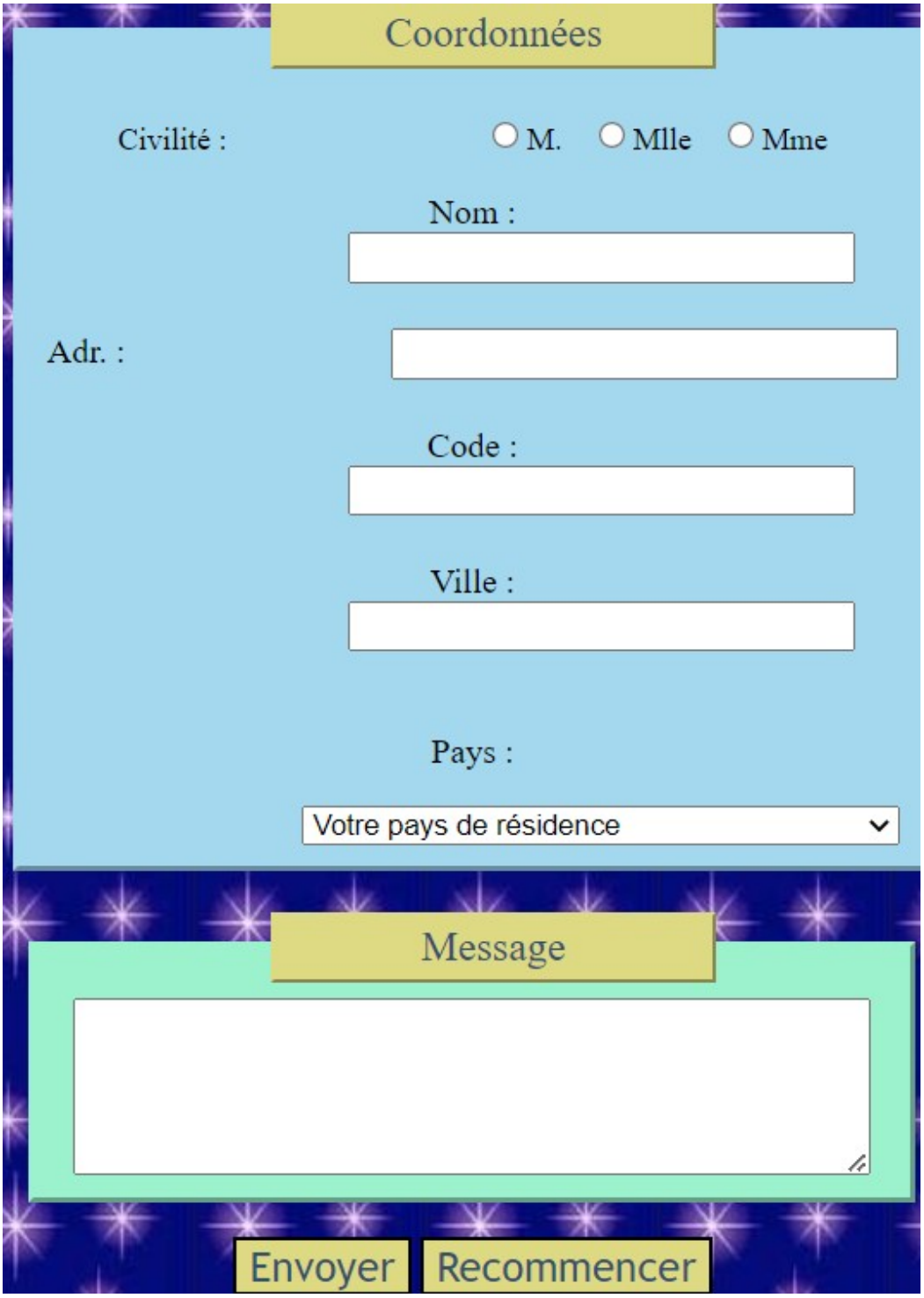
Faire un formulaire HTML5, CSS et PHP

1- Aperçu d'un exemple



2. les balises du formulaire

Nous allons tout d'abord créer le formulaire avec la balise appropriée qui est **<form>.**
Pour créer un formulaire, il faut obligatoirement lui spécifier une méthode d'envoi d'une part (à choisir entre GET et POST) ainsi que l'url du script qui va recevoir les données du formulaire. Celui se traduit par le code html :

```
1  <form method="post" action="traitement.php">
2  </form>
```

<form method="post" action="traitement.php">

</form>.
Sur cet exemple, les données seront envoyées au fichier "traitement.php", dont nous verrons l'utilisation plus loin.
Tous les éléments du formulaire seront placés entre la balise de début <form…> et la balise de fin </form>.

