Model – View – Controller

A Design Pattern

Author: Hunter Chapman

Why you care

- MVC gives your code a solid architecture.
- MVC code pattern makes it super easy for you to follow other Dev's code.

M-V-C Basically:

- MVC is not much different than the code you are already writing.
- Just a bunch of classes and modules!
- The difference being a pattern of organization which allows us to separate out responsibilities in a standardized way.

Model – View – Controller

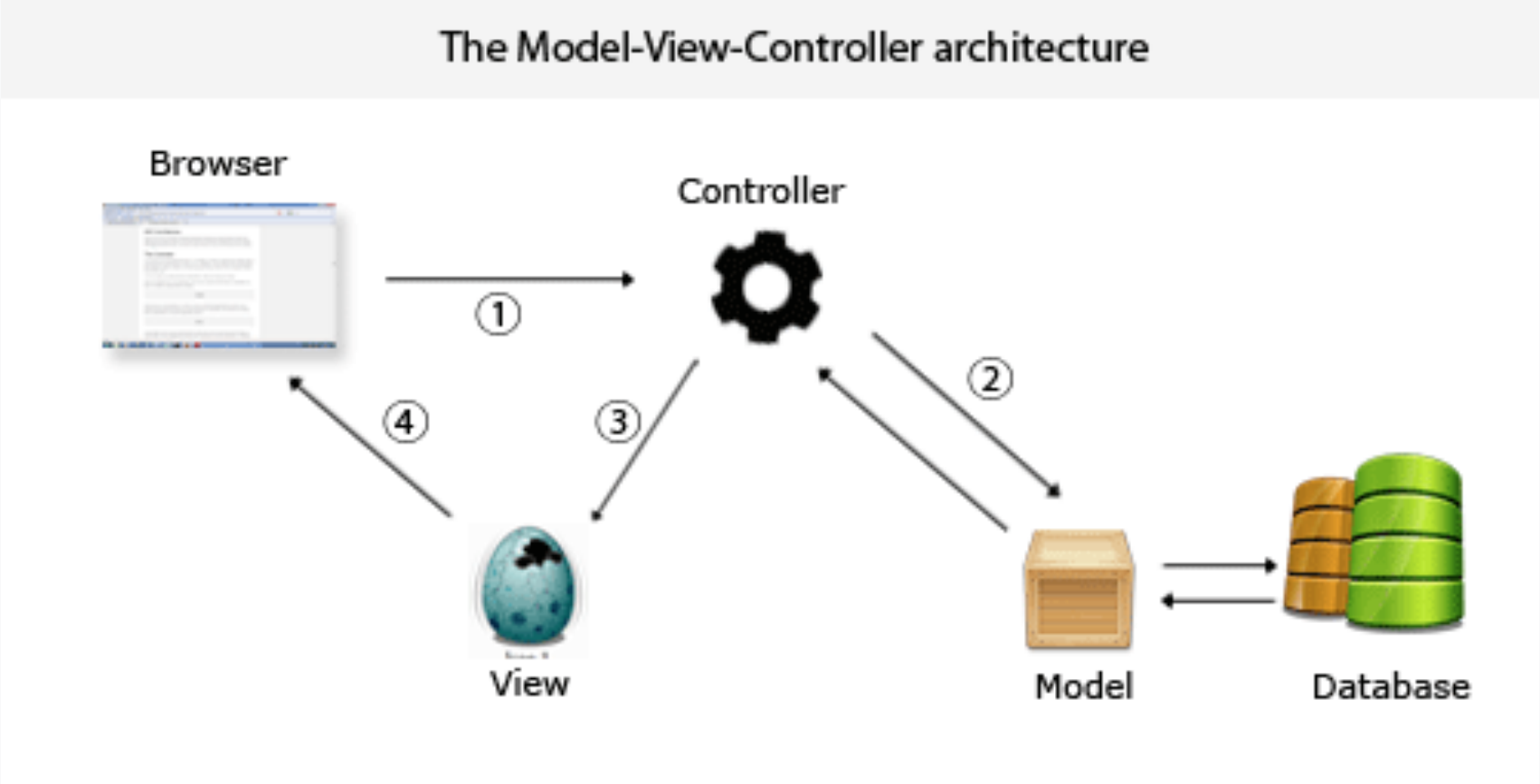
- What is it?
- In short: MVC is a design pattern.
- In not so short: MVC is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

Design Patterns

- A design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.

- Get used to this term. Start thinking of everything you do as implementing some pattern. You're just following a blueprint, filling in the gaps as needed with as little distinction from the standard pattern as possible.

M-V-C Summation:



There's a lot of data out there…

Sure, you can write all the codes… But where do you put it?

- Whose business is it to talk to users?
- What about displaying output?
- Should your classes all know how to parse user input?
- Where do heavy algorithms run?
- How do you separate 10,000 lines of code into a manageable structure?

Think about a restaurant.

- Does your waiter make your food?
- Do they know all the recipes?
- Do they run the kitchen?
- What *is* a waiter responsible for?

Everyone has their own role.
If you want a well-run restaurant,
you need to divvy up the work.

WHAT ROLES DO WE NEED?

- The waiters handle customer interaction
- They take your order and deliver your food

- The Chef combines ingredients to make the food
- They know the recipes and put everything together

- The kitchen manager manages the whole shebang
- They manage the orders coming in, tell the cooks what they need to make, and ensure the orders go to the servers for delivery.

