Sparks

Très heureux de vous rencontrer

Sparks

Formation VUE.JS

Animé par Michel BOCCIOLESI

L’équipe Sparks

Vous souhaite une **bonne formation**

Pour toute question vous pouvez nous joindre au

04.78.22.10.38

de 09:00 à 12h30 et de 14:00 à 18:00

Vous pouvez également nous envoyer un mail à

demande@sparks-formation.com

Table des matières

Vue une librairie ou un framework ?

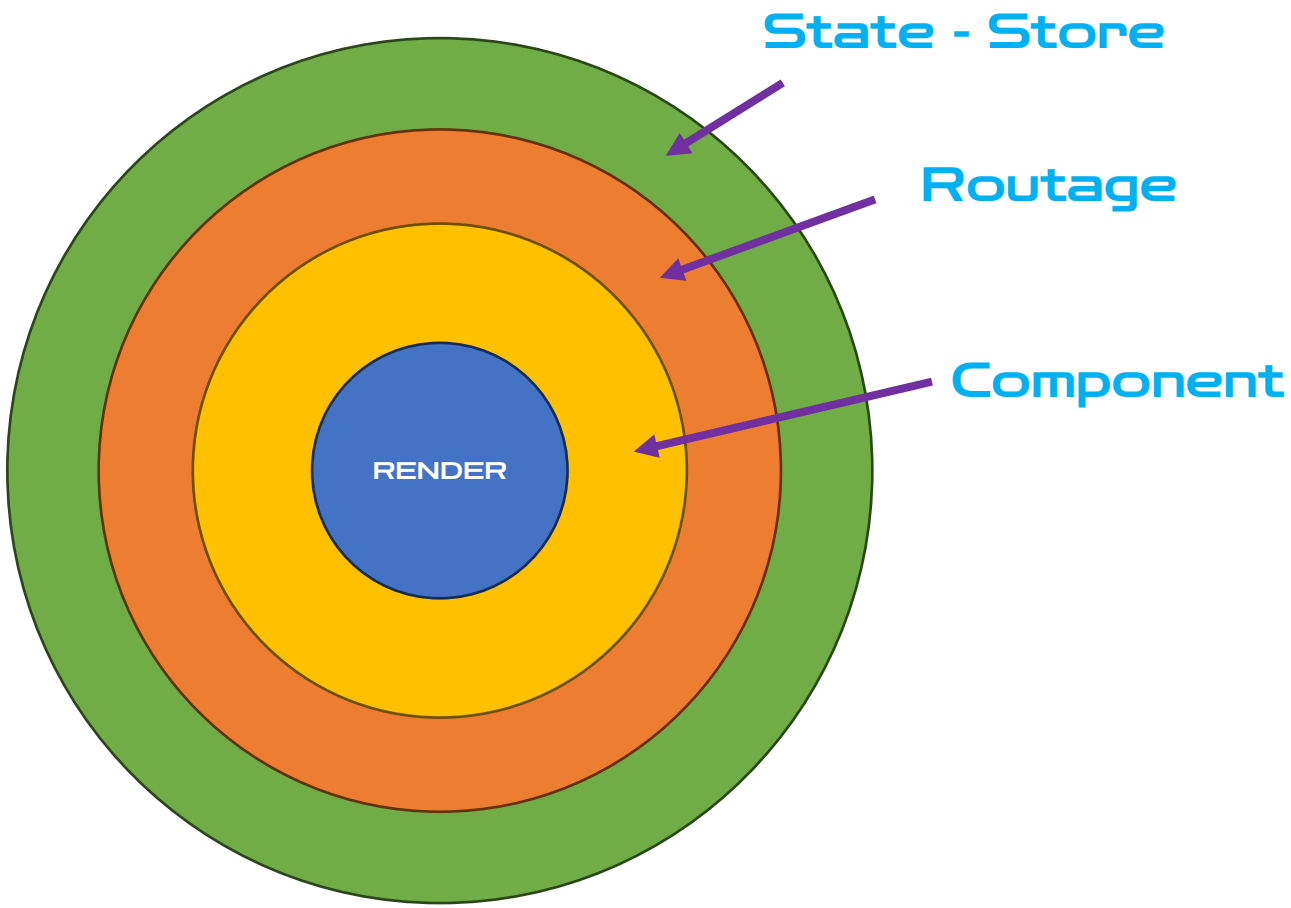
Dans le monde du développement Front, il existe plusieurs catégories de frameworks :

- Angular (2016) est un framework « orienté »
- React (2013) est une librairie
  mais peut être piloté et utilisé par d'autres framework (« NextJS » ou « Vite »)
- Vue (2014) est un framework « progressif [1]» .
  Il peut cependant être utilisé en tant que librairie. 🤔
  Il possède ses propres outils de création npm create vue ou Vite [2]

[1] *Le projet peut être simple initialement et peut facilement évoluer vers un projet beaucoup plus complexe, grâce à sa modularité*

[2] *Vite a été développé en **2020 par Evan You** le créateur de **Vue, Vite et Vitest***

Framework progressif



Vue une librairie ou un framework ?

Tous ces frameworks utilisent NODEJS (2009), un serveur javascript développé par Google.

NodeJS permet de travailler son projet Web en tant que WEBPACK.

C'est beaucoup plus facile de structurer, d'architecturer un projet avec un webpack.

NB : Même en Vanilla, on peut développer avec un Webpack. La toute 1ère étape est de créer un projet avec la commande **npm init** (-y)

Histoire du Web

- **2003** : Création du WHATWG
- **2007** : HTML5 et les API JS – le tout 1er smartphone est présenté
- **2010** : Responsive Web Design
- **2011** : Bibliothèque CSS Bootstrap
- **2015** : le développement Front devient très important, les frameworks, le webpack sont de plus en plus utilisés

EcmaScript : la normalisation de Javascript

- **1999** : version 3
- **2009** : version 5 (la 4 n'a jamais existé)
- **2015**: version 6
- 2016
- 2017
- ....
- ES Next

Réactivité de Javascript

Javascript n’est pas du tout réactif … 🥵

Finalement quel Problème doit-on résoudre ?
Tous les Frameworks modernes essaient de résoudre ce même problème

La notion de réactivité en Front End

	A	B
1	Montant HT	150
2	Tva	0,2
3	Montant TTC	=B1*B2+B1

Un tableur est réactif par défaut, la formule se recalcule à chaque modification des données sources.

Javascript ne sait pas le faire par défaut.
Bien sûr on a depuis bien longtemps trouver des « parades » pour rendre le code javascript réactif.

Chaque Framework propose sa ou ses propres solutions.
Angular : les Observables et les signaux
Vue : les références réactives (ref, reactive, computed, watchEffect …)
React : les références réactives (useRef, useEffect, …)

index.html × script.js ×

```
1  let montantHT=150;
2  const TVA=0.2;
3  let total=montantHT*TVA + montantHT;
4  console.log(total);
5
6  let montantHt=200;
7  console.log('Est ce réactif ? : ', total, '🤔😣');
8
```

Console ×

```
180
Est ce réactif ? :  180 🤔😣
```

Rappels – Révisions Ecmascript

Ecmascript 2015+

Ecmascript, la normalisation du langage Javascript a révolutionné Javascript en 2015 (ES6) en le dotant de fonctionnalités et spécificités dignes d'un « langage de haut niveau ».
Cette transformation totale et globale devenait nécessaire au vu des besoins des développements front web et mobile.

Les améliorations les plus importantes du langage concernent :

1. La définition des variables et leur portée : LET et CONST
   Let et const ont un scope local, leur portée est limitée au bloc {}
   const n'est pas ré-assignable

2. La possibilité de créer des modules de code exportable et importable : IMPORT, EXPORT
   import et export sont très utilisés par tous les frameworks actuels

3. L'écriture simplifiée des fonctions grâce aux FAT ARROWS
   le mot clé « function » n'est plus nécessaire

Ecmascript 2015+

Les améliorations les plus importantes du langage concernent :

1. La string interpolation des variables avec ${maVariable} et la `concaténation avec les back stits`
   il n'est plus nécessaire d'utiliser le + comme opérateur de concaténation
   on parle désormais de string interpolation

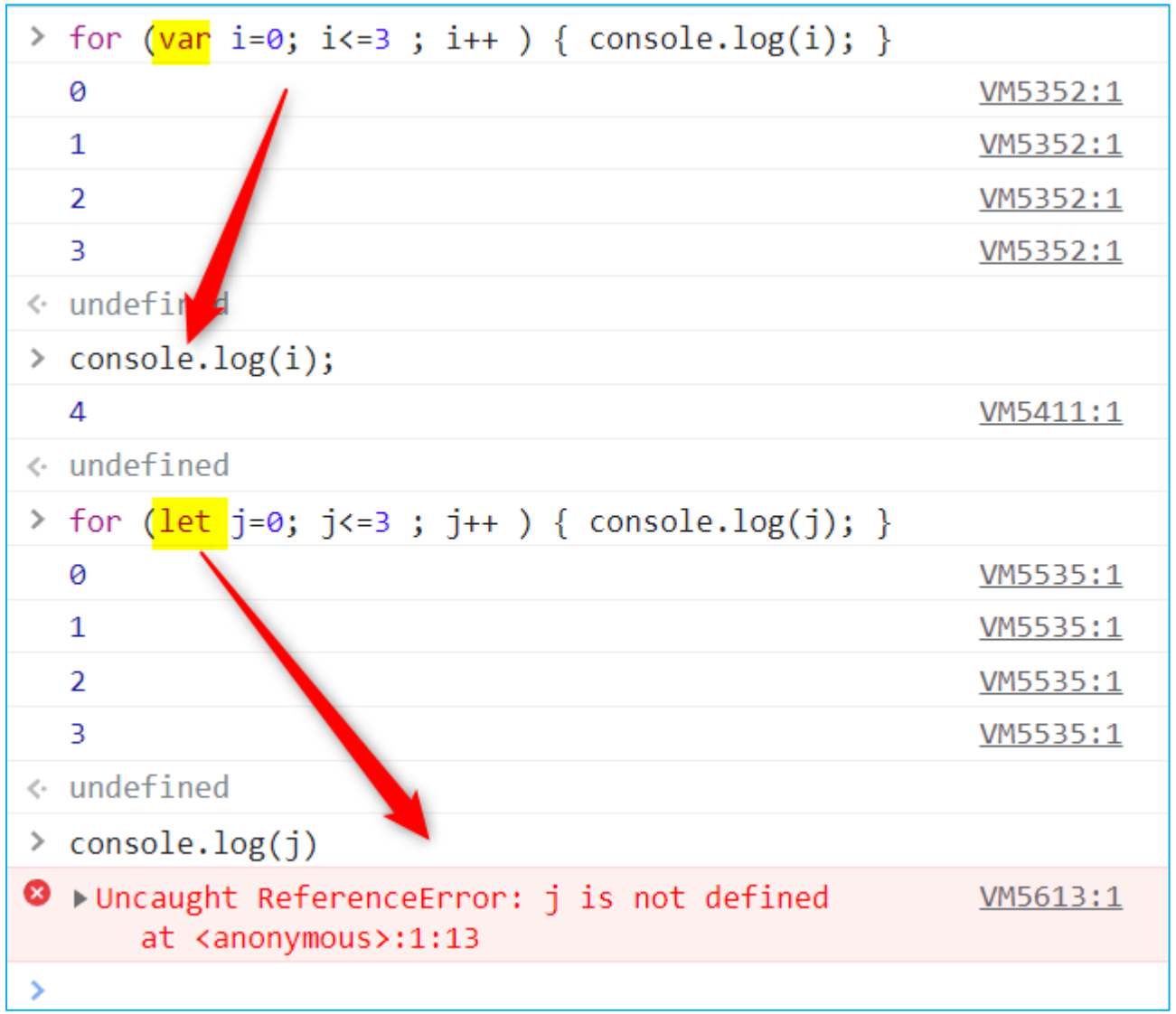
2. Les itérateurs et les boucles FOR OF, FOR IN et FOR OF ENTRIES
   ont beaucoup simplifier l'usage classique du « for »

3. Les spread operators pour déstructurer les tableaux
   permettent de manipuler facilement les arrays

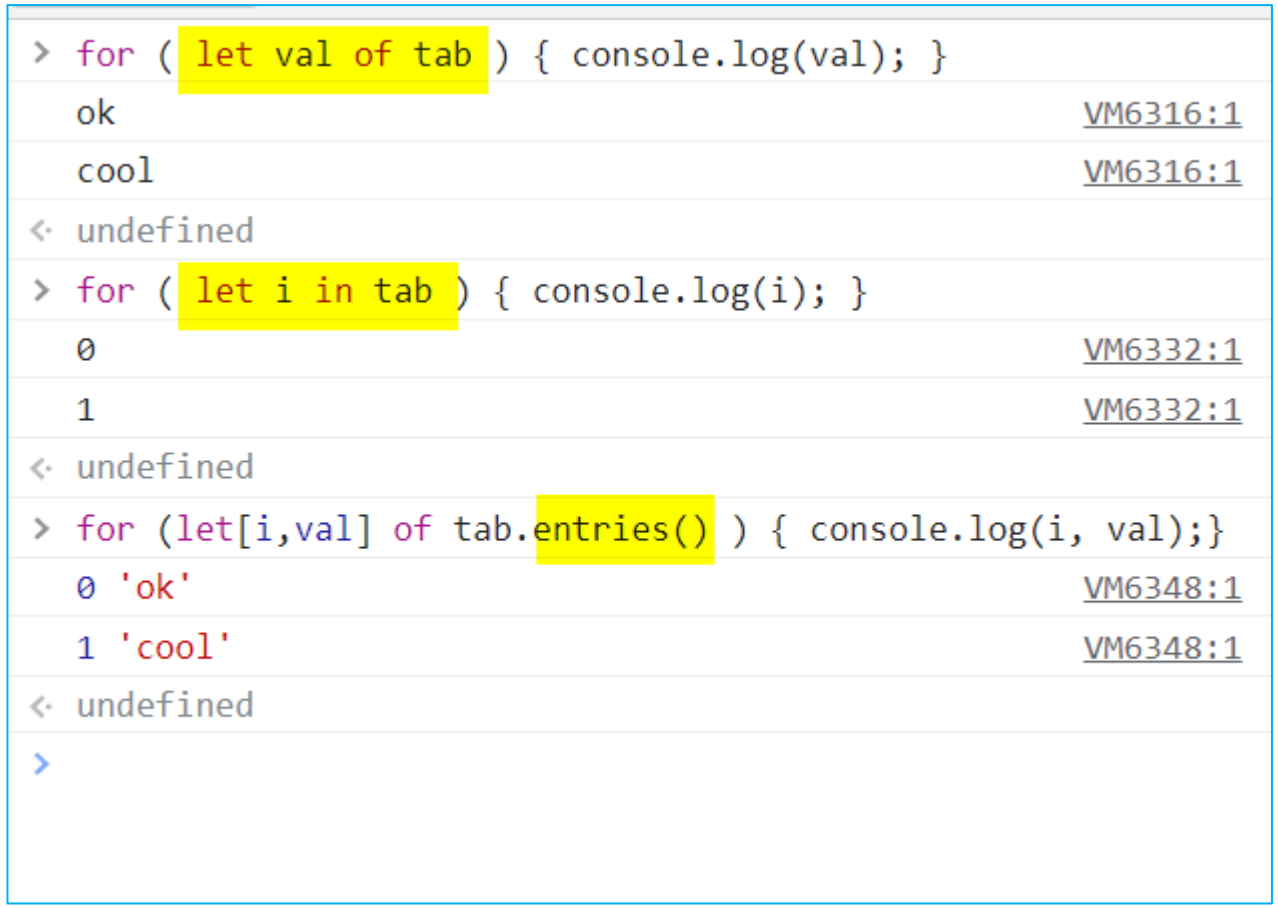
4. La notion de promesses afin de gérer les événements et les procédures asynchrones
   ES2017 a ajouté le couple async/await pour encore mieux gérer l'asynchronisme
   l'api « fetch » qui permet de d'effectuer des queries HHTP en mode asynchrone renvoie une promesse
   , tout comme la fonction « json() » qui permet de récupérer le body d'une requête GET

5. Les objets et les CLASS
   même si « class » est un sucre syntaxique , on peut enfin écrire des class en javascript

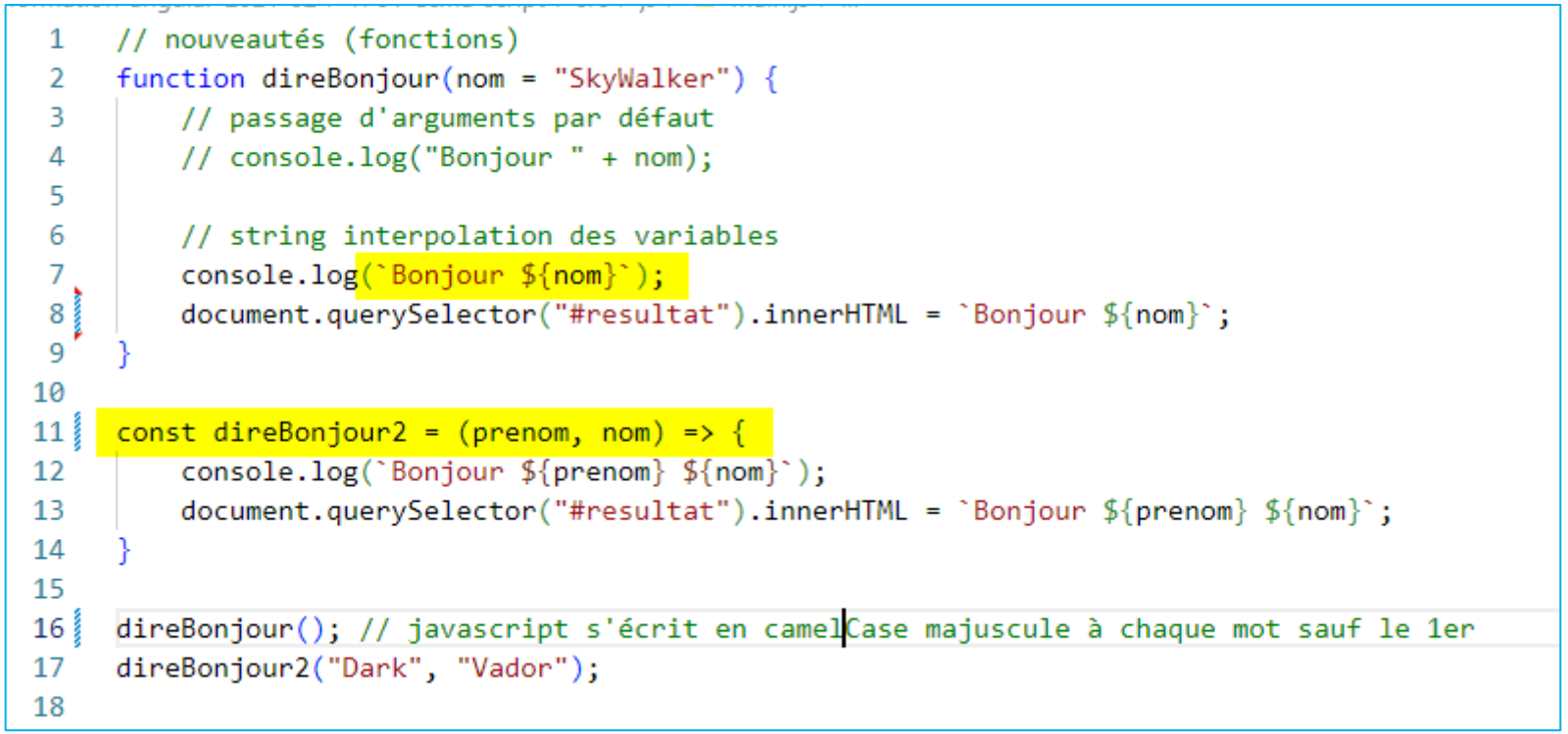
Ecmascript 2015+



Ecmascript 2015+

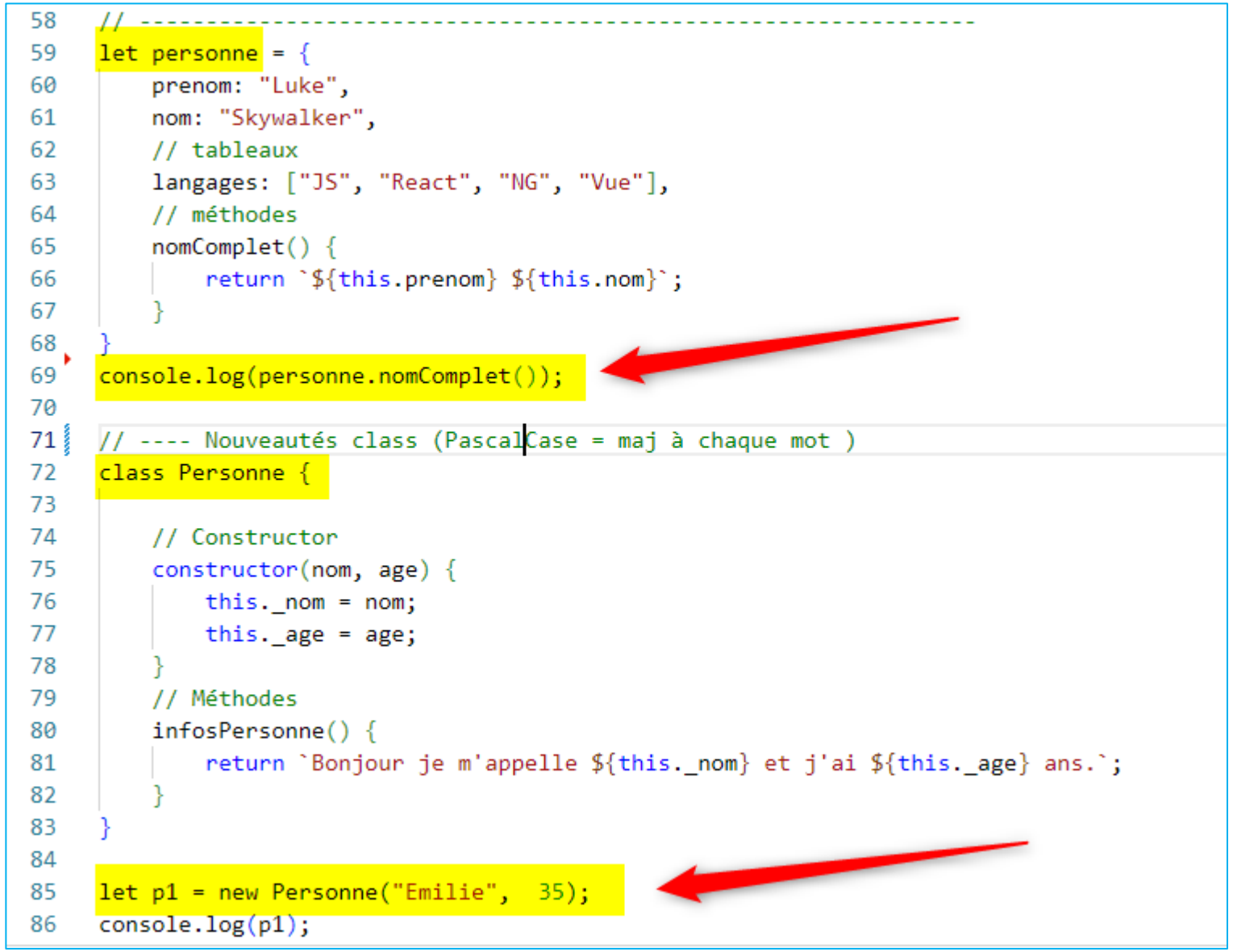


Ecmascript 2015+



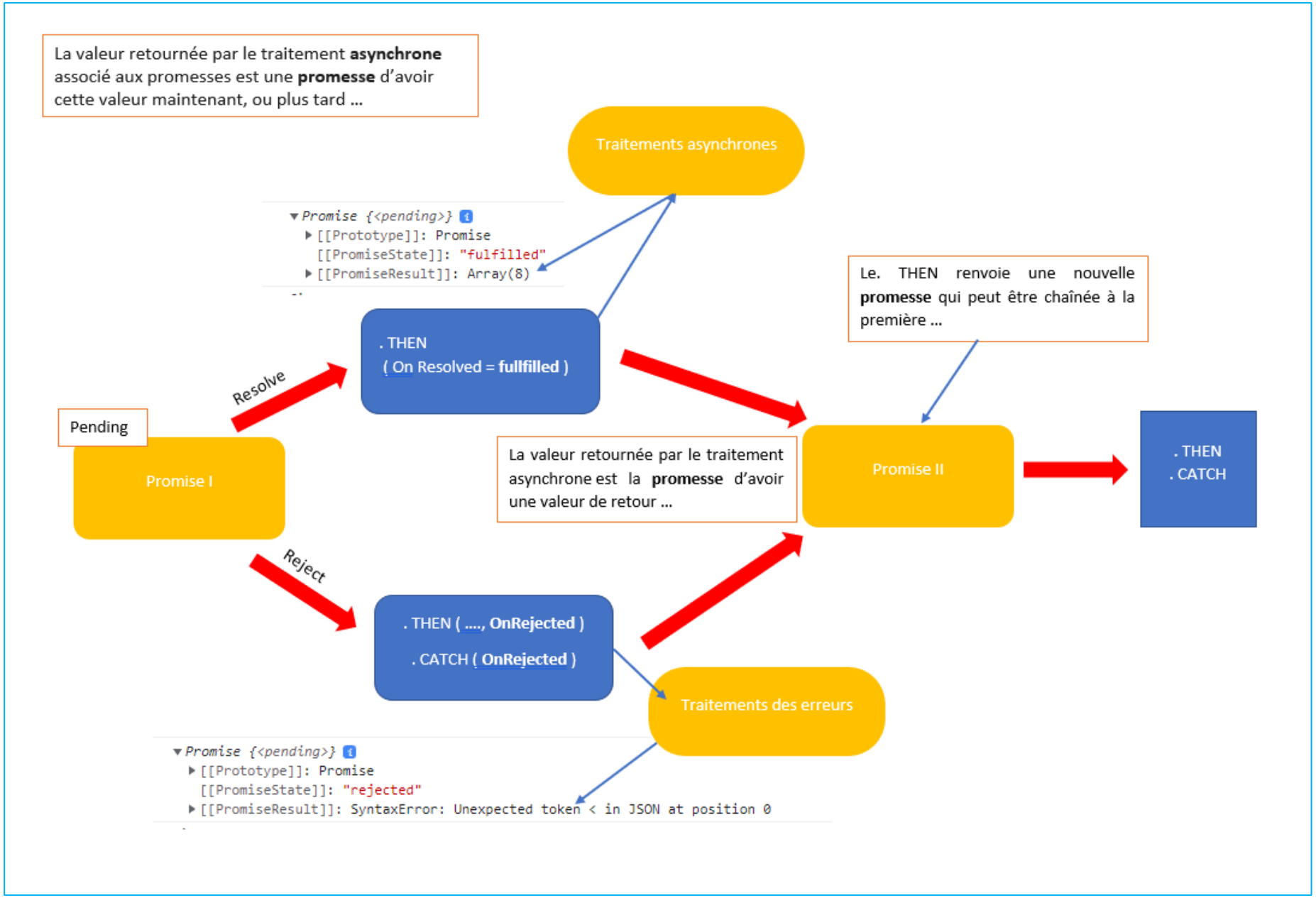
```
1   // nouveautés (fonctions)
2   function direBonjour(nom = "SkyWalker") {
3       // passage d'arguments par défaut
4       // console.log("Bonjour " + nom);
5
6       // string interpolation des variables
7       console.log(`Bonjour ${nom}`);
8       document.querySelector("#resultat").innerHTML = `Bonjour ${nom}`;
9   }
10
11  const direBonjour2 = (prenom, nom) => {
12      console.log(`Bonjour ${prenom} ${nom}`);
13      document.querySelector("#resultat").innerHTML = `Bonjour ${prenom} ${nom}`;
14  }
15
16  direBonjour(); // javascript s'écrit en camelCase majuscule à chaque mot sauf le 1er
17  direBonjour2("Dark", "Vador");
18
```

Ecmascript 2015+



```
58  // ------------------------------------------------------------
59  let personne = {
60      prenom: "Luke",
61      nom: "Skywalker",
62      // tableaux
63      langages: ["JS", "React", "NG", "Vue"],
64      // méthodes
65      nomComplet() {
66          return `${this.prenom} ${this.nom}`;
67      }
68  }
69  console.log(personne.nomComplet());
70
71  // ---- Nouveautés class (PascalCase = maj à chaque mot )
72  class Personne {
73
74      // Constructor
75      constructor(nom, age) {
76          this._nom = nom;
77          this._age = age;
78      }
79      // Méthodes
80      infosPersonne() {
81          return `Bonjour je m'appelle ${this._nom} et j'ai ${this._age} ans.`;
82      }
83  }
84
85  let p1 = new Personne("Emilie",  35);
86  console.log(p1);
```

Le fonctionnement d'une promesse



App.vue × script.js ×

```
// const url='https://dev.webjs.fr/JSON_API/films.json';
const url = 'http://localhost:3001/films1';
// const url = 'http://localhost:3001/filmdssdfdsfs1'; // erreur 404 !

fetch(url, // renvoie une promise
    {
        method: 'get',
        // headers
    }
).then(
    (resHTTP) => {
        console.log(resHTTP);
        resHTTP.json() // json renvoie également une promise
            .then(
                (datas) => {
                    console.log(datas);
                    movies.value = datas; // on passe les adatas à la ref reactive de VUE
                }
            )
    },
    // on rejected
    (e) => {
        console.warn('🙄Erreur du Rejected', e); // arrêter le json-server pour tester
    }
)
// .catch(
//     (e) => {
//         console.warn('😭Erreur globale du CATCH', e); // arrêter le json-server pour tester
//     }
// )
```

Voici un exemple avec « axios » et avec le couple « async await »

App.vue × script.js ×

```
1   // ---------------------------------
2   onMounted(
3
4       async () => {
5           const url='http://localhost:3001/dives';
6           await axios.get(url,{
7           })
8           .then(
9               (resHTTP) => {
10                  console.log('ResHTTP',resHTTP);
11                  console.log(resHTTP.data);
12                  dives=resHTTP.data;
13              }
14          )
15  })
```

Au travers de nos TD et TP, nous prendrons des exemples de cette nouvelle nomenclature EcmaScript 2015+.