3. Création des blocs coordonnées et message

Comme vous l'avez vu sur l'image montrant le résultat final, nous allons diviser notre formulaire en deux blocs principaux : "coordonnées" et "message", avec un titre pour chaque bloc. Pour cela, nous allons créer deux éléments <p> sur lesquels nous allons prévoir une classe "titre" pour la mise en forme CSS par la suite.
Et nos deux groupes de champs vont être créés avec des éléments <fieldset> sur lesquels nous allons prévoir un attribut "id" pour la mise en forme plus tard.

Notre code devient :

```
<form method="post" action="traitement.php">
  <p class="titre">Coordonnées</p>
  <fieldset id="coordonnees">
  </fieldset>
  <p class="titre">Message</p>
  <fieldset id="message">
  </fieldset>
</form>
```

4. Créer des input radio (case à choisir)

Pour le choix de la civilité, nous allons créer des boutons de type radio, c'est-à-dire à choix unique. Lorsque vous ferez un choix, le choix précédemment sélectionné ne le sera plus. Ils se créent avec la balise <input type="radio", un nom name="civilite" et une valeur value="valeur". Vous pouvez même utiliser checked="checked" sur une des cases si vous souhaitez qu'elle soit sélectionnée par défaut. Puis fermez la balise avec /> . La valeur est ce que votre script de traitement du formulaire récupère, veillez donc à éviter tout caractère spécial…

Nous avons donc ajouté trois de ces cases (du même nom pour qu'un seul choix soit possible), précédé d'un élément <label>, qui est le label du champ comme son nom l'indique. Nous le mettrons pour chaque champ qui aura besoin d'une description le précédent. Enfin, on englobe tout cela d'un <p> avec un id que nous allons utiliser dans le CSS encore une fois, donc vous comprendrez son utilité plus tard. Notre code devient :

```
<form method="post" action="traitement.php">
  <p class="titre">Coordonnées</p>
  <fieldset id="coordonnees">
    <p id="civilite"><label>Civilité : </label>
      <input type="radio" name="civilite" value="M." />M.
      <input type="radio" name="civilite" value="Mlle" />Mlle
      <input type="radio" name="civilite" value="Mme" />Mme
    </p>
  </fieldset>
  <p class="titre">Message</p>
  <fieldset id="message">
  </fieldset>
</form>
```

5. Créer les champs de texte

Les champs de texte sont les champs à une ligne dans lesquels nous pouvons entrer du texte : nom, adresse, code postal, ville sur l'exemple. Ce champs de texte ne sont pas très compliqués à mettre en place, il faut tout d'abord créer un **<input type="text"** , puis attribuer un nom avec **name="nom"** et enfin une taille (longueur) avec **size="30"**. Mais une valeur par défaut est optionnellement possible avec **value="valeur"** et un nombre de caractères maximum avec,**maxlength="100"**par exemple.
Dans notre formulaire, nous allons faire précéder chacun des quatre champ de texte d'un label puis faire un retour à la ligne après chaque champ avec**
**, ce qui donnera ceci :

```
<form method="post" action="traitement.php">
<p class="titre">Coordonnées</p>
<fieldset id="coordonnees">
  <p id="civilite"><label>Civilité : </label><br />
    <input type="radio" name="civilite" value="M." />M.
    <input type="radio" name="civilite" value="Mlle" />Mlle
    <input type="radio" name="civilite" value="Mme" />Mme
  </p><br />
<label>Nom : </label>
    <input type="text" name="nom" size="30" /><br />
  <label>Adresse : </label>
    <input type="text" name="adresse" size="30" /><br />
  <label>Code postal : </label>
    <input type="text" name="codepostal" size="30" /><br />
  <label>Ville : </label>
    <input type="text" name="ville" size="30" /><br />
</fieldset>
<p class="titre">Message</p>
<fieldset id="message"><br />
</fieldset>
</form>
```

6. Créer une liste de choix

Nous allons ensuite créer une liste prédéfinie qui contiendra des pays, il faudra que l'utilisateur en choisisse un dans la liste.

Pour créer la liste il faut insérer une balise **<select** sur laquelle sur mettrons un nom **name="pays"** qui nous servira à récupérer sa valeur pour le traitement du formulaire. Ensuite pour chaque option de liste, il faut créer une balise **<option** avec une valeur **value="valeur"** et la fermer avec **</option>**. Entre les deux (<option> et </option>), il faut mettre la valeur de la liste affichée pour l'utilisateur. Puis en dernier lieu, il faut fermer la balise **</select>**.

