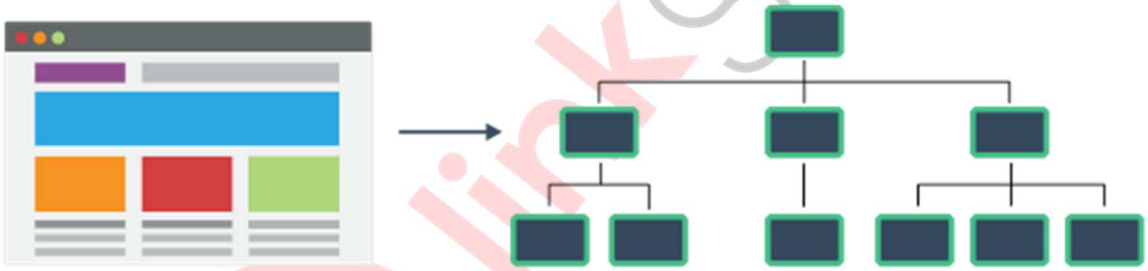
Vue komponente

Korisnička okruženja web aplikacija moguće je apstrahovano posmatrati kao skup manjih jedinica. Takve manje jedinice, koje čine korisničko okruženje jedne aplikacije, drugačije se nazivaju komponente. I sam Vue poznaje pojam komponenata, pa je tako komponentni sistem jedan od najznačajnijih koncepata u Vueu, koji omogućava efikasno kreiranje prezentacione logike web aplikacija. Lekcija pred vama biće posvećena kreiranju i korišćenju Vue komponenata.

Šta su Vue komponente?

Vue komponente su samostalne, nezavisne jedinice, kojima se predstavlja jedan deo korisničkog okruženja web aplikacija. Naime, web aplikacije su veoma često sačinjene iz određenih elemenata ili sekcija koje su zajedničke za veći broj prikaza, odnosno stranica. Na primer, takve sekcije mogu biti navigacija, statusna traka, zaglavlje ili traka sa alatima. Reč je o sekcijama koje poseduje većina web aplikacija, a koje je najbolje realizovati kao samostalne, nezavisne jedinice, što omogućava da se one umetnu na sva ona mesta na kojima se za njihovim prisustvom javi potreba.



Slika 6.1. Vue sistem komponenata

Kako se kreiraju Vue komponente?

Vue komponente su Vue instance koje se kreiraju korišćenjem `Vue.component()` metode:

```
Vue.component('my-component', {
        template: '<div>Hello from component.</div>'
})
```

Na ovaj način je obavljeno kreiranje prve Vue komponente. Možete videti da `Vue.component()` metoda prihvata dva parametra:

- prvi parametar se odnosi na naziv komponente,
- drugi parametar je objekat za konfigurisanje.

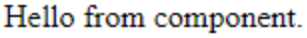
Konfigurisanje je u primeru obavljeno upotrebom samo jednog svojstva – `template`. Svojstvo **template** koristi se za definisanje šablona komponente.

Na ovaj način smo došli do prve razlike između Vue instance i Vue komponenata. Prilikom kreiranja Vue komponenata ne koristi se svojstvo `el`, već se kompletan šablon definiše kao vrednost specijalnog svojstva `template`.

Kako biste ovako kreiranu komponentu prikazali unutar šablona Vue aplikacije, dovoljno je upotrebiti njen naziv, koji je definisan prilikom kreiranja:

```
<div id="container">
    <my-component></my-component>
</div>

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

    Vue.component('my-component', {
        template: '<div>Hello from component.</div>'
    })

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

</script>
```

Komponenta se smešta unutar šablona, definisanjem HTML elementa čiji naziv tagova odgovara nazivu komponente. Prilikom generisanja HTML dokumenta, Vue će `<my-component></my-component>` element da zameni sadržajem ove komponente, odnosno elementom `div` sa sadržajem *Hello from component,* pa će unutar web pregledača da bude dobijen efekat kao na slici 6.2.



