**一、基本概念**

（1）语法

1、Js中的一切（变量，函数名和操作符都区分大小写）；

2、标识符第一个字符不能是数字，后面的可以是字母，数字，下划线，$。标识符命名变量最好是字母间用下划线分隔，函数用驼峰命名法，即首单词首字母小写，后面的单词首字母大写；

（2）变量

用var操作符定义的变量将成为该变量作用域中的局部变量。不用var声明，该变量会归window所有。

（3）数据类型

1、Js中有5种简单数据类型：Undefined，Null，Boolean，String，Number；一种复杂数据类型：Object。null其实也属于object，因为null被认为是一个空对象的引用。Undefined类型只有undefined一个值。

**Number类型：**

2、16进制的前两位是0x。js会将超出数值范围的值自动转换成Infinity或-Infinity，Infinity不能参与数值计算，可以用isFinite(x)来确实值是否有穷。

3、在js中NaN是用来表示本来要返回数值的值未返回数值的情况可以避免报错。任何涉及NaN的操作都会返回NaN，NaN与任何数都不想等包括NaN本身。采用isNaN（x）可以判断x是否是NaN。

4、Number（），parseInt（），parseFloat都可以把非数值转换为数值，Number函数可以用于任何数据类型，转换规则较多可参阅P30，后面两个专门用来把字符串转换为数值。三个方法都归window所有。对于不符合数值格式的字符串返回NaN。

5、3.2e7表示3.2\*107

**String类型：**

6、\加一些字母表示转义字符，例如\n表示换行

7、将一个值转换为字符串有两个方法。toString()方法，除了null和undefined类型没有这个方法别的值类型都有。String()方法，每种数据类型都可以用，null和undefined会被转换为字符串形式的null和undefined。

**Object类型：**

8、object类型是所有它的实例的基础，object类型所具有的任何属性和方法也同样存在于更具体的对象中。后面会详细介绍这个类型。

（4）操作符

**一元操作符：++,--,+,-**

1. 对非数值类型使用该操作符时会进行隐性Number函数转换

**位操作符：~(按位非)，&(按位与)，|(按位或)，^(按位异或)，<<(左移)，>>(有符号右移)，>>>(无符号右移)**

1. 位操作符就是将数值转换为二进制格式再执行操作。例如~20，对非数值采用位操作符依然会进行number函数转换。详情请查阅P41

**布尔操作符：!(逻辑非)，&&(逻辑与),||(逻辑或)**

1. 对于非布尔值采用逻辑非会先利用Boolean函数将其转换为布尔值在进行操作。
2. 逻辑与与逻辑或都是短路操作，对于逻辑与会先判断第一个数是不是true，如果不是则返回第一个数否则返回第二个；对于逻辑或第一个数是true就返回第一个数否则返回第二个。

**乘性操作符：\*，/，%，p48**

**加性操作符：+，-，p49，注意+当有一个数值是字符串时会进行字符串拼接**

**关系操作符：>,<,<=,>=，返回布尔值**

**相等操作符：!=,==（相等与不等先转换数据类型再比较）；!==,===(全等与不全等只比较不转换)**

**条件操作符：d=a?b:c，就是三元操作符**

**赋值操作符：=，可以与\*，/等配合使用，例如\*=**

**逗号操作符：**

5、多用于声明多个变量不过也可以用来赋值。在赋值时，逗号操作符总会返回表达式中的最后一项，例如var num = (1,3,4,5)，num值为5，这里用()将其变为了表达式

（5）语句

**for-in语句:用来枚举对象的属性**

1. 例如for (var a in window){document.write(a)},注意for-in输出的属性名是没有顺序的

**with语句：将代码的作用域设置到一个特定的对象中**

1. 例如var a = location.name,b=location.url可以用with简写为with(location){var a = name,

B = url}

**switch语句:非常适合判断多个判断条件，swtich(a){case 1:..break;…default:…}**

（6）函数

1、js中使用function关键字来声明函数的返回值在不定义的情况下为undefined，执行完return语句会立刻退出。

2、js中函数的参数在内部是用一个类数组来表示的，通过arguments对象可以访问这个参数数组，即第一个元素是arguments[0]以此类推，因此js其实可以不指定函数参数数量，只管传入到时用arguments来调用即可。

