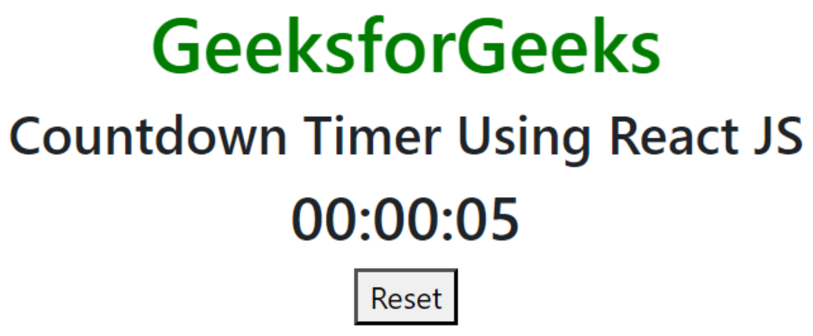
How to Create Countdown Timer in React JS

Last Updated : 04 Mar, 2024

React timers are very common **UI components** that are widely used in various applications and websites to visually display the remaining time for a specific activity or event. React timers are mostly used to highlight the **commencement** or **conclusion of events** or **offers** on commercial websites.

This tutorial explores the process of **creating a countdown timer** using **React JS**, a popular JavaScript user interface toolkit.

Preview of final output:



Предварительный просмотр выходных данных

Предварительные требования

- [NPM & Node.js]
- [React JS]
- [Хуки React JS]
- [Метод setInterval в JavaScript]

Здесь у нас есть 2 подхода к созданию таймера обратного отсчета, приведенные ниже

1. Использование хуков React с функциональными компонентами
2. Используя React State с компонентами на основе классов

Шаги по созданию таймера обратного отсчета с помощью React JS

Прежде чем создавать таймер в **React JS**, вы должны быть знакомы со следующими терминологиями:

- **Сохранение времени:** Вычислите разницу между целевым таймером и текущим временем, которое у нас есть. Эта функция проверит время, оставшееся от целевого таймера, путем вычислений и вернет общее количество часов, минут и секунд.
- **startTimer:** Эта функция начнет отсчет времени после получения общего количества часов, минут и секунд из функции getTimeRemaining.
- **ClearTimer**: Эта функция используется для сброса таймера, что означает, что при перезапуске таймера очищается время, оставшееся от предыдущего обратного отсчета, в противном случае он начинает параллельный двойной отсчет или они могут сворачиваться друг с другом.
- **getDeadTimer:** This function provides the deadline of the timer means it gives time from where you want to start the countdown. In this, you have to add time if you want to extend. We have used this in two scenarios first when the page is loaded and second when someone clicks the reset button.

Шаги по созданию приложения React

Сначала давайте создадим образец веб-приложения React, на которое мы установим таймер React.

Шаг 1: Создайте приложение React с помощью следующей команды:

```
npx create-react-app foldername
```

Шаг 2: После создания папки вашего проекта, т.е. имени папки

```
cd foldername
```

Подход 1: создайте таймер обратного отсчета с помощью React Hooks

Мы можем создать таймер обратного отсчета, используя **перехватчики React** и **метод setInterval** JavaScript.

Мы используем хуки React, такие как **useState, useRef** и **useEffect**

Мы предоставили рабочий код, чтобы должным образом продемонстрировать, как создать таймер обратного отсчета, используя перехватчики React с функциональными компонентами.

Таймер обратного отсчета React с использованием React Hooks Пример

В этом примере реализован таймер обратного отсчета в React с использованием перехватчиков React и метода setInterval() JavaScript.

