



# Tema 4: Ejercicios Resueltos

TYPESCRIPT

STUDIUM

[www.grupostudium.com](http://www.grupostudium.com)  
[informacion@grupostudium.com](mailto:informacion@grupostudium.com)  
954 539 952



## 1 Introducción

Ahora toca practicar con todo lo que se ha visto en el Temario. Desarrollaremos nuestras aplicaciones, en proyectos independientes, utilizando el lenguaje de programación **TypeScript**.

## 2 Ejercicio 1

Crea una clase Empleado con los atributos privados nombre, apellidos y edad así como todos los accesores y constructores que necesites.

Instancia un objeto de tipo Empleado y muestra en consola, los valores dados a los atributos de dicho objeto.

Debes implementar el código de tu aplicación en un único fichero de extensión **.ts** llamado **empleado.ts**.

Adjunta capturas de pantalla del código desarrollado y de la correcta ejecución de tu aplicación.

## Solución

A continuación mostramos el código de la solución propuesta al ejercicio 1:

```
class Empleado {  
    //Atributos  
    private _nombre: string;  
    private _apellidos: string;  
    private _edad: number;  
  
    //constructor por parámetros  
    constructor(nombre: string, apellidos: string, edad: number) {  
        this._nombre = nombre;  
        this._apellidos = apellidos;  
        this._edad = edad;  
    }  
  
    //Getters  
    get nombre() {  
        return this._nombre;  
    }  
  
    get apellidos() {  
        return this._apellidos;  
    }  
}
```



```
get edad() {
    return this._edad;
}

//Setters
set nombre(nombre: string) {
    this._nombre = nombre;
}

set apellidos(apellidos: string) {
    this._apellidos = apellidos;
}

set edad(edad: number) {
    this._edad = edad;
}
}

let empleado1 = new Empleado("María José", "Martínez Navas", 22);
console.log("El empleado " + empleado1.nombre + " " + empleado1.apellidos + "
tiene " + empleado1.edad + " años de edad.");
```

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the project structure with files 'empleado.js' and 'empleado.ts'. The main editor displays the TypeScript code for the 'Empleado' class, including getters, setters, and the instantiation of 'empleado1'. The bottom panel shows the 'TERMINAL' tab with the following commands and output:

```
PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio1ProyectoTS> tsc empleado.ts
PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio1ProyectoTS> node empleado.js
El empleado María José Martínez Navas tiene 22 años de edad.
```



### 3. Ejercicio 2

Crea una clase `FormaGeometrica` con los atributos privados `color`, `grosorBorde` y `tipo`, así como todos los accesores y constructores que necesites. Implementa una función llamada `mostrar()`, que te permita mostrar por consola los valores de los atributos de un objeto de tipo `FormaGeometrica`.

Instancia un objeto de tipo `FormaGeometrica` y muestra en consola, los valores dados a los atributos de dicho objeto.

Debes implementar el código de la clase `FormaGeometrica` en un único fichero de extensión `.ts` llamado `formaGeometrica.ts`.

Adjunta capturas de pantalla del código desarrollado y de la correcta ejecución de tu aplicación.

### Solución

```
class FormaGeometrica {  
  
    //Atributos  
    private _color: string;  
    private _grosorBorde: Number;  
    private _tipo: string;  
  
    //Constructor por parámetros  
    constructor(color: string, grosorBorde: Number, tipo: string) {  
        this._color = color;  
        this._grosorBorde = grosorBorde;  
        this._tipo = tipo;  
    }  
  
    //getters  
    get color() {  
        return this._color;  
    }  
    get grosorBorde() {  
        return this._grosorBorde;  
    }  
    get tipo() {  
        return this._tipo;  
    }  
}
```



```
//setters
set color(color: string) {
    this._color = color;
}
set grosorBorde(grosorBorde: Number) {
    this._grosorBorde = grosorBorde;
}
set tipo(tipo: string) {
    this._tipo = tipo;
}
/*Con esta función mostramos por consola los valores dados a los atributos */
mostrar() {
    console.log(` Soy una forma de tipo ${this._tipo}, color ${this._color} y grosor de
borde ${this._grosorBorde}`);
}
}

let formaGeometrica = new FormaGeometrica("Verde", 0.5, "Genérico");
formaGeometrica.mostrar();
```

The screenshot shows the Visual Studio Code editor with the file `formaGeometrica.ts` open. The code defines a class `FormaGeometrica` with private attributes `_color`, `_grosorBorde`, and `_tipo`. It includes a constructor that initializes these attributes and a `color` getter. The terminal at the bottom shows the command `tsc formaGeometrica.ts` being executed, followed by `node formaGeometrica.js`, which outputs: `Soy una forma de tipo Genérico, color Verde y grosor de borde 0.5`.