3、js中函数没有重载，如果定义了两个相同名字的函数，则该名字只属于后定义的那个。

4、函数的本质是特殊的对象。

**二、变量，作用域和内存问题**

1、js中有两种数据类型的值，一个是基本类型，一个是引用类型。Undefined，null，Boolean，number，string都是基本类型别的都是引用类型，其实也就是object。两者最明显的区别就是基本类型的复制两者是独立的，而引用类型其实传递的是一个指针地址，因此对其中任何一个的修改对会反应在引用对象身上。另外基本类型存储在栈内存上，引用类型在堆内存上。

2、确定一个引用类型值属于哪种对象可以用instanceof，基本类型就是typeof

3、所有变量（包括基本类型和引用类型）都存在于一个执行环境（也称为作用域）当中，这个执行环境决定了变量的生命周期，以及哪一部分代码可以访问其中的变量。

（1）执行环境有全局执行环境和函数执行环境之分。

（2）每次进入一个新执行环境，都会创建一个用于搜素变量和函数的作用域链

（3）函数的局部环境不仅有权访问函数作用域中的变量，而且有权访问其包含（父）环境，乃至全局环境。

（4）全局环境只能访问在全局环境中定义的变量和函数，而不能直接访问局部环境中的任何数据

（5）某个执行环境中的所有代码执行完毕之后该环境被销毁，保存在其中的所有变量和函数定义也随之销毁（全局执行环境直到应用程序退出例如关闭网页时才会被销毁）。

4、js具有自动垃圾收集机制

**三、引用类型**

1、对象是某个特定引用类型的实例，新对象是使用new操作符后跟一个构造函数来创建的，构造函数本身就是一个函数，只不过该函数是出于创建新对象的目的而定义的。

**object类型**

1. object类型创建要么直接new Object()或者对象字面量法即var a= {}，对象字面量法本质上就是new Object()，访问对象属性时一般使用点表示法，不过js也可以用[]，例如a[name],方括号里要是字符串，方括号访问的好处是可以使用变量来访问。一般是使用点表示法。

**Array类型**

1. js中的数组每一项都可以保存任意类型的数据且数组长度是可以动态调整的，可以随着数据的添加自动增长以容纳新增数据。
2. 创建数组也是两种方法，一是Array()构造函数，参数可以是一个数字代表数组长度或者直接是数组内容；一个是字面量即var a =[]。
3. 数组的length属性不是可读的，即设置它可以删减数组项，增加数组长度，新增项为undefined，例如a长度为3，直接a.length = 2就会删除最后一项。也可以直接设置超出当前数组大小的下标值，数组会重新计算长度值，例如a数组原本长度为3，a[100]=”he”。
4. 对数组调用toString()方法本质上是对数组中的每一项调用toString()方法，返回数组中每个值的字符串形式拼接而成的以逗号分隔的字符串。
5. push()方法，接收任意数量参数逐个添加到数组末尾，返回修改后数组长度；unshift()方法，跟push差不多不过是从数组前端推入。pop()方法，从数组末尾移除一项，返回该项；shift()方法，从数组前端移除一项，返回该项。join(“,”)方法，只接收一个参数即用作分隔的字符串，然后返回包含所有数组项的字符串。
6. reverse()方法，反转数组。sort()方法，默认是按升序排列数组不过本质上的排列规则是将每一项转换为字符串再比较字符串因此并不理想，不过sort方法可以接收一个比较函数作为参数，根据函数返回值来确定是否调换次序。
7. concat()，不传参时只是返回当前数组的副本，如果传递一个或多个数组会将他们的内容添加到数组末尾并返回新数组。slice()，接收一个或两个数字参数一个就是从该参数到最后一项，两个就是包括头不包括尾，如果结束位置小于开始位置就返回空数组。两个方法都不改变原数组。splice()方法，数组最强大方法，主要用途是向数组的中部插入项，具体规则参见p95，该方法会改变原数组。
8. indexof(),lastIndexof()，两者的区别是前者从数组头开始查找，后者从数组尾开始查找，可以接受两个参数要查找的项和（可选的）表示起点位置的索引。没找都就返回-1。
9. 迭代方法，every(),filter(),forEach(),map(),some();归并方法，reduce(),reduceRight()，详情参见p97。filter,map都会对数组的每一项调用函数，filter方法返回函数判断条件后为true的项，因此筛选后组成的数组还是原来中的一些项，map方法则是将函数返回的结果组成新数组。reduce方法一般是用来归并数组的，即对所有数组值按某种关系求值。例如nums.reduce(function (pre,n){return pre+n},0),这样会返回nums数组里的各项加起来的和，0的作用是对第一次回调的pre进行初始化，n为数组的每一项，前一次回调的pre+n会作为下一次回调的pre。

