

Components i Vue Cli



Inmaculada Rodríguez Santiago

Jordi Barea Colomer

Factors Humans i Computació, 22-23

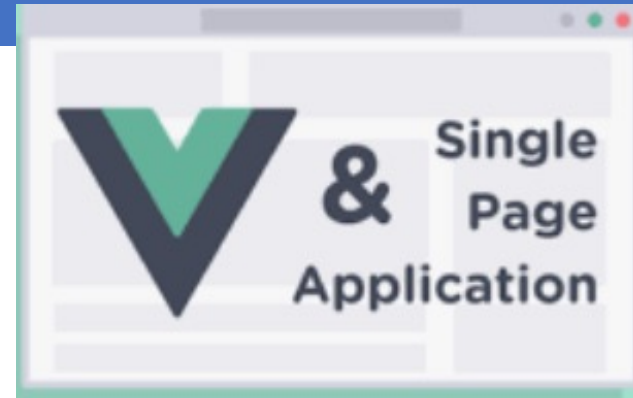
Formas de usar Vue



Aplicació multi-pàgina: Vue controla parts de les pàgines o les pàgines enteres.

COM HEM TREBALLAT FINS ARA

Recordeu! `app.mount("#app")`



Aplicació SPA (Single Page Application):

El servidor envia solament una pàgina HTML que no inclou markup, és el codi Vue qui toma el control de la UI.

HO VEUREM AMB COMPONENTS

Components

- Els Components ens permeten dividir la interfície d'usuari en peces independents i reutilitzables.
- Una aplicació es podria veure llavors com un arbre de Components niats.
- Un component és una nova etiqueta HTML amb una interfície i un comportament. Ex. `<friend-contact></friend-contact>`

Creació de components

Creació de components:

```
app.component ( 'nom-component' , { }
```

Amb nom compost per dues paraules, amb – com a separador

Amb template, data and methods

Creació de components

Creació de components :

```
app.component( 'nom-component', {  
  template: `...codi html...`  
  data: function(){  
    ...  
  },  
  methods: {  
    ...  
  }  
});
```

app.js

```
1  const app = Vue.createApp({
2    data() {
3      return {
4        friends: [
5          {
6            id: 'manuel',
7            name: 'Manuel Lorenz',
8            phone: '01234 5678 991',
9            email: 'manuel@localhost.com',
10          },
11          {
12            id: 'julie',
13            name: 'Julie Jones',
14            phone: '09876 543 221',
15            email: 'julie@localhost.com',
16          },
17        ],
18      };
19    },
20  });
```

```
22  app.component('friend-contact', {
23    template: `
24      <li>
25        <h2>{{ friend.name }}</h2>
26        <button @click="toggleDetails()">
27          {{ detailsAreVisible ? 'Hide' : 'Show' }} Details
28        </button>
29        <ul v-if="detailsAreVisible">
30          <li><strong>Phone:</strong> {{ friend.phone }}</li>
31          <li><strong>Email:</strong> {{ friend.email }}</li>
32        </ul>
33      </li>
34    `,
35    data() {
36      return {
37        detailsAreVisible: false,
38        friend: {
39          id: 'manuel', name: 'Manuel Lorenzo',
40          phone: '01234 5678 991', email: 'manuel@localhost.com',
41        },
42      };
43    },
44    methods: {
45      toggleDetails() {
46        this.detailsAreVisible = !this.detailsAreVisible;
47      },
48    },
49  });
50  app.mount('#app');
```

Exemple de creació de
component friend-
contact

Creació de components

Fitxer amb extensió .vue

Altres maneres de crear components és amb un SFC (single-file component)

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  }
}
</script>

<template>
  <button @click="count++">You clicked me {{ count }} times.</button>
</template>
```

Creació de components

Un altre manera de crear-los és com un objecte JS

Fitxer .js que defineix un únic component i l'exporta.

```
export default {  
  data() {  
    return {  
      count: 0  
    }  
  },  
  template: `  
    <button @click="count++">  
      You clicked me {{ count }} times.  
    </button>`  
}
```


Ús de components

Al HTML utilitzem el component com si fos un element més:

```
<nom-component> </nom-component>
```

Vue detectarà el component per el seu nom (am un -) i entendrà que volem renderitzar la mini aplicació que s'ha definit en aquest component (amb un *template*, *data* i *lògica*)

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Vue Basics</title>
7     <link
8       href="https://fonts.googleapis.com/css2?family=Jost:wght@400;700&display=swap"
9       rel="stylesheet"
10    />
11    <link rel="stylesheet" href="styles.css" />
12    <script src="https://unpkg.com/vue@next" defer></script>
13    <script src="app.js" defer></script>
14  </head>
15  <body>
16    <header>
17      <h1>FriendList</h1>
18    </header>
19    <friend-contact></friend-contact>
20    <section id="app">
21      <ul>
22        <friend-contact></friend-contact>
23        <friend-contact></friend-contact>
24      </ul>
25    </section>
26  </body>
27 </html>
28

