A design patter is a general reusable solution to an occurring problem in software design. It can also be described as a blue print to handle recurring problems in software development. And as the article mentions is a template on how we solve problems. Patterns can be divided into three categories: constructional, structural and behavioral patterns.

**A constructional pattern** focuses on ways to create objects or classes. It’s helpful because it let us avoid repetition and it helps us keep track of the objects created.

A constructor pattern creates objects to be used by functions both locally and globally. It has some drawback too and it need a constructor hence the name.

**A** **Structural** **pattern** focuses on ways to manage relationships between objects so that the application is designed in a scalable way. An important aspect of structural patterns is to ensure that a change in one part of your application does not affect all other parts. So it makes it easy to modify parts of it later on.

A facade is an object that provides a simplified interface to a larger body of code, such as a class library or objects and complex functions. This pattern provides a convenient higher-level interface to a larger body of code, hiding its true underlying complexity.

**A** **Behavioral** **pattern** primarily focuses on communication between objects.

A decorator pattern promotes code reusability and is a flexible alternative to sub-classing. It allows the user to modify existing objects without the need to change the underlying code. It also allows adding functionality or features later on.

<https://github.com/aed7/IS_217> (Sorry for the mess I’ve been having issues with committing changes to another repo for some reason all the files get committed too. The file’s names for the project are example\_1 through example\_4 and the javascript files are: constructor, facade, decorator and two\_patterns. Once again I’m sorry for the mess I’m trying my best to fix it.)