EJERCICIOS DE TYPESCRIPT

EJERCICIO 1 – Constantes numéricas y JSON

Definir una función que recibe un parámetro de cualquier tipo y devuelve un string. Se debe indicar el tipo en los parámetros y valor devuelto de la función.

La función dice si es convertible a número o no ese parámetro. Para ello haremos uso de la constante NaN o las funciones relacionadas con ella. Los posibles textos que devuelve la función se construirán con una **plantilla de texto** y son:

"El parámetro xxxxxxx sí es un número"

o

"El párametro xxxx NO es un número".

Haremos una miniprograma para probar la función, de forma que el resultado de la función lo mostraremos por consola.

SOLUCIÓN:

```
function esConvertibleNumero(x: any): string{
   let paramMostrar = x;
   if (typeof    x === "object")
      paramMostrar = JSON.stringify(x); //convierte un array de json o un json a string
   return  isNaN(Number(x)) ? `El parámetro ${paramMostrar} NO es un número` : `El
parámetro ${paramMostrar} SÍ es un número`;
}

document.body.innerHTML += "Con un número <br>"+esConvertibleNumero(3)+"<br>";
document.body.innerHTML += "Con un string no número <br>" + esConvertibleNumero("3A3") +
                           "<br>";
document.body.innerHTML += "Con un string que es numérico <br>"+
                           esConvertibleNumero("3333") +"<br>";
document.body.innerHTML += "Con un objeto json <br>"+esConvertibleNumero({a:3,b:6}) +
                           "<br>";
```

EJERCICIO 2– Constantes numéricas y JSON

Usando la función anterior, crear un programa en el que tengáis un array de tipo any: con objetos, números, strings,… y debéis imprimir en pantalla si cada elemento del array es convertible a número. Por ejemplo:

Si tenemos este array:

```
let arrayVariado = [
   { nombre: "Pedro", edad: 24 },
   { nombre: "María", edad: 18 },
   45,
   "9.876",
   [3, 4, 5],
   "hola"
];
```

Nos debería escribir en consola:

```
Array(6)

0:"El parámetro {"nombre":"Pedro","edad":24} NO es un número"
1:"El parámetro {"nombre":"María","edad":18} NO es un número"
2:"El parámetro 45 SÍ es un número"
3:"El parámetro 9.876 SÍ es un número"
4:"El parámetro [3,4,5] NO es un número"
5:"El parámetro hola NO es un número"
```

SOLUCIÓN 1: usando el método "map" de los arrays

```
function esConvertibleNumero(x: any): string {
   let paramMostrar = x;
   if (typeof x === "object")
      paramMostrar = JSON.stringify(x);
   return isNaN(Number(x)) ? `El parámetro ${paramMostrar} NO es un número` : `El
parámetro ${paramMostrar} SÍ es un número`;
}

let arrayVariado = [
   { nombre: "Pedro", edad: 24 },
   { nombre: "María", edad: 18 },
   45,
   "9.876",
   [3, 4, 5],
   "hola"
]

let arrayResultados = arrayVariado.map(esConvertibleNumero);
console.log(arrayResultados);
```

OTRAS FORMA EQUIVALENTE QUE SE PODRÍA REALIZAR CON MAP SERÍAN:

a) **Forma 1: llamando a map sobre el propio array**

```
let arrayResultados2=[
            { nombre: "Pedro", edad: 24 },
            { nombre: "María", edad: 18 },
            45,
            "9.876",
            [3, 4, 5],
            "hola",
            "adios"
        ].map(esConvertibleNumero);
        console.log(arrayResultados2);
```

b) **Forma 2: Implementando directamente la función esConvertibleNumero() dentro del map:**

```
let arrayResultados3 = arrayVariado.map(
                (x: any) => {
                let paramMostrar = x;
                if (typeof x === "object")
                    paramMostrar = JSON.stringify(x);
                return isNaN(Number(x)) ? `El parámetro ${paramMostrar}
NO es un número` : `El parámetro ${paramMostrar} SÍ es un número`;
            });
    console.log(arrayResultados3);
```