```

FriendList

Manuel Lorenz

Show Details

Manuel Lorenz

Show Details

Exemple d'ús de
component
friend-
contact

Vàries aplicacions Vue vs varis components

- Les aplicacions Vue son independents una d'altre. Podem tenir varies crides a `createApp()` cadascuna amb el seu punt de montatge. No es poden comunicar.

vs

- Els components es poden comunicar i així compartir dades. Podem tenir una UI connectada si tenim una aplicació arrel que manté varis components. (Aquest és el cas de SPA).

Comunicació entre components: props

Props

- Les props son propietats dels components (atributs HTML).
- Permeten utilitzar el mateix component quantes vegades sigui necessari però amb dades diferents (que venen d'altre component o aplicació).

Parent
component

dades

Child
component

Index.html

```
<ul>
  <friend-contact
    name="Manuel Lorenz"
    phone-number="01234 78992"
    email-address="manuel@localhost.com"
  ></friend-contact>
  <friend-contact
    name="JulieJones"
    phone-number="0987 65431"
    email-address="julie@localhost.com"
  ></friend-contact>
</ul>
```

Comunicació entre components: props

- Les props es defineixen al component com un array de strings.
- Al template, utilitzem props tal com ho fem amb les dades.
- Als mètodes, es pot accedir a les propietats (amb this.) tal com ho fem amb les dades.

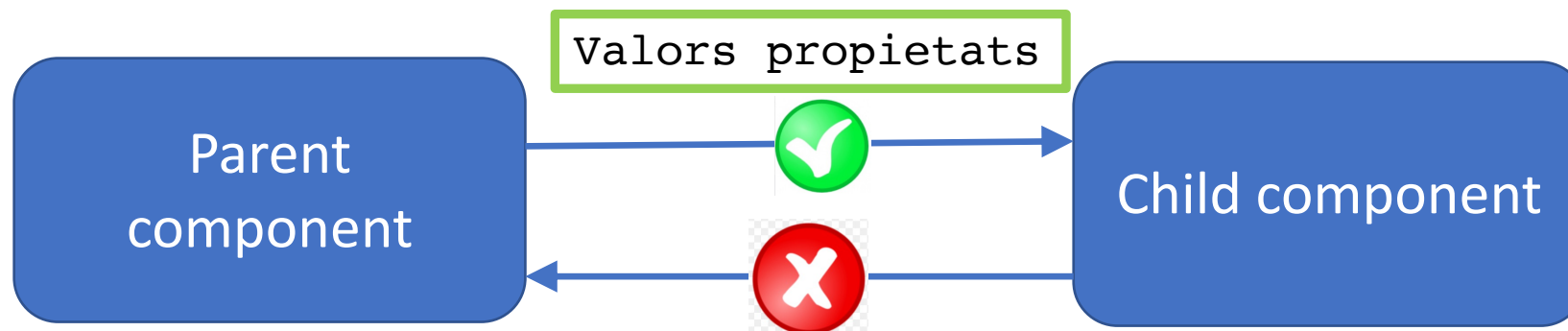
```
22 app.component("friend-contact", {
23   props: ["name", "phoneNumber", "emailAddress"],
24   template: `
25     <li>
26       <h2>{{ name }}</h2>
27       <button @click="toggleDetails()">
28         {{ detailsAreVisible ? 'Hide' : 'Show' }} Details
29       </button>
30       <ul v-if="detailsAreVisible">
31         <li><strong>Phone:</strong> {{ phoneNumber }}</li>
32         <li><strong>Email:</strong> {{ emailAddress }}</li>
33       </ul>
34     </li>
35   `,
36   data() {
37     return {
38       detailsAreVisible: false,
39       friend: {
40         id: "manuel",
41         name: "Manuel Lorenzo",
42         phone: "01234 5678 991",
43         email: "manuel@localhost.com",
44       },
45     };
46   },
47   methods: {
```

Al template, enlloc de **friend.email**, **emailAddress** (idem amb les altres dues props)

Comunicació entre components: props

- Amb les props existeix una comunicació unidireccional. Són immutables.

```
22 app.component("friend-contact", {
23   props: ["name", "phoneNumber", "emailAddress"],
24   template: `
25     <li>
26       <h2>{{ name }}</h2>
27       <button @click="toggleDetails()">
28         {{ detailsAreVisible ? 'Hide' : 'Show' }} Details
29       </button>
30       <ul v-if="detailsAreVisible">
31         <li><strong>Phone:</strong> {{ phoneNumber }}</li>
32         <li><strong>Email:</strong> {{ emailAddress }}</li>
33       </ul>
34     </li>
```



```
this.phoneNumber = '600 20 44 20'
```

```
<ul>
  <friend-contact
    name="Manuel Lorenz"
    phone-number="01234 78
    email-address="manuel@
  ></friend-contact>
```

