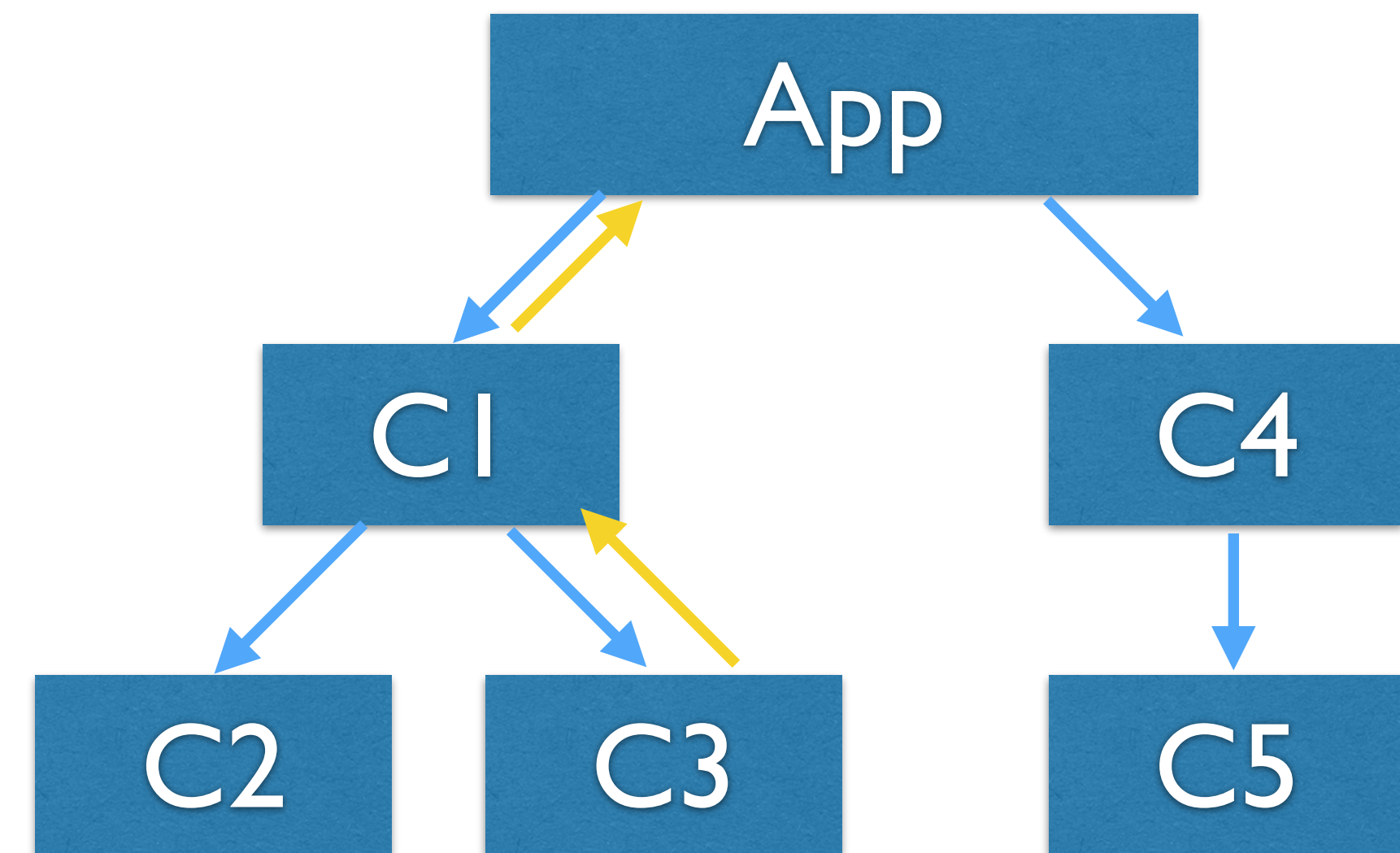


# Web Programming

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

# 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



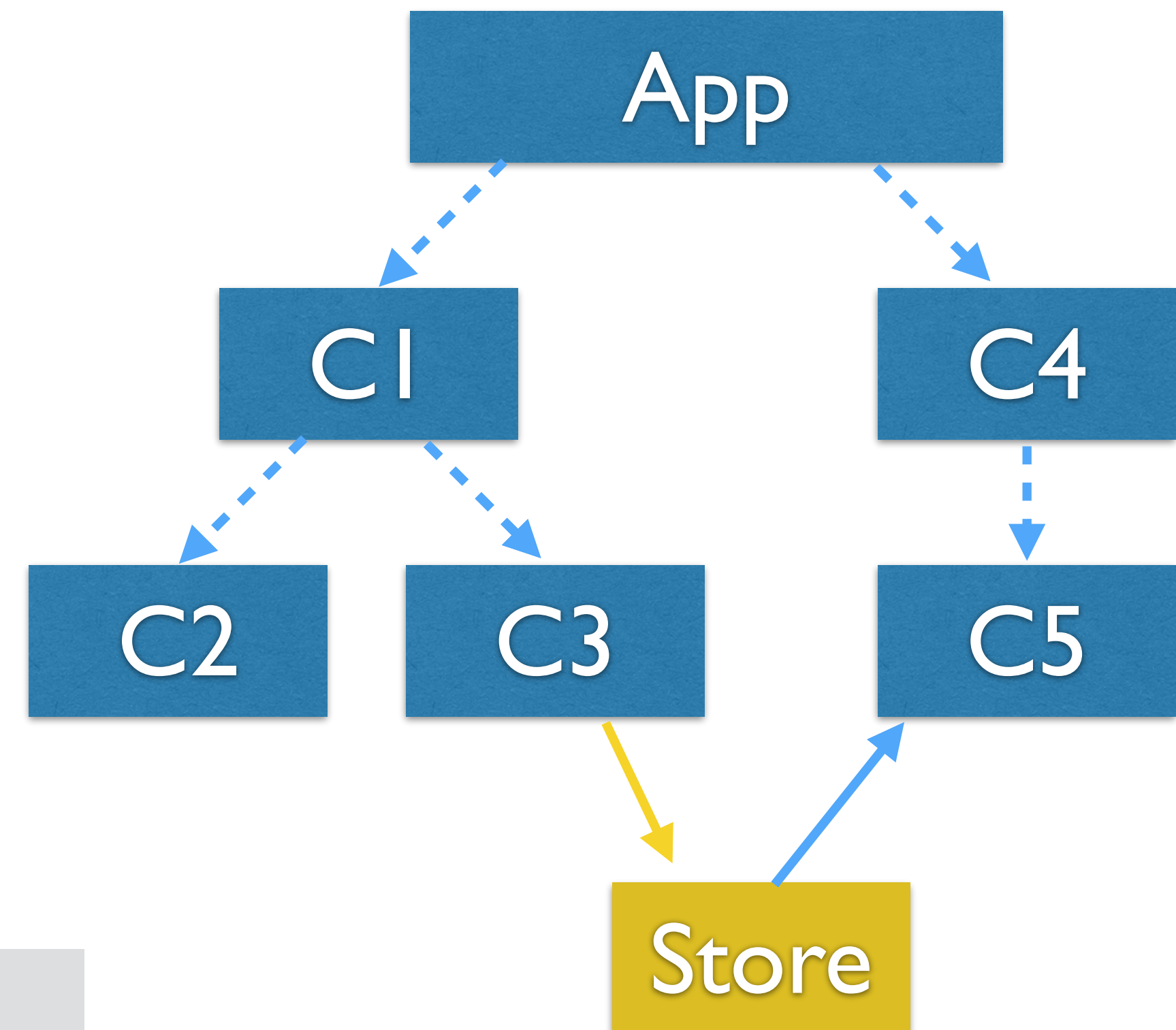
# A different pattern: External store

- Outside of your app, define a store:

```
function DataStore(data){  
  this.data = Vue.reactive(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-fruits/index.html>

list.js

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

Use in reactive properties

data.js

```
let store = {
  fruits: Vue.reactive([
    { 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: function(){
    return this.fruits.filter(
      (fruit) => fruit.favorite);
  },
  //setter
  addFruit(name, isFavorite){
    this.fruits.push({name: name, favorite:
isFavorite});
  }
};
```

# 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](https://github.com/vuejs/vue3-examples/blob/master/js/vue3/global-state-vuex-fruits/index.html)

Not curriculum!

data.js

list.js

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

```
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>

## Grades for DAT320:

Student number	Grade
333333	A

To main

## Add grades

Student 222222: Mary Jane

Grade B

Add grade

**Problem:** Table is not reactive.



# Exercise #1



[github.com/dat310-2022/info/tree/main/](https://github.com/dat310-2022/info/tree/main/exercises/js/vue3)  
**exercises/js/vue3**