M-V-C Bits:

- The view (waiter) handles user interaction
- It shows information to the user and gets user input

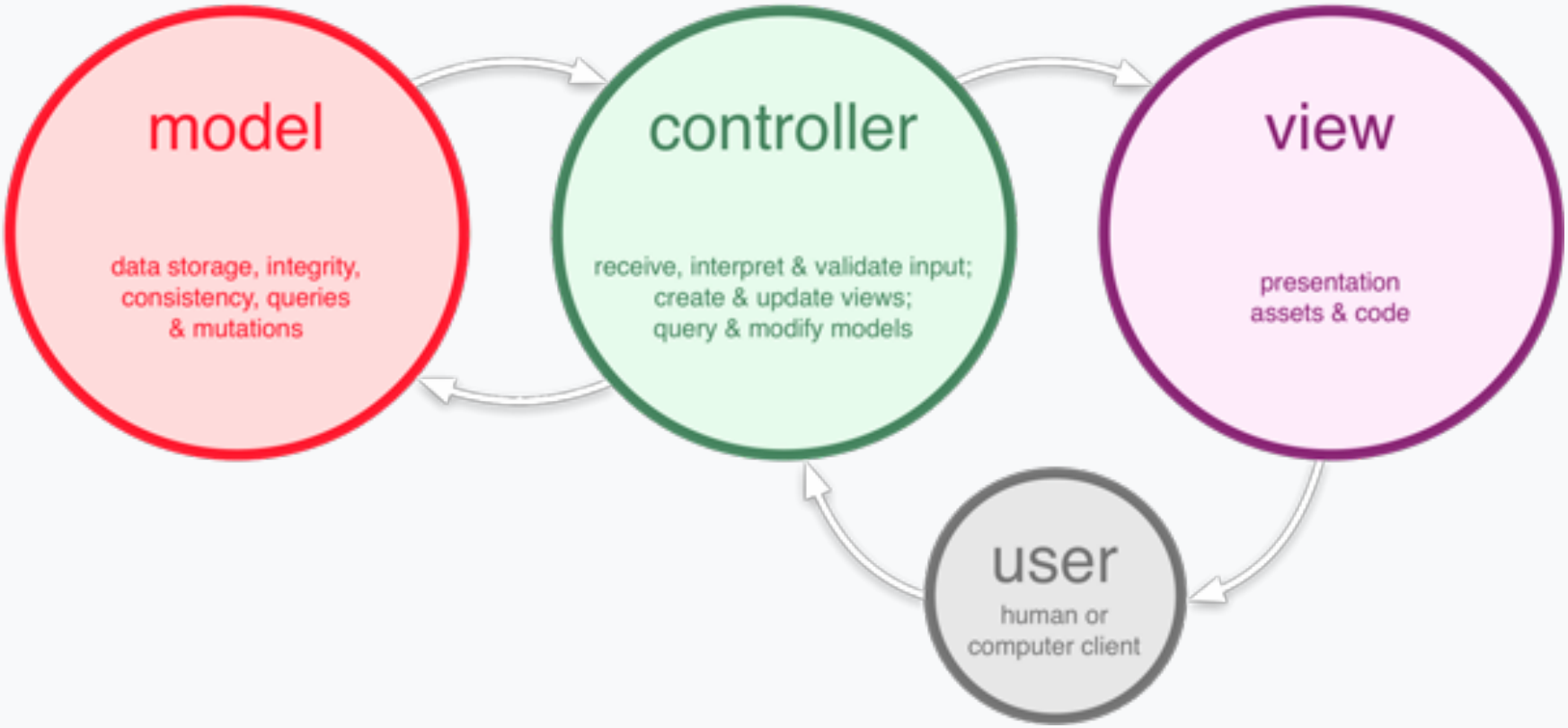
- The model (chef) handles your data
- It interacts with a DB or data store of some kind, crunches the numbers and does the 'heavy lifting' in your application.

- The controller (manager) manages communication between model and view
- It parses user input, gets the information it needs from the model, and passes it back to the view.
- It runs everything! Controllers are very... CONTROLLING.

Recurring Problem for M-V-C

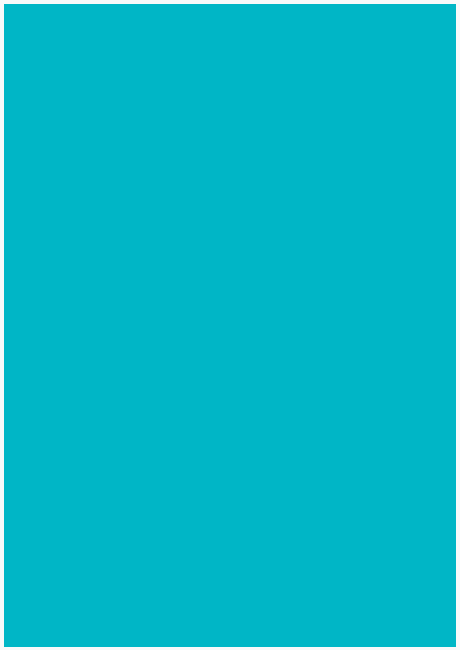
- Data and business logic
- User interface
- Communication between them

