Capítulo 6
Tablas

1. Tablas de dimensión única

En el capítulo Desarrollo a partir de algoritmos, hemos visto de pasada el potencial de las tablas de dimensión única y de dimensiones múltiples. Veamos cómo se gestionan en JavaScript.

1.1 Sintaxis

En JavaScript, una tabla de dimensión única es una variable en memoria "compuesta", en la que va a ser posible almacenar varios datos independientes, de tipos diferentes, con una indexación de cada uno de los valores con un número (o índice).

Por tanto, el acceso a cada dato de la tabla se hará con este valor de índice.

Una particularidad respecto a este índice es que su valor para la primera celda de la tabla es 0.

El lenguaje JavaScript proporciona varias maneras de crear una tabla:

– la sintaxis literal,

– la sintaxis llamada "Programación orientada a objetos".

Con una sintaxis literal, la declaración de una tabla de nombre `tabSemana` de siete celdas, que contiene las etiquetas de los días de una semana, se hace como sigue:

```
var tabSemana = ["Lunes", "Martes", "Miércoles", "Jueves",
"Viernes", "Sábado", "Domingo"];
```

Observe que la declaración está acompañada de la inicialización de cada una de las celdas de la tabla `tabSemana` (de la celda de índice 0 a la celda de índice 6).

Con una sintaxis de "Programación orientada a objetos", tendríamos:

```
var tabSemana = new Array("Lunes", "Martes", "Miércoles", "Jueves",
"Viernes", "Sábado", "Domingo");
```

Hubiéramos podido declarar la tabla `tabSemana` sin asignarle valores. Se pueden considerar asignaciones posteriores, como por ejemplo para el Lunes:

```
tabSemana[0] = "Lunes";
```

Lo que es verdaderamente particular en la gestión de las tablas en JavaScript es la extrema flexibilidad:

– sin dimensionamiento a priori (siempre es posible extender el tamaño de la tabla en función de las necesidades),
– posibilidad de mezclar en una misma tabla datos de tipos diferentes,
– posibilidad de utilizar tablas asociativas (tablas cuyos índices se sustituyen por valores textuales).

En un procesamiento, para acceder al contenido de un valor de tabla asociado a una posición de índice particular, la sintaxis será:

```
document.write("El cuarto día de la semana es " + tabSemana[3]);
```

Observación

Es necesario recordar siempre que la numeración de los índices empieza en cero.

Para terminar, debe saber que JavaScript ofrece muchos métodos que se aplican en las tablas (`Array`). Con estos métodos puede fácilmente insertar, eliminar y encontrar elementos en una tabla. También existen métodos de ordenación (`sort`, `reverse`) para clasificar fácilmente los valores contenidos en una tabla, sin tener que recurrir a la pesada escritura de un algoritmo de ordenación.

1.2 Ejercicio n.°14: Contar los números pares en una tabla

Enunciado

Determinar la cantidad de números pares en una tabla (entrada de datos inicial de valores con el teclado).

Corrección (parcial) en JavaScript

```
/* Declaración de variables locales */
/*
i         : Contador de bucle
nbPares   : Acumulado de la cantidad de números pares
tabla  :    Tabla de números
*/
var i, nb_pares;
var tabla = new Array;

/* Inicializaciones */
nbPares = 0;
for (i=1; i<=5; i++)
{
    tabla[i] = parseInt(prompt("tabla[" + i +"]: "));
}

/* Determinar la cantidad de números pares en la tabla */
for (i=1; i<=5; i++)
{
    if (tabla[i]%2 == 0)
    {
        nbPares = nbPares + 1;
    }
}

/* Visualización del resultado */
document.write("La tabla contiene " + nbPares + " números pares");
```

Comentarios del código JavaScript

No hay nada realmente nuevo en este script, excepto el cálculo del módulo. Este cálculo sirve aquí para determinar la paridad de cada contenido de celdas de la tabla. Se hace con el operador `%`.

Puede que haya observado que, en este script, la celda con índice 0 no se ha utilizado (la numeración del bucle `for` empieza en 1). Esta elección hace más comprensible el algoritmo.

