REACTNATIVE

Presented by:

Farzad Akbarinejad

Ehsan Beheshtian

- about instructors:
  - Android developer
  - React native developer
  - Web frontend developer with ReactJS
  - Backend developer with Django
  - Python, Java, JavaScript
  - React native developer
  - Software Designer and Software Test Designer
  - Web frontend developer with ReactJS
  - Backend developer with Django and NodeJS
  - Python, Java, JavaScript, C++

About Course

- Smart phone, a necessary tool
- Companies want to have mobile Application besides their services
- How to develop a mobile App
- Old fashion
- Web Applications
- Multiplatform frameworks

- Modules:
  - ReactJS
  - ReactNative
  - Redux
  - Advanced ReactNative
  - Graphql_Apollo

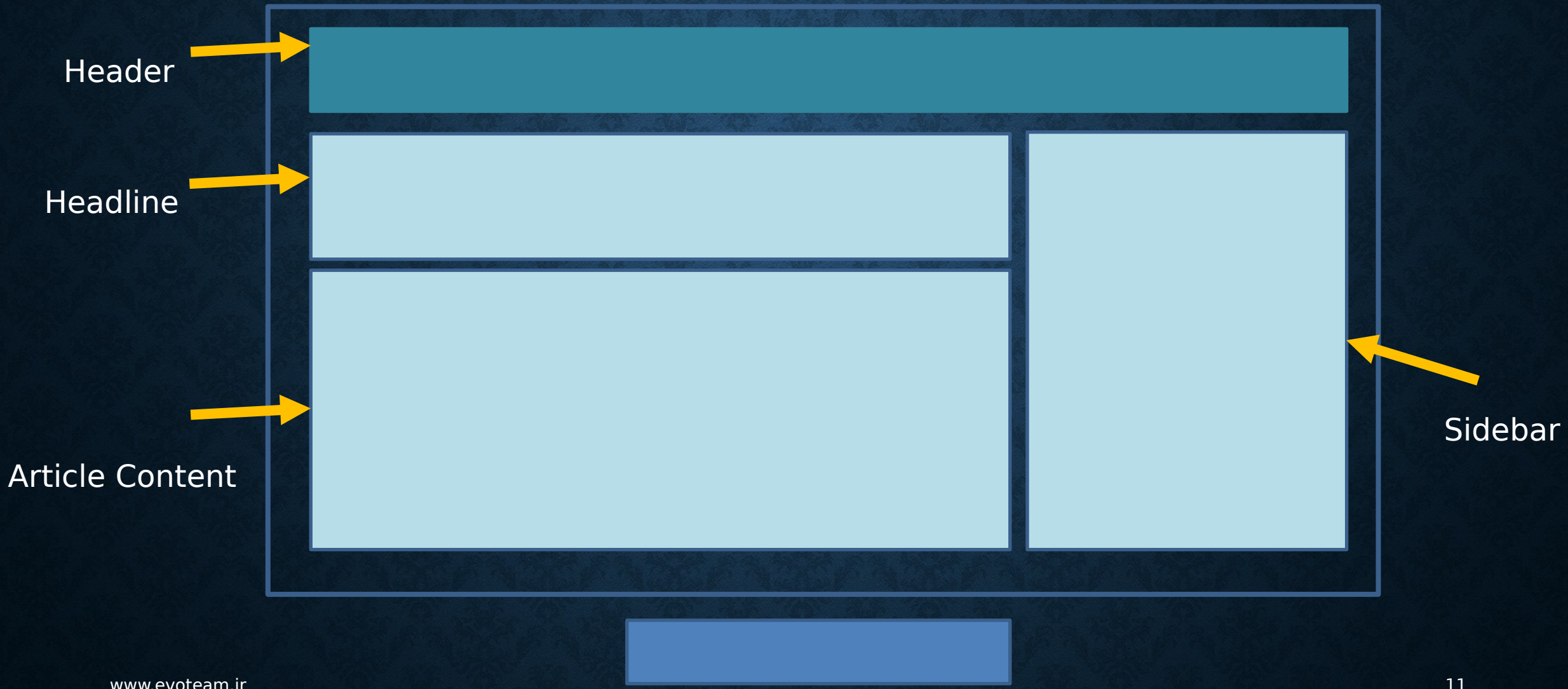
REACTJS

- titles
  - what is React
  - Building Components
    - Stateful
    - Stateless
  - Comparing Stateless and Stateful Components
  - Lifecycle Methods in ReactJS
    - rendering
    - updating

WHAT?

a JavaScript library for building User Interfaces

components?



