Module

# Módulo

Podemos utilizar funciones y cierres para crear módulos. Un módulo es una función u objeto que presenta una interfaz pero que oculta su estado e implementación. Al utilizar funciones para producir módulos, podemos eliminar casi por completo el uso de variables globales, mitigando así una de las peores características de JavaScript.

Por ejemplo, supongamos que queremos aumentar Stringcon un método deentityify. Su función es buscar entidades HTML en una cadena y sustituirlas por sus equivalentes. Tiene sentido guardar los nombres de las entidades y sus equivalentes en un objeto. Pero, ¿dónde debemos guardar el objeto? Podríamos ponerlo en una variable global, pero las variables globales son malvadas. Podríamos definirlo en la propia función, pero eso tiene un coste en tiempo de ejecución porque el literal debe evaluarse cada vez que se invoca la función. El enfoque ideal es ponerlo en un cierre, y quizás proporcionar un método extra que pueda añadir entidades adicionales:

String.method('deentityify', function ( ) {  
  
// The entity table. It maps entity names to  
// characters.  
  
 var entity = {  
 quot: '"',  
 lt: '<',  
 gt: '>'  
 };  
  
// Return the deentityify method.  
  
 return function ( ) {  
  
// This is the deentityify method. It calls the string  
// replace method, looking for substrings that start  
// with '&' and end with ';'. If the characters in  
// between are in the entity table, then replace the  
// entity with the character from the table. It uses  
// a regular expression (Chapter 7).  
  
 return this.replace(/&([^&;]+);/g,  
 function (a, b) {  
 var r = entity[b];  
 return typeof r === 'string' ? r : a;  
 }  
 );  
 };  
}( ));

Fíjate en la última línea. Invocamos inmediatamente la función que acabamos de crear con el operador ( ). Esa invocación crea y devuelve la función que se convierte en el método deentityify.

document.writeln(  
 '&lt;&quot;&gt;'.deentityify( )); // <">

El patrón módulo aprovecha el ámbito de la función y el cierre para crear relaciones vinculantes y privadas. En este ejemplo, sólo el método deentityify tiene acceso a la estructura de datos de la entidad.

El patrón general de un módulo es una función que define variables y funciones privadas; crea funciones privilegiadas que, mediante cierre, tendrán acceso a las variables y funciones privadas; y que devuelve las funciones privilegiadas o las almacena en un lugar accesible.

El uso del patrón módulo puede eliminar el uso de variables globales. Promueve la ocultación de información y otras buenas prácticas de diseño. Es muy eficaz para encapsular aplicaciones y otros singletons.

También puede utilizarse para producir objetos que sean seguros. Supongamos que queremos crear un objeto que produzca un número de serie:

var serial\_maker = function ( ) {  
  
// Produce an object that produces unique strings. A  
// unique string is made up of two parts: a prefix  
// and a sequence number. The object comes with  
// methods for setting the prefix and sequence  
// number, and a gensym method that produces unique  
// strings.  
  
 var prefix = '';  
 var seq = 0;  
 return {  
 set\_prefix: function (p) {  
 prefix = String(p);  
 },  
 set\_seq: function (s) {  
 seq = s;  
 },  
 gensym: function ( ) {  
 var result = prefix + seq;  
 seq += 1;  
 return result;  
 }  
 };  
};  
var seqer = serial\_maker( );  
seqer.set\_prefix('Q');  
seqer.set\_seq(1000);  
var unique = seqer.gensym( ); // unique is "Q1000"

Los métodos no hacen uso de this o that. Como resultado, no hay forma de comprometer el seqer. No es posible obtener o modificar el prefix o el seq excepto si lo permiten los métodos. El objeto seqer es mutable, por lo que los métodos podrían sustituirse, pero eso seguiría sin dar acceso a sus secretos. seqer es simplemente una colección de funciones, y esas funciones son capacidades que otorgan poderes específicos para utilizar o modificar el estado secreto.

Si pasáramos seqer.gensym a la función de un tercero, esa función podría generar cadenas únicas, pero no podría modificar las prefix o seq.