Avec la liste rajoutée, on aura désormais ceci :

```
<form method="post" action="traitement.php">
<p class="titre">Coordonnées</p>
<fieldset id="coordonnees">
  <p id="civilite"><label>Civilité : </label>
    <input type="radio" name="civilite" value="M." />M.
    <input type="radio" name="civilite" value="Mlle" />Mlle
    <input type="radio" name="civilite" value="Mme" />Mme
  </p>
  <label>Nom : </label>
    <input type="text" name="nom" size="30" /><br />
  <label>Adresse : </label>
    <input type="text" name="adresse" size="30" /><br />
  <label>Code postal : </label>
    <input type="text" name="codepostal" size="30" /><br />
  <label>Ville : </label>
    <input type="text" name="ville" size="30" /><br />
  <label>Pays : </label>
    <select name="pays">
      <option value="france">France</option>
      <option value="belgique">Belgique</option>
      <option value="suisse">Suisse</option>
    </select>
</fieldset>
<p class="titre">Message</p>
<fieldset id="message">
</fieldset>
</form>
```

7. Créer des checkbox (cases à cocher)

Maintenant nous allons ajouter des cases à cocher. La différence avec les input de type radio, c'est que le choix peut être multiple ici. Vous pouvez cocher plusieurs cases comme bon vous semble. Il suffit de créer un élement <input type=”checkbox” et comme nous récupérerons les valeurs cochées dans une variable de type tableau, nous allons toutes les nommer

name=”interets[]” et leur donner une valeur avec value=”valeur”. Le code avec notre partie contenant les cases à cocher pour les centre d’intérêts sera donc réalisé avec ce code :

```
<form method="post" action="traitement.php">
<p class="titre">Coordonnées</p>
<fieldset id="coordonnees">
  <p id="civilite"><label>Civilité : </label>
    <input type="radio" name="civilite" value="M." />M.
    <input type="radio" name="civilite" value="Mlle" />Mlle
    <input type="radio" name="civilite" value="Mme" />Mme
  </p>
  <label>Nom : </label>
    <input type="text" name="nom" size="30" /><br />
  <label>Adresse : </label>
    <input type="text" name="adresse" size="30" /><br />
  <label>Code postal : </label>
    <input type="text" name="codepostal" size="30" /><br />
  <label>Ville : </label>
    <input type="text" name="ville" size="30" /><br />
  <label>Pays : </label>
    <select name="pays">
      <option value="france">France</option>
      <option value="belgique">Belgique</option>
      <option value="suisse">Suisse</option>
    </select>
  <p id="interets"><label>Centres intérêts : </label>
    <input type="checkbox" name="interets[]" value="sport" />Sport
    <input type="checkbox" name="interets[]" value="cinema" />Cinéma<br />
    <input type="checkbox" name="interets[]" value="internet" />Internet
    <input type="checkbox" name="interets[]" value="voyages" />Voyages
  </p>
</fieldset>
<p class="titre">Message</p>
<fieldset id="message">
</fieldset>
</form>
```

8. Créer une zône de texte

Une zône de texte est une zône où vous pouvez taper du texte librement et sur plusieurs lignes, au contraire du champ de texte qui est limité et sur une seule ligne. Sa création est assez simple, il faut créer une balise <textarea , lui ajouter un nom avec name=”comments” et éventuellement un nombre de lignes rows=”5″ et de colonnes cols=”40″. Enfin il faudra fermer avec </textarea>. Vous avez également la possibilité de mettre un texte par défaut dedans, pour cela le placer entre <textarea…> et </textarea>.

```
<form method="post" action="traitement.php">
<p class="titre">Coordonnées</p>
<fieldset id="coordonnees">
  <p id="civilite"><label>Civilité : </label>
    <input type="radio" name="civilite" value="M." />M.
    <input type="radio" name="civilite" value="Mlle" />Mlle
    <input type="radio" name="civilite" value="Mme" />Mme
  </p>
  <label>Nom : </label>
    <input type="text" name="nom" size="30" /><br />
  <label>Adresse : </label>
    <input type="text" name="adresse" size="30" /><br />
  <label>Code postal : </label>
    <input type="text" name="codepostal" size="30" /><br />
  <label>Ville : </label>
    <input type="text" name="ville" size="30" /><br />
  <label>Pays : </label>
    <select name="pays">
      <option value="france">France</option>
      <option value="belgique">Belgique</option>
      <option value="suisse">Suisse</option>
    </select>
 <p id="interets"><label>Centres intérêts : </label>
    <input type="checkbox" name="interets[]" value="sport" />Sport
    <input type="checkbox" name="interets[]" value="cinema" />Cinéma<br />
    <input type="checkbox" name="interets[]" value="internet" />Internet
    <input type="checkbox" name="interets[]" value="voyages" />Voyages
```

```
  </p>
</fieldset>
<p class="titre">Message</p>
<fieldset id="message">
<textarea name="comments" rows="5" cols="40"></textarea>
</fieldset>
</form>
```

