# JavaScript简介

功能强大的语言，组成：核心(ECMAScript，核心语言功能)、文档对象模型(DOM，提供访问与操作网页内容的方法和接口)、浏览器对象模型(BOM，提供与浏览器交互的方法与接口)。

# 在HTML中使用JavaScript

<script>元素包含的属性以及写法，加载文件于顺序解析JS时，页面处理会暂时停止。标签放在<head>内，会加载完才处理页面显示，放在<body>后，就是显示完，才解析JS，<script>的defer属性会制定页面处理完才解析此处JS，就是延迟解析，async属性指定异步乱序加载，在XHTML中使用JS，特殊字符要转码，或者使用CData片段包含JS代码，文档模式很重要，影响页面的外观与JS的处理，简而言之就是代表版本的不同，分为混杂模式与标准模式。

# 基本概念

## 3.1 语法

ECMAScript都区分大小写，标识符类C，采用驼峰命名，注释类JAVA，支持严格模式，表明语法中的不确定行为会得到处理，”use strict”;相当于一条编译指令，指定JS引擎执行严格处理，语句末尾加”;”号，多条语句用{}括起来。

## 3.2 关键字和保留字

## 3.3 变量

松散类型，var声明，不初始化的话值为undefined，没有使用var定义的变量为全局变量，定义在window对象下。

## 3.4 数据类型

5种基本类型：undefined、null、Boolean、Number、String；一个复杂类型Object，由基本类型的无序键值对组成。Typeof操作符检测值得类型；null值表示空对象指针，跟undefined有本质区别，undefined是变量基本类型，null是对象空指针；Boolean只有true与false，Boolean()函数可以把其他类型转为true或者false，有值就是true，0，””,null,undefined,NaN等会转换为false，Number类型：整数、8进制与16进制；浮点数，科学计数法；数值范围，isFinite()检测是不是数值是不是超出范围了，NaN是Not a Number，任何涉及到NaN的计算都回得到NaN，isNaN()检测是否是有效数字，Number()转换为Number数据，parseInt与parseFloat()将字符串转换为数字；String类型跟java有些类似，toString()函数完成其他类型到String的转换，也可以使用拼接的方式；Object类型跟Java类似，使用new 创建，对象类型有一些内置的函数，跟java类似，比如：toString()，valueOf()，toLocalString()，constructor()，hasOwnProperty()，isPrototypeOf()，propertyIsEnumerable()。

## 3.5 操作符

一元操作符：++，--，+，-；

位操作符：~,&,|，^，<<，>>，>>>；

布尔操作符：！，&&，||；

乘性操作符：\*，/，%；

加性操作符：+，-，

关系运算符：<，>，<=，>+；

相等运算符：==，!=，===，!==；

条件操作符：？：

赋值运算符：=；

## 3.6 语句

With语句的作用是将代码的作用域设置到一个特定的对象中；

## 3.7 函数

Return返回任何类型的返回值，return;返回undefined；ECMAScript的函数参数个数不做规定，可以传任意个参数个数，类型也不限定，函数内部将所有参数组合成一个数组arguments；可以用它访问传递进来的参数，arguments[]数组对应位置的值与命名参数的对应位置的值始终保持一致，修改了一个，另外一个也会改变。

# 变量、作用域和内存问题

## 4.1 基本类型和引用类型的值

基本类型值是值类型，引用类型是引用的；与C语言里面类似；检测对象的类型使用instanceof操作符。

## 4.2 执行环境及作用域

执行环境代表了javaScript的运行环境上下文，定义了变量或者函数有权访问的的数据，执行环境是一种栈式的结构，每个环境都有一个保存所有数据的对象，就是栈帧，全局执行环境代表全栈即windows对象，每个栈在其执行完后都会销毁，整个栈形成一个作用域琏，栈顶变量作用域的最前端，当碰到一个名字时，将会沿着作用域连向栈底搜寻，作用域连延长的情况：try-catch语句与with语句，函数内可以访问with内定义的变量。Javascript不存在块级作用域，定义的变量隶属于其所属的栈帧整个栈内，就是其最接近的环境；最小的栈就是函数了，所以不存在块级作用域。如果定义时没有使用var声明，会被添加到window对象的属性中，变为全局。

