DevGeekWeek 2018

Extreme SPA with React

By: Yariv Katz

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

Hello from Nerdeez!

Who are we

- We are Nerdz ;)
- We give consulting and teach new technologies to companies
- We release software products for Software developers
- Yariv Katz - CEO & Founder of Nerdeez
- Email: yariv@nerdeez.com
- LinkedIn: https://il.linkedin.com/in/yariv-katz

DevGeekWeek 2018

What's cooking today?

- What is SPA
- What is React
- What is Angular
- Angular VS React - What do I choose
- Tools we need before starting - NPM
- Hello World with React
- The smallest building block - React element
- A larger building block - React Component
- What is Redux
- How to combine React and Redux

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

What is SPA

- Web users today have high standards for web pages they visit
- They expect the page to load quickly even on their mobile phones
- They expect the site interaction to be really fast and responsive
- Today when building a web site we consider users entering our site like they are opening a desktop application
- at first we load the application and then every additional screen and resource we need is loaded via ajax
- We no longer ask for a full page only the data we need
- What are the disadvantages?
- What are the challenges?

What is React

- React is an open source JavaScript library
- Used for building UI components.
- Maintained by Facebook
- The main idea: You create component with a state, when state is changed the component will rerender.
- Together with Application Architecture Design like Flux or Redux becomes a powerful tool for creating complex web applications

DevGeekWeek 2018

What is Angular?

- SPA Framework
- Built with Typescript
- Open source and maintained by Google
- New version released in September 2016 and new version is issued approx every 6 months
- Together with React the two of them are becoming the most popular choices of SPA solution

So what do I choose Angular or React?

- There is no really wrong choice, both technologies are great!
- Both can be easily work with Redux
- React is less opinionated then angular
- React performance is better
- Angular typescript is an advantage for angular
- No templating language for react is an advantage for react
- React Native is more mature and better then the equivelant NativeScript

React building blocks

- In React we are not creating DOM elements. we are create JavaScript Objects which are cheaper
- **React.createElement(<tag name>, <attributes>, <text>)**
- From the building blocks react will create a Virtual DOM
- using diff algorithm react will detect changes between the Virtual DOM and the actual DOM and update the dom with minimum changes

ReactDOM

- ReactDOM is used to connect a React element (or group of react elements) to the DOM
- **ReactDOM.render(<react element>, <where to place>)**
- Let's try to use what we learned and create an hello world app with React

JSX

- Defining the UI with **React.createElement** can be daunting
- It's common to use **JSX** instead
- with **JSX** we can add an XML syntax to our JS files and it will be translated to **React.createElement**
- Browsers can't understand the **JSX** syntax so in order to run in browser we will have to transpile our code
- You can insert JavaScript expression in your JSX by wrapping it in **{}**
- attributes of elements can get a string or js expression
- Some attributes are different for example **class** is now **className**

Babel

- Browsers can’t understand JSX
- Browsers have limited support for ES6
- We need to transpile our code to browser readable code
- we will use **babel-standalone** and load the scripts with **type=”text/babel”**
- It’s not recommended to use **babel-standalone** when running your app, it’s better to use **webpack**
- you can also use **create-react-app** an npm package by facebook to bootstrap a react app

Props

- You can pass data from parent to child using properties
- properties will arrive at component: **this.props**
- With JSX properties are passed similar to attributes in HTML
- if attribute has string syntax like HTML then the attribute will be passed as string
- if the attribute has curly braces than it will be passed as JS expression
- To demonstrate let’s extend the hello world component to recieve property of the message to display
- Can we use it to pass data from child to parent?

State

- Each component in React has a private state
- You init the state in the constructor
- You have to use **this.setState** if you want to cause the component to rerender
- The **setState** can get a dictionary which will be merged with the current state
- The **setState** can also get a function which will be called with the current state and return the dictionary of the new state
- Let’s add a toggle visibility button in our hello world component

Events

- component can have also forms, buttons, and events that will change the state
- events are added like DOM events only **camelCased**
- events get curly braces js inject not like HTML where we pass string
- The an event will be passed to the function with the same specs as the DOM event - **SyntheticEvent**
- It’s common practice that events will be class methods
- Careful by default the context of the events won’t be bound to the class this
- Let’s add a button that will pop an alert with the message we got from the parent