9. Créer des boutons submit et reset

Il ne nous reste plus qu'à créer un bouton pour envoyer le formulaire, qui déclenchera l'action du formulaire (souvenez-vous le fichier traitement.php …). Pour cela il suffit de créer un <input type="submit" et de lui donner une valeur qui sera affichée sur le bouton avec value="Envoyer" par exemple, puis de fermer la balise bien sûr.
De la même façon nous pouvons aussi créer un bouton <input type="reset" qui servira à effacer tout le contenu des champs du formulaire pour recommencer à zéro. Nous mettrons alors la valeur value="Recommencer". Nous allons mettre ces deux boutons dans un   avec un id défini pour appliquer un style CSS dans la partie 2 du tutorial. Voici maintenant le code complet du formulaire :

```
<form method="post" action="traitement.php">
<p class="titre">Coordonnées</p>
<fieldset id="coordonnees">
  <p id="civilite"><label>Civilité : </label>
    <input type="radio" name="civilite" value="M." />M.
    <input type="radio" name="civilite" value="Mlle" />Mlle
    <input type="radio" name="civilite" value="Mme" />Mme
  </p>
  <label>Nom : </label>
    <input type="text" name="nom" size="30" /><br />
  <label>Adresse : </label>
    <input type="text" name="adresse" size="30" /><br />
  <label>Code postal : </label>
    <input type="text" name="codepostal" size="30" /><br />
  <label>Ville : </label>
    <input type="text" name="ville" size="30" /><br />
  <label>Pays : </label>
    <select name="pays">
      <option value="france">France</option>
      <option value="belgique">Belgique</option>
      <option value="suisse">Suisse</option>
    </select>
<p id="interets"><label>Centres intérêts : </label>
    <input type="checkbox" name="interets[]" value="sport" />Sport
    <input type="checkbox" name="interets[]" value="cinema" />Cinéma<br />
    <input type="checkbox" name="interets[]" value="internet" />Internet
    <input type="checkbox" name="interets[]" value="voyages" />Voyages
  </p>
</fieldset>
<p class="titre">Message</p>
<fieldset id="message">
  <textarea name="comments" rows="5" cols="40"></textarea>
</fieldset>
<p id="buttons">
  <input type="submit" value="Envoyer" />
  <input type="reset" value="Recommencer" />
</p>
</form>
```

II. Conclusion du tutorial

Vous venez de voir comment créer un formulaire HTML avec les principaux champs utilisés (champ de texte, case à cocher, liste, etc.). Mais pour le moment nous n’avons pas vu à quoi servent les id et class mis sur certains éléments, ni comment récupérer les données dans un script pour en faire quelque chose. La partie suivante va donc parler de la mise en forme CSS du formulaire, et la troisième partie portera sur le traitement de données du formulaire en PHP.

Faire un formulaire CSS, HTML, PHP (2)

Après avoir créé un formulaire en HTML, nous allons maintenant appliquer des styles CSS sur celui-ci afin de le mettre en forme, vous allez comprendre à quoi sert le CSS.

II. Explications

1. Création du fichier CSS

Vous commencerez par créer une page html de base (avec votre éditeur html ou manuellement)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>
  <body>
  </body>
</html>
```

Ensuite, vous allez créer un fichier que vous nommerez “formulaire.css” dans le même répertoire que le fichier html et vous l’attacherez à cette page html, ce qui donnera :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" /><link href=formula
        <title>Titre</title>
<link href="formulaire.css" rel="stylesheet" type="text/css" />
</head>

<body>
</body>
</html>
```

Entre les balises <body> et </body>, vous placerez le code de notre formulaire créé dans la page 1 précédente. Nous allons maintenant voir le contenu du fichier CSS.

2. Les titres des blocs : coordonnées et message

Commençons par les éléments <p class=”titre”>.
Nous allons leur mettre ceci :

– background : couleur de fond du titre

– color : couleur du texte

– padding : un petit espace pour ne pas écrire juste au bord du cadre du titre

– font-size : pour définir la taille du texte

– border : une bordure de 2 pixels avec un effet sortant (ombre) avec une couleur définie

– position : nous plaçon le titre en position relative à son conteneur

– margin-bottom : nous mettons une marge de bas négative, ce qui fait que le titre viendra s’encastrer dans l’elément en-dessous

– width : la largeur du cadre du titre

– margin-left : marge à gauche du titre

– margin-top : marge au-dessus du titre

```
p.titre {
  background:#DED983;
  color:#345071;
  padding:.2em .3em;
  font-size:1.2em;
  border:2px outset #DED983;
  position:relative;
  margin-bottom:-1em;
  width:10em;
  margin-left:1em;
  margin-top:1em;
}
```

