

Vue šabloni

U prethodnoj lekciji dotakli smo se osnovnih funkcionalnosti koje sistem Vue izlaže na korišćenje. Pre svih, misli se na osnovni način na koji se povezuju podaci i prezentacioni sloj. U takvoj, uvodnoj priči, akcenat je bio na Vue instanci, kao figuri koja zauzima centralnu poziciju u sistemu koji omogućava postizanje reaktivnog ponašanja. U ovoj lekciji nastavljamo upoznavanje ostalih najznačajnijih elemenata od kojih je Vue sačinjen. Tako će u lekciji pred vama biti reči o Vue šablonima.

Šta su šabloni?

Šablon (engl. *template*) je naziv koji se koristi za kod kojim se kreira prezentacioni sloj aplikacije. U prethodnoj lekciji kreirano je nekoliko vrlo jednostavnih šablona. Primer prvog šablona je izgledao ovako:

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

Reč je o vrlo jednostavnom šablonu, sa samo jednim HTML elementom, unutar koga se posredstvom Vue instance ispisuje vrednost promenljive `message`.

Nešto kompleksniji šablon koji smo videli u prethodnoj lekciji je izgledao ovako:

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

Ovaj šablon sačinjen je iz nešto razgranatije strukture, pa su unutar korenog elementa smeštene još dva zasebna `div` elementa za prikazivanje vrednosti svojstava `name` i `email`.

Izgradnja Vue šablona

Iz upravo prikazanih primera možete da vidite da se za izgradnju Vue šablona koristi sintaksa koja se bazira na HTML jeziku. Drugim rečima, svaki Vue šablon ujedno je i validan HTML kod, koji može parsirati bilo koji web pregledač. Možda vam se na prvi pogled to ne čini tako, zbog korišćenja dvostrukih vitičastih zagrada kao sadržaja elemenata, ali ukoliko malo bolje razmislite, takve vitičaste zagrade su u potpunosti validan sadržaj jednog HTML elementa. Dvostruke vitičaste zagrade za Vue sistem imaju posebno značenje, ali i kada se Vue ne bi koristio, takav sadržaj bi bio u potpunosti validan. Upravo zbog toga se kaže da je svaki Vue šablon ujedno i validan HTML kod.

Pored dvostrukih vitičastih zagrada, unutar Vue šablona je moguće koristiti još neke tvorevine koje su po HTML specifikaciji potpuno validne, ali za Vue sistem imaju posebno značenje. Tako se može reći da su dva posebna Vue pojma koja je moguće iskoristiti prilikom kreiranja šablona:

- interpolacija i
- direktive.

Interpolacija je zapravo specifičan naziv za pristup koji smo već videli, a koji podrazumeva korišćenje dvostrukih vitičastih zagrada. Stoga smo mi do sada već videli osnovne primere interpolacije, pa će u nastavku priča o interpolaciji biti dodatno proširena i biće predstavljen pojam direktiva.

Interpolacija

U programiranju, pojam interpolacije se odnosi na proces postavljanja vrednosti promenljivih na unapred definisana rezervisana mesta. Unutar Vuea interpolacija izgleda ovako:

```
{{ name }}
```

Vue interpolacija se postiže korišćenjem posebne sintakse, koja podrazumeva upotrebu dva para vitičastih zagrada, unutar kojih se definiše naziv svojstva, objekta kojim se predstavljaju podaci.

Zbog korišćenja ovakve sintakse, koja podrazumeva upotrebu dvostrukih vitičastih zagrada, povezivanje upotrebom interpolacije se unutar Vuea veoma često drugačije naziva **mustache binding**. Upotreba reči *mustache* u nazivu je slikovita i pokušava da dočara asocijaciju na brkove (engl. *mustache*), na koje podsećaju vitičaste zagrade koje se koriste za postizanje povezivanja.

Upotrebom interpolacije moguće je obaviti povezivanje sadržaja HTML elemenata i odgovarajućih svojstava unutar objekta koji predstavlja podatke. Tako interpolacija omogućava da se unutar HTML elementa ispiše tekstualna vrednost nekog svojstva i da svaka promena vrednosti bude propagirana unutar HTML koda.

Proces interpolacije je delimično moguće prilagoditi, zato što Vue omogućava definisanje JavaScript izraza unutar koda za interpolaciju:

```
<div>{{ name.toUpperCase() }}</div>
```

Na ovaj način će proces interpolacije podrazumevati pretvaranje svih malih slova u velika.

Još jedan primer interpolacije može da izgleda ovako:

```
<div>{{ balance.toFixed(2) }}</div>
```

Na ovaj način se prilikom prikazivanja decimalne vrednosti obavlja njeno zaokruživanje na dve decimale.

