Intermediate React

tsevdos.me / @tsevdos

Agenda

All the content can be found here.

- create react app
- hooks
- forms and events
- custom hooks

Rules

Feel free to interrupt me for:

- questions
- relevant comments

Create react app

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

Create react app

```
npx create-react-app my-app
cd my-app
```

Create react app

Files and directory structure (demo).

Hooks

Let you put state and logic to your functional components!

Hook rules

- only call hooks from React function components
- only call hooks at the top level
- don’t call hooks inside loops, conditions, or nested functions
- custom hooks start with useSomething PascalCase function

React Hooks

- useState
- useEffect
- useReducer
- useContext
- useCallback
- useMemo
- useRef

React Hooks

- useState: is used to declare a state variable and can be initialized with any type of value. It returns an array with the (current) state and a function used to update the state.
- useReducer: An alternative to useState. Accepts a reducer of type (state, action) => newState, and returns the current state paired with a dispatch method.

React Hooks

- useEffect: accepts an effect "action" as an anonymous function as the first argument. Skip applying an effect if certain values haven’t changed between re-renders. To do so, pass an array as an optional second argument to useEffect. Finally, some effects might require cleanup so they return a function.
- useCallback: useCallback will return a memoized version of the callback that only changes if one of the inputs has changed.

React Hooks

- useMemo: Returns a memoized value. Pass a “create" function and an array of inputs. useMemo will only recompute the memoized value when one of the inputs has changed.
- useRef: useRef returns a mutable ref object whose ".current" property is initialized to the passed argument (initialValue). The returned object will persist for the full lifetime of the component.

How do lifecycle methods correspond to Hooks?

- constructor: function components don’t need a constructor. You can initialize the state in the useState call
- getDerivedStateFromProps: schedule an update while rendering instead
- shouldComponentUpdate: React.memo
- render: this is the function component body itself

How do lifecycle methods correspond to Hooks?

- componentDidMount, componentDidUpdate, componentWillUnmount: useEffect Hook
- componentDidCatch and getDerivedStateFromError: there are no Hook equivalents for these methods yet, but they will be added soon

useState

```
const [state, steState] = useState("my value");
```

useEffect

```
// accepts two arguments: a function, and dependency array
useEffect(() => {
  // do stuff
  return () => {}; // run this return function when component unmounts
}, []); // dependency array
```

useReducer

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

Hooks

Examples.

Forms and Events

- inputs events
- form events

Components and events

- SyntheticEvent
- cross-browser wrapper around the browser’s native event
- it has the same interface as the browser’s native event, including stopPropagation() and preventDefault()
- you have access to the native event using event.nativeEvent

Components and events

- react events are named using camelCase, rather than lowercase
- supported events

Forms and Events

Examples.

Custom hooks

Custom hooks are a mechanism to reuse stateful logic (such as setting up a subscription and remembering the current value), but every time you use a custom Hook, all state and effects inside of it are fully isolated.

Custom hooks

- reuse stateful logic between components
- simplify components (easy to understand)
- share logic between different components and lifecycle methods
- easier and more flexible pattern from render props and higher-order components

Custom hooks rules

- only call hooks from React function components
- only call hooks at the top level
- don’t call hooks inside loops, conditions, or nested functions
- custom hooks start with useSomething PascalCase function

Custom hooks

Examples.

Custom hooks

(./src/examples/custom-hooks/exercise/UserCard.js)

1. create a custom "useUser" custom hook
2. keep the same functionality
3. use the custom hook to get the user data on the UserCard component

Mini project: ToDo list

Starting point (./intermediate/workshop/todo-app).

Mini project: exercise 1

Create the add todo functionality.

Mini project: exercise 2

Create the toggle todo functionality.

Mini project: exercise 3

Create the delete todo functionality.

Mini project: exercise 4

Replace the hard-coded todos with server data, using this endpoint.

Recap

- create react app
- hooks and custom hooks
- forms and events

Recap: hooks

- state (useState / useReducer)
- useEffect

Recap: forms and events

- (SyntheticEvent) event object
- input handling
- form handling

That's all folks

Questions / Discussions?