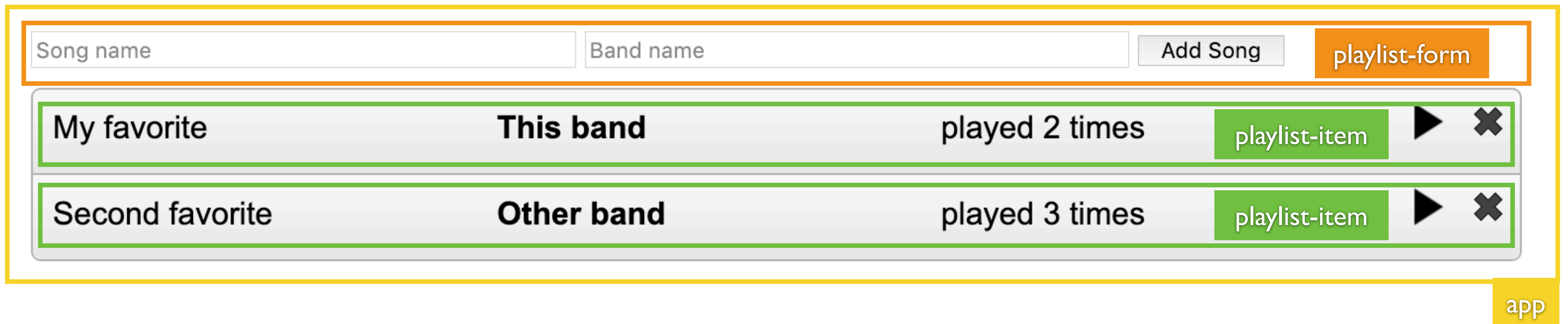


Web Programming

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

# 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](#)

Add Song

playlist-form

My favorite

This band

played 2 times

playlist-item

▶

✕

Second favorite

Other band

played 3 times

playlist-item

▶

✕

app

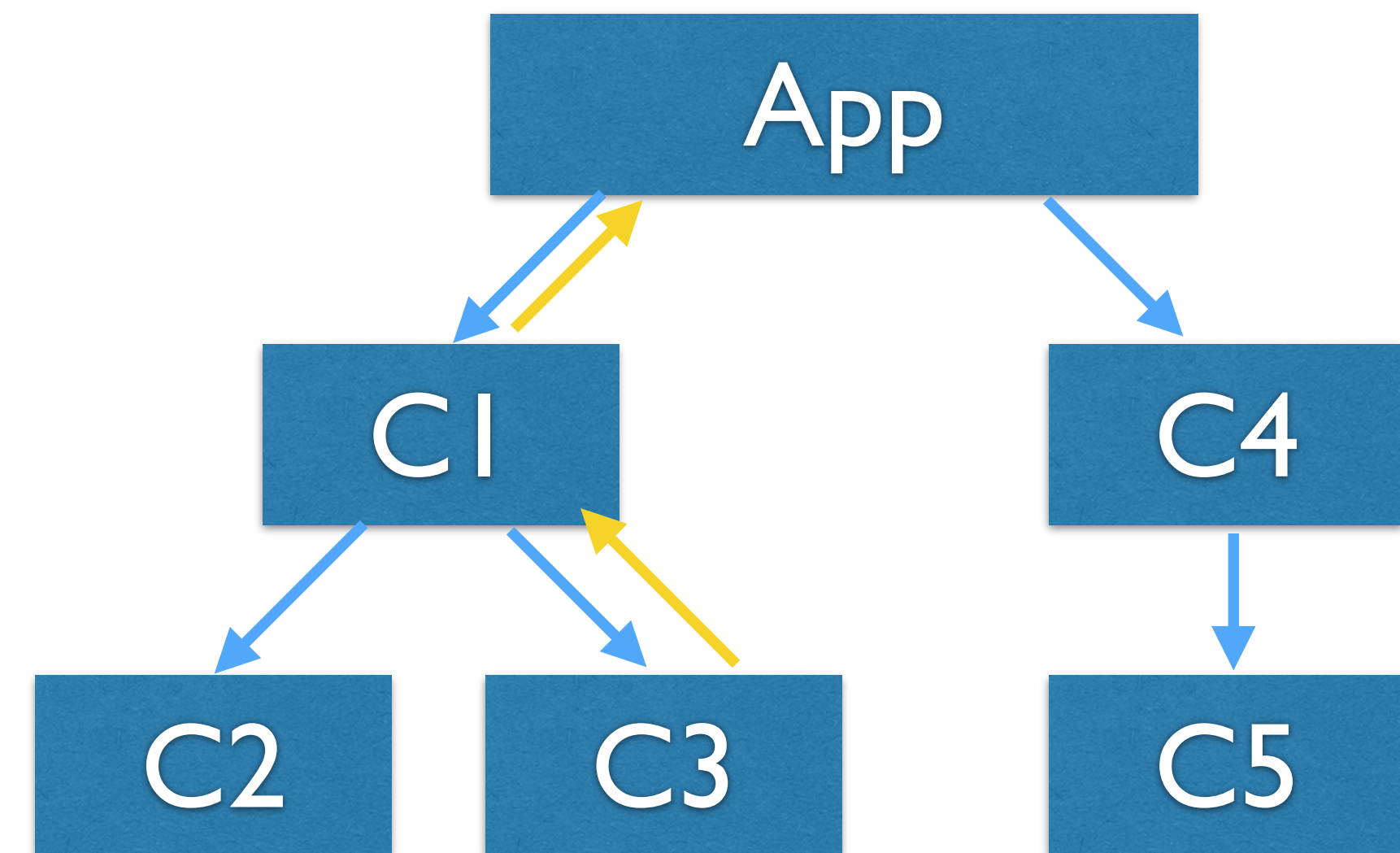
# Exercise #0



[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/js/vue3)  
**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