Direktno definisanje izraza unutar koda za interpolaciju generalno predstavlja lošu praksu, pogotovu ukoliko je reč o složenim izrazima. Upravo zbog toga, Vue obezbeđuje još jedan mehanizam, kojim je izraze moguće premestiti na mnogo prikladnije mesto. Reč je o takozvanim izračunatim svojstvima (engl. *computed properties*).

Pitanje

Da li je prilikom interpolacije moguće definisati JavaScript izraze?

- a) Da
- b) Ne

Objašnjenje:

Proces interpolacije je delimično moguće prilagoditi definisanjem JavaScript izraza unutar koda za interpolaciju.

Computed properties

Izračunata svojstva omogućavaju da se u obliku JavaScript funkcije definiše proizvoljna logika, čija će povratna vrednost moći lako da se interpolira unutar Vue šablona. Tako izračunata svojstva omogućavaju da se izrazi iz šablona premeste unutar Vue instance, a da se unutar šablona, korišćenjem naziva izračunatog svojstva, obavlja interpolacija njegove vrednosti.

Kako biste bolje razumeli značaj izračunatih svojstava, pogledajte sledeći primer:

```
<div id="container">
  <div>
    {{ name.toLowerCase().split(' ').map(word =>
word.charAt(0).toUpperCase() + word.slice(1)).join(' ')}}
  </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>
```

Prikazani primer pokazuje vrlo kompleksnu logiku koja je definisana direktno unutar Vue šablona prilikom obavljanja interpolacije. Reč je o logici kojom se prvi karakteri imena i prezimena pretvaraju u velika slova. Prikazana logika umnogome loše utiče na preglednost šablona, pa samim tim i otežava održavanje. Poseban problem može nastati ukoliko se javi potreba za time da se identično obavi na još nekim mestima. Tada bi bilo neophodno kompletan izraz prekopirati, što krši jedan od osnovnih programerskih principa – DRY (engl. *Don't repeat yourself*).

Rešenje problema se ogleda u upotrebi izračunatih svojstava:

```
<div id="container">
  <div>
    {{ capitalizedName }}
  </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,
  computed: {
    capitalizedName: function () {
      return this.name.toLowerCase().split(' ').map(word
=> word.charAt(0).toUpperCase() + word.slice(1)).join(' ');
    }
  }
});

</script>
```

Sada je unutar Vue instance definisano svojstvo **computed**. Reč je o svojstvu čija je vrednost objekat kojim se definišu metode, čiji se nazivi unutar šablona mogu koristiti kao da je reč o svojstvima. Tako je u primeru logika, koja je malopre bila direktno unutar šablona, prebačena unutar metode `capitalizedName()`. Unutar šablona je umesto izraza definisan naziv ovakve metode, a takav naziv će prilikom konstrukcije prezentacije da bude zamenjen povratnom vrednošću takve metode.

Vue direktive

Vue sistem za izgradnju šablona omogućava mnogo više od onoga što je prikazano u prethodnim redovima, pa je tako moguće obaviti uslovno generisanje HTML koda, pretplatu na događaje, dvosmerno povezivanje i još mnogo toga. Sve su to primeri za čiju realizaciju je neophodno koristiti Vue direktive.

Vue direktive su specijalni atributi koji se postavljaju na HTML elemente, a započinju prefiksom **v-**. Takav prefiks koristi se kako bi se Vue direktive razlikovale od regularnih HTML atributa.

Vue poseduje veliki broj direktiva sa kojima ćemo se mi upoznati u nastavku ove lekcije. Za početak, evo kako izgleda upotreba jedne direktive koja može da zameni interpolaciju:

```

<div id="container">
  <div v-text="name"></div>
  <div v-text="email"></div>
  <div v-text="balance"></div>
</div>

<script src="vue.js"></script>
<script>
  var user = { name: "Ben Torrance", email: "ben@email.com",
balance: 55353.93434 };

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

```

U primeru je upotrebljena direktiva **v-text**, koja je, baš kao i regularan HTML atribut, postavljena na HTML elementima. Vrednosti `v-text` direktiva u primeru su nazivi svojstava objekta koji predstavlja podatke. Na ovaj način se postiže identično što i u prethodnom primeru, upotrebom interpolacije.

Obratite pažnju na to da su i direktive u suštini potpuno legalne HTML tvorevine. Naime, reč je o HTML atributima, pa se tako još jednom potvrđuje činjenica da je kod Vue šablona potpuno validan HTML.

Direktive je moguće koristiti za obavljanje mnogih operacija prilikom kreiranja Vue šablona. Tako ćemo se u nastavku lekcije upoznati sa sledećim direktivama, koja će nam omogućiti da obavimo različite napredne operacije prilikom kreiranja Vue šablona.

