The elements of this language are entities called patterns. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Christopher Alexander

**Design pattern:** Gang of Four

- Creational
  - Ways to create objects
- Structural
  - Ways to combine/compose objects
- Behavioral
  - Ways to communicate between objects

**Anti-patterns:** Signs you are getting it wrong...

- Viscosity
  - Easier to do a hack than do the “Right Thing”
- Immobility
  - Can’t DRY out functionality
- Needless repetition
- Needless repetition
- Needless repetition
- Needless complexity from generality

**Design pattern:** Three tier architecture

- Application Server (Node, Flask, ...)
- Presentation tier
- Logic tier
- Web Server (Apache, nginx, ...)
- Database (PostgreSQL, MongoDB, ...)
- Data tier
Design pattern: Model View Controller

Controller

View

Model

User action

Update

Notify

Update

{ artist: 'Hugo Race',
  title: 'John Lee Hooker’s World Today',
  year: 2017,
  tracks: [{name: 'Hobo Blues'}, {name: 'Love Blues'}, ...]
}

Frameworks/Libraries

Angular

React

Backbone.js

Vue.js

Rails

Ember
Frameworks

- Event based (e.g., Backbone)
  - Changing the data triggers an event
  - Views register event handlers
- Two-way binding (e.g., Angular)
  - Assigning to a value propagates to dependent components and vice versa
- Efficient re-rendering (e.g., React)
  - Re-render all subcomponents when data changes

Frameworks: [React](https://calendar.perfplanet.com/2013/diff/)

Batched updates

Sub-tree re-rendering

https://calendar.perfplanet.com/2013/diff/