Vue.js components and routes

To complete this session, we assume that Node, NPM and Vue.js have been successfully installed, and you have created a Vue.js project. If you did not, please refer to the material of practical session 8.

1. Run your newly created website/project
   
   > npm run serve

2. If everything is ok, you’ll get the following
   App running at:
   - Local:  http://localhost:8080/
   - Network: http://192.168.1.147:8080/

3. Ctrl + press on http://localhost:8080/, and your website will open in the browser

4. Create a new view by right-clicking on the views folder, then, give it the name Products.vue

5. Copy the following code into the newly created Products.vue view

   ```html
   <template>
   <div class="products">
   <h2> Our Products page</h2>
   </div>
   </template>
   ```

6. Define a new route to the newly created view by adding the following code to the index.js file that is located within router folder

   ```javascript
   // do not forget to import the products view by adding the following line
   import Products from "./views/Products.vue";
   ...
   const routes = [
   ...
   {
   path: "/products",
   name: "products",
   component: Products
   },
   ...
   ];
   ```
7. Add a route link that allows navigating to the `Products.vue` by adding the following line of code (highlighted with red color) to the route links in the template section of the `App.vue`.

```html
<template>
  <div id="app">
    <div id="nav">
      <router-link to="/">Home</router-link> | 
      <router-link :to="{ name: 'products' }"> Products </router-link> | 
      <router-link to="/about">About</router-link>
      <router-view />
    </div>
  </div>
</template>
```

**Congratulations!** you have just created your first route to your newly created view, and you should see a new route link (Products) among the Home and About links.

If you press on the Products link, you should be directed to the Products page, which should looks like this:

![Welcome to Your Vue.js App](image)

8. Copy the following code into the `Products.vue` view

```html
<template>
  <div class="products">
    <h2> Our Products page </h2>
    <product_compo_1 :productList="productList">
    </product_compo_1>
  </div>
</template>
```
6. Create two new components by right-clicking on the components folder, then, give them the names `product_compo_1.vue` and `product_compo_2.vue` respectively.

7. Copy the following code into `product_compo_1.vue`

```html
<template>
  <div id="comp-list-one">
    <h1>Book list one</h1>
    <ul>
      <li v-for="product in productList" :key="product.id">
        <span class="author">Author: {{product.author}}</span><br>
        <span class="book">Book: {{product.book}}</span><br>
        <span class="goodreads">Goodreads: {{product.goodreads}}</span><br>
        <span class="price">Price:{{product.price}}</span>
      </li>
    </ul>
  </div>
</template>
<script>
export default {
  name: "product_compo_1",
  components: { product_compo_1, product_compo_2 },
  props: ["productList"],
  data() {
    return {
      productList: [
    }
  }
}
</script>
```
8. Copy the following code into product_compo_2.vue

```html
<template>
  <div id="comp-list-two">
    <h1>Book list two</h1>
    <p v-for="product in productList" :key="product.id">
      <span class="author">Author: {{product.author}}</span>
      <span class="book">Book: {{product.book}}</span>
      <span class="goodreads">Goodreads: {{product.goodreads}}</span>
      <span class="price">Price: € {{product.price}}</span>
    </p>
  </div>
</template>

<script>
export default {
  props: ['productList'],
  data: function() {
    return {  
  }}

  <style scoped>
    .item{
      background: rgb(47, 255, 141);
      margin-bottom: 5px;
      padding: 3px 5px;
    }
    #comp-list-one{
      background: #ffbb1b1;
      box-shadow: 1px 2px 3px rgba(0,0,0.2);
      margin-bottom: 30px;
      padding: 10px 20px;
    }
    #comp-list-one ul{
      padding: 0;
    }
    #comp-list-one li{
      display: inline-block;
      margin-right: 10px;
      margin-top: 10px;
      padding: 20px;
      background: rgba(255,255,255,0.7);
    }
    .price{
      font-weight: bold;
      color: #E8800C;
    }
  </style>
```
By now, the Products page should looks like this

![Products page](image-url)
Installing Vuex

If you followed the installation instructions in Practical session 8, you should have Vuex installed in your IDE (VS Code).

If you did not install it, you can do that now

`> npm install vuex --save`

To be sure it has been installed successfully, open your `package.json` file and you should see it installed in your dependencies.

If the `store` folder is not showing, you need to create it yourself and you need to create a `index.js` file and add the following script to the `index.js` file as shown in the picture.
If your store is not imported into the `main.js` file, import it manually by adding the scripts highlighted in the following picture to the `main.js` file.

```javascript
import Vue from "vue";
import Vuex from "vuex";
Vue.use(Vuex);
export default new Vuex.Store({
});
```
Creating a central store for your App

By now, you should have Vuex installed

1. Move the `productList` array from `Products.vue` file to the `state` section within `index.js` file in the `store` folder, i.e., delete it from `Products.vue` file and add it to the `index.js` file as follows:

```javascript
import Vue from "vue"
import Vuex from "vuex"

Vue.use(Vuex);

export default new Vuex.Store({
  state: {
    productList: [
    ]
  }
});
```

2. Remove the `props:` (highlighted in `red` in the following code) from `product_compo_1.vue` and `product_compo_2.vue` since we do not need them anymore.

```html
<template>
  <div id= "comp-list-two">
    ...
  </template>

<script>
export default {
  name: "product_compo_1",
  props: ["productList"],
  data: function() {
    return {
    }
  })
</script>

<style scoped>
  ...
</style>
```
Note: we are using `product_compo_1.vue` in the script up, yet you should do the same to `product_compo_2.vue`.

