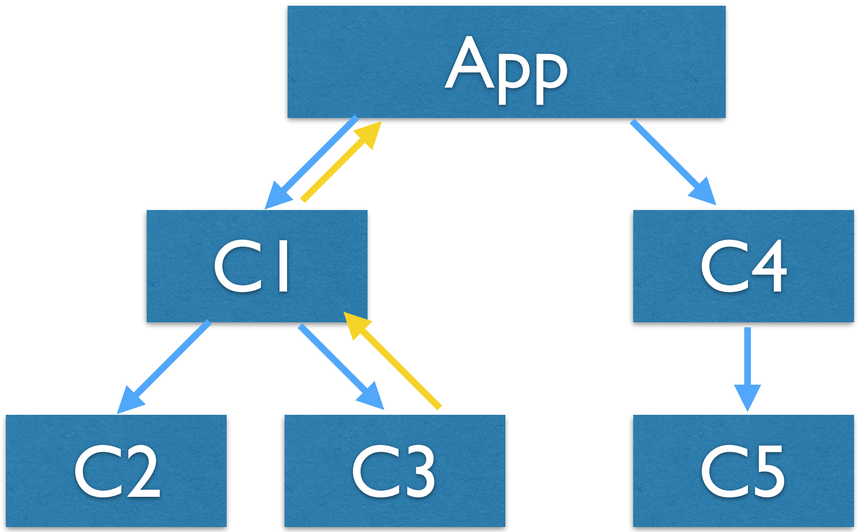
Web Programming

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

Leander Jehl | University of Stavanger

State management cont'd

- 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

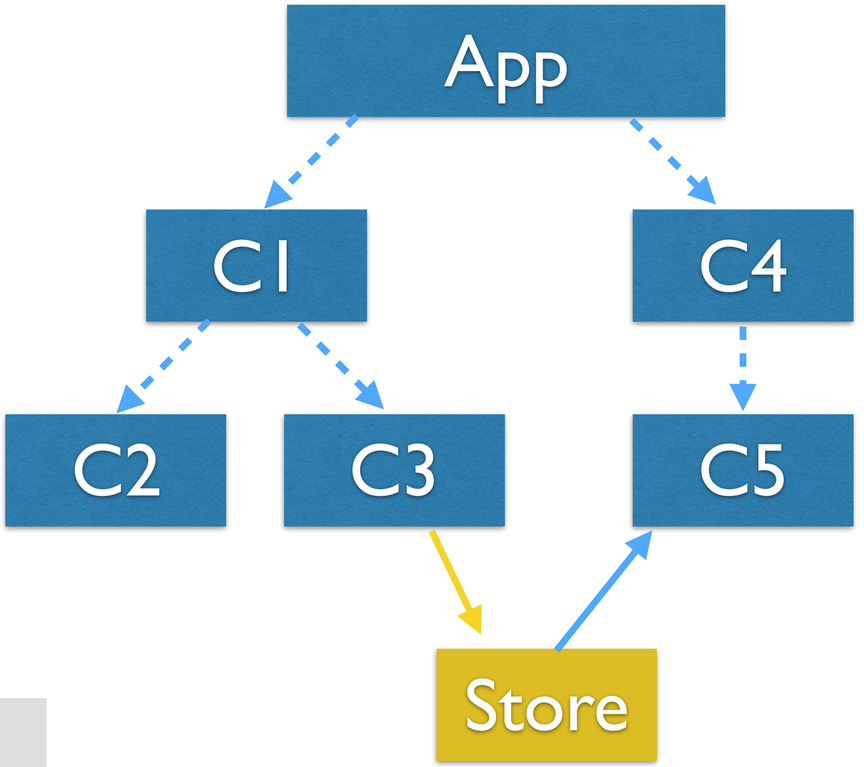
- Outside of your app, define a store:

```
function DataStore(data){
    this.data = data;
    this.getter = function(){}
    this.setter = function(){}
}

let store = new DataStore(data);
```

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

```
   data: {
        course: store.course(this.course_id)
   }
```



Example #1

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

list.js

```
// this component is not reactive
Vue.component("my-favorites",{
    computed: {
        favorites: function(){
            return store.favoriteFruits();
        }
    },
```

Problem: this is not updated reactively.

data.js

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

const store = new DataStore();
```

A vue instance as store

- As store, use a separate vue instance.
  - Store instance does not need template or data element.

