React Fundamentals Module – lifecycle hooks
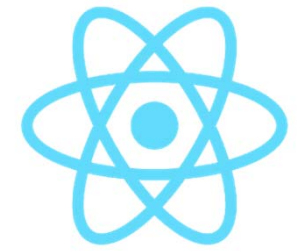
Peter Kassenaar –
info@kassenaar.com

Lifecycle hooks
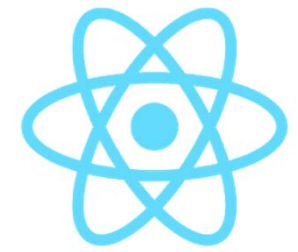
Executing code at specific points in time of the component lifecycle
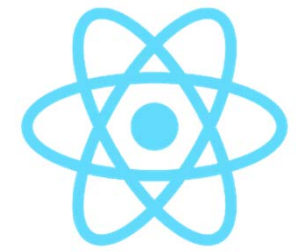
What are lifecycle hooks?

"Every Component follows a cycle from when it's created and mounted on the DOM to when it is unmounted and destroyed. This is what we refer to as the **Component lifecycle**.*"*

https://blog.pusher.com/beginners-guide-react-component-lifecycle/

Available lifecycle hooks

- Three stages in the lifecycle

  1. Creating and Mounting

  2. Updating

  3. Unmounting

- We now cover the lifecyle hooks of `class` based components

- Function based components have a generic `useEffect()` hook

1. Mounting lifecycle hooks

- `constructor()`

    - The default JavaScript constructor of the function or class

- `render()`

    - Your render function

- `componentDidMount()` – used for:

    - Make API calls

    - Set up timers

    - Add event listeners…
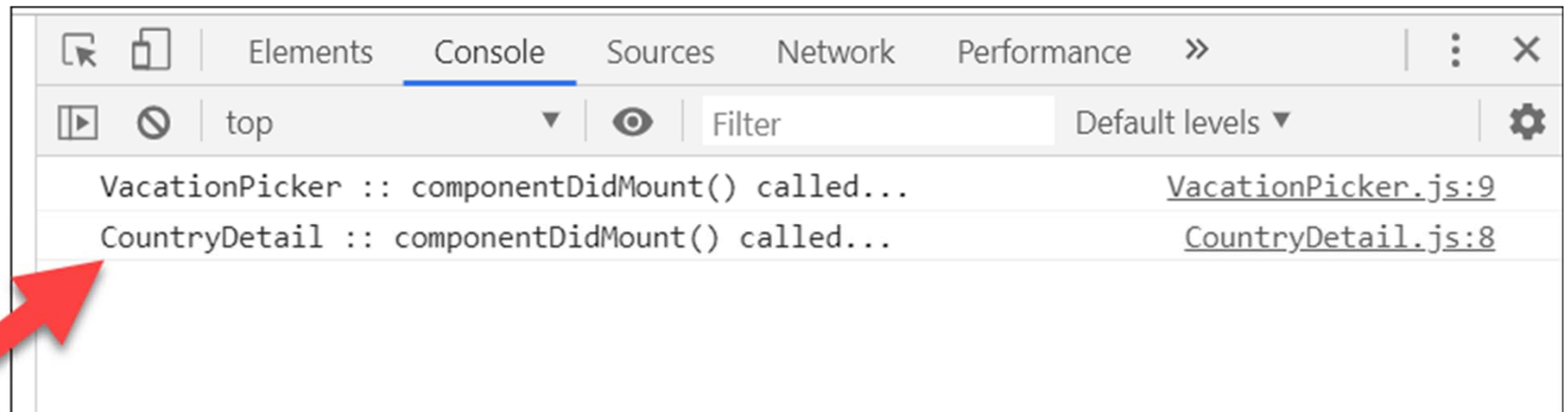
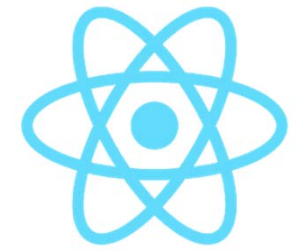- OLD: `componentWillMount()` – don't use anymore!

`componentDidMount()`
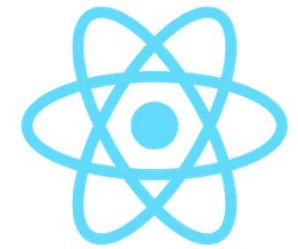
```
// Mounting lifecycle hook
componentDidMount() {
    console.log('VacationPicker :: componentDidMount() called...');
}
```

```
componentDidMount() {
    console.log('CountryDetail :: componentDidMount() called...');
}
```

2. Updating lifecycle hooks

- `shouldComponentUpdate()`
    - Called before the component re-renders after receiving new props or there's a new state.
    - Two parameters: `nextProps` and `nextState`
    - Not used very often – may harm performance