Model – View – Controller



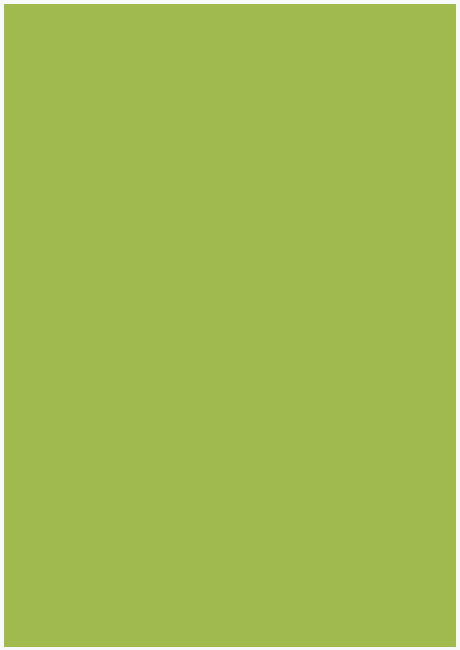
M-V-C Design Pattern

Model



Controller

View



M-V-C Design Pattern

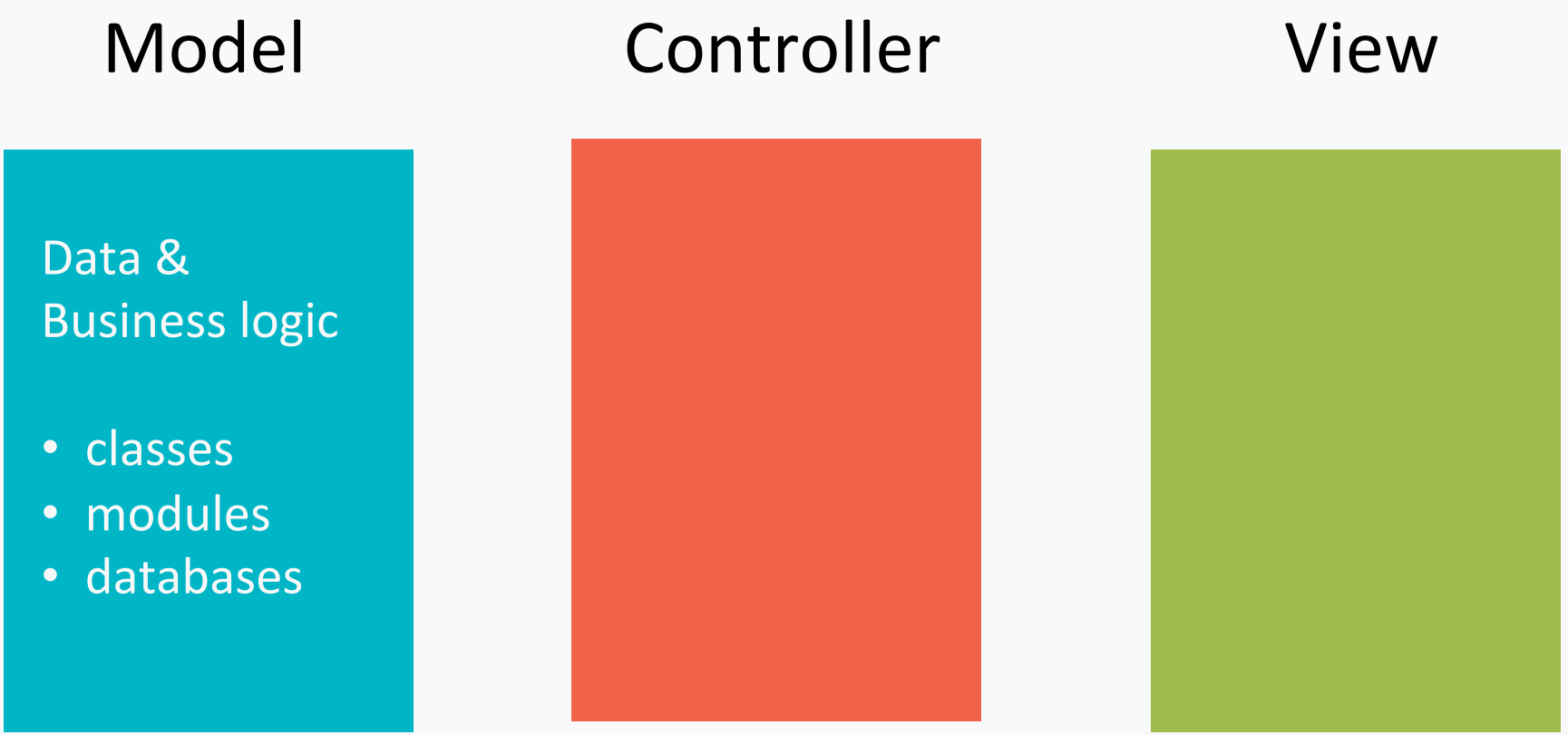
Model

Data &
Business logic

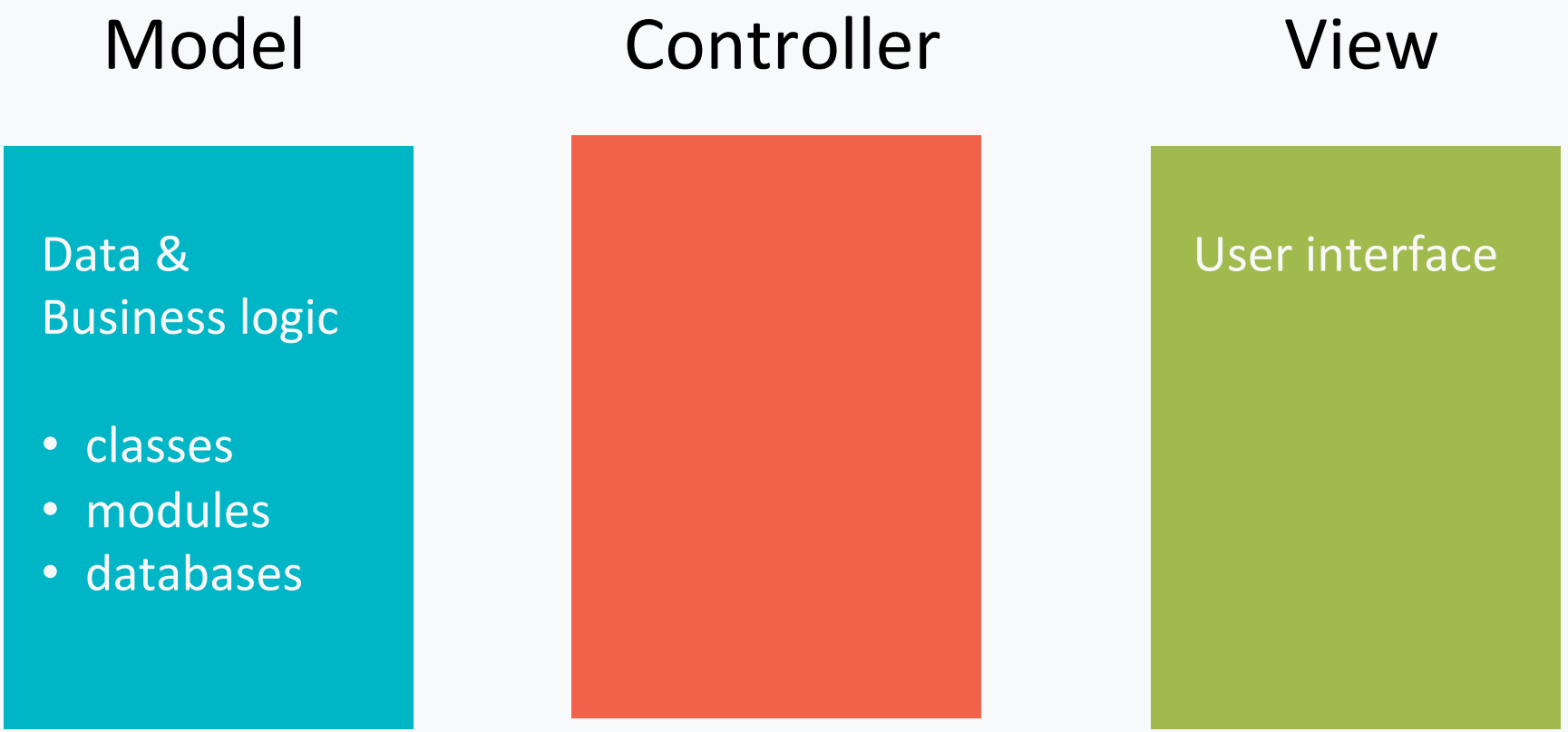
Controller

View

M-V-C Design Pattern



M-V-C Design Pattern



M-V-C Design Pattern

Model

Data &
Business logic

- classes
- modules
- databases

Controller

View

User interface

- get input
- display data from model

M-V-C Design Pattern

Model

Data &
Business logic

- classes
- modules
- databases

Controller

Communicate

View

User interface

- get input
- display data from model

M-V-C Design Pattern

Model

Data &
Business logic

- classes
- modules
- databases

Controller

Communicate

- handle input
- update model
- send data to view

View

User interface

- get input
- display data from model

M-V-C TODO's

