JUNIORDev
DIGITALNA DALMACIJA

Stanja komponente

Na prethodnoj radionici smo se već upoznali sa ***useState*** *hook*-om. Ovaj *hook* nam služi kako bi komponentama dodali mogućnost pamćenja neke vrijednosti (varijabla stanja). Promjena varijable stanja uzrokuje ponovno renderiranje komponente kako bi se prikazala nova vrijednost.

Jedna od značajki osvježavanja stanja u Reactu je da se ono zakazuje i odvija u grupama (*batches*). Vidjeli smo primjer u kojem smo uvećavali dva brojača unutar iste funkcije i kako će React istovremeno osvježiti i prikazati oba stanja. Sada ćemo još malo detaljnije pogledati način na koji React osvježava stanja i kako pisati funkcije za osvježavanje.

Vratimo se na jednostavni primjer sa brojačem. Unutar komponente definiramo stanje pomoću *useState*. Rezultat destrukturiramo u varijablu stanja `brojac` i funkciju za promjenu stanja `postaviBrojac`. Napisati ćemo i pomoćnu funkciju koju ćemo povezati sa *onClick* atributom:

App.jsx

```jsx
import { useState } from 'react'
import './App.css'

function App() {
  const [brojac, postaviBrojac] = useState(0)

  function uvecajBrojac(){
    postaviBrojac(brojac + 1)
  }

  return (
    <div>
      <p>Broj: {brojac}</p>
      <button onClick={uvecajBrojac}>+</button>
    </div>
  )
```

```
}

export default App
```

Možemo to probati i na interaktivnom primjeru:

Broj: 0
+

Uzastopno osvježavanje

Ovo nam je sve poznato i od prije. Sada ćemo napraviti malu promjenu i u našoj pomoćnoj funkciji dva puta pozvati funkciju za uvećanje brojača. Ponovno imamo ispod interaktivni primjer, a prije nego ga testiramo pokušajte predvidjeti koji će biti rezultat nakon prvog pritiska na "+".

App.jsx

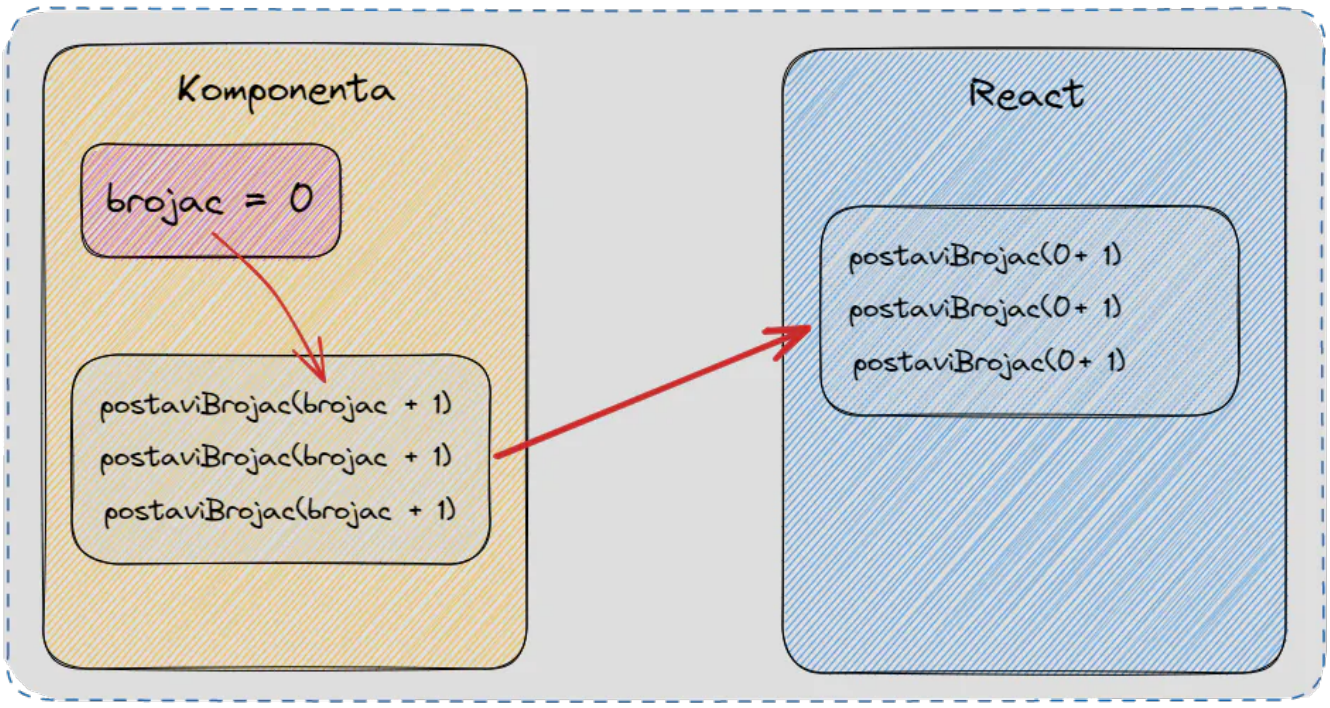
```jsx
import { useState } from 'react'
import './App.css'

function App() {
  const [brojac, postaviBrojac] = useState(0)

  function uvecajBrojac(){
    postaviBrojac(brojac + 1)
    postaviBrojac(brojac + 1)
    postaviBrojac(brojac + 1)
  }

  return (
    <div>
      <p>Broj: {brojac}</p>
      <button onClick={uvecajBrojac}>+</button>
    </div>
  )
}

export default App
```