- `v-once` – omogućava da se interpolacija obavi samo jednom,
- `v-html` – omogućava parsiranje HTML koda,
- `v-bind` – omogućava povezivanje vrednosti atributa,
- `v-model` – omogućava dvosmerno povezivanje,
- `v-if`, `v-else`, `v-else-if` – direktive za kreiranje kondicionala,
- `v-for` – direktiva za kreiranje petlji,
- `v-on` – direktiva za pretplatu na događaje,
- `v-show` – direktiva koja omogućava povezivanje sa `display` CSS svojstvom, pa samim tim i uticanje na vidljivost HTML elementa.

Kao što možete videti iz ovoga kratkog pregleda, direktive imaju vrlo široku upotrebu, pa je tako njih, sem za povezivanje, moguće koristiti i za brojne druge zahvate nad Vue šablonom. O tome će biti reči u nastavku ove lekcije.

v-once

Direktiva `v-once` omogućava da se obavi identično što i običnom interpolacijom ili upotrebom `v-text` direktive, ali samo jednom. To praktično znači da će podatak inicijalno biti renderovan unutar HTML-a, ali da povezivanje u nastavku neće funkcionisati. Tako se eventualne promene nad podacima neće propagirati do sloja prezentacije:

```

<div v-once>{{ balance }}</div>

```

Direktiva `v-once` se koristi u kombinaciji sa dvostrukim vitičastim zagradama, baš kao i u prikazanom primeru. Pri tome, `v-once` direktiva ne poseduje vrednost. Na ovaj način bilo koja promena vrednosti `balance` svojstva neće biti propagirana unutar HTML dokumenta.

v-html

Prilikom obavljanja interpolacije Vue tretira vrednosti svojstva kao običan tekst i na taj način ne dozvoljava parsiranje eventualnog HTML koda. Ovo je odlično ponašanje, s obzirom na to da je dinamičko renderovanje HTML koda veoma opasno i otvara vrata mogućim XSS napadima. Ipak, ukoliko se nekada iz opravdanih razloga javi potreba za dinamičkim umetanjem HTML koda iz pouzdanih izvora, može se koristiti direktiva `v-html`.

Podaci nad kojima će biti ilustrovana upotreba `v-html` direktive mogu da izgledaju ovako:

```
var user = {  
  name: "Ben Torrance",  
  email: "<i>ben@email.com</i>",  
  balance: 55353.93434  
};
```

Interpolacija upotrebom `v-text` direktive može da izgleda ovako:

```
<div v-text="email"></div>
```

U ovakvom slučaju se dobija sledeći ispis:

```
<i>ben@email.com</i>
```

Jasno je da se HTML kod tretira kao običan tekst.

Upotreba `v-html` direktive može da izgleda ovako:

```
<div v-html="email"></div>
```

Sada ispis na stranici izgleda kao na slici 5.1.

ben@email.com

Slika 5.1. HTML oznake su sada parsirane, pa je dobijen italic tekst

v-bind

Interpolaciju, odnosno dvostruke vitičaste zagrade, nije moguće koristiti za povezivanje vrednosti atributa. Za obavljanje takvog posla potrebno je koristiti direktivu `v-bind`:

```

```

`v-bind` direktiva definiše se kao prefiks atributa čiju vrednost je potrebno povezati sa nekim od svojstava objekta sa podacima. U primeru je to atribut `src`, `img` elementa, čime je obavljeno povezivanje njegove vrednosti sa svojstvom `thumb_url`:

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

Na ovaj način će unutar HTML dokumenta da bude dobijen sledeći HTML:

```

```

`v-bind` je jedna od najkorisnijih Vue direktiva, koju je moguće koristiti za obavljanje širokog spektra manipulacija nad Vue šablonom, o čemu će više reči biti u nastavku. Zbog svoje svestranosti, `v-bind` direktivu je moguće definisati i u skraćenom obliku:

```

```

Sada je `v-bind` direktiva definisana u skraćenom obliku, koji podrazumeva korišćenje karaktera dve tačke (:) ispred naziva HTML atributa.

v-model

Svi primeri do sada ilustrovali su postizanje povezivanja kod koga su promene propagirane od modela kao sloja prezentacije. Takvo povezivanje se drugačije naziva jednosmerno povezivanje (engl. **one-way binding**). Ipak, u nekim situacijama može se javiti potreba za postizanjem dvosmernog povezivanja (engl. **two-way binding**).

