

Fondamentaux du Web

HTML, CSS, JavaScript

Lab 3 — CSS en pratique

Styliser Jàng Afrig avec une palette panafricaine

Licence 1 DUT1-Informatique — 2025–2026

Dr. El Hadji Bassirou TOURÉ
École Supérieure Polytechnique
Département Génie Informatique
Université Cheikh Anta Diop de Dakar

Informations pratiques

Type	Lab guidé
Séance	S5 — premier Lab CSS
Prérequis	Lab 2 terminé (5 pages HTML poussées avec tag v0.2) + CM2
Validation	5 pages stylisées + tag v0.3 sur GitHub
Rapport	1-2 pages : ce que j'ai fait + difficultés + lien repo

Objectifs du Lab

À la fin de ce Lab, vous aurez :

1. Créé un fichier **style.css** et l'avoir lié aux 5 pages HTML
2. Mis en place un **reset CSS** et **box-sizing: border-box**
3. Importé **2 polices Google Fonts** (titres + corps)
4. Défini une **palette panafricaine** (vert, jaune, rouge + neutres)
5. Stylisé le **header** et la **navigation Flexbox**
6. Créé des **cartes** pour les modes de jeu
7. Stylisé des **boutons** avec effet **:hover**
8. Stylisé le **tableau** du palmarès (bordures + lignes alternées)
9. Posé le tag v0.3 sur GitHub

Table des matières

1 Avant de coder	2
1.1 Là où on en est	2
1.2 Ce qu'on va construire aujourd'hui	2
2 Étape 1 : Créer style.css et le lier aux 5 pages	3
2.1 Créer le fichier	3
2.2 Lier style.css aux 5 pages	3

3	Étape 2 : Reset CSS et box-sizing	3
4	Étape 3 : La palette panafricaine	5
4.1	Le choix des couleurs	5
4.2	Variables CSS : un seul endroit pour les couleurs	5
5	Étape 4 : Importer les polices Google Fonts	6
5.1	Choisir 2 polices	6
5.2	Méthode 1 : via le <link> HTML (recommandé)	6
5.3	Utiliser les polices dans style.css	6
6	Étape 5 : Styliser le header	7
7	Étape 6 : Styliser la nav avec Flexbox	7
7.1	Pourquoi le « » a disparu ?	8
8	Étape 7 : Cartes pour les modes de jeu	9
8.1	Modifier le HTML de index.html	9
8.2	Ajouter le CSS	9
9	Étape 8 : Boutons stylés	11
9.1	Ajouter une classe au bouton	11
9.2	Styler les boutons en CSS	11
10	Étape 9 : Tableau du palmarès	13
11	Étape 10 : Footer	13
12	Vérification visuelle	15
12.1	Naviguez entre les 5 pages	15
12.2	DevTools : modifier le CSS en direct	15
12.3	Ce qui ne marche pas encore (et c'est normal)	15
13	Exercices d'application	16
14	Aide-mémoire	17
14.1	Propriétés CSS vues dans ce Lab	17
14.2	Variables et pseudo-classes	17
14.3	Structure type d'un fichier style.css	17
15	Routine Git de fin de Lab	18
16	Rapport à produire	18
17	Résumé et transition	19
17.1	Ce que vous avez accompli	19
17.2	Ce qu'on fera au Lab 4	19

1 Avant de coder

1.1 Là où on en est

Au Lab 2, vous avez construit les 5 pages HTML de Jàng Afrig avec une structure sémantique propre. Mais visuellement, c'est encore austère : texte noir sur fond blanc, liens bleus soulignés, tableau sans bordures.

1.2 Ce qu'on va construire aujourd'hui

Un **fichier** `style.css` unique, lié aux 5 pages, qui les transforme en application visuellement attractive.

