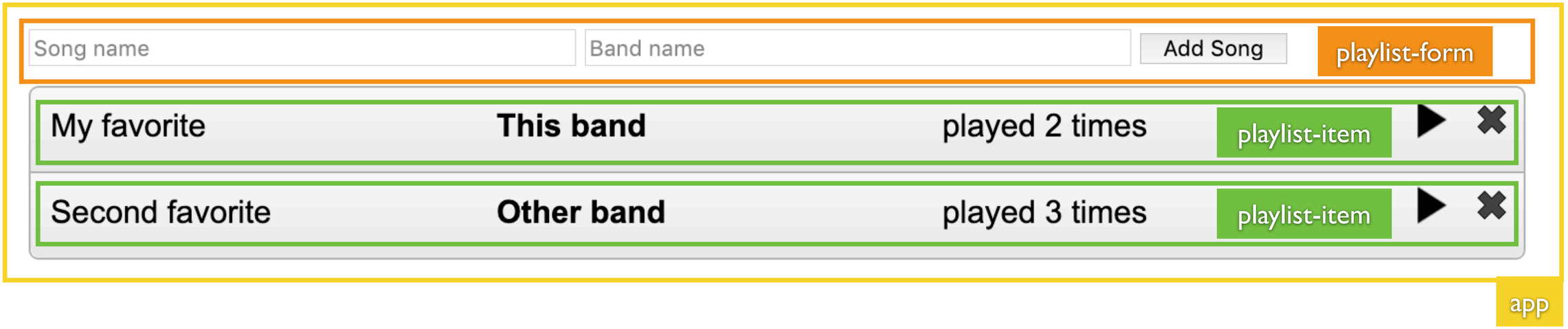
Web Programming

Vue.js III. (components,state, routes)

Leander Jehl | University of Stavanger

Components

- If a Vue application gets too big/bloated we can separate it in multiple components.



Use **components,** do **not** use **multiple app instances.**

Components

- Can have **`state`**, **`methods`**, and **`computed`** properties:

```
app.component('my-counter',{
    template: `
        <div class="counter">
            <span id="count" class="count">{{ count }}</span>
            <button v-on:click="increment">Add</button>
        </div>`,
    data: function(){
        return {count: 0};
    },
    methods: {
        increment: function(){
            this.count++;
        }
    }
})
```

Props: passing values to components

- A component can state properties (props),
  i.e. values it receives from parent component.

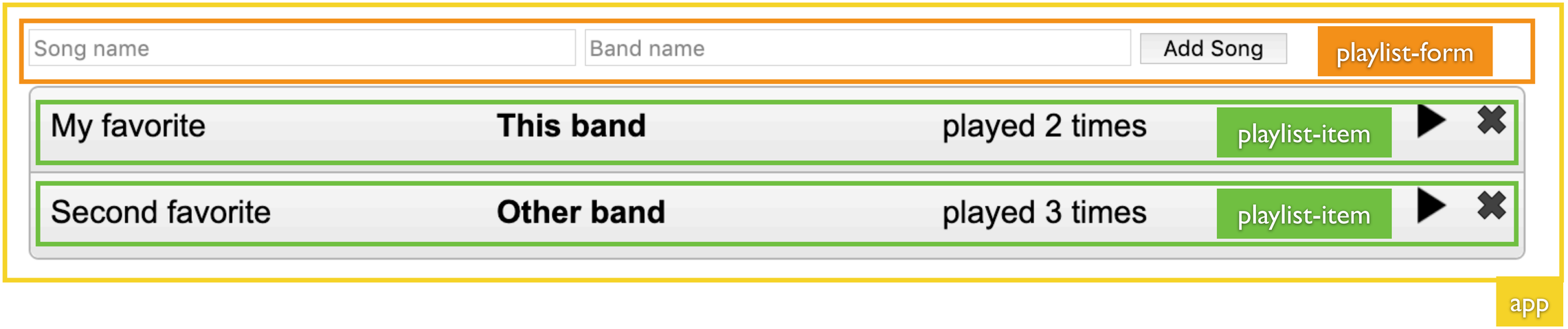
```
app.component('my-box', {
    // specify properties received
    props: [“color", “title”,”nr”],

    // use properties in template
    template: `
    <div class="box"
        v-bind:style="{ backgroundColor: color}">
        #{{ nr }}. {{ title }}
    </div>`
});
```

