WMNFE 2410

FRONT-END DEVELOPMENT

React #1 | Getting started

SAE CREATIVE MEDIA EDUCATION

ΠΕΡΙΕΧΟΜΕΝΑ

Περιεχόμενα

- Welcome to **React**
  - Basic Setup
- Elements
- Components
  - JSX

WELCOME TO REACT

Welcome to React

What is React?

Η **React** (παλαιότερα γνωστή και ως **React.js** ή **ReactJS**) είναι μία **JavaScript** βιβλιοθήκη για τη δημιουργία διεπαφών (*UI*).

Δημιουργήθηκε και αναπτύσσεται από την *Facebook* / *Meta* (πλέον & την *Vercel*) και μπορεί να χρησιμοποιηθεί για τη δημιουργία διαδραστικών σελίδων, *SPA* αλλά και *Mobile* εφαρμογών.

Οι βασικές τις λειτουργίες αφορούν στη διαχείριση της κατάστασης μιας εφαρμογής (*state management*) και την αποτύπωση αυτής στο **DOM**. Για περισσότερες λειτουργίες (π.χ. *Routing*) είναι αναγκαία η χρήση βιβλιοθηκών.

BASIC SETUP

Basic Setup (*μέχρι v18*)

Με *CDN* links

Για την ανάπτυξη μίας απλής εφαρμογής, μπορεί να χρησιμοποιηθεί η βιβλιοθήκη **React** απευθείας από κάποιο *CDN*.

Σε αυτή την περίπτωση αρκεί η προσθήκη των παρακάτω **script** στο **head**.

```
<script src="https://unpkg.com/react@18/umd/react.development.js"
        crossorigin defer></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
        crossorigin defer></script>
```

Basic Setup (*μέχρι v18*)

Γιατί δύο ξεχωριστά **script**;

Η **React** δεν αφορά αποκλειστικά σε *web* εφαρμογές, μπορεί να υποστηρίξει και *native* υλοποιήσεις.

Η βασική λειτουργικότητα βρίσκεται στο **react.js**, ενώ η αποτύπωση στον *browser* (**DOM**) γίνεται μέσω του **react-dom.js**.

Στην ουσία, το **react-dom.js** αναλαμβάνει του *rendering* στη σελίδα και το **react.js** όλα (?) τα υπόλοιπα.

Basic Setup (*μέχρι v18*)

... in production

Τα προηγούμενα *URL* είναι αυστηρά για τη φάση του *development*.

Αν και η προσέγγιση αυτή, με τη χρήση των **script** απευθείας από κάποιο *CDN*, ίσως δεν είναι η απόλυτα ενδεδειγμένη, όταν/αν η εφαρμογή ανέβει σε *production* περιβάλλον, θα χρειαστούμε τα παρακάτω **script**.

```
<script src="https://unpkg.com/react@18/umd/react.production.min.js" defer></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.production.min.js" defer></script>
```

Με τα κατάλληλα εργαλεία υπάρχουν και πιο σύνθετοι τρόποι να ξεκινήσει κανείς με μια εφαρμογή **React**.

Basic Setup (*v19+*)

Έκδοση 19

Στην έκδοση **19** της **React**, η αρχιτεκτονική έχει υποστεί σημαντικές αλλαγές που απαιτούν μια νέα προσέγγιση.

Δεν αρκεί πλέον μια απλή αναφορά των *script* της **React** στη σελίδα, καθώς η νέα δομή στηρίζεται σε ένα πιο modular και σύγχρονο σύστημα.

Μετά και την κατάργηση του εργαλείου **CRA**, προτείνεται η δημιουργία εφαρμογών μέσω **Node**, **Vite** ή με τη χρήση *meta-frameworks* όπως το **Next.js**, **React Router**, **Tanstack Router** κ.λπ.

Vite Setup

Εγκατάσταση

Μια από τις πιο εύκολες και γρήγορες λύσεις για τη δημιουργία μιας νέας εφαρμογής **React** είναι μέσω του **Vite**.

Το **Vite** είναι ένα *build tool* που στηρίζεται στα **ES Modules** και το **Rollup** για τη δημιουργία ενός γρήγορου και αποδοτικού περιβάλλοντος ανάπτυξης.