Comunicació entre components: props

- Per aconseguir una comunicació bidireccional:

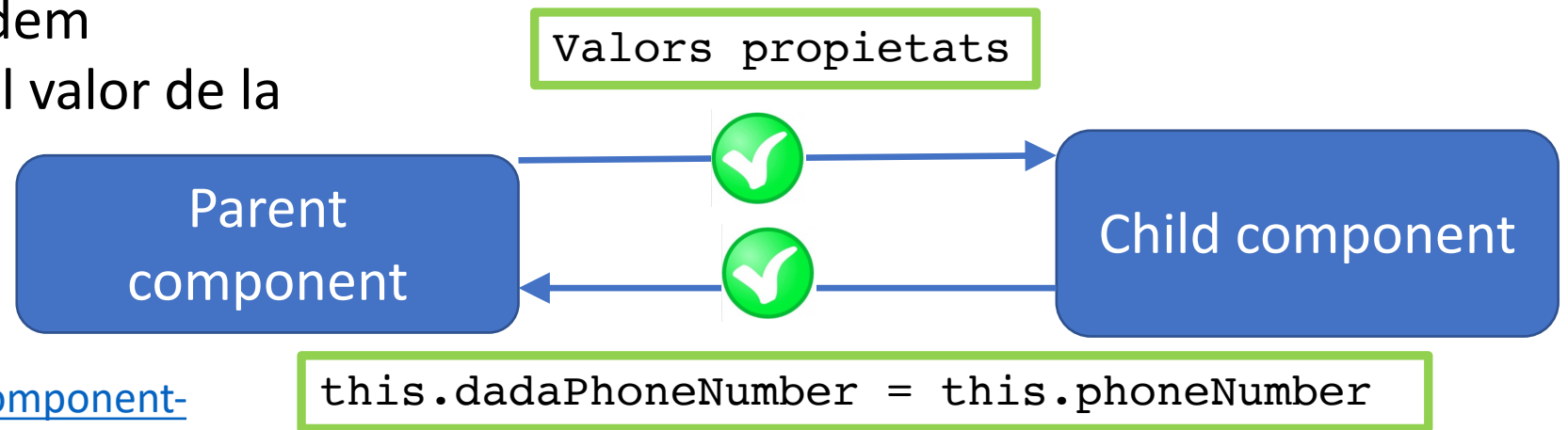
OPCIÓ 1: Creem una dada amb el valor inicial de la propietat (ex. `dadaPhoneNumber`) i ja podem canviar, si es necessita fer, el valor de la dada.

OPCIÓ 2: Usar `$emit`

(veure

<https://vuejs.org/guide/essentials/component-basics.html#listening-to-events>)

```
22 app.component("friend-contact", {
23   props: ["name", "phoneNumber", "emailAddress"],
24   template: `
25     <li>
26       <h2>{{ name }}</h2>
27       <button @click="toggleDetails()">
28         {{ detailsAreVisible ? 'Hide' : 'Show' }} Details
29       </button>
30       <ul v-if="detailsAreVisible">
31         <li><strong>Phone:</strong> {{ phoneNumber }}</li>
32         <li><strong>Email:</strong> {{ emailAddress }}</li>
33       </ul>
34     </li>
```



Components: validació

<https://vuejs.org/guide/components/props.html>

- Les props es poden definir com a objectes i assignar un tipus, una propietat required (true, false) i una lògica de validació (validator)

Si el valor de la propietat isFavourite no és '1' o '0' retorna false i es mostra un missatge d'error per consola

```
// props: ['name', 'phoneNumber', 'emailAddress', 'isFavorite']
props: {
  name: {
    type: String,
    required: true,
  },
  phoneNumber: {
    type: String,
    required: true
  },
  emailAddress: String,
  isFavorite: {
    type: String,
    required: false,
    default: '0',
    validator: function(value) {
      return value === '1' || value === '0';
    }
  },
}
```