Dvosmerno povezivanje je ono kod koga se promene propagiraju u dva smera, odnosno od sloja modela ka sloju prezentacije, ali i od sloja prezentacije nazad do sloja modela. Stoga dvosmerno povezivanje zahteva postojanje elemenata koji korisniku omogućavaju unos podataka:

```
<div id="container">
  <input v-model="name" placeholder="enter your name...">

  <label for="favourite-color">Choose your favourite
color:</label>
  <select v-model="color" id="favourite-color">
    <option>Maroon</option>
    <option>Blue</option>
    <option>Yellow</option>
    <option>Magenta</option>
  </select>

  <div>
    Your name: {{ name }}
  </div>
</div>
```

```

    </div>
    <div>
      Your favourite color: {{ color }}
    </div>
  </div>

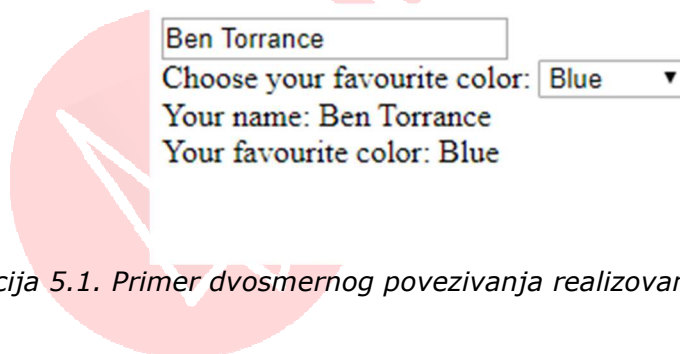
  <script src="vue.js"></script>
  <script>
    var data = {
      name: "Ben Torrance",
      color: "Blue"
    };

    var vm = new Vue({
      el: '#container',
      data: data
    })
  </script>

```

Primer ilustruje dvosmerno povezivanje koje se postiže upotrebom `v-model` direktive. Za realizaciju primera korišćeni su `input` i `select` elementi, koji su povezani sa `name` i `color` svojstvima objekta modela. Inicijalnim otvaranjem ovakvog HTML dokumenta može se videti da se `input` i `select` elementi popunjavaju podacima na osnovu vrednosti svojstava objekta modela. To praktično znači da podaci putuju od modela ka prezentaciji.

Kako bi se prikazalo da se promene nad podacima propagiraju i od sloja prezentacije do modela, unutar Vue šablona su postavljena i dva `div` elementa čiji je sadržaj definisan osnovnim načinom za obavljanje interpolacije. Promenom vrednosti unutar `input` elementa ili promenom selekcije `select` elementa moguće je videti da se promene na odgovarajući način odslikavaju i na takvim elementima (animacija 5.1).



Animacija 5.1. Primer dvosmernog povezivanja realizovanog korišćenjem Vuea

Direktive za kreiranje kondicionala

U Vueu je direktive moguće koristiti i za kondicionalno renderovanje HTML koda unutar Vue šablona. Direktive koje omogućavaju obavljanje kondicionalnog renderovanja su:

- `v-if`
- `v-else`
- `v-else-if`

Korišćenje `v-if` direktive može da izgleda ovako:

```
<div id="container">
  <input type="checkbox" id="age" v-model="adult"> <label
for="age">I am over 18 years old.</label>

  <div v-if="adult">
    Your are over 18 years old.
  </div>
</div>

<script src="vue.js"></script>
<script>
  var data = {
    name: "Ben Torrance",
    adult: false
  };

  var vm = new Vue({
    el: '#container',
    data: data
  })
</script>
```

Prikazani primer poseduje jedan checkbox input element, koji je korišćenjem `v-model` direktive dvosmerno povezan sa `adult` svojstvom objekta podataka. Ispod takvog checkbox elementa nalazi se i jedan `div` element koji se unutar DOM strukture prikazuje samo ukoliko je vrednost svojstva `adult` `true`. To je postignuto korišćenjem `v-if` direktive.

☐ I am over 18 years old.

Animacija 5.2. Uslovno renderovanje korišćenjem Vuea

Nakon `v-if` naredbe, moguće je definisati i `v-else` naredbu:

```
<div v-if="adult">
  Your are over 18 years old.
</div>
<div v-else>Oh no 😞, you are under 18.</div>
```

Element sa `v-else` direktivom mora se naći odmah nakon elementa sa direktivom `v-if`.