Για να δημιουργήσετε μια νέα εφαρμογή **React** με το **Vite**, αρκεί να εκτελέσετε την ακόλουθη εντολή:

```
# Bash | MacOS/Linux
npm create vite@latest my-react-app -- --template react

# PowerShell | Windows
npx create-vite@latest my-react-app --template react
```

REACT ELEMENTS

React Elements

DOM manipulation

Η **HTML** είναι μια σειρά από οδηγίες τις οποίες ακολουθεί ο *browser* για να χτίσει το **DOM**.

Ως γνωστόν, κάποιος μπορεί να διαχειριστεί το **DOM** και μέσω της **JavaScript** και του **DOM API**, χρησιμοποιώντας μεθόδους όπως οι **document.createElement()**, **document.appendChild()**, κ.λπ.

Με τη **React** δεν αναφερόμαστε στο **DOM** απευθείας, αλλά δημιουργούμε *React elements*, τα οποία αναλαμβάνουν να ενημερώνουν το **DOM** σε κάθε αλλαγή.

React Elements

React.createElement()

Η δημιουργία των **React elements** γίνεται μέσω της μεθόδου **React.createElement()**.

```
React.createElement(type, [props], [...children])
```

- **type**: **string** που αντιπροσωπεύει τον τύπο του *element*, π.χ. ένα *HTML tag* ή κάποιο ήδη ορισμένο *React element*.
- **props** (optional): **JSON** με τα χαρακτηριστικά του *element*, ή **null** αν δεν υπάρχουν κάποια.
- **children** (optional): ένα ή περισσότερα *child elements*. Μπορεί να είναι είτε ένα **string**, από το οποίο θα δημιουργηθεί ένα *text node*, είτε άλλα *React elements*.

React Elements

Props

Τα *properties* ενός **element** μπορεί να είναι, είτε από τα ήδη υπάρχοντα **HTML attributes**, είτε *custom* ιδιότητες.

Μιας και δεν μπορούμε να χρησιμοποιήσουμε το *keyword* **class** (ως *reserved word*), στη θέση του γράφουμε **className**.

Ομοίως, στη θέση του *attribute* **for** γράφουμε **htmlFor**.

React Elements

Παράδειγμα #1

```
const h1 = React.createElement('h1',
  { id: 'main-title'},
  'Hello, World!'
);
const h2 = React.createElement('h2',
  { className: 'secondary-title'},
  'Greetings from React!'
);

const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render([h1, h2]);
```

React Elements

Παράδειγμα #2

```
const h1 = React.createElement('h1', null, 'Recipe for muffins');
const ul = React.createElement('ul', { className: 'ingredients' },
    React.createElement('li', null, '2 cups of flour'),
    React.createElement('li', null, '1 cup of sugar'),
    React.createElement('li', null, '1 cup of milk'),
    React.createElement('li', null, '1/2 cup of oil'),
    React.createElement('li', null, '1 egg'),
);

// Render the elements inside the 'root' div
const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render([h1, ul]);
```

React Elements

Παράδειγμα #3

```
const ingredients = [
    { title: 'flour', quantity: '2 cups', vegan: true },
    { title: 'sugar', quantity: '1 cup', vegan: true },
    { title: 'milk', quantity: '1 cup' },
    { title: 'oil', quantity: '1/2 cup', vegan: true },
    { title: 'eggs', quantity: '1' },
];

const ul = React.createElement('ul',
    { className: 'ingredient-list' },
    ingredients.map(
      (item, index) => React.createElement('li',
        { className: 'ingredient', key: index },
        `${item.quantity} ${item.title} ${item.vegan ? '(v)' : ''}`
      )
    )
);

const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render(el);
```

COMPONENTS

Components

Composition & Re-usability

Στη **React**, αν και το βασικό δομικό στοιχείο είναι το *React Element*, το *UI* μιας εφαρμογής χτίζεται με/συντίθεται από ένα σύνολο στοιχείων *React Component*.

Ένα *component*, όχι μόνο στη **React** αλλά γενικά στον κόσμο των *web framework*, είναι ένα κομμάτι επαναχρησιμοποιούμενου κώδικα, το οποίο εμπεριέχει ότι αυτό χρειάζεται για τη λειτουργικότητά του.

Σε αντίθεση με την κληρονομικότητα, από τον κόσμο του *Αντικειμενοστραφούς Προγραμματισμού*, εδώ συναντάμε την αρχή της *Σύνθεσης* (*Composition*).

Components

Functional Programming

Όπως θα γίνει ξεκάθαρο στη συνέχεια, η **React** βασίζεται στη φιλοσοφία του **Functional Programming**.