- Use v-bind on your property to:
  - Define state props based on JS and parent state
  - Reactively update props

```
<my-box v-bind:color=“backgroundColor” title=“Hello” nr=“1” ></my-box>
```

Example #1

examples/js/vue2/list

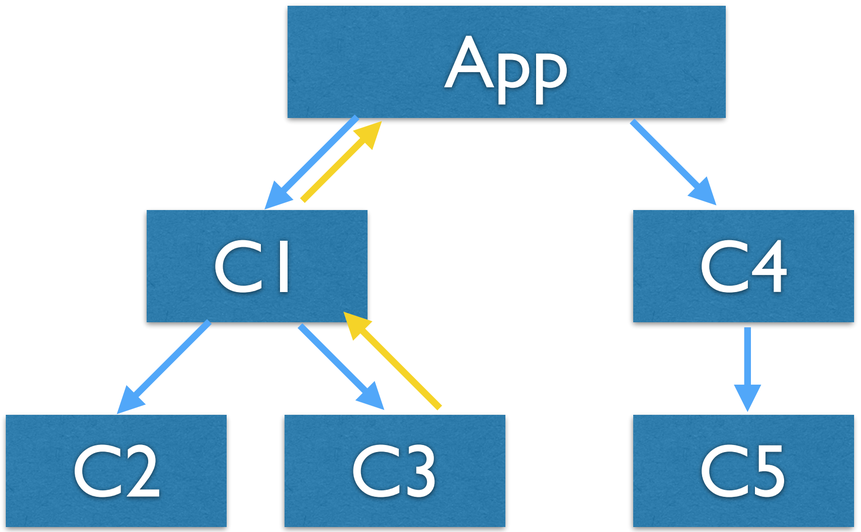


Exercise #1

github.com/dat310-2023/info/tree/master/
exercises/js/vue3

State management

- If multiple components access the same state, it needs to be passed down using props and changed using events.
  - State shared by C3 and C5 must be located in App.
  - If shared state is changed in C3, change is propagated using events and props

