

Kreiranje React aplikacije

U prethodnoj lekciji ste, kroz priču o ručnoj integraciji React biblioteke, imali prilike da se upoznate sa nekim osnovnim postulatima korišćenja takve biblioteke. Ipak, kao što ste mogli da vidite, ručna integracija je vrlo nepraktična, a uz to i vrlo neefikasna kada su u pitanju performanse aplikacije, zbog činjenice da se JSX kod prevodi unutar klijentskog web pregledača. Stoga je Facebook razvio poseban alat koji je specijalno namenjen kreiranju React aplikacija. Reč je o alatu koji znatno ubrzava obavljanje nekih uobičajenih operacija prilikom razvoja React aplikacija. O takvom alatu i načinima za njegovo korišćenje biće reči u ovoj lekciji.

Create React App

Create React App je alat koji je kreirala kompanija Facebook, kako bi olakšala razvoj React aplikacija. Reč je zapravo o jednom npm paketu koji omogućava sledeće:

- kreiranje React projekta i projektne strukture,
- laku izgradnju projekta, što podrazumeva i transformisanje JSX koda,
- pokretanje React aplikacija na živom, razvojnom, lokalnom Node.js serveru.

S obzirom na to da smo se mi u prethodnim lekcijama već upoznali sa Vue bibliotekom, nije teško zaključiti da je u React svetu, *Create React App* isto što i Vue CLI za Vue.

Korišćenje Create React App alata

Preduslov za korišćenje *Create React App* alata jeste postojanje izvršnog okruženja Node.js i npm ili yarn menadžera paketa. Mi ćemo u nastavku koristiti npm menadžer paketa.

Preporučeni način za korišćenje *Create React App* alata jeste njegovo direktno pokretanje, bez prethodnog instaliranja kao globalnog paketa, što je razlika u odnosu na Vue CLI. Direktno izvršavanje npm paketa moguće je postići korišćenjem jednog posebnog alata koji je sastavni deo npm-a. Reč je o alatu npx.

npx

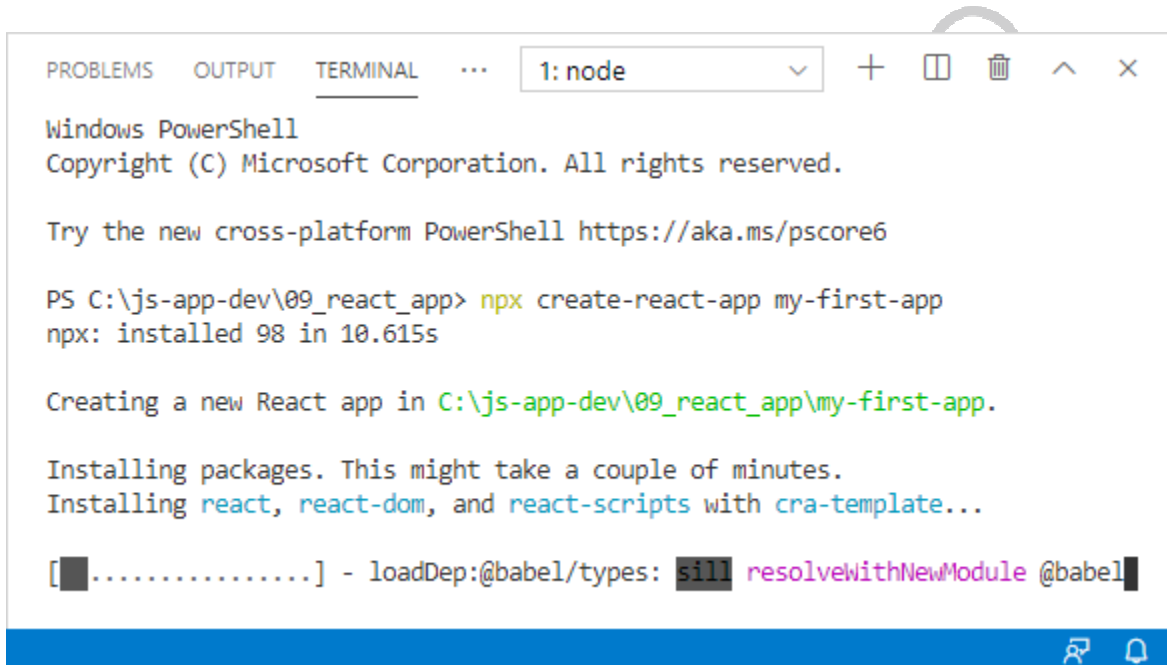
npx je alat koji je sastavni deo npm ekosistema. npm i npx čine osnovni skup klijentskih alata koji su na raspolaganju korisniku za interakciju sa npm repozitorijumom i njegovim paketima. Alat npm koristi se za instalaciju npm paketa, dok je alat npx namenjen direktnom izvršavanju takvih paketa. Oba takva alata postoje kao zasebni izvršni fajlovi koje je moguće pronaći na instalacionoj putanji Node.js-a.

