Vue.js

Vue.js

- Jedan od najpopularnijih JavaScript radnih okvira
- Obično se koristi za pravljenje single-page aplikacija
  - SPA – Single-Page Application
- Stvorio ga je Evan You koji je bio zaposlen u Google-u
- Izvukao je najbolje karakteristike od Angular-a
- Prvi Vue izvorni kod je napisan 2013. godine

Hello World!

Kreiranje projekta

```
npm create vue@latest
cd my-app
npm install
npm run dev
```

Tokom instalacije dodati vue-router modul

Primer Vue komponente

Template deo
Prikaz (HTML)

```
<template>
   <p>{{ message }}</p>
</template>
```

Script deo
Java Script kod

```
<script setup>
import { ref } from "vue"

const message = ref("Zdravo iz Vue-a!")
</script>
```

Style deo
CSS stilovi

```
<style scoped>
   p { font-size: 20px; }
</style>
```

Osnovni elementi

- Model – sadrži podatke
- Šablon – prikazuje podatke
- Direktive – proširenja HTML-a
- Metoda – poziva se kao reakcija na događaje korisničkog interfejsa
- Ajax/fetch pozivi
- Ruter - definiše koja komponenta će se aktivirati kada se klikne na odgovarajući link
- Komponente

Model

```
<script setup>
import { ref, reactive } from "vue"

const loggedIn = ref(true)
const user = ref({id: 1, name: "Alice"})
const employees = ref([
   {id: 1, name: "Bob"},
   {id: 2, name: "Charlie"}
])

</script>
```

- U Composition API-ju promenjivama definisanim pomoću `ref()` se pristupa preko .value (npr. `loggedIn.value`). U template delu nije potrebno dodavati `.value`
- Ako čuvamo objekte ili nizove, možemo koristiti i `reactive()`, kojem se prisupa direktno i nije potrebno koristiti `.value`

Šablon

```
<template>
    <div>
        <h2 class="text-center">Zaposleni</h2>
        <div class="card">
            <h1>{{ employee.firstName }}</h1>
            <h1>{{ employee.lastName }}</h1>
            <p class="title">{{ employee.position }}</p>
        </div>
    </div>
</template>

<script setup>
import { ref, reactive } from "vue"

const employee = ref(
    {
        id: 1,
        firstName: "Pera",
        lastName: "Petrovic",
        position: "Programer",

    }
)

</script>
```

One-way binding

Direktive

- Proširuju HTML dodatnim tagovima i atributima
- Sistemske direktive su atributi koji počinju: “v-” stringom
  - v-model – povezuje model sa input poljem
  - v-if – koristi se kod definisanja uslova
  - v-for – koristi se kod iteriranja kroz kolekcije
  - v-bind – povezuje model sa HTML elementom
  - v-on – event listener (@click)

v-model

- Direktiva koja omogućava **two way binding** između HTML elemenata i modela

```
<template>
   <label for="firstName">Ime:</label>
   <input v-model="employee.firstName" id="firstName" />

   <label for="firstName">Prezime:</label>
   <input v-model="employee.lastName" id="firstName" />

   <label for="position">Pozicija:</label>
   <input v-model="employee.position" id="position" />
</template>

<script setup>
import { ref, reactive } from "vue"

const employee = ref({
       firstName: "Pera",
       lastName: "Petrovic",
       position: "Programer",
   })
</script>
```

Two-way binding

v-if

- Uslovni prikaz

```
<template>
    <span v-if="loggedIn">Ulogovan si</span>
    <div v-else>
        <span>Nisi ulogovan</span>
    </div>
</template>

<script setup>
import { ref, reactive } from "vue"

const loggedIn = ref(false)
</script>
```

v-for

```
<script setup>
import { ref, reactive } from "vue"

const employees = ref(
    [{
        firstName: "Pera",
        lastName: "Petrovic",
        position: "radnik",

    },
    {
        firstName: "Marko",
        lastName: "Markovic",
        position: "menadzer",

    }]
)
</script>
```

```
<template>
    <table>
        <tr>
            <th>Ime</th>
            <th>Prezime</th>
            <th>Pozicija</th>
        </tr>
        <tr v-for="employee in employees" :key="employee.id">
            <td>{{ employee.firstName }}</td>
            <td>{{ employee.lastName }}</td>
            <td>{{ employee.position }}</td>
        </tr>
    </table>
</template>
```

