Proyecto clases

By Ávaro Maseri Apraiz

La expresión class es una forma de definir una clase. Similar a las funciones, las expresiones de clase pueden ser nombradas o no. Si se nombran, el nombre de la clase es local sólo en el cuerpo de la clase. Las clases en JavaScript utilizan herencia basada en prototipos

Mi proyecto se basa en un pequeño constructor de skates y comenzará con una superclase llamada marca

```
export class marcas {
protected _marca:string

    constructor(
        _marca:string
    ){
        this._marca=_marca

    }
```

```
get marca () {
    return this._marca
}
set marca (_marca:string){
    this._marca=_marca
}
get iva_marca () {
    let iva:number
    if (this._marca=='Santa cruz') {
        iva=0.4
    }
    if (this._marca=='Element') {
        iva=0.2
    }
    if (this._marca=='Creature') {
        iva=0.3
    }
    if (this._marca=='family') {
        iva=0.1
    }
    else {
        iva=0.5
    }
    return iva
}
}
```

De la cual depende la clase Skate

```
import {marcas} from '../superclass/marcas';

export class Skate extends marcas  {
protected _ancho: number
protected _ejes: number
protected _ruedas: number
protected _rodamientos: string
protected _tornilleria: string
constructor(
     (parameter) _ancho: number
    _ancho:number,
    _ejes:number,
    _ruedas:number,
    _rodamientos:string,
    _tornilleria:string,
)
{
    super(_marca)
    this._ancho=_ancho
    this._ejes=_ejes
    this._ruedas=_ruedas
    this._rodamientos=_rodamientos
    this._tornilleria=_tornilleria
}
```

```
set marca (_marca:string){
this._marca=_marca
}
get ancho () {
    return this._ancho
}
set ancho (_ancho:number) {
    this._ancho=_ancho
}
get ejes () {
    return this._ejes
}
set ejes (_ejes:number) {
    this._ancho=_ejes
}
get ruedas (){
    return this._ruedas
}
set ruedas (_ruedas:number) {
    this._ejes=_ruedas
}
get rodamientos () {
    return this._rodamientos
}
set rodamientos (_rodamientos:string) {
    this._rodamientos=_rodamientos
}
get tornilleria () {
    return this._tornilleria
}
set tornilleria (_tornilleria:string) {
    this._tornilleria=_tornilleria
}
```

```
precio_ancho():number {
    let precio1:number = 0

    if (this.ancho <= 8){
        precio1 = precio1 + 40
    }
    else if (this.ancho > 8){
        precio1 = precio1 + 5
    }
    return precio1
}
precio_ejes():number{
    let precio2=0
    // 130 para tablas <7.5, 140 entre 7.5-8.5, 150 >8.5
    if (this._ejes == 130) {
        precio2 = precio2 + 43
    }
    else if (this._ejes == 140) {
        precio2 = precio2 + 56
    }
    else if (this._ejes == 150) {
        precio2 = precio2 + 77
    }
    return precio2
}
```

```
precio_torilleria():number{
    let precio4  = 0
    if (this._tornilleria =='negro') {
        precio4= precio4 + 5
    }
    else if (this._tornilleria =='blanco') {
        precio4= precio4 + 7
    }
return precio4
}
precio_rodamientos():number {
    let precio5=0
    if (this.rodamientos =='metal') {
        precio5= precio5 + 5
    }
    else if (this.rodamientos =='cerámica') {
        precio5= precio5 + 5
    }
    return precio5
}

precio_total():number {
    let precio = 0
    precio = this.precio_rodamientos() + this.precio_ancho()
    + this.precio_ejes() + this.precio_torilleria()
    + this.precio_ruedas()
    precio = precio + (precio * this.iva_marca)
return precio
}
}
```

De la cual depende Long

```
import { Skate } from "./skate";
export class Long extends Skate {
protected _largo:number
protected _marca:string
protected _ancho:number
protected _ejes:number
protected _ruedas:number
protected _rodamientos:string
protected _tornilleria:string

constructor(
    _largo:number,
    _marca:string,
    _ancho:number,
    _ejes:number,
    _ruedas:number,
    _rodamientos:string,
    _tornilleria:string
    )
    {
        super (_marca,_ancho,_ejes,_ruedas,
            _rodamientos,_tornilleria)
        this._largo=_largo
        this._marca=_marca
        this._ancho=_ancho
        this._ejes=_ejes
        this._ruedas=_ruedas
        this._rodamientos=_rodamientos
        this._tornilleria=_tornilleria
```

