Les formulaires HTML

Rédigé par: Jean-Luc Colson

Date première rédaction: Mai 2020

SUIVI DES MODIFICATIONS A LA FICHE

Série	Date	Page(s) modifiée(s)	Raison

Contents

Présentation des formulaires HTML et de l'élément form 4
Définition et rôle des formulaires HTML 4
L'élément form et ses attributs 4
Création d'un premier formulaire simple en HTML 5
Création des champs de formulaire avec l'élément input 5
Ajout d'indications sur les données attendues avec l'élément label 7
Ajout du bouton d'envoi de données 7
Identification des données du formulaire 8
Les éléments des formulaires 11
Les éléments de formulaires HTML 11
L'élément input et les valeurs de l'attribut type 12
L'élément textarea 12
Les éléments select, option et optgroup 13
Les éléments fieldset et legend 13
Création d'un formulaire HTML complet 13
Attributs des formulaires et sécurité 20
Le problème des données utilisateurs 20
Quelles solutions pour sécuriser nos formulaires ? 20
Vérification et validation des données de formulaire dans le navigateur 21

Présentation des formulaires HTML et de l'élément form

Nous allons aborder dans cette nouvelle partie une partie essentielle du HTML qui va consister en la création de formulaires.

Nous allons donc définir ensemble ce qu'est un formulaire HTML et à quoi vont servir les formulaires puis nous allons apprendre à créer des formulaires plus ou moins complexes.

Définition et rôle des formulaires HTML

Les formulaires HTML vont permettre à nos visiteurs de nous envoyer des données que nous allons ensuite pouvoir manipuler et / ou stocker.

Nous allons utiliser les formulaires pour permettre à des utilisateurs de s'inscrire sur notre site (formulaires d'inscription), de se connecter (formulaire de connexion), de nous envoyer des messages (formulaire de contact), de laisser des commentaires, etc.

Vous l'aurez donc compris : les formulaires se révèlent indispensables et incontournables dans de nombreuses situations.

Les formulaires HTML vont pouvoir être composés de champs de texte (cas d'un champ de formulaire demandant à un utilisateur de renseigner son adresse mail pour se connecter ou pour s'inscrire sur le site par exemple), de listes d'options (choix d'un pays dans une liste de pays par exemple), de cases à cocher, etc.

Cependant, vous devez bien comprendre ici qu'on touche avec les formulaires aux limites du langage HTML. En effet, nous allons tout à fait pouvoir construire nos formulaires en HTML, mais le HTML ne va nous permettre ni de recueillir les données envoyées par nos visiteurs, ni de les manipuler, ni de les stocker.

Pour réaliser ces différentes opérations, il faudra utiliser d'autres types de langages comme le PHP (pour la manipulation des données) et le MySQL (pour le stockage) par exemple qui vont s'exécuter côté serveur.

L'élément form et ses attributs

Pour créer un formulaire, nous allons utiliser l'élément HTML form. Cet élément form va avoir besoin de deux attributs pour fonctionner normalement : les attributs method et action.

L'attribut method va indiquer comment doivent être envoyées les données saisies par l'utilisateur. Cet attribut peut prendre deux valeurs : get et post.

Que signifient ces deux valeurs et laquelle choisir ? Les valeurs get et post vont déterminer la méthode de transit des données du formulaire. En choisissant get, on indique que les données

doivent transiter via l'URL (sous forme de paramètres) tandis qu'en choisissant la valeur post on indique qu'on veut envoyer les données du formulaire via transaction post HTTP.

Concrètement, si l'on choisit l'envoi via l'URL (avec la valeur get), nous serons limités dans la quantité de données pouvant être envoyées et surtout les données vont être envoyées en clair. Evitez donc absolument d'utiliser cette méthode si vous demandez des mots de passe ou toute information sensible dans votre formulaire.

Cette méthode de transit est généralement utilisée lors de l'affichage de produits triés sur un site e-commerce (car oui, les options de tris sont des éléments de formulaires avant tout !). Regardez bien les URL la prochaine fois que vous allez sur un site e-commerce après avoir fait un tri : si vous retrouvez des éléments de votre tri dedans c'est qu'un get a été utilisé.

En choisissant l'envoi de données via post transaction HTTP (avec la valeur post), nous ne sommes plus limités dans la quantité de données pouvant être envoyées et les données ne sont visibles par personne. C'est donc généralement la méthode que nous utiliserons pour l'envoi véritablement de données que l'on souhaite stocker (création d'un compte client, etc.).

L'attribut action va lui nous servir à préciser l'adresse de la page qui va traiter les données. Je vous rappelle ici qu'on ne va pas pouvoir manipuler les données des formulaires en HTML mais que nous allons être obligé d'utiliser un autre langage comme le PHP pour cela.

Généralement, nous enverrons les données reçues vers un fichier PHP dont le but sera de traiter ces données justement. Nous n'allons bien évidemment pas créer ce fichier dans ce cours, mais je vous propose pour les exemples suivants de faire « comme si » ce fichier existait. Nous pourrons l'appeler form.php.

Création d'un premier formulaire simple en HTML

Il est temps de passer à la pratique et de créer un premier formulaire simple en HTML. Dans ce formulaire, nous allons demander trois choses à l'utilisateur :

- Un pseudonyme à l'utilisateur ;
- Un mot de passe ;
- Une adresse mail.

Création des champs de formulaire avec l'élément input

Pour faire cela, nous allons avoir besoin de créer trois champs de formulaires dans lesquels l'utilisateur pourra remplir les informations demandées. Pour créer ces champs, nous allons utiliser des éléments HTML input.

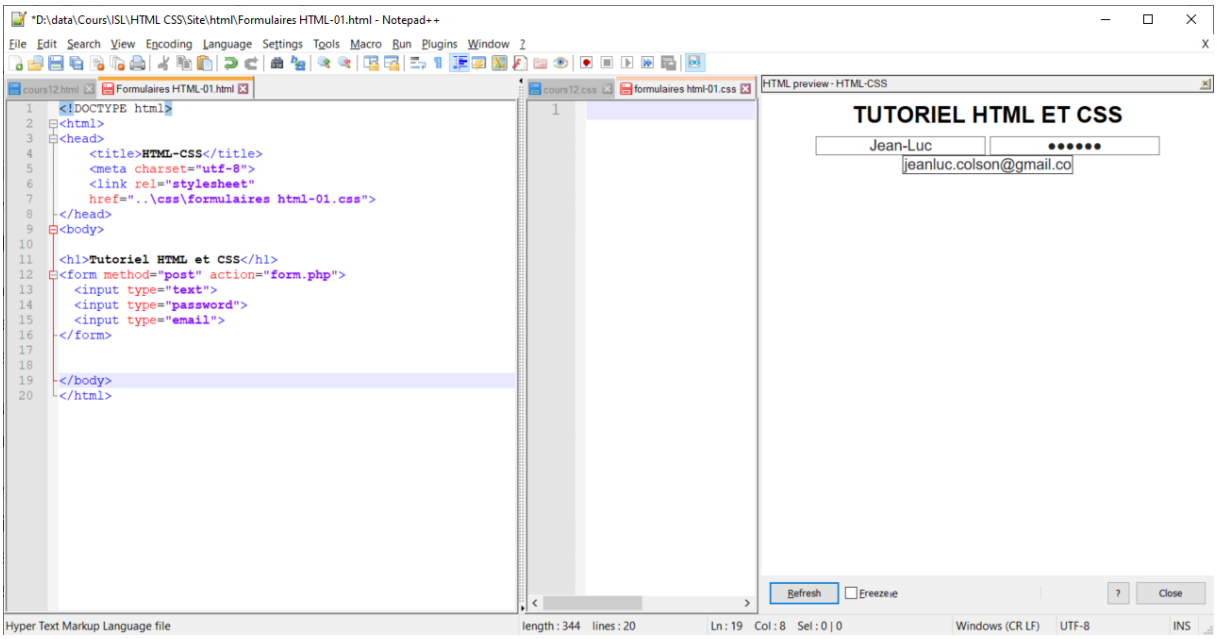
L'élément HTML input est un élément qui va permettre à l'utilisateur d'envoyer des données. Il se présente sous la forme d'une balise orpheline et va obligatoirement posséder un attribut type auquel on va pouvoir donner de nombreuses valeurs.

