**Pomodoro Timer**

The Pomodoro Technique is a time management method developed by Francesco Cirillo in the late 1980s.  
The technique uses a timer to break down work into intervals, traditionally 25 minutes in length, separated by short breaks. Each interval is known as a pomodoro, from the Italian word for 'tomato', after the tomato-shaped kitchen timer that Cirillo used as a university student.

You will implement a Pomodoro timer that follows these steps (simplified from the original technique):

1. Set the focus duration (default to 25 minutes, no less than 5 or more than 60).
2. Set the break duration (default to 5 minutes, no less than 1 or more than 15).
3. When the user clicks the "play" button, the timer starts.
4. When the focus time expires, an alarm plays and then the break timer starts.
5. When the break time expires, the alarm plays again and then the focus timer starts.

This application uses [Bootstrap 4](https://getbootstrap.com/) for styling and [Open-Iconic icons](https://useiconic.com/open) for icons.

This project is designed to test your ability to work with rendering and state management using React. Before taking on this module, you should be comfortable with the following:

* Installing packages via NPM.
* Running tests from the command line.
* Writing React function components.
* Using hooks like useState()
* Debugging React code through console output

**Project setup**

Follow the instructions below to get this project up and running on your own machine:

* Sync this challenge with your computer so that you can work on the project locally.
* Run npm install.

To run the tests, you can run the following command:

npm test

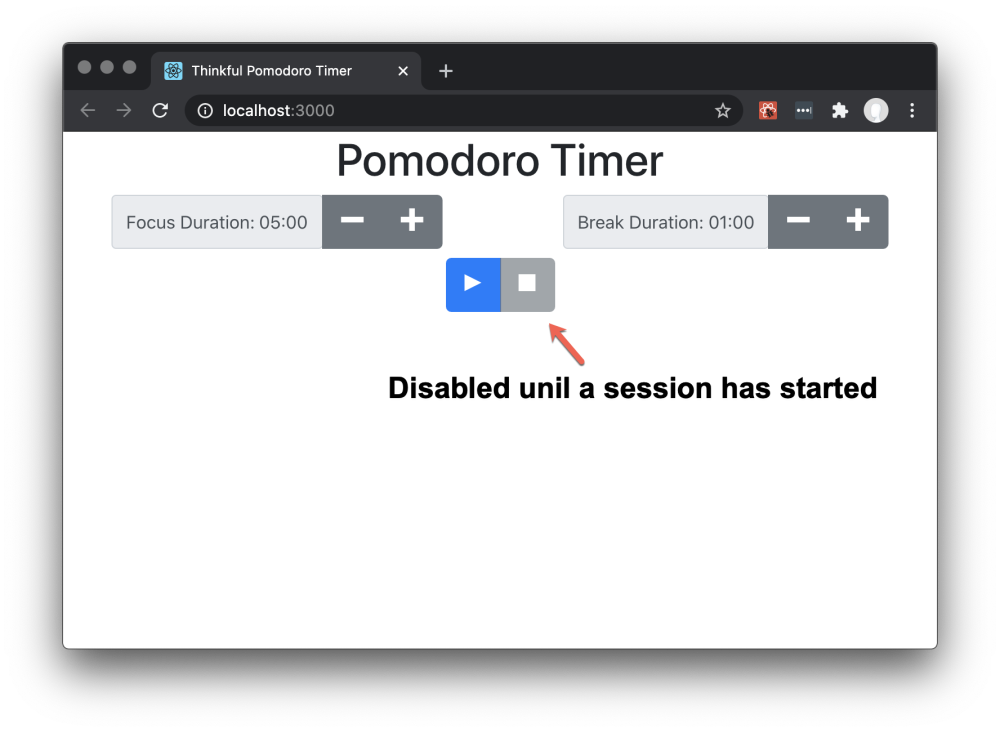
You can run the application using the following command.

npm start

**Initial Screen**

The initial screen lets the user set the length of the focus and break and break sessions.

