



VUE : DU BONHEUR

Expérience de Développement (DX) en VueJS





Qui suis-je ?

- 1ère ligne de code en 1986... sur ZX81
- 1er site internet en... publié en 1995 (sur mygale.org)
- LinkedIn : [Mathieu LALLEMAND | LinkedIn](#)
> Ancien membre du bureau du Club Agile de Caen.
- Code Web : +25 ans d'XP
 - Vanilla JS
 - Mootools
 - JQuery
 - Angular
 - VueJS
- NCI : MOE depuis 2021 / Lead Dev & DevOps
 - Editeur d'applications métier
 - Développement sur Mesure
 - AMOE / Coaching d'équipes de développeurs.

LinkedIn



NCI

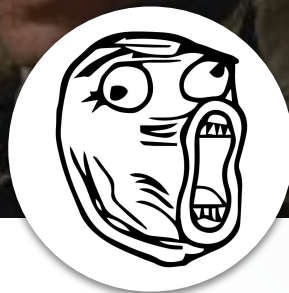


BLOG



ATTENTION : TROLLER ALERT

Vous le remarquerez peut-être, ce talk est volontairement orienté “Pro” Vue JS.



The background features several decorative elements: a series of overlapping white chevron shapes in the top-left corner; a horizontal dashed line in the top-right; a vertical dotted line in the bottom-left; and a cluster of light gray wireframe cubes in the bottom-right.

Pourquoi j'aime VueJS ?

```
render() {
  const { name } = this.state;
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <h1 className="App-title">Welcome to React</h1>
      </header>
      <input type="text" onChange={this.handleChange} />
      <br />
      <b>{name}</b>

      <Home />
    </div>
  );
}
```



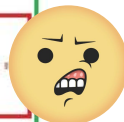
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>tp01php</title>
</head>
<body>
<H3>votre première page en PHP</H3>

<?php
echo '<p>Bonjour tout le monde</p>';
?>

</body>
</html>
```

HTML

code PHP



```
return (
  <Flex direction="column" className='ScoresContent Flex' mb={4} maxH=
  {"80vh"}>
    { /* Scrollable container for arrow values */ }
    <Box flex="1" mb={4} overflowY="auto" >
      {arrowValues.map(callbackfn: (row, rowIndex) => (
        <Flex key={rowIndex} mb={2} border="1px solid black" p={2}>
          {row.map(callbackfn: (value, valueIndex) => (
            <Text key={valueIndex} mx={1} fontSize="x1">
              {value || '-'}
            </Text>
          ))}
        </Flex>
      ))}
    </Box>

    { /* Buttons to Enter Arrow Values */ }
    <Flex wrap="wrap" spacing={2}>
      {Array.from(iterable: { length: 10 }, mapfn: (_, i) => i + 1).map
      (callbackfn: (number) => (
        <Button key={number} onClick={() => handleButtonPress(value: number)}
        >
          {number}
        </Button>
      ))}
    </Flex>
  </Flex>
)
```

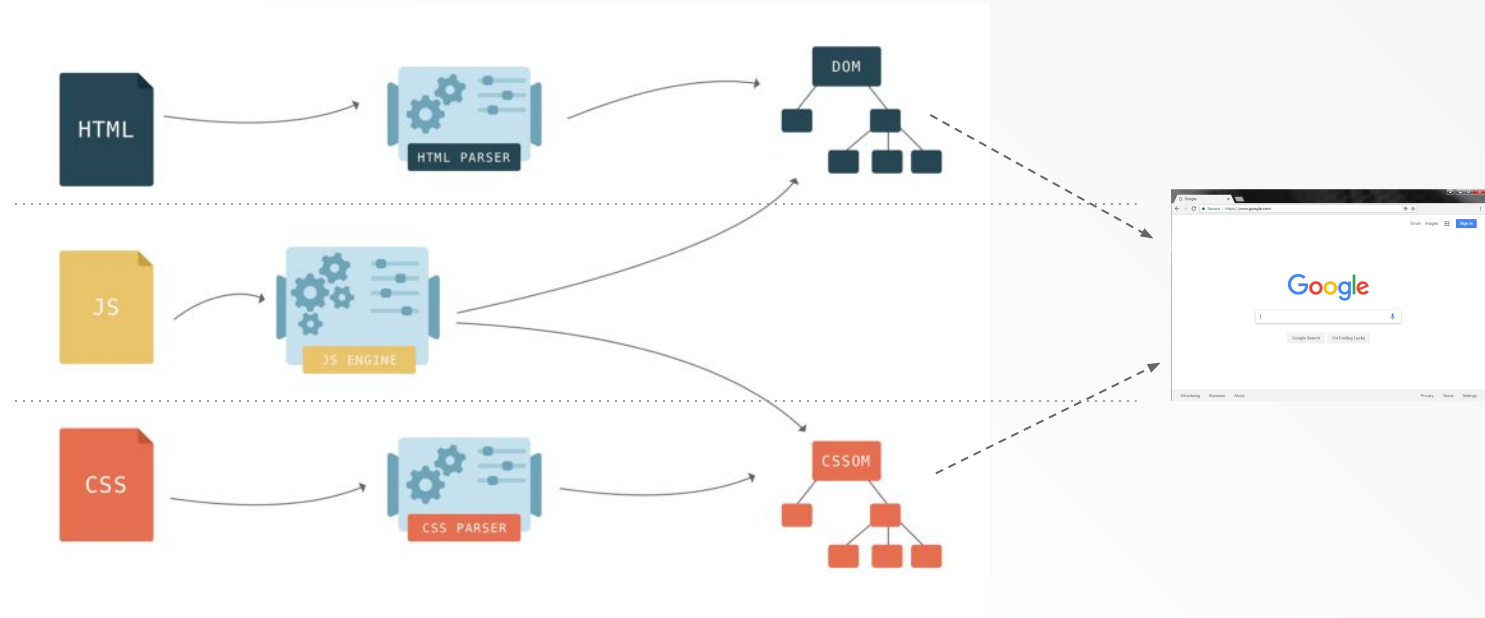




“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

Martin Fowler

C'est quoi cette Page Web ?



Separation of concern



The diagram illustrates the separation of concerns in a code editor. A dark-themed code editor window is shown with three distinct sections highlighted by green borders. Each section contains code for a different concern: HTML, JavaScript, and CSS. To the right of each section, the corresponding concern is labeled in green text. The code editor window also shows a status bar at the bottom indicating 'Line 21, Column 1', 'Spaces: 2', and 'Vue Component'.

<pre>1 <template> 2 <p>{{ greeting }} World!</p> 3 </template></pre>	HTML
<pre>5 <script> 6 module.exports = { 7 data: function () { 8 return { 9 greeting: 'Hello' 10 } 11 } 12 } 13 </script></pre>	Javascript ou TypeScript
<pre>15 <style scoped> 16 p { 17 font-size: 2em; 18 text-align: center; 19 } 20 </style></pre>	CSS ou Scss, ou Stylus, ou etc.



Vue SFC - Composition API / Script Setup + TailWind