Ime	Prezime	Pozicija
Pera	Peric	radnik
Marko	Markovic	menadzer

v-bind

- Povezivanje HTML atributa sa modelom

```
<input type="text" name="text" v-model="employee.firstName" v-bind:disabled="mode == 'VIEW'" />
…
const mode = ref("VIEW") // EDIT, VIEW
```

Može se izostaviti v-bind,

```
<input :disabled="mode" :class="'card'"
```

Metode

```vue
<template>
    <h1>{{ employee.firstName }}</h1>
    <h1>{{ employee.lastName }}</h1>
    <button class="btnSeeMore" @click="showMore">Prikaži detalje</button>
</template>

<script setup>
import { ref, reactive } from "vue"
import { useRouter } from "vue-router"

const router = useRouter()

const employee = ref(
    {
        id: 1,
        firstName: "Pera",
        lastName: "Petrovic",
        position: "Programer",
    }
)

function showMore() {
    router.push(`/employee/${employee.value.id}`)
}

</script>
```

Možete koristiti i arrow funkciju

```vue
const showMore = () => {
    router.push(`/`)
}
```

Osluškivači

- Reaguju na GUI događaje:
  - v-on:click
  - v-on:dbl-click
  - v-on:mousedown
  - v-on:mouseup
  - v-on:mouseenter
  - v-on:mouseleave
  - v-on:mousemove
  - v-on:mouseover
  - v-on:keydown
  - v-on:keyup
  - v-on:keypress
  - v-on:change

v-on: se može zameniti sa **@**

```
<button v-on:click="showMore">
<button @click="showMore">
```

Ajax/fetch pozivi

- Za ajax pozive potrebno je instalirati axios biblioteku : npm install axios

```
axios
  .get("http://localhost:8081/api/employees")
  .then((res) => {
    this.employees = res.data;
  })
  .catch((err) => {
    console.log(err);
  });
```

```
fetch('http://localhost:8081/api/employees')
  .then(response => response.json())
  .then(data => {
    console.log("Success:", data); this.employees = data
  })
  .catch((error) => {
    console.error("Error:", error);
  });
```

Rad sa sesijom

- backend

```
@Configuration
public class CorsRunner implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry){
        registry.addMapping( pathPattern: "/**")
                .allowedMethods("GET", "PUT", "POST", "DELETE")
                .allowCredentials(true);
    }
}
```

Rad sa sesijom

- Frontend – axios

```
axios
    .get("http://localhost:8081/api/employees/" + this.$route.query.id, {withCredentials: true})
    .then((res) => {
        this.employee = res.data
    })
    .catch((err) => {
        console.log(err)
    })
```

```
axios
  .post("http://localhost:8081/api/employees", this.employee, {withCredentials: true})
  .then((res) => {
    console.log(res);
    this.$router.push("/employees");
  })
  .catch((err) => {
    console.log(err);
    alert("Something went wrong!");
  });
```

Ruter

- Definiše koja komponenta će se aktivirati kada se klikne na odgovarajući link
- Klik po hiperlinku, odnosno promenu adrese presreće ruter i postupa po pravilima
- Pravila se zadaju u obliku:

```
const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/employees',
    name: 'employees',
    component: EmployeesView
  }
]
```

Eventi i Props

- Props omogućavaju roditeljskoj komponenti da prosledi podatke u podkomponentu
- Emit Omogućava detetu da šalje događaje prema roditeljskoj komponenti