## 4. Ejercicio 3

Continuando con el ejercicio anterior, crea las clases Cuadrado, Circulo y Triangulo, que **heredan** de la clase FormaGeometrica, y que tendrán los siguientes atributos propios:

- Clase Cuadrado: lado.



- Clase Circulo: diametro y radio.
- Clase Triangulo: base y altura.

Crea, en cada una de estas clases, todos los accesores, constructores y métodos que necesites.

Instancia un objeto de cada tipo Cuadrado, Circulo y Triangulo y muestra en consola los valores dados a los atributos de dichos objetos.

Adjunta capturas de pantalla del código desarrollado y de la correcta ejecución de tu aplicación.

Debes implementar el código de la aplicación en un único fichero de extensión **.ts** llamado **herencia.ts**.

### Solución

```
class FormaGeometrica {  
  
    //Atributos  
    private _color: string;  
    private _grosorBorde: Number;  
    private _tipo: string;  
  
    //Constructor por parámetros  
    constructor(color: string, grosorBorde: Number, tipo: string) {  
        this._color = color;  
        this._grosorBorde = grosorBorde;  
        this._tipo = tipo;  
    }  
  
    //getters  
    get color() {  
        return this._color;  
    }  
    get grosorBorde() {  
        return this._grosorBorde;  
    }  
    get tipo() {  
        return this._tipo;  
    }  
  
    //setters
```



```
set color(color: string) {
    this._color = color;
}

set grosorBorde(grosorBorde: Number) {
    this._grosorBorde = grosorBorde;
}

set tipo(tipo: string) {
    this._tipo = tipo;
}

mostrar() {
    console.log(` Soy una forma de tipo ${this._tipo}, color ${this._color} y grosor de
borde ${this._grosorBorde}`);
}

class Cuadrado extends FormaGeometrica {
    private _lado: number;

    constructor(color: string, grosorBorde: number, tipo: string, lado: number) {
        super(color, grosorBorde, tipo);
        this._lado = lado;
    }

    //get
    get lado() {
        return this._lado;
    }

    //set
    set lado(lado: number) {
        this._lado = lado;
    }

    mostrarCuadrado() {
        this.mostrar(); //Muestro el tipo, color y grosor del borde
        console.log(` La medida del lado del ${this._tipo} es: ${this._lado} cm `);
    }
}
```



```
class Circulo extends FormaGeometrica {
    private _radio: number;

    constructor(color: string, grosorBorde: number, tipo: string, radio: number) {
        super(color, grosorBorde, tipo);
        this._radio = radio;
    }

    //getter
    get radio() {
        return this._radio;
    }

    //setter
    set radio(radio: number) {
        this._radio = radio;
    }

    mostrarCirculo() {
        this.mostrar(); //Muestro el tipo, color y grosor del borde
        console.log(`La medida del radio del ${this.tipo} es: ${this._radio} cm`);
    }
}

class Triangulo extends FormaGeometrica {
    private _base: number;
    private _altura: number;

    constructor(color: string, grosorBorde: number, tipo: string, base: number, altura:
number) {
        super(color, grosorBorde, tipo);
        this._base = base;
        this._altura = altura;
    }

    get base() {
        return this._base;
    }

    get altura() {
        return this._altura;
    }
}
```





```
}

set base(base: number) {
  this._base = base;
}

set altura(altura: number) {
  this._altura = altura;
}

mostrarTriangulo() {
  this.mostrar(); //Muestro el tipo, color y grosor del borde
  console.log(`La medida de la base y de la altura del ${this.tipo} es: ${this._base}
cm y ${
  this._altura} cm`);
}
}

let cuadrado = new Cuadrado("Rojo", 0.3, "Cuadrado", 12);
cuadrado.mostrarCuadrado();

let circulo = new Circulo("Amarillo", 0.1, "Círculo", 10);
circulo.mostrarCirculo();

let triangulo = new Triangulo("Naranja", 0.5, "Triángulo", 3, 5.5);
triangulo.mostrarTriangulo();
```

