JavaScript para principiantes

Harris, Andy

Introducción

Al principio de los 80, mi hermano y yo compramos un ordenador. Mi madre pensaba que estabamos locos, porqué no hacía nada. Tenía razón. Había muy poco software disponible. Pasamos muchas noches tecleando programas (normalmente juegos). Rara vez funcionaban una vez acababamos de teclear, lo que nos obliga a repasar el código con mucho cuidado. A veces, eramos capaces de encontrar los errores tipográficos y conseguir que los juegos funcionaran. Después, nos encontramos cambiando el código, intentando mejorar los programas que tecleabamos. Eso era una forma estupenda de aprender a programar. Trabajabamos en un lenguaje sencillo que no tenía demasiadas opciones complicadas. Escribíamos juegos que eran más divertidos porque los habíamos fabricado nosostros mismos. La programación de juegos es especialmente gratificante, porqué el resultado son programas que estamos deseando utilizar. Nuestras habilidades mejoraron porqué la programación de juegos nos ponía muchos retos. Después, descubrimos que las habilidades que habíamos aprendido desarrollando juegos eran muy útiles para las aplicaciones “serias” también.

Hoy parece difícil aprender a programar de la manera en que lo hicimos mi hermano y yo. Los ordenadores son más complejos que los que nosotros utilizabamos. Los lenguajes de programación se han hecho mucho más complejos y las herramientas que necesita un programador: compiladores, entornos integrados de desarrollo (IDE) y depuradores parecen caros, complejos y prohibitivos para cualquiera que sólo pretende empezar y jugar un poco con el ordenador.

No obstante, aún es posible aprender a programar de una forma parecida a cómo lo hicimos nosotros. Una nueva cosecha de lenguajes sencillos para principiantes están apareciendo. Especialmente JavaScript se está convirtiendo en un lenguaje ideal para que los principiantes aprendan a programar. JavaScript está integrado en los navegadores actuales, luego el lenguaje no cuesta nada. Está disponible en cualquier ordenador que disponga de estos navegadores. El lenguaje tiene una sintaxis razonablemente clara y concisa que permite a los principiantes una introducción amigable a los conceptos más modernos de la programación cómo la orientación a objetos y la programación basada en eventos. Además, no tiene tantas características cómo para necesitar un diploma en ingeniería informática para comprenderlo.

El propósito de este libro es enseñarle los principios básicos de la programación. Aprenderá los conceptos más importantes de la programación aplicados a JavaScript. Usaremos el contexto de la programación de juegos para enseñar los conceptos, pero verá que las técnicas que aprenderemos nos valen para cualquier propósito.

Si ya sabe JavaScript, podrá encontrar algunas ideas en la descripción de los programas. Si ya ha programado juegos, se sorprenderá de algunas de las cosas que puede hacer JavaScript. Si ambas cosas le son nuevas, vamos a pasar un tiempo estupendo explorando estos nuevos conceptos.

Verá que puede aplicar las habilidades adquiridas a otro tipo de programación incluso a otros lenguajes. Vamos a suponer que no tiene ninguna experiencia en programación. Sólo

supondremos que conoce HTML y que sabe cómo crear páginas Web con un editor de textos. Va a necesitar un buen editor de textos, un editor de gráficos e incluso si quiere un editor de sonido.

Cómo utilizar este libro

Para aprender a programar es necesario adquirir una compleja progresión de habilidades y destrezas. En cualquier caso no se aprende a programar leyendo.

A PROGRAMAR SÓLO SE APRENDE PROGRAMANDO.

Tiene que escribir programas para aprender. Este libro está planeado para que el proceso le resulte lo menos penoso posible. Cada capítulo empieza con un programa completo que muestra las ideas principales del caítulo. Luego se analizan las ideas principales. Finalmente se proponen una serie de ejercicios para afianzar y desarrollar las habilidades necesarias. Los programas son lo suficientemente cortos cómo para que se puedan teclear (no cortar y pegar), esta es una forma muy interesante para que analicemos de cerca el código que estamos utilizando.

