FULL STACK COURSE

הרווחה, משרד העבודה
והשירותים החברתיים

Part III - State & Lifecycle

מה ראינו עד כה?

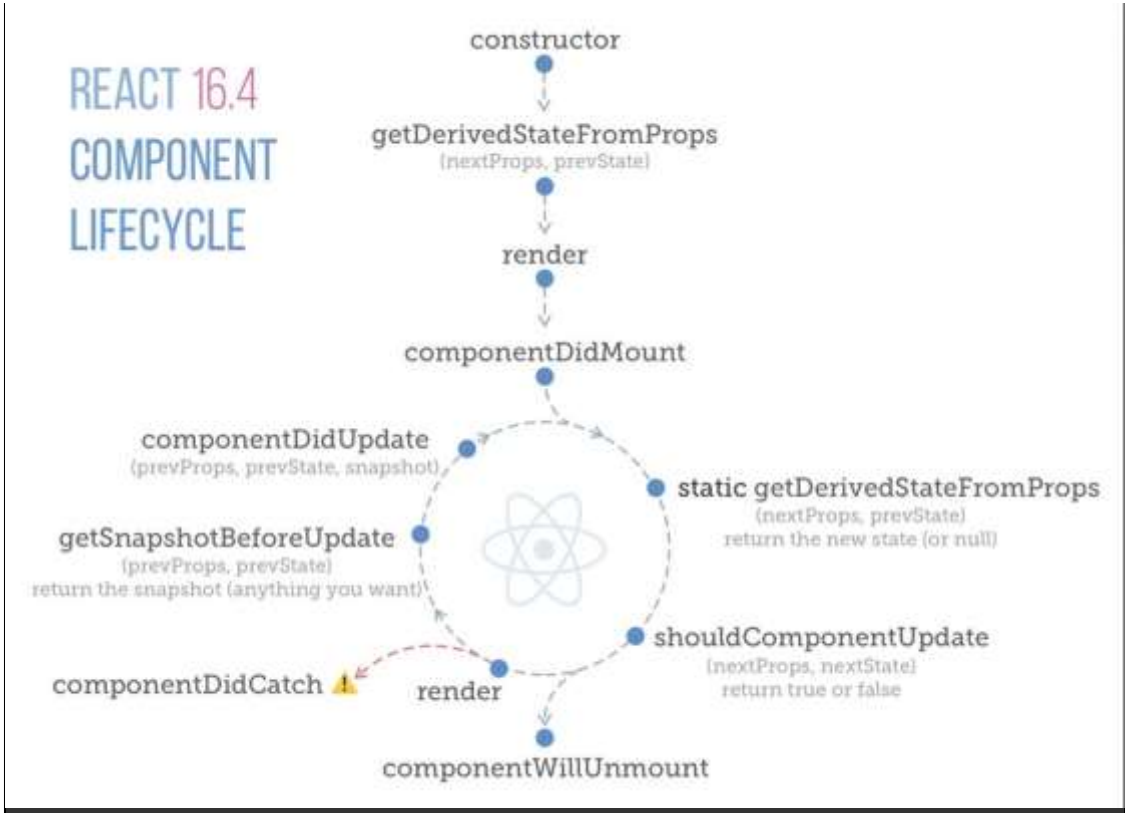
- What is React?
- Components
- JSX
- Props
- PropTypes
- Listes & Keys
- Class Components
- State

מחזור חיים של קומפוננטה

סדרת עדכונים שמתבצעים על הקומפוננטה מהשלב יצירתו עד מותו נקראת מחזור חיים של הקומפמוננטה(lifecycle).

הLifecycle מתחלק ל3 קטגוריות:

- Mounting
- Rendering
- Unmounting



מחזור חיים של קומפוננטה

- Mounting:

שלב יצירת הקומפוננטה וההגה של התוכן שלו.

משתמשים ב Constructor שקורה לפני ה**render** הראשון

(ולכן לפני יצירת הDOM) כדי לאתחל את הstate של

הקומפוננטה.

הפונקציה **componentDidMount** נקראת אחרי ה **render**

הראשון.

משתמשים בפונקציה הזו בשליפות http requests ל

ENDPOINT API.

```
class DataComponent extends
React.PureComponent {
  constructor(props) {
    super(props)
    this.state = {
      data: null
    }
  }

  componentDidMount() {
    getDataFromServer().then(result => {
      this.setState({status: result})
    })

  }

  render() {
    return this.state.data ? (
      <div>
        {this.state.data}
      </div>
    ) : "Loading"
  }
}
```

מחזור חיים של קומפוננט

- Rendering:

שלב מחזורית שחוזרת על עצמה כל עוד או
שיש עדכון ב props או בstate.

משתמשים בפונקציה componentDidUpdate
כדי להשוות בין props/state נוכחים/ישנים
ולפעול לפי זה.

```
componentDidMount() {

this.fetchData(this.props.userID);
}


componentDidUpdate(prevProps) {
  if (this.props.userID !==
prevProps.userID) {

this.fetchData(this.props.userID);
  }
}
```

מחזור חיים של קומפוננטה

- Unmounting:

  שלב השמדה של הקומפוננטה, ונקראת לפני שהקומפוננטה מתעלמת מהDOM.

  משתמשים במתודה

  componentWillUnmount כדי לשחרר

  משאבים.

```
    componentDidMount() {
        this.interval =
setInterval(() => {
            console.log('Hello')
        }, 2000)
    }

    componentWillUnmount() {
        clearInterval(this.interval)
    }
```

Hooks (1) : useState()

this.setState()

```
import React, { Component } from "react";

export default class Button extends Component {
  constructor() {
    super();
    this.state = { buttonText: "Click me, please" };
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(() => {
      return { buttonText: "Thanks, been clicked!" };
    });
  }

  render() {
    const { buttonText } = this.state;
    return <button onClick={this.handleClick}>{buttonText}</button>;
  }
}
```

useState()

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

export default function Button() {
  const [buttonText, setButtonText] = useState("Click me, please");

  return (
    <button onClick={() => setButtonText("Thanks, been clicked!")}>
      {buttonText}
    </button>
  );
}
```

Hooks (2) : useEffect()

componentDidMount()

```
import React, { Component } from "react";

export default class DataLoader extends Component {
  state = { data: [] };

  componentDidMount() {
    fetch("http://localhost:3001/links/")
      .then(response => response.json())
      .then(data =>
        this.setState(() => {
          return { data };
        })
      );
  }

  render() {
    return this.props.render(this.state.data);
  }
}
```

useEffect()

```
import React, { useState, useEffect } from "react";

export default function DataLoader() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch("http://localhost:3001/links/")
      .then(response => response.json())
      .then(data => setData(data));
  });

  return (
    <div>
      <ul>
        {data.map(el => (
          <li key={el.id}>{el.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

Hooks (2) : useEffect() – 1 ב 3

```
useEffect(() => {

  const socket = socketIOClient(ENDPOINT);
  socket.on("FromAPI", data => {
    setResponse(data);
  });

  return () => socket.disconnect();

}, []);
```

componentDidMount()

componentWillUnmount()

componentDidUpdate()