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the project structure with files herencia.js and herencia.ts. The main editor displays the herencia.ts file, which defines a base class `FormaGeometrica` with private attributes `_color`, `_grosorBorde`, and `_tipo`, a constructor, and a `mostrar` method. It also defines three subclasses: `Cuadrado`, `Circulo`, and `Triangulo`, each with its own constructor and `mostrar` method. The terminal at the bottom shows the command `tsc herencia.ts` being executed, followed by `node herencia.js`, which produces the output of the `mostrar` methods for each shape.

```
ts herencia.ts X
TS herencia.ts > FormaGeometrica > (get) grosorBorde
1 class FormaGeometrica {
2
3   //Atributos
4   private _color: string;
5   private _grosorBorde: Number;
6   private _tipo: string;
7
8   //Constructor por parámetros
9   constructor(color: string, grosorBorde: Number, tipo: string) {
10    this._color = color;
11    this._grosorBorde = grosorBorde;
12    this._tipo = tipo;
13  }
14
15  //getters
16
17  mostrar() {
18    console.log(`La medida del lado del Cuadrado es: 12 cm`);
19    console.log(`La medida del radio del Circulo es: 10 cm`);
20    console.log(`La medida de la base y de la altura del Triángulo es: 3 cm y 5.5 cm`);
21  }
22
23  mostrarCuadrado() {
24    console.log(`Soy una forma de tipo Cuadrado, color Rojo y grosor de borde 0.3`);
25  }
26
27  mostrarCirculo() {
28    console.log(`Soy una forma de tipo Circulo, color Amarillo y grosor de borde 0.1`);
29  }
30
31  mostrarTriangulo() {
32    console.log(`Soy una forma de tipo Triángulo, color Naranja y grosor de borde 0.5`);
33  }
34
35  static readonly PI = 3.14159;
36}

ctoS> tsc herencia.ts
PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio3Proye
ctoS> node herencia.js
Soy una forma de tipo Cuadrado, color Rojo y grosor de borde 0.3
La medida del lado del Cuadrado es: 12 cm
Soy una forma de tipo Circulo, color Amarillo y grosor de borde 0.1
La medida del radio del Circulo es: 10 cm
Soy una forma de tipo Triángulo, color Naranja y grosor de borde 0.5
La medida de la base y de la altura del Triángulo es: 3 cm y 5.5 cm
PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio3Proye
```



### 5. Ejercicio 4

Crea un módulo llamado `miGeometria` en el que incluyas todas las clases implementadas en los ejercicios anteriores de forma que se puedan utilizar en cualquier parte de tu aplicación.

Instancia un objeto de cada tipo `Cuadrado`, `Circulo` y `Triangulo` y muestra en consola los valores dados a los atributos de dichos objetos.

Adjunta capturas de pantalla del código desarrollado y de la correcta ejecución de tu aplicación.

Debes implementar el código de la aplicación en un único fichero de extensión `.ts` llamado `main.ts`.

