

TP2 - Un formulaire accessible (5%)

Mandat

Le formulaire doit être fonctionnel

- sur un téléphone mobile
- sur un poste de table en le naviguant au clavier
- par un lecteur d'écran (!)

Des contraintes de saisie doivent être appliquées.

Ces contraintes reposent sur l'API de formulaires de HTML5.

Ce sont principalement des attributs et des valeurs d'attributs qui précisent les données attendues dans les éléments de formulaire.

Utiliser les jeux d'essais pour tester au fur et à la mesure l'interactivité du formulaire en tentant de soumettre le formulaire avec des données pertinentes ou inadéquates.

Ce TP est l'occasion d'utiliser le système de contrôle des versions GIT afin de garder des traces de chaque volet du travail.

Il y a 2 manières d'initier le versionnage soit qu'on initialise localement (git init) comme nous avons fait pour les exercices d'accessibilité, soit qu'on clone (git clone url dossier).

Commencez par fourcher (fork) le répertoire <https://github.com/integration2/tp2-form-a11y>



puis clonez-le sur votre poste local dans votre dossier Sites, en passant par le terminal (ou Git Bash).

```
cd ~/sites
```

```
git clone URL-de-VOTRE-répertoire tp2-form-a11y
```

Avertissement :

Les exemples dans cet énoncé sont puisés dans le formulaire <https://timunix2.csfoyc.ca/~efevrier/tp2/>

Le contexte de ce TP est plutôt un formulaire de réservation pour un gîte du passant.

Voici un site Web comparable à ce que nous allons faire :

<https://chaletsentreamis.ca/nos-chalets/nos-chalets-cote-fleuve/chanteau-de-mer/#1480128990389-f28c41d6-d1c8>

Volet 1 | Compléter le balisage sémantique du HTML

Examiner le code source de base fourni.

1.1 Baliser le formulaire

- Les regroupements d'éléments de formulaire doivent être fait avec des **fieldset**.
- Étiqueter les **fieldset** en leur ajoutant une balise **legend**.
- Assurez-vous que chaque élément de formulaire est correctement étiqueté (le lien se fait par l'attribut **for** du **label** qui a comme valeur le **id** de l'élément de formulaire).
- Veiller à préciser correctement le type du **input**, consultez au besoin cette liste des types : <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input/>
- Chaque élément de formulaire doit avoir un **name** et un **id** unique sauf les boutons radio qui partagent le même **name**.
- Les éléments de formulaire et leurs étiquettes sont des contenus en ligne, vous devrez donc les placer dans un conteneur bloc comme un **p**aragraphe.

1.2 Tester le formulaire sur un téléphone mobile.

Si le type de input a été bien choisi, le clavier de l'appareil mobile facilitera la saisie.

Par exemple, lorsqu'on saisit une adresse courriel le clavier de l'appareil mobile doit offrir l'arobas.

1.3 Ajouter des contraintes de saisie à l'aide d'attributs HTML.

Champs requis

Tous les champs du formulaire sont obligatoires,

sauf la dernière case à cocher "Oui, j'accepte de recevoir de la documentation...".

Ajouter l'attribut **required** à chacun.

Dans le cas des boutons radios, il suffit de mettre l'attribut **required** sur le premier bouton radio du groupe.

Contraintes de saisie

Ajouter à chaque champ obligatoire, des contraintes de saisie en utilisant les attributs de html 5.

Par exemple, si on utilise 2 champs de saisie pour le numéro de téléphone, un premier pour l'indicatif régional et un second pour le numéro, on pourra mettre une contrainte de 3 caractères et 7 caractères avec les attributs **minlength** et **maxlength**. Certains types de champ imposent déjà des contraintes par exemple, un input de type email auquel on ajoute l'attribut **required** imposera le respect des caractéristiques essentielles d'une adresse courriel comme la présence des caractères « @ » et « . ».

Les motifs

Pour certaines données ayant un motif rigoureux comme le code postal, les attributs de base ne suffiront pas. Pour ces champs, ajouter un attribut **pattern** en vous servant des suggestions du site <http://html5pattern.com/>

L'attribut **pattern** reçoit comme valeur une expression régulière.

