Rapport de Développement Web

Projet Bloom Vessels

Technologies : HTML5, Tailwind CSS v4, JavaScript ES6+
Focus : Performance, Accessibilité (a11y), SEO

Réalisé par :
Yahya El Omari

Encadré par :
ABDELMAJID BENCHRIF

https://sprint-3-4.vercel.app/
13 février 2026

Table des matières

1 Introduction et Vision

Ce document retrace l'ingénierie logicielle derrière la création de **"Bloom Vessels"**, une vitrine e-commerce moderne pour des pots de fleurs artisanaux.

L'objectif n'était pas seulement de reproduire une maquette, mais de construire une application web *résiliente*, *rapide* et *accessible*. Nous avons adopté une approche **"Progressive Enhancement"** : une base HTML solide, stylisée avec Tailwind CSS, et enrichie dynamiquement par JavaScript.

2 Phase 1 : Déconstruction du Design (Thinking Process)

Avant d'écrire la moindre ligne de code, nous avons décomposé l'interface utilisateur (UI) en composants logiques.

2.1 Stratégie Responsive : L'Approche Mobile-First

Notre processus mental suit une règle absolue : **"Construire pour le mobile d'abord"**. Sur un petit écran, le contenu est roi et l'espace est rare.

1. **Le Flux Naturel (Flow)** : Par défaut, tous les éléments HTML (‘div‘, ‘section‘) sont des blocs qui s'empilent verticalement. C'est le comportement idéal pour mobile. Nous utilisons ‘flex-col‘ pour forcer cet empilement si nécessaire.

2. **L'Expansion Progressive** : Ce n'est qu'une fois que l'écran s'agrandit que nous introduisons la complexité. Grâce aux préfixes de breakpoints Tailwind (‘md :‘ pour tablette, ‘lg :‘ pour desktop), nous "cassons" la colonne unique.

```
<!-- Mobile : 1 colonne | Desktop : 2 colonnes -->
<div class="flex flex-col md:flex-row">
```

2.2 Le "Couteau Suisse" Tailwind : Classes Indispensables

Pour orchestrer cette responsivité sans écrire de CSS personnalisé, nous nous appuyons massivement sur ces classes clés :

- **Flexbox & Grid** :
  - ‘flex‘, ‘items-center‘, ‘justify-between‘ : Pour aligner navigation et boutons.
  - ‘grid‘, ‘grid-cols-1 md :grid-cols-3‘ : Pour transformer une liste d'articles verticale en une grille élégante sur grand écran.
- **Dimensions Relatives** :
  - ‘w-full‘ : L'élément prend toute la largeur (Mobile).
  - ‘md :w-1/2‘ : L'élément ne prend plus que 50% (Desktop), parfait pour une disposition Texte/Image.
- **Visibilité Conditionnelle** :
  - ‘hidden md :block‘ : Caché sur mobile, visible sur desktop (ex : liens de navigation).
  - ‘block md :hidden‘ : Visible sur mobile, caché sur desktop (ex : bouton menu hamburger).
- **Espacement (Rhythm)** :
  - ‘gap-4 md :gap-8‘ : Espacement plus serré sur mobile, plus aéré sur desktop.
  - ‘px-6 py-12‘ : Marges internes généreuses pour laisser respirer le contenu.

3 Phase 2 : Architecture HTML et Sémantique

Le HTML n'est pas juste une structure, c'est l'API d'accessibilité du site.

3.1 SEO et Métadonnées Avancées

Nous avons intégré un bloc ‘<head>‘ riche pour maximiser la découvrabilité :

Exemple d’Optimisation SEO

```
<meta property="og:type" content="website">
<meta property="og:title" content="Bloom Vessels">
<meta name="description" content="Discover handcrafted...">
<link rel="canonical" href="https://bloomvessels.com/">
```

Ces balises Open Graph (OG) garantissent que les liens partagés sur LinkedIn ou Twitter affichent une image de prévisualisation et un titre correct, augmentant le taux de clic (CTR).

3.2 Performance des Images (Core Web Vitals)

La gestion des images a été pensée pour le score **LCP (Largest Contentful Paint)** :

— **Image Hero** : Utilise ‘<picture>‘ pour servir du format AVIF (plus léger que WebP) et l’attribut ‘loading="eager"‘ + ‘fetchpriority="high"‘. C’est l’élément visuel critique.

