Lecture 3 - Hooks (useState, useEffect)

What is hook?

Hooks are special functions that let you 'hook into' React state and lifecycle features. Hooks allow functional components to manage state and side effects. Commonly used hooks: useState, useEffect, useContext, etc.

What is useState?

useState is a React Hook that lets you add a state variable to your component.
It returns a pair: current state value and a function to update it.

```
const [state, setState] = useState(initialState)
```

useState: declaration

Call useState at the top level of your component to declare a state variable.

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

function MyComponent() {
  const [age, setAge] = useState(28);
  const [name, setName] = useState('Taylor');
  const [todos, setTodos] = useState(() => createTodos());
  // ...
```

useState: initial value

initialState: The value you want the state to be initially. It can be a value of any type, but there is a special behavior for functions. This argument is ignored after the initial render.

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

function MyComponent() {
  const [age, setAge] = useState(28);
  const [name, setName] = useState('Taylor');
  const [todos, setTodos] = useState(() => createTodos());
  // ...
```

useState: set functions

The set function returned by useState lets you update the state to a different value and trigger a re-render. You can pass the next state directly, or a function that calculates it from the previous state:

```
const [name, setName] = useState('Edward');

function handleClick() {
  setName('Taylor');
  setAge(a => a + 1);
  // ...
```

⚠ Pitfall

Calling the `set` function **does not** change the current state in the already executing code:

```
function handleClick() {
  setName('Robin');
  console.log(name); // Still "Taylor"!
}
```

It only affects what `useState` will return starting from the *next* render.

What is useEffect?

useEffect is a React Hook that lets you run side effects in functional components.

```
useEffect(setup, dependencies?)
```

What is side effects?

Side effects are actions that affect something outside the component, like:

- Fetching data from an API
- Subscribing to events or WebSockets
- Updating the document title
- Starting or cleaning up timers

useEffect: declaration

Call useEffect at the top level of your component to declare an Effect.

```
import { useState, useEffect } from 'react';
import { createConnection } from './chat.js';

function ChatRoom({ roomId }) {
  const [serverUrl, setServerUrl] = useState('https://localhost:1234');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
  // ...
}
```

useEffect: Parameters

1. setup:
   The function with your effect's logic. When the component mounts, React runs the setup function.
- On dependency changes: React runs the cleanup (if provided) with old values, then the setup again.
- On unmount: React runs the cleanup one last time.

```
import { useState, useEffect } from 'react';
import { createConnection } from './chat.js';

function ChatRoom({ roomId }) {
  const [serverUrl, setServerUrl] = useState('https://localhost:1234');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
  // ...
}
```

useEffect: Parameters

1. dependencies:
   A list of reactive values (props, state, or variables) used in the effect.

- Written inline like [dep1, dep2].
- React re-runs the effect only if a dependency changes.
- Empty array [] → run once on mount.
- No array → run after every render.

```
import { useState, useEffect } from 'react';
import { createConnection } from './chat.js';

function ChatRoom({ roomId }) {
  const [serverUrl, setServerUrl] = useState('https://localhost:1234');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
  // ...
}
```

useEffect: returns

useState returns undefined.