Using a indeterminate React Checkbox

MAY 16, 2021 BY ROBIN WIERUCH - EDIT THIS POST

This tutorial is part 2 of 2 in this series.

Part 1: How to create a React Checkbox

A short React tutorial by example for beginners on how to create an **indeterminate React Checkbox** which uses an indeterminate state (also called **tri state**).

Let's start with a checkbox example from our previous tutorial:

```
const App = () => {
  const [checked, setChecked] = React.useState(false);

  const handleChange = () => {
```

```
    setChecked(!checked);
  };

  return (
    <div>
      <Checkbox
        label="Value"
        value={checked}
        onChange={handleChange}
      />

      <p>Is checked? {checked.toString()}</p>
    </div>
  );
};

const Checkbox = ({ label, value, onChange }) => {
  return (
    <label>
      <input type="checkbox" checked={value} onChange={onChange} />
      {label}
    </label>
  );
};
```

Now we want to extend the functionality of this checkbox for handling a tri state instead of a bi state. First, we need to transform our state from a boolean to an enum, because only this way we can create a tri state:

```
const CHECKBOX_STATES = {
  Checked: 'Checked',
  Indeterminate: 'Indeterminate',
  Empty: 'Empty',
};

const App = () => {
  const [checked, setChecked] = React.useState(CHECKBOX_STATES.Empty);

  const handleChange = () => {
    let updatedChecked;

    if (checked === CHECKBOX_STATES.Checked) {
      updatedChecked = CHECKBOX_STATES.Empty;
    } else if (checked === CHECKBOX_STATES.Empty) {
      updatedChecked = CHECKBOX_STATES.Checked;
    }

    setChecked(updatedChecked);
  };

  return (
    <div>
```

```
      <Checkbox
        label="Value"
        value={checked}
        onChange={handleChange}
      />

      <p>Is checked? {checked}</p>
    </div>
  );
};

const Checkbox = ({ label, value, onChange }) => {
  return (
    <label>
      <input
        type="checkbox"
        checked={value === CHECKBOX_STATES.Checked}
        onChange={onChange}
      />
      {label}
    </label>
  );
};
```

We have the same behavior as before, but enabled us to have more than two states for our checkbox.

Next comes the indeterminate state of a checkbox. Unfortunately it cannot be assigned via HTML and we need to use an imperative DOM manipulation here. Fortunately React has the concept of refs which gives React developers access to DOM elements:

```
const Checkbox = ({ label, value, onChange }) => {
  const checkboxRef = React.useRef();

  return (
    <label>
      <input
        ref={checkboxRef}
        type="checkbox"
        checked={value === CHECKBOX_STATES.Checked}
        onChange={onChange}
      />
      {label}
    </label>
  );
};
```

By having access to the checkbox element, we can set and unset the checked state imperatively instead of using the HTML in a declarative way:

```
const Checkbox = ({ label, value, onChange }) => {
  const checkboxRef = React.useRef();

  React.useEffect(() => {
    if (value === CHECKBOX_STATES.Checked) {
      checkboxRef.current.checked = true;
    } else {
      checkboxRef.current.checked = false;
    }
  }, [value]);

  return (
    <label>
      <input ref={checkboxRef} type="checkbox" onChange={onChange} />
      {label}
    </label>
  );
};
```

React's useEffect Hook executes its passed side-effect function every time a variable in the dependency array (here: `value`) changes. Then in the side-effect function we evaluate the value: if it is checked, we set the checkbox's internal HTML state programmatically to checked; and vice versa for the unchecked state.

Finally, we can assign the indeterminate state this way too:

```
const Checkbox = ({ label, value, onChange }) => {
  const checkboxRef = React.useRef();

  React.useEffect(() => {
    if (value === CHECKBOX_STATES.Checked) {
      checkboxRef.current.checked = true;
      checkboxRef.current.indeterminate = false;
    } else if (value === CHECKBOX_STATES.Empty) {
      checkboxRef.current.checked = false;
      checkboxRef.current.indeterminate = false;
    } else if (value === CHECKBOX_STATES.Indeterminate) {
      checkboxRef.current.checked = false;
      checkboxRef.current.indeterminate = true;
    }
  }, [value]);

  return (
    <label>
      <input ref={checkboxRef} type="checkbox" onChange={onChange} />
      {label}
    </label>
  );
};
```

And don't forget to assign the proper value on state change in the first place:

```
const App = () => {
  const [checked, setChecked] = React.useState(CHECKBOX_STATES.Empty);

  const handleChange = () => {
    let updatedChecked;

    if (checked === CHECKBOX_STATES.Checked) {
      updatedChecked = CHECKBOX_STATES.Empty;
    } else if (checked === CHECKBOX_STATES.Empty) {
      updatedChecked = CHECKBOX_STATES.Indeterminate;
    } else if (checked === CHECKBOX_STATES.Indeterminate) {
      updatedChecked = CHECKBOX_STATES.Checked;
    }

    setChecked(updatedChecked);
  };

  return (
    <div>
      <Checkbox
        label="Value"
        value={checked}
        onChange={handleChange}
      />

      <p>Is checked? {checked}</p>
    </div>
  );
};
```

That's it. We transformed our React checkbox component from a bi state to a tri state by introducing the indeterminate state. I hope this tutorial is useful to you if you happen to need a checkbox with three states.

MobX is a state management solution. It is a standalone pure technical solution without being opinionated about the architectural state management app design. The 4 pillars State, Actions, Reactions...

HOW TO CREATE A REACT CHECKBOX

A short React tutorial by example for beginners about using a checkbox in React. First of all, a checkbox is just an HTML input field with the type of checkbox which can be rendered in React's JSX...

TAKE PART

NEVER MISS AN ARTICLE ABOUT WEB DEVELOPMENT AND JAVASCRIPT.

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

SUBSCRIBE >

View our Privacy Policy.

PORTFOLIO

Online Courses

Open Source

Tutorials

ABOUT

About me

What I use

How to work with me

How to support me