Analyseur d'Image depuis Disque

Objectif du projet

Ce projet consiste en une interface logicielle simple permettant de :

- Charger une image depuis un disque local
- Afficher l'image dans la page web
- Extraire et présenter un tableau de paramètres détaillés de l'image, tels que :
- La taille du fichier
- La résolution (largeur x hauteur en pixels)
- Le type/format d'image (JPEG, PNG, ...)
- Le taux de compression (approximation)
- La dynamique lumineuse (plage des intensités mesurée)

Structure du projet

Le projet est constitué de trois fichiers principaux dans le même répertoire :

- `index.html` : La page web principale, structure de l'interface
- `styles.css` : La feuille de style CSS moderne et responsive, adaptée mobile
- `script.js` : Le script JavaScript qui gère le chargement, l'affichage, et l'analyse de l'image

Fonctionnalités

- Choix d'image par un champ input type `file` limité aux fichiers image
- Affichage direct de l'image chargée
- Analyse des pixels de l'image dans un Canvas HTML5 pour calculer la dynamique lumineuse
- Calcul du taux de compression approximatif basé sur la taille brute des pixels vs la taille fichier
- Affichage clair et visuel des paramètres dans un tableau
- Interface responsive, optimisée pour écran petit (typiquement mobiles)

Description du Code

HTML (`index.html`)

- Structure simple contenant un champ de fichier, une zone d'affichage d'image, et un tableau pour paramètres
- Inclusion du CSS (`styles.css`) et du JavaScript (`script.js`)
- Balises sémantiques et attributs ARIA pour accessibilité

CSS (`styles.css`)

- Styles modernes (font Segoe UI)
- Responsive limité à 350px largeur et 600px hauteur
- Boutons et champs esthétique bleue moderne (#0078d7)

- Zone image encadrée, tableau avec en-tête fixe
- Couleurs neutres et agréables à l'œil, transitions sur boutons

JavaScript (`script.js`)

- Ecoute l'évènement `change` sur l'input fichier
- Lit le fichier image via FileReader en DataURL
- Affiche l'image dans un élément `<img>`
- Copie l'image dans un canvas pour extraire pixels RGBA
- Calcule min/max luminosité pour dynamique lumineuse (pondération 0.299R+0.587G+0.114B)
- Calcule taux compression approximation avec taille brute pixels
- Met à jour le tableau des paramètres avec ces valeurs

Déploiement et Test

Prérequis

- Aucune installation serveur requise
- Un navigateur web moderne (Chrome, Firefox, Edge, Safari)
- Tous les fichiers dans un même dossier

Étapes

1. Placer les fichiers `index.html`, `styles.css` et `script.js` dans un même dossier sur votre PC
2. Ouvrir le fichier `index.html` par un double-clic ou via le menu « Ouvrir avec… » dans votre navigateur
3. Utiliser le bouton « Choisir un fichier » pour sélectionner une image depuis votre disque
4. L'image s'affiche et les paramètres se mettent à jour automatiquement
5. Tester avec différents formats (JPEG, PNG, etc.) et tailles d'image pour voir les variations des données