WHY?

UI State becomes difficult to handle with Vanilla JavaScript

Focus on business logic, not on preventing your app fro exploding

Huge Ecosystem, Active Community, High Performance

plus

Framework creators probably write better code

COMPONENTS

- Let’s get our hands dirty…

Stateful vs. Stateless Components

Stateful (Containers)	Stateless
class XY extends Component	const XY = (props) => { ... }
✓ Access to State	✗ Access to State
✓ Lifecycle hooks	✗ Lifecycle hooks
Access State and Props via ‘this’	Access Props via ‘props’
this.state.XY & this.props. XY	Props.XY
Use only if you need to manage state or access to Lifecycle hooks	Use in all other cases

Component Lifecycle

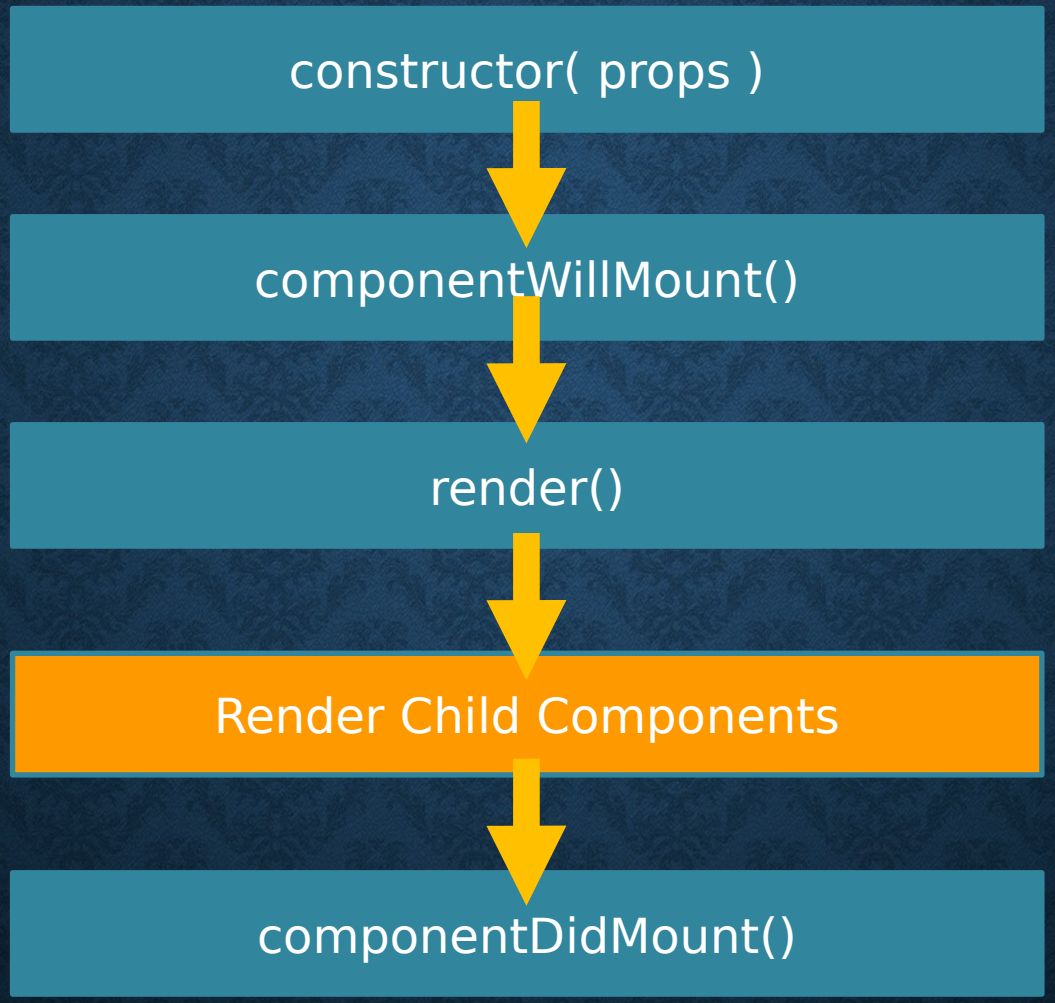
Only available in Stateful Components!