2. Tablas de dimensiones múltiples

Es frecuente que necesitemos una tabla de dimensiones múltiples para gestionar problemas, fundamentalmente en matemáticas, estadística...

JavaScript ofrece esta posibilidad.

2.1 Sintaxis

Como para las tablas de dimensión única, JavaScript permite declarar las tablas de dimensiones múltiples de varias maneras:

– con una sintaxis literal,

– con una sintaxis llamada "Programación orientada a objetos".

Con una sintaxis llamada "Programación orientada a objetos" (incluso llamada JSON - *JavaScript Object Notation*), la declaración de una tabla de nombre `tabMatriz` de dos líneas, divididas en cuatro columnas con inicialización, se hace como sigue:

```
/* Declaración de la tabla tabMatriz */
var tabMatriz tabla = new Array();

/* Declaración de la primera "línea" de la tabla tabMatriz */
tabMatriz[0]=new Array()

/* Inicialización de las 4 "columnas" de la primera "línea" */
tabMatriz[0][0] = "Uno";
tabMatriz[0][1] = "Dos";
```

```
tabMatriz[0][2] = "Tres";
tabMatriz[0][3] = "Cuatro";

/* Declaración de la segunda "línea" de la tabla tabMatriz */
tabMatriz[1]=new Array()

/* Inicialización de las 4 "columnas" de la segunda "línea" */
tabMatriz[1][0] = "Once";
tabMatriz[1][1] = "Doce";
tabMatriz[1][2] = "Trece";
tabMatriz[1][3] = "Catorce";
```

2.2 Ejercicio n.°15: Minihoja de cálculo

Enunciado

A partir de la tabla `tb` de dos dimensiones, con cuatro líneas y cinco columnas, realizar los procesamientos siguientes:

– introducir mediante el teclado valores en las tres primeras líneas y las cuatro primeras columnas (se conservan la última línea y la última columna libres para los totalizadores de líneas y columnas),
– añadir columnas en la última línea y líneas en la última columna.

Corrección (parcial) en JavaScript

```
 /* Declaración de variables locales */
var tb = new Array(5);
var numLinea, numColumna;
var valor;

/* Declaración de 5 "columnas" por "línea" para la tabla tb */
for (var numLinea=1; numLinea<tb.length; numLinea++)
{
    /* Creación de las "columnas" (numeradas de 0 a 5) */
    tb[numLinea]=new Array(6);
}

/* Inicialización de la tabla tb

  (3 filas * 4 columnas) para introducir datos por teclado */
for (numLinea= 1; numLinea<= 3; numLinea++) {
```

```
    for (numColumna = 1; numColumna <= 4; numColumna++) {
        valor = parseInt(prompt("tabla[" + numLinea + "]
[" + numColumna + "] = "));
        tb[numLinea][numColumna] = valor;
    }
}

/* Puesta a cero de los totales en línea n.°4 */
for (numColumna=1; numColumna<=5; numColumna++)
{
    tb[4][numColumna] = 0;
}

/* Puesta a cero de los totales en columna n.°5 */
for (numLinea=1; numLinea<=4; numLinea++)
{
    tb[numLinea][5] = 0;
}

/* Determinación de los totales en línea n.°4 y en columna n.°5 */
for (numLinea=1; numLinea<=3; numLinea++)
{
    for (numColumna=1; numColumna<=4; numColumna++)
    {
        /* Totalización en línea n.°4 */
        tb[4][numColumna] = tb[4][numColumna]
        + tb[numLinea][numColumna];
        /* Totalización en columna n.°5 */
        tb[numLinea][5] = tb[numLinea][5]
        + tb[numLinea][numColumna];
        /* Totalización general en línea n.°4-columna n.°5 */
        tb[4][5] = tb[4][5] + tb[numLinea][numColumna];
    }
}

/* Visualización del total general */
/* NB: Total de 78 suponiendo que se mantiene la técnica de llenado
de la tabla tb */
document.write("Total general en tb[4][5] = " + tb[4][5]);
```