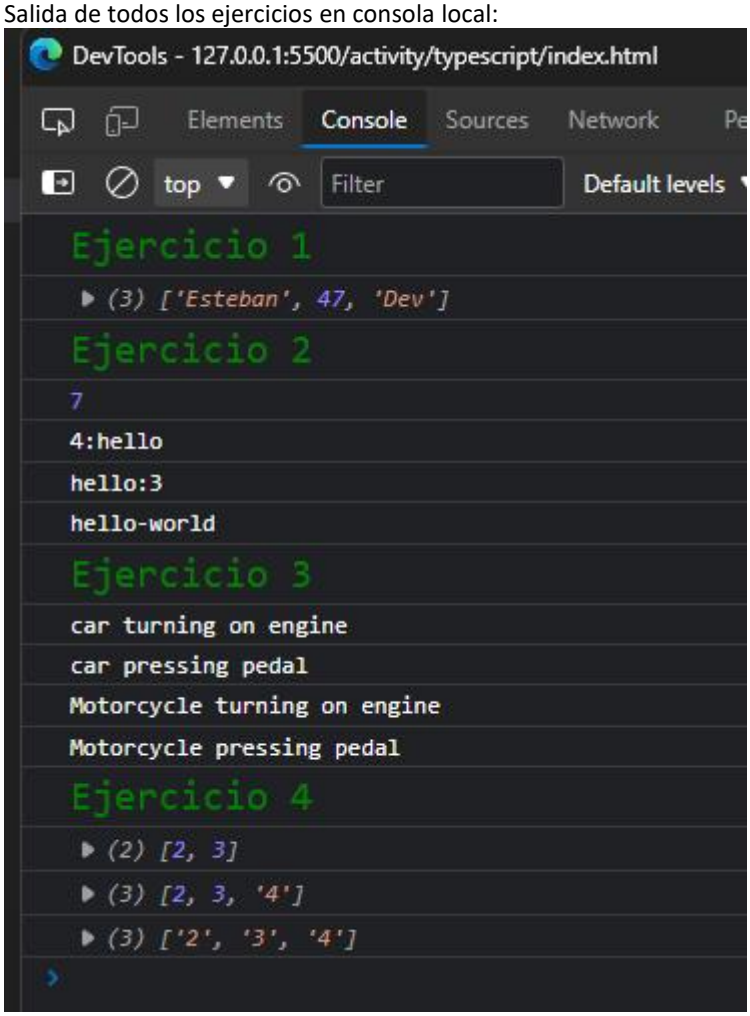
Actividad Sesión 3

Alumno: Esteban Antonio Castro Rojas
Repositorio: https://github.com/esancaro/Javascript-Course/activity/typescript
Fecha: 2022-07-12

Compilación en vsCode

Comando utilizado: tsc --target ESNext index.ts

Salida de todos los ejercicios en consola local:



Ejercicio 1

Crea una interfaz 'Person' que tenga como atributos 'name', 'age' y 'profession'. Ahora define una arrow function que tenga como parámetro esta interfaz y que devuelva un array con el valor de sus propiedades, esta función tiene que tener tipado el parámetro de entrada y el return.

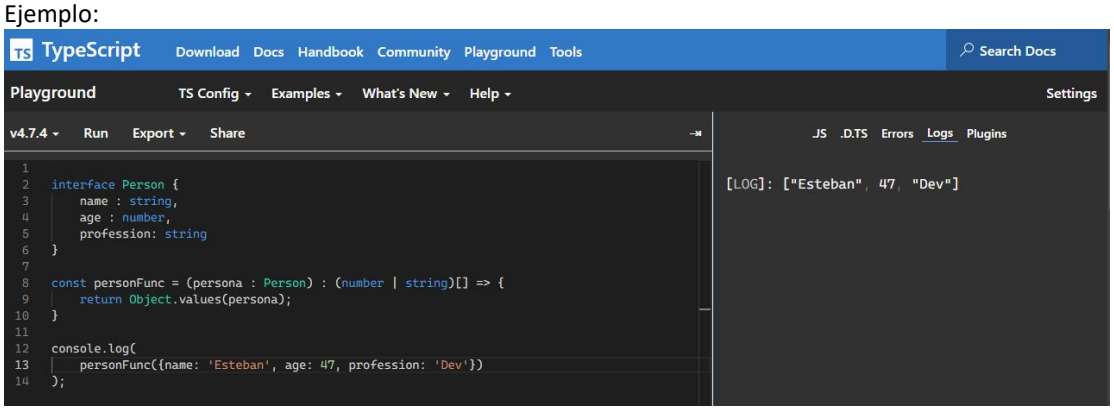
Código:

```
interface Person {
    name: string;
    age: number;
    profession: string;
}

const personFunc = (persona: Person): (number | string)[] => {
    return Object.values(persona);
};
```

```
console.log(personFunc({ name: "Esteban", age: 47, profession:
"Dev" }));
```

