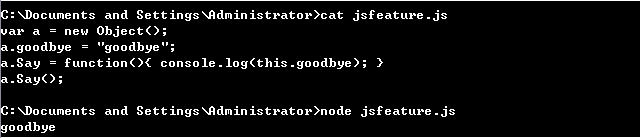
## 有关JavaScript

JavaScript是一门动态、弱类型、基于原型的函数式编程脚本语言。它不仅应用于客户端浏览器同样可以应用于服务器。

## 动态

动态性指，JavaScript对象属性，可以不在声明情况下直接赋值。



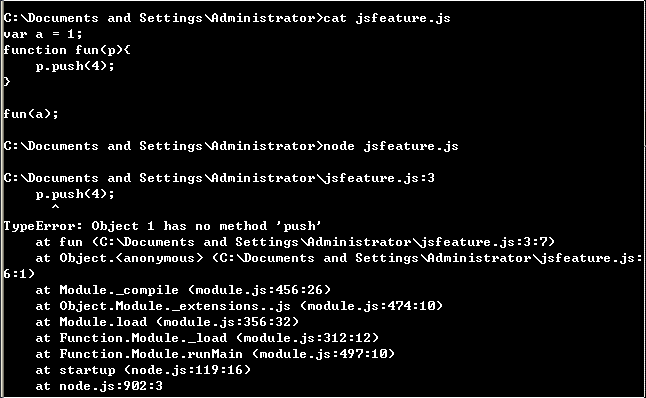
## 弱类型

弱类型指，声明时无需指定类型，用var关键字，它更像一个容器，包容任何数据甚至函数。

|  |
| --- |
| var a = undefined; //undefined var b = null; //undefined var c = true; //boolean var d = ‘test’; //string var e = 10; //nunmber  var f = function(){} //function |

*C++11标准已经支持，关键字auto*

运行期检查类型，不同于C/C++、java这样强类型语言，在编译期检查。



## 基于原型

**基于原型 vs 面向对象**

|  |  |
| --- | --- |
| 面向对象 | 即关心类同时关心对象的创建，注重类设计，例如UML |
| 基于原型 | 基于已有的原型对象来构建，不注重类设计，动态编程 |

**JavaScript构建原型对象**

Number

**\_proto\_**

**prototype**

String

**constructor**

**constructor**

Object

**prototype**

## 函数式编程

**函数是一等公民**

函数不依赖类对象而存在，同时具备函数和对象的特性。它可以拥有内部数据和状态，还可以通过scopechain链修改外部变量，它还可以作为参数也可以作为返回值。

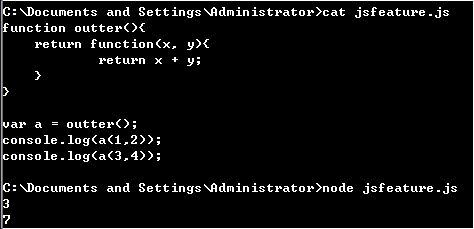
**匿名及自执行函数**

自执行函数，通常是匿名函数，例如function(){}。它类似于C++类的构造函数，一次执行后就没有用处了，例如，著名的jQuery库。

(function(window, undefined){})(windows);

**Lambda函数**

函数可以作为参数，也可以作为返回值，一系列函数形成一个表达式链。

****

**闭包**

闭包（closure）是JavaScript的一种语法特性。关于闭包，有一种经典的提法 – “闭包”是代码块和创建代码块的上下文（环境）中数据的结合。

|  |
| --- |
| function outter(){ //①父函数，返回匿名函数  var x = 0;  return function(){ //②该内部匿名函数使用了父函数的变量x，形成闭包  return x + y;  }  }  var b = outter(); //直接引用了②  console.log(b()); //0  console.log(b()); //1  var c = outter(); //直接引用了②  console.log(c()); //0  console.log(c()); //1  var a = outter; //引用了外层函数①  console.log(a()); //0  console.log(a()); //0  console.log(outter()()); //直接调用0  console.log(outter()()); //0 |

**柯利化**

**JavaScript传值or 传引用？**