- constructor()
- componentWillMount()
- componentWillRecieveProps()
- shouldComponentUpdate()
- componentWillUpdate()
- componentDidUpdate()
- componentDidCatch()
- componentDidMount()
- componentWillUnmount()
- render()

Component Lifecycle - creation

Default ES6 class feature

Exists for historical reasons



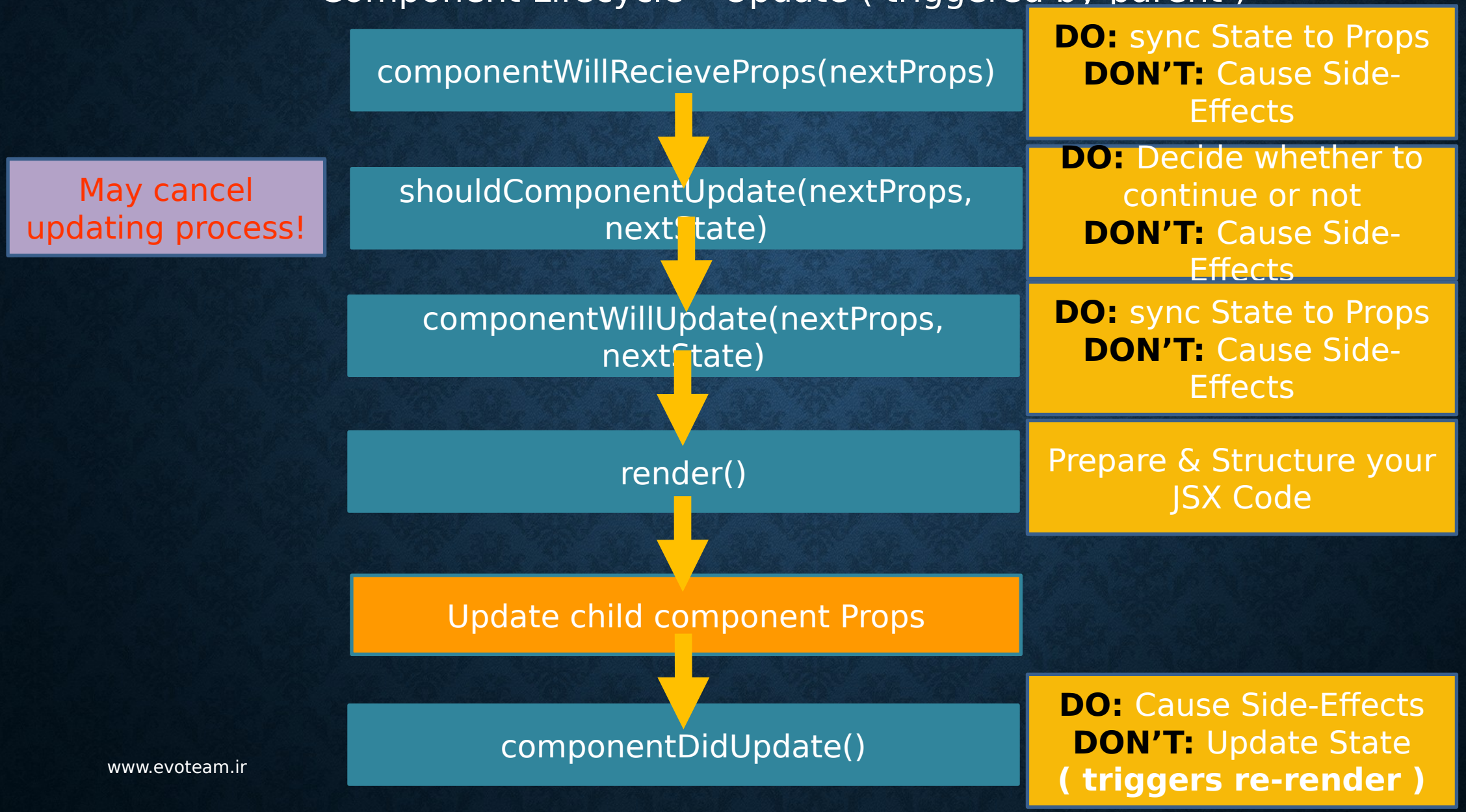
Call super(props)
DO: Set up State
DON'T: Cause Side-Effects

