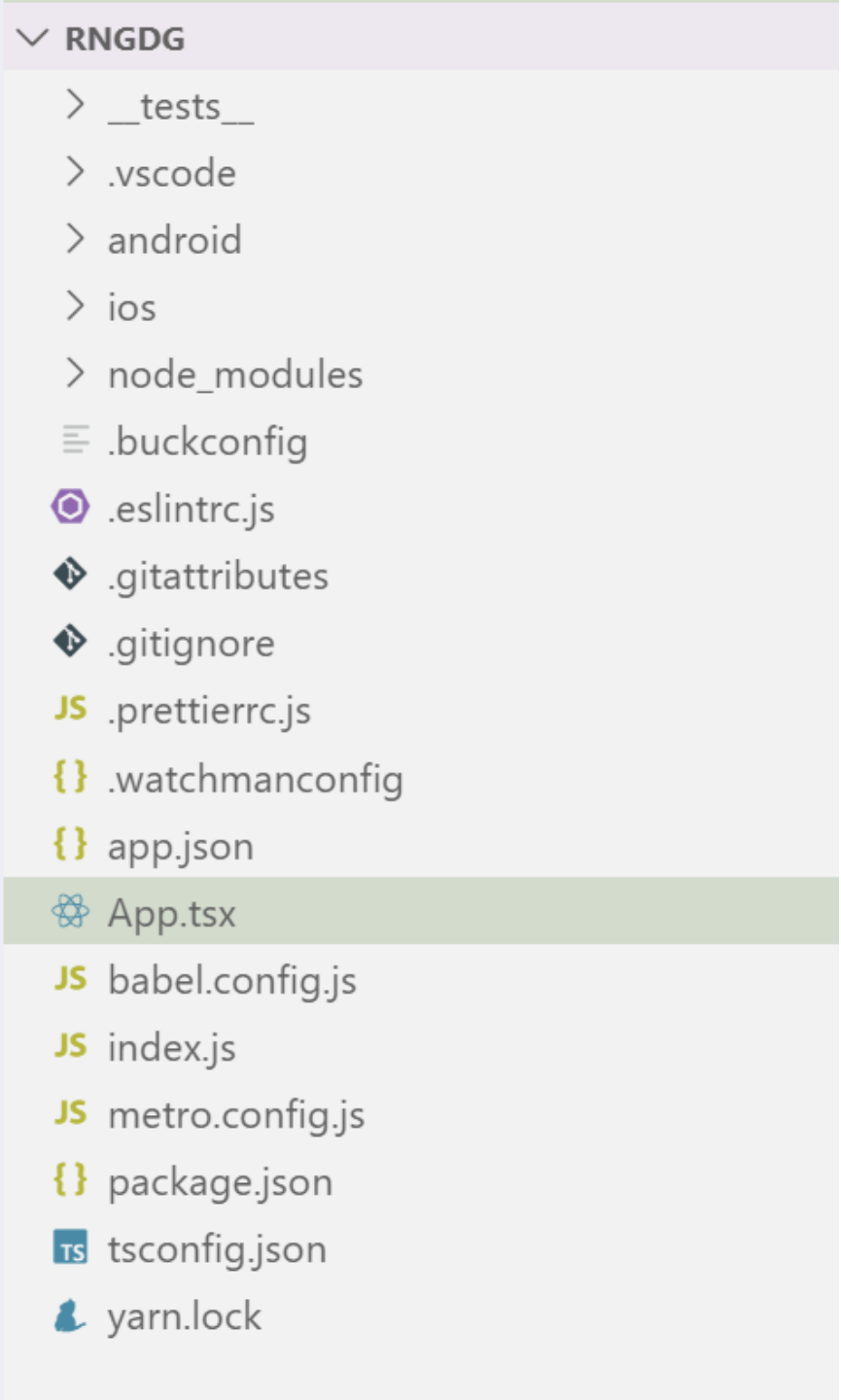
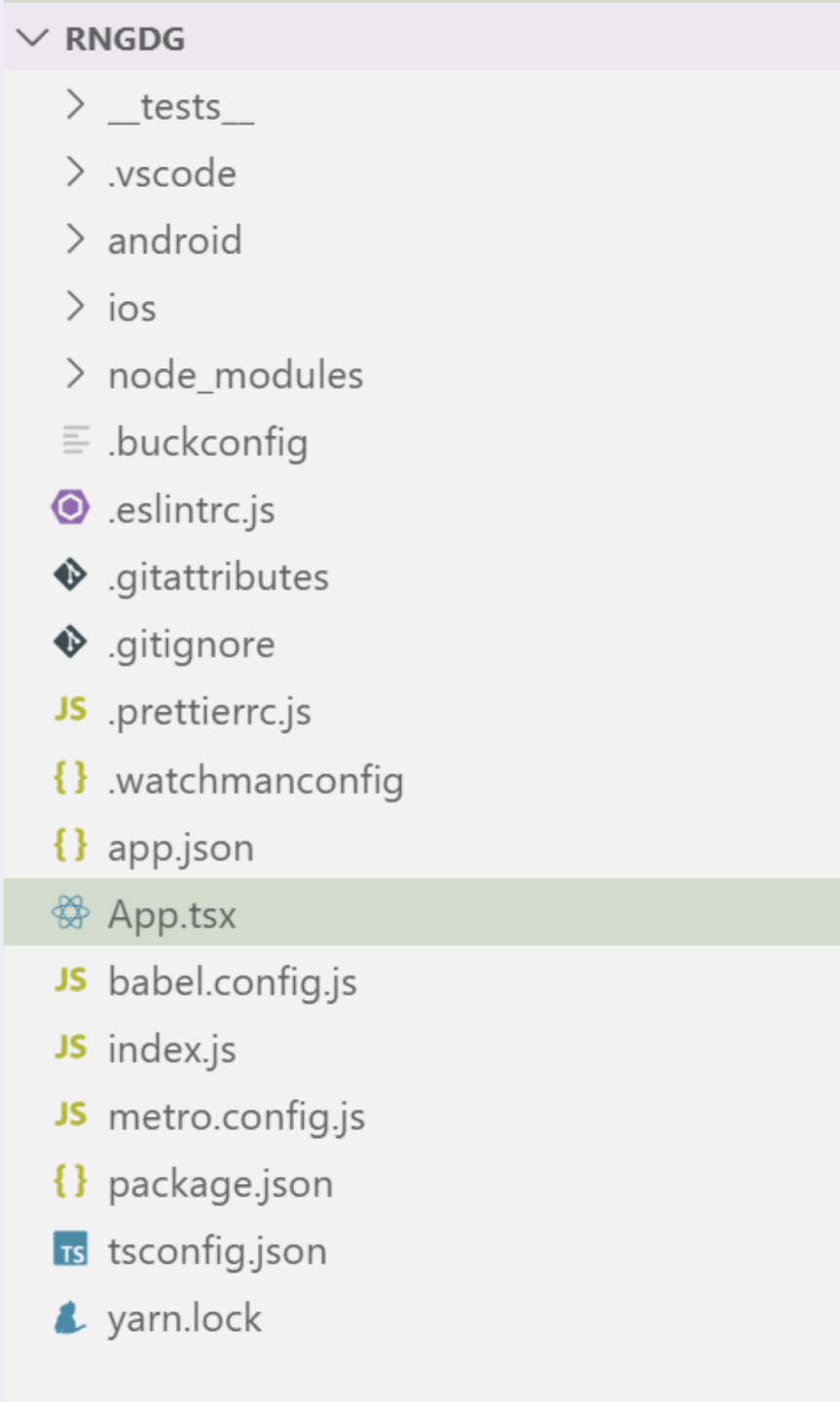
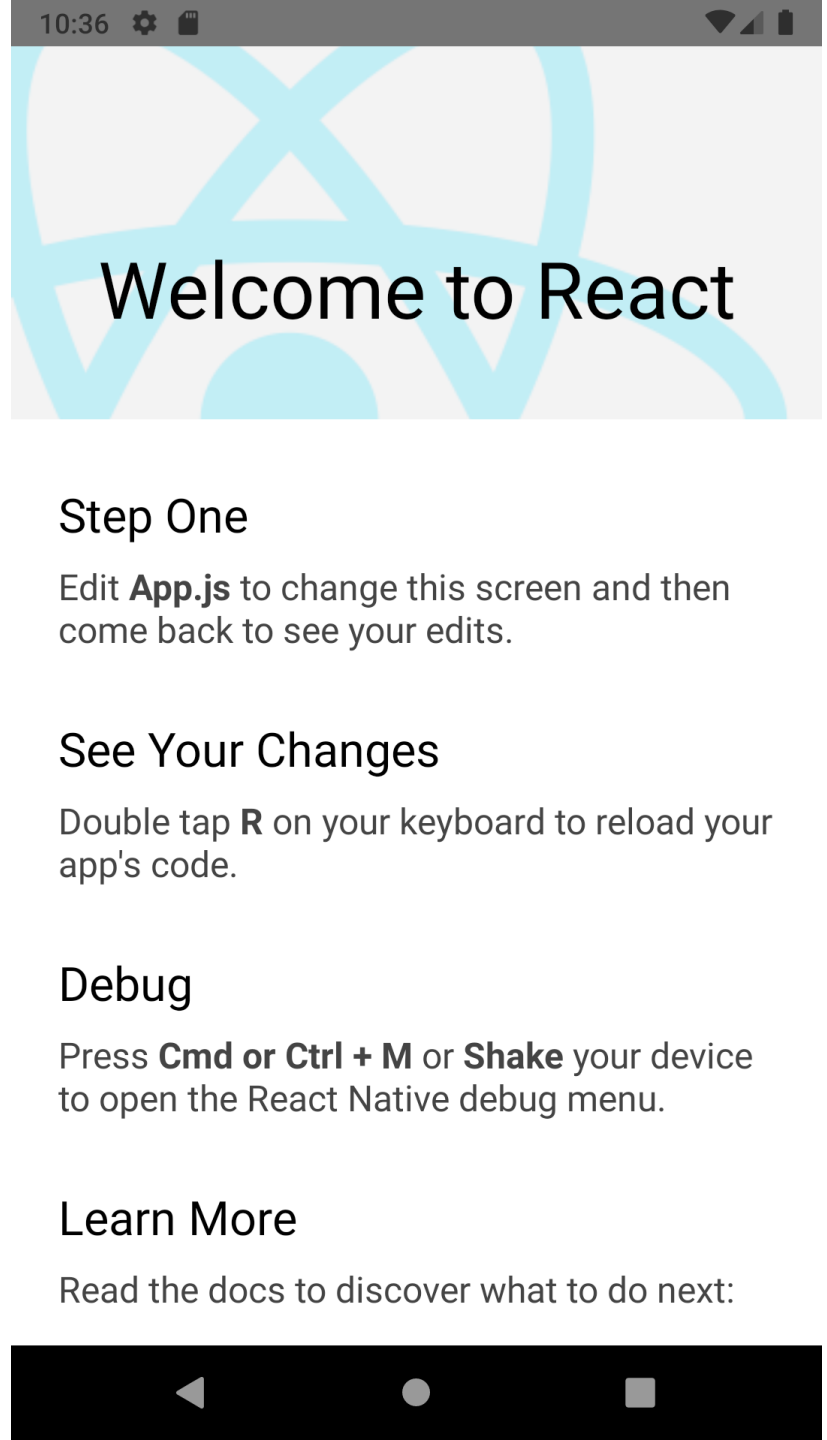
Cross platform mobile App