Slika 6.2. Tekst koji potiče iz Vue komponente

Napomena

Vue komponente se mogu koristiti samo unutar šablona Vue instanci. Zbog toga je u primeru obavljeno kreiranje Vue instance, a za koreni element njenog šablona je definisan element sa id-jem `container`. To na kraju praktično znači da se kreirana komponenta mora naći unutar `container` elementa. Ukoliko bi se navela izvan takvog elementa, Vue ne bi mogao da je prikaže.

Komponente sa sopstvenim podacima

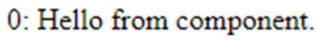
Kao što je već rečeno, Vue komponente su zapravo Vue instance koje se kreiraju na nešto drugačiji način. I instance i komponente prihvataju objekat za konfigurisanje, a svojstva koja je moguće koristiti za konfigurisanje gotovo su identična. Jedini izuzetak jeste svojstvo `el`, koje nije moguće koristiti prilikom konfigurisanja komponenata. Sve ovo praktično znači da je i unutar jedne Vue komponente moguće definisati podatke upotrebom `data` svojstva:

```
<div id="container">
    <my-component></my-component>
</div>

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

    Vue.component('my-component', {
        data: function () {
            return {
                counter: 0
            }
        },
        template: '<div v-on:click="counter++">{{ counter }}:
Hello from component.</div>'
    })

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

</script>
```

Primer je sada modifikovan na nekoliko načina. Prvo, prilikom definisanja komponente, unutar objekta za konfigurisanje je navedeno i svojstvo `data`, kojim je kao lokalni podatak naše komponente definisan jedan objekat sa svojstvom `counter` i vrednošću `0`.

I šablon je sada delimično izmenjen. Dodata je pretplata na `click` događaj, pri čemu je logika JavaScripta definisana u linijskom maniru, odnosno kao direktna vrednost `v-on` direktive. Definisanom logikom je rečeno da će svakim klikom na `div` element komponente vrednost `counter` promenljive da bude uvećavana za jedan.

Na kraju, unutar šablona je obavljena i interpolacija vrednosti promenljive `counter`, tako da se na kraju na ovaj način dobija ponašanje ilustrovano animacijom 6.1.



Animacija 6.1. Komponenta sa lokalnim podacima

Unutar upravo prikazanog primera bitno je da primetite da se podaci ne definišu kao što je to činjeno do sada, unutar Vue instance. Podaci, odnosno **vrednost promenljive `data`, unutar komponenata se definiše u obliku funkcije**. Povratna vrednost takve funkcije jesu podaci koje će koristiti komponente:

```
data: function () {
      return {
            counter: 0
      }
}
```

Podaci se unutar komponenata moraju definisati na ovaj način kako bi svaka komponenta posedovala sopstvenu instancu podataka. Ukoliko se podaci definišu kao objekat, Vue jasno stavlja do znanja da unutar komponenata podaci moraju biti definisani u obliku funkcije:

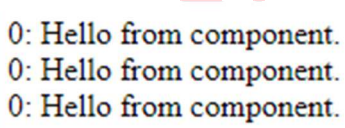
```
[Vue warn]: The "data" option should be a function that returns a per-
instance value in component definitions.
```

Sve ovo omogućava jednu od osnovnih osobina Vue komponenata – ponovnu upotrebljivost:

```
<div id="container">
    <my-component></my-component>
    <my-component></my-component>
    <my-component></my-component>
</div>
```

Sada je unutar šablona Vue aplikacije obavljeno kreiranje 3 `my-component` komponente. Tako primer ilustruje mogućnost ponovne upotrebljivosti Vue komponenata (*jednu komponentu je moguće upotrebiti proizvoljan broj puta*).

S obzirom na to da je svaka komponenta nezavisna celina, svaka će imati sopstvenu instancu podataka (animacija 6.2).



Animacija 6.2. Svaka Vue komponenta poseduje sopstvenu instancu podataka

Pitanje

Jednu Vue komponentu je moguće upotrebiti samo jednom.

a) Tačno
b) Netačno

Objašnjenje:

Jedna od osnovnih osobina Vue komponenata jeste mogućnost ponovne upotrebljivosti. Tako je jednom kreiranu komponentu moguće upotrebiti proizvoljan broj puta.