Ejemplo:



Ejercicio 2

Escribe una función llamada sumOrConcatenate que acepte dos parámetros. Cada uno de estos parámetros podrá ser de tipo number o string. La función devolverá una suma si los dos parámetros son números, una concatenación con el símbolo - si son los dos strings o una cadena concatenada con : si uno es un number y el otro string.

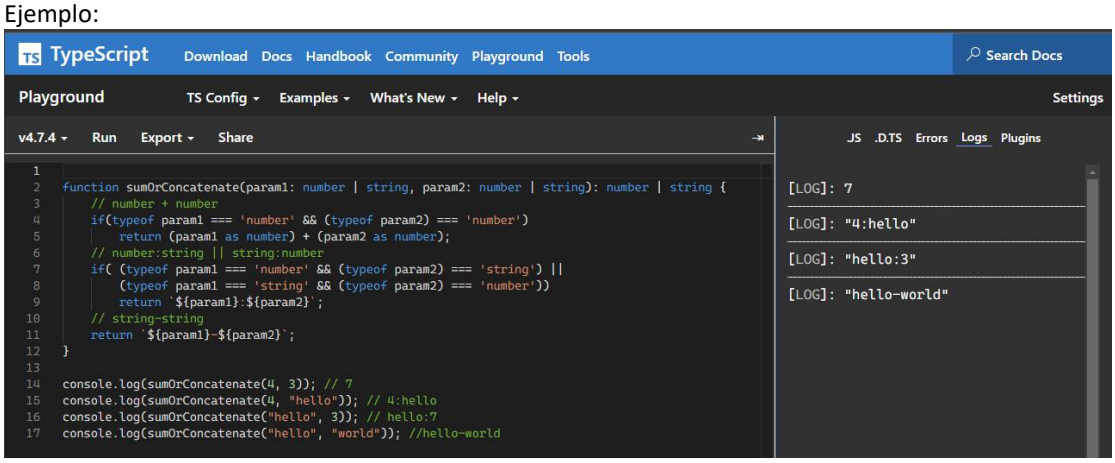
Código:

```
function sumOrConcatenate(
  param1: number | string,
  param2: number | string
): number | string {
  // number + number
  if (typeof param1 === "number" && typeof param2 === "number")
    return (param1 as number) + (param2 as number); // number:string ||
string:number
  if (
    (typeof param1 === "number" && typeof param2 === "string") ||
    (typeof param1 === "string" && typeof param2 === "number")
  )
    return `${param1}:${param2}`; // string-string
  return `${param1}-${param2}`;
```

```
}

console.log(sumOrConcatenate(4, 3)); // 7
console.log(sumOrConcatenate(4, "hello")); // 4:hello
console.log(sumOrConcatenate("hello", 3)); // hello:7
console.log(sumOrConcatenate("hello", "world")); //hello-world
```

Ejemplo:



Ejercicio 3

Crea dos interfaces, una llamada Car y otra Motorcycle. La primera tendrá las propiedades tires (number), turnOnEngine() (función que devuelve void) y pressPedal() (función que devuelve void). La segunda tendrá las propiedades tires (number), turnOnEngine() (función que devuelve void) y openThrottle() (función que devuelve void). Escribe una función que acepte un parámetro que pueda ser Car o Motorcycle que, primero llame a turnOnEngine, y luego si es Car llame a pressPedal pero si es Motorcycle llame a openThrottle().

