Week 5 Research

By: Patrick Corcoran

The four pillars of object-oriented coding are abstraction, encapsulation, inheritance and polymorphism. The idea of abstraction in coding is to hide a complex piece of code in something like a function that is repeatable and gives you an expected outcome without the need to fully understand what it is doing. Having to parse through hundreds of lines of code can be rather inefficient since the wheel has already been invented so to speak, you just need to know how to implement it into what you are currently working on. Encapsulation in programming is also related to hiding certain pieces of code but for a different reason. Encapsulating certain objects or functions can reduce the possibility that certain variables can be overwritten on a global scale. If something doesn't need to be accessed by the entire function encapsulate it off and it can sit by itself and do what it is meant to do without interference. This can greatly reduce the effect of spaghetti code providing unwanted and unintended functionality. Inheritance is the ability of one particular object to receive the properties and methods of a different object that is largely similar except for perhaps a small portion. The benefit to this is reusability and reducing wasted space on repeat coding. It also allows subtle changes to be made to a parent object that will affect the downstream children, reducing time making changes since it only needs to be done once instead of for each particular object. Polymorphism sort of expands upon inheritance, in that the child objects inherit properties and methods from a parent, but

are also able to have their own custom properties and methods as well. This allows global things to be handled and edited in a single spot while still allowing customization where it is needed.

Exceptions are unexpected conditions that occur as your JavaScript is running. JavaScript thankfully has tools to help manage these events, however there are some best practices to keep in mind when implementing them. The first is to not overuse the try/catch/finally statements. They are good at what they are designed for but having too many will unnecessarily complicate your code and cause it to suffer from poor performance. It's unreasonable to apply this to every piece of your code, it's more important to focus it on areas that will have a propensity to cause issues. The next best practice is to avoid using browser specific methods. Some browsers have built in JavaScript error handling, but they may only work in Firefox and potentially cause issues in any other browser. It is best to stick with standardized methods. The last is modifying the try/catch/finally syntax when using asynchronous functions. If you use the standard try/catch/finally you will always get a rejected promise since it will run the catch before the asynchronous function is fully ran. It needs to be used with and await when calling the function so that the catch awaits the function call before running.

References:

https://www.freecodecamp.org/news/four-pillars-of-object-oriented-programming/
https://www.scaler.com/topics/exception-handling-in-javascript/
https://blog.bitsrc.io/javascript-exception-handling-patterns-best-practices-f7d6fcab735d