
 The picture can't be displayed.

DESIGN PATTERNS

INTRODUCTION

 The picture can't be displayed.

What is design pattern ?

Design patterns are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

What does the pattern consist of?

Most patterns are described very formally so people can reproduce them in many contexts.

Here are the sections that are usually present in a pattern description:

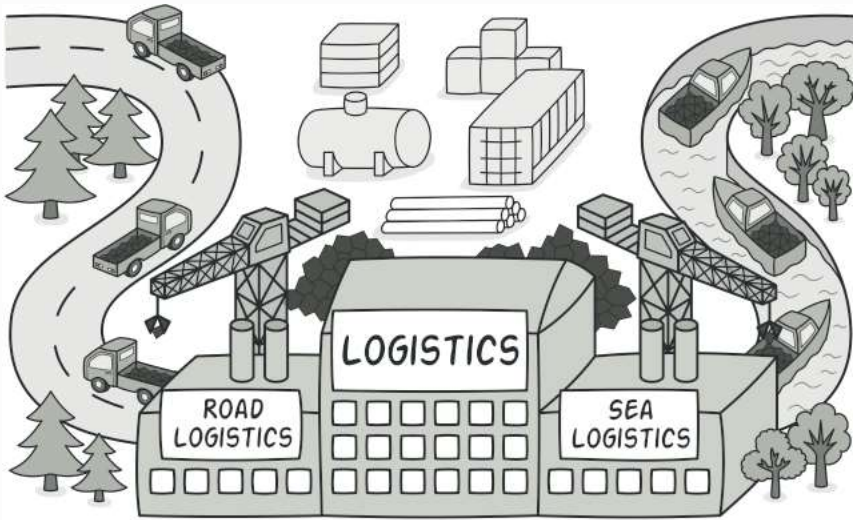
- **Intent** of the pattern briefly describes both the problem and the solution.
- **Motivation** further explains the problem and the solution the pattern makes possible.
- **Structure** of classes shows each part of the pattern and how they are related.
- **Code example** in one of the popular programming languages makes it easier to grasp the idea behind the pattern.

In addition, all patterns can be categorized by their *intent*, or purpose. This book covers three main groups of patterns:

- **Creational patterns** provide object creation mechanisms that increase flexibility and reuse of existing code.
- **Structural patterns** explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.
- **Behavioral patterns** take care of effective communication and the assignment of responsibilities between objects.

Creational patterns

1.Factory Method



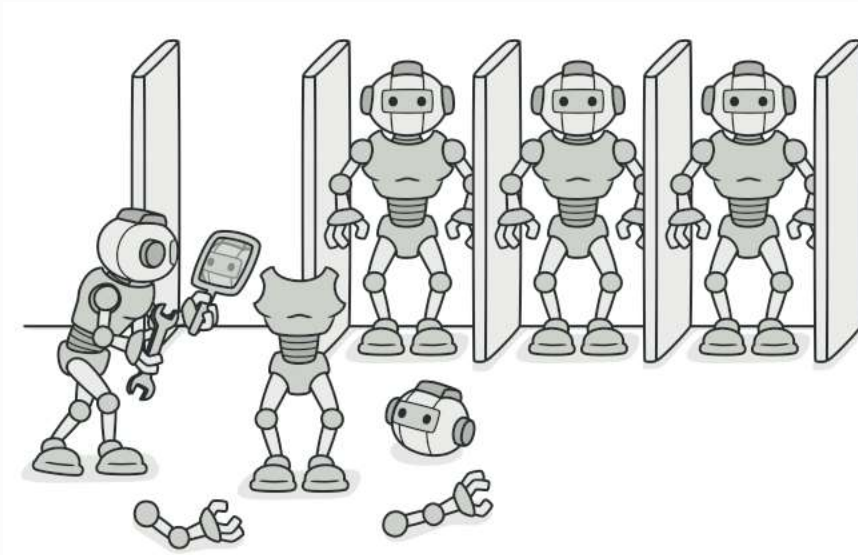
Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