Ensuite nous allons configurer un peu les jeux de champs en appliquant un style sur tous les “fieldset” :

– border : on spécifie de ne pas mettre de bordures (on la mettra sur chaque jeu de champ au cas par cas)

– margin-bottom : marge du dessous

– width : la largeur des cadres de champs

– padding-top : espace en haut, pour laisser la place aux cadres de titres.

```
fieldset {
  border:none;
  margin-bottom:1em;
  width:24em;
  padding-top:1.5em;
}
```

Puis un petit style général à toutes les listes déroulantes :
– margin-left : on spécifie une marge à gauche (pour laisser la place aux “labels” : légende des champs)
– margin-bottom : à 0, pas de marge en dessous.

```
select {
  margin-left:9em;
  margin-bottom:0;
}
```

Puis définissons un style pour tous les labels du bloc coordonnées. Remarquez comment il est possible d’atteindre un sous-élément en CSS en spécifiant tout d’abord l’élément principal (ici dont l’id est “coordonnees”), puis un élément qui se trouve dans ce dernier (les label). Cela évite de devoir mettre des id ou class à tous les labels pour rien, d’autant qu’ils auront tous le même style :

– position : position absolue (relative à aucun autre élément)

– font-size : on spécifie une taille de police, ici 90% de la taille par défaut

– padding-top : un espace au-dessus du texte du label

– left : spécifie la position horizontale du coin haut gauche de l’élément (20 pixels)

```
#coordonnees label {
  position:absolute;
  font-size:90%;
  padding-top:.2em;
  left:20px;
}
```

4. Le bloc message

Maintenant dans le fieldset “message”, cela sera plus vite fait puisque nous n’avons qu’une zone de texte. Nous allons juste :

– background : définition d’une couleur de fond pour le bloc

– border : mise en place d’une bordure “outset” (effet ombre sortante) comme pour le bloc précédent.

```
1  /* fieldset message */
2  fieldset#message {
3    background:#9DF2CE;
4    border:outset #9DF2CE;
5  }
```

5. Les cases radio pour la civilité

Tout comme pour les labels, nous allons passer la taille de la police de caractères à 90% pour le texte des boutons radio :

```
1  #civilite {
2    font-size:90%;
3  }
```

Et mettre une marge à gauche comme pour les autres éléments input du formulaire, pour laisser la place aux légendes :

```
1  #civilite input {
2    margin-left:9em;
3  }
```

Le problème en faisant ce que nous venons de faire ci-dessus, c’est que la marge à gauche va être appliquée aux trois boutons radio…
Nous allons donc utiliser une technique que nous n’avons pas vu encore. Elle consiste à faire un : ” #civilite input + input “, de façon à ce que le style soit appliqué uniquement si le champ input radio est précédé d’un autre. Futé non ?
Si c’est le cas et que le champ input n’est pas le premier de la ligne, nous appliquons une marge beaucoup moins grande (1em).

```
1  #civilite input + input {
2    margin-left:1em;
3  }
```

6. Les cases à cocher pour les centres d'intérêt

Pour les cases à cocher, comme d'habitude nous commencerons à mettre la police à 90%, puis nous utiliserons les mêmes techniques que précédemment pour mettre une marge conséquente au premier élément (case à cocher), et moins grande pour les suivants (#interets input +input).

Ensuite vient quelque chose de nouveau mais que vous comprendrez assez facilement si vous avez compris le coup du "input +input ", c'est le "br+input". Le raisonnement est le même : s'il y a un élément suivi d'un élément input, ce dernier aura une marge à gauche de 9em. Et oui car si l'on va à la ligne, il faut remettre la même marge que le premier input pour les aligner et laisser passer le label.

```
/* cases a cocher des centres d'interet */
#interets {
    font-size:90%;
}
#interets input {
    margin-left:9em;
}
#interets input +input {
    margin-left:1em;
}
#interets br+input {
   margin-left:9em;
}
```

7. La zône de texte du message

Pour la zône de texte du message, voici le style que nous allons appliquer (il n'y a qu'un élément textearea donc nous ne spécifions pas d'id ni class) : – font : la police de caractère ; sa taille sera 0.8em et elle sera de type "Trebuchet MS" (ou Verdana ou sans-serif si la ou les polices précédentes n'existent pas) – width : la largeur de la zone de texte – padding : nous allons espacer le texte de la zone de 0.2em de façon à ne pas écrire trop sur le bord.

```
/* zone de texte du message */
textarea {
   font:.8em "Trebuchet MS", Verdana, sans-serif;
   width:29em;
   padding:.2em;
}
```

