2RING

Moderná architektúra pre SPA aplikácie

Vue + Vuex

Marek Uhlár
marek.uhlar@2ring.com

Marek Spurný
marek.spurny@2ring.com

Obsah

- Prečo práve Vue
- Komponenty
- Options API vs Composition API
- Životný cyklus
- Komunikácia – props, event
- EventBus, Emit
- Vuex
- Praktická ukážka

SPA

- Pri načítaní stránky sa stiahnú všetky potrebné JS súbory
- Hlavné vyhody:
  - Rýchlosť
  - Práca s dátami
  - Práca s UI

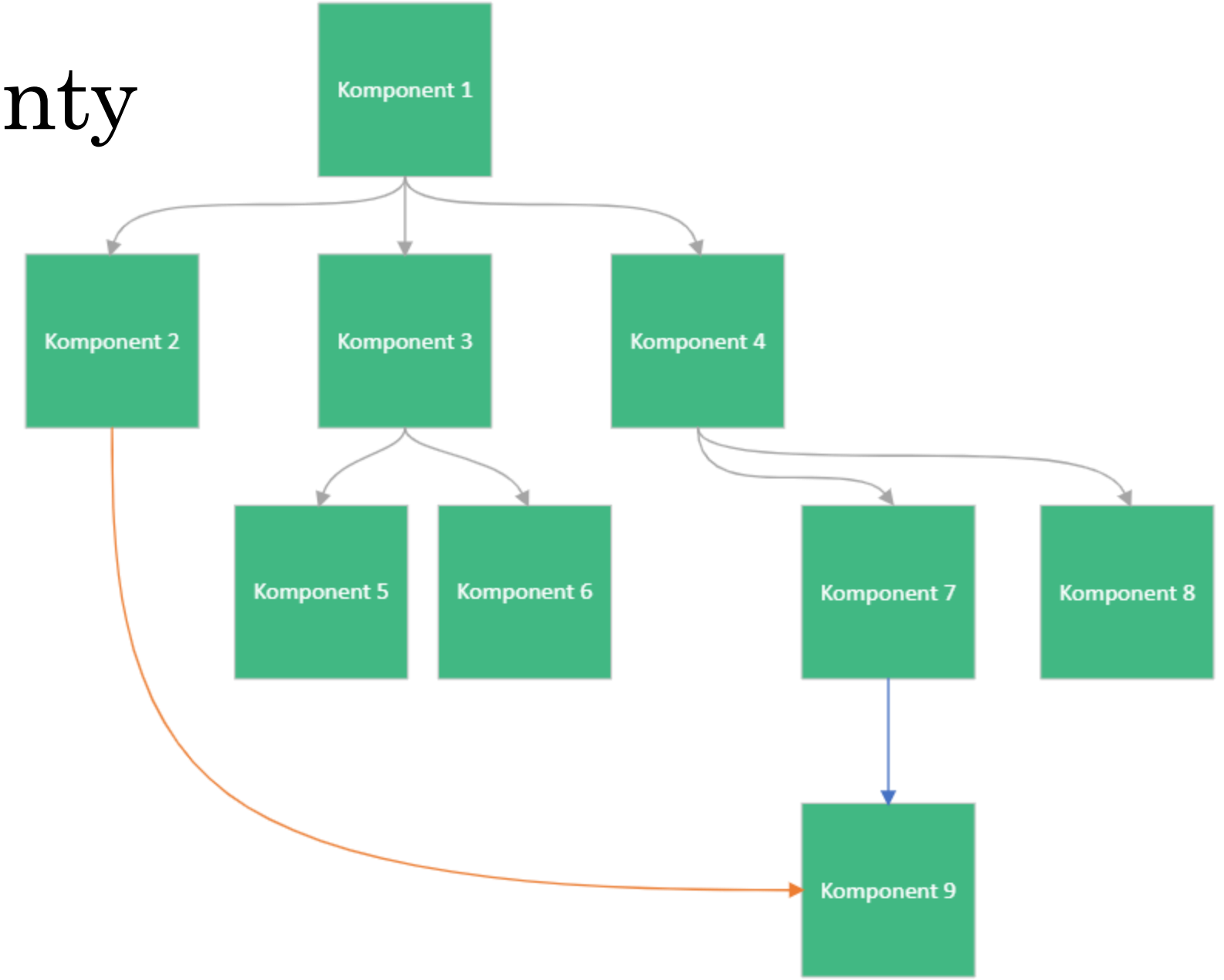
Prečo práve Vue

- Patrí medzi 3 najpoužívanejšie SPA frameworky *
- Malý, rýchly framework
- Jednoduchý na učenie
- Hlavné využitie pre dynamickú prácu s UI
- Základné fungovanie má podobné s Angularom alebo Reactom

* https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

Komponenty

Komponenty


Komponent 1
Komponent 2
Komponent 3
Komponent 4
Komponent 5
Komponent 6
Komponent 7
Komponent 8
Komponent 9

Komponent

- Template
- Script
- Style

```vue
<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<script>
export default {
  data() {
    return {
      count: 0
    }
  }
}
</script>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

API Styles

- Vue komponenty môžu byť vytvorené v dvoch štýloch
  - **Options API**
  - **Composition API**

Options API

- Tradičný spôsob definovania komponentov (VUE 2)
- Definovanie properties, údajov, metód v options objekte
- Vhodný pre jednoduchšie komponenty s menej zložitou logikou
- Pri zložitejších komponentoch sa môže stať, že bude neprehľadný a tažko čitateľný

Options API

```vue
<script>
  export default {
    data() {
      return {
        count: 0
      }
    },

    methods: {
      increment() {
        this.count++
      }
    },

    mounted() {
      console.log(`The initial count is ${this.count}.`)
    }
  }
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Composition API

- Nový spôsob organizácie logiky v komponente (VUE 3)
- Umožňuje zoskupovať súvisiace údaje, computed properties, metódy do opakovane použiteľných funkcií (composition functions)
- Výhoda vytvárania menších, cielenejších funkcií (ktoré sa ľahšie testujú a znovu používajú)
- Odporúča sa na zložitejšie komponenty s väčším množstvom reaktívnych dát a zložitejšou logikou

Composition API

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

  const count = ref(0)

  function increment() {
    count.value++
  }

  onMounted(() => {
    console.log(`The initial count is ${count.value}.`)
  })
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Options API vs Composition API

```vue
<script>
  export default {
    data() {
      return {
        count: 0
      }
    },

    methods: {
      increment() {
        this.count++
      }
    },

    mounted() {
      console.log(`The initial count is ${this.count}.`)
    }
  }
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

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

  const count = ref(0)

  function increment() {
    count.value++
  }

  onMounted(() => {
    console.log(`The initial count is ${count.value}.`)
  })
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Options API

Composition API

https://vuejs.org/assets/composition-api-after.e3f2c350.png

Šablóny (Templates)

- Založené na HTML
- Umožňuje deklaratívne viazať vykreslené DOM elementy s dátami v inštancií komponenty
- VUE optimalizuje prekreslenie DOM elementov pri zmene dát

- Základná forma dátovej väzby – Textová interpolácia

```
<span>Message: {{ msg }}</span>
```

- v-html direktíva

```
<p>Using text interpolation: {{ rawHtml }}</p>
<p>Using v-html directive: <span v-html="rawHtml"></span></p>
```

Directives

- Podmienečné vykresľovanie
  - **v-if, v-else, v-else-if**
- Class a Style bindovanie
  - **:class** (v-bind:class)**, :style**
- Zoznamy
  - **v-for**
- Events
  - **v-on:click** (@click)
- Input
  - **:value**