Εδώ, το βασικό στοιχείο της εφαρμογής είναι οι συναρτήσεις, και μάλιστα οι "*pure*" συναρτήσεις, στις οποίες το επιστρεφόμενο αποτέλεσμα βασίζεται αποκλειστικά στα δεδομένα εισόδου και δεν προκαλούν *side effects*.

Ένα *React Component* είναι μια συνάρτηση (*function*) που επιστρέφει ένα *React Element*.

Μέσα στη συνάρτηση αυτή, προσθέτουμε όλη τη λειτουργικότητα που χρειάζεται ένα *component*.

Components

Παράδειγμα

```
const movies = [{ title: 'Dune', rating: 8.4, sum: 117 }, ...];

const Header = () => React.createElement('h1', null, 'Box Office')

function MovieList({ items, separator }) { // props destructuring
    return React.createElement('ul',
        null,
        items.map((m, i) => React.createElement('li',
            null,
            `${m.title} | $${m.sum}m | ${m.rating}★`
        ))
    )
}

const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render([
    React.createElement(Header, null),
    React.createElement(MovieList, { items: movies })
])
```

Components

JSX

Η **JSX** σύνταξη επιτρέπει να γράφουμε κώδικα όπως ο παρακάτω:

```
const element = <h1>Hello, world!</h1>;
```

Δεν πρόκειται ούτε για **HTML**, ούτε για **JavaScript**, αλλά είναι η προσπάθεια της **React** για πιο ευανάγνωστο κώδικα.

Με τη βοήθεια της **JSX**, δεν χρειαζόμαστε πλέον τη μέθοδο **React.createElement()**.

Σε σύνθετες εφαρμογές, είναι δύσκολο να δομήσουμε έναν όμορφο και ευανάγνωστο κώδικα με εμφωλευμένες κλήσεις της **React.createElement()**.

Components

Not a template engine

Η **JSX** θυμίζει αλλά δεν είναι *template engine*.

Είναι μια επέκταση της **JavaScript** που επιτρέπει να συνδυάζουμε **markup** με κώδικα **JavaScript** με έναν ευπαρουσίαστο τρόπο.

Ως επέκταση της **JavaScript** μπορεί να εκτελέσει οποιονδήποτε κώδικα **JavaScript**.

Χρειάζεται τη βιβλιοθήκη *Babel* για να μπορέσει να εκτελεστεί σε browser.

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js" defer></script>
```

```
const tick = <span>{new Date().toLocaleTimeString()}</span>;
```

Components

JSX Elements / Components

Με τη βοήθεια της **JSX**, η συγγραφή των **elements** & **components** γίνεται ευκολότερη.

Μπορούμε να παρεμβάλουμε σε οποιοδήποτε σημείο (?) *expressions* σε **JavaScript**, αρκεί να βρίσκεται μέσα σε *curly brackets*.

Οτιδήποτε βρίσκεται μέσα σε **{ }** γίνεται *evaluated* (αποτιμάται).

Components

Παράδειγμα #1

```
const styles = { color: '#61DAFB', backgroundColor: 'black' };

// We need Babel in order for this to be parsed
const header = <h1 id="main-header" style={styles}>
    Hello, from JSX! Time is {new Date().toLocaleTimeString()}
</h1>;

// It's still a React element
console.log(header);

const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render(header);
```

Components

Παράδειγμα #2

```
const movie = { title: 'Dune', rating: 8.4, sum: 117 };

const Header = () => (<h1>Box Office</h1>);

const Movie = ({ movie }) => (
  <div>
    <h2>{movie.title}</h2>
    <span>${movie.sum}m</span> -
    <span>{movie.rating}⭐</span>
  </div>
);

const root = ReactDOMClient.createRoot(document.getElementById('app'));
root.render([
  <Header />,
  <Movie movie={movie} />
]);
```

Χρήσιμα links

Quick Start – React
https://react.dev/learn

Running React 19 From a CDN and using esm.sh | Pe…
https://peterkellner.net/2024/05/10/running-react-19-fro…

ESM>CDN
https://esm.sh

Add React to an Existing Project – React
https://react.dev/learn/add-react-to-an-existing-project

Node.js - Download Node.js®
https://nodejs.org/en/download

Getting Started – Vite
https://vite.dev/guide

DOM Elements – React
https://react.dev/docs/dom-elements.html

Your First Component – React
https://react.dev/learn/your-first-component

Writing Markup with JSX – React
https://react.dev/learn/writing-markup-with-jsx

THANK YOU!