DO: update State, last – minute Optimization
DON'T: Cause Side-Effects

Prepare & Structure your JSX Code

DO: Cause Side-Effects
DON'T: Update State
(triggers re-render)

Component Lifecycle – Update (triggered by parent)



Component Lifecycle – Update (triggered by internal change)

May cancel updating process!

shouldComponentUpdate(nextProps, nextState)

DO: Decide whether to continue or not
DON'T: Cause Side-Effects

componentWillUpdate()

DO: sync State to Props
DON'T: Cause Side-Effects

render()

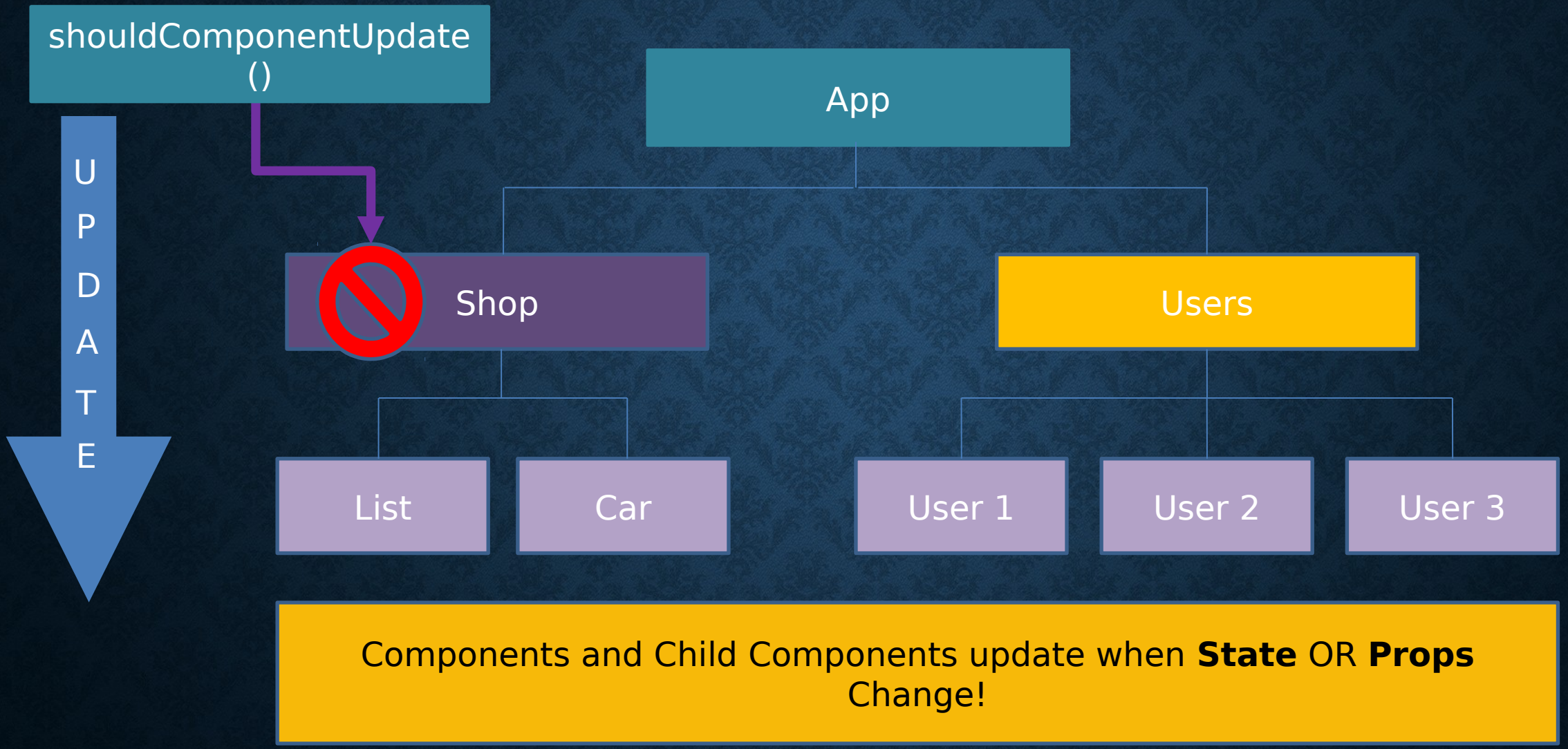
Prepare & Structure your JSX Code

Update child component Props

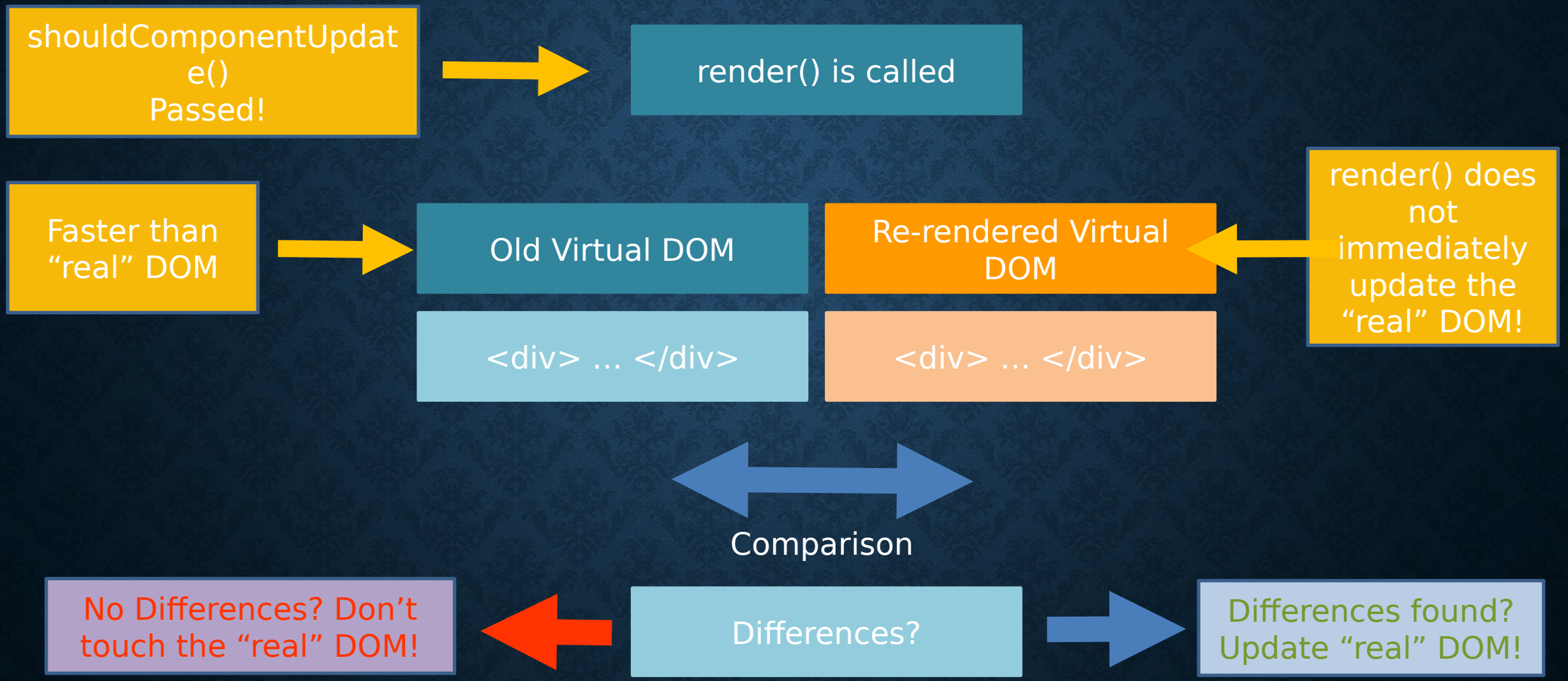
componentDidUpdate()

DO: Cause Side-Effects
DON'T: Update State
(triggers re-render)

Rendering & Updates



How React Updates The DOM



End Of Module 1

React

Questions?