**Date类型：详情参见p98**

**RegExp类型：即正则表达式对象**

1. 创建正则表达式，一是构造函数new RegExp()，接收两个参数一个是要匹配的字符串模式一个是可选的标志字符串。另一个是字面量定义，var express = /pattern/flags;其中pattern就是正则表达式，flags即标志，可以有一个或多个，例如g，i，m，参见p103。
2. RegExp类型具有很多属性取得有关模式的各种信息，例如global，source，lastIndex等，参见p105。捕获组方法exec()，参见p106；测试方法test()，接收字符串参数判断是否符合正则表达式模式，返回布尔值

**Function类型**

1. 函数实际上是对象，每个函数都是Function类型的实例，与其他引用类型一样具有属性和方法，因此函数名实际上也是一个指向函数对象的指针，不会与某个函数绑定。
2. 创建函数使用函数声明与函数表达式对于解析器来说是不一样的，对于函数声明解析器会率先读取并使其在执行前可用，而函数表达式必须要等到解析器执行到它所在的代码行，才会被真正解释执行
3. 函数内部属性，函数内部有两个特殊对象为arguments和this，arguments对象还有个callee属性，该属性是一个指针，指向拥有这个arguments对象的函数，在使用递归时很好用，即使改名也没关系。this引用的是函数执行的环境对象，即函数在哪个环境中执行的，this就指向拥有这个环境的对象。函数对象还有个caller属性，保存着调用当前函数的函数的引用，如果是在全局中调用就为null。
4. 因为函数是对象所以拥有属性和方法。属性：length，表示函数希望接受的命名参数个数。prototype,是个对象都有后面会详细介绍。方法：apply()和call()，这两个方法的作用都是在特定的作用域中调用函数，本质上就是设置函数体内this的值。两者都接收两个参数，第一个是在其中运行函数的作用域（是个对象），第二个参数函数如果有参数就需要添加，其中apply()方法是要么是arguments对象要么是Array实例，而call()方法则是需要将参数逐个列举出来。
5. Es5还定义了一个bind()方法，也有差不多作用，p118

**基本包装类型**

1. 基本包装类型与引用类型类似，有Boolean，string，number三种，有自己的属性和方法，每当读取一个基本类型值的时候后台就会创建一个对应的基本包装类型的对象，从而让我们能够调用一些方法来操作这些数据。
2. 基本包装类型与引用类型最大的区别就是对象的生存期，使用new操作符创建的引用类型的实例在执行流离开当前作用域之前一直都保存在内存中，而自动创建的基本包装类型的对象则只存在于一行代码的执行瞬间然后立即被销毁。
3. 创建基本包装类型直接new相应构造函数即可，不过一般情况下不需要。
4. 字符方法。charAt和charCodeAt，接收一个基于0的字符位置参数，返回相应的字符，字符编码。
5. 字符串操作方法，slice(),substring(),substr(),这三个方法都会返回被操作字符串的一个子字符串，接受两个参数，第一个是起始位置，第二个可选参数对于前两个方法是结束位置，当然包括头不包括尾，而对于substr是截取个数，不设置都以末尾作为结束位置，都不改变原字符串。
6. 字符串位置方法，indexof和lastIndexof跟数组的一样不做赘述。
7. trim()方法，创建字符串副本，删除前置及后缀的所有空格，然后返回结果。
8. 大小写转换，toLowerCase,toUpperCase
9. 字符串模式匹配，match(),search(),replace(),split()参见p127；比较两个字符串，localeCompare()，如果小于返回-1，相等返回0，大于返回1。

**单体内置对象**

1、在所有代码执行之前，作用域中就已经存在两个内置对象Global和Math,在web浏览器中window对象包括包括了global对象，即可以用window对象来代替global对象