Dans chaque partie de la formation, nous aurons l'occasion de mettre en pratique les « nouveautés » EcmaScript. Ce sera l'occasion d'approfondir nos connaissances dans ce domaine.

Prise en main de VUE

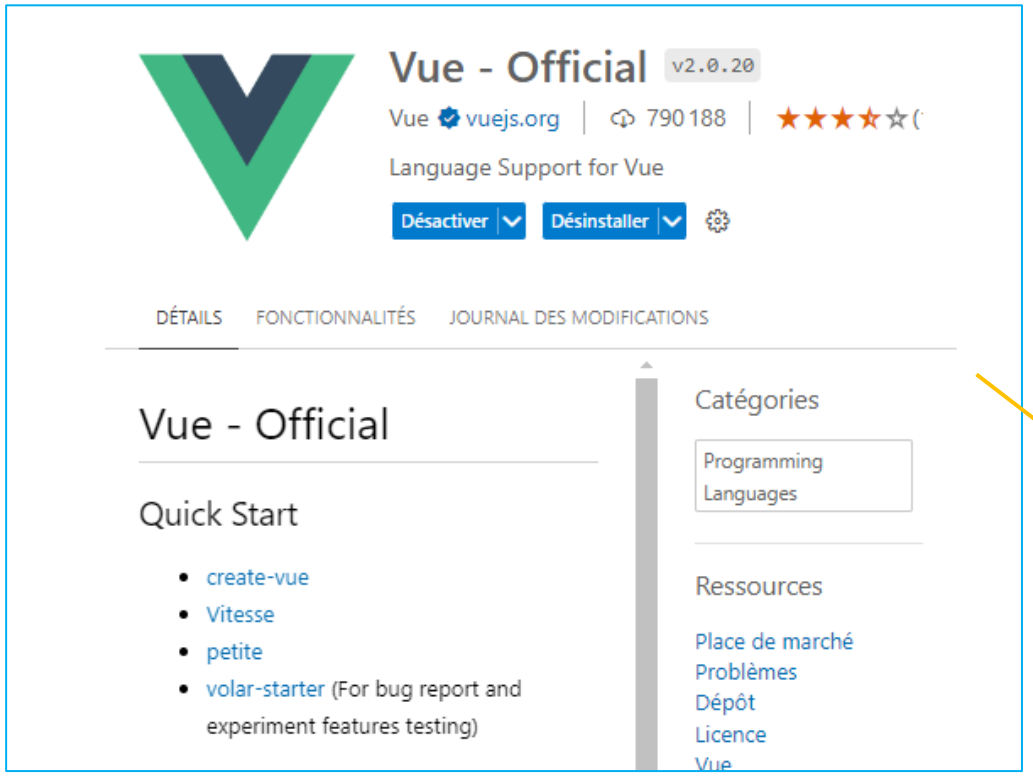
Prise en main de VUE

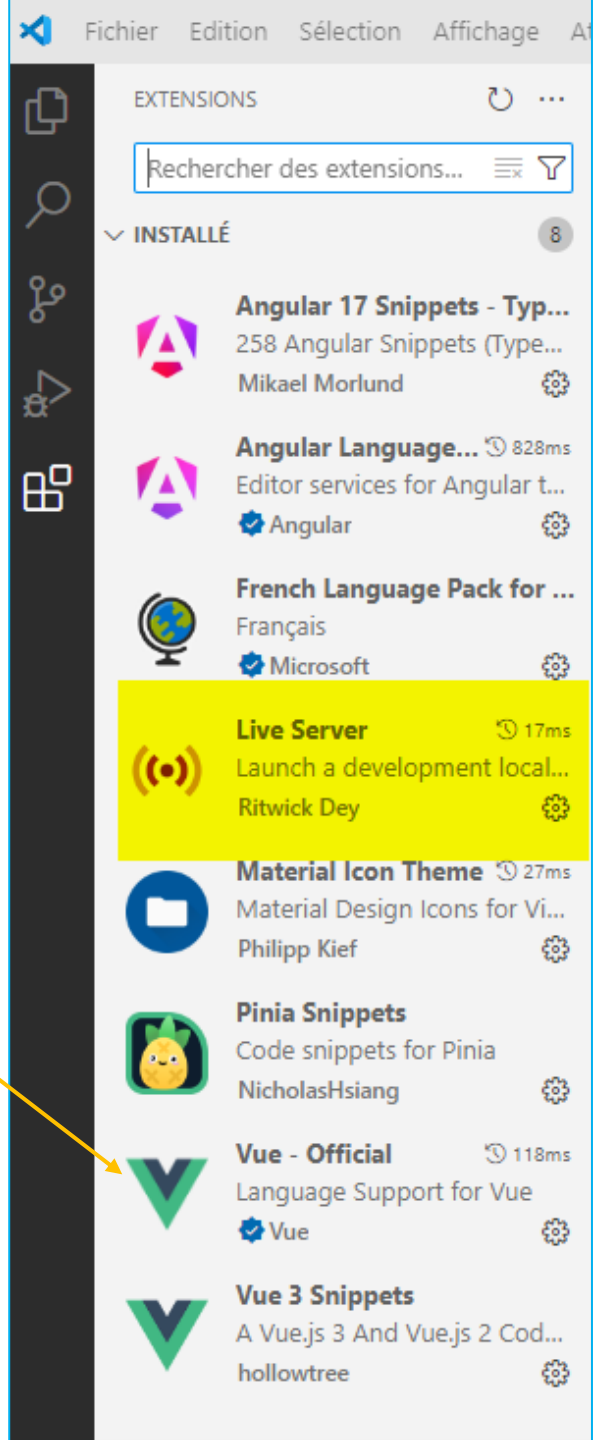
Nous allons utiliser un éditeur de code comme VS Code. Il est intéressant d'ajouter certaines extensions qui vont nous permettre d'avoir de la « completion » de code.

Nous utiliserons également des extensions de navigateur comme « Vue dev.tools »

Il est intéressant de noter qu'il existe des extensions de navigateur pour tous les frameworks.

Outils IDE et extension navigateur





Outils IDE et navigateur

L'extension Vue pour les navigateurs sera très utile

Toutes les extensions

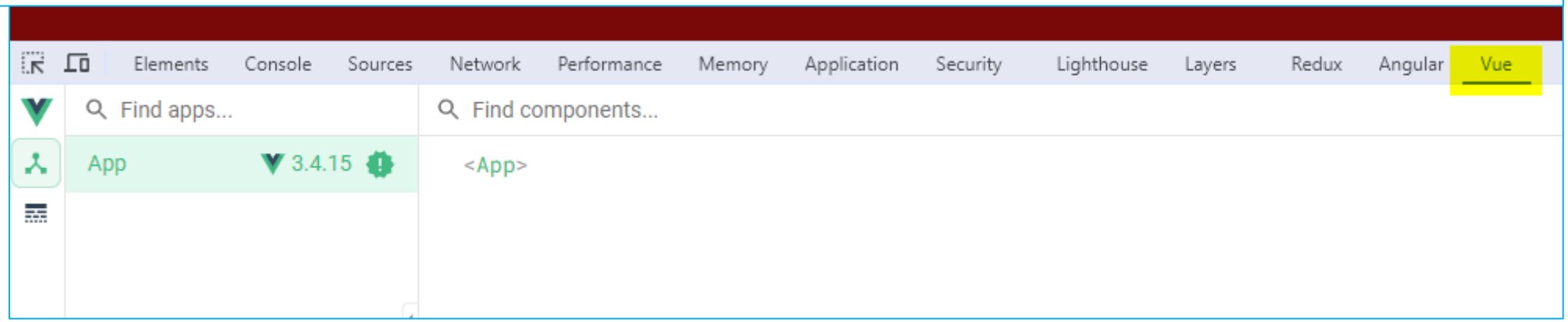
Angular DevTools
Angular DevTools extends Chrome DevTools adding Angular specific debugging and profiling capabilities.
Détails Supprimer

Moesif Origin/CORS Changer & API Logger
Allow cross-domain requests by override Origin and CORS headers. Log/capture XmlHttpRequest API calls for debugging and analytics.
Détails Supprimer

React Developer Tools
Adds React debugging tools to the Chrome Developer Tools. Created from revision c7c68ef842 on 10/15/2024.
Détails Supprimer

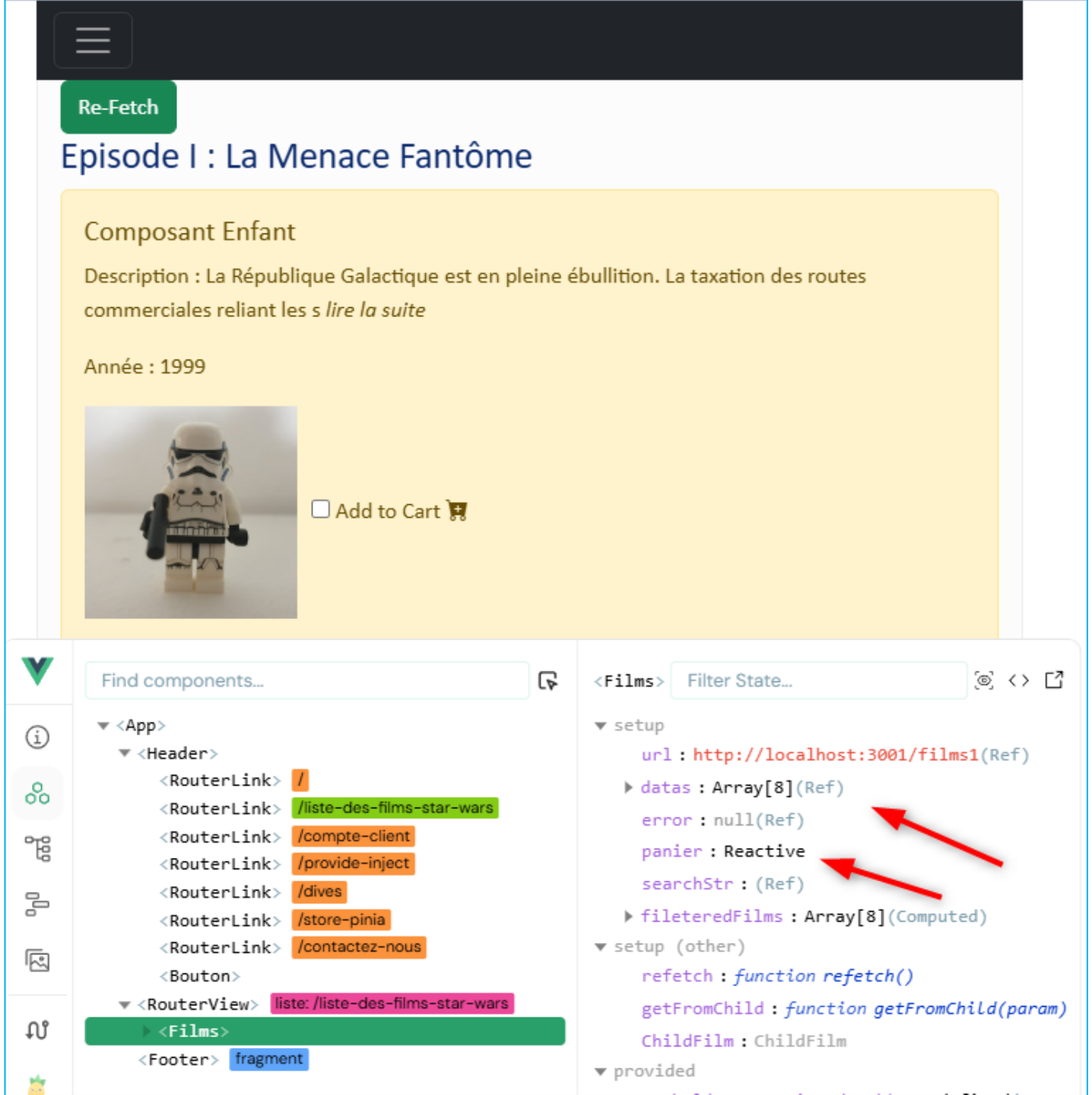
Redux DevTools
Redux DevTools for debugging application's state changes.
Détails Supprimer

Vue.js devtools
DevTools browser extension for Vue.js
Détails Supprimer



Vue dev-tools nous permet de comprendre l’architecture d’un projet:

- L’imbrication des composants
- Le routage
- Le passage des props entre composants
- La gestion du store pinia

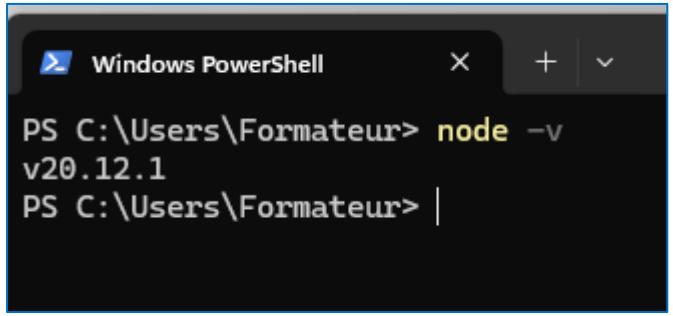


Installation de NODE.JS

- https://nodejs.org/en
- https://www.npmjs.com/
  Le site officiel des repositories NODE

Exemples :

- https://www.npmjs.com/package/vue
- https://www.npmjs.com/package/vite



Vue en tant que librairie …

Vue (https://vuejs.org/) peut s'utiliser en tant que librairie.

Ce n'est pas la meilleure manière de faire évidement comme nous l'avons déjà évoqué en introduction, mais cela est tout à fait possible.

Nous préférerons utiliser Vue avec un environnement de Framework et notamment Vite de « Evan YOU »

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

https://unpkg.com/vue@3/dist/vue.global.js

Vue (https://vuejs.org/) peut s'utiliser en tant que librairie.

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

https://unpkg.com/vue@3/dist/vue.global.js

```
<!DOCTYPE html>
<html lang="en">

<!-- ajouter l'extension live server dans VS Code puis clic droit -->

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"

    <link rel="stylesheet" href="css/styles.css">
    <!-- lien vers le CDN de vue (librairie) -->
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
</head>
```

Prise en main de Vue

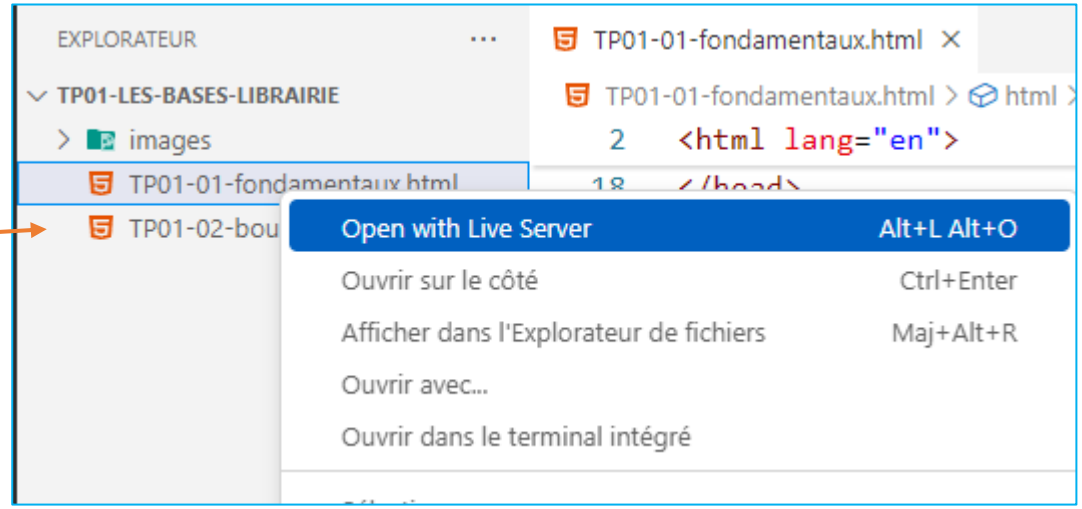
Contrairement à Angular qui sépare le Typescript (le model) du template HTML, Vue et React utilise le même fichier (Typescript, JSX, JS …) pour à la fois manipuler le template HTML et la structure des datas (constructor, properties, methods, lifecycle).

Notre premier exemple va nous permettre de comprendre le « binding » de datas entre le model et le template, nous verrons notamment le « single way binding » :

- du template vers le javascript
- du javascript vers le template

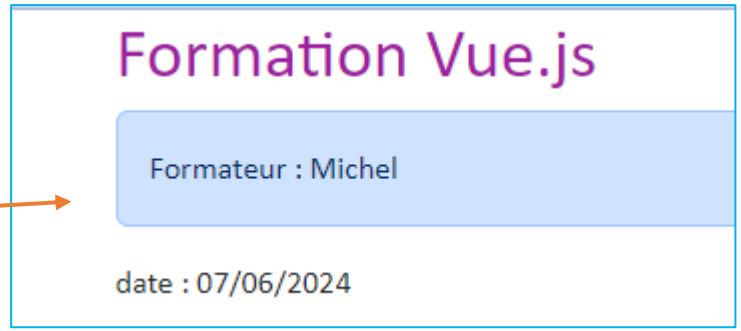
- Et le « double way binding »

- Pour tester notre code, je vous conseille l'extension « live server » de VS Code.
  Une fois installée, au clic droit de la souris, le résultat est actualisé dans le navigateur à chaque modification de code.



Premier exemple

```
<h2 >{{titre}}</h2>
<!-- string interpolation de variable -->
<p class="alert alert-primary">Formateur : {{formateur}}</p>
<p>date : {{date}}</p>
<p>
```




```
<!-- datas controleur -->
<script>
    // 1 - création d'un composant principal App = Application ou le container principal

    // nos composants ou les éléments HTML sont pensés comme des constantes
    const App = Vue.createApp({

        // ----------- LE MODEL DE DONNEES -----------------
        data() {
            // les datas (prop statique ou dynamique) qu'on peut lier à la vue <=> Model de données
            return {
                // raw data
                titre :'Formation Vue.js',
                formateur:'Michel',
                date: new Date().toLocaleDateString(),
```

TP01-01-fondamentaux.html

TP01-01-fondamentaux.html > ...

```
2   <html lang="en">
20  <body>
94      <!-- Single File Component -->
95      <script>
96
97          // le MODEL de données
98          // Création d'un composant principal App = Container principal
99          // les composants sont associés à const
100
101         const App = Vue.createApp({
102
103             data() {
104                 return {
105                     // les props liés à ce composant // states
106                     titre: 'Formation Vue',
107                     version: 3,
108                     date: new Date().toLocaleDateString(),
109                     // camelCase
110                     srcImg: 'images/avatar2.jpg',
111                     srcImg2: 'images/avatar3.jpg',
112                     nom: 'Episode 1',
113                     idNom: 'titre-1',
114                     // rich data
115                     contenuPara: '<span style="color:#3785ce">Bienvenue dans la formation VUE</span>',
116                     // objets
117                     formation: {
118                         name: 'Vue.js',
119                         duration: 3
120                     },
121                     // dble way binding
122                     saisie: '',
123                     reponse: '',
124                     reponse2: '',
125                     position: {
126                         x: 0,
127                         y: 0
128                     },
129                 }
130             },
```

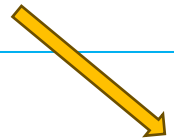
```html
<!-- Template HTML -->

<div id="app-vue">
    <!-- kebab case en HTML et CSS -->
    <h3 class="alert alert-warning">{{titre}} {{version}}</h3>
    <!-- string interpolation ou single way binding -->
    <p>{{date}}</p>
    <p>
        <img v-bind:src="srcImg" style="width:150px">
        <!-- attribut HTML => v-bind = directive modifie un atribut HTML -->
        <!-- single way binding (MODEL => TEMPLATE HTML) -->

        <img :src="srcImg2" style="width:150px">
        <!-- syntaxe simplifiée attribut HTML préfixé par un : syntaxic sugar -->
    </p>

    <p :id="idNom">{{nom}}</p>

    <p>{{contenuPara}}</p>
    <p v-text="contenuPara"></p>
    <!-- équivalent de la prop JS innerText -->
    <p v-html="contenuPara"></p>
    <!-- équivalent de la prop JS innerHTML -->

    <p class="alert alert-primary">{{formation.name}} {{formation.duration}}</p>

    <!-- formulaires -->
    <label>Saisir ...</label>
    <!-- double way binding -->
    <input type="text" class="form-control" v-model="saisie">
    Vous avez saisi : <strong>{{saisie}}</strong>
```

Premier exemple

```
// -------------- LES METHODES ----------
methods: {
    methodAction() {
        console.log("Clic Avec v-on:événement");
        // on utilise this pour cibler l'objet lui-même ( cibler les datas (props))
        this.ouiNon='Oui';
    },
    methodAction2(e) {
        console.log("Clic avec @clic ou @submit");
        console.log(e); // output : Pointer Event
    },
    methodPosition(e) {
        console.log(e.offsetX , e.offsetY);
        this.position.x =  e.offsetX;
        this.position.y =  e.offsetY;

    },
    methodSubmit(e){
        e.preventDefault();
        // méthode JS qui annule l'événement invoqué (le submit du form)
        console.log('Contenu du form à valider et à traiter');
    },
    methodSubmit2(e){
        // plus besoin de prventDefault()
        console.log('Contenu du form à valider et à traiter');
    }
}
```

Les méthodes

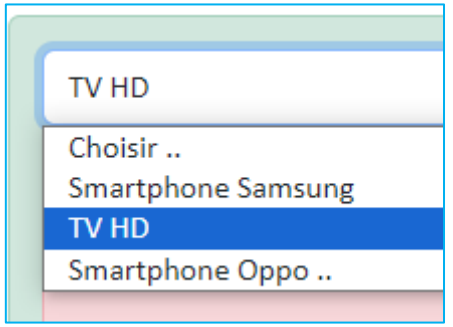
```html
<p>
    <!-- single way binding : Template HTML > datas -->
    <button class="btn btn-primary" v-on:click="methodAction()">Clic</button>
    Vous avez cliqué ? <span>{{ouiNon}}</span>
</p>

<p>
    <!-- syntaxic sugar @event -->
    <button class="btn btn-warning" @click="methodAction2($event)">$event</button>
    <span> L'événement global javascript $event ets très intéressant</span>
</p>
```



```javascript
// -------------- LES METHODES ----------
methods: {
    methodAction() {
        console.log("Clic Avec v-on:événement");
        // on utilise this pour cibler l'objet lui-même ( cibler les datas (props))
        this.ouiNon='Oui';
    },
    methodAction2(e) {
        console.log("Clic avec @clic ou @submit");
        console.log(e); // output : Pointer Event
    },
```

Les structures de boucles v-for

```
<select class="form-control" @change="methodChange($event)">
    <option value="">Choisir ..</option>
    <option v-for="produit in produits" :value="produit">{{produit}}</option>

</select>
```

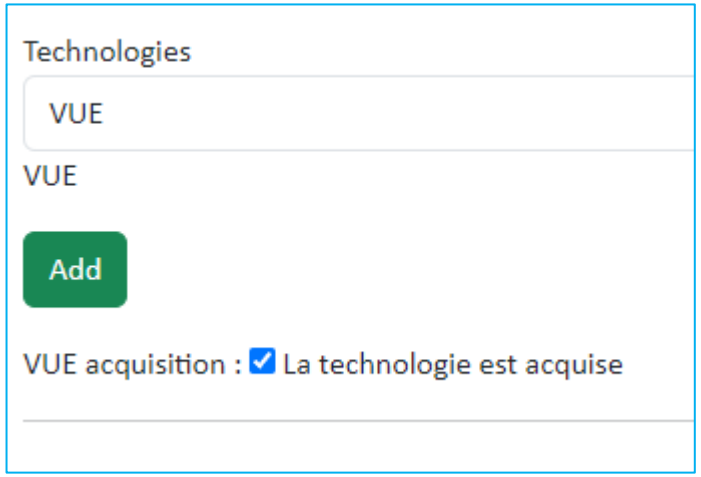



```
methods: {
    methodChange(e) {
        console.warn(e.target.value);
        console.log(this.panier.indexOf(e.target.value)) ; //

        if (this.panier.indexOf(e.target.value) === -1 ) {
        this.panier.push(e.target.value);
        console.log(this.panier);
        }
    }
```

```
<div id="panier" class="alert alert-danger">Panier
    <ul>
        <li v-for="elPanier in panier">{{elPanier}}</li>
    </ul>
</div>
```

