Workshops 🛠️

with Kacper Kula

Developer Evangelist @ SwingDev

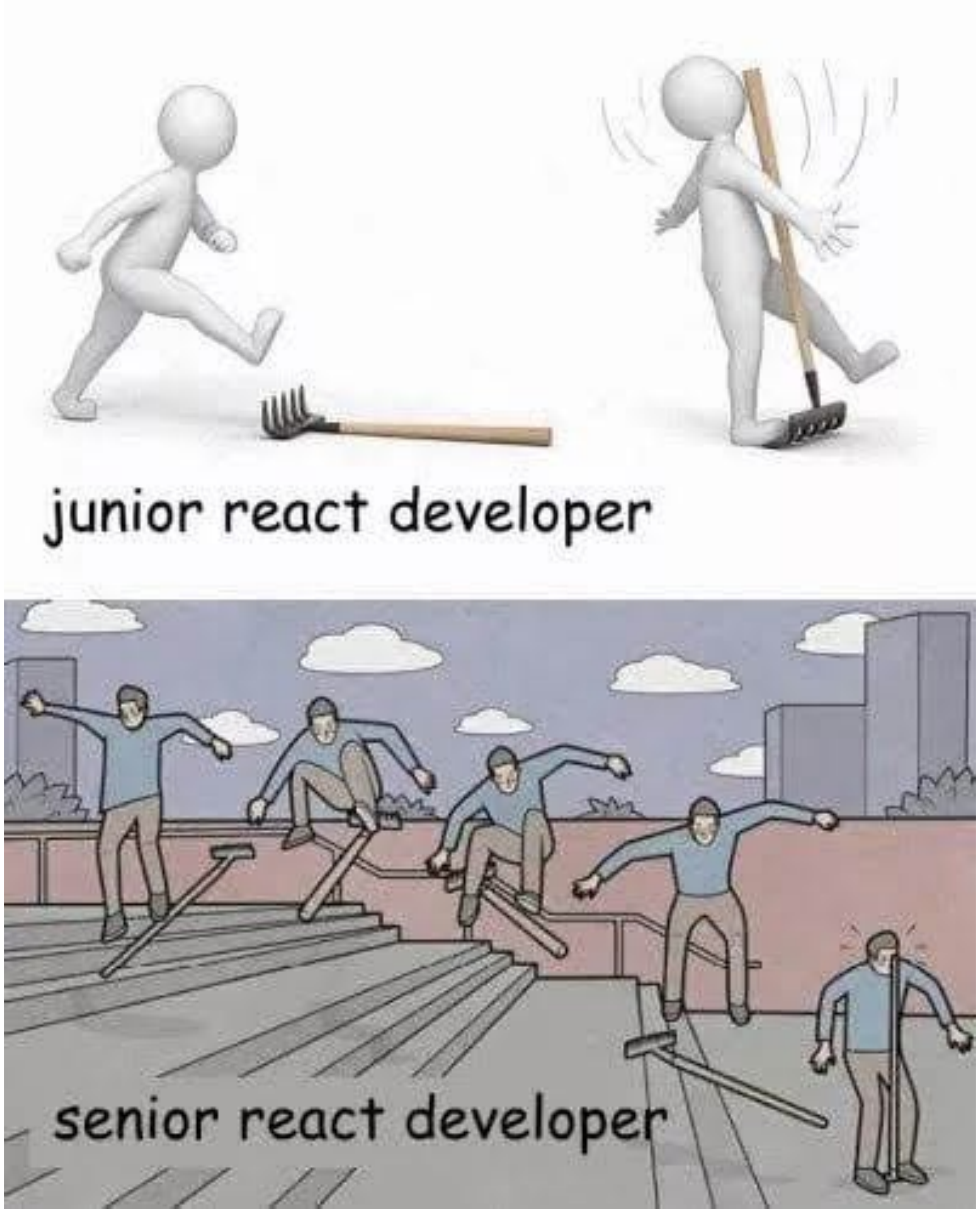
Repository

http://go.swingdev.io/workshops

WIFI

☁️ SwingDev Guest 2

🔒 be.curious


junior react developer
senior react developer

Why is that?