SOLUCIÓN 2: usando el método "forEach" de los arrays. Ojo, este método no me devolverá el array, por tanto, debo modificar mi función para mostrar en pantalla. Fijaros:

```
function esConvertibleNumero2(x: any): void {
    let paramMostrar = x;
    if (typeof x === "object")
        paramMostrar = JSON.stringify(x);
    console.log(isNaN(Number(x)) ? `El parámetro ${paramMostrar} NO es un núme
ro` : `El parámetro ${paramMostrar} SÍ es un número`);
}
```

```
let arrayVariado2 = [
    { nombre: "Pedro", edad: 24 },
    { nombre: "María", edad: 18 },
    45,
    "9.876",
    [3, 4, 5],
    "hola"
]
arrayVariado2.forEach(esConvertibleNumero2);
```

Salida obtenida:

```
El parámetro {"nombre":"Pedro","edad":24} NO es un número        VM242:5
El parámetro {"nombre":"María","edad":18} NO es un número        VM242:5
El parámetro 45 SÍ es un número                                  VM242:5
El parámetro 9.876 SÍ es un número                               VM242:5
El parámetro [3,4,5] NO es un número                             VM242:5
El parámetro hola NO es un número                                VM242:5
```

SOLUCIÓN 3: usando el método "filter" de los arrays para que me muestre, por ejemplo sólo los que sí sean convertibles a número. Fijaros:

```
function esConvertibleNumero3(x: any): boolean {
    return isNaN(Number(x))===false;
}

let arrayVariado3 = [
    { nombre: "Pedro", edad: 24 },
    { nombre: "María", edad: 18 },
    45,
    "9.876",
    [3, 4, 5],
    "hola"
]

let arrayResultados3=arrayVariado3.filter(esConvertibleNumero3);
console.log(arrayResultados3);
```

Salida obtenida:

[45, '9.876']

EJERCICIO 3– Ejemplo de uso de map

Crear un programa que convierta un array de cadenas de texto a mayúsculas:

SOLUCIÓN:

```
let arrayCadenas:string[] = ["Enero", "Febrero", "marzo", "abril", "marzo"];

let arrayMayusculas = arrayCadenas.map(cad => cad.toUpperCase());
console.log(arrayMayusculas);
```

OTRA FORMA MUY INTERESANTE (la más abreviada):

```
console.log(["Enero", "Febrero", "marzo", "abril", "marzo"].map(cad => cad.toUpperCase()));
```

o un poco menos abreviado:

```
console.log(["Enero", "Febrero", "marzo", "abril", "marzo"].map(
   function (cad: string): string {
      return cad.toUpperCase();
   }));
```

EJERCICIO 4– Ejemplo de uso de filter y paso de parámetros a la función que filtra

Crear una aplicación que haga una consulta sobre un array de animales (objetos JSON) y muestra aquellos que sean de un determinado tipo. Pasado por parámetros. Ej: perros, gatos,…

Un posible array sería:

```
let animales: any = [
   {
      nombre: "Poppy",
      especie: "Burro",
      hobbies: ["Comer", "Rascarse", "Galopar"],
      propietario: {
         nombre: "Adrián",
         ciudad: "Avilés",
         tfno:"601112235"

      },
      añoNacimiento: 2000
   },
```

```
  {
    nombre: "Pepa Pig",
    especie: "Cerdo",
    hobbies: ["Comer", "Rascarse", "Dormir"],
    propietario: {
        nombre: "Andrea",
        ciudad: "Avilés",
        tfno:"671182295"

    },
    añoNacimiento: 2010
},
  {
    nombre: "Luck",
    especie: "Perro",
    hobbies: ["Correr", "Jugar con la pelota"],
    propietario: {
        nombre: "Belarmino",
        ciudad: "Sama",
        tfno:"641145235"

    },
    añoNacimiento: 2005
},
  {
    nombre: "Srek",
    especie: "Burro",
    hobbies: ["Comer", "Rascarse", "Galopar"],
    propietario: {
        nombre: "Beatriz",
        ciudad: "Oviedo",
        tfno:"689512266"

    },
    añoNacimiento: 2001
},
  {
    nombre: "Pirata",
    especie: "Perro",
    hobbies: ["Correr", "Buscar cosas"],
    propietario: {
        nombre: "Adrián",
        ciudad: "Avilés",
```

```
            tfno:"601112235"

        },
        añoNacimiento: 2005
    }
];
```

SOLUCIÓN 1: usando función flecha

```
let animales: any = //el array del enunciado

let tipo = "Perro";
let resultAnimales = animales.filter(
   animal =>animal.especie === tipo);

console.log(resultAnimales);
```

SOLUCIÓN 2: usando una función que devuelve la llamada a otra función

```
let tipo = "Perro";

let filtroFuncion = function (parametro: string) {
   return function (elemento) {
      return elemento.especie === parametro;
   }
};

let resultAnimales = animales.filter(filtroFuncion(tipo));

console.log(resultAnimales);
```

