

React

Jan.schulz@devugees.org

Agenda

1. Introduction
2. Set-Up
3. Components
4. JSX
5. Props & States
6. Functional Components
7. Stateful Components Lifecycle
8. AXIOS

1. Introduction

- React
 - Client Side JavaScript framework
 - Created & maintained by Facebook
 - Used to build dynamic user interfaces
 - Renders everything as a component
 - Often to referred as the „V“ in MVC

1. Introduction

- React ...
 - Makes JavaScript easier ... for complex apps
 - Organizes your UI
 - Reusability
 - Scalability & Efficiency
 - Lighter than most frameworks
 - GET A JOB

1. Introduction

- React is used in
 - Facebook
 - Facebook App (React Native)
 - AirBnB
 - Tinder
 - Udemy
 - Instagram
 - Yahoo Mail

1. Introduction

- Requirements:
 - NodeJS
 - Visual Studio Code
 - ES7 React/Redux/GraphQL/React-Native snippets
 - Chrome Extension
 - React Developer Tools
 - Redux Developer Tools

2. Setup

- create-react-app is a predefined Webpack config
 - `$ sudo npm install -g create-react-app`

2. Setup

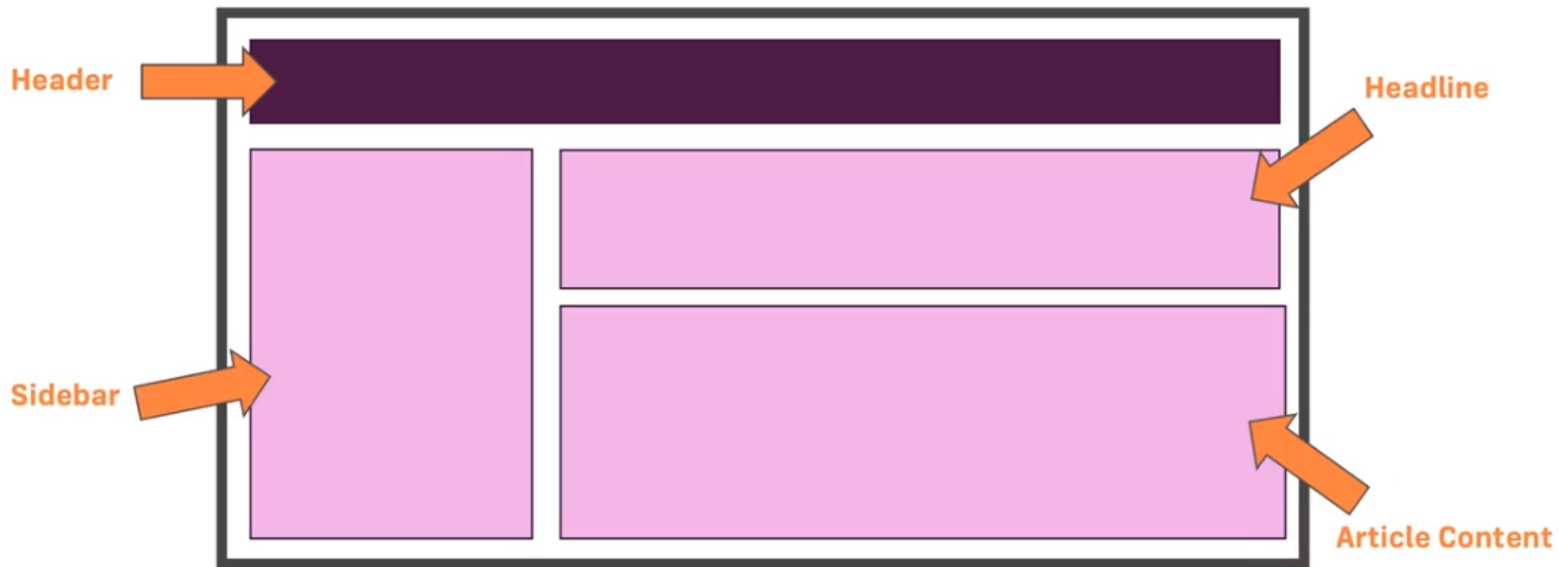
- package.json
 - Manifest that info about the application
- Public folder
 - Entry point to the entire application
 - In index.html, there is **root-div**, where React renders the entire app into

3. Components

- Thinking in React-Way

3. Components

Components:



3. Components

Components:

- Each component contains its separate container of code
- We **do not build** our web app as one bigger picture
- Why?

