```
/* 5--- Método reduce(): ---

El método reduce() ejecuta una función reductora sobre cada elemento de un array,
devolviendo como resultado un único valor.

Sintaxis:

arr.reduce(callback(acumulador, valorActual[, índice[, array]])[, valorInicial])

Parámetros:
callback:
Función a ejecutar sobre cada elemento del array (excepto para el primero, si no se
proporciona valorInicial), que recibe cuatro parámetros:
acumulador:
El acumulador acumula el valor devuelto por la función callback. Es el valor
acumulado devuelto en la última invocación de callback, o el valorInicial, si se
proveyó. (Ver a continuación).
valorActual:
El elemento actual que está siendo procesado en el array.
índice Opcional:
El índice del elemento actual que está siendo procesado en el array. Empieza desde
el índice 0 si se provee valorInicial. En caso contrario, comienza desde el índice
1.
array Opcional:
El array sobre el cual se llamó el método reduce().
valorInicial Opcional:
Un valor a usar como primer argumento en la primera llamada de la función callback.
Si no se proporciona el valorInicial, el primer elemento del array será utilizado y
saltado.

Llamando a reduce() sobre un array vacío sin un valorInicial lanzará un
TypeError.Descripción
*/
const students = [
    {
        name: "Jill",
        lastname: "Doe",
        age: 23,
        course: "Marketing",
    },
    {
        name: "John",
        lastname: "Doe",
        age: 20,
        course: "Web Development",
    },
    {
        name: "Jack",
        lastname: "Doe",
        age: 22,
        course: "Accounting",
    },
    {
        name: "Ryan",
        lastname: "Ray",
        age: 20,
        course: "Web Development",
    },
    {
        name: "Jane",
        lastname: "Doe",
        age: 21,
```

```
        course: "Financial Management",
    },
];

// Vamos a obtener la suma de todas las edades:
// Ciclo for() no es ideal:
let total = 0;
for (let i = 0; i < students.length; i++) {
    total += students[i].age;
    // console.log(total);
}

console.log(total);

// Método reduce():
let result = students.reduce((total, student) => {
    return total + student.age;
}, 0)

console.log(result);

// Vamos a verlo con otro arreglo, developers: Queremos obtener un arreglo que tenga todas las habilidades en uno solo.
const developers = [
    {
        id: 1,
        name: "John",
        skills: ["HTML", "CSS", "Java", "Angular", "Redux", "NodeJS"],
    },
    {
        id: 2,
        name: "Jane",
        skills: ["HTML", "CSS", "JavaScript", "React", "Redux", "NodeJS"],
    },
    {
        id: 3,
        name: "Jack",
        skills: ["HTML", "CSS", "JavaScript", "React", "Redux", "NodeJS"],
    },
];

const reducer = developers.reduce((allSkills, student) => {
    return Array.from(new Set([ ...allSkills,  ...student.skills]))
}, [])

console.log(reducer);

/*
Aquí hemos aprendido como con el objeto Set():
El objeto Set permite almacenar valores únicos de cualquier tipo, incluso valores primitivos o referencias a objetos. Nos permite guardar todos los elementos que no se repiten. Si alguno se repite lo descarta.
También hemos visto cómo ese objeto se puede transformar en un Array con el método Array.from()
Debemos profundizar más en estos métodos porque se ven muy interesantes.
*/
```