Prosleđivanje podataka komponentama

U realnim okolnostima, komponente koje poseduju isključivo lokalne podatke imaju veoma malu upotrebnu vrednost. Stoga se u realnim okolnostima podaci najčešće *ubrizgavaju* unutar komponenata iz spoljašnjeg okruženja.

Kako bi se omogućilo da jedna komponenta dobije podatke, prvo je neophodno obaviti registrovanje svojstava (engl. *properties*). To se postiže upotrebom svojstva **props** unutar objekta za konfigurisanje Vue komponente:

```
Vue.component('my-component', {
    props: ['name'],
    template: '<div>Hello {{ name }}.</div>'
})
```

Sada je prilikom kreiranja komponente obavljeno definisanje vrednosti `props` svojstva. Njime je definisan jedan atribut sa nazivom `name`. Kada se ovakav atribut popuni vrednošću izvan komponente, on automatski postaje objektno svojstvo Vue komponente.

Nakon definisanja vrednosti `props` svojstva, osnovni način na koji je moguće obaviti prosleđivanje podataka jeste njihovo definisanje unutar šablona aplikacije, upotrebom sintakse koja podseća na onu za definisanje atributa:

```
<div id="container">
    <my-component name="Ben"></my-component>
    <my-component name="John"></my-component>
    <my-component name="Tom"></my-component>
</div>

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

    Vue.component('my-component', {
        props: ['name'],
        template: '<div>Hello {{ name }}.</div>'
    })

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

</script>
```

Na ovaj način, unutar web pregledača se dobija ispis kao na slici 6.3.

Hello Ben.
Hello John.
Hello Tom.

Slika 6.3. Jedna Vue komponenta upotrebljena tri puta sa različitim ulaznim parametrima

Ipak, u realnim uslovima, podaci će najčešće postojati unutar objekta koji se pridružuje `data` svojstvu Vue instance:

```
var vm = new Vue({
    el: '#container',
    data: {
        users: [
            { id: 1, name: "Ben" },
            { id: 2, name: "John" },
            { id: 3, name: "Tom" }
        ]
    }
})
```

Sada su podaci definisani unutar `data` svojstva instance Vue, i to u obliku niza objekata, pri čemu svaki objekat predstavlja po jednu osobu sa `id` i `name` svojstvima.

Kao što je rečeno, u realnim okolnostima podaci se najčešće koriste upravo u ovakvom obliku, a ukoliko želite da ih prikažete komponentom koju smo kreirali u prethodnim redovima, neophodno je iskoristiti nekoliko direktiva koje smo upoznali u prethodnoj lekciji. Za svaki objekat unutar niza kojim se predstavljaju osobe potrebno je kreirati po jednu komponentu i njoj proslediti odgovarajuće podatke (`name` svojstva). To se može obaviti na sledeći način:

```
<div id="container">
    <my-component v-for="user in users" v-bind:key="user.id" v-
bind:name="user.name"></my-component>
</div>

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

    Vue.component('my-component', {
        props: ['name'],
        template: '<div>Hello {{ name }}.</div>'
    })

    var vm = new Vue({
        el: '#container',
        data: {
            users: [
                { id: 1, name: "Ben" },
                { id: 2, name: "John" },
                { id: 3, name: "Tom" }
            ]
        }
    })

</script>
```

Primer ilustruje dinamičko renderovanje HTML elemenata korišćenjem `v-for` direktive i naše, odnosno korisnički definisane Vue komponente `my-component`. Ovakav pristup veoma je koristan kada je unutar HTML dokumenta potrebno prikazati određeni skup podataka, čija količina nije unapred poznata.

