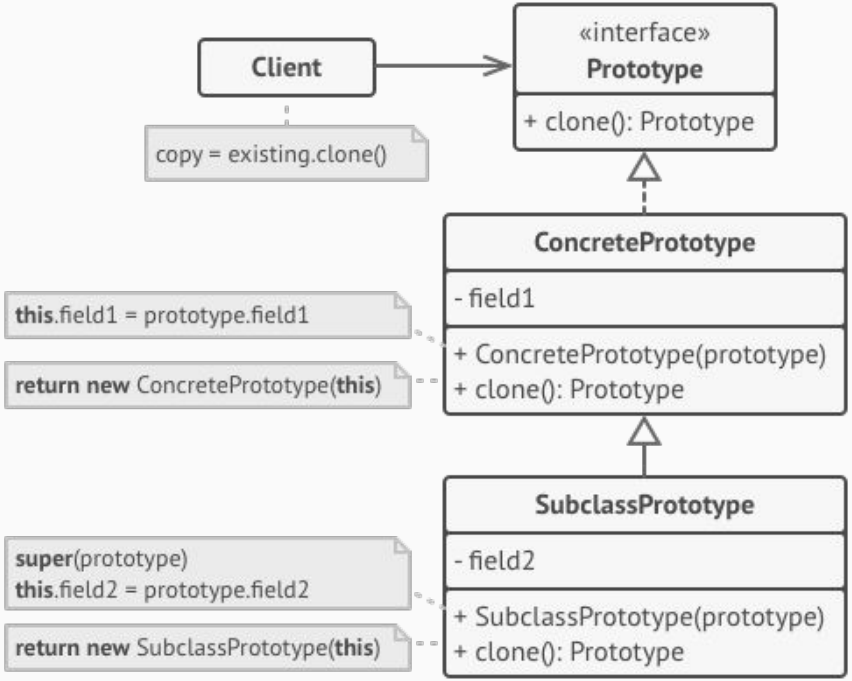
Prototype no TypeScript

Alunos: Mario Matheus, Rafael Coelho, Robson Ribeiro e Wagner Matheus

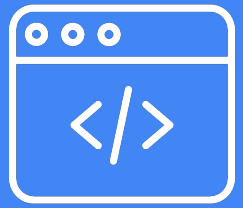
O que é o Prototype?

Também conhecido como Clone, Prototype é um padrão de projeto criacional que permite copiar objetos existentes sem tornar seu código dependente de suas classes.

Estrutura


Client
«interface»
Prototype
+ clone(): Prototype
copy = existing.clone()
ConcretePrototype
- field1
+ ConcretePrototype(prototype)
+ clone(): Prototype
this.field1 = prototype.field1
return new ConcretePrototype(this)
SubclassPrototype
- field2
+ SubclassPrototype(prototype)
+ clone(): Prototype
super(prototype)
this.field2 = prototype.field2
return new SubclassPrototype(this)

Exemplo de Código



```
class Prototype {
    public primitive: any;
    public component: object;
    public circularReference: ComponentWithBackReference;

    public clone(): this {
        const clone = Object.create(this);

        clone.component = Object.create(this.component);
        clone.circularReference = {
            ...this.circularReference,
            prototype: { ...this },
        };

        return clone;
    }
}

class ComponentWithBackReference {
    public prototype;

    constructor(prototype: Prototype) {
        this.prototype = prototype;
    }
}

function clientCode() {
    const p1 = new Prototype();
    p1.primitive = 245;
    p1.component = new Date();
    p1.circularReference = new ComponentWithBackReference(p1);

    const p2 = p1.clone();
    if (p1.primitive === p2.primitive) {
        console.log('Primitive field values have been carried over to a clone. Yay!');
    } else {
        console.log('Primitive field values have not been copied. Booo!');
    }
    if (p1.component === p2.component) {
        console.log('Simple component has not been cloned. Booo!');
    } else {
        console.log('Simple component has been cloned. Yay!');
    }

    if (p1.circularReference === p2.circularReference) {
        console.log('Component with back reference has not been cloned. Booo!');
    } else {
        console.log('Component with back reference has been cloned. Yay!');
    }

    if (p1.circularReference.prototype === p2.circularReference.prototype) {
        console.log('Component with back reference is linked to original object. Booo!');
    } else {
        console.log('Component with back reference is linked to the clone. Yay!');
    }
}

clientCode();
```

Aplicabilidades

- Use o padrão Prototype quando seu código não deveria depender das classes concretas de objetos que você precisa copiar.

- Use o padrão quando desejar reduzir o número de subclasses que diferem apenas na maneira como inicializam seus respectivos objetos. Alguém poderia ter criado essas subclasses para poder criar objetos com uma configuração específica.

⚖ Prós e Contras

- Você pode clonar objetos sem acoplar às suas classes concretas.
- Você pode se livrar do código de inicialização repetido em favor da clonagem de protótipos pré-criados.
- Você pode produzir objetos complexos de forma mais conveniente.
- Você obtém uma alternativa à herança ao lidar com predefinições de configuração para objetos complexos

- A clonagem de objetos complexos com referências circulares pode ser muito complicada.

Demonstração