Broj: 0
+

Problem je u već spomenutom načinu na koji React renderira komponente i osvježava stanje. Svako stanje komponente možemo zamisliti kao jednu snimku (***snapshot***) koja je prikazana (renderirana) na ekranu. Ta snimka je statična, sa fiksnim vrijednostima, sve dok ne zatražimo od Reacta da promjeni vrijednost neke varijable stanja i na zaslonu prikaže novu snimku sa osvježenim vrijednostima.

Pozivom funkcije `postaviBrojac(brojac + 1)` računamo zbroj trenutne vrijednosti brojača (0) i konstante 1 i zapravo pozivamo funkciju `postaviBrojac(1)` . U tom trenutku smo samo zakazali promjenu varijable stanja, nismo je promijenili. Drugi uzastopni poziv funkcije `postaviBrojac(brojac + 1)` i dalje za izračun koristi vrijednost brojača iz trenutnog *snapshot*-a, a ne vrijednost koju će stanje poprimiti nakon idućeg renderiranja.

U gornjem primjeru smo zapravo tri puta pozvali funkciju `postaviBrojac(1)` i zbog toga se nakon prvog klika brojač postavi na `1` , a ne na `3` . Isto tako će svaki idući pritisak na "+" uvećati brojač samo za 1, a ne za 3.



Funkcija osvježavanja

Iako je gornji primjer neobičan (mogli smo jednostavno napisati `brojac + 3`) korisno je naučiti kako riješiti ovaj problem jer se možete naći u situaciji kada morate više puta promijeniti jedno stanje prije idućeg renderiranja (takve situacije su rijetke ali se događaju).

Rješenje je da vrijednost novog stanja ne šaljemo kao konstantu već kao tzv. ***updater*** funkciju koja iduće stanje računa na temelju trenutnog stanja. Ta se funkcija dodaje u red izvršavanja (*queue*) i evaluirati će se tek u trenutku kada React započne sa izračunom novog stanja, a ne u trenutku poziva kao u gornjem primjeru. Samim time *updater* funkcija kao argument prima trenutnu vrijednost varijable stanja u memoriji, a ne u *snapshotu*.

Pogledajmo primjer:

```
function App() {
  const [brojac, postaviBrojac] = useState(0)

  function uvecajBrojac(){
    postaviBrojac(trenutnoStanje => trenutnoStanje + 1)
    postaviBrojac((trenutnoStanje) => { return trenutnoStanje + 1})
    postaviBrojac(br => br + 1)
  }

  return (
    <div>
      <p>Broj: {brojac}</p>
      <button onClick={uvecajBrojac}>+</button>
    </div>
  )
}
```

Sada funkciji `postaviBrojac` kao argument ne šaljemo izraz koji se odmah evaluira (npr `brojac + 1`) već šaljemo funkciju. Funkcija kao parametar prima trenutnu vrijednost stanja kojeg osvježavamo, a kao povratnu vrijednost mora primiti novu vrijednost stanja. Srednji poziv prikazuje funkciju u punom obliku, a prvi i treći u skraćenom obliku kojeg nam omogućuje JS sintaksa kada imamo samo jedan argument i povratni izraz. Brojač nam sada radi ispravno:

Broj: 0
+

Zadatak:

Možete li pretpostaviti koje će biti stanje brojača nakon ovog poziva:

```
function uvecajBrojac(){
  postaviBrojac(br => br + 1)
  postaviBrojac(12)
  postaviBrojac(br => br + 1)
}
```

ili nakon ovog:

```
const [brojac, postaviBrojac] = useState(0)

function uvecajBrojac(){
  postaviBrojac(br => br + 1)
  postaviBrojac(12)
  postaviBrojac(br => brojac + 1)
}
```

Last updated on March 13, 2023

React hooks useRef