第一章

引言

> Web页面应该包含以下三个要素：内容（HTML）、外观（CSS）和行为（JS）。通常来说，*JS*的运行必须依赖于某个宿主环境。

JS现状

关于JS，最有意思的是他必须要在一个宿主环境中运行，其中受欢迎的宿主环境当然就是浏览器了，但这不是我们唯一的选择。
JS完全可以运行在服务器端、桌面以及富媒体环境中，它可以实现的功能有如下：

1. 创建具有强大而丰富的Web应用程序（这种应用程序往往运行于Web浏览器中，例如Gmail）。
2. 编写类似ASP这样的服务器端脚本，或使用Rhino（这是一种用Java实现的JS引擎）这样的框架进行编程。
3. 创建某些富媒体式的应用程序如FLASH Flex等，这其中用到的ActionScript就是一种基于ECMAScript标准的脚本语言。
4. 编写Windows桌面自动化管理脚本任务，我们可以使用Windows自带的脚本宿主环境。
5. 为一些桌面应用程序编写扩展或者插件，例如Firefox
6. 创建一些桌面型Web应用程序，这些应用程序往往会使用离线型数据库来存储信息，例如Google Gears
7. 创建小程序

面向对象的程序设计

需要知道面向对象的具体含义，列出了一系列在面向 对象程序设计(OOP）中最常用到的概念：

- 对象、方法、属性
- 类
- 封装
- 聚合
- 重用和继承
- 多态

a、对象

既然这种程序设计风格叫做面向对象，它的重点就在于对象。所谓对象，实质上是指“事物”在程序设计语言中的表现形式。OPP语义中，这些对象特征就是属性，动作称为方法。

此外，还有一个口语的类比：

- 对象往往是名词
- 方法一般都是动词
- 属性值为形容词

b、类

现实生活中，相似的对象往往都有一些共同的组成特征。OOP中，类实际上就是对象的设计蓝图或者制作配方。“对象”有时候也被称作“实例”。我们可以基于相同的类创建出许多不同的对象。因为类更多的是一种模板，对象就是在这些模板的基础上被创建出来的。

但是我们要明白，JS与C++或Java这种传统的面向对象语言不同，它实际上压根没有类。该语言的一切都是基于对象的，其所依靠的是一套原型系统，而它实际上也是一种对象。

c、封装

封装则是另一个OOP相关的概念，它主要用于阐述对象中所包含的内容，通常由两部分组成：

- 相关的数据（用于存储属性）
- 基于这些数据所能做的事（所能调用的方法）

以一个MP3播放器为例。假设这是一个对象，作为用户，我们无疑需要一些类似于想按钮、显示屏这样的工作接口。而其内部的工作原理我们不知道也不想知道。同样的，在OOP中也是如此。我们在代码中调用一个对象的方法时，无论该对象是原生的还是来自第三方库，我们都不需要该方法时如何工作的。在编译型语言中，我们甚至都无法查看这些对象的工作代码。而由于JS是一种解释型语言，源代码是可以查看的。但至少在概念上是一致的，即我们只需要知道所操作对象的接口，而不必关心其实现。

关于信息隐藏，还有另一方面内容，即方法与属性的可见性。在某些语言中，我们能通过 public、private、protected 这些关键字来限定方法和属性的可见性。这种限定分类定义了对象用户所能访问的层次。例如， private 方法只有其所在对象内部的代码才有权访问，而 public 方法则是任何人都能访问的。在JS中，尽管所有的方法和属性都是public的，但是我们将会看到，该语言还是提供了一些隐藏数据的方法，以保护程序的隐秘性。

d、聚合

所谓聚合，有时候也叫组合，实际上指的是我们将几个现有对象合并成一个新对象的过程。总之，这个概念所强调的就是这种将多个对象合而为一的能力。通过聚合，我们可以将一个问题分解成多个更小的问题。这样一来，问题就会显得更加容易管理。

e、继承

通过继承这种方式，我们可以非常优雅地实现对现有代码的重用。例如我们有个 Person 的一般性对象，其中包含一些姓名、出生日期之类的属性，以及一些功能性函数，如步行、谈话、睡觉、吃饭等。然后，当我们发现自己需要一个 Programmer 对象时，我们可以直接让Prog继承Person，这样就省去了我们不少工作。因为Prog对象只需要实现属于它自己的那部分特殊功能，而其余部分只需重用Person的实现即可。

在传统OOP环境中，继承通常指的是类与类之间的关系，但由于JS中不存在类，因此继承只能发生在对象之间。

当一个对象继承自另一个对象时，通常会往其中加入新的方法，以扩展被继承的老对象。我们通常将这个过程称之为“B继承自A”或“B扩展自A”。我们将这种重定义继承方法的过程叫做覆写。

f、多态

如果prog继承了上一级对象person的所有方法，这意味着二者都执行了talk这一方法。现在，我们的代码中有一个叫做Bob的变量，就算我们不知道他是person对象还是prog对象，依然可以调用talk方法，而不影响代码工作。类似这种不同对象通过相同的方法调用来实现各自行为的能力，我们称其为多态。

OOP概述

例子：

特征描述	相应概念
Bob 是一个男人（后者是一个对象）	对象
Bob出生于1980年，男	属性
Bob能吃喝拉撒	方法
Bob是prog类的一个实例	传统OOP中的类
Bob是Prog对象扩展而来的新对象	基于原型OOP中的原型对象
Bob对象中包含了数据和基于这些数据的方法	封装
我们不知道记录年龄的方法，也不想知道	信息隐藏
Bob只是整个开发团队的一部分，此外还有Jill和Jack	聚合、组合
他们都是扩展自Person对象的新对象	继承
我们随时可以调用他们三个各自的talk方法， 尽管也许会产生不同的结果， 总之每个对象都可以重新自定义他们的继承方法talk	多态，方法覆写

第二章

基本数据类型、数组、循环及条件表达式

将从以下几个方面入手：

- JS的基本数据类型，如字符串和数字
- 数组
- 常用操作符，例如+、-、delete、typeof等
- 控制流语句，如循环和判断

2.1 变量

通常，变量用来储存数据，其使用分为两个步骤:

- 声明变量
- 初始化变量，即给它一个初始值

变量名可以由任何数字、字符及下划线组成。但不能以数字开头。

所谓的变量初始化，实际上是指变量首次被赋值的时机。可以有两种选择。

- 先声明变量，然后再初始化
- 声明变量与初始化同时进行 如 var a = 1; 或者 var a, b, c = ‘hello’;

区分大小写

JS语言中，变量名是区分大小写的。

2.2 操作符

你懂的

2.3 基本数据类型

1. 数字
2. 字符串
3. 布尔值

4. undefined
5. null

JS中的数据类型主要分为以下两部分：

- 基本类型（上述5种）
- 非基本类型（即对象）

2.3.1 查看类型操作符————typeof

在控制台输typeof+变量

2.3.2 数字

八进制与十六进制

例子：var n3 = 0377;
控制台 n3为255

当一个数字以0开头的时候，就表示这是一个八进制数，若对八进制不熟悉，那还有十六进制。

在CSS中，我们定义颜色的方式有以下两种。

- 使用十进制数分别制定R、G、B的值，区直范围都为0~255
- 使用十六进制数，两个数位代表一个色值，一次是R、G、B。例如#000000代表黑，相互转化。

在JS中，我们会用0x前缀来表示一个十六进制值（简称为hex）