Forms

- Often in our app we will need to place forms and grab user input
- ref is used to create a refrence to DOM element or React Component
- ref accepts a function that will be called with DOM element or React class instance
- This function will be called after component is mounted
- It's common usage to use ref to grab input values
- Let's expend the hello world component and add a form with text input and a list that will display the things we type

State Problem

- States in React Component are private to the component
- Sometime we need to transfer the state along multiple components across our app
- React Components are the UI but we need some sort of architecture to organize our application
- MVC is easy to maintain on small scale applications, but it’s considered as a mess for big applications

redux

State manager

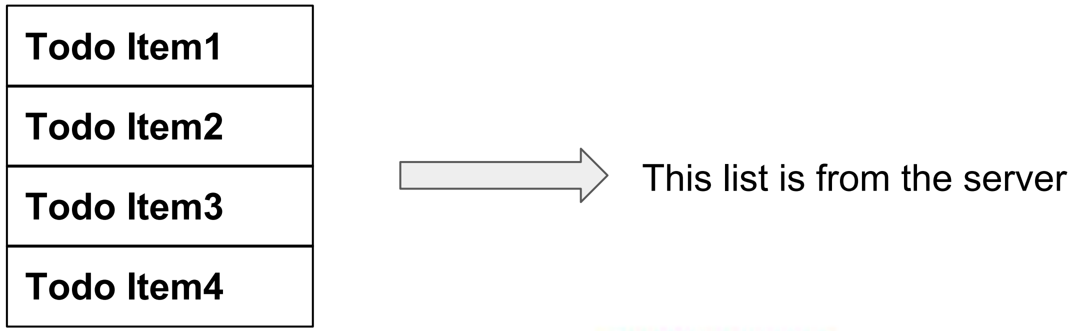
DevGeekWeek 2018

What is a State

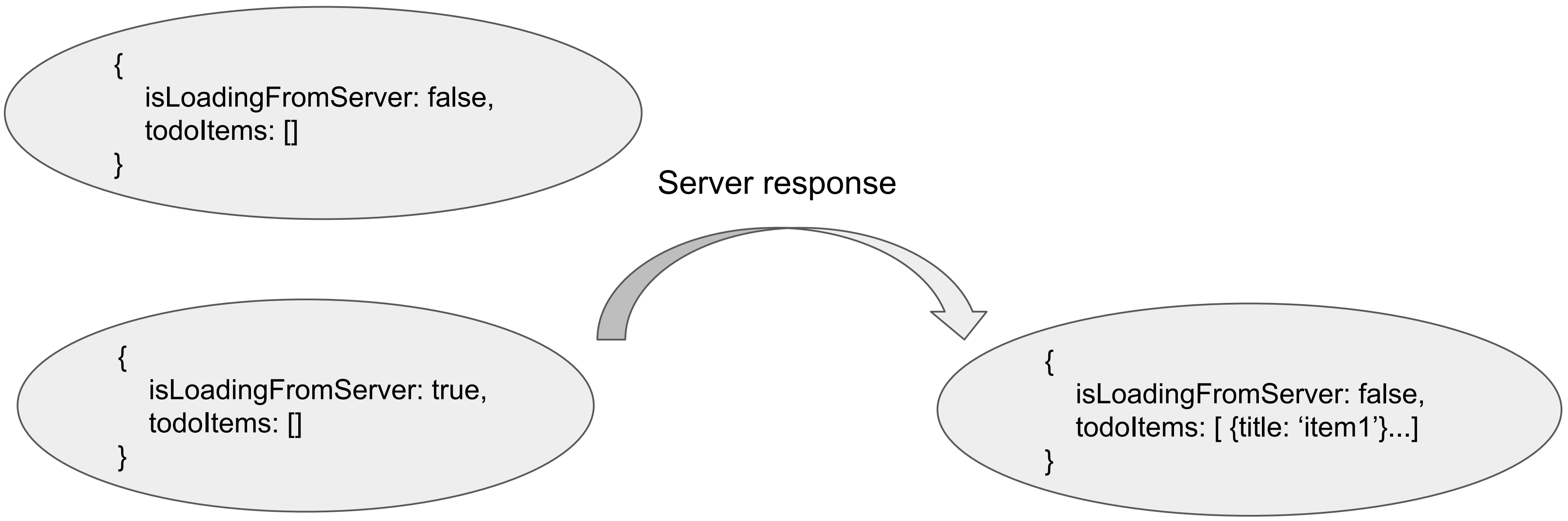
- in college at automation course we learned about Finite-state machine (FSA)
- FSA is a mathematical model of computation
- FSA can be in one state out of a group of final states at any given time
- FSA can transition to a new state based on certain change in input
- Can we look at our frontend application as an FSA?
- What can be the cause of a transition in state in a frontend web app?