Incluso podría hacer algo como:

```
let resuAnimales2=animales.filter(filtroFuncion(tipo2));
let resuAnimales3=animales.filter(filtroFuncion("Burro"));
console.log(resuAnimales2);
console.log(`Mis burros son: ${JSON.stringify(resuAnimales3)}`);
```

SOLUCIÓN 3: es lo mismo que la SOLUCIÓN 2, pero usando la forma abreviada con funciones flecha. Como hay 2 return, hay 2 =>.Fijaros:

```
let filtroFuncion = parametro => elemento =>elemento.especie === parametro;

let resultAnimales = animales.filter(filtroFuncion(tipo));
console.log(resultAnimales);
```

Lo escrito en este color verde equivale a la función interna, llamada en el return.

EJERCICIO 5– Ejemplo de uso de map, forEach y JSON complejos

Crear una aplicación que a partir de un array de empleados (serán objetos JSON) les suba el sueldo a todos un % que se indicará por parámetro. Y muestre el nombre y salario de todos los empleados con el nuevo salario. Usar funciones.

Un ejemplo de salida será:

```
Los nuevos salarios son 1100,1210,2200,1980,1100,2310          (unknown)
Los empleados con el nuevo salario son:                        (unknown)
▶ Array(6)                                                     (unknown)
O mostrando sólo el nombre y salario:                          (unknown)
Nombre: Abel  y  Nuevo Salario: 1100                           (unknown)
Nombre: Ana  y  Nuevo Salario: 1210                            (unknown)
Nombre: Bartolomé  y  Nuevo Salario: 2200                      (unknown)
Nombre: Bárbara  y  Nuevo Salario: 1980                        (unknown)
Nombre: Carlos  y  Nuevo Salario: 1100                         (unknown)
Nombre: Cristina  y  Nuevo Salario: 2310                       (unknown)
```

SOLUCIÓN 1:

```
let empleados = [
   {
      nombre: "Abel",
      edad: 22,
```

```
    salario: 1000,
    direccion: {
        calle: "Sta Rita",
        numero: 3,
        ciudad: "La Felguera"
    }
},
{
    nombre: "Ana",
    edad: 24,
    salario: 1100,
    direccion: {
        calle: "La Rue",
        numero: 32,
        ciudad: "Oviedo"
    }
},
{
    nombre: "Bartolomé",
    edad: 32,
    salario: 2000,
    direccion: {
        calle: "Uria",
        numero: 4,
        ciudad: "Oviedo"
    }
},
{
    nombre: "Bárbara",
    edad: 42,
    salario: 1800,
    direccion: {
        calle: "La nueva",
        numero: 1,
        ciudad: "Gijón"
    }
},
{
    nombre: "Carlos",
    edad: 34,
    salario: 1000,
    direccion: {
        calle: "El Percebe",
```

```
            numero: 9,
            ciudad: "La Felguera"
        }
    },
    {
        nombre: "Cristina",
        edad: 54,
        salario: 2100,
        direccion: {
            calle: "San Agustín",
            numero: 12,
            ciudad: "Gijón"
        }
    }
];


let subida = 10;  //es 10%

let nuevosSalarios=empleados.map(empleado => empleado.salario +=
empleado.salario * subida / 100);
console.log(`Los nuevos salarios son ${nuevosSalarios}`);
console.log("Los empleados con el nuevo salario son:");
console.log(empleados);
console.log("O mostrando sólo el nombre y salario:");
empleados.forEach(empleado => console.log(`Nombre: ${empleado.nombre}  y
Nuevo Salario: ${empleado.salario}`));
```

SOLUCIÓN 2: Usando una función que recibe un parámetro que será el porcentaje a subir y pasarle al map esa función para que la aplique a cada empleado. ESTA ES MEJOR SOLUCIÓN:

```
let funcionSubirSalario = function (porcentaje:number){
    return function(elemento){
        return elemento.salario += elemento.salario * porcentaje / 100;
    }
}

let nuevosSalarios2=empleados.map(funcionSubirSalario(10));
console.log(`Los nuevos salarios son ${nuevosSalarios2}`);
console.log("Los empleados con el nuevo salario son:");
console.log(empleados);
console.log("O mostrando sólo el nombre y salario:");
```

```
empleados.forEach(empleado => console.log(`Nombre: ${empleado.nombre}  y  Nuev
o Salario: ${empleado.salario}`));
```