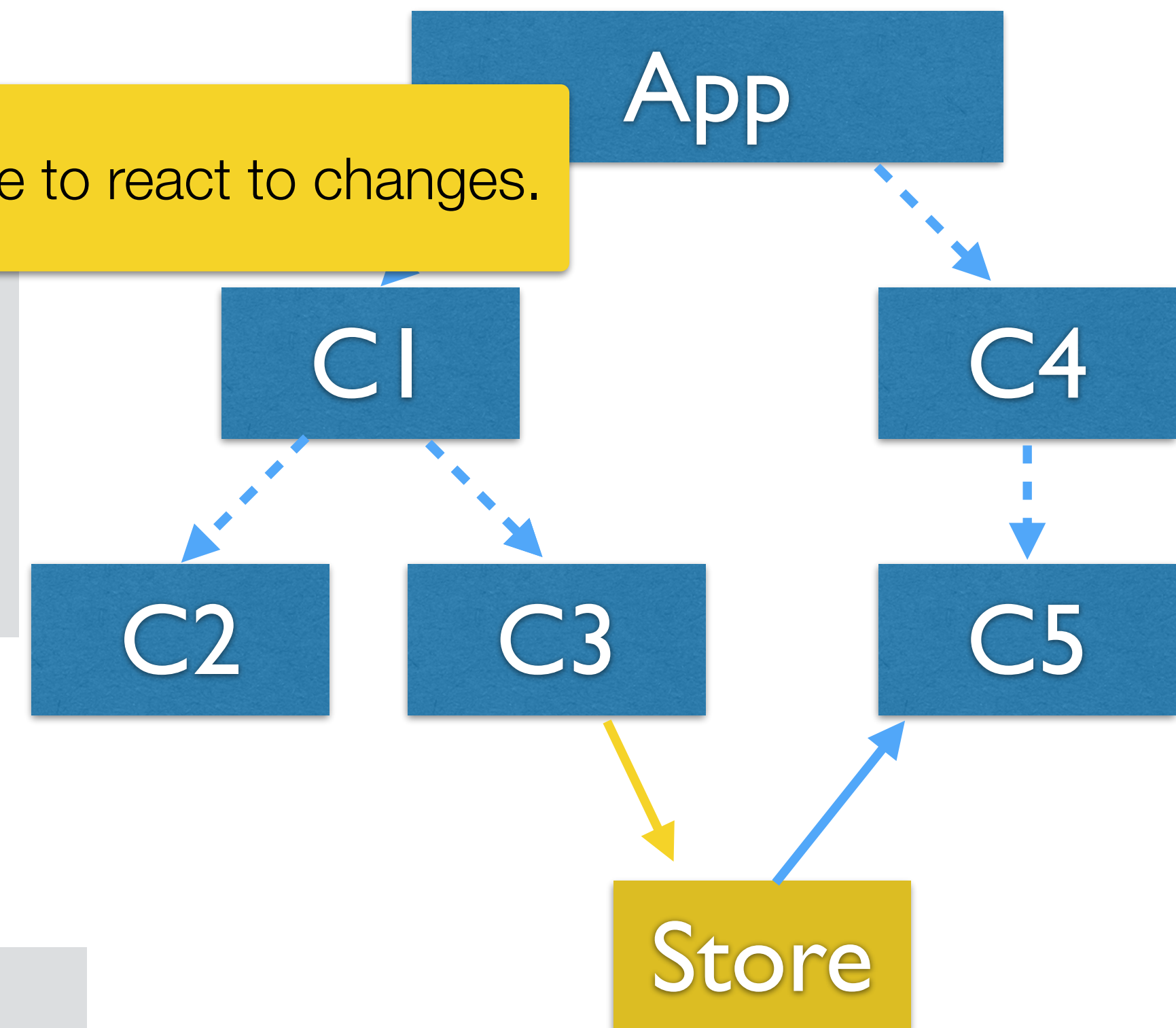
- Outside of your app, define a store:

```
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 #4

🔗 <examples/js/vue3/gradebook/index.html>

## Grades for DAT320:

Student number	Grade
333333	A

[To main](#)

## Add grades

Student

Grade

[Add grade](#)

# Exercise #1



[github.com/dat310-2024/info/tree/master/  
\*\*exercises/js/vue3\*\*](https://github.com/dat310-2024/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

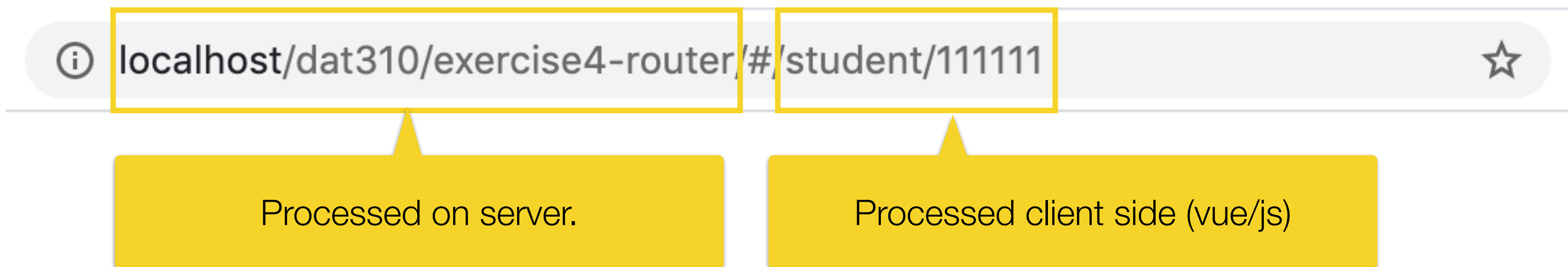
- Components allow to display different “pages”
  - But these are not reflected in the URL
  - Want to bookmark pages
- Use Vue Router

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



Include after vue.js

# Routing



# 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

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

```
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 },  
  ]  
});
```

# 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/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/js/vue3)  
**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/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/js/vue3)  
**exercises/js/vue3**

**Not curriculum!**

This is how you develop in the real world!

# Vite and single file components

- Vite is a tool to set up a new js projects.
  - Uses webpack
  - Web pack avoids including different files in index.html
- Single file components allow to have
  - nicely highlighted templates
  - JavaScript component definition
  - CSS scoped to this component
  - **All in one file**

**Not curriculum!**

This is how you develop in the real world!

# Vite and Single file components

- Requirements:

- Install **node.js** and **npm** <https://nodejs.org/en/download/>