TypeScript
ReasonML
JavaScript
Kotlin
CoffeeScript

React
Aurelia
Vue.js
Angular

Redux
Apollo
MobX
Relay
Jumpsuit

Saga
Dva
Kea
Thunk

But that’s not enough

Reactive Programming

Currying

Promises

Caching

OOP

Exponential Backoff

Styling

Error handling

DAO

DTO

Offline Support

Store Normalisation

TypeScript

ReasonML

JavaScript

Kotlin

CoffeeScript

React

Aurelia

Vue.js

Angular

Redux

Apollo

MobX

Relay

Jumpsuit

Saga

Dva

Kea

Thunk

What we're building?

- Google's Keep clone (notes app)

Features

- Ability to list notes
- Ability to search notes

Repository

http://go.swingdev.io/workshops

WIFI

☁️ SwingDev Guest 2

🔒 be.curious

What we’re building?

- UI Already Created
- Components already there
- We need to create out store and reducers

Sprints

Sprint 1

Load Data

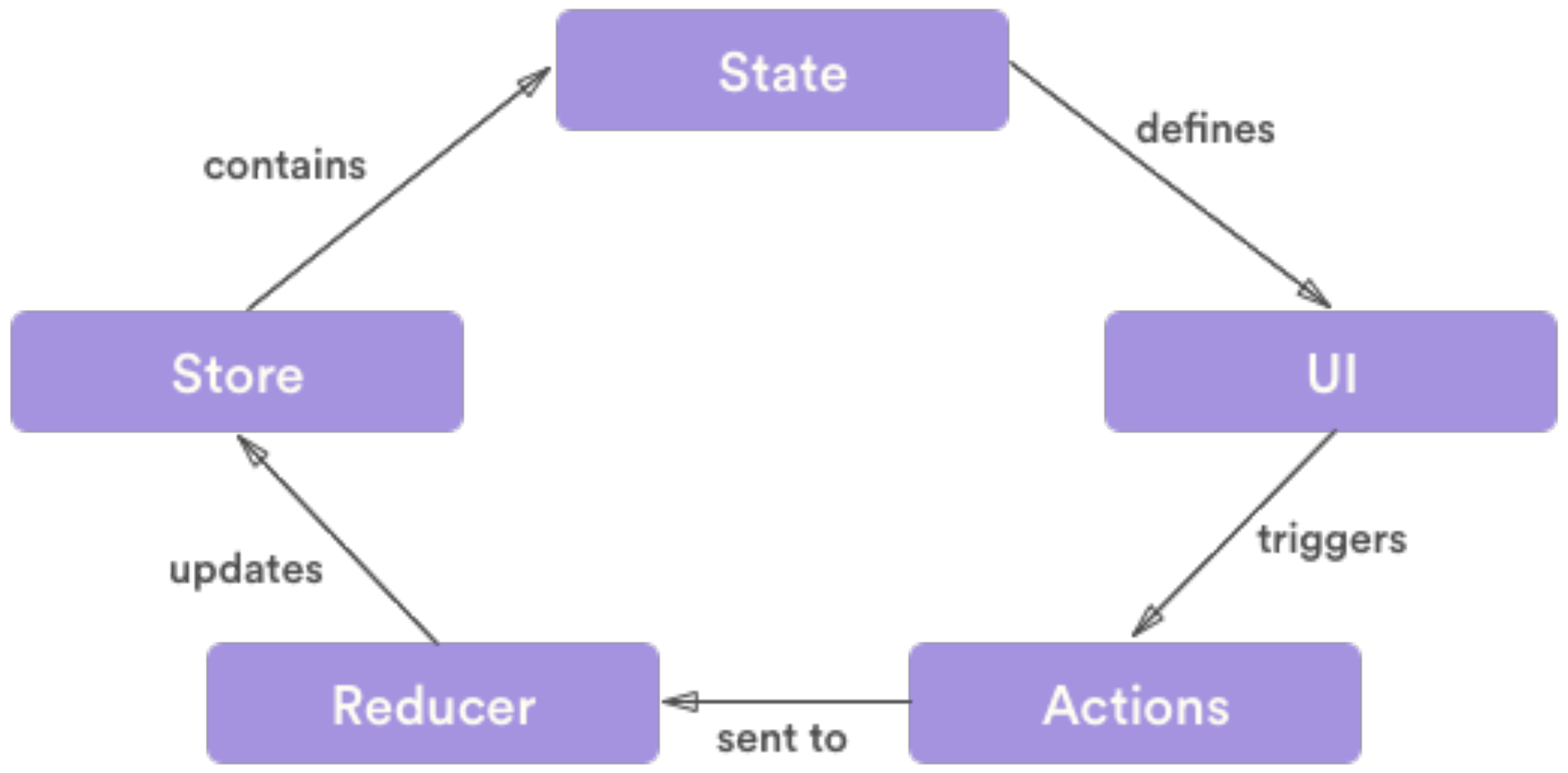
Sprint 2

Add Search

Few decisions