8. Les boutons submit et reset

Pour finir, nous allons voir les boutons submit et reset. Vous l'aurez peut-être remarqué, ils n'ont pas de "class" dans le code html, alors comment allons nous les atteindre ? Et bien nous allons utiliser leur type en faisant input[type="submit"] pour le bouton submit et la meme chose avec le bouton reset. Nous allons leur appliquer le même style, à savoir :

– background : une couleur de fond

– font : une police de 1.2em en taille et de type "Trebuchet MS" (ou Verdana ou sans-serif si la ou les polices précédentes n'existent pas)

– color : une couleur du texte

NB : notez que l'on peut appliquer le même style à plusieurs éléments en les séparant d'un virgule comme ci-dessous.

```
/* les boutons submit et reset */
input[type="submit"], input[type="reset"] {
  background:#DED983;
  font:1.2em "Trebuchet MS", Verdana, sans-serif;
  color:#345071;
}
```

Enfin nous allons centrer les deux boutons.

Attention : ils vont logiquement se mettre au centre de la page web et non au centre du formulaire, pour remédier à cela vous pouvez faire un div conteneur qui contient tous les éléments de notre page web, en lui spécifiant la largeur du formulaire. Si les boutons sont dedans ils s'aligneront alors sur le formulaire et non la page.

```
p#buttons {
  text-align:center;
}
```

III. Conclusion du tutorial

Voilà, après avoir vu comment créer un formulaire HTML avec les principaux champs utilisés (champ de texte, case à cocher, liste, etc.), nous avons vu comment on peut mettre ceux ci en forme avec du CSS.

Dans le cours suivant, nous allons voir comment traiter les données qui sont envoyées par le formulaire, en PHP.

Faire un formulaire PHP, HTML, CSS (3)

I. Introduction au formulaire PHP

Après avoir créé un formulaire en HTML, et appliquer des styles CSS sur celui-ci afin de le mettre en forme, nous allons maintenant voir le fichier traitement.php que nous avions mis dans le champ action lors de la création de notre formulaire HTML. Ce script PHP va traiter les données du formulaire et les insérer dans une base de données MySQL, nous allons voir comment.

II. Explications

1. Création de la table MySQL

Commencez par créer une table que vous nommerez “formulaire” et les champs suivants dans celle ci :

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra
☐	id	int(11)			Non		auto_increment
☐	civilite	varchar(5)	latin1_swedish_ci		Non		
☐	nom	varchar(150)	latin1_swedish_ci		Non		
☐	adresse	varchar(255)	latin1_swedish_ci		Non		
☐	codepostal	varchar(10)	latin1_swedish_ci		Non		
☐	ville	varchar(150)	latin1_swedish_ci		Non		
☐	pays	varchar(150)	latin1_swedish_ci		Non		
☐	interets	varchar(255)	latin1_swedish_ci		Non		
☐	message	text	latin1_swedish_ci		Non		
☐	date	datetime			Non		

Tout cocher / Tout décocher *Pour la sélection :*

Elle contient tous les champs nécessaires pour stocker les données du formulaire + un id qui est un numéro unique pour identifier chaque enregistrement qui sera inséré + un champ “date” dans lequel vous pourrez stocker la date à laquelle le formulaire a été soumis si vous le souhaitez. Pour voir comment faire tout cela avec Phpmyadmin, consultez phpmyadmin pour gérer vos bases MySQL.

2. Connexion à la base de données MySQL

Comme nous allons insérer les données du formulaire dans une base de données MySQL, nous allons commencer par nous connecter à celle-ci.

Commençons par définir quelques paramètres indispensable pour se connecter : le serveur, le nom d’utilisateur, le mot de passe, et le nom de la base de données.

```
1  // Parametres mysql à remplacer par les vôtres
2  define('DB_SERVER', 'localhost'); // serveur mysql
3  define('DB_SERVER_USERNAME', 'root'); // nom d'utilisateur
4  define('DB_SERVER_PASSWORD', 'motdepasse'); // mot de passe
5  define('DB_DATABASE', 'monsite'); // nom de la base
```

Sur une ligne en PHP, tout ce qui est précédé par un double slash // est pris comme un commentaire de code, donc vous mettez ce que vous voulez.
Remarquez aussi l'utilisation de la fonction define() qui permet de définir une constante (valeur qui ne changera jamais à la différence d'une variable)
Une fois ces valeurs constantes définies, vous allez vous connecter à votre serveur de base de données MySQL en utilisant la fonction mysql_connect() puis choisir une base de données à l'aide de mysql_select_db().

```
1  // Connexion au serveur mysql
2  $connect = mysql_connect(DB_SERVER, DB_SERVER_USERNAME,
3  DB_SERVER_PASSWORD)
4  or die('Impossible de se connecter : ' . mysql_error());
5  // sélection de la base de données
6  mysql_select_db(DB_DATABASE, $connect);
```