La valeur passée à l'attribut type va déterminer le type de données que l'utilisateur va pouvoir envoyer : un texte avec input type="text", une adresse mail avec input type="email", une date avec input type="date", etc.

Si le format de données entrées par l'utilisateur ne correspond pas à ce qui est attendu, alors il ne pourra pas envoyer le formulaire et un message d'erreur s'affichera selon le navigateur utilisé.

Notez également que les navigateurs proposent aujourd'hui des présentations différentes en fonction du type d'input et notamment sur mobile : pour un input type="date", par exemple, un calendrier sera souvent affiché pour aider l'utilisateur à choisir une date.

La base de notre formulaire va donc ressembler à cela :



Comme vous pouvez le voir, nos trois champs de formulaires sont bien créés. Si vous placez le curseur de votre souris dans un champ et que vous commencez à écrire, vous devriez voir que le navigateur comprend bien le type de données attendu dans chaque champ : par exemple, le texte inscrit dans le champ demandant un mot de passe par exemple devrait s'afficher sous forme d'étoiles.

De plus, il est également possible que votre navigateur vous propose une auto-complétion des champs si vous avez déjà rempli des formulaires demandant des types de données similaires sur un autre site précédemment.

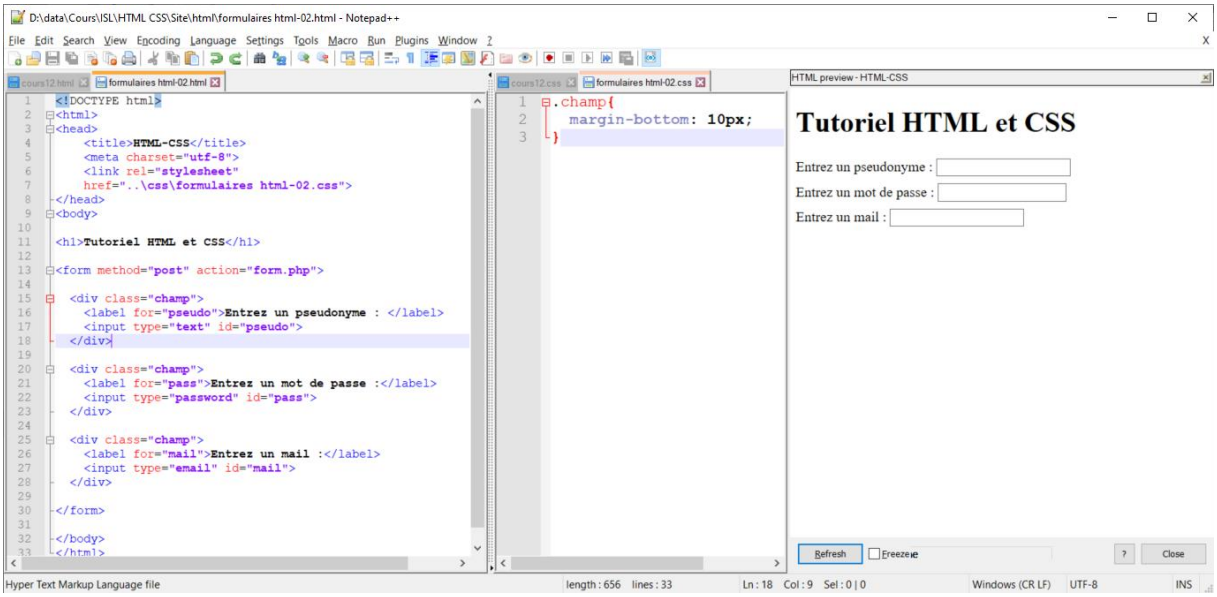
En l'état, notre formulaire n'est cependant pas exploitable pour la simple et bonne raison que l'utilisateur ne possède pas d'indication sur ce qu'il doit rentrer dans chaque champ et également car pour le moment notre formulaire ne possède pas de bouton d'envoi des données !

Ajout d'indications sur les données attendues avec l'élément label

Pour donner des indications sur les données attendues dans chaque champ aux utilisateurs, nous allons utiliser des éléments label. Cet élément va permettre d'attribuer un libellé (c'est-à-dire une étiquette ou une description) à chaque champ de notre formulaire.

Pour des raisons d'ergonomie et de cohérence, il est considéré comme une bonne pratique de lier chaque label à son input correspondant. Ainsi, lorsque l'utilisateur va cliquer sur le label, le curseur de la souris va automatiquement être placé dans l'input correspondant.

Pour lier un label et un input ensemble, nous allons attribuer un attribut id unique à chacun de nos éléments input pour les identifier de manière formelle et allons également rajouter un attribut for à chacun de nos label qui devra avoir la même valeur que l'id de l'input auquel il doit être lié.



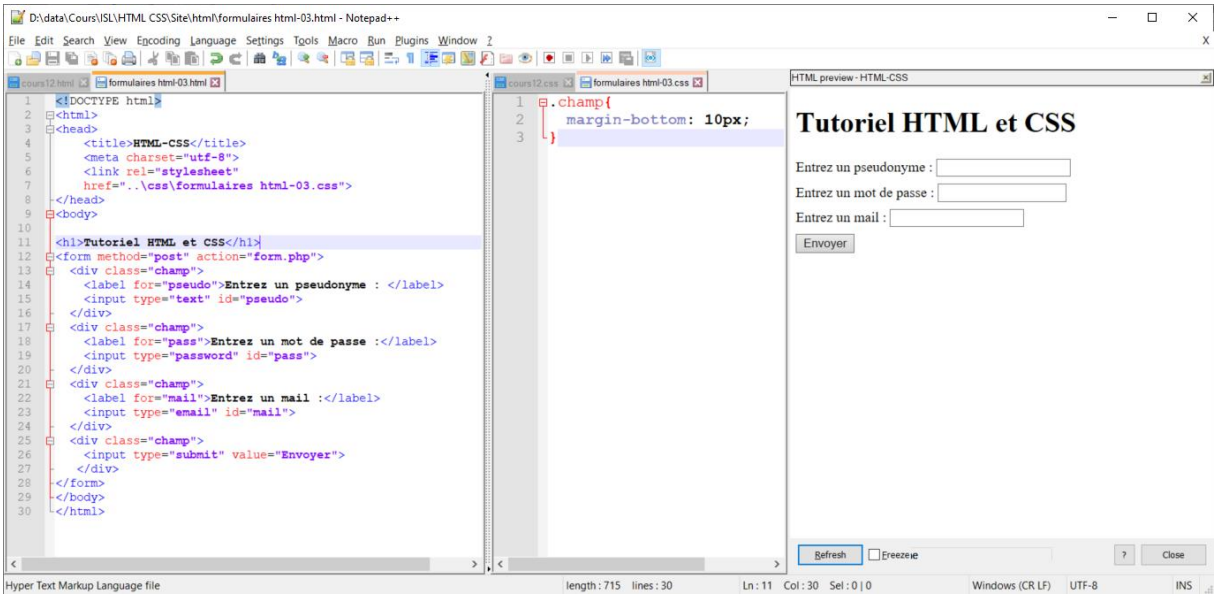
Ici, j'en ai profité pour placer mes couples label + input dans des div afin que chacun d'entre eux soit sur une ligne à eux et j'ai également passé une margin-bottom : 10px aux div.

