PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 8.2: Vue Components

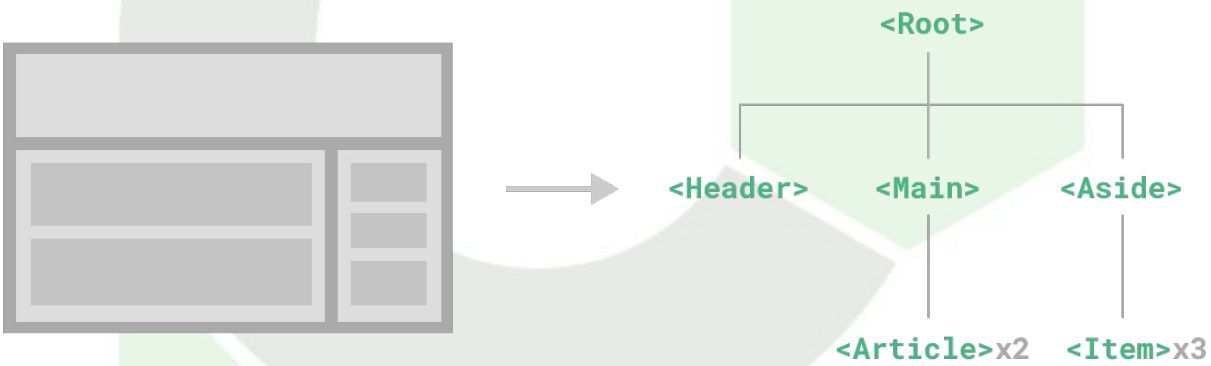
Joan Pons Tugores

Índex de continguts

Components

Fins ara hem utilitzat una única instància de Vue per fer les nostres aplicacions, però això no té per que ser així. Podem utilitzar més instàncies, cada una amb el seu *template* i els seus estils per dividir l'aplicació en components i facilitar la seva reusabilitat així com el manteniment del codi.

Podrem cridar aquests components des de la instància *Vue* arrel i també des d'altres components. Els components de Vue formen un arbre de components, com els elements html, però cada un té el seu propi comportament i contingut.



Declaració

Un component també tendrà les parts de *script, template* i *style*. Normalment es crearan dins un arxiu *vue* a la carpeta *components*.

```
<script setup>
import {ref} from "vue";

const comptador = ref(0)

function incrementar() {
  comptador.value++;
}
</script>

<template>
  <button @click="incrementar">M'has pitjat {{ comptador }}
          vegades</button>
</template>

<style scoped>
  button {
          margin: 2em;
  }
</style>
```

Els components són instàncies reutilitzables de Vue amb un nom, en aquest cas *ButtonCounter*, que podem utilitzar a la instància arrel de Vue de l'aplicació o dins d'altres components, tantes vegades com faci falta.

En Vue3 els noms dels components han de ser multi-paraules per evitar coincidir amb elements HTML. Els podem posar de dues maneres:

- en PascalCase, com a l'exemple: ButtonCounter
- amb guió separador, button-couter

Cada instància que utilitzem del component, cada element que afegim a l'html amb el nom d'aquest component, tendrà les seves pròpies dades i els mètodes treballaran amb aquestes dades.

Utilització

Per utilitzar un component dins un altre, o dins App, s'han de seguir les següents passes:

Script

A l'*script* importam el component amb

```
<script setup>
  import ButtonCounter from "@/components/ButtonCounter.vue";
</script>
```

@ indica l'arrel del projecte i al darrera hi va la ruta fins el fitxer vue que conté el component definit.

Els components importats a l'script setup es poden incorporar directament a la part de *template*.

Template

Finalment utiitzarem el component dins el template com si fos un element HTML més, una etiqueta amb el nom del component que hem registrat a l'script.

```
<ButtonCounter/>
```

Evidentment, el podem inserir tantes vegades com ens faci falta.

```
<h2>Botons</h2>
<p>    <ButtonCounter/>    </p>
<p>    <ButtonCounter/>    </p>
```

Cada un d'aquests botons portarà el compte de les vegades que l'usuari ha clicat en aquell botó.

