

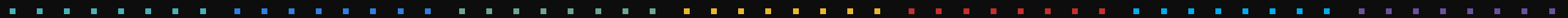
WMNWA 2110 | WEB APPLICATIONS

React #7

Side Effects



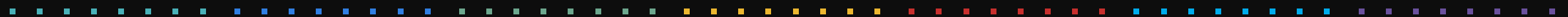
ΠΕΡΙΕΧΟΜΕΝΑ



Περιεχόμενα

- **useEffect**
 - Clean-up
 - The Dependency Array
 - **API** calls
- **useLayoutEffect**

SIDE EFFECTS



Side Effects

"Effectful" Code

Κάποιες ενέργειες δεν μπορούν να γίνουν το κύριο σώμα ενός *Function Component*, αυτό που αναφέρεται και ως *render phase*.

Τέτοιες ενέργειες, συνήθως, έχουν να κάνουν με:

- *mutations*
- *subscriptions*
- *timers*
- *logging*

Side Effects

(Un)intended Consequences

Αν και η *Functional* προσέγγιση βασίζεται σε μεγάλο βαθμό στην «καθαρότητα» (*purity*), κάποια στιγμή, κάπου, κάτι πρέπει να υποστεί αλλαγές.

Χαρακτηριστικά παραδείγματα είναι τα παρακάτω:

- **API** call
- **DOM** manipulation
- *Event listener* registering/de-registering
- *Local/Session storage* access
- Logging
- κ.λπ.

Σε αυτές τις περιπτώσεις, θα χρειαστούμε τη βοήθεια της **useEffect**.

Side Effects

useEffect *hook*

Η **useEffect** δέχεται, ως κύριο όρισμα, ένα *callback function* το οποίο περιέχει *imperative* και *effectfull* κώδικα.

Ο κώδικας αυτός εκτελείται όταν ολοκληρωθεί η *render phase* του **Component**.

```
useEffect (didUpdate) ;
```

Η **useEffect** παρέχει παρόμοια λειτουργικότητα με τα **componentDidMount**, **componentDidUpdate** και **componentWillUnmount** των *Class Components*.

Παράδειγμα

Side Effects

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

function ClickCounter() {
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount((prev) => prev + 1)}>Click me</button>
    </div>
  );
}

export default ClickCounter;
```


Side Effects

Clean-up

Συχνά, χρειάζεται να εκτελεστεί κώδικας όταν κάποιο **Component** «αποπροσαρτηθεί» (*unmount*) από το **DOM**.

Αυτό μπορεί να αφορά σε κάποιο *event listener*, *subscription*, *timer*, κ.λπ.

Ο κώδικας που θα κάνει *remove*, *unregister* ή *clear* κάτι από τα παραπάνω, θα πρέπει να επιστρέφεται ως **function** από το όρισμα της **useEffect**.

```
useEffect(() => {
  // similar to the componentDidMount & componentDidUpdate events
  ...
  return () => {
    // similar to the componentWillUnmount event
    ...
  };
});
```

Παράδειγμα #1

Side Effects

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

const Ticker = () => {
  const getTimestamp = () => new Date().toLocaleTimeString();
  const [timestamp, setTimestamp] = useState(getTimestamp());

  useEffect(() => {
    const handler = setInterval(() => {
      setTimestamp(getTimestamp());
    }, 1000);

    return () => {
      clearInterval(handler);
    };
  }, []);

  return <div>Current time: {timestamp}</div>;
};

export default Ticker;
```

Παράδειγμα #2

Side Effects

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

const MouseTracker = () => {
  const [x, setX] = useState(0);
  const [y, setY] = useState(0);

  const setPosition = ({ x, y }) => { setX(x); setY(y); };

  useEffect(() => {
    window.addEventListener('mousemove', setPosition);
    return () => {
      window.removeEventListener('mousemove', setPosition);
    };
  }, []);

  return <div>Current mouse position: {x}, {y}</div>;
};

export default MouseTracker;
```

Side Effects

The Dependency Array

Στα προηγούμενα παραδείγματα, η **useEffect** έχει κληθεί με δύο ορίσματα. Το πρώτο είναι η *callback function* και το δεύτερο (προαιρετικό) είναι ένας πίνακας με «εξαρτήσεις» (*dependencies*).

Αν το δεύτερο όρισμα απουσιάζει, τότε η *callback function* καλείται σε κάθε *re-render*.

Μιας και αυτό, συχνά, δεν είναι επιθυμητό, παρέχουμε στην **useEffect** (μέσω ενός πίνακα) κάποιες αναφορές, ώστε η *callback function* να εκτελείται μόνο όταν οι «εξαρτήσεις» αυτές έχουν υποστεί αλλαγές.

ΠΡΟΣΟΧΗ: Η σύγκριση που γίνεται στα **deps** είναι *shallow equality check*, για αυτό και θέλει προσοχή όταν κάποιο από τα **deps** δεν είναι *primitive type*.

Side Effects

API calls

Ίσως η πιο συχνή χρήση του **useEffect** *hook* είναι για την κλήση κάποιου **API**.

Είτε το **Fetch API**, είτε κάποια βιβλιοθήκη, όπως η **axios** μπορούν να χρησιμοποιηθούν για αυτή τη δουλειά.

Όταν η κλήση επιστρέψει θα πρέπει να ενημερώσει το **state** του **component**.

Και η ενημέρωση αυτή θα οδηγήσει, εν συνεχεία, σε *re-render*.

Παράδειγμα #1

Side Effects

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

function Movies() {
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    fetch('/movies')
      .then((res) => res.json())
      .then(setMovies);
  }, []);

  if (!movies.length) return <div>Loading...</div>;

  return (
    <ol>
      {movies.map((m) => (
        <li key={m.id}>{m.title}</li>
      ))}
    </ol>
  );
}

export default Movies;
```

