

# WMNWA 2210

## WEB APP DEVELOPMENT



### React 5 | Passing Data





# ΠΕΡΙΕΧΟΜΕΝΑ



## Περιεχόμενα

- Lifting State Up
- Prop Drilling
- Context



# PASSING DATA



# Passing Data

## Lifting State Up

Πολύ συχνά τα **Component** μιας εφαρμογής έχουν την ανάγκη να μοιράζονται *data/state*.

Στην περίπτωση αυτή, δεν έχει το κάθε **Component** τα δικό του **state**, αλλά κρατάμε τα δεδομένα στον πρώτο (ιδανικά) κοινό τους πρόγονο.

Ο στόχος είναι να έχουμε "*a single source of truth*".

Η διαδικασία αυτή είναι αρκετά συχνή και είναι γνωστή ως *lifting state up*.



# Passing Data

## Lifting State Up

Steps:

1. Remove state from the child components.
2. Pass hardcoded data from the common parent.
3. Add state to the common parent and pass it down together with the event handlers.



# Passing Data

## Lifting State Up

Αν κάποιο **Component** έχει την ανάγκη να αλλάξει κάτι σε κοινά *data/state*, τότε θα πρέπει το **Component** που τα "κρατά" να παρέχει μεθόδους για οποιαδήποτε αλλαγή, μιας και δεν υπάρχει απευθείας πρόσβαση.

Στην ουσία, το **Component** γονέας παρέχει **event handlers** στα παιδιά του, που τους επιτρέπει να "επέμβουν" στα κοινά δεδομένα.



# Passing Data

## Lifting State Up

1. When you want to coordinate two components, move their state to their common parent.
2. Then pass the information down through props from their common parent.
3. Finally, pass the event handlers down so that the children can change the parent's state.
4. It's useful to consider components as "controlled" (driven by props) or "uncontrolled" (driven by state).

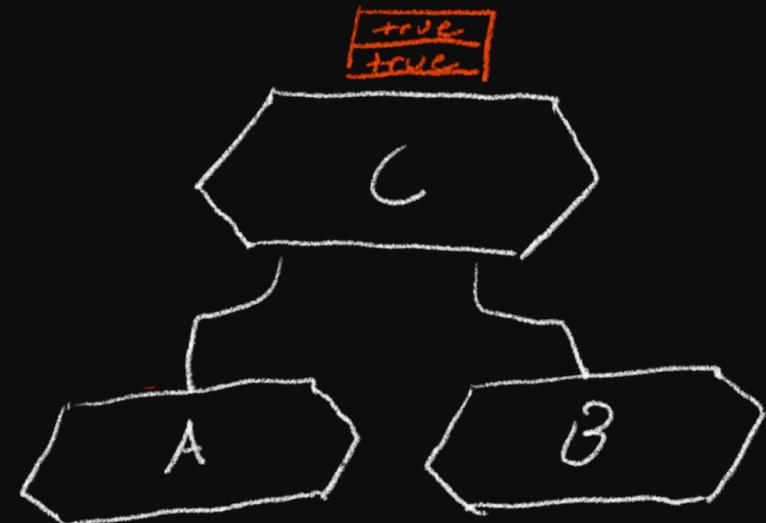


# Passing Data

## Lifting State Up

Independent  
states

"Lifting state up"





# Passing Data

## Prop Drilling

Τα δεδομένα "ταξιδεύουν" πάντα *προς τα κάτω*, συχνά ως **props**, από το ένα **Component** στο άλλο.

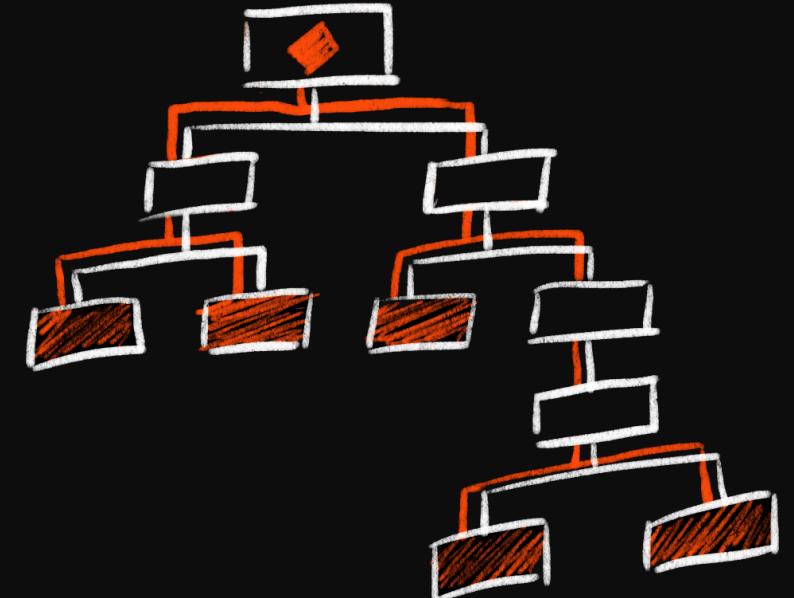
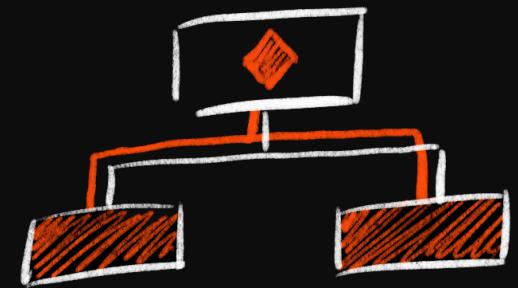
Είναι συνηθισμένη πρακτική, να χρησιμοποιείται μία συνάρτηση **dispatch** για τη δουλειά αυτή.

