TP2: Vue.js

Ibrahim ALAME

4 décembre 2025

1 Initialisation du projet

1.1 Initialisation du projet

Placez-vous où vous voulez pour créer le nouveau projet en ouvrant un terminal. Par exemple dans HOME :

```
cd
```

Créez un nouveau projet Vue.js :

```
npm init vue@latest
```

1. Par défaut, le nom est prérempli avec vue-project mais vous pouvez bien sûr le changer par exemple par boutique.
2. La deuxième question est sur l'utilisation de TypeScript :

   ```
   Add TypeScript? ... No / Yes
   ```

   Comme nous l'avons vu, choisissez oui.
3. Ensuite répondez non pour JSX. Nous n'utiliserons par JSX qui est un langage de template React.
4. Répondez non pour Vue Router, Pinia, Vitest et Cypress car nous les verrons plus tard dans la formation.
5. Répondez oui à ESLint, qui permet de contrôler la qualité du code et répondez oui à Prettier pour le formatage du code.

Vous devez en être là :

```
17:32:32 ✔ erwan:~/code$ npm init vue@latest
Need to install the following packages:
  create-vue@latest
Ok to proceed? (y) yes

Vue.js - The Progressive JavaScript Framework

✔ Project name: … dymaproject
✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit Testing? … No / Yes
✔ Add Cypress for both Unit and End-to-End testing? … No / Yes
✔ Add ESLint for code quality? … No / Yes
✔ Add Prettier for code formatting? … No / Yes

Scaffolding project in /home/erwan/code/dymaproject...

Done. Now run:
```

Allez dans le dossier :

```
cd boutique
```

Bien sûr adaptez boutique avec le nom que vous avez donné au projet.
Installez les dépendances :

```
npm install
```

Lancer WebStorm et ouvrir le projet.

1.2 Inclusion d'une police

Nous modifions index.html pour charger une police depuis Google fonts :

```
<head>
....
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
      href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
      rel="stylesheet"
    />
</head>
```

1.3 Modification de App.vue

Nous enlevons tout le code par défaut et ajoutons l'utilisation de scss :

```
<script setup lang="ts"></script>

<template>
  <h1>Bonjour le monde !</h1>
</template>

<style lang="scss">
@use './assets/base.scss' as *;
</style>
```

1.4 Installation de Sass

Installez Sass en dépendance de développement :

```
npm i -D sass
```

1.5 Modification de assets/base.css

Renommez le fichier base.css en base.scss car nous utilisons Sass et mettez pour le moment :

```
:root {
  --font-family: 'Roboto', sans-serif;
}

body {
  font-family: var(--font-family);
}
```

1.6 Installation de l'extension pour navigateur Vue

Installez l'extension Chrome pour Vue.js ou l'extension Firefox suivant votre navigateur. Le nom de l'extension est Vue.js devtools.

2 Mise en place du style

2.1 Modification deassets/base.scss

Utilisez le style de base suivant (cf site : flatuicolors.com) :

```
:root {
  --primary-1: #3498db;
  --primary-2: #2980b9;
  --danger-1: #e74c3c;
  --danger-2: #c0392b;
  --success-1: #2ecc71;
  --success-2: #27ae60;
```

```
  --gray-1: #f6f6f6;
  --gray-2: #ddd;
  --gray-3: #34495e;
  --text-color: #444;
  --text-primary-color: #ffffff;

  --border: 1px solid var(--gray-2);
  --border-radius: 4px;

  --font-family: 'Roboto', sans-serif;
}
// reset
* {
  box-sizing: border-box;
}
h1,h2,h3,h4 {
  margin: 0;
}
ul {
  list-style: none;
  padding: 0;
}
img {
  max-width: 100%;
}
a {
  color: var(--text-color);
  text-decoration: none;
}
body {
  min-height: 100vh;
  padding: 0;
  margin: 0;
  font-family: var(--font-family);
  color: var(--text-color);
  background-color: var(--gray-1);
}
// flex
.d-flex {  display: flex; }
.flex-row {  flex-direction: row; }
.flex-column { flex-direction: column; }
.justify-content-center {  justify-content: center; }
.align-items-center {  align-items: center; }
.flex-fill {  flex: 1 1 auto; }
// padding
.p-10 {  padding: 10px; }
.p-20 {  padding: 20px; }
.p-30 {  padding: 30px; }
// margin
.m-10 {  margin: 10px; }
.m-20 {  margin: 20px; }
.m-30 {  margin: 30px; }
```

Si vous ne maîtrisez pas certaines propriétés, n'hésitez pas à revoir les chapitres correspondants dans la formation HTML & CSS.

