Vue 3: la chuleta 🥩

ESCUELAVUE

Instalación e uso

La manera mas sencilla de comenzar a utilizar Vue 3 es a través de Vue CLI. Asegúrate de tenerlo actualizado.

```
npm install -g @vue/cli
# O
npm update -g @vue/cli
```

Luego, solo tienes que crear un proyecto con él y **seleccionar Vue 3 en las opciones** (también instalará las nuevas versiones de Vue Router, Vuex y demás).

```
vue create mi-super-proyecto
```

Fragmentos

Por fin 🔥 podemos utilizar varios elementos del DOM como raíz de componentes gracias a los Fragmentos de Vue 3.

```
<!-- Mi Componente -->
<template>
  <div>...</div>
  <div> 🦉</div>
</template>
```

Teleport

Con los teleport de Vue 3 puedes definir la lógica en un sitio pero presentarlo en otro lugar del DOM, aunque físicamente ambas partes estén en el mismo documento.

```
<!-- Mi Modal -->
<teleport to="body">
  <div class="modal">
    ...
  </div>
</teleport>
```

Suspense

Con suspense de Vue 3 podemos mostrar diferentes contenidos en un componente (fallback), una vez todas las promesas (async) involucradas se hayan resuelto.

```
<!-- Mi Componente -->
<Suspense>
  <template #default>
    <HelloWorld />
  </template>
  <template #fallback>
    <p>Loading...</p>
  </template>
</Suspense>
```

Nuevas características

Vue 3 está cargado de novedades que querrás utilizar desde el primer momento. Aquí tienes algunos de los más importantes.

Varios *v-model*

Vue 3 permite usar argumentos en la directiva v-model, facilitando el uso de múltiples v-model en componentes, cada uno asociado una propiedad reactiva.

```
<PersonalForm
  v-model:name="name"
  v-model:age="age"
  v-model:color="color"
/>
```

Cada modelo se sincronizará con una propiedad diferente en el componente.

```
import { ref } from 'vue'
...
setup() {
  const name = ref("John Doe");
  const age = ref(33);
  const color = ref("#FF0000");
  return {
    name, age, color
  }
}
```

Nuevos lifecycle hooks

Vue 3 trae nuevos nuevos hooks en el ciclo de vida de la instancia Vue, especialmente pensados para trabajar con la Composition API.

```
import { onUpdated } from 'vue'
const MiComponente = {
  setup() {
    onUpdated(() => {
      console.log('Updated!')
    })
  }
}
```

Provide & Inject

Para evitar escenarios donde las propiedades se pasan entre diferentes generaciones de componentes, Vue 3 permite *proveer* datos que pueden ser recuperados por los hijos.

```
// Padre
export default {
  provide: {
    location: 'North Pole',
    geolocation: {
      longitude: 90,
      latitude: 135
    }
  }
}
```

```
// Nieto
export default {
  inject:
    [
      'location',
      'geolocation'
    ]
}
```

Vue 3: la chuleta 🥩

ESCUELAVUE

Composition API

La nueva Composition API (compatible con la actual Options API) ha llegado para ayudarnos en la **organización de componentes extensos y/o que necesiten ser estructurados con base en sus características** y funcionalidades.

```
<template>
  <h1>Tienes {{ tareasPendientes }} tareas pendientes 😔</h1>
  <input type="text" v-model="nuevaTarea" required />
  <input type="button" @click="agregarTarea" />
  <hr />
  <ul>
    <li v-for="(tarea, $index) in tareas" :key="$index">
      {{ tarea.titulo }}
    </li>
  </ul>
</template>

<script>
import { ref, computed } from 'vue'
export default {
  setup() {
    const nuevaTarea = ref('')
    const tareas = ref([
      { titulo: 'Aprender Vue', finalizada: true },
      { titulo: 'Aprender Vuex', finalizada: true },
      { titulo: 'Aprender Firebase', finalizada: false }
    ])
    const tareasPendientes = computed(() => {
      return tareas.value.filter((tarea) => !tarea.finalizada)
    })
    function agregarTarea() {
      tareas.value.push({ titulo: nuevaTarea.value, finalizada: false })
      nuevaTarea.value = ''
    }
    return { nuevaTarea, tareas, tareasPendientes, agregarTarea }
  }
}
</script>
```

Importamos los objetos y métodos de la API de Composición

Las referencias reactivas envuelven valores **primitivos** en objetos que reaccionan ante cambios

A través de **computed** creamos propiedades computadas

Exportamos las funciones que hacen de métodos en el componente

Para acceder al valor primitivo debemos utilizar la propiedad **value**

Hacemos accesibles las propiedades y métodos **exponiéndolos** al exportarlos

El método setup()

El método setup() de Vue 3 es llamado antes de la inicialización del componente (*created* hook) y **no tiene acceso a *this***.

```
export default {
  setup() {
   ...
   return { ... }
  }
}
```

Propiedades

Podemos acceder a las propiedades de un componente como **primer parámetro** de *setup()*.

```
export default {
  props: ['cantidad'],
  setup(props) {
    console.log(props.cantidad)
  }
}
```

¿Y el resto?

Podemos acceder a lo antes accedíamos con *this* a través del objeto *context*.

```
export default {
  setup(props, context) {
    context.attrs
    context.slots
    context.emit
  }
}
```