React

MU ENGINEERING

Manulife

Agenda

What is React

- Highlights
- Building blocks
- What’s excluded

Why React

- Because …
- Benefits

Agenda (continued)

Characteristics of React

- Virtual DOM
- One-way data flow
- JSX
- Components
- Hooks
- Render

Ecosystem

- Complimentary libraries
- Runtime
- Build
- Styling

Exercise

What is React?

What is React?

Highlights

- React is a **declarative** JavaScript **library** created by Facebook for rendering views in applications
- At its core, it uses **Virtual DOM** to process updates in the view
- It endorses a component based architecture, where an application is a tree of components
- It is not a MVC framework
- Used to build a **Single Page Application** (SPA)
- It is fast, simple and scalable

What is React?

Building Blocks

Components

- Are the core building blocks in React
- Follow strict encapsulation principles – each is independent & reusable

Composition

- Components can be composed of other components
- Oftentimes, React apps have a top-level 'App' component

Elements

- Components return elements
- They describe what should appear on the screen
- Virtual DOM updates the DOM to match the React elements

What is React?

Building Blocks

Props

- Components accept arbitrary inputs, called 'props’
- Props are read only, making components act like pure functions

State

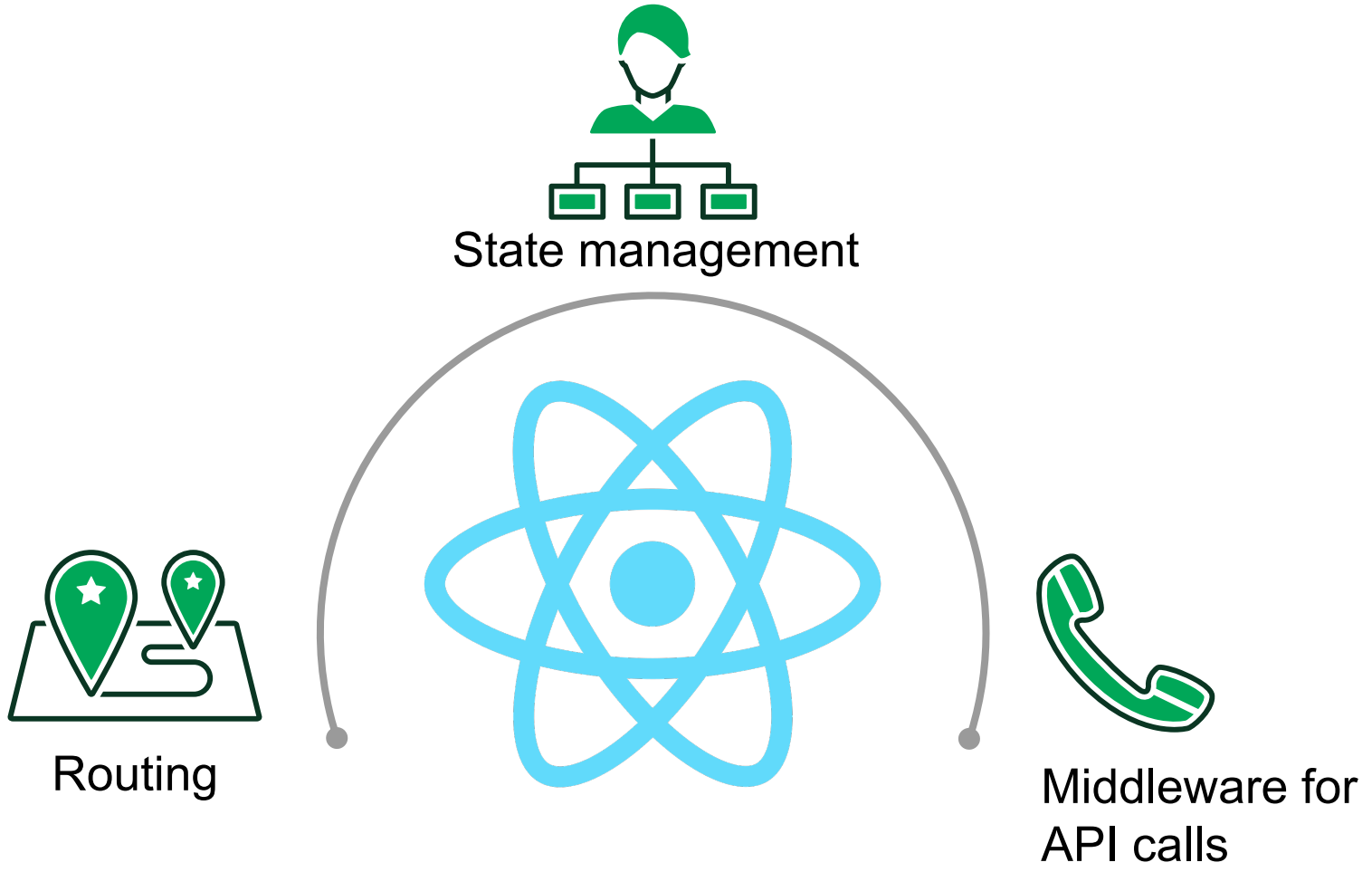
- Components can be stateful or stateless
- In a stateful component, state is local, accessible only from within that component

