PREGRADO

UNIDAD 4 | TRABAJO EN EQUIPOS MULTIDISCIPLINARIOS & WEB APPLICATION FEATURES

BEST PRACTICES

PART 2

SI653 | Aplicaciones Web

Al finalizar la unidad de aprendizaje, el estudiante comunica resultados y proceso de ingeniería aplicado para el desarrollo de una solución distribuida de aplicaciones web, con una arquitectura orientada a servicios, que satisface necesidades u oportunidades detectadas, en un ambiente ágil y colaborativo, bajo un enfoque dirigido por la innovación e inclusion.

AGENDA

VUE TIPS

BEYOND VUE

Composables

Vue 3 gives us the power to create reusable chunks of logic that can even include reactive state. These reusable chunks are commonly called "composables".

```
//FetchPostComposable.js
import { ref, computed } from "vue";

export const useFetchPost = () => {

  // Request States
  const REQUEST_IN_PROGRESS = "REQUEST_IN_PROGRESS";
  const REQUEST_ERROR = "REQUEST_ERROR";
  const REQUEST_SUCCESS = "REQUEST_SUCCESS";
  const requestState = ref(null);
  const loading = computed(() => requestState.value ===
REQUEST_IN_PROGRESS);
  const error = computed(() => requestState.value ===
REQUEST_ERROR);

  // Post
  const post = ref(null);
  const fetchPost = async (id) => {
    post.value = null;
    requestState.value = REQUEST_IN_PROGRESS;
    try {
      const res = await
fetch(https://jsonplaceholder.typicode.com/posts/${id});
      post.value = await res.json();
      requestState.value = REQUEST_SUCCESS;
    } catch (error) {
      requestState.value = REQUEST_ERROR;
    }
  };
  return { post, loading, error, fetchPost };
};
```

```
// PostComponent.vue
<script>
import { useFetchPost } from "./FetchPostComposable";
export default {
  setup() {
    //clear source of data/methods
    const {
      loading: loadingPost, //can rename
      error, fetchPost, post } = useFetchPost();

    return { loadingPost, error, fetchPost, post };
  },
};
</script>
```

Object cloning

If you're creating a component that takes in an object as a property, then make sure to clone the object to a local data property before mutating it.

```
// LoginForm.vue
<template>
 ...
<input type="text" v-model="form.name">
<input type="password" v-model="form.password">
</template>
<script>
export default {
  //...
  data(){
    return {
      // spreading an object effectively clones it's top level properties
      // for objects with nested objects you'll need a more thorough solution
      form: {...this.user}
    }
  },
}
</script>
```

Namespaced State Management Modules

Vuex gives you the ability to break your store up into different modules each of which handles it's own domain (ie. one module handles posts, another users, another comments, etc).

```
// store/index.js
import posts from './modules/posts.js'
import users from './modules/users.js'
import comments from './modules/comments.js'
export default{
  modules:{ posts, users, comments }
}


//store/modules/posts.js
// and the same for each of the other domains with
// common state, actions named the same (possible because of namespacing)
// and any other domain specific state and logic named as fits
export default {
  namespaced: true,
  state:{
    items:[]
  },
  actions:{
    fetchOne(){},
    fetchAll(){}
  }
}
```

Interact with REST API's Via SDK's

SDK as a language specific API to interact with your actual API. So instead of calling axios or fetch directly within your components, you can use something like post.find(1).

```
axios('https://someapi.com/posts/1')
```

It's better an autocompletable method on a resource class.

```
postService.find(1)
```

Wrap Third Party Libraries

Create a wrapper around third party libraries. For instance, instead of using axios directly in your codebase you could create a more generically named class like Http whose methods call axios under the hood.

```
// Http.js
import axios from 'axios'
export default class Http{
  async get(url){
    const response = await axios.get(url);
    return response.data;
  }
  // ...
}
```

It allow changing dependencies without changing interface

```
// Http.js
export default class Http{
  async get(url){
    const response = await fetch(url);
    return await response.json();
  }
  // ...
}
```

Avoid v-if with v-for

Instead of this

```
<ul> <li v-for="list in lists" v-if="showActiveList" :key="list.id" > {{ list.category }} </li> </ul>
```