with react native

```
npx react-native init spike4gdg --template react-native-template-typescript
cd spike4gdg
yarn android # run on device / emulator
```


RNGDG
__tests__
.vscode
android
ios
node_modules
.buckconfig
.eslintrc.js
.gitattributes
.gitignore
.prettierrc.js
.watchmanconfig
app.json
App.tsx
babel.config.js
index.js
metro.config.js
package.json
tsconfig.json
yarn.lock


RNGDG
__tests__
.vscode
android
ios
node_modules
.buckconfig
.eslintrc.js
.gitattributes
.gitignore
.prettierrc.js
.watchmanconfig
app.json
App.tsx
babel.config.js
index.js
metro.config.js
package.json
tsconfig.json
yarn.lock


10:36
Welcome to React
Step One
Edit App.js to change this screen and then come back to see your edits.
See Your Changes
Double tap R on your keyboard to reload your app's code.
Debug
Press Cmd or Ctrl + M or Shake your device to open the React Native debug menu.
Learn More
Read the docs to discover what to do next:

Basics

```ts
const add = (a: number, b: number) => { a + b }; // {} optional
const seven = add(3, 4);

const adder = (a: number) => (b: number) => a + b;
const add3 = adder(3); // of type (b: number) => number
const eight = add3(5); // of type number

const obj = { a: 1, b: 2 }
const { a } = obj; // const a = obj.a;
const copy = { ...obj }; // "copy by value"
const incA = { ...obj, a: a + 1 };  // overwrite some values of copy
const b = 7; // define a variable
const setB = { ...incA, b } // shorthand { b: b }
```

```ts
const add = (a: number, b: number) => { a + b }; // {} optional
const seven = add(3, 4);

const adder = (a: number) => (b: number) => a + b;
const add3 = adder(3); // of type (b: number) => number
const eight = add3(5); // of type number

const obj = { a: 1, b: 2 }
const { a } = obj; // const a = obj.a;
const copy = { ...obj }; // "copy by value"
const incA = { ...obj, a: a + 1 };  // overwrite some values of copy
const b = 7; // define a variable
const setB = { ...incA, b } // shorthand { b: b }
```

```ts
const add = (a: number, b: number) => { a + b }; // {} optional
const seven = add(3, 4);

const adder = (a: number) => (b: number) => a + b;
const add3 = adder(3); // of type (b: number) => number
const eight = add3(5); // of type number

const obj = { a: 1, b: 2 }
const { a } = obj; // const a = obj.a;
const copy = { ...obj }; // "copy by value"
const incA = { ...copy, a: a + 1 };  // overwrite some values of copy
const b = 7; // define a variable
const setB = { ...incA, b } // shorthand { b: b }
```

Your first RN component

```
0  import React, {Component} from 'react';
1  import {Button, Text} from 'react-native';
2
3  const sayHi = () => console.log('Hi console!');
4
5  class HiButton extends Component {
6    public render = () => (
7      <>
8        <Text>Hi there!</Text>
9        <Button title="Say Hi" onPress={sayHi} />
10     </>
11   );
12 }
13
14 export {HiButton};
```

Props and State

```tsx
interface Props {
  greeting: string;
}
interface State {
  hasGreeted: boolean;
}

// Class
class HiButton {…}
// Function
const HiButton = () => {…}

const HiDemo = () =>
  <HiButton greeting="GDG" />;

export {HiDemo};
```

```tsx
interface Props {
  greeting: string;
}
interface State {
  hasGreeted: boolean;
}

// Class
class HiButton {…}
// Function
const HiButton = () => {…}

const HiDemo = () =>
  <HiButton greeting="GDG" />;

export {HiButton, HiDemo};
```

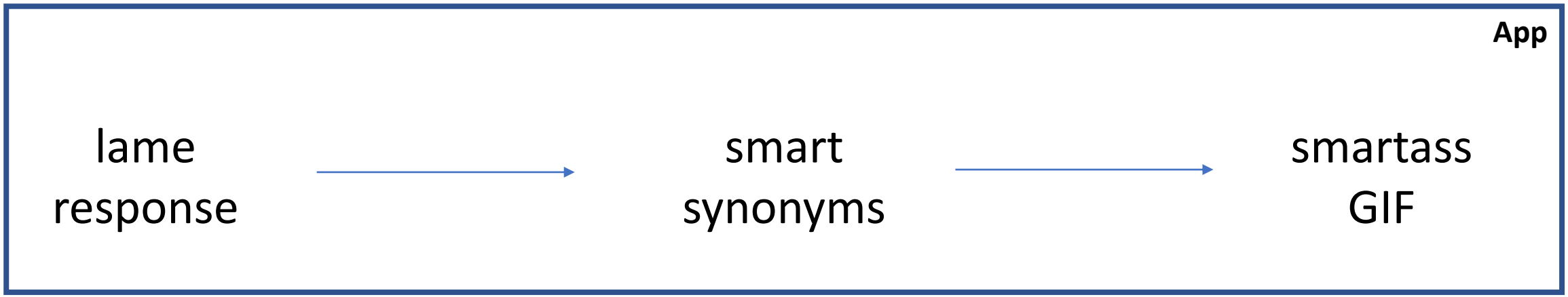
```tsx
class HiButton extends Component<Props, State> {
  state = {hasGreeted: false};

  private sayHi = () => {
    if (this.state.hasGreeted) {
      console.warn('You greeted me already!');
    } else {
      console.log('Hi console!');
      this.setState({hasGreeted: true});
    }
  };

  public render = () => (
    <>
      <Text>{this.props.greeting}</Text>
      <Button title="Say Hi" onPress={this.sayHi} />
    </>
  );
}
```