# Routing

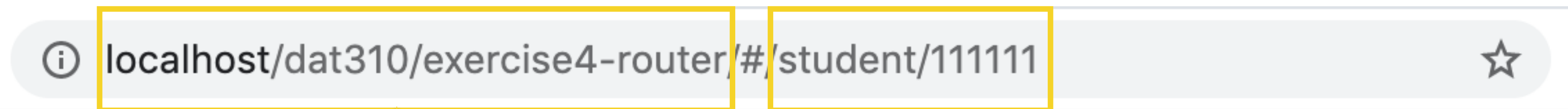
- Components allow to display different “pages”
  - But these are not reflected in the URL
  - Want to bookmark pages
- Use Vue Router

```
<script src="https://unpkg.com/vue-router@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 #5

🔗 [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

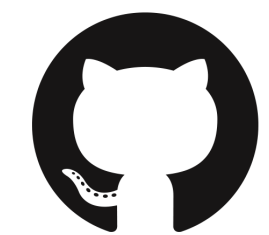
```
{ path: '/student/:id', name: 'student', component: student }
```

- Pass an object `{ name: '', params: { ... } }` to link

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



# Exercise #2



[github.com/dat310-2022/info/tree/main/](https://github.com/dat310-2022/info/tree/main/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/dat310-2022/info/tree/main/](https://github.com/dat310-2022/info/tree/main/exercises/js/vue3)  
**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
  - 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!

# 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
```

may have to run as root

- Create new project

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

- Choose default tools

```
? Please pick a preset:  
  Default ([Vue 2] babel, eslint)  
> Default (Vue 3 Preview) ([Vue 3] babel, eslint)
```

**VSCode:** Install Extension Vetur

# Vue CLI setup

**Not curriculum!**

This is how you develop in the real world!

**VSCode:** Install Extension Vetur

## - 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          // 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/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: './'
}
```

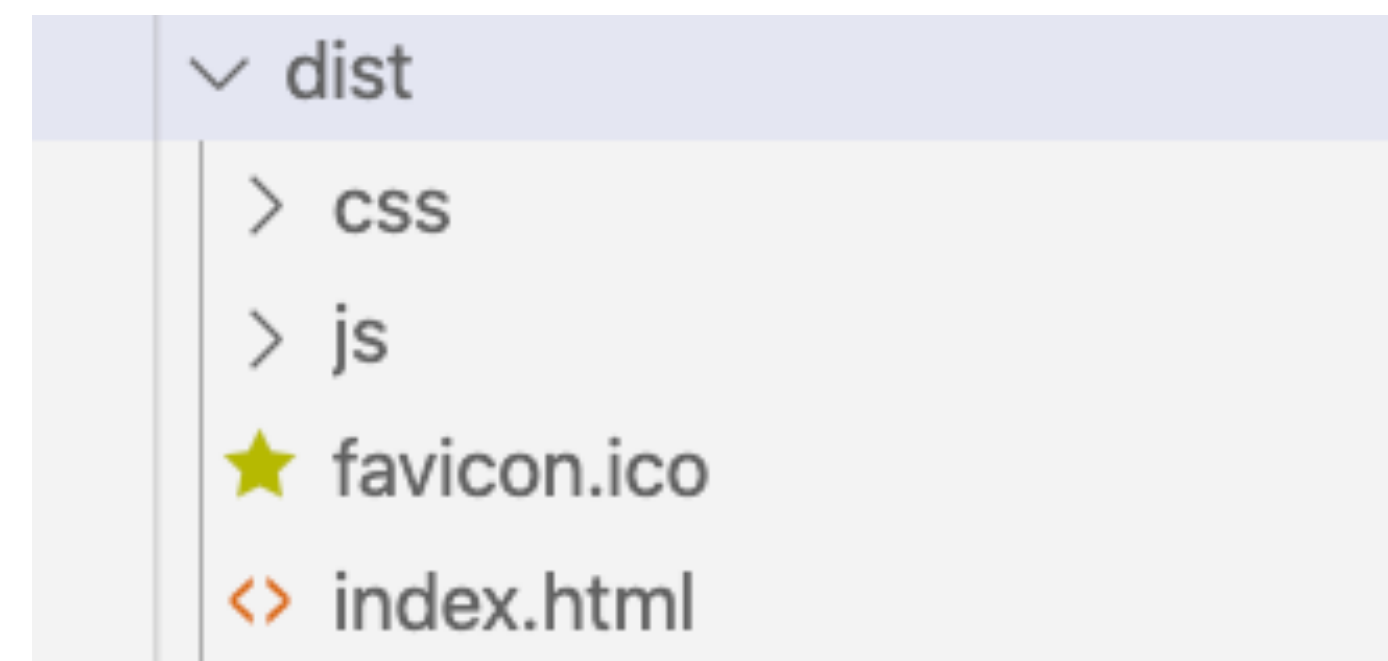
- Run: **npm run build**

```
../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/dat310-2022/info/tree/main/](https://github.com/dat310-2022/info/tree/main/exercises/js/vue3)  
**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 your 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
  }
]
```