ADVANCED LIFECYCLE METHODS

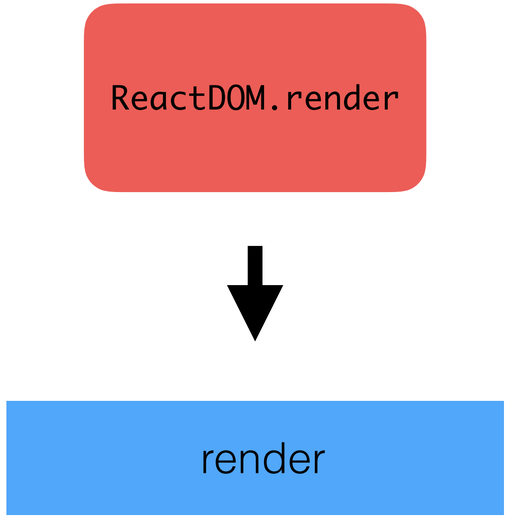
It's the advanced CI-IIIIRRCLE

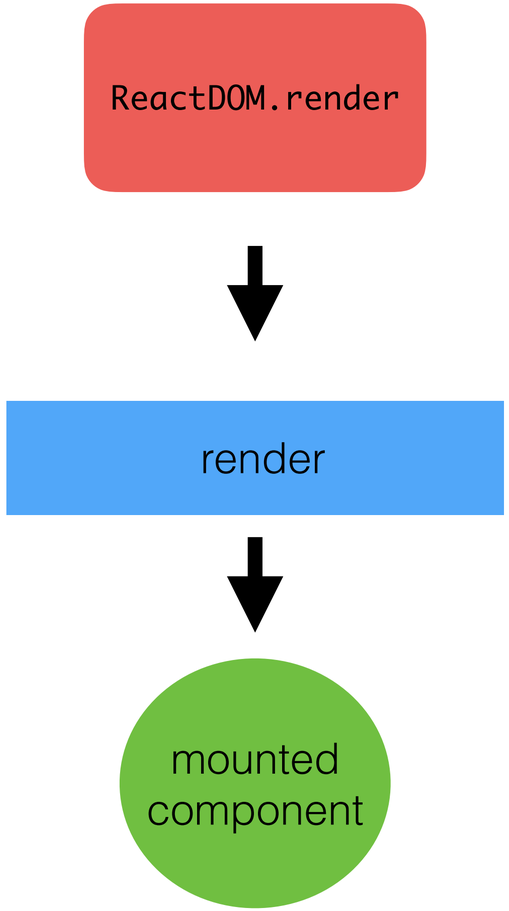
ADVANCED LIFECYCLE

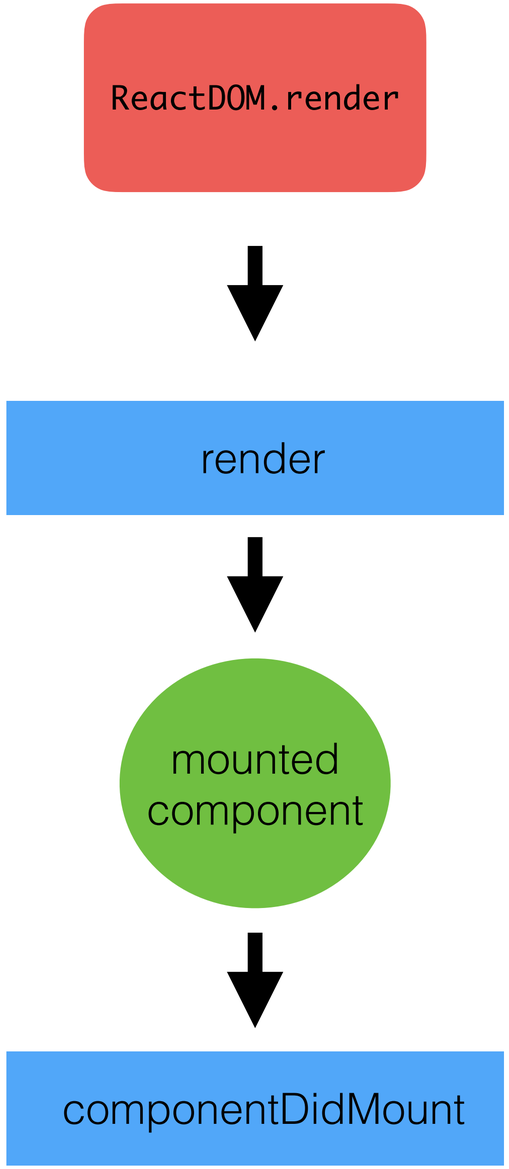
- So far:
  - componentDidMount
  - componentWillUnmount