La ligne commençant par "or" peut vous paraître incompréhensible au premier abord. Si c'est le cas, sachez que le "or" est un "ou logique" et que la fonction mysql_connect() retourne vrai ou faux, suivant si elle réussit ou échoue. Ici si elle retourne "vrai", la ligne du "or" ne se sera pas exécutée. Mais si elle retourne "faux", la ligne sera exécutée et on écrira un message d'erreur avec l'erreur précise qui a eu lieu avec la fonction mysql_error().

3. Vérification du remplissage des champs

Nous allons maintenant vérifier que les champs obligatoire sont bien remplis, ce qui est une étape classique sur un formulaire. Commençons avant par définir quelques variables :

```
1  $msg_erreur = "Erreur. Les champs suivants doivent être obligatoirement remplis :
2  <br/><br/>";
3  $msg_ok = "Votre demande a bien été prise en compte.";
4  $message = $msg_erreur;
```

La variable $msg_erreur contient le début du message d'erreur si un champ obligatoire n'est pas rempli. Nous lui ajouterons le nom des champs à remplir après. La variable $msg_ok contient pour sa part le message qui confirme que le formulaire a bien été envoyé et pris en compte. Nous définissons aussi une variable $message qui prend pour le moment la valeur de $msg_erreur , vous comprendrez pourquoi par la suite. En effet, nous allons maintenant vérifier que certains champs obligatoires (selon notre choix) sont bien remplis, et si ce n'est pas le cas, pour chaque champ non remplit nous allons ajouté son nom dans la variable $message. Cela nous permettra de l'afficher à l'utilisateur. Pour vérifier cela, nous utiliserons la fonction PHP empty() qui retourne "vrai" si la variable passée en paramètre est vide, ou "faux"

dans le cas contraire. Et pour ajouter le nom du champ vide dans la variable $message, nous allons concaténer les deux chaînes (concaténer veut dire ajouter bout à bout) à l'aide d'un ".=". Le point devant le "égal" permet d'ajouter la chaîne à la variable au lieu de remplacer.

```
1   if (empty($_POST['civilite']))
2     $message .= "Votre civilité<br/>";
3   if (empty($_POST['nom']))
4     $message .= "Votre nom<br/>";
5   if (empty($_POST['adresse']))
6     $message .= "Votre adresse<br/>";
7   if (empty($_POST['codepostal']))
8     $message .= "Votre code postal<br/>";
9   if (empty($_POST['ville']))
10    $message .= "Votre ville<br/>";
11  if (empty($_POST['comments']))
12    $message .= "Votre message<br/>";
```

4. Formatage et insertion des données en base de données MySQL Si l'utilisateur n'a pas remplit correctement un des champs obligatoires, il vous faudra commencer par lui afficher le message d'erreur. Mais comment allons-nous savoir si c'est le cas ? Et bien c'est là que vous allez voir l'intérêt de la variable $message , elle sert à conserver la variable $msg_erreur intacte afin de faire une comparaison. L'idée est la suivante : "si le contenu de la variable $message et plus grand que celui de la variable $msg_erreur, c'est que nous venons d'ajouter des champs non remplis", et cela implique d'afficher l'erreur. **NB :** nous aurions pu aussi revérifier que tous les champs obligatoires ne sont pas vides pour faire la même chose, mais cela va faire beaucoup d'appels à la fonction empty() que l'on peut éviter. Voilà ce que cela donne :

```
1   if (strlen($message) > strlen($msg_erreur)) {
2     echo $message;
3   }
```

Par contre si les champs sont bien remplis, vous allez pouvoir insérer leurs valeurs dans la base de données MySQL. Mais avant toute chose, attention à la sécurité car un utilisateur peut mettre du code dans un champ et mettre à mal votre serveur MySQL (voir le tutorial Attaques par injection SQL). C'est pour cela que nous

utiliserons la fonction mysql_real_escape_string() qui protège les données avant insertion et notamment les apostrophes, guillemets, etc.