Propiedades dinámicas de los componentes

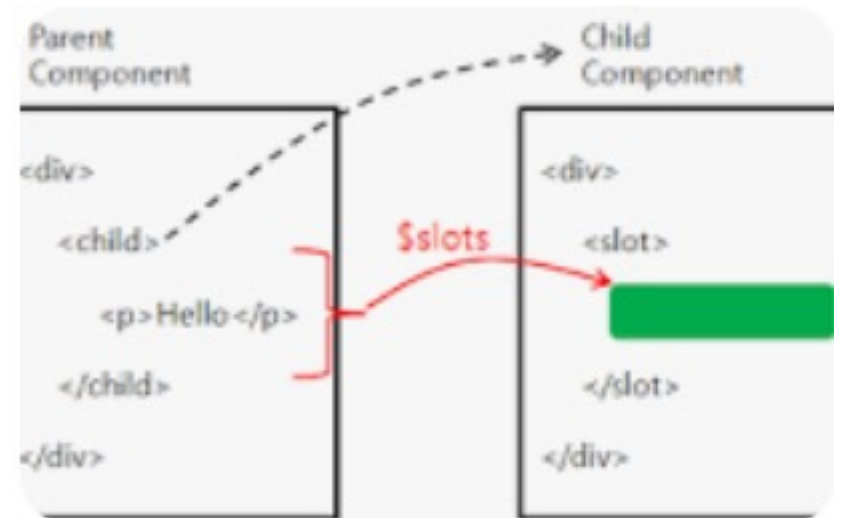
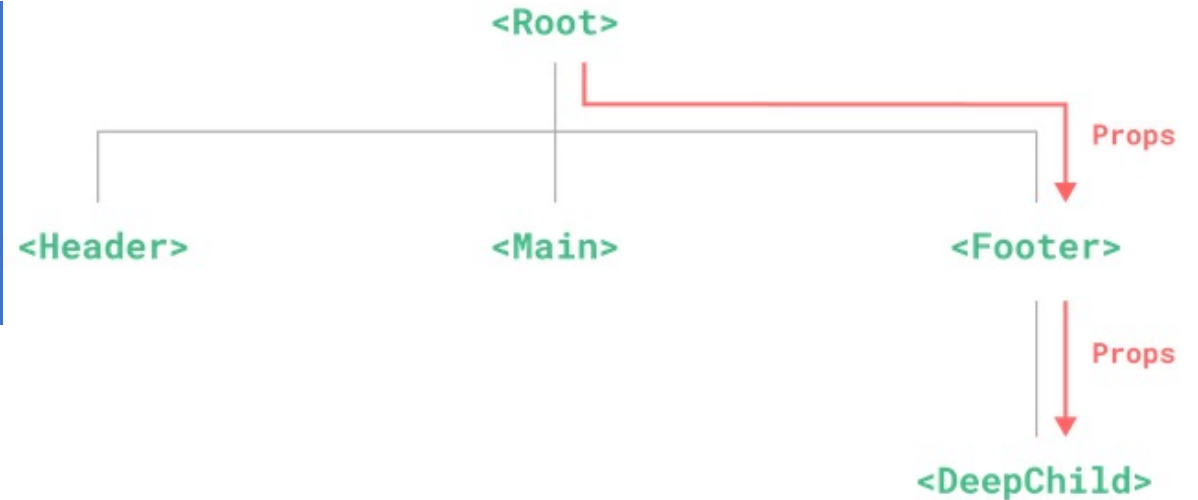
- Podem utilitzar directives v-for o v-bind per a unir propietats d'un component amb dades de la app Vue

- Enlloc de hardcodejar els valors de les propietats de cada amic, fem un v-for.
- Recordeu que :name equival a v-bind:name

```
src > App.vue > {} "App.vue" > template > section >
4      <h1>My friends</h1>
5    </header>
6    <ul>
7      <friend-contact
8        v-for="friend in friends"
9        :key="friend.id"
10       :name="friend.name"
11       :phone-number="friend.phone"
12       :email-address="friend.email"
13       :is-favorite="true"
14     </friend-contact>
15  </ul>
16 </section>
17 </template>
```

Més sobre components

- [Provide/Inject](#): per a comunicar propietats entre components en arbres de components grans.
- [Slots](#): per a passar un fragment de template a un component (de forma similar a com es fa amb les props).
- [Animations](#).



Un millor setup per a desenvolupament

The Vue CLI (Command Line Interface)

i

Vite

The Vue CLI

- Què és?
 - Eina addicional, és un entorn de desenvolupament per a aplicacions Vue, alternativa a utilitzar un editor de text pla i refrescar l'explorador.
- Perquè ho necessitem
 - S'actualitza automàticament la pàgina (sense haver de refrescar)
 - Enlloc d'utilitzar el protocol *file://* (quan fem recarregar) utilitzem *https://* protocol (el protocol amb el qual es serviran les pàgines HTML)
 - Podem treballar més còmodament amb múltiples fitxers.
 - L'editor proporciona millor auto completió.

The Vue CLI

Instal·lació i execució en 5 passos

- PAS 1: Necessita d'un servidor web de desenvolupament: baixar [NodeJS](#) i instal·lar-lo (també s'instal·larà automàticament npm - node package manager).