Definisanje ključeva prilikom upotrebe v-for direktive

U upravo prikazanom primeru prvi put je iskorišćen jedan specijalni atribut – **key**. Reč je o atributu kojim je moguće definisati jedinstveni identifikator čvora unutar Vue šablona. On pomaže Vue sistemu da jednoznačno identifikuje svaki čvor, što može biti vrlo korisno u situacijama u kojima dolazi do promena DOM strukture. Naime, Vue podrazumevano koristi algoritam koji se maksimalno trudi da smanji pomeranje elemenata, odnosno njihovo uništavanje i ponovno kreiranje. Stoga se često dešava da se, u slučaju promena DOM strukture, jedan element iskoristi za kreiranje nekog drugog, kako bi se poboljšale performanse. U većini situacija, podrazumevano ponašanje je odgovarajuće, ipak, prilikom rada sa komponentama dobro je u kombinaciji sa `v-for` direktivom definisati i vrednost specijalnog atributa `key`.

Lokalne i globalne komponente

Pre nego što nastavimo da upoznajemo ostale osobine komponenata, bitno je reći da komponente mogu biti globalne ili lokalne. Sve komponente kreirane u dosadašnjem toku ove lekcije bile su globalne.

Globalne komponente su one koje se kreiraju korišćenjem metode `Vue.component()`:

```
Vue.component('my-component-name', {
      // ... options ...
})
```

Komponente koje se kreiraju na ovaj način dostupne su svuda, pa se upravo zbog toga i nazivaju globalne. Njih je moguće koristiti bilo gde unutar šablona Vue instance, ali takođe i unutar bilo koje druge komponente.

Napomena

Globalne komponente je moguće registrovati samo ukoliko se poziv metode `Vue.component()` obavi pre kreiranja Vue instance.

Lokalne komponente se kreiraju na nešto drugačiji način, koji ne podrazumeva upotrebu metode `Vue.component()`. One se definišu samo korišćenjem objekta za podešavanje, koji je potrebno dodeliti nekoj promenljivoj:

```
const MyLocalComponent = {
    template: '<div>Hello from local component.</div>'
}
```

Evo jednog objekta koji je dodeljen promenljivoj `MyLocalComponent`, a koji će biti iskorišćen za kreiranje lokalne komponente. Za razliku od globalnih, lokalne komponente je neophodno eksplicitno uključiti unutar Vue instance ili neke druge komponente, korišćenjem specijalnog svojstva **components**:

```
<div id="container">
    <my-local-component></my-local-component>
</div>

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

    const MyLocalComponent = {
        template: '<div>Hello from local component.</div>'
    }

    var vm = new Vue({
        el: '#container',
        components: {
            'my-local-component': MyLocalComponent
        }
    })

</script>
```

Prikazani kôd ilustruje kompletan primer kreiranja i uključivanja lokalne komponente unutar Vue instance.

Šabloni komponenata moraju imati jedan koreni element

Do sada je već prikazano da se šabloni komponenata definišu kao vrednost `template` svojstva. Ipak, šabloni svih do sada kreiranih komponenata bili su veoma jednostavni i sačinjeni iz samo jednog HTML elementa. U praksi se šabloni komponenata uglavnom sastoje iz većeg broja elemenata. Tada je bitno znati da se svi elementi moraju grupisati unutar jednog korenog elementa.

Evo jednog primera komponente čiji se šablon sastoji iz većeg broja elemenata:

```
 <div id="container">
    <user-component v-bind:user="user"></user-component>
</div>

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

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

```
    Vue.component('user-component', {
        props: ['user'],
        template: `
            <div class="user-container">
                <h3>{{ user.name }}</h3>
                <p v-text='user.email'></p>
                <p v-text='user.balance'></p>
            </div>
            `
    })

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

</script>
```

Template literals

Upravo prikazani primer ilustruje jednu posebnu funkcionalnost JavaScript jezika koja je uvedena sa pojavom ES2015 specifikacije. Reč je šablonskim literalima. Takva funkcionalnost, između ostalog, omogućava definisanje `string` vrednosti u više redova, bez potrebe za korišćenjem karaktera plus (+) za obavljanje nadovezivanja stringova. Naime, u primeru možete videti da je šablon komponente definisan u više redova, bez ikakvih karaktera za nadovezivanje. Sve je to omogućeno korišćenjem specijalnih navodnika za oivičavanje `string` vrednosti - ``.

Slušanje događaja komponenata

U nekim situacijama može se javiti potreba za slušanjem događaja do kojih može doći na elementima koji pripadaju jednoj komponenti. Događaji se mogu slušati unutar komponenata, ali i izvan njih.

Slušanje događaja unutar komponente postiže se korišćenjem direktive `v-on` u kombinaciji sa svojstvom `methods`, unutar koga se može definisati funkcija za obradu događaja:

