Evolution Engineering

Middle React

Bootcamp [TypeScript]

React Lifecycle Phases

- Mounting phase
- Updating phase
- Unmounting phase
- Error Handling phase

https://reactjs.org/docs/react-component.html#the-component-lifecycle

React Lifecycle Phases

- **Mounting phase** - a component is being created and inserted into the DOM
- **Updating phase** - a component is being re-rendered
- **Unmounting phase** - a component is being removed from the DOM
- **Error Handling phase** - when there is an error during rendering, in a lifecycle method, or in the constructor of any child component

https://reactjs.org/docs/react-component.html#the-component-lifecycle

React Lifecycle Methods

Mounting phase - a component is being created and inserted into the DOM

1. constructor() - is called before component is mounted
2. static getDerivedStateFromProps() - to update the state as the result of changes in props
3. render() - is the only required method in a class component
4. componentDidMount() - is called right after component is inserted into the DOM tree

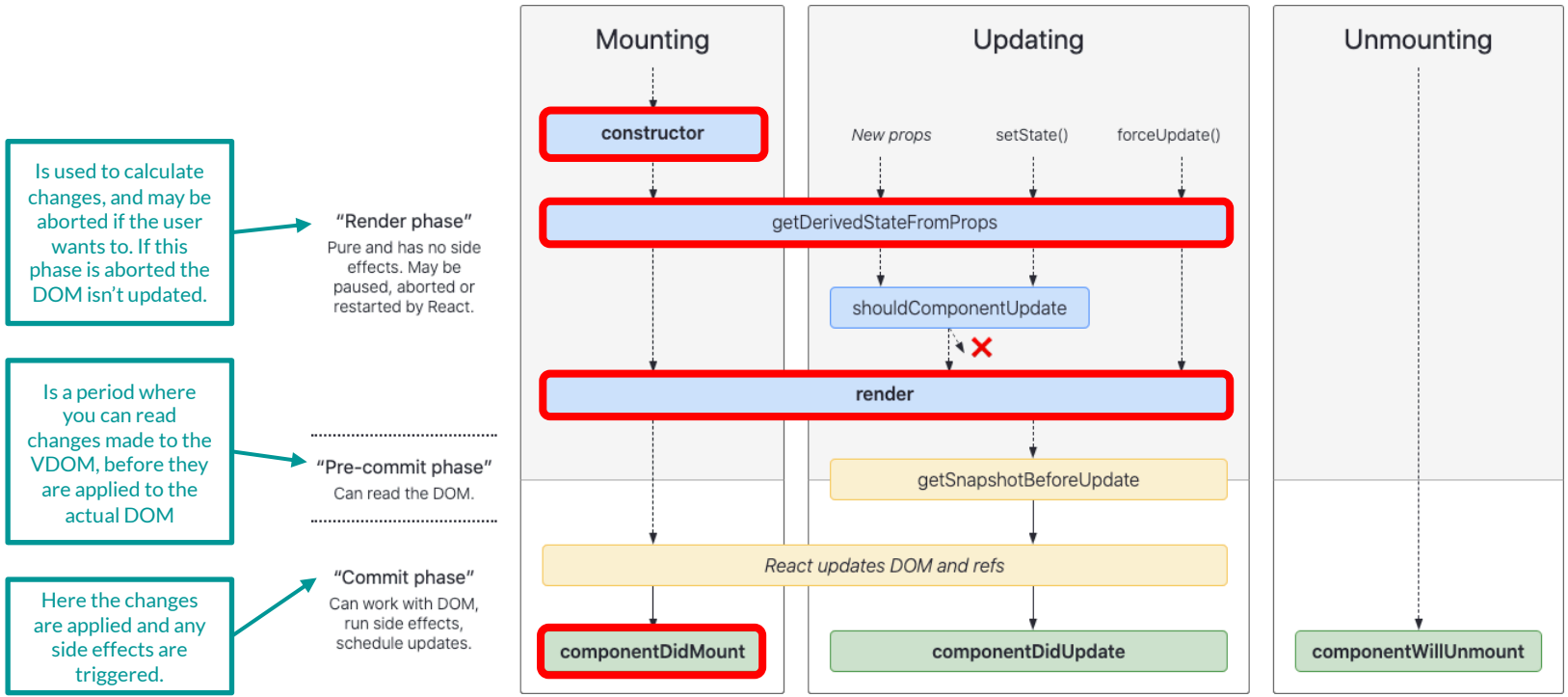
Updating phase - a component is being re-rendered