Nous en avons fini avec les champs de notre premier formulaire HTML en soi et il ne nous reste donc plus qu'à nous préoccuper de l'envoi des données.

Ajout du bouton d'envoi de données

Pour permettre l'envoi de ces données vers notre page form.php, il va nous falloir créer un bouton de soumission de formulaire. Pour cela, nous allons à nouveau tout simplement utiliser un élément input mais cette fois-ci de type input type="submit".

Cet input un peu spécial va se présenter sous forme de « bouton » et nous n'allons pas avoir besoin de lui ajouter de label. A la place, nous allons utiliser un attribut value et lui passer en valeur le texte que doit contenir notre bouton (« soumettre », « valider », ou « envoyer » par exemple).



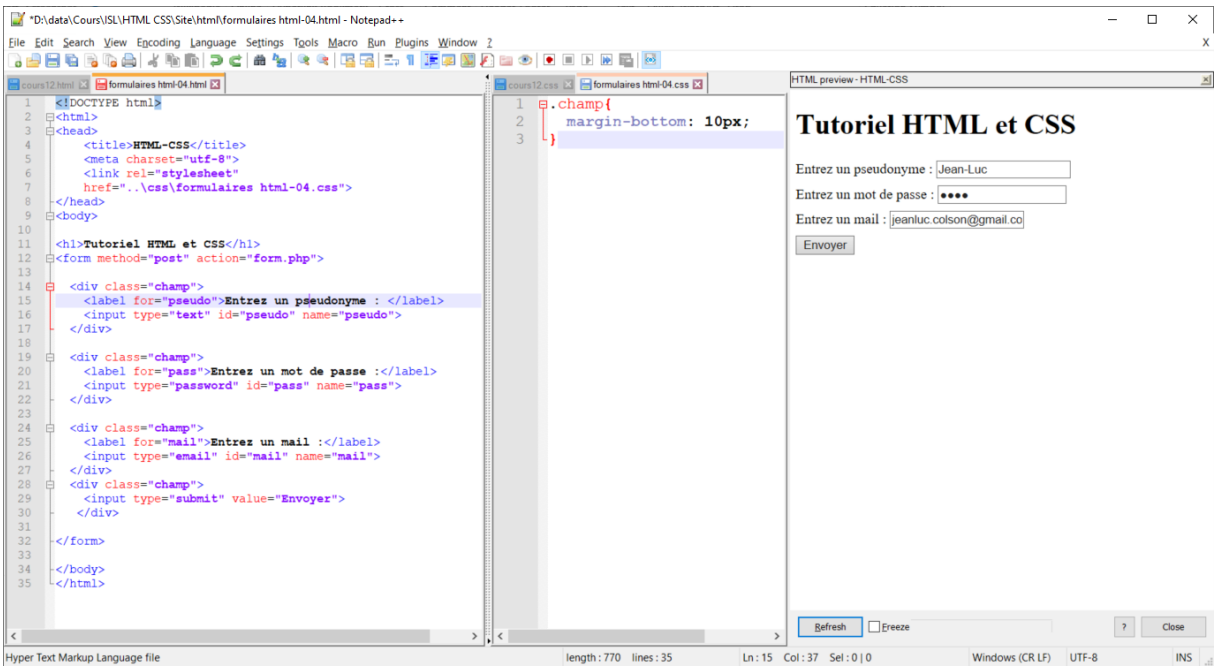
Identification des données du formulaire

En l'état, notre formulaire n'est pas encore tout à fait utilisable. En effet, celui-ci fonctionne côté « façade » et va bien permettre à un utilisateur de remplir des données et de les envoyer.

Cependant, pour le moment, nous ne pourrions pas manipuler les données envoyées car elles n'ont pas été nommées : les données vont bien être envoyées mais il va être impossible pour nous de définir à quel champ correspond chaque donnée envoyée.

Pour identifier les données et pouvoir ensuite les manipuler en PHP ou autre, nous allons devoir ajouter un attribut name qui doit également posséder une valeur unique à chaque élément de formulaire demandant des données à l'utilisateur.

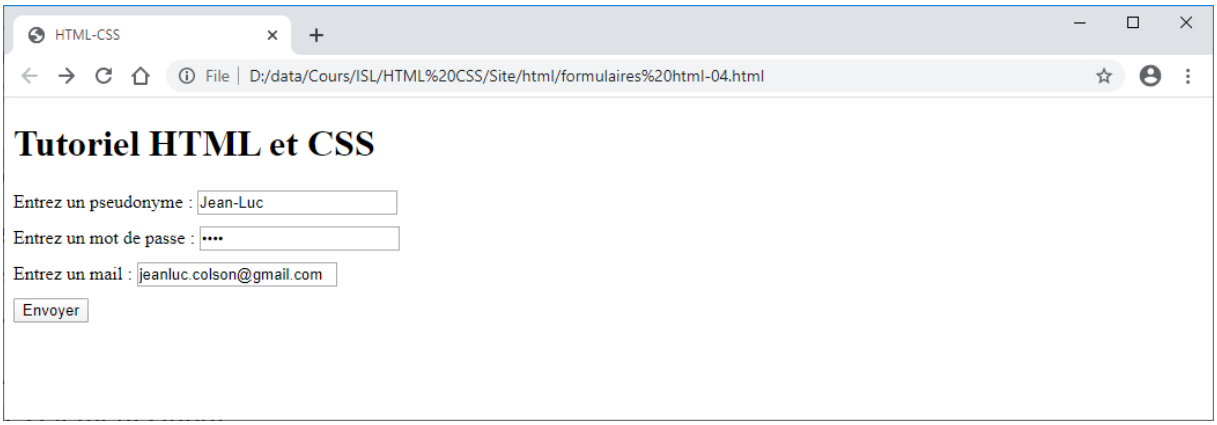
Notez bien ici que chaque attribut name doit posséder une valeur unique par rapport aux autres attributs name mais rien ne nous empêche de choisir la même valeur pour un attribut name et un attribut id par exemple puisqu'il n'y a pas de risque de confusion étant donné que ce sont deux attributs totalement différents.



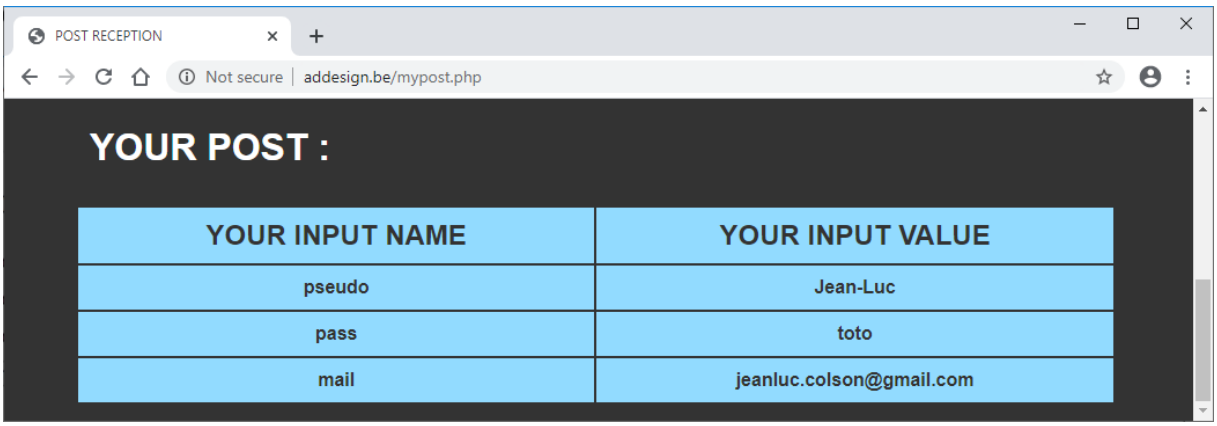
Ça y est, notre premier formulaire HTML est maintenant tout à fait fonctionnel. Expliquons rapidement la logique derrière ce formulaire, c'est-à-dire le fonctionnement de notre formulaire HTML.

