Curry

# Curry

Las funciones son valores, y podemos manipular los valores de las funciones de formas interesantes. *Currying* nos permite producir una nueva función combinando una función y un argumento:

var add1 = add.curry(1);  
document.writeln(add1(6)); // 7

add1 es una función que se creó pasando 1 al método curry de add. La función add1 añade 1 a su argumento. JavaScript no tiene un método curry, pero podemos arreglarlo aumentando Function.prototype:

Function.method('curry', function ( ) {  
 var args = arguments, that = this;  
 return function ( ) {  
 return that.apply(null, args.concat(arguments));  
 };  
}); // Something isn't right...

El método curry funciona creando un cierre que contiene esa función original y los argumentos al curry. Devuelve una función que, al ser invocada, devuelve el resultado de llamar a esa función original, pasándole todos los argumentos de la invocación de curryy de la invocación actual. Utiliza el método Array concat para concatenar las dos matrices de argumentos.

Por desgracia, como hemos visto antes, la matriz argumentsno es una matriz, por lo que no tiene el método concat. Para evitarlo, aplicaremos el método slice a las dos matrices arguments. Esto produce matrices que se comportan correctamente con el método concat:

Function.method('curry', function ( ) {  
 var slice = Array.prototype.slice,  
 args = slice.apply(arguments),  
 that = this;  
 return function ( ) {  
 return that.apply(null, args.concat(slice.apply(arguments)));  
 };  
});