A lo largo del libro nos encontraremos con algunas reseñas del siguiente tipo:

CONSEJO – Buenas ideas que los programadores comparten entre ellos.

CUIDADO – Cuidado, áreas en las que es muy fácil equivocarse. Las señalaremos a lo largo del libro.

TRUCO – Técnicas y atajos que le harán la vida más fácil.

Tema 1 – Variables, entrada y salida

Programar no es más que controlar de forma más directa lo que queremos que haga el ordenador. Seguramente utilizará un ordenador para múltiples aplicaciones y dominará los programas que usa asiduamente.

Aún así, sin la programación, seguirá a la merced de los programas diseñados por otras personas. En este capítulo, vamos a ver cómo podemos empezar a dominar el comportamiento del ordenador. Concretamente vamos a ver:

- **Cómo podemos insertar código en una página HTML**
- **Utilizar cuadros de diálogo para interactuar con el usuario**
- **Aprender cómo el ordenador almacena los datos en variables**
- **Aprender cómo podemos obtener datos del usuario**
- **Realizar operaciones básicas con los datos**

Proyecto: Jugando con el nombre

En la Figura 1.1, un cuadro de diálogo emerge de una página web y pregunta al usuario por su nombre. A continuación, un serie de cuadros preguntan por el apellido y más formas de jugar con el nombre.

Ninguna compañía de videojuegos le va a comprar el juego. No obstante, incluso este sencillo juego aporta algo nuevo a una página web común. La mayoría de las páginas web no permiten al usuario interactuar con ellas. No infravalore el hecho de que la página web trate al usuario de forma personalizada. Esto añade funcionalidad a la página.

Añadiendo código al HTML

Con los conocimientos de HTML que tiene, puede crear páginas web que tengan un aspecto interesante. Por ejemplo, puede controlar el formato del texto, añadir imágenes. Además será capaz de conseguir diseño más perfeccionados con la tecnología de las hojas de estilo. Pero, las páginas HTML siendo adoleciendo de interacción. La única interacción del usuario se limita a pinchar en los enlaces que le interesan.

Creando la aplicación Hola, mundo!

Sería interesante hacer que la página sea más dinámica. La mayoría de los navegadores modernos soporta JavaScript, un lenguaje muy adecuado para añadir interactividad a una página web. Echemos un vistazo al siguiente trozo de código:

```
<html>
<script>
//hola mundo
//el clásico primer programa en cualquier lenguaje
alert("Hola mundo!");
</script>
</html>
```

Si guardamos este programa cómo una página web, y la cargamos en un navegador, veremos la pantalla que se muestra en la Figura 1.2. Este trozo de código está compuesto de HTML normal pero incluye una nueva característica. Las etiquetas `<script></script>` (guiones) le dicen al navegador que el código que se encuentra entre ellas no es HTML, sino un lenguaje de script (habitualmente será JavaScript, pero podrían ser otros). Se pueden colocar las etiquetas `<script> </script>` en cualquier parte del documento HTML pero lo habitual es hacerlo dentro de la cabecera. El código que va entre las etiquetas `<script> </script>` está escrito siguiendo las reglas de JavaScript. JavaScript es un lenguaje diferente a HTML, y sus instrucciones son diferentes. Los caracteres // indican un comentario. El interprete ignora lo que viene detrás hasta la línea siguiente. No obstante, esta información es muy útil para los programadores. Es fundamental añadir comentarios a sus programas no sólo para que otros programadores sepan lo que está haciendo sino sobre todo para que nos sea más fácil modificar o corregir nuestros propios programas cuando haya pasado un cierto tiempo.

NOTA.- Aunque a los usuarios no les gustan las ventanas emergentes que interrumpen el discurrir lógico del programa, las cajas de diálogo son muy útiles. Primero porqué son fáciles de programar. Cualquier otra forma de comunicarse con el usuario es bastante más complicada. Segundo, consiguen llamar la atención del usuario. Tercero, serán de gran utilidad cuando estemos probando los programas.