- Simplified Architecture
- Built with best practices but some shortcuts were needed
- No additional libraries used for store simplification or managing actions
- Backend mocked in service

Redux Architecture Quick Overview



TypeScript + React = ❤️

- **Typing**
  - React
    - Props / State well described
  - Redux
    - Clear definition for actions
    - Clear definition of the store structure

TS + React - Example

```
function searchAction(options) {
  return {
    type: ‘SEARCH’,
    ... options
  };
}
```

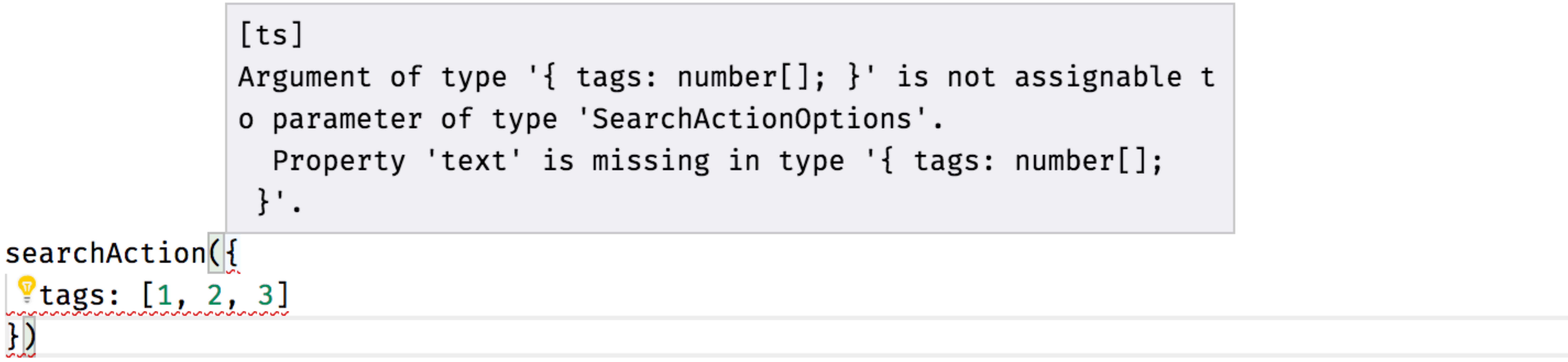
What options can we pass to this function?

```
interface SearchActionOptions {
  text: string,
  tags?: number[]
}

interface ISearchAction extends SearchActionOptions {
  type: string
}

function searchAction(options: SearchActionOptions): ISearchAction {
  return {
    type: 'SEARCH',
    ...options
  };
}
```

