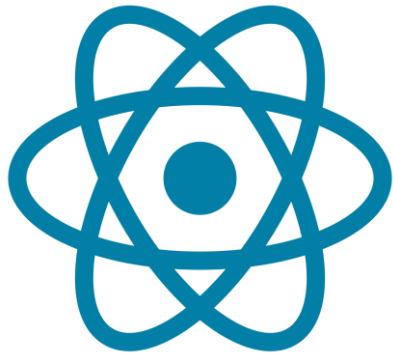


Rauscher Gábor

React alapok

Front-End fejlesztéshez



BBS-INFO Kiadó, 2025.

Minden jog fenntartva! A könyv vagy annak oldalainak másolása, sokszorosítása csak a kiadó írásbeli hozzájárulásával történhet.

A könyv nagyobb mennyiségben megrendelhető a kiadónál:
BBS-INFO Kiadó, www.bbs.hu, Tel.: +36-14-07-17-07

A könyv megírásakor a szerző és a kiadó a lehető legnagyobb gondossággal jártak el. Ennek ellenére, mint minden könyvben, ebben is előfordulhatnak hibák. Az ezen hibákból eredő esetleges károkért sem a szerző, sem a kiadó semmiféle felelősséggel nem tartozik, de a kiadó szívesen fogadja, ha ezen hibákra felhívják figyelmét.

Papírkönyv: ISBN 978-615-6364-40-1
E-book: ISBN 978-615-6364-41-8

Kiadja a BBS-INFO Kft. Budapest

Tartalomjegyzék

1. Bevezetés	5
2. Kezdetek	7
2.1. A React célja.....	7
2.2. A fejlesztési környezet.....	7
2.2.1. Közvetlen HTML fájl használata	8
2.2.2. Projekt használata a create-react-app parancs után	10
3. JavaScript újdonságok a React ES6-ban	14
3.1. Komponens létrehozása React ES6 Class-szal	14
3.2. Rövidebb függvényt meghatározás a React ES6 nyílfüggvénnyel (Arrow Function).....	17
3.3. Változók létrehozása, a React ES6 változó kezelése	18
3.4. Metódusok tömbök, listák kezelésére - React ES6 Array metódusok	19
3.5. Tömb vagy objektum elemeinek szűrése React ES6 Destructuring-gal.....	20
3.6. Tömbök vagy objektumok másolására React ES6 Spread operátor	22
3.7. Változók, függvények átadása React ES6 Modulokkal	22
3.7.1. Exportok.....	23
3.7.2. Importok	23
3.8. Feltételre React ES6 Ternary (háromtagú feltételes) operátor	24
4. HTML elemek létrehozása és megjelenítése React-ben: rendereléssel.....	25
5. A React JSX nyelve.....	27
6. A React komponensek alapjai	29

7. React class komponensek	33
7.1. A class komponensek használatának alapjai	33
7.2. A komponens state objektuma	35
7.3. A komponensek életciklusa	38
8. React props-okhoz kötött adatátvitel	42
9. React eseménykezelők	44
10. React feltételek kezelései	45
11. React listák és tömbök kezelése	47
12. React formok, azaz adatbevitel	48
12.1. Kezdeti tudnivalók a useState-ről	48
12.2. Input használata	49
12.3. Textarea típusú adatbevitel	52
12.4. Select (lenyíló lista) a React-ben	53
13. React Router fájlok navigáció kezelésére	54
13.1. A szerkezet felépítése	54
13.2. A webhely oldalai	57
14. React Memo változatlan komponensre	59
15. A React együttműködése a CSS-sel	61
16. Stílusok SASS használatával	64
17. React Hook-ok életciklusok kezelésére	65
17.1. Mi a Hook?	65
17.2. A useState()	67
17.3. A useEffect()	69
17.4. A useContext()	71
17.5. A useRef()	73
17.6. A useReducer()	75
17.7. A useCallback()	78
17.8. A useMemo()	81
17.9. Egyedi Hook-ok készítése	83
18. Gyakorló feladatok	86
18.1. Csak a szükséges gomb legyen	86
18.2. Mindenre van válasz	90
18.3. Két üdvözlés	96
19. Források, kapcsolat	98

1. Bevezetés

Mi a React?