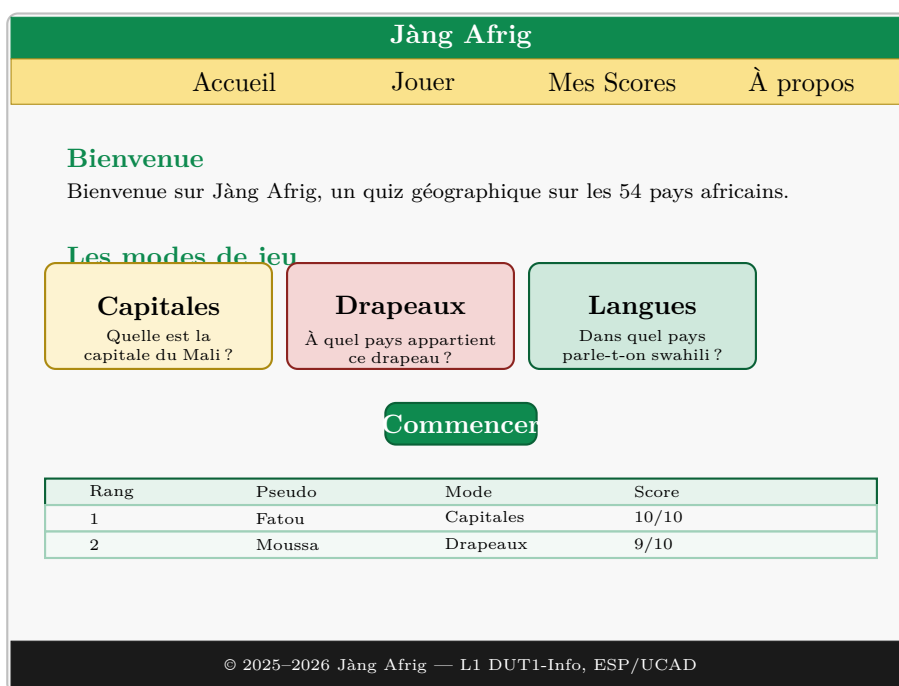


Fig. 1 : Aperçu de l'objectif visuel à atteindre dans ce Lab.

Header vert, nav jaune, cartes colorées (vert/jaune/rouge), bouton vert, palmarès avec lignes alternées, footer noir. On y va.

2 Étape 1 : Créer style.css et le lier aux 5 pages

2.1 Créer le fichier

Dans VS Code, à la racine de votre repo `jàng-afrig` :

1. Clic sur l'icône « Nouveau fichier » dans l'Explorateur
2. Nommez-le : `style.css`
3. Laissez-le vide pour l'instant (on va le remplir étape par étape)

2.2 Lier style.css aux 5 pages

Dans **chaque** fichier HTML (`index.html`, `quiz.html`, `resultat.html`, `scores.html`, `apropos.html`), ajoutez cette ligne dans le `<head>` :

```
<head>
  <meta charset="UTF-8">
  <title>Jàng Afrig</title>
  <link rel="stylesheet" href="style.css">
</head>
```

Enregistrez les 5 fichiers. Pour l'instant, comme `style.css` est vide, rien ne change visuellement. C'est attendu.

Le piège du copier-coller

Si vous oubliez d'ajouter le `<link>` sur *une seule* page, celle-ci restera moche pendant que les autres seront stylées. Vérifiez bien les 5.

3 Étape 2 : Reset CSS et box-sizing

Chaque navigateur applique des styles par défaut différents. Pour démarrer sur une base propre, on ajoute un *reset minimal*.

Tapez ceci au début de `style.css` :

```
/* ===== Reset minimal ===== */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: sans-serif;
  line-height: 1.6;
  color: #1A1A1A;
  background-color: #F8F8F8;
}
```

Enregistrez. La page change d'apparence : tous les espacements sont remis à zéro, les marges des paragraphes disparaissent. C'est moche provisoirement, mais on repart d'une base contrôlée. ✓

Pourquoi ce reset ?

- `box-sizing: border-box` : les calculs de largeur incluent padding et border (vu au CM2)
- `margin: 0; padding: 0` : on annule les marges par défaut du navigateur
- `line-height: 1.6` : un peu d'air entre les lignes pour la lisibilité

4 Étape 3 : La palette panafricaine

4.1 Le choix des couleurs

On utilise les couleurs panafricaines (présentes sur de nombreux drapeaux du continent) :

Couleur	Code hex	Usage
 Vert	#0E8A4F	Header, accents, bouton principal
 Jaune	#F5C518	Nav, mise en valeur
 Rouge	#C8332E	Alertes, mode « Drapeaux »
 Noir	#1A1A1A	Texte principal, footer
 Blanc cassé	#F8F8F8	Fond de page

4.2 Variables CSS : un seul endroit pour les couleurs

Plutôt que de répéter #0E8A4F 20 fois dans le fichier, on définit des **variables CSS** (officiellement appelées *custom properties*).