Lorsqu'un utilisateur arrive sur notre page, il va remplir les différents champs de formulaire créés avec des éléments input.

Ensuite, il va cliquer sur le bouton d'envoi du formulaire. Les données du formulaire vont alors être envoyées via transaction HTTP de type post vers la page form.php.



Rem : la page form.php est ici purement fictive, il convient pour tester l'envoi des valeurs de faire pointer l'attribut action vers une vraie page dynamique sur un serveur qui elle affichera les valeurs envoyées. La création de ce type de page dépasse le cadre de ce cours.



Dans cette page form.php, nous allons ensuite pouvoir manipuler les différentes données du formulaire qu'on va pouvoir identifier grâce aux attributs name donnés à chaque input.

Encore une fois, l'objet de ce tutoriel n'est pas de présenter la manipulation des données en PHP ni le stockage mais bien de nous concentrer sur la création des formulaires en soi c'est-à-dire sur la partie HTML des formulaires.

Les éléments des formulaires

En HTML, nous avons de nombreux éléments spécifiques aux formulaires qui vont nous permettre de définir le formulaire en soi, de créer des zones de saisie de texte courtes ou longues, de proposer des zones d'options à nos utilisateurs, etc.

Dans la leçon précédente, nous nous sommes concentrés sur l'éléments input qui est de loin le plus utilisé car son attribut type permet de créer toutes sortes de champs et le rend très puissant mais il en existe bien d'autres.

L'objet de cette leçon va être de vous présenter la plupart des éléments de formulaire ainsi que les valeurs les plus courantes qu'on va pouvoir passer à l'attribut type de l'élément input.

Les éléments de formulaires HTML

Commençons avec la liste des éléments que l'on va pouvoir utiliser dans nos formulaires HTML ainsi que leur rôle :

Elément	**Définition**
form	Définit un formulaire
input	Définit un champ de données pour l'utilisateur
label	Définit une légende pour un élément input
textarea	Définit un champ de texte long
select	Définit une liste de choix
optgroup	Définit un groupe d'options dans une liste
option	Définit une option dans une liste
fieldset	Permet de regrouper les éléments d'un formulaire en différentes parties
legend	Ajoute une légende à un élément fieldset

L'élément input et les valeurs de l'attribut type

Commençons la présentation des éléments de formulaire avec un élément que nous connaissons : l'élément input.

Cet élément est l'élément à connaitre dans le contexte de la création de formulaires HTML puisqu'il va nous permettre de créer tous types de champs pour récolter des données utilisateurs.

Le type de champ va être défini par la valeur que l'on va donner à son attribut type. L'attribut type peut prendre de nombreuses valeurs. Voici les valeurs les plus utiles et les plus courantes :

Type d'input	**Définition**
text	Définit un champ de saisie monoligne qui accepte du texte
number	Définit un champ qui accepte un nombre décimal
email	Crée un champ qui permet de renseigner une adresse mail
date	Permet à l'utilisateur d'envoyer une date
password	Créer un champ de saisie monoligne dont la valeur va être cachée
checkbox	Permet de créer une case à cocher. L'utilisateur peut cocher une ou plusieurs cases d'un coup
radio	Permet de créer un bouton radio. Par définition, un seul bouton radio peut être coché dans un ensemble
url	Crée un champ qui accepte une URL
file	Permet à un utilisateur de télécharger un fichier
submit	Crée un bouton d'envoi des données du formulaire

L'élément textarea

Pour créer un champ de saisie classique dans nos formulaires, le premier réflexe va être d'utiliser un input type="text" et cela va marcher pour toutes les données de type « texte court ».

Le souci ici est qu'on ne va plus pouvoir utiliser d'input type="text" si on veut par exemple laisser la possibilité à un utilisateur de commenter quelque chose ou si on lui demande de se décrire car le texte qu'on va pouvoir placer dans un input type="text" est limité en taille.

Dans le cas où on souhaite laisser la possibilité à un utilisateur d'écrire un texte long, on utilisera plutôt un élément textarea qui définit un champ de texte long.

Les éléments select, option et optgroup

L'élément select va nous permettre de créer une liste d'options déroulante. Dans cette liste, nous allons placer autant d'éléments option que l'on souhaite donner de choix aux utilisateurs.

Les listes d'options forcent l'utilisateur à faire un choix unique dans la liste et sont généralement utilisées lorsqu'on souhaite proposer de nombreux choix, comme par exemple dans le cas où on demande à l'utilisateur de choisir son pays de résidence.

L'élément optgroup va nous permettre d'ordonner notre liste d'options et groupant des options. Par exemple, dans une liste de choix de pays, on pourrait grouper les différents pays par continent.

Les éléments fieldset et legend

Les éléments fieldset et legend vont nous permettre de structurer et d'améliorer la sémantique d'un formulaire.

L'élément fieldset va nous permettre de grouper plusieurs champs dans un formulaire pour l'organiser en parties.

L'élément legend va nous permettre d'ajouter une légende à une partie de notre formulaire déterminée par fieldset pour expliquer son but par exemple.

Création d'un formulaire HTML complet

Continuons à pratiquer et utilisons certaines des valeurs vues précédemment pour créer un formulaire HTML plus complet que le premier formulaire que nous avons créé.

Nous allons cette fois-ci demander à l'utilisateur de nous transmettre les informations suivantes via un formulaire qui va être divisé en 3 sections :

Section 1 :

- Son nom de famille ;
- Son prénom ;
- Son adresse mail ;
- Son âge ;
- Son sexe ;
- Son pays de résidence.

Section 2 :

- Ses compétences parmi une liste de choix ;
- Une description d'une expérience professionnelle passée / d'un problème résolu.

Section 3 :

- Un mot de passe – obligatoire.

Commençons donc par définir les sections de notre formulaire avec des éléments fieldset et donnons-leur une légende avec l'élément legend.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <form method="post" action="form.php">
            <fieldset>
                <legend>Informations personnelles :</legend>
            </fieldset>
            <fieldset>
                <legend>Compétences / expérience :</legend>
            </fieldset>
            <fieldset>
                <legend>Validation :</legend>
            </fieldset>
        </form>
    </body>
</html>
```

Ensuite, nous allons créer les différents champs section par section et nous nous occuperons de la mise en forme à la toute fin. Notre première section de formulaire contient 6 champs :

- Un premier champ demandant le nom de famille. On utilisera donc un input type="text" ;
- Un deuxième champ demandant le prénom, obligatoire également. On utilisera à nouveau un input type="text"
- Un troisième champ demandant une adresse mail. On utilisera ici un input type="email" ;
- Un quatrième champ demandant un âge. On utilisera un input type="number" ;
- Un cinquième champ qui va demander le sexe de l'utilisateur. Ici, on va proposer un choix parmi deux boutons radio « homme » et « femme ». On utilisera ici des boutons radio plutôt qu'un input type="text" pour recevoir des données bien formatés (choix 1 ou choix 2) plutôt que des résultats comme « homme », « masculin », « garçon », « mec », « fille », « femme », etc. qui seraient très difficile à classer par la suite ;

- Un sixième champ qui va permettre à l'utilisateur d'indiquer son pays de résidence. Ici, nous allons proposer une liste d'options avec une liste préconstruite de pays parmi lesquels l'utilisateur devra choisir. Nous allons également classer les pays par continent.