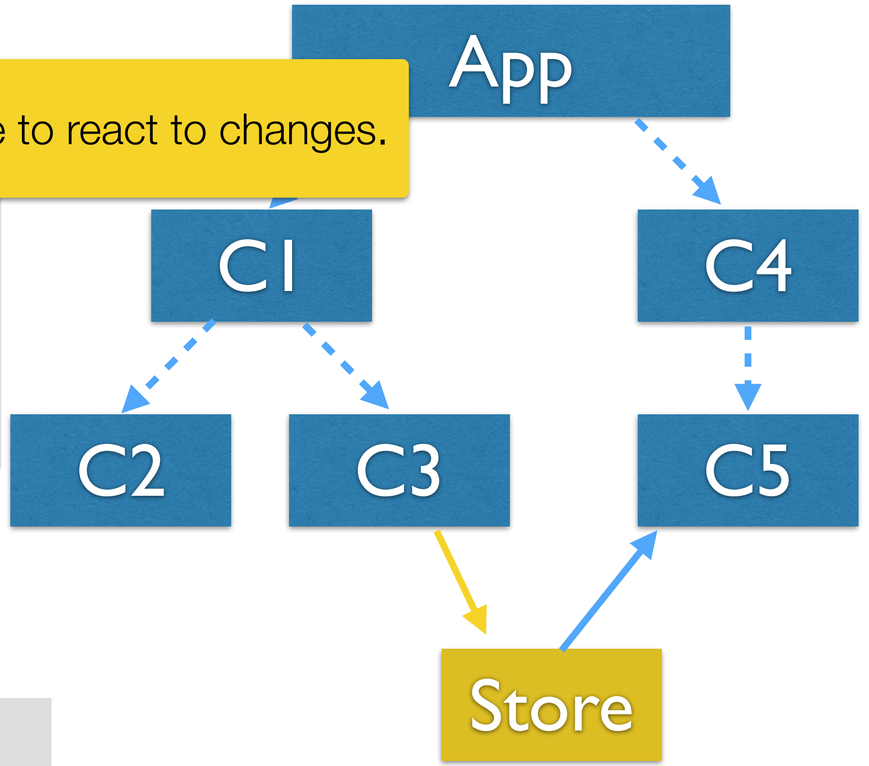


A different pattern: External store

- Outside of your app, define a store:

```
class DataStore{
    constructor(data){
        this.data = data;
    }
    getter(){}
    setter(){}
}
let store = Vue.reactive(new DataStore(data));
```

Vue.reactive( ) allows Vue to react to changes.



- Retrieve data from store,
  e.g. on component creation

```
  data() {
       return store.data;
   }
```

(read the docs)

Example #2

examples/js/vue3/global-state-fruits/index.html

form.js

```
let fruitFormC = { ...
    methods: {
        add: function(){
            store.addFruit(this.name, _);
        }
    }
}
```

Update global state instead of emitting event.

list.js

```
let favoriteC= {
    computed:{
        fruits: function(){
            return store.fruits;
        },
        // using getters inside computed
         // properties works
        favorites: function(){
            return store.favoriteFruits();
        },
    }
}
```

Use getters in computed

data.js

```
class Store{
    constructor(){
        this.fruits = [
            { name: "Apple",   favorite: true },
            { name: "Banana",  favorite: true },
            { name: "Pear",    favorite: true },
            { name: "Grapes",  favorite: false },
            { name: "Oranges", favorite: false },
            { name: "Kiwi",    favorite: false }
        ]
    }
    //getter
    favoriteFruits(){
        return this.fruits.filter(
            (fruit) => fruit.favorite);
    }
    //setter
    addFruit(name, isFavorite){
        this.fruits.push({name: name, favorite:
isFavorite});
    }
}

let store = Vue.reactive(new Store())
```

Use library vuex store

Not curriculum!

- Implements the flux pattern.
  - Lots of debugging features
  - Support for asynchronous updates

```
<script src="https://unpkg.com/vuex@4.0.0/dist/vuex.global.js"></script>
```

Use library vuex store

Not curriculum!

- **`state`** holds **`data`**
- **`getters`** are like **`computed`**
- **`mutations`** are like **`methods`**

```
// Vuex store
let store = new Vuex.Store({
    state: {
      // like data in the vue instance
   },
    getters: {
      // like computed in vue instance,
      //but use state argument instead of this.
   },
    mutations: {
      // mutation takes state as argument,
      // and one additional argument (payload)
   }
})
```

Example #3

examples/js/vue3/global-state-vuex-fruits/index.html

Not curriculum!

