Resumen Unidad 3

- JavaScript aporta varios tipos de estructuras de datos para poder gestionar datos de forma más cómoda.
- Los **arrays** permiten asociar una colección de datos, incluso heterogéneos, a una misma estructura. El acceso al array se realiza mediante una variable que almacena una referencia o enlace a dicho array.
- Para acceder a un elemento individual de un array se utilizan los corchetes y el número de índice del elemento. El primer elemento tiene índice 0, el siguiente el 1 y así sucesivamente.
- Aunque declaremos un array como constante, podemos modificar el contenido del array, ya que lo que almacena la variable es una referencia al propio array. Por tanto, lo que no vamos a poder modificar, es la referencia a ese array, es decir, esa variable declarada como constante no va a poder apuntar a otro array.
- Los arrays son realmente objetos que poseen métodos ya preparados para realizar tareas sobre ellos como ordenar, devolver el tamaño del array, eliminar elementos, añadir elementos, etc.
- El recorrido de los arrays se puede realizar usando un contador que recorra cada índice a través de un bucle **for** normal. Pero disponemos de bucles más apropiados para recorrer arrays, concretamente **for..in** y **for..of**.
- Los **conjuntos**, también conocidos como sets, son una estructura que sirve para almacenar colecciones de datos. La característica diferencial de los conjuntos es que no permite tener valores duplicados. Además, hay más cosas que diferencian a los conjuntos de los arrays, por ejemplo, que los conjuntos no tienen índices.
- Los conjuntos también son un tipo de objetos en JavaScript, y como tal también tiene sus propiedades y métodos para añadir elementos, borrar elementos, buscar valores, etc.
- Para recorrer los conjuntos se utiliza el bucle **for..of**.
- Los **mapas** son una estructura de datos que almacena en cada posición una dupla clave-valor, donde las claves son únicas, no se pueden repetir.
- Los mapas también son objetos con sus propiedades y métodos. También los bucles **for..of** son los idóneos para recorrer estas estructuras.
- Tenemos un método que también podemos usar para recorrer arrays, conjuntos y mapas, es el método **forEach**, el cual recibe como parámetro una **función callback**.
- Las funciones son también un tipo de objetos de JavaScript, que facilitan la modularidad a la hora de programar aplicaciones. Cada función será capaz de resolver una tarea sencilla a partir de una serie de datos que la función requiere para poder ejecutar su tarea.
- Las funciones se declaran con la palabra clave **function** a la que sigue el nombre de la función, los parámetros de la misma (que son los datos que requiere para poder resolver la tarea) y el cuerpo de la función (que es el código que la función ejecuta cuando se la invoque).
- En el cuerpo de la función puede aparecer la palabra **return** que es la que permite que la función devuelva un valor concreto. No es obligatorio que aparezca, ya que

hay funciones que realizan una tarea, pero no requieren devolver ningún valor en concreto.

- Hay **funciones anónimas** cuyo cuerpo se puede asignar a variables o parámetros de funciones, que pasan a ser referencias a la propia función.
- Las **funciones flecha** permiten definir funciones de forma más cómoda sin tener que utilizar la palabra **function** ni, en muchas ocasiones, tener que indicar explícitamente la palabra **return**. Son cómodas para funciones simples, pero no tanto para funciones complejas.
- Cuando una función se invoca indicando tipos simples de datos (números, booleanos, strings, etc.) en los parámetros, estos reciben una copia de esos valores. Pero si enviamos arrays u otros objetos, entonces los parámetros obtendrán una referencia al objeto original. En este último caso, si se modifica el objeto en la función, se estará modificando el objeto original y no una copia.
- JavaScript admite parámetros de funciones con valores predeterminados y, también definir parámetros con el operador de propagación para un número indeterminado de parámetros.
- Cada vez que invocamos funciones, el código de esas funciones se carga en una pila que tiene un tamaño finito y que podemos colapsar (desbordamiento de pila) en caso de excesivas llamadas a funciones sin resolver.
- La **recursividad** en una técnica que permite que una función se invoque a sí misma. Con habilidad permite resolver de forma sencilla, problemas complejos. Aunque puede ser una técnica muy útil hay que tener en cuenta que la iteratividad por norma general es más eficiente que la recursividad.
- Las **funciones callback**, son funciones que se pasan como parámetros a otras funciones.
- Algunos de los métodos que utilizan funciones callback son los métodos **sort** o el método **forEach**.