Chapter 5. Inheritance

# Capítulo 5. Herencia

Divide una cosa entera en muchos objetos; Como las perspectivas, que bien miradas No muestran más que confusión. . .

-WilliamShakespeare, *The Tragedy of King Richard the Second*

La herencia es un tema importante en la mayoría de los lenguajes de programación.

En los lenguajes clásicos (como Java), la herencia (o extends) proporciona dos servicios útiles. En primer lugar, es una forma de reutilización del código. Si una nueva clase es en su mayor parte similar a una clase existente, sólo tienes que especificar las diferencias. Los patrones de reutilización de código son extremadamente importantes porque tienen el potencial de reducir significativamente el coste del desarrollo de software. La otra ventaja de la herencia clásica es que incluye la especificación de un sistema de tipos. Esto libera en gran medida al programador de tener que escribir operaciones de casting explícitas, lo cual es muy bueno porque al hacer casting se pierden las ventajas de seguridad de un sistema de tipos.

JavaScript, al ser un lenguaje poco tipado, nunca hace castings. El linaje de un objeto es irrelevante. Lo que importa de un objeto es lo que puede hacer, no de qué desciende.

JavaScript proporciona un conjunto mucho más rico de patrones de reutilización de código. Puede imitar el patrón clásico, pero también admite otros patrones que son más expresivos. El conjunto de posibles patrones de herencia en JavaScript es enorme. En este capítulo, veremos algunos de los patrones más sencillos. Son posibles construcciones mucho más complicadas, pero normalmente es mejor mantener la sencillez.

En los lenguajes clásicos, los objetos son instancias de clases, y una clase puede heredar de otra clase. JavaScript es un lenguaje prototípico, lo que significa que los objetos heredan directamente de otros objetos.

[Subtema 1 de 2: (Ver todo)](ch05.html)[siguiente](ch05_2.html)