VUE

Vue és un framework JavaScript de codi obert pensat per a la creació de frontend i inteficies d'usuari.

La versió actual, i en la qual es basa aquest curs, és la versió 3. Als següents apartats es mostraran les bases que podeu ampliar amb la documentació oficial.

Composition

Vue suporta dos "estils" d'API: options i composition. Al llarg d'aquest curs ens centrarem en la composition API. Els dos estils són semblants, però divergeixen en la forma de manejar el codi i les variables JS.

Un fitxer `.vue` tendrà tres parts `template`, `script` i `style` encara que prescindirem d'aquest darrer en la majoria de casos.

```
<template>
  <!-- Aqui definirem la plantilla amb html i variables vue -->
</template>

<script setup lang="ts">
  // Dins script hi haurà tota la lògica, definició de variables...
</script>

<style scoped>
  /* Si fes falta disposam d'un apartat styles per a definir estils del
component. */
</style>
```

Tots els fitxers '.vue', ja sigui components o vistes, tenen aquesta estructura, la diferència entre vista i components és lògica.

Template

La sintaxis del template serà la mateixa que html de tota la vida amb l'excepció de certs elements vue per a intercalar les nostres variables.

En el cas de voler pintar un valor dintre d'una etiqueta usarem els caràcters `{{` i `}}` per a delimitar el codi js

```
<span>Nom de la persona: {{ persona.nom }}</span>
```

En el cas de propietat d'una etiqueta html precedirem de `:` l'etiqueta per a indicar que el valor és una propietat js.

```
<button :disabled="formulariOk">Envia</button>
```

Script

Dins l'apartat script es definiran els imports, propietats, mètodes... que usarem al template. Al usar el composition API totes les propietats definides seran automàticament accesibles al template, cosa que no passa amb el options API

```
import { Persona } from "@/model/Persona";

const persona: Persona = new Persona("Joan");

const formulariOk: boolean = false;
```

El codi anterior és un exemple no funcional a falta de veure la reactivitat

Directives

Les directives són atributs especials amb el prefixe `v-` que doten a les etiquetes html d'interoperabilitat amb el codi JS.

Vue disposa d'un conjunt de directives que són suficients per a la majoria de casos, però es poden crear noves directives o usar les que venen en llibreries externes.

A continucació veurem les més representatives.

v-bind

Aquesta directiva, que ja hem vist en la forma abreujada, permet enllaçar una propietat d'una etiqueta html a una propietat JS.

```
<a v-bind:href="url"></a>

<!-- forma abreujada -->
<a :href="url"></a>
```

v-on

De forma semblant a l'anterior aquesta directiva ens permet enllaçar una etiqueta o element html a un event JS executant un mètode definit

```
<button v-on:click="metodeButtonClic">...</button>

<!-- forma abreujada -->
<button @click="metodeButtonClic">...</button>
```

v-if

Aquesta directiva, com el seu nom indica, ens permet renderitzar de forma opcional elements HTML. Es pot combinar, de forma opcional, amb `v-else-if` i `v-else`. En aquests casos els elements han d'anar precedits un darrere l'altre.

```
<div v-if="persona.especialitat === 'T'">Tècnic</div>
<div v-else-if="persona.expecialitat === 'A'">Administratiu</div>
<div v-else>Sense especialitat</div>
```

v-for

Com el seu nom indica ens permet recórrer un llistat per a renderitzar el contingut per a cada element.

```
<div v-for="observacio in observacions">
  {{ observacio.data }}: {{ observacio.text }}
</div>

<!-- Alternativament es pot obtenir també l'índex -->
<div v-for="(titol, index) in titols">{{ index }}): {{titol}}</div>
```

> **NOTA:** No està recomanant l'ús de `v-for` i `v-if` al mateix element HTML degut a que el comportament pot resultar no intuitiu. EL següent codi no funcionaria.

```
<div v-for="nota of notes" v-if="nota > 5"></div>
```

`<template>`

Les directives es poden aplicar a elements HTML, això implica que generen una nova etiqueta. VUE incorpora l'etiqueta `<template>` que no renderitza cap element, simplement aplica la directiva en qüestió. Això ens permet solventar la problemàtica anterior.

```
<div v-for="nota of notes">
  <template v-if="nota > 5"> {{ nota }} </template>
</div>
```

Navegació (Routes)

La navegació a través de les aplicacions VUE consta de dues parts, la primera es la definició de les rutes i la segona l'assignació a un boto o element per poder navegar cap allà.