Les structures de tests v-if v-else

```
<div v-for="elt in technoArray">
    {{elt.technologie}} acquisition : <input type="checkbox" v-model="elt.acquisition">
    <span v-if="elt.acquisition"> La technologie est acquise</span>
    <span v-else="elt.acquisition"> La technologie n'est pas encore acquise</span>
</div>
```

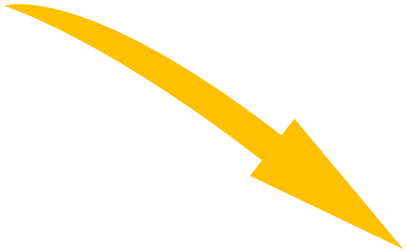


Installation de Vue en tant que framework

https://vuejs.org/guide/quick-start.html

Installation de vue en tant que dépendance globale de Node.
Il existe plusieurs outils pour créer un framework Vue.js *

- npm install -g @vue/cli → vue create projectName 🤔 🙄
- npm create vue@latest 😀
- npm create vite@latest 😉



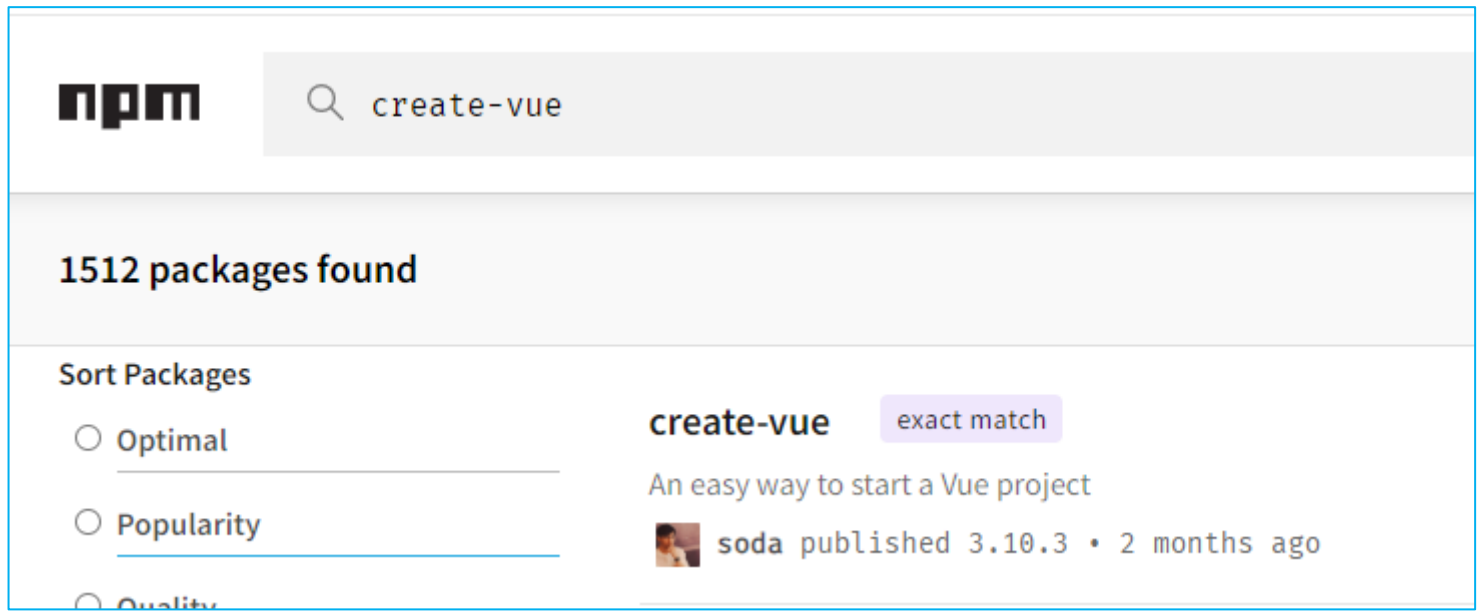
npm | pnpm | yarn | bun

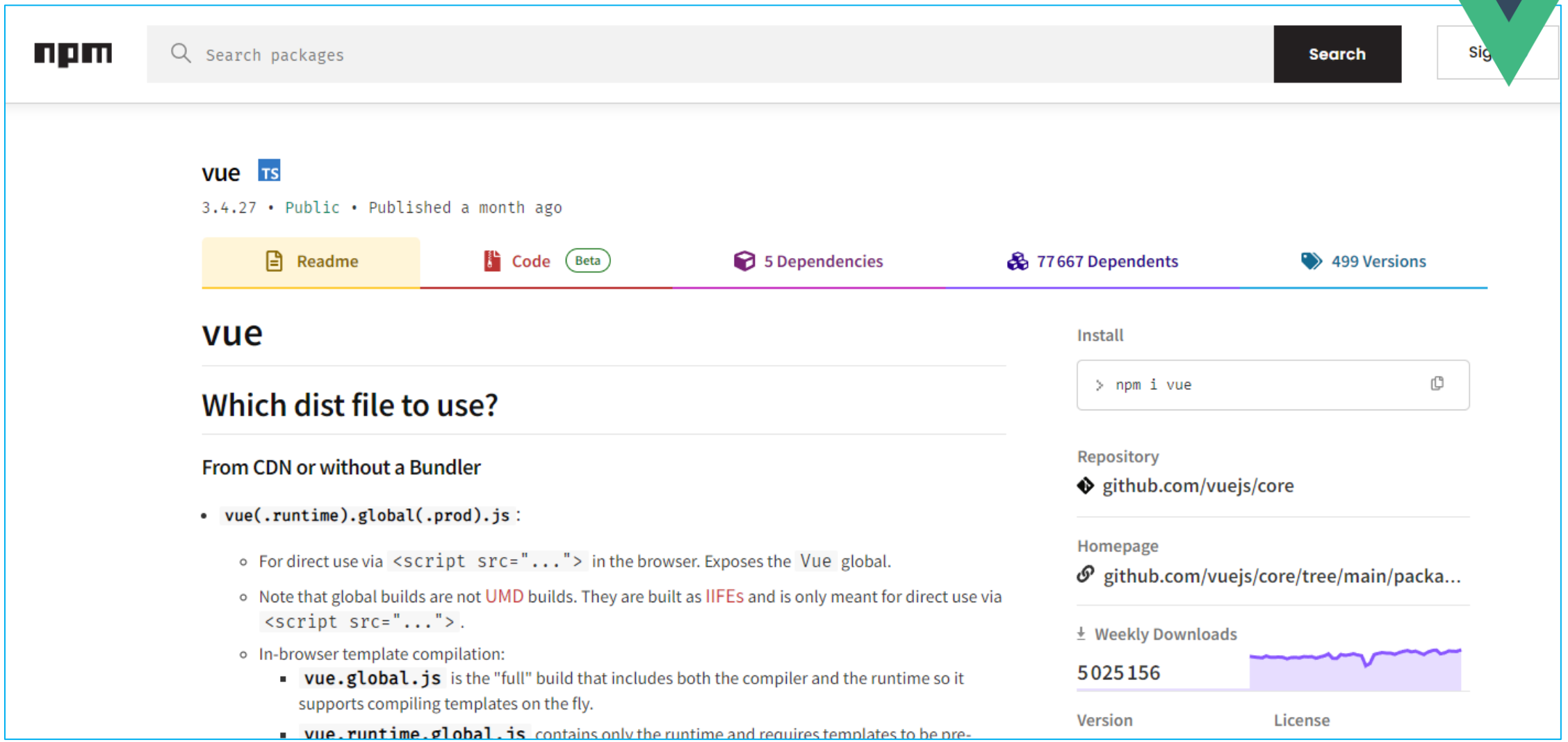
```sh
$ npm create vue@latest
```

This command will install and execute create-vue, the official Vue project scaffolding tool. You will be presented with prompts for several optional features such as TypeScript and testing support:

** Garder toujours une veille technologique sur l'évolution des outils !*

```
PS C:\Users\Michel> npm create vue@latest
Need to install the following packages:
create-vue@3.14.1
Ok to proceed? (y)


> npx
> create-vue


Vue.js - The Progressive JavaScript Framework

✓ Nom du projet : ... formation-vue
✓ Répertoire cible "formation-vue" n'est pas vide. Supprimer les fichiers existants et continuer ? ... Non / Oui
✓ Ajouter TypeScript ? ... Non / Oui
✓ Ajouter le support de JSX ? ... Non / Oui
✓ Ajouter Vue Router pour le développement d'applications _single page_ ? ... Non / Oui
✓ Ajouter Pinia pour la gestion de l'état ? ... Non / Oui
✓ Ajouter Vitest pour les tests unitaires ? ... Non / Oui
✓ Ajouter une solution de test de bout en bout (e2e) ? » Cypress
✓ Ajouter ESLint pour la qualité du code ? » Non

Génération du projet dans C:\Users\Michel\formation-vue...

Terminé. Exécutez maintenant :

  cd formation-vue
  npm install
  npm run dev
```


npm
create-vue
1512 packages found
Sort Packages
Optimal
Popularity
create-vue
exact match
An easy way to start a Vue project
soda published 3.10.3 • 2 months ago


npm
Search packages
Search
vue
TS
3.4.27 • Public • Published a month ago
Readme
Code
Beta
5 Dependencies
77 667 Dependents
499 Versions
vue
Which dist file to use?
From CDN or without a Bundler
vue(.runtime).global(.prod).js :
For direct use via <script src="..."> in the browser. Exposes the Vue global.
Note that global builds are not UMD builds. They are built as IIFEs and is only meant for direct use via <script src="...">.
In-browser template compilation:
vue.global.js is the "full" build that includes both the compiler and the runtime so it supports compiling templates on the fly.
vue.runtime.global.js contains only the runtime and requires templates to be pre-
Install
> npm i vue
Repository
github.com/vuejs/core
Homepage
github.com/vuejs/core/tree/main/packa...
Weekly Downloads
5 025 156
Version
License

https://vitejs.dev/

vite TS

6.0.11 • Public • Published 2 days ago

Readme | Code Beta | 3 Dependencies | 4142 Dependents | 609 Versions

vite

> Next Generation Frontend Tooling

- 💡 Instant Server Start
- ⚡ Lightning Fast HMR
- 🛠️ Rich Features
- 📦 Optimized Build
- 🔩 Universal Plugin Interface
- 🔑 Fully Typed APIs

Vite (French word for "fast", pronounced `/vit/`) is a new breed of frontend build tool that significantly improves the frontend development experience. It consists of two major parts:

- A dev server that serves your source files over native ES modules, with rich built-in features and astonishingly fast Hot Module Replacement (HMR).

Install

```
> npm i vite
```

Repository

github.com/vitejs/vite

Homepage

vite.dev

❤Fund this package

Weekly Downloads

19543911

https://vitejs.dev/

NPM Yarn PNPM Bun

```
$ npm create vite@latest
```

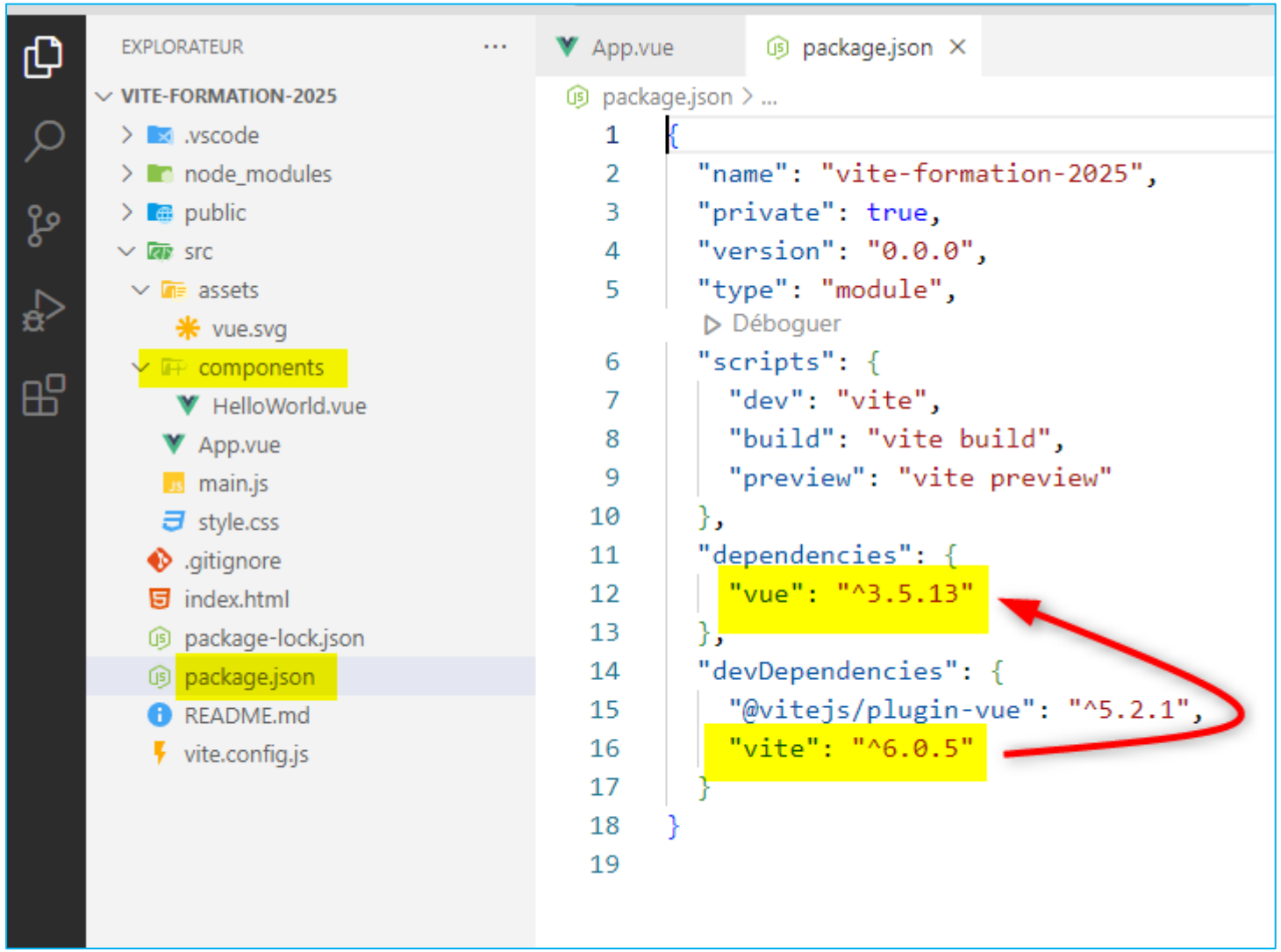
```
PS C:\Users\Formateur\Desktop> npm create vite@latest
Need to install the following packages:
create-vite@6.1.1
Ok to proceed? (y)
```

```
PS C:\Users\Formateur\Desktop> npm create vite@latest

> formateur@1.0.0 npx
> create-vite

√ Project name: ... vite-formation-2025
? Select a framework: » - Use arrow-keys. Return to su
>   Vanilla
    Vue
    React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Angular
    Others
```

```
√ Project name: ... vite-formation-2025
√ Select a framework: » Vue
? Select a variant: » - Use arrow-keys. R
>   TypeScript
    JavaScript
    Customize with create-vue ↗
    Nuxt ↗
```

```
√ Project name: ... vite-formation-2025
√ Select a framework: » Vue
√ Select a variant: » JavaScript

Scaffolding project in C:\Users\Formateu

Done. Now run:

  cd vite-formation-2025
  npm install
  npm run dev
```

Mon 1er Projet Vue en tant que framework



Vite est un environnement de développement, une structure mais in fine c'est bien Vue.js que nous utilisons

Mise en place du TP Fil Rouge

main.js ou main.ts est le point d'entrée (entry point) du projet : son rôle est d'appeler le tout 1er composant App

App.js ne devrait théoriquement pas contenir d'HTML ou de CSS, il est le « composant entry point » qui va structurer la partie principale de l'appli (root) et appelle les sélecteurs principaux des autres composants :

Header, Home ou *RouterView*, Footer

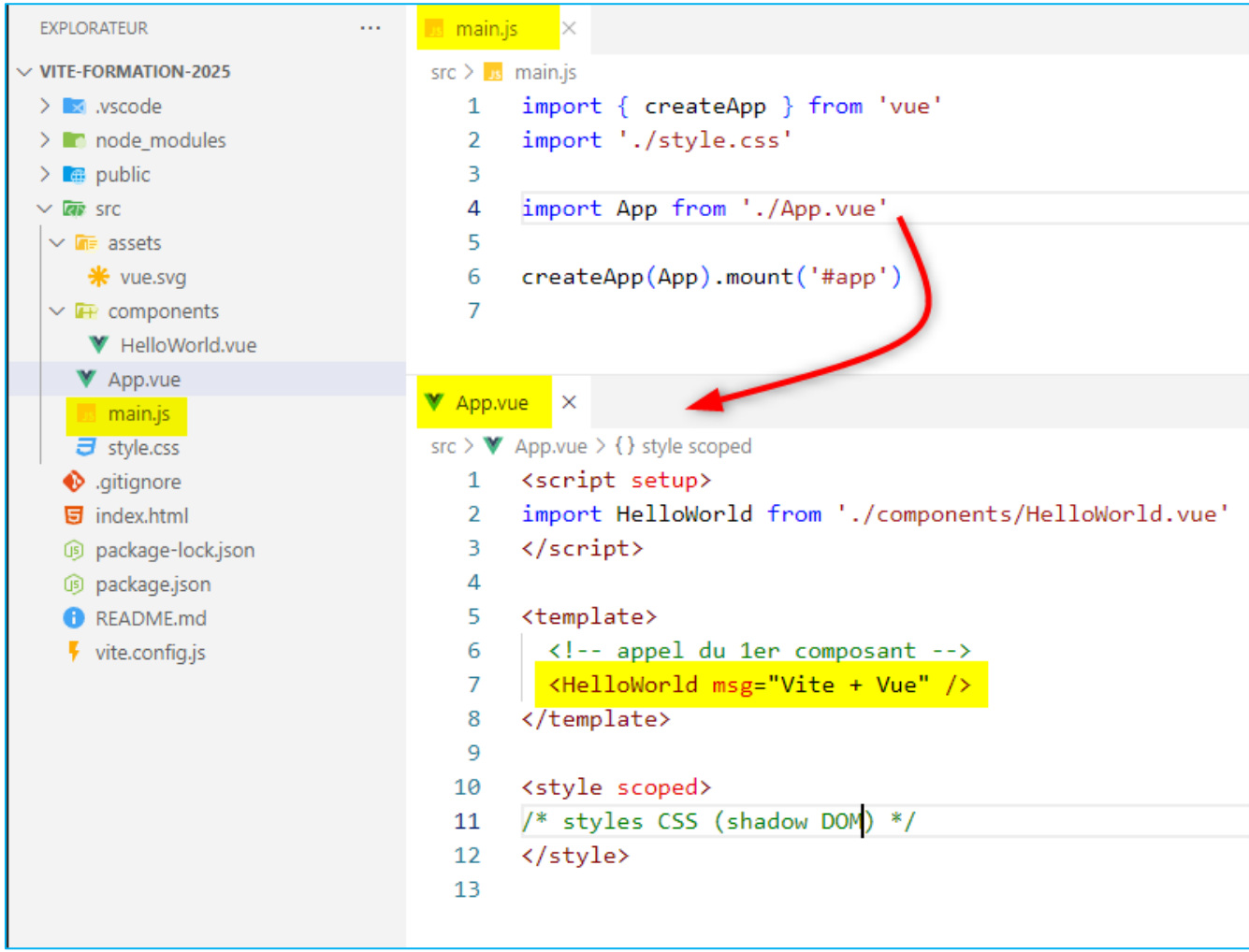
Mon 1er Projet Vue en tant que framework

Main.js est le point d'entrée « entry point » du projet webpack.

Il importe le fichier de styles CSS global à toute l'application
mais surtout son rôle est d'appeler le tout 1er composant App.vue.

App.vue lui-même appelle d'autres composants.

Dès qu'un composant appelle par son sélecteur
un autre composant, il devient un composant parent



Mise en place du TP Fil Rouge

Il est très important de structurer « d'architecturer » son projet !

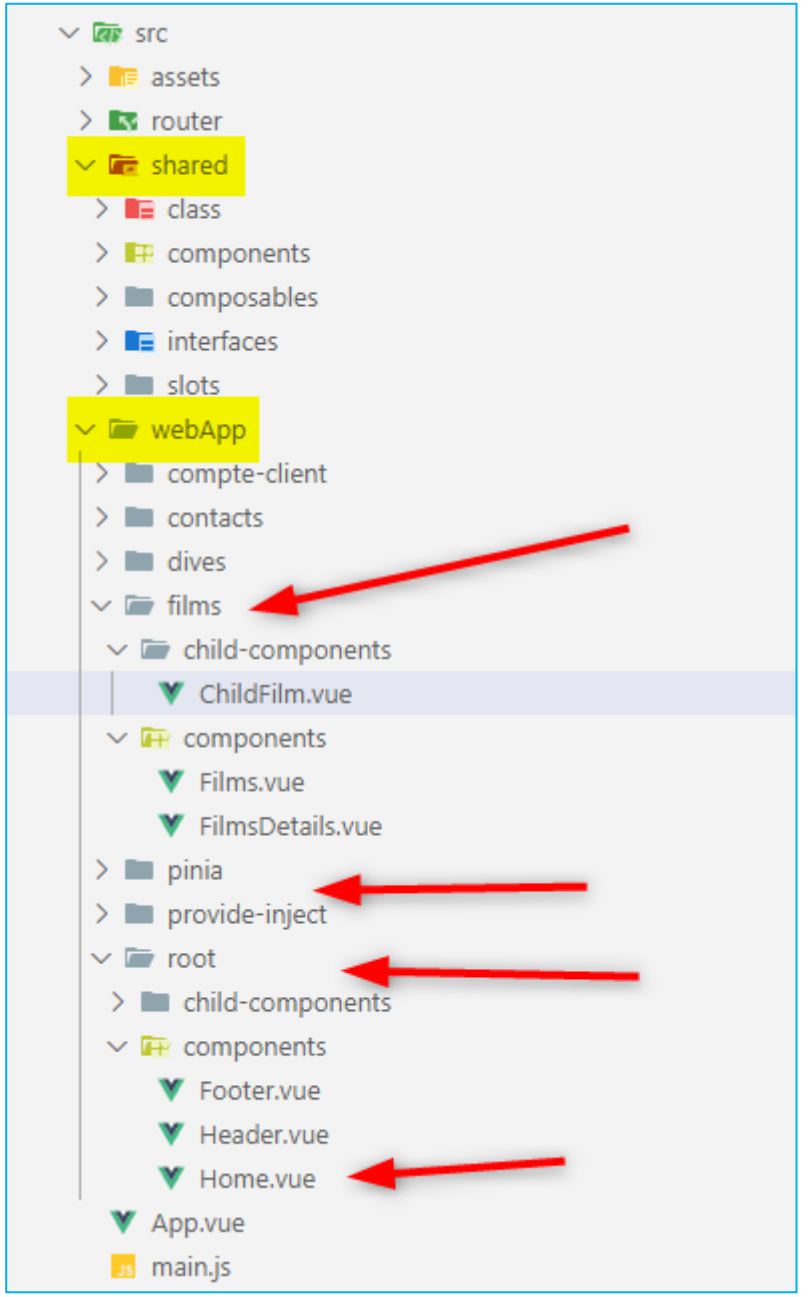
En général, on crée un dossier principal qui contiendra notre application métier.

On peut l'appeler « **core** » « **project** » « **webApp** »

On crée également un dossier « **shared** » « **sharedModules** » « **libs** »

Ce dossier contient les composants redistribuables, les class (typescript), les interfaces(typescript), les slots, les composables, tout ce qui peut être réutilisé dans notre projet.

Dans la diapositive suivante, nous voyons un aperçu de notre application finale, avec tous les menus qui reprendront les grands axes de notre formation.



Liste des Films

Panier

Filtrer l'affichage:

Re-Fetch

Episode I : La Menace Fantôme

Composant Enfant

Description : La République Galactique est en pleine ébullition. La taxation des routes commerciales reliant les s *lire la suite*

Année : 1999

Add to Cart

Plus d'infos ... Ajouter au panier

Episode II : L'Attaque des Clones

Composant Enfant

Description : L'agitation règne au Sénat Galactique. Des milliers de systèmes solaires ont annoncé leur intention *lire la suite*

Année : 2002

Add to Cart

Plus d'infos ... Ajouter au panier

Episode III : La Revanche des Sith

Composant Enfant

Description : C'est la guerre ! La République croule sous les attaques de l'impitoyable Sith, le Comte Dooku. Il y *lire la suite*

Année : 2005

Add to Cart

Plus d'infos ... Ajouter au panier

Le mode de fonctionnement VUE

Les modèles

Le mode de fonctionnement VUE

Nous abordons rapidement un exemple simple du modèle Option API, pour s'assurer de comprendre les projets développés en VUE < 3.

On peut être amené à reprendre un ancien projet Vue et le faire migrer vers une version plus récente. Dans ce cas là , il est important de connaître le modèle OAPI.

Nous utiliserons ensuite uniquement le modèle Composition API CAPI qui est le standard depuis la version 3 de Vue (fin 2020)

Modèle OPTION API : OPI

Ce modèle est présent dans Vue.js jusqu'à vue 2.7 (la dernière version mineure de Vue 2)

Vue 3.0 est sorti le 18 septembre 2020 😏 CAPI Composition API propose une nouvelle manière de faire
Vue 3+ est une version majeure totalement innovante …

(comme React 18 et Angular 19 aussi …) 😳 . Les frameworks évoluent mais nous obligent à repenser la structure du projet et du code..
React 18 s'éloigne des composant à base de class pour privilégier les composants à base de fonctions et de Hooks !

