React Training

> Day Three

React LifeCycle

LifeCycle Terms

- `Mount` : Whenever a component is rendered for the first time
- `Unmount` : Whenever a component is removed from DOM

Special methods that we declare to run when a component mounts or unmounts

Basic LifeCycle

- Mount
  - render()
  - componentDidMount()
- Update
  - shouldComponentUpdate()
  - render()
  - componentDidUpdate()
- Unmount
  - componentWillUnmount()

LifeCycle in Class Based Component

```
class MyComp extends React.Component {
  constructor(props) {
    super(props);
  }

  componentDidMount() {
    // Run after component mounts
  }

  componentWillUnmount() {
    // Run before component unmounts
  }

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

Why do we need useEffect LifeCycle Methods? ♻️

Why do we need useEffect LifeCycle Methods? ♻️

> To define what it needs to do after render
>
> To perform side-effects

LifeCycle in Functional Component

```
useEffect(() => {
  document.title = `You clicked ${count} times`;
});
```

useEffect hook runs on every update

Cleanup

```
useEffect(() => {
  const myTimeout = setTimeout(() => {
    // Do something();
  }, 3000);
  return () => {
    clearTimeout(myTimeout);
  }
});
```

Why Cleanup is necessary & why useEffect runs on each update?

```
useEffect(() => {
  function handleStatusChange(status) {
    setIsOnline(status.isOnline);
  }

  ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
  return () => {
    ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
  };
});
```

Tips on useEffect

- Use multiple useEffects if it serves your need
- Optimize performance by skipping events

Optimize performance by skipping events

```
useEffect(() => {
  document.title = `You clicked ${count} times`;
}, [count]); // Only re-run the effect if count changes
```

Run useEffect only once

```
useEffect(() => {
  document.title = `You clicked ${count} times`;
}, []);
```

Lists and Keys

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li key={number}>{number}</li>
);
```

Keys

- Diffing of lists is performed using keys. Keys should be stable, predictable, and unique.
- Keys Must Only Be Unique Among Siblings

Homework 🏡

Create a Clock app:

- Current time ⌚
- Stop watch ⏱️
- Timer ⏰