La boucle foreach() va passer toutes la variables $_POST envoyées par le formulaire en revue. Pour chacune d'elle nous allons créer une variable spécifique avec $$index, nettoyer sa valeur avec trim() (en enlevant notamment les espaces de fin) et sécuriser sa valeur avec mysql_real_escape_string().

```
1  } else {
2    foreach($_POST as $index => $valeur) {
3      $$index = mysql_real_escape_string(trim($valeur));
4    }
```

Exemple : pour la variable $_POST['adresse'] qui contiendra votre adresse entrée dans le formulaire et si $_POST['adresse'] = "45 rue de l'abbé ", cela va créer une variable $adresse dont la valeur sera "45 rue de l\'abbé".
Les données sont maintenant presque prêtes à être insérées dans la base de données. Il faut juste s'occuper des centre d'intérêts du formulaire avant. En effet pour ceux ci, il est possible de cocher plusieurs valeurs qui sont stockées dans une variable tableau. Or dans notre base de données, nous n'aurons qu'un champ pour les stocker (inutile d'en faire 4 à chaque fois pour en laisser la moitié de vide). Nous allons donc concaténer les centres d'intérêts en les séparant d'une virgule puis un espace afin qu'ils tiennent sur une ligne.

```
1  $interets = $_POST['interets'];
2  $sqlinterets = '';
3  for ($i=0; $i<count($interets); $i++)
4  {
5    $sqlinterets .= $interets[$i];
6    $sqlinterets .= ', ';
7  }
```

Exemple : si la personne coche "sport" et "voyages" comme centres d'intérêt, $_POST['interets'] contiendra un tableau à deux éléments que nous parcourons et ajoutons chaque élément à la variable $sqlinterets. Au final, elle contiendra "sport, voyages, ".

Il ne reste plus qu'à insérer les données dans la table "formulaire" (ou autre nom de table à condition de la changer dans le code suivant) avec une requête sql de type "INSERT" et la fonction mysql_query().
Puis vous vérifierez sur la requête s'est bien effectuée avec la valeur de retour $res. Si c'est le cas, vous affichez $msg_ok, sinon vous affichez l'erreur qui a eu lieu dans MySQL.

```
  $sql = "INSERT INTO formulaire VALUES ('', '".$civilite."', '".$nom."',
    '".$adresse."', '".$codepostal."', '".$ville."', '".$pays."',
    '".$sqlinterets."', '".$comments."', now())";
  $res = mysql_query($sql);

  if ($res) {
    echo $msg_ok;
  } else {
    echo mysql_error();
  }

}
```

5. Le code final de traitement.php

Voici enfin le récapitulatif du code complet que nous venons de voir :

```php
<?php
// Parametres mysql à remplacer par les vôtres
define('DB_SERVER', 'localhost'); // serveur mysql
define('DB_SERVER_USERNAME', 'root'); // nom d'utilisateur
define('DB_SERVER_PASSWORD', 'motdepasse'); // mot de passe
define('DB_DATABASE', 'telechargements'); // nom de la base
// Connexion au serveur mysql
$connect = mysql_connect(DB_SERVER, DB_SERVER_USERNAME,
DB_SERVER_PASSWORD)
or die('Impossible de se connecter : ' . mysql_error());
// sélection de la base de données
mysql_select_db(DB_DATABASE, $connect);
$msg_erreur = "Erreur. Les champs suivants doivent être obligatoirement remplis:
<br/><br/>";
$msg_ok = "Votre demande a bien été prise en compte.";
$message = $msg_erreur;
// vérification des champs
if (empty($_POST['civilite']))
  $message .= "Votre civilité<br/>";
if (empty($_POST['nom']))
  $message .= "Votre nom<br/>";
if (empty($_POST['adresse']))
  $message .= "Votre adresse<br/>";
if (empty($_POST['codepostal']))
  $message .= "Votre code postal<br/>";
if (empty($_POST['ville']))
  $message .= "Votre ville<br/>";
if (empty($_POST['comments']))
```

```php
if (empty($_POST['comments']))
  $message .= "Votre message<br/>";

// si un champ est vide, on affiche le message d'erreur
if (strlen($message) > strlen($msg_erreur)) {

  echo $message;

// sinon c'est ok
} else {

  foreach ($_POST as $index => $valeur) {
    $$index = mysql_real_escape_string(trim($valeur));
  }

  $interets = $_POST['interets'];
  $sqlinterets = '';
  for ($i=0; $i<count($interets); $i++)
  {
    $sqlinterets .= $interets[$i];
    $sqlinterets .= ', ';
  }

  $sql = "INSERT INTO formulaire VALUES ('', '".$civilite."', '".$nom."',
    '".$adresse."', '".$codepostal."', '".$ville."', '".$pays."',
    '".$sqlinterets."', '".$comments."', now())";
  $res = mysql_query($sql);

  if ($res) {
    echo $msg_ok;
  } else {
    echo mysql_error();
  }

}
?>
```

III. Conclusion

Voilà, après avoir vu comment créer un formulaire HTML et mettre en forme celui-ci en CSS, vous venez de voir comment traiter les données du formulaire en PHP par l'exemple.