Direktiva `v-else-if` može se koristiti ukoliko je potrebno kreirati višestruke uslove. S obzirom na to da je u prikazanom primeru kontrolna promenljiva `boolean` tipa, ona može imati samo dve vrednosti. Stoga je neophodno kreirati nešto drugačiji primer, koji će omogućiti definisanje višestrukih uslova:

```

<div id="container">
  <input type="text" v-model="age" placeholder="enter your
age...">

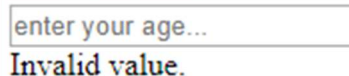
  <div v-if="age > 0 && age < 18">
    You are adolescent.
  </div>
  <div v-else-if="age >= 18 && age < 65">
    You are adult.
  </div>
  <div v-else-if="age >= 65">
    You are senior.
  </div>
  <div v-else>Invalid value.</div>
</div>

<script src="vue.js"></script>
<script>
  var data = {
    name: "Ben Torrance",
    age: undefined
  };

  var vm = new Vue({
    el: '#container',
    data: data
  })
</script>

```

Sada je kontrolno svojstvo (age) numeričkog tipa. Unutar šablona se nalazi jedan `input` element koji je korišćenjem `v-model` direktive povezan sa svojstvom `age`. U nastavku je formirano nekoliko uslovnih blokova, korišćenjem direktiva `v-if`, `v-else` i `v-else-if`. Na taj način obavlja se kontrola prikaza `div` elemenata, u zavisnosti od vrednosti `age` svojstva. Ukoliko `age` ima vrednost između 0 i 18, prikazuje se poruka *You are adolescent*. Kada `age` ima vrednost između 18 i 65, dobija se poruka *You are adult*. Za vrednosti preko 65, poruka je *You are senior*. Na kraju, za sve vrednosti koje nisu numeričke ili su manje od nule (0), prikazuje se `div` element na kojem se nalazi `v-else` direktiva (animacija 5.3).



Animacija 5.3. Višestruko uslovno renderovanje korišćenjem Vuea

Vue poseduje još jednu direktivu koju je moguće koristiti za uslovno prikazivanje HTML elemenata. Reč je o direktivi **v-show**. Za razliku od upravo prikazanih direktiva, `v-show` ne omogućava uslovno renderovanje, već samo uslovno prikazivanje. Naime, element sa `v-show` direktivom uvek postoji unutar HTML strukture dokumenta, dok se njegova vidljivost kontroliše promenom vrednosti `display` svojstva. Direktive iz prvog dela ovoga poglavlja

funkcionišu po drugačijem principu, tako što elemente dodaju i uklanjaju iz HTML strukture dokumenta.

Primer korišćenja `v-show` direktive može da izgleda ovako:

```
<div id="container">
  <input type="checkbox" id="age" v-model="adult"> <label
for="age">I am over 18 years old.</label>

  <div v-show="adult">
    Your are over 18 years old.
  </div>
</div>

<script src="vue.js"></script>
<script>
  var data = {
    name: "Ben Torrance",
    adult: false
  };

  var vm = new Vue({
    el: '#container',
    data: data
  })
</script>
```

Ovo je sada primer identičan onom sa početka priče o Vue kondicionalnima. Jedina razlika je upotreba `v-show` direktive umesto direktive `v-if`. Na prvi pogled sve će izgledati identično kao i prilikom upotrebe `v-if` direktive. Ipak, unutrašnji način funkcionisanja primera je sada drugačiji, pa ukoliko pratite kôd HTML dokumenta, moći ćete da vidite da je `div` element sa porukom uvek prisutan unutar takve strukture. Menja se samo vrednost njegovog CSS svojstva `display`.

Kreiranje petlji

Vue omogućava da se prilikom kreiranja šablona definišu petlje, koje su vrlo korisne kada je potrebno obaviti generisanje lista ili tabela, odnosno kada je podatke predstavljene nizovima potrebno prikazati unutar HTML dokumenta.

Direktiva za kreiranje petlji je **`v-for`**. Ona se definiše u posebnom formatu:

```
<div id="container">
  <ul>
    <li v-for="item in items">{{ item }}</li>
  </ul>
</div>

<script src="vue.js"></script>
<script>
  var data = { items : ["black", "blue", "green", "yellow"]};

  var vm = new Vue({
    el: '#container',
```

```

        data: data
      })
    </script>

```

Kôd ilustruje prvi primer generisanja HTML koda korišćenjem `v-for` direktive. Bitno je primetiti da vrednost `v-for` direktive podseća na `for...in` JavaScript petlju. Tako se `items` odnosi na naziv svojstva koje ukazuje na niz, dok `item` predstavlja promenljivu koja se u svakoj iteraciji popunjava narednim elementom niza. Na kraju se unutar web pregledača dobija ispis kao na slici 5.2.

- black
- blue
- green
- yellow

Slika 5.2. Primer liste generisane korišćenjem v-for direktive

`v-for` direktiva omogućava dolazak do indeksa elementa u svakoj iteraciji:

```

<li v-for="(item, index) in items">{{ index }} : {{ item }}</li>

```

Bitno je primetiti da su sada definisane dve promenljive `item` i `index` i njihove vrednosti se ispisuju unutar stavki jedne liste (slika 5.3).