Unmounting phase - a component is being removed from the DOM

Error Handling phase - when there is an error during rendering, in a lifecycle method, or in the constructor of any child component

https://reactjs.org/docs/react-component.html#the-component-lifecycle

React Lifecycle Methods



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

React Lifecycle Methods

Mounting phase - a component is being created and inserted into the DOM

Updating phase - a component is being re-rendered

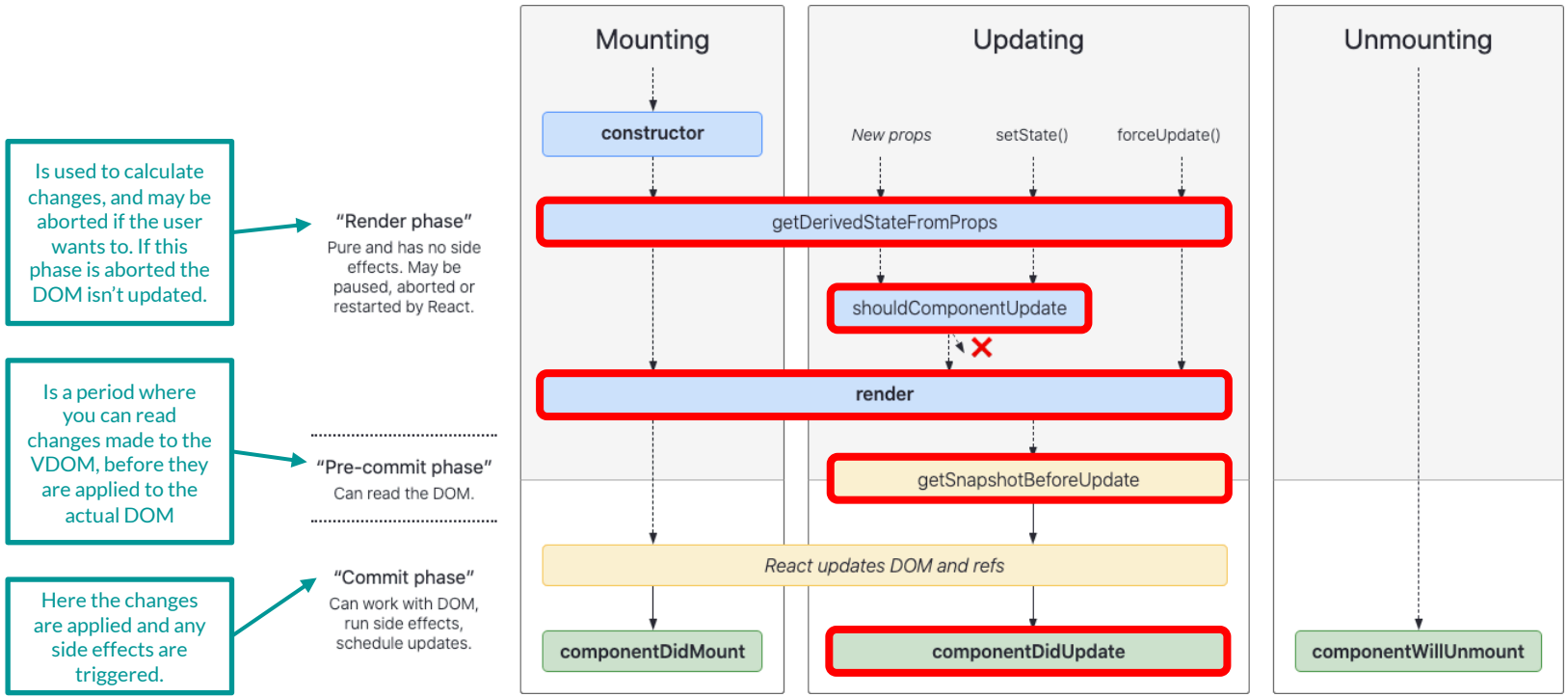
1. static getDerivedStateFromProps() - to update the state as the result of changes in props
2. shouldComponentUpdate() - this method only exists as a performance optimization
3. render() - is the only required method in a class component
4. getSnapshotBeforeUpdate() - is invoked right before the most recently rendered output is committed to the DOM
5. componentDidUpdate() - is invoked immediately after updating occurs