3. Components

Components:

- Each component contains its separate container of code
- We **do not build** our web app as one bigger picture
- Why? – It makes our code more manageable, more maintainable and more reusable
- A react component is like a custom HTML element

3. Components

Why React?

UI State becomes difficult to handle with
Vanilla JavaScript

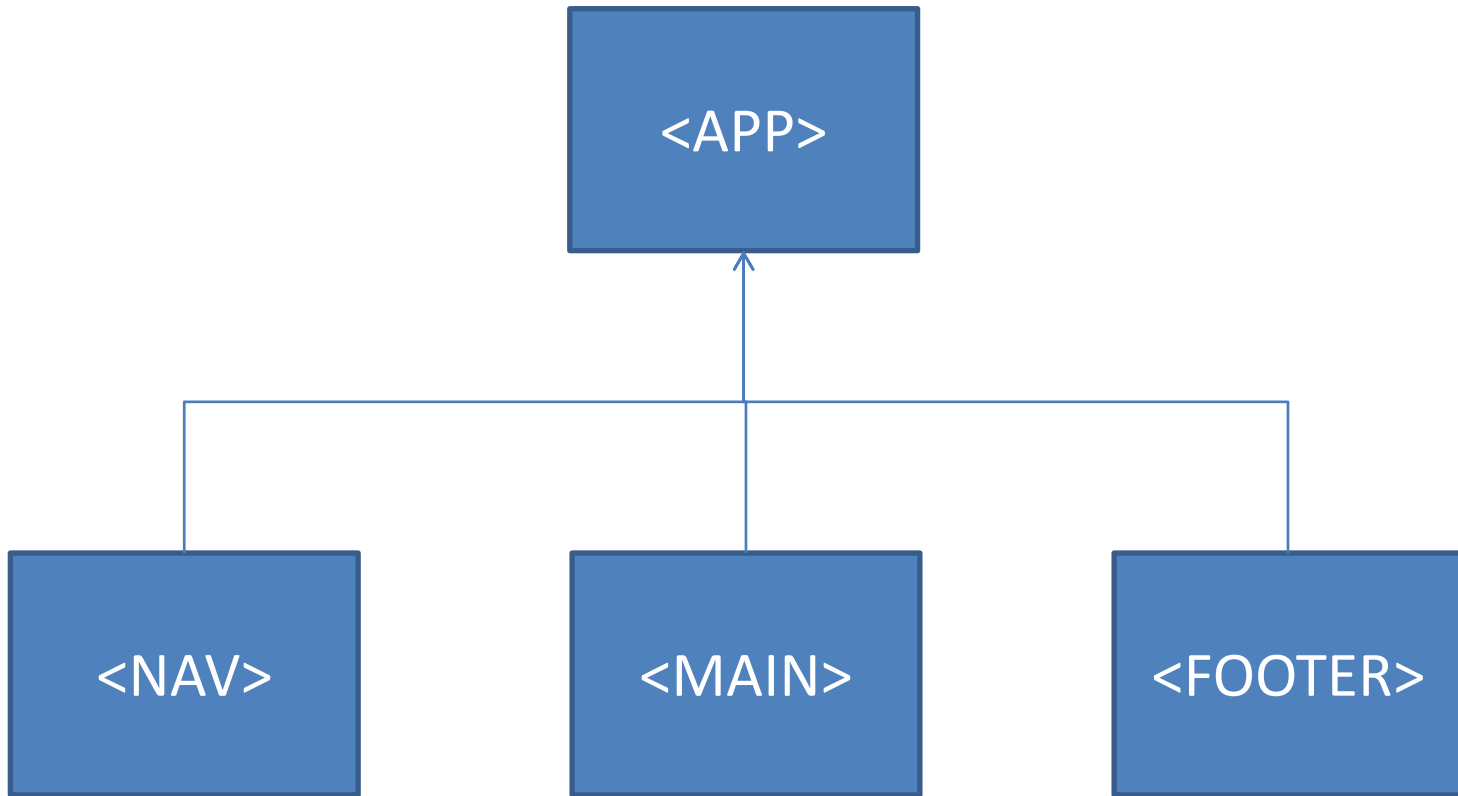
Focus on Business Logic, not on preventing
your App from exploding