A React egy JavaScript könyvtár (framework), amely eszközt kínál a felhasználói felületek komponenseinek leírására. Segítségével olyan felhasználói felületet hozhatunk létre – jellemzően HTML alapú alkalmazást –, amelyben JavaScript működik, és ehhez kapcsolódik a React. A React lehetővé teszi újrafelhasználható összetevők, azaz komponensek megírását a kódban. Ezért a React elsősorban Front-End fejlesztők számára ajánlott, akik a JavaScript lehetőségeit szeretnék kibővíteni.

Milyen előfeltétele van e könyv tanulmányozásának?

A könyv anyagának elsajátításához szükséges a HTML, a CSS és a JavaScript alapismerete. Emellett ajánlott egy kódszerkesztő használata, például a Visual Studio Code. Az összetettebb alkalmazások fejlesztéséhez már nem célszerű csupán egy HTML fájlban hivatkozással behívni a React könyvtárát egy külső webhelyről, hanem érdemes több fájlból álló projektkörnyezetet kialakítani. Ehhez szükséges, hogy a Node.js és az npm (Node Package Manager) telepítve legyenek, használatukhoz pedig parancssort (terminált) kell alkalmaznunk. Ezek telepítése, valamint a terminál használata operációs rendszertől függő, és általános informatikai ismeretnek számít. A könyv feltételezi ezen tudás meglétét az összetettebb eljárások bemutatásakor. A React különféle környezetekben történő elindításáról a 2.2. pontban olvashatunk részletesen.

Miről lesz szó a könyvben?

A könyv első része az ES6 új eszközeit mutatja be. Az ES6 az ECMAScript hatodik verziója, amely a JavaScript szabványosítására jött létre. Bár az itt szereplő eljárások a JavaScriptet jól ismerők számára már ismerősek lehetnek, ebben az esetben a React szempontjából tekintjük át őket. Ez tulajdonképpen egy tudásfelfrissítés a JavaScriptből, amely hasznos alapot nyújt a React eljárások későbbi bemutatásához.

A könyv második fontos része a klasszikus React eljárásokat ismerteti, köztük a React Class típusú komponenseit is. Ennek külön tárgyalása azért indokolt, mert bár a Class típusú komponensek már teljes mértékben helyettesíthetők függvény típusú komponensekkel (Hook-okkal), mégis érdemes mindkét megközelítést megismerni. Gyenge szójátékkal élve: a klasszikus Class jó bevezetés a tömörebb leírású Hook-ok megértéséhez.

A harmadik rész a Hook-okkal foglalkozik, amelyek a React 16.8-as verziójával kerültek bevezetésre. Fontos megjegyezni, hogy a Class típusú komponensek az újabb verziókban is megmaradtak.

A könyv utolsó fejezetében gyakorló feladatokat találunk, amelyek segítenek elmélyíteni és alkalmazni a korábban megszerzett ismereteket.

2. Kezdetek

2.1. A React célja

A React célja, hogy a weboldalak a lehető legtakarékosabban jöjjenek létre. Ezt két módszerrel éri el: az első, hogy az oldalak elemeit a fejlesztő a lehető legtöbb komponensre bontja szét, amelyek aztán újrahasznosíthatók; a második, hogy a komponensek közül a kódfeldolgozó csak azokat hozza létre, amelyek az adott időben és helyen valóban szükségesek, így feleslegesen nem töltődnek be oldal-elemek. Kicsit kézzelfoghatóbban: egyszerű JavaScript kódolásban, ha nem akartuk, hogy egy gomb (button) elem látszódjon egy adott feltétel mellett, akkor valamilyen kóddal láthatatlanná tettük, de az elem ettől még létezett a weboldalon, csak nem volt látható. A React ezzel szemben a gombot nem hozza létre, nem tölti be, ha nem szükséges, amíg az adott feltétel fennáll. Ezt a React úgy éri el, hogy először egy virtuális DOM-ot hoz létre a memóriájában, és nem közvetlenül hat a böngésző DOM-jára, csak a szükséges módosításokkal. A React a weboldalon bekövetkező változások esetén kizárólag azokat az elemeket módosítja, amelyek szükségesek, nem pedig az egész weboldal újratöltését kezdeményezi.