1:1



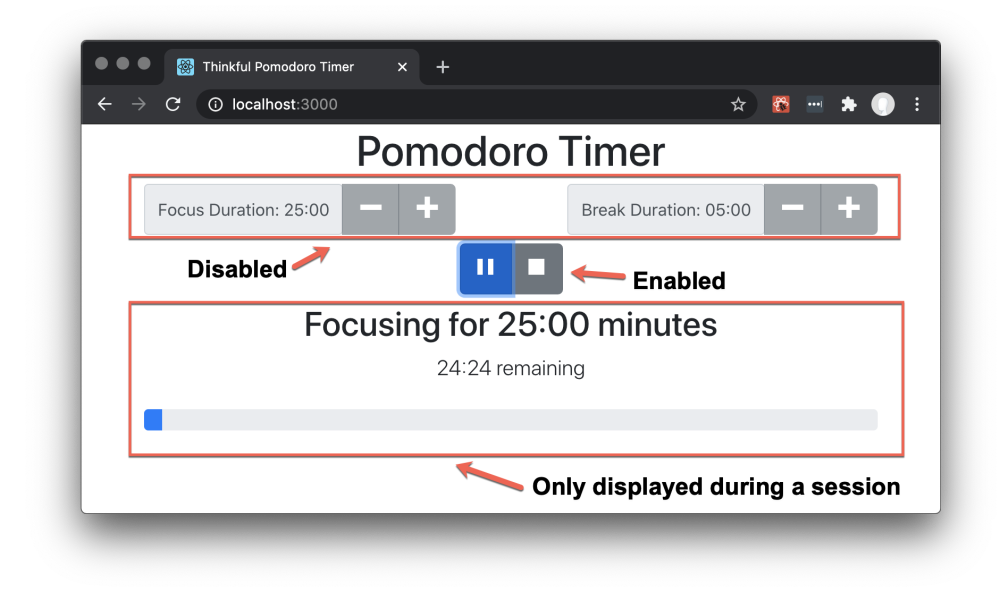
The "stop" button is disabled on the initial screen because the user has not yet started the timer.

When the user clicks the "play" button, the timer will always start a new focus session.

**Active Session Screen**

After the user clicks the "play" button, the buttons to change the focus and break duration are disabled, and the session timer appears.

1:1



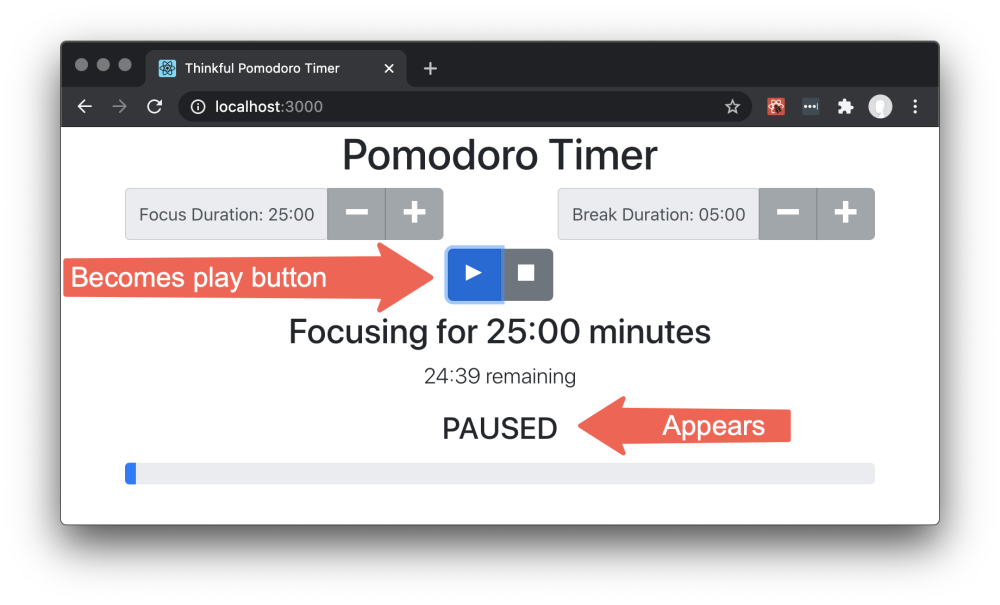
The session timer shows the type of session, either "Focusing" or "On Break", the total duration of the session, the time remaining, and a progress bar showing how much of the session is complete.