```
<h1 v-if="awesome">Vue is awesome!</h1>
```

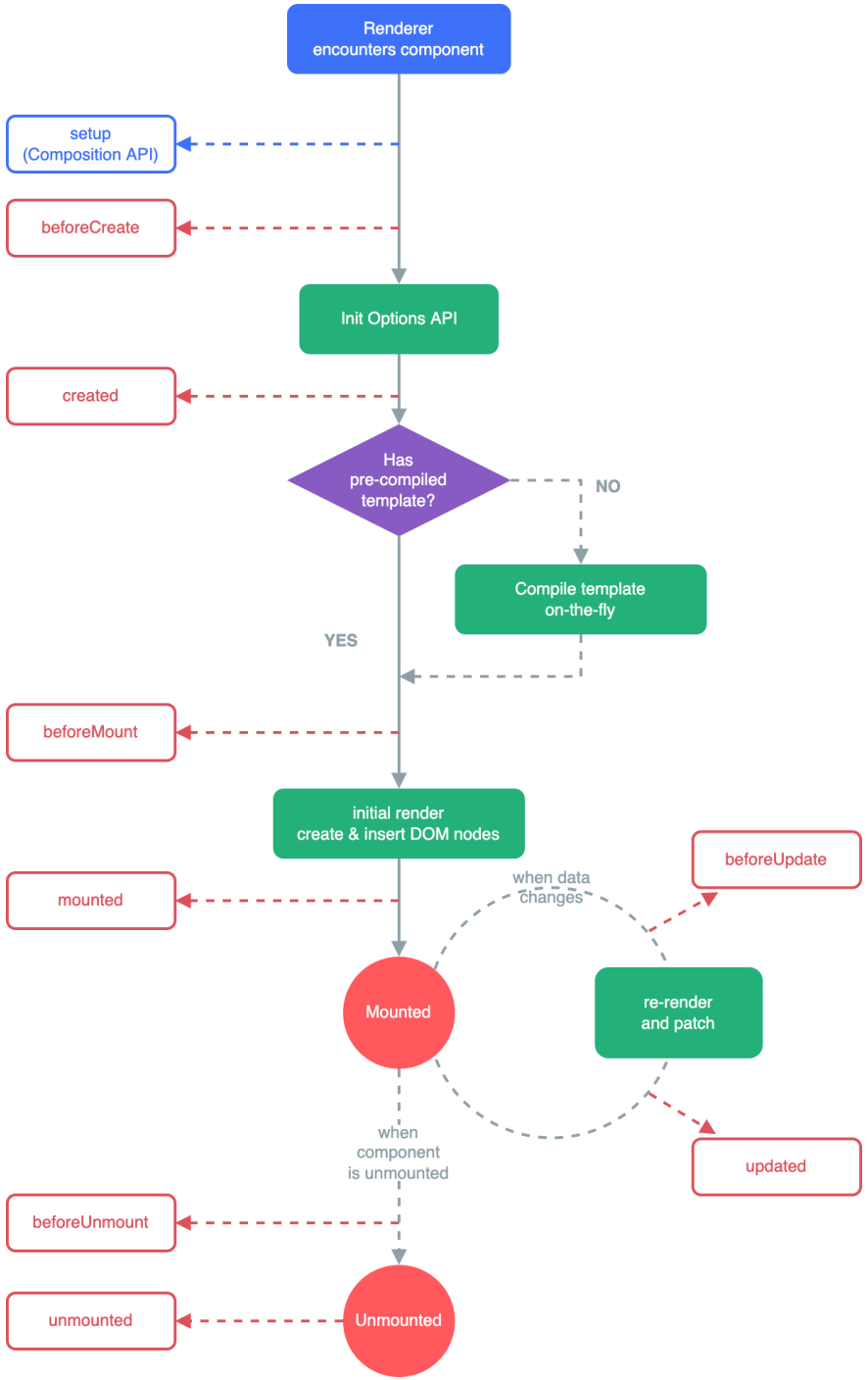
```
<div :class="{ active: isActive }"></div>
```

