***Udemy-REACT-1***

Dobar izbor kazu ljudi

https://react.new to ukucas u browser i preko browsera mozes da kucas react kod (onlice visual code).

REACT KOMPONENTA je jedan delic user interfejsa.

Treba ti Prettier ekstenzija za lakse citanje koda.

import {useState} from "react";

export default function App() {

const [advice, setAdvice] = useState("");

async function getAdvice(){

const res = await fetch('https://api.adviceslip.com/advice');

const data = await res.json();

setAdvice(data.slip.advice);

}

return (

<div>

<h1>{advice}</h1>

<button onClick={getAdvice}> Klikni me!</button>

</div>

);

}

U button mozes da ubacis onClick={getAdvice} koja poziva getAdvice metodu svaki put kada se klikne da dugme.

**STATE KONCEPT** je najosnovniji koncept u React-u. Svaki put kada se nesto menja u user interfejsu mi treba da promenimo state. Svaka promena u user interfejsu updejtuje nesto sto se zove stat.

**STATE** je objekat koji cuva informacije o komponenti koja se moze menjati tokom vremena.

Znaci state je objekat koji ima neki value i neku seter metoda za updejtovanje tog value. Kada se poziva seter metoda kao parametar se prenosi vrednost koja ce biti upisana u value. I tako se pri svakom pozivu upisuje neka vrednost.

**useState** je funkcija koja se importuje iz react biblioteke, ona vraca neki Array koji se destruktuje na [advice, setAddvice]

advice- value odnosno vrednost stat-a

setAdvice-setter funkcija odnosno funkcija za updejtovanje stat-a.

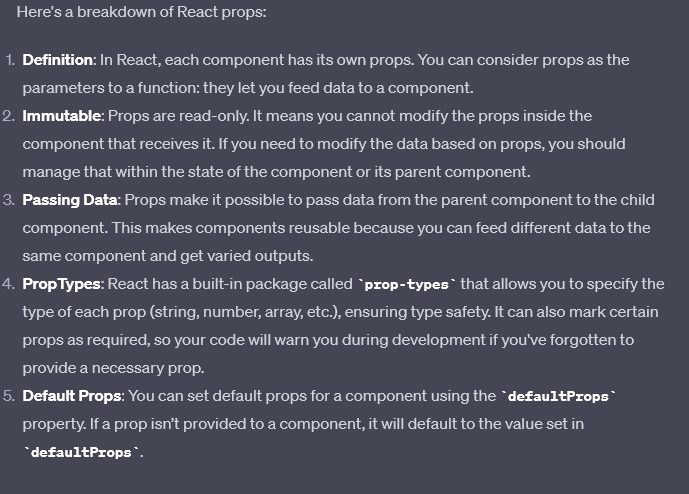
const [count, setCount] = useState(0);

setCount((c => c + 1));

Znaci mozes da ga napravis i kao brojac, gde svaki put kada kliknes se count inkrementira za 1.

useEffect ima dva parametra, prvi je funkcija koju zelimo da se izvrsi odmah na pocetku, a drugi argument je dependencyArray, za pocetak je prazni string. To je neki hook.

Kroz REACT cilj je da korisnicki interfejs podelis na **KOMPONENTE**. A svaka komponenta je neka funkcija. Komponente su building blocks svake react aplikacije. Omogucavaju da enkapsuliras logiku, ponasalje, stilove unutar delica koji se mogu eksportovati i koristiti veci broj puta po potrebi. Ima klasne komponente i funkcijske komponente.

REACT PROPS, props (property) su nesto kao parametri za funckiju. Koriste se za prenos vrednosti sa roditeljske na deciju komponentu.

<Message count={count}/>

Komponenti Message kao parametar preko props-a prenosis vrednost count.

function Message(props){

return(

<p>

Kliknuo si <strong>{props.count}</strong> puta

</p>

);

}

Komponenta Message preko props.count moze da pristupi toj vrednosti. Vrednost je read-only, dete ne moze da je menja.

Linkovi neki:

* <https://vitejs.dev/guide/>
* <https://react.dev/>
* <https://create-react-app.dev/>

***SECTION 3:***

Zasto JS frejmvorci uopste postoje kao sto je React? Zasto se za sve ne koristi vanila js?

Znaci u pocetku klijent bi slao zahtev serveru za nekom web stranicom, server bi izrenderovao i kao odgovor slao klijentu html, css, js fajlove i browser klijenta bi samo interpetirao kod i ucitao stranicu. A sa pojavom single-page aplikacija, renderovanje se vrsi na klijentskoj strani sada podaci stizu preko API-a. To omogucava da se kompletna web aplikacija izvrsava samo na jednom html stranici.

Ima problema kod razvoja single page aplikacija sa vanila javascriptom. Front end deo treba da prihvati neke podatke i da ih prikaze korisniku na ekranu. Interfejs treba da prikazuje uvek trenutno stanje sa podacima. Potrebno je da korisnik uvek ima najnoviju verziju podataka.

