How to useContext in React?

OCTOBER 19, 2019 BY ROBIN WIERUCH - EDIT THIS POST

This tutorial is part 2 of 2 in this series.

Part 1: React Context

React's Function Components come with React Hooks these days. Not only can React Hooks be used for State in React but also for using React's Context in a more convenient way. This tutorial shows you how to use React's useContext Hook. Before, make sure to read the previous tutorial for getting an introduction to React's Context:

- Why React Context?
- What is React Context?
- How to use React Context?
- When to use React Context?

- When to use React Context?

If you can answer these questions by reading the previous tutorial, return to this tutorial to learn about React's useContext Hook. Let's get started ...

REACT'S USECONTEXT HOOK

React's Context is initialized with React's `createContext` top-level API. In this case, we are using React's Context for sharing a theme (e.g. color, paddings, margins, font-sizes) across our React components. For the sake of keeping it simple, the theme will only be a color (e.g. `green`) here:

```
// src/ThemeContext.js

import React from 'react';

const ThemeContext = React.createContext(null);

export default ThemeContext;
```

The Context's Provider component can be used to provide the theme to all React child components below this React top-level component which uses the Provider:

```
// src/ComponentA.js

import React from 'react';
import ThemeContext from './ThemeContext';

const A = () => (
  <ThemeContext.Provider value="green">
    <D />
  </ThemeContext.Provider>
);
```

Somewhere below component A, not component D in this case, but any other child component, let's say component C, will use this theme to style itself:

```
// src/ComponentC.js

import React from 'react';
import ThemeContext from './ThemeContext';
```

```
const C = () => (
  <ThemeContext.Consumer>
    {color => (
      <p style={{ color }}>
        Hello World
      </p>
    )}
  </ThemeContext.Consumer>
);
```

That's the most basic approach of using React's Context API with a single top-level Provider component and one Consumer component in a React child component sitting somewhere below. There can be many more child components using the Consumer component though, but they have to be located somewhere below the component using the Provider component.

Now comes the crucial part where we shift towards React's useContext Hook. As you can see, the Consumer component coming from React's Context is by default a render prop component. In a world where we can use React Hooks, a render prop component isn't always the best choice. Let's see the previous example with React's useContext Hook instead:

```
// src/ComponentC.js

import React from 'react';
import ThemeContext from './ThemeContext';

const C = () => {
  const color = React.useContext(ThemeContext);

  return (
    <p style={{ color }}>
      Hello World
    </p>
  );
};
```

React's useContext just uses the `Context` object -- which you have created before -- to retrieve the most recent value from it. Using the React Hooks instead of the Consumer component, makes the code more readable, less verbose, and doesn't introduce a kinda artificial component -- the Consumer component -- in between.

Show Comments

KEEP READING ABOUT REACT›

REACT'S USEREDUCER HOOK VS REDUX

Since React Hooks have been released, function components can use state and side-effects. There are two hooks that are used for modern state management in React (useState and useReducer) and one…

REACT HOOKS TUTORIAL

React Hooks were introduced at React Conf October 2018 as a way to use state and side-effects (see lifecycle methods in class components) in React function components. Whereas function components…

50.000+ readers.

GET THE BOOK >

Get it on Amazon.

TAKE PART

NEVER MISS AN ARTICLE ABOUT WEB DEVELOPMENT AND JAVASCRIPT.

- ✔ Join 50.000+ Developers
- ✔ Learn Web Development with JavaScript
- ✔ Tips and Tricks
- ✔ Access Tutorials, eBooks and Courses
- ✔ Personal Development as a Software Engineer

Your email address SUBSCRIBE

View our Privacy Policy.

PORTFOLIO **ABOUT**

Online Courses
Open Source
Tutorials
About me
What I use
How to work with me
How to support me

Contact Me Privacy & Terms