```
Vue.component('user-component', {
    props: ['user'],
    template: `<div class="user-container">
                <h3>{{ user.name }}</h3>
                <p v-text='user.email'></p>
                <p v-text='user.balance'></p>
                <button v-on:click="sayHello">Say Hello</button>
            </div>`,
    methods: {
        sayHello: function (event) {
            alert(`Hello ${this.user.name}`);
        }
    }
})
```

Sada je unutar šablona komponente dodat još jedan HTML element. Reč je o `button` elementu, nad kojim je obavljena pretplata na `click` događaj. Korišćenjem `v-on` direktive definisano je da će se prilikom `click` događaja aktivirati funkcija sa nazivom `sayHello`. Takva funkcija je definisana unutar objekta koji je dodeljen svojstvu `methods`. Na ovaj način, klikom na `button` element, dolazi do prikaza modalnog prozora sa porukom *Hello Ben Torrance*.

Pristup instanci komponente

Obratite pažnju na to da su u primeru za formiranje poruke iskorišćeni podaci koji se komponenti prosleđuju od Vue instance i koji se smeštaju unutar `user` svojstva komponente. Kako bi se došlo do takvih podataka koristi se ključna reč `this`. Ključna reč `this`, bilo gde unutar objekta za konfigurisanje komponente, upućuje na instancu komponente. Jedini izuzetak od takvog pravila jeste upotreba Arrow funkcija. Stoga se savetuje izbegavanje korišćenja Arrow funkcija unutar objekata za konfigurisanje Vue komponenata i instanci.

U nekim situacijama može biti korisno obrađivati događaje komponenata izvan njih. Takvo nešto se može obaviti emitovanjem događaja izvan komponente, što omogućava da se događaj obradi na globalnom nivou, odnosno unutar roditeljske komponente ili unutar Vue instance.

```
    <div id="container">
        <user-component v-bind:user="user" v-on:say-hello="sayHello"></user-
component>
    </div>

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

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

        Vue.component('user-component', {
            props: ['user'],
            template: `<div class="user-container">
                        <h3>{{ user.name }}</h3>
                        <p v-text='user.email'></p>
                        <p v-text='user.balance'></p>
                        <button v-on:click="$emit('say-hello',
user.name)">Say Hello</button>
                    </div>`

        })

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

```
        methods: {
            sayHello: function (name) {
                alert(`Hello ${name}`);
            }
        }
    })

</script>
```

Primer je sada modifikovan na nekoliko načina. Za početak, metoda za obradu `click` događaja na dugme koje se nalazi unutar komponente premeštena je unutar Vue instance. Takođe, ona sada prihvata i jedan ulazni parametar kojim se definiše ime osobe.

Unutar komponente više se ne obavlja obrada `click` događaja, već se on emituje izvan okvira same komponente, što se postiže pozivanjem jedne posebne metode Vue biblioteke. Reč je o metodi `$emit()`.

Metoda $emit()

Metoda `$emit()` se koristi za to da proizvede proizvoljan događaj unutar Vue komponente. Njoj je moguće proslediti dva parametra:

```
$emit( eventName, […args] )
```

Parametri imaju sledeće značenje:

- `eventName` – naziv proizvoljnog događaja i
- `args` – parametri koji će da budu prosleđeni funkciji za obradu događaja.

Na osnovu prikazanih parametara, moguće je uvideti da metoda `$emit()`omogućava kreiranje potpuno proizvoljnih događaja koji će biti emitovani iz komponenata. Stoga je za vrednost parametra `eventName` moguće definisati bilo koji naziv.

