JavaScript de Andy Harris

TEMA 2 – Números aleatorios y la sentencia if

En el tema anterior hemos visto, cómo obtener datos del usuario, cómo manipular esos datos, y cómo generar salida para el usuario. En este tema vamos a aprender a hacer aún más con los datos. Vamos a:

- **Generar números aleatorios**
- **Manipular estos números para que estén en un rango determinado**
- **Construir una condición**
- **Usar condiciones para controlar la lógica del programa**
- **Construir estructura condicionales más complejas**

Proyecto: “El oráculo”

La Figura 2.1 muestra el programa “El Oráculo”, que genera una predicción al usuario cada vez que se carga la página.

Generando números aleatorios

Los juegos son mucho más interesantes cuando no son totalmente previsibles. Los programadores de juegos usan habitualmente los números aleatorios para simular el azar. La capacidad de generar números aleatorios en un rango determinado es una destreza importante para los programadores de juegos.

Creando el programa “Generador de números aleatorios”

El programa “Generador de números aleatorios” (ver Figura 2.2) es muy sencillo, pero tiene los fundamentos necesarios para muchos juegos. Cada vez que se cargue la página se obtendrá un número aleatorio entre 0 y 1. Aunque estos números no son especialmente útiles por si mismos,

Figura 2.1

se pueden moldear de forma muy flexible. Cómo veremos en breve, podemos hacer algunos pequeños trucos para obtener números aleatorios para otras aplicaciones más prácticas, cómo por ejemplo un dado. Echemos un vistazo al código del programa, para ver cómo funciona:

```
<html>
<head> <title>Generador de números aleatorios</title>
<script>
// Generador de números aleatorios
// Demuestra el uso del generador de números aleatorios

var numero;
numero = Math.random();
alert ("Hola, este es mi número: " + numero);
```

```
</script>
</head>
<body>
<h1>Generador de números<br /></h1>
<hr>
<h3>Pulse el botón actualizar para ver otro número!</h3>
</body>
</html>
```

Cómo vemos este código no aporta nada nuevo excepto por una línea. Claramente `numero` es la varible dónde se guarda el número. El valor lo genera la instrucción `Math.random()`.

Usando el objeto Math

Se dice que JavaScript es un lenguaje basado en objetos. La implicación exacta de esta declaración se verá más adelante, pero ya hemos empezado a ver algunos objetos. En el Tema anterior vimos el objeto **String**, con sus métodos y sus propiedades. Un objeto es un tipo especial de entidad, que se compone de métodos (cosas que puede hacer el objeto) y de propiedades (valores que tiene el objeto). JavaScript tiene predefinido un objeto **Math**. Este objeto viene con una interesante colección de métodos y propiedades. Cada vez que necesitemos alguna función matemática (cómo calcular el coseno de un ángulo, un logaritmo, etc.), buscaremos en este objeto para comprobar si está disponible. Si lo está, nos ahorraremos un montón de código, nos bastará con utilizar el método que viene definido. En la Tabla 2.1 veremos algunas de las funciones y propiedades más interesantes de Math.

Math	**La clase que utilizamos para realizar cálculos matemáticos de todo tipo.**
Propiedades	**E** - Número E o constante de Euler, la base de los logaritmos neperianos.
	PI - Famoso número para cálculos trigonométricos.
	SQRT2 - Raiz cuadrada de 2
Métodos	**abs()** Devuelve el valor absoluto de un número.
	acos() Devuelve el arcocoseno de un número en radianes.
	asin() Devuelve el arcoseno de un número en radianes.
	atan() Devuelve el arcotangente de un número en radianes.
	ceil() Devuelve el entero igual o inmediatamente superior de un número (techo).
	cos() Devuelve el coseno de un ángulo expresado en radianes.
	exp() Devuelve el resultado de elevar el número E a un número.
	floor() Devuelve el entero igual o inmediatamente inferior de un número (suelo).
	log() Devuelve el logaritmo neperiano de un número.
	max() Retorna el mayor de 2 números.
	min() Retorna el menor de 2 números.
	pow() Recibe dos números como parámetros y devuelve el primer número elevado al segundo número.

	random() Devuelve un número aleatorio entre 0 y 1.
	round() Redondea al entero más próximo.
	sin() Devuelve el seno de un ángulo expresado en radianes.
	sqrt() Devuelve la raiz cuadrada de un número.
	tan() Devuelve la tangente de un ángulo expresado en radianes.