- How would TODO's look following the M-V-C pattern?

M-V-C TODO's

Model	Controller	View
Data & Business logic	Communicate	User interface

M-V-C TODO's

Model

Data &
Business logic

- List
- Task
- CSVParsing

Controller

Communicate

View

User interface

M-V-C TODO's

Model

Data &
Business logic

- List
- Task
- CSVParsing

Controller

Communicate

- TODO-Controller

View

User interface

M-V-C TODO's

Model

Data &
Business logic

- List
- Task
- CSVParsing

Controller

Communicate

- TODO-Controller
- runner.rb

View

User interface

- Views
  - Options
  - Task List
  - Task View

Message Pattern

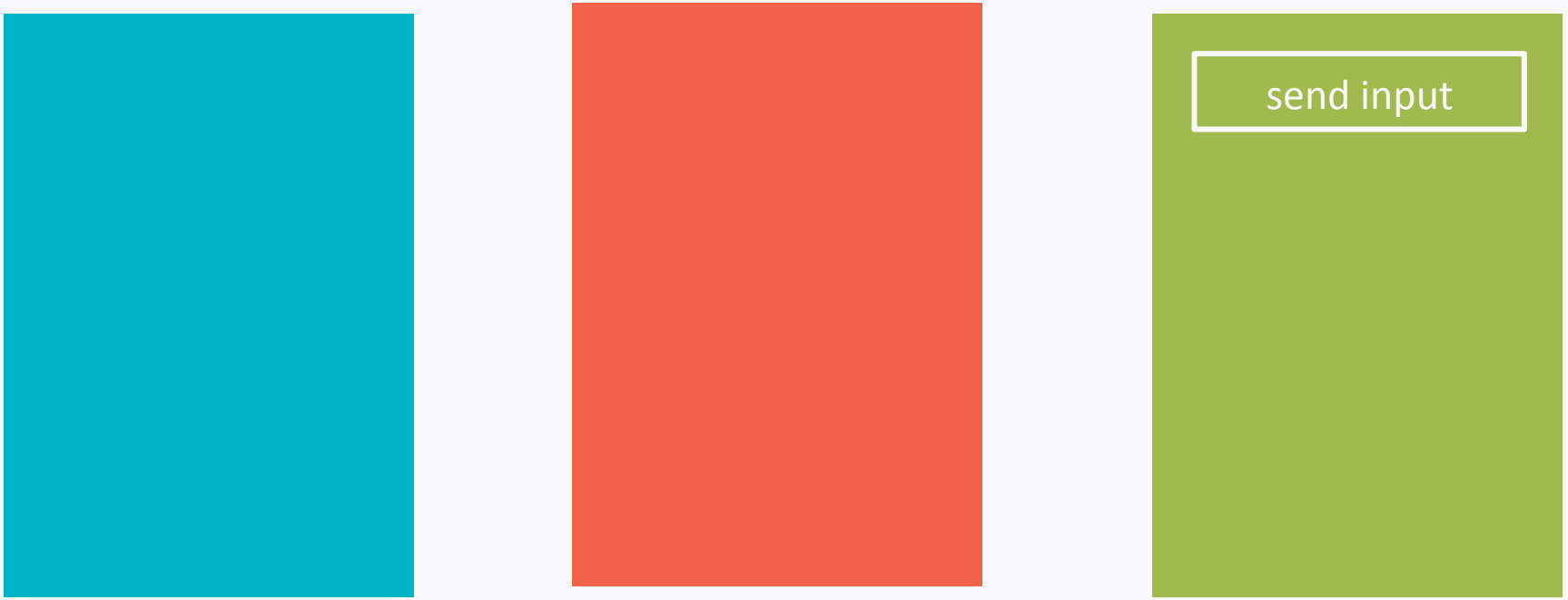
- All of programming is essentially passing messages between all the things.
- Start thinking of your methods/classes as a means of passing a message to a place.
- That place does a thing and passes some message back.