Global对象

1. URI编码方法，可以对uri（通用资源标识符）进行编码，encodeURI和encodeURIComponent，p131
2. eval()方法，接受一个参数要执行的js字符串

Math对象

1. 属性，包括Π，自然对数的值都有p134。
2. 方法，min()和max()接收任意多个数值参数返回最大值，最小值；舍入整数方法，ceil(),向上舍入。floor()，向下舍入。round()，标准舍入，.5向上舍入；随机数方法，random()返回大于等于0小于1的一个随机数。当然还有些绝对值啥的，p136。

**四、面向对象的程序设计**

理解对象

1. js中有两种属性，数据属性和访问器属性。数据属性包含一个数据值的位置，在这个位置可以读取和写入值。数据属性有四个描述其行为的特性，Configurable,Enumerable,Writable,Value，详见p139。访问器属性不能直接定义，详见p141。两种属性都跟Object.defineProperty()方法有关，p141。
2. 定义多个属性也可以利用Object.defineProperties()方法，p142
3. 读取属性特性，Object.getOwnPropertyDescriptor()，p143

创建对象

**工厂模式**

1. 工厂模式就是定义一个函数，在函数中创建对象并赋予属性方法再返回该对象，每次想用新对象就调用函数即可。明显的问题就是没有解决对象识别的问题，即只知道返回了个对象，但却看不到它的类型。

**构造函数模式**

1. 构造函数模式，就是使用new操作符配合函数来使用，在函数中使用this来定义属性和方法，new实际上相当于创建新对象并将函数作用域赋给它然后执行函数代码返回新对象。另外，规定是构造函数首字母应该大写，虽然不强制不过有利于区分开来普通函数，不new的话其实跟普通函数就是一样的。构造函数的问题就是如果对象中包含方法，事实上方法是对象而这样每调用一次就会创建新对象因此对于内存并不友好。当然工厂模式也有这问题。

**原型模式（敲黑板了）**

（1）理解原型

1. 原型模式就是在函数的原型上去定义属性和方法
2. 我们创建的每个函数都有一个prototype(原型)属性，包括Array，Object这种构造函数，这个属性是一个指针，指向一个对象。这个对象的用途是包含可以由特定类型的所有实例共享的属性和方法，即通过该构造函数创建的对象。
3. 无论什么时候，只要创建了一个新函数，就会根据一组特定的规则为该函数创建一个prototype属性，指向函数的原型对象。在默认情况下，所有原型对象都会自动获得一个constructor属性，这个属性是一个指向prototype属性所在函数的指针。当调用构造函数创建一个新实例后，该实例的内部将包含一个指针指向构造函数的原型对象，es5叫它[Prototype]，这个连接存在于实例与构造函数原型对象之间，而不是实例和构造函数。
4. 每当代码读取某个对象的属性时，都会执行一次搜索，目标是具有给定名字的属性，搜索首先从实例本身开始，如果找到该属性就返回值。如果没找到，就继续搜索[Prototype]指针指向的原型对象，如果在原型对象中找到该属性就返回值。
5. 当我们在实例中添加一个属性且跟原型对象中的属性同名时，原型中的属性会被屏蔽，即原型中的该属性并不会变只是并不会被访问到，其实跟第四条一样因为先从实例本身搜索到了该属性，即使这个属性设置为null，不过使用delete可以删除实例属性，从而让我们可以访问到原型中的。

（2）原型与in操作符

1、单独使用in操作符可以判断该属性是否存在于实例或原型中，只要能够访问到就返回true。在利用for-in时，会枚举实例以及原型中所有的属性。

（3）更简单的原型方法

1、其实就是利用对象字面量法来定义原型对象，写起来更方便，不过这种写法要注意两点，一点是要在创建实例前定义，因为不然的话实例的[Prototype]指向的是之前的原型对象，是无法将两则进行联系的。当然根本原因是对象字面量法其实是默认调用Object构造函数，因此并不会重新指向新的原型对象。第二点也是前面提到的因为对象字面量法设置原型对象所以它的默认constructor属性是指向Object构造函数，我们最好重写这个属性使其指向构造函数。