## 4.3 垃圾收集

执行环境负责回收内存，跟Java类似；回收策略：1.标记清除，进入环境时标记，离开环境时标记；2.引用计数，循环引用问题无法解决；IE中：BOM与DOM对象的垃圾回收机制就是引用计数，所以还是存在循环引用问题；垃圾收集器是周期运行的，在IE7以下，当系统中有256个变量、4096个字面量、或者64KB的字符串，垃圾收集器就会运行，这种方式性能低下，后来加入了内存占比考虑，当触发时，内存占比少于15%，则触发数量加倍；一般给Web浏览器分配较少的内存，为了运行速度的原因，一般一个变量不在使用，最好将值设为null，适用于全局变量；局部的可以自动释放，这是方便垃圾收集器下次回收。

# 引用类型

引用类型(类)的实例是对象，是一种数据结构，新对象使用new创建，后跟构造函数创建；常见的引用类型是Object，还有其他很多的引用类型。

## 5.1 Object类型

Object是最基本的引用类型，里面有很少的属性。构建Object的实例有2种方式：new 构造函数(),对象字面量（JSON表示法，不会调用构造函数）；访问Object内的属性与方法，点号表示法，方括号表示法。

## 5.2 Array类型

Array类型里面元素类型可以不同；构建方式：1.[new ]Array(),new可有可无，可以传递数组个数参数或者值参数；2.数组字面量表示法，[value1,value2,,,,]；最后如果没有值得话，浏览器不会解析最后一个逗号，在IE8及以前会把最后一个逗号后面的值解析为undefined；适用方括号[]引用元素，数组能够动态增长，length表示其长度，可以自由设置，相应的数组就变为对应的长度。

Instanceof能够检测对象是不是数组；更新版本的浏览器支持Array.isArray()方法检测数组。

每个对象都有toLocalString()，toString()，valueOf()3个方法，toString()返回逗号分隔字符串，默认在字符串的地方用到对象时，对象调用toString()方法输出自己，toLocalString()显示调用时，将会输出分割的每个元素的toLocalString()方法结果；调用join(delemitor)会改变分割符，不传入分割符默认还是逗号。

数组可以表现的像栈一样，Array提供了pop()与push()方法。Push()追加在末尾返回数组大小，pop()弹出元素，返回元素。

数组也可以模拟队列，shift()移除数组第一项并返回，与push()配合达到队列，unshift()在数组前端添加元素返回长度，与pop()形成反向的队列。

排序方法：reverse()反转；sort()，默认是数组的元素转换成字符串排序，所以排数值是不准的，sort()内可以传入函数对象，由函数对象定义比较规则，第一个<第二个，返回负数……。

操作方法：concat()，拼接传入的值，返回新的数组；slice(),相当于javaString中的substring函数；splice()函数完成删除与插入。Indexof(value,startPosition)与lastIndexOf()查找数据。

迭代方法：every():数组中每一项运行给定函数，如果每一项都返回true，返回itrue；filter():数组中每一项运行给定函数，返回处理函数返回true的数组元素；foreach():数组中每一项运行给定函数,没有返回值；map():数组中每一项运行给定函数,返回处理函数的处理结果组成的数组；some():数组中每一项运行给定函数，如果有一项都返回true，返回itrue；这些方法传入的函数对象都是3个参数的分别是：数组元素值、位置、数组本身。

## 5.3 Date类型

Date类型是在java.util.Date的基础上创建的，使用自1970年1月1日零时的毫秒数保存时间；new Date()区当前时间；new Date(毫秒数)；直接输入毫秒数，开发人员需要计算毫秒数，比较麻烦，提供了2个方法：Date.parse()；里面是时间的字符串表示，支持常见的时间字符串的毫秒数解析；Date.UTC()，里面传入年、月份、日期、小时、分钟、秒及毫秒等参数。Date.now()当前时间的毫秒数；Date类型的对象重写了toString()方法、toLocalString()方法与valueOf()方法，toString()返回带有时区信息的日期时间、toLocalString()不包含时区信息，valueOf()方法返回毫秒数。Date类型还有其他的方法。

## 5.4 RegExp类型

支持正则表达式，语法：

var expression=/pattern/flags；每个pattern是正则表达式字符串、flags是正则表达式的行为：g:全局，i:忽略大小写，m:多行模式，要匹配字符串中的元字符，需要转义，也可以用构造函数定义new RegExp(pattern,flags)；使用构造函数时，参数的字符串中的元字符需要经过双重转义；2种方式额不同是：字面量的形式只会创建一个实例，new的每次都会创建一个全新的。