TODO's: List

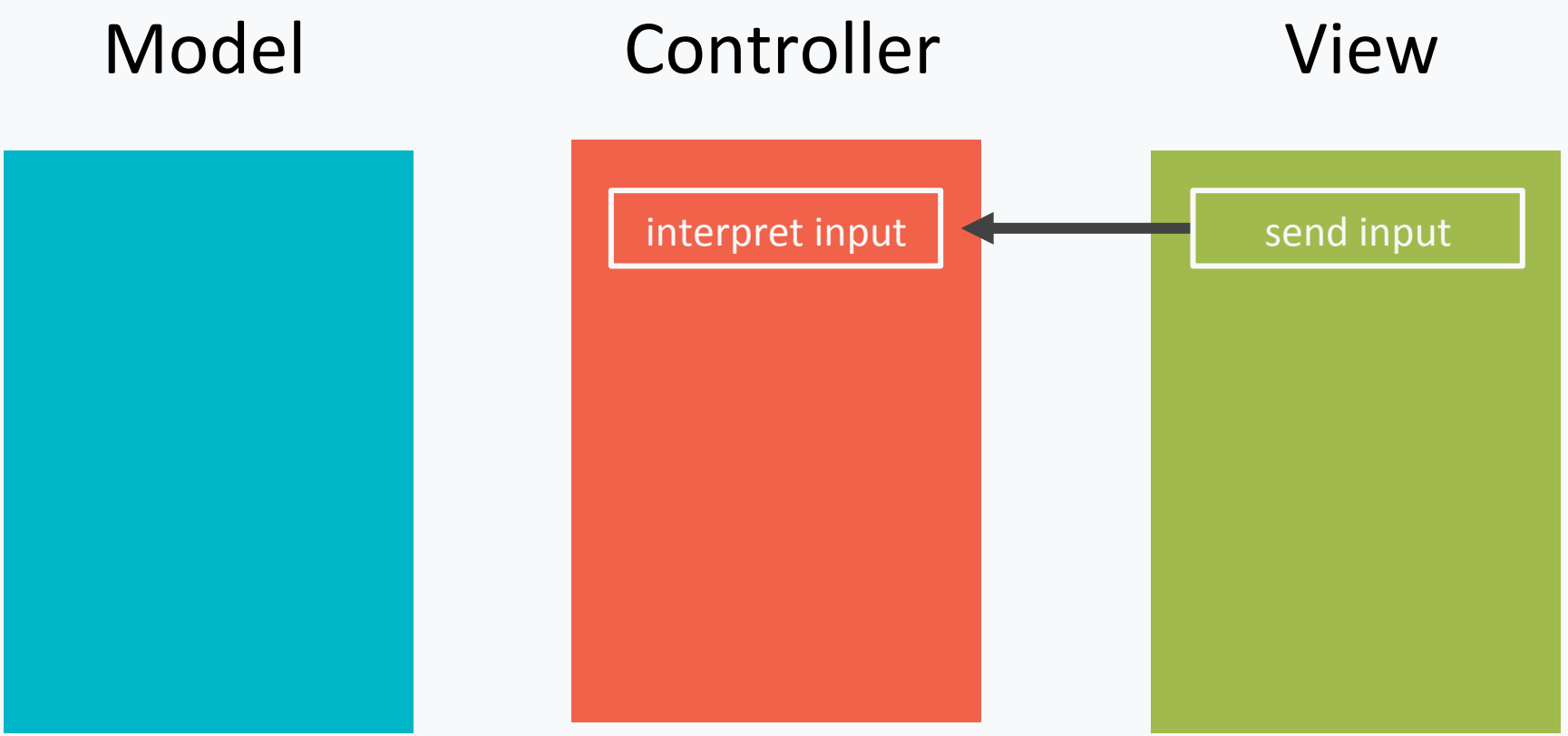
Model

Controller

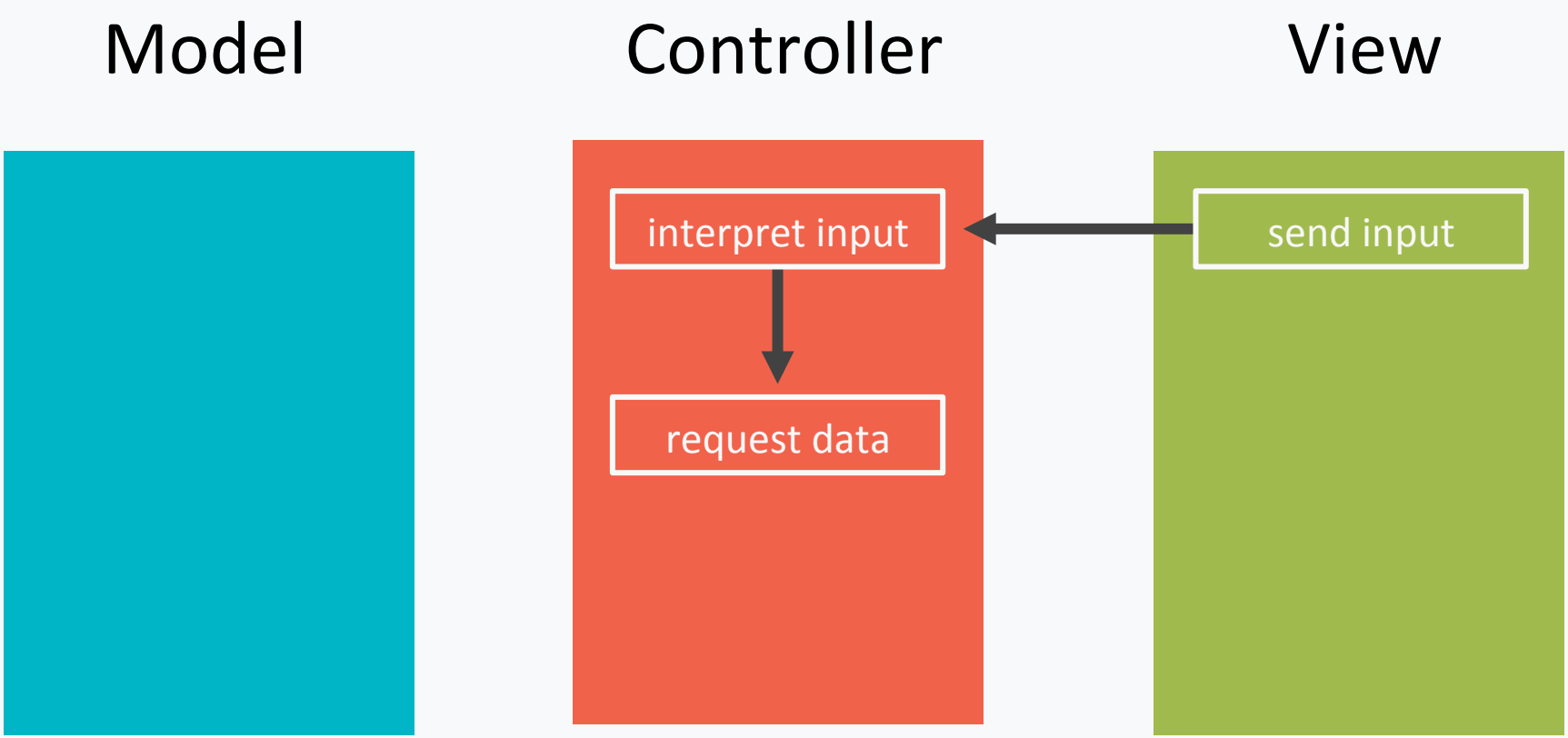
View



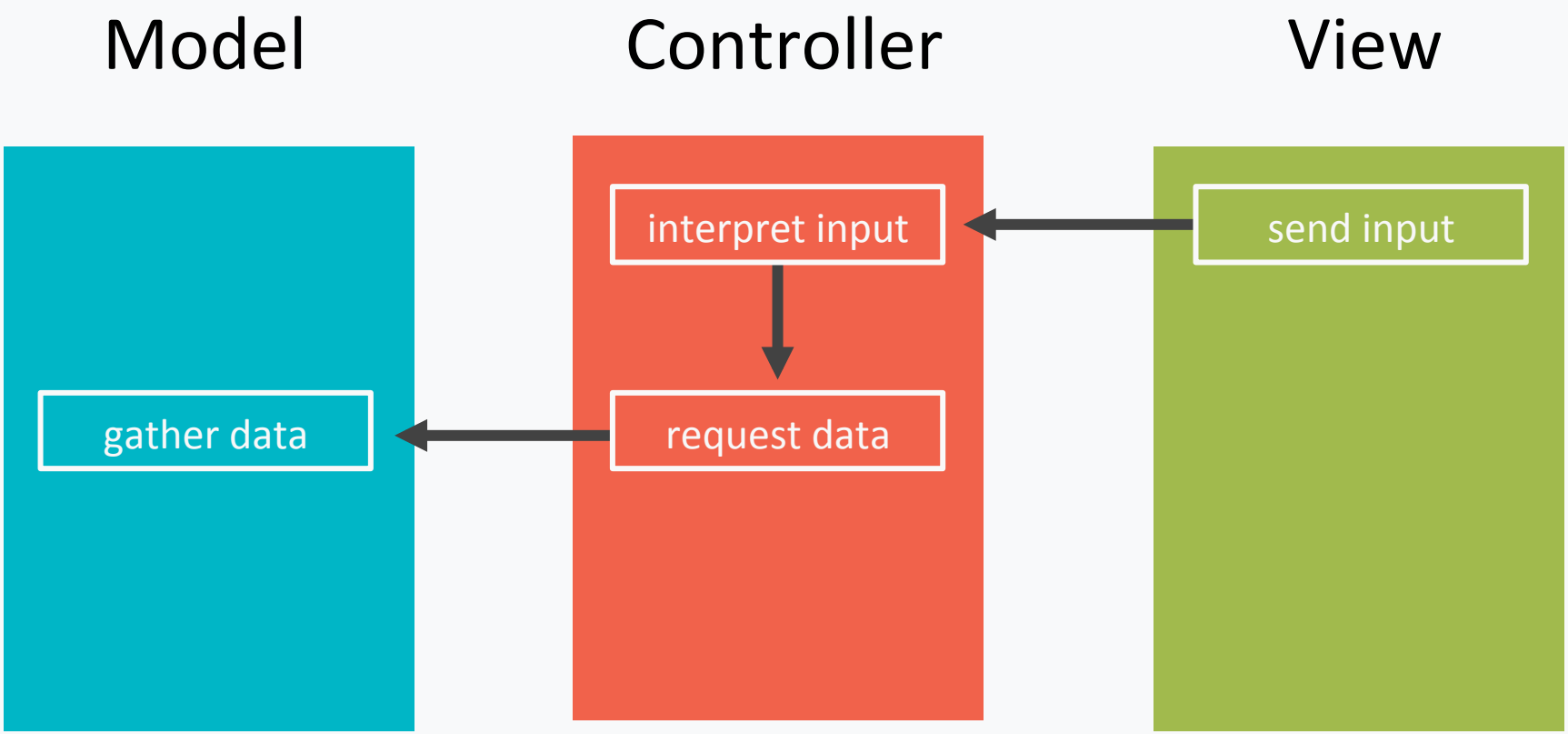
TODO's: List



