The Command Pattern in Adobe Brackets

Patterns in code design are those that are ever more common due to their efficiency, ability to easily be reused, and specific nature. The Gang of Four (GOF) redefined these patterns so that software engineers may come to recognize the patterns that may serve the purpose necessary and keep the code clean and manageable at that. The purpose of this research is to explore a real-world example of the use of a defined and published design pattern. Thus, I will go through a piece of open source software: Adobe Brackets (Menu Section), discussing the pattern, showing the existence of the pattern in the software, and how it is used. First, let me restate my research question.

Research Question: What are the uses of the Command Pattern in "Adobe Brackets", how is it implemented, and what are the similarities and differences in the pattern structure in JavaScript and C++?

Adobe Brackets

Adobe Brackets is a modern text editor made to understand web design code, thus generally made for web development. As one can tell, the software was created by Adobe Systems, but as it is open source it is maintained on GitHub. The software was written in HTML, CSS (via LESS), and JavaScript, and is cross-platform, made available for Mac, Windows, and Linux. Overall, the software was crafted for web developers and front-end developers.

The Command Pattern

The Command pattern is behavioral design pattern with the intent to encapsulate a request as an object. In other words, the method in which an object is used to encapsulate the information required to perform an action, or trigger an event, at a later time is known as “The Command Pattern”. The Gang of Four structure of the Command Pattern utilizes at the very least 4 actor classes:

* Command
  + The Command class is one that would declare the interface for the execution of an operation.
* ConcreteCommand
  + The ConcreteCommand class is one that can be one or many, as it specifies the action to be executed and thus contains the code to handle the desired action.
* Client
  + This is where the Command and Concrete classes are instantiated and depending on the purpose it may be set to a receiver that will utilize it.
* Invoker
  + The Invoker class is one that invokes the specific encapsulated ConcreteCommand by calling its execution method.
* Receiver
  + The Receiver class is a generic class that assists in carrying out requests. It is “generic” since it is not a specific class; any class can be a receiver.

The JavaScript version of the Command Pattern harbors much similarities to the Gang of Four structure. The intention, of course, is still to encapsulate method invocation for requests/ operations into a single object, providing the programmer to parameterize and pass method calls for later execution. The differences are located in the code and class structure of the pattern, as one might expect. Since there is a great difference in the general structure and abilities of objects in JavaScript and C++/ Java, the GOF Command Pattern class structure does not have the same structure in JavaScript. The structure in JavaScript attempts to mimic a C++ class structure, by utilizing JS’s ability to allow encapsulated functionalities within a variable using an Object Literal. Object Literals allow the programmer to define and create an object in one statement, and since it is an “object” one can nest other variables or functions within it, all between two curly braces. Since the pattern rests upon the ability to execute encapsulated actions at will, the JS object ensures the encapsulation by adding a definition for an “execute” method, that allows narrow access to the desired action through execution.

Use of the Pattern in Adobe Brackets

In choosing a specific area of Adobe Brackets to study, I looked at the “Command” section – more specifically, the “Menu” area. It is commonly known that, in terms of software design, it is common to design a Menu with each menu item harboring its own functionalities. It is true that the programmer could simply utilize a function for each menu item, but that would lead to multiple extensive and cluttered functions. It is just like the folks at Adobe Systems to design clean and maintainable code, as they utilized the Command Pattern in their structure of the application’s Menu. It is evident that they utilized the pattern, since it was explained in the Architectural section of Brackets Blog and evident in the code.

I say it is evident in the code that a pattern exists, however it is not designed exactly as expected of a JavaScript version of the Command Pattern. Upon studying the code, the viewer may come to realize that the structure resembles that of the GOF definition of the Command Pattern. For instance, there are a number of JavaScript “class” structures being utilized rather simple JavaScript object (a variable imbued with an Object Literal). The classes utilized in this section are the: Menu, MenuItem, and Command, classes. The Menu class is used to represent a whole “top-level” menu section that can be toggled and opened at any specific location. The MenuItem class is used to represent the individual menu items that will invoke an action/ command – each item nests, or points to, a specific command. The Command class is used to represent the individual command that will carry out the user’s request, thus requires an invoker (MenuItem). The GOF structure would have the “MenuItem” class pose as the interface and make multiple classes that specify the invocation of each Concrete-MenuItem. However, I presume, the code was designed along the lines of the JavaScript version of the Command Pattern, making the “MenuItem” class act as an interface for itself, and making something else (the “Menu” class) the Receiver. Although this may be true, instead of utilizing defined encapsulated functions to determine what action to execute, each action to be executed is stored within the Command class, assigned an ID (to act as a soft pointer), and embedded within a single MenuItem class, which is added to an array of MenuItems in the Menu class. Therefore, while utilizing what first appears to be the GOF structure of the Command Pattern, the contents of the design stayed true to the JavaScript structure.

Conclusion

If we were to discuss and point out the evident differences and similarities between the GOF Command Pattern and the JavaScript Command pattern, I’m sure we’d both state that the two are similar in all aspects other than the code/ class structure. For both, the intent is still to encapsulate requests into objects for later usage, but the bridge that connects the two is built on code structure, which is why the pattern can exist in JavaScript. As it exists in JavaScript, the developers of the Adobe Brackets text editor utilized the pattern to accomplish their goal to structure a menu service for the application client, while keeping their code clean, to-the-point, manageable. In terms of the pattern itself, I have come to recognize that to locate a Command Pattern within code requires one to locate the invocation of executable functions (such as “execute()” or “run()”) and to come to understand the class structure that encapsulates the executable’s definition.

1. Adobe Brackets
   1. Background
   2. Purpose
   3. Implementation
2. The Command Pattern (GOF)
   1. Purpose
   2. Structure
   3. Code Design
3. The Command Pattern (JS)
   1. Purpose
   2. Structure
   3. Code Design
4. Differences & Similarities
5. The Source Code (Menu.js)
   1. Navigate to the file
   2. Where exactly it is
   3. How it is used
6. The Source Code (Brackets Blog & Documentation)
   1. Mentioned Architectural Patterns
   2. Classes
   3. The purpose of the “AddMenuItem” function
7. Conclusion
   1. Differences and Similarities
   2. Reason for using the Command Pattern
   3. How to recognize the pattern
8. .

.