RegExp的每个实例有以下属性：global(是否设置了g标志)、ignoreCase(是否设置了i标志)、lastIndex(开始搜索匹配项的起始位置)、multiline(是否设置了m标志)、source(正则表达式的字符串表示，不包含falgs)。

RegExp的exec(stringMatched)方法实现字符查找，没有匹配项返回null，有的话返回一个数组，数组是Array实例，数据的第一项是匹配的字符串，后面是捕获组（看书中的例子）匹配的字符串，还包含2个属性：index匹配项在字符串中的位置，input则是匹配的字符串。在没有设置g标志时，多次调用返回第一个匹配项，lastIndex保持不变（IE不是这样的），设置了g标志，多次调用会一直向后搜索。Test(string)判断是否匹配。ToString()与toLocalString()会返回包含表达式与flags的完整字符串。

RegExp的构造函数包含一些属性，从类的角度看，这些属性相当于类的公有的静态属性。作用于所有的正则表达式，根据最近一次的正则表达式执行情况而变化，这些属性有2个名字；input($\_)最近一次匹配的字符串、lastMatch($&)最近一次的匹配项、lastParen($+)最近一次匹配的捕获组、leftContext($`)、multiline($\*)、rightContext($’)；还有9个用于存储捕获组的属性，分别用$n表示。

## 5.5 function类型

函数实际上是对象，是Function的实例，具有属性与方法，函数名是函数的指针，2种方式定义；还有一种对象的定义方式：

var functionPointer=new Function(arg1,arg2…function body);

没有重载；重载的话，前面的会被覆盖，因为相当于后面又赋了一遍值。

函数声明与函数表达式会存在一定的区别，当先使用函数，后定义时，函数声明会在代码加载阶段进行函数声明提升操作，把函数声明提到最前面，但是函数表达式相当于指针赋值，是在运行期才会得到函数对象，所以预先调用是会报错的。

函数对象本身是一个值，可以作为参数传递，也可以作为结果传递；

函数有2个特殊的属性：arguments与this，arguments保存参数，还拥有一个callee的属性只想拥有arguments的函数；this：调用函数执行的环境对象，全局函数下就是window对象；caller：调用此函数的上一个函数的指针；

函数包含2个属性：length与prototype；length表示函数的命名参数的个数；prototype名下保存引用类型真正的实例方法；

函数包含2个方法：apply(this，参数或数组)与call(this,arg1,arg2….)；作用是在特定的作用域下调用函数对象本身，this是执行函数的上下文对象；this可以传递任何对象，此时函数内如果访问this，就是传进来的那个this对象。将方法与对象之间进行了解耦。Bind(this)函数也有同样的功能。

toString()、toLocalString()、valueOf()返回函数代码。

## 5.6 基本包装类型

Boolean、Number、String是基本类型的包装类型，在使用这些基本类型值得地方，都会临时创建一个对应的包装引用类型对象来支持一些对象的操作；由于这个对象是执行时临时产生的，所以基本类型值得添加属性与和方法的行为只是对临时对象进行了操作，操作完成后，这个对象就销毁了，所以添加是不成功的；可以用new，创建包装类型对象，new Object(typevalue),也会根据参数值返回对应的包装类型对象；new Number(“25”)与Number(“25”)是不一样的，转型函数返回基本类型值，new的返回对象。

Boolean的类型会造成一定的BUG，Boolean && true ，无论Boolean对象是true还是false都回返回true，因为对象在转为布尔值时都是转为true的。建议不要使用Boolean对象。

Number是数字值的引用类型，new Number(value) 创建Number对象，重写了valueOf()，toString()，toLocalString()等值；toFixed()返回指定小数位数的字符串格式，toExponential()返回指数形式的字符串，toPrecision()返回数值的最合适表达形式。