Todo Application

- Let’s try to create a todo application
- in our todo app we have a component that displays the list of todo items
- at the beginning the list is empty and we query the server for the list of items
- after the server return our response we display the list of tasks
- the todo list component looks like this:



The state of our app



```
{
    isLoadingFromServer: false,
    todoItems: []
}
```

```
{
    isLoadingFromServer: true,
    todoItems: []
}
```

```
{
    isLoadingFromServer: false,
    todoItems: [ {title: ‘item1’}...]
}
```

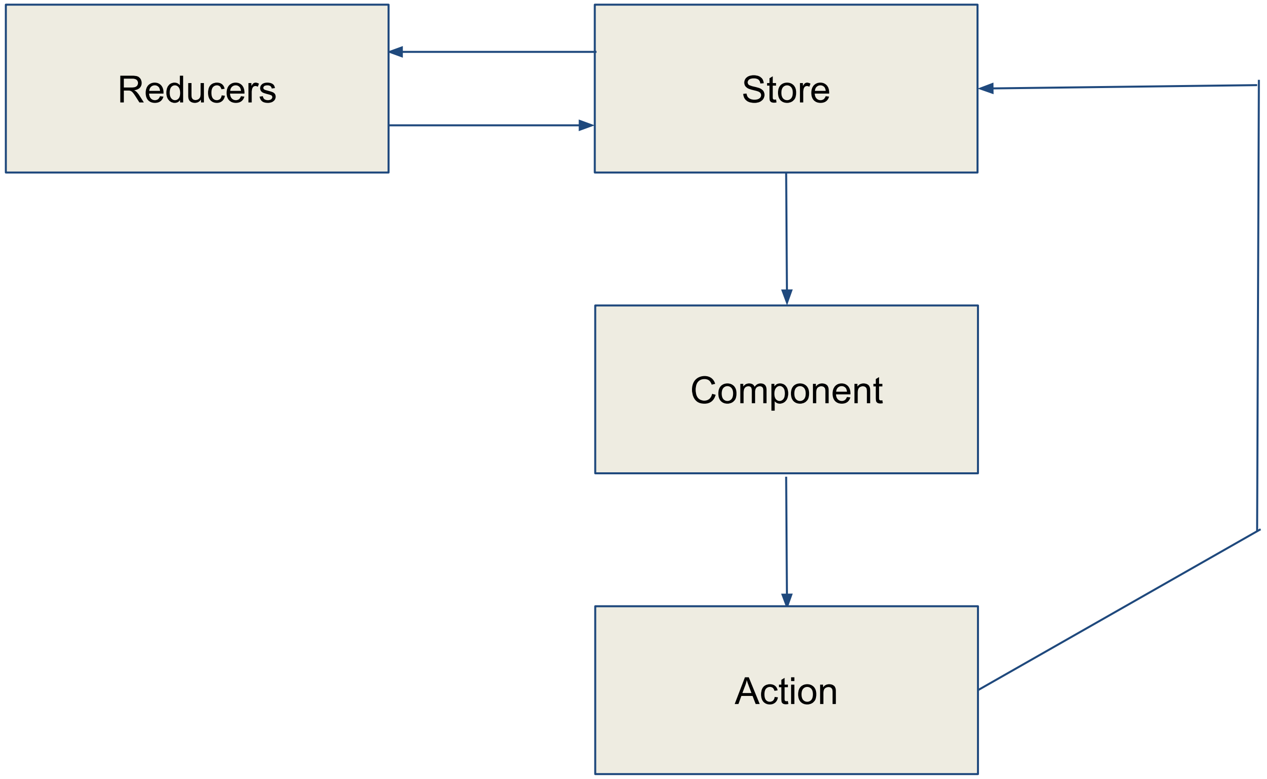
What is Redux

- predictable state container for JS apps
- using redux we have a single object holding the state of our app which is called **store**
- the state in the store can be an Object, Array, Primitive but it's highly recommended that it will be immutable
- The only way to change the state is by calling a method on the store called **dispatch**
  - **store.dispatch(action)**
- An action is a simple object describing **what happened**
- The store decides how the state will change using a pure function called reducer
- The reducer answers the questions **How does the state change?**
  - **(state, action) => state**

