09_装饰器

运行方法

```
npm i
npm start
```

概念介绍

> 装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。
>
> 这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法签名完整性的前提下，提供了额外的功能。
>
> 我们通过下面的实例来演示装饰器模式的用法。其中，我们将把一个形状装饰上不同的颜色，同时又不改变形状类。

类与方法的装饰器

```
// 类装饰器
function anotationClass(id){
    console.log('anotationClass evaluated', id);
    return (target) => console.log('anotationClass executed', id);
}
// 方法装饰器
function anotationMethods(id){
    console.log('anotationMethods evaluated', id);
    return (target, property, descriptor) => console.log('anotationMethods executed',
id);
}

@anotationClass(1)
@anotationClass(2)
class Example {
    @anotationMethods(1)
    @anotationMethods(2)
    method(){}
}
```

日志应用和切面AOP

```
function log(target, name, descriptor) {
    var oldValue = descriptor.value;
```

```
    descriptor.value = function () {
        console.log(`Calling "${name}" with`, arguments);
        return oldValue.apply(null, arguments);
    }
    return descriptor;
}


// 日志应用
class Maths {
    @log
    add(a, b) {
        return a + b;
    }
}
const math = new Maths();
// passed parameters should get logged now
math.add(2, 4);
```