Mandando un mensaje al usuario

Una sóla línea de código realiza todo el trabajo interesante:

```
alert("hello world");
```

Utilizamos la instrucción `alert` para mandar un mensaje. El mensaje aparece en su propio cuadro de diálogo. Un cuadro de diálogo es bastante insistente. Si intenta hacer click en la página antes de cerrarlo, protestará con un pitido y no dejará hacer otra cosa hasta que no se cierre el cuadro.

Se habrá fijado en el carácter punto y coma (;) al final de la línea. Este carácter le indica al interprete que la orden ha terminado y que puede ejecutarla. La mayoría de las líneas de código en JavaScript terminan con el punto y coma. Los comentarios no lo necesitan ya que el interprete los ignora.

El uso de las variables

Uno de los aspectos más importantes de la programación es aprender cómo el ordenador utiliza los datos. Los datos son la información que el odenador almacena y manipula. En los primeros programas los datos suelen ser texto, cómo nombres o frases. Más adelante, veremos otro tipo de datos cómo números. Los lenguajes de programación utilizan las llamadas variables cómo herramienta para manipular los datos. Ahora veremos cómo se usan las variables para almacenar información.

Creando la aplicación Hola, Pepe!

Take a look at the program shown in Figura 1.3. It shows an example of output with a new twist: This time, the computer generates a message already stored in the computer’s memory. This program’s code looks like this:

```
<html>
<head> <title>Hola Pepe</title>
</head>
<body> <h1>Hola, Pepe!</h1>
<script>
//Hola Pepe
//Muestra algunos conceptos básicos de variables

var saludo;
saludo = "Hola Pepe, qué tal?";
alert(saludo);
</script>
</body>
</html>
```

Básicamente, este programa almacena el texto `"Hola Pepe, qué tal?"`, y después muestra un mensaje con ese texto en una caja de diálogo. Este programa nos muestra cómo el ordenador almacena información para reutilizarla posteriormente. Un elemento especial, llamado variable es lo que se utiliza. Los ordenadores trabajan fundamentalmente con información. De ahí la importancia de saber cómo la almacenan. Pensemos de esta manera: una variable es cómo una taquilla en la que dejamos algo guardado, a veces no sabemos ni lo que hemos guardado lo que sí que tenemos que recordar es la identificación de la taquilla dónde está para recuperarlo. Las variables cumplen un función muy similar en un ordenador. Almacenan información hasta que el ordenador la necesita para trabajar con ella.

Figura 1.3

Por ahora, es suficiente recordar que cada vez que queramos que el ordenador guarde información, ya sea un nombre, un mensaje, un precio, etc. vamos a necesitar una variable.

Utilizando la instrucción var

Cada lenguaje de programación ofrece algún tipo de soporte para las variables. En JavaScript, los programadadores utilizan la instrucción `var` para crear una nueva variable. Cuando se crea una variable, hay que darle un nombre. Esto es cómo ponerle una etiqueta a la taquilla para que después podamos localizarla. Veamos lo que ocurre con la instrucción:

```
var saludo;
```

El término `var` le indica al ordenador que cree una variable. La palabra `saludo` es la etiqueta que se le va a poner a la variable. La línea termina con punto y coma cómo la mayoría de las instrucciones.

Después de interpretar esta línea de código, el ordenador genera un espacio de memoria al que se accede por medio de la etiqueta `saludo`. A partir de ahora, podemos escribir código que escriba nuevos valores en este espacio de memoria, o código que lea los valores que hay almacenados.

Pautas para nombres de variables

Los programadores han de ponerle nombre a montones de cosas. Los programadores experimentados han desarrollado algunas pautas para dar nombres, que es bueno conocer: en

primer lugar ojo con las mayúsculas y las minúsculas en muchos lenguajes (incluyendo JavaScript), `nombreusuario`, `nombreUsuario`, y `NOMBREUSUARIO` son nombres completamente diferentes.