Hooks

- Let you use state and other React features without writing a class
- Enables reuse of stateful logic between components

What is React?

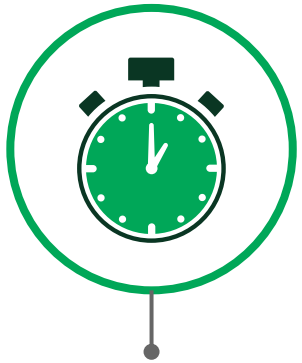
What's excluded?



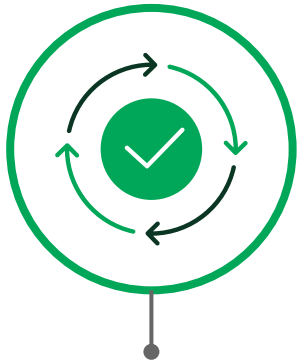
Why React?

Why React?

Makes JavaScript more structured, easier to write



Runs faster than JavaScript with HTML + CSS + jQuery / AJAX



Lightweight & flexible – developer chooses what fits best

Well-supported development platform & ecosystem

Continuously improved by Facebook and a broad community

Why React?

Benefits

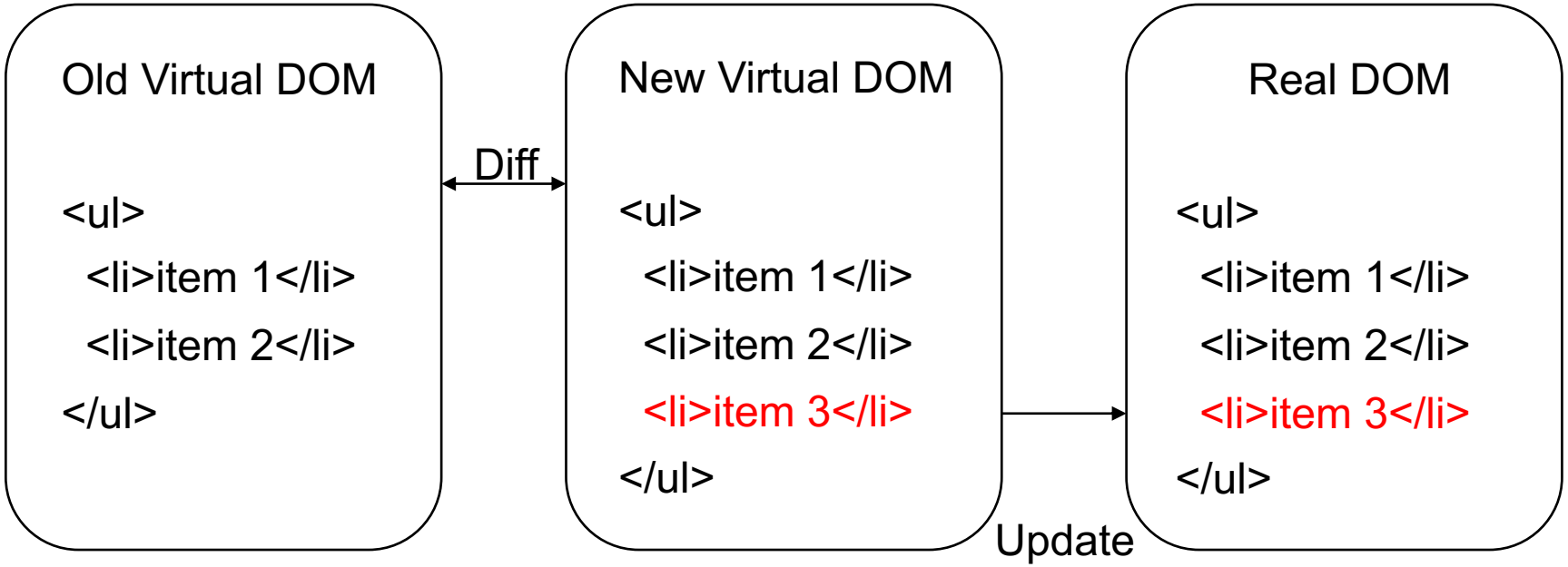
- Decoupling of front end and back end server side development
- Allows for “snappier” apps and rich user interactions
- Delegate business logic to browser and mobile devices to reduce the need for heavy servers
- Allow front end developers to focus on user experience and back end developers to focus on data and security
- Architecture is web and mobile friendly

