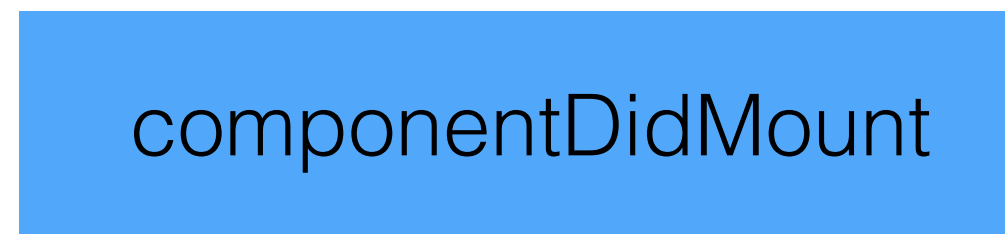
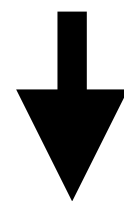
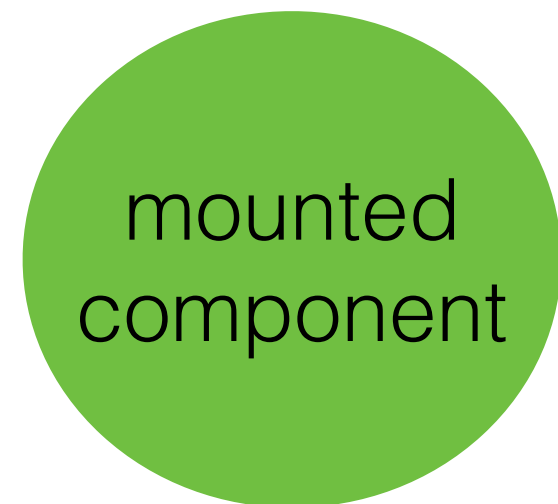
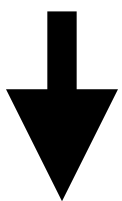
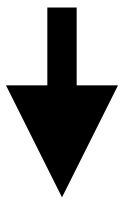
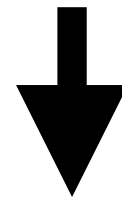