The Vue CLI

- PAS 2:
install the Vue
Cli

Install:

```
npm install -g @vue/cli  
# OR  
yarn global add @vue/cli
```

Create a project:

```
vue create my-project  
# OR  
vue ui
```

- PAS 3:
3.1 Crear
projecte

3.2 Selecciona “Default”:

```
Vue CLI v4.5.0  
? Please pick a preset: (Use arrow keys)  
> Default ([Vue 2] babel, eslint)  
  Default (Vue 3 Preview) ([Vue 3] babel  
  Manually select features
```

The Vue CLI

🎉 Successfully created project **project**.
👉 Get started with the following commands:

```
$ cd project  
$ npm run serve
```

```
(base) Inmaculadas-MacBook-Pro:~ inma$ cd project  
(base) Inmaculadas-MacBook-Pro:project inma$ pwd  
/Users/inma/project  
(base) Inmaculadas-MacBook-Pro:project inma$ npm run serve
```

```
> project@0.1.0 serve  
> vue-cli-service serve
```

INFO Starting development server...

DONE Compiled successfully in 2727ms

App running at:

```
- Local:   http://localhost:8080/  
- Network: http://192.168.1.203:8080/
```

Note that the development build is not optimized.
To create a production build, run `npm run build`.

- PAS 4:

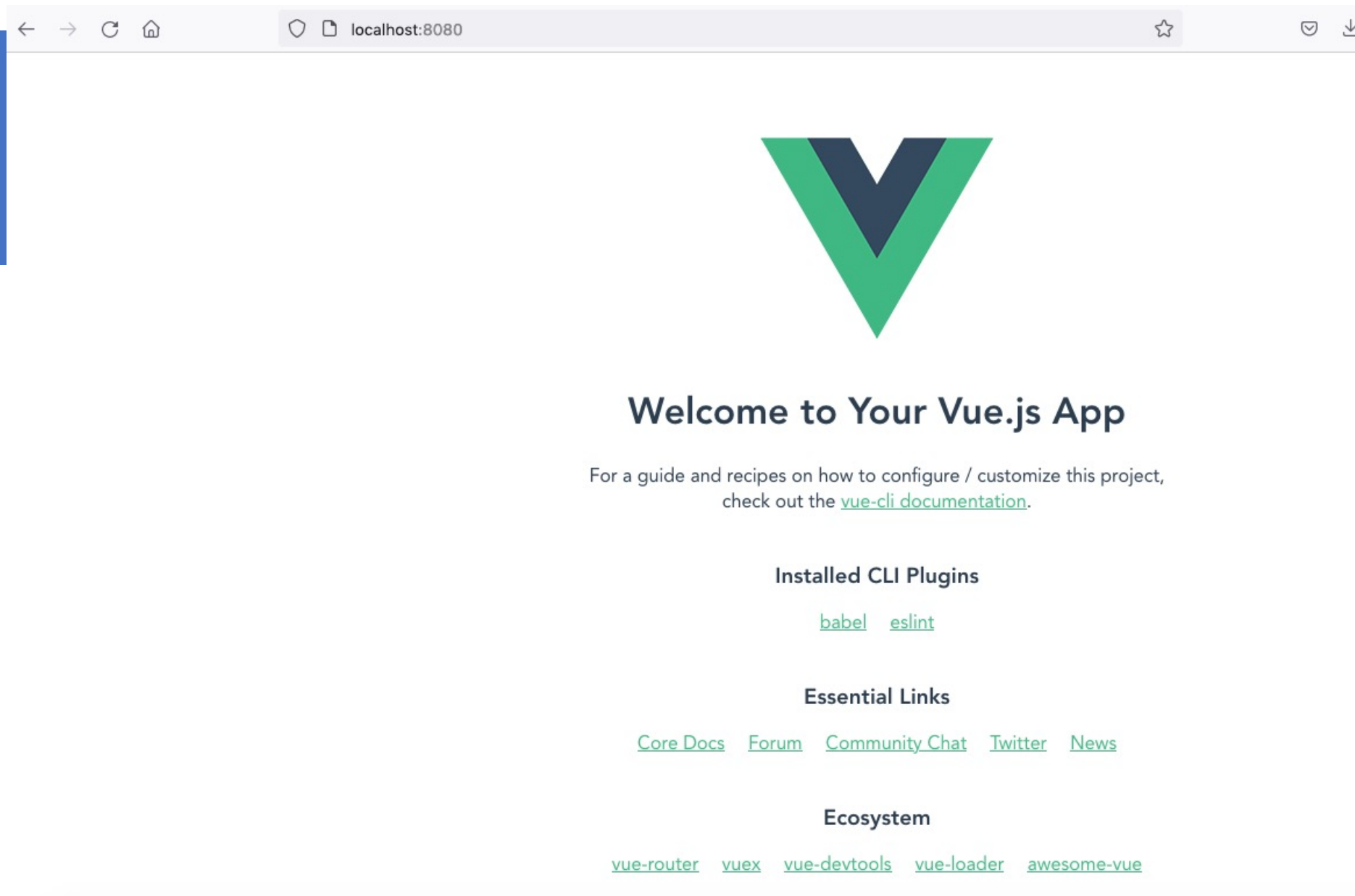
Accedir al
projecte i fer
“npm run serve”
per arrancar el
servidor