npx alat je moguće koristiti za pokretanje globalno instaliranih paketa, ali i za izvršavanje paketa koji uopšte nisu instalirani na lokalnom kompjuteru. Kada se korišćenjem npx alata uputi komanda za pokretanje nekog paketa koji nije instaliran, npm u pozadini obavlja preuzimanje i pokretanje takvog paketa.

Alat **npx** posebno je koristan za izvršavanje jednokratnih komandi. Upravo takva je i komanda **create-react-app**, kojom se obavlja generisanje novog React projekta. Stoga je za kreiranje novog React projekta dovoljno uputiti sledeću komandu korišćenjem konzole ili terminala:

```
npx create-react-app my-first-app
```

Unutar ovakve komande, `my-first-app` se odnosi na naziv React projekta koji će biti kreiran. Upućivanje komande za kreiranje novog React projekta, uz korišćenje Visual Studio Code Terminala, izgleda kao na slici 9.1.



```
PROBLEMS OUTPUT TERMINAL ... 1: node
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\js-app-dev\09_react_app> npx create-react-app my-first-app
npx: installed 98 in 10.615s

Creating a new React app in C:\js-app-dev\09_react_app\my-first-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[.....] - loadDep:@babel/types: sill resolveWithNewModule @babel
```

Slika 9.1. Proces kreiranja novog React projekta korišćenjem `create-react-app` alata

U zavisnosti od brzine internet konekcije, kreiranje React projekta može potrajati i do nekoliko minuta. Nakon završetka kreiranja projekta, unutar foldera u kome ste bili pozicionirani prilikom pokretanja `create-react-app` alata biće kreiran novi folder sa kompletnom projektnom strukturom, unutar koje će automatski biti uključeni i konfigurisani neki vrlo korisni alati koji se koriste tokom razvoja.

Napomena

Možda se pitate zbog čega se novi React projekat kreira na ovaj način, a ne globalnim instaliranjem `create-react-app` paketa i njegovim pokretanjem. React tim se odlučio za direktno izvršavanje `create-react-app` alata, bez njegovog prethodnog instaliranja, kako bi osigurao da se prilikom svakog pokretanja komande za generisanje projekta koristi najnovija verzija `create-react-app` paketa, s obzirom na to da se korišćenjem alata `npx` svaki put iznova obavlja preuzimanje paketa, ukoliko je novija verzija dostupna.

Pitanje

Ono što je Vue CLI u svetu Vue programiranja, to je u React svetu:

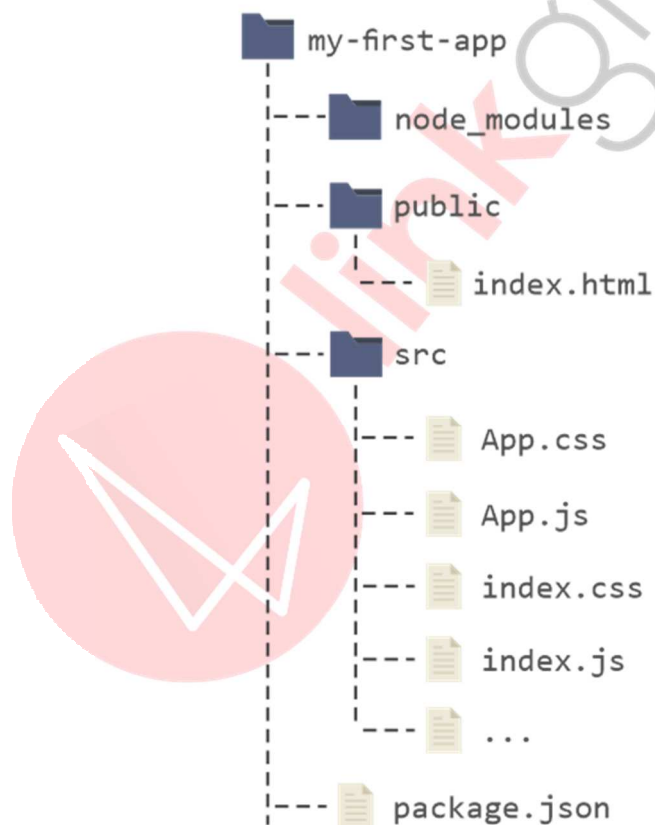
- a) **Create React App**
- b) Create FB App
- c) Create ReactJS App
- d) Create React Proj

Objašnjenje:

U React svetu, Create React App je isto što i Vue CLI za Vue.

Struktura React projekta

Nakon kreiranja React projekta korišćenjem alata `create-react-app`, dobija se projektna struktura ilustrovana slikom 9.2.



Slika 9.2. Projektna struktura React aplikacije

Uloga različitih fajlova i foldera koji se kreiraju tokom generisanja projekta je sledeća:

- `node_modules` – folder za smeštanje različitih npm paketa; unutar ovog foldera smeštaju se različite projektne zavisnosti koje se koriste tokom razvoja (webpack, babel, eslint...),
- `public` – folder za smeštanje statičkih fajlova koji će prilikom izgradnje da budu korišćeni u svom izvornom obliku,
 - `index.html` – fajl sa glavnom HTML strukturom aplikacije, odnosno koreni šablon aplikacije; unutar ovog fajla nalazi se `div` element sa id-jem `root`; unutar takvog elementa se tokom izgradnje projekta umeće kompletna prezentacija React aplikacije,
- `src` – folder sa izvornim fajlovima projekta, odnosno fajlovima koji se koriste tokom razvoja, a na osnovu kojih nastaje finalni, izgrađeni projekat,
 - `App.css` – fajl sa CSS kodom za stilizovanje `App` komponente,
 - `App.js` – fajl sa kodom za kreiranje jedne React komponente sa nazivom `App`,
 - `index.css` – CSS za stilizovanje `index.html` fajla,
 - `index.js` – JavaScript fajl unutar koga se obavlja povezivanje kompletnog projekta,
- `package.json` – JSON fajl za konfigurisanje projekta (npm paketa).

Najznačajniji fajlovi upravo prikazane strukture su `public/index.html` i `src/index.js`. Njihovo postojanje je neophodno kako bi projekat mogao da bude izgrađen. Takođe, ova dva fajla moraju imati prikazane nazive i moraju se naći unutar `public` i `src` foldera, respektivno. Sve ostale fajlove i foldere je moguće obrisati.

Još neke važne činjenice u vezi sa React projektom strukturom su sledeće:

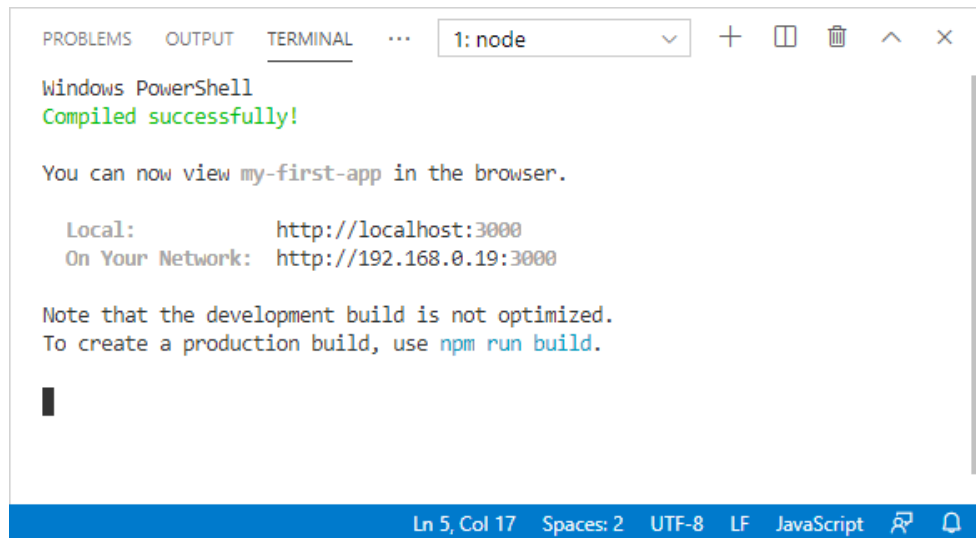
- moguće je kreirati dodatne potfoldere unutar foldera `src`; to se najčešće u praksi i čini, tako što se na primer kreira zaseban folder `components`, unutar koga se izoluju komponente,
- svi JavaScript i CSS fajlovi se moraju smestiti unutar `src` foldera, jer je reč o jedinom folderu čiji sadržaj učestvuje u izgradnji projekta.

Pokretanje React aplikacije

React aplikaciju je moguće pokrenuti posredstvom razvojnog, lokalnog, Node.js servera, korišćenjem sledeće komande:

```
npm start
```

Obratite pažnju na to da je neophodno da budete pozicionirani unutar korenog foldera projekta, prilikom upućivanja ovakve komande. Uspešno okončanje ove komande rezultuje hostovanjem aplikacije na razvojnom serveru, a tom prilikom se unutar konzole dobija poruka kao na slici 9.3.



```
PROBLEMS OUTPUT TERMINAL ... 1: node
Windows PowerShell
Compiled successfully!

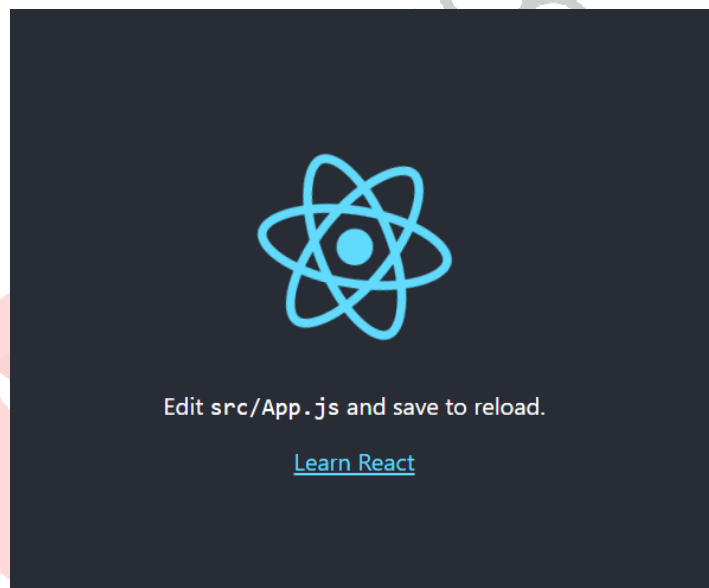
You can now view my-first-app in the browser.

Local:      http://localhost:3000
On Your Network: http://192.168.0.19:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

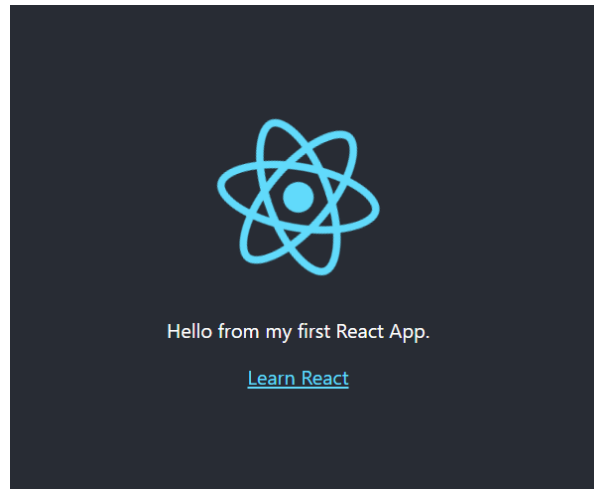
Slika 9.3. Pokretanje React aplikacije na razvojnom serveru

Otvaranjem `http://localhost:3000` adrese, unutar web pregledača se dobija stranica kao na slici 9.4.



Slika 9.4. Izgled React aplikacije koja je automatski kreirana prilikom generisanja projekta

Unutar prikaza koji se dobija ispisan je tekst koji nam govori iz kog izvornog fajla je takav tekst potekao. Stoga, takav tekst možete pronaći unutar JavaScript fajla kojim je kreirana `App` komponenta. Ukoliko promenite sadržaj paragraf elementa unutar koga se nalazi prikazani tekst i sačuvate izmene, moći ćete da vidite da se izmene na izvornim fajlovima projekta, automatski propagiraju i do razvojnog servera (slika 9.5).



Slika 9.5. Izmene na izvornim fajlovima automatski se propagiraju do razvojnog servera

Izgradnja React aplikacije

Izgradnja kompletnog React projekta obavlja se korišćenjem sledeće komande:

```
npm run build
```

Naravno, i za ovu komandu važi identično što i za prethodnu – neophodno je biti pozicioniran unutar korenog foldera React projekta.

```
PS C:\js-app-dev\09_react_app\my-first-app> npm run build

> my-first-app@0.1.0 build C:\js-app-dev\09_react_app\my-first-app
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

 39.39 KB build\static\js\2.4ee2d50b.chunk.js
  778 B   build\static\js\runtime-main.99bb5f77.js
  637 B   build\static\js\main.898a511b.chunk.js
  547 B   build\static\css\main.5f361e03.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  bit.ly/CRA-deploy

PS C:\js-app-dev\09_react_app\my-first-app>
```

Slika 9.6. Proces izgradnje React projekta

Izgradnja React projekta podrazumeva kreiranje foldera **build** unutar koga se smeštaju izgrađeni fajlovi. Procesom izgradnje obavlja se i maksimalna optimizacija projektnih fajlova i resursa. Ona podrazumeva obavljanje minifikacije CSS i JavaScript koda i dodavanje specijalnih kodova (hasheva) unutar naziva fajlova resursa. Takvi kodovi se generišu na osnovu sadržaja fajla. Stoga, kada se sadržaj fajla promeni, menja se i njegov naziv. Na taj način, web pregledači će znati da je reč o novom fajlu i neće koristiti verziju koju možda već poseduju unutar svog keša.

Objavljivanje React aplikacije

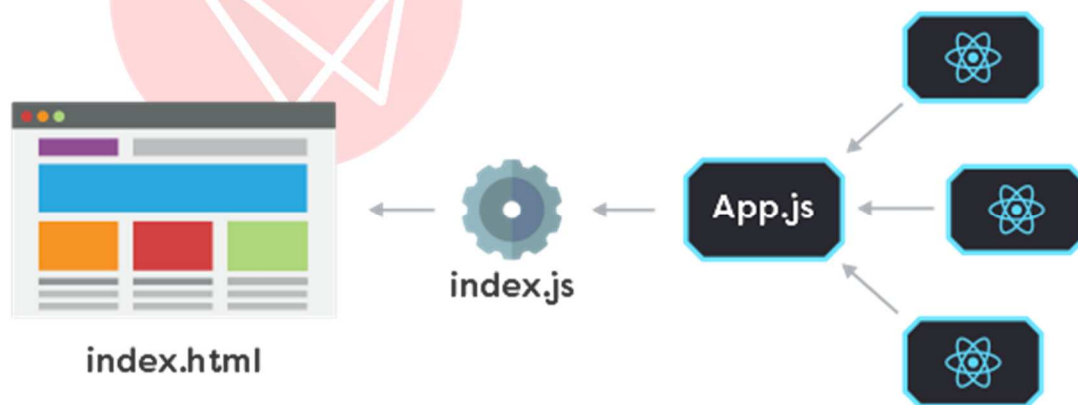
Fajlovi i folderi nastali izgradnjom projekta u prethodnim redovima spremni su za objavljivanje. Stoga je njih dovoljno premestiti na server unutar koga će biti hostovani. Ipak, i ovde važi identično pravilo kao i prilikom objavljivanja Vue aplikacije. Projekat se podrazumevano izgrađuje uz pretpostavku da će biti hostovan na korenoj putanji nekog domena. Ukoliko to nije slučaj, React omogućava da se takav problem prevaziđe na veoma jednostavan način. Unutar `package.json` fajla, koji se inače koristi za opisivanje jednog npm projekta (paketa), dovoljno je dodati sledeći par – ključ i vrednost:

```
"homepage": " . "
```

Na ovaj način se sistemu za izgradnju React projekta govori da putanje do svih resursa i fajlova treba da budu relativne `index.html` fajlu. To će na kraju omogućiti da se React aplikacija hostuje na bilo kojoj putanji unutar servera, odnosno da se bez ikakvih izmena na kodu React aplikacija premešta sa jedne na neku drugu putanju.

Kako funkcioniše kreirani React projekat?

U dosadašnjem toku lekcije ste mogli da vidite da je `create-react-app` alat za nas automatski uspostavio određenu strukturu elemenata unutar projekta. Reč je o strukturi baziranoj na komponentama, koja podrazumeva postojanje jedne korene komponente, unutar koje se eventualno uključuju ostale komponente koje čine grafičko korisničko okruženje. Stoga uprošćena struktura uobičajenih React projekata izgleda kao na slici 9.7.



Slika 9.7. Uobičajena struktura React projekta

Glavna komponenta aplikacije je sadržana unutar `App.js` fajla. Ostale komponente se dodaju unutar takve glavne komponente i na taj način kreiraju strukturu komponenata. Komponentna struktura se unutar statičkog `index.html` fajla uključuje posredstvom logike koja je smeštena unutar `index.js` fajla. Sadržaj takvog fajla izgleda ovako:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Unutar fajla `index.js` možete videti da se korena komponenta aplikacije (`App`) renderuje unutar elementa sa id-jem `root`, koji se nalazi unutar `index.html` fajla. Renderovanje se obavlja korišćenjem već prikazane `ReactDOM.render()` metode. Novina jeste korišćenje `React.StrictMode` elementa. Reč je o jednom specifičnom React elementu, koje ne poseduje vizuelnu reprezentaciju, već se isključivo koristi za pronalazak potencijalnih problema unutar aplikacije. Tako `StrictMode` element omogućava dobijanje poruka i upozorenja o potencijalnim problemima unutar aplikacije, ali samo kada se aplikacija pokreće tokom razvoja. Kada se projekat izgradi i aplikacija objavi, takve poruke se ne dobijaju.

Još jedan segment `index.js` fajla, koji možda ne razumete u potpunosti, tiče se naredbi za importovanje funkcionalnosti iz nekih drugih modula. Naime, modularni dizajn osnova je modernog JavaScript programiranja, pa se tako intenzivno koristi i prilikom kreiranja React aplikacija.

Modularno JavaScript programiranje

Do sada je u nekoliko navrata bilo priče o modularnom JavaScript programiranju, odnosno o podeli JavaScript koda koji pišemo na izolovane celine koje se nazivaju moduli. Na primer, prilikom priče o izvršnom okruženju Node.js, imali ste prilike da vidite da Node.js poznaje pojam modula. Zapravo, moduli su osnovni način za organizovanje izolovanih funkcionalnosti prilikom korišćenja takvog izvršnog okruženja.

Sve do pojave ES6 jezičke specifikacije, JavaScript jezik nije posedovao izvorni mehanizam za kreiranje modula. Takav nedostatak nadomestila je biblioteka CommonJS, koja je JavaScript programerima omogućila modularno programiranje. S obzirom na to da se ES6 pojavio 2015. godine, CommonJS je uvršten unutar izvršnog okruženja Node.js kao podrazumevani sistem za postizanje modularnog programiranja.

React zahvaljujući alatima, odnosno bibliotekama koje se koriste prilikom razvoja (pre svih, zahvaljujući biblioteci webpack), omogućava korišćenje CommonJS, ali i izvornih ES6, JavaScript modula. Preporuka je da se koriste ES6 moduli, a u primeru prve React aplikacije ste mogli da vidite da se oni i podrazumevano koriste unutar koda koji se dobija generisanjem aplikacije korišćenjem `create-react-app` alata.

ES6 moduli funkcionišu po vrlo jednostavnom principu:

- svaki JavaScript fajl je zasebni modul,
- dve osnovne operacije prilikom korišćenja modula su uvoz (engl. *import*) i izvoz (engl. *export*),
- jedan modul može da omogući izvoz proizvoljnog broja entiteta,
- jedan modul može da uveze proizvoljan broj entiteta iz drugih modula.

Eksportovanje elemenata iz jednog modula

Sve promenljive i funkcije definisane unutar jednog modula privatne su i vidljive samo unutar takvog modula. Kako bismo ih učinili vidljivim i izvan modula, neophodno je da ih izvezemo. To se postiže korišćenjem JavaScript naredbe **export**:

```
export var variable1 = 1234;
var variable2 = 56789;
export function sayHello(name) {
  //...
}
```

Prikazani kod ilustruje sadržaj jednog .js fajla. Samim tim, reč je o jednom JavaScript modulu. Unutar takvog modula nalaze se dve promenljive i jedna funkcija. Promenljiva `variable1` i funkcija `sayHello()` obeležene su naredbom `export`. Na taj način su one izvezene iz modula, odnosno, omogućen je njihov uvoz u neke druge module. Ipak, promenljivu `variable2` neće biti moguće uvesti u neki drugi modul, zato što prethodno nije izvezena iz modula u kojem je definisana.

Objedinjeno eksportovanje većeg broja elemenata

Izvoz je moguće obaviti i objedinjeno, na kraju jednog dokumenta:

```
export { variable1, sayHello };
```

Importovanje elemenata iz drugih modula

Kako bi se ovakvi entiteti uvezli unutar nekog drugog modula (fajla), koristi se naredba **import**, na sledeći način:

```
import { variable1, sayHello } from 'app.js';
```

Ovakva `import` naredba funkcionisaće ukoliko je prethodno prikazani kod smešten unutar fajla sa nazivom `app.js`.

Entitete je moguće importovati i pojedinačno:

```
import variable1 from 'app.js';
```

Imenovanje elemenata prilikom importovanja

Prilikom importovanja, elementima koji se uvoze moguće je dati i neko posebno ime, koje će se koristiti u tekućem modulu:

```
import variable1 as Variable from 'app.js';
```

Na ovaj način je uvezen element sa nazivom `variable1`, kojim će se u tekućem dokumentu rukovati korišćenjem naziva `Variable`.

Importovanje svih elemenata odjednom

Moguće je obaviti uvoz svih entiteta koji su izvezeni u nekom modulu, uz korišćenje sledećeg pristupa:

```
import * as App from 'app.js';
```

Karakter zvezdica se odnosi na sve eksportovane elemente jednog modula. Tako je prikazanom naredbom rečeno da će iz fajla `app.js` biti uvezeni svi eksportovani elementi i da će oni biti dostupni korišćenjem naziva `App`. Stoga će biti moguće napisati nešto ovakvo:

```
App.sayHello('Ben');
```

Podrazumevani izvoz i uvoz

Na kraju, postoji mogućnost da se unutar jednog modula jedan element obeleži kao podrazumevani element za izvoz:

```
function sayHello(name) {  
  //...  
}  
  
export default sayHello;
```

Na ovaj način otvara se mogućnost uvoza bez definisanja naziva elementa koji želimo da uvezemo:

```
import Hello from 'app.js';
```

`Hello` se odnosi na naziv koji je podrazumevano izvezenom elementu dodeljen unutar modula u koji se uvozi.

Nakon kratkog upoznavanja sa modularnim sistemom ES6, uvodni deo fajla `index.js` postaje kristalno jasan:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
```

Ovakvim naredbama se obavlja importovanje osnovnih React funkcionalnosti (*React* i *ReactDOM*). Nakon toga obavlja se importovanje fajla za stilizaciju i glavne komponente naše aplikacije (*App*).

