

BOM浏览器对象模型

BOM概述

BOM (Browser Object Model) 浏览器对象模型，它提供了独立于内容而与浏览器窗口进行交互的对象，它的核心对象是window

BOM是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

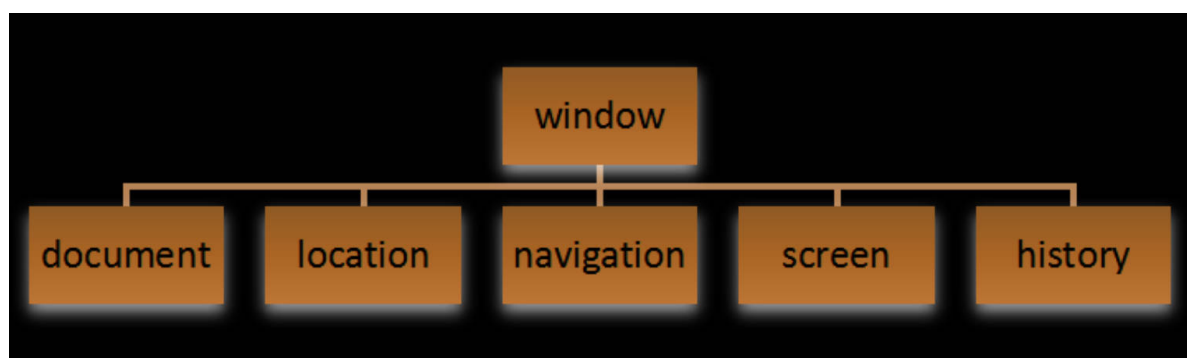
BOM 缺乏标准，JavaScript 语法的标准化组织是 ECMA，DOM 的标准化组织是 W3C，BOM 最初是 Netscape 浏览器标准的一部分。

DOM与BOM之间的区别

DOM	BOM
<ul style="list-style-type: none">• 文档对象模型• DOM 就是把「文档」当做一个「对象」来看待• DOM 的顶级对象是 <code>document</code>• DOM 主要学习的是操作页面元素• DOM 是 W3C 标准规范	<ul style="list-style-type: none">• 浏览器对象模型• 把「浏览器」当做一个「对象」来看待• BOM 的顶级对象是 <code>window</code>• BOM 学习的是浏览器窗口交互的一些对象• BOM 是浏览器厂商在各自浏览器上定义的，兼容性较差

BOM的构成

BOM比DOM更大，它包含DOM



window对象是浏览器的顶级对象，它具有双重角色

- 1、它是js访问浏览器窗口的一个接口
- 2、它是一个全局对象，定义在全局作用域中的变量、函数都会变成window对象的属性和方法

在调用的时候可省略window，前面学习的对话框都属于window对象方法，如alert ()、prompt () 等

注意：window下的一个特殊属性——window.name

window对象的常见事件

窗口加载事件

- `window.onload = function(){}`

window.onload 是窗口 (页面) 加载事件,当文档内容完全加载完成会触发该事件(包括图像、脚本文件、CSS 文件等), 就调用的处理函数。

注意：

1、有了 window.onload 就可以把 JS 代码写到页面元素的上方，因为 onload 是等页面内容全部加载完毕，再去执行处理函数。

2、window.onload 传统注册事件方式 只能写一次，如果有多个，会以最后一个 window.onload 为准。

3、如果使用 addEventListener 则没有限制

- document.addEventListener('load',function() {})

DOMContentLoaded 事件触发时，仅当DOM加载完成，不包括样式表，图片，flash等等。

ie9以上才支持

如果页面的图片很多的话，从用户访问到onload触发可能需要较长的时间，交互效果就不能实现，必然影响用户的体验，此时用 DOMContentLoaded 事件比较合适

调整窗口大小事件

- window.onresize = function(){}

当触发时就调用的处理函数。

只要窗口大小发生像素变化，就会触发这个事件。

我们经常利用这个事件完成响应式布局。 window.innerWidth 当前屏幕的宽度

定时器

setTimeout()

window.setTimeout(调用函数，[延迟的毫秒数]);