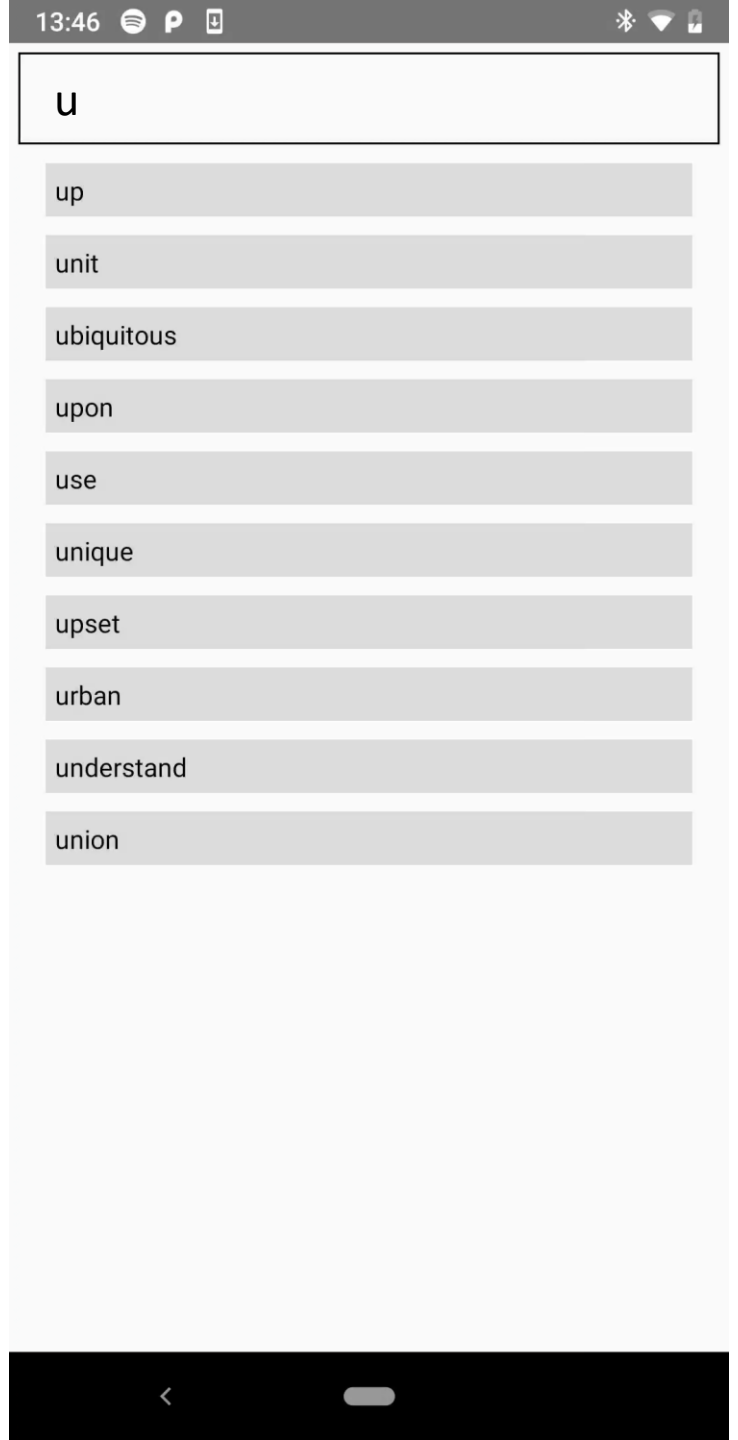
```tsx
interface Props {
  greeting: string;
}
interface State {
  hasGreeted: boolean;
}

// Class
class HiButton {…}
// Function
const HiButton = () => {…}

const HiDemo = () =>
  <HiButton greeting="GDG" />;

export {HiButton, HiDemo};
```

```tsx
const HiButton = (props: Props) => {
  const [hasGreeted, setGreeted] = useState(false);
  const sayHi = () => {
    if (hasGreeted) {
      console.warn('You greeted me already!');
    } else {
      console.log('Hi console!');
      setGreeted(true);
    }
  };
  return (
    <>
      <Text>{props.greeting}</Text>
      <Button title="Say Hi" onPress={sayHi} />
    </>
  );
};
```

Let’s use it…

to find the best GIF you never wanted


App
lame response
smart synonyms
smartass GIF


13:46
u
up
unit
ubiquitous
upon
use
unique
upset
urban
understand
union

```
const renderListItem = ({item}) => (
  <Text>{item}</Text>
);

const App = () => {
  const [search, setSearch] = useState('');
  return (
    <>
      <TextInput
        value={search}
        onChangeText={text => setSearch(text)}
        onSubmitEditing={_ => console.log(`search for ${search}`)}
      />
      <FlatList data={[]} renderItem={renderListItem} />
    </>
  );
};
```

```js
const getTypeAhead = async (query) => {
  const raw = await fetch(`https://api.datamuse.com/sug?s=${query}`);
  const res = await raw.json();
  return res.map(s => s.word);
};
```

```
[ // FROM API
  {
    "word": "code",
    "score": 4081
  },
  {
    "word": "codex",
    "score": 720
  },
  {
    "word": "codec",
    "score": 400
  },
  {
    "word": "codeine",
    "score": 381
  }
]
```

```
[ // OUTPUT
  "code",
  "codex",
  "codec",
  "codeine"
]
```

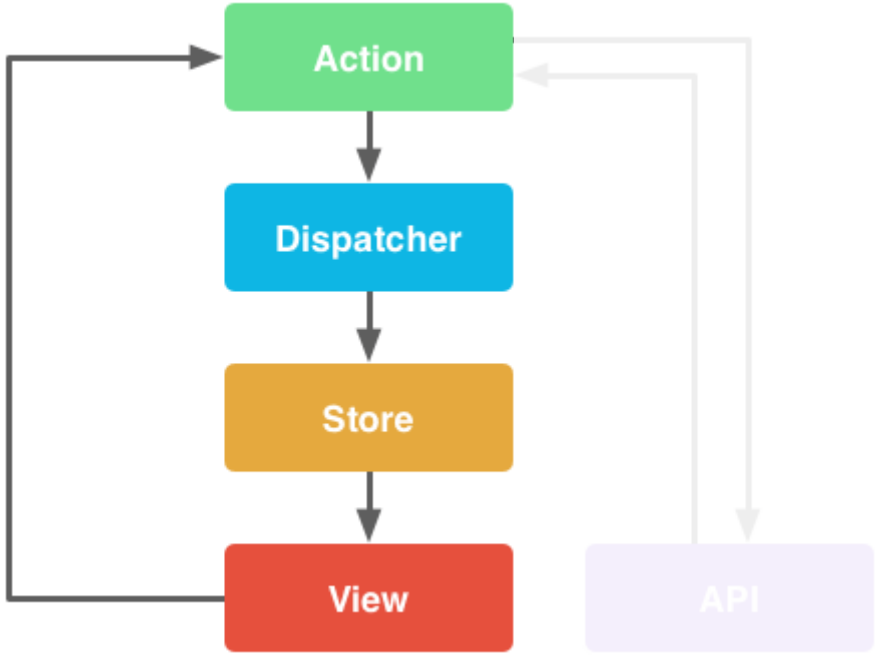
```ts
const getTypeAhead = async (query: string): Promise<string[]> => {
  const raw = await fetch(`https://api.datamuse.com/sug?s=${query}`);
  const res = (await raw.json()) as [{ word: string, score: number }];
  return res.map(s => s.word);
};
```

```tsx
const getTypeAhead = async (query: string): Promise<string[]> => {
  const raw = await fetch(`https://api.datamuse.com/sug?s=${query}`);
  const res = (await raw.json()) as [{ word: string }];
  return res.map(s => s.word);
};

const App = () => {
  const [search, setSearch] = useState('');
  const [results, setResults] = useState<string[]>([]);
  useEffect(() => {
    getTypeAhead(search).then(setResults);
  }, [search]);
  return (
    <>
      <TextInput
        value={search}
        onChangeText={text => setSearch(text)}
      />
      <FlatList
        data={results}
        renderItem={renderListItem}
      />
    </>
  );
};
```