Why Redux

- State in React components are private
- Great way to pass information across the entire app
- performance… components are re rendered only if the part of the state where they are connected to is changed
- single store holds the entire state of the app
- state is immutable
- Easy to test components
- great community

Redux Architecture



Redux Core

Redux core concepts are:

- Actions
- Reducers
- Store

Let’s try to demonstrate redux usage using the state of our todo application

Actions / Action creator

- action is a simple JS Object
- the action needs to hold the following information
  - identifier of the action (string)
  - data associated with the action
  - example: **{type: 'SET_LOADING', payload: true }**
- action creator is a pure function which returns an action

```
function setIsLoading(isLoading) {

    return {type: 'SET_LOADING', payload: isLoading};

}
```

- What action creators do you think our todo app requires?

Reducers

- reducers are pure functions
- reducers get the current state and action as arguments
- the reducer need to decide based on the action and the current state what the next state will be
- the state has to be immutable to guarantee that no one will mutate the state in the reducer we will use a library called **immutable.js**
- when our state grows if we use only a single reducer the reducer will be huge so it is better to split the state to sections and also split the reducers
- redux has a function called **combineReducers** to help us split the app and make each reducer in charge of certain section of the state
- Before we dive into reducers let’s do a small intro on **immutable.js**

Todo Reducer

- Our todo app will contain a single reducer
- When initiated redux will call the reducer with the state undefined so we can utilize this fact to set our initial state
- let’s create the reducer for our todo app

Store

- there is a single store in redux application
- the store holds the state of our app
- to create the store we use **createStore**
    - **createStore(reducers, initialState)**
- we need to pass to the **createStore** the combined reducers and optional initial state
- Let’s create our store

Redux Summary

- we covered the core concepts of redux
- redux has it’s core package and also framework specific packages to connect it to different frameworks
- the next lesson will focus how to combine redux with angular and the benefits of using redux with angular

React & Redux

Connecting react to redux

Todo application

- We will try to create a simple todo application using react and redux
- our app will contain a form to create a new todo item
  - todo item contains title and description
- our app will also contain a list presenting all the items
- how many components do we have?
- how will our state look like?

Task Model

Task is a class containing the following properties:

- title : string
- description : string
- when : Date

Let’s create our task model.

TaskForm

- Our todo application contains a form to create a new task
- that form has a input for title
- textarea for description

TaskList

- The TaskListComponent will contain a list of the tasks we created

TaskList

- The TaskListComponent will contain a list of the tasks we created

Actions

- What action creators will our app have?

Reducer

- What is our initial state?
- Let’s create our reducer

Store

- let’s create our redux store
- we need to pass the store the reducer we created

connecting state and actions to components

- our components need to be connected to the state
- the components need to dispatch actions to the store
- We can use **ReactRedux.connect** decorator to achieve this
- First argument of **connect** is a function that is called with the state and that function should return the state items that will be sent to the component props
- Second argument for **connect** is a function that is called with the **store.dispatch** and should return functions connected to dispatch and action creators, that will be moved to the component props

Provider - wrapping our app

- We need to render all our components to the DOM
- We also need to provide our store to all the components
- For that we can use the **ReactRedux.Provider** and pass the store as prop to the **Provider**

DevGeekWeek 2018

Summary

- Use React to build UI components
- Use Redux with React to build large scale applications
- UniDirectional data flow once you get used to it is far better than MVC
- The lecture aim was to cover React so to cover the main aspects some shortcuts/bad practices had to be used:
  - Use Webpack
  - Use Decorators for connect
  - Use Babel as wepack loader
  - use NPM
  - use ES6 Modules

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

DevGeekWeek 2018

Thanks :)
See you next year

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

DevGeekWeek 2018

תודה :)

נתראה בשנה הבאה

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company