Plus

Huge Ecosystem, Active Community, High
Performance

Framework
Creators
probably
write better
Code

3. Components



3. Components

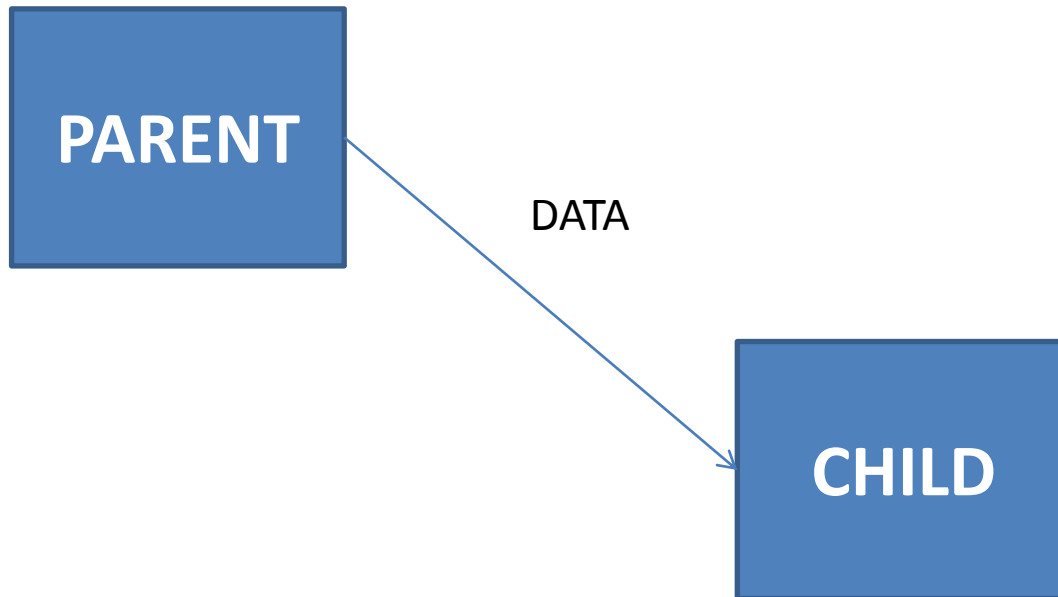
- Stateful Components
 - Have a state object
 - Return JSX
- Functional Components
 - **Do not** have a state object
 - Return JSX

4. JSX

- basically HTML in JavaScript
- Will be transpiled into JavaScript by Babel (as Webpack plugin)

5. Props & States

- Props
 - Data passed down from parent component to child component



5. Props & States

- State
 - Private data of a component

MyComponent

```
state = {  
  active: true,  
  x: 5  
}
```

6. Functional Components

- Functional Components are Components, that do not have a state or lifecycle methods
- Why?

6. Functional Components

- Functional Components are Components, that do not have a state or lifecycle methods
- Why?
 - **For layout purposes**, components do not need to have an own state
 - **For performance issues**, functional components do not inherit stateful component methods and do not need to be instantiated

Task

Create a React App „Fruit Manager“ where one can add, remove and filter fruits.