Now we see that there are two valid properties, required text and optional tags



```
[ts]
Argument of type '{ tags: number[]; }' is not assignable t
o parameter of type 'SearchActionOptions'.
  Property 'text' is missing in type '{ tags: number[];
 }'.
```

```
searchAction({
  tags: [1, 2, 3]
})
```

Sprint 1

- NotesList component can display notes provided as notes from store
- We have actions and actionCreators implemented
- We need to create a store for notes and add reducers
- We need to update connector in Home view

Ideas

SWING'D

Simple Approach (not recommended)

```
export interface NotesListState {
  state: string, // 'INIT', 'LOADING' | 'LOADED' | 'ERROR',
  notes: NoteModel[],
  errorMessage?: string
}
```

reducers/notesList.ts

Simple Approach (not recommended)

```
export interface IActionNotesFetch extends Action {
  type: 'NOTES_FETCH'
}

export interface IActionNotesFetchSuccess extends Action {
  type: 'NOTES_FETCH_SUCCESS',
  notes: NoteModel[]
}

export interface IActionNotesFetchError extends Action {
  type: 'NOTES_FETCH_ERROR',
  errorMessage: string
}

export type AppActions = IActionNotesFetch | IActionNotesFetchSuccess |
IActionNotesFetchError;
```

actions/index.ts

Simple Approach (not recommended)

```
export function notesListReducer(state: NotesListState, action: AppActions): NotesListState {
  if (action.type === 'NOTES_FETCH') {
    return {
      ...state,
      state: 'LOADING',
      notes: [],
    };
  }
  if (action.type === 'NOTES_FETCH_SUCCESS') {
    return {
      ...state,
      state: 'LOADED',
      notes: action.notes,
    };
  }
  if (action.type === 'NOTES_FETCH_ERROR') {
    return {
      ...state,
      state: 'ERROR',
      notes: [],
      errorMessage: action.errorMessage
    };
  }
  return state;
}
```

reducers/notesList.ts

Simple Approach (not recommended)

```
const mapStateToProps = (state: AppState, ownProps: HomeViewProps) => {
  return {
    notes: state.list.notes,
    state: state.list.state,
    errorMessage: state.list.errorMessage
  };
};
```

views/home/index.tsx

Coding Time!

SWING

Working version

git checkout **sprint1-list-finish**

Some issues with our solution

- **What will happen when we add search functionality?**
  - We don’t have a place to persist notes which have been filtered out
  - We would need to fetch them every time when someone is searching
- **What if we have second list with the subset of the notes**
  - Some of the notes could be found on both lists
  - Redundancy
  - Not a single source of truth