Write this

```
<ul> <li v-for="list in lists" :key="list.id" > {{ list.category }} </li> </ul>
```

Use the Computed Property for Filtering Data

It is ideal to use the computed property whenever you want to filter or transform your data, typically you will use a computed value for that purpose.

```
<script>
export default {
  name: 'WebMaterials',
 data: {
            language: '',
            materials: [
               { title: "Eloquent Javascript", author: 'Accessories' language: 'JS'},
               { title: "You don't know JS", author: 'Kyle Simpson' language: 'JS' },
               { title: "CSS Optimization Basics", author: 'Jens Oliver' language: 'CSS'},
               { title: "The HTML Handbook", author: 'Flavio Copes' language: 'HTML'},

],
computed: {
            filterMaterialsByLanguage: function(){
                return this.materials.filter(product =>
!material.language.indexOf(this.language))
            }
        }
    });
}
</script>
```

Always use the :key attributes with v-for

It is very necessary to use the :key attribute with v-for because the virtual DOM in VueJs creates vnodes on each of the listed items so if the :key attribute is not included in the list item when using v-for.

```
<div v-for="item in items" :key="item.id">
 {{ item.value }}
</div>
```

Use Kebab Case to Name and Event

When working, emitting, or listening to events in Vue.js it is a good practice to use a kebab case for event names, because of “HTML attribute case-insensitivity”

```
<template>
  <button @click="$emit('click-event')" />


</template>
<script>
export default {
  methods: {
    onClick () {
      this.$emit('click-event')


      /
    }
  }
}
</script>
```

Component data must be a function

Vue.js recommends that each instance of a component should have its own data object. Because Vue components are static and if we don't do that, all instances will be sharing the same object, and every time we change something, it will reflect in all other instances.

```
export default {
 data () {
   return {
     firstName: 'Ezekiel' ,
     lastName: 'Lawson'
   }
 }
}
```

Use CamelCase for Prop Name Casing

When naming your props in Vue.js you must use a camel case during declaration and a kebab case when working with templates..

```
<Helloworld send-message="Welcome to your Vue JS Application"/>
```

```
props: {
  sendMessage: String
}
```

Scoping Component Style

When working with styles in your application, the styles in your App component should be global and the rest styles in other components should be scoped. The reason is if you apply changes to a particular class in a different component and other components have that same class name, the changes will reflect globally. So, the best solution to this problem is to scope your style.

```
<template>
  <h1 class="heading-text">WELCOME TO MY BLOG</h1>
  <input type="text" class="search-btn" placeholder="Search items">
</template>
```

```
 <!-- scoped attribute example -->
<style scoped>
   .button-close { background-color: red; border: none; border-radius: 4px solid gray; }
</style>
```

Keep npm Packages updated

Try to regularly update npm packages to avoid any dependency errors and to use the latest/updated features provided by individual packages.

```
npm outdated // run to check outdated npm packages
npx npm-check-updates -u // run to check updates outdated npm packages
npm install // run to update npm packages
```

Use suitable Data Type assign for variable

Continuously use proper data types instead of “any” to minimize the casting time and other casting errors.

Avoid any casting inside the loops.

In a case of two data types assigned to the same property, implement type casting using both the types and using conditions.

```
// Wrong
const age: any = 0; [hard to identify response type at the places this variable
used and also hard to track errors on assignments]

// Right
const age: number = 0; [easy to track and identify response type]
```

AGENDA

VUE TIPS

BEYOND VUE

Beyond Vue

Pinia

https://pinia.vuejs.org/

Nuxt

https://v3.nuxtjs.org/

Modular View Applications

Abstracting your dependencies

https://vueschool.io/articles/vuejs-tutorials/abstracting-your-dependencies/

Domain-Driven Design in Vue.js

https://vueschool.io/articles/vuejs-tutorials/domain-driven-design-in-vue-js/

Domain-Driven Design in Nuxt Apps

https://vueschool.io/articles/vuejs-tutorials/domain-driven-design-in-nuxt-apps/

RESUMEN

Recordemos

Vue Tips

Beyond Vue

REFERENCIAS

Para profundizar

https://vueschool.io/articles

https://www.vuemastery.com/blog

PREGRADO

Ingeniería de Software
Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova

LAUREATE
INTERNATIONAL
UNIVERSITIES