THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

SECTION

MODERN JAVASCRIPT DEVELOPMENT: MODULES AND TOOLING

LECTURE

DECLARATIVE AND FUNCTIONAL JAVASCRIPT PRINCIPLES

IMPERATIVE VS. DECLARATIVE CODE

Two fundamentally different ways of writing code (paradigms)

IMPERATIVE

- 👉 Programmer explains **“HOW to do things”**
- 👉 We explain the computer *every single step* it has to follow to achieve a result
- 👉 **Example:** Step-by-step recipe of a cake

```
const arr = [2, 4, 6, 8];
const doubled = [];
for (let i = 0; i < arr.length; i++)
  doubled[i] = arr[i] * 2;
```

DECLARATIVE

- 👉 Programmer tells **“WHAT do do”**
- 👉 We simply *describe* the way the computer should achieve the result
- 👉 The **HOW** (step-by-step instructions) gets abstracted away
- 👉 **Example:** Description of a cake

```
const arr = [2, 4, 6, 8];
const doubled = arr.map(n ⟹ n * 2);
```

FUNCTIONAL PROGRAMMING PRINCIPLES

FUNCTIONAL PROGRAMMING

- **Declarative** programming paradigm
- Based on the idea of writing software by combining many **pure functions**, avoiding **side effects** and **mutating** data

- **Side effect:** Modification (mutation) of any data **outside** of the function (mutating external variables, logging to console, writing to DOM, etc.)
- **Pure function:** Function without side effects. Does not depend on external variables. **Given the same inputs, always returns the same outputs**.
- **Immutability:** State (data) is **never** modified! Instead, state is **copied** and the copy is mutated and returned.

- **Examples:**   Redux

FUNCTIONAL PROGRAMMING TECHNIQUES

- Try to avoid data mutations
- Use built-in methods that don't produce side effects
- Do data transformations with methods such as `.map()`, `.filter()` and `.reduce()`
- Try to avoid side effects in functions: this is of course not always possible!

DECLARATIVE SYNTAX

- Use array and object destructuring
- Use the spread operator (`...`)
- Use the ternary (conditional) operator
- Use template literals

FORKIFY APP: BUILDING A MODERN APPLICATION

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

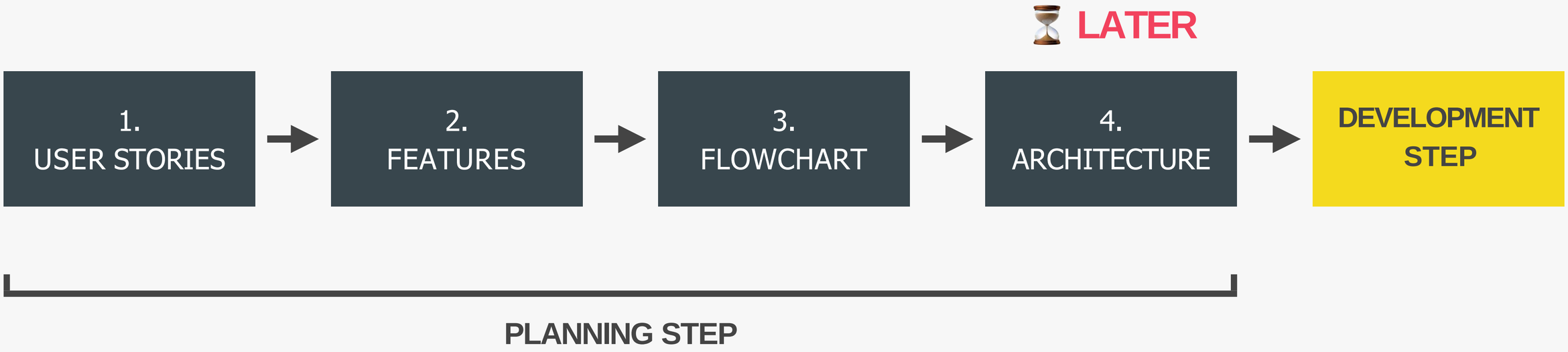
SECTION

FORKIFY APP: BUILDING A MODERN APPLICATION

LECTURE

PROJECT OVERVIEW AND PLANNING

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