（4）原型的动态性

1、这一条的意思其实是，只有在进行实例读取属性时才会进行搜索，因此即使先创建了实例，再定义原型方法，然后用实例调用也是可以的，因为只有当调用时才会去原型上搜索。

（5）原生对象的原型

1、这一条是原生引用类型（Object，Array，String等等）都是在其构造函数的原型上定义了方法，这样方法可以共享，像Array.prototype中就有sort方法。

（6）原型对象的问题

1、原型对象的问题就是当原型对象中有引用类型的属性时，任何一个实例对其进行修改都会反映在原型上，所有实例再去调用它都是已经改变的值。

**组合使用构造函数模式和原型模式**

1. 即结合了构造函数模式和原型模式的优点，并摒弃缺点，利用构造函数模式定义实例属性，原型模式定义方法和共享的属性。这样每个实例都会有自己的一份实例属性的副本，但同时又共享者对方法的引用，最大限度地节省了内存。另外这种混合模式还支持向构造函数传递参数，可谓是集两种模式之长。

继承

**原型链**

1. js中实现继承主要是依靠原型链来实现的。原理就是让一个构造函数的实例成为另一个构造函数地原型，这样另一个构造函数地原型中就会存在一个指向第一个构造函数原型的指针[Prototype]，这样依次类推就会形成原型链即继承。注意，所有函数的默认原型都是Object构造函数的实例，包含一个指向Object.prototype的指针，这正是所有类型都包含toString(),valueOf()等默认方法的根本原因。
2. 注意，利用原型实现继承时，利用一个构造函数实例重写另一个构造函数原型时，另一个构造函数原型一般就没有constructor属性了，需要的话要自己定义。
3. 在利用原型链实现继承时，当需要重写子类方法时要在用超类的实例替换子类原型之后再重写不然会发生覆盖。还有，不能使用对象字面量创建原型方法，这样会重写原型链。因为对象字面量法本质是通过Object构造函数，这样会使继承失效。
4. 原型链的问题其实就是创建对象时原型模式的问题，当一个实例的引用类型属性作为另一个构造函数的原型时就会发生像原型模式一样的问题，即改一个都会改。第二个问题是没有办法在不影响所有对象实例的情况下给超类型的构造函数传递参数。因此时机中很少单独使用原型链。

**借用构造函数**

1. 函数只不过是在特定环境中执行代码的对象，借用构造函数就是通过apply()和call()方法来改变作用域使得在新创建的对象上执行构造函数。例如：

function a(name){

this.name = name  
}

function b(age,name){

a.call(this,name)

this.age = age

}，这样通过new b(2,”nico”)返回的对象拥有的name和age属性就是独立的不会相互影响也可以进行传参。不过借用构造函数也存在问题就是方法没有复用。

**组合继承**

1. 组合继承就是结合了借用构造函数和原型继承的优点，即利用原型链实现对原型属性和方法的继承，通过借用构造函数来实现对实例属性的继承。这样既通过在原型上定义方法实现了函数的复用又保证了每个实例都有它自己的属性。
2. 组合继承也有缺点，就是组合继承调用了两次超类构造函数，一次是给子类原型赋值时，一次是new子类的构造函数时，事实上最后子类实例的属性各个独立本质上是重写的，即它的原型上的那些共享属性还是存在的只是不会读取到那一块。

**寄生组合式继承**

1. 寄生组合式继承就是去除组合继承的缺点，背后的思路是不必为了指定子类型的原型而特意去调用超类型的构造函数，我们所需要的不过是超类型原型的一个副本而已。因此可以通过一个函数来实现它：

function test(son,father){

var proto = object(father.prototype)

proto.constructor = son

son.prototype = proto

}，这里的object函数不是必须的只要能返回对象的一个副本即可。这样当执行test(son,father)了之后，子类原型就有了超类原型中的需要的属性和方法（因为可以通过object函数进行筛选）。