```
<fieldset>
  <legend>Informations personnelles :</legend>
  <div class="champ">
    <label for="nom">Nom de famille :</label>
    <input type="text" id="nom" name="nom">
  </div>
  <div class="champ">
    <label for="prenom">Prénom :</label>
    <input type="text" id="prenom" name="prenom">
  </div>
  <div class="champ">
    <label for="mail">Adresse mail :</label>
    <input type="email" id="mail" name="mail">
   </div>
   <div class="champ">
     <label for="age">Age :</label>
     <input type="number" id="age" name="age">
   </div>
  <div class="champ">
    <input type="radio" id="h" name="sexe" value="homme">
    <label for="h">Homme</label>
    <input type="radio" id="f" name="sexe" value="femme">
    <label for="f">Femme</label>
   </div>
  <div class="champ">
    <label for="pays">Pays de résidence :</label>
    <select id="pays" name="pays">
      <optgroup label="Europe">
        <option value="france">France</option>
        <option value="belgique">Belgique</option>
        <option value="suisse">Suisse</option>
       </optgroup>
      <optgroup label="Afrique">
        <option value="algerie">Algérie</option>
        <option value="tunisie">Tunisie</option>
        <option value="maroc">Maroc</option>
        <option value="madagascar">Madagascar</option>
        <option value="benin">Bénin</option>
        <option value="togo">Togo</option>
       </optgroup>
      <optgroup label="Amerique">
        <option value="canada">Canada</option>
      </optgroup>
    </select>
  </div>
</fieldset>
```

Regardons de plus près le code ci-dessus pour noter les choses intéressantes. Tout d'abord, vous pouvez remarquer que nos deux boutons radio possèdent des attributs name qui partagent la même valeur.

C'est tout à fait normal ici puisque l'attribut name nous sert dans le cas des boutons radio à grouper les options d'un groupe d'options ensemble (cela nous permet de distinguer d'une autre liste de boutons radio qu'il pourrait y avoir dans le formulaire). C'est également ce qui fait qu'on a besoin d'un attribut value supplémentaire pour ensuite pouvoir savoir quelle option a été cochée par l'utilisateur lors du traitement des données.

Ici, on place également le label après l'élément input pour des raisons d'esthétisme tout simplement.

Ensuite, vous pouvez également noter comment se construit une liste d'options de type select. Dans une liste d'options, chaque option est placée dans un élément option.

On utilise à nouveau un attribut value qui va nous permettre de savoir quelle option a été choisie par l'utilisateur lors du traitement des données.

On peut également classer les options par groupe pour structurer notre formulaire grâce à des éléments optgroup.

Les éléments input type="radio" et select servent le même but. De manière générale, pour des raisons d'ergonomie, nous utiliserons les boutons radio pour une liste de choix courte et des listes d'options pour les listes de choix longues.

Dans notre deuxième section de formulaire, nous demandons aux utilisateurs de cocher leurs compétences dans une liste sans limite de nombre. Nous allons donc utiliser des input type="checkbox" pour créer cette liste.

Nous voulons également que l'utilisateur nous raconte une de ses expériences professionnelles. Pour cela, nous allons utiliser un élément textarea.

```
<fieldset>
  <legend>Compétences / expérience :</legend>
  <div class="champ">
    <input type="checkbox" id="html" name="competences" value="html">
    <label for="html">HTML</label>
    <input type="checkbox" id="css" name="competences" value="css">
    <label for="css">CSS</label>
    <input type="checkbox" id="js" name="competences" value="javascript">
    <label for="js">JavaScript</label>
    <input type="checkbox" id="php" name="competences" value="php">
    <label for="php">PHP</label>
    <input type="checkbox" id="sql" name="competences" value="sql">
    <label for="sql">SQL</label>
    <input type="checkbox" id="seo" name="competences" value="seo">
    <label for="seo">SEO</label>
  </div>
  <div class="champ">
    <textarea name="exp" placeholder="Décrivez une expérience pro"></textarea>
   </div>
</fieldset>
```

Les input type="checkbox" vont posséder une syntaxe relativement similaire aux input type="radio" : les différentes cases à cocher d'un même groupe vont partager un même attribut name et nous allons pouvoir distinguer quelles cases ont bien été cochées grâce à l'attribut value.

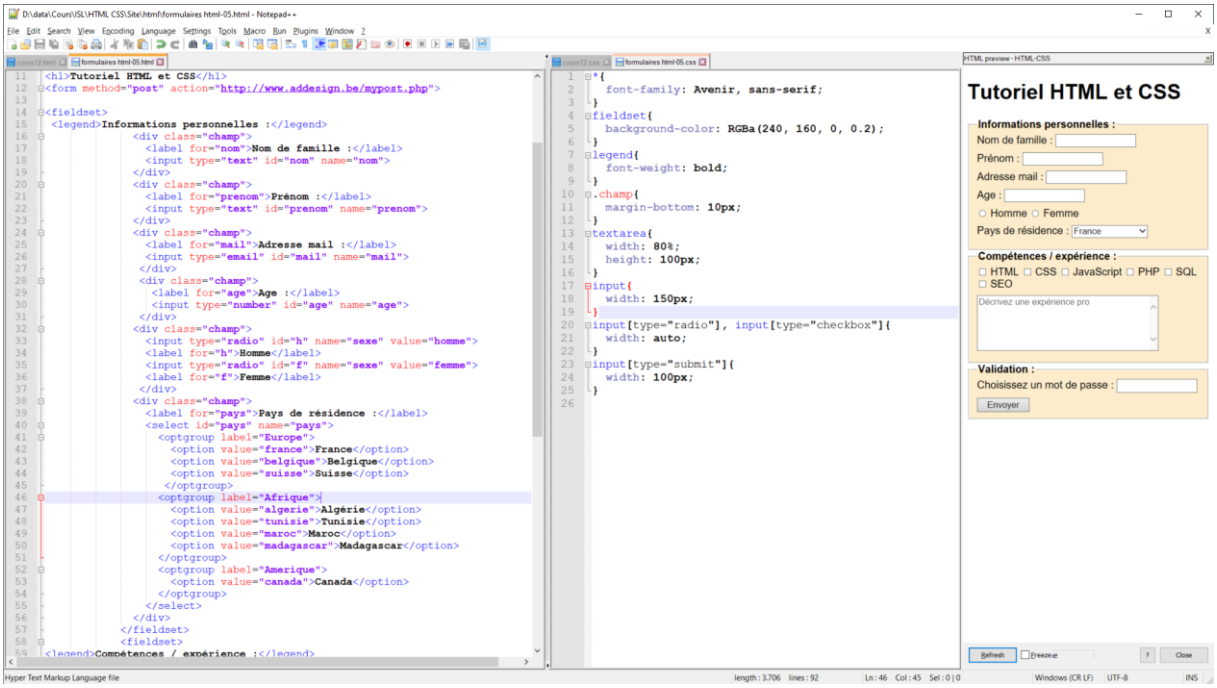
La grande différence entre les input type="checkbox" (cases à cocher) et les input type="radio" (boutons radios) est qu'avec les cases à cocher un utilisateur va pouvoir cocher plusieurs options.

Concernant l'élément textarea, on utilise ici un attribut placeholder qui nous sert à indiquer aux utilisateurs ce qu'ils doivent remplir dans le champ en soi. Le placeholder prend la forme d'un texte semi transparent qui va s'effacer dès qu'un utilisateur place le curseur de sa souris dans le champ de formulaire lié.

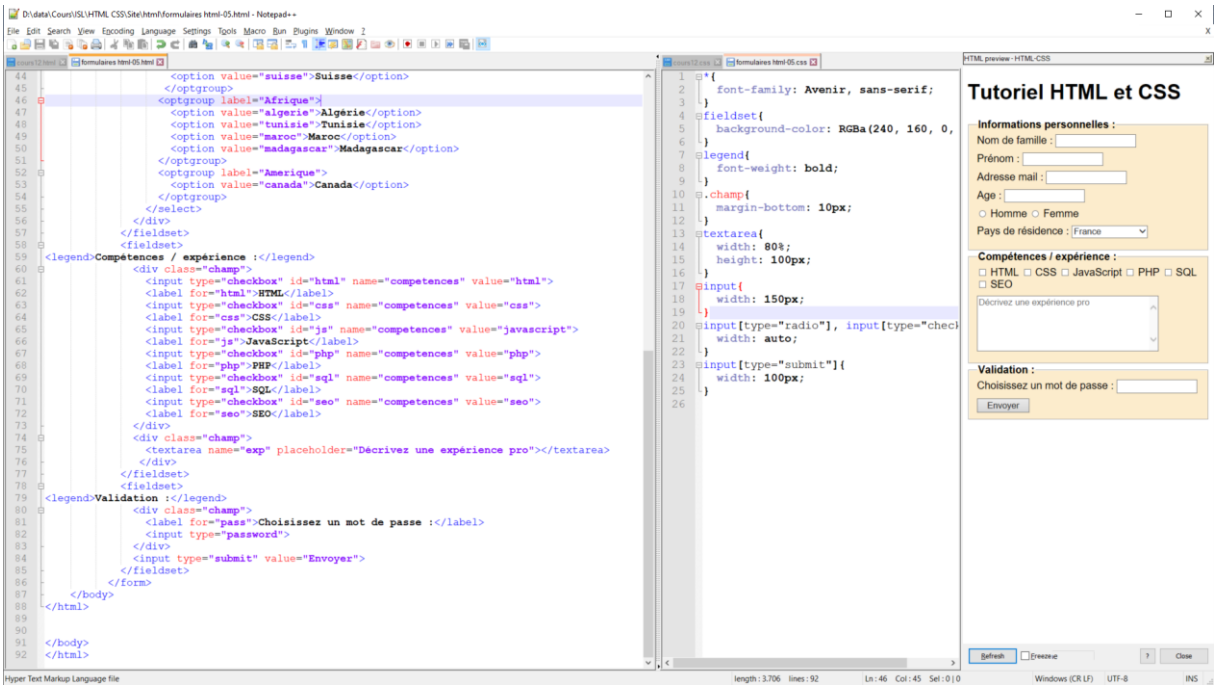
Finalement, notre dernière section de formulaire va tout simplement demander un mot de passe à l'utilisateur et contenir le bouton d'envoi du formulaire.

