Vue integracija i osnove

Priča o bibliotekama i softverskim okvirima koji se danas koriste za izgradnju JavaScript aplikacija započeće softverskim okvirom koji se naziva Vue.

Napomena
Vue se izgovara kao i engleska reč view – /vju:/.

Iako se na zvaničnom sajtu može pročitati da je reč o softverskom okviru, Vue se može koristiti i kao obična JavaScript biblioteka. Stoga će u lekciji pred vama biti ilustrovan osnovni način za integraciju Vue biblioteke unutar jednog projekta. To će nam pomoći da se upoznamo sa osnovnim osobinama Vue sistema. Nakon upoznavanja osnovnih osobina, u narednim lekcijama će biti obrađeni i napredni pristupi korišćenja Vuea.

Šta je Vue?

Vue je softverski okvir koji je namenjen razvoju prezentacionog sloja web aplikacija.

Slika 4.1. Vue logo

- Vue olakšava sledeće aspekte frontend programiranja: povezivanje podataka i prezentacije, generisanje HTML koda, obrada događaja, kreiranje nezavisnih komponenata koje je moguće koristiti na više mesta i kreiranje animacija i tranzicija.

- **Povezivanje podataka i prezentacije**

Pojam povezivanja (engl. *binding*) koristi se da označi posebnu vezu između podataka i prezentacije, po kojoj se promene podataka automatski propagiraju do prezentacionog sloja. U jednoj od prethodnih lekcija ste mogli da vidite koliki je izazov takvo nešto obaviti korišćenjem Vanilla JavaScripta, a u nastavku ove lekcije ćete imati prilike da vidite kako Vue olakšava obavljanje takvog posla.

- **Generisanje HTML koda**

Vue omogućava veoma lako kreiranje HTML elemenata i njihovog sadržaja. Stoga nema potrebe za korišćenje DOM API-ja, zato što Vue njegovo korišćenje u potpunosti apstrahuje za nas.

- **Obrada događaja**

Obrada događaja je još jedan aspekt frontend programiranja koji zahteva pažljivu implementaciju, a veoma često potrebno je i dosta ponavljanja jednog istog koda. Vue omogućava da se pretplata na DOM događaje obavi veoma jednostavno, bez potrebe za korišćenjem `addEventListener()` metode.

- **Kreiranje nezavisnih komponenata koje je moguće koristiti na više mesta**

Vue omogućava da se korisnička okruženja web sajtova kreiraju kao skup većeg broja nezavisnih komponenata koje međusobno sarađuju i tako grade zaokruženo korisničko okruženje. Takav pristup izgradnje prezentacije i njene logike naziva se arhitektura bazirana na komponentama (engl. *Component Architecture*). Komponentna arhitektura je osnovna osobina Vuea, ali i ostalih biblioteka koje se koriste za frontend razvoj.

- **Kreiranje animacija i tranzicija**

Vue apstrahuje i obavljanje nekih uobičajenih tranzicija i animacija nad HTML elementima. Tako je, na primer, korišćenjem Vuea na lak način moguće definisati tranziciju prilikom dodavanja ili uklanjanja elemenata iz DOM strukture.

Integracija Vue biblioteke

Pre početka korišćenja, Vue je neophodno integrisati unutar projekta u kome će biti korišćen. Integraciju je moguće obaviti na dva osnovna načina. Pristupi za integraciju su sledeći:

1. direktna integracija unutar HTML dokumenta korišćenjem `<script>` elementa,
2. integracija korišćenjem Node.js menadžera paketa (npm) i Vue Command-Line Interfacea (Vue CLI),

U ovoj lekciji će biti prikazana direktna integracija, a u jednoj od narednih lekcija, nakon što se upoznamo sa osnovama Vue sistema, biće ilustrovana i integracija korišćenjem npm menadžera paketa.

Direktna integracija Vuea korišćenjem <script> elementa

Najjednostavniji način za integraciju Vuea jeste direktno postavljanje `<script>` elementa unutar HTML dokumenta. Ovakvu vrstu integracije moguće je obaviti na dva načina:

- integracijom lokalnog `vue.js` fajla, koji je prethodno potrebno preuzeti i smestiti unutar projektne strukture, ili
- integracijom `vue.js` fajla uz korišćenje CDN servera.

