Jérôme Fort

Activité : un formulaire interactif

Source : https://www.lamsade.dauphine.fr/~cbeji/fichiers/TP4_OI.pdf
https://www.pierre-giraud.com/html-css/cours-complet/test9.php

Contexte

Durée : 2h sur ordinateur en autonomie.

Les documents sont là pour bien définir l'objectif de l'activité, pour les détails techniques il est préférable d'encourager les participants à chercher eux-mêmes les réponses sur le web.

Les élèves sont regroupés par 2, pour favoriser l'entraide.

L'objectif, est de découvrir en HTML/CSS comment faire pour insérer des zones de texte, des boutons et des cases à cocher dans une page web.

Prérequis : HTML/CSS

Programme : Interactions entre l'homme et la machine sur le *Web*

Contenus	Capacités attendues	Commentaires
Formulaire d'une page *Web*	Analyser le fonctionnement d'un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.

NSI	Thème : Interactions entre l'homme et la machine sur le Web

Activité : Formulaire simple

A l'aide des documents 1,2 et 3, répondre aux questions suivantes

1. Pourquoi touche-t-on aux "limites du HTML" avec les formulaires ?

2. Lequel de ces éléments HTML sert à définir un formulaire ?
   a) L'élément form
   b) L'élément input
   c) L'élément label

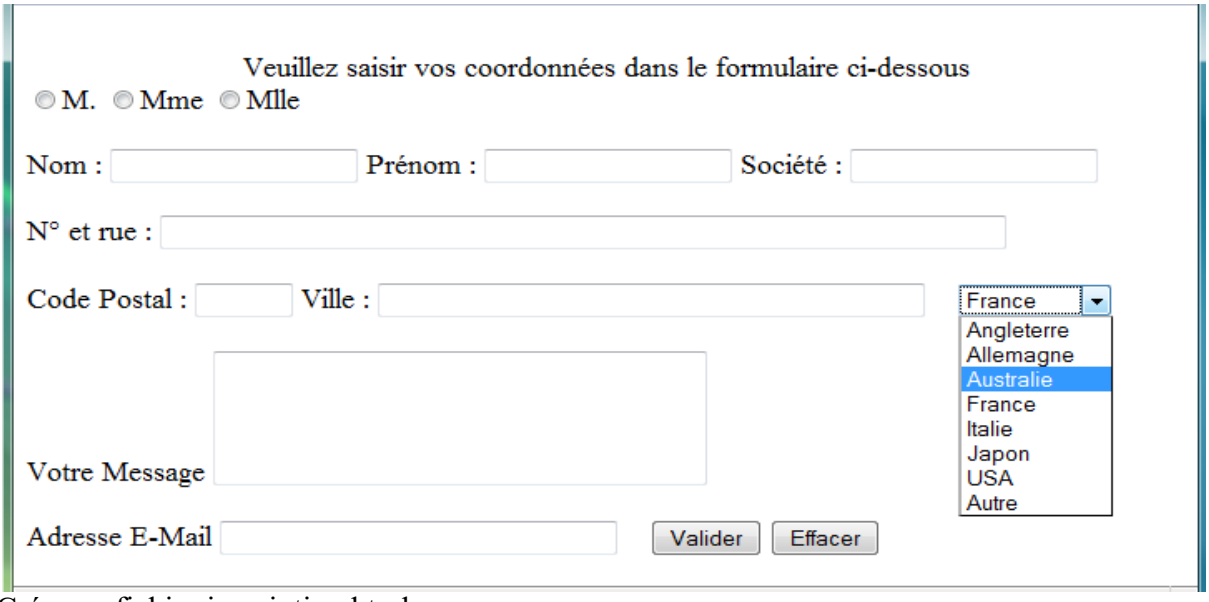
3. L'élément form doit obligatoirement être accompagné...
   a) D'un attribut method
   b) D'un attribut action
   c) De deux attributs method et action

4. Si je demande un mot de passe dans mon formulaire, j'utilise plutôt...
   a) La valeur get de l'attribut method
   b) La valeur post de l'attribut method
   c) Aucune de ces deux valeurs

5. Lequel de ces éléments sert à créer un champ de type "date" dans un formulaire ?
   a) L'élément input
   b) L'élément date
   c) L'élément select

6. On va utiliser l'élément fieldset pour...
   a) Donner un titre à notre formulaire
   b) Donner un titre à une partie de notre formulaire
   c) Créer des parties dans notre formulaire

Le Défi : Réaliser le formulaire ci-dessous



Créer un fichier inscription.html

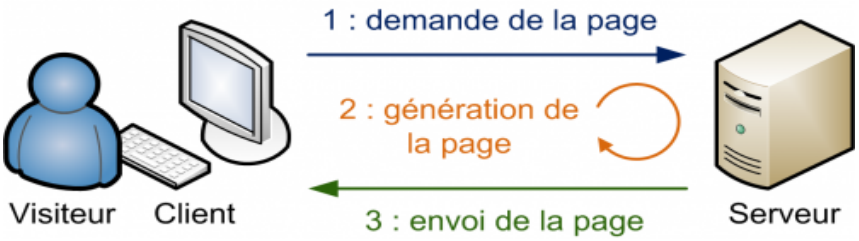
Pour les plus rapides, créer un fichier CSS, appelé "inscriptionstyle.css" qui modifiera l'apparence de cette page Web.

Document 1 : Qu'est-ce qu'un formulaire ?

