REACT ESSENTIALS

Prashanth Puranik
Web Developer and Trainer

COPYRIGHT NOTICE

INSTALLATIONS

Node JS v20: https://nodejs.org/en/

Visual Studio Code: https://code.visualstudio.com/

ES2015+

Features from JavaScript version 6+

ES2015+

Basic JS including functions and objects

- Object.assign()

- Array forEach(),map()

ES6 features used heavily in React

- let, const

- String template literal

- Default values for function arguments

- Array and object destructuring

- Rest and spread, including object rest and spread

- Arrow functions

- Classes and inheritance

- Promises

- async..await

- Modules

https://javascript.info/

TYPESCRIPT

Typed Superset of JavaScript

TYPESCRIPT

- Primitive types
- Type inference
- Union type
- Type literals
- Type alias
- Type assertion (type casting)
- Array and object types
- Function types
- Interfaces
- Classes

https://www.typescriptlang.org/docs/handbook/intro.html

BEFORE GETTING STARTED

What is React?

Single Page Application (SPA)

Build tools

WHAT IS REACT?

- UI library for building frontend apps
- Free and open-source
- Handles DOM manipulations efficiently
- Generally used to build Single Page Applications (SPAs)

https://react.dev/

THE SINGLE PAGE APPLICATION (SPA) ARCHITECTURE

- App includes a single web page. The content changes based on the route.
- All HTML, CSS, JS assets are fetched on first page load
- Only data is fetched on need basis
- **Advantages**
    - Page navigation is faster
    - Less load on the server
- **Disadvantages**
    - First page load is slow

ROLE OF MODULE BUNDLERS IN SPA

- Babel transpilation
- ES2015+ features can be used without issue
- Supports JSX

- Bundling
- Must for SPA. Reduces page load time.

- Hot reloading
- Faster development

https://babeljs.io/

https://webpack.js.org/

https://vitejs.dev/

GETTING STARTED

Scaffolding the app

JSX

Component-based architecture

Props

React Bootstrap

SCAFFOLDING THE APP

- Scaffolding a React application using boilerplate code (**create-react-app**)

    npx create-react-app project-name

- Understanding the Project Structure and build process
    - Babel
    - Webpack
    - Webpack Dev Server

- Start the server

    npm start

https://create-react-app.dev/

JAVASCIPT XML (JSX)

- Easy way to create "React element" (UI definition)
- An HTML like-syntax (but is actually a non-standard extension to JavaScript)
- **Features**
    - Conditional expressions and hiding and showing elements conditionally
    - Rendering an array of React elements
    - Setting a key for efficient DOM re-rendering
    - Styling React elements

INTRODUCTION TO COMPONENTS

- Component-based architecture is a modular approach to building the UI
- A component is a small part of the UI

    Example: A button, a dialog box

- **Features**
    - Declarative
    - Reusable
    - Composable

DEFINING COMPONENTS

- Function syntax
- Class syntax
- **Which syntax to use?**
  - We prefer function syntax in modern React
    - as capable as class components
    - simpler to set up use
    - most importantly, it is easy to share functionality between different components ("custom hooks")

PROPS

- Taking input from outside the component using props
- Used to customize an instance of a component

COMPOSING COMPONENTS

- JSX makes it easy to put together components to form larger components.
- This way we build the UI incrementally

DESIGN LIBRARY

- UI/Design libraries provide useful UI components
- Create good-looking and functional UI with less effort

https://mui.com/material-ui/

https://getbootstrap.com/ (Underlying HTML/CSS/JS library for React Bootstrap)

https://react-bootstrap.github.io/ (This has the React components)

STATE, EVENT HANDLING & SIDE-EFFECTS

State

Hooks

Side-effects

Event-handling

Fetching data and handling state updates

Parent-child communication

STATE

- Data the component maintains
- Data that changes with time and affects the UI
    - **Example**: Page number in a list of search results, the results for the selected page etc. These change as the user interacts.

- **Managing state**
    - Function components: The useState hook
    - Class components: setState() method

HOOKS

- Set of methods available only to function components

- Provide capabilities to function components, which previously (before v16.8) only class components had

- Throw an error if used within a class component, or outside of a function component (i.e not during the rendering of a function)

- Introduced to simplify implementing components that share common functionality

- useState(), useEffect(), useRef(), useContext() etc.

SIDE-EFFECTS

- Logic in a component apart from generating and returning the UI
- Handling asynchronous operations during the lifetime of a component like fetching and displaying data from the backend

- **Managing side-effects**
  - Function components: The useEffect hook
  - Class components: lifecycle methods (componentDidMount, componentDidUpdate etc.)

ADVANCED HOOKS

- useCallback() – to wrap callback props (function props) and prevent unnecessary re-rendering (when combined with memo())

- useReducer() – alternative to useState() that makes code with complex state changes more readable

- useMemo() – to cache result of complex computation when calculation is based purely on state, props or context

BASICS OF EVENT HANDLING

- A separate handler - receives event object as argument
- Inline handler - to pass custom arguments

PARENT-CHILD UPSTREAM/DOWNSTREAM COMMUNICATION

- Shared state is maintained in the common ancestor
- State and callback functions are passed as required to children
- Communication from child to parent component using callback function passed as prop

FORMS AND FORM VALIDATION

Creating Forms

Handling form validation using React Hook Form

WORKING WITH FORMS AND VALIDATING INPUTS

- Two patterns exists for working with forms
    - Uncontrolled components
    - Controlled components – give more control over setting / resetting values, passing form state elsewhere etc.