- `componentDidUpdate()`
    - Called after any rendered HTML has finished loading
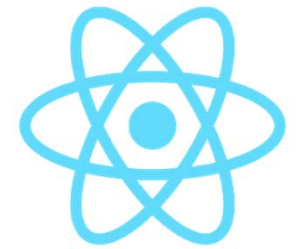   - Two parameters: `prevProps` and `prevState`
    - Used very often
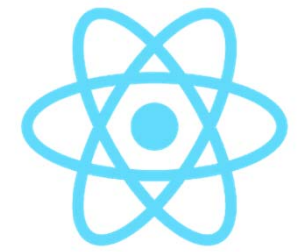
- Deprecated – don't use anymore

  - `componentWillReceiveProps()`

  - `componentWillUpdate()`

- https://reactjs.org/docs/react-component.html

Note:

These methods are considered legacy and you should avoid them in new code:

- `UNSAFE_componentWillUpdate()`

- `UNSAFE_componentWillReceiveProps()`

Result

```
state={
    currentCountry:{},
    prevCountry:{}
};
```

```
// 2. Update lifecycle hook
componentDidUpdate(prevProps, prevState, snapshot) {
    console.log('CountryDetail updated!');
    console.log({prevProps}, {prevState});
    // store previous and current country in state
    if(prevProps.country.name !== this.props.country.name) {
        this.setState({
            currentCountry: this.props.country,
            prevCountry: prevProps.country
        })
    }
}
```
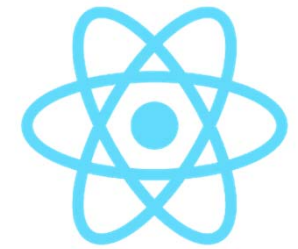
Result
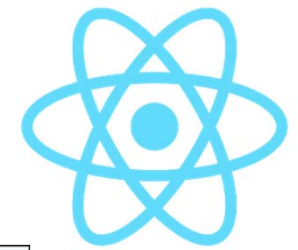
3. Unmounting lifecycle hooks
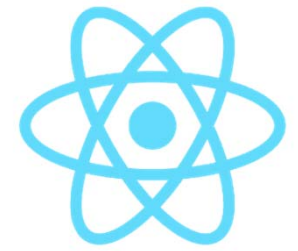
Only one hook available: `componentWillUnmount()`

"`componentWillUnmount()` *is called right before a component is removed from the DOM"*

Result

CountryDetail.js

```
// 3. Unmount lifecycle hook
componentWillUnmount() {
    console.log('CountryDetail :: Component unmounted...');
    // flush caches, close db-connections, update localStorage, etc.
}
```
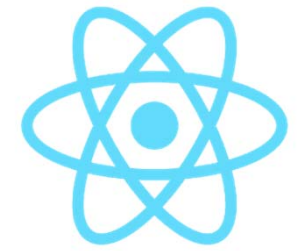
```
// Show or hide country details in the UI
toggleCountries() {
    this.setState({
        showCountries: !this.state.showCountries
    })
}                                               App.js
```

```
{                                               App.js
    // conditional rendering
    this.state.showCountries &&
    <CountryDetail country={this.state.currentCountry}/>
}
```

…/240-lifecycle-hooks

Summary - visual overview
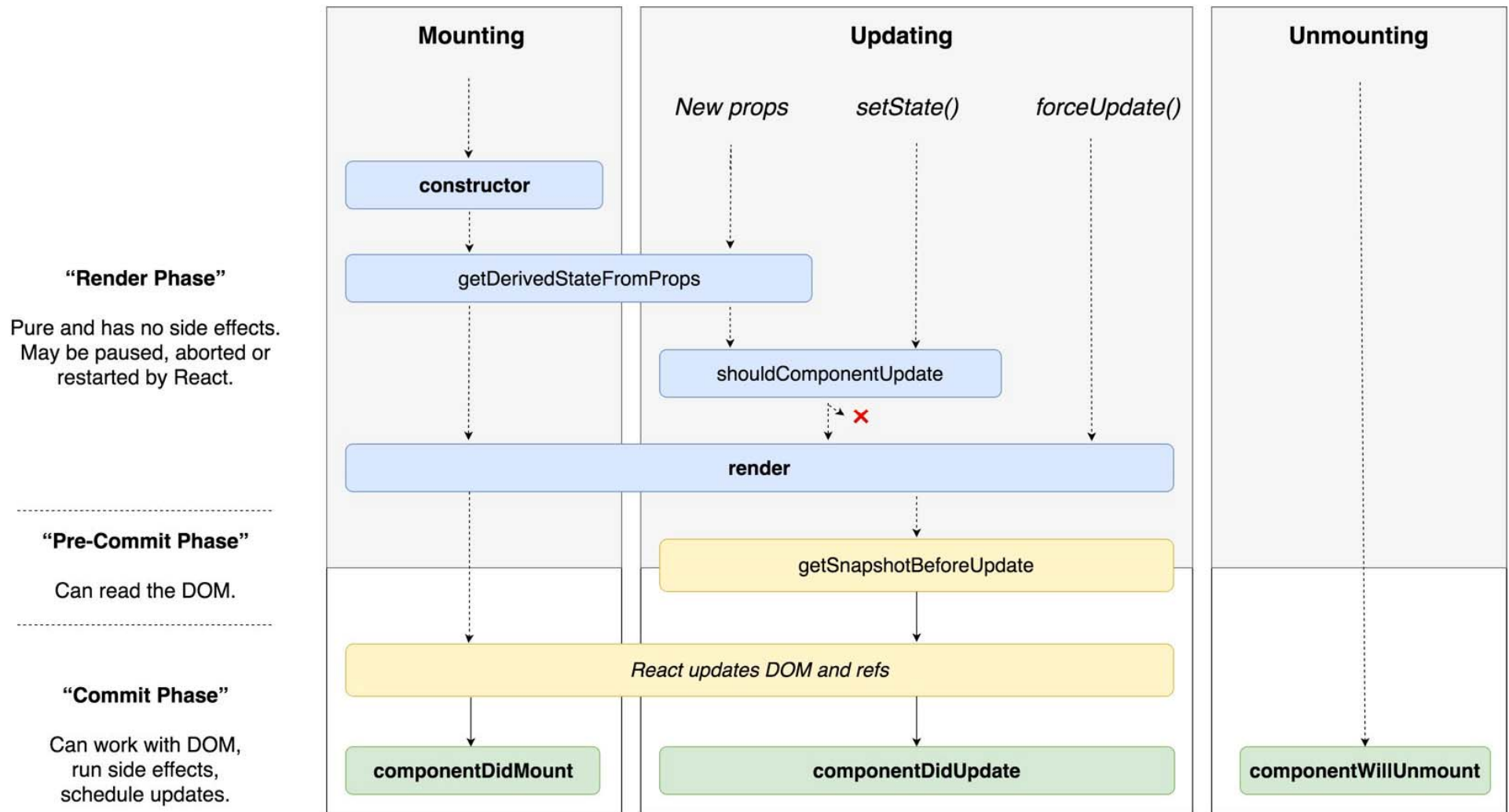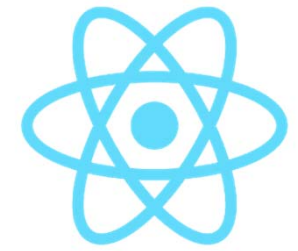
Workshop