Javascript

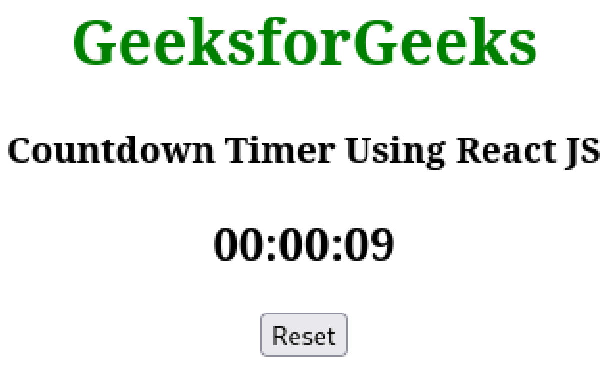
```
// Filename - App.js

import React, { useState, useRef, useEffect } from "react";

const App = () => {
    // We need ref in this, because we are dealing
    // with JS setInterval to keep track of it and
    // stop it when needed
    const Ref = useRef(null);

    // The state for our timer
    const [timer, setTimer] = useState("00:00:00");

    const getTimeRemaining = (e) => {
        const total =
            Date.parse(e) - Date.parse(new Date());
        const seconds = Math.floor((total / 1000) % 60);
        const minutes = Math.floor(
            (total / 1000 / 60) % 60
        );
        const hours = Math.floor(
            (total / 1000 / 60 / 60) % 24
            minutes,
            seconds,
        };
    };

    const startTimer = (e) => {
        let { total, hours, minutes, seconds } =
            getTimeRemaining(e);
        if (total >= 0) {
            // update the timer
            // check if less than 10 then we need to
            // add '0' at the beginning of the variable
            setTimer(
                (hours > 9 ? hours : "0" + hours) +
                ":" +
                (minutes > 9
                    ? minutes
                    : "0" + minutes) +
                ":" +
                (seconds > 9 ? seconds : "0" + seconds)
            );
        }
    };

    const clearTimer = (e) => {
        // If you adjust it you should also need to
        // adjust the Endtime formula we are about
        // to code next
        setTimer("00:00:10");

        // If you try to remove this line the
        // updating of timer Variable will be
        // after 1000ms or 1sec
        if (Ref.current) clearInterval(Ref.current);
        const id = setInterval(() => {
            startTimer(e);
        }, 1000);
        Ref.current = id;
    };

    const getDeadTime = () => {
        let deadline = new Date();

        // This is where you need to adjust if
        // you entend to add more time
        deadline.setSeconds(deadline.getSeconds() + 10);
        return deadline;
    };

    // We can use useEffect so that when the component
```

```
        useEffect(() => {
            clearTimer(getDeadTime());
        }, []);

        // Another way to call the clearTimer() to start
        // the countdown is via action event from the
        // button first we create function to be called
        // by the button
        const onClickReset = () => {
            clearTimer(getDeadTime());
        };

        return (
            <div
                style={{ textAlign: "center", margin: "auto" }}>
                <h1 style={{ color: "green" }}>
                    GeeksforGeeks
                </h1>
                <h3>Countdown Timer Using React JS</h3>
                <h2>{timer}</h2>
                <button onClick={onClickReset}>Reset</button>
            </div>
        );
    };

    export default App;
```

Steps To Run Application: Run the application using the following command from the root directory of the project:

```
npm start
```

Output: Now open your browser and go to **http://localhost:3000**



- **useEffect Hook** starts the timer by calling **clearTimer**.
- **getTimeRemaining Function** calculates the remaining time from a future deadline
- **startTimer Function** updates the timer display every second based on the remaining time.
- **clearTimer Function** resets the timer to 10 seconds and starts the countdown.
- **getDeadTime Function** calculates the deadline time for the countdown.
- **onClickReset Function** resets the timer to 10 seconds when the "**Reset**" button is clicked.
- **Return JSX** displays the timer value and a "Reset" button in the UI.

Approach 2: Create Countdown Timer Using React State in Class Components

We can create a countdown timer using React states in class components and the **setInterval method** of JavaScript.

We will be using the React setState method and createRef methods to updated the timer in the state variable.

We have provided the working code to properly demonstrate how to create a countdown timer using the React States with Class Components.

Countdown timer using React States Example:

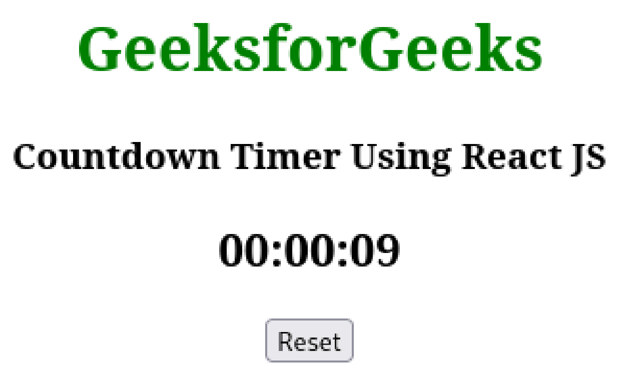
Javascript

```
// Filename - App.js

import React, { Component } from "react";

class App extends Component {
    constructor(props) {
        super(props);

        // We need ref in this, because we are dealing
        // with JS setInterval to keep track of it and
        // stop it when needed
        this.Ref = React.createRef();

        // The state for our timer
```

```
getTimeRemaining = (e) => {
    const total = Date.parse(e) - Date.parse(new Date());
    const seconds = Math.floor((total / 1000) % 60);
    const minutes = Math.floor((total / 1000 / 60) % 60);
    const hours = Math.floor((total / 1000 / 60 / 60) % 24);
    return {
        total,
        hours,
        minutes,
        seconds,
    };
};

startTimer = (e) => {
    let { total, hours, minutes, seconds } = this.getTimeRemaining(e
    if (total >= 0) {
        // update the timer
        // check if less than 10 then we need to
        // add '0' at the beginning of the variable
        this.setState({
            timer:
                (hours > 9 ? hours : "0" + hours) +
                ":" +
                (minutes > 9 ? minutes : "0" + minutes) +
                ":" +
                (seconds > 9 ? seconds : "0" + seconds),
        });
    }
};

clearTimer = (e) => {
    // If you adjust it you should also need to
    // adjust the Endtime formula we are about
    // to code next
    this.setState({ timer: "00:00:10" });

    // If you try to remove this line the
    // updating of timer Variable will be
    // after 1000ms or 1sec
    if (this.Ref.current) clearInterval(this.Ref.current);
    const id = setInterval(() => {
        this.startTimer(e);
    }, 1000);
    this.Ref.current = id;
};

getDeadTime = () => {
    let deadline = new Date();
```

```
        };

        // We can use componentDidMount so that when the component
        // mount the timer will start as soon as possible
        componentDidMount() {
            this.clearTimer(this.getDeadTime());
        }

        // Another way to call the clearTimer() to start
        // the countdown is via action event from the
        // button first we create function to be called
        // by the button
        onClickReset = () => {
            this.clearTimer(this.getDeadTime());
        };

        render() {
            return (
                <div style={{ textAlign: "center", margin: "auto" }}>
                    <h1 style={{ color: "green" }}>GeeksforGeeks</h1>
                    <h3>Countdown Timer Using React JS</h3>
                    <h2>{this.state.timer}</h2>
                    <button onClick={this.onClickReset}>Reset</button>
                </div>
            );
        }
    }

    export default App;
```

Output: Now open your browser and go to **http://localhost:3000**



Explanation:

- In `componentDidMount` initiate the countdown by calling the `clearTimer` function.
- **getTimeRemaining Function** calculates the remaining time from a future deadline
- **startTimer Function u**pdates the timer display every second based on the remaining time.
- **clearTimer Function** resets the timer to 10 seconds and starts the countdown.
- **getDeadTime Function** calculates the deadline time for the countdown.
- **onClickReset Function** resets the timer to 10 seconds when the "**Reset**" button is clicked.
- **Return JSX** displays the timer value and a "Reset" button in the UI.

Wrapping Up

React timer is a very useful component in website UI, as it helps in making events more interactive. This tutorial teaches how to create a countdown timer using React JS for your projects. Explained with live working code, this guide provides an easy solution to build a React timer.

We have used React Hooks and setTimer() method to create a countdown timer in React JS. Hope this guide, helps you build your first timer using React JS.

H harsh... 7

Next Article

Similar Reads

How to Create Countdown Timer using React Native ?

Our main objective focuses on constructing a straightforward and user-friendly countdown timer that impeccably showcases the remaining time left in terms of years, days, hours,...

4 min read

Event Countdown Timer Using React

Event Countdown Timer is an application where you can create an event with the name, date, and category of the event and it will create a card that will show you the remaining...

5 min read

How To Create A Countdown Timer Using JavaScript

A countdown timer is an accurate timer that can be used for a website or blog to display the countdown to any special event, such as a birthday or anniversary. The basics of a...

5 min read

Create a Event Countdown App with React Native

In this article, we will create an event countdown time application using React Native. The Event Countdown App is a mobile application developed with React Native. It enables...

7 min read

Create Timer Based Phone Silencer App using React-Native

In this project, we will create a mobile application using React Native that acts as a timer-based phone silencer. The app allows users to set a timer, and when the timer finishes, it...

3 min read

View More Articles

Article Tags : React-Questions ReactJS Web Technologies