2.2. A fejlesztési környezet

Kétféle megoldás használható. Az egyszerűbb módszer a tananyag végéig megfelelő, amíg nem kerül sor több fájl kezelésére; rövidebb kódok esetén jól alkalmazható, amikor mindent egyetlen HTML fájlban írunk. A React kódokat a script elemek, címkék (tag-ek) közé helyezzük egy kódszerkesztő segítségével, és a futás eredményét a kódszer-

kesztőhöz kapcsolt localhost-on (helyi szerveren) látjuk. A szükséges kiegészítések a második eszköz bemutatása után következnek.

A másik megoldás egy projekt indítása, amely több fájlt hoz létre egy könyvtárban (alkönyvtárakkal együtt). Ezek egy része automatikusan generálódik a létrehozás során (a projekt kezeléséhez), míg más fájlokat a saját kódjaink megírására használunk. A projekt létrehozásához szükség van az npm és a Node.js előzetes telepítésére, ezek megléte alapfeltétel. Ebben az esetben a projekt létrehozása és elindítása a parancssorból (terminálból) történik, amely egyúttal elindítja a localhost-ot is. A kódok és fájlok szerkesztése ezután egy kódszerkesztőben könnyebb, mivel az a projekt fájljait faszerkezetben is megjeleníti. Nem tanulás, hanem munka során a projektmódszer használatos. A részletek szintén az alábbiakban következnek.

2.2.1. Közvetlen HTML fájl használata

A közvetlenül HTML fájlba írandó React kódok előtt a következőket tesszük:

A head részbe három scriptet illesztünk be, melyből az első kettő a React könyvtárakat hozzák, a harmadik a JSX nyelvhez szükséges Babel fordítót. Ezek a CDN-ek (Content Delivery Network – tartalomszétosztó netes hálózat) a cdnjs.com vagy az unpkg.com webhelyekről jönnek, mint források, az alábbi példában mindjárt látszik. A JSX-ről külön lesz majd szó, a lényege, hogy ezzel a szintaktikai kiegészítéssel React-ben írhatunk HTML-t.

A React kódunkat a body részbe illesztett script tag-ek közé írjuk majd. Itt a nyitó script tag-ben meg kell adni a script típusát, mely most: text/babel.

Ezek alapján, így néz ki a kezdeti fájlunk a React 18-as verziójának behívásával:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>

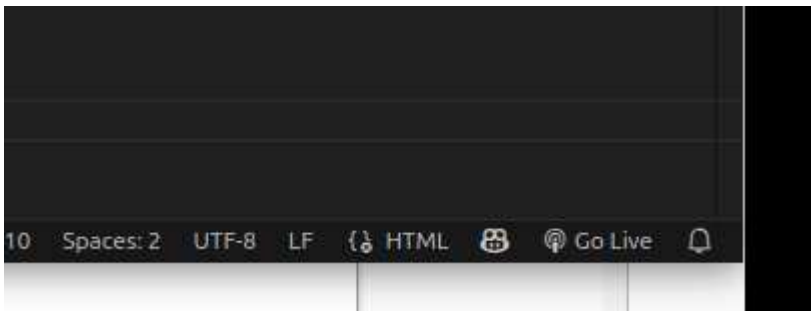
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>

<script src="https://unpkg.com/@babel/stand-alone/babel.min.js"></script>
</head>
<body>
  <div id="root">
    </div>