- Create a component with a `button` and a `counter`

- In a lifecycle hook, log how many times `counter` is updated.
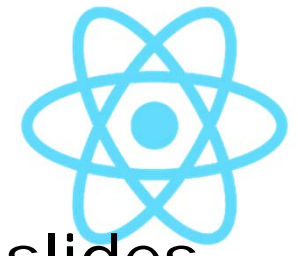
- After initial rendering it should read '`you updated 0 times`'

- After that, '`you updated 1 time, 2 times, 3 times`', etc.

- Embed the counter in another component where you can show/hide it.

- Make sure the lifecycle hooks still work as expected

- Example `../240-lifecycle-hooks`
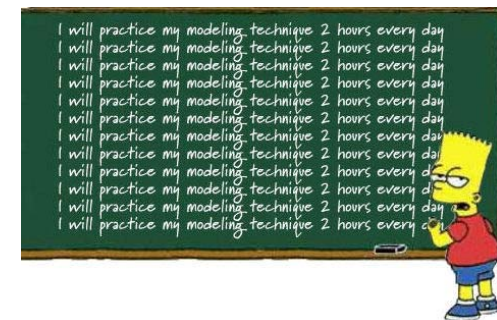
Final Workshop

- Create the app shown in the beginning of the slides

  - One country is visible at a time

  - Its name and capital are shown

  - There are two buttons to cycle through the countries

  - User can hide/show details with a button

  - There are badges if visiting the country is Expensive, or On Sale

- Example: `../workshops/40-data-binding-wrap-up`

  - But first try it for yourself!

More info

- https://blog.pusher.com/beginners-guide-react-component-lifecycle/
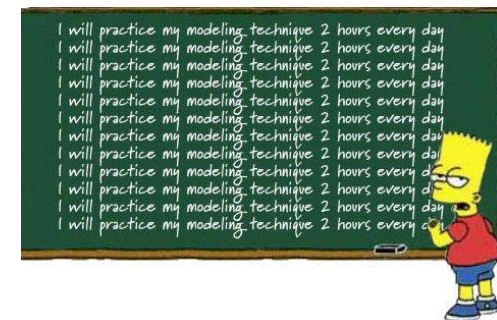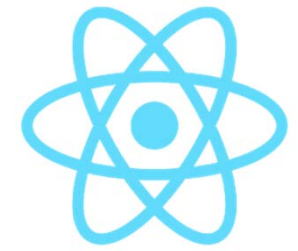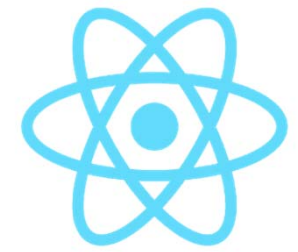
Checkpoint

- You know about component lifecycle hooks

- There are three important hooks:

  - `componentDidMount()`

  - `componentDidUpdate()`

  - `componentWillUnmount()`

- Previous versions of React had additional hooks that are now deprecated

- Lifecycle hooks can be used in class based components