ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

React JS
State Management Μέρος 2ο

Δαμιανός Μαρινάτος

Group Related State

Ομαδοποίηση σχετικών δεδομένων στο state

- Όταν θέλουμε να **ενημερώνουμε δύο ή περισσότερες μεταβλητές state ταυτόχρονα**, είναι προτιμότερο να τις συγχωνεύσουμε σε μία κοινή μεταβλητή state.

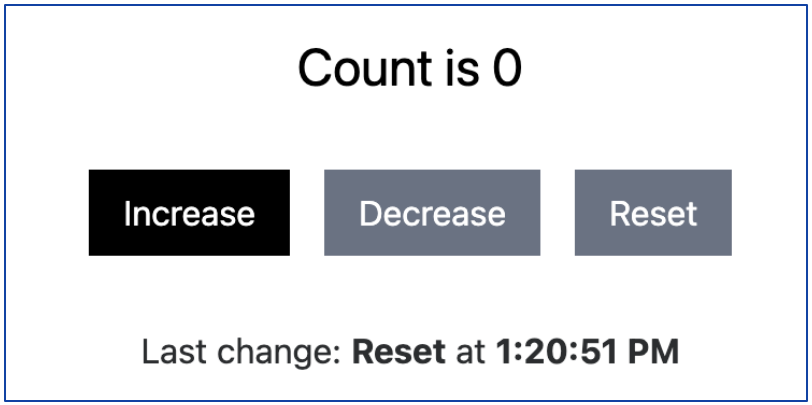
```
const [count, setCount] = useState( initialState: 0);
const [lastAction, setLastAction] = useState( initialState: "");
const [time, setTime] = useState( initialState: "");
```

```
const [state, setState] = useState<CounterState>({
  count: 0,
  lastAction: "",
  time: "",
});
```

Παράδειγμα Counter (1)

Θα δημιουργήσουμε ένα component που:

- χρησιμοποιεί το hook useState για να διαχειριστεί την κατάσταση του μετρητή.
- το state θα περιλαμβάνει την τρέχουσα τιμή του μετρητή (count), την τελευταία ενέργεια που πραγματοποιήθηκε (lastAction) και την ώρα της τελευταίας ενέργειας (time).
- εμφανίζει στην οθόνη: τη τιμή του μετρητή, τρία κουμπιά: **Αύξηση**, **Μείωση**, **Επαναφορά** και τη τελευταία ενέργεια με την ώρα που έγινε.



Παράδειγμα Counter (2)


```tsx
import { useState } from "react";
import CounterButton from "./CounterButton.tsx";

type CounterState = {
  count: number;
  lastAction: string;
  time: string;
};

const CounterAdvanced : () => Element  = () : Element  => {  Show usages  new *

  const [state, setState] = useState<CounterState>({
    count: 0,
    lastAction: "",
    time: "",
  });
```

- Εισάγει το hook **useState** από τη βιβλιοθήκη react.
- Εισάγει ένα component με όνομα **CounterButton**.
- Δημιούργει έναν τύπο για το State του component όπου περιλαμβάνει 3 πεδία.
- Δημιουργεί ένα state object που ακολουθεί τον τύπο **CounterState**.

Παράδειγμα Counter (3)


```
const getCurrentTime : () => string  = () : string  => new Date().toLocaleTimeString();

const increaseCount : () => void  = () : void  => {  Show usages  new *
  setState({
    count: state.count + 1,
    lastAction: "Increase",
    time: getCurrentTime(),
  });
};
const decreaseCount : () => void  = () : void  => {  Show usages  new *
  if (state.count > 0) {
    setState({
      count: state.count - 1,
      lastAction: "Decrease",
      time: getCurrentTime(),
    });
  }
};
const resetCount : () => void  = () : void  => {  Show usages  new *
  setState({
    count: 0,
    lastAction: "Reset",
    time: getCurrentTime(),
  });
};
```

- Η **getCurrentTime** επιστρέφει την τρέχουσα ώρα σε μορφή string όπου χρησιμοποιεί το Date object της JavaScript.
- Η μέθοδος **.toLocaleTimeString()** επιστρέφει την ώρα σύμφωνα με τις τοπικές ρυθμίσεις (locale) του browser.
- Οι συναρτήσεις **increaseCount**, **decreaseCount**, **resetCount** αλλάζουν την τιμή του μετρητή και εμφανίζουν την τελευταία ενέργεια και την ώρα αυτής.