[Design Pattern Link](#)

[JS Implementation](#)

Creational patterns

2. Prototype



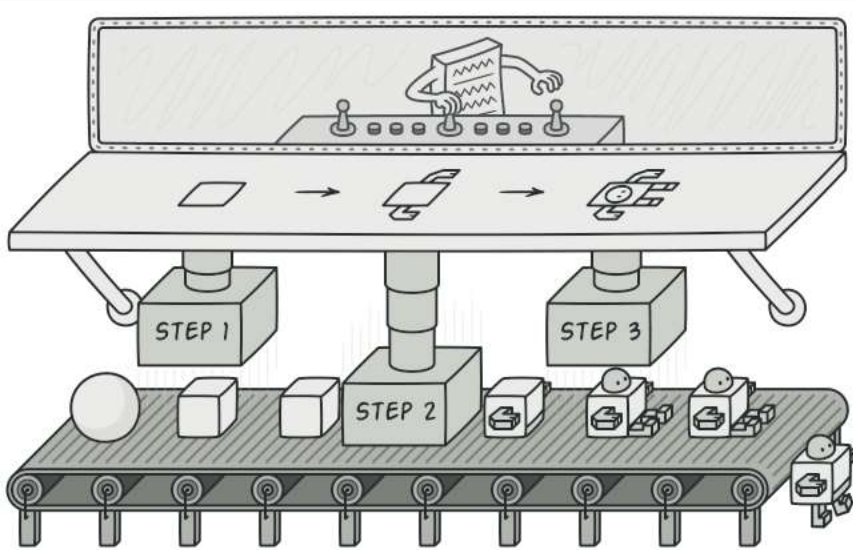
Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

[Design Pattern Link](#)

[JS Implementation](#)

Creational patterns

2. Bulder



Builder is a creational design pattern that lets you construct complex objects step by step.

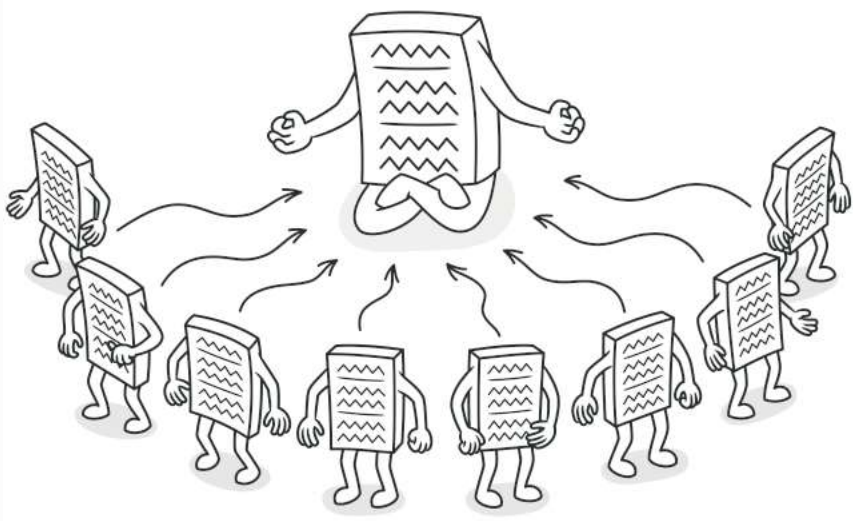
The pattern allows you to produce different types and representations of an object using the same construction code.