Use nombres descriptivos. No se deben usar nombres como `r` o `x`, porqué luego será muy difícil recordar exactamente qué dato es el que almacena la variable. Nombres como `tasaIVA` o `saludo` harán que el código sea mucho más fácil de entender. No use espacios en blanco ni de puntuación. La mayoría de lenguajes no permiten nombres de variables formados por varias palabras. Muchos programadores utilizan una mayúscula (tasaIVA) o el guión bajo (tasa_iva) para crear nombres de variables formados por múltiples palabras y que sean fáciles de leer. Los signos de puntuación suelen tener significados especiales en los lenguajes por lo que es una mala idea utilizarlos en los nombres de variables. No utilice nombres de variable tan largos que sea muy difícil teclearlos. Nombres entre 5 y 15 caracteres es una buena medida.

Asignando un valor a una variable

Echemos un vistazo a esta línea de código:

```
saludo = "Hola Pepe, qué tal?";
```

Básicamente, esta línea asigna el texto `"Hola Pepe, qué tal?"` a la variable `saludo`. Cualquier cosa que se encuentra entre comillas se llama cadena de caracteres. String es el término que usan los programadores para referirse a las cadenas de caracteres, o sea al texto. El signo igual (=) indica asignación. La forma correcta de leer esta instrucción sería: almacena la cadena de caracteres `"Hola Pepe, qué tal?"` en la variable `saludo`.

CUIDADO – No sería correcto en absoluto decir que saludo es igual a "Hola Pepe, qué tal?”. La igualdad es un concepto totalmente diferente que veremos en otro capítulo.

Resumiento, la palabra `saludo` es el nombre de una variable que contiene la cadena de caracteres `"Hola Pepe, qué tal?"`. Para guardar un valor en una variable se usa el operador asignación (=). Hubiese sido más correcto usar cómo operador de asignación (<-).

Usando el contenido de una variable

Usamos el nombre de la variable para hacer referencia a la misma. Echemos un vistazo a la línea:

```
alert(saludo);
```

Cuando el usuario carga la página web, no aparece `saludo` en el cuadro de diálogo. Si no que lo que aparece es el contenido que hay almacenado en la variable, en este caso `"Hola Pepe, qué tal?"`. Cualquiera que sea el contenido almacenado en la variable este será el que se muestre cuando se llame a la variable.

Obteniendo datos del usuario

Además de mostrar información al usuario, los ordenadores también obtienen información del usuario. Este tipo de información se llama entrada.

CONSEJO – A veces es fácil no tener claro si algo es entrada o salida. Por ejemplo, supongamos que estamos leyendo un mensaje de texto en la pantalla del ordenador. Conforme está leyendo el mensaje este se convierte en entrada para su cerebro; no obstante, desde la perspectiva del ordenador se trata de una salida. El convenio en programación es que cuando hablamos de entrada o salida, lo hacemos desde el punto de vista del ordenador.

Creando la aplicación Hola usuario!

Veamos este programa, que muestra un tipo sencillo de entrada: ahora, el ordenador le pregunta al usuario por su nombre y utiliza esa información en otra instrucción. Este es el código:

```
<html>
<head>
<title>Hola usuario</title>
</head>
<body>
<h1>Hola, usuario!</h1>
 <script>
//hola usuario
//pregunta al usuario por su nombre

var nombreUsuario;
nombreUsuario = prompt("Cómo te llamas?");
alert(nombreUsuario);
</script>
</body>
</html>
```

Este programa tiene una variable (`nombreUsuario`), pero ahora el valor que se guarda en la variable no lo determina el programa. Al contrario, el usuario tiene la oportunidad mediante un cuadro de diálogo especial de introducir el valor que se va a almacenar. Ahora que sabemos obtener datos del usuario, los programas se pueden hacer mucho más flexibles. Por ejemplo, podrá escribir programas que llamen al usuario por su nombre, aunque no tenga ni idea de cuál va a ser cuando escribe el programa (ver Figura 1.4).

