JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

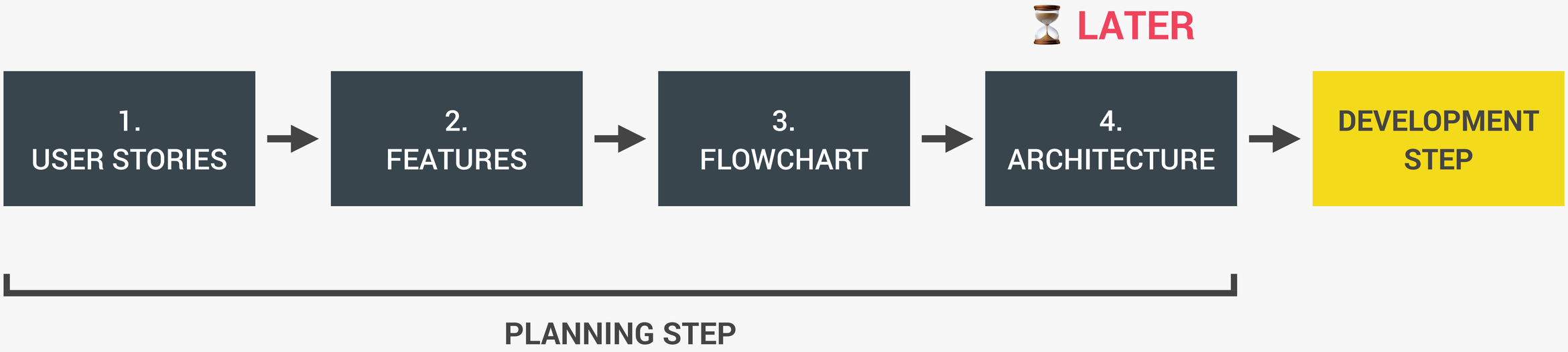
FORKIFY APP: BUILDING A MODERN APPLICATION

LECTURE

PROJECT OVERVIEW AND PLANNING

JS

PROJECT PLANNING



1. USER STORIES

forkify

👉 **User story:** Description of the application's functionality from the user's perspective.

👉 **Common format:** As a *[type of user]*, I want *[an action]* so that *[a benefit]*

1. As a user, I want to **search for recipes**, so that I can find new ideas for meals
2. As a user, I want to be able to **update the number of servings**, so that I can cook a meal for different number of people
3. As a user, I want to **bookmark recipes**, so that I can review them later
4. As a user, I want to be able to **create my own recipes**, so that I have them all organized in the same app
5. As a user, I want to be able to **see my bookmarks and own recipes when I leave the app and come back later**, so that I can close the app safely after cooking

2. FEATURES

forkify

USER STORIES	FEATURES
1 Search for recipes	👉 Search functionality: input field to send request to API with searched keywords 👉 Display results with pagination 👉 Display recipe with cooking time, servings and ingredients
2 Update the number of servings	👉 Change servings functionality: update all ingredients according to current number of servings
3 Bookmark recipes	👉 Bookmarking functionality: display list of all bookmarked recipes
4 Create my own recipes	👉 User can upload own recipes 👉 User recipes will automatically be bookmarked 👉 User can only see their own recipes, not recipes from other users
5 See my bookmarks and own recipes when I leave the app and come back later	👉 Store bookmark data in the browser using local storage 👉 On page load, read saved bookmarks from local storage and display

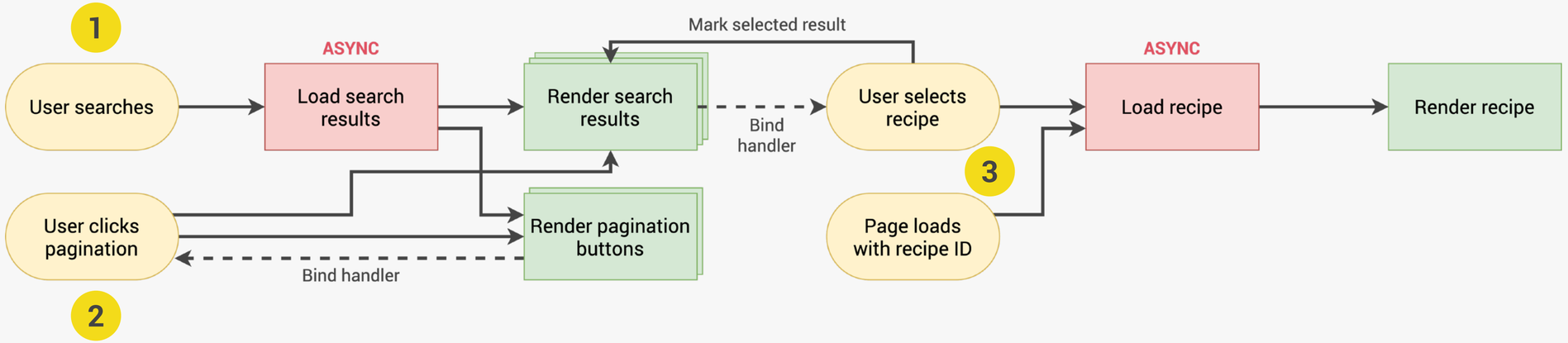
3. FLOWCHART (PART 1)