Fajlove za lokalnu integraciju Vuea možete preuzeti sa sledećih linkova:

- `vue.js` – razvojna verzija, koja je korisna tokom razvoja i testiranja, zato što poseduje različite mehanizme za praćenje izvršavanja i lak uvid u izvorni kod,
- `vue.min.js` – produkciona verzija, koju je potrebno koristiti kada je aplikacija spremna za isporuku.

`script` element za integraciju Vuea, kada se fajlovi nalaze lokalno, može da izgleda ovako:

```
<script src="vue.js"></script>
```

Ukoliko se koristi CDN integracija, `script` element može da izgleda ovako:

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

Prvi primer Vue aplikacije

Odmah nakon integracije, moguće je kreirati jednostavan primer koji će prikazati osnovne osobine Vuea:

```
<div id="container">
    {{ message }}
</div>

<script src="vue.js"></script>
<script>

    var obj = {message: "Hello World!"};

    var vm = new Vue({
        el: '#container',
        data: obj
    })

</script>
```

Upravo prikazani kod jeste sadržaj tela HTML dokumenta, stoga je njega potrebno da prekopirate unutar `body` dela HTML dokumenta. Kada takav dokument otvorite unutar web pregledača, moći ćete da vidite poruku:

```
Hello World!
```

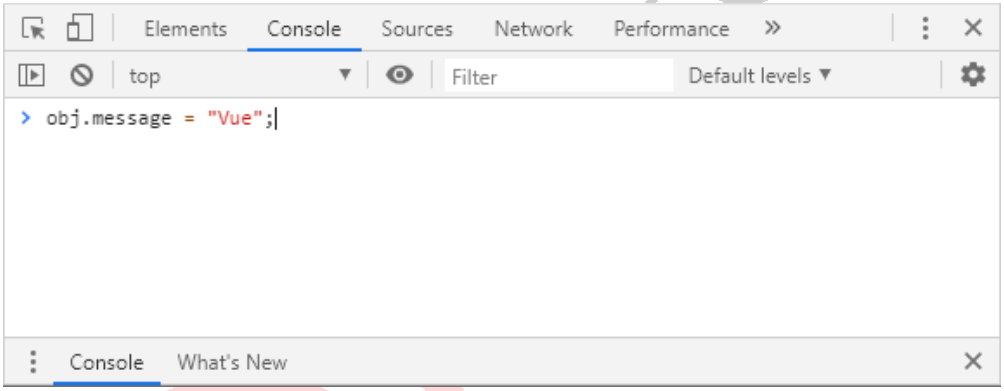
Na osnovu upravo prikazanog primera moguće je videti osnovnu Vue funkcionalnost na delu. U primeru je kreiran jedan `div` element sa `id`-jem `container`. Ipak, njegov sadržaj nije uobičajen, već predstavlja specifičnu tvorevinu softverskog okvira Vue:

```
{{ message }}
```

Korišćenjem ovakvog posebnog formata za formiranje sadržaja HTML elemenata deklarativno se postavljaju sadržaji direktno unutar objektnog modela dokumenta. Drugim rečima, Vue, na osnovu specijalno upotrebljenog formata, koji podrazumeva dve otvorene i dve zatvorene vitičaste zagrade, zna da je ono što se nalazi između njih – naziv promenljive čiju vrednost je potrebno da umetne kao sadržaj ovakvog `div` elementa. Naziv promenljive čija vrednost će biti postavljena unutar `div` elementa je `message`, a takva promenljiva je definisana unutar objekta na koji upućuje promenljiva `obj`.

Promenljiva `obj` je postavljena za vrednost svojstva `data`, jednog posebnog objekta, koji se naziva Vue objekat ili **Vue instanca**. Takav objekat je u primeru pridružen promenljivoj sa nazivom `vw`.

Odmah na početku je bitno razumeti da se prikazanim kodom ne obavlja samo puko ispisivanje vrednosti jedne promenljive unutar HTML elementa, već potpuno povezivanje podataka izraženih `obj` objektom i objektnog modela dokumenta. To omogućava da se bilo kakva promena nad podacima automatski propagira i unutar objektnog modela dokumenta. Kako biste se u to uverili, dovoljno je da promenite vrednosti promenljive `message` unutar konzole samog web pregledača, a takva promena će automatski da bude propagirana i unutar HTML dokumenta (slika 4.2).