Characteristics of React

Characteristics of React

Virtual DOM

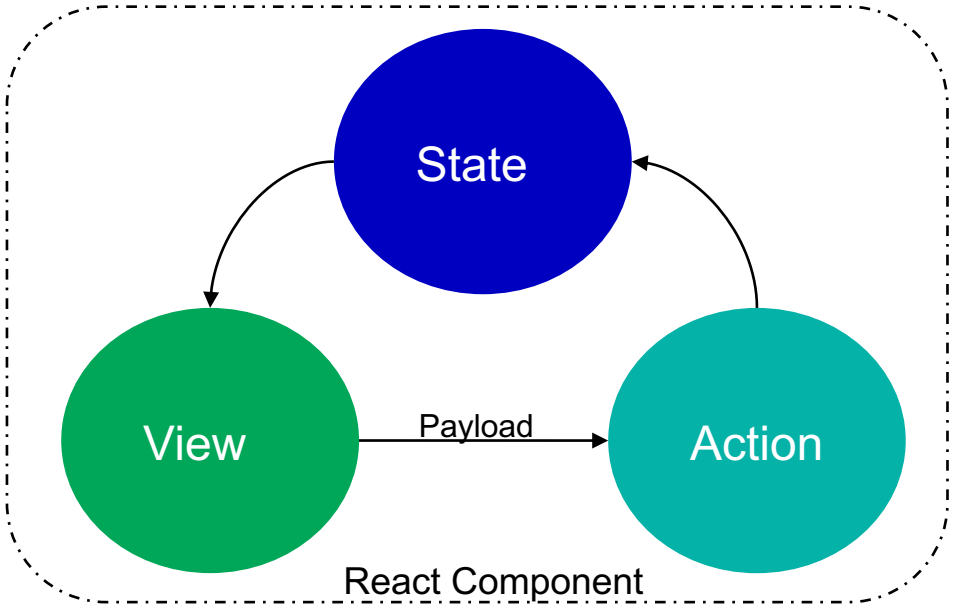
- React uses a virtual DOM to compare changes between render function calls in memory and makes minimal changes to the real DOM



Characteristics of React

One-way Data Flow

- Immutable data is passed to React components to be rendered
- No more two-way data binding
- Components cannot modify data being passed to it, but instead they can trigger actions to change component state
- Overall application data flow is done with other frameworks such as Redux