Usando el método Math.random()

Dentro de los métodos de la clase Math, el método Math.random() de especial interés para los desarrolladores de juegos, porque genera números aleatorios. El valor estará entre 0 y 1. La mayoría de los lenguajes de programación tienen algún tipo de generador de números aleatorios, y es muy habitual que devuelvan un valor entre 0 y 1. Concretamente, la línea siguiente obtiene el valor aleatorio y lo guarda en la variable `numero`:

```
numero = Math.random();
```

CONSEJO – Técnicamente, los valores devueltos son sólo pseudoaleatorios, se derivan de un fórmula matemática y están basados en el reloj del sistema. Esto no suele ser un problema pues son lo suficientemente aleatorios para usarlos en los juegos.

Creando números aleatorios especializados

Ahora sabemos generar un número aleatorio entre 0 y 1. Pero estos números no son espcialmente interesantes. Normalment vamos a necesitar que los números estén en un rango diferente. Por ejemplo, si estamos haciendo un juego en el que se use un dado necesitaremos números entre 1 y 6.

Figura 2.3

Creando “Lanzar el dado”

El programa “Lanzar el dado”, simula el lanzamiento de un dado de 6 caras. La Figura 2.3 muestra un par de ejecuciones del programa y podemos ver el resultado. El programa genera un número entre 0 y 1, hace ciertas manipulaciones y lo convierte en un número entre 1 y 6. Este program muestra los pasos intermedios, aunque en un programa normal esto no se hace. El código del programa “Lanzar el dado” es el siguiente:

```
<html>
<head> <title>Lanzar el dado</title>
<script>
// Lanzar el dado
// convierte un número aleatorio en la cara de un dado

var semilla = 0;
var mayor = 0;
var entero = 0;
var final = 0;
var resultado = "";
```

```
semilla = Math.random();
mayor = semilla * 6;
entero = Math.floor(mayor);
final = entero + 1;
resultado = "Semilla: " + semilla + "\n";
resultado += "Mayor: " + mayor + "\n";
resultado += "Entero: " + entero + "\n";
resultado += "Final: " + final + "\n";
alert(resultado);
</script>
</head>
<body>
<h1>Lanza el dado<br />
</h1>
<hr>
</body>
</html>
```

El programa empieza, cómo siempre, con la creación de variables. Hemos creado un grupo de variables para guardar los distintos pasos del proceso. Aunque no son necesarias tantas variables, a veces descomponer los procesos en pasos más sencillos ayuda a comprenderlos mejor. Aunque JavaScript no es exigente con el tipo de las variables, es buena costumbre indicar qué tipo de datos va a guardar una variable con la inicialización.

Obteniendo el valor inicial

El primer paso del proceso es obtener la semilla (un número decimal entre 0 y 1). Lo hacemos con el código que ya hemos visto:

```
semilla = Math.random();
```

Obteniendo números más grandes

Ahora tenemos un número entre 0 y 1 con un montón de decimales almacenado en `semilla`. Buscamos un valor entre 1 y 6, sin decimales. Vamos a necesitar algunos pasos para conseguirlo. El primer paso será obtener un número entre 0 y 5. Esta línea realiza el cambio de rango:

```
mayor = semilla * 6;
```

Esta sentencia multiplica el valor guardado en `semilla` por 6. El resultado será un valor mayor que 0 y menor que 6, pero que seguirá teniendo un montón de decimales. Aunque nos estamos acercando a la solución los decimales siguen siendo un problema.