该函数用于设置一个定时器，该定时器到期后执行调用函数

注：

- window可以省略
- 这个调用函数可以直接写函数，或者写函数名或者采取字符串'函数名()'三种形式，第三种不推荐
- 延迟的毫秒数省略默认是 0，如果写，必须是毫秒
- 因为定时器可能有很多，所以我们经常给定时器赋值一个标识符。

setTimeout() 这个调用函数我们也称为回调函数 callback

普通函数是按照代码顺序直接调用。

而这个函数，需要等待时间，时间到了才去调用这个函数，因此称为回调函数。简单理解：回调，就是回头调用的意思。上一件事干完，再回头再调用这个函数。以前我们讲的 element.onclick = function() {} 或者 element.addEventListener("click", fn); 里面的 函数也是回调 函数。

案例：5秒后自动关闭的广告

停止定时器

停止setTimer()定时器

window.clearTimeout(timeoutID)

该方法取消了先前通过调用setTimeout()建立的定时器

注意：window可以省略

里面的参数就是定时器的标识符

setInterval

`window.setInterval(回调函数,[间隔的毫秒数])`

该方法重复调用一个函数，每隔这个时间，就去调用一次回调函数

间隔的毫秒数省略默认是 0，如果写，必须是毫秒，表示每隔多少毫秒就自动调用这个函数。

第一次执行也是间隔毫秒数之后执行，之后每隔毫秒数就执行一次。

案例：倒计时

难点：计算目标时间与当前时间的时间差，以及时间差表示的具体格式

采用Date类型

停止setInterval () 定时器

`window.clearInterval(intervalID);`

该方法取消了先前通过setInterval()建立的定时器

注意：

- 1、window可以省略
- 2、里面的参数就是定时器的标识符

案例：发送短信

input中的text类型的禁用：`disabled = true`

设置setInterval（在中间这个间隔期不能点击发送）以及clearInterval（到时间后就可以继续发送了）

this

this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，一般情况下this 的最终指向的是那个调用它的对象

现阶段，我们先了解一下几个this指向

- 1、全局作用域或者普通函数中this指向全局对象window（注意定时器里面的this指向window）
- 2、方法调用中谁调用this指向谁
- 3、构造函数中this指向构造函数的实例

js执行队列

js是单线程

JavaScript 语言的一大特点就是单线程，也就是说，同一个时间只能做一件事。这是因为 Javascript 这门脚本语言诞生的使命所致——JavaScript 是为处理页面中用户的交互，以及操作 DOM 而诞生的。比如我们对 某个 DOM 元素进行添加和删除操作，不能同时进行。应该先进行添加，之后再删除。

这样所导致的问题是：如果 JS 执行的时间过长，这样就会造成页面的渲染不连贯，导致页面渲染加载阻塞的感觉。

同步异步

为了解决这个问题，利用多核 CPU 的计算能力，HTML5 提出 Web Worker 标准，允许 JavaScript 脚本创建多个线程。于是，JS 中出现了同步和异步。