forkify

FEATURES

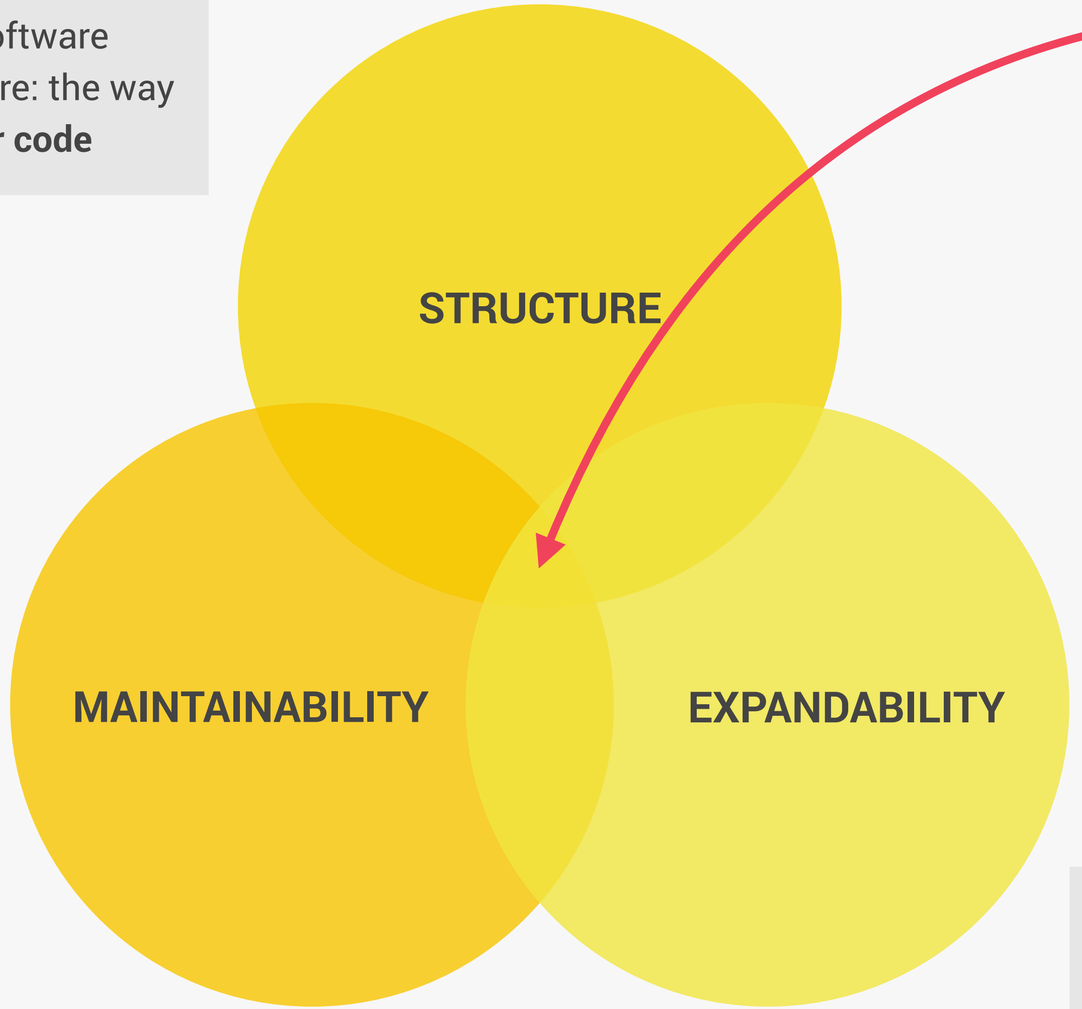
1. Search functionality: API search request
2. Results with pagination
3. Display recipe

Other features later



WHY WORRY ABOUT ARCHITECTURE?