The Vue CLI

- PAS 5:

accedir a la url

<http://localhost:8080/>

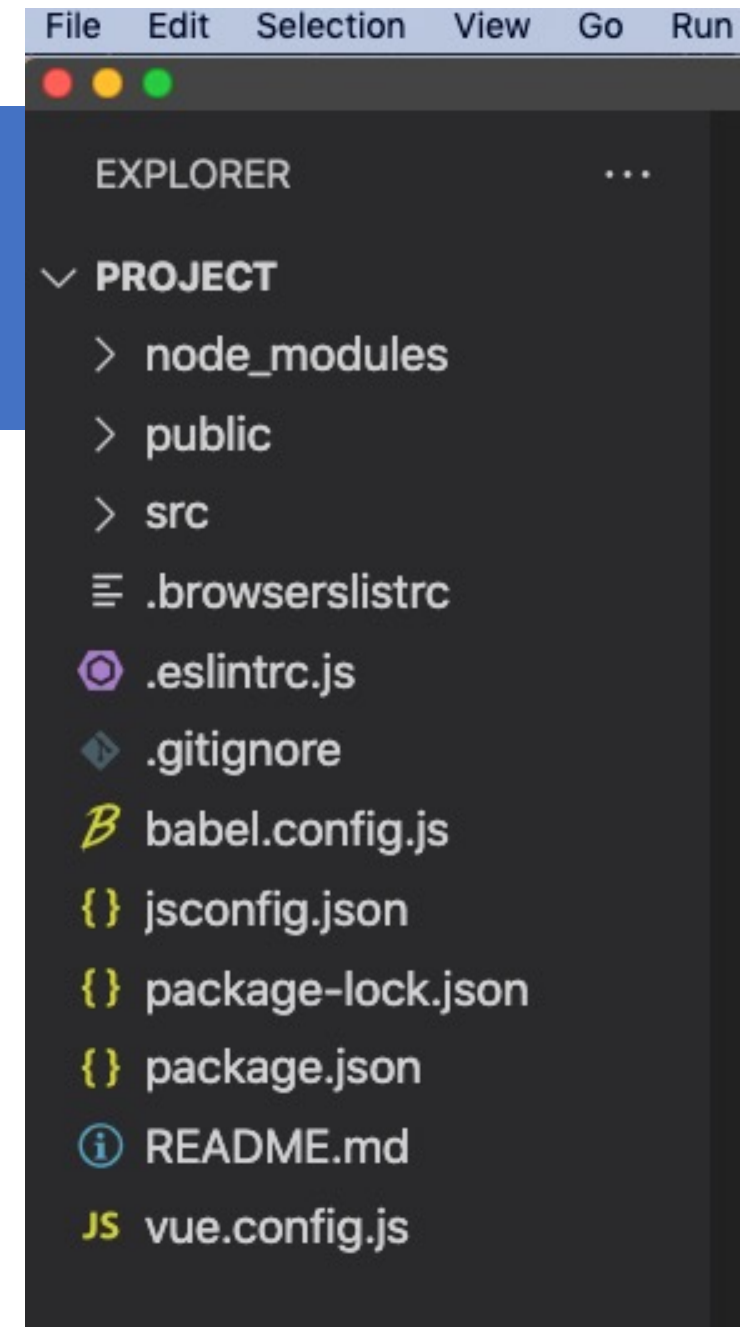
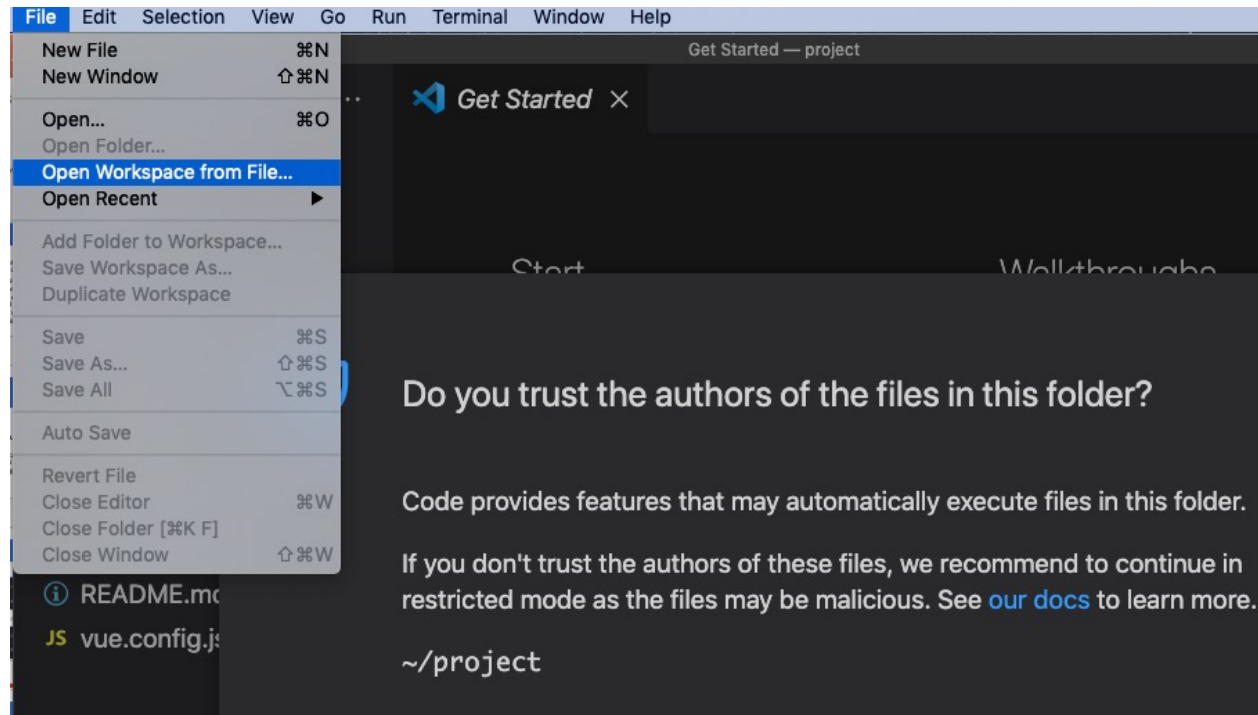