Convirtiendo a entero

Recordaremos que los números positivos y negativos sin decimales se llaman enteros. Los números con decimales se llaman números reales. Concretamente, los ordenadores utilizan un forma especial de representar los números reales llamada en coma flotante. Muy a bajo nivel, los números enteros y los reales se almacenan y manipulan de forma muy diferente. Aunque los lenduajes de programación nos aislan de estos detalles, esta diferencia es bastante importante

para el ordenador. La mayoría de lenguajes de programación nos ofrecen la posibilidad de cambiar de un tipo de número al otro. Obtener un número entero a partir de un número en coma flotante es relativamente sencillo. JavaScript nos ofrece hasta tres formas distintas de hacerlo. En el programa hemos utilizado una de ellas:

```
entero = Math.floor(mayor);
```

La clase Math tiene un método `floor()`, que simplemente elimina la parte decimal del número y se queda con la parte entera. El programa coge el valor almacenado en `mayor` (un valor con decimales entre 0 y 6) y corta cualquier cosa después del entero. A continuación se guarda el valor entero resultante en la variable `entero`. Su valor será 0, 1, 2, 3, 4 ó 5

CONSEJO – Antes, hemos mencionado que había tres formas de convertir un número real en entero. La clase Math ofrece también el método ceil() y el método round(). Se puede usar cualquiera de las opciones, pero los resultados serán diferentes. ceil() siempre redondea hacia arriba, y round() redondea cómo lo hacemos habitualmente, es decir al entero más cercano.

Obteniendo valores mayores que 0

La variable `entero` tiene ya unos valores muy próximos a los que vamos buscando, pero seguimos teniendo un problema. El rango no es exactamente el que queremos, necesitamos entre 1 y 6 y ahora tenemos entre 0 y 5. Esta línea resuelve este pequeño problema:

```
final = entero + 1;
```

Ahora cada vez que ejecutamos el programa el número generado estará entre 1 y 6.

Desarrollando un algoritmo para generar números aleatorios

Obtener números aleatorios en un cierto rango es un problema muy habitual en programación. He aquí un resumen del proceso que hemos seguido:

1. Obtener un número aleatorio entre 0 y 1 (semilla).
2. Multiplicar el valor por el tamaño de nuestro rango.
3. Convertir el valor en entero.
4. Sumarle el valor del extremo inferior del rango.

Usaremos esta técnica a menudo cuando escribamos juegos. Aunque utilicemos otros lenguajes de programación la estrategia que hemos descrito sigue siendo válida. Este tipo de estrategias en programación se llaman algoritmos (Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema). Una vez tenemos un buen algoritmo, suele ser muy sencillo ponerlo en forma de código. Luego en la **programación** lo **importante** no es tanto el código cómo el **algoritmo** que nos permite alcanzar una solución.

Toma de decisiones con la sentencia if

Hasta ahora, nuestros programas han sido totalmente secuenciales. Han sido sólo una lista de intrucciones que el ordenador ha realizado. Esta es una forma de programar perfectamente

legítima, pero a menudo necesitaremos que el ordenador realice diferentes operaciones dependiendo de la situación.

La forma en la que conseguimos que los ordenadores **tomen decisiones** sigue un proceso muy similar a cómo lo hacemos nosotros mismos. Por ejemplo, al levantamos, mucha gente pone la radio para escuchar la previsión del tiempo. Les gusta saber qué tiempo va a hacer antes de decidir cómo vestirse. Siguen un proceso similar al siguiente “Va a hacer frío, hoy. Voy a ponerme un jersey.” Aunque no seamos plenamente conscientes, este el proceso que sucede. Este es un ejemplo de una acción lógica. **Se basa en** una construcción muy importante llamada **condición**.

Una condición es una expresión que puede evaluarse a verdadero o falso. En el problema del tiempo, la condición “Va a hacer frío, hoy.” Esta expresión es verdadera o falsa. En términos humanos, las condiciones son preguntas que tienen una respuesta del tipo si/no. Cuando estemos diseñando algoritmos, a menudo necesitaremos pensar en estos términos y expresar nuestra lógica en forma de preguntas con respuesta si/no.