👉 Like a house, software needs a structure: the way we **organize our code**



The perfect architecture

👉 We can create our own architecture (Mapty project)

👉 We can use a well-established architecture pattern like MVC, MVP, Flux, etc. (**this project**)

👉 We can use a framework like React, Angular, Vue, Svelte, etc.

👉 A project is never done! We need to be able to easily **change it in the future**

👉 We also need to be able to easily **add new features**

COMPONENTS OF ANY ARCHITECTURE

BUSINESS LOGIC

- 👉 Code that **solves the actual business problem**;
- 👉 Directly related to what business does and what it needs;
- 👉 **Example:** sending messages, storing transactions, calculating taxes, …

STATE

- 👉 Essentially **stores all the data** about the application
- 👉 Should be the “single source of truth”
- 👉 UI should be kept in sync with the state
- 👉 State libraries exist


HTTP LIBRARY

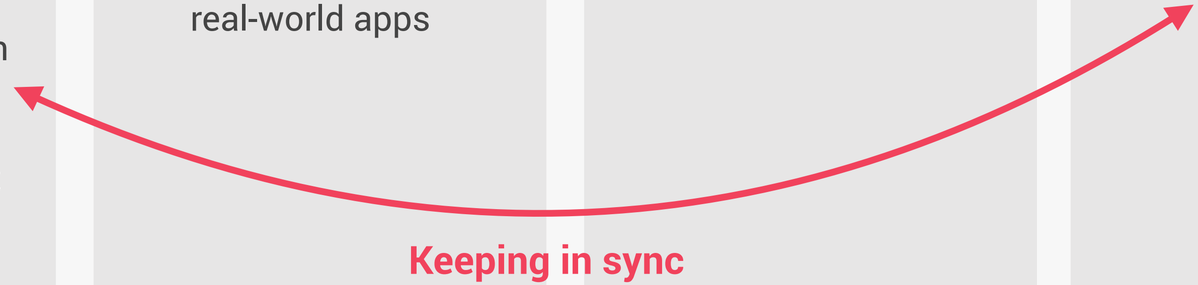
- 👉 Responsible for making and receiving AJAX requests
- 👉 Optional but almost always necessary in real-world apps

APPLICATION LOGIC (ROUTER)

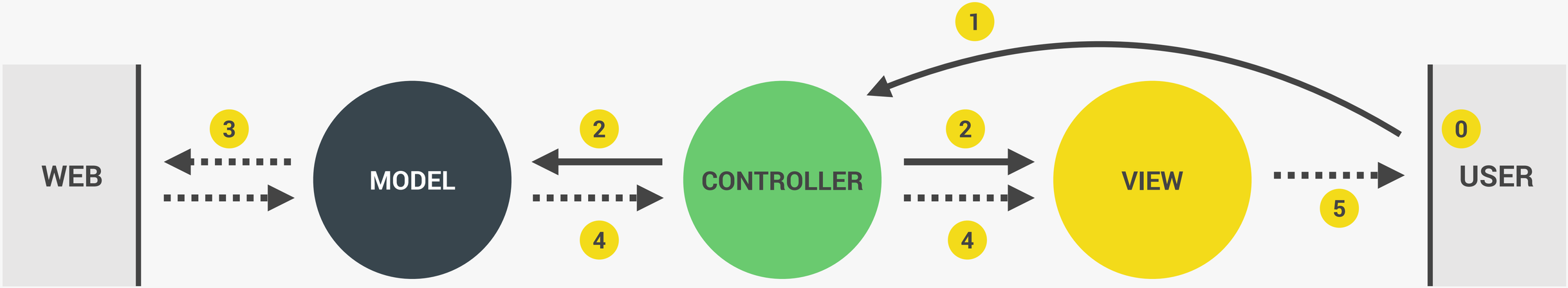
- 👉 Code that is only concerned about the **implementation of application itself**;
- 👉 Handles navigation and UI events

PRESENTATION LOGIC (UI LAYER)

- 👉 Code that is concerned about the **visible part** of the application
- 👉 Essentially displays application state



THE MODEL-VIEW-CONTROLLER (MVC) ARCHITECTURE



BUSINESS LOGIC

STATE

HTTP LIBRARY

APPLICATION LOGIC

- 👉 Bridge between model and views (which don't know about one another)
- 👉 Handles UI events and **dispatches tasks to model and view**