```
1  <script setup>
2  import { ref } from "vue"
3  const greeting = ref('Hello');
4  </script>
5
6  <template>
7    <p>{{ greeting }} World !</p>
8  </template>
9
10 <style scope>
11   p {
12     @apply text-lg text-center;
13   }
14 </style>
```



React peut-il “vraiment” rivaliser ?



NON

Une **librairie** vient s'intégrer au code source initial pour lui **ajouter** de nouvelles **fonctionnalités**.

Un **Framework** transforme le code source initial en un **autre code source** utilisable par la machine / le navigateur.

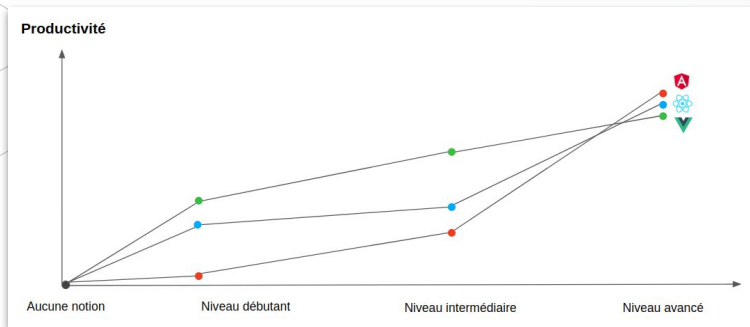
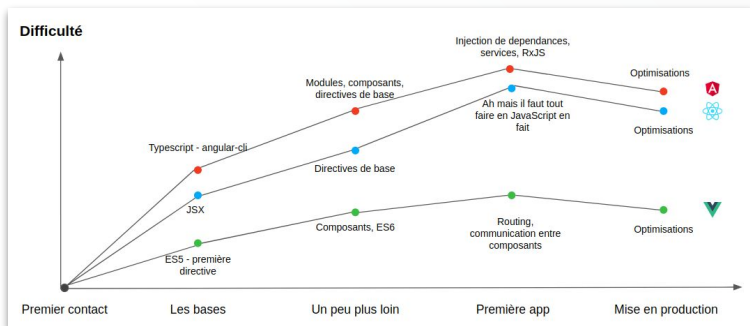
React est une **librairie**, **Vue** est un **Framework**.

*“React est à Vue, ce que le ciment, le sable, les graviers et les algos
sont à un mur déjà construit.”*



T'es sûr ?

Pour l'instant oui.



Name Duration for...	vanillajs-1	vue-v3.2.26	svelte-v3.46.2	react-hooks-v18.0.0	angular-v13.0.0
Implementation notes	772				
create rows creating 1,000 rows	81.7 ± 6.2 (1.00)	103.5 ± 4.0 (1.27)	120.8 ± 6.4 (1.48)	112.6 ± 3.8 (1.38)	113.7 ± 3.5 (1.39)
replace all rows updating all 1,000 rows (5 warmup runs).	77.5 ± 0.5 (1.00)	92.2 ± 1.3 (1.19)	97.9 ± 1.3 (1.26)	94.7 ± 2.0 (1.22)	104.8 ± 2.4 (1.35)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	20.2 ± 1.4 (1.00)	33.1 ± 1.2 (1.64)	29.9 ± 0.9 (1.48)	60.9 ± 2.1 (3.02)	52.8 ± 1.3 (2.62)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	46.1 ± 0.4 (1.00)	48.9 ± 0.6 (1.06)	49.1 ± 0.7 (1.07)	328.0 ± 3.2 (7.11)	352.4 ± 5.8 (7.64)