```
function DataStore(data){
    this.data = data;
    this.getter = function(){}
    this.setter = function(){}
}

let store = new DataStore(data);
```

```
let store = new Vue({
    data: { data  },
    computed: {
        // getters here
   },
    methods: {
        // setters here
   }
});
```

Example #2

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

data.js

```
let store = new Vue({
    // el: "#app",
    data: {
        fruits: [
            { name: "Apple",   favorite: true },
           ...]
    },
    computed: {
        // getter
        favoriteFruits: function(){
            return this.fruits.filter(
                (fruit) => fruit.favorite);
        }
    },
    methods: {
        // setter
        addFruit(name, isFavorite){
            this.fruits.push(
                {name: name, favorite: isFavorite});
        }
    }
});
```

list.js

```
// this component is reactive
Vue.component("my-favorites",{
    computed: {
        favorites: function(){
            return store.favoriteFruits;
        }
    },
```

This will update reactively.

Use library vuex store

Not curriculum!

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

```
<script src="https://unpkg.com/vuex@3.1.2/dist/vuex.js"></script>
```

Use library vuex store

Not curriculum!

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

```
// Vue component as store
let store = new Vue({
    data: { data  },
    computed: {
        // getters here
   },
    methods: {
        // setters here
   }
});
```

```
// 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!

index.html

```
let app = new Vue({
        el: "#app",
        store: vuexStore
    })
```

list.js

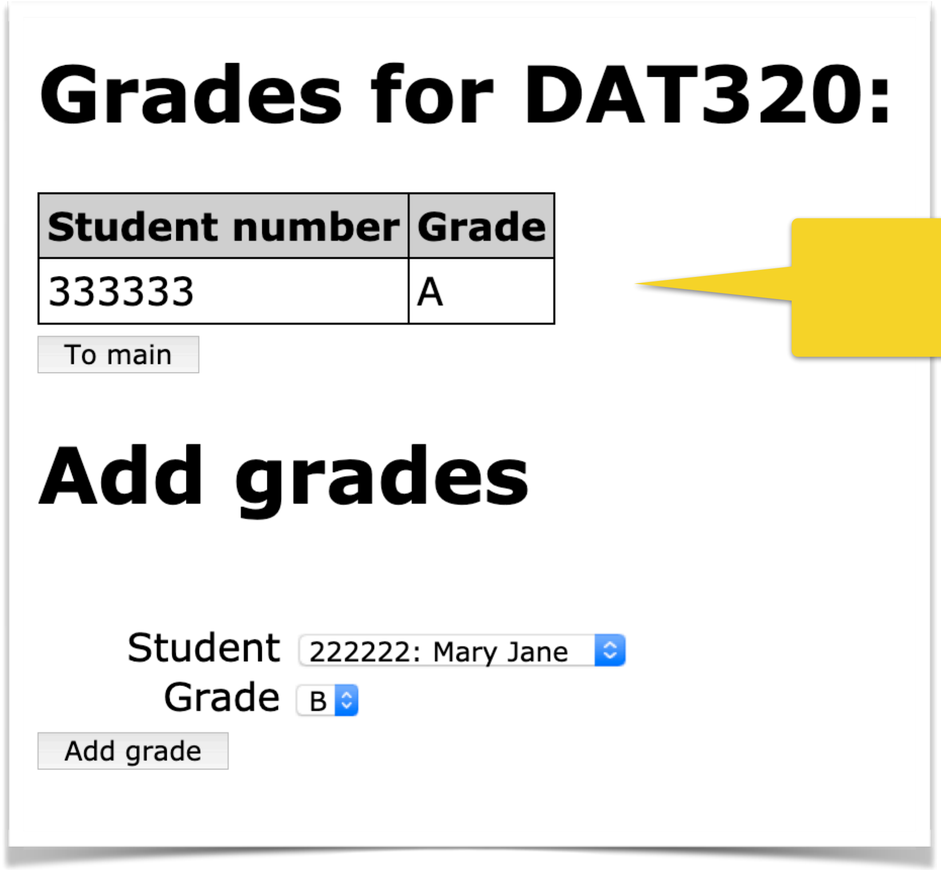
```
Vue.component("my-favorites",{
    computed: {
        favorites: function(){
            return this.$store
                       .getters
                       .favoriteFruits;
        }
    },
```

data.js

```
const vuexStore = new Vuex.Store({
    state: {
        // like data in the vue instance
        fruits: [
            { name: "Apple",   favorite: true },
           ...]
    },
    getters: {
        // like computed in vue instance, but
        // use state argument instead of this.
        favoriteFruits: function(state){
            return state.fruits.filter(
                (fruit) => fruit.favorite);
        }
    },
    mutations: {
        // mutation takes state as argument,
        // and one additional argument (payload)
        addFruit(state, newFruit){
            state.fruits.push(newFruit);
        }
    }
});
```

Example #4

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



Problem: Table is not reactive.

Exercise #1, (#1b)