1. 这种继承模式比较理想，没有什么缺点。

**五、函数表达式**

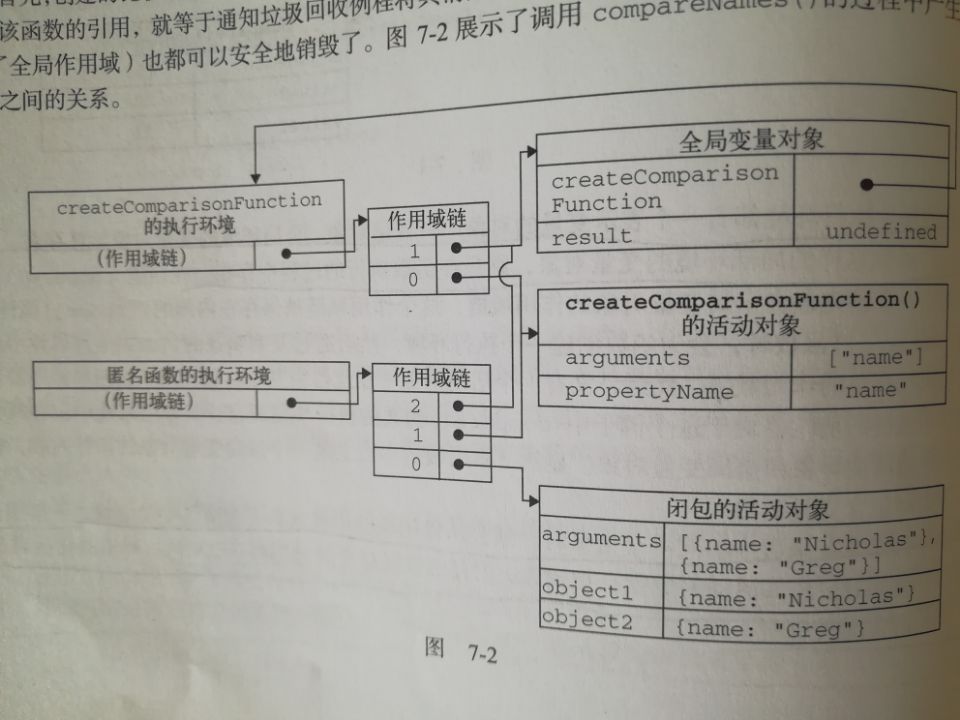
1、函数定义有函数声明和函数表达式两种，函数声明具有一个特征叫做函数声明提升，在执行代码之前会先读取函数声明，函数表达式只有执行到它才会执行。

**递归**

1. 递归就是一个函数调用自身，记住最好是通过arguments.callee来调用这样函数改名也不影响，注意递归一定要在某种情况下返回确定值。

**闭包（敲黑板）**

1. 闭包是指有权访问另一个函数作用域中的变量的函数，创建闭包的常见方式，就是在一个函数内部创建另一个函数。
2. 闭包的本质原理：当某个函数被调用时，会创建一个执行环境，该执行环境中包含作用域链，作用域链包含本身函数的活动对象以及外部函数的活动对象，当然外部函数的活动对象始终处于第二位，外部的外部处于第三位。。。直至作用域终点的全局执行环境，该活动对象是使用arguments和其他命名参数的值来初始化函数的活动对象。
3. 对于每个函数来说当它执行完毕之后它的执行环境作用域链就会被销毁。不过对于活动对象普通函数和闭包则不同，对于普通函数来说函数执行完毕后活动对象就会销毁，内存中仅保存全局作用域（全局执行环境的变量对象）。但是对于闭包来说，它会将外部函数的活动对象添加到它的作用域链中，直到闭包函数被销毁后外部函数的活动对象才会被销毁。



1. 对于this对象，每个函数在被调用时都会自动取得两个特殊变量:this和arguments，内部函数在搜索这两个变量时，只会搜索到其活动对象为止，因此永远不可能访问外部函数的这两个变量，不过把外部函数的this对象保存在一个闭包能够访问的变量里可以让闭包访问该对象。

**私有作用域**

1. 创建一个私有作用域，每个开发人员既可以使用自己的变量又不必担心搞乱全局作用域。

方法就是定义一个匿名的立即执行函数，因为函数表达式后面可以跟括号所以只需要在函数声明外部加一对括号将其变为函数表达式再去执行即可。例如(function (){..})()

**六、BOM**

**window对象**

1. 在浏览器中window对象扮演者双重角色，既是通过js访问浏览器窗口的一个接口，又是es规定的global对象。这意味着在网页中定义的任何一个对象，变量和函数都以window作为其global对象，因此有权访问parseInt()等方法。
2. 窗口关系及框架，窗口位置，窗口大小，导航和打开窗口，计时器，系统对话框，P198