Slika 4.2. Izmenom vrednosti promenljive message, unutar konzole web pregledača automatski se menja tekst div elementa

Reaktivnost

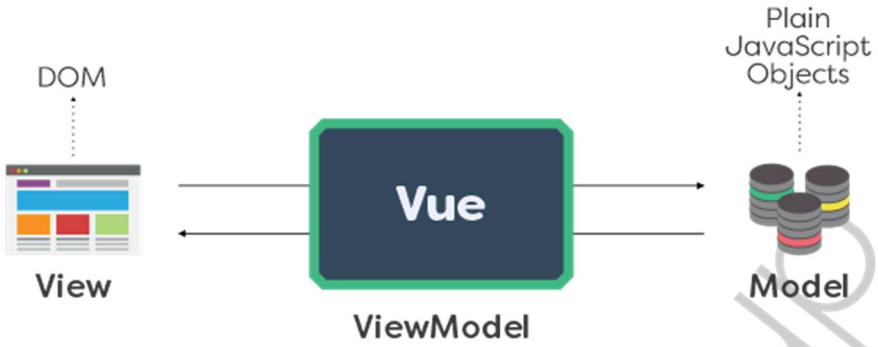
Mogućnost automatskog propagiranja promena koje se dešavaju nad podacima drugačije se naziva reaktivnost. Reč je zapravo o jednoj od programerskih paradigmi kojom se definiše način na koji se podaci kreću kroz različite slojeve aplikacije. Programiranje kojim se postiže reaktivnost drugačije se naziva reaktivno programiranje.

Reaktivnost je jedna od osnovnih osobina gotovo svih biblioteka koje su namenjene kreiranju prezentacionih delova web aplikacija (Vue, Angular, React...).

Vue anatomija

Upravo prikazani prvi primer korišćenja Vuea ilustrovao je njegovu najznačajniju osobinu – automatsko propagiranje promene podataka. Na koji se način takvo nešto postiže i koji se specifični pojmovi Vuea tom prilikom koriste biće objašnjeno u narednim redovima, koji će se baviti anatomijom Vuea.

Vue je namenjen kreiranju dinamičkih korisničkih okruženja web aplikacija i direktno je inspirisan MVVM softverskim šablonom. Ukoliko se posmatra iz ugla MVVM šablona, Vue je zadužen za kreiranje ViewModel sloja, s obzirom na to da povezuje sloj prezentacije (View) i sloj podataka (Model).



Slika 4.3. Arhitektura Vuea

Na slici 4.3. možete da vidite osnovnu MVVM strukturu prezentacionog dela web aplikacija, kada se za njihov razvoj koristi Vue. Na levoj polovini slike 4.3. je prezentacioni sloj (View), odnosno HTML struktura i njena objektna reprezentacija (DOM). U našem prvom primeru prezentacioni sloj je zapravo bio sačinjen iz jednog `div` elementa sa id-jem `container`.

Na desnoj polovini slike 4.3. nalazi se sloj podataka (Model). Prilikom rada sa Vueom podaci se predstavljaju u formi običnih JavaScript objekata (engl. *plain JavaScript object*). U nešto ranije prikazanom primeru podaci su bili predstavljeni objektom sa jednim svojstvom `message`.

Centralna figura MVVM arhitekture prilikom upotrebe Vuea jeste ViewModel komponenta koja se predstavlja Vue objektom, odnosno Vue instancom. Stoga uopšte nije slučajno da je nešto ranije, u prvom prikazanom primeru, promenljiva koja čuva referencu na takav objekat imenovana nazivom `vm` (što je skraćenica od ViewModel). Unutar takve promenljive smeštena je referenca na Vue objekat koji je kreiran korišćenjem konstruktorske funkcije `Vue()`.

Pitanje

Ukoliko se Vue posmatra iz ugla MVVM softverskog šablona, za realizaciju kog sloja je on zadužen?

a) View
b) ViewModel
c) Model
d) ModelView

Objašnjenje:

Ukoliko se posmatra iz ugla MVVM šablona, Vue je zadužen za kreiranje ViewModel sloja, s obzirom na to da povezuje sloj prezentacije (View) i sloj podataka (Model).

