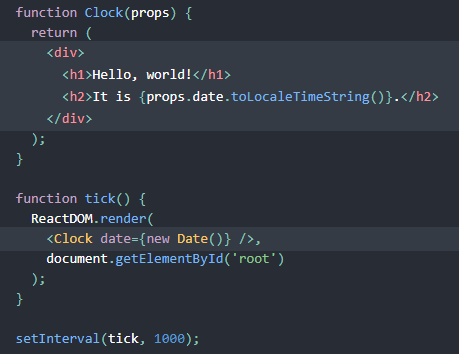
Resource

<https://reactjs.org/docs/state-and-lifecycle.html>

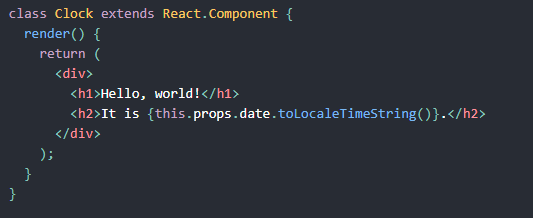
Currently we are using setInterval() to manually calling function



By using states, without calling setInterval(), components can update themselves.

State are private to component, and are fully controlled by a component

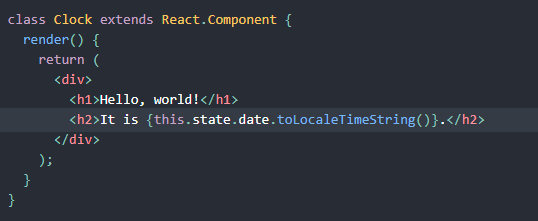
Class version of clock:



Render() will be called each time an update happens, but what update? Props?

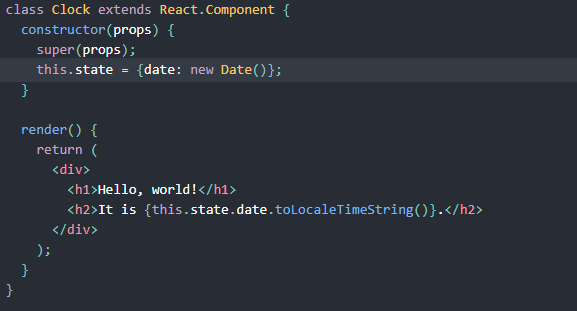
Remove the props and replace it with state in 3 steps:

1. Change this.props.date to this.state.date



2.

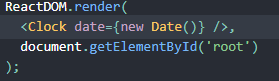
Add a constructor, constructor always need to pass in props; in constructor we initiate the state.



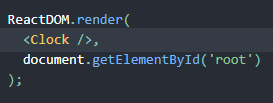
Super(props) are always needed to be called in constructor.

1. remove the props of date when we create Clock component.

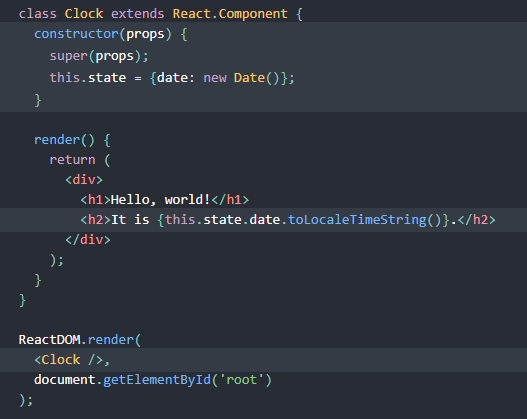
From



To



Result looks like



But now, the clock is freezed, it doesn’t update itself.

We need to add Lifecycle method to let it update itself.

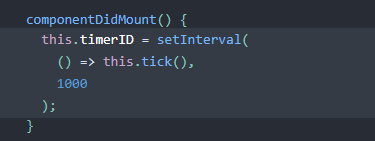
Mounting: whenever the Clock is rendered to the DOM for the first time, it’s called mounting.

Unmounting: when the DOM produced by the Clock is removed.

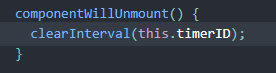
Lifecycle Method: Special methods that will run when the components mount or unmount.

componentDidMount(): run after the component has rendered to the DOM.

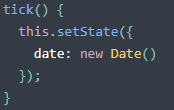
We set up a timer inside componentDidMount()



We tear down the timer in component will unmount.



Finally in the tick() we update the state



Question in my mind:

When will a component mount?

Ans: when a component is rendered to the DOM, when the component is showing on your screen, it’s being mounted.

When will a component unmount?

Ans: when a component no longer getting displayed on your screen, the component is unmounted.

Examples in our project, in the navbar, when we create another link in the navbar, the entire display of the DOM changes, and that’s the moment of the old component on the DOM get unmounted and the new component get mounted.