  <script type="text/babel">

</script>
</body>
</html>
```

A localhost indítása Visual Studio Code szerkesztő esetén a jobb alsó sarokban található: „Go Live” gomb megnyomásával történhet.



2.2.2. Projekt használata a create-react-app parancs után

Projekt létrehozása a parancssorból az `npx create-react-app` paranccsal történik.

Ehhez azonban előtte installálni kell ezt a létrehozó eszközt az `npm`-ben. Vagyis: a `Node.js` telepítésével megkapjuk a kezelő `npm`-et is, azonban a projekt létrehozó eszközt is telepítenünk kell az első `React` projekt előtt egy ehhez hasonló paranccsal:

```
npm install -g create-react
```

A konkrét parancs a `node.js` verziótól (operációs rendszertől) és a választott eszköztől függ. (Van például eszköz, mely a `React` mellett a hasonló `Vue` framework-öt is kezeli.)

Tehát fentiek megléte után írhatjuk a `npx create-react-app` parancsot, mégpedig abban a könyvtárban, ahol a `React` projektünket futtatni szeretnénk. Azaz a parancssorban belépünk a könyvtárunkba, majd egyszerűen a parancs után adjuk meg a projektünk nevét, célszerűen utalva arra is, hogy egy `React` applikációt hozunk létre. Így a `create-react-app` létrehoz egy alkönyvtárat a projektünk nevével és a szükséges fájlokkal. Példa:

```
npx create-react-app elso-react-app
```

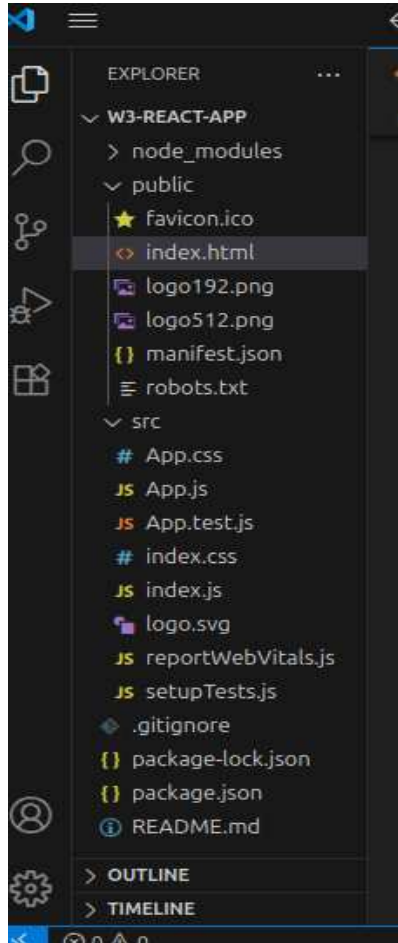
A projekt futtatása pedig a parancssorból a következőképpen lehetséges: Először elmegyünk a projektünk könyvtárába, majd egyszerűen az `npm start` parancs indítja a szerkeszthető és futtatható projektet, továbbá megnyitja a helyi szerveret (localhostot, általában a 3000-es porton) a böngészőben.

```
cd elso-react-app  
npm start
```

A képernyőn, a böngészőben megjelenik a React logója és tanulásra, szerkesztésre buzdító felirat.

Mivel ezek a kezdő elemek zavaróak lehetnek, az alábbiak szerint ezeket töröljük, csakúgy, mint a kezdés által a tanulásra kínált felesleges fájlokat. Ehhez és a továbbiakhoz kódszerkesztő használata indokolt, a könyv a Visual Code Studio-ból mutatja az eredményeket.

A szerkesztőben látjuk, hogy milyen könyvtárakat és fájlokat hozott létre a create-react-app a projektnkhöz, nagyjából (eszköztől, verziótól függően) úgy, ahogy a csatolt kép is mutatja.



A számunkra most lényeges könyvtárak a public és az src. A többi fájl és könyvtár többnyire a React futását biztosítja a háttérből. Ezekkel most nem kell törődnünk.

A public könyvtár tartalmazza a megjelenő fájlokat, mindenekeelőtt az index.html fájlt, ezt látjuk majd futás után