**Programming Paradigms**

A programming paradigm is a style or way of programming.

* **Imperative**: Programming with an explicit sequence of commands that update state.
* **Declarative**: Programming by specifying the result you want, not how to get it.
* **Structured**: Programming with clean, goto-free, nested control structures.
* **Procedural**: Imperative programming with procedure calls.
* **Functional** (Applicative): Programming with function calls that avoid any global state.
* **Function**-**Level** (Combinator): Programming with no variables at all.
* **Object-Oriented:** Programming by defining objects that send messages to each other. Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:
* **Class-based:** Objects get state and behavior based on membership in a class.
* **Prototype-based**: Objects get behavior from a prototype object.
* **Event-Driven**: Programming with emitters and listeners of asynchronous actions.
* **Flow-Driven:** Programming processes communicating with each other over predefined channels.
* **Logic** (Rule-based): Programming by specifying a set of facts and rules. An engine infers the answers to questions.
* **Constraint**: Programming by specifying a set of constraints. An engine finds the values that meet the constraints.
* **Aspect-Oriented**: Programming cross-cutting concerns applied transparently.
* **Reflective:** Programming by manipulating the program elements themselves.
* **Array:** Programming with powerful array operators that usually make loops unnecessary.