Pokedex React Tutorial

Jeroen Nouws

Table of Contents of Pokedex React Tutorial

Click here if you rather have a PDF

Preface

Asumptions

The reader of this tutorial should have a basic understanding of web based technologies such as html, css and javascript.

Versions used

- node: 14 (LTS)
- yarn: 1.22.5
- react: 17

Code Examples

Reading a tutorial can give you insights into a tech stack, but writing code and testing it gives you a good feeling of learning to work with React. All code is available on Github.

Commands and CLI

All commands in this tutorial are run in a bash shell on linux and not tested in windows based terminals like command prompt or Powershell.

I. Introduction

Chapter 1. Typescript Primer

1.1. What is Typescript? A quick overview

- superset of javascript
- adding typescript to your project
  - installing typescript
  - adding tsconfig
  - adding package.json script
- typing
  - types
  - interfaces
- classes
- enums

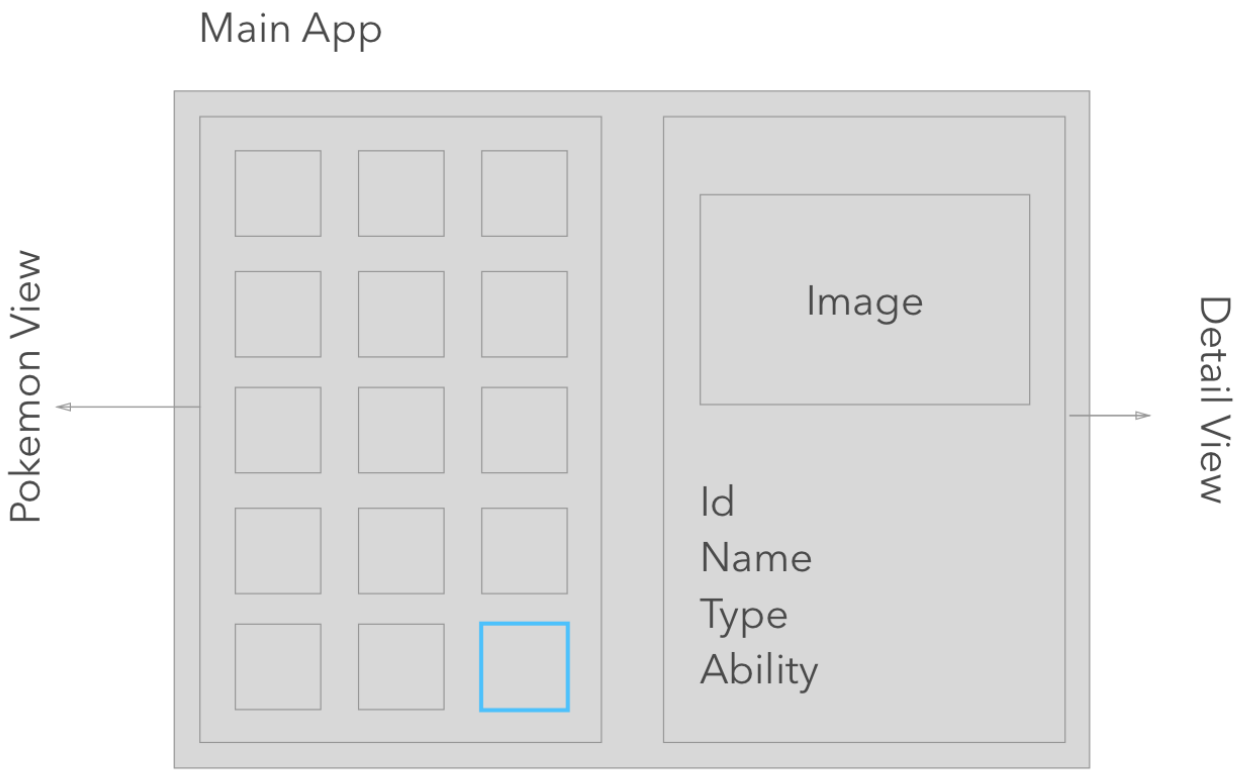
Chapter 2. Project Setup

2.1. What are we building?

In this tutorial we'll be building a pokedex react app. We'll do so by implementing the commonly used component/container pattern (representational/logical components).
We'll also touch visual testing and documentation by writing stories for Storybook and use Jest for component testing and snapshot testing.
Finally we'll introduce redux into our project for managing state and thunk middleware to handle asyncronous api calls.
This setup might seem overkill for a project of this size and there are better solutions to handle state than through redux for such a small project, but the purpose of this tutorial is to teach you some commonly used technologies and patterns within the React ecosystem.