3. Remove the passed data (highlighted in red in the following code) from the components call in the `Products.vue` file since we also do not need to pass the data in this way anymore.

```html
<template>
  <div class="products">
    <h2>Our Products page</h2>
    <product_compo_1 :productList="productList" />
    <product_compo_2 :productList="productList" />
  </div>
</template>
<script>
  ...
</script>
<style scoped>
  ...
</style>
```

4. Since we are not receiving the data as a prop anymore, we need to reach out to the store to get the data. This can be done by adding the following code in both `product_compo_1.vue` and `product_compo_2.vue`

```html
<script>
  export default {
    name: "product_compo_1",
    data: function() {
      return {
        productList: () => {
          return this.$store.state.productList
        }
      }
    },
    computed: {
      productList() {
        return this.$store.state.productList
      }
    }
  }
</script>
```

Note: you should remember how we define a computed property (productList()). 
$store.state.productList refers to the array you have defined in the state of the store.

By now, your page should look exactly the same as it used to look before using Vuex.
5. Concerning **getters**, add the following code into the computed properties section in the script section of the `product_compo_1.vue` component.

```javascript
<script>
export default {
  name: "product_compo_1",
  data: function() {
    return {
    }
  },
  computed: {

    productListsale(){
      var productListsale = this.$store.state.productList.map(product => { 
        return{ author: "** " + product.author+ " **",
          price: product.price/2,
          book: product.book,
          goodreads: product.goodreads
        }
      });
      return  productListsale
    },
    productList() { return this.$store.state.productList }
  }
</script>
```

Also change the name of the array you use in the `template section` of the `product_compo_1.vue` component from `productListsale` to `productListsale`, as follows:
If everything is ok, your page should look like this

6. Now let's move the new computed property to the store to create a getter, which makes it available to be used by any other component. To do that, move the `productListsale()` from the `product_compo_1.vue` to the store and place it within the `getters: {}` section and modify the code as shown in the picture:
Now you can use it within the computed properties section of the `product_compo_1.vue` or any other component (shown in the following picture), and you should have the same results.

7. Concerning **mutations**, add two buttons that trigger two functions in the template section of the `Products.vue`, as follows:

```html
<template>
  ...
  <product_compo_1 :productList="productList"></product_compo_1>

  <button v-on:click="IncreasePrice"> Increase price </button>
  <button v-on:click="DecreasePrice"> Decrease price </button>

  <product_compo_2 :productList="productList"></product_compo_2>

  ...
</template>
```

8. Now, add the definitions of two functions to the script section of the `Products.vue`. Then, test if they work correctly.

```javascript
<script>
  ...

  export default {
    methods: {
      IncreasePrice: function() {
        this.$store.state.productList.forEach(product => {
          product.price -= 1;
        });
      },
      DecreasePrice() {
        this.$store.state.productList.forEach(product => {
          product.price += 1;
        });
      }
    }
  }
</script>
```
9. Check the Vuex tab while clicking on the button, do you notice anything?

10. Now, modify the functions in the template section of the Products.vue to commit a mutation in the store, as follows:

```html
<script>
...
export default {
  methods: {
    IncreasePrice: function() {
      this.$store.commit("IncreasePrice")
    },
    DecreasePrice: function() {
      this.$store.commit("DecreasePrice")
    }
  }
}
</script>
```

Then, add the definitions of the created mutations to the mutation section in the store, as follows:

```javascript
...
export default new Vuex.Store({
  state: {},
  mutations: {
    IncreasePrice: state => {
      state.productList.forEach(product => {
        product.price += 1;
      });
    }
  }
});
```
12. Check the Vuex tab again while clicking on the button, do you notice anything?

13. Concerning actions, modify the functions in the template section of the Products.vue to dispatch related mutations in the store, as follows:

```javascript
<script>
...
export default {
...
methods: {
  IncreasePrice: function() {
    this.$store.dispatch("IncreasePriceAct")
  },
  DecreasePrice() {
    this.$store.dispatch("DecreasePriceAct")
  }
}}
</script>
```

14. Add the definitions of the created actions to the actions section in the store, as follows:
export default new Vuex.Store({
...

    actions: {
        IncreasePriceAct: act => {
            setTimeout(function() {
                act.commit("IncreasePrice")
            }, 1000)
        },
        DecreasePriceAct: act => {
            setTimeout(function() {
                act.commit("DecreasePrice")
            }, 1000)
        }
    },
    modules: {},
});

15. Test if and how it works