```
<li v-for="item in items">
  {{ item.message }}
</li>
```

```
<button @click="greet">Greet</button>
```

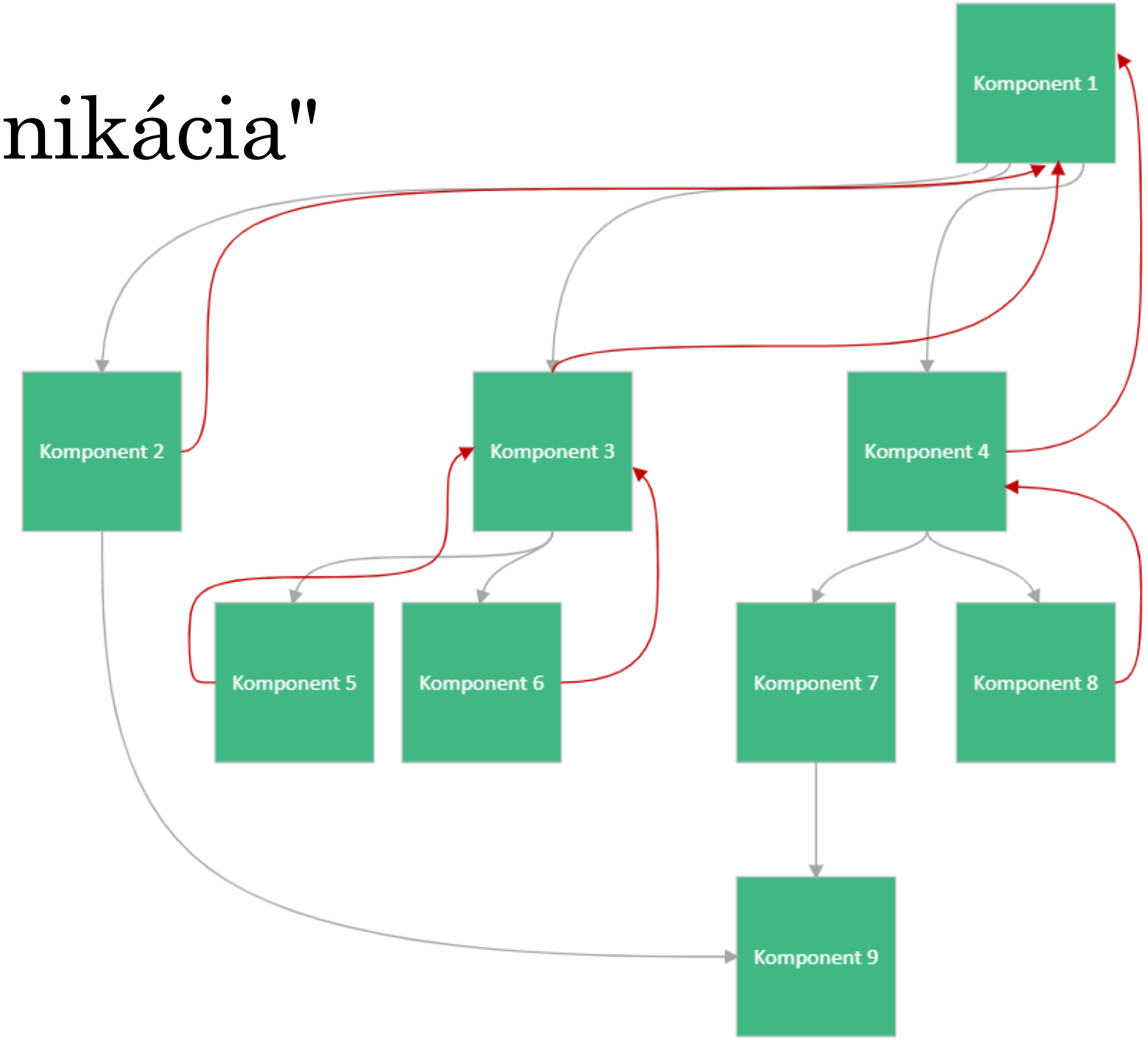
```
<input :value="text">
```

Lifecycle diagram



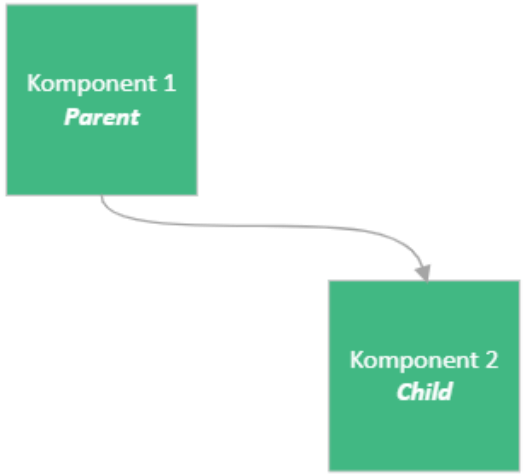
https://vuejs.org/guide/essentials/lifecycle.html#lifecycle-diagram

"Komunikácia"



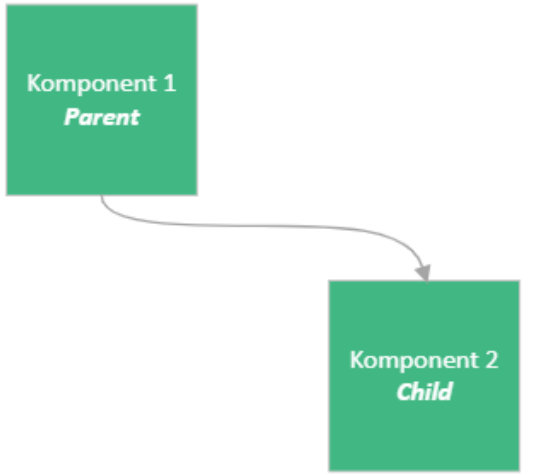
Props

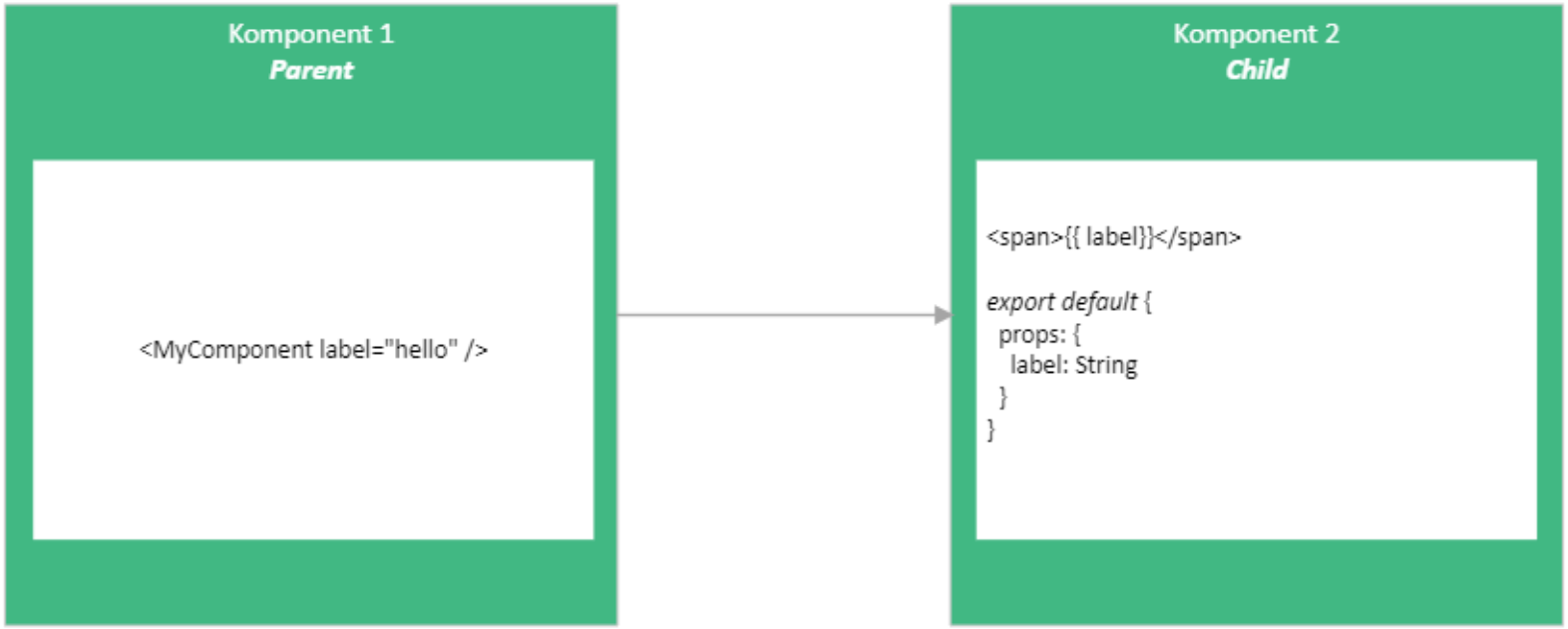
- Posielanie dát z parent komponetu do child komponentu



