Decorator

Dentre os 23 padrões de projeto descritos no livro “Design Patterns”, os padrões GoF, o padrão estrutural *Decorator* apresenta a possibilidade de adicionar comportamentos a classes existentes no sistema de forma dinâmica. Ele pode ser usado para modificar sistemas existentes onde desejamos adicionar funcionalidades adicionais a objetos sem a necessidade de modificar de forma acentuada o código. Isso se torna mais útil quando os programas possuem funcionalidades que requerem uma grande quantidade de objetos, como, por exemplo, um jogo criado em JS. Nesse caso o programador poderia ter que criar centenas de subclasses para diferentes combinações de personagens de um jogo.

O padrão Decorator não é focado em como os objetos são criados, mas sim na extensão de suas funcionalidades. Com o Decorator vamos trabalhar com uma única classe base, adicionando funcionalidades adicionais de forma progressiva. Ou seja, invés de adicionar subclasses adicionamos propriedades ou métodos na classe base.

Porém, existem desvantagens. Se usado de forma imprudente, o padrão pode complicar significantemente a arquitetura da aplicação, já que vai adicionar vários objetos pequenos e similares no código. Isso deixará o código difícil de manipular e desenvolvedores que não estão familiarizados com o padrão podem ter dificuldade em entender o motivo pelo qual ele está sendo usado.

Fontes:

-https://medium.com/@washingtonbr/padrões-de-projeto-e-boas-práticas-em-javascript-d3b49c5c61a5

-https://www.patterns.dev/posts/classic-design-patterns/#decoratorpatternjavascript