React jegyzet

# HTML, CSS és JavaScript Alapok

## HTML

Ismerni kell a HTML alapjait, mivel a React komponensek végül HTML kódot renderelnek.

## CSS

A stílusok kezeléséhez szükséges alapok. A React-ben gyakran használnak CSS-t vagy CSS-in-JS megoldásokat.

## JavaScript

A React JavaScript alapú, így nélkülözhetetlenek a JavaScript ismeretek. pl.:

* változók deklarálása
* arrow function-ek,

### destrukturálás (tömbökből vagy objektumokból értékek kiemelése),

* template literals stb.
* Aszinkron JavaScript (Promise-ok, async/await)
* Array és Object metódusok (pl. map, filter, reduce)

# React Alapok

## Új poject létrehozás

**npm create vite@latest my-react-app -- --template react**

**cd my-react-app**

**npm install**

**npm run dev**

Az alábbi könyvtárszerkezetet kapjuk:

hello-vite-app/

├── node\_modules/

├── public/

│ ├── favicon.ico

│ └── vite.svg

├── src/

│ ├── assets/

│ │ └── react.svg

│ ├── App.css

│ ├── App.jsx

│ ├── index.css

│ └── main.jsx

├── .gitignore

├── index.html

├── package.json

├── README.md

└── vite.config.js

Az alapértelmezett project szerkesztését az *src* mappában lévő fájlokkal, leginkább az App.jsx fájl-al kezdjük.

## React komponensek

Funkcionális és osztály alapú komponenseket is készíthetünk.

### Funkcionális komponensek

**import** React from 'react'**;**

// Egy egyszerű funkcionális komponens, amely köszönti a felhasználót

**const** Greeting **=** **({** name **})** **=>** **{**

**return** **(**

**<**div**>**

**<**h1**>**Szia**,** **{**name**}!</**h1**>**

**<**p**>**Üdvözlünk a React világában**!</**p**>**

**</**div**>**

**);**

**};**

// Ezt a komponenst használhatod máshol is:

**export** **default** Greeting**;**

Használata:

**import** React from 'react'**;**

**import** Greeting from './Greeting'**;**

**const** App **=** **()** **=>** **{**

**return** **(**

**<**div**>**

**<**Greeting name**=**"Anna" **/>**

**</**div**>**

**);**

**};**

**export** **default** App**;**

### Osztály alapú komponensek

Az osztály alapú komponenseket akkor használták, amikor a komponenseknek belső állapotra (state) vagy lifecycle metódusokra volt szükségük. Azonban a React 16.8[[1]](#footnote-1) verziótól kezdve, a hooks bevezetésével a funkcionális komponensek is képesek kezelni az állapotot és az életciklusokat, így az osztály alapú komponensek használata ritkábbá vált.

**import** React**,** **{** Component **}** from 'react'**;**

// Egy osztály alapú komponens, amely köszönti a felhasználót

**class** Greeting **extends** Component **{**

render**()** **{**

**const** **{** name **}** **=** **this.**props**;** // Props de-strukturálása

**return** **(**

**<**div**>**

**<**h1**>**Szia**,** **{**name**}!</**h1**>**

**<**p**>**Üdvözlünk a React világában**!</**p**>**

**</**div**>**

**);**

**}**

**}**

// Ezt a komponenst használhatod máshol is:

**export** **default** Greeting**;**

Használat:

**import** React from 'react'**;**

**import** Greeting from './Greeting'**;**

**class** App **extends** React**.**Component **{**

render**()** **{**

**return** **(**

**<**div**>**

**<**Greeting name**=**"Anna" **/>**

**</**div**>**

**);**

**}**

**}**

**export** **default** App**;**

## JSX

A React egyedi szintaxisa, ami lehetővé teszi a JavaScript és HTML kombinációját.

## Props

Adatokat átadása komponensek között (*props*)

## State

Állapotok kezelése egy komponensen belül (*state*).

## Komponens Életciklus

Ismerd meg a komponensek életciklus metódusait, hogy megértsd, mikor és miért történnek bizonyos események a komponensekkel (pl. *componentDidMount, componentDidUpdate*).

# 3. Fejlettebb React Fogalmak

## Hook-ok

Ismerd meg a React hook-okat, különösen a useState és useEffect hook-okat. Ezek a funkcionális komponensek állapotkezelésére és mellékhatások kezelésére szolgálnak.

## Context API

Tanuld meg, hogyan lehet megosztani adatokat a komponensek között anélkül, hogy props-okat kellene lefelé küldeni minden szinten.

## React Router

Ismerd meg az útvonalkezelést, hogy hogyan lehet különböző oldalakra navigálni egy egyoldalas alkalmazásban (SPA).

# 4. Állapotkezelés

# Redux vagy MobX

Tanuld meg, hogyan lehet komplex állapotokat kezelni egy alkalmazásban. A Redux a legnépszerűbb állapotkezelő könyvtár Reacthez, de az alternatívák, mint a MobX vagy a Context API is érdemesek lehetnek a tanulásra.

# 5. Styling Technológiák

## CSS-in-JS

Tanuld meg, hogyan lehet CSS-t írni JavaScript-ben, például a Styled Components vagy az Emotion segítségével.

## CSS keretrendszerek

Ismerkedj meg néhány népszerű CSS keretrendszerrel, mint a Bootstrap vagy a Tailwind CSS, amelyeket React-tel gyakran használnak.

# 6. Integrációk és API Hívások

## REST API és GraphQL

Ismerd meg, hogyan hívj meg külső API-kat, és hogyan dolgozz fel adatokat React alkalmazásban. A fetch API és az Axios a két leggyakrabban használt megoldás a REST API-khoz, míg az Apollo Client a legnépszerűbb a GraphQL-hez.

## Build eszközök és deployment

### Create React App

Kezdj ezzel az eszközzel, amely gyorsan felállít egy React fejlesztői környezetet minden szükséges konfigurációval.

### Webpack és Babel

Érdemes megérteni a modern JavaScript alkalmazások felépítéséhez és fordításához használt eszközöket.

### Deployment

Tanuld meg, hogyan lehet a React alkalmazásokat különböző környezetekbe deploy-olni, például Netlify, Vercel, Heroku vagy saját szerverre.

# 8. Tesztelés

## Unit tesztelés

Tanuld meg a komponensek unit tesztelését, például a Jest és a React Testing Library segítségével.

## End-to-End tesztelés

Tanuld meg az alkalmazás teljes folyamatainak tesztelését eszközök segítségével, mint például a Cypress.

# 9. Projektek Építése

Készíts saját kis projekteket, hogy gyakorold a tanultakat. Kezdd egyszerűbb projektekkel, mint például egy teendő lista, majd haladj komplexebb alkalmazások felé, mint például egy e-kereskedelmi oldal vagy egy blog platform.

# 11. Frissítések és Új Technológiák Követése

# A JavaScript és a React gyorsan változó világában fontos, hogy naprakész maradj. Kövess blogokat, podcastokat, és React konferenciákat, hogy megtudd, mik az új trendek és best practice-ek.

1. 2019 [↑](#footnote-ref-1)