- 0 : black
- 1 : blue
- 2 : green
- 3 : yellow

Slika 5.3. Primer liste generisane korišćenjem v-for direktive (2)

Direktivu `v-for` je moguće koristiti i za prolazak kroz svojstva običnih JavaScript objekata:

```

<div id="container">
  <p v-for="(value, name, index) in user">{{ index }}. {{
name }}: {{ value }}</p>
</div>

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

  var data = {
    user: {
      name: "Ben Torrance",
      email: "ben@email.com",
      balance: 55353.93434
    }
  };

  var vm = new Vue({

```

```

        el: '#container',
        data: data
    })
</script>

```

Podaci su sada definisani u obliku objekta sa jednim svojstvom – `user`. Svojstvo `user` ukazuje na novi objekat koji poseduje tri svojstva. Nešto ranije u ovoj lekciji su ovakvi podaci korisnika unutar šablona ispisivani pojedinačno, korišćenjem osnovnih principa interpolacije. Ipak, ovoga puta je za generisanje HTML koda iskorišćena `v-for` direktiva sa tri parametra:

- `value` – vrednost svojstva,
- `name` – naziv svojstva,
- `index` – indeks svojstva, koji kao i kod nizova započinje od nule.

Na ovaj način se unutar web pregledača dobija ispis kao na slici 5.4.

```

0. name: Ben Torrance
1. email: ben@email.com
2. balance: 55353.93434

```

Slika 5.4. Prikaz svojstava objekta dobijenih korišćenjem `v-for` direktive

Pretplata na DOM događaje

Još jedan aspekt razvoja prezentacionog dela web aplikacija, koji Vue znatno olakšava, jeste obrada događaja. DOM događaje je moguće obrađivati korišćenjem **`v-on`** direktive.

Vrednost `v-on` direktive može biti jednostavan JavaScript izraz koji će biti izvršen kada dođe do pojave nekog događaja. Takav pristup sličan je linijskoj obradi događaja korišćenjem čistog JavaScripta. Evo kako može da izgleda jedan takav primer:

```

<div id="container">
  <button v-on:click="code = Math.random()">Generate
code</button>
  <p>{{ code }}</p>
</div>

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

    var vm = new Vue({
      el: '#container',
      data: {
        code: 0
      }
    })

</script>

```

U primeru je `v-on` direktiva upotrebljena za pretplatu na `click` događaj na jedan `button` element. Iz primera možete videti da se na `v-on` direktiva dodaje na naziv događaja koji se sluša, pri čemu se `v-on` i naziv događaja razdvajaju karakterom dve tačke (:). Svakim klikom na `button` element iz primera biće aktiviran kod koji je definisan kao vrednost `v-on` direktive. Reč je o kodu kojim se postavljaju vrednosti `code` svojstva korišćenjem vrednosti koja je generisana upotrebom `Math.random()` metode.

Iako u nekim situacijama definisanje JavaScript izraza kao vrednosti `v-on` direktive može biti pogodno (ukoliko je reč o jednostavnoj logici), u većini situacija logika za obradu događaja će biti znatno obimnija. Stoga je korišćenjem `v-on` direktive moguće obaviti referenciranje metoda koje se definišu unutar Vue instance:

```
<div id="container">
  <button v-on:click="sayHello">Say Hello</button>
</div>

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

  var vm = new Vue({
    el: '#container',
    data: {
      code: 0
    },
    methods: {
      sayHello: function(event){
        alert("Hello World");
      }
    }
  })
</script>
```

Vrednost `v-on` direktive sada je naziv metode (`sayHello`) koja je definisana unutar Vue instance. Metode se unutar Vue instance definišu unutar objekta koji se priključuje **methods** svojstvu.

Napomena

Unutar metoda koje se definišu u Vue instanci, promenljiva `this` ukazuje na Vue instancu, dok je parametar (u primeru `event`) identičan regularnim objektima kojima se predstavljaju DOM događaji. To praktično znači da je u prethodnom primeru korišćenjem promenljive `event` moguće pristupiti svim uobičajenim svojstvima i metodama na koje smo navikli – `event.preventDefault()`, `event.stopPropagation()`, `event.target...`

Sada smo prvi put unutar Vue instance, pored podataka, definisali i ponašanja, odnosno metode. Kao što vidite, svojstvo `data` ukazuje na podatke, a svojstvo `methods` na metode. Bitno je znati da metode koje se definišu kao vrednost svojstva `methods` ne moraju da budu korišćene isključivo za obradu događaja. Jednostavno, podaci i metode koji se definišu svojstvima `data` i `methods` automatski postaju svojstva i metode Vue objekta. Stoga je metodu `sayHello()` iz prethodnog primera moguće pozvati i na sledeći način:

```
vm.sayHello();
```