[Design Pattern Link](#)

[JS Implementation](#)

Creational patterns

3. Singleton



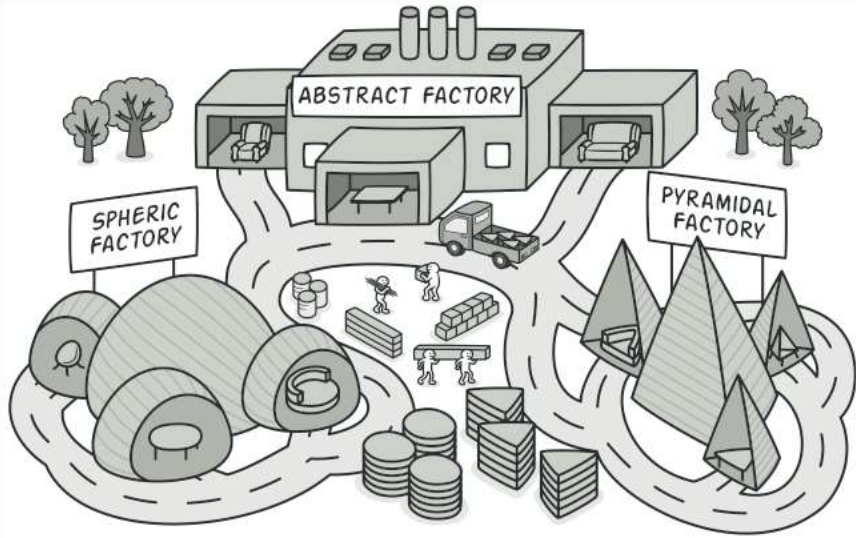
Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.

[Design Pattern Link](#)

[JS Implementation](#)

Creational patterns

3. Abstract Factory



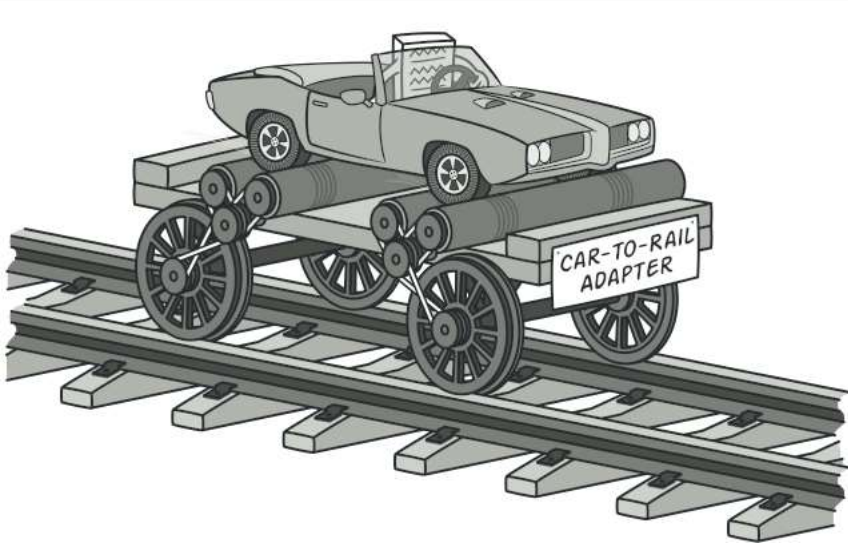
Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.

[Design Pattern Link](#)

[JS Implementation](#)

Structural patterns

1. Adapter (Wrapper)



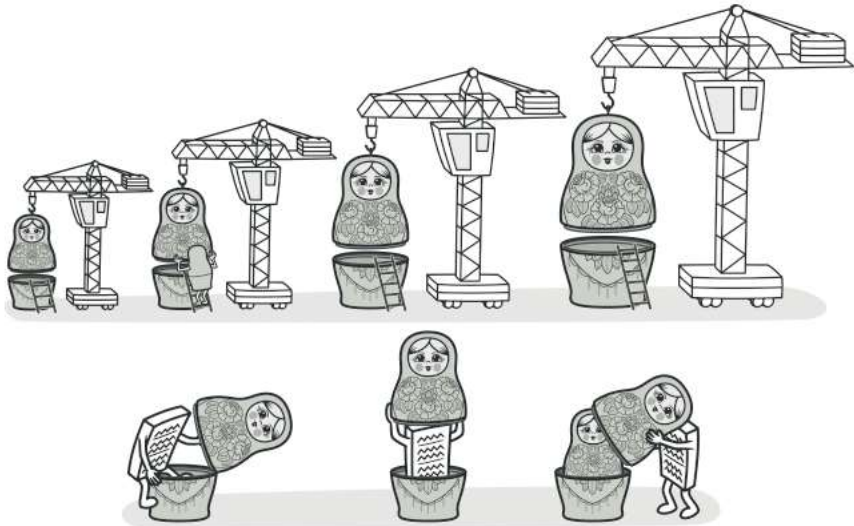
Adapter is a structural design pattern that allows objects with incompatible interfaces to collaborate.