### Solución

```
module miGeometria {  
  export class FormaGeometrica {  
  
    //Atributos  
    private _color: string;  
    private _grosorBorde: Number;  
    private _tipo: string;  
  
    //Constructor por parámetros  
    constructor(color: string, grosorBorde: Number, tipo: string) {  
      this._color = color;  
      this._grosorBorde = grosorBorde;  
      this._tipo = tipo;  
    }  
  
    //getters  
    get color() {  
      return this._color;  
    }  
    get grosorBorde() {  
      return this._grosorBorde;  
    }  
    get tipo() {  
      return this._tipo;  
    }  
    //setters
```



```
set color(color: string) {
  this._color = color;
}
set grosorBorde(grosorBorde: Number) {
  this._grosorBorde = grosorBorde;
}
set tipo(tipo: string) {
  this._tipo = tipo;
}

mostrar() {
  console.log(` Soy una forma de tipo ${this._tipo}, color ${this._color} y grosor
de borde ${this._grosorBorde} `);
}
}

export class Cuadrado extends FormaGeometrica {
  private _lado: number;

  constructor(color: string, grosorBorde: number, tipo: string, lado: number) {
    super(color, grosorBorde, tipo);
    this._lado = lado;
  }

  //get
  get lado() {
    return this._lado;
  }
  //set
  set lado(lado: number) {
    this._lado = lado;
  }

  mostrarCuadrado() {
    this.mostrar(); //Muestro el tipo, color y grosor del borde
    console.log(` La medida del lado del ${this._tipo} es: ${this._lado} cm `);
  }
}

export class Circulo extends FormaGeometrica {
  private _radio: number;
```



```
constructor(color: string, grosorBorde: number, tipo: string, radio: number) {
    super(color, grosorBorde, tipo);
    this._radio = radio;
}

//getter
get radio() {
    return this._radio;
}

//setter
set radio(radio: number) {
    this._radio = radio;
}

mostrarCirculo() {
    this.mostrar(); //Muestro el tipo, color y grosor del borde
    console.log(`La medida del radio del ${this.tipo} es: ${this._radio} cm`);
}
}

export class Triangulo extends FormaGeometrica {
    private _base: number;
    private _altura: number;

    constructor(color: string, grosorBorde: number, tipo: string, base: number,
altura: number) {
        super(color, grosorBorde, tipo);
        this._base = base;
        this._altura = altura;
    }

    get base() {
        return this._base;
    }
    get altura() {
        return this._altura;
    }
}
```



```
set base(base: number) {
    this._base = base;
}

set altura(altura: number) {
    this._altura = altura;
}

mostrarTriangulo() {
    this.mostrar(); //Muestro el tipo, color y grosor del borde
    console.log(`La medida de la base y de la altura del ${this.tipo} es:
    ${this._base} cm y ${this._altura} cm`);
}
}
}
```

let cuadrado = new miGeometria.Cuadrado("Rojo", 0.3, "Cuadrado", 12);  
cuadrado.mostrarCuadrado();

let circulo = new miGeometria.Circulo("Amarillo", 0.1, "Círculo", 10);  
circulo.mostrarCirculo();

let triangulo = new miGeometria.Triangulo("Naranja", 0.5, "Triángulo", 3, 5.5);  
triangulo.mostrarTriangulo();

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the file structure with 'main.ts' selected. The main editor displays the TypeScript code for a class 'miGeometria' with methods for setting base, height, and displaying shapes. The terminal at the bottom shows the command 'tsc main.ts' and 'node main.js' being executed, followed by the output of the program, which displays the details of a square, a circle, and a triangle.

```
main.ts - Ejercicio4ProyectoTS - Visual Studio Code
```

EXPLORER

- OPEN EDITORS
  - TS main.ts
- EJERCICIO4PROYECTOTS
  - main.js
  - main.ts

```
TS main.ts > {} miGeometria > FormaGeometrica > (get) color
1 module miGeometria {
2   export class FormaGeometrica {
3
4     //Atributos
5     private _color: string;
6     private _grosorBorde: Number;
7     private _tipo: string;
8
9     //Constructor por parámetros
10    constructor(color: string, grosorBorde: Number, tipo: string) {
11      this._color = color;
12      this._grosorBorde = grosorBorde;
13      this._tipo = tipo;
14    }
15  }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

powershell + - - - ^ x

```
oyectoTS> tsc main.ts
PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio4Pr
oyectoTS> node main.js
Soy una forma de tipo Cuadrado, color Rojo y grosor de borde 0.3
La medida del lado del Cuadrado es: 12 cm
Soy una forma de tipo Círculo, color Amarillo y grosor de borde 0.1
La medida del radio del Círculo es: 10 cm
Soy una forma de tipo Triángulo, color Naranja y grosor de borde 0.5
La medida de la base y de la altura del Triángulo es: 3 cm y 5.5 cm
```

PS G:\Curso 2023-2024 (Preparación)\DAW\DWEC\Cursos Angular\Libro JavaScript y Angular\Capítulo 7\Ejercicio4Pr

Ln 18, Col 32 Spaces: 4 UTF-8 CRLF TypeScript

29/04/2023