Vue objekat

Centralna figura Vue sistema jeste Vue objekat. On se kreira korišćenjem `Vue()` konstruktorske funkcije:

```
var vm = new Vue({
      // options
})
```

Na osnovu upravo prikazane sintakse `Vue()` konstruktorske funkcije, može se videti da ona kao ulazni parametar prihvata jedan objekat. Reč je o objektu kojim se konfiguriše Vue instanca koja se kreira. U prvom Vue primeru, koji je prikazan nešto ranije, kreiranje Vue instance je izgledalo ovako:

```
var vm = new Vue({
      el: '#container',
      data: obj
})
```

Možete videti da objekat koji se prosleđuje `Vue()` konstruktorskoj funkciji poseduje dva veoma značajna svojstva:

- `el` – koreni element u objektnom modelu dokumenta i
- `data` – podaci.

Korišćenjem dva upravo prikazana svojstva povezuju se prezentacija (View) i podaci (Model). Svojstvom `el` definiše se HTML element unutar koga će da budu prikazivani podaci koji su definisani svojstvima `data` objekta.

Definisanje prezentacije

Definisanje prezentacije, odnosno HTML elementa unutar koga će da budu smeštani podaci, obavlja se korišćenjem svojstva `el`. Prvi prikazani primer podrazumevao je samo jedan HTML element i jedno objektno svojstvo čija je vrednost umetana unutar takvog elementa. Ipak, `el` svojstvom se zapravo definiše koreni element, što znači da će i svi elementi koji se nalaze unutar takvog elementa automatski da budu uključeni u reaktivni Vue sistem:

```
<div id="container">
      <div>{{ name }}</div>
      <div>{{ email }}</div>
</div>

<script src="vue.js"></script>
<script>

      var user = { name: "Ben Torrance", email: "ben@email.com" };

      var vm = new Vue({
            el: '#container',
            data: user
      })

</script>
```

Sada je primer delimično modifikovan tako što je napravljena nešto razgranatija HTML struktura. `container` element unutar sebe sada poseduje još dva `div` elementa. Unutar takvih `div` elemenata biće prikazano ime i email adresa jednog korisnika:

```
Ben Torrance
ben@email.com
```

Iz ovog proširenog primera se može videti da Vue automatski u svoj reaktivni sistem uključuje i sve elemente potomke elementa koji je korišćenjem `el` svojstva definisan kao koreni.

Definisanje podataka

Podaci se definišu korišćenjem svojstva `data`. Reč je o svojstvu koje kao svoju vrednost može da prihvati JavaScript objekat. Sva svojstva takvog objekta automatski se uključuju u Vue reaktivni sistem i moguće ih je koristiti po nazivu kako bi se obavilo povezivanje sa slojem prezentacije.

Bitno je reći da svojstva `data` objekta automatski postaju i svojstva Vue instance. To praktično znači da je moguće napisati nešto ovakvo:

```
vm.name === user.name  //true
```

Ovakvo poređenje će da proizvede logičku vrednost `true`, što dokazuje da je svojstvo `name`, `user` objekta, automatski postalo i svojstvo Vue instance.

Svojstva i metode Vue instance

Vue instance poseduju i skup veoma korisnih svojstava, koja sistem automatski kreira. Kako bi se razlikovala od korisnički definisanih, sva takva svojstva započinju karakterom dolar ($). Na primer, ugrađenom svojstvu `data` moguće je pristupiti na sledeći način:

```
vm.$data
```

Već ste videli da je `$data` svojstvo kojim se predstavljaju podaci, te je stoga njegova vrednost identična kao i vrednost promenljive kojom se predstavljaju podaci:

```
vm.$data === user //true
```

Ovakvo poređenje stvoriće `true` logičku vrednost.

Po identičnom principu, Vue omogućava pristup korenom prezentacionom elementu:

```
vm.$el === document.getElementById("container") //true
```

I ovakvo poređenje stvoriće `true` logičku vrednost.