[Design Pattern Link](#)

[JS Implementation](#)

Structural patterns

2. Decorator



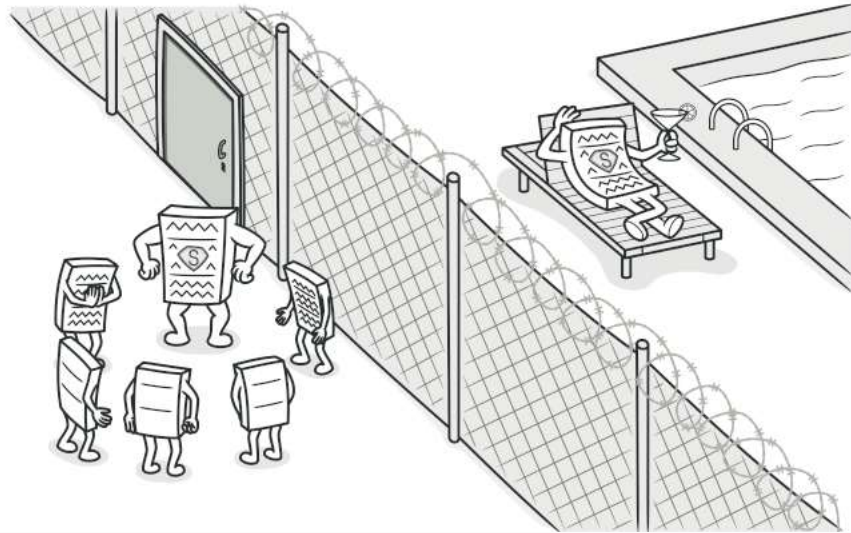
Decorator is a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors.

[Design Pattern Link](#)

[JS Implementation](#)

Structural patterns

3. Proxy



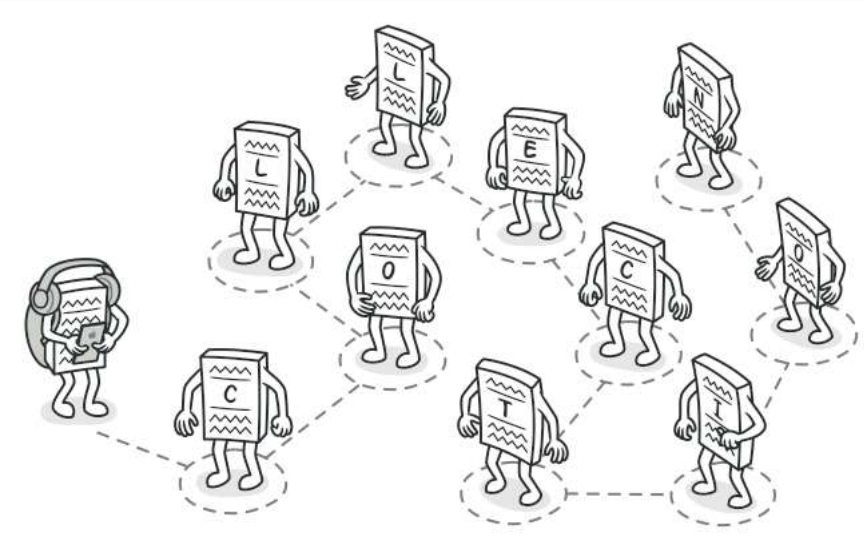
Proxy is a structural design pattern that lets you provide a substitute or placeholder for another object. A proxy controls access to the original object, allowing you to perform something either before or after the request gets through to the original object.