Manulife

Characteristics of React

JSX

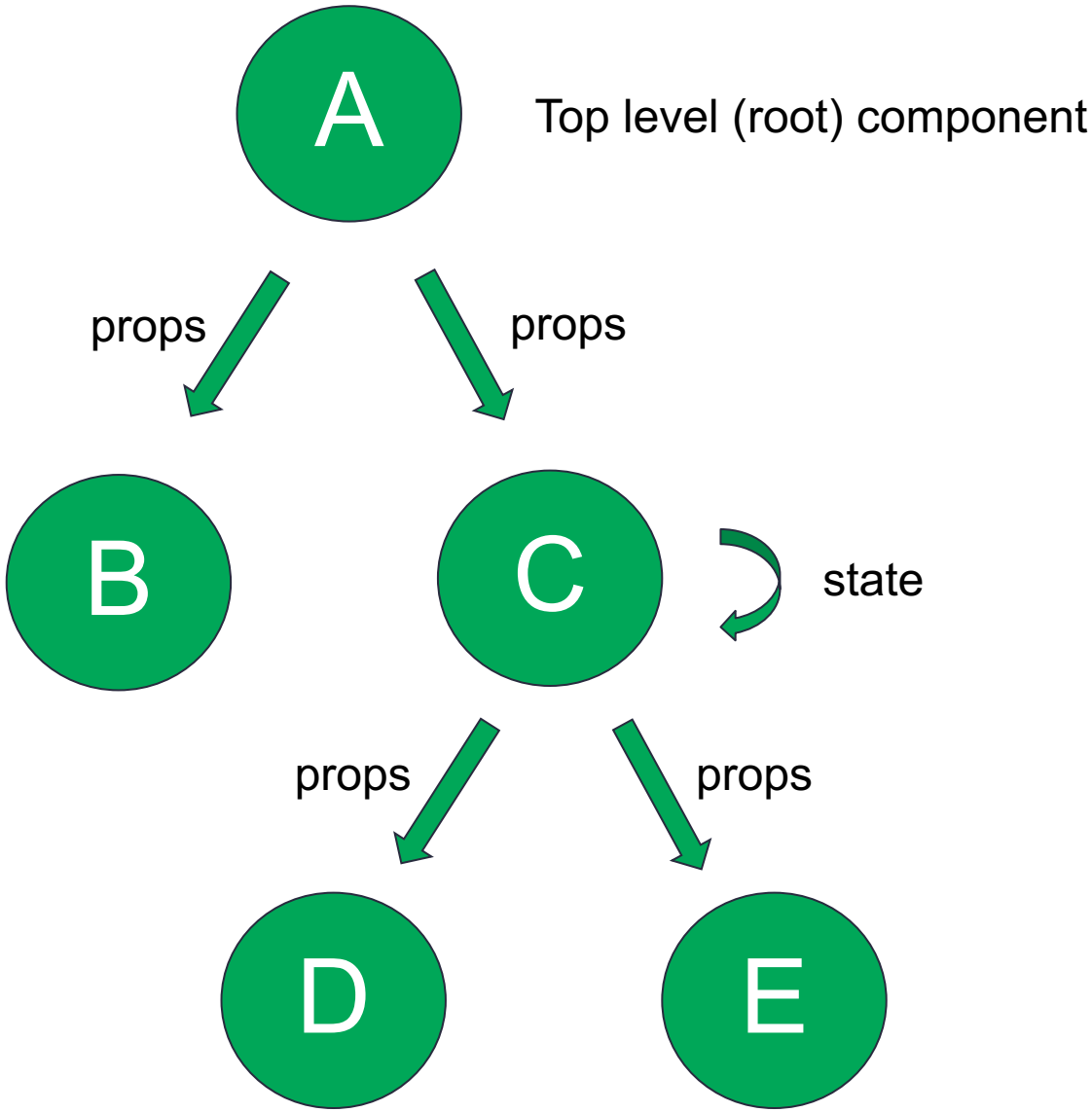
- React doesn't use HTML templates like other SPA frameworks
- Instead, it uses JSX, an HTML like extension syntax that gets processed into JavaScript
- JSX creates a development experience that is familiar without the performance limitations of HTML templates

```
// JSX
function JsxExample(props) {
    return (
        <div>
            <label htmlFor='example-text'>{this.props.labelText}</label>
            <input id='example-text' className='textbox' type='text' />
        </div>
    );
}
```