Ως **dispatch** αναφέρεται η συνάρτηση που προκύπτει από την κλήση του **useReducer hook**.

# Passing Data

## Prop Drilling

“Lifting state up” → “prop drilling”





# Classwork

## Classwork #1

Δημιουργήστε μία εφαρμογή *Todos*, με τη εξής δομή:

- <**Todos** />
  - <**TodoInput** /> (για την εισαγωγή νέων *items*)
  - <**TodosList** />
    - <**TodoItem** />

To <**Todos** /> **Component** κρατά τα απαραίτητα δεδομένα, είτε με τη χρήση του **useState** ή του **useReducer hook**.

Περνά τα δεδομένα αυτά, μέσω **props** στα υπόλοιπα **Component**.



# Passing Data

## Context

Αν η δομή της εφαρμογής έχει αρκετά επίπεδα, τότε η διαδικασία "διαβίβασης" των δεδομένων από **Component** σε **Component**, μπορεί να γίνει δύσκολη στη διαχείρισή της.

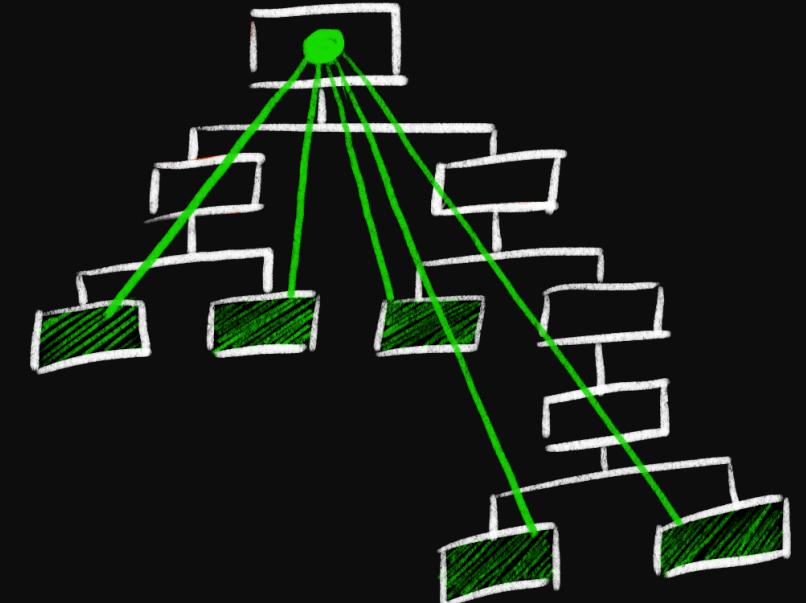
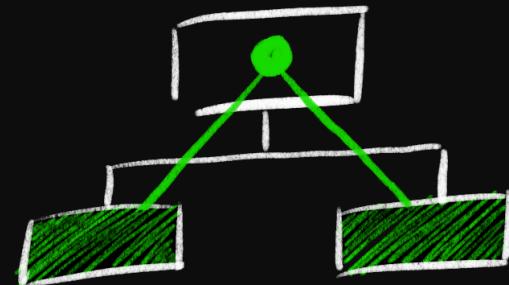
Η **React** παρέχει το μηχανισμό **Context** ώστε τα δεδομένα να είναι προσβάσιμα σε όλα τα στοιχεία, κάτω από ένα στοιχείο "γονέα".

Με το **Context** δεν μας απασχολεί πόσο χαμηλά, στην ιεραρχία, βρίσκεται ένα στοιχείο που χρειάζεται πρόσβαση στα δεδομένα.

# Passing Data

## Context

providing data with context





# Passing Data

## Context

Steps:

1. *Create a context.*
2. *Use that context from the component that needs the data.*
3. *Provide that context from the component that specifies the data.*



# CONTEXT



# Context

## Create a Context

Για τη δημιουργία ενός **Context**, αρκεί να καλέσουμε τη συνάρτηση **createContext**, με όρισμα ένα *initial state*.

```
const MyContext = createContext({ data: [], setData: () => {} });
```

Μέσα στο **Context** που δημιουργείται, υπάρχουν δύο **Component**.  
Ένας **Provider** και ένας **Consumer**.