- New:
  - shouldComponentUpdate
  - componentDidUpdate

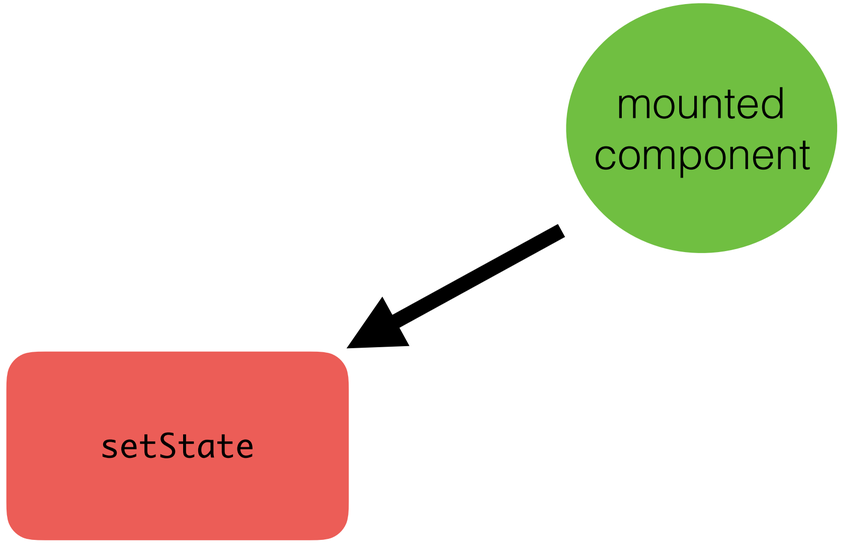

ReactDOM.render

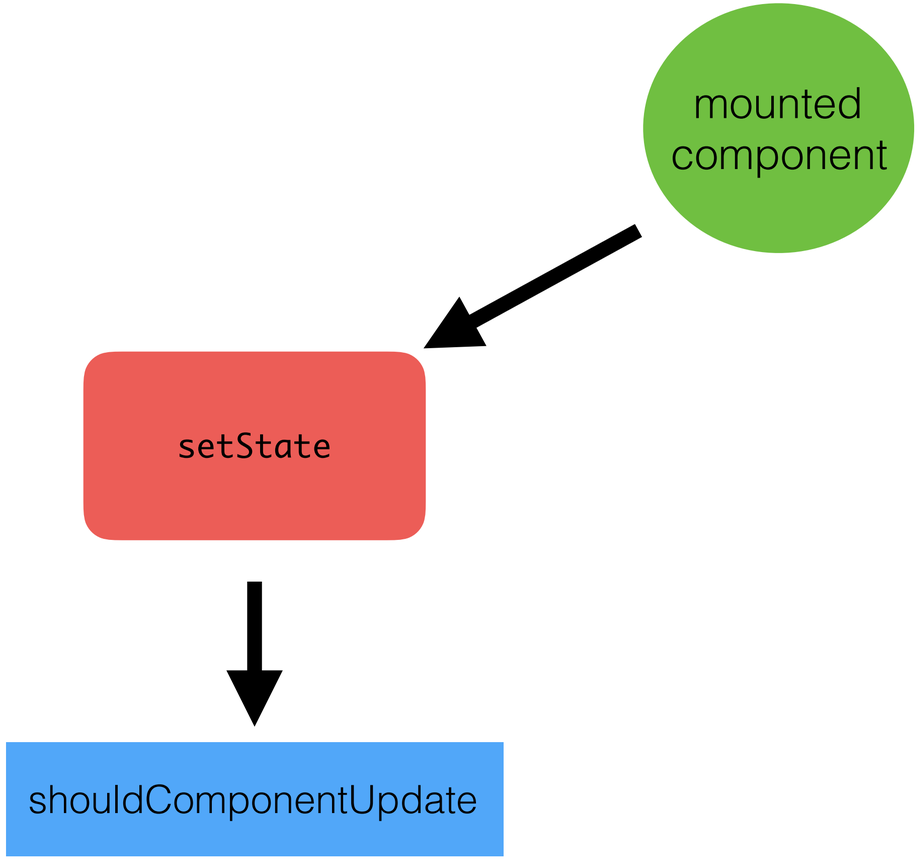

ReactDOM.render
render

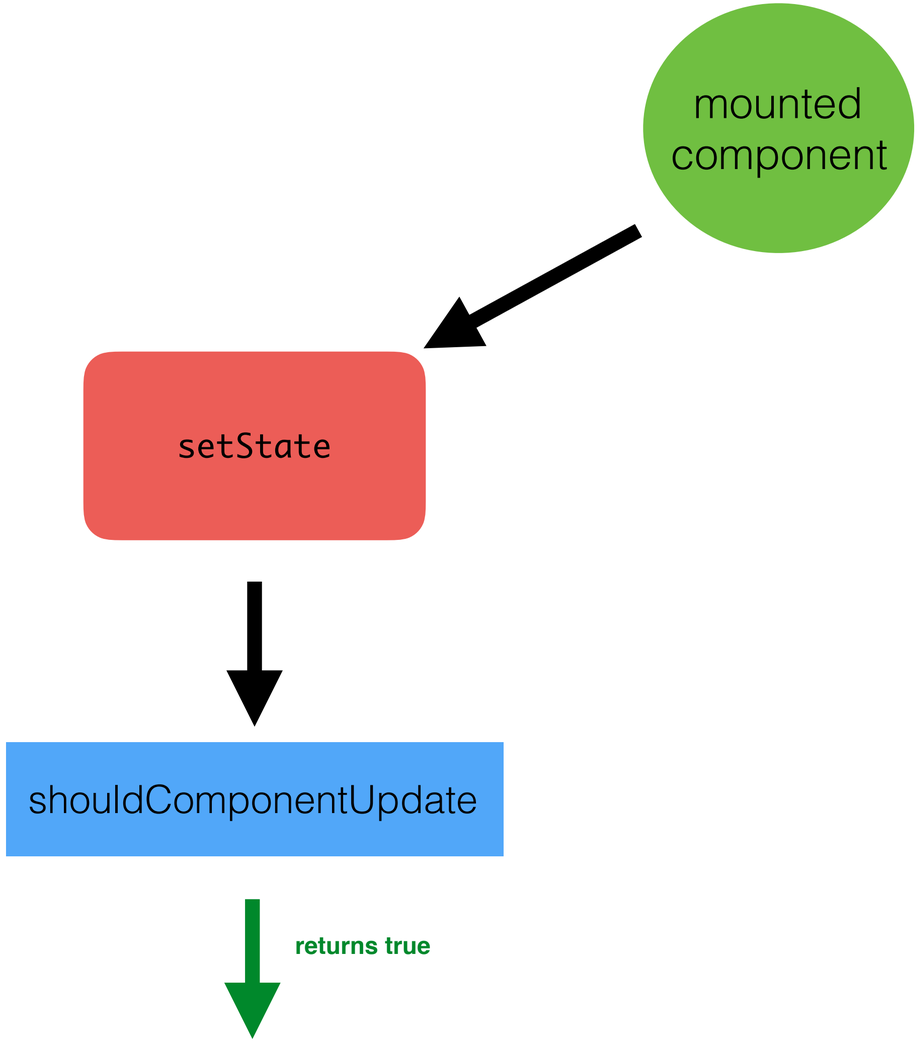

ReactDOM.render
render
mounted
component

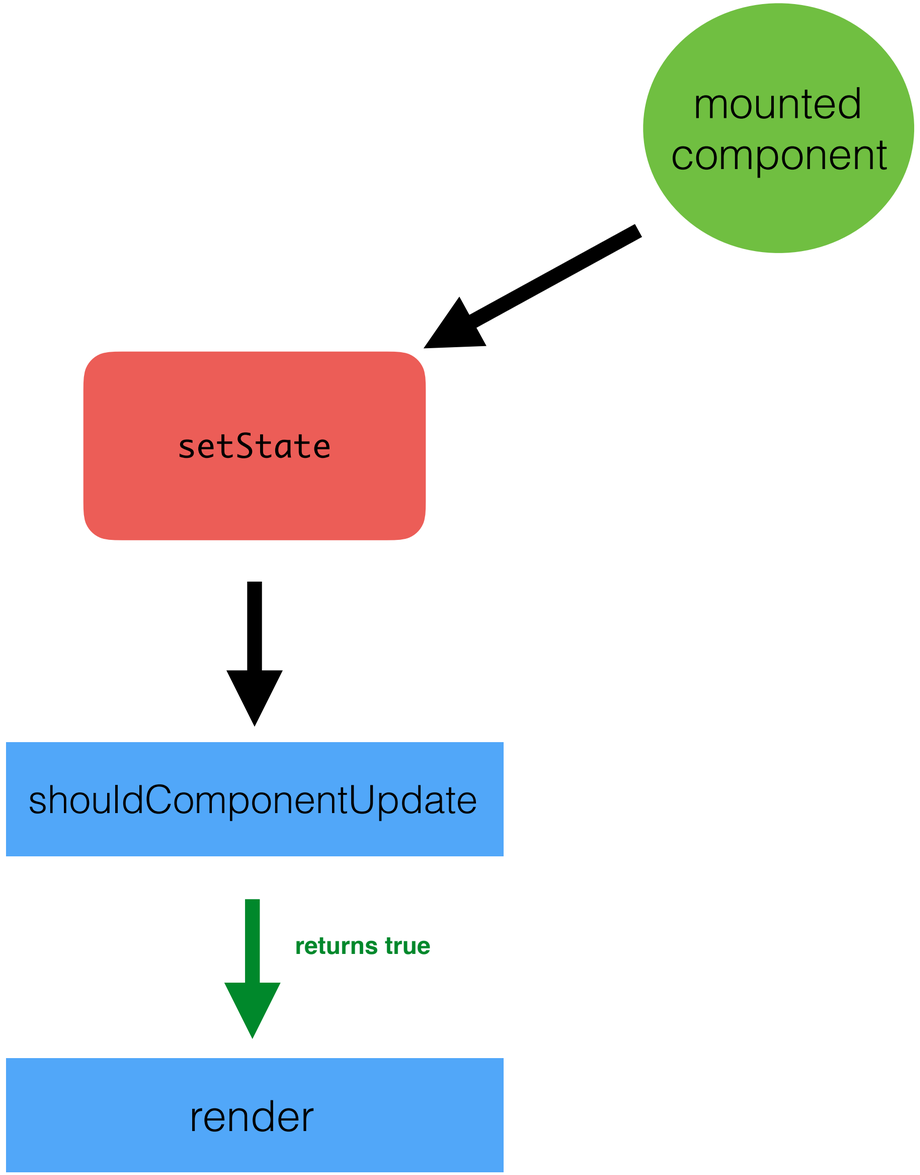

ReactDOM.render
render
mounted
component
componentDidMount

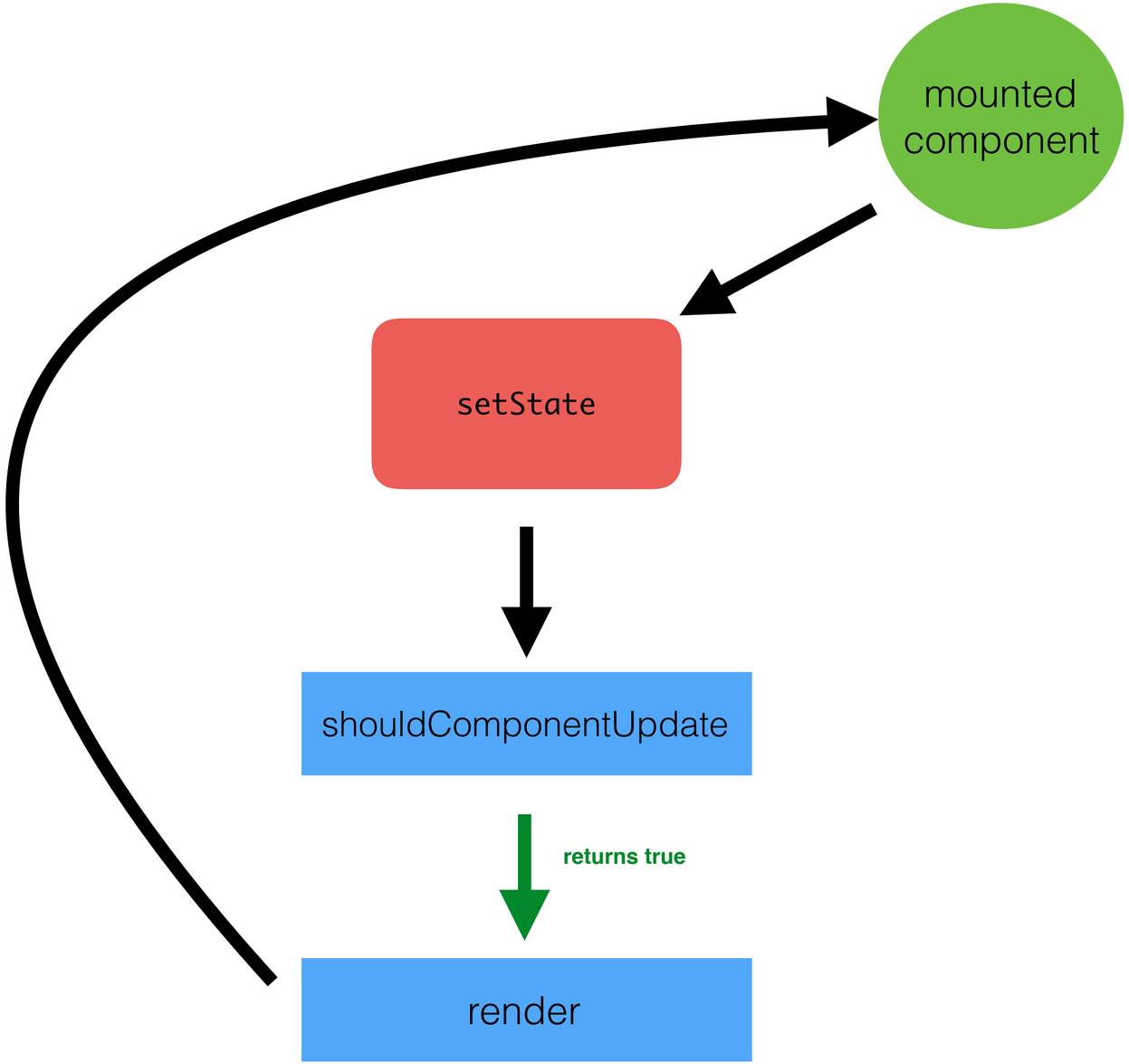

mounted
component

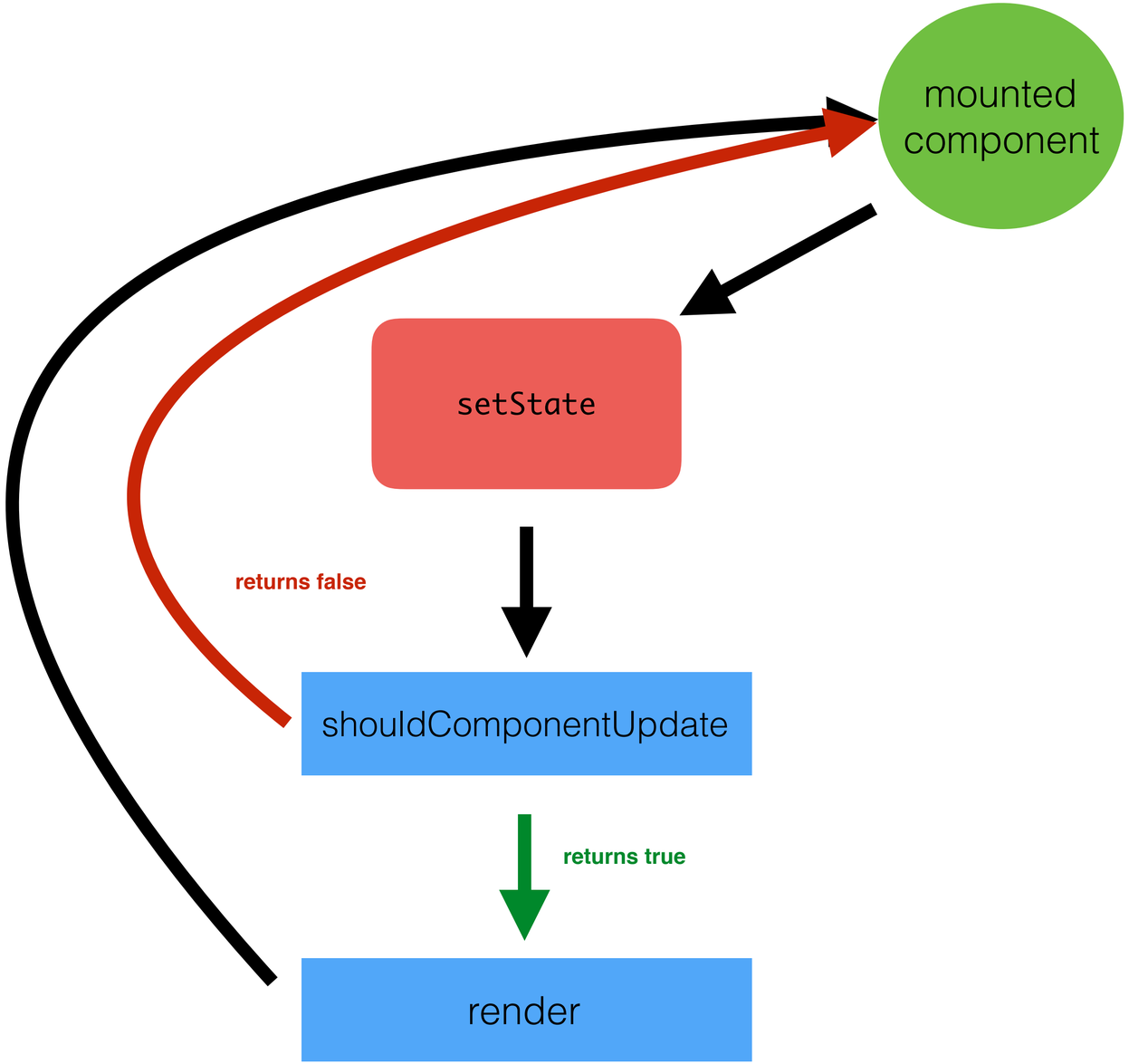