Con el método get permitimos llamar a ese valor mientras que con el set podemos alterarlo. En este ejemplo no necesitamos definir los métodos ya creados en Skate debido a que son heredados

También admite la implementación de funciones con mismo nombre debido al polimorfismo por ello hay dos funciones iguales en ambas clases llamadas precio_total

```
    get largo (){
        return this._largo
    }
    set largo (_largo:number){
        this._largo==_largo
    }

    precio_total():number{
        let precio:number
        precio = super.precio_total();
            if (this._largo>110){
                precio += 20
        }
            else if (this.largo<=110){
                precio +=10
            }
    return precio
    }
}
```

En ambas esta dos últimas clases defino las partes que definen a mi objeto por ejemplo en el caso de Skate serán marca, ancho, ejes, ruedas, rodamientos y tornillería.
Mientras Long tendrá las mismas mas el añadido de largo.
Es importante mencionar que la marca está heredada de la clase marcas donde dependiendo de la misma se aplicará un porcentaje de IVA

```
import {leerTeclado} from '../util/entradaTeclado'

export const menu = async () => {
    let numero : number
    console.log('Pofavor elija una de las siguientes opciones para continuar \n 1--
    console.log(' 5-- Actualizar skate \n 6-- Actualizar long \n 29 --exit')
    numero = parseInt(await leerTeclado(''))
    return numero
}
```

Un pequeñito menú donde se encuentran las distintitas posibilidades a tomar en relación con las clases

y por último un Index donde se encuentra lo siguiente

```
import { leerTeclado } from "./util/entradaTeclado";
import { Skate } from "./tipos/skate";
import { Long } from "./tipos/long";
import { menu } from "./util/menusito";
const new_skate = async ()=> {
    let skate: Skate

    const marca = await (leerTeclado('introduzca la marca deseada  \n Santa cruz \n Element \n Creature \n family \n'))
    const ancho = parseInt(await leerTeclado('Elija el ancho de la tabla,\n este depende del tamaño de su pie asi como: \n
    const ejes = parseInt(await leerTeclado('Elija los ejes:\n Ancho tabla -- Numero de eje \n < 7.5 -- 130 \n 7.5< x <8.5
    const ruedas = parseInt(await leerTeclado('Elija las ruedas: \n para patinar street -- <56 \n para patinar bowl 56>'))
    const rodamientos = await leerTeclado('Elija el tipo: \n rodamientos de metal \n  rodamientos de cerámica')
    const tornilleria = await leerTeclado('Elija const rodamientos: string aca \n tornilleria negra')
    skate = new Skate(marca,ancho,ejes, ruedas, rodamientos, tornilleria)
```

función new_skate la cual crea el nuevo objeto Skate

```
const show_skate = (show: Skate |undefined) => {
    if (show != undefined){
        console.log(`La marca es ${show.marca}`)
        console.log(`Los ejes son de tipo: ${show.ejes}`)
        console.log(`La marca son de grosor: ${show.ruedas}mm`)
        console.log(`La rodamientos son de: ${show.rodamientos}`)
        console.log(`La marca es de color: ${show.tornilleria}`)
        console.log(`El precio total es: ${show.precio_total()}`)
    }
    else {
        console.log('no has creado el skate')
    }
}
```

y show_skate mostrara dicho objeto

Creamos skate el cual es el nuevo objeto tipo Skate. Rellenamos dicho objeto con el alias new Skate(la clase) mas los parámetros declarados en el constructor (cap1)