Παράδειγμα #2

Side Effects

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

function FetchEmployees() {
  const [employees, setEmployees] = useState([]);

  useEffect(() => {
    async function fetchEmployees() {
      const response = await fetch('/employees');
      const fetchedEmployees = await response.json(response);
      setEmployees(fetchedEmployees);
    }

    fetchEmployees();
  }, []);

  return (
    <div>
      {employees.map(name => <div>{name}</div>)}
    </div>
  );
}
```

Παράδειγμα #3

Side Effects

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

function useFetch(uri) { // Custom hook
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    setLoading(true);
    setError(null);

    fetch(uri)
      .then((response) => response.json())
      .then((json) => { setData(json); setLoading(false); })
      .catch((err) => { setError(err); setLoading(false); });
  }, [uri]);

  return { data, loading, error };
}

export default useFetch;
```


Side Effects

useLayoutEffect

Ο τρόπος κλήσης της **useLayoutEffect** είναι πανομοιότυπος με αυτόν της **useEffect**.

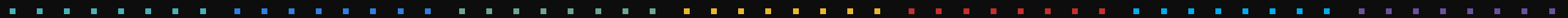
Σε αντίθεση με την **useEffect**, η **useLayoutEffect** εκτελείται «σύγχρονα» μετά από όλες τις αλλαγές στο **DOM**, αλλά πριν το *paint* στο *browser*.

Αυτό είναι χρήσιμο όταν θέλουμε να κάνουμε αλλαγές στο **DOM** που θα αποτυπωθούν στο *paint* εξαρχής και όχι ασύγχρονα σε δεύτερο χρόνο.

Εάν κάποιο *component* τρεμοπαίζει (**flickering**) όταν ενημερώνεται η κατάσταση του - π.χ γίνεται *render* σε μια αρχική κατάσταση και κατόπιν (μέσω της **useEffect**) - αυτό είναι μια καλή ένδειξη ότι πρέπει να χρησιμοποιηθεί η **useLayoutEffect**.

Διαφορετικά η **useEffect** είναι 99% η σωστή επιλογή.

HOMework



Homework


Github user info

Δημιουργήστε ένα **React component**, το οποίο θα δείχνει πληροφορίες για έναν χρήστη του **Github**.

Το **component** πρέπει να δέχεται ως **prop** το **username** του χρήστη και με αυτό να καλεί, μέσω της **useEffect**, το **endpoint** `"https://api.github.com/users/{username}"`.

Στη συνέχεια, ενσωματώστε το **component** σε ένα **React app** όπου κάποιος θα μπορεί να επιλέξει/εισάγει ένα **username** και θα εμφανίζονται οι πληροφορίες του αντίστοιχου χρήστη, μέσω του **component**.

Χρήσιμα links

 Using the Effect Hook – React
<https://reactjs.org/docs/hooks-effect.html>

 Hooks API Reference - useEffect – React
<https://reactjs.org/docs/hooks-reference.html#useeffect...>

 Keeping Components Pure
<https://beta.reactjs.org/learn/keeping-components-p...>

 The last guide to the useEffect Hook you'll ever need - LogRocket Blog
<https://blog.logrocket.com/guide-to-react-useeffect-...>

 A Simple Explanation of React.useEffect()
<https://dmitripavlutin.com/react-useeffect-explanation/>

 Hooks API Reference – useLayoutEffect - React
<https://reactjs.org/docs/hooks-reference.html#uselay...>

 useEffect vs useLayoutEffect
<https://kentcdodds.com/blog/useeffect-vs-uselayout...>

 When to useLayoutEffect Instead of useEffect (example)
<https://daveceddia.com/useeffect-vs-uselayouteffect/>

Extra info

🔗 Is it safe to omit functions from the list of dependencies? - Hooks FAQ – React
<https://reactjs.org/docs/hooks-faq.html#is-it-safe-to-o...>

🔗 ESLint Plugin - Rules of Hooks – React
<https://reactjs.org/docs/hooks-rules.html#eslint-plugin>

🔗 How to Use the React Hook useDeepEffect | by Marco Antonio Ghiani | Better Programming
<https://betterprogramming.pub/how-to-use-the-react...>

🔗 useLocalStorage React Hook - useHooks
<https://usehooks.com/useLocalStorage/>

🔗 donavon/use-persisted-state: A custom React Hook that provides a multi-instance, multi-tab/browser shared and persistent state.
<https://github.com/donavon/use-persisted-state>

🔗 React Query - Hooks for fetching, caching and updating asynchronous data in React
<https://react-query.tanstack.com/>

THANK YOU!