React elementi i JSX

Centralne figure React sistema su komponente koje omogućavaju da se korisničko okruženje jedne aplikacije efikasno organizuje, odnosno da se podeli na manje jedinice, pri čemu svaka takva jedinica poseduje jedinstveno zaduženje. Stoga je poznavanje različitih načina za kreiranje, konfigurisanje komponenti React aplikacije i rukovanje njima najveći izazov prilikom učenja korišćenja React aplikacija. Ipak, komponente React aplikacija nisu najmanji gradivni blokovi od kojih je sačinjeno korisničko okruženje React aplikacija. Naime, komponente React aplikacija sačinjene su iz jednog elementa ili od više pojedinačnih elemenata.

Stoga, pre nego što se upustimo u detaljniju analizu komponenata React aplikacija, neophodno je da se upoznamo sa pojmom koji omogućava apstrahovanje HTML koda koji korisnik direktno vidi unutar jedne React aplikacije. Sastavni deo priče o React elementima jeste i JSX jezik, s obzirom na to da je reč o jeziku koji olakšava kreiranje React elemenata. Stoga će lekcija pred vama biti posvećena React elementima i njihovom kreiranju korišćenjem JSX jezika.

React elementi

Elementi su najmanji gradivni blokovi jedne React aplikacije. Njima se opisuje ono što želimo da vidimo na displeju. Osnovni način za kreiranje React elemenata jeste korišćenje metode `React.createElement()`:

```
React.createElement(
  type,
  [props],
  [...children]
)
```

Možete videti da se React elementi opisuju korišćenjem tri informacije:

- tip elementa (`div`, `p`, `section`, `h1`…),
- svojstva elementa (`className`, `id`, `style`…)a
- sadržaj, odnosno potomci elementa.

Primer jednog React elementa može da izgleda ovako:

```
let elem = React.createElement(
  'h1',
  {
    className: 'heading',
    id: "my-heading"
  },
  "I am React Element!");
```

Na ovaj način kreiran je element React aplikacije kojim se opisuje jedan DOM element. To će biti DOM element tipa `h1`, on će na sebi imati klasu `heading` i id `my-heading`, a kao svoje potomke imaće tekstualni čvor sa sadržajem *I am React Element!*.

Prikaz React elemenata

U jednoj od prethodnih lekcija mogli ste da vidite da je osnovni način za generisanje neke prezentacije – korišćenje metode `ReactDOM.render()`. To je metoda kojom je moguće prikazati neki React element:

```
ReactDOM.render(element, container[, callback])
```

Kao svoj prvi parametar, metoda `ReactDOM.render()` prihvata element na osnovu koga formira prikaz unutar React aplikacije. Kompletan kod kojim se kreira i prikazuje jedan element React aplikacije može da izgleda ovako:

```
let elem = React.createElement(
  'h1',
  {
    className: 'heading',
    id: "my-heading"
  },
  "I am React Element!");

ReactDOM.render(elem, document.getElementById('root'));
```

