Predavanje React Router i Class Komponente

Proširen je primjer klijentske aplikacije sa prethodnog predavanja. Dodane su još dvije class komponente Details i Carousel. https://reactjs.org/docs/react-component.html

1. **Details.js**

   Ako usporedimo function komponente i class komponente, ono što je Hook useffect() u function komponentama, u class komponentama je to componentDidMount() lifecycle metoda. Kako samo ime kaže, ova metoda se poziva odmah nakon prvog renderanja elemenata (komponente) na stranicu. Operacije koje se u nju stavljaju, uglavnom su dohvaćanje podataka sa Web API-a. Osim toga, componentDidMount() je idealno mjesto za mijenjanje *stanja*, odnosno poziva metode za update stanja *setState()*.

   U našem primjeru bi bilo idealno kada bi se preko url-a mogli dobiti podaci o detaljima filma. Primjerice za žanr komedija i id 1. Nažalost nemamo pravi backend, nego mockane podatke na vanjskom servisu. Zbog toga je izvedena svojevrsna gimnastika sa mockanim podacima i prenošenjem propertija u children-e. Ovo je nešto što će se izbjeći za predavanje/dva kada budemo radili backend tehnologiju i izradu REST API-a.

   Details komponenta ima dva property-a *id* i *genre* (gore su spomenuti razlozi). Details komponenta ima nekoliko stanja: name, images i loading.

   U Details.js datoteci se koristi komponenta Carousel te joj se kao property šalje image (podaci o slikama koje opisuju pojedini film - u našem slučaju šalju se url-ovi na slike).

2. **Carousel.js**

- predstavlja komponentu koja će renderirati galeriju slika za odabrani film
- ima jedno stanje *active* (states https://reactjs.org/docs/faq-state.html):

active: predstavlja indeks slike iz galerije na koju se kliknulo-preciznije predstavlja aktivnu sliku

Što se tiče stanja active - budući da se njegova vrijednost mijenja u DOM-u, a sve što iz DOM-a dođe je string, za potrebe našeg primjera treba ga castati u integer. Za to je korišten unarni operator plus.

Unarni operator `+ active: +event.target.dataset.index`
https://scotch.io/tutorials/javascript-unary-operators-simple-and-useful#toc-unary-plus-

Još je jedna stavka koju je bitno istaknuti u ovom primjeru, a to je funkcija handleIndexClick. Napisana je u dvije varijante. Kao function declaration i kao arrow function. Niže pogledajte obješnjenje obje varijante.

Function Declaration. U tijelu funkcije, ključna riječ *this* ima svoj kontekst. Taj kontekst nije ono što u ovom slučaju želite/mislite da je. Vjerojatno očekujete da je to *this* koji pripada objektu klase Carousel. Ali nažalost nije. Funkcije unutar klase za koje možemo garantirati da im *this* pripada instanci su lifecycle metode kao što su constructor, render ili prethodno spomenuta funkcija componentDidMount jer ih React poziva nad ispravnom instancom komponente. U metodama koje poziva DOM, kao na primjer funkcije koje su postavljene na onClick i slično, DOM ih poziva bez objekta, stoga je u tom slučaju this undefined. Problematičan dio koda je *this.setState()* u metodi *handleIndexClick()*. Ukoliko želite osigurati da *this* pripada instanci, a ne nečem drugom, u jednoj od lifecycle metoda ćete dodati sljedeću liniju: *this.handleIndexClick = this.handleIndexClick.bind(this)*. Što se točno događa u ovoj liniji: Povrat funkcije bind je kopija funkcije *handleIndexClick* za koju je vezan onaj *this* koji je proslijeđen kao argument u funkciju *bind()*. Od ove linije pa nadalje smo sigurni da nam this unutar funkcije *handleIndexClick* pripada instanci.

Arrow Function. U drugom načinu implementacije funkcije *handleIndexClick*() koristimo arrow funkciju. Arrow funkcija nasljedjuje kontekst za ključnu riječ this, te smo s njom sigurni u da se *this* iz *this.setState()* odnosi na *this* iz instance klase Carousel. This unutar arrow funkcije je lexicaly scoped (vezan je uz mjesto deklaracije, ne uz mjesto/nacin poziva). Kako ne biste upadali u probleme vezane uz prethodni primjer, preporuča se koristiti arrow funkcije kada su one callback funkcije vezane uz neki event iz DOM-a.

3. App.js

U App.js datoteci Primjećujemo korištenje komponente Router (koja dolazi iz paketa @reach/router). Unutar komponente Router nalaze se komponetne SearchParams i Details. Svaka od njih ima path property na sebi. Path property predstavlja rutu. React Router automatski renderira komponentu čiji se path property poklapa sa url-om koji je u address baru. Postavlja se sljedeće pitanje: na koji se način url u address baru mijenja? Tu dolazi u igru komponenta Link koja je također dio @react/router paketa. U našem primjeru pogledajte datoteku Movie.js. Link komponenta ima atribut to te se vrijednost tog atributa upisuje u address bar, u trenutku kada se klikne na dijete komponente Link (u našem primjeru klik na određeni film).

Ukratko: Klik na film ubacuje vrijednost to atributa iz komponente Link u address bar (napravljen je AJAX-oidni request). App komponenta se iznova renderira. Unutar Router komponente nalaze se SearchParams i Details komponente. Renderirati će se ona čiji se path atribut poklapa sa adresom u address baru (traži se najspecifičinja ruta).

Link komponenta nam omogućuje SPA navigaciju (Single Page App) – tj. Navigaciju kroz različite stranice web mjesta bez ponovnog učitavanja (reload) cijele stranice.

ReactRouter-više pročitati na https://reactrouter.com/.