```
  <fieldset>
    <legend>Validation :</legend>
    <div class="champ">
      <label for="pass">Choisissez un mot de passe :</label>
      <input type="password">
    </div>
    <input type="submit" value="Envoyer">
  </fieldset>
```

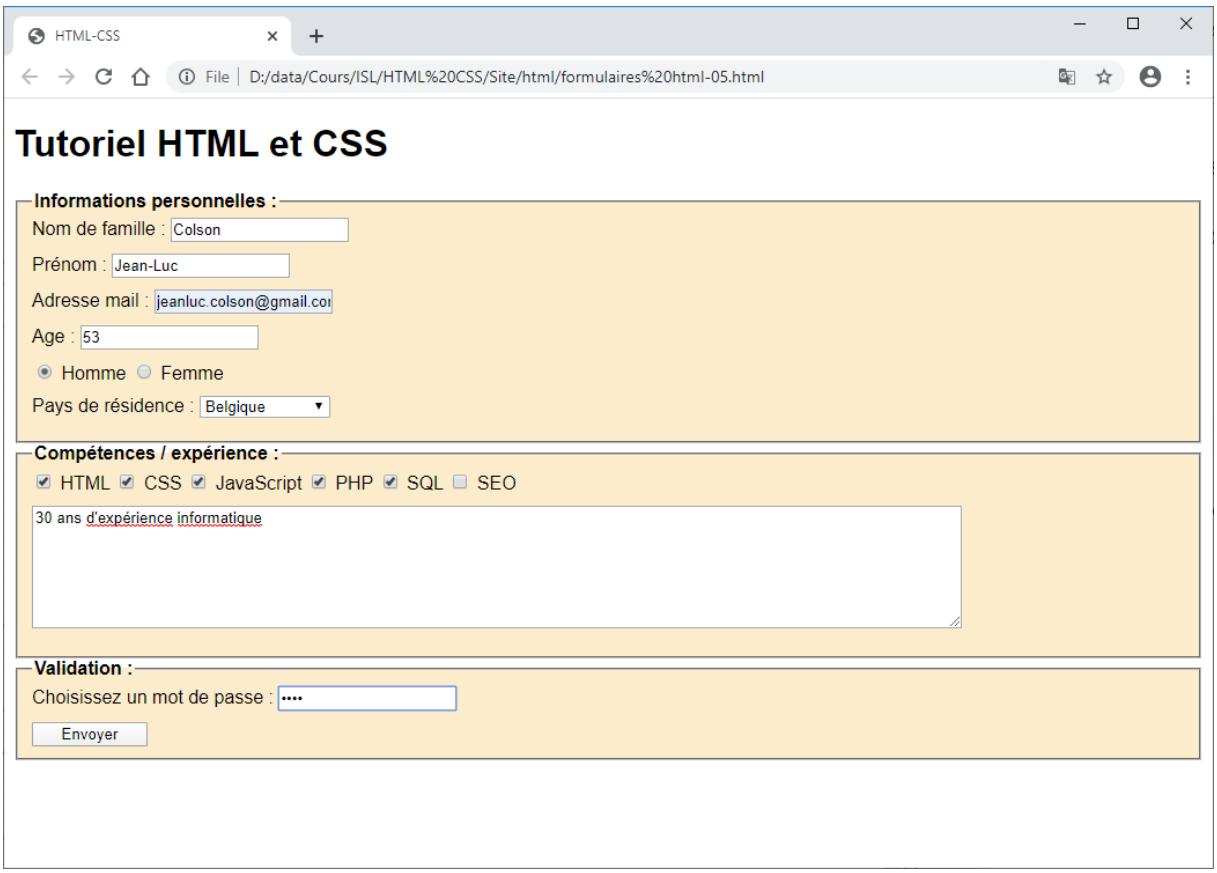
Voici le code complet de ce formulaire. Je me suis contenté d'y ajouter quelques styles CSS simples que vous ne devriez avoir aucun mal à comprendre pour le rendre un peu plus présentable :

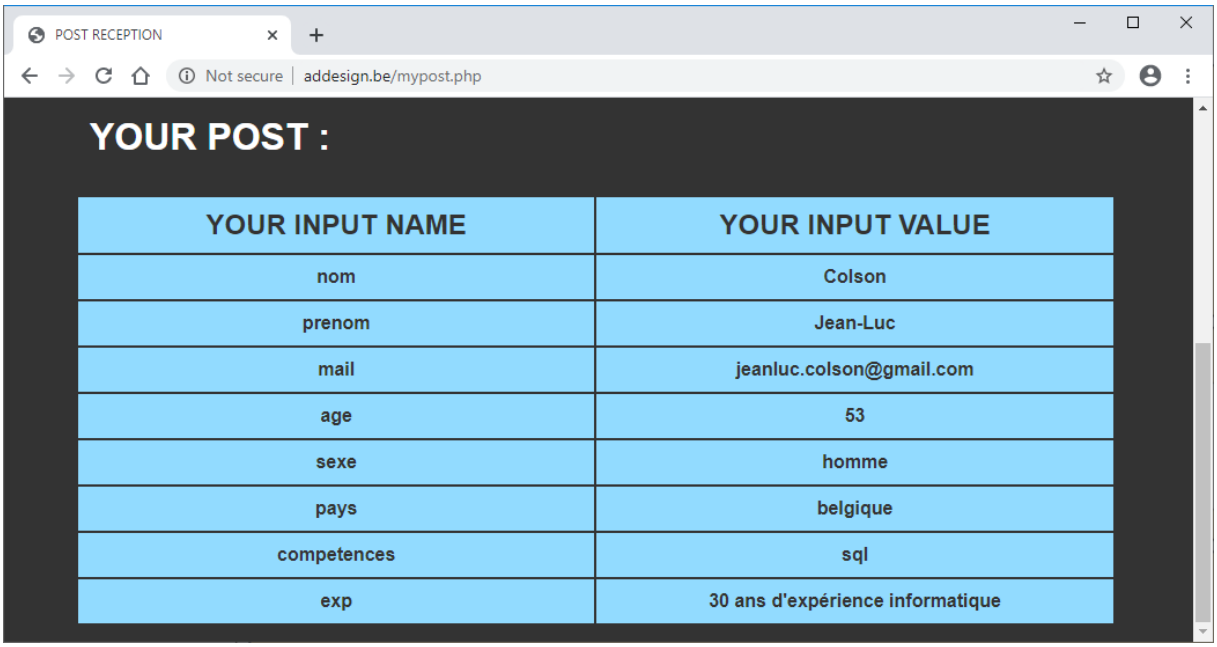


Suite :



Résultat :


HTML-CSS
File | D:/data/Cours/ISL/HTML%20CSS/Site/html/formulaires%20html-05.html
Tutoriel HTML et CSS
Informations personnelles :
Nom de famille : Colson
Prénom : Jean-Luc
Adresse mail : jeanluc.colson@gmail.co
Age : 53
Homme Femme
Pays de résidence : Belgique
Compétences / expérience :
HTML CSS JavaScript PHP SQL SEO
30 ans d'expérience informatique
Validation :
Choisissez un mot de passe :
Envoyer


POST RECEPTION
Not secure | addesign.be/mypost.php
YOUR POST :
YOUR INPUT NAME
YOUR INPUT VALUE
nom
Colson
prenom
Jean-Luc
mail
jeanluc.colson@gmail.com
age
53
sexe
homme
pays
belgique
competences
sql
exp
30 ans d'expérience informatique

Attributs des formulaires et sécurité

Dans cette dernière leçon liée aux formulaires HTML, nous allons discuter de l'importance de sécuriser ses formulaires et de toujours traiter les données envoyées par les utilisateurs et je vais vous présenter quelques attributs HTML qu'il peut s'avérer intéressant d'utiliser.

Le problème des données utilisateurs

Il n'y a qu'une règle que vous devez absolument retenir concernant les formulaires HTML : il ne faut jamais faire confiance aux données envoyées par les utilisateurs.

En effet, vous ne devez jamais vous attendre à ce que vos formulaires soient correctement remplis : certains utilisateurs vont faire des fautes d'inattention ou vont avoir la flemme de remplir tous les champs de votre formulaire tandis que d'autres, malveillants, vont essayer d'exploiter les failles de votre formulaire pour récupérer des données dans votre base de données ou pour tromper vos utilisateurs.

Les mécanismes d'intrusion et de protection des formulaires sont relativement complexes pour quelqu'un qui n'a pas une vue d'ensemble sur les langages utilisés mais je vais essayer d'illustrer le problème et les solutions possibles le plus simplement possible.

Reprenons le formulaire créé à la partie précédente par exemple. Rien n'empêche un utilisateur, dans ce formulaire, d'envoyer des valeurs aberrantes comme 2000 ans pour son âge ou des données de type invalide comme des éléments HTML à la place de son nom ou encore de renvoyer le formulaire à moitié ou complètement vide.

Dans ces cas-là, les données reçues vont être inexploitables et ce n'est pas ce que nous voulons.

De plus, ici, rien n'empêcherait un utilisateur malveillant d'injecter du code JavaScript dans notre formulaire pour ensuite essayer de tromper des utilisateurs ou encore pour tenter d'injecter le code malveillant dans notre base de données.

