

## MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Sử dụng các loại decorators.

## NỘI DUNG

### Bài 1: Class decorator

Thực hiện bài tập ví dụ

```
function Logger(constructor: Function) {
    console.log('Logging...');
    console.log(constructor);
}

@Logger
class Person {
    name = 'Max';

    constructor() {
        console.log('Creating person object...');
    }
}

const pers = new Person();

console.log(pers);
```

### Bài 2: Decorator factory

Thực hiện bài tập ví dụ

```
function Logger(logString: string) {
    return function(constructor: Function) {
        console.log(logString);
        console.log(constructor);
    };
}

@Logger('LOGGING - PERSON')
class Person {
    name = 'Max';

    constructor() {
        console.log('Creating person object...');
    }
}
```

## Bài 3: Property decorator

### Thực hiện bài tập ví dụ

```
function Log(target: any, propertyName: string | Symbol) {  
    console.log('Property decorator!');  
    console.log(target, propertyName);  
}  
  
class Product {  
    @Log  
    title: string;  
    private _price: number;  
  
    set price(val: number) { }  
  
    constructor(t: string, p: number) {  
        this.title = t;  
        this._price = p;  
    }  
  
    getPriceWithTax() {}  
}
```

## Bài 4: Method decorator

```
function Log3(target: any, name: string | Symbol, descriptor: PropertyDescriptor) {
    console.log('Method decorator!');
    console.log(target);
    console.log(name);
    console.log(descriptor);
}

class Product {

    title: string;
    private _price: number;

    set price(val: number) { }

    constructor(t: string, p: number) {
        this.title = t;
        this._price = p;
    }

    @Log3
    getPriceWithTax() { }
}
```

## Bài 5: Autobind decorator

```
function Autobind(_: any, _2: string, descriptor: PropertyDescriptor) {
    const originalMethod = descriptor.value;
    const adjDescriptor: PropertyDescriptor = {
        configurable: true,
        enumerable: false,
        get() {
            const boundFn = originalMethod.bind(this);
            return boundFn;
        }
    };
    return adjDescriptor;
}

class Printer {
    message = 'This works!';

    @Autobind
    showMessage() {
        console.log(this.message);
    }
}

const p = new Printer();
p.showMessage();

const button = document.querySelector('button')!;
button.addEventListener('click', p.showMessage);
```

## Bài 6: LEARNING TYPESCRIPT

Tạo simple project có cấu trúc sau:

- Tên project: LEARNING\_TYPESCRIPT
- Project gồm: src/app.ts (biên dịch đến dist/app.js), index.html, style.css, tsconfig.json

*app.ts*

Sử dụng decorator để validate input nhập vào

Login

@Required  
title: string;

Username:

Start

### Gợi ý

```
interface ValidatorConfig {  
  [property: string]: {  
    [validatableProp: string]: string[]; // ['required', 'positive']  
  };  
}  
  
const registeredValidators: ValidatorConfig = {};  
  
function Required(target: any, propName: string) {  
  registeredValidators[target.constructor.name] = {  
    ...registeredValidators[target.constructor.name],  
    [propName]: ['required']  
  };  
}  
  
function PositiveNumber(target: any, propName: string) {  
  registeredValidators[target.constructor.name] = {  
    ...registeredValidators[target.constructor.name],  
    [propName]: ['positive']  
  };  
}
```

### \*\*\* Yêu cầu nộp bài:

SV nén file (hoặc share thư mục google drive) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---