SOLUCIÓN 3: es la misma que la SOLUCIÓN 2, pero indicando el tipo que devuelve la función "funcionSubirSalario2" que realmente devuelve una función. Pero, ¿cómo indico que devuelve una función? Respuesta:

- **Se pone ":" para indicar el tipo devuelto y**
- **Para indicar que un tipo es una función hay que concretar qué aspecto tiene esa función. Es decir, qué parámetros lleva y de qué tipo y qué devuelve. En este ejemplo concreto sería:**

(ele : any) => number

```
let funcionSubirSalario2 = function (porcentaje:number):(ele:any)=>number {
    return function(elemento){
        return elemento.salario += elemento.salario * porcentaje / 100;
    }
}

let nuevosSalarios3=empleados.map(funcionSubirSalario2(10));
console.log(`Los nuevos salarios son ${nuevosSalarios3}`);
console.log("Los empleados con el nuevo salario son:");
console.log(empleados);
console.log("O mostrando sólo el nombre y salario:");
empleados.forEach(empleado => console.log(`Nombre: ${empleado.nombre}  y  Nuev
o Salario: ${empleado.salario}`));
```

SOLUCIÓN 4: es la misma que la SOLUCIÓN 2, pero pasando a notación flecha las funciones.

```
let funcionSubirSalario3 = porcentaje => elemento => elemento.salario += elemento.salario *
porcentaje / 100;

let nuevosSalarios4=empleados.map(funcionSubirSalario3(10));
console.log(`Los nuevos salarios son ${nuevosSalarios4}`);
console.log("Los empleados con el nuevo salario son:");
console.log(empleados);
console.log("O mostrando sólo el nombre y salario:");
empleados.forEach(empleado => console.log(`Nombre: ${empleado.nombre}  y  Nuevo Salario: ${e
mpleado.salario.toFixed(2)}`));
```

EJERCICIO 6– Ejemplo de uso de map, desestructuración y resto de parámetros

Crear una aplicación que a partir de un array de empleados (serán objetos JSON) les suba el sueldo a todos un % que se indicará por parámetro y **nos muestre cuál es el salario mayor y a quien pertenece**. Usar funciones.

```
let empleados = [
    {
        nombre: "Abel",
        edad: 22,
        salario: 1000,
        direccion: {
            calle: "Sta Rita",
            numero: 3,
            ciudad: "La Felguera"
        }
    },
    {
        nombre: "Ana",
        edad: 24,
        salario: 1100,
        direccion: {
            calle: "La Rue",
            numero: 32,
            ciudad: "Oviedo"
        }
    },
    {
        nombre: "Bartolomé",
        edad: 32,
        salario: 2000,
        direccion: {
            calle: "Uria",
            numero: 4,
            ciudad: "Oviedo"
        }
    },
    {
        nombre: "Bárbara",
        edad: 42,
        salario: 1800,
        direccion: {
            calle: "La nueva",
```

```
            numero: 1,
            ciudad: "Gijón"
        }
    },
    {
        nombre: "Carlos",
        edad: 34,
        salario: 1000,
        direccion: {
            calle: "El Percebe",
            numero: 9,
            ciudad: "La Felguera"
        }
    },
    {
        nombre: "Cristina",
        edad: 54,
        salario: 2100,
        direccion: {
            calle: "San Agustín",
            numero: 12,
            ciudad: "Gijón"
        }
    }
];


let subida = 10;  //es 10%

//ejemplo de uso de map
let nuevosSalarios = empleados.map(
    empleado => empleado.salario += empleado.salario * subida / 100
);
console.log(nuevosSalarios);

//ejemplo de desestructuración y de resto de parámetros ...
let { nombre, salario }=empleados[nuevosSalarios.indexOf(Math.max(...nuevosSalarios))];
console.log(`El empleado ${nombre} con un salario de ${salario} es el que más
gana`);

//el array de empleados despues de subir el sueldo es:
console.log("El array de empleados después subida sueldo es:");
console.log(empleados);
```

EJERCICIO 7– Ejemplo de uso de map

Modificar el ejercicio anterior para que la subida de salario sea sólo para los empleados que tengan veintipico años. No hay que calcular el mayor sueldo.