create many rows creating 10,000 rows	811.6 ± 39.7 (1.00)	1,029.1 ± 14.5 (1.27)	975.8 ± 24.9 (1.20)	1,293.4 ± 39.2 (1.59)	1,112.5 ± 10.4 (1.37)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	46.3 ± 1.8 (1.00)	62.7 ± 0.7 (1.35)	69.8 ± 1.6 (1.51)	63.9 ± 1.8 (1.38)	139.3 ± 1.2 (3.01)
geometric mean of all factors in the table	1.00	1.28	1.32	2.07	2.32
compare: Green means significantly faster, red significantly slower	com- pare	com- pare	com- pare	com- pare	com- pare

Angular vs Vue.js vs React vs Svelte: What Do The Statistics Say?
(kodaps.dev)

Vue 3 - Vapor Mode "devrait" permettre d'être à moins de 10% d'une performance 'vanilla' en consommant moins de RAM.

Qui utilise Vue ?

C'est là que c'est rigolo.

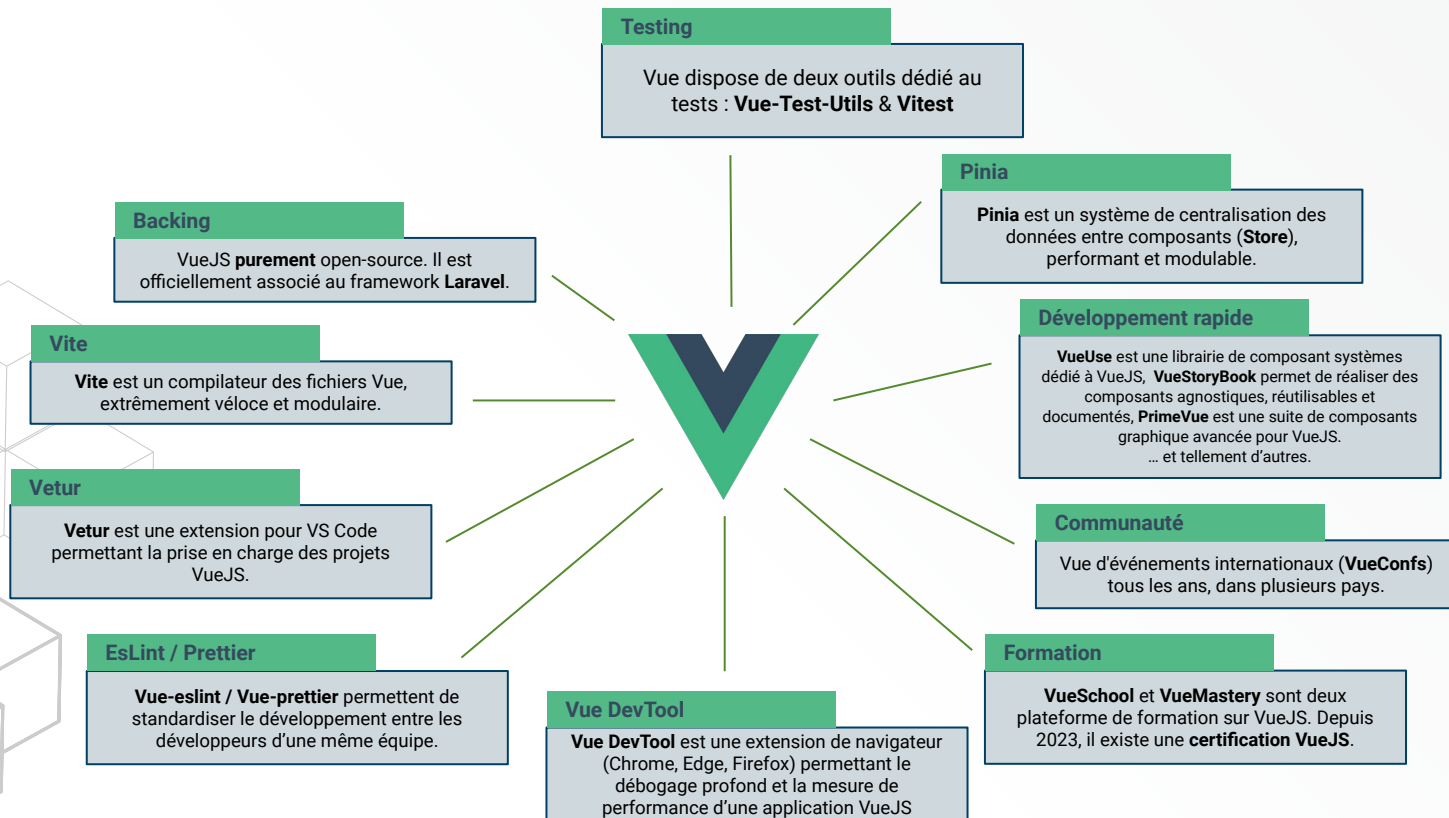


[Top 10 Global Companies using Vue.js - TatvaSoft Blog](#)

[15 Examples of Global Websites Using Vue.js in 2023 | Trio Developers](#)



Est-ce qu'il y a un "vrai" éco-système ?





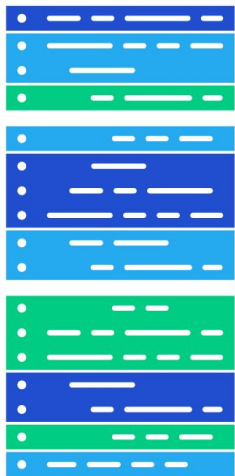
Vue 3

En mode TL;DR

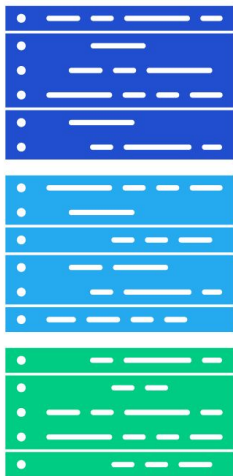


Composition API (vue >3.0)

Options API



Composition API



