



TypeScript (parte5 Clases)

LABORATORIO DE COMPUTACIÓN III

UTNFRA – TÉCNICO UNIVERSITARIO EN PROGRAMACIÓN --

Classes

```
1
2  class Avenger {
3      nombre:string;
4      equipo:string;
5      nombreReal:string;
6      puedePelear:boolean;
7      peleasGanadas:number;
8  }
9
10 let antman:Avenger= new Avenger();
11
12 console.log(antman);
13
```

Constructor

```
1  class Avenger1{
2      nombre:string;
3      equipo:string;
4      nombreReal:string;
5
6      puedePelear:boolean;
7      peleasGanadas:number;
8
9      //Declaracion del constructor
10     constructor(nombre:string, equipo:string, nombreReal:string){
11
12         this.nombre = nombre;
13         this.nombreReal = nombreReal;
14         this.equipo = equipo;
15     }
16 }
17
18 //declaro un objeto
19 let antman1:Avenger1 = new Avenger1("Antman", "Cap", "Scott Lang");
20
```



Propiedades: públicas, protegidas y privadas

```
5  class Avenger2{
6
7      public nombre:string;
8      protected equipo:string;
9      private nombreReal:string;
10
11     private puedePelear:boolean;
12     private peleasGanadas:number;
13
14
15     constructor(nombre:string, equipo:string, nombreReal:string){
16
17         this.nombre = nombre;
18         this.nombreReal = nombreReal;
19         this.equipo = equipo;
20     }
21 }
22
23 antman2.nombre = "Nick Fury";
24 antman2.equipo = "Ironman";
```

Por defecto todas las propiedades son públicas

Métodos: públicos, protegidos y privados

```
1 class Avengerr {
2     public nombre:string;
3     protected equipo:string;
4     private nombreReal:string;
5
6     private puedePelear:boolean;
7     private peleasGanadas:number;
8
9
10    constructor(nombre:string, equipo:string, nombreReal:string){
11
12        this.nombre = nombre;
13        this.nombreReal = nombreReal;
14        this.equipo = equipo;
15    }
16
17    public mostrar():string{
18        return `${this.nombre} ${this.nombreReal}, ${this.equipo}`;
19    }
20 }
21
22
23 let antmann:Avengerr = new Avengerr("Antman", "Cap", "Scott Lang");
24
25 antmann.
```

 mostrar (method) Avengerr.mostrar(): string
 nombre

Por defecto todos los métodos son públicos

Herencia: super y definición de propiedades

```
1  class Avengerr1{
2
3      constructor( public nombre:string, private nombreReal:string){
4
5      }
6  }
7
8  class Xmen extends Avengerr1{
9
10 }
11
12 let ciclope:Xmen = new Xmen("Ciclope", "Scott");
13
14 console.log(ciclope);
```

Las clases heredadas si no tienen un constructor definido explícitamente, toman el constructor de la clase de la cual heredan

Herencia

```
20 class Avengers1{
21
22     constructor( public nombre:string, private nombreReal:string){
23
24     }
25 }
26
27 class Xmen extends Avengers1{
28
29     constructor( a:string, b:string){
30         // llamada al constructor de la clase padre
31         super(a, b);
32     }
33 }
34
35 let ciclope:Xmen = new Xmen("Ciclope", "Scott");
36
37 console.log(ciclope);
38
```

Si definimos un constructor en una clase hija siempre debe haber una llamada al constructor del padre mediante el método super

Get y Set

```
1  class Avengerrr {
2
3      private _nombre: string;
4
5      constructor(nombre?: string) {
6          this._nombre = nombre;
7      }
8
9      get nombre(): string {
10
11          if (this._nombre) {
12              return this._nombre;
13          } else {
14              return "No tiene ningun nombre el Avenger";
15          }
16      }
17
18      set nombre(nombre: string) {
19          this._nombre = nombre;
20      }
21  }
22
23  let ciclope1: Avengerrr = new Avengerrr("Ciclope");
24
```


Métodos y propiedades estáticos

```
1  class Xmenn{
2      static nombre:string = "Wolverine";
3
4      constructor(){
5
6      }
7
8      static crearXmen(){
9          return new Xmenn();
10     }
11
12 }
13
14 console.log(Xmenn.nombre)
15
16 let deadpool:Xmenn = Xmenn.crearXmen();
17
18
```

Classes Abstractas

```
3  
4  abstract class Mutante{  
5  
6      constructor(public nombre:string, public nombreReal:string){  
7  
8      }  
9  }  
10 class Xmenx extends Mutante{  
11  
12 }  
13  
14 let wolverine = new Xmenx("Wolverine", "Logan");|
```

Constructor privado

```
1 // Creamos una clase de la que solo puede haber una en nuestro programa
2
3 class Apocalipsis{
4     static instancia:Apocalipsis;
5
6     private constructor(public nombre:string){
7
8     }
9
10    static llamarApocalipsis():Apocalipsis{
11        if(Apocalipsis.instancia){
12            Apocalipsis.instancia = new Apocalipsis("Soy Apocalipsis!!!");
13        }
14
15        return Apocalipsis.instancia;
16    }
17 }
18
19
20
21 //let apocalipsisFalso = new Apocalipsis("Soy Apocalipsis!!!");
22 let real = Apocalipsis.llamarApocalipsis();
23
24 console.log(real);|
```