Avantages OPI : structure comme une class 😀

- name
- comp
- props
- data
- computed
- methods
- lifeCycle

Inconvénients

- plus lourd à manipuler que CAPI
- utilisation du this
- difficilement redistribuable
- impossibilité d'importer du code js composable

Modèle OPTION API : OPI

```
export default {
  name:'App',
  components:{
    Header, Footer,
    // CapiComponent, ListeDesFilms
},
  props:[],
  data() {
    return {
      titre :'Formation Vue.js',
      nb:0
    }
  },
  methods: {

  },
  // lifeCycle
  // https://vuejs.org/api/options-lifecycle.html
  beforeCreate () {
    console.log('BeforeCreate');
    // init du composant
    // Data et events ne sont pas appelés
  },

  created() {
    console.log('Created');
    // Accès au datas + events
    // setInterval(() => {   this.nb++  }, 3000 );

},

  beforeMount() {
      console.log('BeforeMount');
      // méthodes du model sont accessibles
      // au niveau du rendu Virtual DOM
      // juste avant le 1er rendu réel
  },
  mounted() {
      console.log('Mounted');
      // le 1er rendu réel depuis le dom virtuel de Vue
      // accéder aucx éléments HTML
  },
```

Plus besoin de nommer le composant

OAPI vs CAPI

ModelOAPI.vue

```vue
<script>
export default {
    name: 'ModelOAPI',
    components: {},
    props: ['msg'],
    data() {
        return {
            titre: 'Formation Vue',
            posX: 0,
            posY: 0
        }
    },
    // lifeCycle
    mounted(){
        window.addEventListener('mousemove', this.updatePosMouse )
    },
    unmountedmounted(){
        window.addEventListener('mousemove', this.updatePosMouse )
    },
    methods: {
        updatePosMouse(e) {   // console.log(e);
            this.posX = e.pageX;
            this.posY = e.pageY;
        }}
}
</script>

<template>
    <h5>Titre : {{ titre }}</h5>
    Position de la souris : {{ msg }}
    <ul>
        <li>pos x : {{ posX }}</li>
        <li>pos y : {{ posY }}</li>
    </ul>
</template>

<style scoped>
Codiumate: Options | Test this class
h5 {
    color: #8f0aaa;
}
</style>
```

ModelCAPI.vue

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

defineProps({ msg: String});

const titre = ref('Formation Vue')
const posX = ref(0);
const posY = ref(0);

// // created n'existe plus
// // on peut directemnt écrire nos fcts de datas
onMounted(
    () => window.addEventListener('mousemove', updatePosMouse)
)

onUnmounted(
    () => window.addEventListener('mousemove', updatePosMouse)
)
Codiumate: Options | Test this function
function updatePosMouse(e) {//   console.log(e);
    posX.value = e.pageX;
    posY.value = e.pageY;
}
</script>


<template>
    <h5> Titre : {{ titre }} </h5>
    Position de la souris : {{ msg }}
    <ul>
        <li>pos x : {{ posX }}</li>
        <li>pos y : {{ posY }}</li>
    </ul>
</template>

<style scoped>
Codiumate: Options | Test this class
h5 {
    color: #27cc18;
}
</style>
```

Modèle COMPOSITION API : CAPI

Vue 3.0 est sorti le 18 septembre 2020 😋 CAPI Composition API propose une nouvelle manière de faire

```
<h5>Composition API Component</h5>
<capi-component msg="Cool 😀"></capi-component>
```

CapiComponent.vue

TP04-vue-vite-TP-fil-rouge > src > components > CapiComponent.vue > { } script setup

```
1   <script setup>
2
3   defineProps({
4     msg: String
5   });
6
```

Setup : https://fr.vuejs.org/api/sfc-script-setup.html

est un sucre syntaxique de compilation pour l'utilisation de la Composition API dans les composants monofichiers (SFC).
C'est la syntaxe recommandée si vous utilisez à la fois les SFC et la Composition API. Elle offre un certain nombre d'avantages par rapport à la syntaxe classique <script> :

- Un code plus succinct avec moins de répétitions
- Possibilité de déclarer des props et des événements émis en utilisant du Type Script pur.
- Meilleures performances d'exécution (le Template est compilé dans une fonction de rendu dans la même portée, sans proxy intermédiaire).
- Le script est pré-traité et utilisé comme fonction setup() du composant, ce qui signifie qu'il sera exécuté pour chaque instance du composant.

Modèle COMPOSITION API : CAPI

Ref, OnMounted, ...

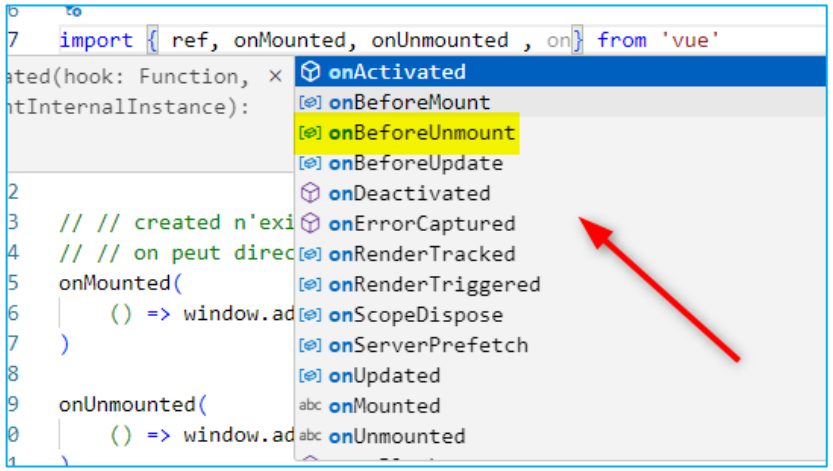
```vue
<script setup>

defineProps({
  msg: String
});

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

const nom = ref('Formation Vue')
const posX = ref(0);
const posY = ref(0);

// // created n'existe plus
// // on peut directemnt écrire nos fcts de datas
onMounted(
    () => window.addEventListener('mousemove', updatePosMouse)
)

onUnmounted(
    () => window.addEventListener('mousemove', updatePosMouse)
)
Codiumate: Options | Test this function
function updatePosMouse (e) {
    console.log(e);
    posX.value = e.pageX;
    posY.value = e.pageY;
}
```

fct perso

ref()

Takes an inner value and returns a reactive and mutable ref object, which has a single property `.value` that points to the inner value.

https://vuejs.org/api/reactivity-core.html#ref

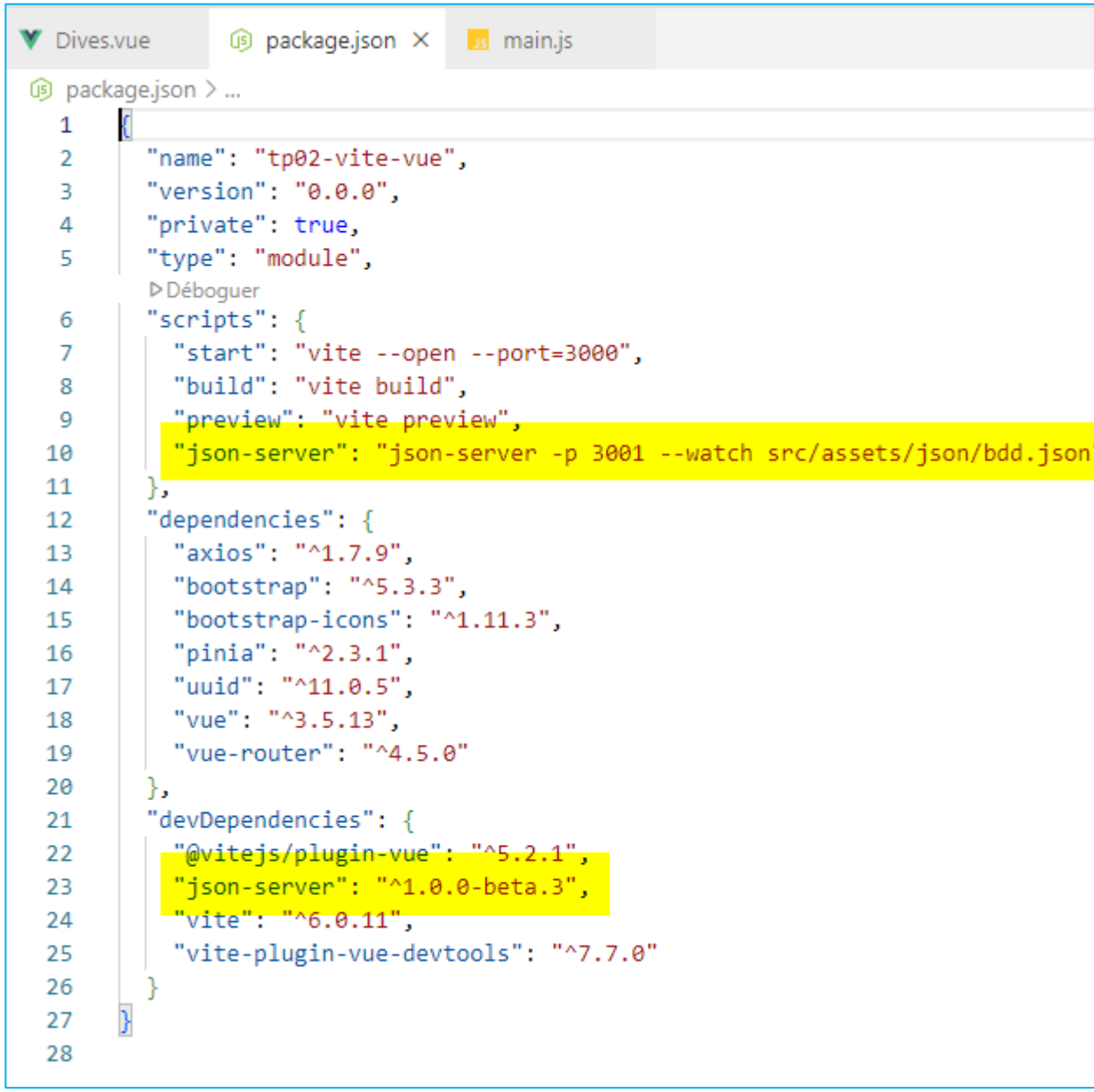


La gestion des datas

La gestion des datas

Nous aurons besoin de datas pour nos requêtes, nous pouvons utiliser le serveur de dév que je mets à disposition : https://dev.webjs.fr/JSON_API/

Nous pouvons également utiliser une librairie à installer dans les devDependencies du « package.json »
Npm install -D json-server



Le composants Datas.vue

Utilisation de l’API FETCH

App.vue

```vue
<script setup lang="js">
import { reactive, ref } from 'vue';

// props qui viennent du parent
defineProps({})
// (datas du composant lui même)
let movies = ref(null); // ref null

// methods
// const url='https://dev.webjs.fr/JSON_API/films.json';
const url = 'http://localhost:3001/films1';

fetch(url, // renvoie une promise
    {
        method: 'get',
        // headers
    }
).then(
    (resHTTP) => {
        console.log(resHTTP);
        resHTTP.json() // json renvoie également une promise
            .then(
                (datas) => {
                    console.log(datas);
                    movies.value = datas; // on passe les adatas à la ref reactive de VUE
                }
            )
    },
    // on rejected
    (e) => {
        console.warn('🙄Erreur du Rejected', e); // arrêter le json-server pour tester
    }
)
</script>
```

Le composants Datas.vue

```vue
<template>
    <div class="alert alert-warning">

        <h3>Composant - Datas</h3>

        <!-- {{ movies }} -->

        <table class="table">
            <tr>
                <th scope="col">Titre</th>
                <th scope="col">Description</th>
                <th scope="col">Année</th>
            </tr>
            <tr v-for="movie in movies">
                <td>{{ movie.title }}</td>
                <td>{{ movie.desc.substring(0, 100) }} ...</td>
                <td>{{ movie.year }}</td>
            </tr>
        </table>

    </div>

</template>
```

CAPI : La notion de Composable

https://vuejs.org/guide/reusability/composables

CAPI introduit la notion de **Composable**

Ce sont des fonctions qui « factorisent » le code et peuvent être réutilisés par différents composants, voire plusieurs fois dans le même composant.

Par convention les composables sont toujours préfixées par use:

- useMouse
- useFetch

NB : cela ressemble beaucoup aux fonctions « Hooks » de React 😉

CAPI : La notion de Composable

https://vuejs.org/guide/reusability/composables

ModelCAPI.vue

TP04-OPI-CAPI > src > components > ModelCAPI.vue > { } script setup

```vue
<script setup>
import { ref, onMounted, onUnmounted  } from 'vue'
import { useMouse } from '@/functions/mouse';

defineProps({ msg: String});

const { posX, posY } = useMouse()
</script>


<template>
    <h5> Titre : {{ titre }} </h5>
    Position de la souris : {{ msg }}
    <ul>
        <li>pos x : {{ posX }}</li>
        <li>pos y : {{ posY }}</li>
    </ul>
</template>

<style scoped>
Codiumate: Options | Test this class
h5 {
    color: #27cc18;
}
</style>
```

mouse.js

TP04-OPI-CAPI > src > functions > mouse.js > useMouse > update

```js
import { ref, onMounted, onUnmounted } from "vue";

// Par convention , les fonctions composables sont préfixées par use...
Codiumate: Options | Test this function
export function useMouse() {
  // 2 varaibales d'états contrôlées par le comosable
  const posX = ref(0);
  const posY = ref(0);

  // MAJ de l'état du comosable
  function update(event) {
    posX.value = event.pageX;
    posY.value = event.pageY;
  }

  onMounted(() => window.addEventListener("mousemove", update));
  onUnmounted(() => window.removeEventListener("mousemove", update));

  return { posX, posY };
}

```

TP1 : Créer une fonction composable Event.js

Fonction nommée **event.js** qui factorisera le code des addEventListener de la fonction mouse.js

La fonction useMouse.js importera elle-même la fonction composable **event.js**

Solution

ModelCAPI.vue

TP04-OPI-CAPI > src > components > ModelCAPI.vue > { } template

```vue
<script setup>
import { ref, onMounted, onUnmounted  } from 'v
import { useMouse } from '@/functions/mouse';

defineProps({ msg: String});

const { posX, posY } = useMouse()
</script>


<template>
    <h5> Titre : {{ titre }} </h5>
    Position de la souris : {{ msg }}
    <ul>
        <li>pos x : {{ posX }}</li>
        <li>pos y : {{ posY }}</li>
    </ul>
</template>

<style scoped>
Codiumate: Options | Test this class
h5 {
    color: #27cc18;
}
</style>
```

mouse.js

TP04-OPI-CAPI > src > functions > mouse.js > useMouse > useEventListener('mousemove') callback

```js
import { ref, onMounted, onUnmounted } from "vue";
import { useEventListener } from "./event";

// Par convention , les fonctions comosables sont préfixées par use...
Codiumate: Options | Test this function
export function useMouse() {
  // 2 varaibales d'états contrôlées par le comosable
  const posX = ref(0);
  const posY = ref(0);

  // MAJ de l'état du comosable
//   function update(event) {
//     posX.value = event.pageX;
//     posY.value = event.pageY;
//   }

//   onMounted(() => window.addEventListener("mousemove", update));
//   onUnmounted(() => window.removeEventListener("mousemove", update));

useEventListener(window, 'mousemove', (event) => {
    posX.value = event.pageX;
    posY.value = event.pageY;
})

  return { posX, posY };
```

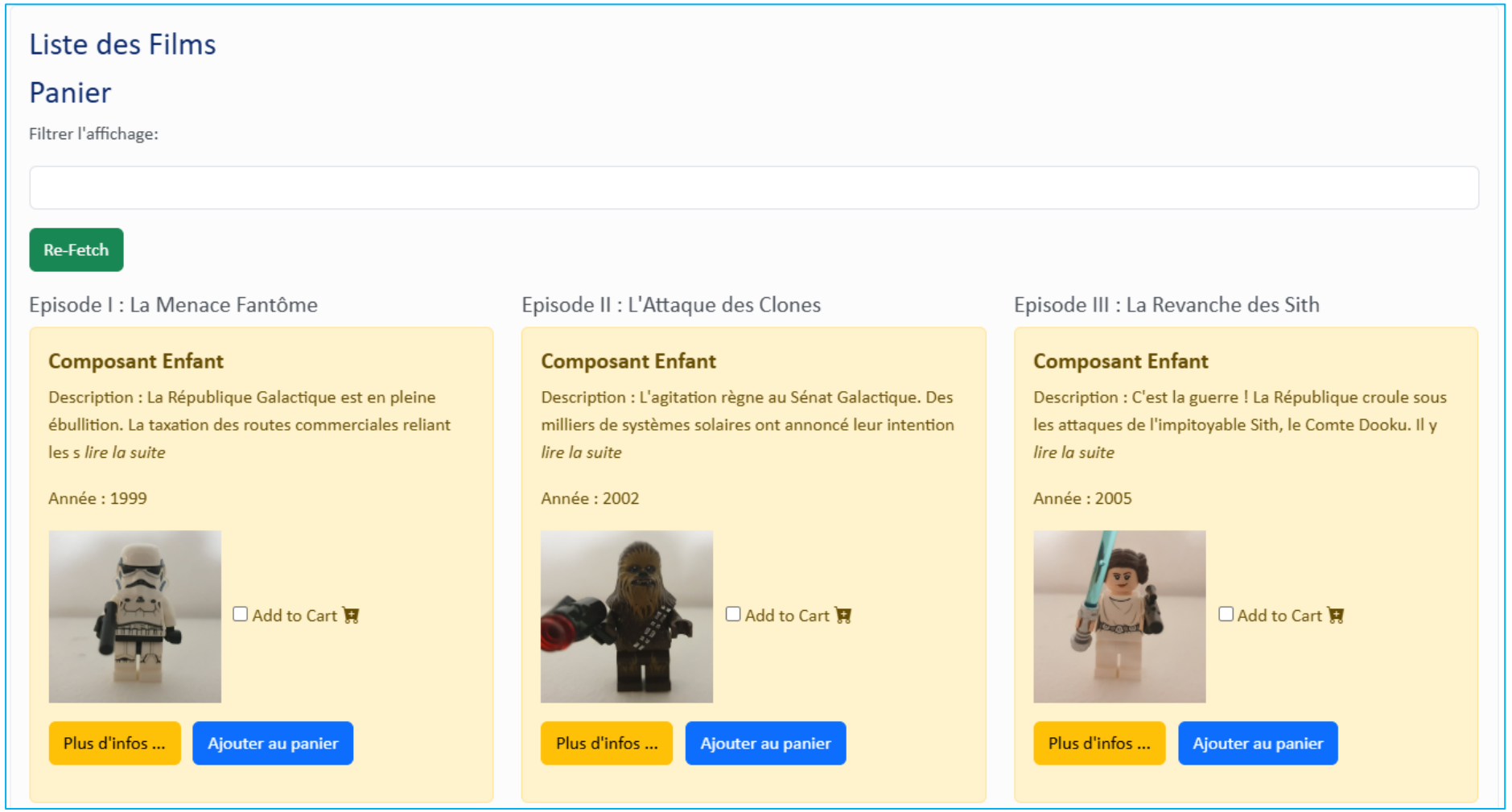
event.js

TP04-OPI-CAPI > src > functions > event.js > ...

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

Codiumate: Options | Test this function
export function useEventListener(target, event, callback) {
  onMounted(() => target.addEventListener(event, callback))
  onUnmounted(() => target.removeEventListener(event, callback))
}
```

TP2 : Créer une fonction composable fetch.js

Fonction nommée **fetch.js** qui factorisera le code du fetch lors des appels HTTP

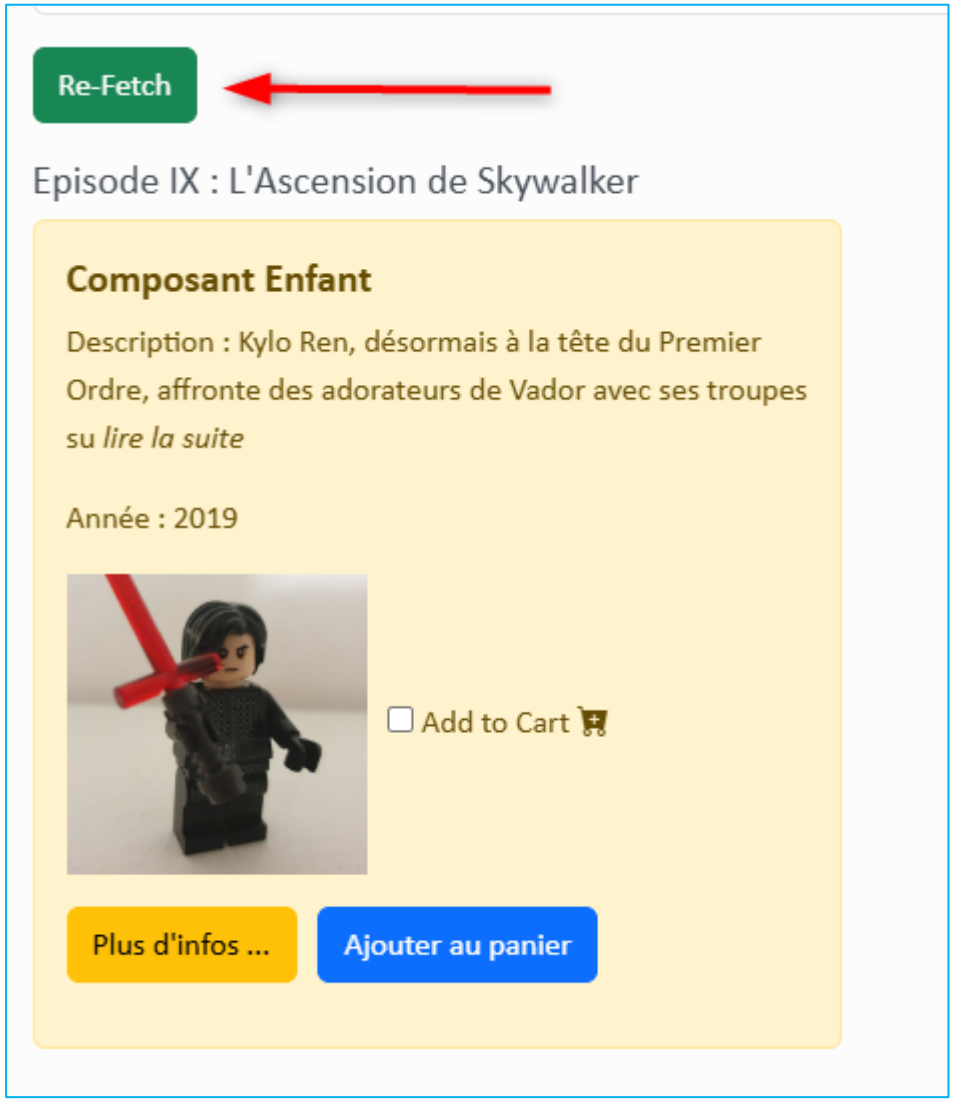


Solution :

```vue
<script setup>
import { onMounted, ref, reactive, computed } from 'vue';
import { useFetch } from '../composables/Fetch'


defineProps({
})

const { datas, error } = useFetch('http://localhost:3001/films1', 'get');

console.log('Dans le composant : ', datas);


</script>

<template>

    <h3>Composant #2 Datas</h3>
    <!-- TP : afficher tous les titres, et plus -->

    <div v-for="movie in datas">
        Titre : {{ movie.title }}
    </div>
```

```js
import { ref } from "vue";

export const useFetch = (_url, _method) => {
  const _headers = new Headers();
  _headers.append("Content-Type", "text/json");

  const datas = ref(null);
  const error = ref(null);

  // ------------------------------
  fetch(_url, {
    method: _method,
    headers: _headers,
  })
    .then((resHTPP) => {
      console.log("Body res HTTP : ", resHTPP);
      resHTPP.json().then((datasJSON) => {
        console.log("Dans le composable : ", datasJSON);

        //   ref de datas
        datas.value = datasJSON;
      });
    })
    .catch((err) => (error.value = err));
  //   ------------------
  return { datas, error };
};
```

TP3 : Le composable fetch.js doit devenir réactive si un de ses paramètres change « _url »



Nous ajoutons un bouton dans le Template qui modifie l'_url passée en entrée du composable useFetch./

Comment rendre réactive notre fonction de composable ?

TP3 : Le composable fetch.js doit devenir réactive si un de ses paramètres change « _url »

Aide : Nous aurons besoin des « Watchers » qui permettent d'observer manuellement une valeur réactive pour déclencher un comportement spécial lors de l'update de cette valeur.

La fonction watchEffect() est observe automatiquement la source d'une valeur et se « *triggera* » automatiquement lorsque cette valeur s'update 😀

🚩 https://vuejs.org/guide/essentials/watchers.html
- watchEffect
- toValue

Vue 3.3+

Solution

Films.vue

src > webApp > films > components > Films.vue > { } script setup

```vue
<script setup lang="js">
import { useFetch } from '@/shared/composables/Fetch';
import { reactive, ref } from 'vue';
import ChildFilm from '../child-components/ChildFilm.vue';
import { useFilter } from '@/shared/composables/Filter';

defineProps({})

// methods
// const url='https://dev.webjs.fr/JSON_API/films.json';
const url = ref('http://localhost:3001/films1');

const { datas, error } = useFetch(url, 'get');
// console.warn('😀',datas);

const refetch = () => {
    url.value = 'http://localhost:3001/films2';
}
console.warn(url);

const panier = reactive([]);
const getFromChild = (param) => {
    console.log(param);
    if (param.paramChecked) { panier.push(param.paramMovie); }
    else {
        let keyPanier = panier.indexOf(param.paramMovie);

        if (keyPanier >= 0) { panier.splice(keyPanier, 1); }
    }
}

// -------------------------
const {searchStr, filteredElements:fileteredFilms} = useFilter(datas

</script>

<template>
    <div class="alert alert-light">

        <h3>Liste des Films</h3>
        <h3>Panier</h3>
        <div v-for="item in panier" :key="item.id" :id="item.id" cla
```

Fetch.js

src > shared > composables > Fetch.js > useFetch

```vue
import { ref, toValue, watchEffect } from "vue";

export const useFetch = (_url, _method, _body=null) => {

    const _headers = new Headers();
    _headers.append("Content-Type", "text/json");

    const datas = ref(null);
    const error = ref(null);

    // console.warn(_url) // films1
    // -------------------------
    const fetchDatas = async () => {

        await fetch(
            toValue(_url),
            // _url.value,
            {
                method: _method,
                headers:_headers,
                body:_body
            }
        ).then(
            (resHTTP) => {
                // console.log('Dans le Composable', resHTTP);
                resHTTP.json() // json renvoie également une promise
                    .then(
                        (datasJSON) => {
                            // console.log('😊', datasJSON);
                            datas.value = datasJSON; // on passe les adatas à la ref rea
                        }
                    )
            })
            .catch(
                (e) => error.value = e
            )
    }
    // ----- Ecoute reactive du fetch ou de la fct grâce à watchEffect
    // rend la fonction réactive => si un des params change(_url), le fct est réexcutée
    watchEffect(
        () => { fetchDatas() }
    )
```

Les Reactives States dans les composables : watchEffect, toValue (*Vue 3.3+)*

ModelCAPI.vue

TP05-OPI-CAPI > src > components > ModelCAPI.vue > { } template > p

```vue
import { ref, onMounted, onUnmounted } from 'vue'
import { useMouse } from '@/composables/mouse';
Codiumate: Options | Test this function
import { useFetch } from '@/composables/fetch';

const url = ref('http://localhost:3000/films1')
const { datas, error } = useFetch(url, 'get')
url.value = 'http://localhost:3000/films2'// trigger a re-fetch

console.log(datas);
defineProps({ msg: String });

const refetch = () => { url.value = 'http://localhost:3000/films1' }

const { posX, posY } = useMouse()
</script>


<template>
    <h5> Titre : {{ titre }} </h5>
    Position de la souris : {{ msg }}
    <ul>
        <li>pos x : {{ posX }}</li>
        <li>pos y : {{ posY }}</li>
    </ul>
    <button class="btn btn-primary" @click="refetch()">Re-Fetch</button>
    <p></p>
    <div v-for="data in datas">
        <p class="alert alert-primary">Titre du film{{ data.title }}</p>

    </div>


</template>
```

fetch.js

TP05-OPI-CAPI > src > composables > fetch.js > ...

```vue
import { ref, watchEffect, toValue } from "vue";

Codiumate: Options | Test this function
export const useFetch = (_url, _method) => {

  const _headers = new Headers();
  _headers.append("Content-Type", "text/json");

  const datas = ref(null);
  const error = ref(null);

  const fetchData = () => {
    datas.value = null
    error.value = null

    fetch(toValue(_url), {
      method: _method,
      headers: _headers,
    })
      .then((responseHTTP) => {
        responseHTTP
          .json()

          .then((datasJSON) => {
            console.log("Dans la fonction :", datasJSON);
            datas.value = datasJSON;
          });
      })
      .catch((err) => (error.value = err));
  }
  watchEffect( () => fetchData());
  return { datas, error };
};
```

useFetch() prend en entrée une URL statique. La Query HTTP n’est effectuée qu’un seule fois.

