Lifecycle Triggers and Events

- from when a component is invoked to when it is destroyed, it goes through a series of lifecycle events.
- these functions give you the opportunity to make decisions and take appropriate actions.
- there are four triggers that kick off these lifecycle events that we will examine.

Lifecycle Event Triggers

- Initialization
- Updating State
- Updating Props
- Unmounting

Trigger: Initialization

Events:

1. getDefaultProps
2. getInitialState
3. componentWillMount
4. render
5. componentDidMount

getDefaultProps:

- used to define any default props which can be accessed via `this.props`
- these are the values give to a prop in the case that it isn't passed in.
- in an es6 class, this is a static property defined on the class.

getInitialState

- used to set initial states for your component
- only called once when initially rendering the component
- common to use some value from the props being passed in.
- in an es6 class, this is set in the `constructor` function
- must call 'super(props)', when using 'this' in the constructor.

componentWillMount

- called once immediately before the render method is executed
- calling setState in this function will not cause render to be called more than once.

render

- returns the component markup, which can be a single child component, a set of components, null, or false (in case you don't want any rendering)

componentDidMount

- called once immediately after the initial rendering has occured
- the DOM is now available at this point,
- this is where you'll want to use setInterval, setTimeout, and ajax requests.

Example

```
// within your component class
class MyComponent extends Component {
    constructor(props) {
        super(props);
        this.state = {
            name: props.name
        }
    }

    componentWillMount() {...}

    render () {...}

    componentDidMount() {
      // window.setTimeout(function() {}, 100)...
    }
}

MyComponent.defaultProps = {
    myProp1: 'someVal'
}
```

Exercise

Trigger: Updating State

Events:

1. shouldComponentUpdate
2. componentWillUpdate
3. render
4. componentDidUpdate

shouldComponentUpdate

- this function determines whether or not the component should update itself.
- if the function returns true, the component will update and the rest of the lifecycle steps will be called
- if the function returns false, the component will not update and no other lifecycle functions will be called.

componentWillUpdate

- invoked immediately before rendering when new props or state are received
- you can't use `this.setState()` in this method!!
- should be used only to prepare for an update, not trigger an update itself

componentDidUpdate

- invoked immediately after the component's updates are sent to the DOM
- can be used to operate on the DOM after a component has been updated

Example

```
class MyComponent extends Component {

    shouldComponentUpdate(nextProps, nextState) {
      // return true if it should update; false otherwise
    }

    componentWillUpdate(nextProps, nextState) {
      // some code here
    }

    render() {...}

    componentDidUpdate(prevProps, prevState) {
      // some code here
    }
}
```

Trigger: Updating Props

- Used when the Parent element modifies the props of a child.

Events:

1. componentWillReceiveProps
2. shouldComponentUpdate
3. componentWillUpdate
4. render
5. componentDidUpdate

componentWillReceiveProps

- invoked when a child component is receiving new/updated props
- it's not called for the initial render

Example

```
class MyComponent extends Component {

    componentWillReceiveProps(nextProps) {
      // do something with nextProps here
    }

    shouldComponentUpdate(nextProps, nextState) {
      // return true if it should update; false otherwise
    }

    componentWillUpdate(nextProps, nextState) {
      // some code here
    }

    render() {...}

    componentDidUpdate(prevProps, prevState) {
      // some code here
    }
}
```

Trigger: Unmounting

Events:

1. componentWillUnmount

componentWillUnmount

- invoked immediately before a component is unmounted/removed from the DOM
- perform any necessary cleanup in this method, such as clearing timers or cleaning up any DOM elements that were created in componentDidMount

```
class MyComponent extends Component {

    render() {...}

    componentWillUnmount() {
      // some cleanup code here
    }
}
```

Exercise