El codi complet de *App.vue* és aquest:

```
<script setup>
        import ButtonCounter from "@/components/ButtonCounter.vue";
</script>

<template>
 <div>
  <img alt="Vue logo" class="logo" src="./assets/logo.svg"
              width="125" height="125"/>
 </div>
 <div>
  <ButtonCounter/>
 </div>
</template>

<style scoped>
 div {
        display: block;
        clear: both;
 }
</style>
```

Reutilització de components

Una vegada tenim el component definit el podem reutilitzar tantes vegades com volguem.

```
<h2>Botons</h2>
<p>    <ButtonCounter/>    </p>
<p>    <ButtonCounter/>    </p>
```

Per a cada un d'ells es crearà una instància del component, evidentment amb les seves pròpies dades. A l'exemple anterior, cada botó mantindrà el seu comptador independent dels altres.

Passant dades als components

Si tenim més d'una instància d'un component segurament el voldrem "personalitzar" de manera que cada instància mostri informació diferent o es comporti de manera distinta.

Les **propietats** són atributs personalitzats que es poden registrar al component, de manera que li podem passar informació. Per exemple, si volem que cada botó del nostre exemple mostri un nom diferent podem modificar el component de la següent manera:

Dins la l'<*script setup*> utilitzam la funció *defineProps*. L'argument és un array de cadenes de text amb el nom de les propietats que volguem que tengui el component.

```
<script setup>
  import {ref} from "vue";

  const comptador = ref(0)

  function incrementar() {
          comptador.value++;
  }

  defineProps(['nomBoto'])
</script>
```

Al *template* podem utilitzar aquestes propietats com si fossin part de l'entorn de reactivitat, per exemple amb {{ }}.

```
 <button v-on:click="incrementar">
   {{ nomBoto }} ({{ comptador }} vegades)
 </button>
```

Finalment, quan cream una instància del component, per exemple a *App.vue*, afegim a l'element un atribut amb el nom de la propietat:

```
<p>     <ButtonCounterPersonalitzat nomBoto="Primer"/>  </p>
<p>     <ButtonCounterPersonalitzat nom-boto="Segon"/> </p>
```

D'aquesta maner a la pàgina tendrem un botó amb el nom *Primer* i un altre amb el nom *Segon*. Per cert, les dues notacions del nom de la propietat funcionen.

Si volem passar un dels elements del component pare, per exemple *cadena*, haurem d'utilitzar v-bind:

```
<p><ButtonCounterPersonalitzat :nom-boto="cadena"/></p>
```

Evidentment les propietats no tenen perquè ser un valor simple, poden contenir objectes.

Les propietats són immutables, és a dir, dins del component no podem canviar els seus valors.

No obstant, les propietats són reactives, és a dir, si canvia el valor al component pare el component fill canvia.

A l'exemple anterior, si el component pare canvia el valor de *cadena* el component *ButtonCounterPersonalitzat* es refrescarà i mostrarà el nou valor de la propietat *nom-boto*.

Les propietats també es poden definir com un objecte, especificant el seu tipus:

```
<script setup>
        defineProps({
                nomBoto: String
        })
</script>
```

Fer-ho d'aquesta manera té l'avantatge que a la consola apareix un *warning* a la consola del navegador quan passam un valor d'un tipus equivocat.

Si un component té diverses propietats podem utilitzar un objecte amb v-bind. La única condició és que les propietats de l'objecte han de tenir el mateix nom que les propietats del component.

```
const dadesBoto = {
        id: 1,
        nom: 'Botonot'
}
```

Si cream un component botó

```
<Boto v-bind:"dadesBoto"/>
```

serà equivalent a

```
<Boto :id="dadesBoto.id" :nom="dadesBoto.nom"/>
```

Components en iteracions

És molt probable que haguem de crear un component per a cada element d'una llista. Per exemple si tenim definit un array amb els noms dels botons:

```
data: function () {
  return {
    noms:["Primer","Segon","Tercer","Quart"]
  };
},
```

Podem crear els botons utilitzant *v-for*

```
<p>
  <ButtonCounterPersonalitzat
          v-for="nom in noms"
          :key="nom"
          :nomBoto="nom"/>
</p>
```

En aquest cas hem de posar forçosament l'atribut *key* que identifiqui cada component. L'enllaçam a la dada que volguem utilitzar d'identificador amb *v-bind.*

Per enllaçar la propietat amb el seu valor també utilitzam *v-bind*:

```
  :nomBoto="nom"
```