Cool, but …

- single responsibility principle
- cast to: `[{ word: string }]` depends on external API
- every tap calls service
- requests can return in any order
- `async` calls not easy to read (`useEffect`)
- will not scale
- Prop drilling
- …

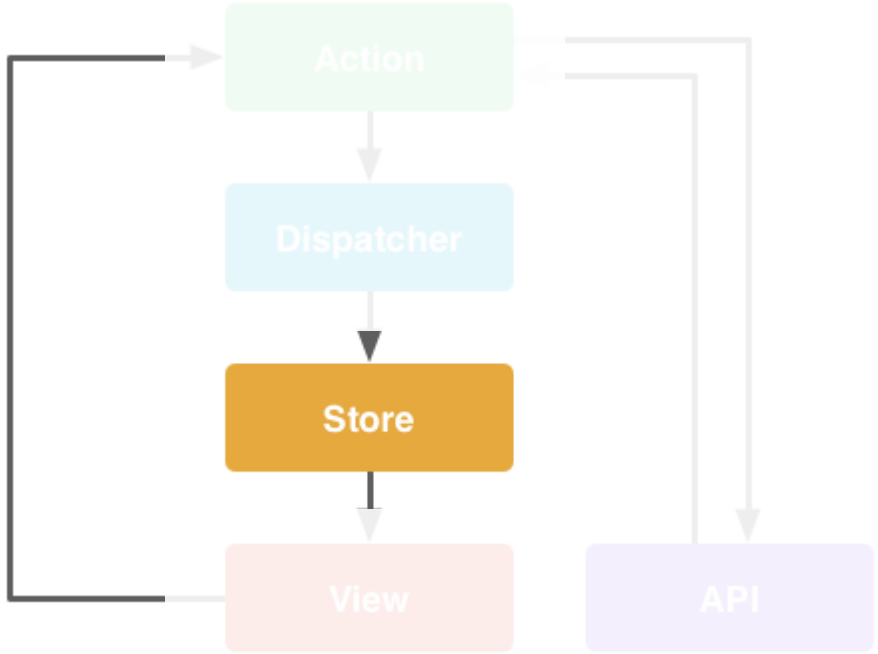
Separate state management from UI



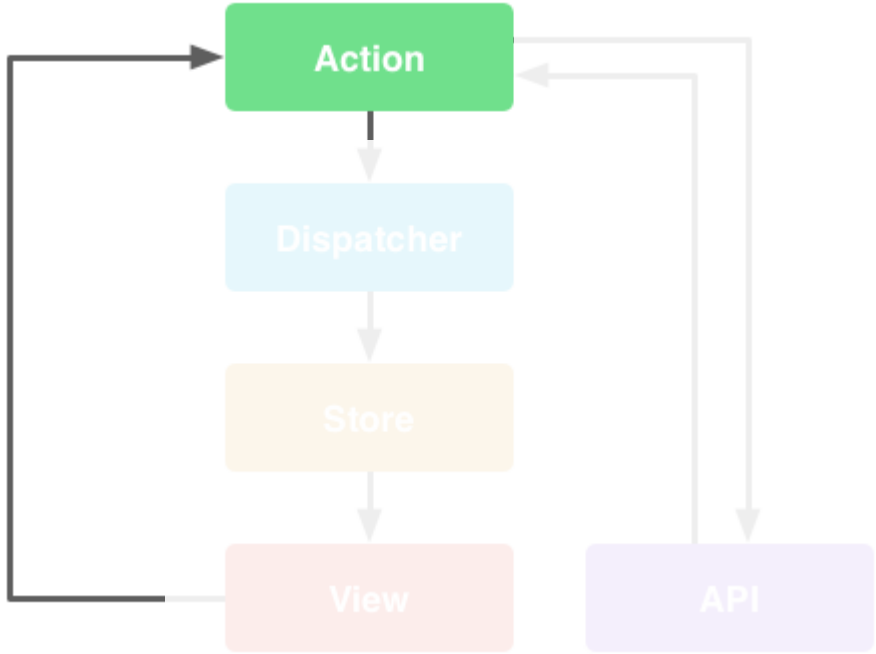
https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture

Load and manage images

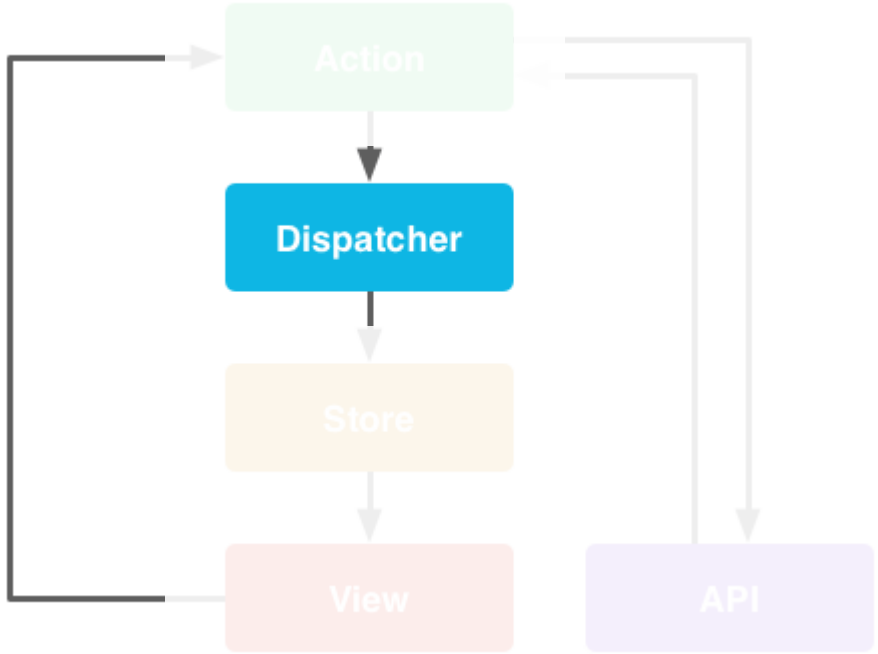
```ts
/// TYPES
interface IState {
  query: string;
  images: Array<ImageURISource>;
}
/// immutable state
const defaultState: IState = {
  query: '',
  images: [],
};
```



```ts
/// ACTIONS … (FSA)
interface SetQueryAction {
  type: typeof 'SET_QUERY';
  payload: { query: string };
}

interface SetImagesAction {
  type: typeof 'SET_IMAGES';
  payload: { images: Array<ImageURISource> };
```



```
/// REDUCER
const reducer = (state, action) => {
  switch (action.type) {
    case 'SET_QUERY': {
      return { ...state,
               query: action.payload.query };
    }
    case 'SET_IMAGES': {
      return { ...state,
               images: action.payload.images };
    }
    default:
      return state;
  }
};
```



```tsx
interface Props {
  hasGifs: boolean;
  searchGifs: (query: string) => void;
}

class StupidComponent extends Component<Props> {
  public render = () => (
    <Button
      title={this.props.hasGifs ? "much GIFs" : "such empty"}
      onPress={() => this.props.searchGifs("moar")} />
  )
}

const mapStateToProps = (state) => ({
  hasGifs: state.images.length > 0,
});

const mapDispatchToProps = (dispatch) => ({
  searchGifs: search =>
    dispatch({type: 'SET_QUERY', payload: {query: search}}),
});

export default connect(
  mapStateToProps,
```

```
6  class StupidComponent extends Component<Props> {
7    public render = () => (
8      <Button
9        title={this.props.hasGifs ? "much GIFs" : "such empty"}
10       onPress={() => this.props.searchGifs("moar")} />
11   )
12 }
13
14 const mapStateToProps = (state) => ({
15   hasGifs: state.images.length > 0,
16 });
17
18 const mapDispatchToProps = (dispatch) => ({
19   searchGifs: search =>
20     dispatch({type: 'SET_QUERY', payload: {query: search}}),
21 });
22
23 export default connect(
24   mapStateToProps,
25   mapDispatchToProps,
26 )(StupidComponent);
```

```
class StupidComponent extends Component<Props> {
  public render = () => (
    <Button
      title={this.props.hasGifs ? "much GIFs" : "such empty"}
      onPress={() => this.props.searchGifs("moar")} />
  )
}

const mapStateToProps = (state) => ({
  hasGifs: state.images.length > 0,
});

const mapDispatchToProps = (dispatch) => ({
  searchGifs: search =>
    dispatch({type: 'SET_QUERY', payload: {query: search}}),
});

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(StupidComponent);
```