String是字符串的引用类型，length属性表明长度，具体的方法可以看书。

## 5.7 单体内置对象

内置对象就是ECMAScript语言定义的对象，这些对象不依赖于浏览器，程序执行前就已经存在的对象，前面的都属于内置对象，还有Global与Math等，global对象是一个终极对象，是最高层级的，所有没归属的属性与方法都是这个对象内的内容，很多我们觉得内置的方法都是global对象的方法，此外还有：1.URI编码方法-encodeURI()与encodeURIComponent()可以对URI进行编码，用UTF8编码替换所有无效的字符（比如空格），然后发送给浏览器；encodeURI()只转化不符合URI的特殊字符编码，可以对这个URI进行编码，encodeURIComponent()对所有特殊字符进行编码，所以只能对特定的部分进行编码，典型的就是/字符；对应的还有解码方法：decodeURI与decodeURIComponent()；

eval()方法：一个完成的ECMAscript解析器，接受一个javascript语句编写的命令字符串；解析后，把执行结果插入到eval调用所在的位置，里面的字符串语句与调用eval的环境内作用域相同，就是说参数内与外部作用域的变量与对象可以互相引用。

所有原生引用类型的构造函数都是Global对象的属性。在浏览器中，globa对象就是window对象，同时window对象又加入了一些其他的功能。

还有一个全局对象Math，里面有很多跟数学相关的属性与方法；

# 面向对象的程序设计

Javascript中的对象就相当于java中的散列表，是无序属性的集合。创建对象以前是new Object()然后添加属性，现在是字面量的方式（JSON）。

## 6.1 理解对象

ECMAScript用于描述属性的特性用双方括号标识[[enumerable]]，这表示这些特性是内部值，是用于JavaScript引擎使用的。

JavaScript有2种属性：数据属性与访问器属性。

数据属性是一个数据值，有4个属性：[[Configurable]]表示属性能否修改，重新定义（可配置）等，一般为true；[[Enumerable]]能否通过for-in循环返回属性；[[Writable]]表示能否修改属性的值，[[Value]]属性的数据值，默认值是undefined。

Object.defineProperty(ObjectName,propertyName,{writable:false,value:…}(例子))用于修改对象属性的特性，参数分别是对象、属性以及描述符对象；如果configurable:false设定后，属性变成不可配置的，相当于变成只读属性，同时

Delete object.property操作也会失败；也不能再变回可配置的。非严格模式下会忽略，严格模式下会抛出异常。

访问器属性不包含数据值，包含一对setter、getter函数，有4个属性：[[Configurable]]表示属性能否修改，重新定义（可配置）等，一般为true；[[Enumerable]]能否通过for-in循环返回属性；[[Get]]读取属性时调用的函数，默认undefeined，[[Set]]写入属性时调用的函数，默认值是undefined。

Book={\_year:2014,edition:1}\_的属性表示属性只能通过对象方法访问的属性，

Object.defineProperty(book,”year”,{get:function(){return this.\_year;})定义了一个访问器属性year,用于访问\_year的值。这个特性加入以前，是用的\_defineGetter\_与\_\_defineSetter\_\_方法实现的访问器属性。

Object.defineProperties定义一个对象多个属性的特性，例子可以看书。

Object.getOwnPropertyDescriptor(ObjectName,ObjectProperty)方法可以返回属性的描述符对象。

## 6.2 创建对象

new Object()与{}的方式创建对象会产生重复代码。

1. 工厂模式：就是一个函数，里面可以return Object，可以根据参数生成特定的Object。可以解决创建多个相似对象的问题，但是不能识别对象类型。
2. 构造函数模式：构造函数是工厂模式的进一步发展，可以构造特定类型的对象，instanceof时有用，相当于java中的类了；比如:

Function Person(name){

This.name=name;

}

里面没有显示的创建对象，属性与方法直接付给this对象，没有return语句，有点类似于匿名的函数对象的感觉。new Person(“”)构造函数创建对象经历4步：(1)创建一个新对象，(2)将构造函数的作用域赋给新对象，(3)执行构造函数的代码，(4)返回新对象。对象的constructor属性指向了构造函数；构造函数与普通函数没什么区别，可以一样当成普通函数使用；缺陷：里面的方法都是独有的，会制造出很多执行同样过程的方法对象，如果把方法定义在外部在内部通过变量引用，又破坏了实例的封装性，全局下要定义很多这种方法。

1. 原型模式：每个函数都有一个prototype属性，一个指针，指向一个对象，这个对象包含特定类型的实例共有的属性与方法；类似于java中的Class类，里面的属性类似于类属性，在javascript中，这个叫做原型对象（Class，类）。