Figura 1.4

Usando la instrucción prompt

La instrucción que hace posible que el usuario introduzca un valor en una variable es `prompt`. Se utiliza así:

```
nombreUsuario = prompt("Cómo te llamas?");
```

Cómo vemos, la línea empieza de forma similar a la sentencia de asignación del programa Hola Pepe, pero ahora, el valor que se asigna a la variable `nombreUsuario` no está fijado en el programa. La instrucción `prompt` hace aparecer un cuadro de diálogo. Pero este cuadro de diálogo es diferente al que produce la instrucción `alert`. No sólo manda un mensaje al usuario, sino que además presenta un cuadro de texto para que el usuario teclee una respuesta. El principal propósito de la instrucción `prompt` es obtener un valor del usuario. Cada sentencia con `prompt`

incluye algún tipo de variable lista para almacenar ese valor. Una sentencia de entrada, cómo la instrucción `prompt`, se usa cuando necesitamos almacenar en una variable la respuesta del usuario a alguna pregunta. La instrucción `prompt` genera un cuadro de diálogo que presenta al usuario una pregunta y un cuadro de texto para teclear una respuesta. Se usa casi siempre cómo parte de una sentencia de asignación para guardar la respuesta del usuario en una variable.

TRUCO – Cuando generamos un cuadro de diálogo de entrada (prompt), tenemos que determinar la pregunta que le vamos a hacer al usuario, y tener lista una variable dónde almacenar la respueta.

Diciendo Hola al usuario

Ahora que tenemos el nombre del usuario guardado en una variable, es fácil devolver el valor al usuario. Es lo que hace la siguiente línea:

```
alert(nombreUsuario);
```

Cómo `nombreUsuario` no está entre comillas, el ordenador interpreta que es un nombre de variable, la localiza y muestra al usuario su contenido. Si ponemos comillas alrededor de `nombreUsuario`, lo que se mostraría sería "`nombreUsuario`" y no el valor almacenado en la variable `nombreUsuario`.

Creando un texto más elaborado

El mensaje al usuario podía haber sido más familiar. Si el nombre del usuario es Susana, hubiese quedado mejor decir "Hola, Susana!!". La Figura 1.5 muestra esta salida mejorada.

Creando el programa de concatenación (Unir o enlazar unas cosas con otras)

Para crear la pantalla de la Figura 1.5, debemos combinar cadenas de caracteres (la parte "Hola," y la parte "!!") con el valor guardado en una variable. En los programas anteriores, la salida era el valor guardado en la variable sin más texto alrededor. En este programa, vamos a incluir el valor de una variable junto con el resto del texto.

Concatenando Strings

Concatenando Strings: la combinación de dos o más cadenas de caracteres. En JavaScript, la **concatenación** se realiza con el **signo** más (+).

Figura 1.5

Veamos el código:

```
<html>
<head> <title>Concatenacion</title>
</head>
<body>
<h1>Concatenacion</h1>
<script>
//concatenacion
//preguntar por el nombre del usuario
```

```
var nombreUsuario;
var saludo;
nombreUsuario = prompt(""Cómo te llamas?");
saludo = "Hola, " + nombreUsuario + "!!";
alert(saludo);
</script>
</body>
</html>
```

Uniendo variables y literales

El programa utiliza dos variables: `nombreUsuario` para guardar el nombre del usuario y `saludo` que contiene el texto que se mostrará al usuario. El programa obtiene el valor de `nombreUsuario` mediante una instrucción `prompt`, al igual que en el programa anterior. Esta es la única novedad:

```
saludo = "Hola, " + nombreUsuario + "!!";
```

Ya debemos ser capaces de reconocer que el programa asigna un valor a la variable `saludo`. Para formar ese valor, el programa concatena el literal `"Hola, "` con el contenido de la variable `nombreUsuario` y por último le añade el literal `"!!"`. Podemos utilizar la concatenación para crear cadenas de texto realmente largas y complejas.

Trabajando con números

Los ordenadores son muy buenos trabajando con valores de texto. Pero son aún mejores trabajando con números. Muy a bajo nivel, el texto y los números se almacenan de la misma forma valores en binario. Los lenguajes de alto nivel cómo JavaScript nos abstraen de la complejidad de trabajar a ese nivel. Entre otras cosas es capaz de adivinar cuando el programador está hablando de números y cuando de texto. Aunque, a veces, es necesario suministrarle algo de información para que acierte.