**Hint**: In React, if you want to hide/show something, use conditional rendering rather than CSS styles. In general, conditional rendering is preferred. The tests in this project require the use of conditional rendering to show/hide the session timer.

**Paused Session Screen**

If the user clicks the "pause" button, "paused" appears below the time remaining.

1:1



The session timer shows the type of session, either "Focusing" or "On Break", the total duration of the session, the time remaining, and a progress bar showing how much of the session is complete.

**Stopping a session**

Stopping a session returns the application to the initial screen and the user is able to change the focus and break duration.

Clicking the "play" button will always start a new focus session.

**Specific instruction**

1. The code has various TODO items that should help you build the project as expected. With that said, feel free to make the changes you feel are necessary to accomplish the tasks.
2. Break up the code into at least two additional components that have a single responsibility.
3. The user cannot change the duration of the focus or break during a focus or break session.
4. Display durations as mm:ss. i.e. 05:00 for 5 minutes or 18:45 for eighteen minutes and forty-five seconds.
5. The tests use the data-testid="..." attributes on elements. Removing these will break one or more tests.

**Using setInterval in React**

Using setInterval with React functional components requires a custom hook.

We have provided a custom useInterval hook in src/utils/useInterval/index.js for you to use. The useInterval hook is already setup to start and stop with the play/pause buttons

You may not have learned about hooks yet, but don't worry, this function works exactly like setInterval except you don't need to use clearInterval to stop it.

As it is currently configured, the useInterval will execute the code in the callback every second, unless isTimerRunning is set to false.

This should be sufficient to implement the pomodoro timer.

**Playing Audio alarm**

Use the following code to play an alarm when the time expires. You can upload your own sound or use the one provided in the link below.

new Audio(`https://bigsoundbank.com/UPLOAD/mp3/1482.mp3`).play();

**classNames function**

import classNames from "../utils/class-names";

Use this function to dynamically assign the className property of react components.

Usage:

<span

className={classNames({

oi: true,

"oi-media-play": currentState.isPaused,

"oi-media-pause": !currentState.isPaused,

})}

/>

if currentState.isPaused === true, the className will be "oi oi-media-play" otherwise it will be "oi oi-media-pause"

classNames takes a map of a class name to a boolean value. If the boolean value is true, the class name is included, otherwise it is excluded.

returns: A space delimited string of the class names which have a value of true.

**minutesToDuration function**

**minutesToDuration** formats a number of minutes as 'mm:00'. For example,

import { minutesToDuration } from "../utils/duration";

minutesToDuration(3); *// '03:00'*

minutesToDuration(45); *// '45:00'*

**secondsToDuration function**

**secondsToDuration** formats a number of seconds as 'mm:ss'. For example,

import { secondsToDuration } from "../utils/duration";

secondsToDuration(305); *// '05:05'*

secondsToDuration(930); *// '15:30'*

**Debugging**

If you have a failing test, but the application appears to be working correctly when you view it in the browser, try the following debugging steps:

1. In [./src/pomodoro/Pomodoro.js](https://www.qualified.io/assess/src/pomodoro/Pomodoro.js), find isTimerRunning ? 1000 : null and temporarily change it to isTimerRunning ? 100 : null.
   * This will make the timer run 10 times faster, making it easier to debug.
2. Start the app and open it in the browser.
3. Set the focus and break times to the minimum values.
4. Click the play button to start the pomodoro timer.
5. Observe the application going through *multiple* focus/break sessions. Let it run through at least two transitions from "Focusing" to "On break" and back to "Focusing", just like a real user.
   * Check the values displayed in session title, session sub-title, and progress bar.
   * At this point you will likely see the problem.
6. In [./src/pomodoro/Pomodoro.js](https://www.qualified.io/assess/src/pomodoro/Pomodoro.js), change isTimerRunning ? 100 : null back to isTimerRunning ? 1000 : null so the timer runs at normal speed.

**Note:** In addition to needing to pass the tests and requirements in the instructions here, please review the Rubric Requirements for the human-graded part of this project in your Thinkful curriculum page.