Ajoutez en haut de `style.css`, juste après le reset :

```
/* ===== Palette panafricaine ===== */
:root {
  --vert: #0E8A4F;
  --jaune: #F5C518;
  --rouge: #C8332E;
  --noir: #1A1A1A;
  --blanc-casse: #F8F8F8;
}
```

Comment les utiliser dans n'importe quelle règle :

```
header {
  background-color: var(--vert);
}
```

L'avantage des variables

Si demain vous décidez de changer le vert pour un #1A9A5F (un peu plus clair), vous changez **une seule ligne** dans `:root` et toute l'application est mise à jour. Sans variables, il faudrait chercher et remplacer 20 occurrences à la main.

:root

`:root` est un sélecteur spécial qui cible la racine du document (c'est-à-dire `<html>`). On l'utilise pour déclarer des variables globales accessibles partout dans le CSS.

5 Étape 4 : Importer les polices Google Fonts

Les polices par défaut du navigateur sont moches. **Google Fonts** propose plus de 1000 polices gratuites, libres d'utilisation.

5.1 Choisir 2 polices

On va utiliser :

- **Poppins** pour le corps du texte (claire, ronde, moderne)
- **Montserrat** pour les titres (forte, bien lisible en gros)

5.2 Méthode 1 : via le <link> HTML (recommandé)

Dans le <head> de **chaque** page HTML, **avant** la ligne <link rel="stylesheet" href="style.css">, ajoutez :

```
<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=Poppins:wght@400;600&family=Montserrat"
  rel="stylesheet">
```

5.3 Utiliser les polices dans style.css

Mettez à jour le body dans votre CSS :

```
body {
  font-family: 'Poppins', sans-serif;
  line-height: 1.6;
  color: var(--noir);
  background-color: var(--blanc-casse);
}

h1, h2, h3 {
  font-family: 'Montserrat', sans-serif;
  font-weight: 700;
}
```

Enregistrez. La page utilise maintenant Poppins pour le texte courant et Montserrat pour les titres. ✓

Connexion internet obligatoire

Google Fonts est servi en ligne. Sans Internet, les polices ne se chargent pas et le navigateur retombe sur **sans-serif** par défaut. C'est **normal** et ce n'est pas un bug. En production sérieuse, on téléchargerait les fichiers de police pour les servir localement, mais pour ce cours on garde la simplicité.

6 Étape 5 : Styliser le header

Ajoutez à `style.css` :

```
/* ===== Header ===== */
header {
  background-color: var(--vert);
  color: white;
  padding: 30px 20px;
  text-align: center;
}

header h1 {
  font-size: 2.5rem;
  margin-bottom: 8px;
  letter-spacing: 1px;
}

header p {
  font-size: 1.1rem;
  opacity: 0.9;
  font-style: italic;
}
```

Enregistrez. Le bandeau du haut est maintenant vert vif avec un gros titre blanc « Jàng Afrig » et un sous-titre italique en dessous. ✓

Quelques propriétés nouvelles

- `padding: 30px 20px` raccourci pour 30px haut/bas et 20px gauche/droite
- `font-size: 2.5rem` 2.5 fois la taille de base ($\approx 40\text{px}$)
- `letter-spacing: 1px` espacement entre les lettres
- `opacity: 0.9` transparence à 90% (1 = opaque, 0 = invisible)

7 Étape 6 : Styliser la nav avec Flexbox

Ajoutez à `style.css` :

```
/* ===== Navigation ===== */
nav {
  background-color: var(--jaune);
  padding: 0;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}

nav a {
  color: var(--noir);
  text-decoration: none;
  font-weight: 600;
  padding: 15px 25px;
  transition: background-color 0.2s;
}

nav a:hover {
```

```
background-color: var(--vert);  
color: white;  
}
```

Enregistrez. La nav est maintenant un bandeau jaune avec les liens centrés en gras noir. **Survolez un lien** : il devient vert avec texte blanc, en douceur. ✓

L'effet :hover avec transition

La propriété `transition: background-color 0.2s` fait que le changement de couleur au survol se fait **progressivement** sur 0.2 seconde au lieu d'être brutal. Petit détail, gros effet de qualité perçue.