Note:

- Use both stateful and stateless components
- Use flexbox or grid for the arrangement of the items

Task

Add: Adds a new fruit with random color

Remove: Removes all fruits that contain the substring of the text input

Filter: Shows only the fruits that contain the substring of the text input

Click on X: Removes the fruit

Fruit Manager

Add

Remove

Filter

Apple X

Orange X

Mango X

Yellow X

Pine Apple X

Lemon X

Kiwi X

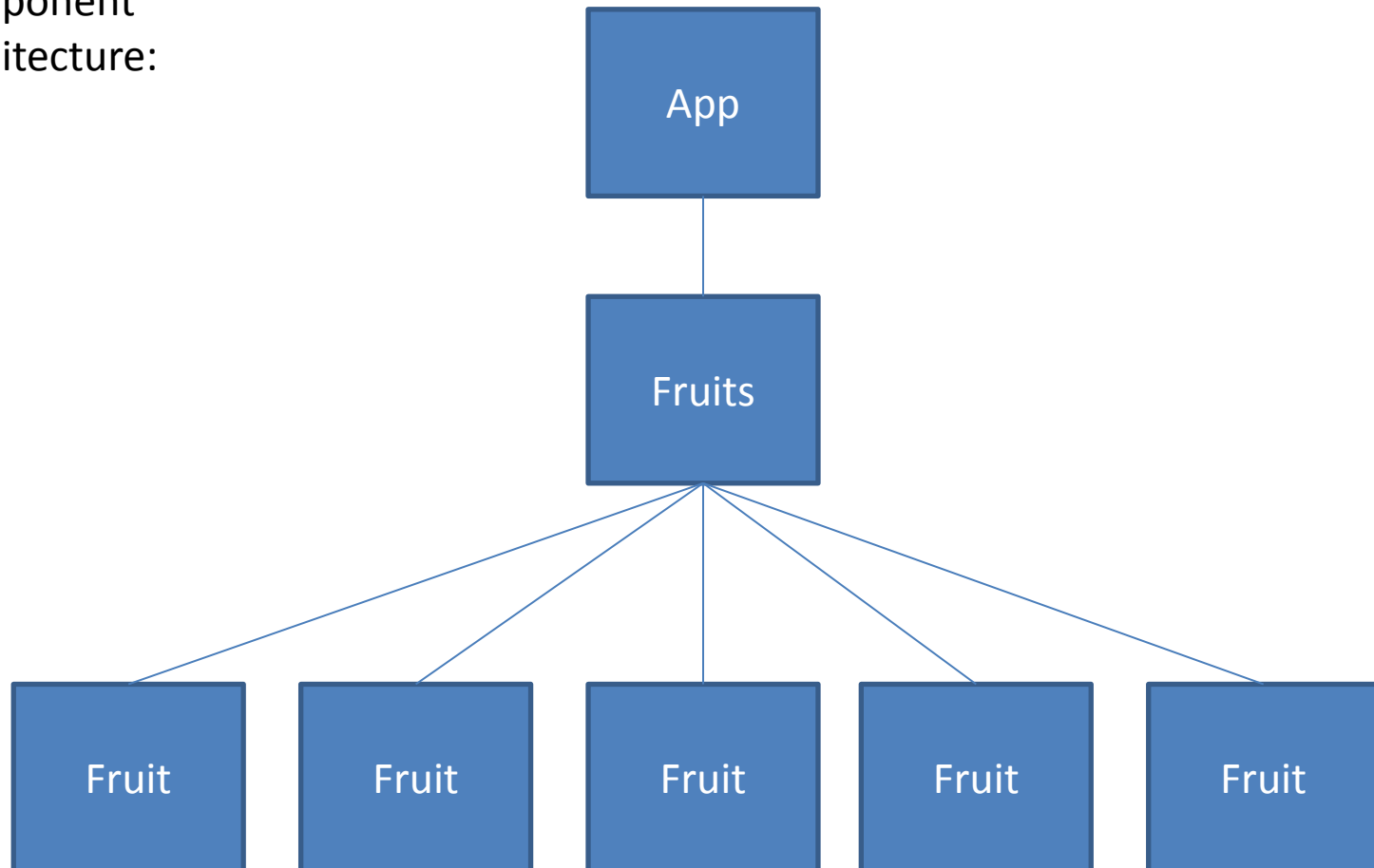
Apple X

Apple X

Melon X

Task

Suggested
Component
Architecture:



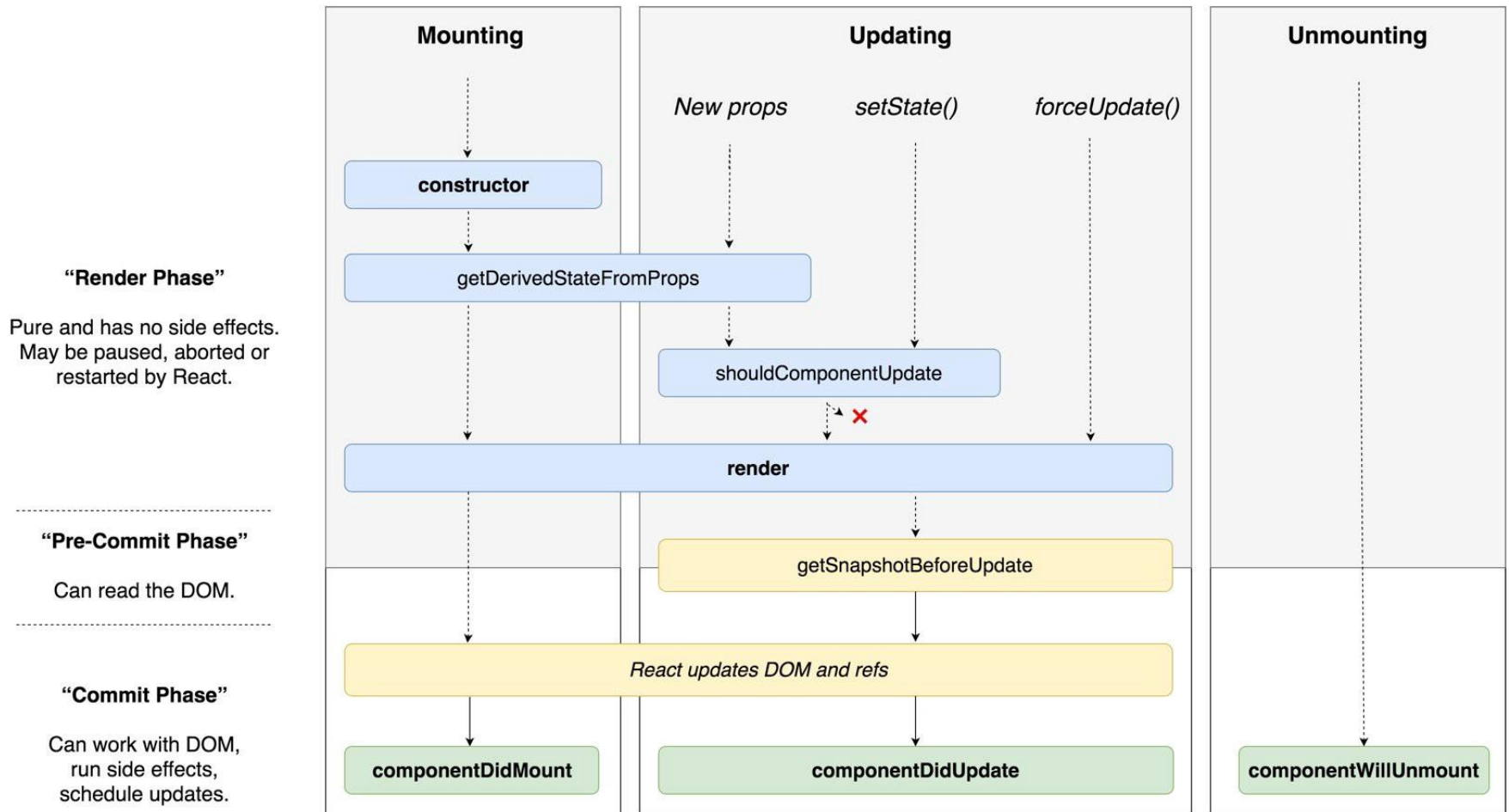
7. Stateful Component Lifecycle

- Each stateful component is
 1. Is born
 2. Lives
 3. Dies

7. Stateful Component Lifecycle

- Each stateful component is
 1. Is born = **mounts**, the parent container includes it in the virtual DOM
 2. Lives = **updates**, new props will be sent to the component or it decides by itself to set a new state