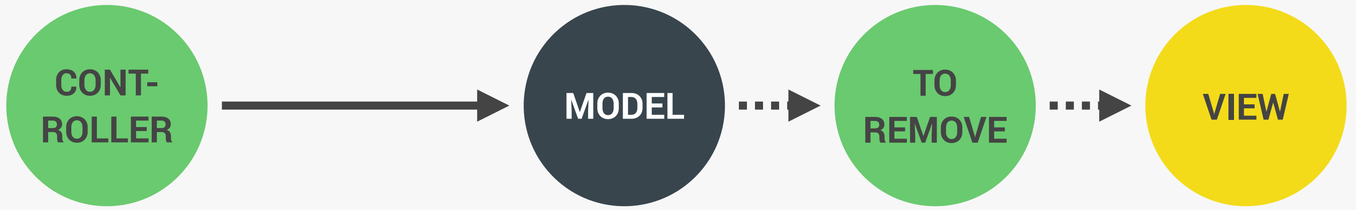
PRESENTATION LOGIC

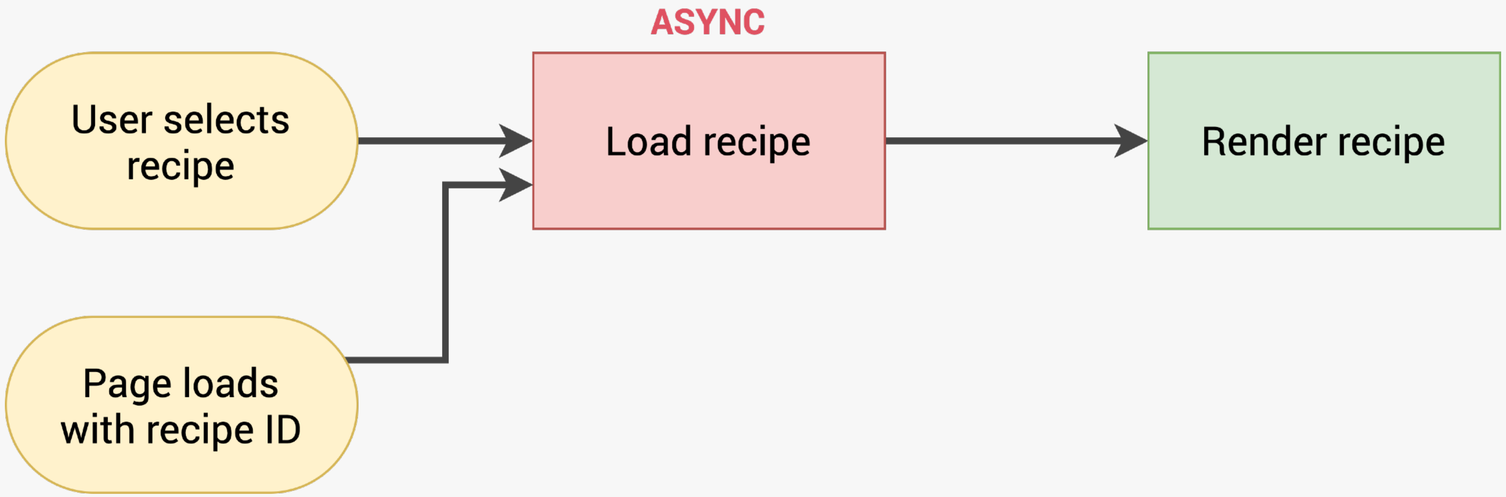
→ Connected by function call and import

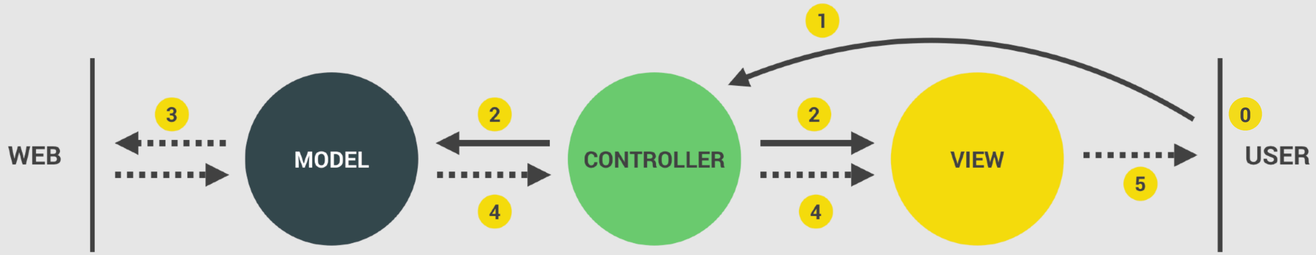
⇢ Data flow

MODEL, VIEW AND CONTROLLER IN FORKIFY (RECIPE DISPLAY ONLY)

