TP : Accessibilité Web avec HTML5 et ARIA

Introduction

> L'accessibilité numérique vise à garantir que tous les utilisateurs, y compris ceux en situation de handicap, puissent naviguer, comprendre et interagir avec le contenu d'un site web.
>
> Dans ce TP, vous allez enrichir la structure du code HTML grâce aux attributs **ARIA** (`role`, `aria-label`, etc.) afin de rendre votre site plus compréhensible pour les **technologies d'assistance** (lecteurs d'écran, navigation clavier).
>
> Ces bonnes pratiques sont aujourd'hui essentielles pour respecter les normes d'accessibilité (WCAG) et proposer des sites **inclusifs**, utilisables par tous.
>
> Ce travail s'inscrit dans une démarche de **développement responsable et citoyen**, encouragée dans les milieux professionnels, éducatifs et institutionnels.

Objectifs :

- Identifier les balises sémantiques HTML5 utiles à l'accessibilité
- Utiliser les rôles ARIA (`banner`, `navigation`, `main`, etc.)
- Ajouter des **descriptions alternatives** (`aria-label`) pour clarifier les zones
- Rendre les **éléments interactifs accessibles au clavier**
- Comprendre les **bonnes pratiques** de base pour concevoir des interfaces inclusives

---

Explication du CSS fourni

Élément	Explication
`:root`	Définit les variables CSS (couleurs) pour une meilleure maintenance.
`.dark`	Classe activée sur `<body>` pour changer de thème via JS.
`nav`, `main`, `header`, `footer`	Structure claire et moderne (HTML5).
`flex`	Disposition horizontale (ou verticale sur mobile) entre la colonne gauche et droite.
`@media`	Gère le responsive sur petits écrans.
JS simple	`toggleTheme()` ajoute ou enlève `.dark` à `<body>`.

Exercices HTML/CSS/JS (Thème, Mise en page, Accessibilité)

Exercice 1 : Comprendre la structure de la page

Objectif : Identifier les grandes parties d'une page web HTML5

> **Instructions :**

1. Ouvrir le fichier HTML dans un éditeur (VS Code ou autre).
2. Souligner ou commenter les balises suivantes :
   - `<header>`
   - `<nav>`
   - `<main>`
   - `<section class="left">`
   - `<aside class="right">`
   - `<footer>`
3. Expliquer en 1 phrase le rôle de chacune dans le contexte de ce site.

Exercice 2 : Ajouter un lien dans le menu de navigation

Objectif : Modifier la barre de navigation.

> **Instructions :**

1. Dans la `<nav>`, ajouter un nouveau lien vers une page fictive `contact.html`.
2. Utiliser une icône Font Awesome (par exemple, `fa-envelope` pour une enveloppe).
3. Tester que le lien s'affiche bien.

Exercice 3 : Ajouter une nouvelle ligne dans le tableau

Objectif : Travailler avec les balises `<table>`, `<tr>`, `<td>`, etc.

> **Instructions :**

1. Recopier une ligne existante dans le `<tbody>`.
2. Modifier les valeurs : image, nom, email, etc.
3. Vérifier que la ligne s'affiche bien.

Exercice 4 : Modifier la couleur principale du thème

Objectif : Manipuler les variables CSS

> **Instructions :**

1. Dans le style, modifier la valeur de `--primary` dans `:root`.
2. Observer les changements dans le menu et le bouton actif.

Exercice 5 : Créer un bouton pour activer/désactiver la colonne de droite

Objectif : Manipuler le DOM avec JavaScript

> **Instructions :**

1. Ajouter un bouton sous le tableau (dans `.left`).

```
<button onclick="toggleAside()">Afficher/Masquer la colonne
droite</button>
```

2. En bas du fichier, ajouter le script suivant :

```
function toggleAside() {
  const aside = document.querySelector('.right');
  aside.style.display = (aside.style.display === 'none') ? 'block' :
'none';
}
```

3. Cliquer sur le bouton et observer l'effet.

Exercice 6 : Rendre le site plus accessible

Objectifs :

- Comprendre l'intérêt de l'**accessibilité numérique**
- Ajouter des attributs ARIA pour améliorer l'accessibilité.
- Apprendre à utiliser les **attributs ARIA** et les **rôles sémantiques**
- Appliquer des bonnes pratiques simples sur une page HTML existante

---

Contexte : Améliorer l'accessibilité

L'accessibilité vise à rendre un site utilisable par **tout le monde**, y compris :

- Les personnes utilisant un **lecteur d'écran**
- Les personnes ayant des troubles de la vision, de la motricité ou de la cognition
- Les personnes naviguant uniquement au **clavier**

HTML5 offre déjà une structure **sémantique** avec `<header>`, `<nav>`, `<main>`, etc., mais **ARIA** peut améliorer ou compléter cette structure, notamment pour les **technologies d'assistance**.

Étape 1 : Ajouter les rôles sémantiques

Les rôles indiquent à un lecteur d'écran la **fonction** d'une partie du document.

```
<header role="banner" aria-label="En-tête du site">
  <h1>BTS CIEL</h1>
  <p>Développement Web : HTML – CSS</p>
</header>
```

```
<nav role="navigation" aria-label="Menu principal">
  ...
</nav>

<main role="main">
  <section class="left" aria-label="Contenu principal – Informatique">
    ...
  </section>

  <aside class="right" aria-label="Contenu secondaire – Python">
    ...
  </aside>
</main>

<footer role="contentinfo" aria-label="Pied de page">
  <p>Carnus Enseignement Supérieur, 12000 Rodez</p>
</footer>
```

Étape 2 : Comprendre les rôles ARIA utilisés

Élément	Rôle	Pourquoi ?
`<header>`	`role="banner"`	Partie d'en-tête commune à toutes les pages
`<nav>`	`role="navigation"`	Permet aux lecteurs d'écran de sauter directement à la navigation
`<main>`	`role="main"`	Contenu principal de la page
`<footer>`	`role="contentinfo"`	Infos générales sur la page ou le site
`<section>`	`aria-label="..."`	Décrit le contenu de manière accessible si aucun `<h2>` n'est présent
`<aside>`	`aria-label="..."`	Contenu secondaire ou complémentaire

Note : Si la section contient un titre clair (comme un `<h2>`), l'`aria-label` est parfois redondant, mais reste utile pour les lecteurs d'écran.