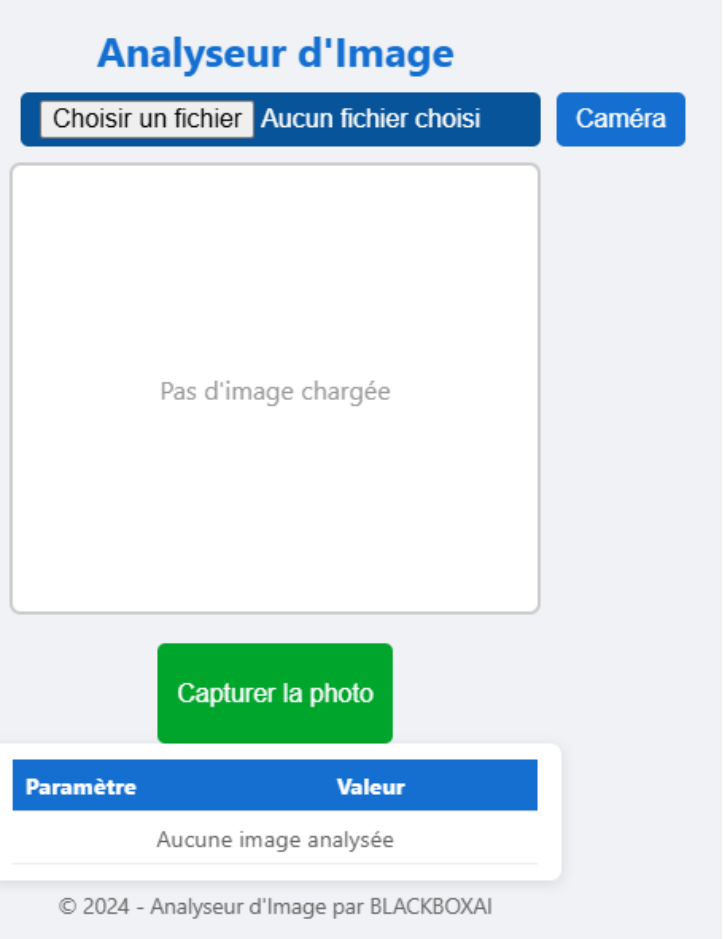
Points importants

- La dynamique lumineuse estimée est une approximation basée sur la gamme de luminosité des pixels visibles
- Le taux de compression est calculé approximativement en comparant la taille compressée du fichier vs la taille brute pixel (RGBA)
- L'interface est entièrement responsive et fonctionnelle sur mobiles
- Pas besoin de serveur web ni d'installation complexe
- Démonstration instantanée, idéale pour présentations pédagogiques ou techniques

Perspectives d'améliorations possibles

- Support de la capture vidéo/caméra
- Sauvegarde des données d'analyse dans un fichier CSV
- Analyse avancée : histogramme, entropie, profil de couleurs
- Support de gros fichiers avec affichage progressif

1. `index.html` : La page web principale, structure de l'interface




```
<!DOCTYPE html>

<html lang="fr">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1, user-scalable=no" />
 <title>Analyseur d'Image depuis Disque</title>
 <link rel="stylesheet" href="styles.css" />
</head>
<body>
 <h1>Analyseur d'Image (Depuis Disque)</h1>
 <div class="controls">
  <input type="file" id="file-input" accept="image/*"
title="Charger une image depuis le disque" aria-
label="Choisir image fichier"/>
 </div>
 <div id="image-container" aria-live="polite" aria-
label="Aire d'affichage de l'image">
  <span>Pas d'image chargée</span>
 </div>
 <div id="parameters" aria-label="Paramètres de
l'image">
  <table>
   <thead>
    <tr><th>Paramètre</th><th>Valeur</th></tr>
   </thead>
   <tbody id="params-tbody">
    <tr><td colspan="2" style="text-align:center; color:#666;">Aucune image analysée</td></tr>
   </tbody>
  </table>
 </div>
 <div class="info-text">
  &copy; 2024 - Analyseur d'Image par Wahid Dridi SRT en ligne 2024/2025
 </div>

 <script src="script.js"></script>
</body>
</html>
```

2- `styles.css` — Le CSS moderne et responsive pour l'interface.

```
/* Styles modernes et responsives pour l'interface Analyseur d'Image */

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  margin: 0;
  padding: 10px;
  background: #f0f2f5;
  color: #333;
  display: flex;
  flex-direction: column;
  align-items: center;
  max-width: 350px;
  height: 600px;
  overflow: hidden;
  box-sizing: border-box;
}

h1 {
  font-size: 1.6rem;
  margin-bottom: 10px;
  text-align: center;
  color: #0078d7;
}

.controls {
  width: 100%;
  display: flex;
  justify-content: center;
  margin-bottom: 10px;
}

input[type="file"] {
  padding: 8px 12px;
  font-size: 1rem;
  cursor: pointer;
  background-color: #0078d7;
  color: white;
  border-radius: 5px;
  border: none;
  transition: background-color 0.25s ease;
}

input[type="file"]:hover {
  background-color: #005a9e;
}

#image-container {
  width: 100%;
  height: 280px;
```

```
  background: #fff;
  border: 2px solid #ccc;
  border-radius: 7px;
  display: flex;
  justify-content: center;
  align-items: center;
  overflow: hidden;
  margin-bottom: 12px;
  position: relative;
}

#image-container img {
  max-width: 100%;
  max-height: 100%;
  display: block;
  border-radius: 5px;
}

#parameters {
  width: 100%;
  background: white;
  border-radius: 7px;
  box-shadow: 0 2px 8px rgb(0 0 0 / 0.1);
  padding: 10px 15px;
  font-size: 0.9rem;
  max-height: 210px;
  overflow-y: auto;
}

#parameters table {
  width: 100%;
  border-collapse: collapse;
}

#parameters th, #parameters td {
  text-align: left;
  padding: 6px 8px;
  border-bottom: 1px solid #eee;
}

#parameters th {
  background-color: #0078d7;
  color: white;
  position: sticky;
  top: 0;
  z-index: 1;
}

.info-text {
  font-size: 0.85rem;
  text-align: center;
  color: #666;
  margin-top: 6px;
```