github.com/dat310-spring20/course-info/tree/master/
exercises/js/vue3

Routing

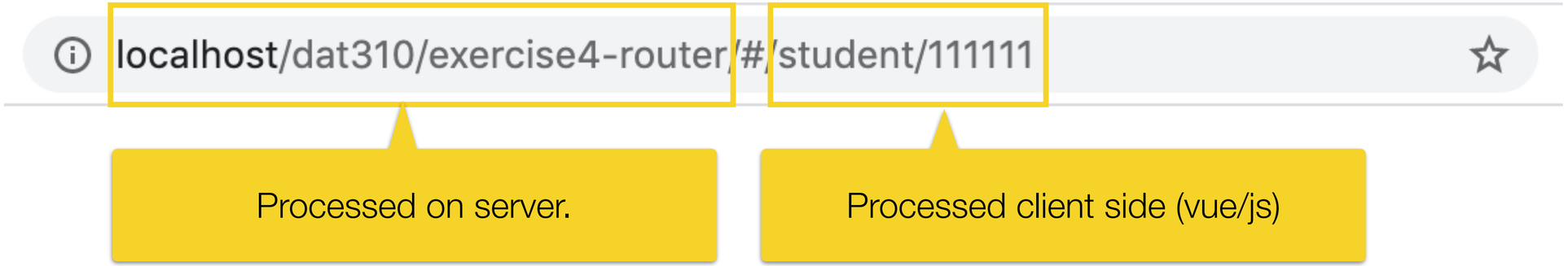
- Components allow to display different “pages”
  - But these are not reflected in the URL
  - Want to bookmark pages
- Use Vue Router

```
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></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 = new VueRouter({ … });
```

- Add router to app:

```
new Vue({ el: "#app", router: 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 hello = { template: `<div>Hello component</div>` }

let router = new VueRouter({
    routes: [
      { path: '/', component: mainComponent },
      { path: '/hello', component: hello }
    ]
});
```

Like components, but without **Vue.component**.

More routes

- Define routes
  - Match all other routes

```
{ path: '*', component: unknown }
```

  - 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 }">
```

Example #5

examples/js/vue3/fruits-router

```
let router = new VueRouter({
    routes: [
      { path: '/', component: myFavorites },
      { path: '/all', component: allFruits}
    ]
});
```

Exercise #2

github.com/dat310-spring20/course-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-spring20/course-info/tree/master/
exercises/js/vue3

Not curriculum!
This is how you develop in the real world!

CLI and single file components

- CLI is a tool to set up a new vuejs project.
  - Uses webpack
- 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!

CLI and Single file components

- Requirements:
  - Install **`node.js`** and **`npm`** https://nodejs.org/en/download/

```
~: node -v
v12.10.0
~: npm -v
6.11.3
```

  - Install vue cli

```
~: npm -g install @vue/cli
```

  - Create new project

```
~: vue create my-test-project
```

  - Choose default tools

```
? Please pick a preset: default (babel, eslint)
```

VSCode: Install Extension Vetur

Not curriculum!
This is how you develop in the real world!

Vue CLI setup

- 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
 babel.config.js  // Babel configuration
 package.lock.json // Dependency versions (for npm)
 package.json      // npm configuration
 README.md
 vue.config.js     // add this file
```

- src folder

```
src
 > assets          // More static assets, e.g. images
 > components      // Your components
 App.vue           // Main component
 main.js           // Dependency versions (for npm)
```

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:
Vue.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/vue3/playlist-CLI

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

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

- Add config file **`vue.config.js`**

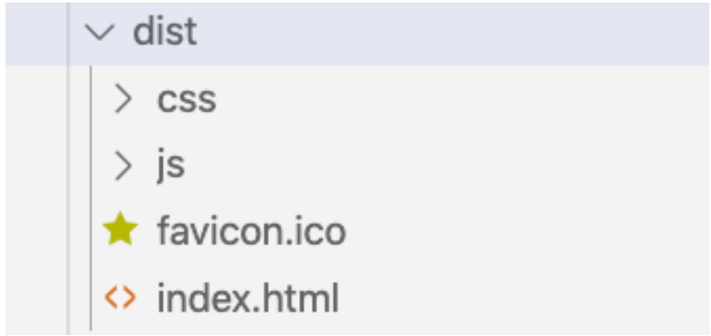
```
// vue.config.js
module.exports = {
    // change to relative path
    publicPath: './'
}
```

- Run: **`npm run build`**

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

You can submit like this.
In grading/approving, we will not consider your code, only functionality.

- Submit files from **`dist`** folder



Exercise #4, #4b

github.com/dat310-spring20/course-info/tree/master/
exercises/js/vue3

Routing and vuex

Not curriculum!
This is how you develop in the real world!

- You can choose a setup including vuex and routing

```
~: vue create my-test-project2
```

```
? Please pick a preset:
  default (babel, eslint)
❯ Manually select features
```

```
? Check the features needed for your project:
 ◉ Babel
 ○ TypeScript
 ○ Progressive Web App (PWA) Support
 ◉ Router
❯◉ Vuex
 ○ CSS Pre-processors
 ◉ Linter / Formatter
 ○ Unit Testing
 ○ E2E Testing
```

Select Vuex and Router using <space>

Not curriculum!
This is how you develop in the real world!

Vue CLI setup

- Folder structure with router and vuex (store)
  - src folder

```
src
 > assets            // More static assets, e.g. images
 > components        // Your components
 > router
   index.js          // Define you routes here
 > store
   index.js          // Add state mutations getters,… here
 > views             // Usually holds components that are used for routing
 App.vue             // Main component
 main.js             // Dependency versions (for npm)
```

Not curriculum!
This is how you develop in the real world!

Example #7

examples/js/vue3/grades-router-vuex-CLI

```
../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
  }
]
```