Design Patterns

Design patterns ou padrões de projetos são soluções generalistas para problemas recorrentes durante o desenvolvimento de um software. Não se trata de um framework ou um código pronto, mas de uma definição de alto nível de como um problema comum pode ser solucionado.

# A Pattern Language

* 1978;
* Christopher Alexander, Sara Ishikawa e Murray Silverstein;
* 253 tipos de problemas/desafios de projetos

# Formato de um pattern

* Nome;
* Exemplo;
* Contexto;
* Problema;
* Solução.

# Using Patterns Languages for Object-Oriented Programs

* 1987;
* Kent Beck e Ward Cunningham;
* 5 padrões de projetos.

# Design Patterns: Elements of Reusable Object-Oriented Software

* 1994;
* Gang of Four (GoF);
* Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides.

# Tipos

* Criação;
* Estruturais;
* Comportamentais.

# Padrões de criação

Os padrões de criação são aqueles que abstraem e/ou adiam o processo de criação dos objetos. Eles ajudam a tomar um sistema independente de como seus objetos são criados, compostos e representados.

# Padrões de criação mais famosos

* Abstract Factory;
* Builder;
* Factory;
* Factory Method;
* Prototype;
* Singleton;

# Padrões estruturais

Os padrões estruturais se preocupam com a fama como classes e objetos são compostos para formar estruturas maiores.

# Padrões estruturais mais famosos

* Adapter;
* Brigde;
* Composite;
* Decorator;
* Facade;
* Business Delegate;
* Flyweight;
* Proxy.

# Padrões comportamentais

Os padrões de comportamento se concentram nos algoritmos e atribuições de responsabilidades entre objetos. Eles não descrevem apenas padrões de objetos ou de classes, mas também os padrões de comunicação entre objetos.

# Padrões comportamentais mais famosos

* Chain of Responsability;
* Command;
* Interpreter;
* Iterator;
* Mediator;
* State;
* Strategy;
* Template Method
* Visitor.

# Patterns mais utilizados

* Factory;
* Singleton;
* Decorator;
* Observer;
* Module;

# Factory

Todas as funções que retornam um objeto, sem a necessidade de chamá-las com o “new”, são consideradas funções Factory.

Obs: ver arquivo pasta js

# Singleton

O objetivo desse pattern é criar uma única instância de uma função construtora e retorná-la toda vez em que for necessário utilizá-la.

jQquery

# Decorator

Uma função decorator recebe uma outra função como parâmetro e estede o seu comportamento sem modificá-la explicitamente.

Proposta:

<https://github.com/tc39/proposal-decorators>

usar decorator com typescript

# Observer

É um pattern muito popular em aplicações javascript. A instância (subscriber) mantém uma coleção de objetos (observers) e notifica todos eles quando ocorrem mudanças no estado.

Vue

RxJs

# Module

É um pattern que possibilita organizarmos melhor o nosso código, sem a necessidade de expor variáveis globais