Vue omogućava da se, prilikom poziva metoda za obradu događaja, takvim metodama prosledi i neki parametar, i to direktno iz same `v-on` direktive:

```
<div id="container">
  <button v-on:click="sayHello('Ben', $event)">Say Hello to
Ben</button>
  <button v-on:click="sayHello('John', $event)">Say Hello to
John</button>
</div>

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

  var vm = new Vue({
    el: '#container',
    data: {
      code: 0
    },
    methods: {
      sayHello: function(name, event){
        alert("Hello " + name);
      }
    }
  })
</script>
```

Sada su kreirana dva `button` elementa kojima se aktivira identična metoda (`sayHello`). Ipak, metoda je sada modifikovana tako da, pored `event` parametra, prihvata i `name` parametar. Oba parametra se prosleđuju direktno unutar `v-on` direktive.

Na kraju Vue omogućava i da se na veoma lak način obave neke uobičajene modifikacije događaja. Na primer, prilikom obrade formi veoma često se obavlja otkazivanje podrazumevanog ponašanja pozivanjem metode `preventDefault()`. Takve i slične modifikacije događaja moguće je obaviti unutar same `v-on` direktive korišćenjem sledećih dodataka:

- `.stop` – zaustavlja propagiranje događaja,
- `.prevent` – sprečava podrazumevano osvežavanje stranice prilikom prosleđivanja forme ili klika na link,
- `.capture` – obrađuje događaj u capture fazi umesto podrazumevanog ponašanja po kome se događaj obrađuje u bubbling fazi,
- `.self` – čini da događaj bude obrađen samo ukoliko je inicijator događaja sam element na kome je izvršena pretplata,
- `.once` – čini da događaj bude obrađen samo jednom,
- `.passive` – omogućava kreiranje pasivnih funkcija za obradu događaja koje povećavaju performanse prilikom skrolovanja, oslobađanjem web pregledača od

potrebe za čekanjem logike koja je pridružena `touchstart` ili `touchmove` događajima.

Na primer, kako bi se sprečilo podrazumevano ponašanje koje se sastoji od prosleđivanja forme od web pregledača i osvežavanja stranice, dovoljno je napisati:

```
<form v-on:submit.prevent="onSubmit"></form>
```

Rukovanje stilizacijom

Nešto ranije je predstavljena `v-bind` direktiva i tada je rečeno da je u pitanju jedna od najsvestranijih Vue direktiva. S obzirom na to da se koristi za definisanje vrednosti atributa, unutar Vue šablona nju je moguće koristiti i prilikom rada sa klasama i linijskom stilizacijom (oba segmenta se definišu odgovarajućim HTML atributima – `class` i `style`, pa potpadaju pod radar `v-bind` direktive).

Evo primera najjednostavnijeg korišćenja `v-bind` direktive za uticanje na klase koje su prisutne na jednom HTML elementu:

```
<div id="container">
  <div v-bind:class="{ completed: completed }"></div>
</div>

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

  var vm = new Vue({
    el: '#container',
    data: {
      completed: true
    }
  })
</script>
```

Na `div` elementu postavljena je `v-bind` direktiva kojom se utiče na prisutnost klase `.completed`. Ukoliko je vrednosti svojstva `completed` `true`, kao što je to u primeru, na `div` elementu će biti prisutna `.completed` klasa. Tako će prikazani primer da proizvede `div` element koji će imati sledeću strukturu:

```
<div class="completed"></div>
```

Kada svojstvo `completed` ima vrednost `false`, klase `.completed` neće biti na elementu:

```
<div class=""></div>
```

Više svojstava je moguće objediniti unutar zasebnog objekta, a onda se takav objekat može povezati sa `class` atributom, korišćenjem `v-bind` direktive:


```

<div id="container">
  <div v-bind:class="classes"></div>
</div>

<script src="vue.js"></script>
<script>
  var vm = new Vue({
    el: '#container',
    data: {
      classes: {
        completed: true,
        deleted: false
      }
    }
  })
</script>

```

Sada je korišćenjem `v-bind` direktive `class` atribut povezan sa jednim objektom (`classes`). Takav objekat poseduje dva svojstva boolean tipa – `completed` i `deleted`. Nazivi svojstava sa `true` vrednošću biće postavljeni kao vrednosti klasa na `div` elementu. Tako će prikazani primer stvoriti ovakav element:

```
<div class="completed"></div>
```

Kada oba svojstva imaju vrednost `true`, `div` element će izgledati ovako:

```
<div class="completed deleted"></div>
```