Creando la aplicación Cuenta

Este programa, mostrado en la Figura 1.6 presta un pequeño servicio. Mira cuánto vale la comida, calcula el 10 % de IVA (IVA de hostelería y restaurantes), lo suma y obtiene el total de la cuenta. Veamos el código:

```
<html>
<head> <title>Cuenta</title>
</head>
<body>
<h1>La cuenta, por favor!</h1>
<script>
//Cuenta
//Cómo se hacen cálculos básicos

var comida = 22.50;
var iva = comida * 0.10; //tipo actual para restaurantes
var total = comida + iva;
alert ("La comida son €: " + comida);
```

```
alert ("El IVA son €" + iva);
alert ("El importe total son: €" + total);
</script>
</body>
</html>
```

Figura 1.6

Usando variables numéricas

El programa Cuenta tiene variables, pero los valores que guardan no son texto. En este programa, queremos que el ordenador haga cálculos matemáticos, luego las variables son de tipo numérico. Fíjese que no hay comillas alrededor del valor 22.50. También, que se puede combinar la instrucción `var` con una sentencia de asignación, de forma que la variable guarda un valor de forma inmediata. La línea siguiente calcula el valor del IVA multiplicando el valor guardado en la variable `comida` por 0.10 (o sea el 10%):

```
var iva = comida * 0.10;
```

En programación, la multiplicación se suele representar por el asterisco (*) (la x se utiliza cómo variable, y el punto para otras funciones). La línea siguientes crea la variable `total`:

```
var total = comida + iva;
```

Esta sentencia suma el contenido de la variable `comida` con el de la variable `iva`, y guarda el resultado en la variable `total`. La instrucción `alert` funciona tal y cómo esperamos: el programa convierte de forma automática los números en texto al generar la salida.

Depurando la aplicación Mala cuenta

Nuestro anterior programa sería mucho más práctico si permitiera que el usuario introdujese el importe de la comida. Veamos esta variante del programa Cuenta:

```
<html>
<head> <title>mala Cuenta</title>
</head>
<body>
<h1>La mala Cuenta</h1>
<script>
//malaCuenta
//Demuestra un fallo potencial

var comida;
//obtener el importe de la comida del usuario
comida = prompt("Cuánto cuesta la comida?");
var iva = comida * 0.10;
var total = comida + iva;
alert ("La comida son €: " + comida);
alert ("El IVA son €" + iva);
alert ("El importe total son: €" + total);
</script>
</body>
```

```
</html>
```

El programa parece correcto, pero los resultados, cómo se ve en la Figura 1.7, no lo son. Algo ha fallado. Antes de continuar, veamos si podemos intuir cuál es el problema.

Figura 1.7

Interpretando números y texto

Esto es lo que ha sucedido: la instrucción `prompt` devuelve una cadena de caracteres. El ordenador guarda ese valor cómo texto, lo que no supone ningún problema hasta que intenta operar con él. En la línea siguiente, `comida` almacena una cadena de caracteres que es lo que le devuelve `prompt`:

```
comida = prompt("Cuánto cuesta la comida?");
```

La línea siguiente multiplica el valor de `comida` por 0.10:

```
var iva = comida * 0.10;
```

No tiene sentido multiplicar una cadena de caracteres por un número, luego JavaScript en su afán por ayudar convierte el String almacenado en `comida` en un número, y la multiplicación funciona. En la línea siguiente es dónde aparece el problema:

```
var total = comida + iva;
```

El ordenador sigue interpretando comidoa como un String e iva cómo un número. El problema es el signo (+), que es el operador que le dice al ordenador que sume números. Pero que **si se utiliza con cadenas de caracteres lo que hace es concatenarlas**. En esta sentencia, el signo (+) tiene un número en un lado y una cadena de caracteres en el otro, tiene un dilema. En este caso, decide tratar ambos valores cómo cadenas de caracteres y concatenarlas. Luego, el resultado de concatenar “22.50” y “2.25” es otra cadena de caracteres, “22.502.25”.