7.1 Pourquoi le « | » a disparu ?

Au Lab 2, la nav contenait des séparateurs | entre les liens. Maintenant qu'on utilise Flexbox + padding, on n'en a plus besoin : chaque lien a son propre espace. **Vous pouvez retirer les |** dans le HTML des 5 pages. La nav devient :

```
<nav>  
  <a href="index.html">Accueil</a>  
  <a href="quiz.html">Jouer</a>  
  <a href="scores.html">Mes Scores</a>  
  <a href="apropos.html">À propos</a>  
</nav>
```

8 Étape 7 : Cartes pour les modes de jeu

Les modes de jeu sont actuellement une simple liste . On va les transformer en **cartes** colorées disposées en grille avec Flexbox.

8.1 Modifier le HTML de index.html

Remplacez la <section> des modes de jeu dans index.html par :

```
<section>
  <h2>Les modes de jeu</h2>
  <div class="modes-grille">
    <div class="carte-mode carte-jaune">
      <h3>Capitales</h3>
      <p>« Quelle est la capitale du Mali ? »</p>
    </div>
    <div class="carte-mode carte-rouge">
      <h3>Drapeaux</h3>
      <p>« À quel pays appartient ce drapeau ? »</p>
    </div>
    <div class="carte-mode carte-vert">
      <h3>Langues</h3>
      <p>« Dans quel pays parle-t-on le swahili ? »</p>
    </div>
    <div class="carte-mode carte-jaune">
      <h3>Voisins</h3>
      <p>« Combien de voisins a l'Algérie ? »</p>
    </div>
    <div class="carte-mode carte-rouge">
      <h3>Chrono</h3>
      <p>Maximum de bonnes réponses en 60 secondes</p>
    </div>
  </div>
</section>
```

8.2 Ajouter le CSS

Ajoutez à style.css :

```
/* ===== Section principale ===== */
main {
  max-width: 1100px;
  margin: 30px auto;
  padding: 0 20px;
}

main section {
  margin-bottom: 40px;
}

main h2 {
  color: var(--vert);
  border-bottom: 3px solid var(--jaune);
  padding-bottom: 8px;
  margin-bottom: 20px;
}
```

```
/* ===== Grille de cartes (Flexbox) ===== */
.modes-grille {
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
}

.carte-mode {
  flex: 1 1 200px;
  padding: 20px;
  border-radius: 8px;
  color: var(--noir);
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
}

.carte-mode h3 {
  margin-bottom: 8px;
}

.carte-jaune { background-color: var(--jaune); }
.carte-rouge { background-color: var(--rouge); color: white; }
.carte-vert { background-color: var(--vert); color: white; }
```

Enregistrez. Les modes de jeu apparaissent maintenant en **cartes colorées en grille**, qui se réorganisent automatiquement si la fenêtre rétrécit. ✓

flex: 1 1 200px

Cette syntaxe condensée signifie : chaque carte peut grandir et rétrécir, mais sa largeur *idéale* est 200px. Combiné avec **flex-wrap: wrap** sur le parent, les cartes se mettent automatiquement sur plusieurs lignes si l'écran est trop étroit. C'est le début du *responsive*.

9 Étape 8 : Boutons stylés

Le bouton « Commencer » de la page `quiz.html` est moche. On le transforme en bouton vert avec effet de survol.

9.1 Ajouter une classe au bouton

Dans `quiz.html`, modifiez le bouton :

```
<button type="submit" class="btn btn-primary">Commencer le quiz</button>
```

9.2 Styler les boutons en CSS

Ajoutez à `style.css` :

```
/* ===== Boutons ===== */
.btn {
  display: inline-block;
  padding: 12px 28px;
  font-size: 1rem;
  font-weight: 600;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  text-decoration: none;
  transition: background-color 0.2s, transform 0.1s;
}

.btn-primary {
  background-color: var(--vert);
  color: white;
}

.btn-primary:hover {
  background-color: #0a6e3f; /* vert plus foncé */
}

.btn-primary:active {
  transform: scale(0.97);
}

.btn-secondary {
  background-color: var(--blanc-casse);
  color: var(--noir);
  border: 2px solid var(--noir);
}

.btn-secondary:hover {
  background-color: var(--noir);
  color: white;
}
```

Enregistrez. Le bouton est maintenant vert avec coins arrondis. **Survolez-le** : il devient vert foncé. **Cliquez-le** : il rétrécit légèrement (effet enfoncé). ✓

Pseudo-classes pour les boutons

- **:hover** au survol de la souris
- **:active** pendant le clic (souris enfoncée)
- **:focus** quand sélectionné au clavier (Tab)

Pour l'accessibilité, traitez **:hover** et **:focus** de la même façon (pour les utilisateurs au clavier).