Roditeljska komponenta

```
<template>
   <Child :message="parentMessage"
          @childClicked="handleClick" />
</template>

<script setup>
import Child from './Child.vue'
import { ref } from 'vue'

const parentMessage = ref("Poruka od roditelja")

function handleClick() {
   alert("Dugme u detetu je kliknuto!")
}
</script>
```

Podkomponenta

```
<template>
   <div>
       <p>{{ message }}</p>
       <button @click="$emit('childClicked')"
          >Klikni me</button>
   </div>
</template>

<script setup>
    defineProps({ message: String })
</script>
```

Computed & Watch

Computed

- Reaktivne vrednosti koje se automatski računaju kada se zavisni podaci promene

Watch

- Prati promene određene promenljive.
- Koristi se za izvršavanje operacije kada se stanje promeni (npr roditeljska komponenta ažurira prop).

```
<template>
   <input v-model="firstName" placeholder="Ime" />
   <input v-model="lastName" placeholder="Prezime" />
   <p>Full Name: {{ fullName }}</p>
</template>


<script setup>
import { ref, computed, watch } from "vue"

const firstName = ref("Petar")
const lastName = ref("Petrovic")

// computed vrednost
const fullName = computed(() => firstName.value + " " +
lastName.value)

// watch primer
watch(fullName, (newVal, oldVal) => {
   console.log("Full name changed:", oldVal, "→", newVal)
})
</script>
```

Lifecycle hooks

- Lifecycle hooks su funkcije koje se automatski pozivaju u određenim fazama životnog ciklusa komponente.

- Omogućavaju da reaguješ na događaje kao što su montiranje, ažuriranje ili uklanjanje komponente iz DOM-a.

```
<script setup>
import { ref, onMounted, onUpdated, onUnmounted } from 'vue'

onMounted(() => {
   console.log("Komponenta montirana")
   // Ucitavanje podataka, setupovanje event listener-a, itd.
})

onUpdated(() => {
   console.log("DOM ažuriran, counter:") // Retko se koristi
})

onUnmounted(() => {
   console.log("Komponenta uklonjena")
   // Clean-up, uklanjanje event listener-a, itd.

})

</script>
```

Setup

```vue
<template>
   <p>{{ message }}</p>
   <button @click="changeMessage">Promeni</button>
</template>
```

```vue
<script setup>
import { ref } from 'vue'

const message = ref('Zdravo!')

const changeMessage = () => {
   message.value = 'Poruka izmenjena!'
}
</script>
```

```vue
<script>
import { ref } from 'vue'

export default {
   setup() {
        const message = ref('Zdravo!')

        const changeMessage = () => {
            message.value = 'Poruka izmenjena!'
        }
        return {
            message,
            changeMessage
        }
   }
}
</script>
```

Ako koristite `<script setup>`, Vue interno pravi `setup()` i vraća sve što definišete, pa nije potrebno eksplicitno returnovati vrednosti.

Composition API vs Option API

Composition API

```
<template>
    <div>
        <p>Broj klikova: {{ counter }}</p>
        <button @click="increment">Povećaj</button>
    </div>
</template>

<script setup>
import { ref } from 'vue'

const counter = ref(0)

function increment() {
    counter.value++
}
</script>

<style scoped>
p {font-size: 20px;}
</style>
```

Option API

```
<template>
    <div>
        <p>Broj klikova: {{ counter }}</p>
        <button @click="increment">Povećaj</button>
    </div>
</template>

<script>
export default {
    data() {
        return {
            counter: 0
        }
    },
    methods: {
        increment() {
            this.counter++
        }
    }
}
</script>

<style scoped>
p {font-size: 20px;}
</style>
```

Vue 2 koristi Options API, gde je kod organizovan po tipovima (data, methods, computed), što otežava održavanje velikih komponenti. Vue 3 uvodi Composition API, koji grupiše logiku po funkcionalnosti, olakšavajući ponovnu upotrebu i skaliranje koda.