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

React JS
State Management

Δαμιανός Μαρινάτος

State Management

- Το state είναι ένα σύνολο εσωτερικών δεδομένων που διατηρεί ένα component και μπορεί να μεταβάλλεται με την πάροδο του χρόνου, επηρεάζοντας άμεσα το περιεχόμενο που εμφανίζεται στο UI.
- Η διαφορά με τα props είναι ότι τα δεδομένα τους είναι **εξωτερικά** και **αμετάβλητα** από το component.

React re-renders

- Η React παρακολουθεί το **state:**
  Όταν ορίζουμε μια μεταβλητή με το useState, η React «θυμάται» το αρχικό της περιεχόμενο και το συνδέει με το συγκεκριμένο component.
- Όταν αλλάζει το state με **setState**:
  - Δεν αλλάζει το UI άμεσα.
  - **Ξανατρέχει** τη function του component.
  - Δημιουργεί νέο εικονικό DOM (Virtual DOM).
  - Συγκρίνει το νέο με το παλιό (Diffing).
  - Κάνει update στο Real DOM μόνο όποτε χρειάζεται.

State in Class Components (1)


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

type State = {
  count: number;
};

class ClassComponentWithState extends Component<{}, State> {  Show usages  new *
  constructor(props: {}) {  no usages  new *
    super(props);
    this.state = { count: 0 };
  }

  increase : () => void  = () : void  => {  Show usages  new *
    this.setState({ count: this.state.count + 1 });
  };

  render() : Element  {  no usages  new *
    return (
      <>
        <h1 className="text-center">Count is {this.state.count}</h1>
        <div className="text-center">
          <button
            className="bg-black text-white py-2 px-4 "
            onClick={this.increase}
          >
            Increase
          </button>
        </div>
      </>
    );
  }
}

export default ClassComponentWithState;  Show usages  new *
```

- Είναι **κλασικό component** που χρησιμοποιεί **state** με την κλάση Component της React.
- Το **state** περιέχει μόνο ένα πεδίο **count**, το οποίο ξεκινά από 0.
- Η μέθοδος **increase()** καλείται με onClick και αυξάνει την τιμή του count κατά 1.
- Το **this.setState()** μας επιτρέπει να αλλάξουμε το state σε class components.
- Κάθε αλλαγή στο state προκαλεί **re-rendering** του component και ενημερώνει το UI.

State in Class Components (2)


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

type State = {
  count: number;
};

class ClassComponentWithState extends Component<{}, State> {  Show usages  new *
  constructor(props: {}) {  no usages  new *
    super(props);
    this.state = { count: 0 };
  }

  increase : () => void  = () : void  => {  Show usages  new *
    this.setState({ count: this.state.count + 1 });
  };

  render() : Element  {  no usages  new *
    return (
      <>
        <h1 className="text-center">Count is {this.state.count}</h1>
        <div className="text-center">
          <button
            className="■bg-black ■text-white py-2 px-4 "
            onClick={this.increase}
          >
            Increase
          </button>
        </div>
      </>
    );
  }
}

export default ClassComponentWithState;  Show usages  new *
```

constructor(props: {})

- Είναι ο **constructor** της κλάσης. Εκτελείται **μόνο μία φορά** όταν δημιουργείται το component.
- Παίρνει ως όρισμα τα props που στέλνονται στο component από τον γονέα του.
- Το {} σημαίνει ότι **δεν περιμένουμε κανένα prop**.

super(props)

- Καλεί τον constructor της **γονικής κλάσης** (Component της React).
- Είναι **απαραίτητο** να καλέσουμε super(props) πριν χρησιμοποιήσουμε το **this** μέσα στον constructor.
- Επιτρέπει στο component να έχει πρόσβαση στο **this.props**.

this.state = { count: 0 };

- Αρχικοποιεί το **εσωτερικό state** του component.
- Δημιουργεί το πεδίο count και του δίνει αρχική τιμή 0.
- Το state αυτό είναι **τοπικό** και **μπορεί να αλλάξει με this.setState()**.