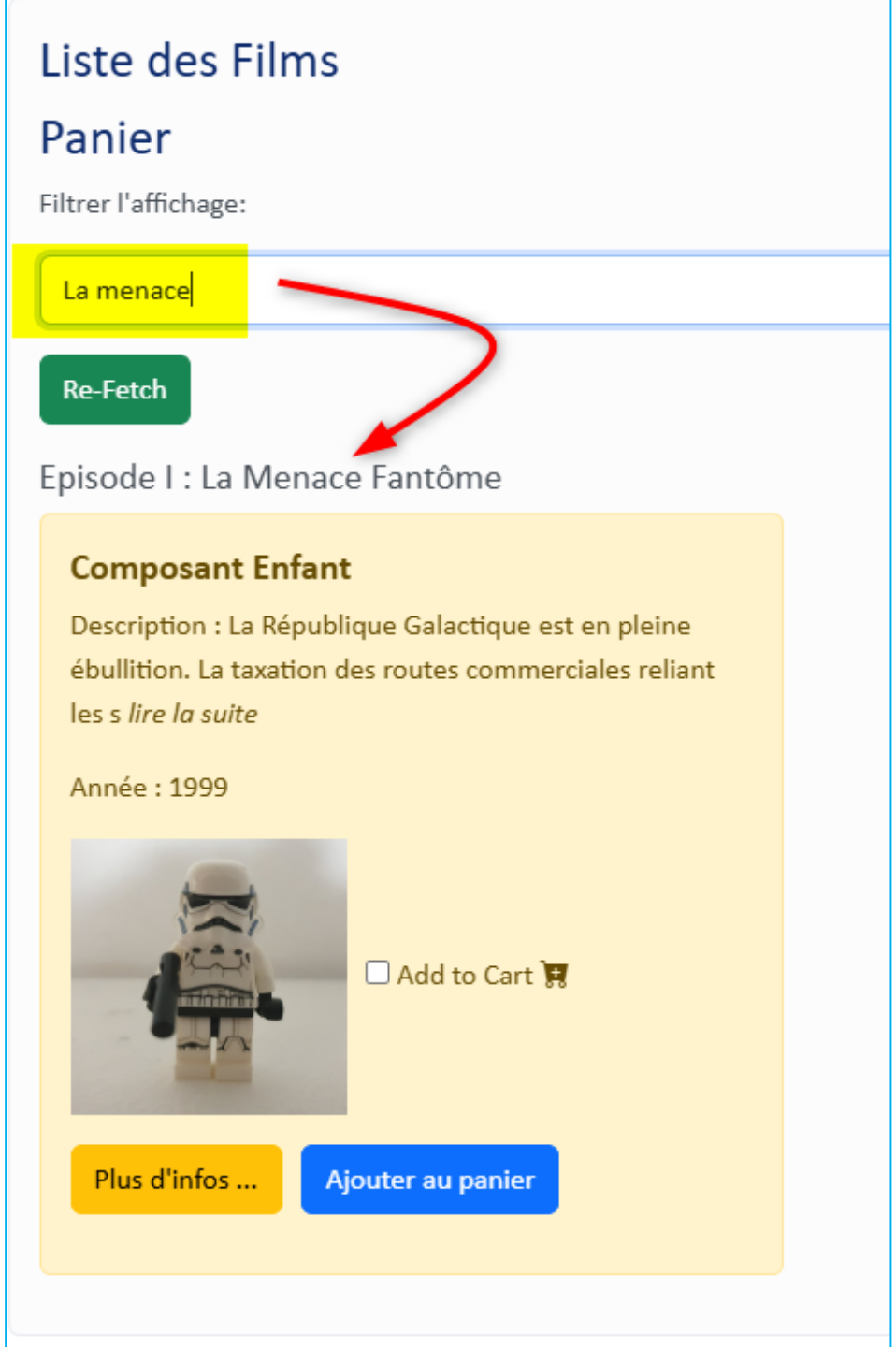
PB : Nous voulons qu'elle récupère à nouveau l'URL à chaque fois qu'elle change.

Pour cela, il est nécessaire de passer un « state réactif » à la fonction composable.
Ce composable deviendra un « observer » et mettra à jour les réponses HTTP.

Le concept est totalement réactif 😀 https://vuejs.org/guide/reusability/composables

TP4 : Créer un composable « useFilter » qui modifie de manière « réactive » le Template HTML

Nous rajoutons un input de saisie dans le Template.
Lors de la saisie de caractères, le Template HTML est filtré dynamiquement par le titre des films.



TP4 : Solution

```vue
const url = ref('http://localhost:3001/films1');

const { datas, error } = useFetch(url, 'get');

// -------------------------
const {searchStr, filteredElements:fileteredFilms} = useFilter(datas)

</script>

<template>
    <div class="alert alert-light">

        <!-- TP useFilter -->
        <div>
            <label>Filtrer l'affichage:</label>
            <p></p>
            <input class="form-control" type="text" v-model="searchStr" />

        </div>

        <div class="row">

            <div v-for="movie in fileteredFilms" :key="movie.id" :id="movie.id" class="col-4">
```

Filter.js

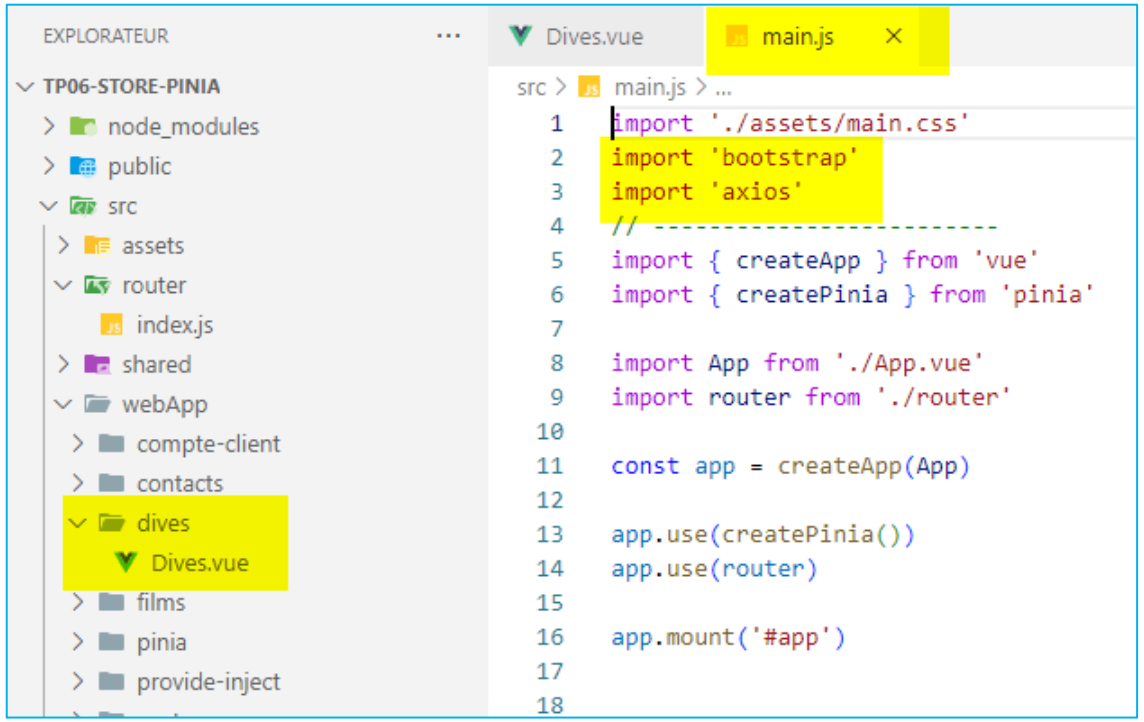
formation-vue-sopra-02-2025 > TP06-store-pinia > src > shared > composables > Filter.js > ...

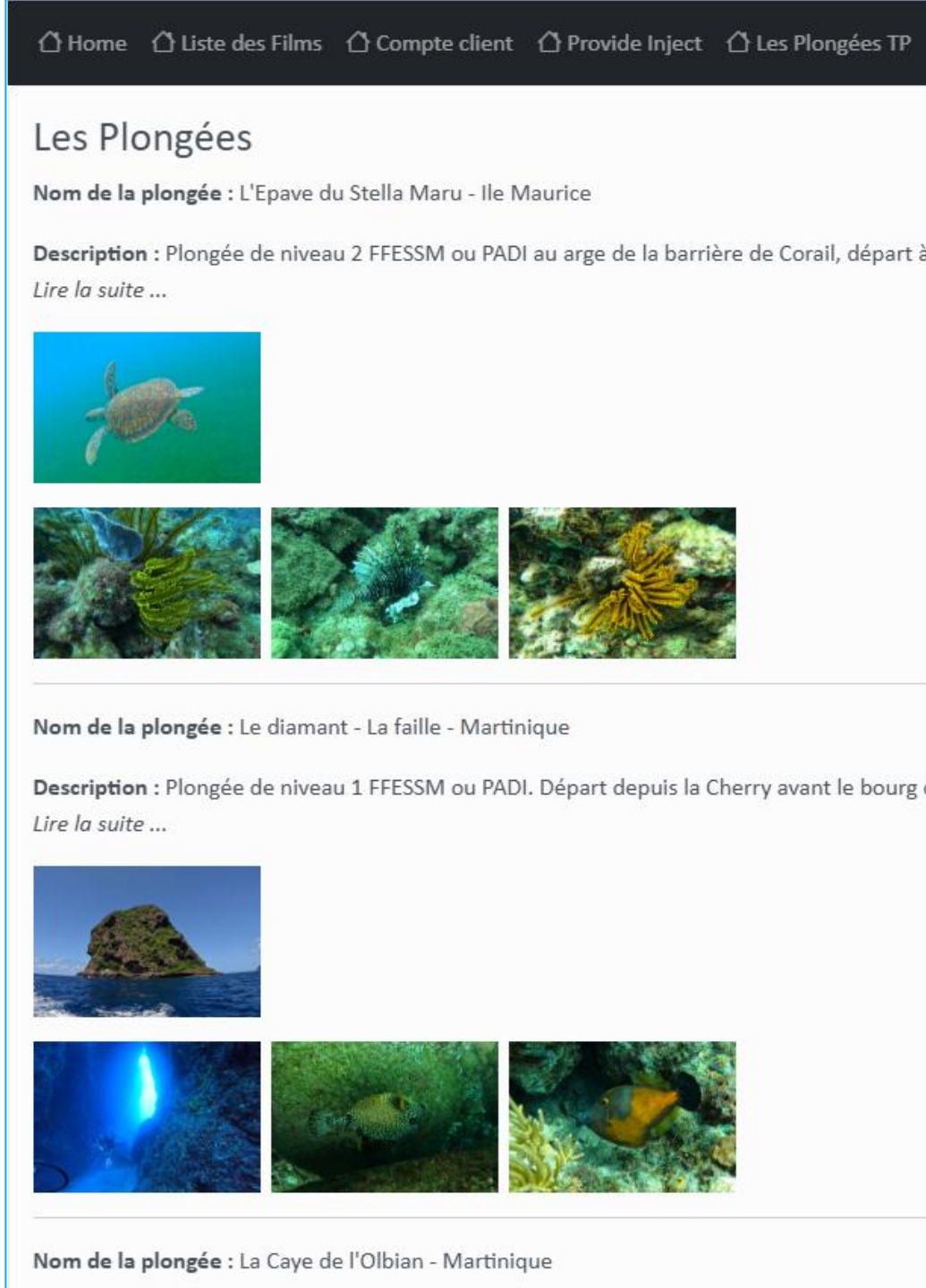
```js
import { ref, computed, reactive } from "vue";

// ce composable permet de filtrer un affichage en fonction d'un motif passé
// searchStr
export const useFilter = (elements, filters = ["title"]) => {
    // le composable prend en paramètres une liste d'éléments
    // et par défaut la prop sur laquelle on pose le filtre
    // ici le title des films...

    const searchStr = ref("");

    const filteredElements = computed(() => {
        console.log(searchStr.value);
        console.log("Depuis Filter : ", elements);

        return elements.value?.filter(
            (elt) => {
                return filters.some((filter) => {
                    return elt[filter]
                        .toLocaleLowerCase()
                        .includes(searchStr.value.toLocaleLowerCase());
                });
            });
    });

    return {
        searchStr,
        filteredElements,
    };
};
```

TP5 : créer une partie de l'application nommée « dives »

Le composant principal Dives vue utilisera la librairie « axios »

1ère étape : créer l'arborescence
2nde étape : créer le routage
3ème étape : installer axios : npm install axios
4ème étape : éditer le fichier mains.js





Dives.vue | main.js

src > webApp > dives > Dives.vue > { } script setup > Dive > photos

```vue
<script setup lang="ts">
import axios from 'axios';
import { onMounted, reactive, ref } from 'vue';

interface Dive {
    id:number;
    name:string;
    description:string;
    photo:string;
    photos:string;
}

let dives = ref([] as Dive[]);

// --------------------------------
onMounted(
    async () => {
        const url='http://localhost:3001/dives';
        await axios.get(url,{
        })
        .then(
            (resHTTP) => {
                console.log('ResHTTP',resHTTP);
                console.log(resHTTP.data);
                dives=resHTTP.data;
            }
        )
})
</script>

<template>
    <div class="alert alert-light">
        <h3>Les Plongées </h3>

        <div v-for="d in dives" :key="d.id">
            <p><strong>Nom de la plongée :</strong> {{ d.name }}</p>
            <p><strong>Description  :</strong> {{ d.description.substring(0,150) }}
            <!-- <p><img :src="`${d.photo}`"></p> -->
            <p><img :src=d.photo ></p>

            <span  v-for="p in d.photos">
                <!-- <img :src="`${p}`">  -->
                <img :src=p> 
            </span>
            <hr>
```

Les slots permettent de transmettre un fragment HTML réexploitable dans les composants Vue

Communication parents enfants

Les Datas : communication entre composants parents et enfants

Les composants parents et enfants communiquent grâce **aux inputs et aux outputs**.

L'input est passé lors de l'appel du composant enfant par le parent

L'output est un Event qui provient de l'enfant pour informer le parent : $emit

```
<!-- le sélecteur du composant enfant -->
<child-film
 :filmDetail="film"
 @nomEventEmitter="getFromChild"

 ></child-film>
```

```
const panier = reactive([]);
const getFromChild = (param) => {
    console.log(param);
    if (param.paramChecked) { panier.push(param.paramMovie); }
    else {
        let keyPanier = panier.indexOf(param.paramMovie);

        if (keyPanier >= 0) { panier.splice(keyPanier, 1); }
    }
}
```

Films.vue

TP06-store-pinia > src > webApp > films > components >

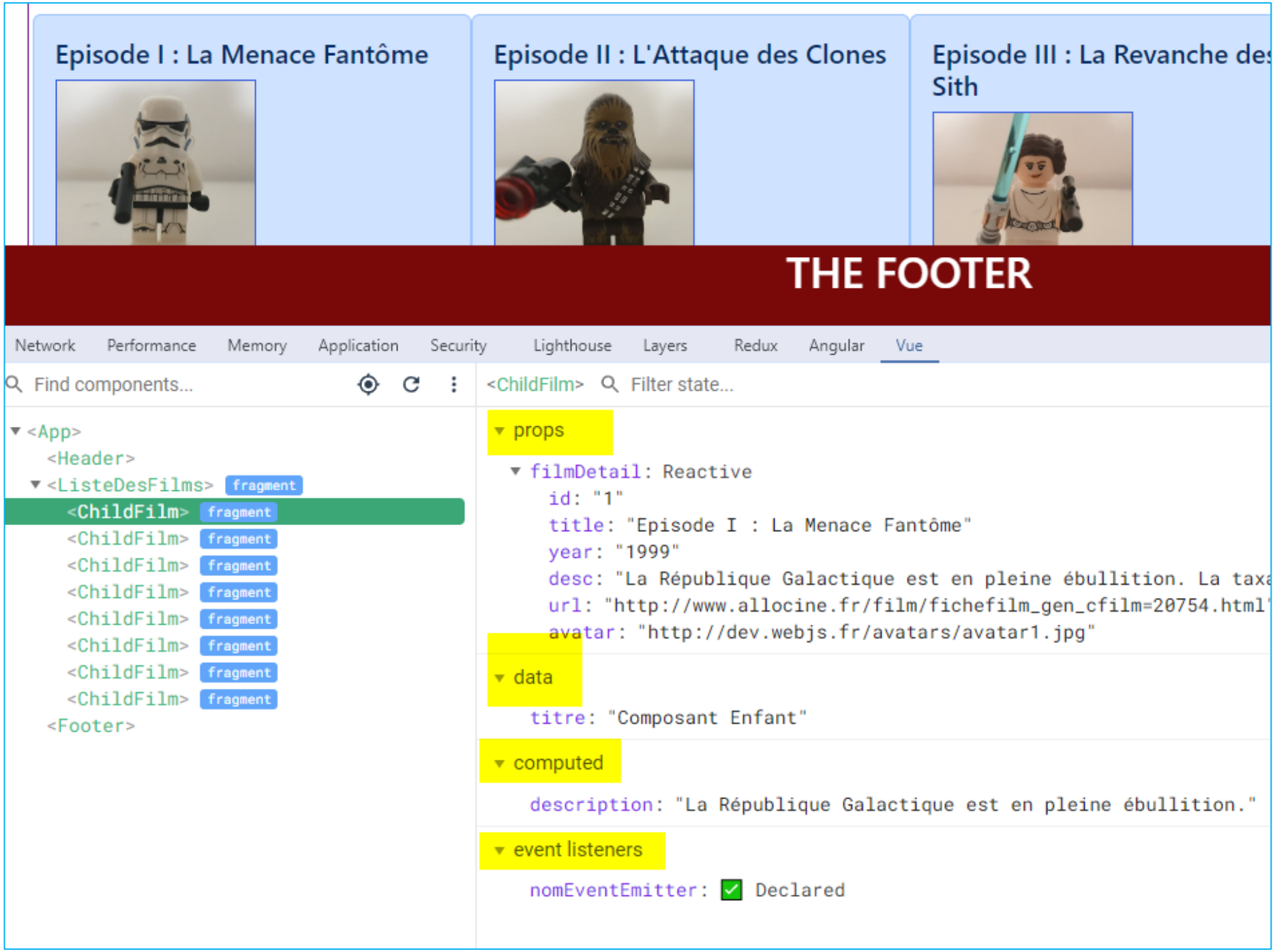
```
</script>

<template>
    <div class="alert alert-light">

        <h3>Liste des Films</h3>
        <h3>Panier</h3>
        <div v-for="item in panier" :key="item.id" :id="item.id" class="alert alert-primary">
            <h5>{{ item.title }} </h5>
        </div>

        <!-- TP useFilter -->
        <div>
            <label>Filtrer l'affichage:</label>
            <p></p>
            <input class="form-control" type="text" v-model="searchStr" />

        </div>

        <p></p>

        <button class="btn btn-success" @click="refetch()">Re-Fetch</button>

        <p></p>

        <div class="row">

            <div v-for="movie in filteredFilms" :key="movie.id" :id="movie.id" class="col-4">
            <!-- :key rendu virtuel du dom soit bien recalculé => optimisation -->
            <h5 class="text-grey">{{ movie.title }}</h5>

            <ChildFilm :propMovie="movie" @nomEventEmitter="getFromChild($event)" />

        </div>

        </div>
```

h5.text-grey

ChildFilm.vue

TP06-store-pinia > src > webApp > films > child-components > ChildFilm... > mplate >

```
<script setup>

import Bouton from '@/shared/slots/Bouton.vue';

defineProps({
    propMovie: Object
})
</script>

<template>

    <div class="alert alert-warning">
        <h5><strong>Composant Enfant</strong></h5>
        <p>Description : {{ propMovie.desc.substring(0, 100) }} <em>l
        <p>Année : {{ propMovie.year }}</p>
        <img :src="propMovie.avatar" alt="">

         
        <input type="checkbox" @change="$emit('nomEventEmitter', {
            'paramFilmId': propMovie.id, //2
            'paramMovie': propMovie, // {title:.....,desc:....}
            'paramChecked': $event.target.checked // $event evt JS
        })"> Add to Cart <i class="bi bi-cart-plus-fill"></i>
        <p></p>
        <p>
            <!-- Router Link -->
            <router-link :to="{
                // path /liste-des-films-star-wars/3
                path: `/liste-des-films-star-wars/${propMovie.id}`,
                query:{
                    title:propMovie.title,
                    description:propMovie.desc,
                    img:propMovie.avatar,
                    objet:JSON.stringify(propMovie)
                }

            }"
```


Episode I : La Menace Fantôme
Episode II : L'Attaque des Clones
Episode III : La Revanche des Sith
THE FOOTER
Network
Performance
Memory
Application
Security
Lighthouse
Layers
Redux
Angular
Vue
Find components...
<ChildFilm>
Filter state...
<App>
<Header>
<ListeDesFilms> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<ChildFilm> fragment
<Footer>
props
filmDetail: Reactive
id: "1"
title: "Episode I : La Menace Fantôme"
year: "1999"
desc: "La République Galactique est en pleine ébullition. La taxa
url: "http://www.allocine.fr/film/fichefilm_gen_cfilm=20754.html"
avatar: "http://dev.webjs.fr/avatars/avatar1.jpg"
data
titre: "Composant Enfant"
computed
description: "La République Galactique est en pleine ébullition."
event listeners
nomEventEmitter: Declared

CAPI : communication entre composants parents et enfants

```
<ChildFilm :propMovie="movie" @nomEventEmitter="getFromChild($event)" />
```

```
defineProps({
    propMovie: Object
})
```

```
 
<input type="checkbox" @change="$emit('nomEventEmitter', {
    'paramFilmId': propMovie.id, //2
    'paramMovie': propMovie, // {title:.....,desc:....}
    'paramChecked': $event.target.checked // $event evt JS
})"> Add to Cart <i class="bi bi-cart-plus-fill"></i>
<p></p>
```

CAPI : communication entre composants parents et enfants

En composition API, les méthodes **Ref**, **Reactive** sont utilisées pour gérer la réactivité et la communication.

Ref : crée une simple variable réactive, elle retourne un Objet. Elle peut être typé avec les types primitifs de javascript (String, Number, Object, Array, Date)

Reactive : est initialisé uniquement à partir d'un Objet (ex : le panier)

$emit est toujours utilisé depuis le composant enfant

ref.vue | DetailFilm.vue | ChildFilm.vue

TP06-routage > src > composables > ref.vue > {} script setup

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

const nb = ref(3)
const titre =ref('Formation Vue')
const format =ref('distanciel')

const datas = reactive({
        nb:3,
        titre:'Formation Vue',
        format:'distanciel'
})

</script>
```

CAPI : communication entre composants parents et enfants

ListeDesFilms.vue

TP06-routage > src > components > ListeDesFilms.vue > {} template

```vue
<script setup>

import { ref, reactive } from 'vue'
import { useFetch } from '../composables/fetch.js';
import ChildFilm from '../childComponents/ChildFilm.vue';

const url = ref('http://localhost:3000/films1')
const { datas, error } = useFetch(url, 'get')

const titre = ref('Liste des Films');
const films = ref([]);
const panier = reactive([]);

const getFromChild = (args) => {
  console.log('From Parent ', args);
  console.log(args.objet, args.evt);

  let isChecked = args.evt;
  let filmSelect = args.objet;
  // --
  if (isChecked) {  panier.push(filmSelect); }
  else {
    let keyPanier = panier.indexOf(filmSelect);
    if (keyPanier >= 0) { panier.splice(keyPanier, 1);}
  }
}
</script>

<template>
  <h5 class="alert alert-primary">{{ titre }}</h5>
  <div class="alert alert-success">
    <h5><em>Panier</em></h5>
    <div v-for="item in panier" :key="item.id" :id="item.id">
      <h5>{{ item.title }} </h5>
    </div>
  </div>
```

ChildFilm.vue

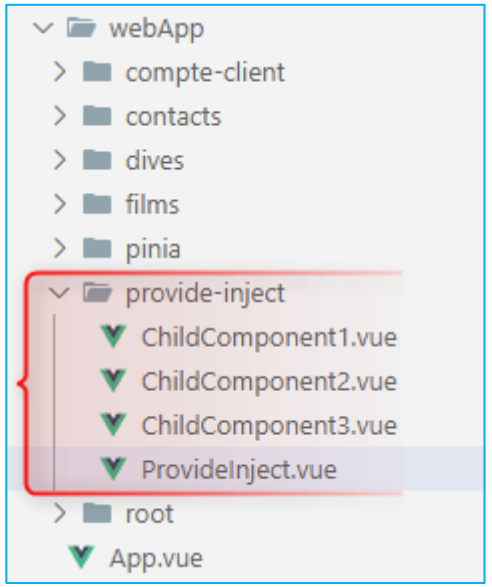
TP06-routage > src > childComponents > ChildFilm.vue > {} template > p.alert.alert-warning

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

defineProps({
    msg:String, filmDetail:Object
})

const titre = ref('Composant Enfant');
</script>

<template>

    <h5 class="alert alert-danger">{{ titre }} - {{ msg }} </h5>
    <!-- <p> <strong>Résumé</strong> : {{ filmDetail.desc.substring(0,100) }} ...</p>  -->
    <p> <strong>Résumé</strong> : {{ filmDetail.desc }} <em>lire la suite .</em></p>

    <p class="alert alert-warning">
        Add to Cart <i class="bi bi-cart-plus-fill"></i>
        <input type="checkbox" @change=" $emit('nomEventEmitter',
                {
                    'objet': filmDetail,
                    'evt': $event.target.checked
                });">
    </p>
    <div></div>


</template>

<style scoped>
Codiumate: Options | Test this class
img {
    width: 150px;
    border: 1px solid #345ce0;

}
</style>
```

Provide et Inject

Imaginons qu'un composant souhaite faire passer une donnée à un petit-composant enfant très éloigné, utiliser les props serait coûteux.