10 Étape 9 : Tableau du palmarès

Le tableau de `scores.html` est sans aucune mise en forme. On lui donne un look propre.

Ajoutez à `style.css` :

```
/* ===== Tableaux ===== */
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 15px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.05);
}

th {
    background-color: var(--vert);
    color: white;
    padding: 12px;
    text-align: left;
}

td {
    padding: 10px 12px;
    border-bottom: 1px solid #e0e0e0;
}

/* Lignes alternées (zébrures) */
tr:nth-child(even) td {
    background-color: rgba(245, 197, 24, 0.08);
}

tr:hover td {
    background-color: rgba(14, 138, 79, 0.1);
}
```

Enregistrez et allez sur `scores.html`. Le tableau est maintenant : en-tête vert, lignes claires alternées avec un soupçon de jaune, surlignage vert au survol. ✓

`tr:nth-child(even)`

`:nth-child(even)` sélectionne les éléments *pairs* (la 2^e, la 4^e, la 6^e...). Combiné avec une couleur très claire, ça crée l'effet zébrures qui aide à suivre les lignes du tableau.

11 Étape 10 : Footer

Ajoutez à `style.css` :

```
/* ===== Footer ===== */
footer {
    background-color: var(--noir);
    color: var(--blanc-casse);
    padding: 25px 20px;
    text-align: center;
    margin-top: 50px;
    font-size: 0.9rem;
}

footer a {
```

```
    color: var(--jaune);  
    text-decoration: underline;  
}  
  
footer a:hover {  
    color: white;  
}
```

Le pied de page devient un bandeau noir avec texte clair et liens jaunes. ✓

12 Vérification visuelle

12.1 Naviguez entre les 5 pages

Avec Live Server, parcourez :

1. `index.html` : header vert, nav jaune, cartes colorées, footer noir
2. `quiz.html` : formulaire avec bouton vert « Commencer »
3. `resultat.html` : page de résultat
4. `scores.html` : tableau du palmarès stylé
5. `apropos.html` : sections informatives

Toutes les pages partagent le même look. C'est la magie d'un fichier CSS externe lié à toutes les pages. ✓

12.2 DevTools : modifier le CSS en direct

Inspecter et modifier

Dans le navigateur :

1. Clic droit sur un élément → **Inspecter**
2. L'onglet **Styles** (panneau de droite) montre toutes les règles CSS qui s'appliquent
3. **Modifiez une valeur** (par exemple changez `background-color: var(--vert)` en `red`) : la page change instantanément
4. Vos changements sont **temporaires** (ils disparaissent au rafraîchissement). C'est parfait pour expérimenter sans toucher au code.

C'est l'outil que tous les développeurs utilisent pour bricoler.

12.3 Ce qui ne marche pas encore (et c'est normal)

- Le formulaire ne fait toujours rien quand on clique Commencer → c'est pour le **Lab 7** avec JS
- L'app n'est pas encore optimisée pour téléphone → c'est pour le **Lab 4** (responsive)
- Les images ne sont pas dans une grille → c'est pour le **Lab 4** aussi

13 Exercices d'application

Trois mini-exercices à enchaîner

Exercice 1 — Bouton secondaire

Sur `resultat.html`, transformez les liens Rejouer, Voir mes scores, Retour à l'accueil en **boutons** en utilisant les classes `.btn` et `.btn-secondary` (déjà dans votre CSS) :

```
<a href="quiz.html" class="btn btn-secondary">Rejouer</a>
<a href="scores.html" class="btn btn-secondary">Voir mes scores</a>
<a href="index.html" class="btn btn-secondary">Retour à l'accueil</a>
```

Bonus : disposez-les en ligne avec `display: flex; gap: 10px;` sur leur conteneur.

Exercice 2 — Personnaliser une carte de mode

Modifiez la carte Chrono sur `index.html` pour :

- Lui donner une nouvelle classe `.carte-noire`
- Définir cette classe en CSS : fond noir, texte blanc, bordure jaune épaisse

Exercice 3 — Inverser les couleurs au survol des cartes

Faites en sorte qu'au survol d'une carte de mode, ses couleurs s'inversent (texte sur fond clair devient fond foncé sur texte clair). Indice : utilisez `.carte-mode:hover` et `transition`.

Bonus : changez la palette! Remplacez le vert par une autre couleur dominante (par exemple un rouge brique `#a04436`) et observez que toute l'app se met à jour grâce aux variables CSS.