State in Class Components (3)


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

type State = {
  count: number;
};

class ClassComponentWithState extends Component<{}, State> {  Show usages  new *
  constructor(props: {}) {  no usages  new *
    super(props);
    this.state = { count: 0 };
  }

  increase : () => void  = () : void  => {  Show usages  new *
    this.setState({ count: this.state.count + 1 });
  };

  render() : Element  {  no usages  new *
    return (
      <>
        <h1 className="text-center">Count is {this.state.count}</h1>
        <div className="text-center">
          <button
            className="■bg-black ■text-white py-2 px-4 "
            onClick={this.increase}
          >
            Increase
          </button>
        </div>
      </>
    );
  }
}

export default ClassComponentWithState;  Show usages  new *
```

increase = () => { ... }

- Είναι μια **arrow function** μέθοδος της κλάσης.
- Ορίζεται έτσι για να **κρατήσει σωστά το this**, ώστε να δείχνει στο component.
- Αυτό είναι σημαντικό όταν τη χρησιμοποιούμε σε onClick, γιατί αλλιώς το this μπορεί να χαθεί ή να είναι undefined.

this.setState({ count: ... })

- Είναι η **μοναδική σωστή μέθοδος** για να αλλάξουμε το state σε class components.
- Δεν αλλάζουμε το state **άμεσα** (π.χ. this.state.count++ είναι λάθος).

this.state.count + 1

- Παίρνουμε την τρέχουσα τιμή του count και προσθέτουμε 1.

State in Functional Components


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

const Counter : () => Element = () : Element => { Show usages new *
  const [count, setCount] = useState( initialState: 0);

  const increaseCount : () => void = () : void => { Show usages new *
    setCount(count + 1);
  };

  return (
    <>
      <h1 className="text-center">Count is {count}</h1>
      <div className="text-center space-x-4">
        <button
          className="bg-black text-white py-2 px-4 "
          onClick={increaseCount}
        >
          Increase
        </button>
      </div>
    </>
  );
};

export default Counter; no usages new *
```

Χρησιμοποιεί το Hook useState

- Αρχικοποιεί μια μεταβλητή count με τιμή 0
- Η **setCount** είναι η συνάρτηση που χρησιμοποιούμε για να αλλάξουμε την τιμή του count.
- Το useState δίνει **state** σε functional components, όπως το **this.state** σε class components.

Η συνάρτηση increaseCount

- Όταν πατηθεί το κουμπί, η increaseCount καλεί την setCount και αυξάνει τον count κατά 1.
- Αυτό προκαλεί **re-rendering** του component και το UI ενημερώνεται με τη νέα τιμή.

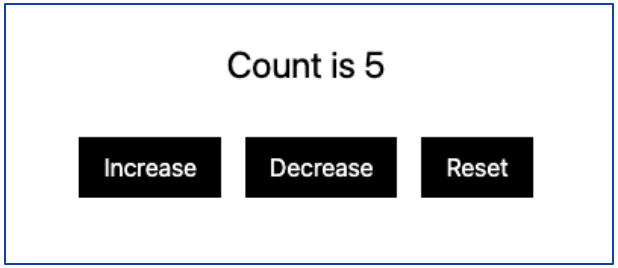
Class vs Functional Components

Functional με useState	Class με this.state
useState(0)	this.state = { count: 0 }
setCount(...)	this.setState(...)
Καθαρότερος κώδικας	Περισσότερο boilerplate
Λιγότερη διαχείριση του this	Χρήση this, constructor, binding

Παράδειγμα Counter (1)

Θα φτιάξουμε ένα **Component** με **Counter** που:

- Χρησιμοποιεί το useState hook για να διαχειρίζεται την τιμή του μετρητή.
- Εμφανίζει το τρέχον count στην οθόνη.
- Παρέχει τρία buttons για **αύξηση**, **μείωση** και **επαναφορά** του μετρητή.
- Το state και οι συναρτήσεις χειρισμού (handlers) βρίσκονται στον γονικό Component (Counter) και περνιούνται στα buttons μέσω props.