 3. Dies = **unmounts**, the parent container excludes it from the virtual DOM

7. Stateful Component Lifecycle



8. AJAX with AXIOS

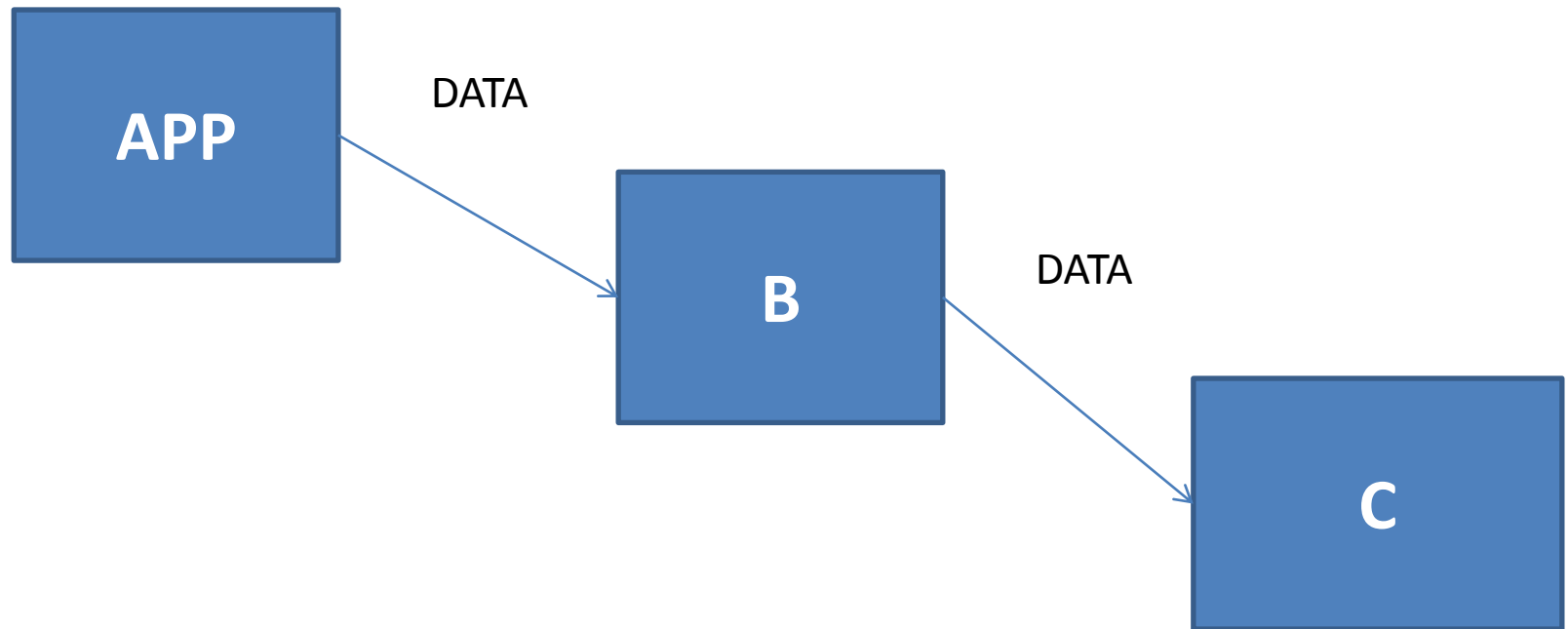
- AXIOS is a library that implements the functionality of AJAX using promises
- NodeJS Servers can be integrated into React Apps using a Proxy feature