mounted
component
setState

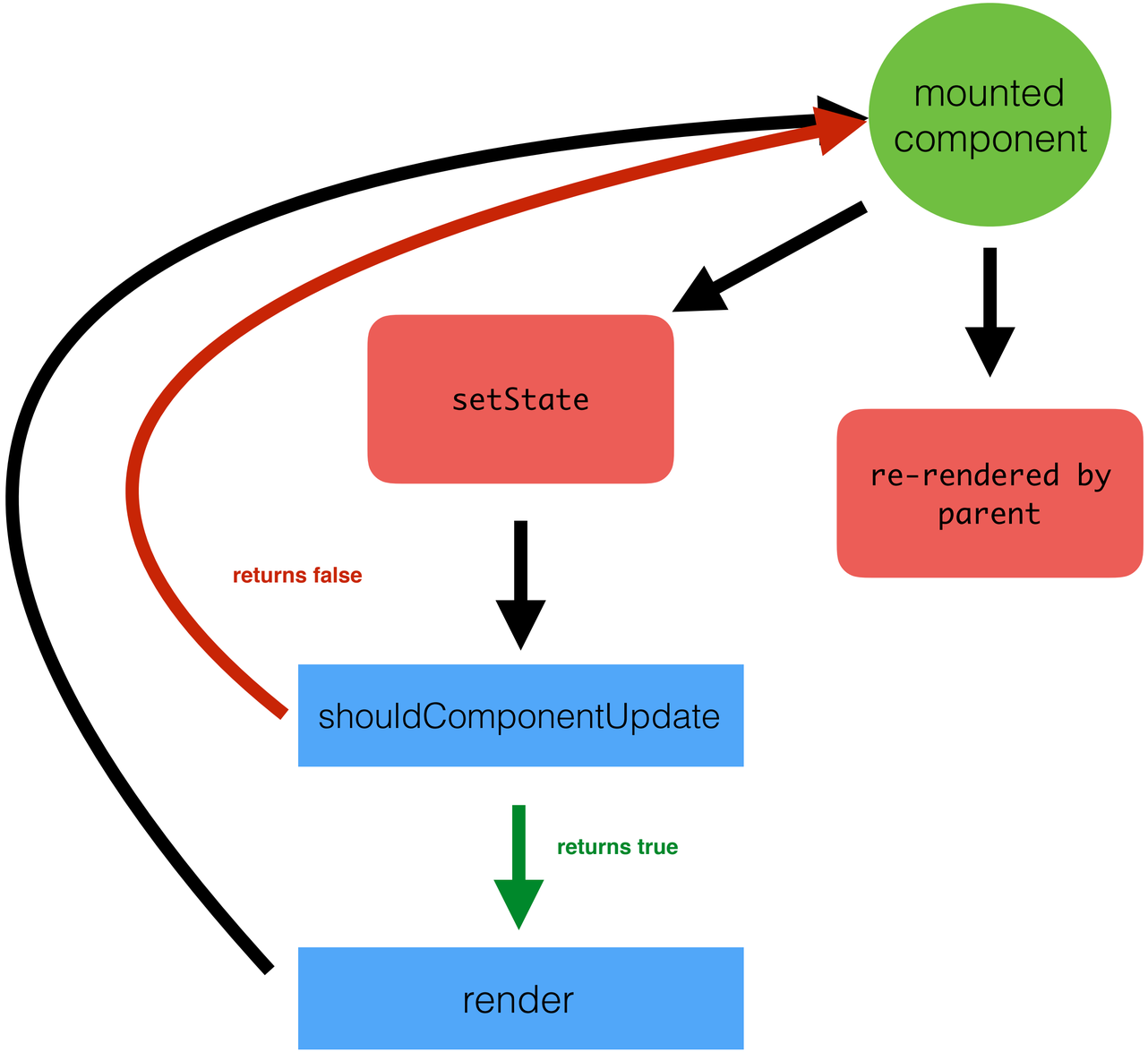

mounted
component
setState
shouldComponentUpdate

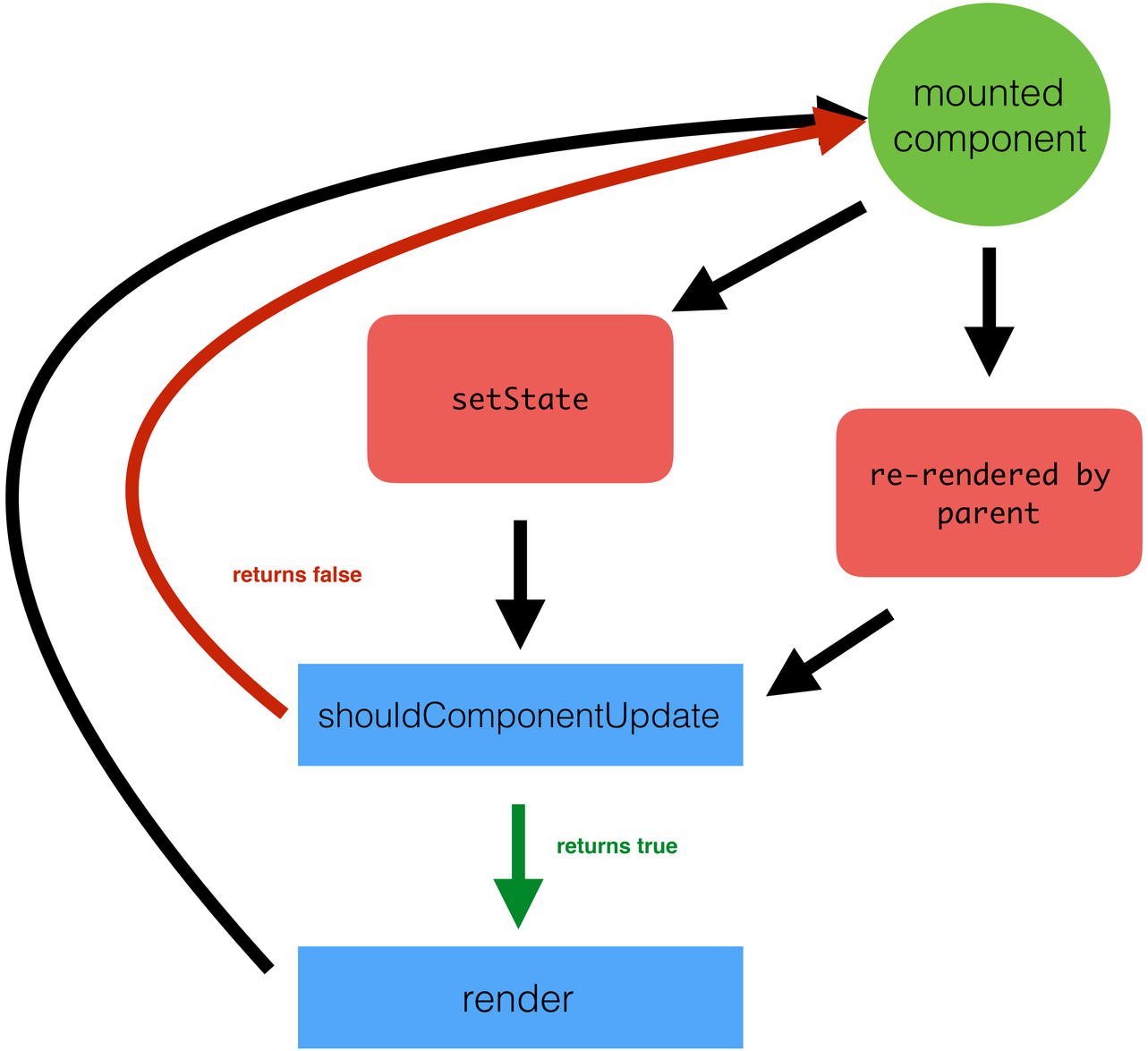

mounted component
setState
shouldComponentUpdate
returns true

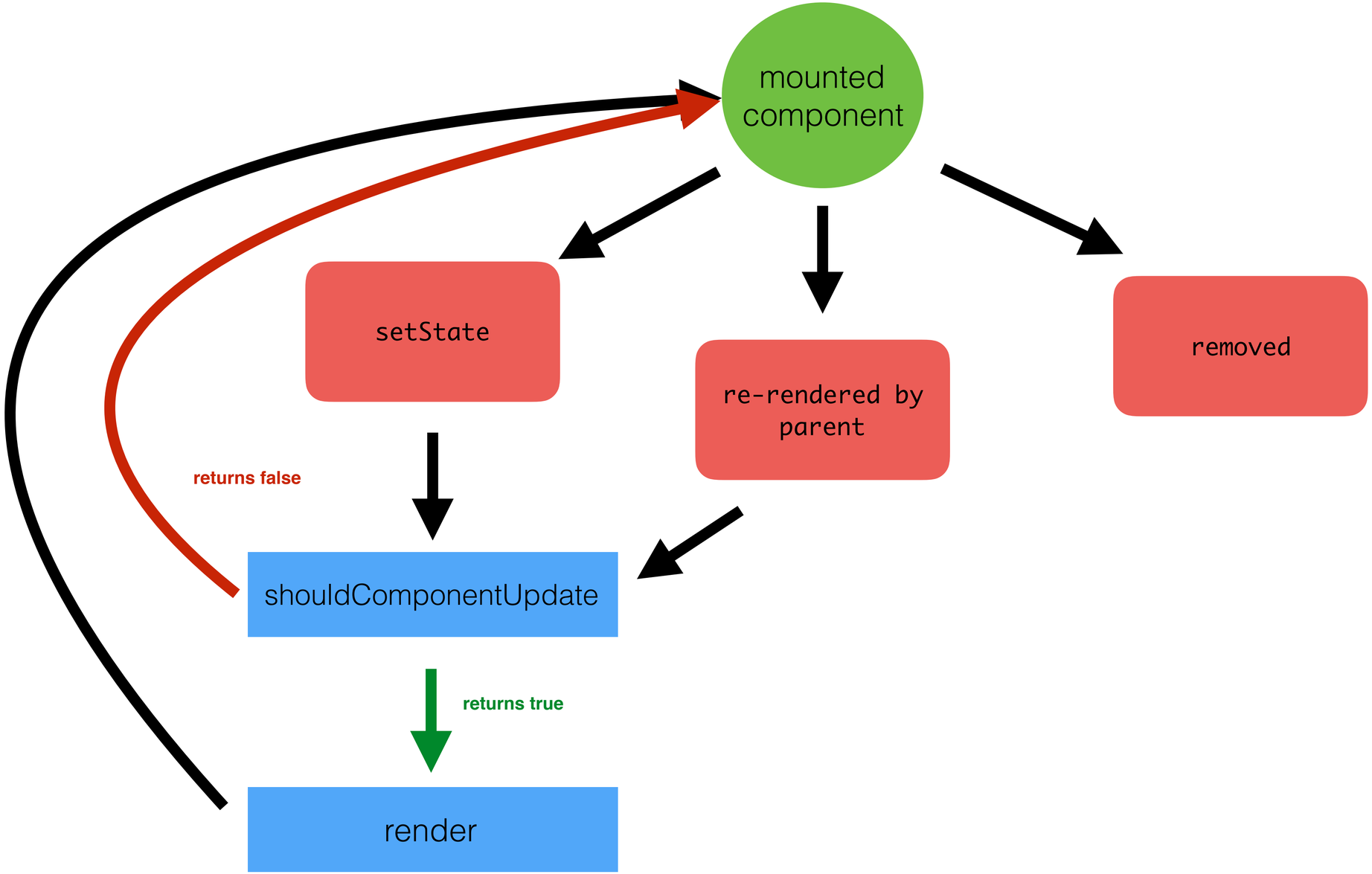

mounted component
setState
shouldComponentUpdate
returns true
render

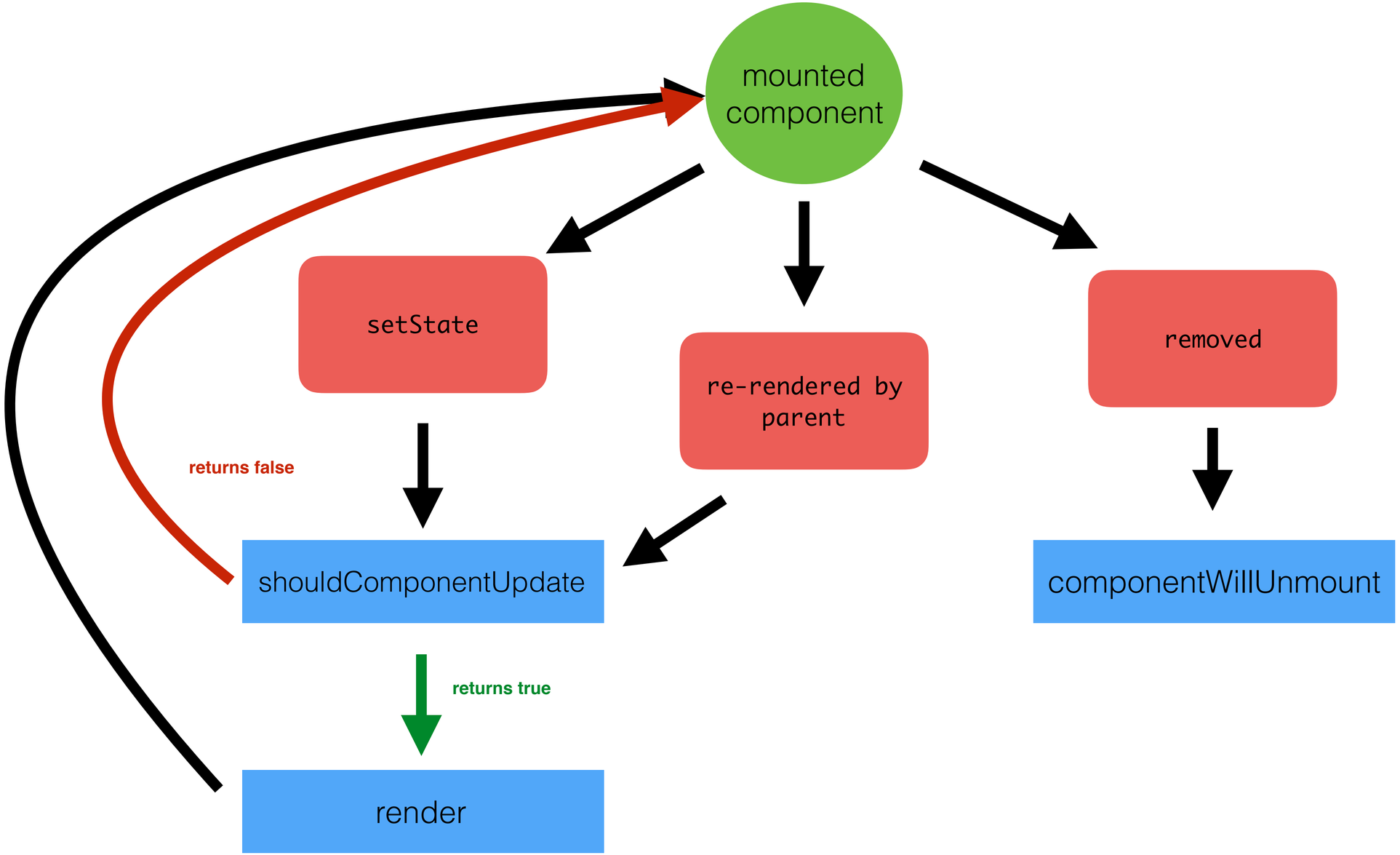

mounted component
setState
shouldComponentUpdate
returns true
render


mounted component
setState
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
removed
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
removed
returns false
shouldComponentUpdate
componentWillUnmount
returns true
render

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 {









}
```

```
class FavoritePuppy extends Component {

  shouldComponentUpdate (                          ) {









}
```

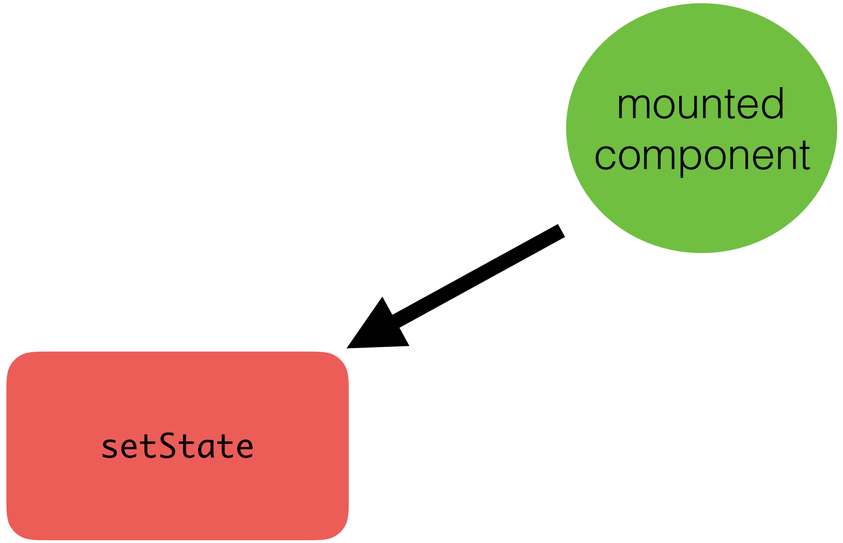
```
class FavoritePuppy extends Component {