Les formulaires enrichissent les pages web en permettant à l'utilisateur de saisir des informations sous différentes formes : zone de texte, cases à cocher, listes déroulantes, etc.

Classiquement, les données saisies par l'utilisateur dans un formulaire sont ensuite envoyées via le réseau à un **serveur web** qui va les traiter et renvoyer une réponse adaptée au navigateur client sous la forme d'une nouvelle page web.

Pour réaliser cette tâche, un serveur web emploie une technologie adaptée, comme par exemple le langage PHP.



Nous touchons, avec les formulaires, aux limites du HTML.
En effet, le HTML va nous permettre de créer notre formulaire sans problème. Cependant, nous n'allons ni pouvoir traiter les données envoyées par les utilisateurs, ni pouvoir les stocker.
Pour effectuer ces opérations, nous allons avoir besoin de connaître de nouveaux langages informatiques qui vont traiter les informations envoyées côté serveur, comme le PHP par exemple.

Document 2 : Créer un formulaire simple

Délimitation :
Pour insérer un formulaire dans une page HTML, vous devez pour commencer écrire une balise<form>. C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.
La balise <FORM> possède les attributs suivants :
name : il est parfois utile de prévoir un nom pour un traitement ultérieur du formulaire (name="nom"),
method : indique sous quelle forme seront envoyées les réponses. Cette transmission peut s'effectuer selon la méthode « POST » ou la méthode « GET ».
action : indique l'adresse d'envoi (soit une adresse d'un script de traitement situé sur un serveur (action="http://wwww….."), soit une adresse email (action="mailto:adresse.email@machine")

Champs de données :

On utilisera la balise **`<input />`** pour :
une zone de texte monoligne : `<input type="text" name="pseudo" />`,
un mot de passe : `<input type="password" name="pass" />` , un E-mail : `<input type="email" />`

un numéro de téléphone : `<input type="tel" />` ou une Date : `<input type="date" />`.

Des cases à cocher : `<input type="checkbox" name="frites" id="frites" />`
Ou des zones d'options : `<input type="radio" name="age" value="moins15" id="moins15" />`

On utilisera la balise **`<select />`** pour les listes déroulantes :

```
<select name="pays" id="pays">
    <option value="france">France</option>
    <option value="espagne">Espagne</option>
    …
</select>
```

Et enfin la balise <**`textarea`**>. pour un texte long.

```
<textarea cols="80" rows="4" name="article">
  Tapez ici votre article
</textarea>
```

Description du champ
Afin que l'utilisateur puisse savoir à quoi sert ce champ de texte il est nécessaire de lui donner une légende ou un titre. Les descriptions de champs sont faites à l'aide de la balise </label>

Regrouper les champs
Si votre formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises<fieldset>. Chaque<fieldset>peut contenir une légende avec la balise<legend>.

Rendre un champ obligatoire
Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut required.

Déclenchement de l'envoi
Jusqu'à présent nous avons rempli notre formulaire mais nous n'avons toujours aucun élément permettant de déclencher l'envoi. En ajoutant un champ de type "submit" (<input type="submit" />) vous obtiendrez un bouton qui déclenche la soumission des données. Le texte par défaut du bouton est déterminé par le navigateur, vous pouvez le changer grâce à l'attribut "value".

Document 3 : Cible du formulaire

Un formulaire est habituellement destiné à être traité par un programme sur le serveur.
L'attribut `action` permet de définir l'URL où sera envoyé le formulaire pour traitement.
Il vous est recommandé d'utiliser la méthode get quand aucune information envoyée n'est exagérément secrète (pas de mot de passe), que la quantité d'information est limitée et que deux soumissions avec les mêmes paramètres renvoient toujours la même information ; si un de ces trois critères n'est pas vérifié, vous devriez envisager la méthode post.

Faut-il utiliser plutôt la méthode GET, ou la méthode POST ?

La méthode GET ajoute les données à l'URL :
Avec cette méthode, les données du formulaire seront encodées dans une URL. Celle-ci est composée du nom de la page ou du script à charger avec les données de formulaire empaquetée dans une chaîne.
Les données sont séparées de l'adresse de la page pas le code ? et entre elles par le code &.
Ainsi si on envoie à page.html les valeurs "couleur bleu" et "forme rectangle", l'URL construite par le navigateur sera:

https://www.xul.fr/page.html?couleur=bleu&forme=rectangle

Les données de formulaire doivent être uniquement des codes ASCII. La taille d'une URL est limitée à par le serveur, souvent un peu plus de 2000 caractères, en comprenant les codes d'échappement.

La méthode POST n'a pas de taille limite :
Elle envoie un en-tête et un corps de message au serveur. Le corps est généralement constitué des données entrées dans le champ de formulaire par l'utilisateur.
Les données du formulaire n'apparaissent pas dans l'URL. En conséquence, il n'est pas possible de récupérer directement les données en JavaScript , il faut ajouter du code PHP dans la page:

Conclusion :
La méthode GET est la valeur de méthode par défaut. On l'utilise de préférence sauf si on ne veut pas que les paramètres soient ajoutés à l'URL.
Elle permet de récupérer les données passées à la page avec du code JavaScript.
La méthode POST est indispensable pour des codes non ASCII, des données de taille importante, et elle est recommandée pour modifier les données sur le serveur, et pour les données sensibles .
Si l'on utilise POST, on doit intégrer du code PHP (ou autre langage) dans la page ou les données seront utilisées.