**location对象**

1. location对象是最有用的BOM对象之一,它既是window对象的属性，也是document对象的属性
2. location对象常用属性见p207，replace()方法可以接收一个url参数并导航，特点是不会生成历史记录。reload()方法重新加载当前显示页面。

**navigator对象**

1. 主要用处是识别客户端浏览器的一些信息。见p211

**screen对象**

1. 用处不大，只是用来表示客户端的能力，例如浏览器窗口外部显示器的信息。P214

**history对象**

1、保存着用户的历史记录，p215

**七、DOM**

Node类型

1. js中的所有节点类型都继承自Node类型，因此所有节点类型都共享着相同的基本属性和方法。每个节点都有一个nodeType属性用于表明节点的类型，属性值有1-12每个属性值都表示不同的节点类型。

**nodeName和nodeValue属性**

1. 这两个属性可以了解到节点的具体信息，属性值对于不同的节点类型不同

**节点关系**

1. 每个节点都有一个childNodes属性，属性值是一个NodeList对象，该对象为一个类数组存储着该节点的子节点，也有length属性，通过方括号或者item()方法都可以进行访问。注意该对象是动态的，每次对其进行操作都会重新读取所以当中间dom树变化时，该对象的一些属性也会变化。
2. 每个节点都有一个parentNode属性，指向文档树中的父节点。包含在childNodes列表中的所有节点都具有相同的父节点，即parentNode属性都指向同一个节点。childNodes列表中每个节点都互为同胞节点，使用previousSibling和nextSibling可以访问前后子节点，不存在时值为null。父节点的firstChild和lastChild属性分别指向childNodes列表中的第一个和最后一个节点。
3. 节点有一个方法叫hasChildNodes()，这个方法在存在一个或多个子节点的情况下返回true。有一个属性叫ownerDocument指向表示整个文档的文档节点。

**操作节点**

1. 最常用的节点方法appendChild()用于向childNodes列表的末尾添加一个节点，如果该节点已经是文档一部分就将该节点从原来的位置转移到新位置。
2. insertBefore()方法参数有两个，要插入的节点和参考节点，如果参考节点为null就执行和appendChild()相同的操作。该方法由插入节点的父节点调用，可以直接利用插入节点的parentNode属性。
3. replaceChild()，替换节点，也是两个参数要插入的节点和要替换的节点。removeChild()，移除节点，一个参数为要移除的节点。
4. 每个节点类型都有的方法，cloneNode()，用于创建调用这个方法的节点的一个完全相同的副本，接受一个布尔值参数，当为true时执行深复制即复制节点本身及其整个子节点树，当为false时只复制节点本身。

Document类型

1. document对象的documentElement属性指向html元素，body属性直接指向body元素。还有一些属性，像title指向浏览器窗口的标题栏，URL指向url等，p255。
2. 文档写入，dom一致性检测，查找元素p257

Element类型

1. 该种类型一般用于表示html元素，提供了对元素标签名，子节点及特性的访问，元素的属性可以直接通过例如a.name,a.titel来进行读取和设置，注意class的话是a.className。不过如果是自定义属性只能通过setAttribute这种方法来设置，直接通过属性这种方式是不行的。
2. 取得特性getAttribute，移除特性removeAttribute，都是一个参数特性名字符串。设置特性setAttribute，两个参数特性名和要设置的值。
3. 两类特殊的特性，第一类是style，当通过getAttribute访问时获得的是style值中的css文本，而通过属性来访问会返回一个对象。第二类是类似onclick这样的事件处理程序，通过getAttribute方法返回js代码的字符串，通过属性返回一个js函数。

**创建元素**

1. document.createElement()，参数为元素名

**元素的子节点**

1. 在一般的浏览器中，换行是属于文本节点的

**创建文本节点**

1. document.createTextNode()，参数为文本字符串
2. 如果在一个包含两个或多个文本节点的父元素上调用normalize()方法可以将所有文本节点合并为一个文本节点。splitText()，分割文本节点，p273。