Redux

- Very weird at the beginning
- Makes sense once you start using it
- RTK (redux toolkit) allows to write less boilerplate, but you should first understand what happens under the hood

API Explorer

Take our API for a spin by inputting some sample queries and view live responses!

Request

Choose an app / API Key

GDG demo: USQe5n2uShHwsMQMKcEBaaGY3skWACjn

Choose a resource

GIPHY Public API

Choose an endpoint

Search

Request URL

https://api.giphy.com/v1/gifs/search?api_key=USQe5n2uShHwsMQMKcEBaaGY3skWACjn&q=springer&limit=25&offset=0&rating=G&lang=en

Send Request

Parameters

q REQUIRED

springer

limit

25

offset

0

rating

G

lang

en

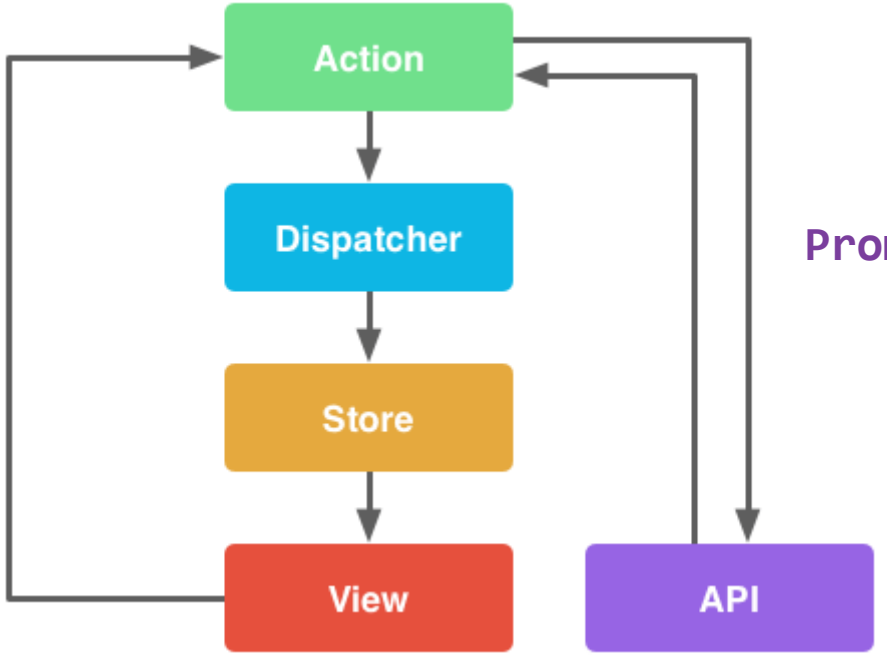
Response


```
{
   "data":
   [
       {
           "type": "gif",
           "id": "l1J9NGOVsxcDzgqOI",
           "url": "https://giphy.com/gifs/mlb-l1J9NGOVsxcDzgqOI",
           "slug": "mlb-l1J9NGOVsxcDzgqOI",
           "bitly_gif_url": "https://gph.is/2gYVqkR",
```

```
  "scripts": {
    "swagger-download": "wget http://…/swagger-codegen-cli.jar",
    "swagger-codegen": "java -jar swagger-codegen-cli.jar
       generate -i giphy.yml -l typescript-fetch"
  },

/**
 * Search all GIPHY GIFs for a word or phrase. Punctuation will be stripped …
 * @summary Search GIFs
 * @param {string} q Search query term or phrase.
 * @param {number} [limit] The maximum number of records to return.
 */
searchGifs(q: string, limit?: number, …)… => Promise<InlineResponse200> {…}

export interface Image {
    /**
     * The number of frames in this GIF.
     */
    frames?: string;
    /**
     * @memberof Image
     */
    height?: string;
…
```

```
  "scripts": {
    "swagger-download": "wget http://…/swagger-codegen-cli.jar",
    "swagger-codegen": "java -jar swagger-codegen-cli.jar
       generate -i giphy.yml -l typescript-fetch"
  },

/**
 * Search all GIPHY GIFs for a word or phrase. Punctuation will be stripped …
 * @summary Search GIFs
 * @param {string} q Search query term or phrase.
 * @param {number} [limit] The maximum number of records to return.
 */
searchGifs(q: string, limit?: number, …)… => Promise<InlineResponse200> {…}

export interface Image {
    /**
     * The number of frames in this GIF.
     */
    frames?: string;
    /**
     * @memberof Image
     */
    height?: string;
…
```

`async` actions?




```
Promise<InlineResponse200>
```

https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture

Solutions

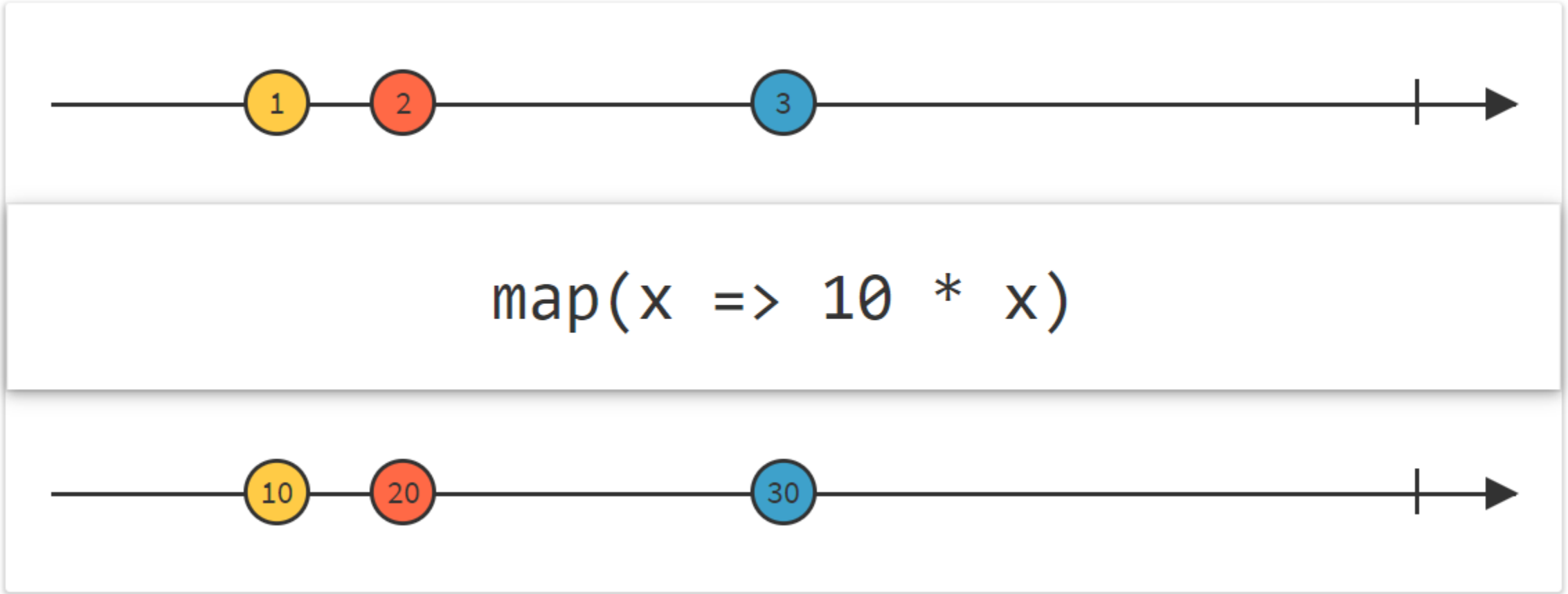
- redux-thunk → dispatch an (async) function
- redux-saga → use ES6 Generators
- redux-observable → use RxJS

redux-thunk

```
const mapDispatchToProps = (dispatch) => ({
  loadTypeAhead: (search) => dispatch(typeAheadThunk(search)),
});
```