The Vue CLI

- Una aplicació Vue serà una combinació de components.
- Els components són com “mini aplicacions” amb el seu html, js, i css.
- Els components es poden comunicar.
- En el projecte que hem generat abans al pas 3.1:
 - HelloWorld és un component definit en `HelloWorld.vue`, amb el seu *template (html)*, *script*, i *style* i que importem des de l'aplicació `App.vue`
- A continuació inspeccionem el projecte...






The Vue CLI

- Inspeccionar proyecto creat:



The Vue CLI

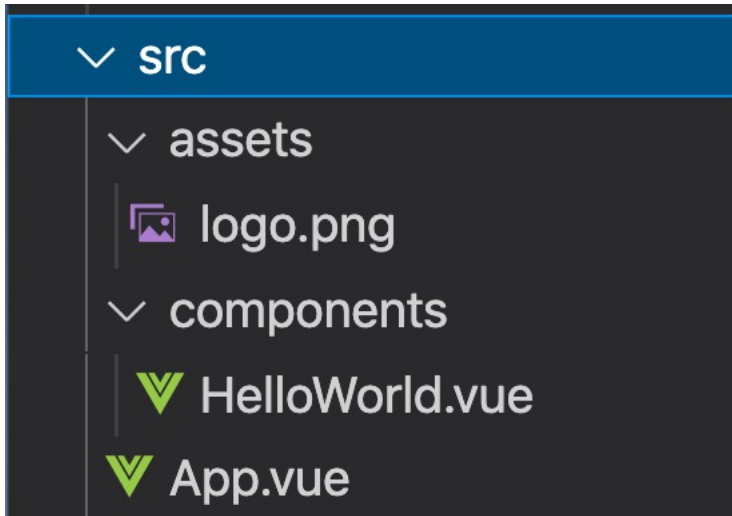
- Inspeccionar projecte creat:
 - A package.json tenim les dependències core-js i vue, no hem de fer els imports des de el nostre codi, s'han afegit automàticament (veure que hi ha fitxers a la carpeta "node_modules").
 - Si no tinguéssim la carpeta "node_modules" hem d'executar al terminal (Terminal-New Terminal) de VS Code "npm install".