REACT_NATIVE

- titles

  - Installing on Windows
  - Creating a Project
  - Getting to know the project folders
  - React VS. ReactNative
  - Building Components Using JSX
  - Building Reusable Components
  - Working With Lists
  - Linking between Applications
  - Debugging
  - New Error Handling Feature (React 16)

INSTALLING

- Java Sdk
- NodeJS
- Python
- Android Studio
- react-native-cli
- Running the project on android studio
- Creating android simulator
- Adding environment variables
  - JAVA_HOME
  - ANDROID_HOME: platform-tools

react-native run-android

React VS. ReactNative

React

- Knows how a component should behave
- Knows how to take bunch of components and make them work together

ReactNative

- Knows how to take the output from a component and place it on the screen
- Provides default core components (image, text)

Module 1 Overview

End Of Module 2

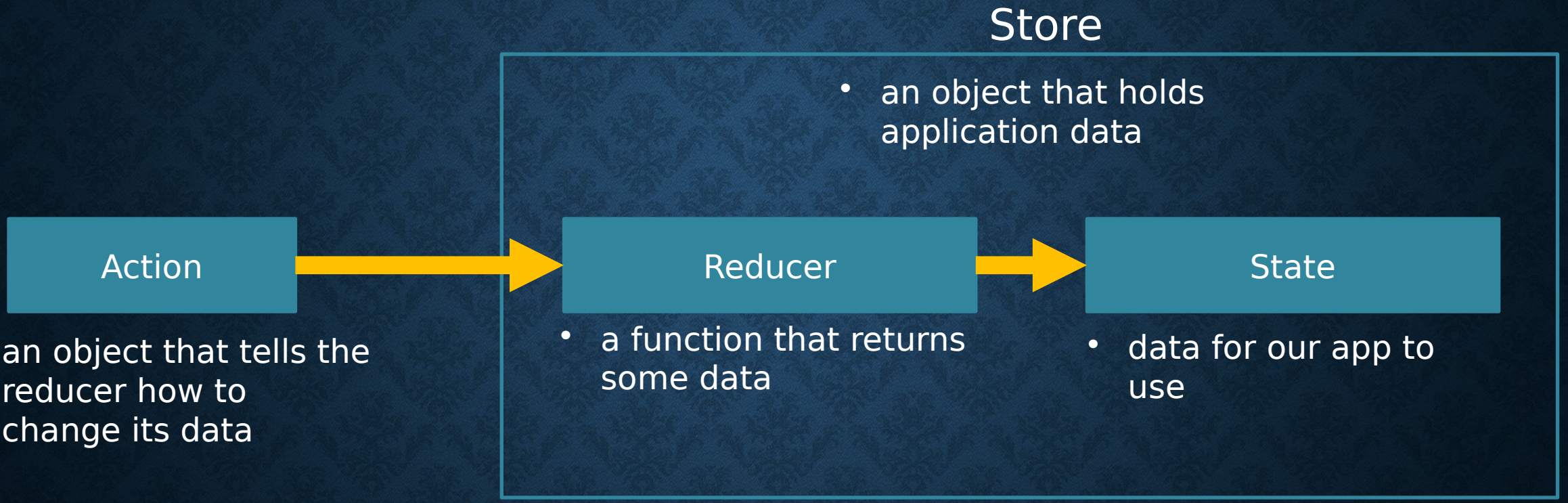
React Native

Questions?