```
~: node -v  
v21.6.1  
~: npm -v  
10.2.4
```

- Create new project and possibly install parts

```
~: npm create vue@latest
```



# Vite and Single file components

## - Configure project

```
[✓] Project name: ... myproject
[✓] Add TypeScript? ... No / Yes
[✓] Add JSX Support? ... No / Yes
[✓] Add Vue Router for Single Page Application development? ... No / Yes
[✓] Add Pinia for state management? ... No / Yes
[✓] Add Vitest for Unit Testing? ... No / Yes
[✓] Add an End-to-End Testing Solution? > No
[✓] Add ESLint for code quality? ... No / Yes
[✓] Add Prettier for code formatting? ... No / Yes
```

```
Scaffolding project in /Users/leanderjehl/dev/dat310/tmp/myproject...
```

```
Done. Now run:
```

```
cd myproject
npm install
npm run format
npm run dev
```

# Vite setup

**Not curriculum!**

This is how you develop in the real world!

## - Folder structure

```
my-test-project
> node_modules    // JS libraries and dependencies, e.g. vue
> public           // Static files, contains index.html
> src              // All your code is here
index.html         // A minimal html file
package.lock.json  // Dependency versions (for npm)
package.json       // npm configuration
README.md
vite.config.js      // vite config
```

## - src folder

```
src
> assets           // More static assets, e.g. images
> components       // Your components
App.vue            // Main component
main.js            // create and mount app
```

**Not curriculum!**

This is how you develop in the real world!

# Single file components

- Components can now be specified in .vue files:

```
<template>
  <!-- The template for your component -->
  <form>
    <input type="text" v-model="song">
    ...
  </form>
</template>

<script>
  // define your component in JavaScript
</script>

<style scoped>
  // CSS queries applied only to this component
</style>
```

**Not curriculum!**

This is how you develop in the real world!

# ES6 import and export

- Using CLI, components are not defined globally,

```
// globally defined component:  
app.component("song-form",{ });
```

- Instead the definition of a component is exported

## SongForm.vue

```
// export component configuration  
export default {  
  template: ...  
  methods: ...  
};
```

Only one default export per file.

## App.vue

```
// import component  
import songForm from './components/SongForm'  
  
export default {  
  template: ...,  
  // use songForm in this component  
  components: {  
    songForm,  
  }  
};
```

**Not curriculum!**

This is how you develop in the real world!

# Example #6

 [examples/js/vue-vite/playlist-vite](#)

```
../playlist: npm run dev
```

Starts a development server, serving your app.

```
<template>
  <div id="app">
    <song-form></song-form>

    <ul id="playlist">
      <song-list-item
        v-for="(song, index) in playlist"
        v-bind:song="song"
        v-bind:index="index"
        v-bind:key="index"
      ></song-list-item>
    </ul>
  </div>
</template>
```

```
<script>
import gState from './data.js'

import songForm from './components/SongForm'
import songListItem from './components/SongListItem'