Creando el programa “Baja temperatura”

Las Figuras 2.4, 2.5, y 2.6 muestran algunas ejecuciones del programa “Baja temperatura”. Este programa genera un número aleatorio para indicar la temperatura actual, a continuación aconseja una vestimenta adecuada si la temperatura está por debajo de un límite de 18 grados.

Figura 2.4
Figura 2.5
Figura 2.6

Este es el código del programa:

```
<html>
<head><title>Baja temperatura</title>
<script>
// Baja temperatura
// Demuestra el uso de la sentencia if

var temp = 0;
var tempPerfecta = 18;
temp = Math.floor(Math.random() * 40) + 1;
alert ("Hace " + temp + " grados fuera. ");
if (temp < tempPerfecta){
   alert("Coge un jersey!!");
}// end if
</script>
</head>
<body>
<h1>Baja temperatura<br /></h1>
<hr />
<h3>Pulse el botón recargar para ver otra temperatura</h3>
</body>
</html>
```

Generando la temperatura

La línea que genera la temperatura asusta un poco:

```
temp = Math.floor(Math.random() * 40) + 1;
```

Si nos fijamos bien, veremos que no es más que la implementación del algorimo para generar números aleatorios, sólo que en una sóla línea con la ayuda de los paréntesis. Los paréntesis marcan el orden en que se ejecutan las operaciones, exactamente igual que en matemáticas, luego lo primero que hace el ordenador es ejecutar `Math.random()`. A continuación, multiplica ese valor por 40. Calcula el suelo del número obtenido y por último le suma 1. Con lo que tendremos un rango de temperaturas entre 1 y 40 que para Sevilla parece bastante ajustado.

NOTA – Ver precedencia de operadores

Toma de decisiones con condiciones

Por supuesto, el ordenador nunca será tan flexible cómo la mente humana. Analicemos las siguientes expresiones:

- Hará frío.
- Hace frío.
- Hoy hará frío.
- Hará fresco hoy.
- Las temperatura estará por debajo de los 18 grados.
- Hoy será más o menos cómo ayer.

La mente humana es suficientemente flexible para interpretar correctamente todas estas sentencias (y muchas más) que significan básicamente lo mismo. Los seres humanos están capacitados para entender diferentes estructuras sintácticas (la estructura de la frase) y determina su significado correcto (la semántica). Los lenguajes de programación se llevan mal con las sutilezas. La mayoría de las veces, habrá pocas formas diferentes de explicarle lo anterior al ordenador para que lo entienda. El arte de la programación consiste muchas veces en afinar la excesiva expresividad del lenguaje humano y ser capaces de convertirlo a un lenguaje de programación que es mucho más restrictivo sin perder significado.

Operador	**Descripción**
==	" Igual a" devuelve true si los términos son iguales
===	"Igual a" estricto
!=	" **No igual** a" devuelve true si los términos no son iguales
!==	" No igual a" estricto
>	" Mayor que" devuelve true si el término de la izquierda es mayor que el de la derecha.
>=	" Mayor o igual que " devuelve true si el término de la izquierda es

	mayor o igual que el de la derecha.
<	" Menor que" devuelve true si el término de la izquierda es menor que el de la derecha.
<=	"Menor o igual que" devuelve true si el término de la izquierda es menor o igual que el de la derecha

Para expresar este tipo de condiciones en JavaScript, necesitamos construir una condición. Habitualmente, una condición compara el valor de una variable con el valor de otra. Por ejemplo, en el programa anterior: `temp < 18`. Echemos un vistazo al funcionamiento: `temp` es una variable. Suponemos que el programador ha creado e inicializado la variable anteriormente. Podemos usar cualquier tipo de variable en una condición, pero no podemos comparar variables de distintos tipos (Más tarde ampliaremos esto). El signo menor que (<) es un operador de comparación, en la tabla anterior tenemos los operadores de comparación que usa JavaScript.