list.js

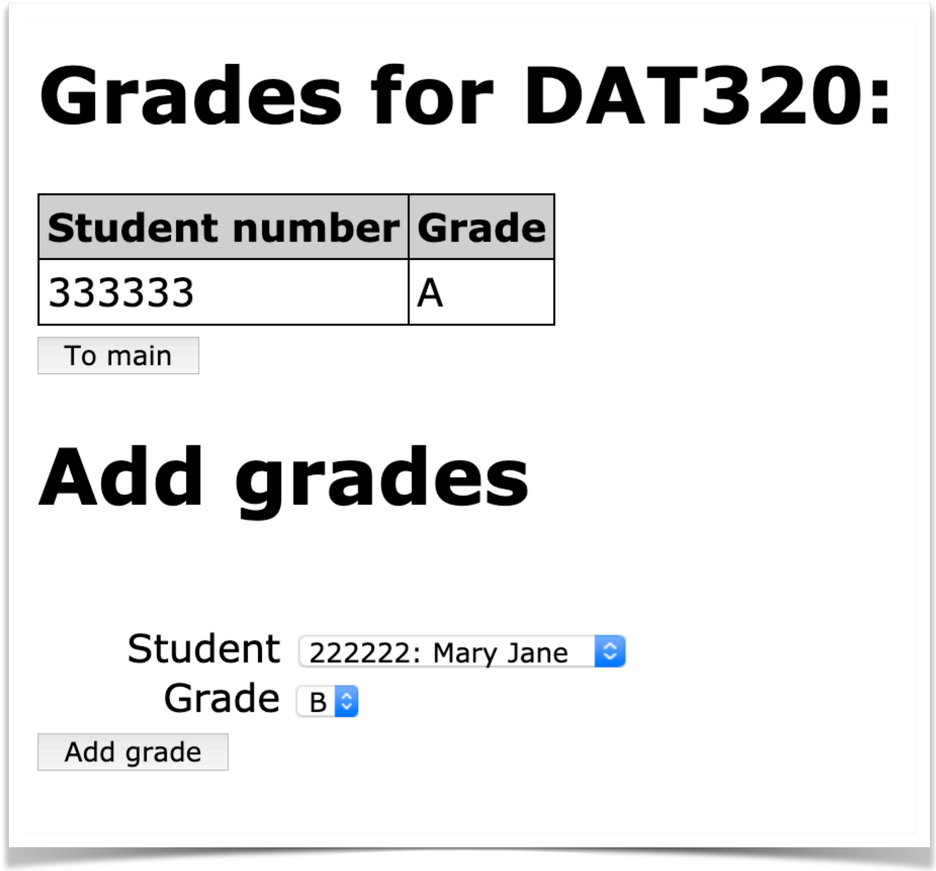
```
let favoritC= {
    computed: {
        // using getters
        // in computed properties
        favorites: function(){
            return store.getters.
                   favoriteFruits },
    },
```

data.js

```
let store = Vuex.createStore({
    state: function(){
       return {
            fruits: [
                { name: "Apple",   favorite: true },
                ...
            ]
        }
    },
    getters: {
        // a getter takes the state as argument.
        favoriteFruits(state){
            return state.fruits.filter(
                (fruit) => fruit.favorite);}
    },
    mutations: {
        // setters are called mutations.
        addFruit(state, fruit){
               state.fruits.push(fruit);
        }
    }
});
```

Example #4

examples/js/vue3/gradebook/index.html



Exercise #1

github.com/dat310-2023/info/tree/master/
exercises/js/vue3

Example #5

examples/js/vue2/global-store-playlist

gstate.js

```
class GState { … }

Let gState = Vue.reactive(
    new GState());

let app = Vue.createApp({
    data() {
        return gState.state;
    }
});
```

songListItem.js

```
Vue.component("song-list-item",{
    props: ['song'],
    template: ...
    methods: {
        remove: function(){
            gState.remove(this.song);
        }
    },
}
```

songForm.js

```
Vue.component("song-form",{
    template: ...
    methods: {
        addSong: function() {
            gState.add(new Song(this.song, this.band));
        }
    },
}
```

Update global state instead of emitting event.

