

## TP4 - Un formulaire accessible (5%)

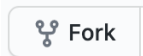
### Mandat

D'après le texte fourni (textes.docx), coder les balises html de manière à

- favoriser l'accessibilité du formulaire
- appliquer des contraintes au niveau de la saisie des données par l'utilisateur

Le formulaire doit être fonctionnel sur un téléphone mobile ou sur un poste de table en le naviguant au clavier. Les contraintes de saisie reposent sur l'API de formulaires de HTML5. Ce sont principalement des attributs et des valeurs d'attributs qui permettent de réaliser des validations html et css. Il faudra tester au fur et à la mesure l'interactivité du formulaire en tentant de soumettre le formulaire avec des données pertinentes ou inadéquates (utilisation de jeux d'essais).

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/evefevrier/concours>, puis clonez-le sur votre poste local en passant par le terminal (ou Git Bash).

```
cd ~/sites  
git clone URL-de-VOTRE-répertoire-concours tp4
```

### Volet 1 | Balisage HTML

1. Ajouter une page index.html et lier la feuille de styles.css.

Importer dans le fichier styles.css, dans l'ordre prescrit par l'ébauche de table des matières :  
normalize, utilitaires, typo, form et grille.

Jeter un coup d'œil aux contenus de ces fichiers. Ce sont ceux du cadriceil proposé au cours 12.

Un fichier form.css a été ajouté pour compartimenter les règles qui seront liées à l'interactivité des formulaires.

**Le volet 1 du TP se fait sans ajouter aucun style.**

Concentrez-vous sur le HTML pour ce premier volet.

Le dossier captures-ecrans > sans-les-styles vous guidera.

Les textes doivent être copié-collé à partir du fichier textes.pdf.

Ne vous occupez pas tout de suite des erreurs, nous le ferons à la fin du volet 1.

## 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** « TP4 » 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 **aside** et un **form** (avec leurs attributs role).
- 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.

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.

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

Par exemples, 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 « . ».

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.

Vous verrez en détail dans le cours de programmation comment écrire et reconnaître une expression régulière. En attendant, vous pourrez utiliser cet attribut de manière exploratoire.

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.

Les tâches 7 et 8 sont effectuées en prévision des validations JavaScript qui seront ajoutées dans le cours de programmation.

## 7. Ajouter les conteneurs pour les messages d'erreur.

Pour chaque élément ou groupe d'éléments de formulaire à valider, ajouter tout de suite après un élément **p** avec la classe « erreur ». 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>
```

## 8. Ajouter un conteneur HTML commun pour

- le conteneur bloc de l'élément de formulaire et son étiquette
- le paragraphe « erreur » consécutif

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

Ce conteneur HTML sera utile lorsque vous ajouterez les validations en JavaScript.

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** ».

## Consulter les guides visuels avec styles pour accomplir la tâche #9

## 9. 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.

```
<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>
```

## 10. Valider le code html.

## 11. Versionner

Git add, git commit, git push avec un message "volet 1 – html"

## Volet 2 | Styles CSS

### 1. Schématisation

Pour ce TP, la schématisation des stratégies d'intégration est fournie en partie (voir le fichier `schematisation.png` dans le dossier `guides-visuels`).

Compléter le HTML en respect de la structure définie dans le schéma.

### 2. Débuter la mise en page du formulaire

#### Méthode

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.

#### 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 vous devrez les redéclarer dans une classe de contexte comme `.nomcomplet`, `.telephone` ou `.date`.

(Voir la capture-écran en page 4 pour comprendre où appliquer ces classes dans la structure html)

Exemple :

```
.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%;  
}
```

### 3. Charte typographique

Comme dans le cadriciel, 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. Changer cela par une instruction `@import` de *Google Fonts* puisque la police *News Cycle* y est disponible.

Supprimer ensuite les déclarations `@font-face` et le dossier polices qui seront maintenant inutiles.

### 4. Contrôler les marges et les espacements intérieurs.

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

Un sélecteur **root** `:{}` est placé au début du fichier utilitaires.css. Compléter ce sélecteur avec des variables pour les couleurs et les espacements récurrents. Utiliser ensuite ces variables.

### Styler l'interactivité.

### 6. Adapter la couleur du style des éléments qui ont le focus.

La feuille de styles de Chrome donne un halo bleu aux éléments de formulaire lorsqu'ils reçoivent le focus.

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

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

Redéfinissez la couleur du outline avec la valeur hexadécimale **#349194**

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 its corresponding developer tools on the right. The form has fields for 'Adresse', 'Ville', 'Pays', 'Code postal (X1X 1X1)', 'Courriel', and 'Téléphone (indicatif - numéro)'. A red arrow points to the 'Code postal' field, which is currently focused. The developer tools on the right show the DOM tree with the selected element being the 'input' for 'adresse'. The 'Styles' pane shows the 'Force element state' section with the ':focus' checkbox checked. The 'Filter' section shows ':hov' selected. The 'element.style' section shows the 'outline' property set to 'auto 5px;'. The 'input[type=number]' section shows the 'outline' property set to '#276062 auto 5px;'. The 'styles.cs' file is also visible in the bottom right corner.

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

Cette étape s'appuie sur le fait que lorsqu'elle est correctement reliée, la boîte d'un **label** permet de changer l'état de son bouton radio ou de sa case à cocher.

### Une base HTML robuste

Il est préférable d'inclure la liste des boutons radio 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.




### Cacher visuellement les boutons radio

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
photo en noir et blanc	coloration de la photo	coloration de la photo + contour noir du label
 Le Taj Mahal	 Le Taj Mahal	 Le Taj Mahal

**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.

## 8. Styler les erreurs

Cette section est à compléter dans le fichier `_form.css`.

### Styler les messages d'erreur (classe `erreur`)

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

## 9. Compléter la documentation de la feuille de styles

Regroupement thématiques & entêtes de section

Table des matières contenant toutes les entêtes de section

@author contenant vos coordonnées



## Critères d'évaluation

Items réalisés dans le travail	Pts	Habiletés
Structure et sémantique		
<b>Regrouper</b> 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.
<b>Étiqueter</b> 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 <b>navigable</b> au clavier.	1	
Baliser avec précision les éléments de formulaire. Bien choisir le type du input. <b>Code sémantique et valide</b> pour l'ensemble du document.	1	
Identifier par un attribut approprié les <b>champs obligatoires</b> du formulaire. Ajouter des <b>contraintes de saisie</b> sur les champs de formulaire.		
Styles		
<b>Aligner</b> les éléments de formulaire. Contrôler les espacements.	1.5	Utiliser de manière précise et créative les styles CSS pour positionner et mettre en valeur les contenus en respect des maquettes visuelles.
<b>Styler l'interactivité : états des hyperliens et des éléments de formulaires.</b>	1.5	
Styler les <b>boutons radio</b> en les gardant accessibles au clavier.		
Utiliser des <b>sprites CSS</b> . <b>Styler les validations</b> de formulaire.	1.5	
<b>Organiser et documenter la feuille de styles</b> Bonne utilisation du <b>cadriciel</b> (normalize, utilitaires, ...) Définir des <b>variables CSS</b> et les utiliser		
<b>Utiliser le contrôle des versions GIT</b>		
	1.5	
<b>Total</b>	10	

### Modalités de remises

#### Sur Teams dans Devoirs

- Remise d'une archive .zip

#### Sur Github dans le compte personnel

- Avoir fourché (fork) le répertoire concours et sauvegarder (commit) au minimum 2 étapes

Groupes 1 et 2 – Mardi 30 mars minuit