2.1.1. Wireframe

Our Application has a very basic layout consisting of 2 columns, in the left hand column we'll display a list of selectable pokemons, while in the right side column we'll display either the details of the selected pokémon or a placeholder text.



2.1.2. Datasource

All our Pokémon information will be pulled from the Poke API. This is a standard REST-Api, altough at the time of writing they are going into beta with their GraphQL api.
This API holds an extensive source of information about all different Pokémons, going from names,

to types, to abilities and more. If you like this tutorial go give those guys your support.

2.2. Creating a new React Project

Before we start creating a new React project we should have a little talk about what React is and isn't.
React is an open source javascript **library** that's used for building user interfaces and single page applications. Altough there are other popular frameworks around, React stands out compared to other frameworks like Angular and Vue.js by being:

- **Declerative** instead of imperative
- **Not opinionated**, except on how to render views, React doesn't care how you structure or build your project.
- **Lightweight**
- **Library not a framework**, React focusses on creating components and building pages, solutions like state management and routing aren't part of React.js, but are managed by the community.

NOTE	Since React isn't opinionated you'll often find different solutions for the same problem i.e.: Flux vs Redux for state management, css vs styled components for styling,...

2.2.1. Using create-react-app to scaffold a new React project

Creating a new react app is as easy as running one command, so let's get started. Open up your terminal and run `yarn create react-app pokedex --template typescript`, this will create a new directory pokedex and populates it with a default react project structure and files. If all went well you should see output similar to this

```
Success! Created pokedex at
Inside that directory, you can run several commands:

  yarn start
    Starts the development server.

  yarn build
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd pokedex
  yarn start

Happy hacking!
Done in 45.15s.
```

Inside your pokedex folder you should have a file structure that looks like this:

```
pokedex
|_ public
  |_ ...
|_ src
  |_ ...
|_ package.json
|_ tsconfig.json
|_ README.md
```

Let's quickly discuss the structure and some of the files in our pokedex folder:

- **public**/ this folder contains our static files like index.html, favicon, etc.
- **src**/ this folder contains all your source that makes up your application.
- **package.json** this file contains the details of your project including but not limited to dependencies, tslint config, build scripts,...
- **tsconfig.json** in order to compile typescript we need a tsconfig file telling how to compile the project.

So now that we have our project setup under our belts, let's get started by creating our first component.

II. Components & Containers

In the next chapters we'll focus on building the UI with React, we'll do so by implementing the component/container pattern. By doing so we'll make a clear seperation of presentational components and logical components.

NOTE	*Seperation of Concerns* is a design principle for seperating your code into distinctive sections or concerns. In our case presentational vs logical.

Chapter 3. First Component: PokeListItem

Before we start building our first component we need to make a clear distinction between representational components and logical components.

- **Presentational Components**
  - Main concern: how things look
  - Usually contain both DOM markup and styles of their own
  - Have no dependencies on other part of the application like state
- **Logical Components**
  - Main concern: how things work
  - Is composed of presentational components, can but mostly don't have DOM markup
  - Loads data from the store and provide it to the presentational components

3.1. Component: PokeListItem

The first component we are going to create is a PokeListItem. This is just a styled button that handles an onClick event.
Since this is a presentational component it needs to receive all it's props from the Container above and bubble it's events upwards.
We'll start by creating this component with basic html styling, afterwards we'll refac this to an implementation with styled-components.

3.1.1. Styling with css

- Create a new folder `src/component/PokoListItem`
- Create a new file inside this folder `index.tsx` with following content

```
import React, { Component } from 'react'; ①
import 'pokelistitem.css'; ②

export interface PokeListItemProps { ③
    sprite: string;
    onClick: (event?: MouseEvent) => void ④
}

class PokeListItemComponent extends Component<PokeListItemProps, {}> { ⑤

    render() { ⑥
        return (
            <button
                role="button"
                class="pokelistitem"
                style={{
                    backgroundImage: this.props.sprite
                }}
            >);
    }
}

export default PokeListItemComponent; ⑦
```

① foobar

② foobar

③ foobar

④ foobar

⑤ foobar

⑥ foobar

⑦ foobar

- Create a new file inside this folder `pokelistitem.css` with following content in it.

```
.pokelistitem {
  width: 120px;
  height: 120px;
  border: none;
  border-radius: 5px;
  box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.16);
  margin: 10px;
  background-position: center;
  background-repeat: no-repeat;
}
```

3.1.2. Styling with Styled Components

3.2. Visual testing & documentation with Storybook

3.2.1. Setting up storybook

3.3. Quality through testing

3.3.1. Testing components with Jest

3.3.2. Snapshot testing with Jest

III. State management & Middleware

Chapter 4. redux stuff