```
1 <script>
2   export default {
3     props: ['color'],
4     emits: ['update:name'],
5     data() {
6       return {
7         name: 'John Doe',
8         age: 30,
9         users: ['Jane', 'Mark', 'Bob'],
10      }
11    },
12    computed: {
13      details() {
14        return `${this.name} is ${this.age} years old`;
15      },
16      userList() {
17        return this.users.join(', ');
18      }
19    },
20    methods: {
21      updateName(newName) {
22        this.name = newName;
23        this.$emit('update:name', newName);
24      },
25      updateAge(newAge) {
26        this.age = newAge;
27      },
28      addUser(user) {
29        this.users.push(user);
30      },
31      removeUser(username) {
32        this.users = this.users.filter(user => user !== username);
33      }
34    },
35    watch: {
36      name(newName, oldName) {
37        // Do something when name changes
38      }
39    },
40    mounted() {
41      // Do something when component is mounted
42    },
43    updated() {
44      // Do something when component is updated
45    }
46  }
47 </script>
```

Options API

<https://www.webmound.com>

```
1 <script setup>
2   import { computed, onMounted, onUpdated, ref, watch } from 'vue';
3
4   const emits = defineEmits(['update:name']);
5
6   // Personal details section
7   const name = ref('John Doe');
8   const age = ref(30);
9   const details = computed(() => `${name.value} is ${age.value} years old`);
10  const updateName = (newName) => {
11    name.value = newName;
12    emits('update:name', newName);
13  }
14  const updateAge = (newAge) => {
15    age.value = newAge;
16  }
17  watch(name, (newName, oldName) => {
18    // Do something when name changes
19  });
20
21  // Users list section
22  const users = ref(['Jane', 'Mark', 'Bob']);
23  const userList = computed(() => users.value.join(', '));
24  const addUser = (user) => {
25    users.value.push(user);
26  }
27  const removeUser = (username) => {
28    users.value = users.value.filter(user => user !== username);
29  }
30
31  // Lifecycle hooks section
32  onMounted(() => {
33    // Do something when component is mounted
34  });
35
36  onUpdated(() => {
37    // Do something when component is updated
38  });
39 </script>
```

Composition API



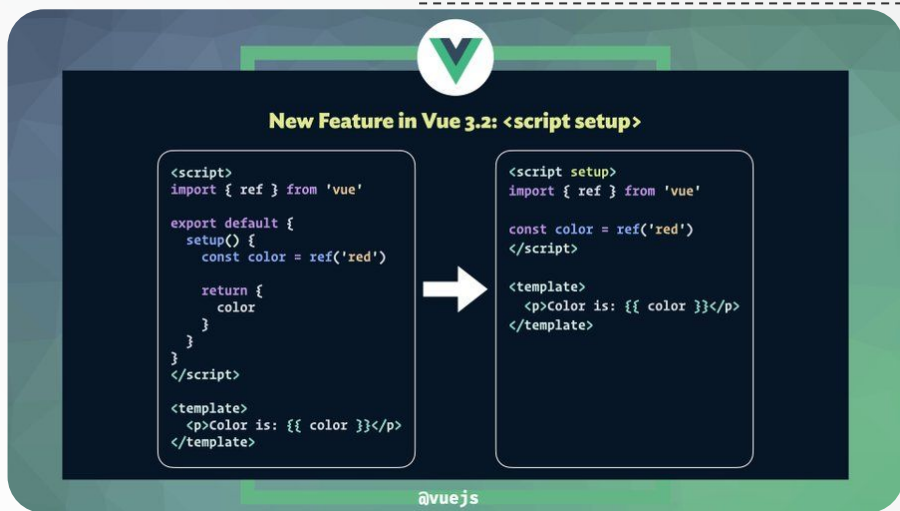
<script setup> (vue >3.2)

La balise **script setup** permet de basculer le code complètement en **composition API**.

Le code est plus **compact** et plus **simple** à organiser.

L'**import** de sous-composants dans le composant est **natif** en script setup.

Il sera **requis** pour activer le futur **vapor-mode**.



New Feature in Vue 3.2: <script setup>