Nous n'allons pas entrer dans les détails ici car c'est assez complexe et car vous ne pourrez pas comprendre sans connaitre le JavaScript mais grosso modo retenez qu'ici un utilisateur peut tout à fait envoyer du code dans vos champs de formulaire et que le code envoyé risque d'être exécuté à un moment ou à un autre de notre côté.

Quelles solutions pour sécuriser nos formulaires ?

Ici, il va falloir faire la différence entre deux types d'utilisateurs qui vont être gérés de façons différentes : les utilisateurs maladroits qui vont envoyer des données invalides par mégarde et les utilisateurs malveillants qui vont tenter d'exploiter des failles de sécurité dans nos formulaires pour par exemple récupérer les données personnelles d'autres utilisateurs.
Pour ce premier groupe d'utilisateurs qui ne sont pas mal intentionnés, la première action que nous allons pouvoir prendre va être d'ajouter des contraintes directement dans notre formulaire pour limiter les données qui vont pouvoir être envoyées.

Pour cela, nous allons pouvoir utiliser des attributs HTML comme required, min et max que nous allons étudier dans la suite de la leçon et allons devoir faire le maximum pour préciser les bonnes valeurs dans l'attribut type de l'élément input à chaque fois.

Nous allons ensuite également pouvoir tester que les données nous conviennent dès le remplissage d'un champ ou au moment de l'envoi du formulaire grâce au HTML ou au JavaScript (principalement) et bloquer l'envoi du formulaire si des données ne correspondent pas à ce qu'on attend.

Tout cela ne va malheureusement pas être suffisant contre les utilisateurs malintentionnés pour la simple et bonne raison que n'importe qui peut neutraliser toutes les formes de vérification effectuées dans le navigateur.

Pour cela, il suffit par exemple de désactiver l'usage du JavaScript dans le navigateur et d'inspecter le formulaire pour supprimer les attributs limitatifs avant l'envoi.

Contre les utilisateurs malveillants, nous allons donc également devoir vérifier les données après l'envoi du formulaire et neutraliser les données potentiellement dangereuses. Nous allons effectuer ces vérifications en PHP, côté serveur.

Ces deux niveaux de vérifications (dans le navigateur / côté serveur) doivent être implémentés lors de la création de formulaires. En effet, n'utiliser qu'une validation dans le navigateur laisse de sérieuses failles de sécurité dans notre formulaire puisque les utilisateurs malveillants peuvent désactiver ces vérifications.

N'effectuer qu'une série de vérifications côté serveur, d'autre part, serait également une très mauvaise idée d'un point de vue expérience utilisateur puisque ces vérifications sont effectuées une fois le formulaire envoyé.

Ainsi, que faire si des données aberrantes mais pas dangereuses ont été envoyées par un utilisateur maladroit ? Supprimer les données ? Le recontacter pour qu'il soumette à nouveau le formulaire ? Il est bien plus facile dans ce cas de vérifier directement les données lorsqu'elles sont saisies dans le navigateur et de lui faire savoir si une donnée ne nous convient pas.