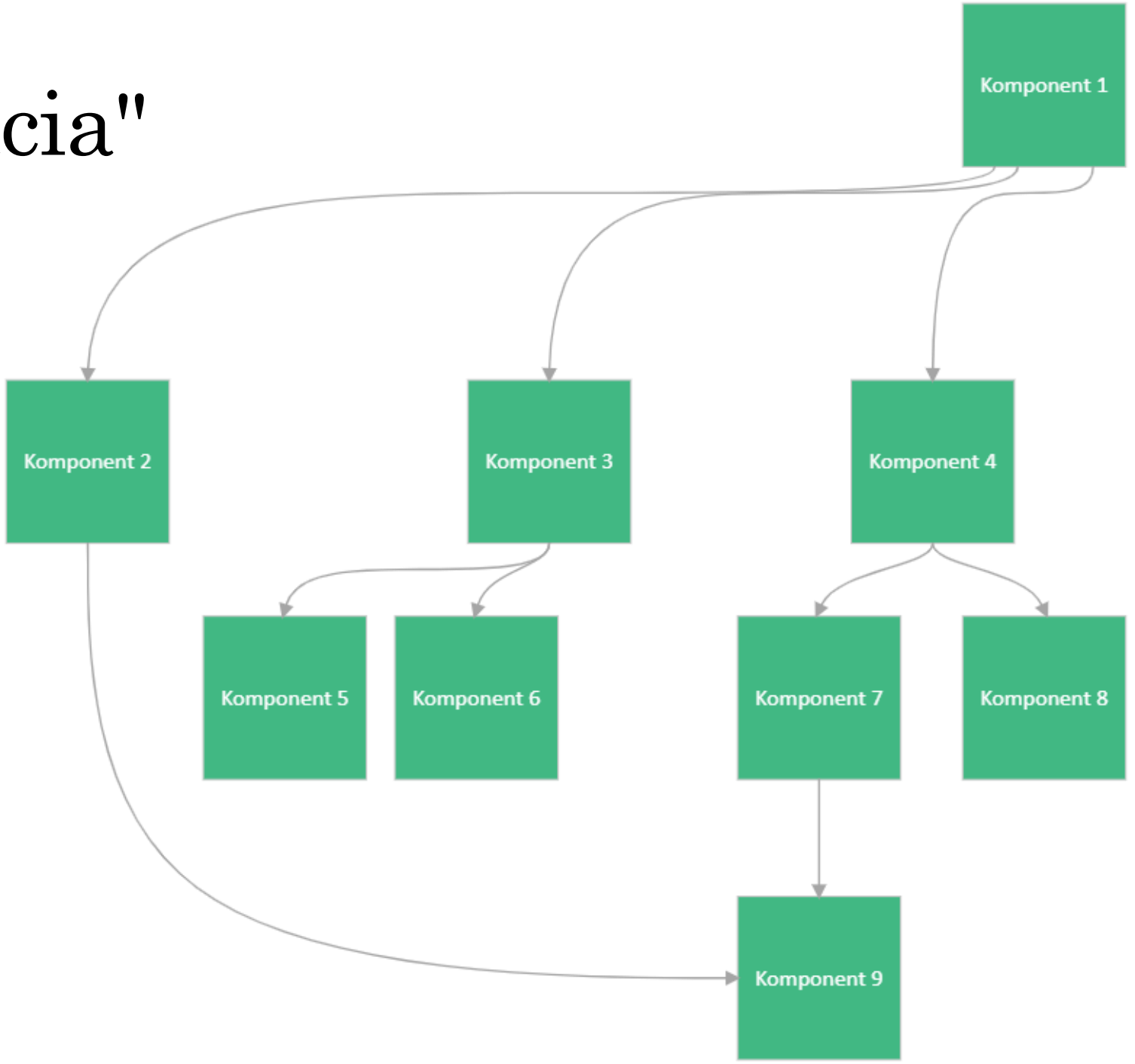
Props

- Posielanie dát z parent komponentu do child komponentu

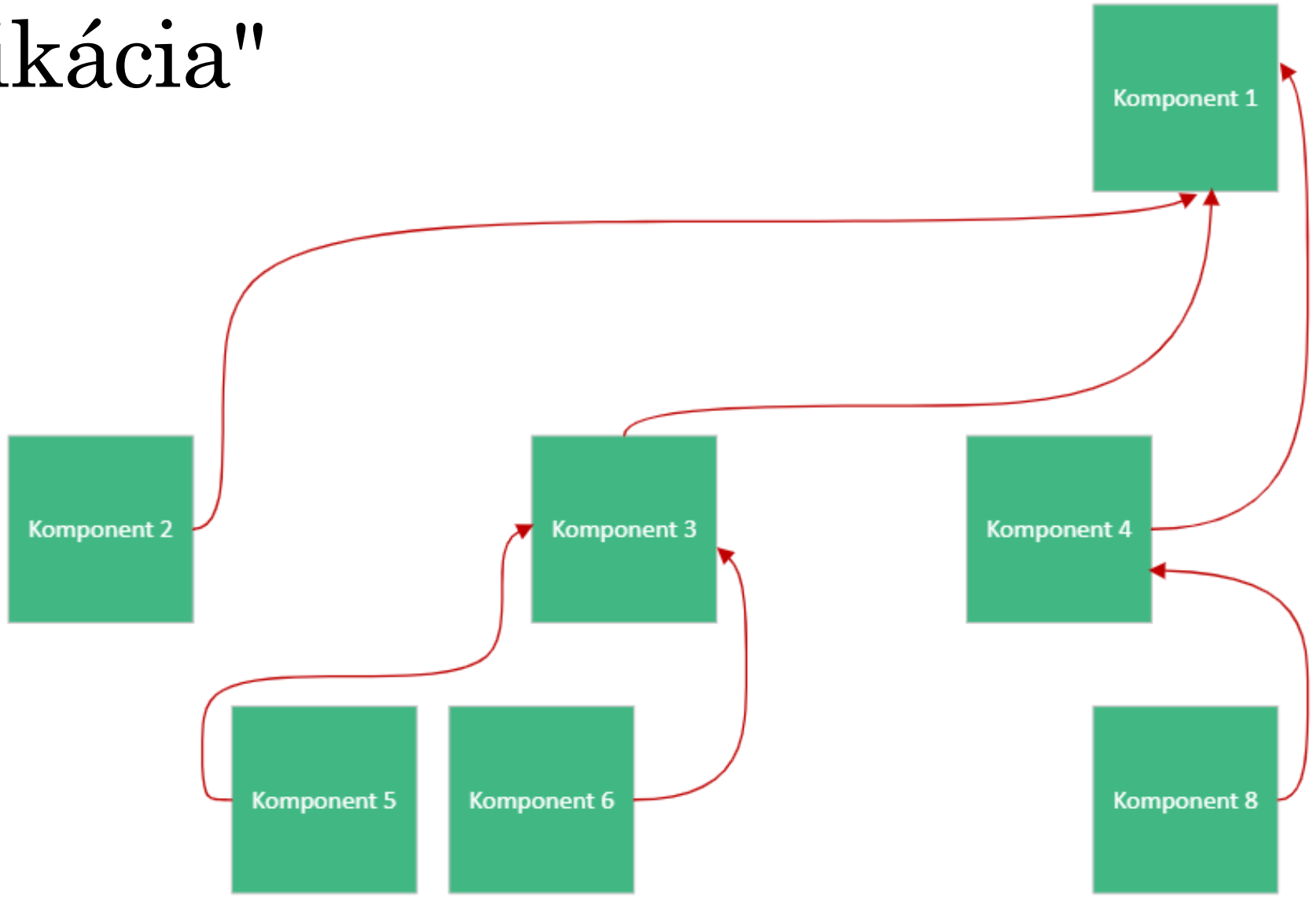




"Komunikácia"

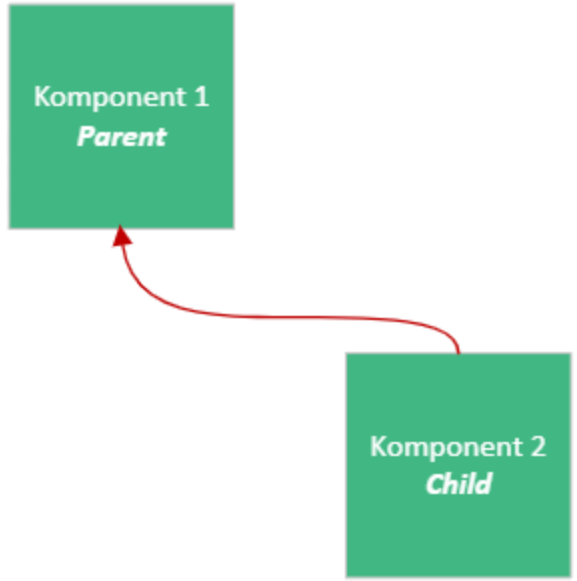


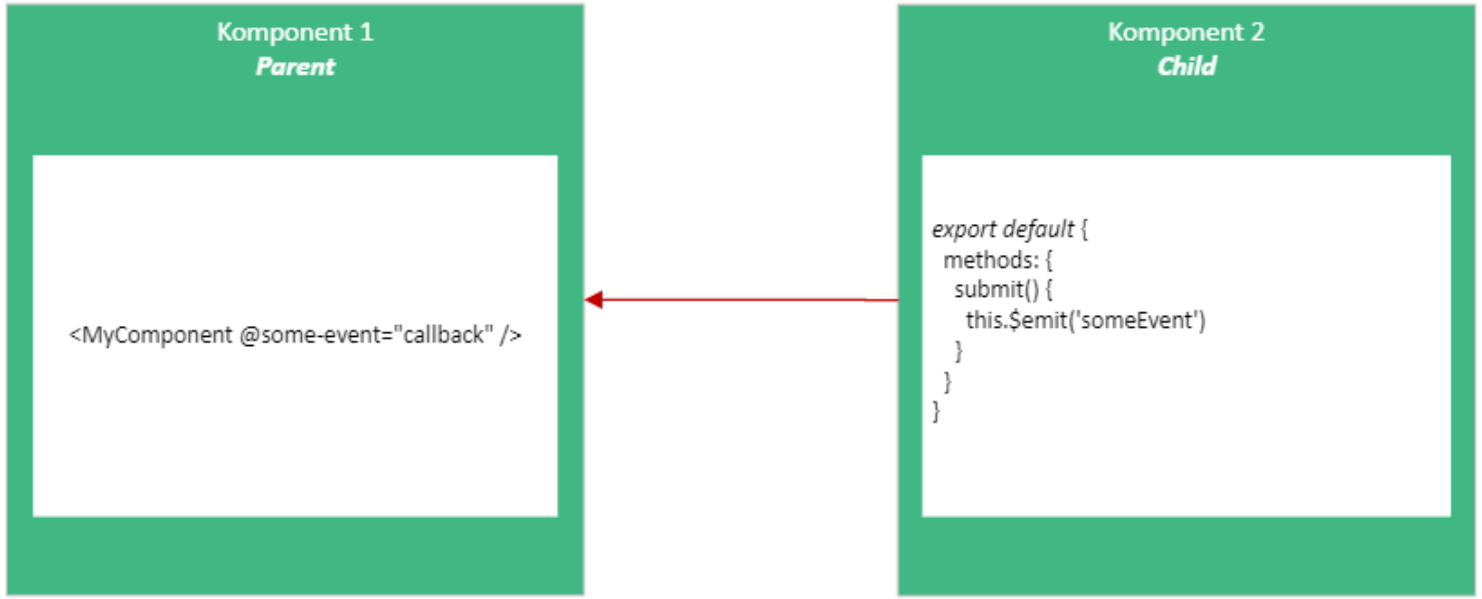
"Komunikácia"



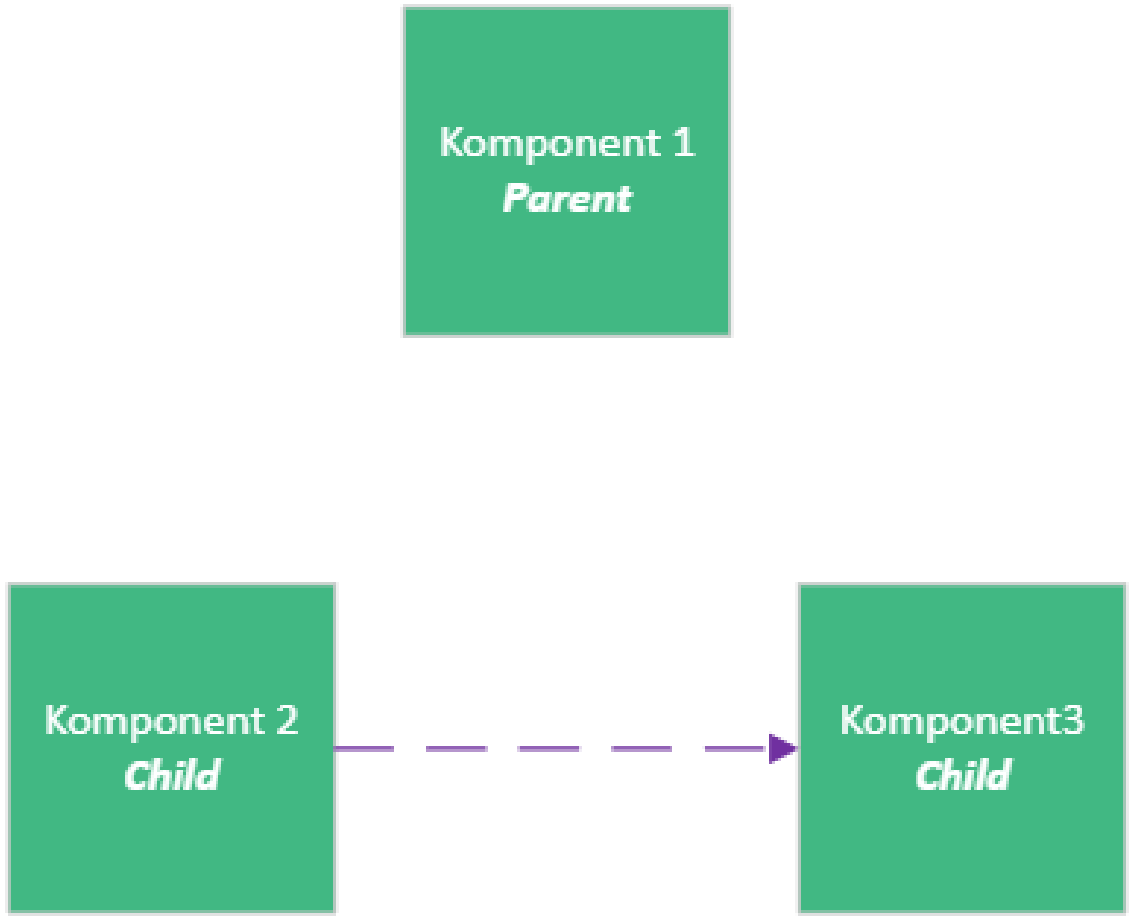
Events

- Komunikácia z child komponentu do parent komponentu

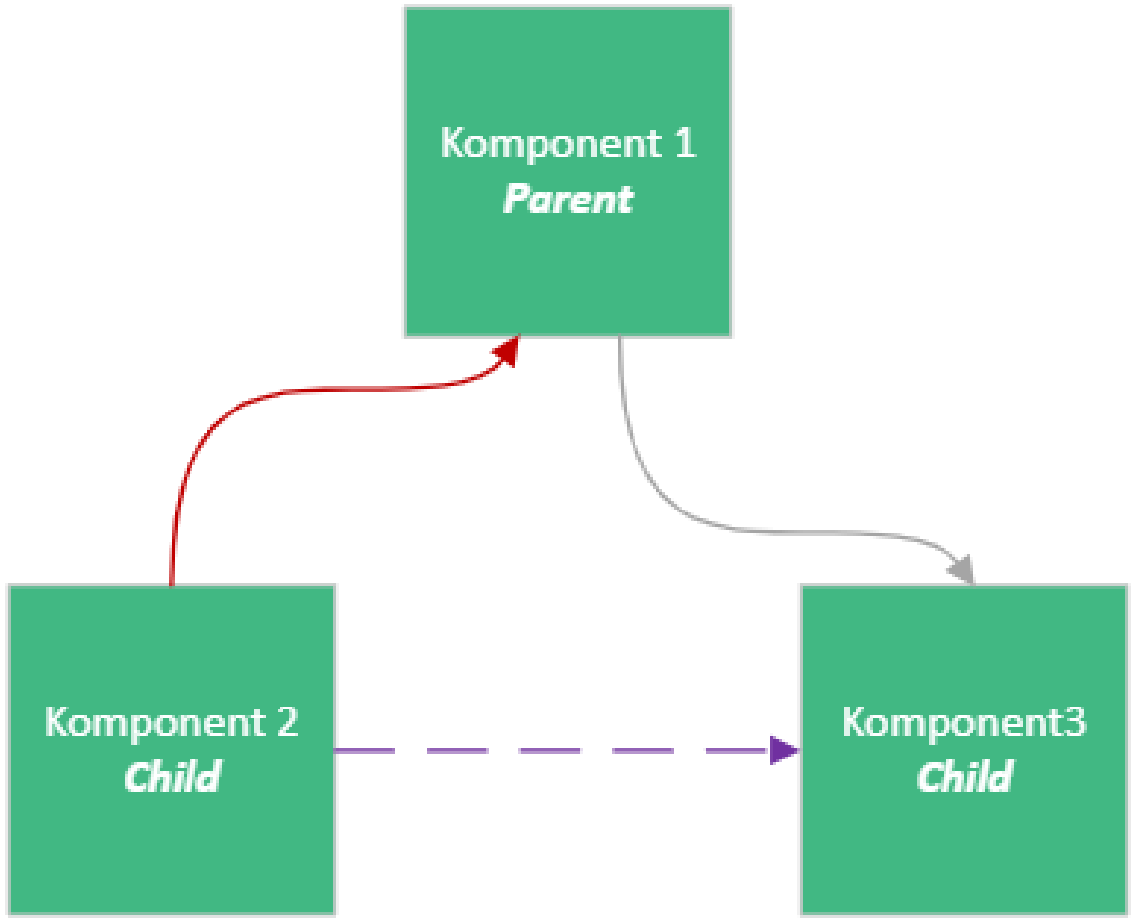




Event-bus