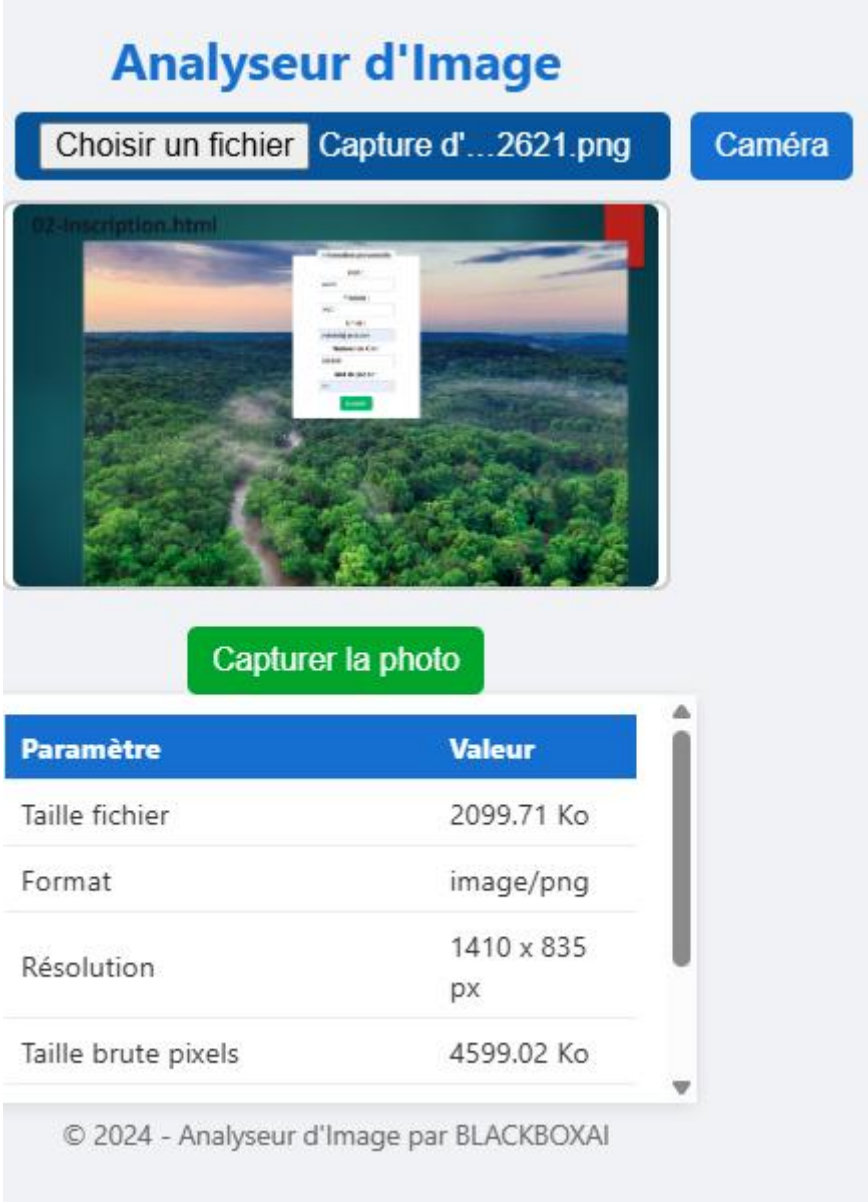
```
}
```

3- `script.js` — Le JavaScript qui charge l'image depuis disque, l'affiche, extrait et affiche ses paramètres.

```
// Références DOM
const fileInput = document.getElementById('file-input');
const imageContainer = document.getElementById('image-container');
const paramsTbody = document.getElementById('params-tbody');

// Fonction pour afficher un message dans la zone image
function showImageMessage(msg) {
  imageContainer.innerHTML = `<span style="color:#999;">${msg}</span>`;
}

// Fonction pour afficher une image HTML dans la zone imageContainer
function displayImageElement(imgElement) {
  imageContainer.innerHTML = "";
  imageContainer.appendChild(imgElement);
}

// Fonction pour nettoyer la zone paramètres
function clearParameters() {
  paramsTbody.innerHTML = '<tr><td colspan="2" style="text-align:center; color:#666;">Aucune image
analysée</td></tr>';
}

// Fonction pour mettre à jour le tableau des paramètres
function updateParameters(imgFile, imageElement, imagePixelsData) {
  const fileSizeKo = (imgFile.size / 1024).toFixed(2) + ' Ko';
  const width = imageElement.naturalWidth || imageElement.width;
  const height = imageElement.naturalHeight || imageElement.height;
  const type = imgFile.type || 'Inconnu';
  const rawSizeBytes = width * height * 4; // RGBA 8 bits * 4 composants
  let compressionPercent = 0;
  let dynamicRange = 'N/A';

  if (imagePixelsData) {
    let minLum = 255;
    let maxLum = 0;
    for (let i = 0; i < imagePixelsData.length; i += 4) {
      const r = imagePixelsData[i];
      const g = imagePixelsData[i + 1];
      const b = imagePixelsData[i + 2];
      const lum = 0.299 * r + 0.587 * g + 0.114 * b;
      if (lum < minLum) minLum = lum;
      if (lum > maxLum) maxLum = lum;
    }
    dynamicRange = (maxLum - minLum).toFixed(1);
    compressionPercent = (100 * (1 - (imgFile.size / rawSizeBytes))).toFixed(1);
    if (compressionPercent < 0) compressionPercent = 0;
  }
```

```
  paramsTbody.innerHTML = `
    <tr><td>Taille fichier</td><td>${fileSizeKo}</td></tr>
    <tr><td>Format</td><td>${type}</td></tr>
    <tr><td>Résolution</td><td>${width} x ${height} px</td></tr>
    <tr><td>Taille brute pixels</td><td>${(rawSizeBytes / 1024).toFixed(2)} Ko</td></tr>