Manulife

Characteristics of React

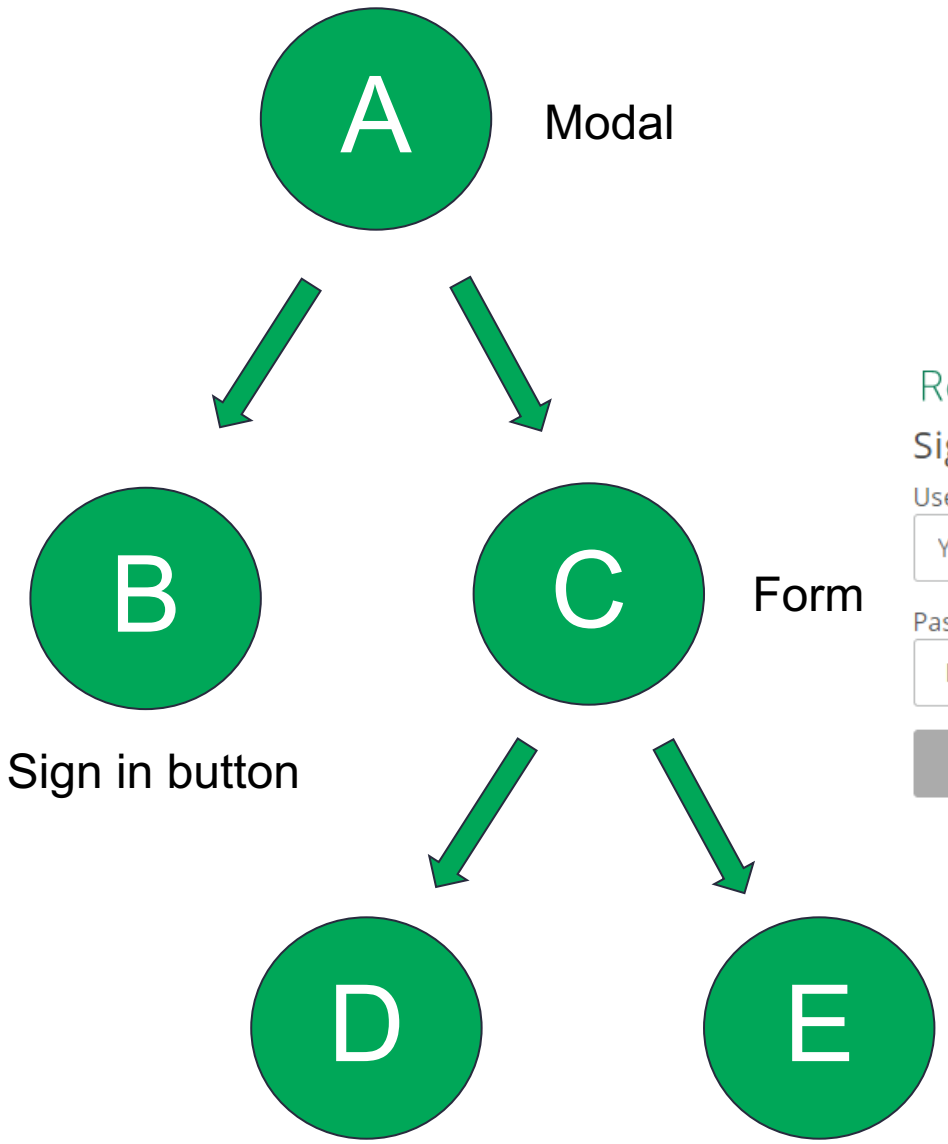
Component Tree

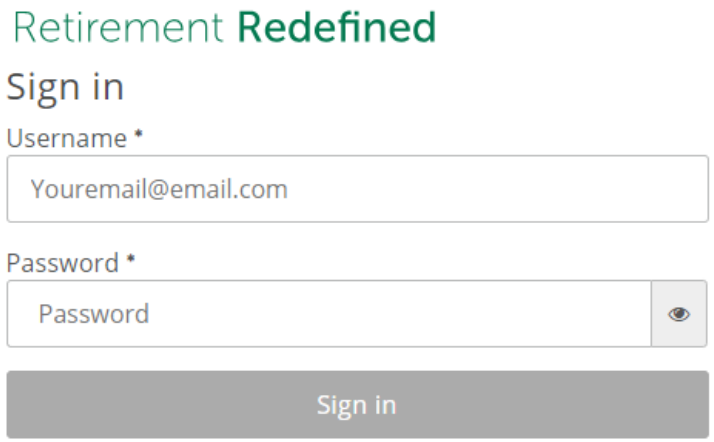


Manulife

Characteristics of React

Component Tree (irl example)