 .gitignore	7	"build": "vue-cli-service
 babel.config.js	8	"lint": "vue-cli-service
 jsconfig.json	9	},
 package-lock.json	10	"dependencies": {
 package.json	11	"core-js": "^3.8.3",
	12	"vue": "^3.2.13"

The Vue CLI

- Inspeccionar projecte creat:

Al directori `src` estarà el nostre codi



```
1  <template>
2    
3    <HelloWorld msg="Welcome to Your Vue.js App"/>
4  </template>
5
6  <script>
7    import HelloWorld from '../components/HelloWorld.vue'
8
9    export default {
10      name: 'App',
11      components: {
12        HelloWorld
13      }
14    }
15  </script>
```

The Vue CLI

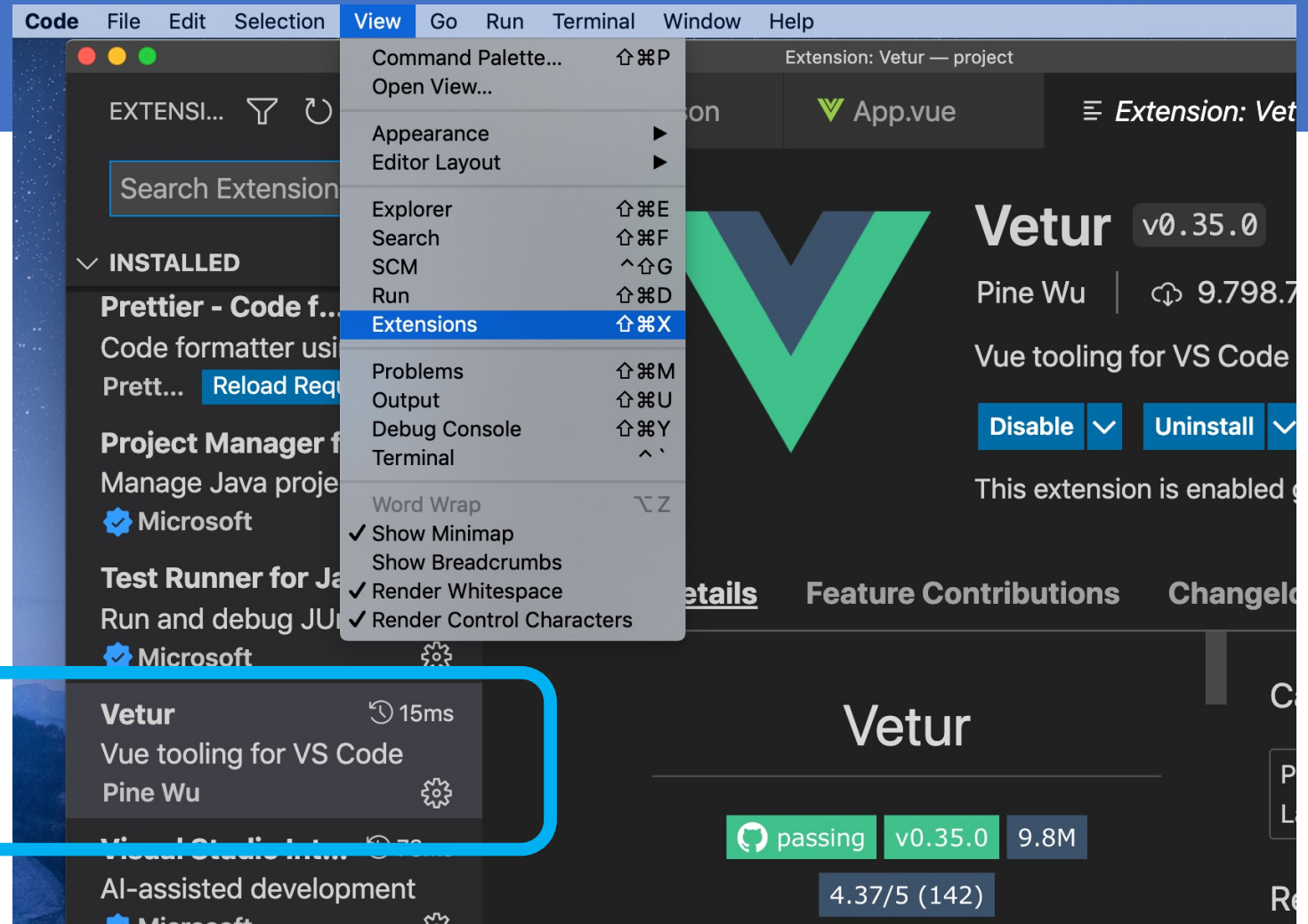
Una vegada fets els passos anteriors, per a crear un nou projecte anomenat `proyecto-ejemplo`, anar a la consola (o a *vue UI*) i escriure:

```
> vue create proyecto-ejemplo
```

The Vue CLI

- Extensions

Instal·lar la extensió **Vetur** per a tenir una experiència de desenvolupament millor amb codi Vue a VS studio (highlighting, autocompletion, ...)



Projectes basats en Vite

El successor de Vue Cli es [Vite](#)

Vue CLI is in Maintenance Mode!

For new projects, please use `create-vue` to scaffold Vite-based projects. Also refer to the [Vue 3 Tooling Guide](#) for the latest recommendations.

Bibliografia

<https://vuejs.org/guide/essentials/component-basics.html>

<https://cli.vuejs.org/>