React
state & lifecycle

Hello!

I am Mateusz Choma

I am a scientific mind, passionate of technology, an engineer "squared" - a graduate of two universities in Lublin :)
As well, I am a JS developer, entrepreneur and owner of small software house - Amazing Design.

1.
State

State
What is it?

State on object that is like components memory. We can store in that object stuff that can change in component, and use them in components methods and in JSX that will be returned form component render method.

We should modifying state **ONLY** by calling `this.setState` method. By using that method we notify React that something in components displaying could change, and React must re-render component.

Do not call `this.setState` in render because it cause an infinite loop!

State
this.setState

`this.setState` method takes object that will be merged with previous state as first argument and callback that will be invoked when component is updated as second argument.

When we want to change counter property of state object to 1, we can write:

```
this.setState({
    counter: 1
})
```

More -> https://reactjs.org/docs/react-component.html#setstate

State
props .VS. state

- available in smart (class) and dumb (function) components
- there are data from “outside” - passed from parent component
- component can’t change them itself

- available only in smart (class) components
- there are data “inside” component
- component can change them itself by calling `this.setState`

Component re-renders when state is changed AND when it will receives new props!

State
this.forceUpdate

We can force React to re-render component by calling

```
this.forceUpdate()
```

It can be useful when our component render method depends on something else than state and props

More -> https://reactjs.org/docs/react-component.html#forceupdate

2.
Default props & prop types

Default props & prop types
Default props

We can define default props as a property of class itself. Default props will be used when no props will be passed from parent component.

```
class CustomButton extends React.Component {
    ...
}

CustomButton.defaultProps = {
    color: 'blue'
}
```

Default props & prop types
Default props

As you can see in previous example this is a property that belongs to class itself not to the object that will be created from this class.
In programming that type of properties are called static.

From ES6 we can write (same as previous slide):

```
class CustomButton extends React.Component {
    static defaultProps = {
        color: 'blue'
    }
}
```

Default props & prop types

Prop types

We can check if component gets props of the right type by setting static property propTypes on the component class.

From React 15.5 prop types are external library and must be imported!

```
import PropTypes from 'prop-types'
```

More -> https://reactjs.org/docs/typechecking-with-proptypes.html

Default props & prop types
Prop types

```
class Greeting extends React.Component {
  render() {
    return (
      <h1>Hello, {this.props.name}</h1>
    );
  }
}
Greeting.propTypes = {
  name: PropTypes.string
};
```

3.
Component lifecycle

Component lifecycle
What is it?

Component lifecycle is that React is calling certain special methods of smart (class) components every time that something happens with component.

Component lifecycle methods starts when React wants to mount component to DOM and ends when React wants to remove it.

Methods prefixed with **will** are called right before something happens, and methods prefixed with **did** are called right after something happens.

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

Component lifecycle
Example

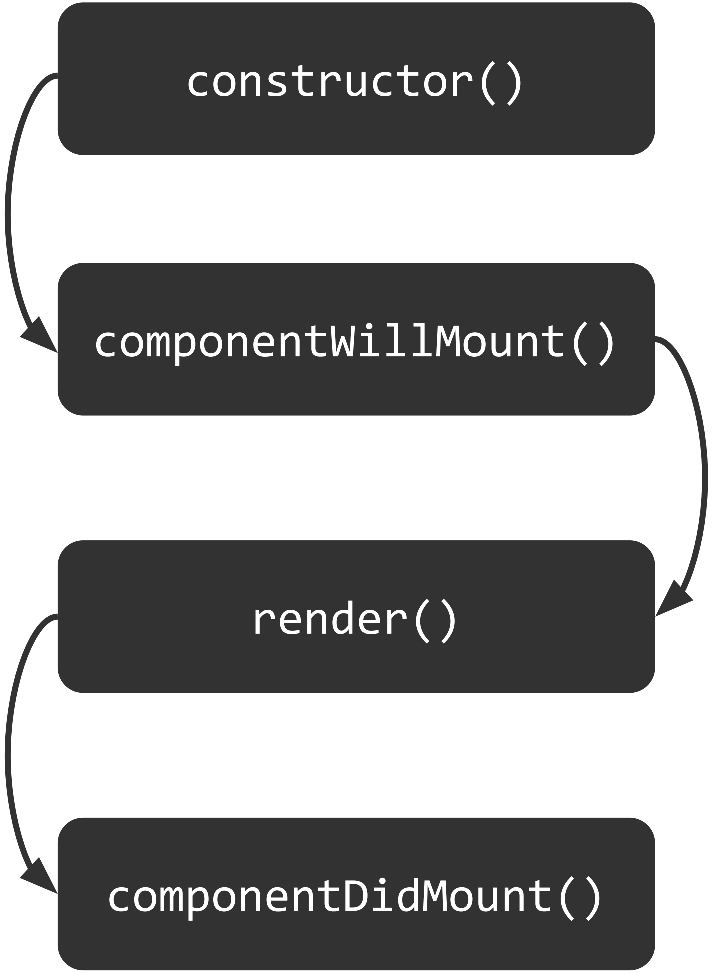
```
class Foo extends React.Component {
  componentDidMount() {
    console.log("I’m ready!")
  }

  render() {
    return Hello
  }
}
```

