LA PROGRAMACIÓN: INICIATE EN UN MUNDO APASIONANTE

5. LIBRERÍAS DE CÓDIGO

Este módulo trata sobre librerías de código. Primero vamos a ver cómo podemos construir una librería de código por nosotros mismos. Y después veremos algunos ejemplos con las librerías de código que nos ofrece el sistema.

Librerías de código

Hasta ahora hemos conocido la estructura de datos de diccionario que nos permitía almacenar datos como pares de *clave : valor*. Pero esta misma estructura también nos permite almacenar funciones. Volviendo a nuestro ejemplo de la baraja de cartas, vamos a modelar la figura del croupier como un diccionario que almacena dos funciones: `barajar` y `repartir`.

Ejemplo 1: librería Croupier

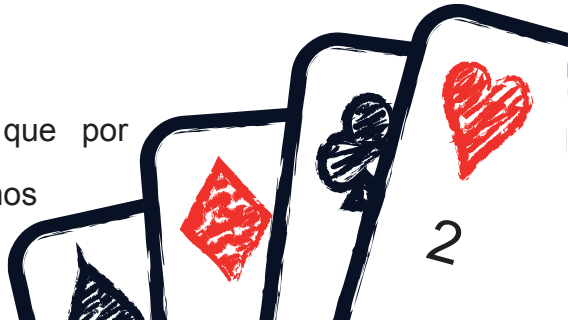
```
var cartas = [{palo: "c", valor: 1},
              {palo: "d", valor: 1},
              {palo: "p", valor: 1},
              {palo: "t", valor: 1}];

function barajar(cartas){
  return "Barajando…";
}

function repartir(cartas){
  return "Su carta, gracias";
}

var Croupier = {baraja: barajar, reparte: repartir};
Croupier.baraja(cartas);
Croupier.reparte(cartas);
```

Dada una colección de cartas, que por simplicidad son sólo los 4 ases, vamos



a definir dos funciones que un croupier puede realizar sobre ellas: *barajar* y *repartir*. En este ejemplo, no están implementadas de verdad las funciones y sólo devuelven una cadena de texto a modo explicativo de lo que hacen. La implementación real se deja como ejercicio al alumno del curso.

Definimos luego la variable *Croupier*, para la que usamos un nombre en mayúsculas que es una convención que se usa en muchos lenguajes de programación para denotar un objeto. Para nuestro croupier elegimos una estructura de datos de diccionario para almacenar las funciones anteriores. Elegimos un nombre para las claves (*baraja*, *reparte*) y referenciamos a las funciones que hemos declarado anteriormente por su nombre.

Librerías del sistema

Vamos a ver algunas funciones útiles de la librería estándar de JavaScript. Las iremos conociendo con ejemplos.

console

Ejemplo 2: console

```
console.log("Hola, buenas tardes");


console.clear();
```

En este caso, vamos a conocer las funcionalidades de la librería *console* que ya conocemos de otros módulos. Con la función *log* podemos imprimir un texto en la consola. Y con *clear* borrar la consola, del mismo modo que usando el botón con el mismo nombre. Esta función en el navegador Firefox no está implementada en *console* sino que se ejecuta escribiendo directamente *clear()*.

Date

Ejemplo 3: Date

```
Date.now();


Date.parse("01/02/2007");
Date.parse("2007-01-02");
Date.parse("Jan 2, 2007");
```

La función *now* de la librería *Date* nos devuelve la fecha actual. Pero el formato en que nos lo devuelve es curioso: es un número muy largo. Ese número corresponde al número de milisegundos (milésimas de segundo) que han pasado desde el 1 de enero de 1970. Lo llamamos *timestamp* o marca de tiempo, porque indica una fecha y hora con precisión de milisegundos. Esta es la forma en que JavaScript trata las fechas de forma interna.

La función *parse* analiza una cadena de texto que le pasamos como parámetros y nos devuelve el timestamp de esa fecha. Admite distintos tipo de formatos de fecha, algunas con texto o también sólo números.

Math

Ejemplo 4: Math

```
Math.sqrt(9);
Math.sqrt(9786);
var numeroGrande = 30000000;
Math.sqrt(numeroGrande);


Math.PI
```

Para terminar los ejemplo de la librería estándar de JavaScript, indagaremos en la librería de matemáticas *Math*. Con ella podremos realizar cálculo matemáticos como la raíz cuadrada o *square root* en inglés, con la función *sqrt*. Para usarla, simplemente invocamos este método pasándole como parámetros el número al que realizar la raíz cuadrada.

Además de funciones, también podemos almacenar valores constantes en una librería. En *Math* podremos acceder al número *PI* si lo necesitamos para realizar algún cálculo matemático. Como convención, a la hora de nombrar constantes siempre se hace con su nombre completo en mayúsculas. Nota que, al tratarse de una constante y no una función, para acceder a ella no utilizamos paréntesis.