iteratec

KOMPETENZ, DIE ENTLASTET

Wrocław

miasto spotkań

PROUDLY PRESENTS

ANGULAR WITH NGRX

+ INTRODUCTION TO REDUX ARCHITECTURE

Paweł Kamiński

LET'S GET KNOW EACH OTHER

- Angular / Web experience
- Why I'm here

AGENDA

Learn

- What is application state?
- What is Redux?
- How to write code with Redux?

Build

- Live coding

Further learning

Feedback

WHAT IS AN APPLICATION STATE?

All the data that an application needs at given moment:

- To display UI of given structure and content (User Interactions and Input)
- To perform some actions (Authentication Tokens)
- To provide behaviour persistency (Server Responses)

WHAT IS REDUX?

- Simple, reliable and scalable application architecture (e.g. Facebook)
- Small set of building blocks used to manage application state
- A way of thinking about where to put business logic

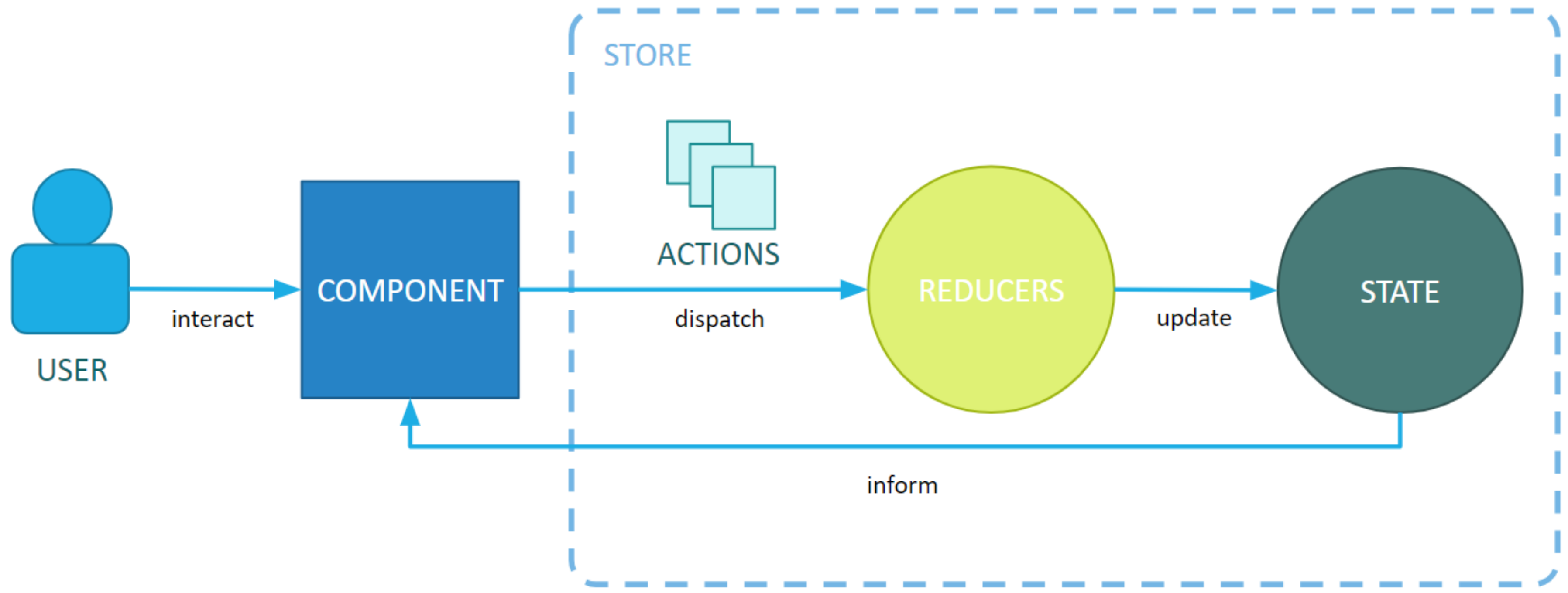
THREE PRINCIPLES OF REDUX

- Single Source of Truth
- State is read only
- Changes are made with pure functions (i.e. reducers)

GLOSSARY

- Store - objects which stores our state, and provides mechanism to access it
- Reducers - functions to change the state
- Action - 'event' to trigger change of the state

GENERAL REDUX SCHEMA



SIMPLE EXAMPLE

Codepen

LIVE CODING SESSION

STEP 1

- Write your first reducer
  hint: (State, Action) => State
- Add initial, default state to your reducer
- Wire up the reducer with the module using *StoreModule.forRoot method* and *ActionReducerMap*

STEP 2

Use the store to render tasks in :
TaskBoardComponent

- Stop using the service in *getData*
- Fetch all tasks from the store using *select* operator,
- *Temporarily* subscribe to the store and assign them to the existing *tasks* field
- ...

STEP 2

- ...
- Map the tasks collection to a filtered one, based on *this.mode (if set)*
- Write a unit test for your reducer

Since we're not relying on internal component state anymore, adding tasks won't work for a while.