Component lifecycle
Mounting

These methods are called (in that order) when an instance of a component is being created and inserted into the DOM:

- constructor()
- componentWillMount()
- render()
- componentDidMount()

Mounting

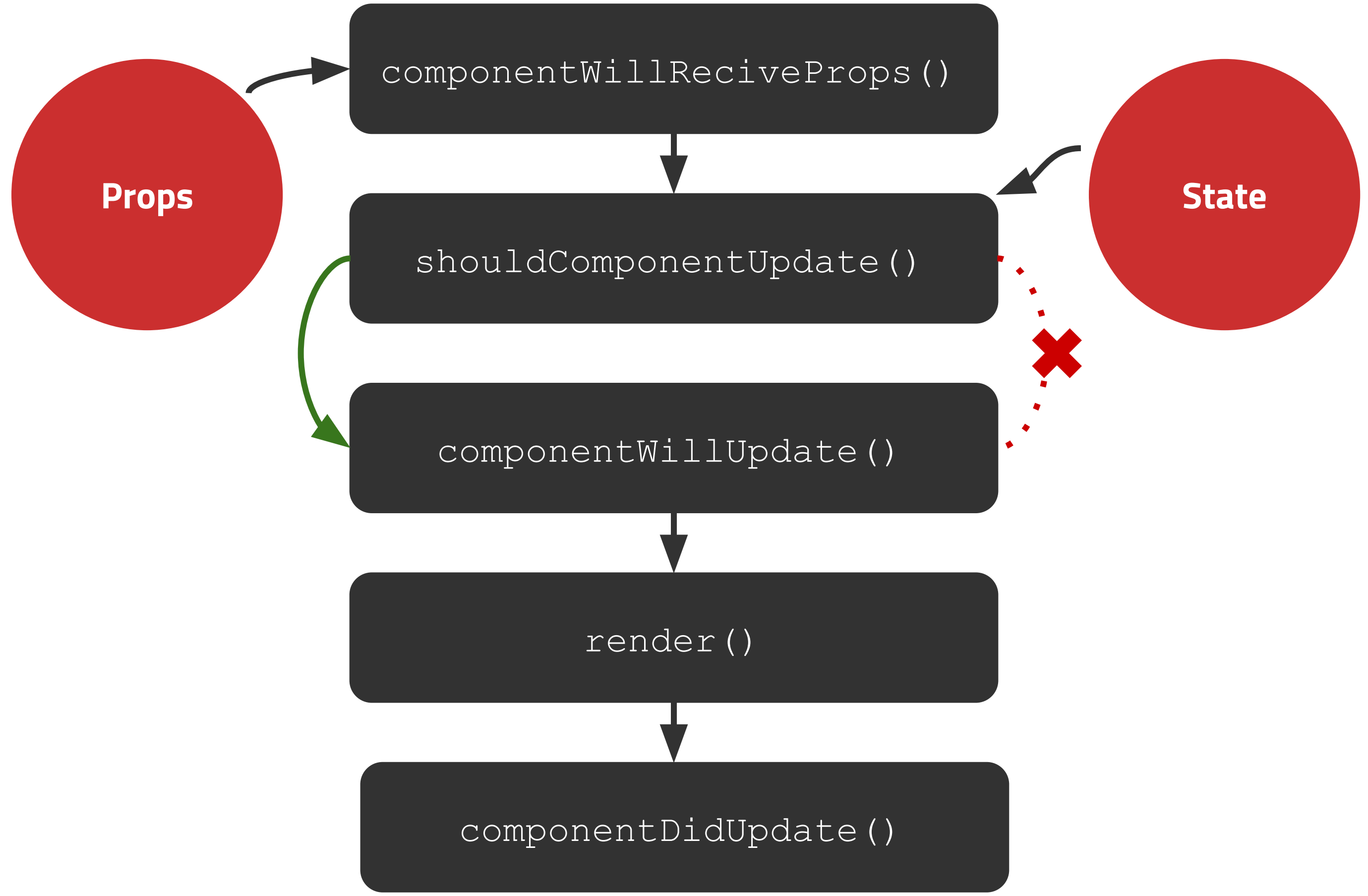

constructor()
componentWillMount()
render()
componentDidMount()

Component lifecycle
Updating

An update can be caused by changes to props or state. These methods are called when a component is being re-rendered:

- componentWillReceiveProps()
- shouldComponentUpdate()
- componentWillUpdate()
- render()
- componentDidUpdate()

Updating


Props
componentWillReciveProps()
State
shouldComponentUpdate()
componentWillUpdate()
render()
componentDidUpdate()

Component lifecycle
Unmounting

This method is called when a component is being removed from the DOM:

- componentWillUnmount()

Unmounting

```
componentWillUnmount()
```

Component lifecycle
Most cases

In most cases we will use only two of them:

1. `componentDidMount` - to fetch data that components need to display, and save it in state
2. `render` - because every component have to render something ;)