Par exemple, pour le champ prénom, on peut utiliser **pattern="[a-zA-ZÀ-ÿ \-]+"**.

Cette expression autorise la présence des minuscules a-z, des majuscules A-Z, des caractères accentués À-ÿ ainsi que les espaces et les traits d'union. Le trait d'union est échappé par la barre oblique qui le précède.

Vérifier au besoin sur le site de référence MDN, quels sont les attributs de contrainte disponible pour le **type** de **input** choisi. Par exemple, sur un type **number** on ne peut pas ajouter l'attribut **pattern**.

1.4 Tester les contraintes de saisie en vous servant des jeux d'essai fournis.

1.5 Valider le code html.

1.6 Versionner.

Volet 2 – Analyser la mise en page, compléter le html et styler

2.1 Analyser, schématiser d'après les esquisses fonctionnelles ou le visuel fourni

- Identifier où il faut ajouter des conteneurs flex (classe `rangee`).
- Identifier où il faut ajouter des conteneurs pour la validation (classe `ctnValidation`).
- Placer des **classes de contexte** sur l'élément qui contient chaque bloc ou regroupement d'éléments.

Les classes de contextes permettent de rédiger des sélecteurs contextuels légers. Exemples :

- `.date` input
// un champ de saisie dans le contexte d'un parent qui a la classe « date »
- `.telephone` p:nth-of-type(2)
// le second paragraphe dans le contexte d'un parent qui a la classe « téléphone »

Ajouter, au besoin seulement, des éléments HTML qui pourront servir de conteneur Flex.

Notez que dans cet exemple, la classe « rangée » définit un conteneur flex (display :flex).

Les « cols_4_de_12 » définissent un flex-basis de 32% pour les items flex.

```
<fieldset class="date ctnValidation">
  <legend class="h4">
    Date à laquelle la question a été posée?
  </legend>
  <div class="rangee">
    <p class="cols_4_de_12"...>
    <p class="cols_4_de_12"...>
    <p class="cols_4_de_12"...>
  </div>
  <p class="erreur"...>
</fieldset>
```

HTML des boutons radio.

Comme il s'agit d'une liste, les 3 boutons radios doivent être dans un `ul`.

Celui-ci pourra servir de conteneur Flex.

Chaque label doit être correctement relié à son input par la valeur de son attribut `for` correspondant à la valeur du `id` de son input. La balise label doit contenir une balise `picture` suivi d'un `span` contenant le texte de description de l'image.

Les paragraphes pour les messages d'erreur JavaScript.

Ajouter un **p** avec la classe « erreur » pour chaque élément ou groupe d'éléments de formulaire à valider. Prévoir le balisage accessible d'une icône d'avertissement.

```
<p class="erreur">
  <span>
    <span class="screen-reader-only">Avertissement</span>
    <span class="icone icone--avertissement" aria-hidden="true"></span>
  </span>
  Le prénom ou le nom comporte un ou plusieurs caractères interdits.
</p>
```