```
<script>
import { ref } from 'vue'

export default {
  setup() {
    const color = ref('red')
  }

  return {
    color
  }
}
</script>

<template>
<p>Color is: {{ color }}</p>
</template>
```

→

```
<script setup>
import { ref } from 'vue'

const color = ref('red')
</script>

<template>
<p>Color is: {{ color }}</p>
</template>
```

@vuejs

```
script-setup-demo - App.vue

1 <template>
2   <HelloWorld message="Hello World!!!" /> <!-- PascalCase -->
3   <hello-world message="Hello World!!!" /> <!-- kebab-case -->
4 </template>
5
6 <script setup>
7 import HelloWorld from "../components/HelloWorld";
8 </script>
```

Les Slots

Permettent d'**injecter** du contenu dans un **composant**.

Il est possible d'en définir **plusieurs** en les **nommants** afin de réaliser de **composants complexes** mais malléables.

```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot></slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

```
<BaseLayout>
  <template #header>
    <h1>Here might be a page title</h1>
  </template>

  <!-- implicit default slot -->
  <p>A paragraph for the main content.</p>
  <p>And another one.</p>

  <template #footer>
    <p>Here's some contact info</p>
  </template>
</BaseLayout>
```



Les composables c'est la joie

```
import { ref, onMounted, onUnmounted } from 'vue'

export function useMouse () {

  const x = ref(0)
  const y = ref(0)
  function update(event) {
    x.value = event.pageX
    y.value = event.pageY
  }
  onMounted(() => window.addEventListener('mousemove', update))
  onUnmounted(() => window.removeEventListener('mousemove', update))

  return { x, y }
}
```

Les **composables** sont des **fonctions réactives** permettant une grande réutilisabilité du code.

```
<script setup>
import { useMouse } from './mouse.js'
const { x, y } = useMouse ()
</script>
<template>
  Mouse position is at: {{ x }}, {{ y }}
</template>
```

Mouse position is at: 856, 2265



Des perfs qui s'améliorent toutes seules.

Vue 3.0.0

- 41% Lighter tree shaking
- initial render 55% faster
- updates up to 133% faster
- memory usage 54% less

Vue 3.2.0

- ref : 260% faster read / 50% faster write
- 40 % faster dependency tracking
- 17% less memory usage
- 200% faster creation of plain vNodes

3.3.6

- WeakMaps & WeakSets (less memory usage)

...

[core/CHANGELOG.md at main · vuejs/core \(github.com\)](#)



Eco-Système de rêve

*“La simplicité est la sophistication suprême”
Léonard de Vinci*

Vue dispose d'extension “officielles”

Vue Router

Vue Router

Vue Routeur est le coeur des **SPA** (Single Page Applications). Son objectif est de **remplacer** un **composant** par un **autre** au sein de VueJS pour **simuler une navigation** entre différentes “pages”.

VueRouter est “transparent” avec **NuxtJS**

Installation

```
npm install vue-router@4
```

Définitions des routes (fichier /routes.js)

```
export const routes = [
  { path: '/', component: () => import('@/components/TheHome.vue') },
  { path: '/about', component: () => import('@/components/TheAbout.vue') },
  ...
]
```

Ajout du routeur dans le fichier “main.js”

```
import { routes } from '@/routes.js'
const router = VueRouter.createRouter({
  history: VueRouter.createWebHashHistory(),
  routes
})

const app = Vue.createApp({})
app.use(router)
app.mount('#app')
```

Utilisation du “router-link” / “router-view” dans App.vue

```
<div>
  <h1>Hello App!</ h1>
  <p>
    <router-link to="/">Go to Home</ router-link>
    <router-link to="/about">Go to About</ router-link>
  </p>
  <router-view></router-view>
</div>
```



VueX était un problème. Pinia est une solution.