Unmounting phase - a component is being removed from the DOM

Error Handling phase - when there is an error during rendering, in a lifecycle method, or in the constructor of any child component

https://reactjs.org/docs/react-component.html#the-component-lifecycle

React Lifecycle Methods



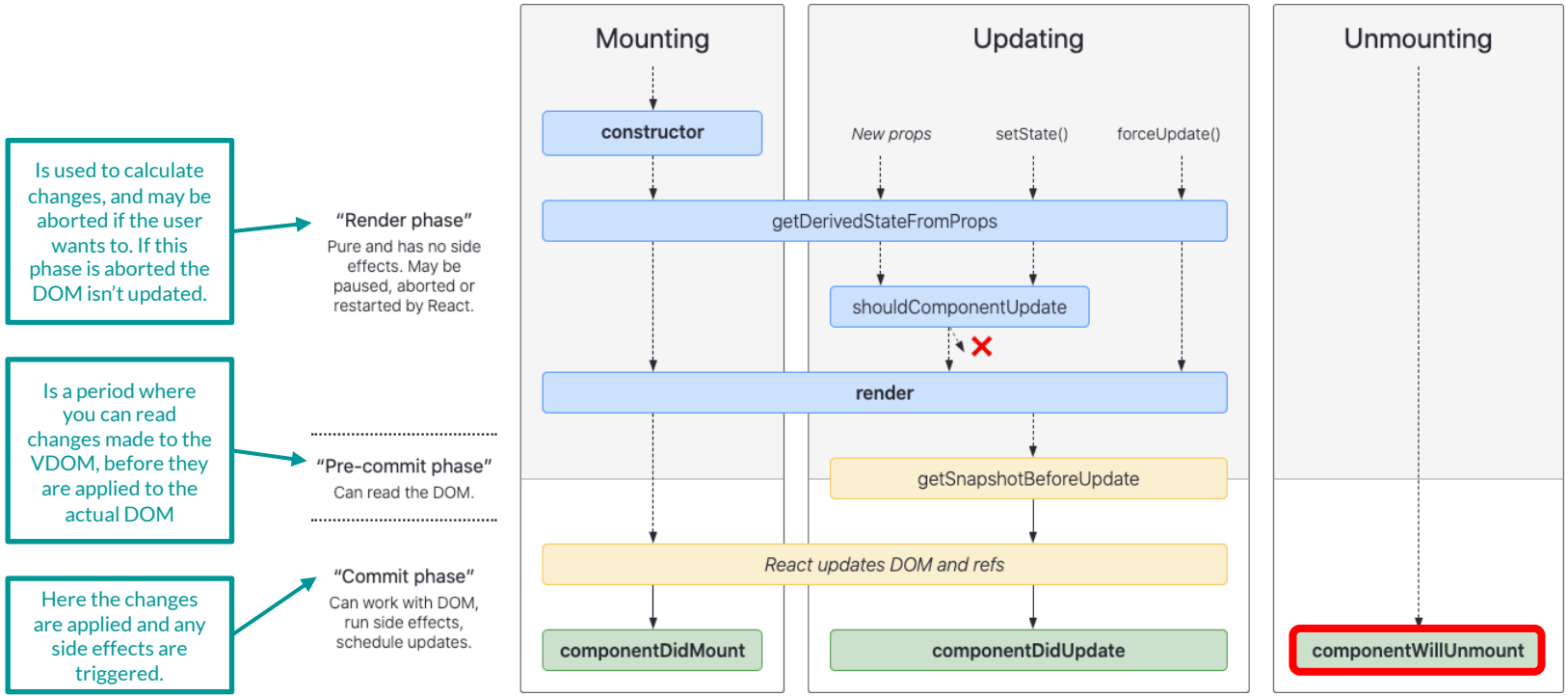
https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