-> PIECE OF DATA == PIECE OF STATE

Bez odredjenih frejmworka je skoro pa nemoguce da se ogromne kolicine podataka sinhronizuju sa super-kompleksnim korisnickim interfejsom. Jednsotavno vanila js bi bio davao jako nepregledan i tezak kod za razumevanje.

Zasto front-end frejmworci uopste postoje?

Postoje kako bi se olaksalo sinhronizacija korisnickih in terfejsa sa podacima sa kojima web aplikacija radi. Fremworci olaksavaju posao programerima. Razliciti frejmworci ima razlicite pristupe resenju ali svi imaju zajednicki cilj a to je da podaci i UI (User interface) budu sinhronizovani. Oni takodje pruzaju pravilan nacin za pisanje i strukturu koda kako bi kod bio sto laksi za razumevanje. Moderni web dev se vrti oko js frejmworka.

Znaci kod React sve je odradjeno kroz js dok kod vanila js se idalje koriste html elementi.

**RAZLIKA JE** kad se kod vanila js vrsi promena vrednosti neke promenljiva ona nece da se odmah odrazi na korisnicki interfejs vec je neophodno da programer manuelno onese novu vrednost u interfejs, dok se sa react-om koriste states gde svaka promena nad statom automatski vrsi updejt korisnickog interfejsa. Znaci velika prednost REACT u odnosu na vanila js jesu **STATES**. Znaci veoma je bitno da vrednosti koje se prikazuju na UI budu sinhronizovane sa DATA. React smanjuje kolicinu koda u web aplikacijama.

**STA JE REACT?**

React je javascript biblioteka za izgradnju user interfejsa. Bazira se na komponentama i state-u. Kreiran od strane fejsbuka i veoma je popularan. Reakt je skup komponenti. Komponente su gradivni blokovi UI u react-u. Komponenta moze biti neko dugme, heder, div sa drugim komponentama i tako dalje. React uzima komponente i kreira web app na osnovu njih.

**JSX** je deklarativna sintaksa. Koristi se za opis kako komponente izgledaju na osnovu trenutnog stanja. React je apstrakcija DOM-a. Nikad se ne radi sa DOM direktno kao u vanilla js. JSX je sintaksa koja kombonuje html, css, js i omogucava struktuiranje komponenti.

Glavni cilj React-a je da UI bude u skladu sa DATA. Na osnovu trenutnog stanja REACT renderuje UI, onda se stanje moze promeniti nakon klika na dugme, svaki put kad se stanje promeni react automatski re-renderuje UI kako bi se reflektovale promene na korisnicki interfejs. Znaci nakon svake promene STATE, react re-renderuje UI.

**REACT REACTS TO STATE CHANGES BY RE-RENDERING THE UI.**

Zbog toga se React zove React. Da li je React biblioteka ili framework. On je ustvari samo biblioteka. Jer biblioteka je alat a framework je skup alata. A React je samo jedan alat za izgradnju SPA aplikacija. Da bi se izgradila aplikacija potrebni su jos neki dodatni alati kao sto je Next.js.

React je popularan i ostace popularan jer veliki broj kompanija, cak i vecih, su prihvatile react odavno. Sve vise manje kompanije koriste React. Zato nastaje velika potreba za React poslovima i to su dobro placeni poslovi. Sada postoji i React community. Uvek ce biti tutoriala, pitanja i pomoci oko react dev-a.

Takodje veliki broj 3-rd party alata mozes da inkludujes u svoju react aplikaciju.

React je kompletno transformisao moderni web dev. React je dobar za renderovanje komponenti na UI na osnovu trenutnog stanja.Takodje je jako dobar i odrzava sinhronizaciju DATA sa UI jer reaktuje svaki put kad se izvrsi promena nad state rerenderuje se UI. UI prati svaku promenu state-a.

Znaci treba imas Visual Studio Code i Node.js novu verziju.

**ESLint** ekstenzija obavezno (MNOGO JAKO, AUTOMATSKI TI FORMATIRA KOD)

**Prettier** jer lepo formatira kod

**Material Icon Theme** isto ekstenzija koju mora skines

Settings -> default format -> Editor: Default Formatter -> Prettier

Settings -> format on save -> check

Settings -> eslint run -> run -> onSave //eslint proverava kod svaki put kad ga zapamtimo

Treba instaliras i najnoviju verziju node // node –v da proveris koja je verzija

Treba neki **snippet** da napravis, da prekopiras nesto

***PURE REACT***

**HTML TEMPLATE** u VsCode: Ukucas ! + Enter

root Div ima className root.

Ukljucis dve biblioteke, prva je kao neki interfejs a druga je rendering layer koja renderuje react komponente u DOM.

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

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

Iz ovih biblioteka se uzima React objekat i metode koje se pozivaju nad njim.