Characteristics of React

Declaring a React Component

As a function

```
// functional component (arrow function)
const HelloWorld = props => <div>Hello World</div>;
```

```
// functional component
function HelloWorld(props) {
  return (
    <div>HelloWorld</div>
  );
}
```

As a class

```
// class-based component
class HelloWorld extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <div>Hello World</div>
    );
  }
}
```

Signal-to-Noise Ratio

Manulife

Characteristics of React

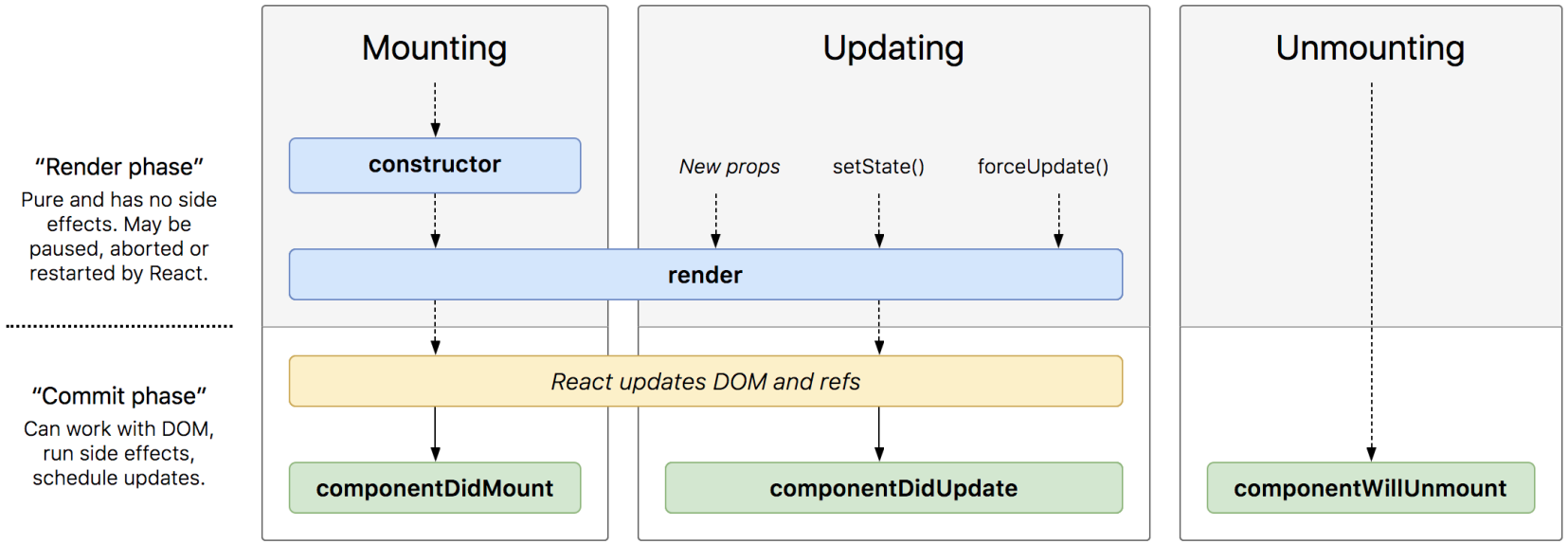
Function vs Class Component

- As a function:
  - Fewer lines of code, easy to understand
  - Performant
  - Easy to test
  - Prior to React v16.8.3, could not have local state

- As a class:
  - More verbose
  - More boilerplate
  - Access to component lifecycle methods
  - Prior to React v16.8.3, only component that can have local state