原型模式：每个函数都有他的属性prototype指向原型对象，里面有个constructor属性，指向prototype属性所在的函数；函数里可以继续向prototype里添加属性；原型对象默认只有constructor属性，其他的继承与Object，JavaScript中用[[ProtoType]]这个特性描述对象（具体实例，前面的prototype是构造函数内的内的原型对象，非IE浏览器在所有对象内部都内置一个\_\_proto\_\_属性，用于存储原型对象，这个属性连接的是对象与原型对象，不是对象与构造函数。一般无法访问到对象内的[[Prototype]]，可以用isPropertyOf(Object)函数确定调用者是不是参数内对象的[[Prototype]]，Object.getPropertyOf()返回[[Prototype]]。

代码读取对象时，都要搜索对象里面的内容，如果是对象里面的属性，返回属性值，没找到则到原型对象里面去找。

对象内的属性会屏蔽掉原型对象内的同名属性，对象属性的修改值或者添加都不会影响原型对象内的属性。

hasOwnProperty()检测一个属性是不是存在于实例中，给定属性存在于实例中才会返回true。

in操作符不仅用于for-in循环，也可以单独使用，属性在给定的对象或者原型里面时，返回true。

FunctionName.prototype={}，可以用定义实例常量的方式定义对象原型，这种方式创建原型后，constructor属性不再指向函数，此时一般指向Object（新对象的constructor属性），可以手动添加。

原型具有动态性，针对原型的任何添加修改，实例都能马上使用，因为原型只有一份，是通过指针指向的；但是如果创建实例后，重写原型对象，那么实例内的原型指针还是指向原来的原型，而不是新的原型。

上文中很多原生对象类型就是通过构造函数+原型的方式定义的。原型模式也有一些缺点，比如共享原型属性的实例是引用类型的，那么针对这个属性的修改会反映到所有实例上，导致创建的实例同质化。

创建自定义类型的常见方式是组合使用构造函数模式与原型模式；构造函数用于定义实例属性，原型模式用于定义方法与共享属性，保证了自己的独特，又共享了相同的部分。

动态原型模式允许将原型的更改定义放到构造函数中进行。

寄生构造函数模式与工厂模式类似。

稳妥构造函数模式。

## 6.3 继承

OO语言支持2种继承方式：接口继承与实现继承；JS只支持实现继承。依靠原型链的方式实现。

### 6.3.1 原型链

主要思想是利用原型让一个引用类型继承另一个引用类型的属性与方法；定义就是让原型对象等于一个对象的实例，那么对象->原型对象（实例）->原型对象(实例)，就形成了一个链条，这个就是原型链；这种继承的本质要给原型对象重新赋值。

所有的引用类型都继承了Object，就是通过原型链的方式实现的，所有函数默认的原型就是Object的实例，会包含一个指向Object.prototype原型的指针，所以会继承来一些默认的方法。

原型与实例之间的关系可以用instanceof判断，只要是实例继承的原型，都会返回true，或者isPrototypeOf(instance)也可以判断。

使用原型链实现继承时，不能使用对象字面量定义原型。

### 6.3.2 借用构造函数

# 函数表达式

函数对象都有一个name属性，用于等于函数的名字；函数声明的一点就是函数声明提升，就是调用可以发生在声明前；函数表达式也可以创建函数，创建的函数叫做匿名函数；函数声明类似于编译期确定函数，函数表达式是执行期确定要执行的函数；匿名函数对象也可以作为返回值。

## 7.1 递归

递归时，函数名是确定的，但是如果后面将函数赋给其他的变量，那么此时执行递归函数可能会出现错误，会找不到函数，此时可以使用arguments.callee这个指针执行递归，因为它总是指向当前正在执行的函数。

## 7.2 闭包

定义：有权访问另一个函数作用域中变量的函数；常见的方式就是在函数内部创建另一个函数，此时创建的函数可以访问外部函数的所有的属性；及时这个创建的函数被返回给上层，这个内部函数的作用域也是包含外部函数的作用域的。