REDUX

- titles
  - Getting to know Redux
  - What are
    - Actions
    - Reducers
    - State
  - How Redux works
  - When to / not to use redux
  - Pulling logics out of the components

- Example


Store
an object that holds application data
Action
Reducer
State
an object that tells the reducer how to change its data
a function that returns some data
data for our app to use

Store

- turn ‘asdf’ into an array of characters

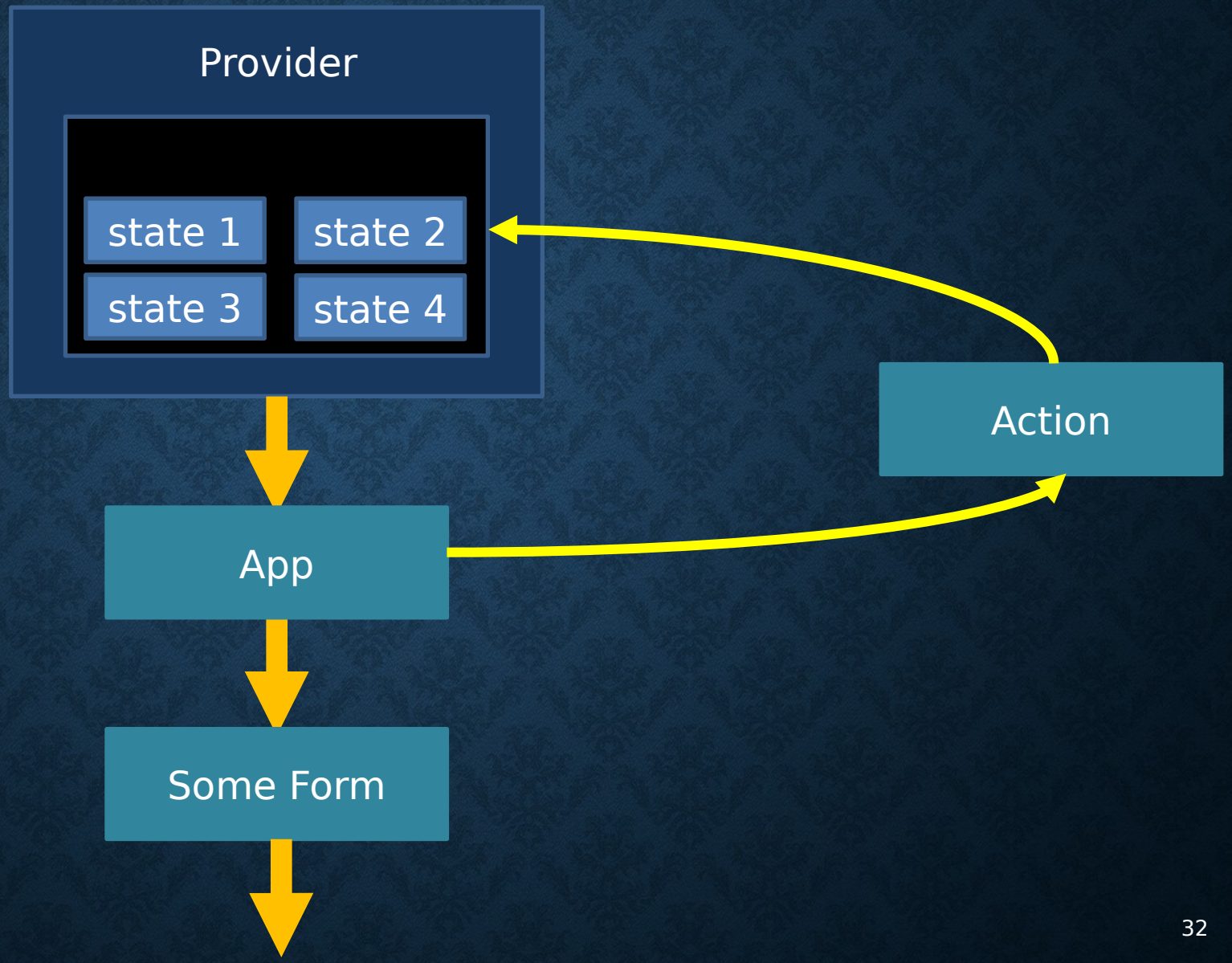
Action

- if the action’s type is ‘split’, I will take a string of characters and turn it into an array.