Παράδειγμα Counter (4)


```
  return (
    <>
      <h1 className="text-center text-2xl">Count is {state.count}</h1>
      <div className="text-center space-x-4 pt-8">
        <CounterButton onClick={increaseCount} label="Increase" />
        <CounterButton
          onClick={decreaseCount}
          label="Decrease"
          disabled={state.count === 0}
        />
        <CounterButton
          onClick={resetCount}
          label="Reset"
          disabled={state.count === 0}
        />
      </div>
        <p className="text-center text-cf-gray pt-8">
          Last change: <strong>{state.lastAction || "-"}</strong> at{" "}
          <strong>{state.time || "-"}</strong>
        </p>
    </>
  );
};

export default CounterAdvanced; Show usages  new *
```

- Το Component επιστρέφει έναν μετρητή, Button για Increase, Decrease, και Reset.
- Εμφανίζει την τελευταία ενέργεια και την ώρα που έγινε.
- Κάνει export το component ώστε να μπορεί να χρησιμοποιηθεί εξωτερικά.

Custom Hooks

- Είναι απλές συναρτήσεις JavaScript που ξεκινούν με use και μας επιτρέπουν να επαναχρησιμοποιήσουμε μια λειτουργία που βασίζεται σε hooks, όπως το useState, useEffect, useContext, κ.ά., χωρίς να επαναλαμβάνουμε τον ίδιο κώδικα σε πολλά components.
- Χαρακτηριστικά:
  - Ξεκινάνε πάντα με **use** (π.χ. useCounter, useForm, useFetch, useLocalStorage)
  - Μπορούν να καλούν **άλλα hooks** (useState, useEffect, κλπ.)
  - Δεν επιστρέφουν JSX, αλλά μόνο τιμές ή συναρτήσεις.

Παράδειγμα Custom Hook (1)


```tsx
import { useCounter } from "../hooks/useCounter.ts";
import CounterButton from "./CounterButton.tsx";

const CounterWithCustomHook : () => Element  = () : Element  => {  Show usages  new *
  const { count, lastAction, time, increase, decrease, reset } = useCounter();

  return (
    <>
      <h1 className="text-center text-2xl">Count is {count}</h1>
      <div className="text-center space-x-4 pt-8">
        <CounterButton onClick={increase} label="Increase" />
        <CounterButton onClick={decrease} label="Decrease" disabled={count === 0} />
        <CounterButton onClick={reset} label="Reset" disabled={count === 0} />
      </div>
      <p className="text-center  text-cf-gray pt-8">
        Last change: <strong>{lastAction || "-"}</strong> at{" "}
        <strong>{time || "-"}</strong>
      </p>
    </>
  );
};

export default CounterWithCustomHook;  Show usages  new *
```

- Από το προηγούμενο παράδειγμα με τον Counter κρατήσαμε μόνο την εμφάνιση του component και μεταφέραμε όλη την λειτουργία του στο custom hook που δημιουργήσαμε (useCounter) το οποίο το καλέσαμε μέσα στο component.

Παράδειγμα Custom Hook (2)


```tsx
import { useState } from "react";

type CounterState = {
  count: number;
  lastAction: string;
  time: string;
};

export const useCounter : () => { count: number; lastAction: string... = () : { count: number; lastAction: string; time... => {
  const [state, setState] = useState<CounterState>({
    count: 0,
    lastAction: "",
    time: "",
  });

  const getCurrentTime : () => string = () : string => new Date().toLocaleTimeString(); Show usages new *
```

- Εισάγει το hook **useState** από τη βιβλιοθήκη react.
- Εισάγει ένα component με όνομα **CounterButton**.
- Δημιούργει έναν τύπο για το State του component όπου περιλαμβάνει 3 πεδία.
- Δημιουργεί ένα state object που ακολουθεί τον τύπο **CounterState**.
- Η **getCurrentTime** επιστρέφει την τρέχουσα ώρα σε μορφή string όπου χρησιμοποιεί το Date object της JavaScript.

Παράδειγμα Custom Hook (3)


```
const increase : () => void  = () : void  => {
  setState({
    count: state.count + 1,
    lastAction: "Increase",
    time: getCurrentTime(),
  });
};

const decrease : () => void  = () : void  => {
  if (state.count > 0) {
    setState({
      count: state.count - 1,
      lastAction: "Decrease",
      time: getCurrentTime(),
    });
  }
};

const reset : () => void  = () : void  => { Sho
  setState({
    count: 0,
    lastAction: "Reset",
    time: getCurrentTime(),
  });
};

return {
  count: state.count,
  lastAction: state.lastAction,
  time: state.time,
  increase,
  decrease,
  reset,
};
};
```

- Οι συναρτήσεις **increaseCount**, **decreaseCount**, **resetCount** αλλάζουν την τιμή του μετρητή και εμφανίζουν την τελευταία ενέργεια και την ώρα αυτής.
- Το Hook επιστρέφει το **state** και τις **συναρτήσεις διαχείρισης** για χρήση σε ένα component. Έτσι, το component μπορεί να εμφανίζει και να χειρίζεται το μετρητή χωρίς να γνωρίζει πώς λειτουργεί εσωτερικά.