Usando la sentencia if

Una vez que hemos entendido las condiciones, la sentencia **if** es sencilla de entender. Veamos el código del programa Baja temperatura:

```
if (temp < tempPerfecta){
   alert("Coge un jersey!!");
}// end if
```

AMPLIACIÓN – La sentencia if es un ejemplo de estructura lógica. Las estructuras lógicas son los elementos que nos permiten escribir programas flexibles. La mayoría de las estructuras lógicas que vamos a aprender están basadas en condiciones, por lo que es vital entender cómo funcionan las condiciones. Y esto es válido para cualquier lenguaje de programación. Las condiciones se pueden combinar en condiciones compuestas.

La sentencia if se compone de una condición entre paréntesis, y a continuación una llave de apertura ({). Esta línea le dice al ordenador que analice la condición. Cualquier condición se evalúa a verdadero (**true**) o falso (**false**). El ordenador ejecutará el código contenido entre la llave de apertura ({) y la llave de cierre (}) únicamente si la condición se evalúa a verdadero (**true**). En este caso sólo hay una línea de código entre las llaves, pero puede haber un montón de código.

La sentencia **if** es una de las más importantes que tenemos en programación, ya que nos permite escribir código que se ejecutará sólo en determinadas circunstancias. En este programa, el mensaje `"Coge un jersey!!"` sólo aparecerá cuando el valor de la variable `temp` sea menor que el de la variable `tempPerfecta`. En cualquier otro caso, el código que está entre llaves no se ejecuta y no sucede nada.

Sangrando líneas y usando el punto y coma

Habrá observado algunas cosas sobre la estructura de la sentencia if. Primero, la línea con la instrucción `alert` está sangrada respecto al margen. JavaScript no le echa cuentas a los espacios, sangrías, retornos de carro, pero los programadores ha aprendido a seguir algunas

normas que les facilitan leer el código de sus programas y sobre todo el de los demás. Se debe sangrar el código que se encuentre entre llaves. Algunos usan una sangría de 3 espacios, otros de 5, no importa mucho siempre que se sea consistente. A continuación de la llave de cierre, ayuda poner un comentario de `//end if` sobre todo cuando estamos trabajando con estructuras condicionales complejas. También se habrá fijado que no hay punto y coma después de la llave de cierre. Se entiende que la llave de cierre termina la sentencia y el punto y coma sería redundante.

Usando la estructura else

La sentencia **if** se usa para ejecutar código cuando se cumple una condición. A veces, es necesario ejecutar otro código cuando no se cumple la condición. Por ejemplo, puede querer llevar un jersey o una camiseta. Si hace frio llevará un jersey, si no llevará una camiseta. El programa siguiente simula esta situación.

Creando el programa “Alta o baja”

Las Figures 2.7 y 2.8 muestran la salida del programa “Alta o baja”. De nuevo, el programa genera una temperatura de forma aleatoria, a continuación muestra un mensaje acorde con la temperatura.

Figura 2.7
Figura 2.8

El programa “Alta o baja” empieza de forma similar al anterior, pero incluye una nueva capacidad. A ver si la puede localizar en el código:

```
<html>
<head>
<title>Alta o baja</title>
<script>
// Alta o baja
// Demuestra la estructura if / else
var temp = 0;
var tempPerfecta = 25;
temp = Math.floor(Math.random() * 40) + 1;
alert ("Hace " + temp + " grados fuera. ");
if (temp < tempPerfecta){
   alert("Coge un jersey!!");
} else {
   alert("Coge una camiseta!!");
}// end if
</script>
</head>
<body>
<h1>Alta o baja<br /></h1>
<hr />
<h3>Pulse el botón recargar para ver otra temperatura</h3>
</body>
</html>
```