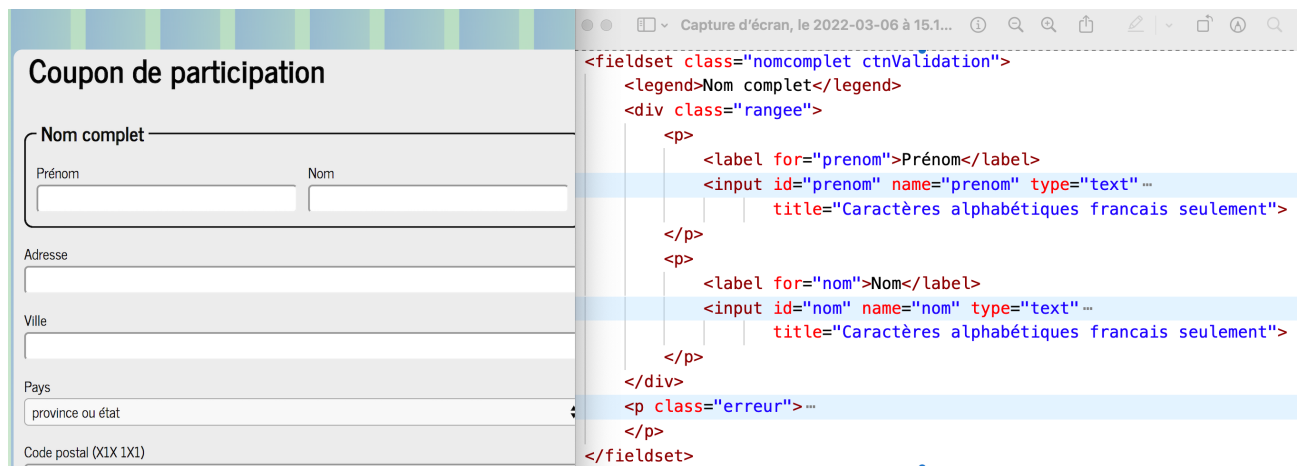
Regrouper le paragraphe d'erreur et l'élément de formulaire

Il s'agit de définir un parent commun pour le paragraphe contenant l'élément de formulaire et le paragraphe d'erreur pour cet élément de formulaire. Vous aurez parfois besoin d'ajouter un div pour créer ce parent commun aux 2 paragraphes. AJOUTER à ce parent commun la classe « **ctnValidation** » en plus d'une classe de contexte.

Exemple pour un élément simple à valider

```
<div class="adresse ctnValidation">
  <p><label for="adresse">Adresse</label>
  <input id="adresse" name="adresse" type="text"...>
</p>
<p class="erreur"...>
</div>
```

Exemple pour un groupe d'éléments à valider



```
<fieldset class="nomcomplet ctnValidation">
  <legend>Nom complet</legend>
  <div class="rangee">
    <p>
      <label for="prenom">Prénom</label>
      <input id="prenom" name="prenom" type="text" ...
        title="Caractères alphabétiques français seulement">
    </p>
    <p>
      <label for="nom">Nom</label>
      <input id="nom" name="nom" type="text" ...
        title="Caractères alphabétiques français seulement">
    </p>
  </div>
  <p class="erreur">...
</p>
</fieldset>
```

2.2 Débuter la mise en page du formulaire

Méthode *Mobile First*

Écrire les styles pour l'écran étroit et ce qui est commun à toutes les variantes, puis ajouter des requêtes media au fur et à la mesure que se présente les variantes.

2.3 Styler l'interactivité.

Styler l'état focus

La feuille de styles de Chrome donne un halo bleu aux éléments de formulaire lorsqu'ils reçoivent le focus. Utiliser la cascade pour redéfinir la propriété `outline`.

`outline : -webkit-focus-ring-color auto 5px;`

// le nom de la couleur par défaut est « `-webkit-focus-ring-color` »

Redéfinissez la couleur du outline pour obtenir un vert lumineux.

Utilisez le filtre `:hov` de *devtools* et cochez `:focus` pour tester le rendu de cette règle.

The image shows a web form on the left and the Chrome DevTools on the right. The form has fields for 'Adresse', 'Ville', 'Pays', 'Code postal (X1X 1X1)', 'Courriel', and 'Téléphone (indicateur - numéro)'. A red arrow points to the 'Code postal' field, which is currently focused. In the DevTools, the 'Styles' pane is open, showing the 'input#adresse' element. The 'Force element state' section has the ':focus' checkbox checked. The 'Filter' section has ':hov' selected. The 'element.style' section shows the 'outline' property being set to '#276062 auto 5px;'. The 'Styles' pane also shows the 'outline' property being set to '-webkit-focus-ring-color auto 5px;'. A red arrow points to the 'Filter' section, which has ':hov' selected.

dans l'exemple ci-dessus la couleur du nouveau outline est `#276062`,
c'est un peu trop foncé, cela n'attire pas suffisamment l'attention !




Cacher visuellement les boutons radio et rendre interactive la boîte du label.

Cacher visuellement les input de type radio tout en conservant leur accessibilité.