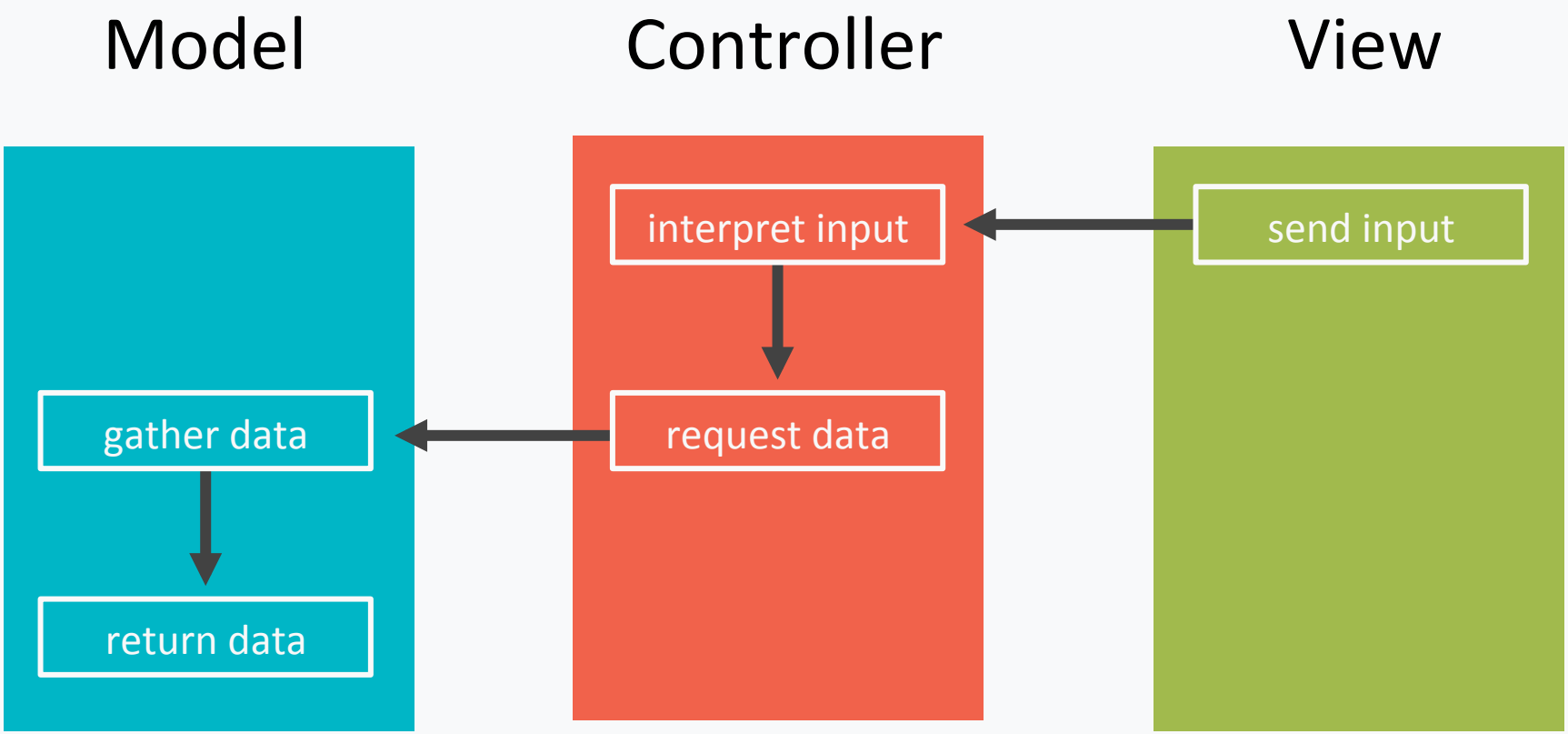
TODO's: List


Model
Controller
View
interpret input
request data
send input

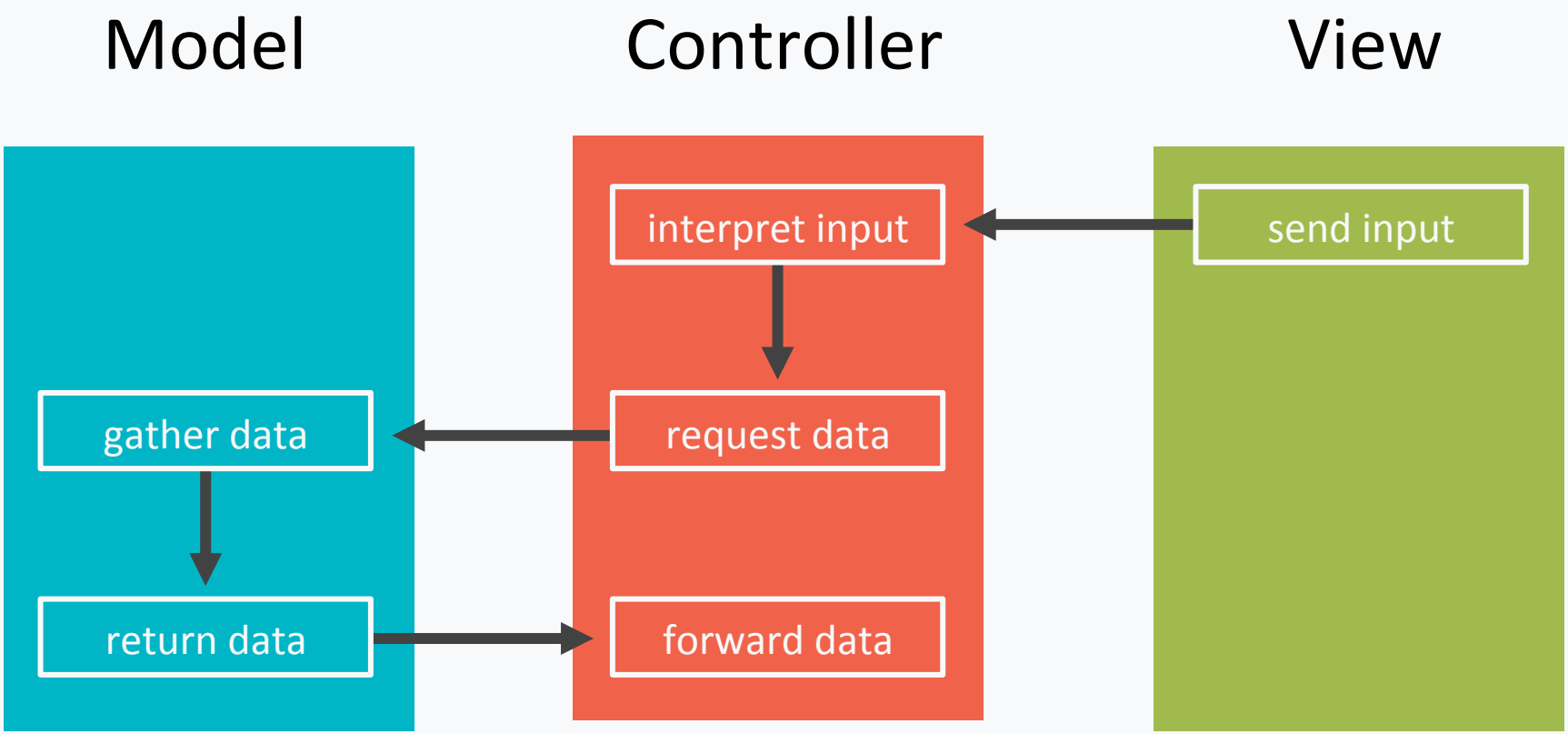
TODO's: List


Model
Controller
View