**操作表格**

1. tbody元素的属性和方法，rows：保存着tbody元素中行的集合对象；deleteRow(n):删除指定位置的行；insertRow(n):向rows集合中的指定位置插入一行，返回对新插入行的引用。
2. tr元素的属性和方法，cells：保存着tr元素中单元格的对象集合（即td集合）；deleteCell(n):删除指定位置的单元格；insertCell(n)，向cells集合中的指定位置插入一个单元格，返回对新插入单元格的引用。
3. table元素也有些属性和方法，p282。

**八、DOM扩展**

1、querySelector()，接收一个css选择符，返回与该模式匹配的第一个元素。通过document类型调用时是在全局搜索，通过element类型调用就是在该元素后代查找。querySelectorAll()，参数也是一个css选择符，不过返回的是一个类数组，别的与querySelector一样。

2、DOM元素新属性

childElementCount：返回子元素（不包括文本节点和注释）的个数。

firstElementChild：指向第一个子元素

lastElementChild：指向最后一个子元素

previousElementSibling：指向前一个同辈元素；nextElementSibling：指向后一个同辈元素

1. 元素有一个属性叫classList，有四个操作类名的方法。add(value)：添加类名，如果已存在就不添加。contains(value)：确认列表中是否含有该值，返回布尔值。remove(value):移除类名；toggle（value）：如果该类名已存在就删除，否则就添加。
2. document.head可以直接引用head元素。innerHTML,innerText,outerHTML,outerText，outer的不常用，inner的不用我多说，p296。
3. element新增了个children属性，只包含子元素节点。还有个contains方法，参数为后代节点，检查是否存在返回一个布尔值。

**九、DOM2和DOM3**

1、通过元素的style属性的不同属性可以设置元素的样式，不过要注意驼峰命名法。Style对象还有两个方法，getPropertyValue()参数为特姓名，例如border这种，style对象也是个类数组所以也可以直接通过方括号来访问。移除特性，removeProperty()。

2、元素的偏移量。offsetHeight：border+padding+height；offsetWidth：border+padding+width；offsetLeft：元素的左边框至包含元素的左内边框之间的像素距离。

offsetTop：元素的上边框至包含元素的上内边框之间的像素距离。

3、元素的客户区大小，clientWidth：width+padding；clientHeight：height+padding；

1. 滚动大小，scrollHeight：在没有滚动条的情况下，元素内容的总高度。scrollWidth：在没有滚动条的情况下，元素内容的总宽度。scrollLeft：被隐藏在内容区域左侧的像素数，通过设置这个属性可以改变元素的滚动位置。scrollTop:被隐藏在内容区域上方的像素数，通过设置这个属性可以改变元素的滚动位置。scrollTop和scrollLeft都是针对自身而言。

**十、事件**

1、js与html之间进行交互是通过事件来实现的，事件流描述的是从页面中接收事件的顺序。DOM2级事件规定的事件流包括三个阶段，事件捕获阶段（从模糊到具体），处于目标事件阶段，事件冒泡阶段（从具体到模糊），一般都是冒泡事件。

**事件处理程序**

1. html内联js
2. DOM0级事件处理程序，就是类似于element.onclick这种，一般事件前缀都有on。
3. DOM2级事件处理程序，addEventListener()，removeEventListener()，接收三个参数，第一个参数是事件名，然后是作为事件处理程序的函数和一个布尔值，如果这个布尔值是true表示在捕获阶段调用事件，false表示在冒泡阶段。通过addEventListener添加的事件处理程序只能通过removeEventListener来移除。如果是匿名函数就无法移除了。注意这种方法的事件名一般没有on前缀。
4. 在触发DOM上的某个事件时，会产生一个事件对象event，这个对象中包含着所有与事件有关的信息，包括导致事件的元素，事件的类型以及其他与特定事件相关的信息，也有些方法像preventDefault()，见p355。
5. 事件类型，p362。

**十一、表单脚本**

1. 基本属性，p412
2. 获取表单元素的独有方法，每个表单都有elements属性，该属性是个类数组，是表单中所有表单元素（字段的）集合，可以通过位置或者name特性来访问它，例如form1.elements[0]，form1.elements[‘hehe’]，第一个返回表单中第一个字段元素，第二个返回表单中name特性为hehe的元素，如果有多个name相同的就返回一个NodeList即一个类数组。