Routing

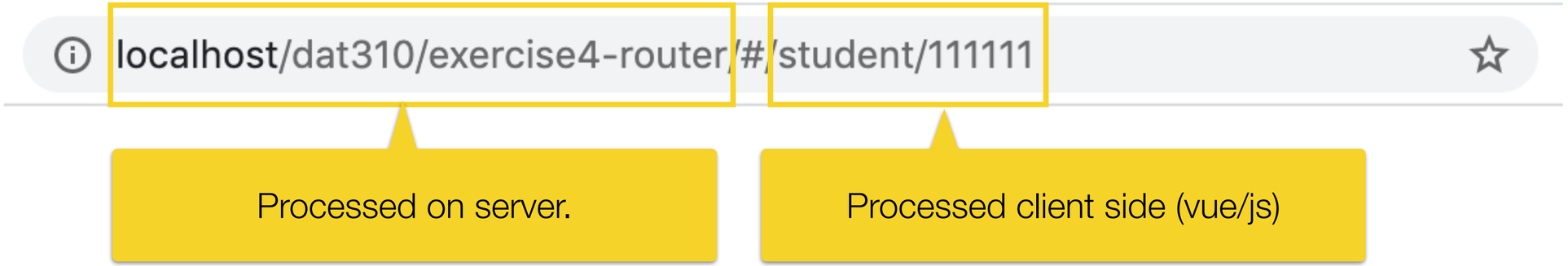
- Components allow to display different “pages”
  - But these are not reflected in the URL
  - Want to bookmark pages
- Use Vue Router

```
<script src=“https://unpkg.com/vue-router@4”></script>
```

Include after vue.js

Routing


Vue list
localhost/dat310/exercise4-router/#/student/111111
Error
Grades for John Smith (111111)
Course
Grade
MAT100
B
MAT200
C
DAT100
B
DAT200
C
DAT220
C
MAT100
B
Error
To main


localhost/dat310/exercise4-router/#/student/111111
Processed on server.
Processed client side (vue/js)

Routing

- Create router

```
let router = VueRouter.createRouter({ ... });
```

- Add router to app:

```
let app = Vue.createApp({});
app.use(router);
```

- Add rout component to template

```
<div id="app">
    <router-view></router-view>
</div>
```

<router-view> is replaced with component, depending on route.

Routes

- Define routes

```
let mainComponent = {
    template: `<div>Main component</div>`,
    // data, computed, ...
}
let helloC = { template: `<div>Hello component</div>` }

let router = VueRouter.createRouter({
    // this will tell vue to use routes including #
    history: VueRouter.createWebHashHistory(),
    routes: [
      { path: '/', component: mainComponent },
      { path: '/favorite', component: helloC},
    ]
});
```

Like components, but no need to call app.component( )

Example #6

examples/js/vue3/fruits-router

```
let router = VueRouter.createRouter({
    // this will tell vue to use routes including #
    history: VueRouter.createWebHashHistory(),
    routes: [
      { path: '/', component: allC },
      { path: '/favorite', component: favoriteC},
    ]
});
```

More routes

- Define routes
  - Parametrized routes, use **`:name`** to define a route parameter

```
{ path: '/student/:id', component: student }
```

  - Access parameters in route component as **`$route.params.name`**

```
let student = { template: '<div>Student {{ $route.params.id }}</div>' }
```

Links

- Use **<router-link to="path"></router-link>**

```
<router-link to="/hello">Hello</router-link>
```

  - Can use **v-bind** to bind parameters

```
<router-link v-bind:to="'/student/' + student_no">Me</router-link>
```

- Named routes

```
{ path: '/student/:id', name: 'student', component: student }
```

  - Pass an object **{ name: '', params: { … }}** to link

```
<router-link v-bind:to="{ name: 'student', params: { id: 123456 }">
```

Exercise #2

github.com/dat310-2023/info/tree/master/
exercises/js/vue3

Navigation in JS

- To move to a different route in JS
  - In event handler in component do **`this.$router.push()`**
  - ```
    this.$router.push(‘/all');
    ```
  - To go back to the last page use **`this.$router.go(-1)`**
  - ```
    this.$router.go(-1);
    ```

Routes with props

- Route parameters can also be passed as props:
  - Set **`props=true;`** in route

```
{ path: ‘/student/:id’, component: student, props: true }
```

  - Parameter will be passed as prop

```
let student = { props: ['id'], template: `<div>Student {{ id }}</div>`}
```

- It is also possible to pass static props to reuse components

```
{ path: '/favorite', component: fruitList, props: { showAll: false } },
{ path: '/all',      component: fruitList, props: { showAll: true } }
```

Exercise #3

github.com/dat310-2023/info/tree/master/
exercises/js/vue3