interpret input
send input
gather data
request data

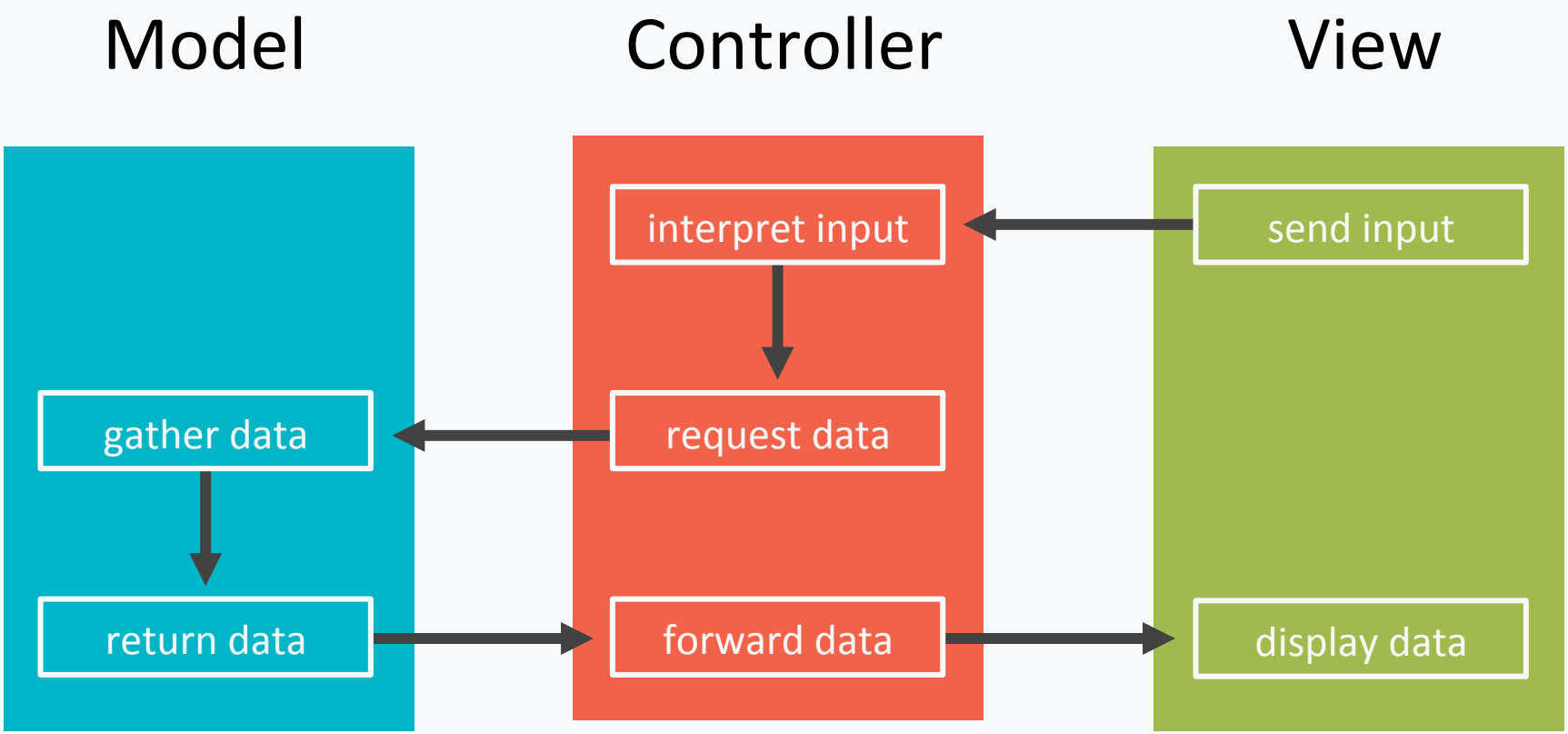
TODO's: List


Model
Controller
View
interpret input
send input
gather data
request data
return data

TODO's: List


Model
Controller
View
interpret input
send input
gather data
request data
return data
forward data

Everything you need to visualize the MVC pattern summed up in 1 slide



FIN: