setInterval in a React app

I'm still fairly new at React, but I've been grinding along slowly and I've encountered something I'm stuck on.

I am trying to build a "timer" component in React, and to be honest I don't know if I'm doing this right (or efficiently). In my code below, I set the state to return an object `{ currentCount: 10 }` and have been toying with `componentDidMount`, `componentWillUnmount`, and `render` and I can only get the state to "count down" from 10 to 9.

Two-part question: What am I getting wrong? And, is there a more efficient way of going about using setTimeout (rather than using `componentDidMount` & `componentWillUnmount`)?

Thank you in advance.

```
import React from 'react';

var Clock = React.createClass({
  getInitialState: function() {
    return { currentCount: 10 };
  },

  componentDidMount: function() {
    this.countdown = setInterval(this.timer, 1000);
  },

  componentWillUnmount: function() {
    clearInterval(this.countdown);
  },

  timer: function() {
    this.setState({ currentCount: 10 });
  },

  render: function() {
    var displayCount = this.state.currentCount--;
    return (
      <section>
        {displayCount}
      </section>
    );
  }

});

module.exports = Clock;
```

javascript reactjs settimeout state

asked Mar 30 '16 at 3:34

Jose
543 1 5 21

`bind(this)` is no longer needed, react does this on its own now. – Derek Mar 30 '16 at 3:36

1 your timer method does not update currentCount – Bryan Chen Mar 30 '16 at 3:37

1 @Derek are you sure? I just got mine working by adding `this.timer.bind(this)` as this.timer on it's own did not work – The worm Nov 23 '16 at 10:51

1 @Theworm @Derek is wrong, kind of. React.createClass (which is deprecated) autobinds methods, but `class Clock extends Component` doesn't auto bind. So it depends on how you're creating your components whether you need to bind. – CallMeNorm Jan 10 at 18:54

2 Answers

I see 4 issues with your code:

- In your timer method you are always setting your current count to 10
- You try to update the state in render method
- You do not use `setState` method to actually change the state
- You are not storing your intervalId in the state

Let's try to fix that:

```
componentDidMount: function() {
   var intervalId = setInterval(this.timer, 1000);
   // store intervalId in the state so it can be accessed later:
   this.setState({intervalId: intervalId});
},

componentWillUnmount: function() {
   // use intervalId from the state to clear the interval
   clearInterval(this.state.intervalId);
},

timer: function() {
   // setState method is used to update the state
   this.setState({ currentCount: this.state.currentCount -1 });
},

render: function() {
    // You do not need to decrease the value here
    return (
      <section>
       {this.state.currentCount}
      </section>
    );
}
```

This would result in a timer that decreases from 10 to -N. If you want timer that decreases to 0, you can use slightly modified version:

```
timer: function() {
   var newCount = this.state.currentCount - 1;
   if(newCount >= 0) {
       this.setState({ currentCount: newCount });
   } else {
       clearInterval(this.state.intervalId);
   }
},
```

edited Oct 28 '16 at 16:58
Dherik
2,133 1 23 45

answered Mar 30 '16 at 3:41
dotnetom
18.8k 8 29 43

Thank you. This makes a lot of sense. I'm still very much a beginner and I'm trying to get a hold of how state works and what goes in which "chunks", like render. – Jose Mar 30 '16 at 3:47

I am wondering, though, is it necessary to use componentDidMount and componentWillUnmount to actually set the interval? EDIT: Just saw your most recent edit. :) – Jose Mar 30 '16 at 3:49

@Jose I think `componentDidMount` is the right place to trigger the client side events, so I would use it to initiate the countdown. What other method are you thinking about for initializing? – dotnetom Mar 30 '16 at 3:52

I didn't have anything else in particular in mind, but it seemed clunky to use so many "chunks" inside a component. I suppose it is just myself getting accustomed to how the bits and pieces work in React. Again, thank you! – Jose Mar 30 '16 at 3:54

@dotnetom why would mine be potentially failing saying cannot read currentCount of undefined in the timer method. I have mine set up exactly the same as yours just have the ES6 way of doing getinitialstate. i.e. constructor(props) { super(props); this.state = { currentCount: 10 }; }; – The worm Nov 23 '16 at 10:10

Updated 10-second countdown using `class Clock extends Component`

```
import React, { Component } from 'react';

class Clock extends Component {
  constructor(props){
    super(props);
    this.state = {currentCount: 10}
  }
  timer() {
    this.setState({
      currentCount: this.state.currentCount - 1
    })
    if(this.state.currentCount < 1) {
      clearInterval(this.intervalId);
    }
  }
  componentDidMount() {
    this.intervalId = setInterval(this.timer.bind(this), 1000);
  }
  componentWillUnmount(){
    clearInterval(this.intervalId);
  }
  render() {
    return(
      <div>{this.state.currentCount}</div>
    );
  }
}

module.exports = Clock;
```

edited Apr 13 at 6:13

answered Apr 12 at 11:29

Greg

96 3 7