CLASES

CLASES

enlace a la documentación

Fundamentos

- Declaración con class: variables y métodos

```
class  Point {
    x, y: number;
    move(x: number, y: number) {this.x = x; this.y = y; }
}
```

- Acceso a variables/métodos de instancia con **this.XXX**

- Constructor (sólo uno!!) con constructor

```
class Point {
    x, y: number;
    constructor(x: number, y: number) { this.x = x; this.y = y; }
    move(x: number, y: number) { this.x = x; this.y = y; }
}
```

- Instanciación con new

```
let p: Point = new Point(0,0);
```

CLASES

enlace a la documentación

Modificadores de acceso

- Public (por defecto), private, protected (*acceden las clases heredadas también*)

```
class Point {
    private x, y: number;
    constructor(x: number, y: number) { this.x = x; this.y = y; }
    public move(x: number, y: number) { this.x = x; this.y = y; }
}
```

- Readonly
  - Hace que las propiedades a las que acompaña sean **sólo de lectura**.
  - Las propiedades deben ser inicializadas al declararlas o en el constructor.

Ejemplo:

```
class Octopus {
    readonly name: string;
    readonly numberOfLegs: number = 8;
    constructor (theName: string) { this.name = theName; }
}

let dad = new Octopus("Man with the 8 strong legs");
dad.name = "Man with the 3-piece suit"; // error! name is readonly.
```

CLASES

enlace a la documentación

- Parámetro en **constructor + propiedad 2x1**

```
class Point {
    constructor(private x: number, private y: number) { }
    public move(x: number, y: number) { this.x = x; this.y = y; }
}
```

ES EQUIVALENTE A

```
class Point {
    private x, y: number;
    constructor(x: number, y: number) { this.x = x; this.y = y; }
    public move(x: number, y: number) { this.x = x; this.y = y; }
}
```

CLASES

enlace a la documentación

Getters y Setters:

- Se pueden crear getters y setters pero, **no se llaman como una función normal**:
  - El get será llamado cuando preguntemos por el valor de:

    objeto.propiedad

  - El set será llamado cuando asignemos un valor a:

    objeto.propiedad

    Se suele usar para filtrar valores antes de asignarlos a una propiedad: *ej: comprobar que no sea null, que sea > que un valor,…*

- **Ejemplo de get**:

```
get nombre(): string {
        return this.nombre;
}
```

Ejemplo de set:

```
set nombre(n: string) {
  if (n === "Fuego")
     this._nombre = "IGNIS";
  else if (n === "")
     this._nombre = "SIN NOMBRE";
}
```

CLASES

enlace a la documentación

- **Getters y Setters:**
  - **Ejemplo de uso**:

```
class Animal {

constructor(private _nombre: string, public especie: string, public edad: number) { }

get nombre(): string {
     return this.nombre;
}

set nombre(n: string) {
    if (n === "Fuego")
        this._nombre = "IGNIS";
    else if (n === "")
        this._nombre = "SIN NOMBRE";
    }
}

let miAnimal: Animal = new Animal("Estrella", "yegua", 8);
miAnimal.nombre = "Fuego";       //estoy usando el set
console.log(miAnimal.nombre);    //estoy usando el get
```

CLASES

enlace a la documentación

Herencia

- Con extends: acceso a padre con super

```
class Shape {
    constructor(protected color: number) { }
}

class Circle extends Shape {
    constructor(color: number, private r: number, private center: Point) {
            super(color);
    }
}
let shp: Shape = new Shape(222);
let cir: Circle = new Circle(222, 10, new Point(0,0));
let shp2: Shape = new Circle(222, 10, new Point(0,0)); // polimorfismo!!
```

CLASES

enlace a la documentación

Más ...

- Clases y variables/métodos abstractos con abstract
- Variables/métodos de clase con static

```
abstract class Shape {
        constructor(protected color: number) { Shape.numOfShapes++; }
        abstract area(): number;
        static numOfShapes: number = 0;
}

class Circle extends Shape {
        constructor(color: number, private r: number, private center: Point) {
                super(color);
        }
        area(): number { return Math.PI*r*r; }
}
let shp: Shape = new Shape(222); // error!!
let cir: Circle = new Circle(222, 10, new Point(0,0));
console.log('area:' + cir.area() + ';shapes:' + Shape.numOfShapes);
```