    <tr><td>Taux de compression approximatif</td><td>${compressionPercent} %</td></tr>
    <tr><td>Plage dynamique (luminosité)</td><td>${dynamicRange} (0-255)</td></tr>
  `;
}

// Fonction pour analyser l'image en extrayant pixels via canvas
function analyzeImage(imageElement, file) {
  const canvas = document.createElement('canvas');
  canvas.width = imageElement.naturalWidth || imageElement.width;
  canvas.height = imageElement.naturalHeight || imageElement.height;
  const ctx = canvas.getContext('2d');
  ctx.drawImage(imageElement, 0, 0);

  try {
    const imgData = ctx.getImageData(0, 0, canvas.width, canvas.height);
    updateParameters(file, imageElement, imgData.data);
  } catch (e) {
    updateParameters(file, imageElement, null);
  }
}

// Fonction de chargement et affichage d'image depuis un fichier local
function loadImageFromFile(file) {
  if (!file || !file.type.startsWith('image/')) {
    alert('Veuillez sélectionner un fichier image valide.');
    return;
  }
  const reader = new FileReader();
  reader.onload = function (e) {
    const img = new Image();
    img.onload = function () {
      displayImageElement(img);
      analyzeImage(img, file);
    };
    img.src = e.target.result;
  };
  reader.readAsDataURL(file);
}

// Gestion événement sur sélection fichier
fileInput.addEventListener('change', (e) => {
  if (e.target.files.length > 0) {
    loadImageFromFile(e.target.files[0]);
  } else {
    showImageMessage("Pas d'image chargée");
    clearParameters();
  }
});
```

```
// Initialisation
showImageMessage("Pas d'image chargée");
clearParameters();
```



Pour faire un test localement en suivant ces étapes simples :

1. On crée un dossier sur l'ordinateur, par exemple `analyse-image`.
2. Dans ce dossier, crée ces trois fichiers avec le contenu que je t'ai donné :
   - `index.html`
   - `styles.css`
   - `script.js`
3. Ouvre le fichier `index.html` avec un navigateur web moderne (Chrome, Firefox, Edge, Safari).
4. Utilise le bouton "Choisir un fichier" pour sélectionner une image depuis ton disque.
5. L'image s'affichera et ses paramètres seront calculés et affichés automatiquement.

or create a new repository on the command line

```
echo "# Analyseur_d-Image_html_css_js" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/wahidridi/Analyseur_d-Image_html_css_js.git
git push -u origin main
```