Παράδειγμα Counter (2)

- /Components/Counter.tsx

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

const Counter : () => Element = () : Element => { Show usages new *
  const [count, setCount] = useState( initialState: 0);
  const increaseCount : () => void = () : void => { Show usages new *
    setCount(count + 1);
  };
  const decreaseCount : () => void = () : void => { Show usages new *
    if (count > 0) {
      setCount(count - 1);
    }
  };
  const resetCount : () => void = () : void => { Show usages new *
    setCount( value: 0);
  };
  return (
    <>
      <h1 className="text-center text-2xl">Count is {count}</h1>
      <div className="text-center space-x-4 pt-8">
        <CounterButton
          onClick={increaseCount}
          label="Increase"
        />
        <CounterButton
          onClick={decreaseCount}
          label="Decrease"
          disabled={count === 0}
        />
        <CounterButton
          onClick={resetCount}
          label="Reset"
          disabled={count === 0}
        />
      </div>
    </>
  );
};

export default Counter; Show usages new *
```

Παράδειγμα Counter (3)

- /Components/CounterButton.tsx

```tsx
type ButtonProps = {
  onClick: () => void;
  disabled?: boolean;
  label: string;
};

const CounterButton : {{ onClick, disabled, label }: ButtonProp... = ({ onClick, disabled = false, label }: ButtonProps) : Element => {
  return (
    <>
      <button
        className="bg-black disabled:bg-gray-500 text-white py-2 px-4 "
        onClick={onClick}
        disabled={disabled}
      >
        {label}
      </button>
    </>
  );
};

export default CounterButton; Show usages  new *
```

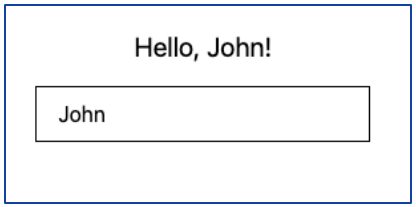
Παράδειγμα Counter (4)

- App.tsx

```tsx
import Counter from "./components/Counter.tsx";

function App() : Element  {  Show usages  new *

  return (
    <>
      <Counter/>
    </>
  )
}

export default App  Show usages  new *
```

Παράδειγμα NameChanger(1)

- Φτιάχνουμε ένα component όπου ο χρήστης πληκτρολογεί το όνομά του σε ένα input.
- Το όνομα αποθηκεύεται στο state και εμφανίζεται δυναμικά στην οθόνη.
- Αν δεν έχει δοθεί όνομα, εμφανίζεται το μήνυμα "Hello, stranger!".



Παράδειγμα NameChanger(2)

- Πώς ένα input πεδίο μπορεί να ενημερώνει δυναμικά το state και να εμφανίζει το όνομα στην οθόνη

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

const NameChanger : () => Element  = () : Element  => {  Show usages  new *
  const [name, setName] = useState( initialState: "");

  const handleChange : (e: ChangeEvent<HTMLInputElement>) => voi...  = (e: React.ChangeEvent<HTMLInputElement>) : void  => {
    setName(e.target.value);
  };

  return (
    <>
      <h1 className="text-center text-xl">Hello, {name || "stranger"}!</h1>
      <div className="text-center mt-4">
        <input
          type="text"
          value={name}
          onChange={handleChange}
          placeholder="Enter your name"
          className="border px-4 py-2"
        />
      </div>
    </>
  );
};

export default NameChanger;  Show usages  new *
```

Παράδειγμα NameChanger(3)

- **React.ChangeEvent<T>** είναι ένας generic τύπος **(type)** που αντιπροσωπεύει ένα change event στη React.
- Το **React.ChangeEvent<HTMLInputElement>** είναι τύπος στη TypeScript που χρησιμοποιείται για να περιγράψει το αντικείμενο event που επιστρέφεται από ένα onChange σε ένα input στοιχείο στη React.
- Το **<HTMLInputElement>** λέει ότι το στοιχείο που πυροδοτεί το event είναι ένα <input>.
- Το **<HTMLSelectElement>** λέει ότι το στοιχείο που πυροδοτεί το event είναι ένα <select>
- Και το **<HTMLTextAreaElement>** λέει ότι το στοιχείο που πυροδοτεί το event είναι ένα <textarea>

Παράδειγμα NameChanger(4)

- App.tsx

```tsx
import NameChanger from "./components/NameChanger.tsx";

function App() : Element  {  Show usages  new *

  return (
    <>
      <NameChanger/>
    </>
  )
}

export default App  Show usages  new *
```

Εργασία

- Φτιάξετε ένα **μικρό project** της επιλογής σας (2-3 components) χρησιμοποιώντας **components, props & states**.