- Using react-hook-form for handling forms and validations

https://react-hook-form.com/

ROUTING

Routing using React Router

ROUTING

- Handle navigation between "pages" - changes UI according to the "address" (path)
  - React Router
  - Popular router library
  - v6

https://reactrouter.com/en/main

USING REACT ROUTER (1/2)

- **BrowserRouter**
    - Refreshes child (usually App) when path changes ("address")
- **Routes, Route**
    - Used for conditionally rendering components based on path

USING REACT ROUTER (2/2)

- **Link**
    - Render links that does not cause browser to reload pages
- **NavLink**
    - Same as Link but highlights link if it matches page path ("address")
- **Handling dynamic path params**
    - Parts of the path used to match a component can be dynamic
    - /path/to/:dynamic/:part/in/the/route
    - useParams() helps extract the dynamic parts from within the matched component
    - useNavigate() for programmatic navigation

REDUX

Avoid props drilling and share state globally, in a predictable way

REDUX

- Redux is a predictable state management library for JavaScript apps
- Redux is independent of React and can be used with plain JS apps, Angular apps etc.
- Avoid prop drilling and manage global state
- Implements a slight variation of the Flux architecture

REDUX DATA FLOW

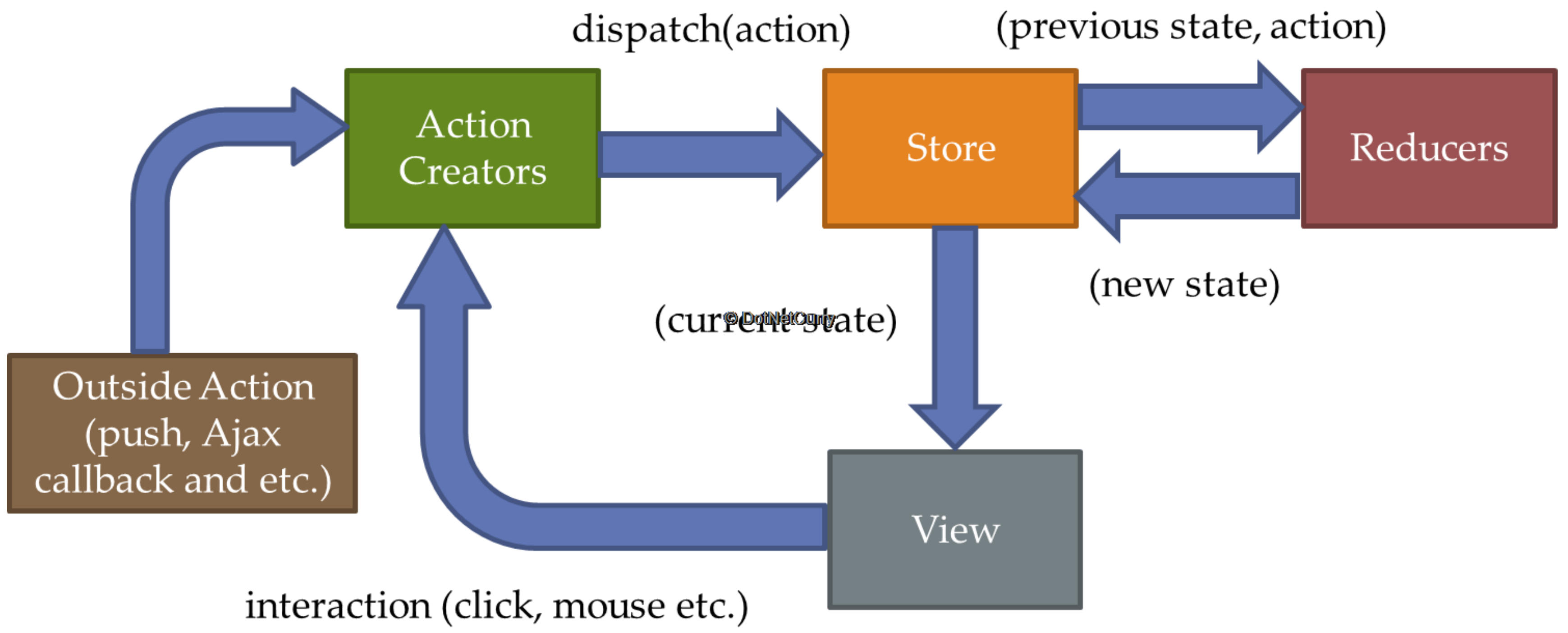

Image credits: https://www.dotnetcurry.com/reactjs/1356/redux-pattern-tutorial

REDUX – SUPPORTING LIBRARIES / TOOLS

- Redux toolkit - reduces boilerplate
- React Redux – Provider, useDispatch(), useSelector
- Redux Thunk – to simplify creating of middleware for async task

BUILD & DEPLOY

Configuration management

Build

Deployment

BUILD & DEPLOY

- Configuration management in a create-react-app application

- Creating a production build

    npm run build

- Deployment on Netlify

https://www.netlify.com/

THANK YOU

https://www.linkedin.com/in/prashanth-puranik

Learn more…

https://www.reactiflux.com/learning

https://redux.js.org/

https://react.dev

https://nextjs.org/

https://jestjs.io/

https://testing-library.com/docs/