

BO-HUB

B-INN-000

Introduction à Vue.js

Crée ta première application

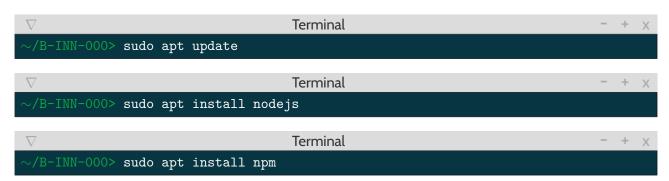


EPITECH.



INSTALATION DES OUTILS

La première chose à faire va être d'installer nodejs et npm.



Pour créer un projet Vue.js, exécutez cette ligne à l'endroit ou vous voulez créer le projet.

```
Terminal - + x

~/B-INN-000> npm init vue@latest

Project name: ... your-project-name

Add TypeScript? ... (No) / Yes

Add JSX Support? ... (No) / Yes

Add Vue Router for Single Page Application development? ... (No) / Yes

Add Pinia for state management? ... (No) / Yes

Add Vitest for Unit testing? ... (No) / Yes

Add Cypress for both Unit and End-to-End testing? ... (No) / Yes

Add ESLint for code quality? ... (No) / Yes

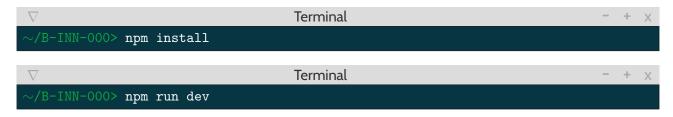
Add Prettier for code formatting? ... (No) / Yes

Scaffolding project in ./your-project-name...

Done.
```

Voilà votre premier projet Vue a été créer.

Maintenant vous pouvez rentrer dans le dossier de votre projet et lancer les commandes pour installer les dépendances puis faire tourner le site.



Vous pouvez maintenant accéder à votre site en localhost





ARCHITECTURE DE L'APPLICATION

Notre site commence dans le fichier index.html

Et c'est à l'intérieur de la balise suivante que notre application Vue.js va être ajouté.

```
<div id="app"></div>
```

Pour rendre notre code soit plus simple on va supprimer tout les fichiers dans le dossier src/components et on va remplacer le contenu du fichier src/App. vue par le code suivant.

```
<template>
</template>
<script setup>
</script>

<style>
@import './assets/base.css';
</style>
```

La balise template est destinée au code HTML et à l'utilisation des components Vue. C'est ici, que les éléments de la page sont (texte, image, tableau, etc...).

La balise script en Javascript va être utiliser pour pour coder la logique de notre page et d'autres... On peut par exemple créer des variables.

Et enfin la balise style en CSS pour ajouter du style à nos éléments.





UTILISATION DES TEMPLATES

Nous allons maintenant voir ce qui est possible avec les templates en Vue.js.

Vous pouvez utilisez une variable de cette manière grâce à la "Mustache" syntax

```
<template>
  <span>Message: {{ msg }}</span>
</template>

<script setup>
    const msg = "Bonjour j'apprends le Vue.js";
</script>

et vous pouvez aussi faire ça:
{{ number + 1 }}
{{ ok ? 'YES' : 'NO' }}
{{ message.split('').reverse().join('') }}
<div :id="`list-${id}`"></div>
```

Cette syntax ne marche pas pour les attributs des balises HTML. A la place Vue permet d'utiliser "v-bind:" ou seulement ":". Cela permet d'avoir des valeurs dynamiques pour les attributs.

On peut par exemple faire:

```
<div v-bind:id="dynamicId"></div>
<!-- ou-->
<div :id="dynamicId"></div>
```

dans ces cas l'attribut id de la balise va prendre la valeur d'une variable nomée dynamic I d créé dans script.

On peut aussi donner des valeurs booléenne (true/false) pour activer et désactiver un élément. Vous pouvez essayer de faire comme si dessous avec une variable isButtonDisabled avec une valeur true/false.

```
<button :disabled="isButtonDisabled">Button</button>
```

Et on peut également appeler des fonction de cette manière.

```
<span :title="toTitleDate(date)">
   {{ formatDate(date) }}
</span>
```





CLICS, CONDITIONS ET BOUCLES

Vue.js intègre de nombreuses directives qui commencent par v-.

On peut trouver v-html, v-bind, v-on, v-for, v-if, v-slot.

Toutes ces directives sont utilent et permettent de simplifier et rendre plus compréhensible le html avec le javascript.

Pour commencer, une des directives les plus utilisé est v-on. Elle est très utilisé pour savoir s'il y a eu un click ou si on est au dessus d'un élément (hover). Comme elle est très utilisé de la même façon que pour le v-bind il a un raccourci @. Voici on exemple où quand on clic sur notre élément la fonction doSomething() va être appelé.

```
<a v-on:click="doSomething"> lien </a>
<!-- shorthand -->
<a @click="doSomething"> lien </a>
```

Ensuite voici un exemple d'une utilisation du v-if qui va nous permettre de mettre des conditions sur des blocs pour dire s'ils doivent être afficher ou non sur la page. seen est une variable javascript (recherchez à quoi sert ref avec Vue.js)

```
Now you see me
```

Et enfin la directive v-for va nous permettre de ne pas recopier un bloc de code plusieurs fois grâce à une itération sur une liste d'objet javascript ou une suite d'index. Exemple :

```
const items = ref([{ message: 'Foo' }, { message: 'Bar' }])

    {{ item.message }}

<span v-for="n in 10">{{ n }}</span>
```





EXERCICE

Maintenant que vous avez les bases, essayez de faire une liste de boutons verticalement. Et quand on clic sur un bouton un texte doit apparaître juste à droite du bouton et disparaître au bout de 5 secondes.

Pour les textes des bouton et les textes qui s'affichent vous devrez utiliser cet objet:

```
const datas = [
   {textButton: 'Button 1', textClick: 'Ananas'},
   {textButton: 'Button 2', textClick: 'Banane'},
   {textButton: 'Button 3', textClick: 'Citrouille'},
   {textButton: 'Button 4', textClick: 'Datte'},
   {textButton: 'Button 5', textClick: 'Endive'}
]
```

Si vous réussissez appeler un encadrant pour qu'il vienne vérifier.

ETAPE SUIVANTE

Vous avez réussi a utiliser les v-for, v-if et plus. Vous allez désormais essayer de faire du routage de page avec Vue.js. Vous allez devoir créer une barre de navigation avec un bouton Accueil, Contact, A Propos. Et chaque bouton devra emmener vers une page différente.

Bonus : si une page avec un lien inconnu est demandé, envoyez une page "404 Not Found"

