TP n°2 : Introduction aux formulaires

1 Formulaires en HTML

1.1 Balises et attributs d'un formulaire

Il existe de nombreux possibilités de formulaires, on peut y entrer des mots clés pour effectuer des recherches, des informations qui seront ensuite envoyées à des bases de données, des valeurs pour effectuer des calculs etc. L'usage d'un formulaire posent cependant certaines problématiques :

1. Comment envoyer l'information entrée par l'utilisateur ?
2. Comment traiter cette information ?

La création d'un formulaire est conditionnée par l'usage de la balise `<form></form>` et a pour attributs possibles :

Attribut	Usage
`method`	permet de spécifier la méthode de transmission de l'information ; GET et POST sont les deux possibilités.
`action`	le nom d'un script à charger lorsque le formulaire est fourni, cela peut-être un fichier PHP ou HTML par exemple ; que se passe-t-il si l'on met une adresse URL ?
`onSubmit`	évènement déclenché lorsque le formulaire est soumis et avant l'exécution de l'action. Elle peut permettre de déterminer si l'action doit être exécutée ou non, par exemple si l'utilisateur a entré une information invalide, on pourra lui demander de la corriger.
`target`	indiquant où afficher la réponse après avoir envoyé le formulaire. `blank` pour ouvrir une nouvelle fenêtre et `top` pour rester dans la même fenêtre.
`id`	identifiant unique.
`name`	nom (contrairement à l'identifiant, il n'est pas unique).

Exercice 1. Parmi les attributs ci-dessus, quels sont ceux répondant aux deux problématiques ci-dessus ?

Remarque : si l'on veut faire figurer du texte dans le formulaire, il faut obligatoirement utiliser les balises `<p></p>`.

1.2 Créations des champs et des boutons

1.2.1 Champs et labels

Pour créer un **champ d'entrée**, il peut utiliser la balise `<input>` (on remarquera que celle-ci est orpheline), par exemple :

```
<input type="text" name="pseudo" />
```

L'attribut `type` permet de spécifier le type de la valeur attendue en entrée, cela peut être un texte comme dans l'exemple ci-dessus, des nombres, un mail, une date, etc. L'attribut `name` permet de donner un nom à la variable dans laquelle sera stockée la valeur entrée ; il est essentiel pour qu'elle puisse être traitée par le fichier PHP. La balise `<input>` peut aussi avoir pour attribut :

Attribut	Usage
`size`	spécifie la taille du champ.
`maxlength`	le nombre maximal de caractères que l'utilisateur peut entrer.
`value`	permet de préremplir le champ avec une valeur par défaut.
`placeholder`	indique ou donne un exemple de ce que doit entrer l'utilisateur.
`id`	son identifiant unique.

La balise `<input>` n'indique pas *a priori* ce que doit entrer l'utilisateur dans le champ crée par celle-ci. Si l'on veut afficher un texte **label** afin d'indiquer à l'utilisateur ce qu'il doit entrer, on peut utiliser la balise `<label></label>`. En reprenant l'exemple précédent, on pourrait faire :

```
<label>Votre pseudo</label> : <input type="text" name="pseudo" />
```

Toutefois, ainsi écrit, le code ne lie pas le label à son champ. Pour indiquer à quel champ correspond quel label, on utilise l'attribut `for` auquel on donnera la valeur de l'`id` du champ (donc du `<input>`). On n'utilise pas l'attribut `name` car celui-ci sert pour identifier la variable dans laquelle on stockera la valeur et non la balise qui, elle, est identifiée par son `id`. On aura par exemple :

```
<label for="pseudo">Votre pseudo</label> :
<input type="text" name="pseudo" id="pseudo" />
```

La balise `<input></input>` permet de créer des champs **monolignes**, si l'on veut créer des champs **multilignes** pour que l'utilisateur ait plus de place (pour écrire un commentaire par exemple), on utilisera `<textarea></textarea>`. De la même façon, on lui liera une balise `<label></label>`, par exemple

```
<label for="commentaire">Votre commentaire</label> :
<textarea type="text" name="commentaire" id="commentaire"></textarea>
```

1.2.2 Bouton

La création d'un bouton d'envoi se faire encore grâce à la balise `<input>`. Cinq attributs `type` permettent de spécifier le bouton.

`type`	**Usage**
`submit`	principal bouton d'envoi de formulaire ; le visiteur sera conduit à la page indiquée dans l'attribut `action` du formulaire.
`reset`	remise à zéro du formulaire.
`image`	pour mettre une image à la place du bouton ; à compléter avec `scr` pour indiquer quelle fichier charger.
`button`	bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page.