export default {
  name: 'App',
  data: function(){
    return {
      playlist: gState.playlist,
    }
  },
  components: {
    songForm,
    songListItem
  }
}
</script>
```

**Not curriculum!**

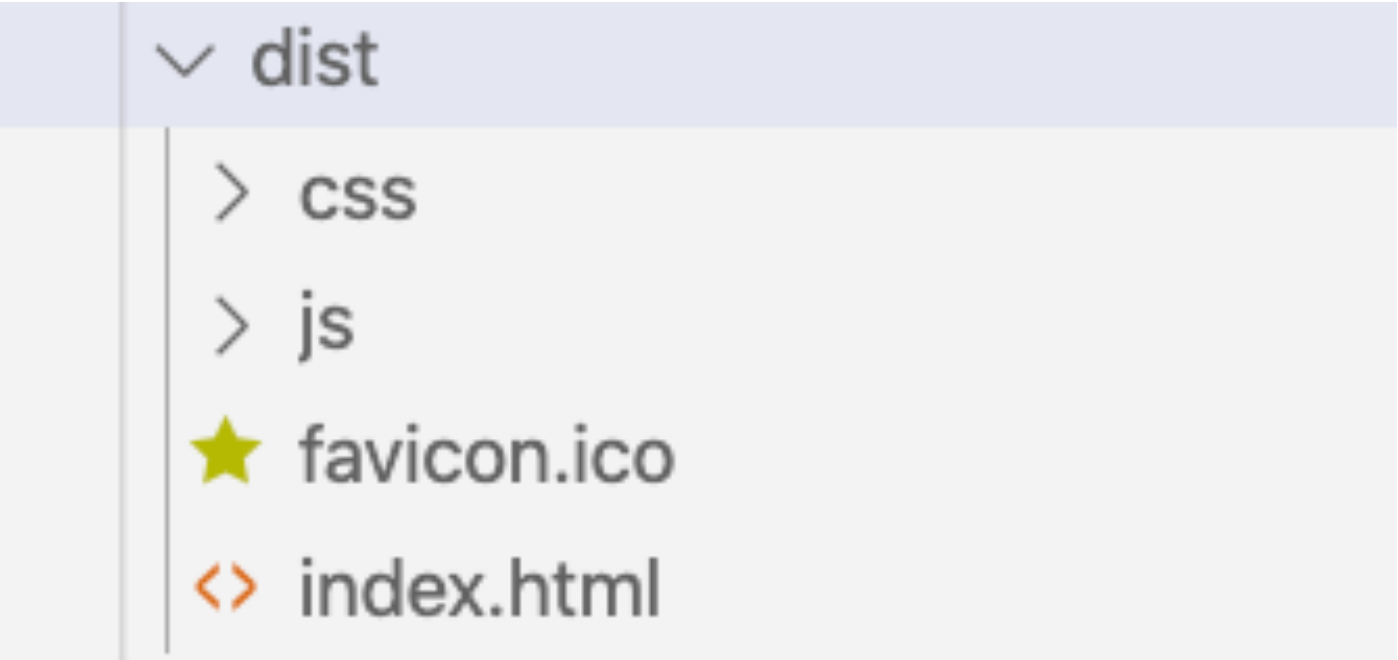
This is how you develop in the real world!

# Submitting

- Run: **npm run build**

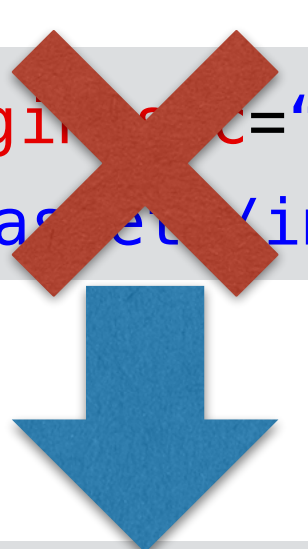
```
../playlist: npm run build
```

- Submit files from **dist** folder



```
▼ dist  
  > css  
  > js  
  ★ favicon.ico  
  <> index.html
```

- Fix paths



```
<script type="module" crossorigin src="/assets/index-xxx.js"></script>  
<link rel="stylesheet" href="/assets/index-xxx.css">
```

```
<script type="module" crossorigin src="assets/index-xxx.js"></script>  
<link rel="stylesheet" href="assets/index-xxx.css">
```

You need to submit both  
distribution to be run and code.

**Not curriculum!**

This is how you develop in the real world!

# Example #7

🔗 `examples/js/vue-vite/grades-router`

```
../playlist: npm run serve
```

Starts a development server, serving your app.

**App.vue**

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

**router/index.js**

```
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/student/:student_no',
    name: 'Student',
    props: true,
    component: Student
  },
  {
    path: '/course/:course_id',
    name: 'Course',
    props: true,
    component: Course
  }
]
```