Gracias al get puedo enseñar los parámetros del objeto sin llegar a tocarlos directamente (cap2)

con new_long creare mi objeton Long

```
const new_long = async () => {
    let long: Long
    const largo =  parseInt(await leerTeclado('Defina el largo del long porfavor \n <110 -- long corto \n >110 --
    const marca = await (leerTeclado('introduzca la marca deseada  \n Santa cruz \n Element \n Creature \n family
    const ancho = parseInt(await leerTeclado('Elija el ancho de la tabla,\n este depende del tamaño de su pie asi
    const ejes = parseInt(await leerTeclado('Elija los ejes:\n Ancho tabla -- Numero del eje \n < 7.5 -- 130 \n 7.
    const ruedas = parseInt(await leerTeclado('Elija las ruedas: \n para patinar street -- <56 \n para patinar bo
    const rodamientos = await leerTeclado('Elija el tipo: \n  rodamientos de metal \n  rodamientos de cerámica')
    const tornilleria = await leerTeclado('Elija el tipo tornilleria blaca \n   tornilleria negra')
   long = new Long(largo ,marca ,ancho,ejes, ruedas, rodamientos, tornilleria)
return long
```

mientras que con show_long lo enseñaré

```
const show_Long = (show: Long | undefined) =>{
    if (show!=undefined){
        console.log('/n ')
        console.log(`la marca es: ${show.marca}`)
        console.log(`El ancho es: ${show.ancho}`)
        console.log(`Los ejes son: ${show.ejes}`)
        console.log(`Las ruedas son: ${show.ruedas}`)
        console.log(`Los rodamientos son: ${show.rodamientos}`)
        console.log(`La tornilleria es: ${show.tornilleria}`)
        console.log(`El precio es: ${show.precio_total()}`)
    }
    else {
        console.log('primero crea el long porfavor')
    }
}
```

show_long a la par de ser tipo Long también lo considero indefinido y así me ahorro ir al objeto vacío, se podría llamar medida de seguridad debido a que hasta que no creemos el objeto Long este esta vacío es decir indefinido

Mediante el método set podemos cambiar los valores del objeto pero sin tocarlos directamente, esto es gracias al modificador de acceso protected declarado en la clase Skate

```
protected _ejes: number
```

```
set ejes (_ejes:number) {
    this._ejes=_ejes
}
```

Referenciamos a los valores del objeto mediante la variable de referencia this

```
const actualizar_skate = async (skate: Skate | undefined ) => {
    if ( skate != undefined ){
        const leer = parseInt(await leerTeclado('elegir que actualizar \n
            switch(leer){
                case 1:
                skate.ancho = parseInt( await leerTeclado('Elija el ancho
            break;
                case 2:
                skate.ejes = parseInt (await leerTeclado('Elija los ejes:\
            break ;
                case 3:
                skate.ruedas = parseInt ( await leerTeclado('Elija las rue
            break ;
                case 4:
                skate.rodamientos = await leerTeclado('Elija el tipo:\n rc
            break;
                case 5:
                skate.tornilleria = await leerTeclado ('Elija el tipo:\n t
            break;
            case 6:
                skate.marca = await leerTeclado ('Elija nueva marca:')
            break
            }
        }
        else {
            console.log('porfa crea la tabla')
        }
    }
```

Otro sencillo ejemplo con la clase Long, con el método parseInt transformamos el tipo string a number, mientras que con el await esperamos a respuesta por consola de la funcion leerTeclado

```
const actualizar_long = async (_new_long: Long | undefined ) => {
    if ( _new_long != undefined ){
        const leer = parseInt(await leerTeclado('elegir que actualizar \n'))
            switch(leer){
                case 1 :
                _new_long.largo = parseInt ( await leerTeclado ('dame el largo'))
            break;
                case 2:
                _new_long.ancho = parseInt( await leerTeclado('nuevo ancho '))
            break;
                case 3:
                _new_long.ejes = parseInt (await leerTeclado('nuevo eje'))
            break ;
                case 4:
                _new_long.ruedas = parseInt ( await leerTeclado('Elija las ruedas:
            break ;
                case 5:
                _new_long.rodamientos = await leerTeclado('Elija el tipo: \n Rodam
            break;
                case 6:
                _new_long.tornilleria = await leerTeclado ('Elija el tipo:\n torni
            break;
            case 7:
                _new_long.marca = await leerTeclado ('Elija nueva marca:')
            break

            }
        }
    else{
        console.log('primero crea la tabla')
        }
    }
```