Characteristics of React

Common Life Cycle Methods (Class components only)



All lifecycle methods:

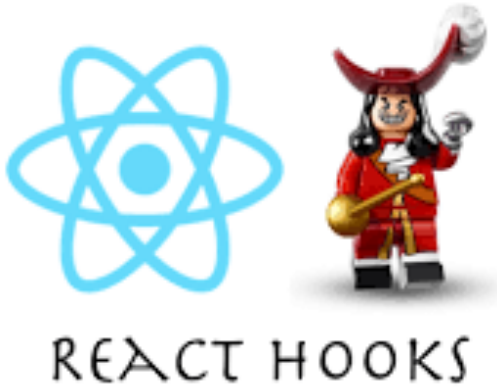
http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Manulife

Characteristics of React

Hooks - Overview

- Let you use state and other React features without writing a class
- Completely opt-in, you can try Hooks in a few components without rewriting any existing code
- 100% backwards-compatible. Hooks don’t contain any breaking changes.
- There are no plans to remove classes from React



Characteristics of React

Hooks - Overview

- Why Hooks?
  - It’s hard to reuse stateful logic between components
  - Complex components become hard to understand

- Rules for Hooks
  - **Don’t** call Hooks inside loops, conditions, or nested functions
  - Only call Hooks **from React function components**. Don’t call Hooks from regular JavaScript functions
  - There is just one other valid place to call Hooks — your own custom Hooks. We’ll learn about them later
  - **https://reactjs.org/docs/hooks-rules.html**

Characteristics of React

Hooks – built-in

useState

- Call it inside a function component to give it some state
- React will preserve this state between re-renders
- Returns a pair: the *current* state value and a function that lets you update it
- You can use the State Hook more than once in a single component

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Manulife

Characteristics of React

Hooks – built-in

useEffect

- Adds the ability to perform side effects from a function component
- Serves the same purpose as componentDidMount, componentDidUpdate, and componentWillUnmount in React classes, but unified into a single API
- By default, React runs the effects after every render — *including* the first render
- You can use more than a single effect in a component

```
const [count, setCount] = useState(0);

// Similar to componentDidMount and componentDidUpdate:
useEffect(() => {
  // Update the document title using the browser API
  document.title = `You clicked ${count} times`;
});
```

Manulife

Characteristics of React

Hooks – built-in

Other Hooks

- **useContext** lets you subscribe to React context without introducing nesting
- **useReducer** lets you manage local state of complex components with a reducer
- **useCallback** will return a memoized version of the callback that only changes if one of the dependencies has changed
- **useMemo** will only recompute the memoized value when one of the dependencies has changed
- **useRef** returns a mutable ref object whose .current property is initialized to the passed argument (initialValue). The returned object will persist for the full lifetime of the component.
- Find more at **https://reactjs.org/docs/hooks-reference.html**

Characteristics of React

Hooks – custom

- Are a way to reuse **stateful logic**, not state itself
- Each **call** to a Hook has a completely isolated state — so you can even use the same custom Hook twice in one component
- More of a **convention** than a feature. If a function's name starts with "use" and it calls other Hooks, we say it is a custom Hook
- Can cover a **wide range** of use cases like form handling, animation, declarative subscriptions, timers, and many more
- Some useful examples of custom hooks can be found at **https://useHooks.com**

Characteristics of React

Hooks – FAQ

Do Hooks cover all use cases for classes?

- There are no Hook equivalents to the uncommon getSnapshotBeforeUpdate and componentDidCatch lifecy cles yet

What do Hooks mean for popular APIs like Redux connect() and React Router?

- They'll continue to work, React Redux supports the Hooks API and exposes hooks like useDispatch or useSelector

Characteristics of React

Render Phase

When does a component re-render?

Any time **state** or **props** change.

Q. Why would it be a bad idea to update state in render?

How to prevent unnecessary re-renders?