React Lifecycle Methods

- **Mounting phase** - a component is being created and inserted into the DOM
- **Updating phase** - a component is being re-rendered
- **Unmounting phase** - a component is being removed from the DOM
  1. componentWillUnmount() - is invoked before a component is unmounted and destroyed
- **Error Handling phase** - when there is an error during rendering, in a lifecycle method, or in the constructor of any child component

https://reactjs.org/docs/react-component.html#the-component-lifecycle

React Lifecycle Methods



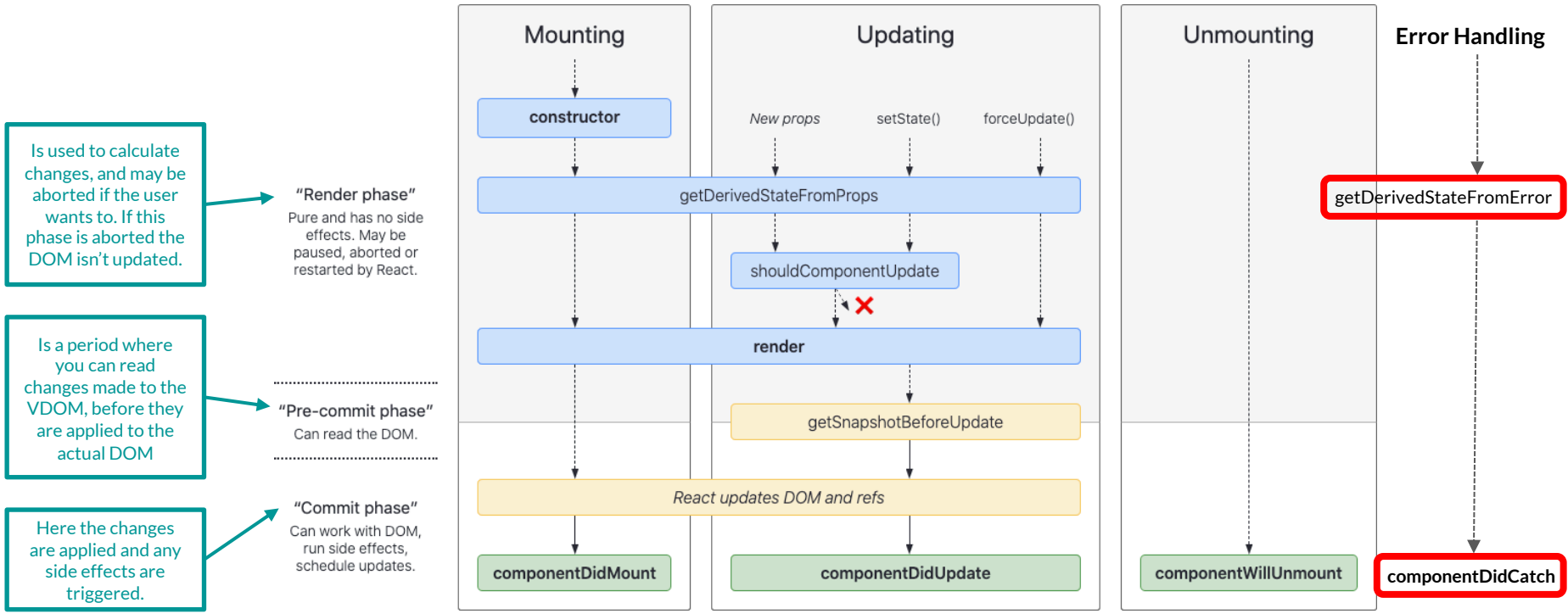
https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

React Lifecycle Methods

- **Mounting phase** - a component is being created and inserted into the DOM
- **Updating phase** - a component is being re-rendered
- **Unmounting phase** - a component is being removed from the DOM
- **Error Handling phase** - when there is an error during rendering, in a lifecycle method, or in the constructor of any child component
  1. static getDerivedStateFromError() - is invoked after an error has been thrown by a descendant component
  2. componentDidCatch() - is invoked after an error has been thrown by a descendant component

https://reactjs.org/docs/react-component.html#the-component-lifecycle

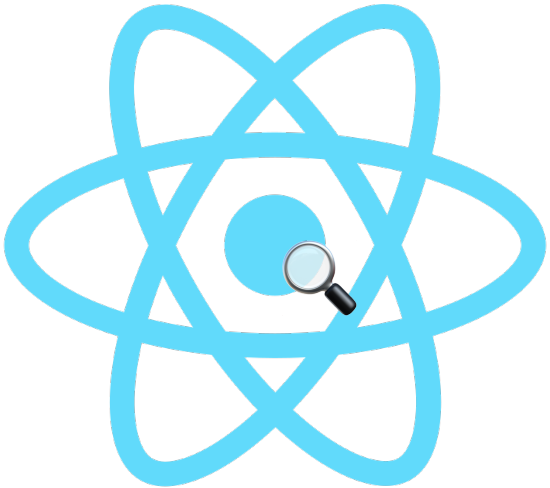
React Lifecycle Methods



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

React Lifecycle Methods

Let’s see…



https://codesandbox.io/s/lifecycle-methods-hdzsu?file=/src/index.tsx

Pure Components

1. Unlike the React.Component, the React.PureComponent **has a default implementation of shouldComponentUpdate** under the hood. This default implementation is based on a shallow prop and state comparison (it doesn't compare object attributes, only their references are compared).