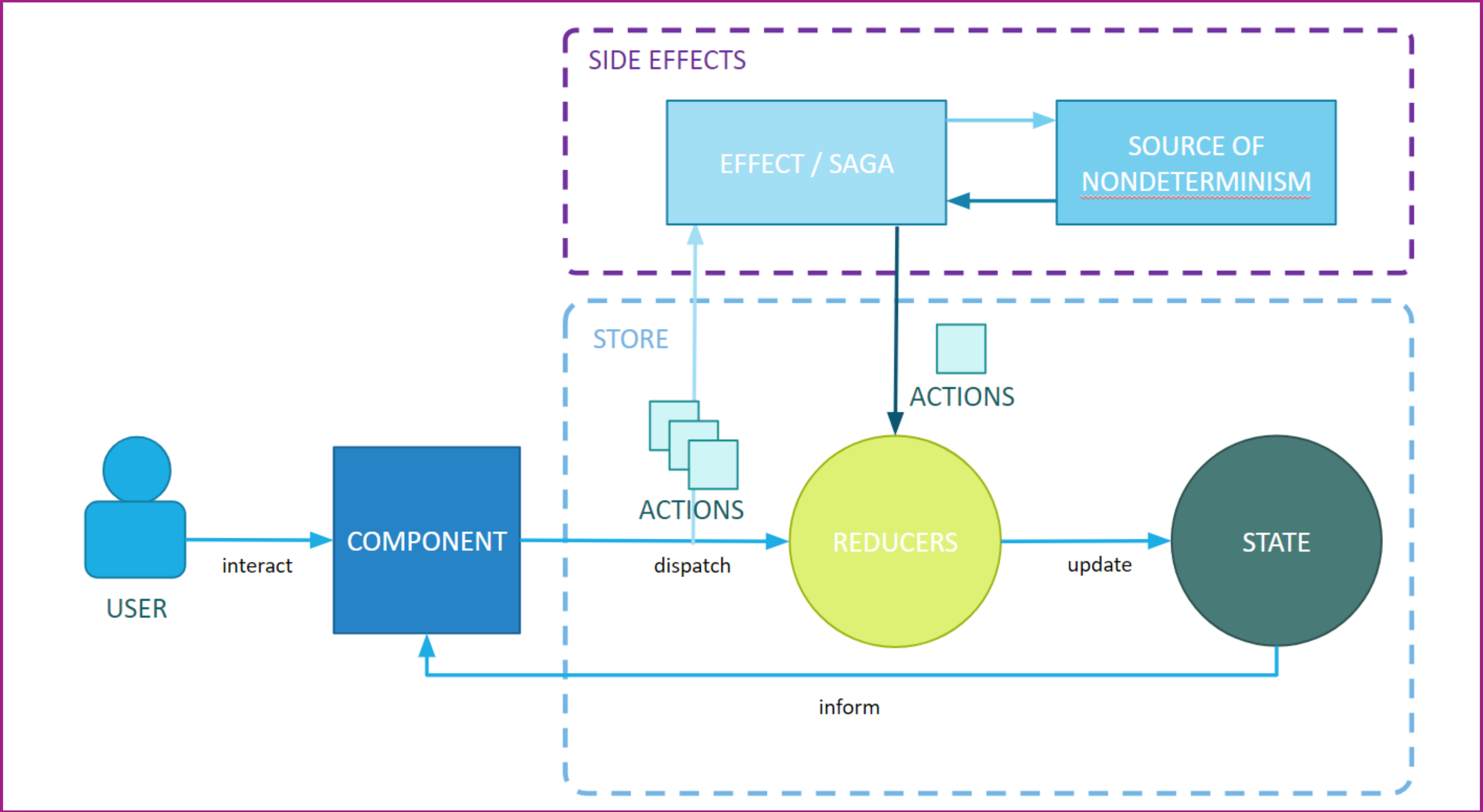
STEP 3

- Refactor *TaskBoardComponent* to use *async* instead of subscribing to *tasks$*
- Extract tasks selection logic to *store/selectors.ts* use *createFeatureSelector* and *createSelector* functions,

STEP 4

- Implement *UpdateTask* action that will be dispatched when e.g. moving a task
- Handle updating a task in the reducer, use task *id* field to find proper one
- Test the reducer
- Dispatch the update action on task move *TaskBoardComponent*

REDUX SCHEMA WITH EFFECTS



STEP 5

- Add *TasksService.fetchTasks* method that will return task with some delay, to emulate the HTTP request behaviour (use *timer* from rxjs)
- Be sure that *TasksService* returns other tasks than in the default state
- Add *LoadTask* and *TasksLoaded* actions, handle them in the reducer (add *loading: boolean* state)

STEP 5

- Write an effect that reloads tasks on *LoadTask* action and emits *TasksLoaded* action
- Register the effect with *EffectsModule.forRoot*
- Add a temporarily fetch button to *TaskBoardComponent* that will dispatch *LoadTask* action

STEP 6

- Implement adding a new task:
  - Create *AddTask* action
  - Handle the action in the reducer
  - Dispatch the action in *TaskBoardComponent* in appropriate handler

STEP 7

- Use loading value from state to show/hide spinner when loading data

STEP 8

- Extend *NgrxModuleState* to have router key of type *RouterReducerState* - the *RouterStateUrl* was implemented for you in *store/router-store.ts*
- Import and register *routerReducer* in your present *ActionReducerMap* (suggested key: `router`)
- Wire up *StoreRouterConnectingModule.forRoot({ serializer: CustomSerializer }),* with your module

STEP 9

- Use router *ROUTER_NAVIGATION* to start fetching the data
- Write selector to extract mode from url

STEP 10

- Adjust *TaskBoardComponent* component to use just observables
- Remove fetch button from step 5

STEP 11

- Adjust selectors
- Fix issue with too often calling fetch data
- Since *ROUTER_NAVIGATION* is used to load data clean initial state in *task-reducer* file

WHAT WE HAVE MISSED

- state normalization
- tooling
- testing of effects and selectors
- alternatives for NgRx

LEARN MORE

- redux.js.org
- NgRx Home Page
- NgRx State Management - ultimateangular.com
- Getting Started with Redux
- Todd Motto - Understand NgRx by building a Store - Keynote

FEEDBACK

[illegible]

Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, custodian of the stolen plans that can save her people and restore freedom to the galaxy....

THE END