Reducer

- [‘a’, ‘s’, ‘d’, ‘f’]

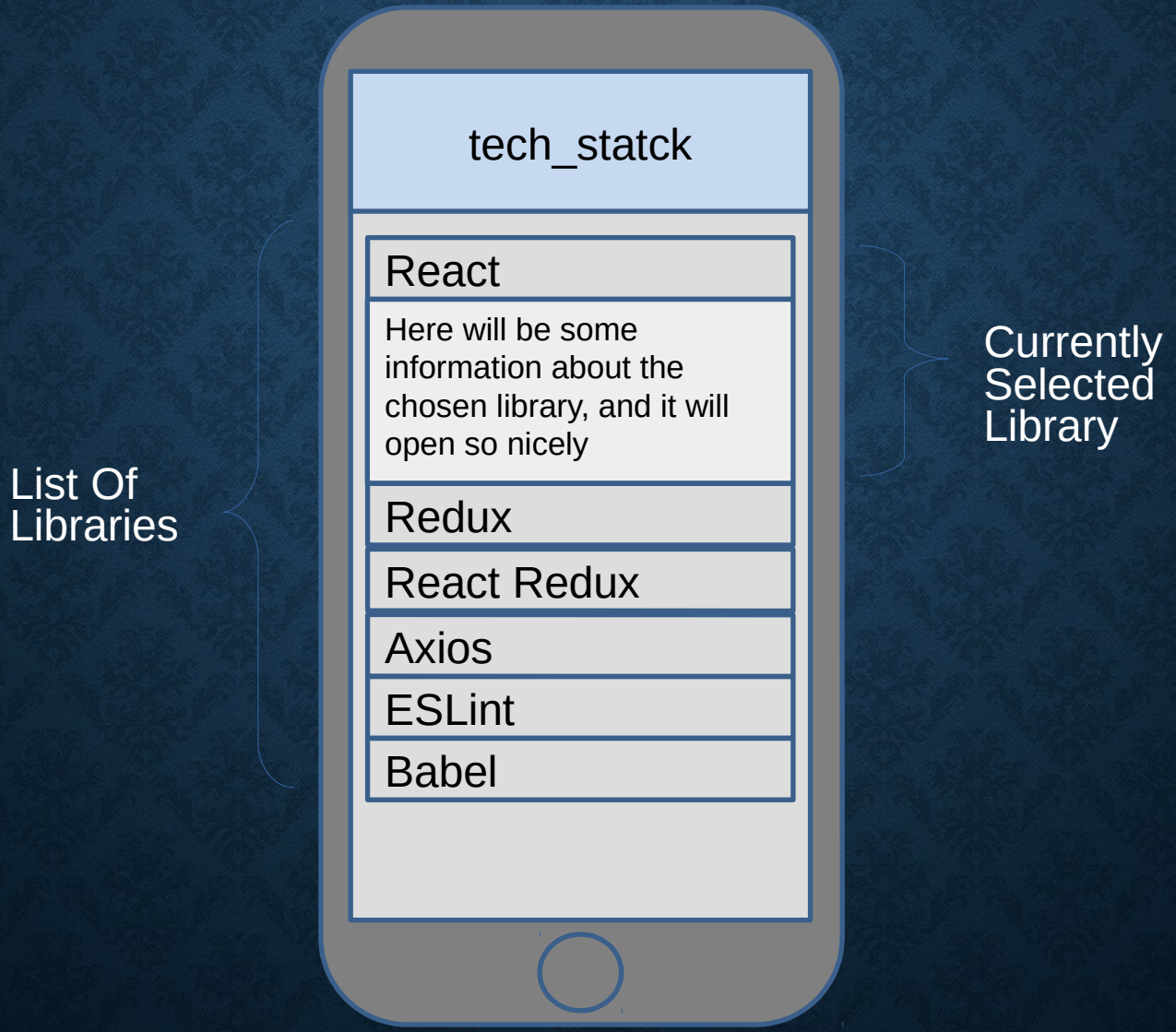
State


Provider
state 1
state 2
state 3
state 4
Action
App
Some Form

Types of State

Type	Example	User Redux?
Local UI State	Show / Hide Modal	Mostly handled within components
Persistent State	All Users, All Posts, ...	Stored on Server, relevant slice managed by Redux
Client State	Is Authenticated? Filters set by User, ...	Managed via Redux

Module 2 Overview



End Of Module 3

Redux

Questions?