U upravo prikazanom primeru, kojim se ilustruje emitovanje događaja iz komponente, obavljeno je sledeće:

- unutar šablona komponente obavljena je pretplata na `click` događaj `korišćenjem` `v-on` direktive,
- kao vrednost `v-on` direktive postavljen je poziv `$emit()` funkcije, čime se prilikom klika na `button` element iz komponente emituje događaj sa nazivom `say-hello`,
- `say-hello` događaju se pridodaje i parametar koji se odnosi na naziv korisnika,
- `say-hello` događaj je obrađen unutar Vue instance tako što je unutar šablona Vue aplikacije na `user-component` elementu postavljena direktiva `v-on` sa vrednošću koja ukazuje na naziv funkcije koja će obraditi događaj `say-hello`,
- na kraju, kada korisnik klikne na `button` element unutar komponente, komponenta emituje `say-hello` događaj, što na kraju kao posledicu ima aktiviranje funkcije `sayHello()`, koja je definisana unutar objekta koji je priključen `methods` svojstvu Vue instance.

Slotovi

Nešto ranije je prikazan osnovni način na koji je moguće prosleđivati podatke komponentama. Reč je o pristupu koji podrazumeva korišćenje svojstava koja se definišu upotrebom `props` svojstva. Vrednosti takvih svojstava se definišu kao atributi koji se postavljaju na komponentama prilikom njihove integracije unutar šablona.

Još jedan način koji omogućava da se komponentama proslede podaci koje je potrebno prikazati unutar šablona jeste korišćenje slotova. `slot` je zapravo specijalni element, koji se može postaviti unutar šablona komponente, a čija vrednost će biti popunjena tekstom koji se definiše kao sadržaj prilikom integracije komponente u šablon Vue instance. Sledeći primer ilustruje korišćenje slotova:

```
<div id="container">
    <info-box>
        This is message 1.
    </info-box>
    <info-box>
        This is message 2.
    </info-box>
</div>

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

    Vue.component('info-box', {
        template: `
        <div>
            <strong>Message:</strong>
            <slot></slot>
        </div>
        `
    })

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

</script>
```

U primeru je kreirana jedna Vue komponenta sa nazivom `info-box`. Unutar njenog šablona je postavljen `slot` element. Ovakva komponenta je dva puta uvrštena u šablon Vue instance, pri čemu je sadržaj kojim će biti popunjen `slot` element definisan kao sadržaj elementa kojim se unutar šablona predstavlja komponenta. Sve ovo će na stranici da proizvede rezultat kao na slici 6.4.

Message: This is message 1.
Message: This is message 2.

Slika 6.4. Komponente čiji je sadržaj delimično definisan korišćenjem slotova

Single-file komponente

Vrhunac rada sa Vue komponentama jeste mogućnost definisanja svake komponente u zasebnom fajlu. Takve komponente se nazivaju Single-file komponente (engl. *Single-file components*).

Single-file komponente omogućavaju da se unutar jednog fajla smesti kompletan kôd kojim se definiše jedna Vue komponenta. To praktično znači da je unutar jednog fajla moguće definisati šablon, stilizaciju i JavaScript logiku kojom se definiše komponenta.

Single-file Vue komponente smeštaju se unutar fajlova sa .**vue** ekstenzijom. Stoga je prvi korak u procesu kreiranja Single-file Vue komponente – kreiranje praznog fajla sa ekstenzijom `.vue`.

Evo kako jedna od komponenata iz dosadašnjeg toka ove lekcije može da izgleda kada se pretvori u Single-file komponentu:

```
<template>
    <div class="user-container">
      <h3>{{ user.name }}</h3>
      <p>{{ user.email }}</p>
      <p class="balance">{{ user.balance }}</p>
    </div>
</template>

<script>
  export default {
    name: "MyVueComponent",
    props: {
      user: Object
    }
  };
</script>

<style scoped>
  h3 {
    text-decoration: underline;
  }

  .balance {
    font-style: italic;
  }
</style>
```

U kodu kojim se kreira prva Single-file Vue komponenta možete da uočite tri osnovne celine:

- `template` element – reč je o specijalnom elementu unutar koga se definiše šablon komponente,
- `script` element – element unutar koga se definiše JavaScript kôd kojim se konfiguriše komponenta,
- `style` element – element za definisanje stilizacije koja će se odnositi na HTML elemente koji se koriste za izgradnju komponente.