i. React.PureComponent is still can be re-rendered with forceUpdate() method.

https://reactjs.org/docs/react-api.html#reactpurecomponent

Pure Components

So, your Component acts like a Pure Function?
Consider extending the React.PureComponent!

```tsx
import React from "react";

interface MyClassComponentProps {
    a: number;
    b: number;
}

export default class MyClassComponent extends React.PureComponent<MyClassComponentProps> {
    render() {
        const { a, b } = this.props;

        console.log("render MyClassComponent");

        return (
            <h2>a + b = {Math.round(a + b)}</h2>
        )
    }
}
```

https://codesandbox.io/s/pure-components-55yqs?file=/src/MyClassComponent.tsx

https://reactjs.org/docs/react-api.html#reactpurecomponent

Function Components

Also called Stateless Components.

```tsx
import React from "react";

interface MyFunctionComponentProps {
    a: number;
    b: number;
}

export default function MyFunctionComponent({ a, b }: MyFunctionComponentProps) {
    console.log("render MyFunctionComponent");

    return (
        <h2>a + b = {Math.round(a + b)}</h2>
    );
}
```

https://codesandbox.io/s/pure-components-55yqs?file=/src/MyFunctionComponent.tsx

https://reactjs.org/docs/components-and-props.html#function-and-class-components

React Error Handling

A JavaScript error in a part of the UI shouldn't break the whole app. You can use **Error Boundaries** to catch errors during rendering, in a lifecycle method, or in the constructor.

```
import React from "react";

export default class MyErrorBoundary extends React.Component {
    state = { hasError: false };

    static getDerivedStateFromError(error: Error) {
        return { hasError: true };
    }

    componentDidCatch(error: Error, errorInfo: React.ErrorInfo) {
        // Can perform side effects here
        console.log(error, errorInfo);
```

https://codesandbox.io/s/error-boundaries-it91l?file=/src/MyErrorBoundary.tsx

https://reactjs.org/docs/error-boundaries.html

Synthetic Events

React implements a wrapper over native browser events to smooth out cross-browser differences. This wrapper is called **SynteticEvent**.

```
// TODO: change any to unknown when moving to TS v3
interface BaseSyntheticEvent<E = object, C = any, T = any> {
    nativeEvent: E;
    currentTarget: C;
    target: T;
    bubbles: boolean;
    cancelable: boolean;
    defaultPrevented: boolean;
    eventPhase: number;
    isTrusted: boolean;
    preventDefault(): void;
    isDefaultPrevented(): boolean;
    stopPropagation(): void;
    isPropagationStopped(): boolean;
    persist(): void;
    timeStamp: number;
    type: string;
}
```

https://reactjs.org/docs/events.html

Work with Forms

HTML form elements work a little bit differently from other DOM elements in React, because form elements naturally keep some internal state.

- Controlled Components
- Uncontrolled Components

```
import MyControlledComponentForm from "./MyControlledComponentForm";
import MyUncontrolledComponentForm from "./MyUncontrolledComponentForm";

import "./styles.css";

export default function App() {
  return (
```

https://reactjs.org/docs/forms.html

https://codesandbox.io/s/work-with-forms-29t95?file=/src/MyForm.tsx

Additional Materials

1. The React Lifecycle, step by step: https://medium.com/@vmarchesin/the-react-lifecycle-step-by-step-47c0db0bfe73
2. You Probably Don't Need Derived State: https://reactjs.org/blog/2018/06/07/you-probably-dont-need-derived-state.html
3. Everything about event bubbling/capturing: https://transang.me/everything-about-event-bubbling/
4. Virtual DOM and Internals: https://reactjs.org/docs/faq-internals.html
5. React Virtual DOM Explained: https://programmingwithmosh.com/react/react-virtual-dom-explained/

Thank you!

Middle React ✅