Pinia est un “**store**” de données. **Remplaçant Vuex** il est beaucoup plus **flexible** et **simple** à utiliser.

Il permet de **centraliser** les données manipulées par **plusieurs composants**, et de **synchroniser** en un **point de confiance** unique (SPOT) les données à diffuser aux composants.

Pinia est **modulaire** et **extensible** via un système de **plugins** très flexible.

Le use-case privilégié par NCI est de réaliser un pinia par **ressource** (ou **modèle**), chaque pinia incluant une **collection d'éléments** du modèle, ainsi que le formulaire pour créer / modifier le modèle.

Installation

```
npm install pinia
```

Ajout de pinia dans le "main.js"

```
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import App from './App.vue'

const app = createApp(App)
app.use(createPinia()).mount('#app')
```

Création du store "counter"

```
// stores/counter.js
import { defineStore } from 'pinia'
export const useCounterStore = defineStore('counter', {
  state: () => ({ count: 0 }),

  actions: {
    increment () { this.count++ },
  },
})
```

Utilisation dans le composant du store "counter"

```
<script setup>
import { useCounterStore } from '@/stores/counter'
const counter = useCounterStore ()

counter.count ++ // with autocompletion ✨
counter.increment ()
</script>

<template>
  <!-- Access the state directly from the store -->
  <div>Current Count: {{ counter.count }}</div>
</template>
```

Vue I18n

Vue **I18n** permet de localiser (**traduire**) très **facilement** les applications VueJS.

Il permet de **gérer** les **pluriels** sur différents niveaux (0,1,N) ou (0, 1, X, Y, N) selon certaines langues.

Il prends aussi en compte les **dates** et les **nombres**.

Augmente la **maintenabilité** du code et **simplifie** l'étape de **traduction**.

```
npm install vue-i18n@8
```

Installation

```
export default {  
  hello: 'こんにちは、世界'  
}
```

Trad. Japonaise

```
export default {  
  hello: 'hello world'  
}
```

Trad. Anglaise

Définitions des locales dans le fichier locales/index.js

```
import locale_en from "@/locales/en.js"  
import locale_ja from "@/locales/ja.js"  
  
export default const messages = {  
  en: { message: locale_en },  
  ja: { message: locale_jp }  
}
```

Configuration du fichier main.js

```
import locales from "@/locales"  
const i18n = new VueI18n({  
  locale: 'en', // set locale  
  messages, // set locale messages  
})  
  
const app = Vue.createApp({})  
app.use(i18n)  
app.mount('#app')
```

Utilisation dans le template .vue

```
<template>  
  <span>{{ $t('hello') }}</span>  
</template>
```





Je vous montre ?

a quel point c'est du bonheur ?



Vue c'est S.O.L.I.D.

Single Responsibility Principe : “L'élément ne doit faire qu'une et une seule chose.”

> Components, Composables, Pinia, Router, etc...

Open/Closed Principe : “L'élément doit être ouvert à l'extension, fermé à la modification”

> Props & Emits, Composables, Slots

Liskov Substitution Principe : “L'élément doit pouvoir être remplacé par un élément de type similaire”

> Slots

Interface Segregation Principe : “L'élément ne doit pas imposer des dépendances à des méthodes inutiles aux autres éléments”

> Props & Emits, Default values

Dependency Inversion Principe : “L'élément doit dépendre des

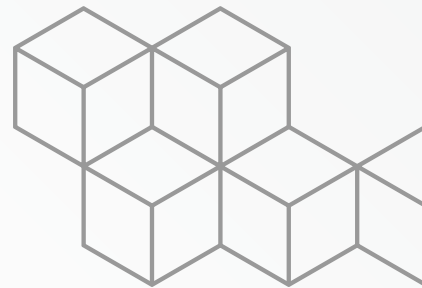
> Pinia, Props & Emits.





Les Librairies qui font du bien

**Trouve des solutions,
pas des problèmes**



Vitest

Vitest est un outil permettant de réaliser des **tests unitaires** sur des **fonctions** en javascript. Bien qu'initialement conçu par Antony Fu (**développeur** de la **CoreTeam** de VueJS).

Il n'est pas uniquement réservé à VueJS.