同步：

前一个任务结束后再执行后一个任务，程序的执行顺序与任务的排列顺序是一致的、同步的。比如做饭的同步做法：我们要烧水煮饭，等水开了（10分钟之后），再去切菜，炒菜。

同步任务都在主线程上执行，形成一个执行栈

异步：

你在做一件事情时，因为这件事情会花费很长时间，在做这件事的同时，你还可以去处理其他事情。比如做饭的异步做法，我们在烧水的同时，利用这10分钟，去切菜，炒菜。

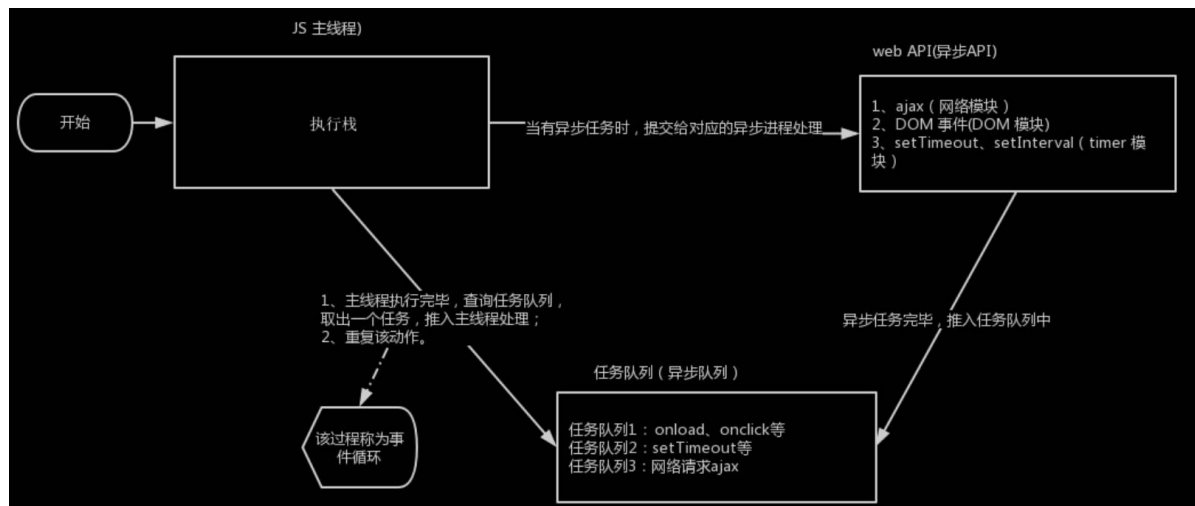
js的异步通过回调函数实现的。一般而言，异步任务有以下三种类型：

- 1、普通事件：click、resize等等
- 2、资源加载：load、error等等
- 3、定时器：包括setInterval、setTimeout等等

异步任务相关回调函数添加到任务队列中（任务队列也成为消息队列）

js执行机制

- 1、先执行执行栈中的同步任务。
- 2、异步任务（回调函数）放入任务队列中。
- 3、一旦执行栈中的所有同步任务执行完毕，系统就会按次序读取任务队列中的异步任务，于是被读取的异步任务结束等待状态，进入执行栈，开始执行。



由于主线程不断的重复获得任务、执行任务、再获取任务、再执行，所以这种机制被称为事件循环（event loop）

location对象

概述

window 对象给我们提供了一个 location 属性用于获取或设置窗体的 URL，并且可以用于解析 URL。因为这个属性返回的是一个对象，所以我们将这个属性也称为 location 对象。

URL

Uniform Resource Locator，URL统一资源定位符

是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的 URL，它包含的信息指出文件的位置以及浏览器应该怎么处理它。

一般语法：

protocol://host[:port]/path/[?query]#fragment

eg:

<http://www.itcast.cn/index.html?name=andy&age=18#link>

组成	说明
protocol	通信协议 常用的http,ftp,maito等
host	主机（域名） www.itheima.com
port	端口号 可选，省略时使用方案的默认端口 如http的默认端口为80
path	路径 由 零或多个'/'符号隔开的字符串，一般用来表示主机上的一个目录或文件地址
query	参数 以键值对的形式,通过 & 符号分隔开来
fragment	片段 #后面内容 常见于链接 锚点

location对象属性

location对象属性	返回值
location.href	获取或者设置 整个URL
location. host	返回主机（域名） www.itheima.com
location.port	返回端口号 如果未写返回 空字符串
location.pathname	返回路径
location. search	返回参数
location. hash	返回片段 #后面内容 常见于链接 锚点

案例：5秒后自动跳转页面

利用之前的定时器做倒计时效果

时间到了，跳转页面：location.href = '<http://www.baidu.com>'

location对象方法

location对象方法	返回值
location.assign()	跟 href 一样，可以跳转页面（也称为重定向页面）
location.replace()	替换当前页面，因为不记录历史，所以不能后退页面
location.reload()	重新加载页面，相当于刷新按钮或者 f5 如果参数为true 强制刷新 ctrl+f5

navigator对象

navigator 对象包含有关浏览器的信息，它有很多属性，我们最常用的是 userAgent，该属性可以返回由客 户机发送服务器的 user-agent 头部的值。

下面前端代码可以判断用户那个终端打开页面，实现跳转

```
if((navigator.userAgent.match(/(phone|pad|pod|iPhone|iPod|ios|iPad|Android|Mobile|BlackBerry|IEMobile|MQQBrowser|JUC|Fennec|WOSBrowser|BrowserNG|WebOS|Symbian|Windows Phone)/i))) {  
    window.location.href = ""; //手机  
} else {  
    window.location.href = ""; //电脑  
}
```

history对象

window对象给我们提供了一个history对象，与浏览器历史记录进行交互，该对象包含用户（在浏览器窗口中）访问过的URL

history对象方法	作用
back()	可以后退功能
forward()	前进功能
go(参数)	前进后退功能 参数如果是 1 前进1个页面 如果是-1 后退1个页面