SOLUCIÓN 1: usando primero un filter y sobre el resultado del filtro se aplica el map:

```
let empleados = [
   {
      nombre: "Abel",
      edad: 22,
      salario: 1000,
      direccion: {
         calle: "Sta Rita",
         numero: 3,
         ciudad: "La Felguera"
      }
   },
   {
      nombre: "Ana",
      edad: 24,
      salario: 1100,
      direccion: {
         calle: "La Rue",
         numero: 32,
         ciudad: "Oviedo"
      }
   },
   {
      nombre: "Bartolomé",
      edad: 32,
      salario: 2000,
      direccion: {
         calle: "Uria",
         numero: 4,
         ciudad: "Oviedo"
      }
   },
   {
      nombre: "Bárbara",
      edad: 42,
      salario: 1800,
      direccion: {
         calle: "La nueva",
```

```
            numero: 1,
            ciudad: "Gijón"
        }
    },
    {
        nombre: "Carlos",
        edad: 34,
        salario: 1000,
        direccion: {
            calle: "El Percebe",
            numero: 9,
            ciudad: "La Felguera"
        }
    },
    {
        nombre: "Cristina",
        edad: 54,
        salario: 2100,
        direccion: {
            calle: "San Agustín",
            numero: 12,
            ciudad: "Gijón"
        }
    }
];



let subida = 10;  //es 10%

//ejemplo de uso de filter combinado con map

let nuevosSalarios = empleados.filter(
    empleado => empleado.edad >= 20 && empleado.edad < 30)
    .map(empleado => empleado.salario += empleado.salario * subida / 100);

//el array de empleados despues de subir el sueldo es:
console.log("El array de empleados después subida sueldo es:");
console.log(empleados);
```

SOLUCIÓN 2: es equivalente al código anterior, pero sin usar filter, haciendo nosotros un recorrido con for y un if para quedarnos con los que tienen veintipico años. Tiene MÁS TRABAJO, es peor:

```
let subida = 10;  //es 10%

//ejemplo de uso de map
let nuevosSalarios = empleados.map(
   empleado => {
      if (empleado.edad >= 20 && empleado.edad < 30)
         return empleado.salario += empleado.salario * subida / 100
   });

//el array de empleados despues de subir el sueldo es:
console.log("El array de empleados después subida sueldo es:");
console.log(empleados);
```

SOLUCIÓN 3: es modificar la SOLUCIÓN 1 para crear una función filtro por edad que reciba la edad mínima y máxima a filtrar y pasaríamos esa función al filtro. Sería como en otros ejemplos que hicimos una función que devuelve otra función. ES LA MEJOR SOLUCIÓN DE LAS TRES:

```
let subida3 = 10;  //es 10%

// Función filtrar por edad según una edad mínima y máxima
let filtroPorEdad=function (edadMin,edadMax){
    return function(ele){
        return ele.edad >= edadMin && ele.edad < edadMax;
    }
}

/* Esa misma función filtroPorEdad() en notación flecha sería:

   let filtroPorEdad = (edadMin,edadMax) => ele => ele.edad >= edadMin && ele.
edad < edadMax;
*/

let nuevosSalarios3 = empleados.filter(filtroPorEdad)
    .map(empleado => empleado.salario += empleado.salario * subida / 100);

//el nuevo array de salarios es
console.log("El nuevo array de salarios es:");
console.log(nuevosSalarios3);
//el array de empleados despues de subir el sueldo es:
console.log("El array de empleados después subida sueldo es:");
console.log(empleados);
```

EJERCICIOS DE ANGULAR para practicar HTML y TypeScript (sólo usamos un componente Angular)

EJERCICIO 8 – Acceder desde TypeScript a propiedades y elementos del HTML, como background, color, head,…

Hacer que se muestren 5 cuadrados de diferentes colores y al pasar por encima de ellos cambie el color del fondo de la página web con el color de ellos. PISTA: mirar los eventos de ratón.

Además habrá una etiqueta debajo de los cuadros. Esta etiqueta tendrá un fondo de color que podremos elegir con selector de color situado a su izquierda (*es un input de tipo color en HTML5).*