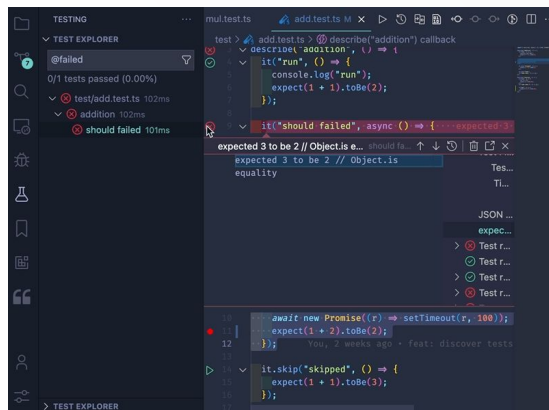
Basé sur les patterns de ViteJS, il est très rapide et permet une exécution en continue de la suite de tests.

Bonus : Le plugin vscode est top.

```
DEV v0.34.5 /home/mat/Sources/LHOTTELIER/Demo1
✓ src/tests/demo.spec.js (1)
  ✓ adds 1 + 2 to equal 3

Test Files 1 passed (1)
Tests 1 passed (1)
Start at 15:43:33
Duration 453ms (transform 33ms, setup 0ms, collect 0ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```



Installation
`npm install -D vitest`

Fichier à tester

```
// sum.js
export function sum(a, b) {
  return a + b
}
```

Fichier de test

```
// sum.test.js
import { expect, test } from 'vitest'
import { sum } from './sum'

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3)
})
```

Modification du fichier package.json

```
{
  "scripts": {
    ...
    "test": "vitest"
    ...
  }
}
```



Vue Test Utils

Vue Test Utils

Vue Test Utils permet de monter des composants dans un **DOM synthétique** afin de **tester** les comportements du composant.

Les tests sont **très rapides** et permettent de valider la **non-régression** des composants.

Installation

```
npm install @vue/test-utils --save-dev
```

Ajout du fake-DOM dans la config vite.config.js

```
...  
test: {  
  environment: "happy-dom",  
},  
...
```

Exemple d'utilisation

```
import { mount } from '@vue/test-utils'  
  
// The component to test  
const MessageComponent = {  
  template: '<p>{{ msg }}</p>',  
  props: ['msg']  
}  
  
test('displays message', () => {  
  const wrapper = mount(MessageComponent, {  
    props: { msg: 'Hello world' }  
  })  
  