# Context

## Provide the Context

Ο **Provider** πρέπει να τοποθετηθεί πάνω στη ρίζα του υποδέντρου που μοιράζεται τα δεδομένα.

Έχει ένα **prop** με το όνομα **value**, μέσω του οποίου θα "περασούν" τα δεδομένα και, ίσως, κάποιες μέθοδοι για την διαχείριση τους.

```
<MyContext.Provider value={{ data, setData }}>  
  ...  
</MyContext.Provider>
```



# Context

## Παράδειγμα

```
export const MovieDataContext = createContext();

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

  return (
    <>
      <h1>Box Office</h1>
      <MovieDataContext.Provider value={{ movies, setMovies }}>
        <TopMovies />
        <MovieList />
      </MovieDataContext.Provider>
    </>
  );
}
```



# Context

## Use the Context

Ο **Consumer** πρέπει να τοποθετηθεί σε κάθε στοιχείο "απόγονο" που θέλει πρόσβαση στα δεδομένα.

Η χρήση του **Consumer** δεν είναι απαραίτητη και συχνά, στη θέση του, χρησιμοποιείται μία κλήση στο **useContext hook**.

```
const { data, setData } = useContext(MyContext);
```



# Context

## Παράδειγμα

```
import { useContext } from 'react';
import { MovieDataContext } from './App';

function TopMovies() {
  const { movies } = useContext(MovieDataContext);

  return (<>
    <h2>Top Movies</h2>
    <ol>
      {movies
        .sort((a, b) => b.gross - a.gross)
        .slice(0, 3)
        .map((movie, index) => (
          <li key={index}>
            `${movie.title} ${movie.gross}m ${movie.favorite ? '❤️' : '🖤'}` 
          </li>
        ))
      )
    </ol>
  </>);
}
```



# Context

## Custom hook

Είναι καλή πρακτική να μεταφέρουμε όλη τη λογική ενός **Context** σε ξεχωριστό αρχείο.

```
// in MyContextProvider.jsx

const useMyContextProvider = () => useContext(MyContextProvider);

function MyContextProvider({ children }) {
  const [data, setData] = useState([]);
  return (
    <MyContext.Provider value={{ data, setData }}>
      { children }
    </MyContext.Provider>
  );
}
```

```
// in SomeComponent.jsx

const { data, setData } = useMyContextProvider();
```



# Classwork

## Classwork #2

Τροποποιήστε την εφαρμογή *Todos*, ώστε να κάνει χρήση ενός **Context**.

Κατά προτίμηση, φτιάξτε ένα **custom hook** για τη δουλειά αυτή.



# HOMEWORK



# Homework

## Box Office

Δημιουργήστε έναν **MovieDataProvider** που θα διαχειρίζεται τα δεδομένα ταινιών.

Τα δεδομένα θα πρέπει να έχουν τη μορφή:

```
{ "title": "Some Movie", "rating": 8.1, "sum": 5.1, "favorite": false }
```

Στη συνέχεια, μέσα σε ένα **React** app, χρησιμοποιήστε τον **MovieDataProvider**, ώστε:

- να εμφανίζεται μία λίστα με τα δεδομένα των ταινιών
- να δίνεται η δυνατότητα να οριστεί μια ταινία ως "favorite"

Και οι δύο αυτές λειτουργίες πρέπει να γίνονται μέσα σε κάποιο *sub-component*.



# Χρήσιμα links

Thinking in React

<https://beta.reactjs.org/learn/thinking-in-react>

useReducer - Hooks API Reference – React

<https://reactjs.org/docs/hooks-reference.html#usere...>

Sharing State Between Components

<https://beta.reactjs.org/learn/sharing-state-between-...>

Passing Data Deeply with Context

<https://beta.reactjs.org/learn/passing-data-deeply-wi...>

Scaling Up with Reducer and Context

<https://beta.reactjs.org/learn/scaling-up-with-reducer...>



# Extra info

👉 Prop Drilling

<https://kentcdodds.com/blog/prop-drilling>

DEV React doesn't need state management tool, I said -

DEV Community

[https://dev.to/tolgee\\_i18n/react-doesnt-need-state-...](https://dev.to/tolgee_i18n/react-doesnt-need-state-...)

👉 Getting started with state management using

useReducer and Context · Emma Goto

<https://www.emgoto.com/react-state-management/>

👉 How To useContext With useReducer | Harry Wolff

<https://hswolff.com/blog/how-to-usecontext-with-user...>

👉 React Context API Global State Management -

Modus Create

<https://moduscreate.com/blog/react-context-api-sta...>

👉 How to destroy your app performance using React contexts

<https://thoughtspile.github.io/2021/10/04/react-conte...>

👉 How to use React Context effectively

<https://kentcdodds.com/blog/how-to-use-react-cont...>

👉 How to optimize your context value

<https://kentcdodds.com/blog/how-to-optimize-your-...>

DEV useReducer TypeScript: React Context with

useReducer and Typescript. - DEV Community



<https://dev.to/elisealcala/react-context-with-useredu...>



# THANK YOU!