1. 函数也是对象，可以为函数设置属性、方法等。如Object函数的prototype属性，Object.create（）方法等等。
2. 函数声明语句，实际上声明了一个变量（ES5中只有两种生命变量的方式），然后把一个函数对象赋值给它。同时，函数名称会成为函数内部变量对象中以的一个局部变量。
3. 嵌套的函数和对象的方法函数还是很不一样的，this值。This永远是谁调用了它，谁激活了这段上下文，谁就是this。嵌套函数中this如果不指定的话，ES5非严格模式下为全局对象，严格模式下为undefined。
4. 构造函数new会新建一个对象，并将这个对象的\_proto\_属性指向构造函数的prototype，同时new 会调用构造函数，并设置执行上下文中this的值为这个新创建的对象。
5. 函数也是对象，任意函数都从Function.prototype上继承到了apply（）和call（）两个方法，都允许显示指定this的值。也就是说，任何函数都可以作为任意对象的方法来调用，哪怕这个函数不是那个对象的方法。
6. JS的函数调用不会检查传入实参的个数。如果传入的实参的个数比形参要少，那么剩下的形参将会是undefined，多出来的自动省略。
7. arguments对象  
   arguments标识符是指向实参对象的引用，是一个类数组对象。非严格模式下，arguments的数组元素是形参所对应实参的别名。严格模式废除了这一特性。  
   arguments.callee表示正在执行的函数，arguments.caller表示正在执行的函数的函数。通过caller可以访问调用栈。严格模式废除了，也就是削弱了访问调用栈的能力。
8. 函数本身的length属性  
   函数本身也有一个length属性，为形参的个数，也就是期望传入实参的个数。有时候可以比较arguments.length和arguments.callee.length是否一致。
9. 函数的prototype属性  
   每一个函数都可以成为构造函数，每一个函数都有一个prototype属性。
10. 函数的call（）和apply（）方法  
    每一个函数都有这两个方法。ES5严格模式下传入的第一个参数无论如何都会成为this的值，即使传入的是undefined和null。ES5的非严格模式下，如果传入的是undefined和null，则会替代为全局对象。  
    call（）第一个实参之后，所有的实参都是要传入待调用函数的。  
    apply（）不同，它的实参都需要放入一个数组中，传入的可以是真实的数组，也可以是类数组对象。
11. 函数的toString（）方法  
    返回字符串，大部分浏览器都返回完整的源码，内置函数往往返回一个类似"function tanh() { [native code] }"，这是Math.tanh.toString（）。
12. 函数的bind（）方法  
    ES5新增的方法。它的作用是返回一个新函数，将该新函数绑定到某个对象上，调用这个新函数就像是调用这个对象的方法一样。实际上，他也是指定this的值。如：  
    function f(y){return this.x+y}  
    var o={“x”:1};  
    var g=f.bind(o);  
    g(2);这个值就是3  
    除此之外，bind（）除了第一个实参会被绑定到this，其他的实参也会被绑定，“柯里化”。如：  
    var sun=function(x,y){return x+y}  
    var succ=sun.bind(null,1);  
    succ(2);这时候返回3