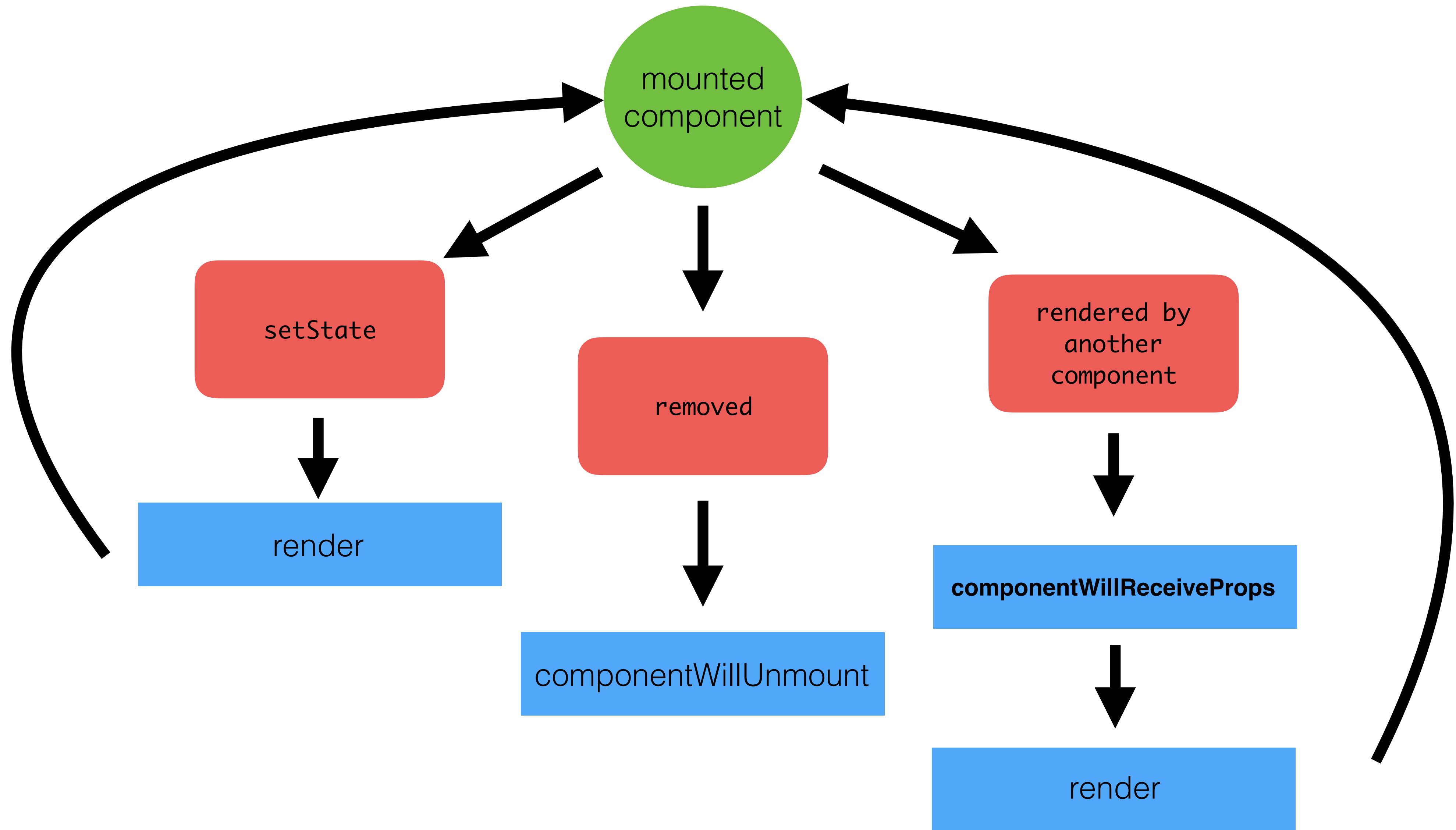
ADVANCED LIFECYCLE METHODS

It's the advanced CI-IIIIRRCLE

ADVANCED LIFECYCLE

- **So far:**
 - **componentWillMount**
 - **componentDidMount**
 - **componentWillUnmount**
- **New:**
 - **componentWillReceiveProps**
 - **shouldComponentUpdate**