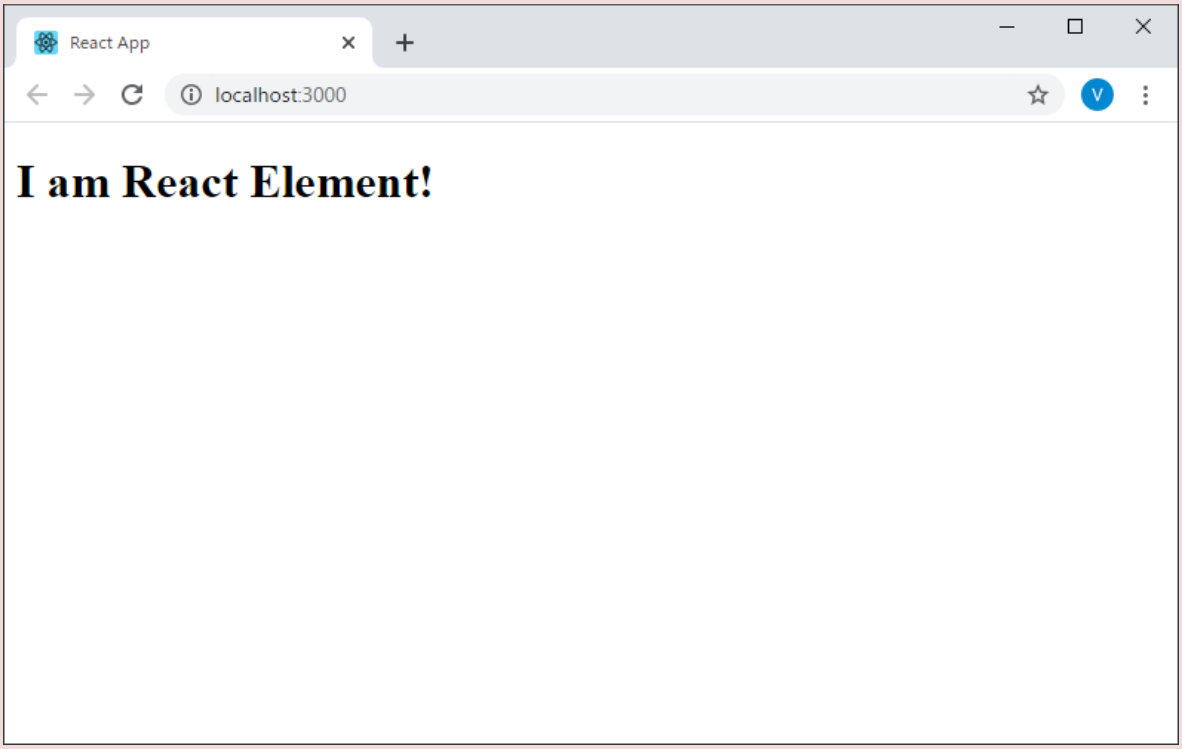
Ovakav kod postavite unutar `index.js` fajla React projekta na kome radimo, nakon naredbi za importovanje React modula:

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

let elem = React.createElement(
  'h1',
  {
    className: 'heading',
    id: "my-heading"
  },
  "I am React Element!");

ReactDOM.render(elem, document.getElementById('root'));
```

Sačuvajte izmene i pokrenite aplikaciju na razvojnom serveru upućivanjem komande:

```
npm start
```

Unutar web pregledača moći ćete da vidite prikaz kao na slici 10.1.



Slika 10.1. React element renderovan unutar HTML dokumenta

HTML kod kreiranog elementa izgleda ovako:

```
<h1 class="heading" id="my-heading">I am React Element!</h1>
```

Kreiranje elemenata korišćenjem JSX-a

Prethodni redovi su pokazali osnovni način za kreiranje elemenata React aplikacije, koji je podrazumevao direktnu upotrebu `React.createElement()` metode. Ipak, React obezbeđuje i znatno jednostavniji način na koji je moguće kreirati elemente. Reč je o pristupu koji podrazumeva upotrebu JSX jezika.

React element iz prethodnog primera se uz korišćenje JSX-a lako može kreirati na sledeći način:

```
      let const = <h1 className="heading" id="my-heading">I am React
Element!</h1>;
```

Kao što vidite, JSX omogućava da se elementi kreiraju korišćenjem mnogo jednostavnije sintakse, koja predstavlja kombinaciju JavaScript i HTML jezika. Ipak, JSX je bliži JavaScript nego HTML jeziku, što možete videti i iz prikazanog primera. Za definisanje klase je iskorišćen naziv DOM svojstva (`className`), a ne HTML atribut `class`.

Unutar JSX-a atributi se tretiraju kao DOM svojstva

S obzirom na to da je JSX bliži JavaScript jeziku nego HTML jeziku, atributi se definišu navođenjem naziva DOM svojstava. Zbog toga se, umesto HTML atributa `class`, koristi svojstvo `className`.

JSX se u pozadini pretvara u pozive već prikazane metode `React.createElement()`, pa je tako prikazani JSX ekvivalentan već viđenoj naredbi:

```
let elem = React.createElement(
  'h1',
  {
    className: 'heading',
    id: "my-heading"
  },
  "I am React Element!");
```

Definisanje izraza unutar JSX-a

JSX omogućava da se unutar koda za kreiranje React elemenata umetnu izrazi. To se postiže korišćenjem vitičastih zagrada – `{}`:

```
const content = "I am React Element!";
const elem = <h1 className="heading" id="my-heading">{content}</h1>;
```

Sada je sadržaj našeg React elementa izmešten unutar jedne konstante sa nazivom `content`. Zatim je takva konstanta uvršćena u JSX kod, unutar vitičastih zagrada. Na taj način će biti obavljena klasična interpolacija vrednosti konstante, pa će efekat da bude identičan kao i kada je sadržaj bio direktno unutar elementa.

Unutar vitičastih zagrada je moguće postavljati bilo koji validan JavaScript izraz:

```
const user = {
  firstName: 'Ben',
  lastName: 'Torrance'
};

const elem = <h1 className="heading" id="my-heading">{user.firstName +
" " + user.lastName}</h1>;
```

Sada je unutar vitičastih zagrada definisana konkatenacija dve `string` vrednosti koje su predstavljene svojstvima objekta `user`.

Unutar JSX-a je moguće definisati i pozive metoda:

```
const user = {
  firstName: 'Ben',
  lastName: 'Torrance',
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};

const elem = <h1 className="heading" id="my-
heading">{user.fullName()}</h1>;
```

Unutar JSX-a je sada postavljen poziv metode `fullName()` objekta `user`. Stoga će renderovanjem sadržaj elementa da postane ime i prezime osobe.

Pitanje

JSX predstavlja kombinaciju dva jezika:

a) CSS i HTML
b) HTML i JavaScript
c) JavaScript i PHP
d) JavaScript i CSS

Objašnjenje:

JSX omogućava da se elementi kreiraju korišćenjem mnogo jednostavnije sintakse, koja predstavlja kombinaciju JavaScript i HTML jezika.

JSX za kreiranje struktura elemenata

JSX nije ograničen na kreiranje elemenata sa jednostavnim, tekstualnim sadržajem. Moguće je definisati elemente koji poseduju razgranatu strukturu elemenata potomaka. Evo jednog takvog primera:

```
const elem = <ul>
  <li>Amaranth</li>
  <li>Auburn</li>
  <li>Chestnut</li>
  <li>Frostbite</li>
</ul>;
```

Ovo je primer jednog React elementa sa složenijom strukturom elemenata potomaka. Na ovaj način će unutar HTML dokumenta da bude prikazana jedna neuređena lista. Jedini uslov prilikom definisanja razgranatih struktura elemenata jeste da svi oni budu obuhvaćeni jednim korenim elementom (u primeru je to `ul` element).

JSX unutar JavaScript koda

Do sada su prikazani primeri koji su podrazumevali definisanje JSX-a kao vrednosti JavaScript konstanti. Ipak, JSX je moguće pisati i unutar JavaScript funkcija ili koristiti unutar uslovnih blokova i petlji. Na taj način je moguće obaviti dinamičku izgradnju elemenata.

```
function makeMessage(age) {

  if (age > 17) {
    return <div>You are welcome!</div>
  } else {
    return <div>You can't enter!</div>
  }

}
```

Unutar prikazane funkcije obavlja se kreiranje React elementa. U zavisnosti od vrednosti koja se prosledi, kao povratna vrednost funkcije će biti emitovan element sa različitim sadržajem. Evo kako se ovakva funkcija može upotrebiti:

```
ReactDOM.render(makeMessage(18), document.getElementById('root'));
```

Mnogo jednostavniji način za postizanje identičnog efekta jeste korišćenje ternarnog operatora:

```
let age = 17;
const elem = (age > 17) ? <div>You are welcome!</div> : <div>You can't
enter!</div>;

ReactDOM.render(elem, document.getElementById('root'));
```

Upravo prikazani primer je predstavljao način na koji se može obaviti uslovno renderovanje u Reactu. Sledeći primer će ilustrovati dinamičko generisanje HTML elemenata na osnovu vrednosti jednog niza:

```
const colors = ['Amaranth', 'Auburn', 'Chestnut', 'Frostbite'];

const elem = <ul>
  {colors.map((value, index) => {
    return <li key={index}>{value}</li>
  })}
</ul>;

ReactDOM.render(elem, document.getElementById('root'));
```

Primer će proizvesti identičan efekat kao nešto ranije prikazani kod kojim se dobijala jedna neuređena lista. Ipak, ovoga puta takva lista nastaje dinamički, na osnovu elemenata jednog niza.

Metoda map()

Upravo prikazani primer je realizovan korišćenjem jedne posebne metode. Reč je o metodi `map()`, objekta `Array`. Metoda `map()` omogućava da se obavi efikasno transformisanje vrednosti jednog niza, u niz sa drugačijim osobinama. To se obavlja tako što metoda `map()`, kreira novi niz na osnovu povratnih vrednosti koje se dobijaju od funkcije koja se poziva po jednom za svaki element nekog niza.

U prikazanom primeru je metoda `map()` pozvana nad `colors` nizom. Kao parametar je definisana jedna callback funkcija koja se aktivira za svaki element niza `colors`. Unutar takve funkcije obavlja se transformisanje vrednosti elementa niza, tako što se vrednost umeće unutar `<li> </li>` tragova.

Prilikom dinamičkog generisanja lista, savetuje se da svaka stavka liste poseduje i sopstveni, jedinstveni ključ, koji jednoznačno identifikuje takvu stavku. Takav ključ se definiše svojstvom **key**, a nakon renderovanja takva vrednost nije vidljiva unutar HTML koda. Vrednost se interno koristi u Reactu za optimizaciju prikazivanja elemenata.

Definisanje vrednosti HTML atributa

Važan aspekt rada sa DOM elementima jeste i mogućnost definisanja vrednosti njihovih atributa. To je pomoću React aplikacije moguće obaviti korišćenjem već viđenih pristupa:

```
var user = {
  name: "Ben Torrance",
  email: "ben@email.com",
  balance: 55353.93434,
  thumb_url: "/img/sdg24bv2.png"
};


const elem = <div>
  <p><img src={user.thumb_url} alt="profile img"></img></p>
  <p>{user.name}</p>
  <p>{user.email}</p>
  <p>{user.balance}</p>
</div>;

ReactDOM.render(elem, document.getElementById('root'));
```

U primeru se kreira element aplikacije React kojim se prikazuju podaci jednog korisnika. Među podacima je i putanja do profilne slike. Takav podatak se postavlja za vrednost `src` atributa, kao i interpolacija ostalih podataka (korišćenjem vitičastih zagrada). Na kraju, u prikazanom primeru možete da vidite da je vrednost atributa moguće obavljati i na tradicionalni način – korišćenjem `string` vrednosti između navodnika, što je učinjeno na primeru `alt` atributa.

Unutar JSX-a atributi se tretiraju kao DOM svojstva

JSX je bliži JavaScript jeziku nego HTML jeziku. Stoga se atributi definišu navođenjem naziva DOM svojstava. Tako se, umesto HTML atributa `class`, koristi `className`, a umesto na primer atributa `tabindex`, svojstvo `tabIndex`. Kod `tabindex` atributa razlika je samo u upotrebi camelCase notacije, koju je potrebno poštovati.

Obrada događaja React elemenata

Događaje koje proizvode DOM elementi moguće je slušati definisanjem naziva funkcije za obradu, unutar vitičastih zagrada:

```
function sayHelloWorld(e) {
  alert("Hello world");
}

const elem = <button onClick={sayHelloWorld}>Say Hello World</button>;

ReactDOM.render(elem, document.getElementById('root'));
```

Na ovaj način će klikom na renderovani `button` element da bude prikazan modalni prozor sa porukom *Hello World*.

Funkcije za obradu događaja dobijaju objekat događaja (u primeru e), baš kao i prilikom korišćenja funkcije `addEventListener()`. Ipak, bitno je znati da se podrazumevano ponašanje kod nekih događaja (formi i linkova) ne može sprečiti emitovanjem povratne vrednosti `false`, već isključivo pozivanjem metode `preventDefault()`:

```
function sayHelloWorld(e) {
  e.preventDefault();
  alert("Redirect to Google is prevented!");
}

const elem = <a href="https://www.google.com"
onClick={sayHelloWorld}>Go to Google</a>;

ReactDOM.render(elem, document.getElementById('root'));
```

Funkcijama za obradu događaja je moguće prosleđivati neke proizvoljne parametre na sledeći način:

```
function sayHello(e, name) {
  alert("Hello " + name);
}

const elem = <button onClick={(e) => sayHello(e, "Ben")}>Say hello to
Ben</button>;

ReactDOM.render(elem, document.getElementById('root'));
```

React elementi su nepromenljivi

React elementi su nepromenljivi (engl. *immutable*). To praktično znači da nakon kreiranja jednog elementa nije moguće menjati njegov sadržaj ili atribute:

```
let counter = 0;

function onClick(e) {
  counter++;
}

const elem = <button onClick={onClick}>{counter}</button>;

ReactDOM.render(elem, document.getElementById('root'));
```

Prikazani primer ilustruje nepromenljivost React elemenata. Kreirano je jedno dugme, čiji je tekst formiran na osnovu vrednosti promenljive `counter`. Na dugmetu je registrovana i funkcija koja će se aktivirati prilikom klika. Unutar nje se uvećava vrednost promenljive `counter`, prilikom svakog klika. Ipak, ukoliko primer isprobate unutar web pregledača, videćete da se na stranici ništa ne dešava. Jednostavno, jednom prikazani React elementi se ne mogu promeniti. To je i razlog zbog koga su u svim primerima iz ove lekcije, za predstavljanje elemenata i njihovih podataka, korišćene konstante (ključna reč `const`).

React elementi se koriste za predstavljanje komponenata

U prethodnim primerima ste mogli da vidite React elemente kojima su direktno opisivani DOM elementi, koji su zatim renderovani unutar HTML dokumenta. Ipak, pored predstavljanja običnih DOM elemenata, React elementi imaju mogućnost da predstavljaju i React komponente. O tome će biti reči u narednoj lekciji.

Rezime

- Elementi su najmanji gradivni blokovi jedne React aplikacije.
- React elementima se opisuje ono što želimo da vidimo na displeju.
- Osnovni način za kreiranje React elemenata jeste korišćenje metode `React.createElement()`.
- Metoda kojom je moguće prikazati (renderovati) neki React element jeste `ReactDOM.render()`.
- JSX omogućava da se elementi kreiraju korišćenjem mnogo jednostavnije sintakse, koja predstavlja kombinaciju JavaScript i HTML jezika.
- JSX je bliži JavaScript jeziku nego HTML jeziku.
- JSX se u pozadini pretvara u pozive metode `React.createElement()`.
- Unutar JSX-a moguće je umetnuti JavaScript izraze korišćenjem jednog para vitičastih zagrada – `{}`.
- JSX je moguće pisati i unutar JavaScript funkcija ili koristiti unutar uslovnih blokova i petlji.
- Unutar JSX-a atributi se tretiraju kao DOM svojstva.
- React elementi su nepromenljivi, što znači da nakon kreiranja jednog elementa nije moguće menjati njegov sadržaj ili atribute.
- Pored predstavljanja običnih DOM elemenata, React elementi imaju mogućnost da predstavljaju i React komponente.