Código:

```
interface Car {
  tires: number;
  turnOnEngine: () => void;
  pressPedal: () => void;
}

interface Motorcycle {
  tires: number;
  turnOnEngine: () => void;
  openThrottle: () => void;
}
```

```
// type predicate
const isCar = (vehicle: Car | Motorcycle): vehicle is Car => {
  return (vehicle as Car).pressPedal != undefined;
};
```

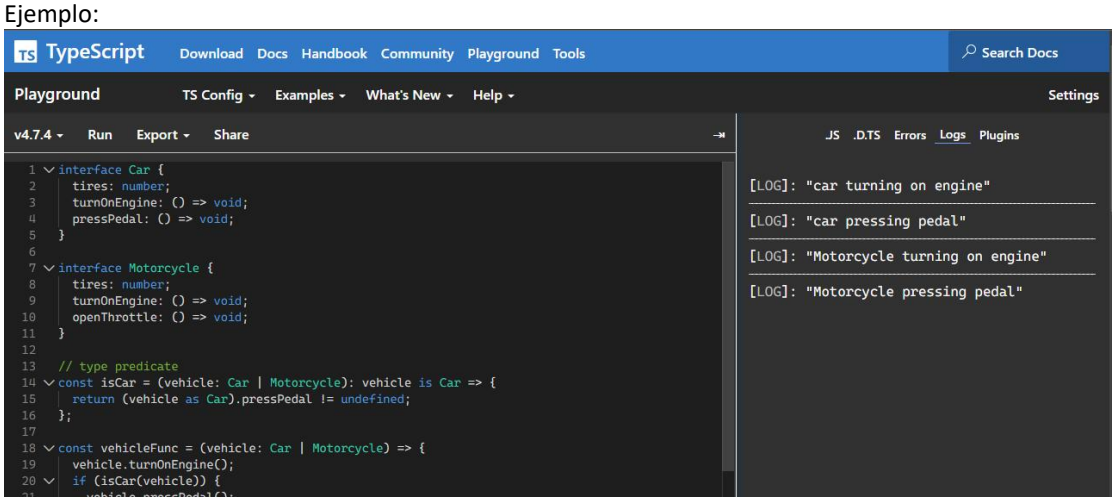
```
const vehicleFunc = (vehicle: Car | Motorcycle) => {
  vehicle.turnOnEngine();
```

```
  if (isCar(vehicle)) {
    vehicle.pressPedal();
  } else {
    vehicle.openThrottle();
  }
};
```

```
vehicleFunc(
  // Car
  {
    tires: 4,
    turnOnEngine: () => {
      console.log("car turning on engine");
    },
    pressPedal: () => {
      console.log("car pressing pedal");
    },
  }
);
```

```
vehicleFunc(
  // Motorcycle
  {
    tires: 2,
    turnOnEngine: () => {
      console.log("Motorcycle turning on engine");
    },
    openThrottle: () => {
      console.log("Motorcycle pressing pedal");
    },
  }
);
```

Ejemplo:



Ejercicio 4

Crea una función genérica, que acepte un array que pueda contener strings y numbers y devuelva el mismo array sin el primer elemento.

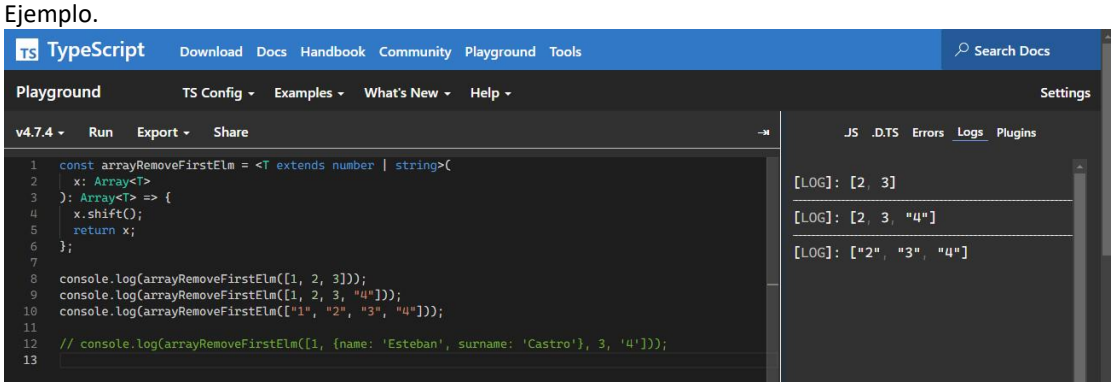
Código.

```
// Ejercicio 4
const arrayRemoveFirstElm = <T extends number | string>(
  x: Array<T>
): Array<T> => {
  x.shift();
  return x;
};

console.log(arrayRemoveFirstElm([1, 2, 3]));
console.log(arrayRemoveFirstElm([1, 2, 3, "4"]));
console.log(arrayRemoveFirstElm(["1", "2", "3", "4"]));
```

```
// console.log(arrayRemoveFirstElm([1, {name: 'Esteban', surname:
'Castro'}, 3, '4']));
```

Ejemplo.



Ejemplo de error.