Pour cela, appliquer la classe `.screen-reader-only` sur chaque input de type radio.

Rendre interactive la boîte du label

Ajouter les styles pour que le `label` qui contient l'image forme un « bouton » interactif à 3 états.

Initial	Hover et focus	Checked
 Le Taj Mahal	 Le Taj Mahal	 Le Taj Mahal
photo en noir et blanc	coloration de la photo	coloration de la photo + contour noir du label

Notez que ces états sont plutôt les états du input qui précède le label dans le HTML.

Servons-nous de cette structure HTML pour rédiger des sélecteurs.

Si le bouton radio est au focus, on cible le label qui est son frère adjacent

```
.type-de-chambre[type=radio]:focus + label {}
```

ou l'image qui se trouve dans ce label

```
.type-de-chambre[type=radio]:focus + label img {}
```

Tester de nouveau la navigation au clavier du formulaire.

Assurez-vous que les boutons radios sont *focusables* et que les changements de styles se font bien selon les états *normal*, *hover*, *focus* et *checked*.

Styler les erreurs

Les messages d'erreur seront affichés par le JavaScript.

Pour tester, les styles, écrivez au moins un message directement dans le HTML.

Le message d'erreur doit débuter par une icône d'avertissement et afficher son texte d'une couleur contrastante. Les icônes peuvent être intégrés avec des sprites CSS ou une police d'icônes.

Le texte alternatif doit être « erreur ».

Utiliser la nomenclature BEM et le principe de double-classe au besoin.

Exemple :

```
.icone {  
    display: inline-block;  
    width: 24px;  
    height: 24px;  
    background-image: url("../images/good-warning-error.png");  
    background-size: cover;  
    position: relative;  
    top: 3px;  
}  
  
.icone--ok {  
    background-position: 0 0; /* la classe modificateur positionne */  
}
```

Compléter la documentation de la feuille de styles

- Faire des regroupements thématiques & mettre des commentaires d'entêtes
- Compléter la table des matières contenant toutes les entêtes de section
- Vérifier que votre feuille de styles se navigue bien à partir de la table des matières
- Compléter @author avec vos coordonnées (prénom nom, courriel)

Critères d'évaluation

Items réalisés dans le travail	Pts	Habiletés	
Structure, sémantique et validations en html			
Regrouper les éléments de formulaire de même nature. Utiliser des fieldset. Faire des groupes d' options dans une liste déroulante.	1	Intégrer les contenus multimédias en respect des meilleures pratiques d'accessibilité, de performance et de portabilité.	
Étiqueter les regroupements d'éléments de formulaire à l'aide de la balise Legend. Étiqueter un groupe d'<option>s d'une liste déroulante. Étiqueter les champs de formulaire.	1		
Rendre (garder) le formulaire navigable au clavier.	1		
Baliser avec précision les éléments de formulaire. Bien choisir le type du input. Code sémantique et valide pour l'ensemble du document.	1		
Identifier par un attribut approprié les champs obligatoires du formulaire. Ajouter des contraintes de saisie sur les champs de formulaire.			
Styles			
Travailler Mobile First. Contrôler les espacements et les alignements.	2	Utiliser de manière précise et créative les styles CSS pour positionner et mettre en valeur les contenus	
Styler l'interactivité : État focus, état checked des éléments de formulaires États des hyperliens (link, visited, hover, active) Styler les messages d'erreur. Styler les boutons radio en les gardant accessibles au clavier.	2		
Organiser et documenter la feuille de styles			
Navigation et composants accessibles			
Menu réactif javascript (d'après celui du cadriciel)	1		
Utiliser le contrôle des versions GIT Versionner toutes les étapes en s'efforçant de décrire la tâche accomplie dans chaque message de commit.	1		
Total	10		
Bonus de 2 points Intégration et programmation de la page d'accueil incluant son carrousel et des onglets pour les contenus « description, équipements, activités, tarifs et forfaits, emplacement ».	2		

Modalités de remises

Sur Github dans le compte personnel

Groupes 1 et 2 – Vendredi 10 mars minuit