  // Assert the rendered text of the component  
  expect(wrapper.text()).toContain('Hello world')  
})
```

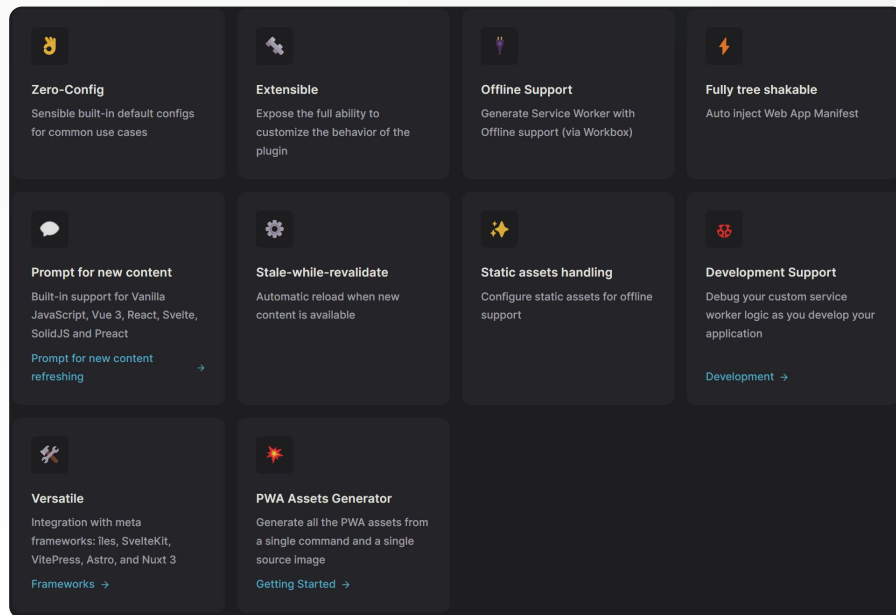
Vite PWA

Vite **PWA** (Progressive Web App) permet de transformer **n'importe quelle application javascript en application mobile**.

L'application est **stockée sur le device**, puis exécutée au travers d'un navigateur sans que la fenêtre du navigateur ne soit forcément visible, donnant un **look & feel "natif"** à l'application PWA.

L'application PWA peut fonctionner **sans accès à internet** ou en **mode hybride** grâce à une **détection de la connectivité**.

L'application PWA s'installe **sans passer par un store** et peut se mettre à jour **automatiquement** dès qu'une mise à jour est **disponible**.



PrimeVue

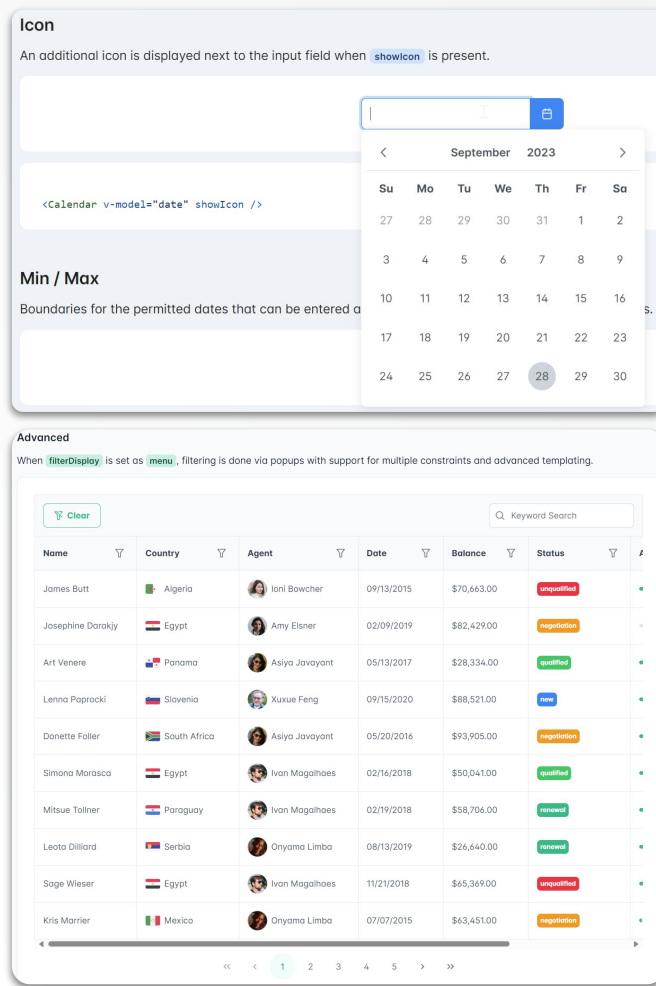
PrimeVue

PrimeVue est une **bibliothèque de composants UI** pour Vue 3 prêts à l'emploi.

Évite de “réinventer la roue” concernant la **création de composants**

Dispose de **nombreux composants complexes** (calendar, org-chart, datatable, dock, ripple effect, terminal, etc.)

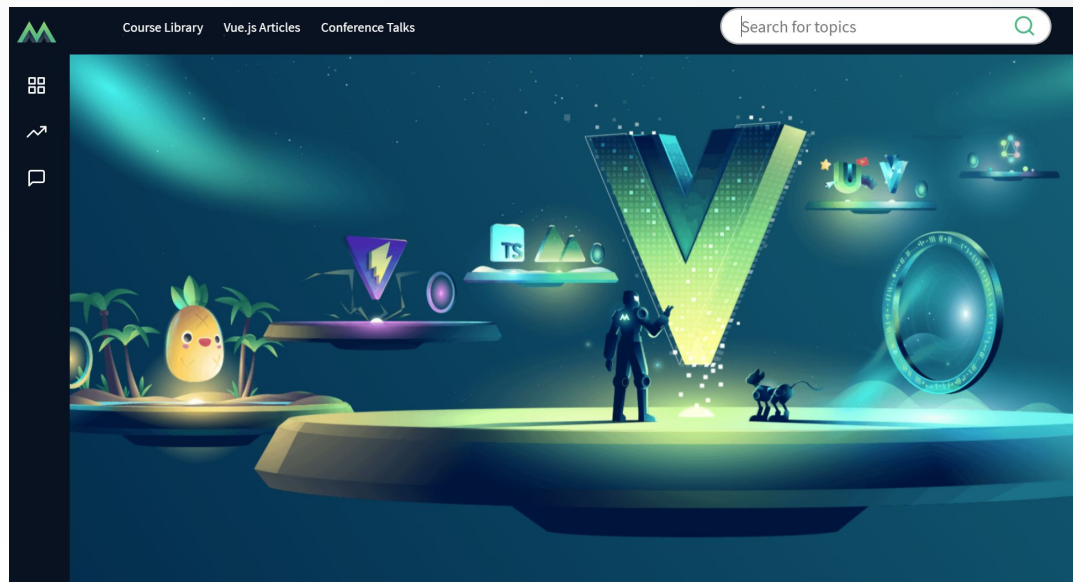
Définition du Design System avec TailwindCSS possible



Vue Mastery

Env. \$15 / mois

- **Gros volume** de formations **pertinente** et à jour.
- **Séparation** des formations **vue2** et **vue3**.
- **Séparation** des formations **"Option API"** et **"Composition API"**.
- **Formations** par le créateur de vue (**Evan You**) disponibles.
- **Retours d'expériences** par la **Core Team** disponibles (Evan, Eduardo, etc.)
- Formations très **techniques disponibles**
- Learning Path
- Sous-titres **anglais** uniquement.
- Vitesse x1.25 / x1.5 / x2 disponible.



VOX MEDIA





MERCI

des questions ?