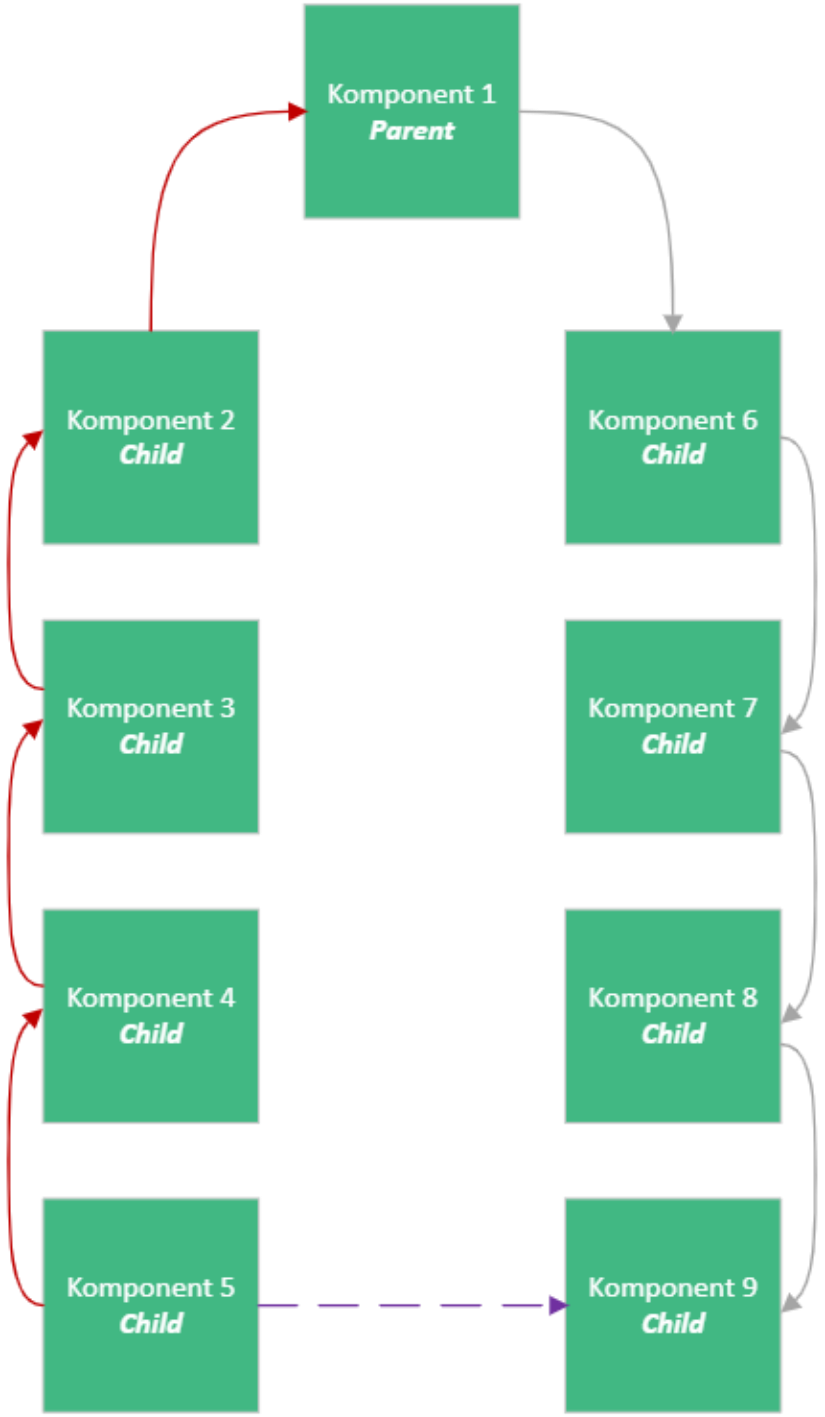
Event-bus



Event-bus


Komponent 1
Parent
Komponent 2
Child
Komponent 6
Child
Komponent 3
Child
Komponent 7
Child
Komponent 4
Child
Komponent 8
Child
Komponent 5
Child
Komponent 9
Child

Event-bus


Komponent 1
Parent
Komponent 2
Child
Komponent 6
Child
Komponent 3
Child
Komponent 7
Child
Komponent 4
Child
Komponent 8
Child
Komponent 5
Child
Komponent 9
Child

Komunikácia a dáta medzi komponentami

- Vo Vue3 odstránená podpora Event-bus
