

TP3 - 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 (!)

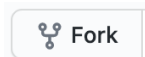
Des contraintes de saisie doivent être appliquées.

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.



Commencez par fourcher (fork) le répertoire <https://github.com/integration2/tp3> 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 tp3
```

Volet 1 | Balisage HTML

1. Glisser dans le répertoire fraîchement cloné le fichier index.html débuté au cours 11.

Ce volet 1 du TP se fait sans ajouter aucun style.

On se concentre sur le HTML et la fonctionnalité du formulaire.

Débuter l'intégration à l'aide des textes fournis et de la capture-écran.

Pour éviter les fautes de frappe, copier-coller les textes.

2. Définir la structure HTML de base

Assurez-vous que le fichier index.html contient:

- Le meta du jeu de caractère utf8 (doit être inséré avant la balise **title**)
- Le meta du viewport (essentiel pour l'affichage correct sur les appareils mobile)
- Un **title** « TP3 – formulaire accessible »
et une balise `<link rel="icon" href="favicon.ico" />`
- Une structure minimale dans le body qui inclus un **header**, un **main** et un **footer** (avec leurs attributs role).
- Le **main** devrait comporter une balise **section** et un **form**.
- Dans le header, il vous faudra ajouter une balise **picture** pour offrir soit le fichier bandeau_600 sur l'écran étroit ou le fichier bandeau_1200 pour l'écran large.

3. Baliser le formulaire

- Déterminer d'abord les regroupements d'éléments de formulaire et ajouter des **fieldset** pour les créer.

- Étiqueter les **fieldset** en leur ajoutant une balise **legend**.

Une balise **legend** peut contenir un élément d'entête **h1-h6**.

NOTE : Il est tout à fait possible de placer un fieldset dans un autre fieldset.

Par exemple : le **fieldset** du *Coupon de participation*

inclus le **fieldset** du *Nom complet*.

- Ajouter les éléments de formulaire (**input** ou **select**) en les accompagnant d'une étiquette (balise **label**).
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**.
- Les éléments de formulaire et leurs étiquettes sont des contenus en ligne, vous devrez donc les placer dans un conteneur bloc (généralement un **p**).

4. Déposer le projet dans votre répertoire sur timunix2 et 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.

5. 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, les deux champs de saisie associés au numéro de téléphone doivent recevoir respectivement 3 caractères et 7 caractères. Les attributs **minlength** et **maxlength** permettront de préciser ces limites.

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 par exemple 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**.

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

7. Valider le code html.

8. Versionner.

Volet 2 – Analyser, schématiser pour vérifier et compléter le balisage HTML

2.1 D'après les guides visuels et votre actuel code HTML

- 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 les éléments parents :
`entete`, `intro`, `nom-complet`, `adresse`, `ville`, `region`, `code-postal`, `courriel`, `telephone`, `date`, `question`,
`reglement`, `documentation`, `soumission`, `credits`.

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.

Ceci sera nécessaire seulement lorsqu'un groupe d'élément de formulaire doit être placé côte à côte comme par exemple pour la date.

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.

Pour pouvoir le repérer facilement en remontant l'arbre HTML à partir de l'élément de formulaire en cours de validation, nous lui ajouterons la classe « **ctnValidation** » en plus d'une classe de contexte.

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>
```

Groupe d'éléments à valider

Coupon de participation

Nom complet

Prénom Nom

Adresse

Ville

Pays

province ou état

Code postal (X1X 1X1)

```
<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.

Base (entête, corps et pied de page)

Les classes de centrage (**.conteneur**), de conteneur Flex (**.rangee**) et de dimensionnement (**colsX_de_12**) permettront d'obtenir la mise en page de base.

Voir le **demo-grille-12-cols**.

Dans le formulaire

Utiliser les flexbox pour disposer côte à côte les éléments de formulaire qui doivent l'être.

Dans certains cas, le dispositif doit être présent sur l'écran étroit aussi hors, dans le fichier grille.css, les classes **rangee** et **colsX_de_12** sont déclarés dans une requête media. Pour les rendre disponibles sur l'écran étroit des règles contextuelles ont été ajoutés pour le nom complet, le téléphone et la date.

Exemple tiré du fichier `_grille.css` :

```
.nomcomplet .rangee,  
.telephone .rangee,  
.date .rangee {  
    display: flex;  
    justify-content: space-between;  
    align-items: stretch; /* valeur par défaut */  
    flex-wrap: wrap;  
}  
  
.nomcomplet .cols_6_de_12 {  
    width: 49%;  
}
```

Charte typographique

Les polices de caractères sont déclarées ici par des instructions `@font-face` et distribuées à partir d'un dossier de polices dans la structure de répertoires du projet.

Contrôler les marges et les espacements intérieurs.

Utiliser des variables CSS pour définir les valeurs qui se répètent souvent.

Un sélecteur **root: {}** est placé au début du fichier utilitaires.css.

Il contient quelques variables pour les couleurs et les espacements récurrents.

Utiliser ces variables. Ajoutez-en si besoin.

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 'Styles' panel on the right. The form contains fields for 'Adresse', 'Ville', 'Pays', 'Code postal (X1X 1X1)', 'Courriel', and 'Téléphone (indicatif - numéro)'. The 'Adresse' field is highlighted with a blue outline. In the DevTools 'Styles' panel, the 'input#adresse' element is selected. The 'Filter' dropdown is set to ':hov .cls'. Under 'Force element state', the ':focus' checkbox is checked. The 'element.style' section shows the following CSS rules:

```
input[type=text], input[type=email], input[type=tel], styles.c
input[type=number], select {
  width: 100%;
  padding: 0.5rem;
  border-radius: 0.5rem;
}

:focus {
  outline: 2px solid #276062 auto 5px;
}
```

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` des boutons radio 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

```
input[type=radio]:focus + label {}
```

ou l'image qui se trouve dans ce label

```
input[type=radio]:focus + label img {}
```

Tester de nouveau la navigation au clavier du formulaire.

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

Styler les erreurs

Le message d'erreur doit débuter par une icône d'avertissement et afficher son texte en rouge.

Utiliser des sprites CSS pour l'icône d'avertissement.

Ils doivent avoir un attribut class avec deux valeurs :

- une classe **icone**
qui définit la boîte qui servira à afficher le sprite et lier la background-image
- une classe **icone--avertissement**
qui ajuste le background-position pour voir seulement l'une des images du fichier

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 | Habilités |
|---|-----|--|
| Structure et sémantique | | |
| Regrouper les éléments de formulaire de même nature. Utiliser des fieldsets. Faire des groupes d'<option>s 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 ; nommer le regroupement. 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 | | |
| Aligner les éléments de formulaire. Contrôler les espacements. Intégrer tous les contenus selon les guides visuels (ou mieux !) | 2 | Utiliser de manière précise et créative les styles CSS pour positionner et mettre en valeur les contenus en respect des maquettes visuelles. |
| Styler l'interactivité : État focus, état checked des éléments de formulaires États des hyperliens (link, visited, hover, active) Styler les messages d'erreur. Utiliser des sprites CSS . | 2 | |
| Styler les boutons radio en les gardant accessibles au clavier. | | |
| Méthodes de travail favorisant la collaboration | | |
| Organiser et documenter la feuille de styles | 1 | |
| Utiliser le contrôle des versions GIT Un minimum de 3 commits est attendu pour les étapes html, css, contrôle qualité finale. | 1 | |
| Total | 10 | |

Modalités de remises

Sur Github dans le compte personnel

Groupes 1 et 2 – Lundi 14 mars minuit