COMPONENT WILL RECEIVE PROPS

- **Fired only after the initial render**
- **Fired when a component is rendered by its parent**
- **Chance to compare incoming props with previous props and do something based on that info**

```
class SinglePuppy extends Component {
```

```
}
```

```
class SinglePuppy extends Component {  
  async fetchPuppy (puppyId) {  
  
  }  
  
}
```

```
class SinglePuppy extends Component {  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
}
```



```
class SinglePuppy extends Component {  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
  
  }  
  
}
```

```
class SinglePuppy extends Component {  
  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
    const puppyId = this.props.match.params.puppyId  
    this.fetchPuppy(puppyId)  
  }  
  
}
```

```
class SinglePuppy extends Component {  
  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
    const puppyId = this.props.match.params.puppyId  
    this.fetchPuppy(puppyId)  
  }  
  
  componentWillReceiveProps (      ) {  
  
  
  }  
}
```

```
class SinglePuppy extends Component {

  async fetchPuppy (puppyId) {
    const res = await axios.get(`/api/puppies/${puppyId}`)
    this.setState({puppy: res.data})
  }

  componentDidMount () {
    const puppyId = this.props.match.params.puppyId
    this.fetchPuppy(puppyId)
  }

  componentWillReceiveProps (nextProps) {

  }

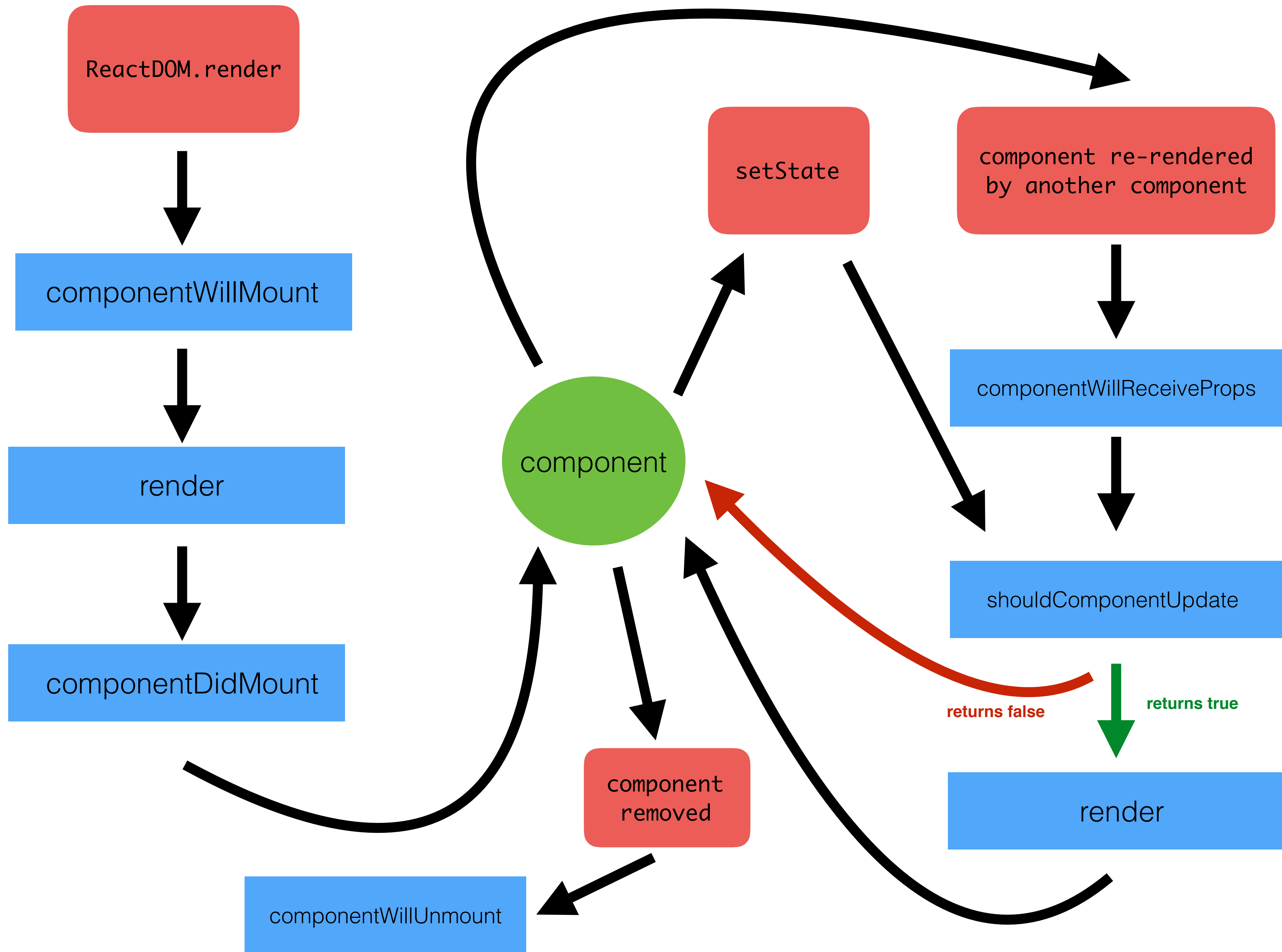
}
```

```
class SinglePuppy extends Component {  
  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
    const puppyId = this.props.match.params.puppyId  
    this.fetchPuppy(puppyId)  
  }  
  
  componentWillReceiveProps (nextProps) {  
    const newPuppyId = nextProps.match.params.puppyId  
  
  }  
}
```

```
class SinglePuppy extends Component {  
  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
    const puppyId = this.props.match.params.puppyId  
    this.fetchPuppy(puppyId)  
  }  
  
  componentWillReceiveProps (nextProps) {  
    const newPuppyId = nextProps.match.params.puppyId  
    const oldPuppyId = this.props.match.params.puppyId  
  
  }  
}
```



```
class SinglePuppy extends Component {  
  
  async fetchPuppy (puppyId) {  
    const res = await axios.get(`/api/puppies/${puppyId}`)  
    this.setState({puppy: res.data})  
  }  
  
  componentDidMount () {  
    const puppyId = this.props.match.params.puppyId  
    this.fetchPuppy(puppyId)  
  }  
  
  componentWillReceiveProps (nextProps) {  
    const newPuppyId = nextProps.match.params.puppyId  
    const oldPuppyId = this.props.match.params.puppyId  
    if (newPuppyId !== oldPuppyId) {  
      this.fetchPuppy(newPuppyId)  
    }  
  }  
}
```



SHOULD COMPONENT UPDATE

- **Fired only after the initial render**
- **Fired when a component sets state or is re-rendered by another component**
- **Must return true or false**
- **Compare previous props/state with new props/state, and prevent re-rendering if it's unnecessary to do so**

```
class FavoritePuppy extends Component {
```

```
}
```

[illegible]

```
class FavoritePuppy extends Component {  
    shouldComponentUpdate (nextProps, nextState) {
```

```
class FavoritePuppy extends Component {  
  shouldComponentUpdate (nextProps, nextState) {  
    if (nextProps.favorite === this.props.favorite) {  
  
    } else {  
  
    }  
  }  
}
```

```
class FavoritePuppy extends Component {  
  shouldComponentUpdate (nextProps, nextState) {  
    if (nextProps.favorite === this.props.favorite) {  
      // don't bother to re-render  
      return false  
    } else {  
      }  
    }  
  }  
}
```

```
class FavoritePuppy extends Component {  
  shouldComponentUpdate (nextProps, nextState) {  
    if (nextProps.favorite === this.props.favorite) {  
      // don't bother to re-render  
      return false  
    } else {  
      // there's a new favorite in town! Re-render!  
      return true  
    }  
  }  
}
```