TP : Créer l'arborescence de provide-inject et le routage associé



Un composant peut « provider » une donnée quelconque qui pourra être injectée beaucoup plus bas dans l'arbre des composants 🙂
https://vuejs.org/guide/components/provide-inject

ProvideInject.vue

src > webApp > provide-inject > ProvideInject.vue > {} template

```vue
<script setup>
import { provide, ref } from 'vue';
import ChildComponent1 from './ChildComponent1.vue';


const titre1='Provide et Inject';
const texte= ref(`Private et Inject c'est Cool 😊`)
provide('message',texte)
// provide défdinit une prop avec une valeur associée

</script>

<template>

    <div class="alert alert-warning">

        <h3>Composant #1</h3>
        <p><strong>Titre: </strong>{{titre1}}</p>
        <hr>
        <ChildComponent1 :title="titre1"/>

```

ChildComponent1.vue

src > webApp > provide-inject > ChildComponent1.vue > {} script setup

```vue
<script setup>
import ChildComponent2 from './ChildComponent2.vue';
defineProps({
    title:String
})

</script>

<template>

    <div class="alert alert-success">
        <h3>Composant #2</h3>
        <p>{{title}}</p>
        <hr>
        <ChildComponent2 />

    </div>

</template>
```

ChildComponent2.vue

src > webApp > provide-inject > ChildComponent2.vue > {} script setup

```vue
<script setup>
import ChildComponent3 from './ChildComponent3.vue';



</script>

<template>

    <div class="alert alert-danger">
        <h3>Composant #3</h3>
        <p>{{title}}</p>
        <hr>
        <ChildComponent3 />

    </div>

</template>
```

ChildComponent3.vue

src > webApp > provide-inject > ChildComponent3.vue > {} script setup

```vue
<script setup>
import { inject } from 'vue';

const msg = inject('message')
// inject permet de récupérer la valeur de la prop providée

</script>

<template>

    <div class="alert alert-primary">
        <h3>Composant #4</h3>
        <p>{{msg}}</p>
        <hr>

    </div>

</template>
```

Le ROUTAGE

Routage

Le routage nécessite d'installer vue-router, s'il n'a pas été proposé lors de l'installation par le(s) framework(s)

npm install vue-router

Chaque **route** (path ou fragment d'URL) est associé au Template HTML d'un composant

Les routes se définissent dans le dossier **router**

index.js main.js

TP05-routage > src > main.js > ...

```
1  import './assets/main.css';
2
3  // import bootstrap js
4  import 'bootstrap';
5
6  import { createApp } from 'vue'
7  import App from './App.vue'
8
9  import router from './router'
10
11 const app = createApp(App)
12 // composant principal = container = Appli
13
14 app.use(router)
15 // routage
16
17 app.mount('#app')
18 // le 1er rendu
19
```

Routage

```
const router = createRouter({

  history: createWebHistory(import.meta.env.BASE_URL),

  routes: [
    {
      path: '/', // www.domain.tld/ == page de démarrage
      name: 'home',
      component: HomeComponentVue
    },
    {
      path: '/liste-des-films',
      name: 'liste',
      component: ListeDesFilmsVue
    },
    ,
```

index.js | Header.vue ×

TP05-routage > src > components > Header.vue > {} script setup

```
19                </button>
20
21                <div class="collapse navbar-collapse" id="menu">
22                    <ul class="navbar-nav">
23
24                        <li class="nav-item ">
25                            <router-link to="/">
26                                <a class="nav-link"><i class="bi bi-house"></i> Home
27                                </a>
28                            </router-link>
29                        </li>
30
31                        <li class="nav-item ">
32                            <router-link to="/liste-des-films">
33                                <a class="nav-link">
34                                    <i class="bi bi-house"></i> Liste des Films
35                                </a>
36                            </router-link>
37                        </li>
```

index.js | Header.vue | App.vue ×

TP05-routage > src > App.vue > {} script

```
41
42  <template>
43
44  <div class="container">
45
46      <Header></Header>
47
48      <div class="alert alert-warning">
49          <!-- le router-view permet de charger les templates HTML des composants -->
50
51          <router-view></router-view>
52
53      </div>
54
55      <Footer></Footer>
```

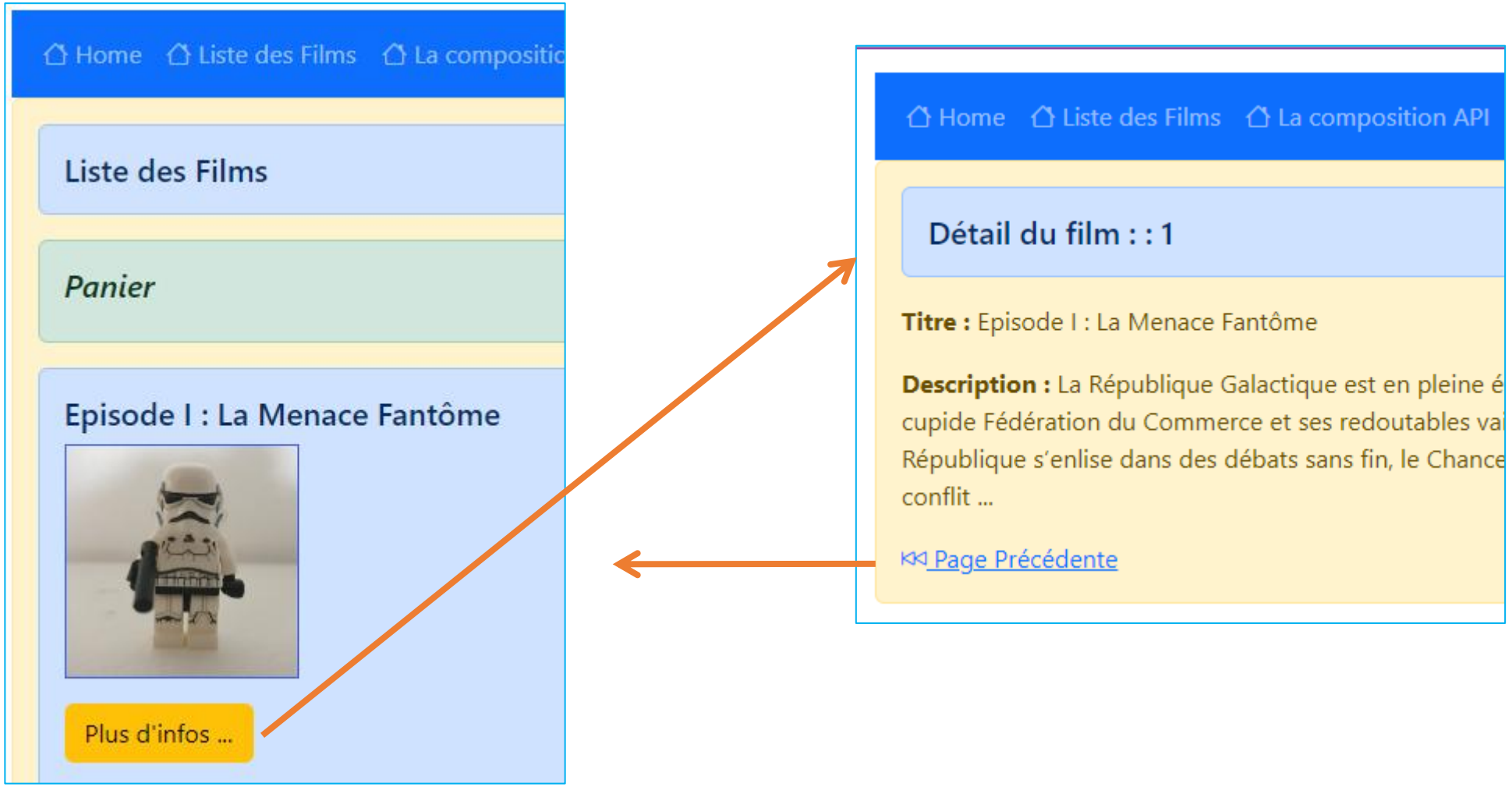
Routage Avancé

Le routage nécessite d'installer vue-router, s'il n'a pas été proposé lors de l'installation par le(s) framework(s)

npm install vue-router

Chaque **route** (path ou fragment d'URL) est associée au Template HTML d'un composant

Les routes se définissent dans le dossier **router**



Routage Avancé

/router/index.js

```
{
  path:'/liste-des-films/:id',
  component:DetailFilmVue,
},
```

```
<router-link :to="{
  path:`/liste-des-films/${film.id}`,
  query:{
    title:film.title,
    desc:film.desc,
    img:film.img}
}">
```

ListeDesFilms.vue

index.js | DetailFilm.vue

TP05-routage > src > components > DetailFilm.vue > {} script

```
<script>

export default {

  name:'DetailFilm',
  props:[],
  data() {
    return {
      titre:'Détail du film :'
    }
  },
  computed : {

    title () { return this.$route.query.title},
    desc () { return this.$route.query.desc},
    img () { return this.$route.query.img},
  }

}

// avec CAPI : const route = useRoute()
// à la place de $route en OAPI
```

```
<template>

  <h5 class="alert alert-primary">{{titre}} :  {{ $route.params.id }}</h5>
  <!-- queryParams qui récupère les paramètres passés -->

  <p><strong>Titre : </strong>{{ title }}</p>
  <p><strong>Description : </strong>{{ desc }}</p>


  <router-link class="link" to="/liste-des-films">
    <i class="bi bi-skip-backward"></i> Page Précédente
  </router-link>
```

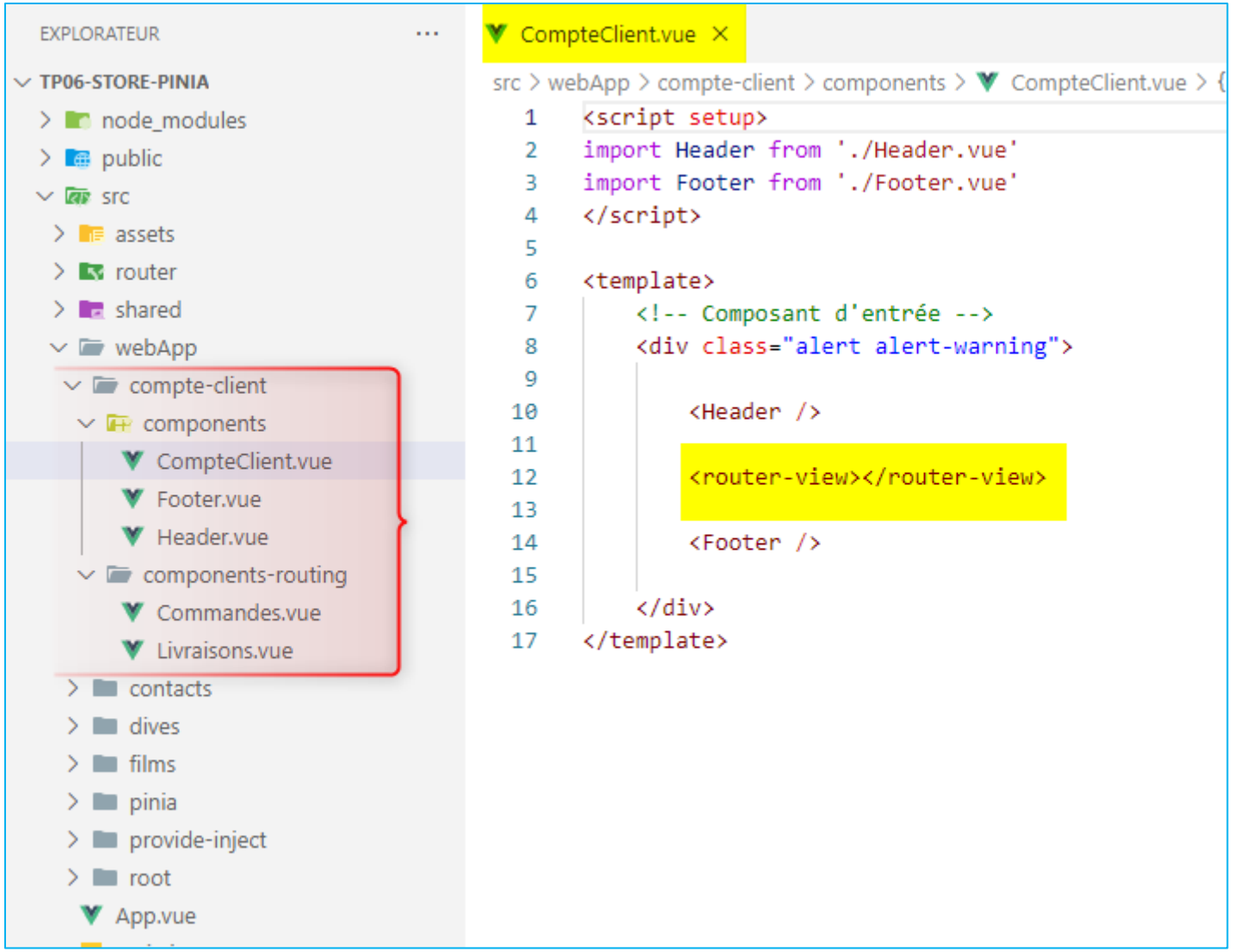
Le routage

ListeDesFilms.vue | DetailFilm.vue

TP06-routage > src > components > DetailFilm.vue > {} script setup

```vue
<script setup>
import { ref, reactive, watch } from 'vue'
import { useRoute } from 'vue-router'

const titre = ref('Détail du film :')
const route = useRoute()


const title = ref(route.query.title)
const desc = ref(route.query.desc)
const img = ref(route.query.img)



</script>

<template>

  <h5 class="alert alert-primary">{{ titre }} : {{ $route.params.id }}</h5>
  <!-- queryParams qui récupère les paramètres passés -->

  <p><strong>Titre : </strong>{{ title }}</p>
  <p><strong>Description : </strong>{{ desc }}</p>
  <p><img :src="img" alt=""></p>


  <router-link class="link" to="/liste-des-films">
    <i class="bi bi-skip-backward"></i> Page Précédente
  </router-link>

```

TP : Créer l'arborescence de la partie compte-client



TP : Créer l’arborescence de la partie compte-client

En général, le composant point d’entrée « entry-point » porte le même nom que la partie de l’application concernée.

Ce composant structure cette partie de l’application. Il est composé de sélecteurs de composants et généralement d’un <router-view>

TP : Solution

CompteClient.vue | Header.vue

src > webApp > compte-client > components > Header.vue > { } template

```vue
<template>

    <div class="alert alert-light">
        <h5 class="text-primary">Header Compte Client</h5>
        <ul>

            <li>
                <!-- <router-link :to="{name:'cmd'}"> -->
                <router-link to="/compte-client/mes-commandes">
                    <a>Mes commandes</a>
                </router-link>
            </li>

            <li>
                <!-- <router-link :to="{name:'livraisons'}"> -->
                <router-link to="/compte-client/mes-livraisons">
                    <a>Mes Livraisons</a>
                </router-link>
            </li>


        </ul>


    </div>

</template>
```

index.js

src > router > index.js > ...

```js
const router = createRouter({
  routes: [
    // component: Contacts,
    name: 'contacts',
    // Lazy Loading : permet de compiler un chunk file final séparé du b
    // optimisation du poids du build principal car le fichier ne sera c
    component: () => import('@/webApp/contacts/Contacts.vue')
  }
  ,
  {
    path:'/compte-client/',
    component:CompteClient,
    name:'client',
    children:[
      { path:'mes-commandes', component:Commandes, name:'cmd'},
      { path:'mes-livraisons', component:Livraisons, name:'livraisons'},
    ]
  },
  {
    path: '/provide-inject',
    component: ProvideInject
  },
  {
    path: '/dives',
    component: Dives
  },
  {
    path: '/store-pinia',
```

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/', // www.domain.tld/ => page de démarrage
      component: Home,
      // pour donner un nom à cette route
      // plus facile à mettre à jour (alias)
      name: 'home'
    },
    {
      path: '/liste-des-films-star-wars', // SEO
      component: Films,
      name: 'liste'
    },
    {
      path: '/liste-des-films-star-wars/:id', // SEO
      component: FilmsDetails
    },
    {
      path: '/contactez-nous', // SEO
      // component: Contacts,
      name: 'contacts',
      // Lazy Loading : permet de compiler un chunk file final séparé du build principal,
      // optimisation du poids du build principal car le fichier ne sera chargé que si la route (path) est invoquée
      component: () => import('@/webApp/contacts/Contacts.vue')
    }
    ,
    {
      path:'/compte-client/',
      component:CompteClient,
      name:'client',
      children:[
        { path:'mes-commandes', component:Commandes, name:'cmd'},
        { path:'mes-livraisons', component:Livraisons, name:'livraisons'},
      ]
    },
```

```
// pour capturer tous les url non répertoriées
  {
    path:'/:pathMatch(.*)',
    component:Page404
    // redirect:to => {
    //   return { path:'/'}
    // }
  },
  // { path:'nouvelle-url', component:ComponentX},
  // { path:'ancienne-url-seo-déjà-référencée',
  // redirect:to => {
  //   return {
  //     path:'/nouvelle-url'
  //   }
```

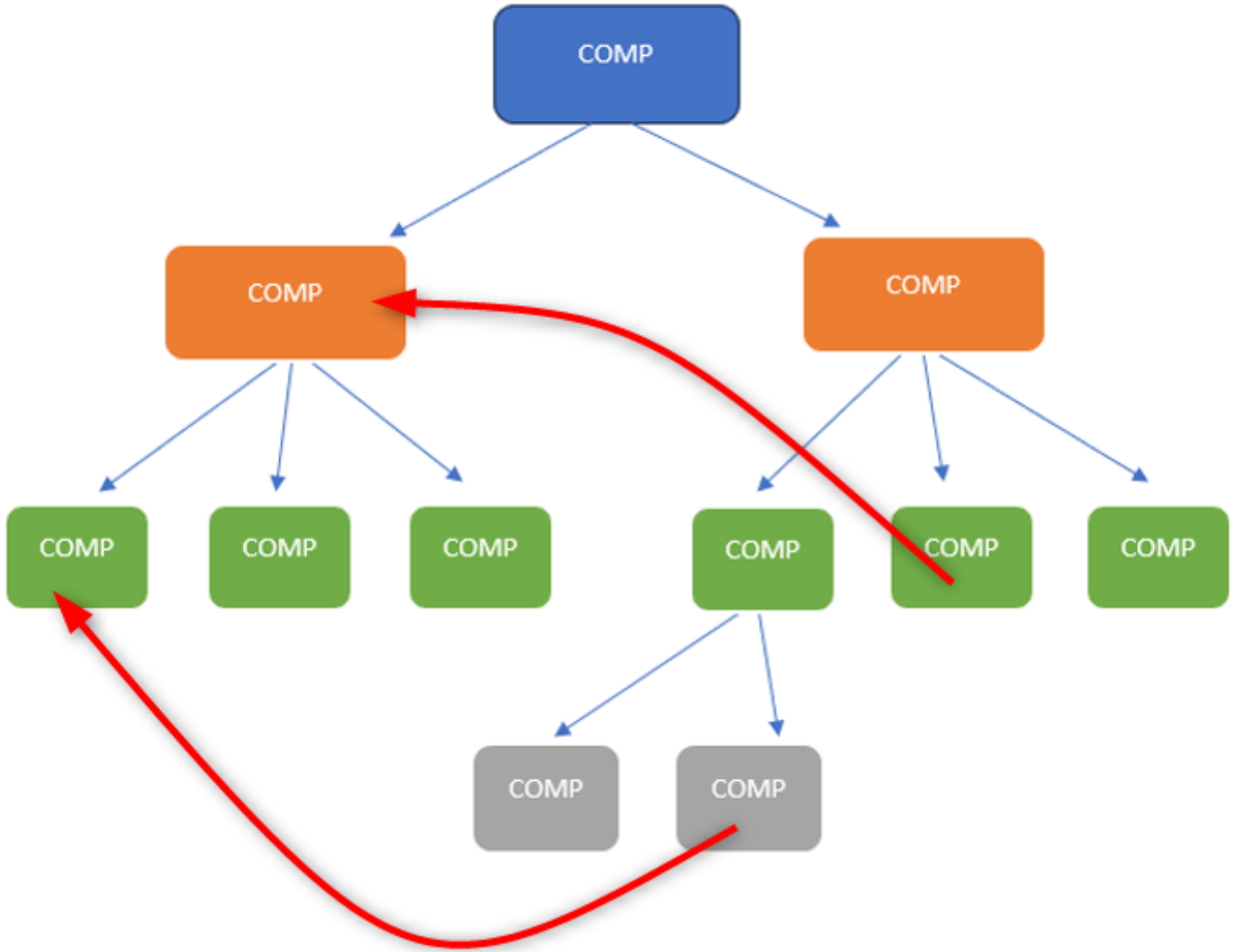
PINIA – La notion de STORE

Le store pourquoi ?

Lorsqu'il y a beaucoup d'interactions entre composants et composants-enfants,
Lorsque le projet devient très conséquent en termes de développement,
Lorsque le projet grandit et se construit au fur et à mesure …

Le projet est de plus en plus difficile à maintenir.
On peut rencontrer des effets de bords non désirés.
La structure « complexe » du projet rend de plus en plus difficile les opérations d'amélioration et d'enrichissement.

Imaginons qu'un composant éloigné dans l'arbre des composants souhaite communiquer avec un autre composant également éloigné.



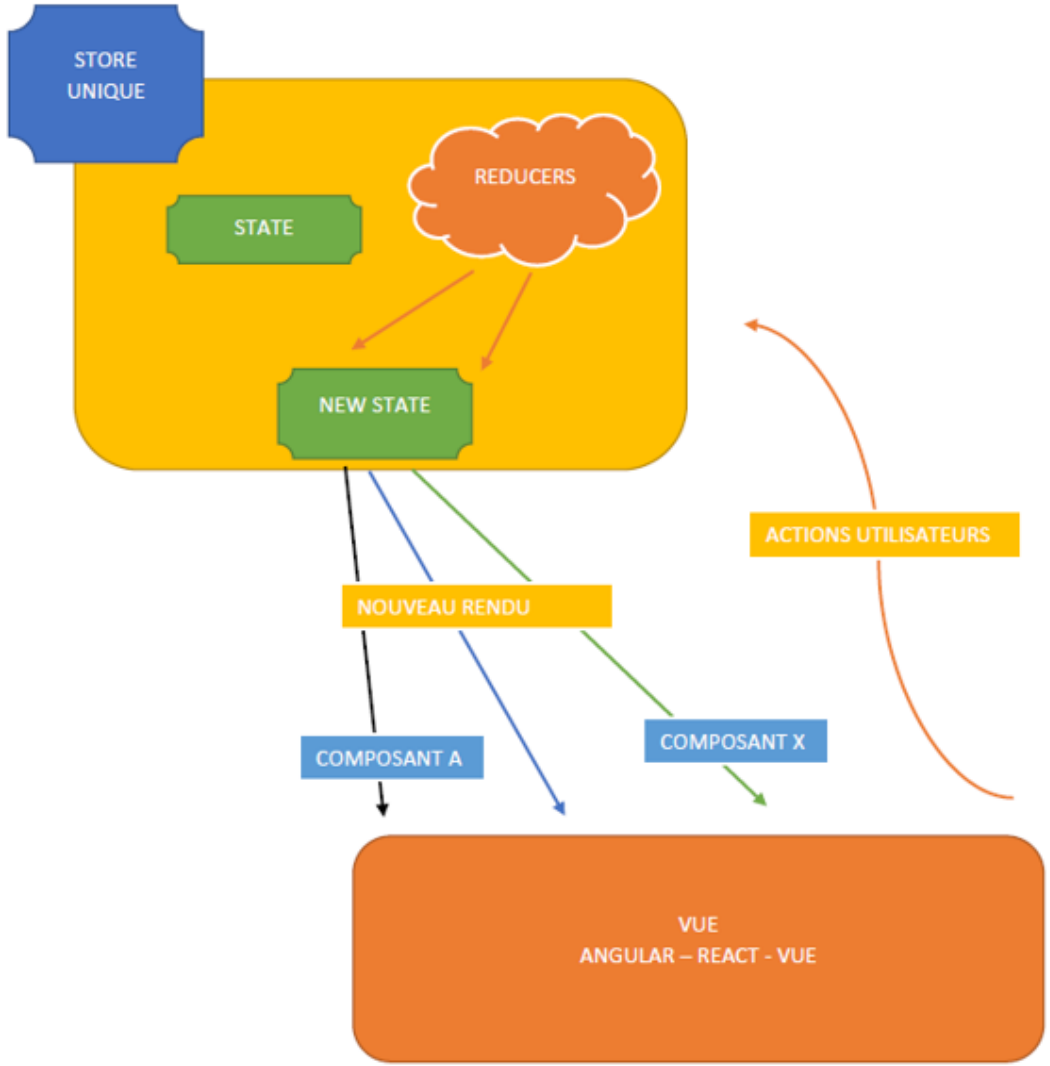
Utiliser les props et $emit ou même provide et inject peut vite devenir « assez » complexe.

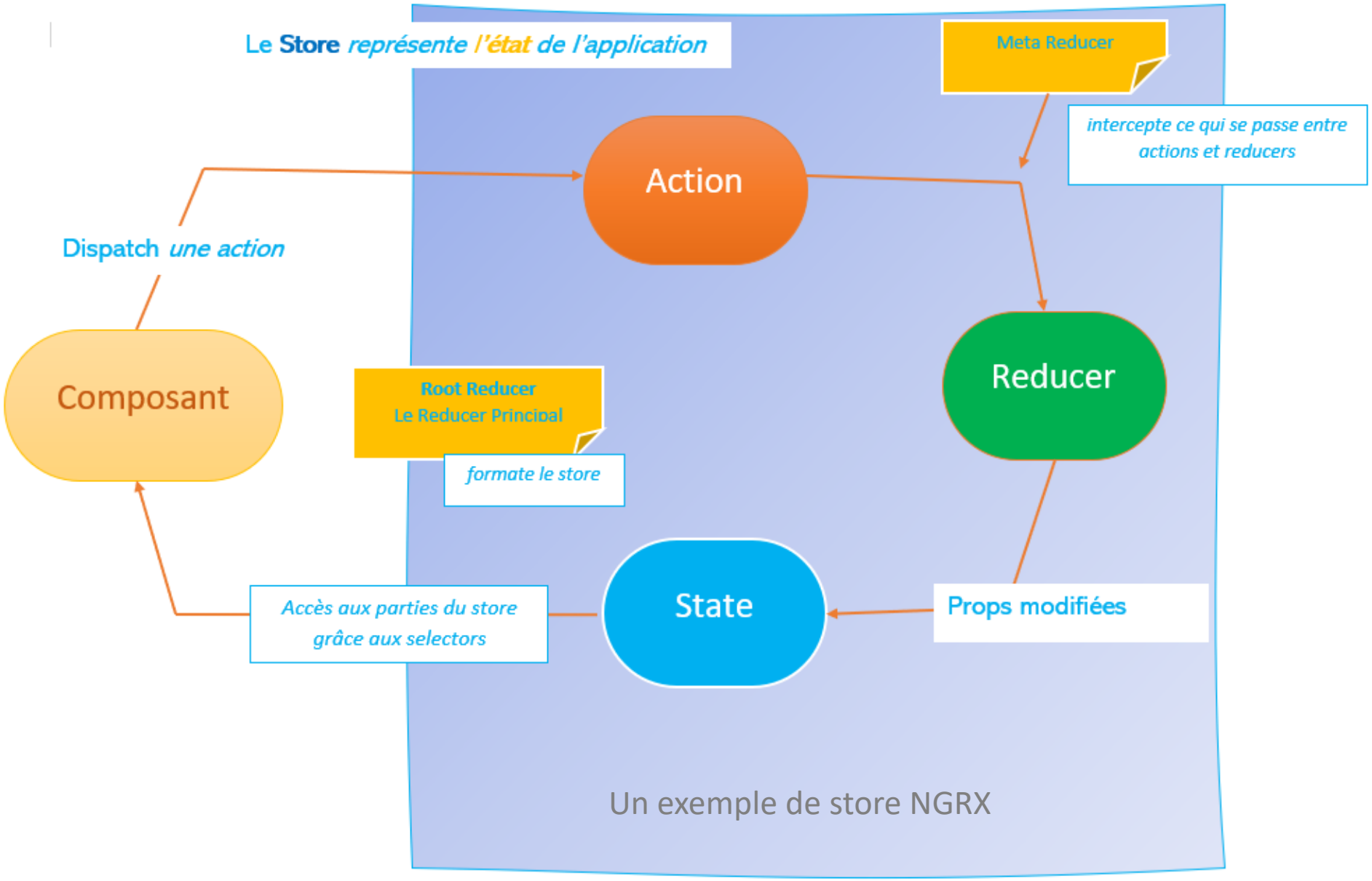
VueX – PINIA - la solution

Au lieu de traverser tout l'héritage des Input et des Output des composants parents-enfants, le composant peut s'inscrire auprès du Store et dispatcher une Action.
C'est-à-dire demander quelque chose ou demander à faire quelque chose.

L'accès est direct sans passer par la structure parfois « trop rigide » des web-components imbriqués.

Système de centralisation de DATAS et d'ACTIONS





Un exemple de store NGRX

Notion de STORE

https://pinia.vuejs.org

Le store nécessite d'installer **pinia**, s'il n'a pas été proposé lors de l'installation par le(s) framework(s)

npm install pinia

```
PS C:\Users\Michel\Desktop> npm create vite@latest
√ Project name: ... TP-store-pinia
√ Package name: ... tp-store-pinia
√ Select a framework: » Vue
√ Select a variant: » Customize with create-vue 

Vue.js - The Progressive JavaScript Framework

√ Nom du package : ... tp-store-pinia
√ Ajouter TypeScript ? ... Non / Oui
√ Ajouter le support de JSX ? ... Non / Oui
√ Ajouter Vue Router pour le développement d'applications _single page_ ? ... Non / Oui
√ Ajouter Pinia pour la gestion de l'état ? ... Non / Oui
√ Ajouter Vitest pour les tests unitaires ? ... Non / Oui
√ Ajouter une solution de test de bout en bout (e2e) ? » Non
√ Ajouter ESLint pour la qualité du code ? ... Non / Oui
√ Ajouter l'extension Vue DevTools 7 pour le débogage ? (expérimental) ... Non / Oui

Génération du projet dans C:\Users\Michel\Desktop\TP-store-pinia...