Korišćenjem Single-file komponenata mi zalazimo u polje naprednog korišćenja Vue ekosistema i uopšte u oblast naprednog frontend programiranja. Naime, s obzirom na to da se kod Single-file komponenata smešta unutar fajla sa ekstenzijom `.vue`, web pregledači nisu u stanju da razumeju sadržinu takvih fajlova. Stoga je u priču o razvoju potrebno uključiti još neke alate koji se koriste za izgradnju koda koji će biti razumljiv web pregledačima. Reč je o alatima koji se drugačije nazivaju *Build Tools*. Oni omogućavaju da se Vue komponente razdvoje i pretvore u čist HTML, CSS i JavaScript kod koji će web pregledači biti u mogućnosti da razumeju.

Danas postoji nekoliko popularnih alata za izgradnju, koje je moguće koristiti prilikom razvoja Vue aplikacija. Pre svih, to su alati *Browserify* i *Webpack*, koji su već spomenuti u jednoj od prethodnih lekcija kada je bilo reči o izvršnom okruženju Node.js i npm menadžeru paketa. Ipak, mi u nastavku takve alate nećemo koristiti direktno, zato što Vue ekosistem poseduje specijalizovani alat koji je posebno prilagođen razvoju Vue aplikacija, a koji korišćenje spomenutih alata olakšava i apstrahuje. Reč je o Vue CLI-ju, komandnom interfejsu koji obezbeđuje skup različitih alata koji ubrzavaju i olakšavaju programiranje u Vueu.

Rezime

- Vue komponente su samostalne, nezavisne jedinice, kojima se predstavlja jedan deo korisničkog okruženja web aplikacija.
- Vue komponente su Vue instance koje se kreiraju korišćenjem `Vue.component()` metode.
- `Metoda Vue.component()`prihvata dva parametra, pri čemu se prvi odnosi na naziv komponente, a drugi na objekat za konfigurisanje.
- Unutar Vue komponenata svojstvo `template` koristi se za definisanje šablona komponente.
- Šabloni komponenata moraju imati jedan koreni element.
- Vue komponente se unutar šablona aplikacije smeštaju definisanjem HTML elementa čiji nazivi tagova odgovaraju nazivima komponenata.
- Lokalni podaci Vue komponenata se definišu korišćenjem svojstva `data`, čija vrednost mora biti JavaScript funkcija.
- Podaci se komponentama mogu prosleđivati korišćenjem proizvoljnih atributa, čiji se nazivi definišu kao vrednosti svojstva `props` unutar objekta za konfigurisanje Vue komponente.
- Vue komponente mogu biti lokalne i globalne.
- Globalne komponente su one koje se kreiraju korišćenjem metode `Vue.component()`.
- Lokalne komponente se kreiraju kao obični JavaScript objekti koji se dodeljuju nekoj promenljivoj, a zatim unutar Vue instance ili neke druge komponente uključuju korišćenjem svojstva `components`.
- Metoda `$emit()` se koristi da proizvede proizvoljan događaj unutar Vue komponente.
- Komponentama se podaci mogu proslediti korišćenjem specijalnog elementa `slot`, koji se može postaviti unutar šablona komponente, a čija vrednost će biti popunjena tekstom koji se definiše kao sadržaj prilikom integracije komponente u šablon Vue instance.
- Single-file komponente omogućavaju da se unutar jednog fajla smesti kompletan kôd kojim se definiše jedna Vue komponenta – šablon, stilizacija i JavaScript.
- Single-file Vue komponente smeštaju se unutar fajlova sa `.vue` ekstenzijom.
- Kako bi mogle biti korišćene, Single-file Vue komponente je neophodno prevesti u oblik razumljiv web pregledačima, za šta se koriste specijalni alati za izgradnju.