Ahora declaro la variable main la cual esta relacionada con el menú anteriormente mostrado en la cual esperamos una respuesta de la función menu, con el case creamos opciones a elegir en donde llamamos a las distintas funciones creadas anteriormente

```
const main = async () =>{
    let numero : number
    let skate : Skate | undefined
    let long : Long |undefined
    do{
    numero = await menu()
    switch(numero){
        case 1 :
            skate = await new_skate()
        break
        case 2:
            long = await new_long()
        break
        case 3:
            show_skate(skate)
        break
        case 4:
            show_Long(long)
        break
        case 5:
            await actualizar_skate(skate)
        break
        case 6:
            await actualizar_long(long)
        break
    }
}
while (numero != 29)
}
```

Un ejemplo en consola

```
PS H:\Base_de_datos\1ºtrismetre\Visual-estudios\09_proyecto> node dist
Pofavor elija una de las siguientes opciones para continuar
 1-- Nuevo skate
 2-- Nuevo long
 3-- Mostrar skate
 4-- Mostrar long
 5-- Actualizar skate
 6-- Actualizar long
 29 --exit
:
```

Elegiré Nuevo skate

```
introduzca la marca deseada
 Santa cruz
 Element
 Creature
 family
: Creature
```

```
Elija el ancho de la tabla,
 este depende del tamaño de su pie asi como:
 numero de pie -- ancho
 <37 -- <8
 37/42 -- 8/8.5
 42> -- 8.5>
: 8.6
```

```
Elija los ejes:
 Ancho tabla -- Numero de eje
 < 7.5 -- 130
 7.5< x <8.5 -- 140
 8.5 > -- 150: 150
```

```
Elija las ruedas:
 para patinar street -- <56
 para patinar bowl 56>: 54
```

```
Elija el tipo:
 rodamientos de metal
  rodamientos de cerámica: metal
```

```
Elija el tipo:
 tornilleria blnaca
 tornilleria negra: negra
```

Ya estaría creado¡¡

Ahora seleccionaré mostrar skate skate

```
Pofavor elija una de las siguientes opciones para continuar
 1-- Nuevo skate
 2-- Nuevo long
 3-- Mostrar skate
 4-- Mostrar long
 5-- Actualizar skate
 6-- Actualizar long
 29 --exit
: 
```

```
La marca es Creature
El ancho es 8
Los ejes son de tipo: 150
La marca son de grosor: 54mm
La rodamientos son de: metal
La marca es de color: negra
El precio total es: 235.5
```

por último actualizaremos skate

```
 : 5
elegir que actualizar
  1-- ancho 2-- ejes 3-- ruedas
 4-- rodamientos 5-- tornilleria 6-- marca:
```

```
Elija el ancho de la tabla,
 este depende del tamaño de su pie asi como:
 numero de pie -- ancho
 <37 -- <8
 37/42 -- 8/8.5
 42> -- 8.5>
: 9
```

Volvemos a ver el skate

```
La marca es Creature
El ancho es 9
Los ejes son de tipo: 150
La marca son de grosor: 54mm
La rodamientos son de: metal
La marca es de color: negra
El precio total es: 250.5
```

Uy a cambiado el precio¡

Por ultimo voy a mostrar una asignación de clase manual dentro de un array

```
//asignación manual la haré en forma de array
const new_array= ()=>{
    let array = new Array<Skate>();
        array[1] = new Skate('Element',8.7,140,55,'metal','negra');
        array[2] = new Long (110,'family',9.5,150,54,'cerámica','blanca')
        array[3] = new Skate('Santa cruz',8.5,130,59,'metal','blanca')
        array[4] = new Long (135,'Creature',9,150,56,'metal','negra')
        delete array[1]
    return array
}
  console.log(new_array())
```

Aqui creamos cuatro clases dentrod el array y borramos una

```
Long {
  _marca: 'family',
  _ancho: 9.5,
  _ejes: 150,
  _ruedas: 54,
  _rodamientos: 'cerámica',
  _tornilleria: 'blanca',
  _largo: 110
},
```

```
Skate {
  _marca: 'Santa cruz',
  _ancho: 8.5,
  _ejes: 130,
  _ruedas: 59,
  _rodamientos: 'metal',
  _tornilleria: 'blanca'
},
```

```
Long {
  _marca: 'Creature',
  _ancho: 9,
  _ejes: 150,
  _ruedas: 56,
  _rodamientos: 'metal',
  _tornilleria: 'negra',
  _largo: 135
}
```