* 什么是闭包（闭包的原理）--函数嵌套函数

闭包就是能够读取其他函数内部变量的函数。由于在Javascript语言中，只有函数内部的子函 数才能读取局部变量，因此可以把闭包简单理解成"定义在一个函数内部的函数"。所以，在本质 上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

* 闭包的弊端

闭包特性1: 由于内部函数所引用的外部变量会被保存下来 所以变量会一直在内存中保存直到 浏览器被关闭，垃圾回收机制失效。（造成内存消耗很大，所以不能滥用闭包，否则会造成网页 的性能问题）

* 闭包的应用场景
  + 1、函数外部获取函数内部的局部变量，但是函数内部的函数所引用的变量会被保存下来。
  + 2、避免全局变量的污染。
  + 3、实现模块化。
  + 4、自执行的匿名函数保存循环变量
* 写js代码:三种方式

1.闭包

2.命名空间（namespace）:定义不同的名称,当作对象的标识符，进行开发避免冲突。

3.面向对象

4.模块化--seajs/require.js

* 应用

1. 掌握闭包的特点和原理
2. 掌握闭包的应用场景

* 构造函数继承

将父对象的构造函数绑定在子对象上,子对象能够继承父对象的属性和方法

* call/apply继承

在子对象构造函数中利用call或apply将父对象指向子对象的实例对象。

* Prototype(原型)--原型的概念

Javascript规定，每一个函数都有一个prototype对象属性，指向另一个对象（原型链）。

prototype(对象属性)的所有属性和方法，都会被构造函数的实例继承。这意味着，我们可以把 那些不变(公用)的属性和方法，直接定义在prototype对象属性上。

* prototype就是调用构造函数所创建的那个实例对象的原型（\_\_proto\_\_）。
* prototype可以让所有对象实例共享它所包含的属性和方法。也就是说，不必在构造函数中定义对象信息，而是可以直接将这些信息添加到原型中。
* 构造函数与实例

构造函数（类）---不具备方法和属性，只能用来构造实例对象，构造函数里面添加的 是属性（因为一般情况下属性各不相同）原型里面添加的是方法（因为方法都一样）

实例（实例对象），具有一定的功能，被构造函数（类）构造出来的。

* Prototype模式的其他验证方法
* hasOwnProperty() : 看是不是对象自身下面的属性， 只在属性存在于实例中时才返回 true。
* in操作符用来判断某个属性属于某个实例对象，可以是对象的直接属性，也可以是通过prototype继承的属性(包括Object下面的属性)。
* toString() :系统对象下面都是自带的 , 系统对象都有toString(),自己写的对象都是通过原型链找Object下面的，可以把对象转成字符串，可以做进制转换或者类型的判断。
* constructor：实例对象的构造函数 (实例对象.constructor)
* instanceof：判断一个对象是否是一个构造函数（类）的实例。
  + isPrototypeOf (): 判断一个实例对象是否指向了该构造函数的原型对象。
* 原型链（JS原型与原型链）

实例对象与原型之间的连接，叫做原型链。\_\_proto\_\_( 隐式连接 )

JS在创建对象的时候，都有一个叫做\_\_proto\_\_的内置属性，用于指向创建它的函数对 象的原型对象prototype。

内部原型(\_\_proto\_\_)和构造器的原型（prototype）

1、每个对象都有一个\_\_proto\_\_属性,原型链上的对象正是依靠这个属性连结在一起

2、作为一个对象，当你访问其中的一个属性或方法的时候，如果这个对象中没有这个 方法或属性，那么Javascript引擎将会访问这个对象的\_\_proto\_\_属性所指向上一个对 象，并在那个对象中查找指定的方法或属性，如果不能找到，那就会继续通过那个对象 的\_\_proto\_\_属性指向的对象进行向上查找，直到这个链表结束。

原型链的最外层：Object.prototype-->null

* 混合继承（构造函数+原型混合方式实现js的继承）
  + 属性的继承和构造函数的继承一致的，call/apply
  + 方法的继承1：拷贝继承，一一赋值。
  + 方法的继承2：原型继承。
* 应用

1. 掌握继承的原理

* 综合应用

1. 飞机大战游戏

Object(Array,Date,RegExp...)

js是一个公司-->boss(Object.prototype)

对比：Object.prototype Function.prototype Function Object

Function(function)