# BOM

ESMAScript是JavaScript的核心，BOM提供对象用于与浏览器交互。

## 8.1 window对象

Window代表一个浏览器实例，也是JavaScript中所有对象的Global对象，全局作用域中声明的变量或者函数都是window对象的属性，定义全局变量与定义window属性一点不同：全局变量不能通过delete删除，而且访问未声明的全局变量会抛出错误，而访问不存在的window属性则不会抛出错误，返回undefined。

页面中包含框架，每个框架都代表一个window对象，子框架的window对象保存在父window对象的frames数组中，按照从左到右，从上到下；可以通过索引或者框架的名称（window的name属性）来访问数组；top，永远指向最外面最上层的window对象，parent用于指向子window对象的父window对象，当只有一个frame时，2个相等；除非窗口是通过window.open()打开的，否则window的name属性没有值；self对象始终指向window对象，所以window对象之间是一种树的关系，子节点还保存父节点对象的地址；注意：每个框架中的window对象都包含各自的原生类型的构造函数，所以，同名的构造函数构造的对象在不同的window对象是不相等的。

大多数浏览器都提供了screenLeft与screenTop，表示可见页面相对于屏幕的左界与上界的距离；firefox则提供screenX与scrennY提供这些信息；在非IE浏览器中screenLeft（screenTop）保存的都是浏览器窗口到屏幕左侧的距离；多个框架时，IE浏览器会返回每个框架相对于边界的值，在其他浏览器中会返回top.screenX与top.screenY的值，moveTo与moveBy可以将浏览器窗口位置移到新的位置。

浏览器大小的4个属性：innerWidth，innerHeight、outerWidth、outerHeight;一般来说，outer返回的带浏览器窗口的值，inner是页面视图区的大小，具体浏览器还有不同；DOM取得视图区大小的方法(支持移动浏览器)

document.documentElement(body).clientWidth(clientHight)

document.documentElement(body).clientWidth(clientHight)

在移动浏览器下，body的大小是手机屏幕的大小，documentElement的大小是实际页面的大小；resizeTo与resizeBy方法调整窗口大小。

window.open()方法打开一个URL，4个参数：URL、目标窗口框架名字、特性字符串、关于历史记录的布尔值；其中目标窗口框架还可以是：\_self,\_parent,\_top,\_blank；如果不是一个已经存在的窗口或者框架，那么会根据低个参数打开新的窗口或者页签，如果没有第三个参数则是浏览器的默认设置；第三个参数是一个逗号分隔的设置字符串；有哪些可以设置可以看书；open()方法会返回新窗口对象window的引用；top.close()关闭自己，关闭了引用还在，可以检测closed状态；opener属性保存打开它的原始窗口对象；只在top类型中的window对象中有定义；设置opener=null，会让2个页面运行在独立的进程中。

即使在第三个参数里设置了很多参数，但是很多浏览器也会根据自身的安全限制让一些参数失效；

弹出窗口屏蔽程序，如果是浏览器内置的屏蔽程序，那么window.open()的返回值是null。如果是拓展程序，window.open()会抛出异常。

JavaScript是单线程的语言，可以通过设置超时值与间歇时间来调度代码在特定的时刻执行；前者指在指定的时间过后执行代码，后者指每隔指定的时间就执行一次代码；超时调用的函数是setTimeout()方法，到了指定时间后，不一定立即执行，因为是单线程执行，所以会保存一个任务队列，在空闲的时间取任务队列来执行；setTimeout返回一个ID，表示任务地址，clearTimeout(ID)可以在未执行前取消这个任务；间歇执行的代码是setInterval()，递归的超时调用可以代替间歇调用。

alert()、confirm()、prompt()、print()、find()一些跟操作系统有关的对话框。

## 8.2 location对象

Location对象提供当前文档的信息与URL信息，既是window的属性也是document的属性；它的很多属性可以看书，location对象可以更改URL并立即生效，location.assign(url)，立即打开新的URL，并生成历史记录；直接为location.href或者window.location设置一个URL，默认也是调用assing函数的；修改lcoation的其他属性也会直接触发页面加载，reload()会重新加载页面。

## 8.3 navigator对象

通常用来检测浏览器或者操作系统的一些信息，

## 8.4 screen对象

Screen对象表示客户端的能力，包含一些与外部的显示器相关的信息，一些属性可以详细看书。

## 8.5 history对象

History对象保存用户的历史记录，go()、back()、forward()等方法跳转页面。

# 第十章 DOM

# 第二十一章 Ajax与Comet