  shouldComponentUpdate (nextProps, nextState) {




  }
}
```

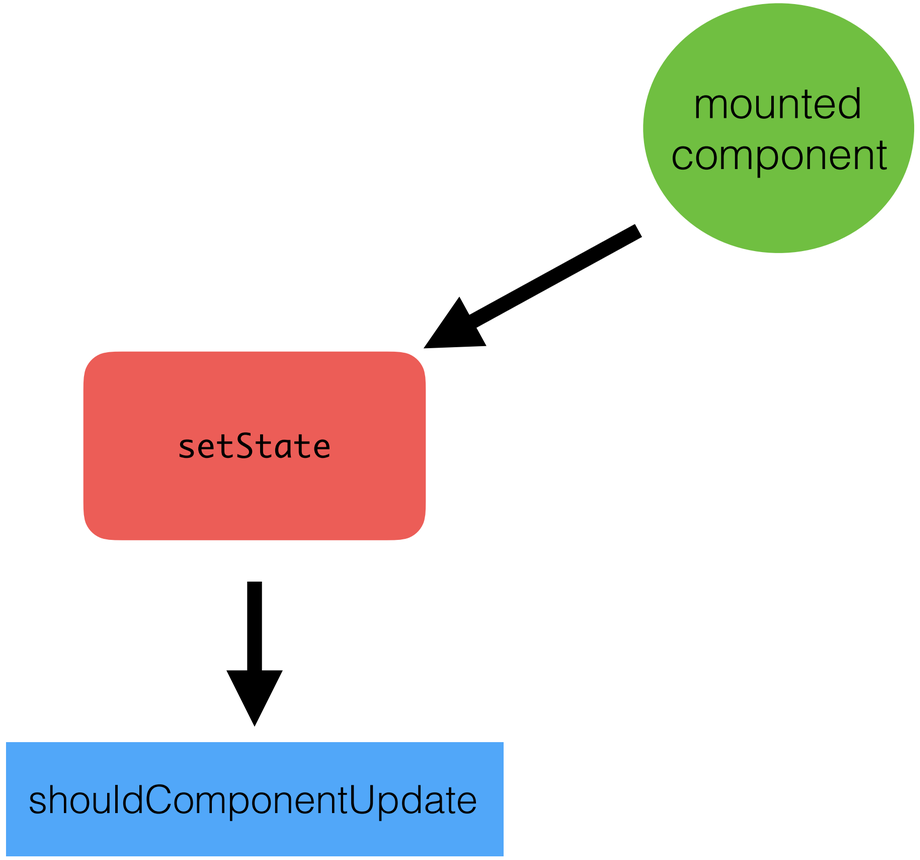
```
class FavoritePuppy extends Component {

  shouldComponentUpdate (nextProps, nextState) {
    if (nextProps.favorite === this.props.favorite) {


    } else {


    }
  }
}
```

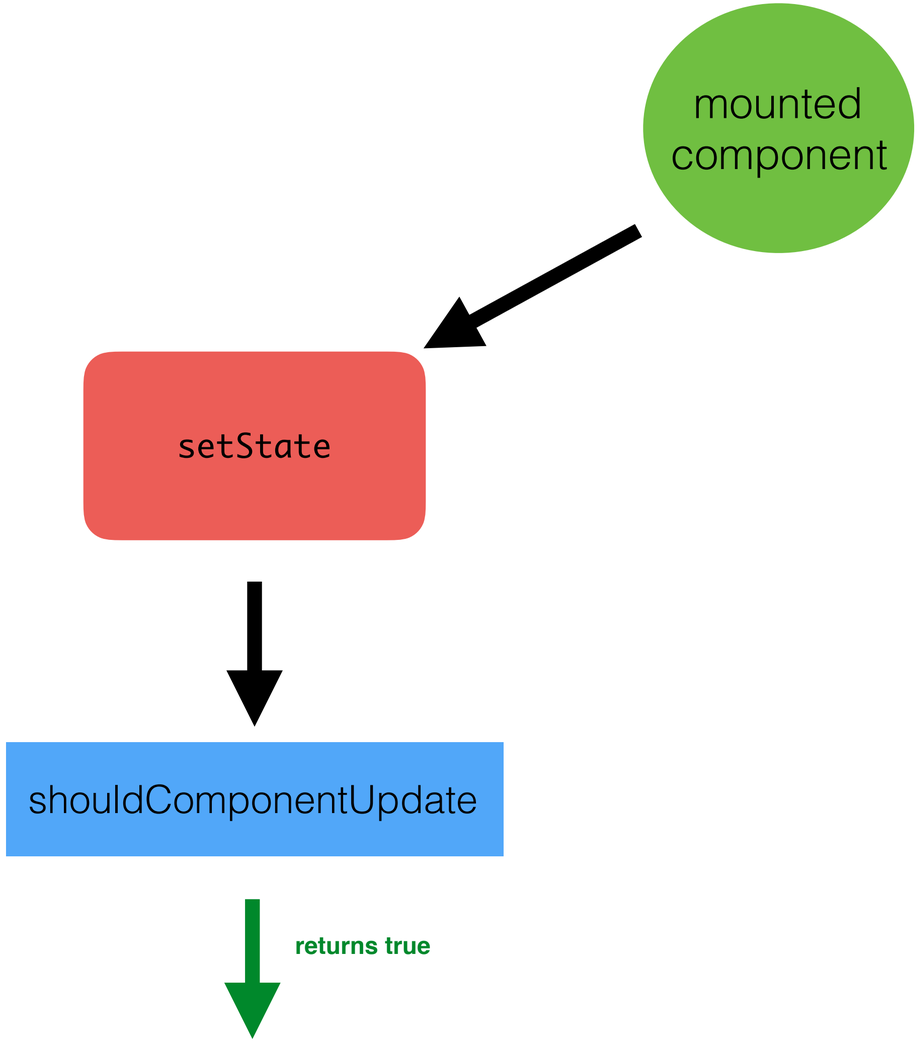
```
class FavoritePuppy extends Component {

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


    }
  }
}
```

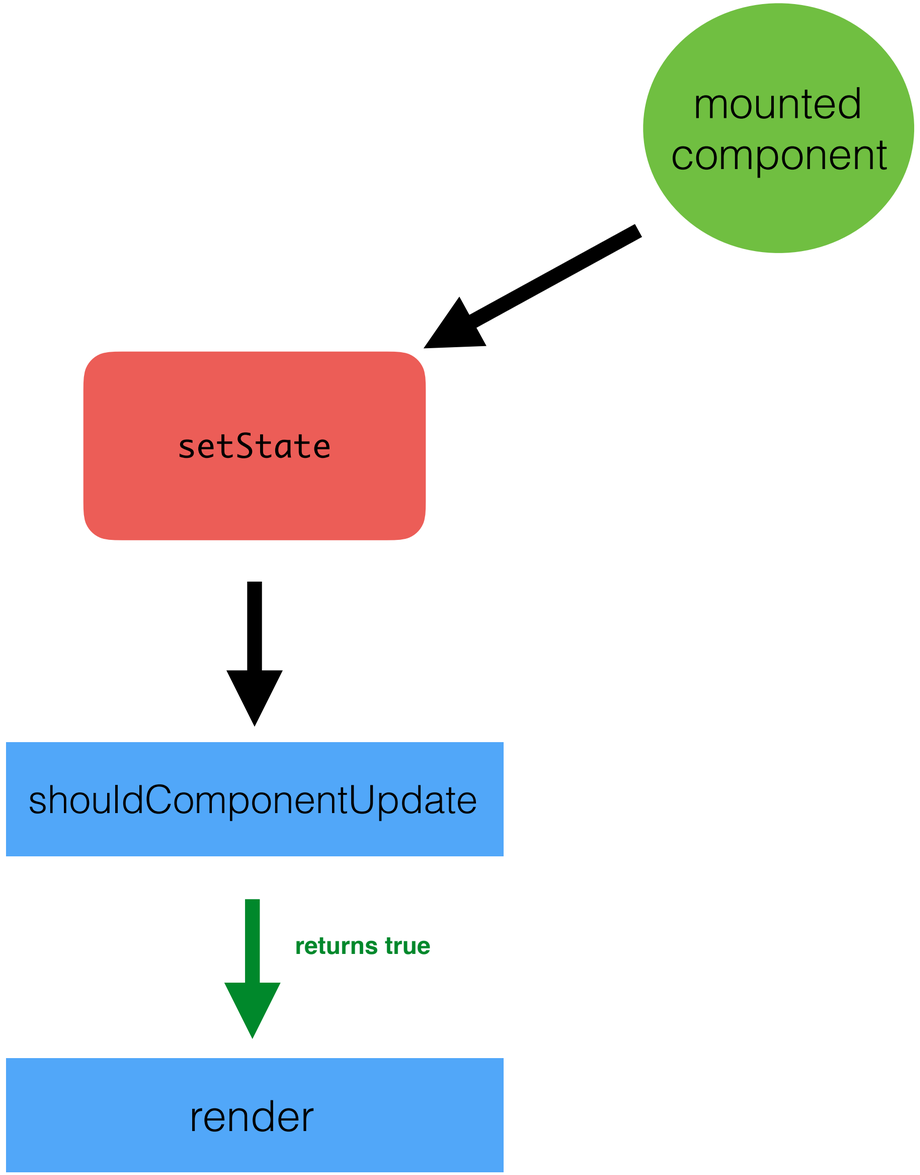
```
class FavoritePuppy extends Component {