2.2 Création du fichier assets/debug.scss

Créez les classes pour le débug plus facile du CSS :

```
.b1 {  background-color: red; }
.b2 {  background-color: blue; }
.b3 {  background-color: yellow; }
.b4 {  background-color: green; }
.b5 {  background-color: purple; }
```

2.3 Modification de App.vue

N'oubliez pas de modifier App.vue pour l'importer :

```
<script setup lang="ts"></script>

<template>
  <h1>Bonjour le monde !</h1>
</template>

<style lang="scss">
@use './assets/base.scss' as *;
@use './assets/debug.scss' as *;
</style>
```

Exemple :

```
<template>
  <h1 class="b1">Bonjour le monde !</h1>
</template>
```

3 Mise en page globale

3.1 Création de l'architecture

Créez un dossier components dans le dossier src.
Dans ce dossier, créez les fichiers Footer.vue, Header.vue, Shop.vue et Cart.vue.

3.1.1 Composant Footer.vue

Mettez dans le composant :

```
<script setup lang="ts"></script>

<template>
  <footer>
```

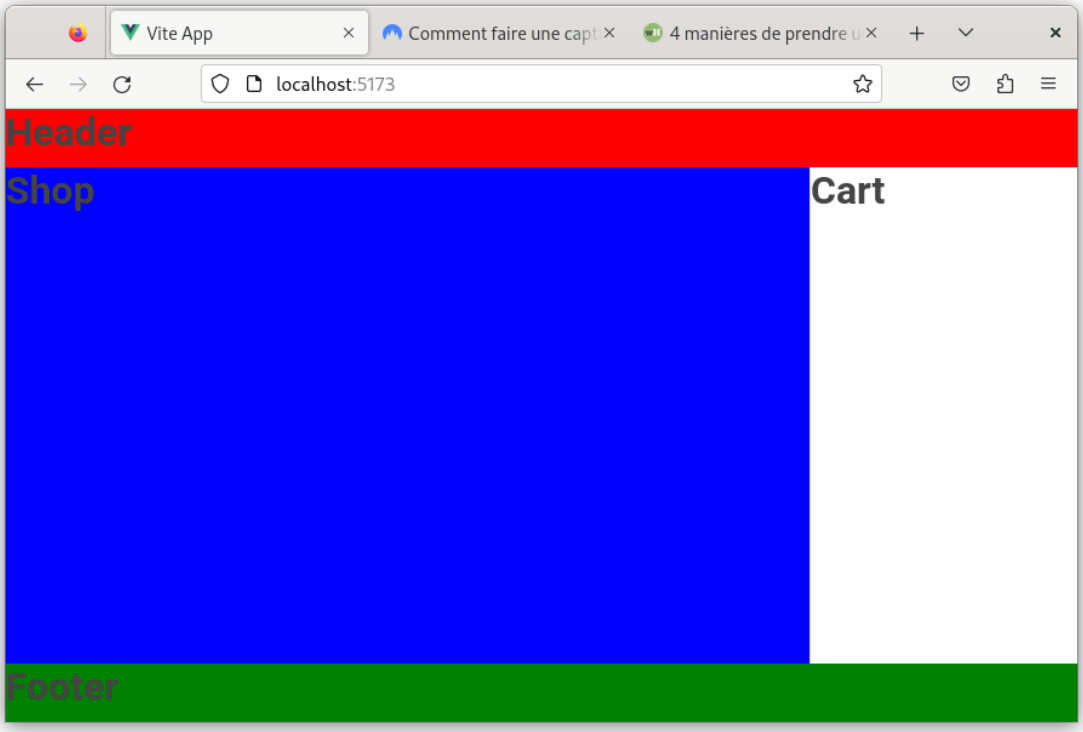
```
    <h1>Footer</h1>
  </footer>
</template>

<style lang="scss" scoped></style>
```

Faire de même pour les autres composants.

Notez que nous utilisons scoped pour limiter la portée des styles déclarés dans ces composants.

3.1.2 Modification de App.vue

Modifier notre composant racine pour importer et utiliser les composants que nous avons créés. La mise en page est définie dans une div par une grille css :



4 Mise en page boutique et panier

4.1 Architecture

1. Dans le dossier `components` créez un dossier `Cart` et un dossier `Shop`. Déplacez le composant `Cart.vue` dans `Cart` et `Shop.vue` dans `Shop`. Vérifier les chemins d'imports dans `App.vue`.
2. Dans le dossier `Cart`, créez les composants `CartProductList.vue` et `CartProduct.vue`.
3. Même choise dans le dossier `Shop`, créé deux composants `ShopProductList.vue` et `ShopProduct.vue`.