Creando la aplicación Buena cuenta

JavaScript ofrece varias soluciones para este problema, una de las más fáciles es la función eval(). Veamos la nueva versión del código:

```
<html>
<head><title>Buena cuenta</title>
</head>
<body> <h1>La buena Cuenta</h1>
<script>
//buenaCuenta
//Demuestra la funcion eval()

var comida;
//obtener el importe de la comida del usuario
comida = prompt("Cuánto cuesta la comida?");
//convertir el valor de comida en un número
comida = eval(comida);
var iva = comida * 0.10;
```

```
var total = comida + iva;
alert ("La comida son €: " + comida);
alert ("El IVA son €" + iva);
alert ("El importe total son: €" + total);
</script>
</body>
</html>
```

Sólo hay una nueva línea de código. La línea:

```
comida = eval(comida);
```

Esta sentencia simplemente evalúa el valor de la cadena de caracteres que ha introducido el usuario. El programa vuelve a guardar el resultado en la misma variable comida. En este caso, el programa devuelve un valor numérico. Veamos los resultados en la Figura 1.8, y veremos que el programa ahora funciona correctamente.

Figura 1.8

Usando los métodos de String

Además de la posibilidad de trabajar con números, la mayoría de los lenguajes de programación nos permiten manipular las cadenas de texto. Incluyen capacidades para convertir el contenido de una variable en mayúsculas, en minúsculas, comandos que nos permiten saber la longitud de la cadena, y técnicas para concatenar cadenas. Podemos mezclar todas estos comandos, técnicas y capacidades para programar “Jugando con el nombre”.

String	**La clase que nos sive para manejar cadenas de caracteres**. Estudiamos sus propiedades y los métodos más interesantes.

Propiedades	**length** La clase String sólo tiene una propiedad: length, que guarda el número de caracteres del String
Métodos	**charAt(indice)** Devuelve el carácter que hay en la posición indicada como índice. Las posiciones de un String empiezan en 0.
	indexOf(carácter,desde) Devuelve la posición de la primera vez que aparece el carácter indicado por parámetro en un String. Si no encuentra el carácter en el String devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir de que posición se desea que empiece la búsqueda.
	lastIndexOf(carácter,desde) Busca la posición de un carácter exáctamente igual a como lo hace la función indexOf pero desde el final en lugar del principio. El segundo parámetro indica el número de caracteres desde donde se busca, igual que en indexOf.
	replace(substring_a_buscar,nuevoStr) Sirve para reemplazar porciones del texto de un String por otro texto, por ejemplo, podríamos uilizarlo para reemplazar todas las apariciones del substring "xxx" por "yyy". El método no reemplaza en el String, sino que devuelve un resultante de hacer ese reemplazo. Acepta expresiones regulares como substring a buscar.
	substring(inicio,fin) Devuelve el substring que empieza en el carácter de inicio y termina en el carácter de fin. Si intercambiamos los parámetros de inicio y fin también funciona. Simplemente nos da el substring que hay entre el carácter menor y el mayor.
	toLowerCase() Pone todas los caracteres de un String en minúsculas.
	toUpperCase() Pone todas los caracteres de un String en mayúsculas.
	toString() Este método lo tienen todos los objetos y se usa para convertirlos en cadenas.

Jugando con el nombre

Vamos a escribir el programa “Jugando con el nombre” aplicando los conceptos que hemos visto en este capítulo. También llamaremos la atención sobre algunas cosas.

```
<html>
<head> <title>Jugando con el nombre</title>
</head>
<body>
<h1>Jugando con el nombre</h1>
```

```
<script>
//Jugando con el nombre
//hace varias manipulaciones con el nombre del usuario

//usa los métodos de String
var nombre = "";
var apellido = "";
var numLetras = 0;
nombre = prompt("Hola, cómo te llamas?", "");
alert ("Tienes un bonito nombre, " + nombre);
alert ("Creo que voy a gritarlo: " + nombre.toUpperCase());
apellido = prompt("Y cuál es tu apellido, " + nombre + "?");
alert ("Oh. " + nombre + " " + apellido + ".");
alert ("A veces te llaman " + apellido + ", " + nombre);
numLetras = nombre.length + apellido.length;
alert ("Sabías que tu nombre y apellido tienen " + numLetras + "
letras?");
</script>
</body>
</html>
```

Hemos visto la mayoría de este código anteriormente. No es más que algunas sentencias de entrada y salida con alguna manipulación de cadenas de caracteres.

