有指定则打开空白窗口。
 *   name   指定target属性或窗口的名称。
 *       _blank  - 在新的窗口加载页面(默认)
 *       _parent - 在窗口中加载页面
 *       _self   - 在当前窗口中加载页面(替换)
 *       _top    - URL替换任何可加载的框架集
 *       name    - 窗口名称
 *   features  可选的字符串，声明了新窗口要显示的标准浏览器的特征
 *   replace   可选的布尔值，设置浏览历史的处理(true表示替换当前条目，false表示新建条目)
 */
/*01-打开空白窗口*/
window.open();

/*02-打开新的标签页，加载www.wendingding.com站点*/
window.open("http://www.wendingding.com");

/*03-打开新的窗口并控制窗口的外观*/
//toolbar      浏览器工具栏是否显示
//location     是否显示地址字段
//directories 是否添加目录按钮(IE)
//status       是否添加状态栏
//menubar      是否显示菜单栏
//scrollbars   是否显示滚动条
//resizable    窗口是否可调节尺寸
//width        窗口的宽度
//height       窗口的高度
window.open("http://www.wendingding.com","_blank","toolbar=yes, location=yes, " +
 "directories=no, status=no, menubar=yes, scrollbars=yes, " +
  "resizable=no,width=400, height=400");

/*04-关闭当前窗口*/
window.close();

/*05-该方法用来调出打印窗口*/
window.print();
```

系统弹框

`alert()`	弹出对话框(确定)
`confirm()`	弹出警告框，返回布尔值（确定&取消）
`prompt()`	弹出输入框，返回消息或 null

```
//console.log(alert("确定"));              /*提示框 该方法没有返回值*/
//console.log(confirm("确定删除吗?"));;    /*警示窗 确定返回true,取消返回false*/

var res = prompt("好汉请留下尊姓大名","座山雕");

//如果点击了取消那么就返回null
//如果点击了确定那么就返回输入框的内容
console.log(res);
```

window的 history 对象

window对象中有一个 `history` 属性，它本身也是一个对象保存着网页的历史记录(从窗口打开时计算)。出于安全方面的考虑，开发人员无法得知用户浏览过的URL详情，但却可以利用 `history` 对象来实现前进和后退的功能。

`history` 对象通过内部的 `length` 属性来记录浏览历史的数量，该数据包含了向前和向后的所有浏览记录，默认加载到窗口的第一个页面其 `history.length` 的值为0。

go( )	跳转到任意的浏览历史记录，负数表示后退。
back( )	后退一页
forward( )	前进一页

```
/*方法演示   下面代码中的window前缀可以省略*/
console.log(window.history.length);    //获取历史记录长度
window.history.go(1);                  //前进1页
window.history.go(-1);                 //后退1页
window.history.back();                 //回退
window.history.forward();             //前进
```

window的事件补充

`onresize` 事件会在窗口大小调整的时候被触发。
`onscroll` 事件在页面滚动条滚动的时候会被触发。
`onload` 事件会在页面加载完成(HTML+CSS+其它资源)后触发。