- Pass only the props we need
- If new props === old props, re-render will happen by default, unless:
  - Class extends **React.PureComponent**
  - Functional component uses **React.Memo()**

Ecosystem

Ecosystem

Complimentary libraries

- **MUX** is a React component library that implements Manulife branding out of the box
- Inner source library, everyone can contribute
- Interactive storybook provided to learn components
- Benefits of component libraries
  - Collaborate efficiently
  - Staying consistent
  - Customization
  - Flexibility
  - Allows for iteration

Git: https://git.platform.manulife.io/cdt-ux-engineering/mux

Slack: Canadian Division **#topic-mux**

Manulife

Ecosystem

Complimentary libraries

React-router for page navigation

Redux enables complex data-driven API calls and state management

Redux-forms or **React-final-form** for managing form state

Manulife

Ecosystem

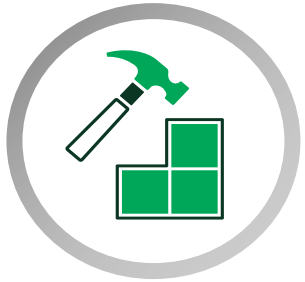
Runtime environment and dependency management

- JavaScript runtime environment is **Node.js** for the server and a browser for the client side
- **NPM** is the default Node.js package manager to handle dependencies
- **Webpack** is a modular build tool that has two sets of functionality, loaders and plugins
  - Loaders transform the source code of a module (i.e. transpile)
  - Plugins can do things loaders can’t, like minify and uglify js

Ecosystem

Transpile and build

Babel transpiles modern JavaScript to browser-compatible JavaScript



Webpack builds all code into optimized bundle(s)

Ecosystem

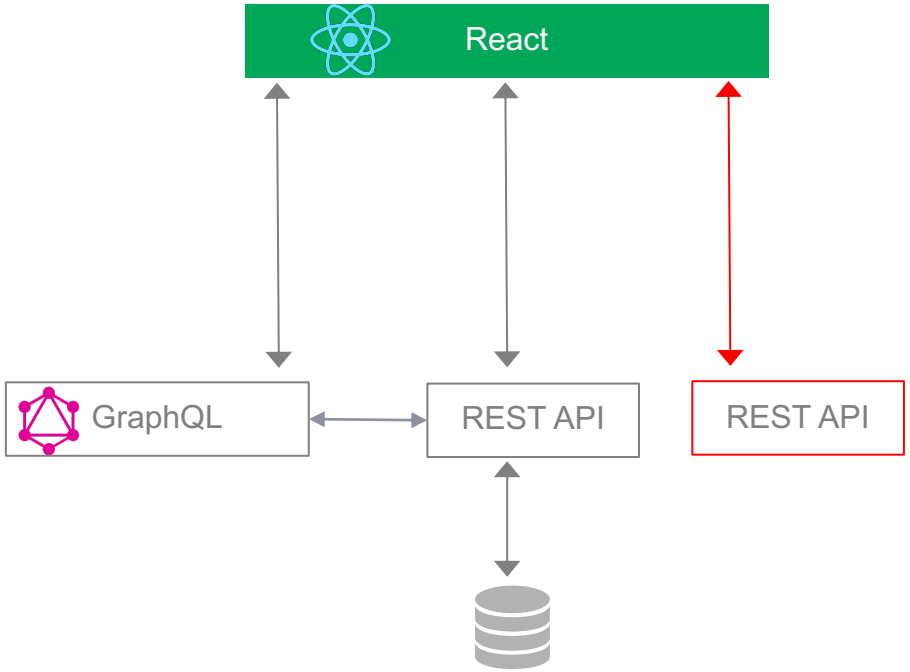
Code style and unit test

Prettier & **ESLint** validate code style and best practices

Jest & **React-testing-library** are used for unit testing

Exercise

Code Along



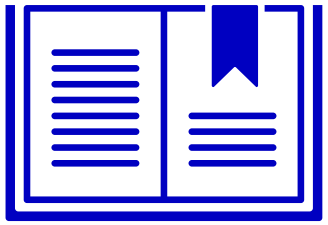
Manulife

Exercise

Code Along

Instructions for this exercise can be found in GitLab:

https://git.platform.manulife.io/mu-materials/react-exercises



Additional reading

Official React docs	https://reactjs.org
Full Stack React	https://www.fullstackreact.com/


MU
ENGINEERING