**ARROW FUNCTIONS**

1. Convertí las siguientes funciones a arrow functions

**var sumarLosTres = function (num1, num2, num3) {**

**return num1 + num2 + num3;**

**}**

**let cuadrado = function (num) {**

**return num \* num;**

**}**

**const decirHola = function () {**

**console.log('¡Hola!');**

**}**

**/// SOLUCION**

**const sumarLosTres = (num1, num2, num3) => num1 + num2 + num3;**

**const cuadrado = num => num \* num;**

**const decirHola = () => console.log('¡Hola!');**

2. Convertí la siguiente función a una arrow functions

**function saludar (nombre) {**

**if (nombre === undefined) {**

**return "hola anónimo";**

**} else {**

**return "hola " + nombre;**

**}**

**}**

**// SOLUCION**

**const saludar = nombre => {**

**if (nombre === undefined) {**

**return "hola anónimo";**

**} else {**

**return "hola " + nombre;**

**}**

**}**

3. Arreglá las siguientes arrow functions

**const nombreCompleto = nombre, apellido => {**

**return `${nombre} ${apellido}`;**

**}**

**const exclamar = str => {**

**`¡${str}!`**

**}**

**const mayusculas = (str) => return str.toUpperCase()**

**const minusculas = (str) => str.toLowerCase()**

**console.log( nombreCompleto('Ada', 'Lovelace') ); // Ada Lovelace**

**console.log( exclamar('Hedy') ); // ¡Hedy!**

**console.log( mayusculas('grace') ); // GRACE**

**console.log( minusculas('SHERYL') ); // sheryl**

**// SOLUCION**

**const nombreCompleto = (nombre, apellido) => `${nombre} ${apellido}`;**

**const exclamar = str => `¡${str}!`;**

**const mayusculas = str => str.toUpperCase();**

**const minusculas = str => str.toLowerCase();**

**console.log( nombreCompleto('Ada', 'Lovelace') ); // Ada Lovelace**

**console.log( exclamar('Hedy') ); // ¡Hedy!**

**console.log( mayusculas('grace') ); // GRACE**

**console.log( minusculas('SHERYL') ); // sheryl**

4. La siguiente función multiplica todos los números por 3 y finalmente filtra solo aquellos que sean múltiplos de 5 (más adelante veremos estos métodos que utilizamos en la función). Mejorá el siguiente código para reemplazar las funciones con **arrow functions**.

**function tripleAndFilter(arr) {**

**return arr.map(function (n) {**

**return n \* 3;**

**}).filter(function (n) {**

**return n % 5 === 0;**

**})**

**}**

**tripleAndFilter([1, 2, 3, 4, 5]);**

**// [ 15 ]**

**/// SOLUCION**

**// const tripleAndFilter = arr => {**

**// return arr.map(n => n \* 3).filter(n => n % 5 === 0);**

**// };**

**const tripleAndFilter = arr => arr.map(n => n \* 3).filter(n => n % 5 === 0);**

**tripleAndFilter([1, 2, 3, 4, 5]);**

5. Crear una una arrow function que se llame **multiplicar**, que recibe dos números y retorna la multiplicación de ambos.

**console.log( multiplicar(2, 3) ); // 6**

**console.log( multiplicar(5, 5) ); // 25**

**console.log( multiplicar(4, 9) ); // 36**

**// SOLUCION**

**const multiplicar = (a, b) => a \* b;**

6. Crear una una arrow function que se llame **multiplicarPor2**, que recibe un número y retorna la multiplicación de ese número por 2. Para eso, tenemos que utilizar la función que creamos en el punto anterior.

**console.log( multiplicarPor2(2) ); // 4**

**console.log( multiplicarPor2(5) ); // 10**

**console.log( multiplicarPor2(4) ); // 8**

**// SOLUCION**

**const multiplicarPor2 = (a) => multiplicar(a, 2);**

**DEFAULT PARAMETERS**

1. Modificá la siguiente función, agregando parámetros por default, para que al ejecutar los casos de prueba no tire ningún error.

**const longitudDelNombre = (nombre) => nombre.length**

**console.log( longitudDelNombre() ); // 0**

**console.log( longitudDelNombre('Ada') ); // 3**

**// SOLUCION**

**const longitudDelNombre = (nombre = '') => nombre.length**

**console.log( longitudDelNombre() ); // 0**

**console.log( longitudDelNombre('Ada') ); // 3**

2. ¿Cómo modificarías la siguiente función para seguir obteniendo el mismo resultado, pero con un código más reducido?

**const saludarVisitanteWeb = (nombreUsuario) => {**

**if (nombreUsuario === undefined) {**

**return '¡Hola anónimo!';**

**} else {**

**return `¡Hola ${nombreUsuario}!`;**

**}**

**}**

**console.log(saludarVisitanteWeb()); // ¡Hola anónimo!**

**console.log(saludarVisitanteWeb('Hedy')); // ¡Hola Hedy!**

**// SOLUCION**

**const saludarVisitanteWeb = (nombreUsuario = 'anónimo') => `¡Hola ${nombreUsuario}!`;**

**console.log(saludarVisitanteWeb()); // ¡Hola anónimo!**

**console.log(saludarVisitanteWeb('Hedy')); // ¡Hola Hedy!**

3. ¿Cómo modificarías la siguiente función para seguir obteniendo el mismo resultado, pero con un código más reducido, y que no tire un error cuando no se le pasa un parámetro?

**function primerElemento (arr) {**

**return arr[0];**

**}**

**console.log(primerElemento([])); // undefined**

**console.log(primerElemento([3, 2, 1])); // 3**

**console.log(primerElemento()); // VM224:2 Uncaught TypeError: Cannot read property '0' of undefined**

**// SOLUCION**

**const primerElemento = (arr = []) => arr[0];**

**console.log(primerElemento([])); // undefined**

**console.log(primerElemento([3, 2, 1])); // 3**

**console.log(primerElemento()); // undefined**

4. Crear una función llamada **nombreCompleto**, que recibe dos parámetros (nombre y apellido), y retorna la concatenación de ambas variables. Tanto nombre como apellido tienen que tener un valor default de string vacío. Utilizar arrow funcions y template literals.

**// SOLUCION**

**const nombreCompleto = (nombre = '', apellido = '') => `${nombre} ${apellido}`;**