Store Normalisation

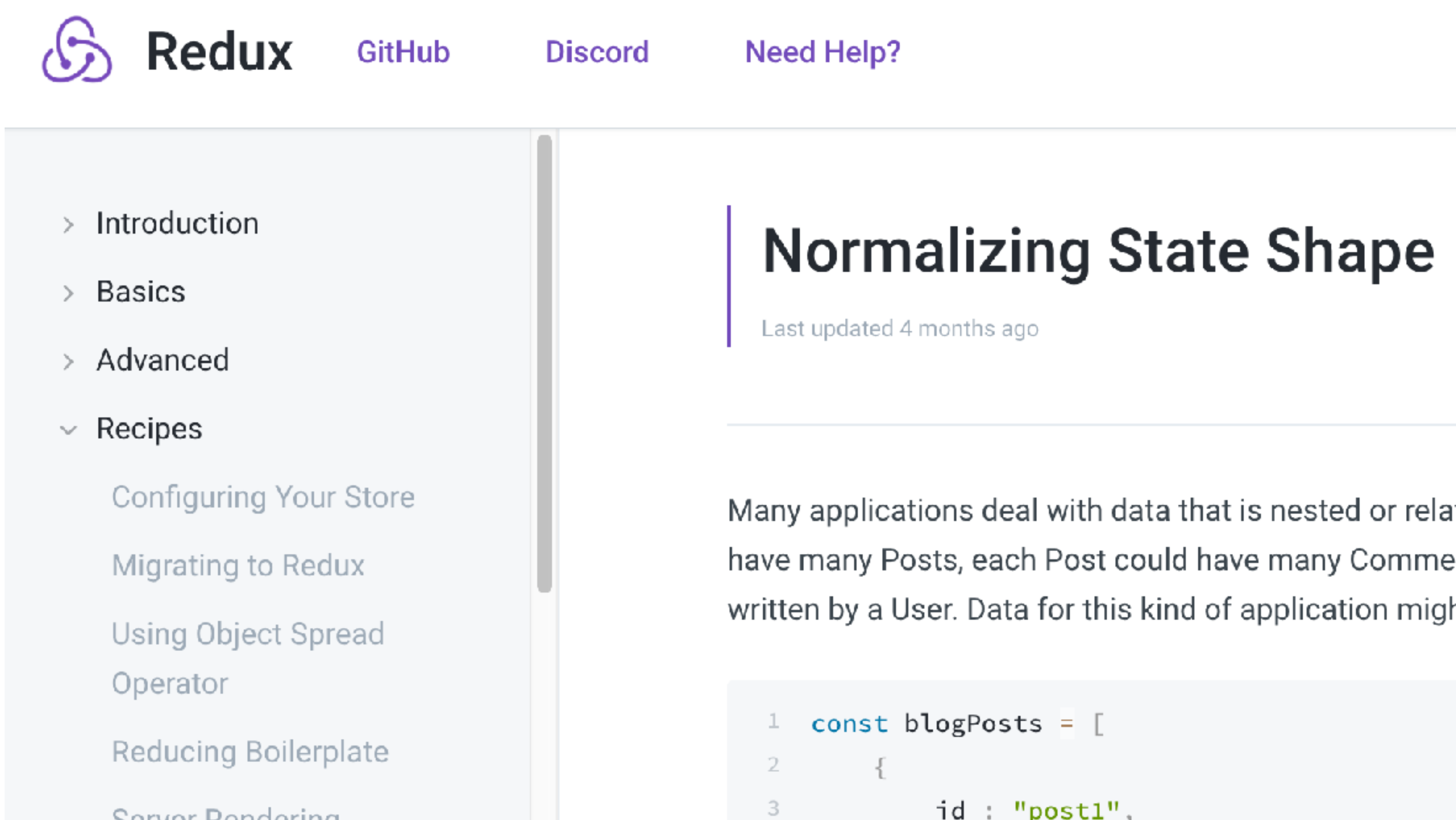
Instead of keeping all the data in one store, we will separate our stores into two categories

1. **Data Stores**
2. **UI Stores**

Store Normalisation

```
{
  entities: {
    notes: {
      byId: {
        1: {
          id: 1,
          title: 'First Note',
            description: 'Lorem Ipsum'
        },
        3: {
          id: 3,
          title: 'Another Note',
            description: 'Dolor Sit Amet'
        }
      },
      allIds: [1, 3]
    },
  },
  ui: {
    dashboardList: {
      state: 'LOADED',
        notes: [1, 3]
    }
  }
}
```

Store Normalisation



Redux GitHub Discord Need Help?