  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
    }
  }
}
```

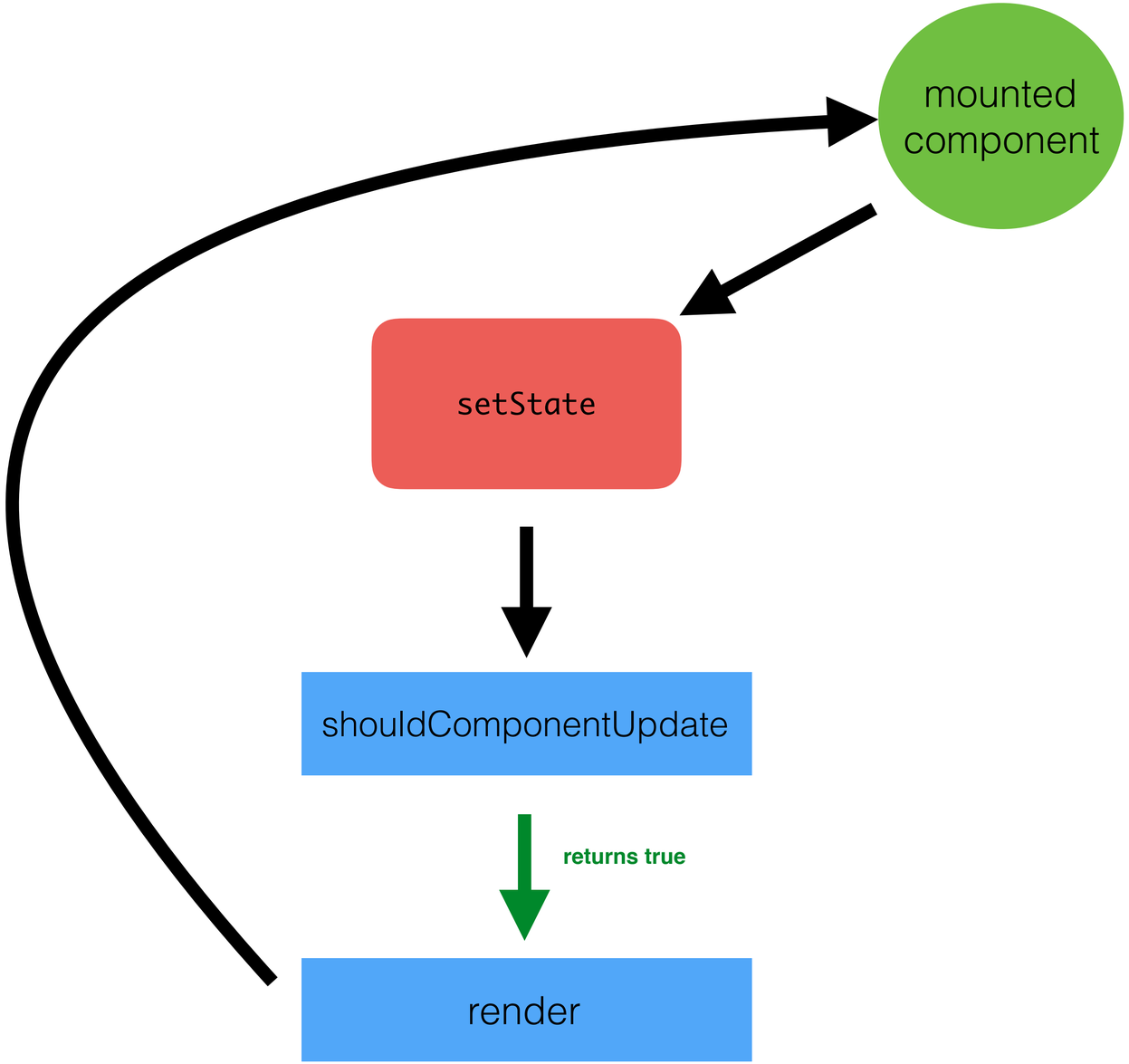

mounted
component

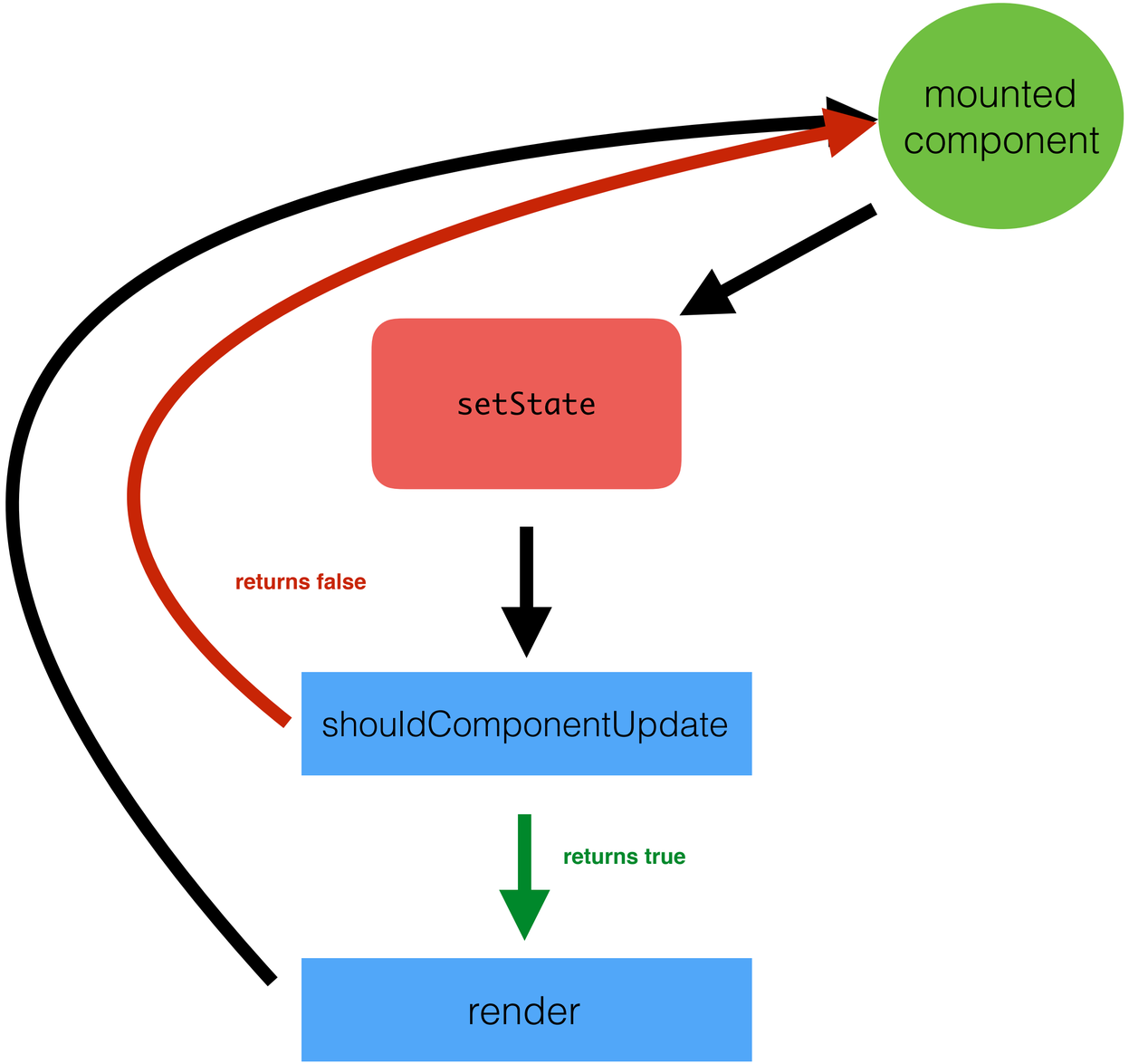

mounted
component
setState

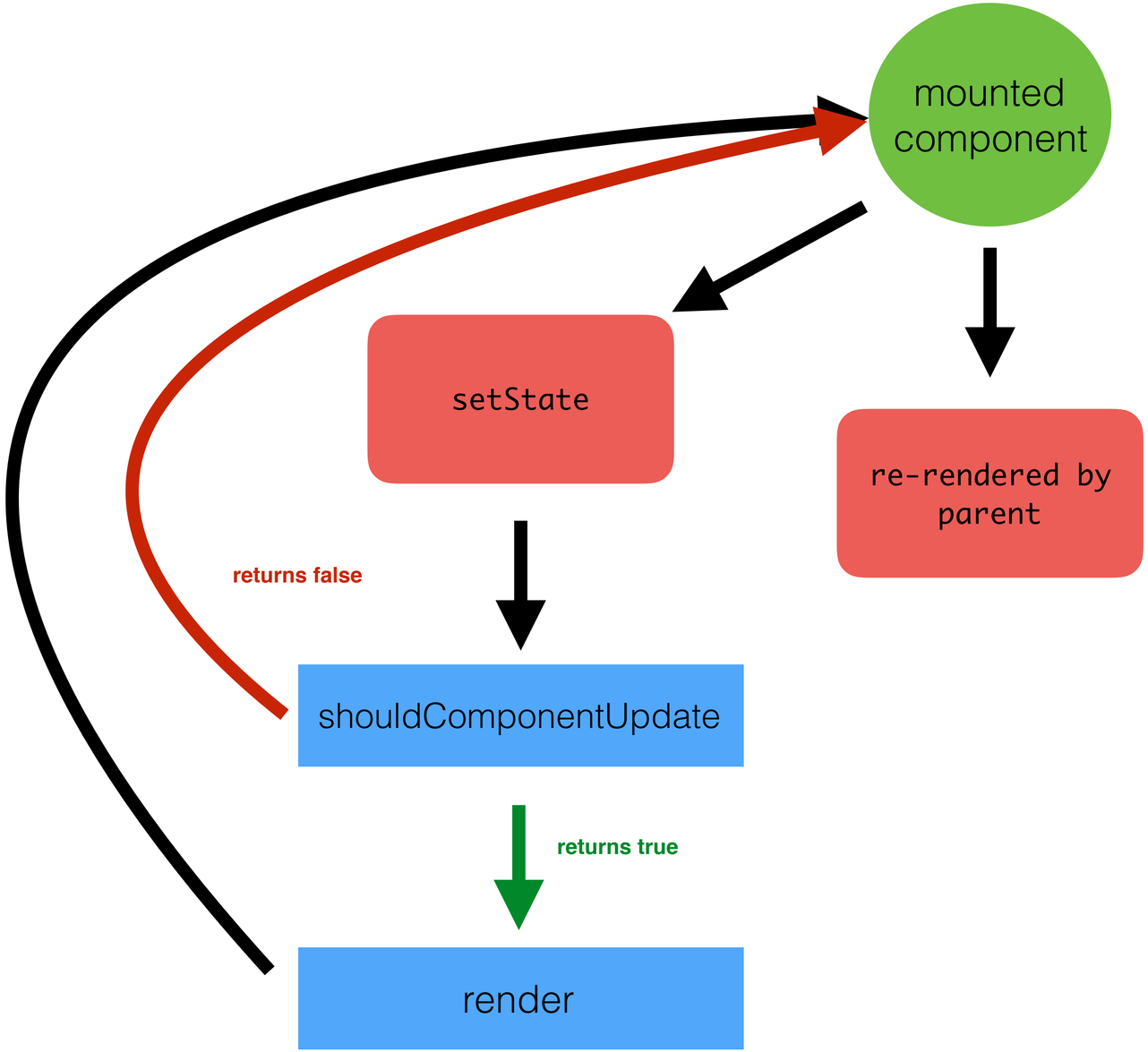

mounted
component
setState
shouldComponentUpdate

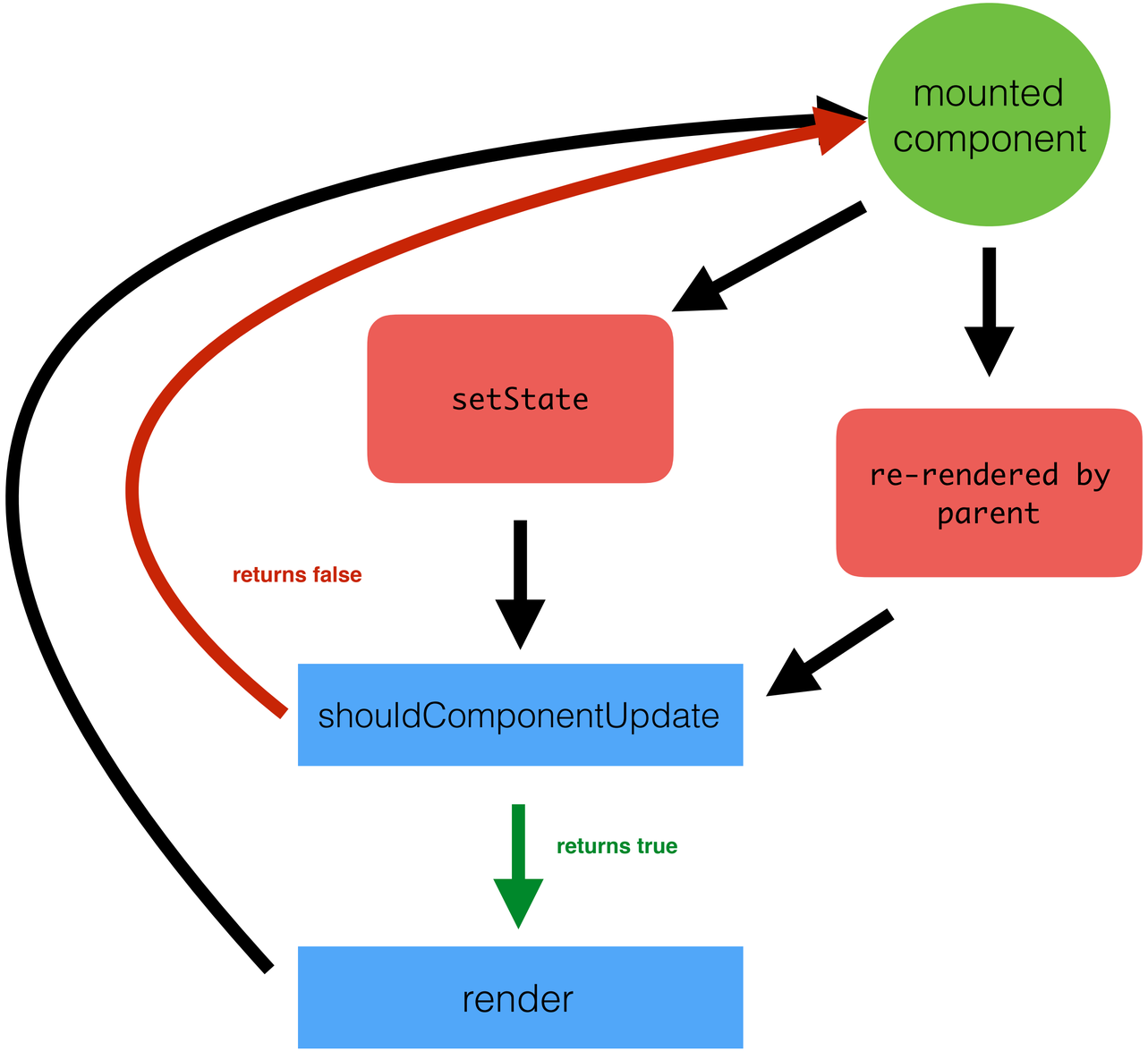

mounted component
setState
shouldComponentUpdate
returns true

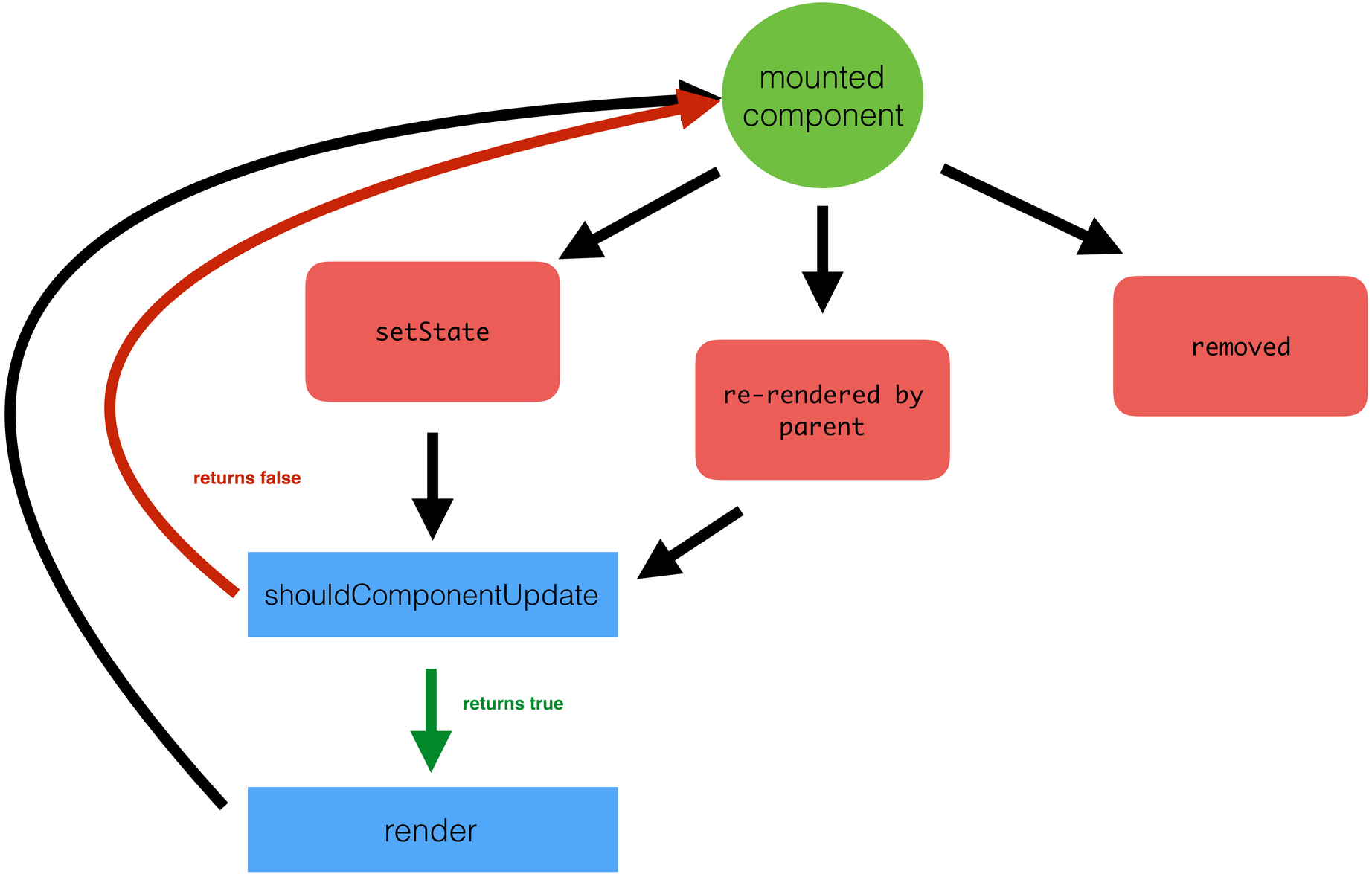

mounted component
setState
shouldComponentUpdate
returns true
render

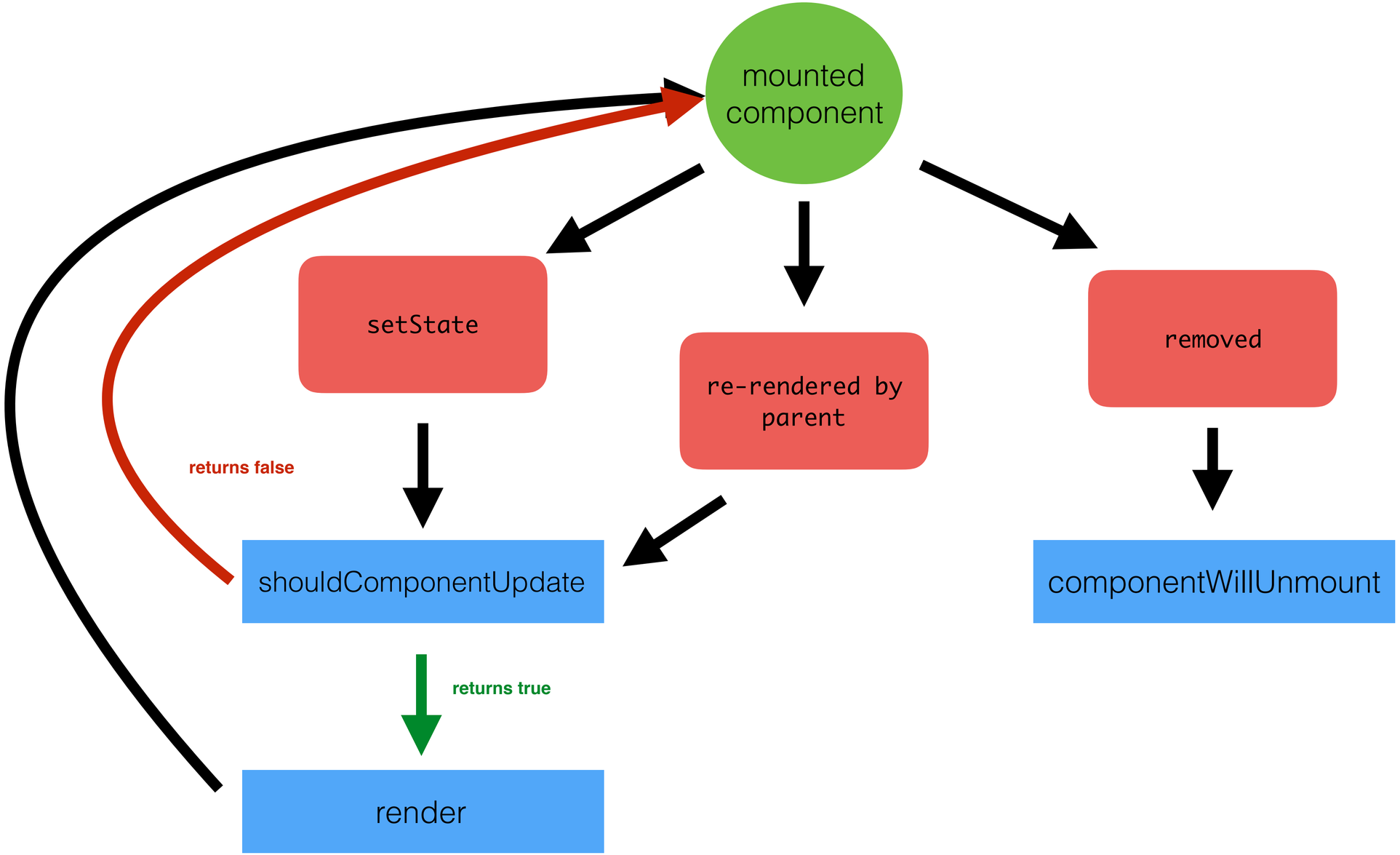

mounted component
setState
shouldComponentUpdate
returns true
render

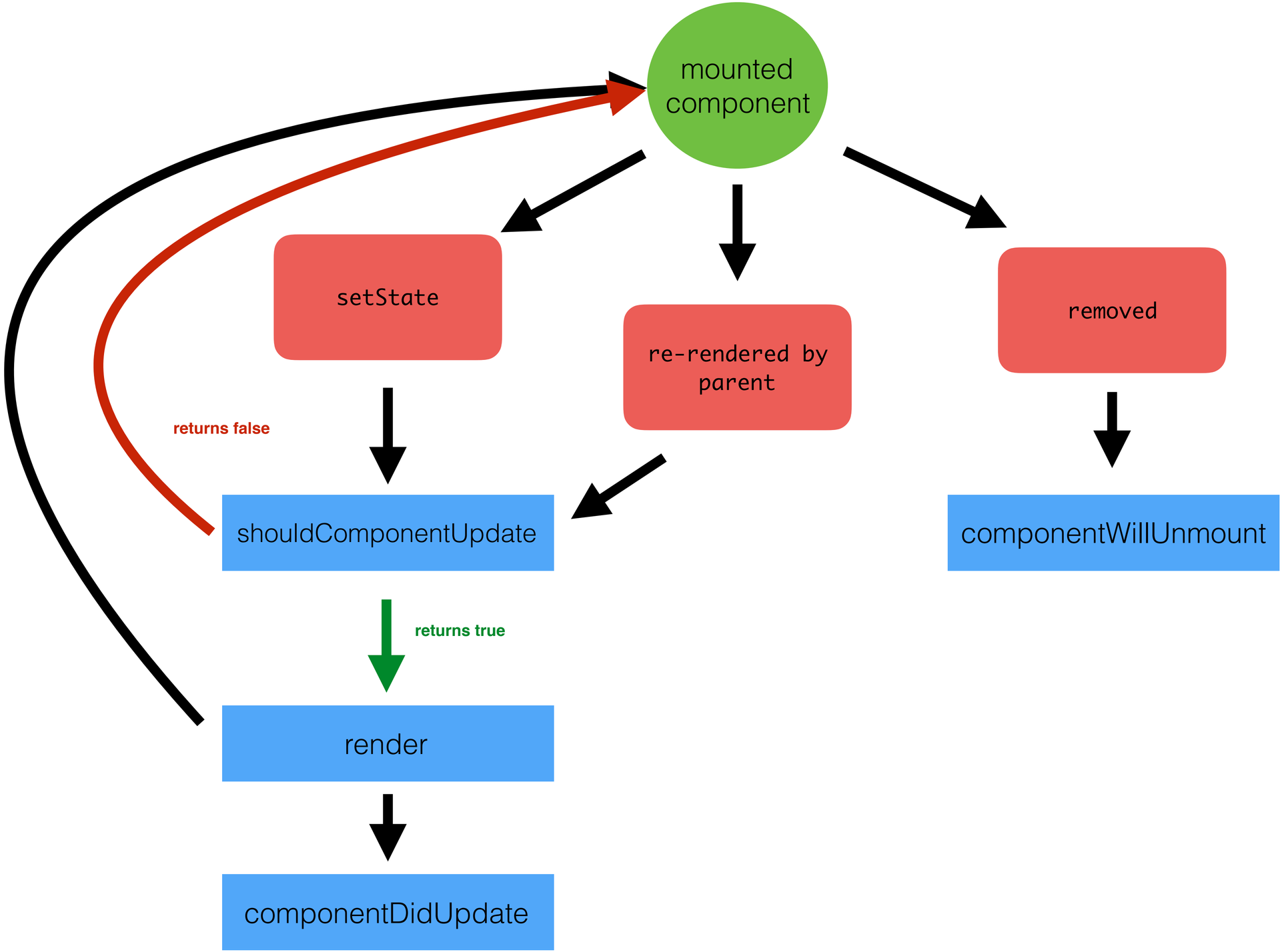

mounted component
setState
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
removed
returns false
shouldComponentUpdate
returns true
render


mounted component
setState
re-rendered by parent
removed
returns false
shouldComponentUpdate
componentWillUnmount
returns true
render


mounted component
setState
re-rendered by parent
removed
returns false
shouldComponentUpdate
componentWillUnmount
returns true
render
componentDidUpdate

COMPONENT DID UPDATE

- Does not occur after the initial render
- Fired whenever a component is updated (either via setState or from being rendered by its parent)
- Chance to compare incoming props with previous props and do something based on that info (like make network requests)

