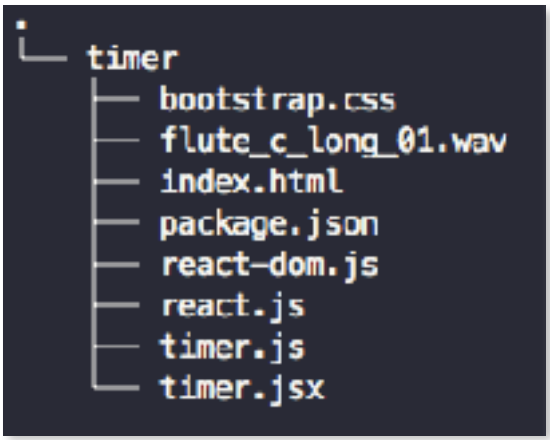
Build a Timer App with React

This exercise is to create a simple React timer countdown app, counting down from a specified time, and sounding off an alarm at the end.

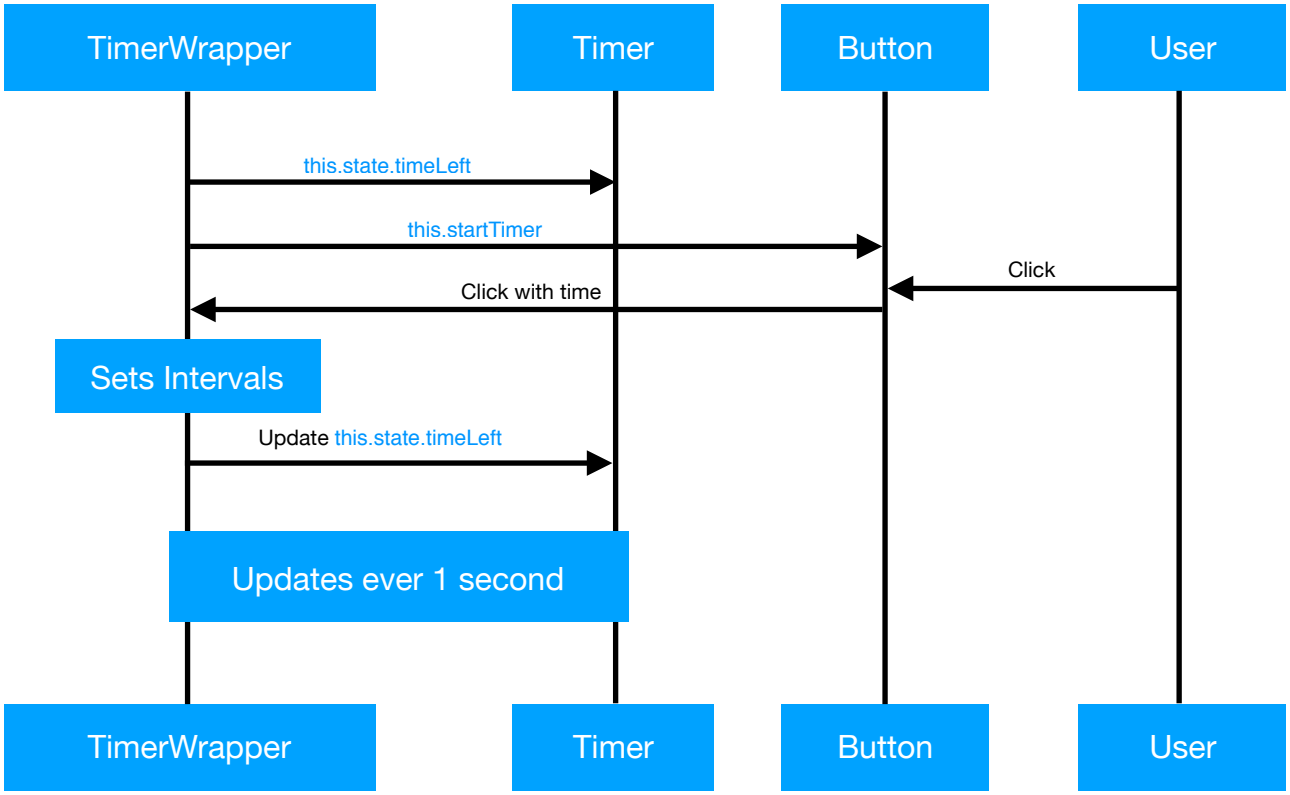
1. You should be greeted with a project structure like the following.



2. Just to make sure that your development environment is in sync, run npm install within the **timer** folder before you develop.

3. For starters, we are editing timer.jsx, and Babel will transpile your code into timer.js. The included HTML file serves as a shell for React to mount your components with timer.js. Note that we are using Bootstrap here for basic styling.

4. Open up timer.jsx and you should find 3 empty main components, namely TimerWrapper, Timer, and Buttons. We are mounting TimerWrapper on the DOM, it should be well assumed that TimerWrapper is the main component containing all other components in this case.



5. The timer app execution is illustrated in the figure below. Note the flow of events from top to bottom.

6. TimerWrapper renders Timer and Buttons by passing TimerWrapper's states as properties. User interacts with a button, which triggers an event in the button. The event in the button calls the function in TimerWrapper with the time value in seconds.TimerWrapper sets the interval and updates Timer. The updates continue until there are 0 (zero) seconds left.

7. First of all, we need to be able to save time left and reset the timer. When the app first loads, the timer app shouldn't run, so we need to set the time state to null. You will come to appreciate the handiness of this, as setting it null as opposed to 0 will allow us to tell the difference between the first load, and when the time is up. Create a 2 data fields in the TimerWrapper state, namely timeLeft and timer and set them to null. timeLeft will save the time left state of the component, while timer will hold the reference to the incoming setInterval function that will do the countdown.

8. Create a handler for when a user clicks a button. If a user clicks a button when the timer is already running, then we need to clear the previous interval and start anew. The first thing that this handler should do is to stop the previous countdown by clearing setInterval. You can set and clear intervals in a browser using browser API methods, clearInterval() and setInterval(). After clearing the interval, set a new one and that reduces the current time left by 1, and clears the interval if the time left is 0. The interval should end with the handler setting timeLeft state with a newly updated one. Use setState() for this, so that our React component picks up the changes.

9. In the mandatory render() method for TimerWrapper method, create your interface for your timer app. It should have 3 Buttons, and maybe a simple title. Feel free to use Bootstrap classes for styling and placement. Make sure that the button's props are passed with the startTimer method, and a respective time values for countdown. Next, add our Timer component, which we pass in our our timeLeft state for it to display the time left if there's any, and display nothing if there isn't. Lastly, create an audio element, passing in the included .wav file into the src attribute. Google if you're not sure as to how to create an audio element, and playing them using JavaScript.

10. Going into the Timer component, the crux of this component is just to display the passed down timeLeft state if there's any, and display nothing if there isn't anything left. If the timer runs out, it should play the audio element that you've just created in the TimerWrapper component.

11. The Button component is slightly more verbose. Noting that we previously had a handler in TimerWrapper for when a user clicks a button. Make sure that your Button component captures user clicks and executes the passed down handler. You can use onClick event handler to launch a certain callback for when users click on your component. Make sure that your Button component also displays the time that the countdown is going to start from for each buttons.

12. You can see here that most of the logic resides in TimerWrapper, while Button and Timer are more of a complementary components to TimerWrapper.

13. Run npm run build to build a distributable timer.js from timer.jsx. The timer app should run by loading the index.html file in your browser.