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(React.createElement(App)); //Iscrtava komponentu App unutar root-a, ovako se ne radi nego se koristi JSX sintaksa, nego bajca samo da pokaze da moze.

file->preferences->settings-> Type **editor.copyWithSyntaxHighlighting->** untick the box labelled **Controls whether syntax //Kad kopiras iz VS da nemas crnu ppozadinu.**

**useEffect je hook koji se aktivira uvek kada dodje do necega do cega treba da dodje. U ovom primeru useEffect se aktivira svaki put kada prodje 1000ms i tada poziva setTime koja menja state time.**

**const [time, setTime] = React.useState(new Date().toLocaleTimeString());**

**React.useEffect(function () {**

**setInterval(function () {**

**setTime(new Date().toLocaleTimeString());**

**}, 1000);**

**}, []);**

**->Escape Hatches je dobro poglavlje u react.dev dokumentaciji.**

**->DOBAR DEVELOPER UVEK ZNA GDE DA NADJE INFORMACIJE KOJE SU MU POTREBNE a officialna dokumentacija je cesto najbolji izbor.**

**Za kreiranje React App imas dve opcije:**

**1. ) CREATE-REACT-APP (kompetan “starter kit” za react app, sve je vec configurisano (ESLint, Prettier, Jest, etc...) problem je sto je napravljeno davno i koristi spore i outdated tehnologije, narocito webpack budles. Trenutno preporuka je da se ovo vise ne koristi za projekte jer nema vise podrske za to, jedino za tutoriale da se koristi. NE KORISTI ZA PRAVE APLIKACIJE.**

**2. ) VITE je perfektan izbot za izgradnju React app. Moderan build tool koji sadrzi template za podesavanje React aplikacije. Problem je sto moras manuelno da podesis Prettier, EsLint, Jester i ostale zebancije. VITE je ekstremno brz u refresovanjus stranice kada se desi neka promena (hot module replacement). Bundiling je isto jako brzo. Stranica se instantno refresuje svaki put kada sejvujemo kod.**

**React ima frejmworke kao sto su Next.js i Remix. Da bi naucio Next.js treba prvo React da znas.**

**ZA KREIRANJE PREKO PRVOG NACINA KUCAS ->**

**npx create-react-app@5 imeAplikacije //@5 oznacava verziju 5 react-a, da bi radilo treba i ovo iskucas-**npm install -g npm@latest

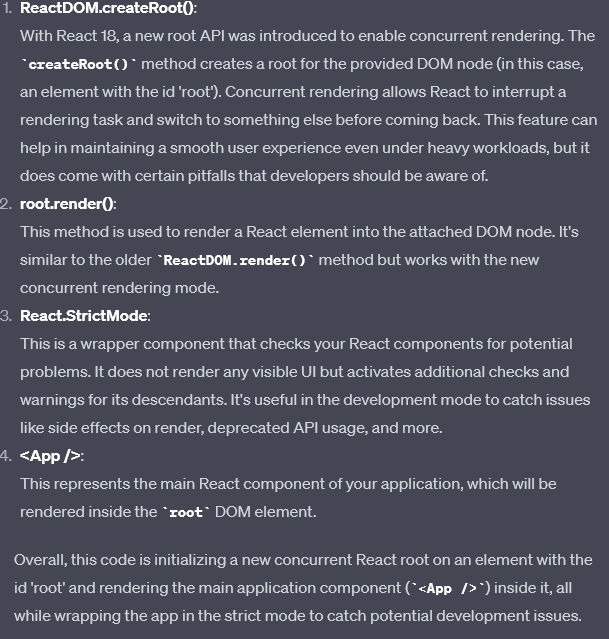
npm cache clean --force

npm rm -rf node\_modules && rm package-lock.json

npm install

npm install create-react-app

npx create-react-app myapp

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<React.StrictMode>**

**<App />**

**</React.StrictMode>**

**);**

**//KREIRA I RENDERUJE ROOT**

U reactu nije bitno gde se sta nalazi, koji fajlovi u kojim folderima... sve je proizvoljno.

Da bi pokrenuo React app treba da ukucas: npm run start (pokrece skriptu “start” koja se nalazi u package.json)

***SECTION 4 js obnavljanje***

React je javascript biblioteka. U section4 ce da obnove ono najbitnije za javascript kao sto su:

Destructing, spread operator, template literals, ternaries, promises, Async/Await, Map,Filter,Reduce,Sort

Sta su immutable arrays? Map, Filter,Reduce,Sort jel to?

* Treba skines ekstenziju **Quokka** (omogucava da se skripta izvrsava direktno iz Vs Coda) da pokrenes treba Manage -> Command pallete -> Quokka start on Curent file

--Ovde obnavljaju JavaScript

ObjectDestructuring: const {ime, prezime,pages,genres} = book;

-----------------------------------------------------------------------------------------------------------------------------------------

***SECTION 5***

Kako se kreiraju i ponovno koriste komponente koriscenjem JSX sintakse.

Rendering lists

Conditional rendering

Pravi se pizza-menu. Prvi fajl uvek mora da ima naziv index.js. Sve iz foldera src izbrisi i napravi novi fajl.

//React v18

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

Strict mode ce renderuje dva puta kako bi pronasao potencijalne greske. Po konvenciji bi trebalo da se pise ali nije obavezan.