Definició de les rutes

Per a definir una ruta hem d'emplenar un objecte tipus `RouteRecordRaw` que te la seguent forma:

```
  {
    path: 'expedients', // path de la ruta al navegador ex:
el.nostre.domini/expedients
    component: ExpedientList, // Component al que fa referencia
    name: 'ExpedientList', // Nom que assignam a la ruta
    children: [] // En cas de tenir fills, les rutes serien relatives al
pare ex: el.nostre.domini/expedients/<path.fill>
  }
```

Es important que el path no contengui una `/` inicial ja que resetejariem la ruta (a no ser que aquesta sigui la intenció).

Al path es poden especificar variables, això es fa precedint el nom de la variable per `:`, per exemple `path: 'expedients/:id'` defineix una propietat `id` poguent agafar qualsevol valor. Per tant les rutes `expedient/1`, `expedient/2348` o `expedient/nous` serien valides per accedir a aquesta vista.

Amb l'exemple anterior pot ser hem notat que les rutes `expedients/:id` i `expedient/nous` poden entrar en conflicte ja que les dues poden resoldre una mateixa adreça. Hi ha diverses formes de tractar aquests casos, però la més senzilla és la classica de l'ordre.

La propietat component es una constant que fa referencia a la nostra vista. Per a que es carregui de forma dinàmica ho hem de fer de la forma següent:

```
const ExpedientList = () => import('../views/ExpedientList.vue')
```

Com es va veure al tema d'estructura cada mòdul té les seves rutes i aquestes s'importen dins `/src/config/routes.ts`.

Navegació

Hi ha distintes formes d'especificar la navegació a una ruta en concret. Es pot fer de la forma tradicional, per el path (`/expedients`), pero seria més recomanable usar la navegació per nom. El nom es correspon a la propietat `name` de la ruta definida.

Els objectes que permeten la navegació solen tenir una propietat `:to` que es la que hem d'emplenar per indicar el canvi de vista.

```
<button :to="{ name: 'ExpedientList'} />

// En cas de requirir parametres
<button :to="{ name: 'ExpedientView', params: { id: expedient.id } } />
```

Si volem navegar des de el codi js hem d'usar el composable `useRouter` amb el mateix format.

```

```

```
const router = useRouter()
router.push({ name: 'ExpedientList' })
```

En cas d'una ruta amb variables probablement necessitarm accedir a elles, en aquest cas hem d'usar el composable `useRoute` (no confondre amb l'anterior)

```
const route = useRoute()
const { id } = route.params
```

Menú

El menú principal és una incorporació a l'esquelet i es pot configurar des de el fitxer `src/config/menu.ts` emplenant objectes tipus `ImasNavbarItemHeader` que tenen la següent forma:

```
{
    title: tc('ofi.oficina'), // Nom que es mostrarà
    icon: 'fas fa-building', // Icona de l'item de menú
    to: { name: 'ExpedientList' }, // Ruta a la que accedirà
    active: () => {...}, // Mètode que s'executa per determinar si es
mostra o no l'item. (per exemple per verificar el rol)
    children: [] // Subnivell del mateixos elements
}
```

Fer notar que les propietats `to` i `children` definides al mateix element causen un malfuncionament del menú.