Terminé. Exécutez maintenant :

  cd TP-store-pinia
  npm install
  npm run dev

PS C:\Users\Michel\Desktop> cd .\TP-store-pinia\
PS C:\Users\Michel\Desktop\TP-store-pinia> npm install
```

EXPLORATEUR

ÉDITEURS OUVERTS

package.json

TP-STORE-PINIA

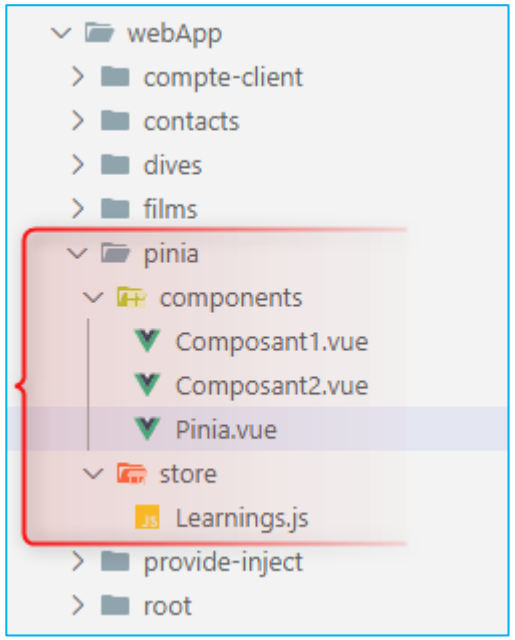
- .vscode
- public
- src
- .gitignore
- env.d.ts
- index.html
- package.json
- README.md
- tsconfig.app.json
- tsconfig.json
- tsconfig.node.json
- tsconfig.vitest.json
- vite.config.ts
- vitest.config.ts

package.json

package.json > ...

```
{
  "name": "tp-store-pinia",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  ▷ Déboguer
  "scripts": {
    "dev": "vite",
    "build": "run-p type-check \"build-only {@}\" --",
    "preview": "vite preview",
    "test:unit": "vitest",
    "build-only": "vite build",
    "type-check": "vue-tsc --build --force"
  },
  "dependencies": {
    "pinia": "^2.1.7",
    "vue": "^3.4.21",
    "vue-router": "^4.3.0"
  },
  "devDependencies": {
    "@tsconfig/node20": "^20.1.4",
    "@types/jsdom": "^21.1.6",
    "@types/node": "^20.12.5",
    "@vitejs/plugin-vue": "^5.0.4",
    "@vue/test-utils": "^2.4.5",
    "@vue/tsconfig": "^0.5.1",
    "jsdom": "^24.0.0",
    "npm-run-all2": "^6.1.2",
    "typescript": "~5.4.0",
    "vite": "^5.2.8",
    "vite-plugin-vue-devtools": "^7.0.25",
    "vitest": "^1.4.0",
    "vue-tsc": "^2.0.11"
  }
}

```

TP : Créer l’arborescence de store ainsi que le routage associé, « Pinia.vue » est le composant point d’entrée de cette partie de l’application



Composition du store Pinia

Le fichier de store est composé de plusieurs parties :

1. Le state : c'est l'équivalent de ref()

Learnings.js

formation-vue-sopra-02-2025 > TP06-store-pinia > src > webApp > pinia > store > Learnings.js > useLearningStore

```
1   // 1- définition du store
2
3   import { defineStore } from "pinia";
4
5   export const useLearningStore = defineStore('learnings', {
6
7       // 1- notion de state : le state représente l'état global de l'appplication storée 😉
8       // state = équivalent de ref
9       state: () => {
10          return {
11              learnings:[],
12              selectedLearning:null,
13              loaded:false,
14              films:[],
15              dives:[],
16              titre:'Le Store Pinia',
17              folowersNumber:150
18
```

Composition du store Pinia

Le fichier de store est composé de plusieurs parties :

2. Les actions : sont les équivalents des méthodes
   Elles seront appelées par des interactions utilisateurs principalement

```
// 2- Actions
//   equivalent des méthods
actions: {
    // 1ère action : sauvegarder une data dans le store (dans le tableau learnings[])
    actionCreateLearning(obj) {
        console.log('Depuis le store : ', obj);
        this.learnings.push(obj);
    },
    // -- 2nde Action
    actionRemoveLearning(data) {
        console.log(data);

        this.learnings = this.learnings.filter(
            (learn) => learn.id !== data.id
        );
        // localStorage.removeItem(key);
    },
    // -- 3ème Action : persister les données
    actionStorage(){
        localStorage.setItem('store_learnings', JSON.stringify(this.learnings))
    }
},
```

Composition du store Pinia

Le fichier de store est composé de plusieurs parties :

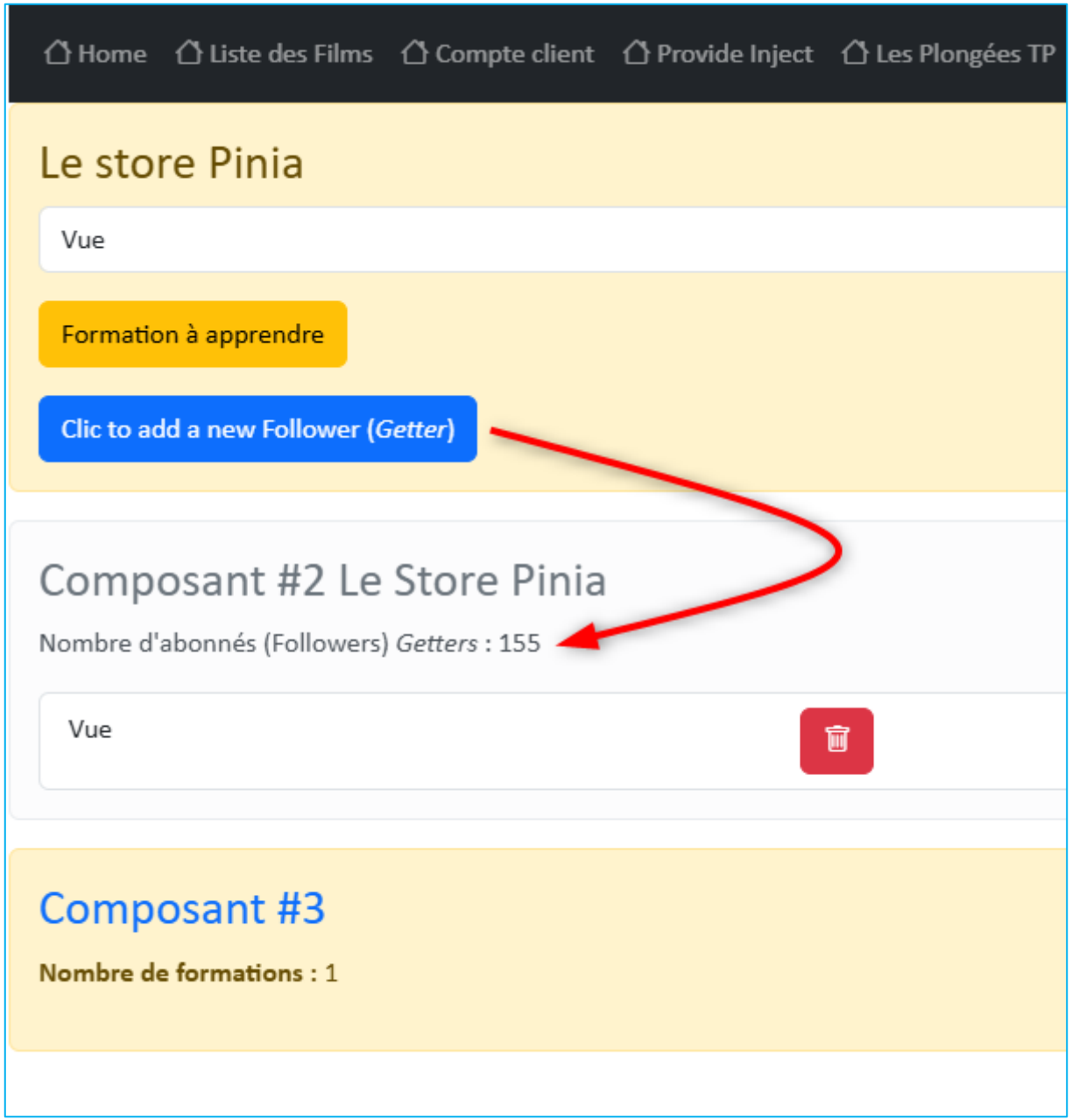
3. Les getters : sont les équivalents des computed properties

   Quelle est la différence entre une méthode et une computed properties ?

   Les méthodes s'éxécutent lorsqu'un appel (interaction) du DOM HTML est requise.
   Les computed properties sont mises à jour automatiquement lorsque la source de la dépendance est modifiée.
   Il s'agit bien d'une valeur dérivée d'une properties de state.

```
state: () => {
    return {
        learnings:[],
        selectedLearning:null,
        loaded:false,
        films:[],
        dives:[],
        titre:'Le Store Pinia',
        folowersNumber:150
```

```
},
getters :{
    getUpadteFolowersNumber : (state) => { return state.folowersNumber += 1; }
}
```


Home
Liste des Films
Compte client
Provide Inject
Les Plongées TP
Le store Pinia
Vue
Formation à apprendre
Clic to add a new Follower (Getter)
Composant #2 Le Store Pinia
Nombre d'abonnés (Followers) Getters : 155
Vue
Composant #3
Nombre de formations : 1

Home | Liste des Films | Compte client | Provide Inject | Les Plongées TP | Le store Pinia | Nous contacter | Go Back

Le store Pinia

Angular

Formation à apprendre"

Composant #2 Le Store Pinia

React

Vue

Angular

Composant #3

Nombre de formations : 3

construction du formulaire dans le composant Pinia

```vue
<script setup>
import { ref } from 'vue';
import { v4 as uuid } from 'uuid'

// --------------------------
// state (datas) du composant
const learningName = ref('')
// --------------------------

// --------------------
// functions
// --------------------
const handleSubmit = () => {
  const learning = {
    name: learningName.value,
    completed: false,
    id:uuid()
  }
  console.log(learning);
}

</script>
<template>

  <form @submit.prevent="handleSubmit()">

    <h3 class="text-light">Deviens un Super Dev !</h3>
    <img src="/images/superdev.png" style="max-width:250px">

    <input v-model="learningName" placeholder="Quelle techno t
    <p></p>
    <p><button class="btn btn-primary">
        OK
      </button></p>

  </form>

</template>

<style scoped></style>
```

création du fichier dans stores learning.js

Learnings.js

src > webApp > pinia > store > Learnings.js > ...

```
1   // 1- définition du store
2
3   import { defineStore } from "pinia";
4
5   export const useLearningStore = defineStore('learnings', {
6
7       // 1- notion de state : le state représente l'état global de l'appplication storée 😉
8       state: () => {
9           return {
10              learnings:[],
11              selectedLearning:null,
12              loaded:false,
13              films:[],
14              dives:[],
15              titre:'Le Store Pinia'
16          }
17      },
18      // 2- Actions
19      actions: {
20          // 1ère action : sauvegarder une data dans le store (dans lke tableau learnings[])
21          actionCreateLearning(obj) {
22              console.log('Depuis le store : ', obj);
23              this.learnings.push(obj);
24          },
```

Connexion du composant au STORE

Learnings.js

src > webApp > pinia > store > Learnings.js > ...

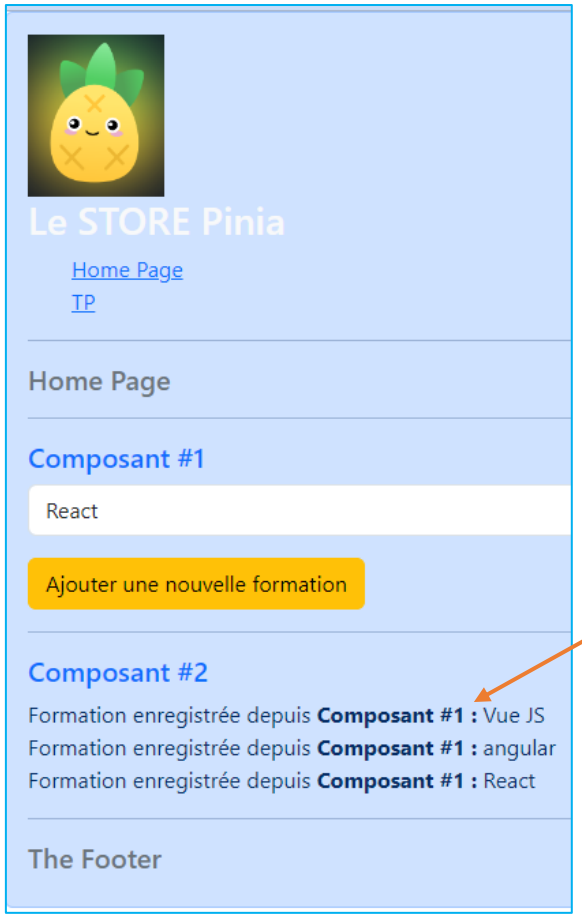
```
1  // 1- définition du store
2
3  import { defineStore } from "pinia";
4
5  export const useLearningStore = defineStore('learnings', {
6
7      // 1- notion de state : le state représente l'état global de l'
8      state: () => {
9          return {
10             learnings:[],
11             selectedLearning:null,
12             loaded:false,
13             films:[],
14             dives:[],
15             titre:'Le Store Pinia'
16         }
17     },
18     // 2- Actions
19     actions: {
20         // 1ère action : sauvegarder une data dans le store (dans l
21         actionCreateLearning(obj) {
22             console.log('Depuis le store : ', obj);
23             this.learnings.push(obj);
24         },
25         // -- 2nde Action
```

Pinia.vue

src > webApp > pinia > components > Pinia.vue > { } script setup

```
1  <script setup lang="ts">
2  import { ref } from 'vue';
3  import { v4 as uuid} from 'uuid';
4  import { useLearningStore } from '../store/Learnings';
5  import Composant1 from './Composant1.vue';
6  import Composant2 from './Composant2.vue';
7
8  // props
9  const learningName = ref('');
10 // methods
11 const handleSubmit = () => {
12     const objLearning = {
13         // /liste de params
14         name:learningName.value,
15         completed:false,
16         id:uuid()
17     }
18     // appel de l'action du store
19     const _store = useLearningStore(); // instance du store Learnings
20     console.log(_store);
21     _store.actionCreateLearning(objLearning);
22     _store.actionStorage()
23
24 }
25
```

Utilisation des valeurs du Store
depuis n'importe quel composant !



Component1.vue | Component2.vue | learnings.js | Home.vue

src > child-components > Component2.vue > {} template > div > div

```vue
<script setup>
import { useLearningStore } from '../stores/learnings';
import { storeToRefs } from 'pinia';


// ------------------------------------------------------------------
// Connexion du composant au store et utilisation des refs du store
//------------------------------------------------------------------
const learningStore = useLearningStore();
const { learnings } = storeToRefs(learningStore);

</script>

<template>
  <h5 class="text-primary">Composant #2</h5>

  <div v-for="learn in learnings">

    <div v-bind:id="learn.id">

      Formation enregistrée depuis
      <strong>Composant #1 :</strong>
      {{ learn.name }}
    </div>

  </div>

</template>

<style scoped></style>
```

TP : Créer un autre composant (TP1) appelé par le routage et appeler les données du store

Edition d’une donnée du store

Component2.vue × Component1.vue

src > child-components > Component2.vue > {} script setup

```
26    // on supprime l'ancienne formation
27
28  }
29
30  </script>
31
32  <template>
33    <h5 class="text-primary">Composant #2</h5>
34
35    <div v-for="learn in learnings">
36
37      <ul class="list-group" v-bind:id="learn.id">
38
39        <li class="list-group-item">
40          <div class="row">
41            <div class="col-4">{{ learn.name }}</div>
42
43            <div class="col-4">
44              <button class="btn btn-danger" @click="remove(learn)"><i class="bi bi-trash"></i></button>
45            </div>
46
47            <div class="col-4">
48              <button class="btn btn-success" @click="edit(learn)"><i class="bi bi-pencil-fill"></i></button>
49            </div>
50
51          </div>
52        </li>
53
54      </ul>
55
56    </div>
57
58  </template>
```

Mise à jour des contenus du Template

Component2.vue | Component1.vue

src > child-components > Component1.vue > { } template > form

```
62  </script>
63  <template>
64
65    <form @submit.prevent="handleSubmit()">
66
67      <h5 class="text-primary">Composant #1</h5>
68
69      <h5 class="text-warning">{{ learningStore.selectedLearning ? "Mettre à jour" : "Formation à apprendre" }}</h5>
70
71      <input v-model="learningName" placeholder="Quelle techno tu veux apprendre ?" class="form-control" />
72      <p></p>
73      <p>
74        <button class="btn btn-warning">
75          {{ learningStore.selectedLearning ? "Mettre à jour" : "Formation à apprendre" }}
76        </button>
77      </p>
78      <p v-if="learningStore.selectedLearning">
79        <button class="btn btn-danger" @click="cancelUpdate">Annuler</button>
80      </p>
81    </form>
82
83  </template>
84
85  <style scoped></style>
```

Component2.vue | Component1.vue

src > child-components > Component1.vue > { } script setup > handleSubmit

```vue
// --------------------
// functions
// --------------------
const handleSubmit = () => {
  const learning = {
    name: learningName.value,
    completed: false,
    // id:  uuid()
    id:learningStore.selectedLearning ? learningStore.selectedLearning.id : uuid()
  }
  console.log(learning);

  // appel de la 1ère action
  learningStore.actionCreateNewLearning(learning);

  // pour remettre le bouton d'origine et effacer les 2 pmettre à jour et annu
  if (learningStore.selectedLearning) {
    learningStore.selectedLearning = null  }

  // RAZ du FORM
  learningName.value = "";
}
```

```vue
//--------------------------
// MAJ du champ learningName
// --------------------------
learningStore.$subscribe(
  Codiumate: Options | Test this function
  (mutation) => {
    const selected = mutation.events.target.selectedLearning;

    if (selected) {
      console.log(selected.name);
      learningName.value = selected.name
    }
  }
);
// --------------------------
//  cancelUpdate
// --------------------------
const cancelUpdate = () => {
  // pour récréer la formation qu'on a effacé en commençant l'action editer
  learningStore.actionCreateNewLearning(learningStore.selectedLearning)
  learningStore.selectedLearning = null;
  learningName.value = "";
}
```

Aller plus loin avec le store PINIA

https://pinia.vuejs.org/core-concepts/state.html

https://pinia.vuejs.org/core-concepts/actions.html

TP : Supprimer une donnée du store

faire une action nommée « actionRemoveLearning »

```
// -- 2nde Action
actionRemoveLearning(data) {
    console.log(data);

    this.learnings = this.learnings.filter(
        (learn) => learn.id !== data.id
    );
```

TP : Sauvegarder les données du Store dans le local Storage

Persister les données avec le local Storage, faire une action nommée « actionStorage »