- Externé event huby mitt alebo tiny-emitter
- Sú aj iné možnosti ?

Vuex

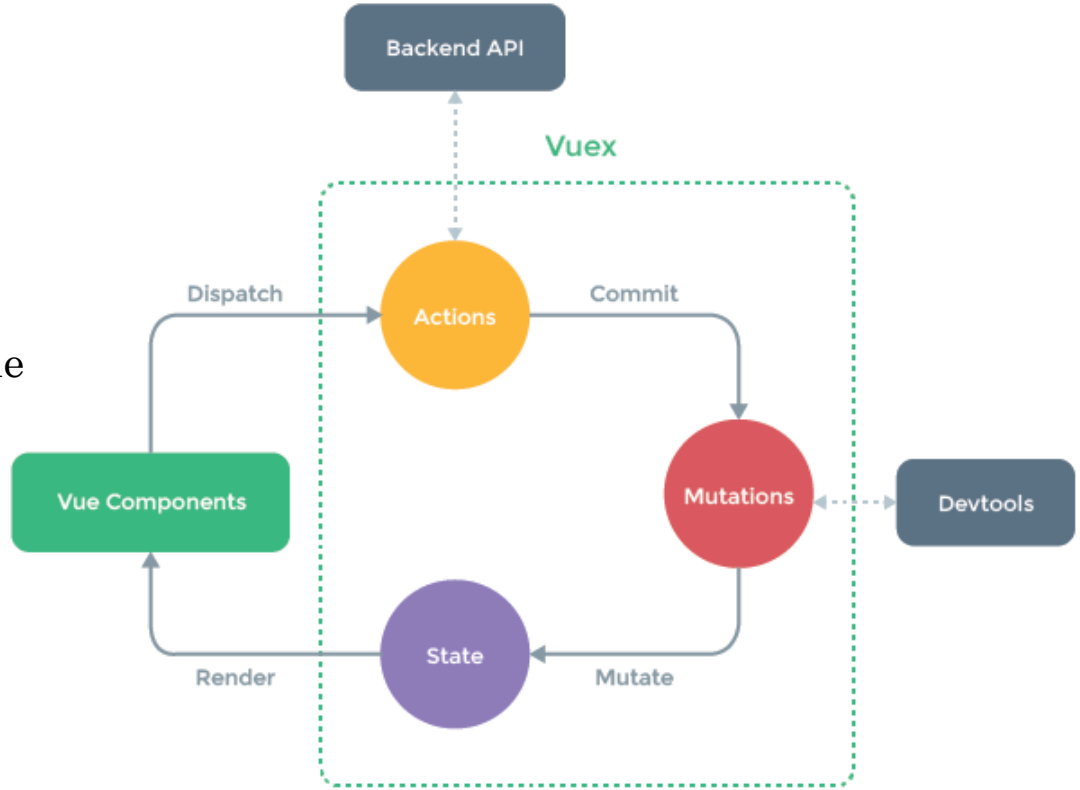
- Management stavu celej aplikácie
- Centralizovaná "správa" dát v aplikácii - konzistencia
- Odpadá synchornizácia dát
- Vhodné pre väčšie projekty
- State management (centralizovaný store pre celú aplikáciu)
- Vuex moduly

- React
  - Redux

- Angular
  - NgRX

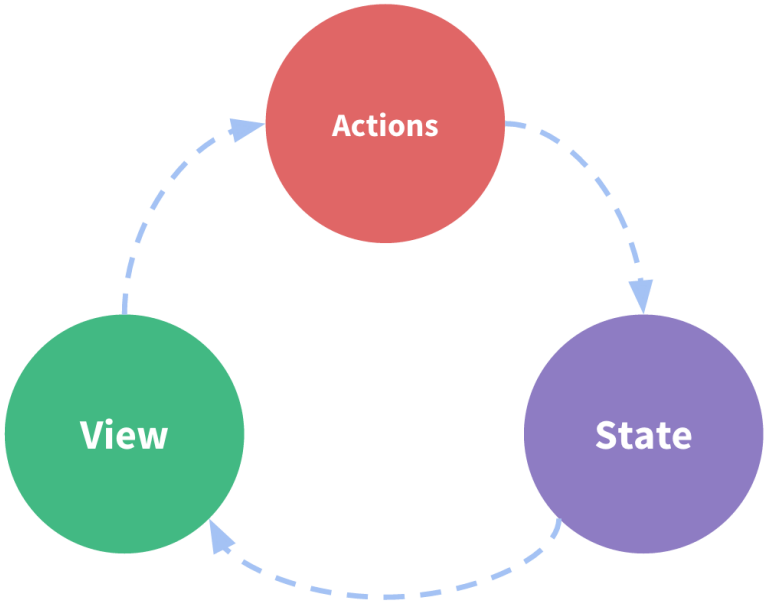
VUEX – architektúra

- Getters
  - Metódy, ktoré vracajú dáta zo stavu
- Mutations
  - Funkcie, za pomoci ktorých sa modifikuje stav
- Actions
  - Iné metódy, komplexnejšie metódy, ktoré volajú mutácie



VUEX – State

- State – jediný objekt, ktorý obsahuje všetky stavy v aplikácii
- Jediný zdroj pravdy
- Aj keď je to jednostavový strom, nie je to problém pri modularite (modules)



https://vuex.vuejs.org/flow.png

VUEX – Getters

- Potreba vrátenia upravenej hodnoty zo store statu