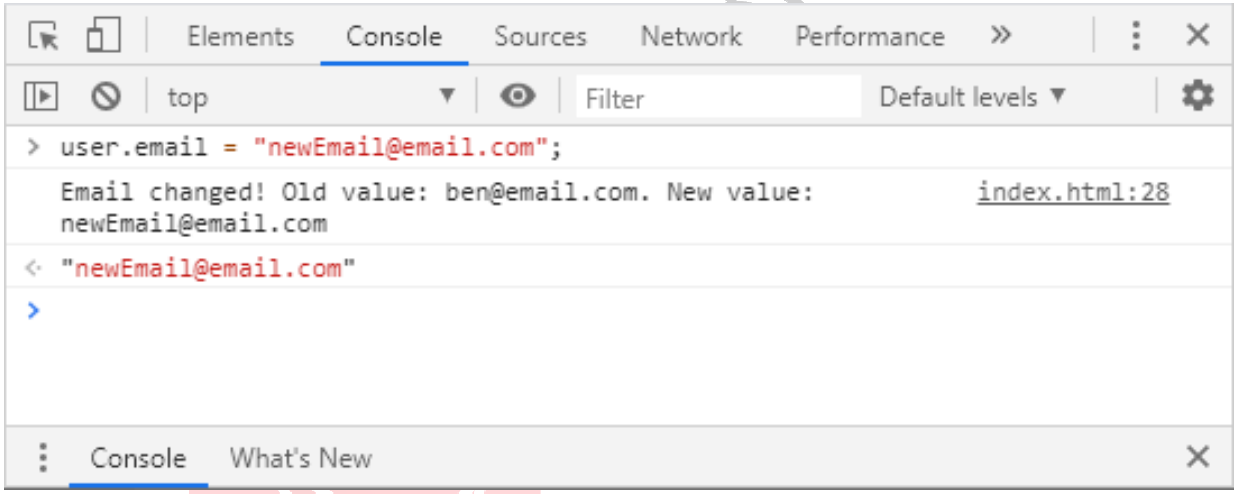
Pored ugrađenih svojstava, Vue instance poseduju i skup vrlo korisnih ugrađenih metoda. Baš kao i svojstva, i one su imenovane korišćenjem prefiksa dolar ($). Jedna od takvih metoda je i metoda `$watch()`, kojom je moguće pratiti promene vrednosti svojstava objekta koji je deo reaktivnog sistema Vue:

```
            var user = { name: "Ben Torrance", email: "ben@email.com" };

            var vm = new Vue({
                el: '#container',
                data: user
            })

            vm.$watch('email', function (newValue, oldValue) {
                console.log("Email changed! Old value: " + oldValue + ". New
value: " + newValue);
            })
```

U prikazanom kodu, metoda `$watch()` je iskorišćena za dobijanje dojava o promeni vrednosti svojstva `email`, koje je sastavni deo reaktivnog sistema. Stoga svaka promena vrednosti svojstva `email` kao posledicu ima aktiviranje funkcije `$watch()` (slika 4.4).



Slika 4.4. Promenom vrednosti svojstva email aktivira se logika definisana $watch() funkcijom

`$watch()` je jedna od ugrađenih metoda Vue instance, a sa ostalima ćemo se upoznavati kada se za upotrebom neke od njih javi potreba.

Rezime

- Vue je softverski okvir koji je namenjen razvoju prezentacionog sloja web aplikacija.
- Vue se izgovara kao i engleska reč *view.*
- Pre početka korišćenja, Vue je neophodno integrisati unutar projekta u kome će biti korišćen.
- Integraciju Vuea je moguće obaviti na dva načina: direktnom integracijom jedne `.js` biblioteke korišćenjem `script` elementa, ili upotrebom npm menadžera paketa i Vue CLI-ja.
- Vue je namenjen kreiranju dinamičkih korisničkih okruženja web aplikacija i direktno je inspirisan MVVM softverskim šablonom.
- Vue je zadužen za kreiranje ViewModel sloja, s obzirom na to da povezuje sloj prezentacije (View) i sloj podataka (Model).
- Centralna figura sistema Vue jeste Vue objekat, odnosno Vue instanca.
- Vue instanca se kreira korišćenjem `Vue()` konstruktorske funkcije.
- `Vue()` konstruktorskoj funkciji se prosleđuje objekat za podešavanje instance.
- Unutar objekta za konfigurisanje Vue instance definisanje prezentacije obavlja se korišćenjem svojstva `el`.
- Unutar objekta za konfigurisanje Vue instance definisanje podataka obavlja se korišćenjem svojstva `data`.
- Vue instance poseduju i skup svojstava i metoda koje sistem automatski kreira, a karakteriše ih prefiks `$`.