function Foo(){  
      getName = function(){  
        alert(1);  
      }  
        return this;  
    }  
    Foo.getName = function(){  
      alert(2);  
    }  
    Foo.prototype.getName = function(){  
      alert(3);  
    }  
    var getName = function(){  
      alert(4);  
    }  
    function getName(){  
      alert(5);  
    }  
    Foo.getName();    // 2;  
    getName();       // 4;  
    Foo().getName();   // 1;  
    getName();         //  1  
    new Foo.getName();  // 2  
    new Foo().getName();   // 3;  
    new new Foo().getName()  // 3

answer:

Foo.getName();  相当于所谓的静态方法，调用的是Foo.getName=这一方法，所以是2

getName();    全局方法，编释顺序决定，var getName覆盖了函数声明，所以是4

Foo().getName()   用Foo当函数调用，函数内有全局的getName,用var getName给重写了，所以是1

getName();   因为上一条，所以还是1

new Foo.getName()  把静态方法getName当构造函数，执行了静态的getName，所以是2

new Foo().getName()  这是正常的对象化方法，调用了自身的prototype.getName，所以是3

最后这个，不太清楚是干嘛，但最终会执行上一条，所以还是3.