webpack importovanje i eksportovanje

Ukoliko izvršite pažljivu analizu svih `import` i `export` direktiva unutar fajlova koji su automatski generisani prilikom kreiranja React projekta, moći ćete da vidite jednu zanimljivu osobinu – takve direktive nisu korišćene samo za uključivanje JavaScript fajlova. Na primer, unutar `index.js` fajla možete videti da je `import` direktiva korišćena i za uključivanje jednog `.css` fajla. Unutar `App.js` fajla možete videti još jednu takvu direktivu, koja služi za uvoz jednog `.svg` fajla. ES6 moduli omogućavaju rad isključivo sa `.js` fajlovima. Pa kako je onda nešto ovakvo moguće?

webpack biblioteka koja se koristi prilikom izgradnje React projekta je krivac za ovakvo ponašanje. Naime, webpack dodatno proširuje mogućnosti ES6 modula, pa tako obezbeđuje da se na identičan način rukuje i ostalim fajlovima. To na kraju omogućava da se korišćenjem `import` direktive obavi uvoz čak i onih fajlova koji nisu `.js`.

Priprema projekta za naredne lekcije

U dosadašnjem izlaganju o biblioteci React ste mogli da vidite da se prilikom kreiranja React projekta korišćenjem `create-react-app` alata, automatski dobija određena projektna struktura sa unapred kreiranim različitim elementima. Ipak, kao što je već rečeno, dva fajla čije je prisustvo obavezno su:

- `public/index.html` i
- `src/index.js`.

Svi ostali fajlovi su fakultativni i moguće ih je obrisati. Stoga ćemo mi sada uprostiti našu projektnu strukturu i ukloniti kod koji nam nije potreban, te na taj način pripremiti projekat za naredne lekcije.

Unutar `src` foldera obrišite sve fajlove osim fajla `index.js`.

Sadržaj `index.js` fajla modifikujte tako da unutar njega ostanu samo dve naredbe za uključivanje dva osnovna React modula:

```
import React from 'react';
import ReactDOM from 'react-dom';
```

Na ovaj način će se dobiti potpuno čist projekat, bez ijedne komponente. To će nam omogućiti da u nastavku ovoga kursa samostalno kreiramo React elemente i komponente.

Rezime

- *Create React App* je alat koji je kreirala kompanija Facebook kako bi olakšala razvoj React aplikacija.
- Preporučeni način za korišćenje *Create React App* alata jeste njegovo direktno pokretanje, bez prethodnog instaliranja kao globalnog paketa.
- Direktno izvršavanje npm paketa moguće je postići korišćenjem alata `npx`.
- *Create React App* unutar npm repozitorijuma ima naziv `create-react-app`.
- Novi React projekat kreira se korišćenjem komande `npx create-react-app app_name`, gde se `app_name` odnosi na naziv projekta koji će biti kreiran.
- *Create React App* automatski uspostavlja projektну strukturu i unutar projekta uključuje brojne dodatne alate, koji se koriste prilikom razvoja i izgradnje projekta.
- Unutar projektne strukture koja se dobija korišćenjem alata *Create React App*, folder `public` koristi se za statičke fajlove, čiji se sadržaj ne menja tokom izgradnje, a folder `src` za smeštanje izvornih fajlova projekta.
- Najznačajniji fajlovi projektne strukture koja se dobija generisanjem projekta korišćenjem alata *Create React App*, su `public/index.html` i `src/index.js`.
- Fajl `public/index.html` predstavlja osnovni šablon aplikacije unutar koga se umeće kompletna prezentacija React aplikacije.
- Fajl `src/index.js` koristi se za povezivanje kompletnog projekta i za iniciranje renderovanja prezentacije React aplikacije.
- React aplikaciju je moguće pokrenuti posredstvom razvojnog, lokalnog, Node.js servera, korišćenjem komande `npm start`.
- Izgradnja kompletnog React projekta obavlja se korišćenjem komande `npm run build`.
- Izgradnja React projekta podrazumeva kreiranje foldera `build`, unutar koga se smeštaju izgrađeni fajlovi, spremni za objavljivanje.
- React omogućava korišćenje CommonJS i ES6 modula; preporuka je da se koriste ES6 moduli.