```
const store = createStore({
    state: {
        todos: [
            { id: 1, text: '...', done: true },
            { id: 2, text: '...', done: false }
        ]
    },
    getters: {
        doneTodos (state) {
            return state.todos.filter(todo => todo.done)
        }
    }
})
```

VUEX – Getters

```
getters: {
    getTodoById: (state) => (id) => {
        return state.todos.find(todo => todo.id === id)
    }
}

//
store.getters.getTodoById(2)
```

VUEX – Mutations

- Store state sa mení (aktualizuje) pomocou mutácií

```
const store = createStore({
    state: {
        count: 1
    },
    mutations: {
        increment (state) {
            // mutate state
            state.count++
        }
    }
})

//
store.commit('increment')
```

VUEX – Actions

- Podobné ako mutácie, rozdiely:
  - Namiesto priamej mutácie stavu sa vykoná mutácia
  - Akcie môžu obsahovať ľubovoľné asynchrónne operácie

```
const store = createStore({
    state: {
        count: 0
    },
    mutations: {
        increment (state) {
            state.count++
        }
    },
    actions: {
        increment (context) {
            context.commit('increment')
        }
    }
})
```

VUEX – Actions

```
actions: {
    loadBooks({ commit }) {
        commit('startLoading');
        get('/api/books').then((response) => {
            commit('setBooks', response.data.books);
            commit('stopLoading');
        });
    }
}
```

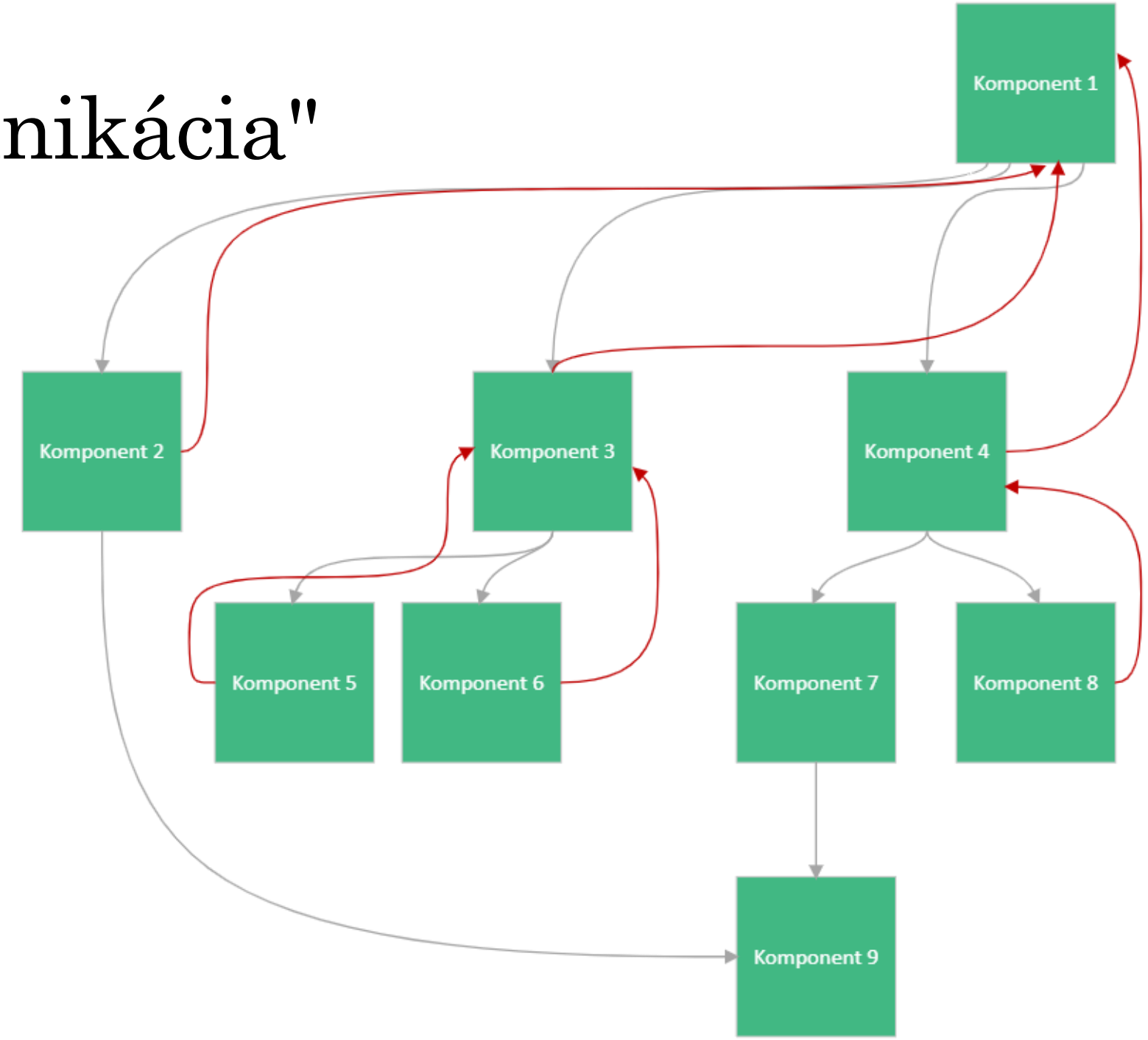
VUEX – Modules

- Pri použití jedného state tree sú všetky stavy obsiahnuté v jednom objekte
- Možnosť rozdelenia storu do viacerých modulov
- Každý modul obsahuje vlastný state, mutations, actions, getters, ...

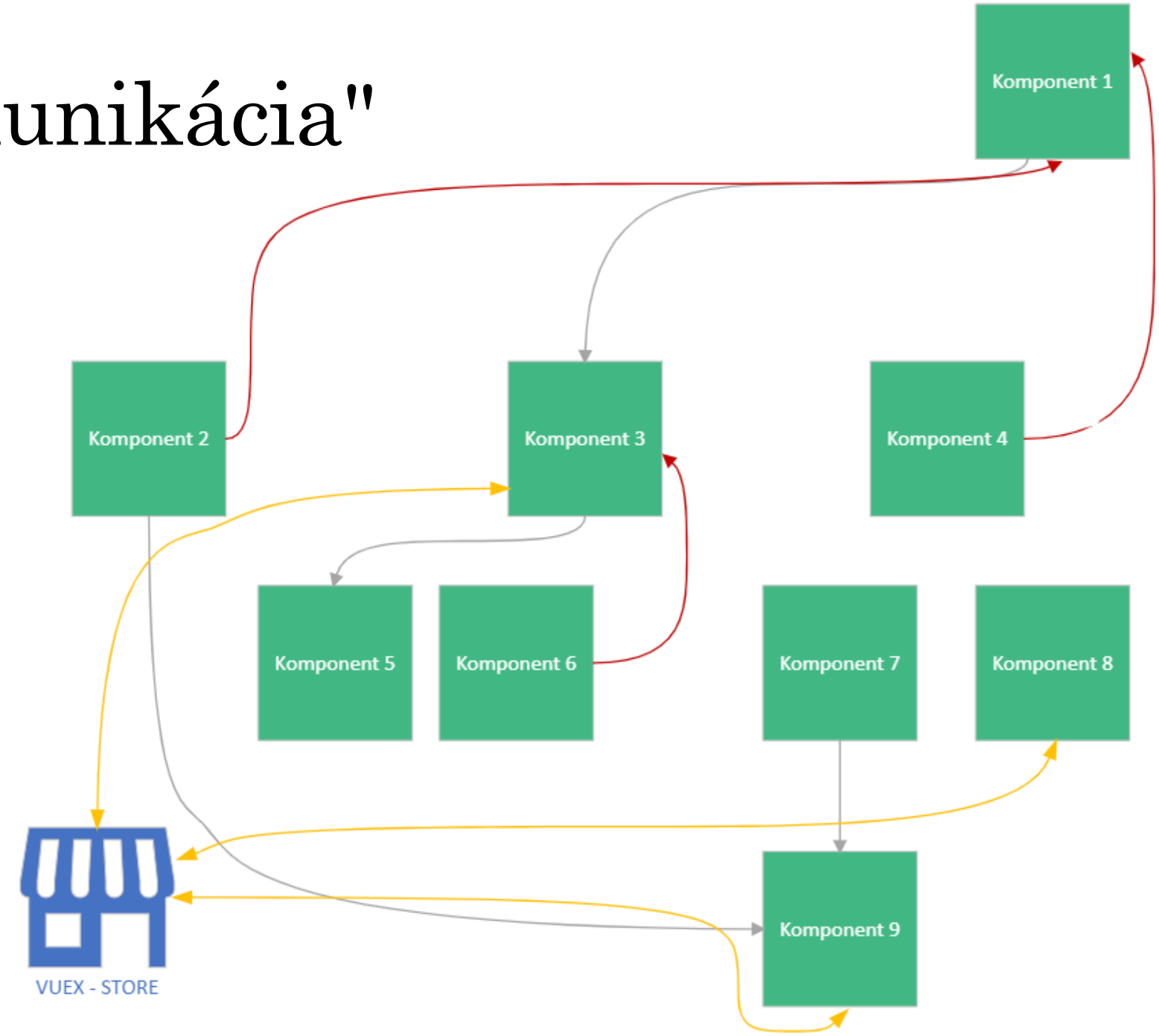
VUEX – Modules

```
const moduleA = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... },
  getters: { ... }
}

const moduleB = {
  state: () => ({ ... }),
  mutations: { ... },
  actions: { ... }
}

const store = createStore({
  modules: {
    a: moduleA,
    b: moduleB
  }
})

store.state.a // -> `moduleA`'s state
store.state.b // -> `moduleB`'s state
```

"Komunikácia"



"Komunikácia"



Linky

- https://vuejs.org/guide
- https://vuex.vuejs.org/
- https://github.com/MarekUhlar/FIIT_prednaska_VUE_VUEX