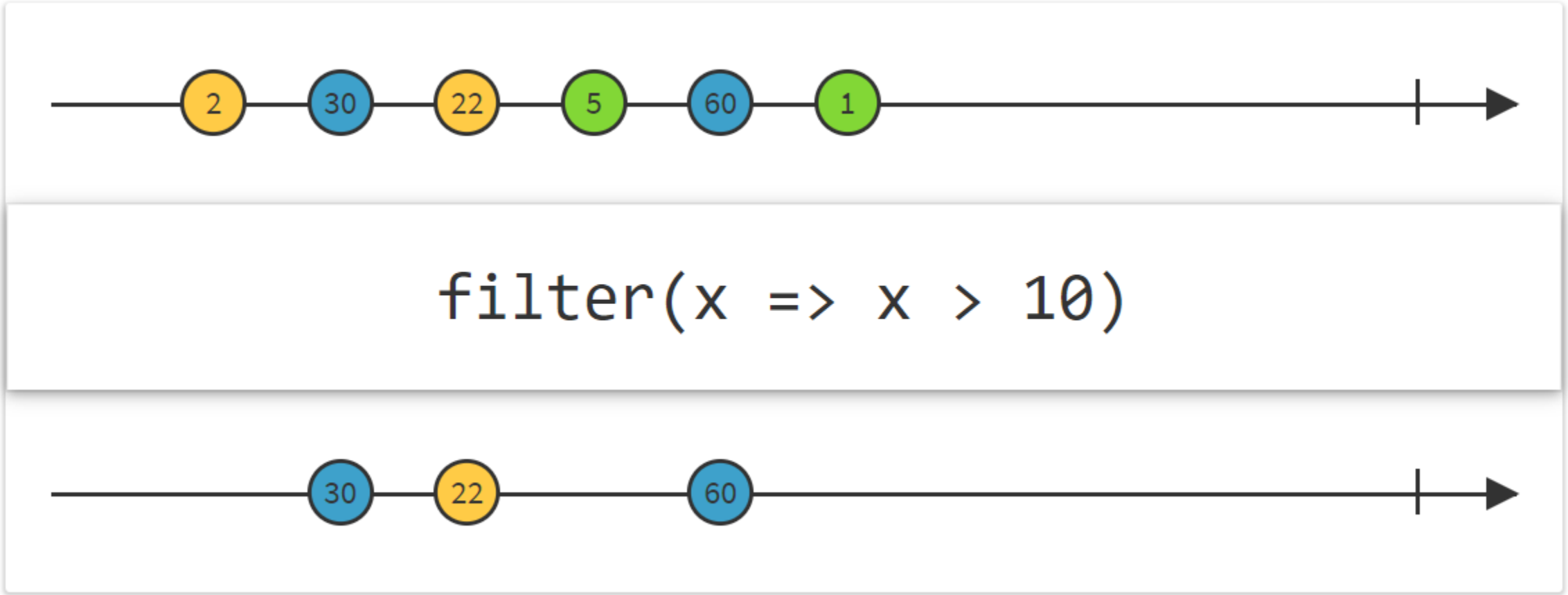
- Easy to start with
- Move logic out of component
- Rewrite same logic over and over
- No common place for retry / auth flow logic
- explicit imperative async code

Meet RxJS (before we have a look at redux-observable)



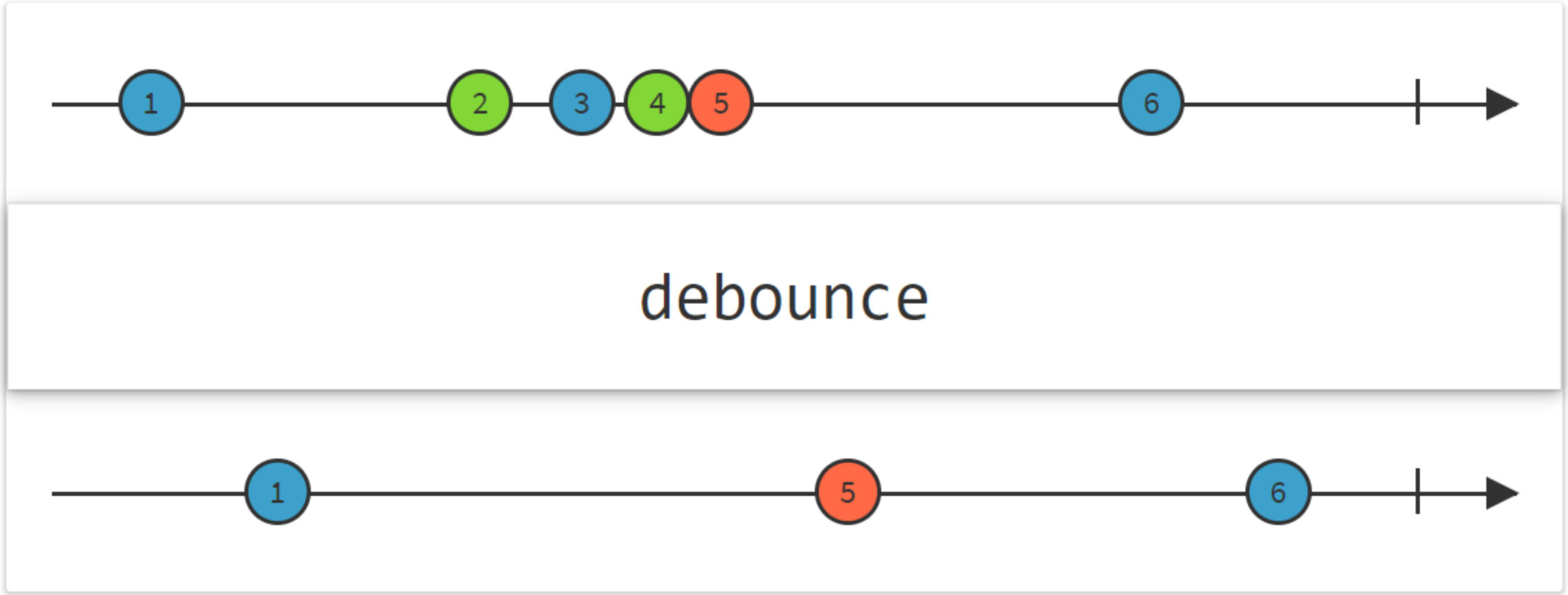
http://reactivex.io/documentation/operators/map.html

