ALTERCLASS.IO

REACT HOOKS CHEATSHEET

v16.12.0


The Easy Way To Learn React
IMPROVE YOUR JAVASCRIPT SKILLS WITH NO PRIOR EXPERIENCE IN REACT
Start Learning now
AlterClass.io

"Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class."

reactjs.org

Rules of Hooks

1. Only call hooks at the top level

- Hooks cannot be used in regular JavaScript functions.

- Hooks can only be called from React function components or custom Hooks.

2. Only call React Hooks from React functions

- Hooks cannot be used within loops, conditionals, or nested functions.

State Hook - *useState* (1/3)

```
import React, { useState } from 'react';
```

- *useState* is a Hook that lets you add React state to function components.

- *useState* returns the current state and a function to update it.

- *useState* take as argument the initial state value.

```
const MyComponent = (props) => {
   const [showModal, setShowModal] = useState(false);

   return (
      <div>
         <button onClick={() => setShowModal(true)}>Show</button>
         { show && <Modal /> }
      </div>
   );
};
```

State Hook - *useState* (2/3)

- *useState* can be used for multiple state variables.

```
const MyComponent = (props) => {
   const [name, setName] = useState("Greg");
   const [country, setCountry] = useState("France");

   return <div>Hello! I'm {name}, from {country}.</div>;
};
```

- *useState* can hold primitives, objects or arrays.

```
const [people, setPeople] = useState({
   name: "Greg",
   country: "France"
});
```

State Hook - *useState* (3/3)

- *useState* always replace the state variable when updating it instead of merging it.

- The update state function can accept a function. This function receives the previous state value, and returns an updated value.

```
const [people, setPeople] = useState({
   name: "Greg",
   country: "France",
   age: 28
});

...

setPeople(prevState => {
   return { ...prevState, age: prevState.age + 1 };
});
```

Effect Hook - *useEffect* (1/3)

```
import React, { useEffect } from 'react';
```

- *useEffect* is a Hook that lets you perform "side effects" in function components, such as data fetching, manual DOM manipulation, and so on...

- *useEffect* accepts a function as argument.

- *useEffect* runs after every render.

```
const MyComponent = ({ userId }) => {
   const [user, setUser] = useState({});

   useEffect(() => {
      fetch(`http://api.example.com/v1/users/${userId}`)
         .then(res => res.json())
         .then(data => setUser(data));
   });
   ...
```

Effect Hook - *useEffect* (2/3)

```
    ...
    return (
        <ul>
            { friends.map(friend => <li key={friend.id}>{friend.name}</li>) }
        </ul>
    );
};
```

- *useEffect* accepts a second argument: the dependencies array. It tells React to run the effect function only if one of the dependencies as changed.

- You can pass an empty array ([]) to run it only once.

```
useEffect(() => {
    fetch(`http://api.example.com/v1/users/${userId}`)
        .then(res => res.json())
        .then(data => setUser(data));
}, [userId]);
```

Effect Hook - *useEffect* (3/3)

- useEffect lets you clean up any effect you have used by returning clean up function.

```
useEffect(() => {
   window.addEventListener("mousedown", eventhandler);

   return () => {
      window.removeEventListener("mousedown", eventhandler);
   };
}, []);
```

Ref Hook - *useRef*

```
import React, { useRef } from 'react';
```

- *useRef* lets you access DOM element.

```
const MyComponent = (props) => {
   const inputRef = useRef(null);

   useEffect(() => {
      inputRef.current.focus();
   });

   return (
      <form>
         <input ref={inputRef} type="text" />
      </form>
   );
};
```


The Easy Way To Learn React
IMPROVE YOUR JAVASCRIPT SKILLS WITH NO PRIOR EXPERIENCE IN REACT
Start Learning now
AlterClass.io

Callback Hook
useCallback (1/2)

```
import React, { useCallback } from 'react';
```

- *useCallback* returns a memoized version of a callback.

- *useCallback* accepts as arguments a callback and a dependencies array.

- The callback only changes if one of the dependencies has changed.

Callback Hook
useCallback (2/2)

```
const MyComponent = ({ eventhandler }) => {
   ...
   const handleEventHandler = useCallback(
      event => {
         if (typeof eventHandler === "function") {
            eventHandler(event);
         }
      },
      [eventHandler]
   );
   ...
```

Context Hook - *useContext* (1/3)

```
import React, { useContext } from 'react';
```

- *useContext* lets you read the context and subscribe to its changes from a function component.

- The context still needs to be created by *React.createContext* and provided by a context *Provider* component.

- *useContext* accepts as argument the context itself created by React.createContext.

- It returns the current context value for that context.

Context Hook - *useContext* (2/3)

```
const UserContext = React.createContext(null);

function App() {
   const [userName, setUserName] = useState("Greg");

   return (
      <UserContext.Provider value={{ name: userName }}>
         <Main />
      </UserContext.Provider>
   );
}

const Main = (props) => (
   <>
      <Header />
      <UserInfo />
      <Footer />
   </>
);
```

Context Hook - *useContext* (3/3)

```
const UserInfo = (props) => {
   const user = useContext(UserContext);

   return <div>Hello, {user.name}!</div>;
};
```

Memoization Hook
useMemo

```
import React, { useMemo } from 'react';
```

- *useMemo* helps with performance optimization by returning a memoized value of an expensive computation.

- *useMemo* accepts as arguments a function and a dependencies array.

- *useMemo* will only recompute the memoized value if one of the dependencies has changed.

```
const MyComponent = ({ a, b }) => {
   const memoizedValue = useMemo(() => expensiveComputation(a, b),
   [a, b]);
   ...
};
```

Reducer Hook
useReducer (1/2)

```
import React, { useReducer } from 'react';
```

- *useReducer* lets you use reducers to manage your application state. This is an alternative to the state hook, *useState.*

- *useReducer* accepts as argument a reducer of type *(state, action) => newState*. It returns the current state and a dispatch method.

```
const initialState = { isAuth: false, user: null };

function reducer(state, action) {
   switch (action.type) {
      case "login": return { isAuth: true, user: action.user };
      case "logout": return { isAuth: false, user: null };
      default: return state;
   }
}
...
```

Reducer Hook
useReducer (2/2)

```
function App() {
   const [state, dispatch] = useReducer(reducer, initialState);

   const handleLogin = () => dispatch({ type: 'login', user: { ... } });

   const handleLogout = () => dispatch({ type: 'logout' });

   if (state.isAuth) {
      return (
         <>
            <p>Welcome, {state.user.name}!</p>
            <button onClick={handleLogout}>Logout</button>
         </>
      );
   }

   return <button onClick={handleLogin}>Login</button>;
}
```

Custom Hook

- You can create your own React hooks to extract component logic into reusable functions.

```
import { useEffect, useCallback } from "react";

// Custom React Hook
export default function useEventListener(name, handler, element) {

   const handleEventHandler = useCallback(
      event => {
         if (typeof handler === "function") {
            handler(event);
         }
      },
      [handler]
   );
   ...
```

Custom Hook

```
    ...
    useEffect(
        () => {
            // Check if the element supports the addEventListener method
            const checked = element && element.addEventListener;
            // Stop here if not supported
            if (!checked) return;
            // Add event listener
            element.addEventListener(eventName, handleEventHandler);
            // Remove event listener on cleanup
            return () => {
                element.removeEventListener(name, handleEventHandler);
            };
        },
        [name, element, handleEventHandler]
    );
}
```


The Easy Way To Learn React
IMPROVE YOUR JAVASCRIPT SKILLS WITH NO PRIOR EXPERIENCE IN REACT
Start Learning now
AlterClass.io