Napomena

Obratite pažnju na imenovanje svojstava prilikom njihovog korišćenja za definisanje klasa. Na primer, ustaljena praksa za definisanje klasa koje se sastoje iz više reči jeste njihovo razdvajanje karakterom srednja crta (-). Takav karakter se ne može upotrebiti unutar JavaScript identifikatora, ali zato JavaScript jezik omogućava definisanje naziva svojstava u `string` obliku:

```

<div id="container">
  <div v-bind:class="classes"></div>
</div>
<script src="vue.js"></script>
<script>
  var vm = new Vue({
    el: '#container',
    data: {
      classes: {
        'is-completed': true,
        deleted: true
      }
    }
  })
</script>

```

Jedno od svojstava je sada izmenjeno, tako da ima naziv 'is-completed'. S obzirom na to da se koristi karakter srednja crta, naziv je postavljen između navodnika.

Na ovaj način će da bude dobijen ovakav `div` element:

```
<div class="is-completed deleted"></div>
```

Prilikom definisanja klasa korišćenjem `v-bind` direktive, njoj je moguće proslediti i niz svojstava, kao u sledećem primeru:

```
<div id="container">
  <div v-bind:class="[ completedClass, deletedClass ]"></div>
</div>

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

  var vm = new Vue({
    el: '#container',
    data: {
      completedClass: "completed",
      deletedClass: "deleted"
    }
  })
</script>
```

Sada će na `div` elementu da budu postavljene klase koje odgovaraju vrednostima svojstava `completedClass` i `deletedClass`, pa će `div` element da izgleda ovako:

```
<div class="completed deleted"></div>
```

Identične načine je moguće upotrebiti i prilikom definisanja vrednosti `style` atributa. Najefektniji način jeste korišćenje objekta sa svojstvima čiji nazivi odgovaraju nazivima CSS svojstava:

```
<div id="container">
  <div v-bind:style="styleObj"></div>
</div>

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

  var vm = new Vue({
    el: '#container',
    data: {
      styleObj: {
        color: 'blue',
        'font-size': '14px'
      }
    }
  })
</script>
```

Korišćenjem `v-bind` direktive u primeru je definisana vrednost `style` atributa. `style` atribut je povezan sa `styleObj` objektom koji poseduje dva svojstva – `color` i `font-size`. Zato što se sastoji iz dve reči, naziv `font-size` je smešten ispod navodnika. Na kraju, na ovaj način se dobija ovakav `div` element:

```
<div style="color: blue; font-size: 14px;"></div>
```

Rezime

- Šablon (engl. *template*) je naziv koji se koristi za kod kojim se kreira prezentacioni sloj aplikacije.
- Svaki Vue šablon ujedno je i validan HTML kod, koji se može parsirati od bilo kog web pregledača.
- Dva osnovna specifična pojma u Vueu, koja je moguće koristiti prilikom izgradnje šablona, jesu interpolacija i direktive.
- Interpolacija se odnosi na proces postavljanja vrednosti promenljivih na unapred definisana rezervisana mesta.
- Vue interpolacija se postiže korišćenjem posebne sintakse, koja podrazumeva upotrebu dva para vitičastih zagrada.
- Povezivanje korišćenjem interpolacije se zbog specifične sintakse unutar Vuea veoma često drugačije naziva *mustache binding*.
- Izračunata svojstva (engl. *computed properties*) omogućavaju da se definiše proizvoljna logika u obliku JavaScript funkcije, čija će povratna vrednost moći lako da se interpolira unutar Vue šablona.
- Izračunata svojstva definišu se kao funkcije unutar objekta koji se dodaje svojstvu `computed`.
- Vue direktive su specijalni atributi koji se postavljaju na HTML elemente, a započinju prefiksom `v-`.
 - Direktiva za obavljanje jednostavne interpolacije je `v-text`.
- Direktiva `v-once` omogućava ispisivanje vrednosti svojstva, ali bez propagiranja promena koje mogu nastati u budućnosti.
- Direktiva `v-html` omogućava da se prilikom interpolacije obavi parsiranje HTML koda koji je sadržan unutar vrednosti svojstva.
- Za definisanje vrednosti atributa, moguće je koristiti direktivu `v-bind`, ili njen skraćeni oblik – `:`
- Dvosmerno povezivanje se može postići korišćenjem direktive `v-model`.
- Uslovno renderovanje HTML koda je moguće postići direktivama `v-if`, `v-else`, `v-else-if`.
- Generisanje HTML koda korišćenjem petlje moguće je obaviti upotrebom direktive `v-for`.
- Pretplata na DOM događaje se može obaviti korišćenjem direktive `v-on`.
- Korišćenjem direktive `v-show` moguće je uticati na vrednosti `display` CSS svojstva.