Meet RxJS



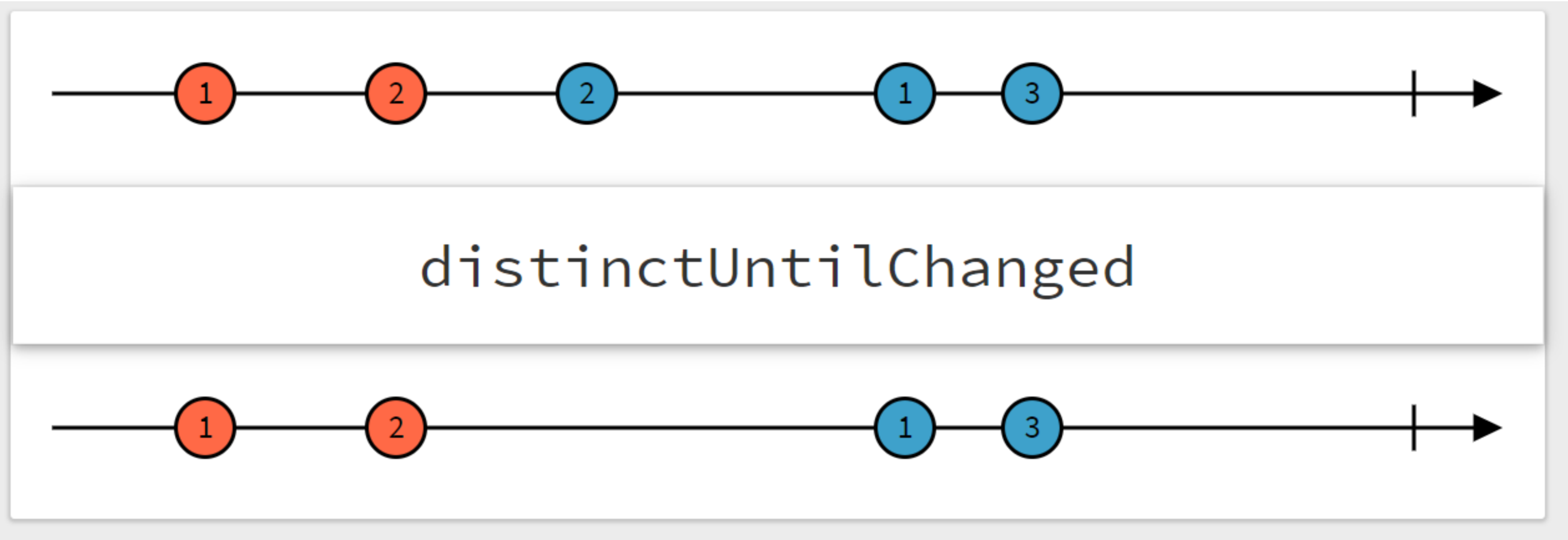
http://reactivex.io/documentation/operators/filter.html

Meet RxJS



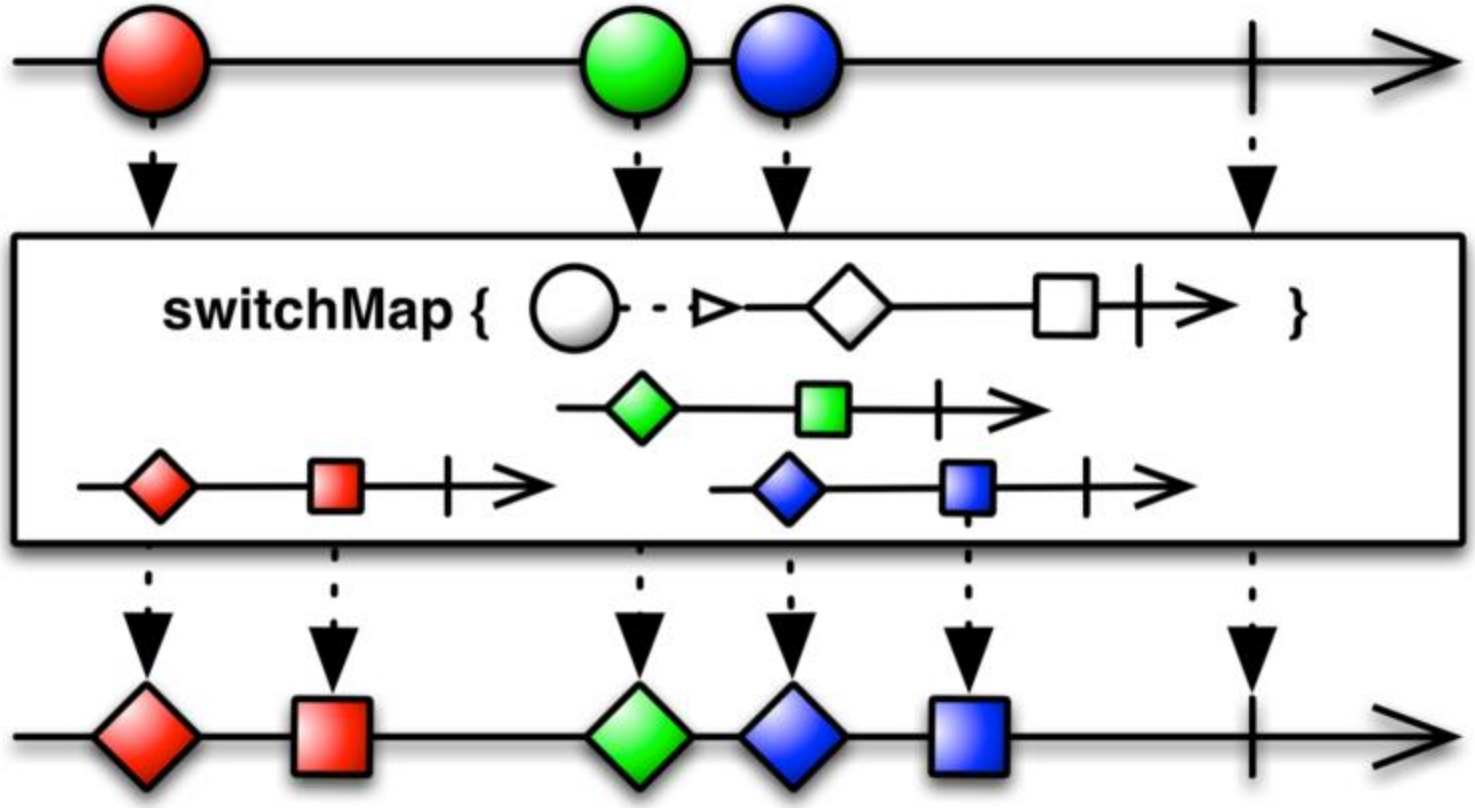
http://reactivex.io/documentation/operators/debounce.html

Meet RxJS



https://rxmarbles.com/#distinctUntilChanged

Meet RxJS



https://yakovfain.com/2017/09/19/rxjs-essentials-part-6-the-switchmap-operator/

Meet RxJS

```
const suggestions$ = fromEvent(changeSearch).pipe( // stream the taps
    map(event => event.value), // extract information
    filter(search => !!search), // check it’s not empty
    debounceTime(300), // go beyond this point only if typing pauses
    distinctUntilChanged(), // do nothing if value did not change
    switchMap(getSuggestionsFromApi) // call API and return results
                                     // discard old if new request
  );
```

redux-observable / NgRx

```
const _ = { type: "SET_SEARCH", payload: { query: "g" } };
const _ = { type: "SET_SEARCH", payload: { query: "go" } };
const _ = { type: "SET_SEARCH", payload: { query: "goo" } };

const epic = action$ /* action in */ => action$.pipe(
    filter(action => action.type === "SET_SEARCH"),
    map(action => action.payload.query),
    debounceTime(300),
    distinctUntilChanged(),
    switchMap(getTypeAhead), // call to async function
    map(s => ({ type: "SET_SUGGESTIONS", payload: { suggestions: s }}))
); /* action(s) out */
```