```
    },
    // -- 3ème Action : persister les données
    actionStorage(){
        localStorage.setItem('store_learnings', JSON.stringify(this.learnings))
    }
}
```

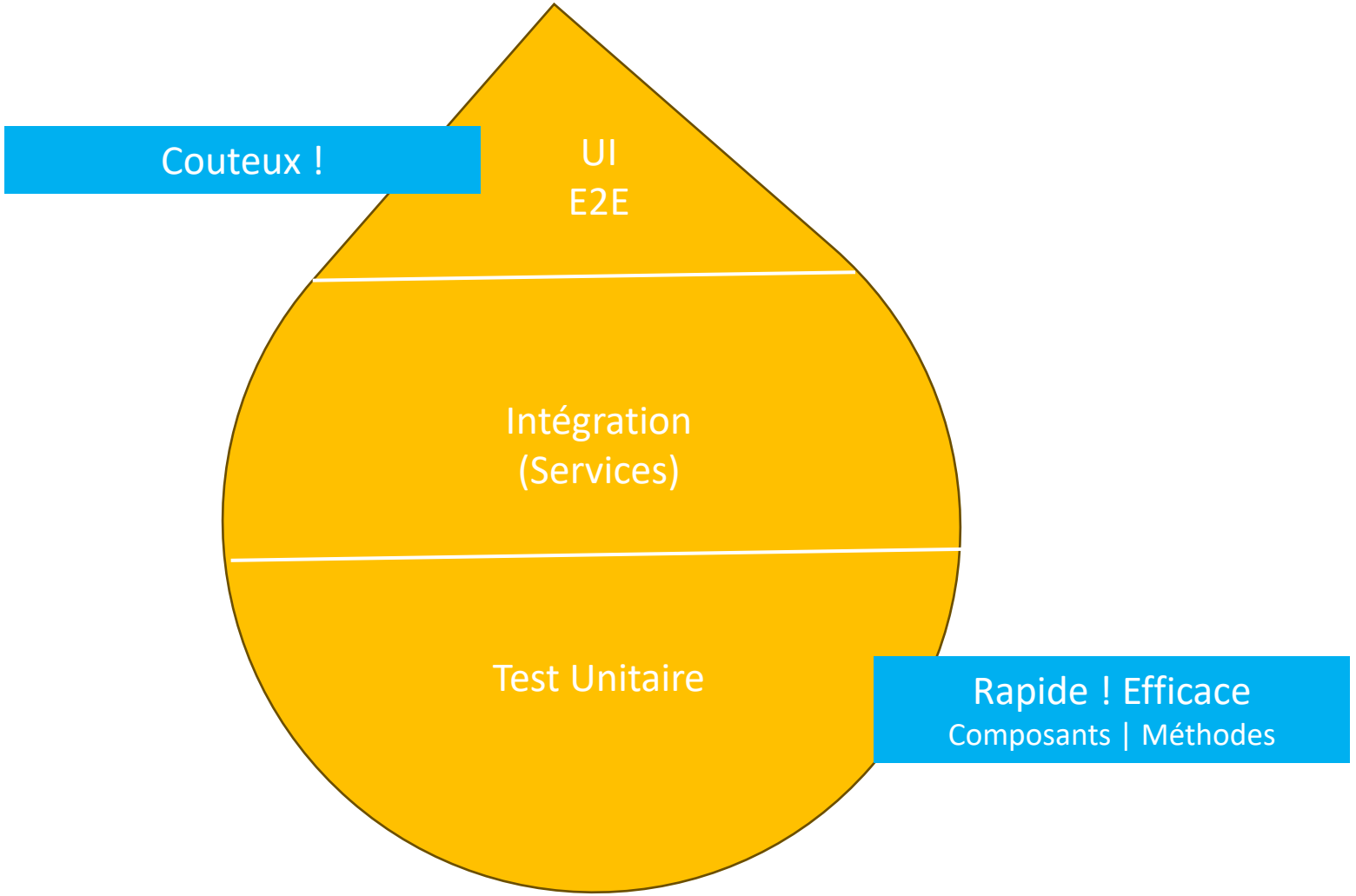
Les Tests

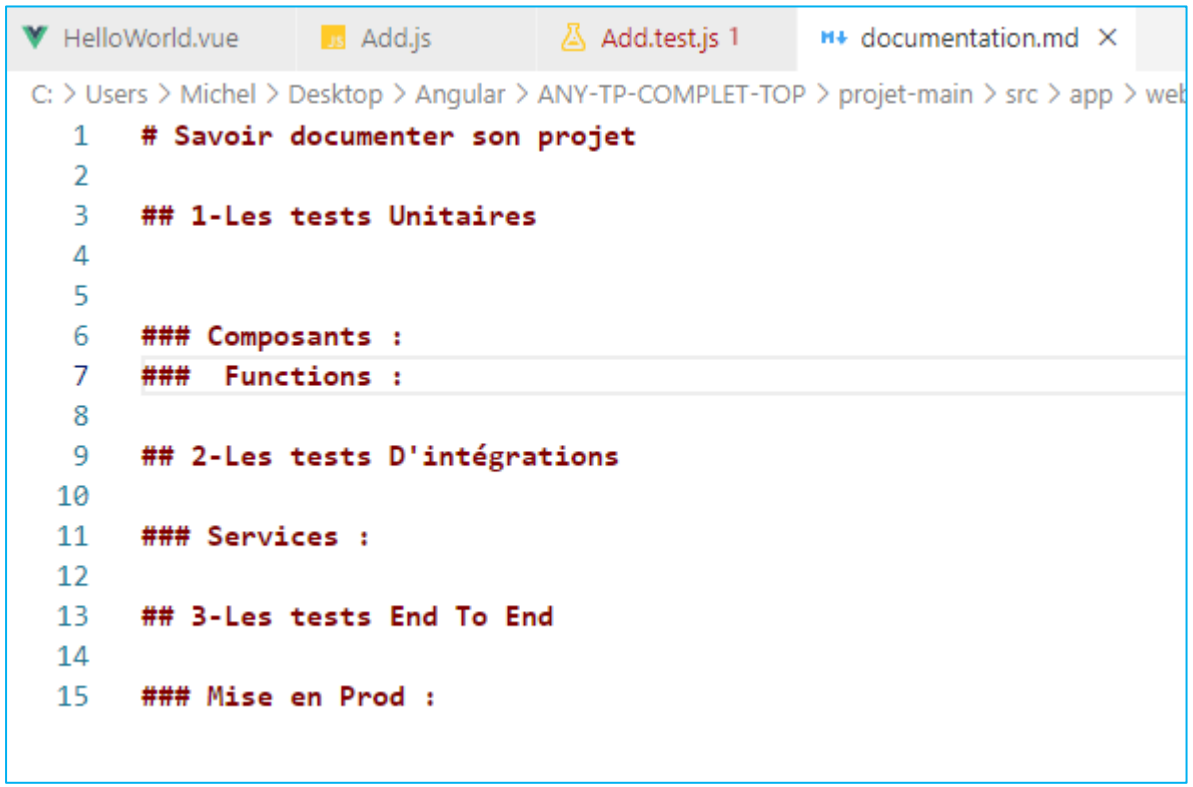
Le Test

Le test (ou le Test Driven Development TDD) permet de :

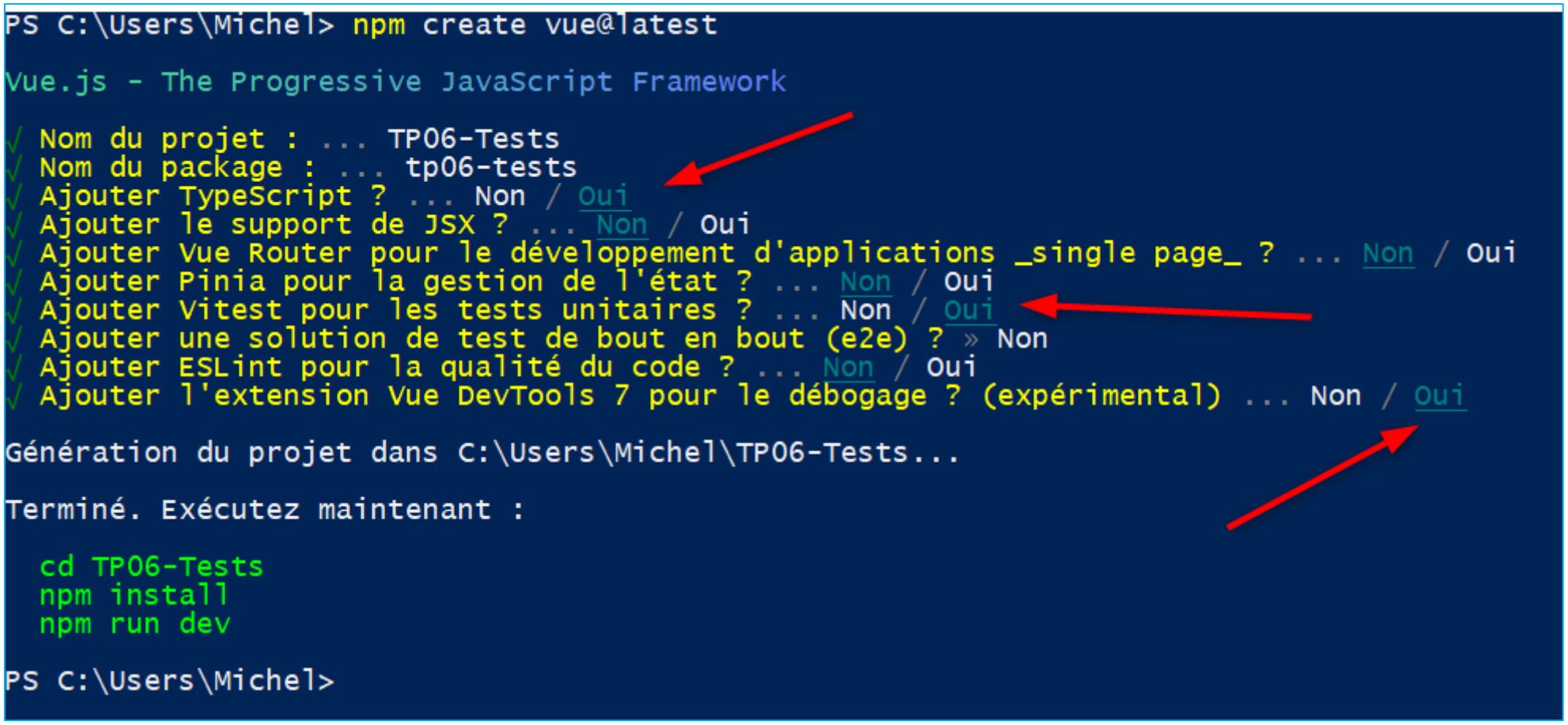
- Documenter le projet au fur et à mesure de sa création
- Faciliter les migrations entre versions majeures
- Simplifier la recherche d'erreurs
- Isoler l'objet de son contexte et de tester son intégration (Mock)

Le Test et la Documentation




HelloWorld.vue
Add.js
Add.test.js 1
documentation.md
C: > Users > Michel > Desktop > Angular > ANY-TP-COMPLET-TOP > projet-main > src > app > web
1 # Savoir documenter son projet
2
3 ## 1-Les tests Unitaires
4
5
6 ### Composants :
7 ### Functions :
8
9 ## 2-Les tests D'intégrations
10
11 ### Services :
12
13 ## 3-Les tests End To End
14
15 ### Mise en Prod :

Installation d’un projet avec Vitest



1er Exemple

```
import { describe, expect, test } from 'vitest'
import Add from './Add';

// const { Add } = useAdd();


// création d'un objet de test
Codiumate: Add more tests
describe('Test ADD', ()=> {

  // création d'une expectation avec la fonction it
  Codiumate: Add more tests
  it('Should get a correct result' , ()=> {

      const result = Add(3,2);
      expect(result).

  })

})
```

```
Arguments                    (property) Arguments: Assertion<any>
NaN
Throw
a
above
all
an
and
any
apply
approximately
arguments
at
```

Matchers

2nd Exemple : test d'un composable useFilter

Fetch.vue | useFilter.js

TP08-tests > src > composables > useFilter.js > useFilter

```
1  import { ref, computed } from "vue";
2
3
4  // ce composable permet de filtrer un affichage en fonction d'un motif passé
5  // searchStr
   Codiumate: Options | Test this function
6  export const useFilter = (elements, filters = ["title"]) => {
7    // le composable prend en paramètres une liste d'éléments
8    // et par défaut la prop sur laquelle on pose le filtre
9    // ici le title des films...
10
11   const searchStr = ref("");
12
13   const filteredElements = computed(() => {
14
15     console.log(searchStr.value)
16
17     return elements.value.filter((elt) => {
18
19       return filters.some((filter) => {
20         return elt[filter]
21           .toLowerCase()
22           .includes(searchStr.value.toLowerCase());
23       });
24     });
25   });
26
27   return {
28     searchStr,
29     filteredElements,
30     changeSearchStr: (texte) => (searchStr.value = texte),
31   };
32 };
```

Array.prototype.some()

Baseline Widely available

The `some()` method of `Array` instances tests whether at least one element in the array passes the test implemented by the provided function. It returns true if, in the array, it finds an element for which the provided function returns true; otherwise it returns false. It doesn't modify the array.

useFilter.spec.js

TP08-tests > src > __tests__ > useFilter.spec.js > describe("useFilter") callback > it("😀 Retouner les valeurs d'origine sans appliquer de

```
import { expect, describe, it } from "vitest";
import { useFilter } from "../composables/useFilter";

Codiumate: Add more tests
describe("useFilter", () => {

  Codiumate: Add more tests
  it("😀 Retouner les valeurs d'origine sans appliquer de filtre ...", () => {

    const elements = {value: [{ name: "1" }, { name: "2" }]};

    const { searchStr, filteredElements } = useFilter(elements, ['name']);

    expect(searchStr.value).toBe("");

    expect(filteredElements.value).toEqual(elements.value);
  });

  Codiumate: Add more tests
  it("😎 Appliquer un filtre ", () => {

    const elements = { value: [{ name: "1" }, { name: "2" }]};

    const { searchStr, filteredElements, changeSearchStr } = useFilter(elements, ['name']);

    changeSearchStr("1");

    expect(searchStr.value).toBe("1");

    expect(filteredElements.value).toEqual([{ name: "1" }]);
  });
});
```

node - TP08-tests

```
RERUN  src/__tests__/useFilter.spec.js x14

stdout | src/__tests__/useFilter.spec.js > useFilter > 😀 Retouner les valeur
ne sans appliquer de filtre ...


stdout | src/__tests__/useFilter.spec.js > useFilter > 😎 Appliquer un filtre
1

 ✓ src/__tests__/useFilter.spec.js (2)
   ✓ useFilter (2)
     ✓ 😀 Retouner les valeurs d'origine sans appliquer de filtre ...
     ✓ 😎 Appliquer un filtre

 Test Files  1 passed (1)
      Tests  2 passed (2)
   Start at  16:48:22
   Duration  92ms


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

npm run test

HelloWorld.vue | Add.js | Add.test.js | package.json

TP07-tests > package.json > { } scripts > test

```
{
  "name": "tp07-tests",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  ▷ Déboguer
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "test": "vitest"
  },
  "dependencies": {
    "bootstrap": "^5.3.3",
    "bootstrap-icons": "^1.11.3",
    "vue": "^3.4.21",
    "vue-router": "^4.3.0"
  },
  "devDependencies": {
```

PROBLÈMES | SORTIE | CONSOLE DE DÉBOGAGE | TERMINAL | PORTS

```
npm ERR! A complete log of this run can be found in: C:\Users\Mich
PS C:\Users\Michel\Desktop\Formation-Vue> cd .\TP07-tests\
PS C:\Users\Michel\Desktop\Formation-Vue\TP07-tests> npm run test

> tp07-tests@0.0.0 test

 RERUN  src/components/__tests__/functions/Add.test.js x2

 ✓ src/components/__tests__/functions/Add.test.js (1)
   ✓ Test ADD (1)
     ✓ Should get a correct result

 Test Files  1 passed (1)
      Tests  1 passed (1)
   Start at  14:33:50
   Duration  34ms


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

Aller plus loin …

Aller plus loin …

Le SSR : Server Side Rendering
https://fr.vuejs.org/guide/scaling-up/ssr

L'avantage du SSR par rapport à une application monopage SPA côté client :

- **Temps d'affichage plus rapide** : cela est plus important sur une connexion internet ou sur des périphériques lents. Le rendu côté serveur permet qu'une page soit rendue plus rapidement car elle n'a pas besoin d'attendre le téléchargement et l'exécution du JavaScript. La récupération des données se fait également côté serveur, ce qui peut améliorer la vitesse de connexion à la base de données. Cela peut entraîner une amélioration de l'expérience utilisateur et des métriques Core Web Vitals, en particulier pour les applications où le temps d'affichage est important pour le taux de conversion.
- **Modèle mental unifié** : vous pouvez utiliser le même langage et le même modèle mental déclaratif orienté composant pour développer votre application entière, au lieu de sauter sans arrêt entre un système de modèle de backend et un framework frontend.
- **Meilleur référencement** : les robots d'indexation de moteur de recherche verront directement la page entièrement rendue.

SSR ou CSR

Le « **S**erver **S**ide **R**endering » existe depuis les années 1990 !
La page est rendue depuis le back et est affichée dans le navigateur du client.

Début 2010, les premiers « Webpack » et les « **S**ingle **P**age **A**pplication » font leur arrivée dans le monde du développement Web.

On parle alors de **C**lient **S**ide **R**endering, puisque le projet final contient seulement une page index.html qui fait elle même appel à des « chunk files javascript » (*des fichiers JavaScripts buildés*).

SSR ou CSR

Pourquoi revient-on depuis peu au SSR ?

- Il y aura de meilleure performance en SEO, bien que les moteurs dont Google garantit que les pages CSR seront correctement indexées, cela nécessite plus de temps mais les moteurs peuvent le faire !

- Le temps de chargement du contenu est un peu plus rapide, car on doit attendre que le contenu total de l'applis soit chargé (NB: le lazy loading solutionne cette problématique …)

- React, Vue et Angular l'implémentent facilement !

SSR ou CSR

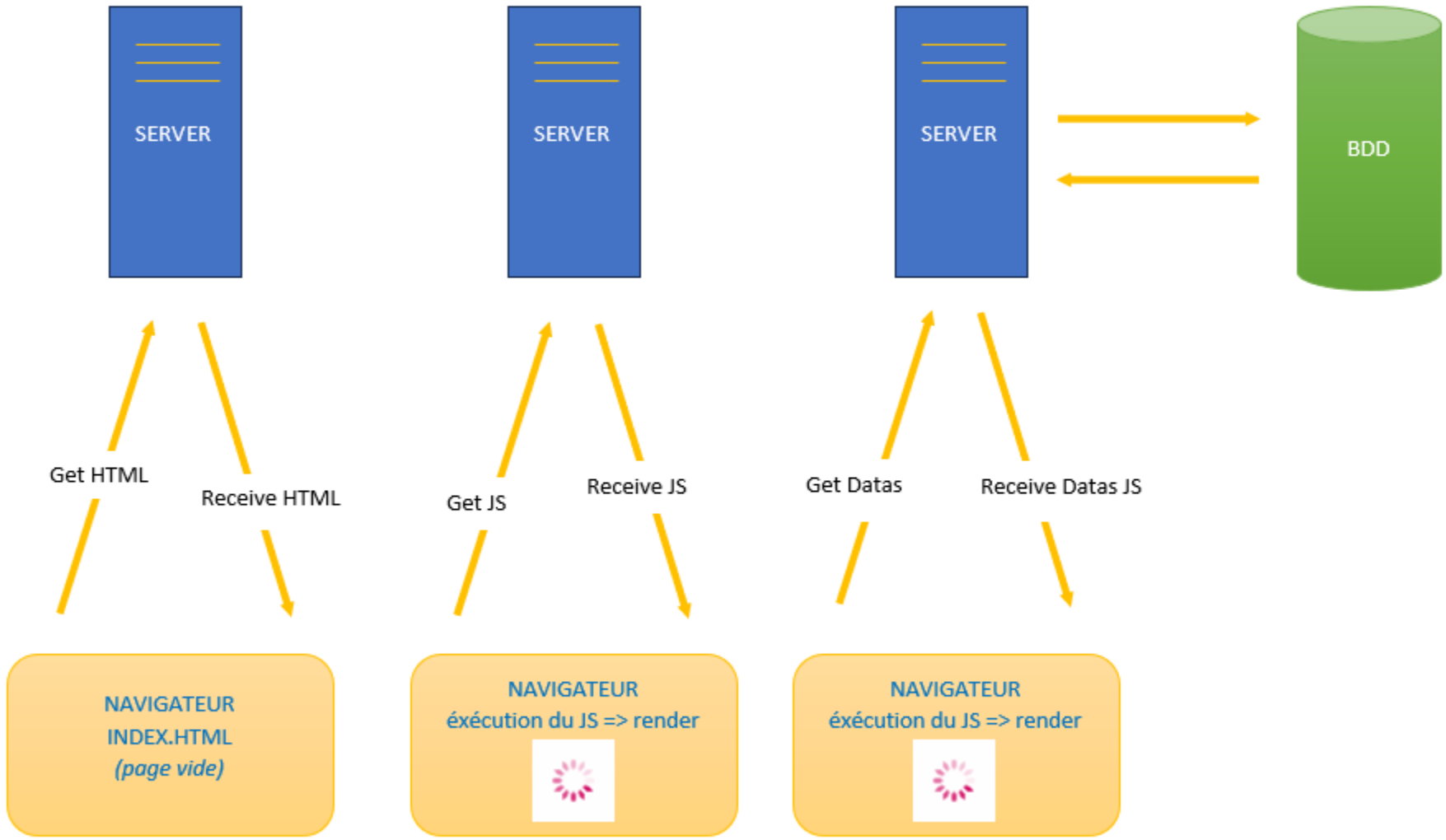
Pourquoi revient-on depuis peu au SSR ?

- Il y aura de meilleures performance en SEO, bien que les moteurs dont Google garantit que les pages CSR seront correctement indexées, cela nécessite plus de temps mais les moteurs peuvent le faire !
- Le temps de chargement du contenu est un peu plus rapide, car on doit attendre que le contenu total de l'applis soit chargé (NB: le lazy loading solutionne cette problématique …)
- React, Vue et Angular l'implémentent facilement !

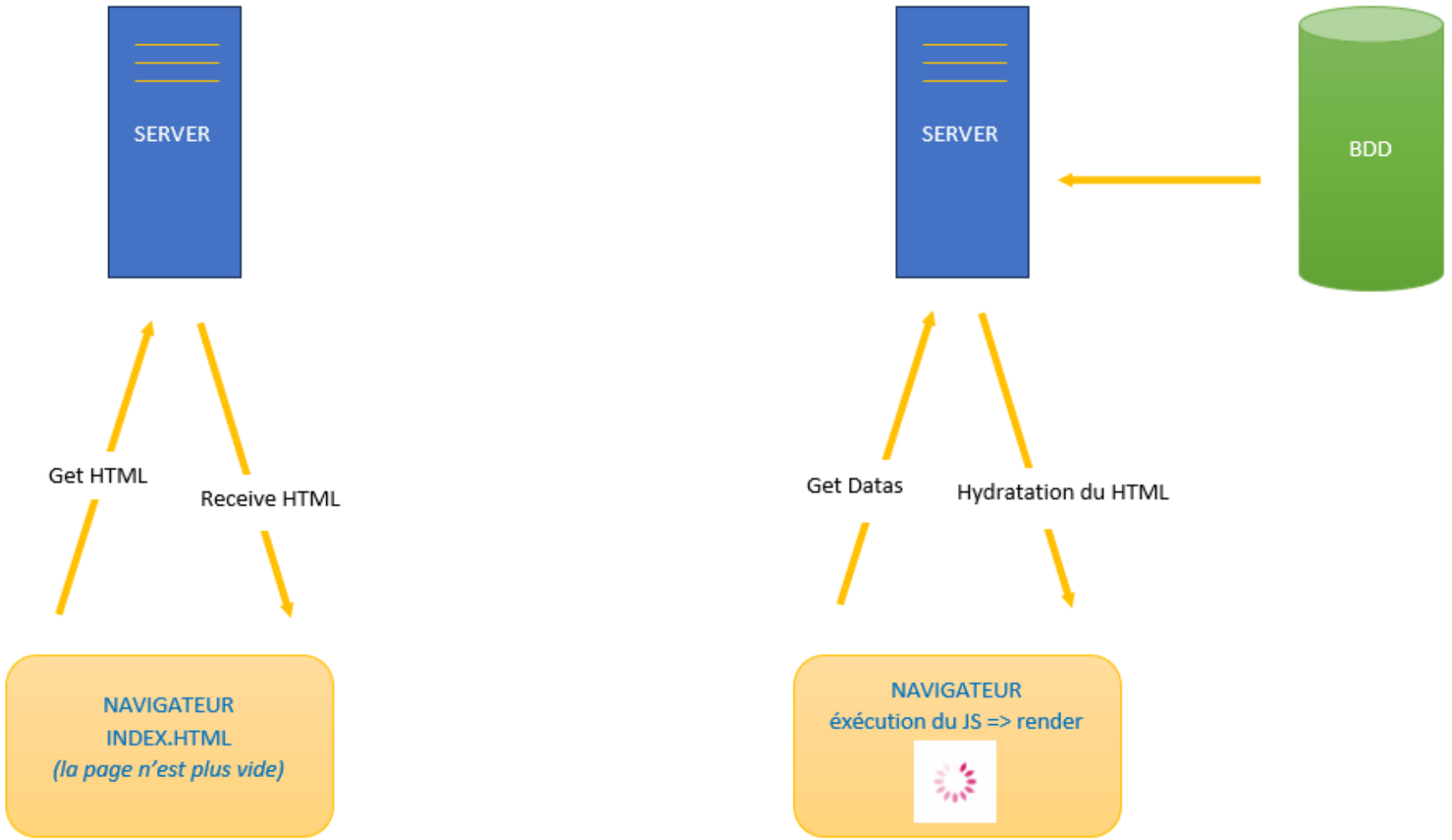
Inconvénients :

- La charge « back » est évidement plus couteuse
- Il est nécessaire d'implémenter Node.js car ce n'est plus un simple hébergement web statique !

CSR : Client-Side Rendering | Single Page Application : Le rendu se fait côté client



SSR : Server-Side Rendering



Mise en pratique

TP guidé : installer « express » et mettre en place une configuration à base de Node.
https://fr.vuejs.org/guide/scaling-up/ssr

I

Aller plus loin …

- Compilation, optimisation, mise en production sur un serveur
  TP : Créer un build, rechercher les fichiers de « lazy-loading »

Aller plus loin …

- Configuration de Vite :
  Comprendre les variables du fichier « vite.config.js »

- https://vite.dev/config/

- TP : trouver la directive qui permet de changer le dossier de build par défaut « dist »

vite.config.js

TP06-store-pinia > vite.config.js > ...

```
1   import { fileURLToPath, URL } from 'node:url'
2
3   import { defineConfig } from 'vite'
4   import vue from '@vitejs/plugin-vue'
5   import vueDevTools from 'vite-plugin-vue-devtools'
6
7   // https://vite.dev/config/
8   export default defineConfig({
9     plugins: [
10      vue(),
11      vueDevTools(),
12    ],
13    resolve: {
14      alias: {
15        '@': fileURLToPath(new URL('./src', import.meta.url))
16      },
17    },
18  })
19
```

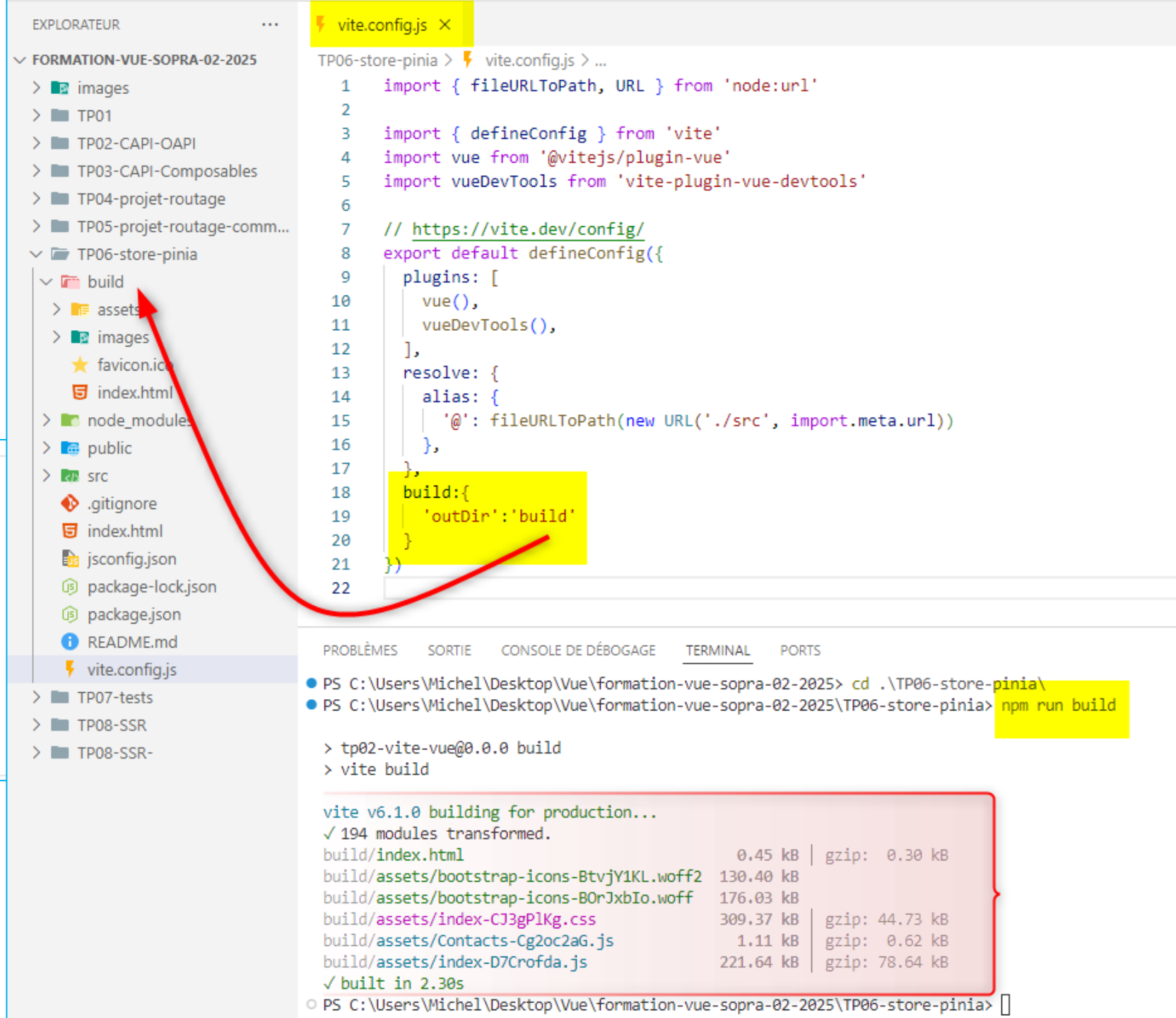
Aller plus loin ...

- Solution

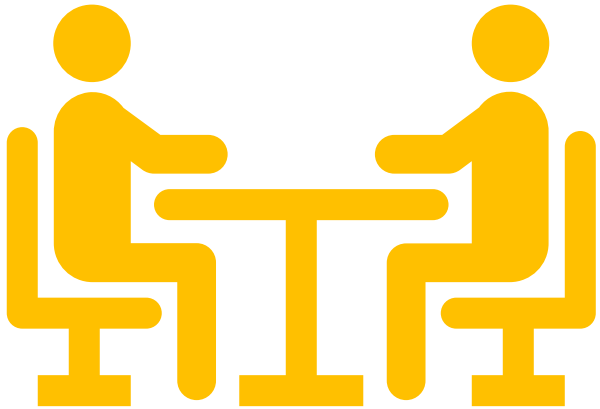
build.outDir

- **Type:** `string`
- **Default:** `dist`

Specify the output directory (relative to project root).



Bonne fin de formation



Sparks vous remercie