14 Aide-mémoire

14.1 Propriétés CSS vues dans ce Lab

Propriété	Rôle
background-color	Couleur de fond
color	Couleur du texte
font-family	Police de caractères
font-size	Taille de police (px, rem, em)
font-weight	Graisse (400 normal, 600 demi-gras, 700 gras)
text-align	Alignement texte (left , center , right)
text-decoration	Soulignement (underline , none)
padding / margin	Espacement intérieur / extérieur
border / border-radius	Bordure / arrondi des coins
box-shadow	Ombre portée
display: flex	Active Flexbox sur le conteneur
flex (raccourci)	Comportement d'un flex item
gap	Espace entre les flex items
transition	Animation douce sur changement de propriété
cursor: pointer	Curseur en main au survol
opacity	Transparence (0 à 1)

14.2 Variables et pseudo-classes

- Définir une variable : `:root { --maCouleur: #0E8A4F; }`
- Utiliser une variable : `color: var(--maCouleur);`
- `:hover` au survol
- `:active` pendant le clic
- `:focus` quand sélectionné
- `:nth-child(even)` lignes paires

14.3 Structure type d'un fichier style.css

```

/* 1. Reset minimal */
* { box-sizing: border-box; margin: 0; padding: 0; }

/* 2. Variables */
:root { --vert: #0E8A4F; }

/* 3. Body de base */
body { font-family: 'Poppins', sans-serif; }

/* 4. Sélecteurs d'éléments (header, nav, main, footer) */
header { ... }
nav { ... }

/* 5. Sélecteurs de classes (composants réutilisables) */
.btn { ... }
.carte-mode { ... }

/* 6. États (hover, focus, active) */
.btn:hover { ... }

```

15 Routine Git de fin de Lab

Dans le terminal intégré de VS Code :

```
git status
git add .
git commit -m "v0.3 : CSS panafricain + Google Fonts"
git push
git tag v0.3
git push --tags
```

Vérifiez sur <https://github.com/votre-username/jang-afrig> :

- Le fichier `style.css` est bien là
- Les 5 pages HTML contiennent le `<link>`
- Le tag `v0.3` apparaît dans l'onglet **Tags**

16 Rapport à produire

Rapport du Lab 3 — obligatoire

Format : 1-2 pages, déposé dans `rapports/rapport-lab-3-NOM-Prenom.pdf`.

Contenu (4 sections) :

1. **Ce que j'ai fait** : `style.css` créé et lié, palette définie, polices importées, header/nav/cartes/boutons/tableau/footer stylés, exercices d'application complétés.
2. **Difficultés rencontrées** : variables qui ne fonctionnent pas, polices qui ne se chargent pas, problèmes de spécificité, conflits de styles...
3. **Comment j'ai résolu** : DevTools, MDN, aide d'un camarade.
4. **Lien vers mon repo avec le tag v0.3** :
ex. <https://github.com/fatou-diop/jang-afrig/releases/tag/v0.3>

Bonus dans le rapport (facultatif) : une capture d'écran de votre page d'accueil stylée. C'est le premier rendu visuel de votre projet, ça mérite d'être archivé.

17 Résumé et transition

17.1 Ce que vous avez accompli

- Créé un fichier `style.css` unique pour les 5 pages ✓
- Mis en place reset CSS et `box-sizing: border-box` ✓
- Importé 2 polices Google Fonts ✓
- Défini une palette panafricaine en variables CSS ✓
- Stylisé header, nav, cartes, boutons, tableau, footer ✓
- Ajouté des effets `:hover` et `transition` ✓
- Posé le tag `v0.3` ✓

17.2 Ce qu'on fera au Lab 4

- Faire en sorte que l'app soit **lisible sur téléphone**
- Découvrir les **media queries**
- Construire un **layout responsive** (1 colonne mobile, 2-3 colonnes desktop)
- Gérer une grille d'images (drapeaux côte à côte au lieu de empilés)
- Tag `v0.4`

À ce stade du semestre

Votre application a maintenant un **vrai look professionnel sur ordinateur**. La structure HTML est solide (Lab 2), le style est appliqué (Lab 3). Manque le **responsive** (Lab 4) avant qu'on attaque la partie interactivité (JavaScript).

Lab 3 — Terminé ✓

Vous avez transformé un document brut en application stylée.
Place au responsive : que ça marche aussi sur téléphone.

Rendez-vous au Lab 4 !