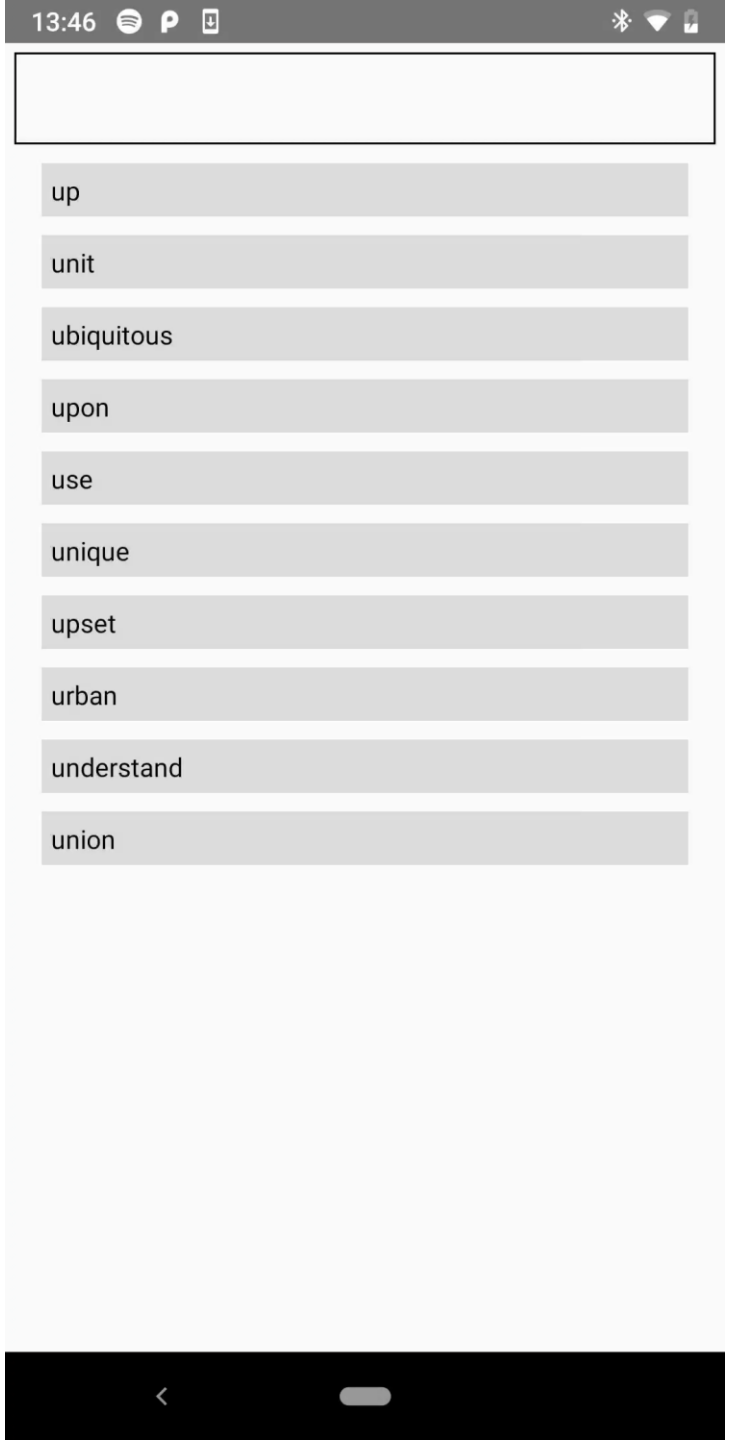
redux-observable / NgRx

```
const _ = { type: "SET_SEARCH", payload: { query: "g" } };
const _ = { type: "SET_SEARCH", payload: { query: "go" } };
const _ = { type: "SET_SEARCH", payload: { query: "goo" } };

const epic = action$ /* action in */ => action$.pipe(
    filter(action => action.type === "SET_SEARCH"),
    map(action => action.payload.query),
    debounceTime(300),
    distinctUntilChanged(),
    switchMap(getTypeAhead), // call to async function
    map(s => ({ type: "SET_SUGGESTIONS", payload: { suggestions: s }))
); /* action(s) out */
```

Yay

- http://bit.ly/rn-gdg

- But
  - nodejs-lts version
  - swagger troubles
  - redux-observable error handling



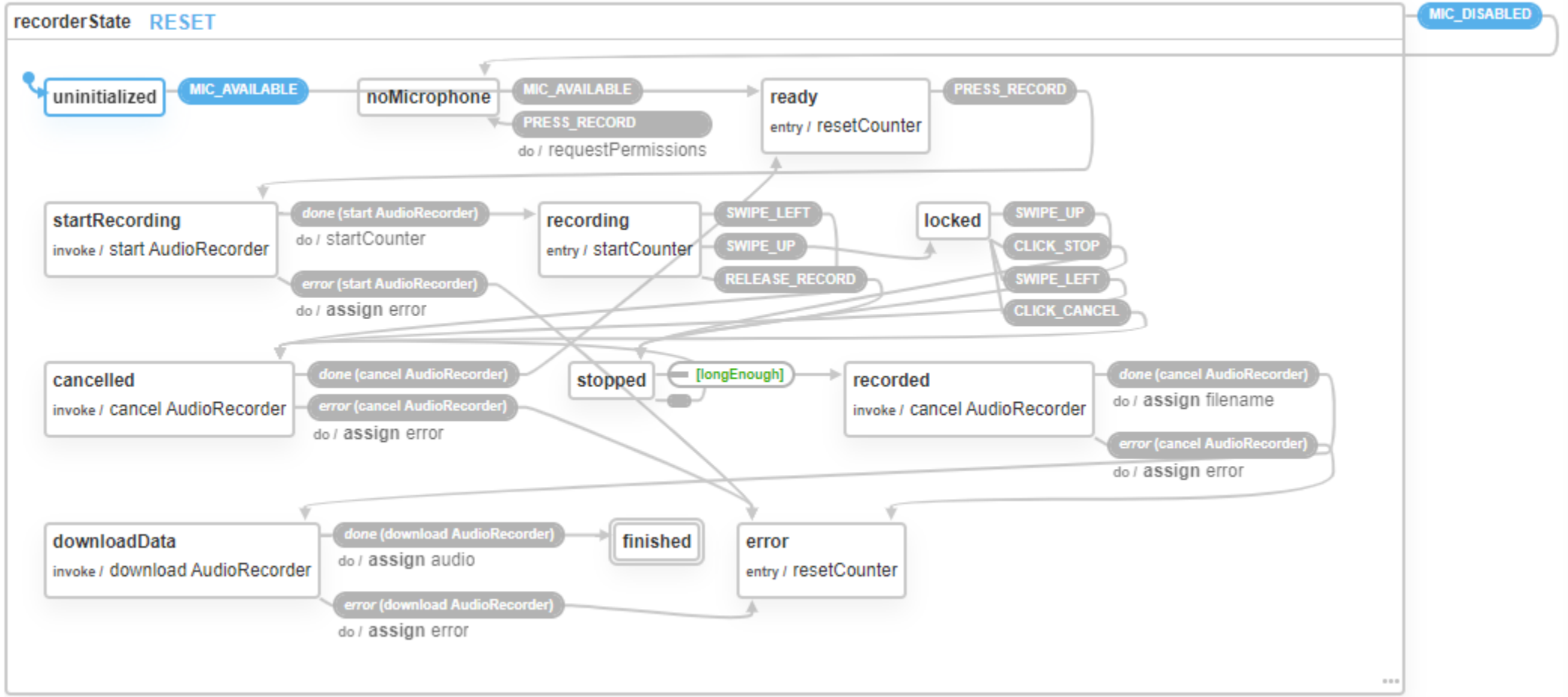
other cool concepts

Voice recording styles IMAGE

(image was removed)

```ts
export interface State {
    // is the microphone supported?
    microphoneSupported: boolean;
    // recording started
    isRecording: boolean;
    // recording is locked
    isLocked: boolean;
    // ...

    // ...

    // ...

    // ...

    // ...

    // ...
```

prime example of a footgun!



https://xstate.js.org/viz/

Generated from your live code!

and what about native?

```java
@ReactMethod // Make available to JS
public void stop(Promise promise) {
    if (null != recorder) {
        audioFile = recorder.stop();
        recorder = null;
        promise.resolve(audioFile);
    } else {
        promise.reject(E_FILE_ERROR, "Recording has not been started.");
    }
}


// Send events to JS code
private void sendEvent(AudioSpec audioSpec) {
    WritableMap params = Arguments.createMap();
    params.putInt("payload", RNReactNativeVoicerecorderModule.EVENT_AUDIOSPEC_VALUE);
    params.putDouble("average", audioSpec.avg);
    mReactContext.getJSModule(SOME.class).emit("VoiceRecorderEvent", params);
}
```

Looking back

- really cool concepts, active community
- little native code, easy to inspect packages
- typescript is awesome, but still JS
- AppCenter (build, distribution, code-push, analytics, …)
- swagger is nice, but also has its problems
- give Rx** a try
- Higher-Order-Components: `withAuthentication(MyScreen)`
- Parent Component: `<Auth><MyScreen /></Auth>`