How To Pass Props Between React Components To Control Modals

linkedin.com/pulse/how-pass-props-between-react-components-control-modals-melissa-wong

Photo by Tim Mossholder on Unsplash

Melissa Wong

Senior Full Stack Engineer

Fecha de publicación: 20 de ene de 2021

Seguir

A while ago I was asked to explain how to make modals work in React Native.

Let’s suppose you have an awesome to-do app. And you want to nudge the users to upgrade to premium tier after posting five to-dos.

You are probably thinking about creating a modal that (1) opens when the state of number of posts is five and (2) closes when user presses “close”. But how does the modal or child component know about the parent’s state? And how does the parent know when the user presses “close” in the child component so we can close the modal?

In this article we will look at a quick example on how to open and close modals in React Native. Or rather, how do you pass the state and callback as props between the parent and child component to control the modal behaviors.

Setting things up

For starter, we have modalVisible, a state variable to determine when the modal is shown. And we will pass it as a prop to the CustomModal component.

```
import React, {useState} from 'react';
import CustomModal from './CustomModal';


const Home = (props) => {
    ...

        const [modalVisible, setModalVisible] = useState(false);


        const closeModal = () => {
            //close modal when user presses "close"
        };


        return(
            <CustomModal modalVisible={modalVisible} closeModal=
        {closeModal} />
        );
};
```

What about setModalVisible? This function will help us update the modalVisible state to close the modal when the user presses “closes”. We will use it in the closeModal function below.

```
const closeModal = () => {
    setModalVisible(false);
};
```

Note that you could also pass the setModalVisible setter function without defining a separate function. You would write <CustomModal functions={[modalVisible, setModalVisible]} /> in the parent component and const [modalVisible, setModalVisible] = props.functions in the CustomModal component instead.

Show Modal Conditionally With useEffect Hook

Now that we've got the state and state setter method setup, let's look at how to make the upgrade reminder modal show up when our user makes the fifth post.

For the sake of simplicity, we'll pretend we already defined some list called toDos to keep track of the user's posts.

Inside our useEffect hook, we can now invoke setModalVisible to set modalVisible to true if the length of toDos equal five.

And we have to add toDos as a second argument or dependency to the hook. This ensures useEffect will fire only when toDos changes.

```
import React, {useState, useEffect} from 'react';
import CustomModal from './CustomModal';


const Home = (props) => {
        ...

        const [modalVisible, setModalVisible] = useState(false);


    const closeModal = () => {
                setModalVisible(false);
    };

    useEffect(() => {
        if(toDos.length === 5) setModalVisible(true);
    }, [toDos]);


    return(
        <CustomModal modalVisible={modalVisible} closeModal=
        {closeModal} />
    );
};
```

Child Component Uses Prop From Parent To Set Modal Visible

Now we are ready to create the CustomModal component. We'll keep it simple and make it render some text only for now.

Notice we set modal's visible prop as props.modalVisible. This is how we use the modalVisible state from the parent to influence the modal's behavior within the child.

```
import React from 'react';
import { Text, Modal } from 'react-native';


const CustomModal = (props) => {
    return(
        <Modal visible={props.modalVisible} transparent>
            <Text>Enjoy this awesome to-do app? Upgrade to premium!</Text>
        </Modal>
    );
};




export default CustomModal;
```

Finally, we add the two Pressables to complete our modal (this core component is recently released by React in version 0.63. You can read more about it here).

We surely hope the user selects the "Upgrade!" Pressable. But maybe our user is not quite ready just yet.

So how do we close the modal if they press on the "Not now…" Pressable?

Trigger The Callback Function To Close The Modal

We need to pass the closeModal callback function to the Pressable's onPress prop as shown below.

So you may have noticed the parentheses in props.closeModal(). Why don't we see those parentheses in the parent component when we passed the function to child? That is because we want to invoke closeModal when the user presses "Not now…"

```
import React from 'react';
import { Text, Pressable, Modal } from 'react-native';


const CustomModal = (props) => {
    const onUpgrade = () => {
        //route to Upgrade component
    };


    return(
        <Modal visible={props.modalVisible} transparent>
            <Text>Enjoy this awesome to-do app? Upgrade to premium!</Text>

            <Pressable onPress={() => props.closeModal()}>
                <Text>Not now...</Text>
            </Pressable>


            <Pressable onPress={() => onUpgrade()}>
                    <Text>Upgrade!</Text>
            </Pressable>
        </Modal>
    );
};



export default CustomModal;
```

I hope this example demonstrates how you can pass state and function as props between the parent and child components.

Controlling the behavior of a modal is a pretty common use case. But you definitely see this pattern in many more ways as you continue to dive in the sea of code.