[Design Pattern Link](#)

[JS Implementation](#)

Behavioral patterns

1. Iterator



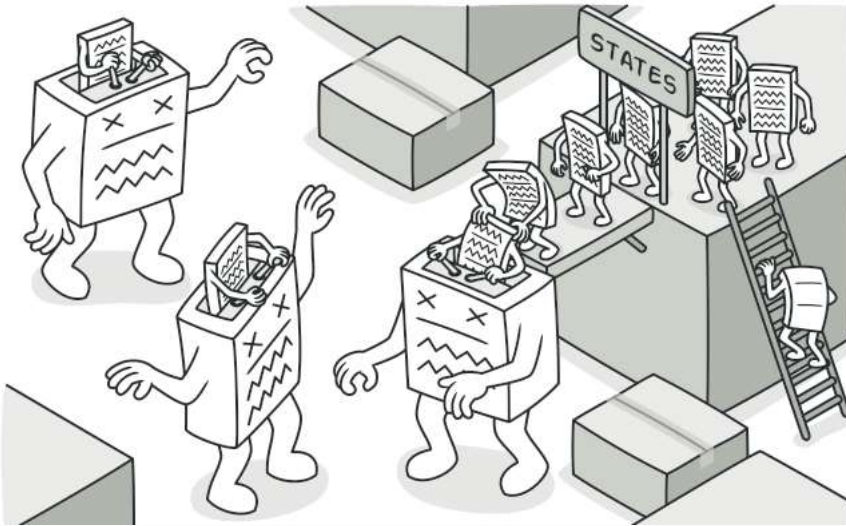
Iterator is a behavioral design pattern that lets you traverse elements of a collection without exposing its underlying representation (list, stack, tree, etc.).

[Design Pattern Link](#)

[JS Implementation](#)

Behavioral patterns

2. State



State is a behavioral design pattern that lets an object alter its behavior when its internal state changes.

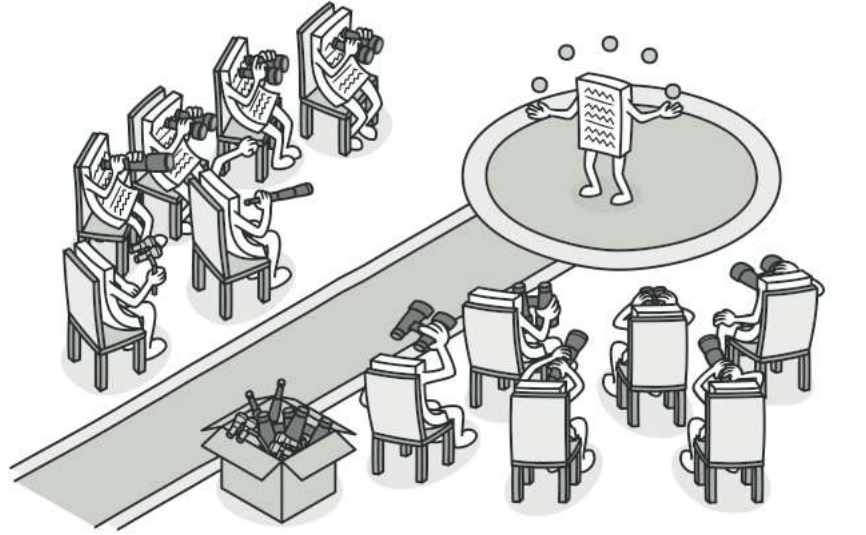
It appears as if the object changed its class.

[Design Pattern Link](#)

[JS Implementation](#)

Behavioral patterns

3. Observer



Observer is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing.

[Design Pattern Link](#)

[JS Implementation](#)