Étape 3 : Ajouter des attributs pour les boutons

Exemple sur le **bouton de changement de thème** :

```
<div class="theme-toggle" role="button" tabindex="0" aria-label="Changer
le thème" onclick="toggleTheme()" onkeydown="if(event.key==='Enter')
{toggleTheme()}">
  <i class="fa-solid fa-circle-half-stroke"></i>
</div>
```

Attribut	Rôle	Pourquoi ?
`role="button"`	Rend un élément `<div>` compréhensible comme un bouton	
`tabindex="0"`	Permet de tabuler jusqu'à l'élément au clavier	
`aria-label="Changer le thème"`	Texte lu par un lecteur d'écran	
`onkeydown`	Active le bouton avec la touche Entrée (comme un vrai `<button>`)	

Résumé :

- Utiliser des **balises sémantiques HTML5**
- Compléter avec des **rôles ARIA** si besoin
- Ajouter des **libellés (`aria-label`)** pour les éléments non textuels
- Rendre tous les éléments **accessibles au clavier** (via `tabindex`, `role`, `onkeydown`)

> Explorer la signification de ces rôles avec MDN Web Docs.

Exercice 7 (bonus) : Ajouter un enregistrement dynamique

Objectif : Ajouter une ligne au tableau avec JavaScript.

> **Instructions :**

1. Créer un petit formulaire au-dessus du tableau :

```
<form onsubmit="addRow(event)">
  <input type="text" placeholder="Nom" id="name" required>
  <input type="email" placeholder="Email" id="email" required>
  <button type="submit">Ajouter</button>
</form>
```

2. En JS :

```
function addRow(e) {
  e.preventDefault();
  const name = document.getElementById('name').value;
  const email = document.getElementById('email').value;
  const tbody = document.querySelector('table tbody');
  const row = document.createElement('tr');
  row.innerHTML = `
    <td><img class="circular" src="./img/LOGO.png" alt="Avatar"></td>
    <td>${name}</td>
```

```
      <td>${email}</td>
      <td>--</td>
      <td>--</td>
    `;
    tbody.appendChild(row);
    e.target.reset();
  }
```

Conclusion

Vous avez maintenant :

- Structuré une page HTML5 complète
- Ajouté du style et des effets via CSS
- Rendu la page plus interactive avec JavaScript
- Intégré les bonnes pratiques d'accessibilité web

Passons maintenant à la mise en pratique complète avec un projet synthèse.

Exercice de synthèse – Construire une page web accessible et dynamique

Projet final – Créer une interface web structurée, responsive et accessible

Objectif :

> Concevoir une page web simple, structurée avec HTML5, stylée avec CSS, accessible pour tous, et enrichie d'interactions JavaScript.

Consignes :

> À partir d'une page HTML vierge ou du modèle fourni en début de TP, créer une **interface web complète** comportant les éléments suivants :
>
> 1. Une **structure HTML5 sémantique** (avec `<header>`, `<nav>`, `<main>`, `<section>`, `<aside>`, `<footer>`)
> 2. Une **barre de navigation** avec au moins 3 liens (dont un actif), et un bouton de **changement de thème (light/dark)** en JavaScript
> 3. Un **tableau de données** avec au moins 3 lignes, contenant :
>   - Une image circulaire
>   - Nom, Email, Téléphone, Commentaire
> 4. Un **formulaire simple** permettant d'ajouter une ligne dans le tableau dynamiquement (JS)
> 5. Un bouton permettant d'**afficher/masquer la colonne de droite** (JS)

> 6. Une interface **responsive** (colonnes empilées en mobile)
>
> 7. Une interface **accessible** :
>
>   - utilisation des **rôles ARIA**
>   - `aria-label` sur les zones principales
>   - navigation **clavier** possible sur les boutons
>   - balisage propre et lisible

Livrables attendus :

- Un fichier **HTML** bien structuré
- Un fichier **CSS** dans une balise `<style>` ou externe
- Le script JS dans une balise `<script>` (ou fichier séparé)
- Des **commentaires dans le code** expliquant les parties clés
- Les éléments accessibles au clavier doivent être testables (tabulation, touche Entrée, aria-label...).
- Bonus : icones, police personnalisée, ou design amélioré

Temps estimé : 30mn à 45mn

Critères d'évaluation :

Critère	Points
Structure HTML sémantique correcte	**2**
Navigation avec lien actif + thème JS	**2**
Accessibilité (rôles ARIA, aria-label, navigation clavier)	**4**
Tableau correctement rempli	**2**
Formulaire fonctionnel (JS)	**3**
Affichage/Masquage colonne (JS)	**2**
Responsive (Flexbox + media query)	**2**
Clarté, propreté, commentaires dans le code	**3**
Total	**20 / 20**
Bonus : Design, icônes, animations, etc.	**+1 (facultatif)**

> Bonus : design personnalisé, usage créatif des icônes, transitions CSS, etc.

Note :

Vous pouvez vous appuyer sur le modèle vu ensemble dans ce TP.