— **Images Secondaires** : Toutes les autres images ont ‘loading="lazy"‘, différant leur chargement, économisant ainsi la bande passante.

4 Phase 3 : Styling Industriel avec Tailwind CSS v4

Nous avons utilisé la toute dernière version de Tailwind (v4) pour sa performance.

4.1 Configuration ”Input-First”

Plus de fichier JS complexe. Tout est centralisé dans le CSS via la directive ‘@theme‘ :

Avantage Majeur : Les variables CSS ‘–color-brand-pink‘ (maintenant Rouge) deviennent accessibles partout.

4.2 Responsivité et Dark Mode

Le site est conçu **Mobile First**. Pour le mode sombre, nous utilisons la classe ‘dark‘ sur la balise ‘<html>‘.

— ‘bg-white‘ devient ‘dark :bg-gray-900‘.

— ‘text-slate-900‘ devient ‘dark :text-white‘.

5 Phase 4 : Ingénierie JavaScript

L’interactivité est gérée par une architecture modulaire en trois fichiers, séparant clairement les données, la logique de rendu et les interactions.

5.1 1. ‘data.js‘ : Source de Vérité

Ce fichier exporte des constantes (‘articleData‘, ‘faqData‘) contenant des **objets JSON** purs. *Pourquoi ?* Cela découple le contenu du code. Un rédacteur peut ajouter un article en ajoutant simplement un objet dans le tableau, sans risquer de casser le HTML. Nous avons ajouté un flag ‘enabled : true‘ pour permettre la désactivation temporaire de contenu sans suppression.

5.2 2. ‘generator.js‘ : Le Constructeur de DOM

Ce script injecte le contenu au chargement. Il contient des mécanismes de sécurité robustes :

— **Safe Guards** : Utilisation de blocs ‘try-catch‘ pour qu’une erreur dans le carrousel ne brise pas le reste de la page.
— **Fallbacks** : Si une image manque, une image par défaut grise est injectée grâce à ‘onerror‘.
— **Pagination Côté Client** : Une logique de ‘slice()‘ permet d’afficher les articles par paquets de 3, simulant une pagination serveur pour une fluidité instantanée.

5.3 3. ‘script.js‘ : Le Gestionnaire d’Événements

Ce fichier orchestre l’expérience utilisateur (UX). Un point technique clé est la **Délégation d’Événements** pour l’accordéon FAQ.

Pro Tip : Event Delegation

Au lieu d’attacher un écouteur sur chaque bouton (qui n’existent peut-être pas encore au chargement), nous attachons un seul écouteur sur ‘document‘. Lors d’un clic, nous vérifions si la cible est un ‘.accordion-btn‘. Cela permet de gérer des éléments générés dynamiquement sans ré-attacher des listeners, améliorant la performance mémoire.

6 Résultat Visuel

Nous avons intégré des visuels haute qualité pour refléter l’identité "Artisan" de la marque.

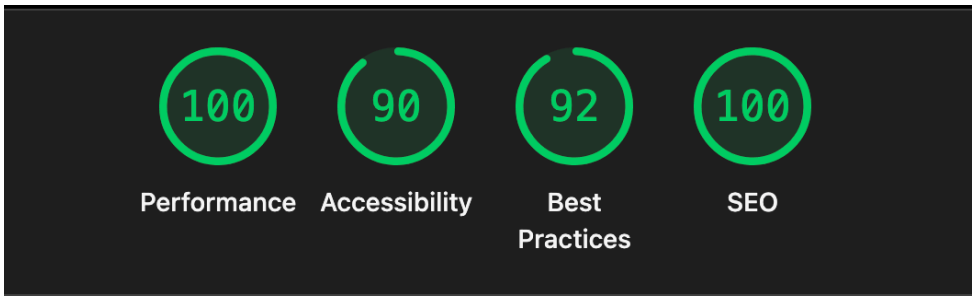

FIGURE 1 – Ambiance visuelle de la section Héro

7 Conclusion

Le projet **Bloom Vessels** démontre une maîtrise complète de la stack Front-End moderne. L’alliance de la sémantique HTML5, de la puissance de Tailwind CSS et d’un JavaScript modulaire et défensif aboutit à un produit final qui est non seulement esthétique, mais techniquement solide, maintenable et évolutif.