12 Mar 2019 / ۵ رجب ۱۴۴۰

Node.js Design paterns

Reusable, reliable solutions to problems that we face everyday in development.
- well documented
- Improve architecture
- write better code

Design paterns categories:

Creational	structural	Behavioral
Abstract Factory	Adaptor	Interpreter
Builder	Bridge	Iterator
Factory Method	Composite	Mediator
Prototype	Facade	Observer
Singleton	Proxy	Visitor
	⋮	⋮

creational: Deal with classes instentiation, the creation of object instences in our application.
structural: Deal with the way objects composed or put together.
Behavioral: Defines how objects interact with each other.

روز بزرگداشت نظامی گنجوی

Anti-Pattern are bad solutions that cause problems.

- modifying the prototype on an instance

```
person.__proto__.address={};
```

- call back hell

:

---

singletons: sometimes you need to make sure that you have one & only one instance of an obj.

Prototype: we have the exact same property in objects, but there are little customization. We can have clone of a master design.

Factory: in js we need different objects to be instantiate. we can improve our code by encapsulate all of the constructors into a single module, and creating a function that will create the objects for us.
it will define an interface for creating an obj, but let subclasses decide which class to instantiate.
Factory method lets a class defer instantiation to subclasses.

Builder: it will use when you want to customise

10 Mar 2019 / ۱۱۴۴ رجب

the objects you'll create. this would seperate the construction of a complex object from its representation so that the same construction process can create different representations.

Adapter: convert the interface of a class into another clients expect. Adapter lets classes work together that couldn't otherwise! because of incompatible interfaces. for ex: we want to use codes that worked in browser and written with javscript now we wanna use the exact same codes in nodejs.

proxy: proxy is an object that controls access to another object. for example ceo is bussy so we call to his assistant, here assistant is proxy for ceo. In general instead of using an object that needs a lot of resource we can use a proxy. In addition to managing expensive obj proxies can be use to manage a remote resource, provide data validation of input, provide cache data, even log actions

Composite: composite work with trees. trees are use everywhere like using file systems. compose obj into tree structures to represent part-whole hirerarchies. composite lets clients treat individual obj & composition of objects uniformly.

Decorator: allows us to dynamically attach properties to our existing objects. Sometimes in our applications we have base objects we need, we can decorate our methods and properties to create many custom variations of the same obj.

chain of responsibility: This design pattern allows us to chain together objects to handle a request, a request is sent to an object to handle it and then that handler could process it and return the result or it could pass the result to next handler.

Command: commands are objects that contain all of the data necessary to execute a specific action on the client.

Iterator: uniform interface to interacting with lists, collections, arrays, or any type of aggregate object. It provides a way to access the elements of an aggregate object sequentially.

observer: define one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

۱۶ پنجشنبه اسفند

۲۹ جمادی‌الثانیه ۱۴۴۰ / 07 Mar 2019

strategy: with strategy pattern you create multiple algorithms for specific task and dynamically decide which algorithm to run on runtime. define a family of algorithms, encapsulate each one, and make them interchangeable.

۱۷ جمعه اسفند