forkify

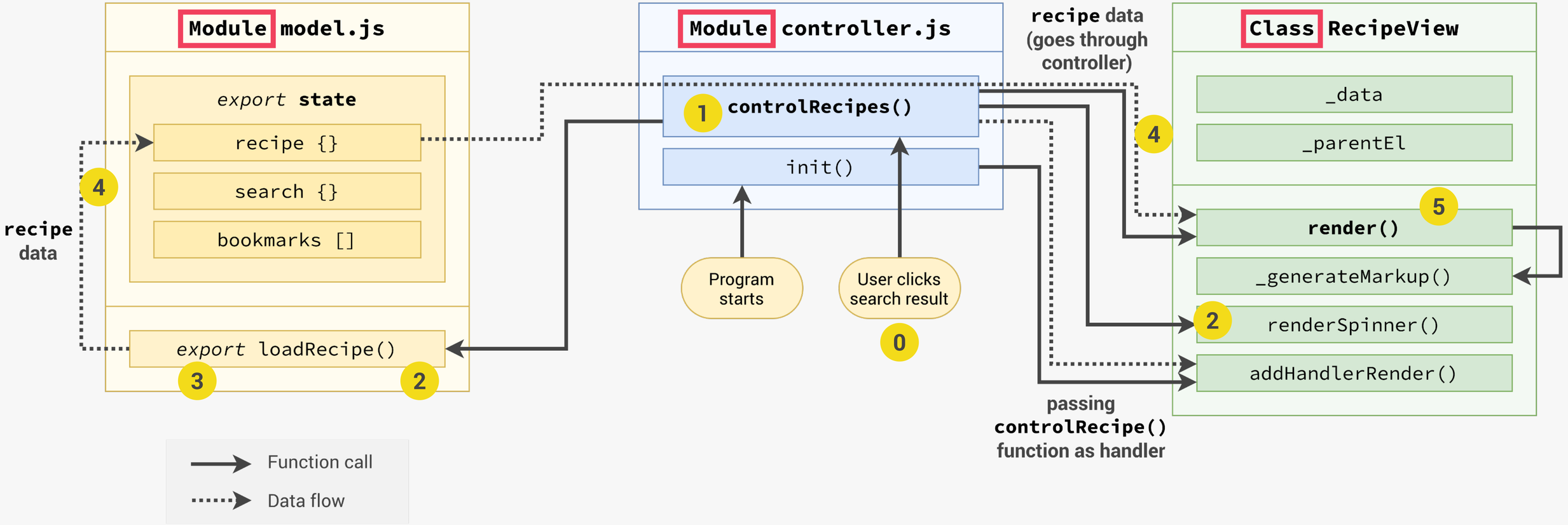


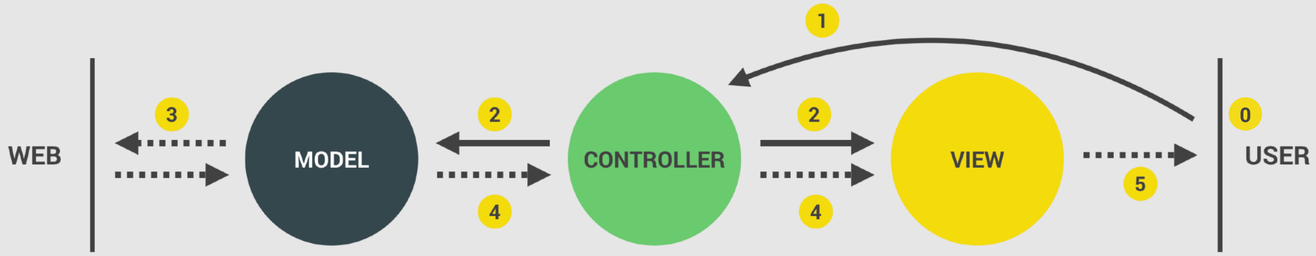




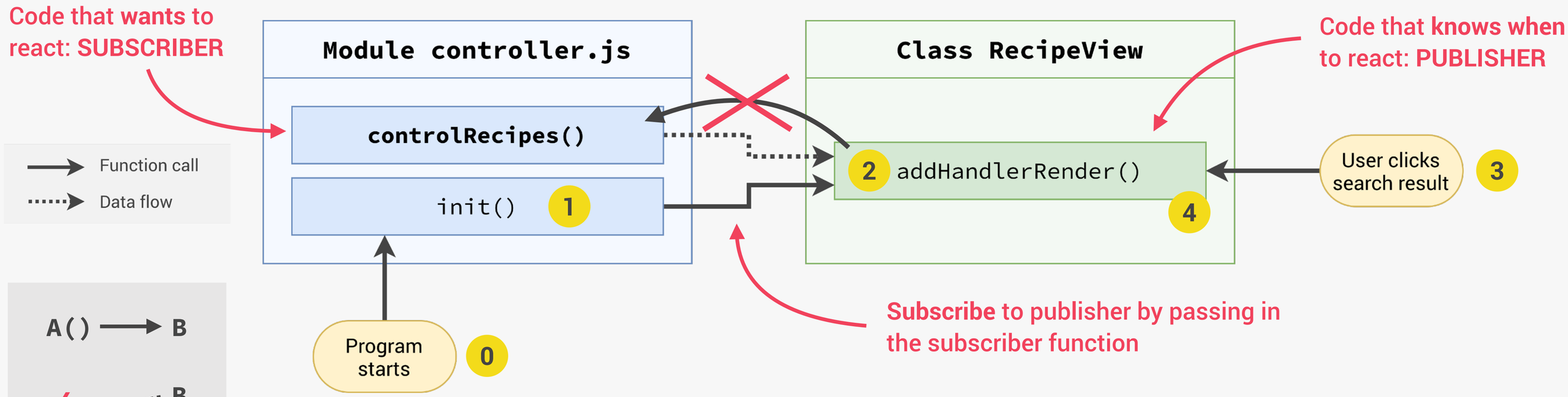
MVC IMPLEMENTATION (RECIPE DISPLAY ONLY)

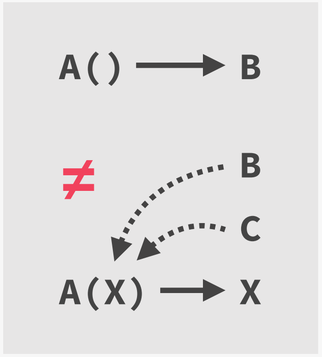
forkify





EVENT HANDLING IN MVC: PUBLISHER-SUBSCRIBER PATTERN





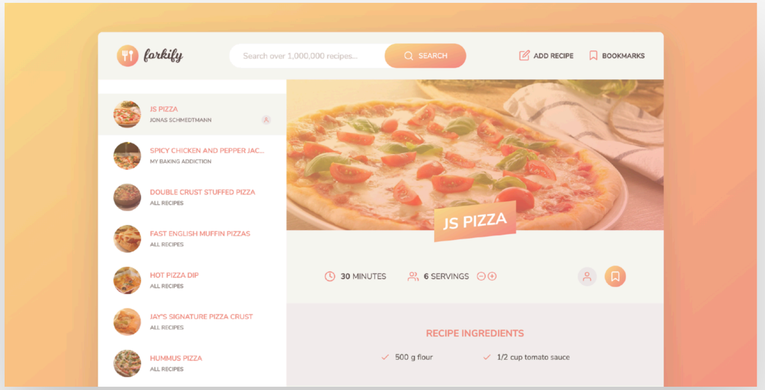
- 👉 `controlRecipes` will be passed into `addHandlerRender` when program starts
- 👉 `addHandlerRender` listens for events (`addEventListener`), and uses `controlRecipes` as callback

- 👉 Events should be **handled** in the **controller** (otherwise we would have application logic in the view)
- 👉 Events should be **listened for** in the **view** (otherwise we would need DOM elements in the controller)

IMPROVEMENT AND FEATURE IDEAS: CHALLENGES 🤓

forkify

- 👉 Display **number of pages** between the pagination buttons;
- 👉 Ability to **sort** search results by duration or number of ingredients;
- 👉 Perform **ingredient validation** in view, before submitting the form;
- 👉 **Improve recipe ingredient input**: separate in multiple fields and allow more than 6 ingredients;



- 👉 **Shopping list feature**: button on recipe to add ingredients to a list;
- 👉 **Weekly meal planning feature**: assign recipes to the next 7 days and show on a weekly calendar;
- 👉 **Get nutrition data** on each ingredient from spoonacular API (https://spoonacular.com/food-api) and calculate total calories of recipe.