Usando la cláusula else

Lo únicao nuevo en el programa “Alta o baja” es la cláusula **else**. Echemos un vistazo a la sentencia **if**:

```
if (temp < tempPerfecta){
   alert("Coge un jersey!!");
} else {
   alert("Coge una camiseta!!");
}// end if
```

La sentencia **if** funciona exactamente igual que en el programa anterior, pero ahora tenemos una cláusula **else**. La parte entre `else {` y la llave de cierre (}) sólo se ejecutará si la condición se evalúa a falso (**false**). Si la condición se evalúa a verdadero (**true**) el mensaje del jersey es el que aparece y el programa sigue después del else. Si la condición se evalúa a falso (**false**), se ejecutará el código del else y aparecerá el mensaje de la camiseta.

Usando estructuras if anidadas

Es habitual tener condiciones más complejas. Por ejemplo, si se quieren recomendar más de dos opciones de vestimenta. Podemos recomendar una chaqueta si hace frio, un jersey si hace fresco, una camisa si hace bueno y una camiseta si hace calor. Podemos anidar las sentencias if para gestionar estas situaciones.

Creando el programa “Varias temperaturas”

Las Figuras 2.9 a 2.12 muestran el programa “Varias temperaturas”, que muestra cuatro mensajes diferentes dependiendo del rango de temperatura que el programa elige de forma aleatoria.

Figura 2.9
Figura 2.10
Figura 2.11
Figura 2.12

Cómo vemos, esta versión del programa es capaz de presentar mensajes diferentes a cuatro rangos de temperatura. Veamos el código:

```
<html>
<head><title>Varias temperaturas</title>
<script>
// Varias temperaturas
// Demuestra la estructura if/else
var temp = 0;
temp = Math.floor(Math.random() * 40) + 1;
alert ("Hace " + temp + " grados fuera. ");

if (temp < 21){
   if (temp < 10){
      alert("Coje una chaqueta!!");
   } else {
      alert("Coje un jersey!!");
   } //end if 10
```

```
} else {
   if (temp > 30){
      alert("Coge una camiseta!!");
   } else {
      alert("Coge una camisa!!");
   } // end if 30
} // end if 21
</script>
</head>
<body><h1>Varis temperaturas<br /></h1><hr />
<h3>Pulse el botón recargar para ver otra temperatura</h3>
</body>
</html>
```

El código empieza cómo la mayoría de los programas anteriores. Una sentencia if comprueba si la temperatura es mayor de 21 grados. El programa incluye dos nuevas sentencias if que nos permiten identificar hasta cuatro rangos de temperatura.

Anidando niveles de sentencias if

La primera sentencia if comprueba si la temperatura es menor de 21 grados:

```
if (temp < 21){
```

Cualquier código entre esta línea y la llave de cierre correspondiente (}) se ejecutará sólo cuando la temperatura sea menor de 21. A continuación viene la línea:

```
if (temp < 10){
```

Este segundo if está anidado con el primero. Comprueba temperaturas por debajo de 10 grados. Cualquier código entre esta línea y la llave de cierre correspondiente sólo se ejecutará cuando la temperatura sea menor de 10 grados. (Por supuesto, debido al if anterior, el programa ya ha determinado que la temperatura es menor de 21 grados). En estas condiciones se ejecuta la línea:

```
      alert("Coje una chaqueta!!");
```

Este mensaje sólo aparece cuando la temperatura es menor de 10 grados. La línea siguiente es:

```
} else {
 alert("Coje un jersey!!");
```

El programa mostrará el mensaje del jersey cuando la temperatura esté entre 10 grados y 21 grados. La línea siguiente termina la estructura if interna:

```
} //end if 10
```

Podemos apreciar lo útil que resultan las sangrías y los comentarios cuando tenemos sentencias anidadas. Sin ellos, se hace realmente muy difícil saber en qué punto de la estructura estamos. La siguiente cláusula else es la del primer if:

```
} else {
```

Cualquier código entre esta línea y la correspondiente llave de cierre se ejecutará cuando la temperatura sea mayor o igual de 21 grados. Esta cláusula else contiene su propia sentencia if para comprobar si la temperatura cae en el rango 21 – 30 o está por encima:

```
 if (temp > 30){
    alert("Coge una camiseta!!");
} else {
    alert("Coge una camisa!!");
} // end if 30
```

Finalmente, el programa cierra la primera sentencia if con la siguiente línea:

```
} // end if 21
```

Además de la sangría y los comentarios, los editores y los IDE modernos también nos ayudan a movernos por estas estructuras anidadas haciendo más fácil el trabajo del programador.

CONSEJO – Si aún le cuesta ver cómo funcionan estas sentencias if anidadas, corra el programa con el código por delante. Cuando el programa muestre el dato de la temperatura, intente ver qué línea se ejecutará a continuación. Esta técnica es muy útil y se utiliza habitualmente para depurar programas.

Usando la estructura switch

JavaScript otra estructura que puede ser muy útil cuando tenemos una variable que puede tomar varios valores. Vamos a verlo con un programa que produce una salida gráfica un poco tosca al programa “Lanzar el dado” que hemos visto al principio del Tema:

Creando el programa “Dado aleatorio”

La Figura 2.13 muestra un programa “Lanzar el dado” más completo al que hemos llamado “Dado aleatorio”. Más adelante incorporaremos mejores gráficos, por ahora nos vale para ver cómo generar algunas imágenes básicas. Cómo podemos ver, el programa genera un número aleatorio y, a continuación, dibuja una imagen del dado formada por texto en un cuadro de diálogo. Este es el código, aunque parece muy largo en el fondo no es más que repeticiones del mismo código básico.

Figura 2.13

```
<html>
<head><title>Dado aleatorio</title>
<script>
// Dado aleatorio
// Demuestra la sentencia switch

var lanzamiento = 0;
var dado = "";
lanzamiento = Math.floor(Math.random() * 6) + 1;
switch (lanzamiento){
case 1:
```

```
dado = "|-------|\n";
dado +="|       |\n";
dado +="|   *   |\n";
dado +="|       |\n";
dado +="|-------|\n";
break;
case 2:
dado = "|-------|\n";
dado +="|   *   |\n";
dado +="|       |\n";
dado +="|   *   |\n";
dado +="|-------|\n";
break;
case 3:
dado = "|-------|\n";
dado +="|   *   |\n";
dado +="|   *   |\n";
dado +="|   *   |\n";
dado +="|-------|\n";
break;
case 4:
dado = "|-------|\n";
dado +="|  * *  |\n";
dado +="|       |\n";
dado +="|  * *  |\n";
dado +="|-------|\n";
break;
case 5:
dado = "|-------|\n";
dado +="|  * *  |\n";
dado +="|   *   |\n";
dado +="|  * *  |\n";
dado +="|-------|\n";
break;
case 6:
dado = "|-------|\n";
dado +="|  * *  |\n";
dado +="|  * *  |\n";
dado +="|  * *  |\n";
dado +="|-------|\n";
break;
default: dado = "ERROR!"
} // end switch
alert(dado);
</script>
</head>
<body>
<h1>Dado aleatorio<br /></h1>
<hr /><h3>Pulsar recargar para ver otro valor</h3>
</body>
</html>
```

El código empieza de forma ya familiar con la declaración de variables:

```
var lanzamiento = 0;
```

```
var dado = "";
lanzamiento = Math.floor(Math.random() * 6) + 1;
```

Estas líneas declaran las variables y obtienen un valor aleatorio por el método que hemos visto en programas anteriores. La variable dado contiene caracteres que simulan la cara de una dado.

Usando la sentencia switch

Lo nuevo empieza con la sentencia `switch`:

```
switch (lanzamiento){
```

Esta sentencia acepta el nombre de una variable. Le dice al ordenador que empiece a pensar en el valor de la variable `lanzamiento`. Acaba con una llave de apertura que indica el inicio de una estructura lógica. El resto de la estructura analiza posibles valores de la variable:

```
case 1:
```

Esta línea comprueba si el valor de `lanzamiento` es igual a 1. El código entre esta línea y la siguiente sentencia `case` se ejecuta si `lanzamiento` es igual a 1. El código actual no es más que una serie de asignaciones de texto a la varible `dado`:

```
dado = "|-------|\n";
```

Esta línea asigna un valor para la línea superior del dado. El carácter `\n` es un carácter especial que significa nueva línea. Las demás líneas son muy similares:

```
dado +="|       |\n";
dado +="|   *   |\n";
dado +="|       |\n";
dado +="|-------|\n";
```

La única diferencia es el uso del operador +=. Esto no es más que un atajo:

```
dado +="|        |\n";
```

es exactamente lo mismo que

```
dado = dado + "|        |\n";
```

Usando la sentencia the break

Al final de cada sentencia `case`, veremos una línea cómo la siguiente:

```
break;
```

La sentencia `break` le dice al ordenador que salte al final de la estructura `switch`.

CUIDADO – No olvide poner una sentencia break al final de cada bloque case. Si no, el ordenador evaluará el siguiente bloque case. Este es un error bastante frecuente, sobre todo en programadores de otros lenguajes que no lo necesitan.

Usando la cláusula default

Podemos añadir una claúsula `default` a una estructura `switch`. El ordenador ejecuta la clásula `default` únicamente cuando ninguna de las demás condiciones es verdadera. Hemos añadido una cláusula `default` a nuestra estructura `switch`, aunque el ordenador no debería ejecutarla nunca. A pesar de esto, es conveniente incluirla aunque esperemos que no se ejecute nunca. Así, el programa estará preparado para lo inesperado.

Table 2.3 – Displaying dynamic content

Volviendo al programa “El Oráculo”

Finalmente estamos listos para examinar el programa “El Oŕaculo” del inicio del Tema. Resultará muy sencillo después de todo lo que hemos aprendido. De hecho, no es más que le programa “Lanzar el dado”, usando mensajes en lugar de imágenes. Este es el código:

```
<html>
<head>
<title>El Oráculo</title>
<script>
// El Oráculo

var lanzamiento = 0;
var futuro = "";
lanzamiento = Math.floor(Math.random() * 5) + 1;
switch(lanzamiento){
case 1:
futuro = "El futuro pinta muy negro.";
break;
case 2:
futuro = "El futuro no es nada alagüeño.";
break;
case 3:
futuro = "El futuro pinta así así.";
break;
case 4:
futuro = "El futuro pinta bien.";
break;
case 5: futuro = "Estas en un día perfecto!";
break;
default:
futuro = "ERROR";
} // end switch
alert(futuro);
</script>
</head>
<body>
<h1>El Oráculo<br /></h1>
<hr />
<h3>Pulsar recargar para otra predicción</h3>
</body>
</html>
```

Resumen

En este Tema, hemos aprendido a generar números aleatorios y a acotarlos dentro de un rango específico. Hemos aprendido cómo el ordenador maneja la lógica. Hemos visto varios ejemplos de condiciones y cómo los programadores las utilizan en las sentencias `if`. Hemos visto variantes más elaboradas de `if` con la cláusula `else`, sentencias `if` anidadas, y la estructura `switch`. También hemos visto propiedades y métodos de la clase **Math**.

EJERCICIOS – Tema 2

1. Escribir un sencillo programa que haga un examen con cinco preguntas sobre JavaScript. Calcular los aciertos y los errores del usuario. Mostrar los resultados y un comentario en función del resultado.

2. Escribir un programa que presente una combinación de la lotería primitiva (6 números del 1 al 49).

3. Escribir el programa “Lanzar el dado”, con un dado trucado. La mitad de las veces da un resultado aleatorio y la otra mitad da 6.

4. Algunos juegos necesitan más de un dado. Escribir un programa que pregunte al usuario cuántos dados quiere (1 a 4). Muestre tantos resultados cómo dados haya pedido el usuario.