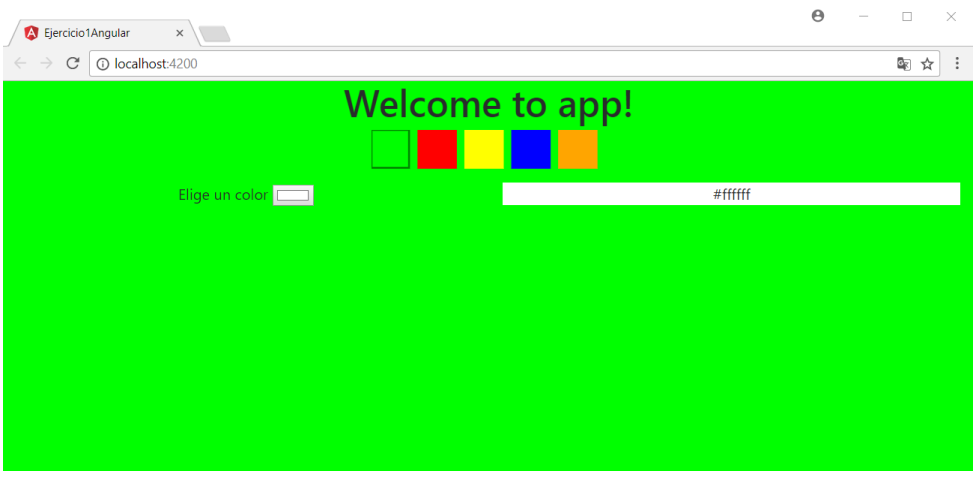
Hay varias formas de dibujar el rectángulo:

1) Con divs
2) Con canvas de HTML
3) Con SVG de HTML. He usado esta. El código para dibujar un rectángulo en HTML5 es:

```
<svg width=200 height=200>
    <rect x=10 y=30 width=180 height=140 stroke="green" stroke-width=2
fill="lime"/>
    </svg>
```

Para acceder al color del fondo de la página se hace con javascript/typescript desde *.ts y la línea que accede a esa propiedad es:

```
document.body.style.backgroundColor=….;
```



VIDEO EXPLICACIÓN: ¿qué se pide y cómo hacerlo? Pinchar aquí

SOLUCIÓN: en un proyecto angular, fuera de este Word.
VIDEO EXPLICACIÓN de la solución: pincha aquí

EJERCICIO 9 – eventos javascript/typescript

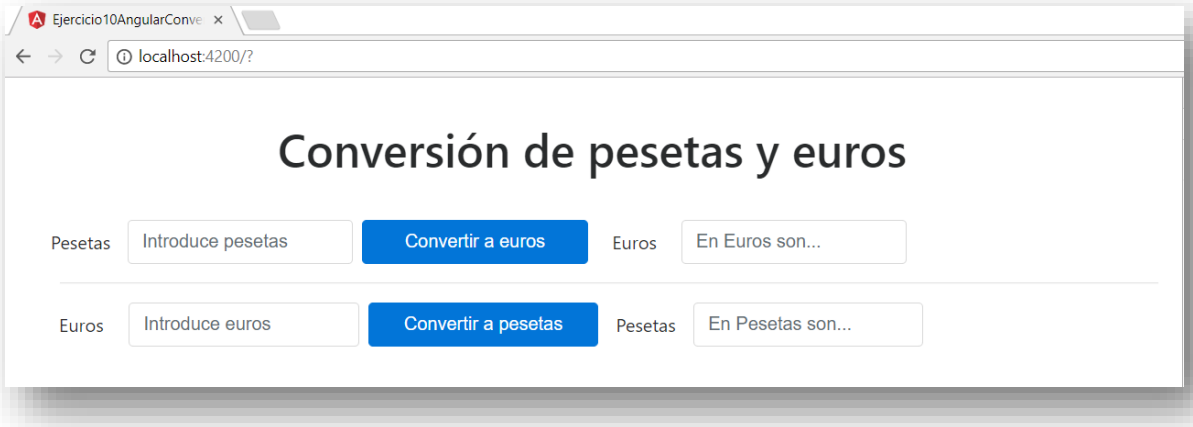
Version 2 del EJERCICIO 8: al quitar el ratón de encima, que vuelva a su color original el fondo de la página. PISTA: mirar los eventos que hay de ratón:

VIDEO EXPLICACIÓN: ¿qué se pide y cómo hacerlo? Pinchar aquí

SOLUCIÓN: en un proyecto angular, fuera de este Word.

EJERCICIO 10 – user input y data binding y uso de NaN

Realiza una página que muestre un formulario para la conversión de Euros a Pesetas o viceversa. Los campos del formulario han de poder ser limpiados. Se debe comprobar que si se introduce algo no numérico muestre una alerta indicándolo.



SOLUCIÓN: en un proyecto angular, fuera de este Word.
VIDEO EXPLICACIÓN DE LA SOLUCIÓN: pincha aquí