4.2 Modification de Shop.vue

Nous allons importer et utiliser les nouveaux composants relatifs à la liste des produits :

```
<script setup lang="ts">
import ShopProductList from './ShopProductList.vue';
</script>

<template>
  <div>
    <ShopProductList />
  </div>
</template>

<style lang="scss" scoped></style>
```

Dans ce composant nous plaçons le composant qui va être responsable de gérer la liste des produits.

4.3 Modification de ShopProductList.vue

Dans ce composant nous allons utiliser plusieurs instances de notre composant `ShopProduct` et les disposer en utilisant une grille CSS en 4 colonnes et de `300px` et de gap `20px`.

4.4 Modification de ShopProduct.vue

Dans ce composant nous affichons juste un titre pour le moment :

```
<script setup lang="ts"></script>

<template>
  <div class="b5">
    <h1>Shop Product</h1>
  </div>
</template>

<style lang="scss" scoped></style>
```

4.5 Modification de Cart.vue

Dans ce composant nous utilisons le composant `CartProductList` responsable de gérer l'affichage des produits du panier. Dans le template de `Cart.vue` insérer dans une division div de padding 20px les deux éléments :

- une balise `h2` dont le `margin-bottom` est `10px` contenant le titre : Panier. (on pourra créer une classe `mb-10` dans assets/base.scss)

  ```
  .mb-10 {
    margin-bottom: 10px;
  }
  ```

- un composant `CartProductList`.

4.6 Modification de CartProductList.vue

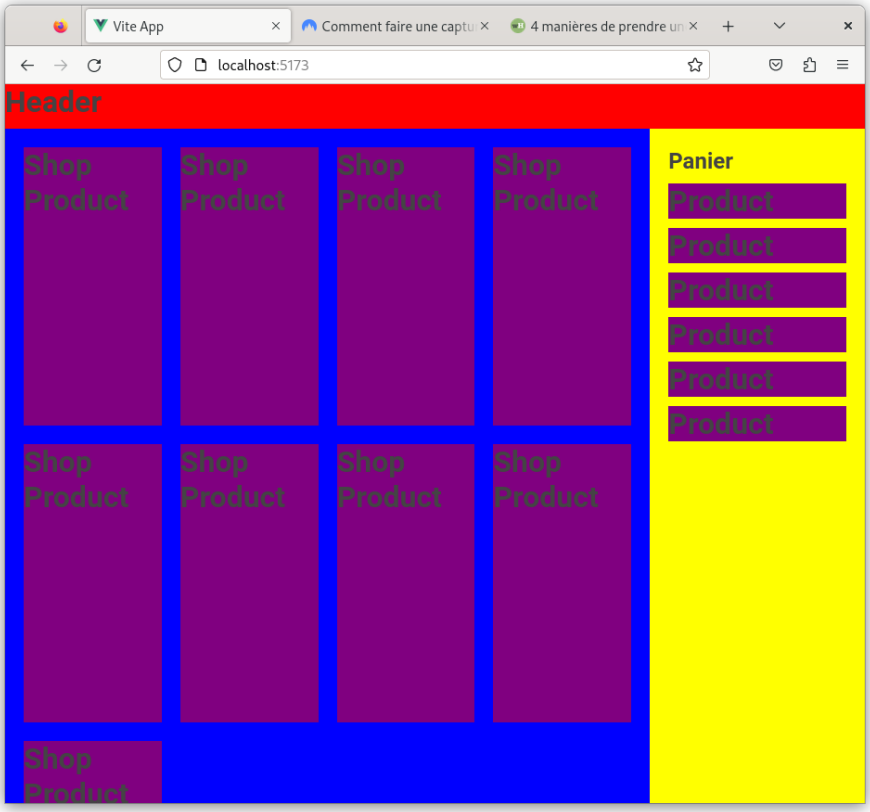
Dans ce composant nous allons utiliser plusieurs instances du composant `CartProduct` qui sont les produits dans le panier. Nous utilisons des boîtes flexibles pour positionner les produits.

```
<template>
  <div class="d-flex flex-column">
    <CartProduct />
    <CartProduct />
    <CartProduct />
    <CartProduct />
    <CartProduct />
    <CartProduct />
  </div>
</template>
```

4.7 Modification de CartProduct.vue

Dans ce composant nous affichons pour le moment simplement un titre :

```
<template>
  <div class="mb-10 b5">
    <h1>Product</h1>
  </div>
</template>
```


Vite App
Comment faire une captu
4 manières de prendre un
localhost:5173
Header
Shop Product
Shop Product
Shop Product
Shop Product
Shop Product
Shop Product
Shop Product
Shop Product
Shop Product
Panier
Product
Product
Product
Product
Product
Product