Empezamos con la declaración de variables

Al inicio de los programas suele ir la declaración de las variables que vamos a necesitar:

```
var nombre = "";
var apellido = "";
var numLetras = 0;
```

La mayoría de las sentencias van a depender de estas variables, por lo que es importante crearlas al principio. También es interesante que estén en un sitio dónde sea fácil localizarlas. Cuando los programas se hagan más largos será útil tener las variables localizadas. Además de crear la variables les damos un valor inicial. JavaScript no necesita que se determine el tipo de la variable, pero es importante. Es una buena costumbre que nos vendrá bien cuando usemos otros lenguajes más exigentes.

Convirtiendo a mayúsculas

After the program obtains the user's first name, it does some manipulation:

```
alert ("Creo que voy a gritarlo: " + nombre.toUpperCase());
```

En JavaScript, los Strings vienen provistos de una serie de capacidades que se llaman métodos. Podemos ver en la tabla resumen algunos de los más importantes. La instrucción `nombre.toUpperCase()` convierte el valor de la variable `nombre` todo a mayúsculas. A continuación, se concatena el nuevo valor con el literal `"Creo que voy a gritarlo: "`.

Concatenando Strings más complejos en entradas y salidas

Vemos en este programa que es habitual concatenar cadenas de caracteres en otras más complejas para generar entradas o salidas. La siguiente sentencia concatena una variable y dos literales para formular la pregunta:

```
apellido = prompt("Y cuál es tu apellido, " + nombre + "?");
```

Contando letras en Strings

La última sentencia `alert` nos dice cuántas letras tiene el nombre y el apellido del usuario.

```
numLetras = nombre.length + apellido.length;
```

`numLetras` es una variable numérica, `nombre.length` devuelve el número de caracteres de la variable. Asimismo, `apellido.length`, devuelve el número de caracteres del apellido. El programa suma ambos valores y los almacena en la variable `numLetras`.

CONSEJO – Puede parecerle extraño que nombreString.toUpperCase() tenga paréntesis al final y nombreString.length no. Esto es porqué length no es un método de las variables de tipo String, sino una propiedad. Más adelante veremos esta diferencia más a fondo, por ahora recordar que los métodos llevan paréntesis y las propiedades no.

Combinando números y texto

Llegados a este punto, puede estar inquieto por saber cuando algo es un número y cuando es una cadena de caracteres. JavaScript es un lenguaje amigable. Trata de adivinar lo que el programador quiere, y, la mayoría de las veces, acierta. Compruebe sus programas, y si ve una concatenación cuando está esperando una suma, use la sentencia eval con las variables que JavaScript debe interpretar cómo números. Si lo que tiene son números que su programa tiene que concatenar con texto, esto suele suceder correctamente, cómo en esta línea:

```
alert ("Sabías que tu nombre y apellido tienen " + numLetras + "
letras?");
```

Resumen

En un solo tema, hemos aprendido muchas cosas. Hemos visto la forma de incorporar programas JavaScript en páginas HTML. Hemos visto la forma de generar salida con la instrucción `alert` y la de obtener información del usuario con la instrucción `prompt`. Hemos visto las variables. Hemos empezado con las operaciones que podemos realizar con números. Hemos visto los métodos y las propiedades de la clase String. Hemos empezado a programar y a utilizar un entorno de desarrollo.

EJERCICIOS

1. Escribir un programa en JavaScript que pregunte al usuario su nombre (1ª entrada) y su apellido (2ª entrada) y que los devuelva en el formato: apellido, nombre (1ª salida).

2. Escribir un programa en JavaScript que pregunte al usuario su nombre y su apellido en el formato: apellido, nombre (1ª entrada) y que devuelva el nombre (1ª salida) y el apellido (2ª salida). (Consejo: utilizar los métodos de la clase String).

3. Escribir un programa que pregunte al usuario dos números (2 entradas) y muestre la suma, la resta, la multiplicación y la división (4 salidas). (Consejo: la división utiliza el símbolo /, y la multiplicación el símbolo *.)