Lifecycle Hooks

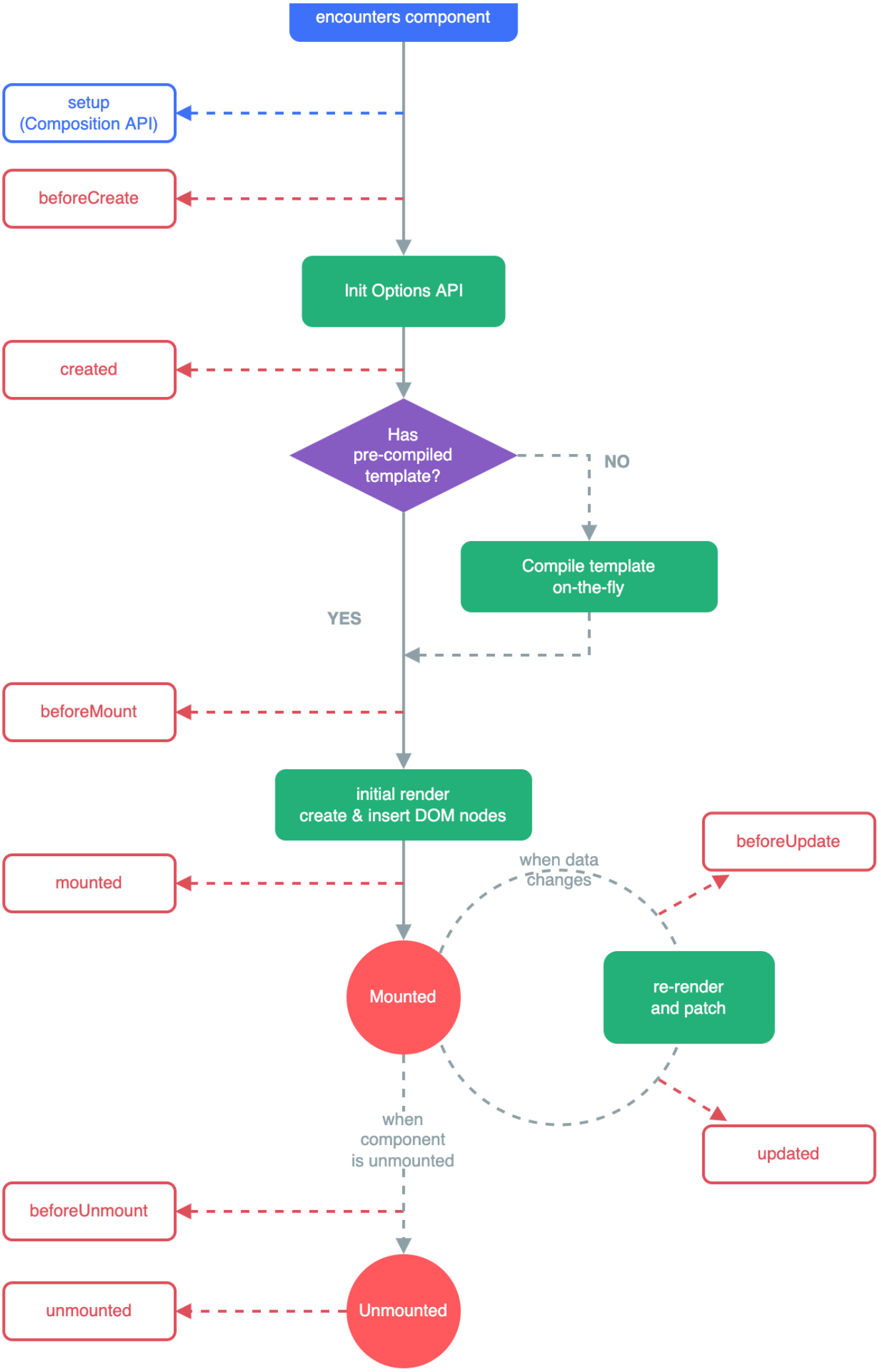
La part de l'script s'executa al carregar el component en questió. Això pot suposar un problema en certs casos que volem executar un codi en un moment concret, per exemple, al carregar o tancar el component.

Per aquest motiu VUE disposa de varis hooks que s'executaran en moments concrets del cicle de vida del component.

Per a usar un hook s'ha de definir aquest amb un mètode que s'executarà arribat el moment. El més usat és onMounted que s'executa una vegada el component està carregat.

```
onMounted(() => {
  // Codi a executar
});
```

A la següent imatge es pot veure el cicle complet.


encounters component
setup
(Composition API)
beforeCreate
Init Options API
created
Has
pre-compiled
template?
NO
Compile template
on-the-fly
YES
beforeMount
initial render
create & insert DOM nodes
mounted
beforeUpdate
when data
changes
Mounted
re-render
and patch
updated
when
component
is unmounted
beforeUnmount
unmounted
Unmounted