L'attribut `value` permet de changer le texte affiché sur le bouton. Un exemple complet de formulaire avec bouton d'envoi :

```
<form method="post" action="traitement.php">
     <p>
          <label for="pseudo">Votre pseudo :</label>
          <input type="text" name="pseudo" id="pseudo" placeholder="Ex : Buzz"/>
          <input type="submit" value="Vers l'infini et au delà !" />
     </p>
</form>
```

1.3 Méthodes GET et POST

1.3.1 La méthode GET

La méthode GET est limitée à 255 caractères. 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 empaquetées dans une chaîne. Les données sont séparées de l'adresse de la page pas le code `?` et entre elles par le code `&`. Dans le cas où une de ces variables serait une chaîne de caractères constituées de plusieurs mots, ceux-ci sont séparés par des `+`.

Par exemple, si on entre « loutre » dans le champ de recherche du formulaire défini par le code ci-dessus (le champ de recherche de la page principale du site exemple donc), la méthode GET prendra l'adresse de l'action puis y ajoutera un `?`, le nom de la variable à laquelle on a affecté la valeur entrée (avec un `=` pour l'affectation), ici `q=`, et enfin sa valeur, `loutre`, pour obtenir :

`https://search.lilo.org/?q=loutre`

Si on entre « raton laveur » dans le champ de recherche, l'adresse obtenue sera :

`https://search.lilo.org/?q=raton+laveur`

Exercice 2. Quelle sera l'adresse obtenue si l'utilisateur entre dans la barre de recherche du site exemple « dark vador » ?

Exercice 3. Dans les adresses ci dessous, identifier les attributs et les variables (et leurs valeurs) qui seraient nécessaires à l'usage de la méthode GET dans un formulaire de recherche.

1. `https://search.lilo.org/?q=raton+laveur&tab=images&page=1` ;
2. `https://duckduckgo.com/?q=dark+vador&t=ffab&ia=web`.

1.3.2 La méthode POST

La méthode POST, elle, n'a pas de limite de taille sur les données à envoyer et celles-ci ne transitent pas par la barre d'adresse. De fait, on ne peut pas récupérer les données directement avec JavaScript ; il faut d'abord passer par du PHP, que cela soit dans un fichier à part ou dans du code intégré au HTML.

1.3.3 Choix entre les méthodes GET et POST

La méthode GET est la valeur de méthode par défaut. Elle permet d'intégrer les données directement dans une URL et de les récupérer avec du code JavaScript.

La méthode POST est indispensable pour des codes non ASCII, des données de taille importante. Elle est recommandée pour modifier les données sur le serveur et pour des données sensibles qui ne doivent pas figurer dans l'URL. Si on l'utilise, on doit intégrer du code PHP (ou un autre langage) dans la page où les données seront utilisées.

Exercice 4. On a retranscrit ci dessous le code HTML du site exemple créer le formulaire permettant d'effectuer une recherche sur le Web.

```
<form method="GET" action="https://search.lilo.org/">
     <label for="recherche"><strong>Rechercher ailleurs sur le Web</strong></label>
     <input type="text" name="q" id="recherche"/>
     <input type="submit"/>
</form>
```

1. Quelle est la méthode utilisée ici ?
2. Quelle est l'action à effectuer lorsque le formulaire est soumis ?

2 Projet Web : partie 3

Il s'agit d'ajouter à votre site des formulaires. Il doit maintenant comporter (toujours en respectant le cahier des charges précédemment établi) les deux éléments suivants.

Formulaire de création de compte :

- le formulaire d'inscription pointe vers une page disant que le compte a été crée ;
- cette page doit au moins contenir les champs suivants :
  - nom * ;
  - prenom * ;
  - mail * + champ de confirmation * ;
  - pseudo ;
  - mot de passe * + champ de confirmation * ;
  - date de naissance * ;
  - validation des conditions d'utilisations (checkbox) * ;
  - choix du pays parmi une liste déroulante ;
- les items avec * ci-dessus seront à remplir obligatoirement par l'utilisateur ;
- chaque champ doit contenir un exemple ;
- le choix du pays est par défaut sur France.

Formulaire de contact :

- le formulaire de contact pointe vers une page disant que la demande a été prise en compte et qu'un mail de confirmation va être envoyé ;
- cette page doit au moins contenir les champs suivants :
  - mail * ;
  - sujet du contact parmi une liste déroulante (ex. : problème technique, suppression de compte, autre, etc) * ;
  - champ permettant à l'utilisateur d'expliquer sur plusieurs ligne sa demande * ;
- les items avec * ci-dessus seront à remplir obligatoirement par l'utilisateur ;

Les pages correspondant à ces formulaires devront être intégrées et liées de façon explicite au reste de votre site.

3 Ressources

- site exemple ;
- validateur HTML du W3C ;
- validateur CSS du W3C ;
- mémo HTML ;
- mémo CSS ;
- cours d'OpenClassRoom sur la création de formulaire.