9. Communication between multi-layered components

1. Props Drilling
2. Context API
3. Redux

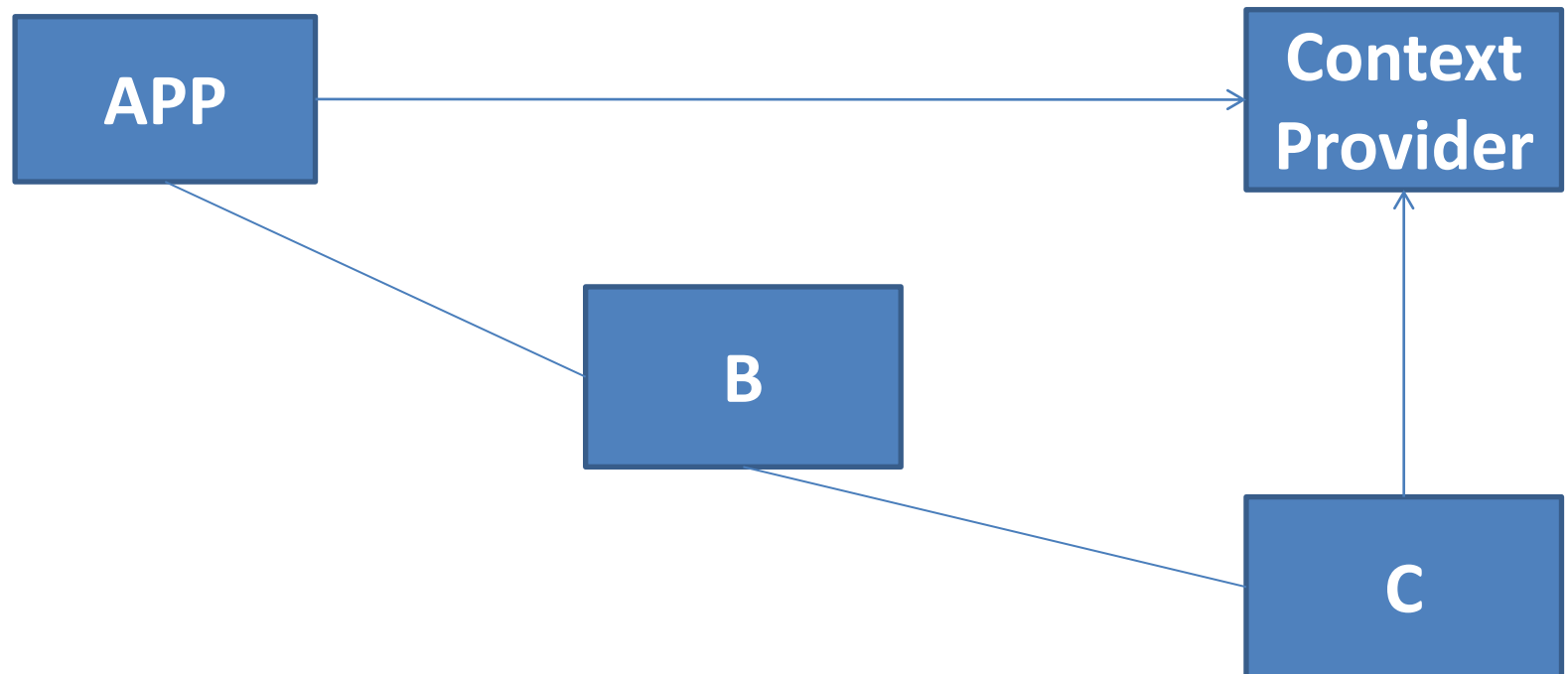
9. Communication between multi-layered components

1. Props Drilling = Passing data down child by child



9. Communication between multi-layered components

2. Context API = A component, that shares its one state with other components



9. Communication between multi-layered components

3. Redux = Multiple Shared States

