These are core principles of JavaScript (SOLID).

* **Single Responsibility Principle**. Everything should have only one reason to change. This will help developers understand the context and responsibility of what they are building and when there is a need for change.
* **Open-Closed Principle**. A change in behaviour should be possible without changing existing code, e.g. by using extension points and creating code that can be plugged in.
* **Liskov Substitution Principle**. Derived objects or types must be substitutable for their base. For Derick this is a more focused version of the Open-Closed principle.
* **Interface Segregation Principle**. A client should not be forced to depend on interfaces it doesn’t use. A problem is that there are no explicit interfaces in JS, but there are ways around this.
* **Dependency Inversion Principle**. Consists of two concepts, abstraction which states that we should depend on abstractions, not on concrete implementations and ownership that states that low level implementation should depend on high level concepts.

It states that if you have large monolithic chunks of code in your system, SOLID will help you break these into individual pieces. It will not decrease complexity, but will help you create abstractions and group details into larger concepts that we can reason about.