```
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)
  }









}
```

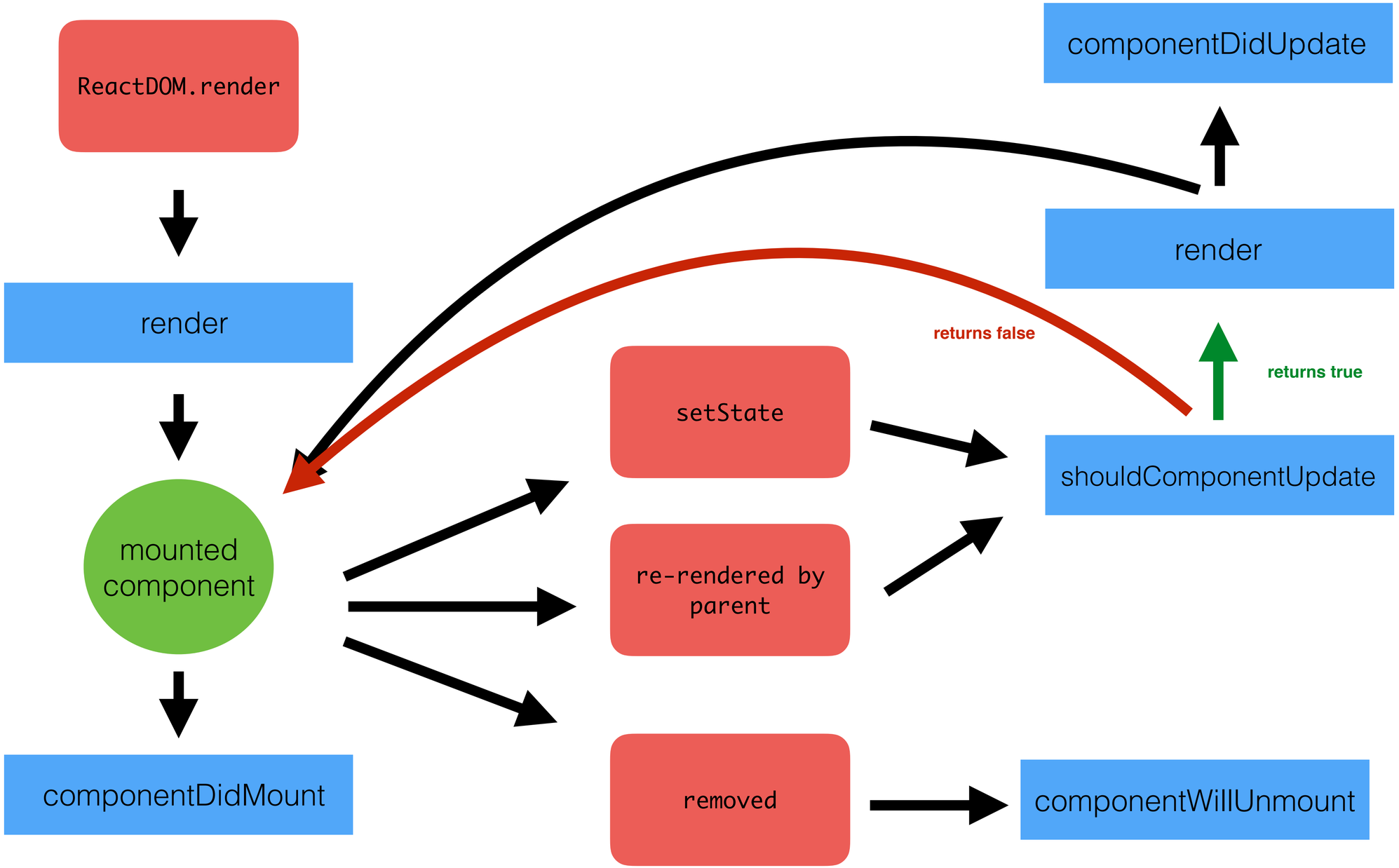
```
class SinglePuppy extends Component {

  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)
  }

  componentDidUpdate (                        ) {




  }
}
```

```
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)
  }

  componentDidUpdate (prevProps, prevState) {









  }
}
```

```
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)
  }

  componentDidUpdate (prevProps, prevState) {
    const newPuppyId = 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)
  }

  componentDidUpdate (prevProps, prevState) {
    const newPuppyId = this.props.match.params.puppyId
    const oldPuppyId = prevProps.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)
  }

  componentDidUpdate (prevProps, prevState) {
    const newPuppyId = this.props.match.params.puppyId
    const oldPuppyId = prevProps.match.params.puppyId
    if (newPuppyId !== oldPuppyId) {
      this.fetchPuppy(newPuppyId)
    }
  }
}
```


ReactDOM.render
render
componentDidMount
mounted component
setState
re-rendered by parent
removed
shouldComponentUpdate
returns false
returns true
render
componentDidUpdate
componentWillUnmount