Introduction
Basics
Advanced
Recipes
Configuring Your Store
Migrating to Redux
Using Object Spread Operator
Reducing Boilerplate
Server Rendering

Normalizing State Shape

Last updated 4 months ago

Many applications deal with data that is nested or rela
have many Posts, each Post could have many Comme
written by a User. Data for this kind of application migh

```
const blogPosts = [
    {
        id : "post1",
```

Store Normalisation

For real projects:

Normalizr

Store Normalisation Readings

- Using the Redux Store Like a Database by Nick Sweeting
- Normalizing State Shape by Redux
- Redux - Why normalize? on StackOverflow

Improved Solution

```
export function notesListReducer(state: NotesListState, action: AppActions): NotesListState {
  if (action.type === 'NOTES_FETCH') {
    return {
      ...state,
      state: 'LOADING',
      notes: [],
    };
  }
  if (action.type === 'NOTES_FETCH_SUCCESS') {
    return {
      ...state,
      state: 'LOADED',
      notes: action.notes.map((n: NoteModel) => n.id),
    };
  }
  if (action.type === 'NOTES_FETCH_ERROR') {
    return {
      ...state,
      state: 'ERROR',
      notes: [],
      errorMessage: action.errorMessage
    };
  }
  return state;
}
```

reducers/notesList.ts

Improved Solution

```
export type NotesDict = {
  [Key: number]: NoteModel
};

export type NotesDictState = {
  byId: NotesDict,
  allIds: number[]
};
```

reducers/notesDict.ts

Improved Solution

```typescript
export function notesDictReducer(state: NotesDictState, action: AppActions): NotesDictState {
  if (action.type === 'NOTES_FETCH_SUCCESS') {
    return {
      byId: action.notes.reduce((acc, note) => ({ ...acc, [note.id]: note}), state),
      allIds: action.notes.map((n) => n.id)
    };
  }
  return state;
}
```

reducers/notesDict.ts

Improved Solution

```
export interface AppState {
  entities: {
    notes: NotesDictState,
  },
  ui: {
    list: NotesListState
  }
}
```

reducers/index.ts

Improved Solution

```
export function mainReducer(state: AppState = defaultState(), action: Action) {
  return {
    entities: {
      notes: notesDictReducer(state.entities.notes, action),
    },
    ui: {
      list: notesListReducer(state.ui.list, action)
    }
  };
}
```

reducers/index.ts

Improved Solution

```
const mapStateToProps = (state: AppState, ownProps: HomeViewProps) => {
  return {
    notes: state.ui.list.notes.map((noteId) => state.entities.notes.byId[noteId]),
    state: state.ui.list.state,
    errorMessage: state.ui.list.errorMessage
  };
};
```

views/home/index.ts

Coding Time!

SWING

Working version

git checkout **sprint1-finish**

🍕 🍺

Sprint 2

- Add Search Functionality

We have SearchComponent

We need to create new action and create proper reducer

Coding Time

SWING

Sprint 2

```
export interface IActionSearchNotes {
  type: 'NOTES_SEARCH',
  options: {
    searchText: string
  }
}
```

actions/index.ts

Sprint 2

```
export type AppActions = IActionNotesFetch | IActionNotesFetchSuccess
| IActionNotesFetchError | IActionSearchNotes;
```

actions/index.ts

Sprint 2

```
export function notesListReducer(state: NotesListState, action: AppActions, notes: NotesDictState):
NotesListState {
  /* all other reducers */
  if (action.type === 'NOTES_SEARCH') {
    return {
      ...state,
      notes: notes.allIds
        .map((id) => notes.byId[id])
        .filter(filterByText(action.options.searchText.toLowerCase()))
        .map((n) => n.id)
    };
  }
  return state;
}
```

reducers/notesList.ts

Sprint 2

```
function filterByText(text: string): (n: NoteModel) ⇒ boolean  {
  return (note: NoteModel): boolean ⇒ {
    return note.title.toLowerCase().indexOf(text) > -1 ||
      note.content.toLowerCase().indexOf(text) > -1;
  };
}
```

reducers/notesList.ts

Working version

git checkout **sprint2-finish**

Bonus

SWING

Ideas for extending app

- Add ability to remove an element
- Add Tagging functionality
- Add option to search by tag

https://twitter.com/kulak_at

https://medium.com/@kulak

https://github.com/kulak-at

#warsawmicroconf