Vérification et validation des données de formulaire dans le navigateur

Les processus de validation des données que nous allons pouvoir mettre en place dans le navigateur vont s'effectuer avant ou au moment de la tentative d'envoi du formulaire.

L'objectif va être ici de bloquer l'envoi du formulaire si certains champs ne sont pas correctement remplis et de demander aux utilisateurs de remplir correctement les champs invalides.

Une nouvelle fois, cette série de validations va s'adresser aux utilisateurs étourdis / flemmards mais ne va pas bloquer les utilisateurs malveillants car n'importe quel utilisateur peut supprimer à la main des parties de code HTML en inspectant le code de la page et désactiver le JavaScript dans son navigateur.

La première chose à faire ici va déjà être de s'assurer qu'on a bien précisé les bons types d'input pour chaque champ de formulaire car la plupart des navigateurs récents proposent des systèmes de contrôle.

Par exemple, les navigateurs bloquent aujourd'hui l'envoi de code JavaScript et empêchent l'envoi du formulaire selon que certaines données ne soient pas de la forme attendue.

Essayez par exemple de rentrer un prénom dans le champ de formulaire demandant un email : votre navigateur devrait renvoyer une erreur et vous demander d'ajouter un signe @ dans vos données.

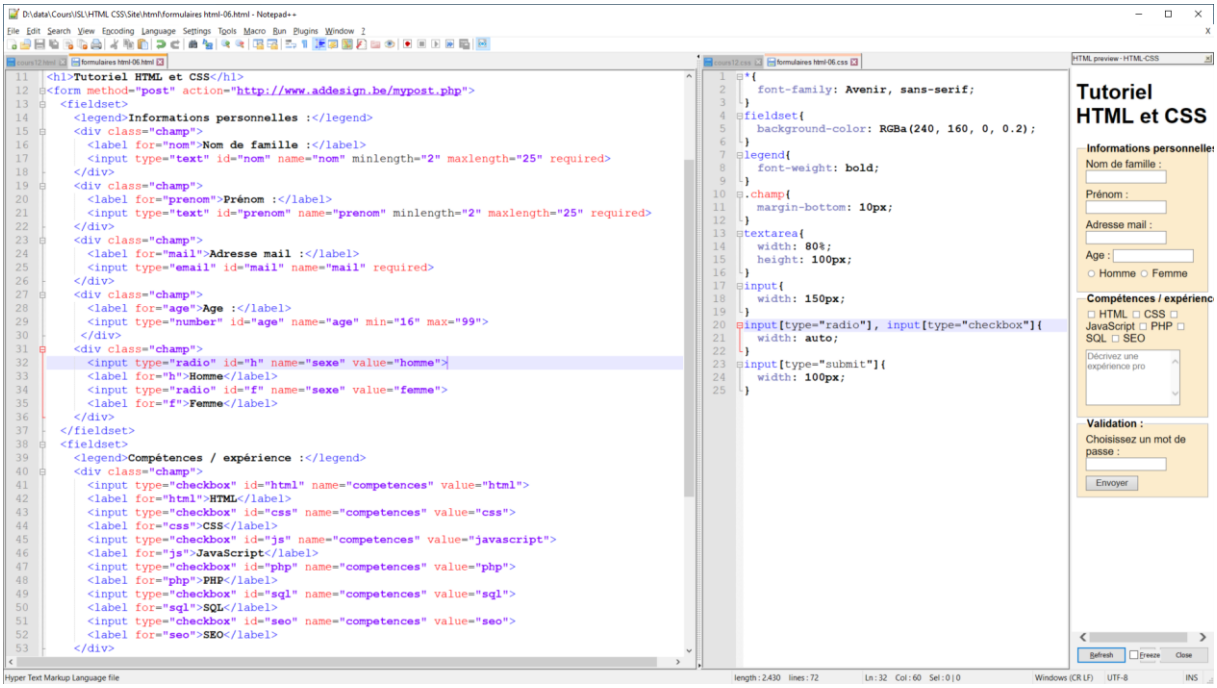
En plus de ces vérifications faites par le navigateur, le HTML propose aujourd'hui des options de validation relativement puissantes. La validation des données en HTML va principalement passer par l'ajout d'attributs dans les éléments de formulaire. Nous allons ainsi pouvoir utiliser les attributs suivants :

Attribut	**Définition**
size	Permet de spécifier le nombre de caractères dans un champ
minlength	Permet de spécifier le nombre minimum de caractères dans un champ
maxlength	Permet de spécifier le nombre maximum de caractères dans un champ
min	Permet de spécifier une valeur minimale pour un champ de type number ou date
max	Permet de spécifier une valeur maximale pour un champ de type number ou date
step	Permet de définir un multiple de validité pour un champ acceptant des donnés de type nombre ou date. En indiquant step="4" , les nombres valides seront -8, -4, 0, 4, 8, etc.
autocomplete	Permet d'activer l'auto complétion pour un champ : si un utilisateur a déjà rempli un formulaire, des valeurs lui seront proposées automatiquement lorsqu'il va commencer à remplir le champ
required	Permet de forcer le remplissage d'un champ. Le formulaire ne pourra pas être envoyé si le champ est vide
pattern	Permet de préciser une expression régulière. La valeur du champ devra respecter la contrainte de la regex pour être valide

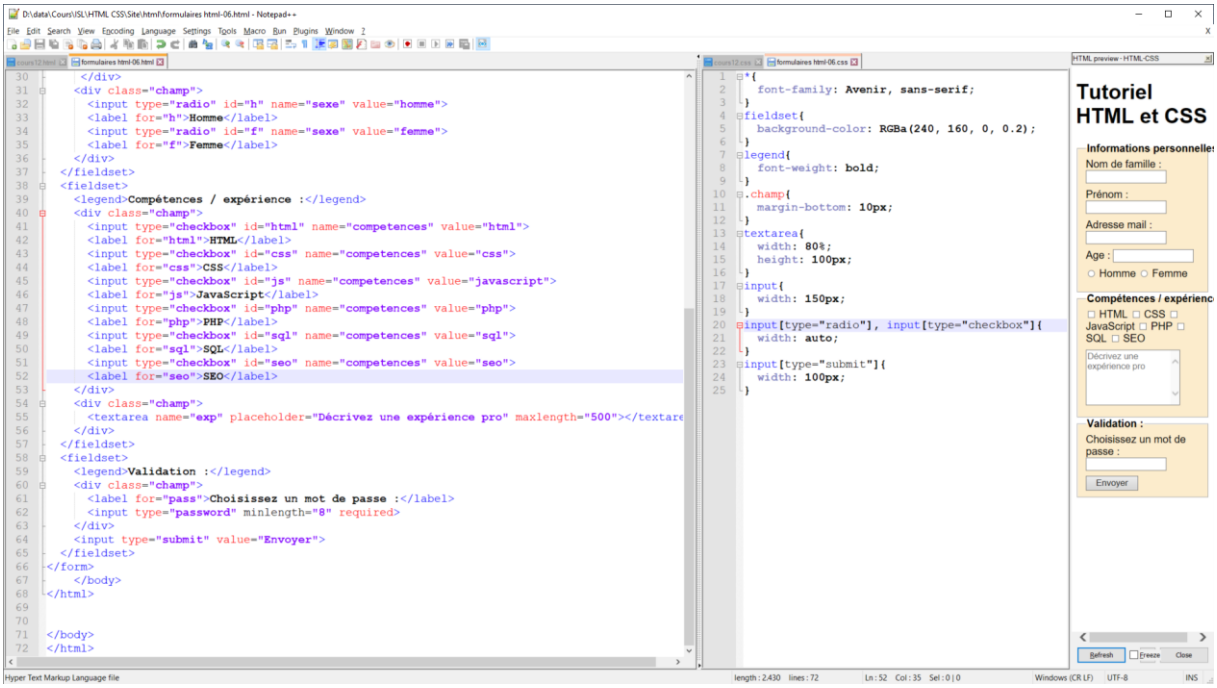
Note : les expressions régulières constituent un langage à part. Elles vont consister en la création de « motifs » ou de chaine de caractères constituées de caractères à la signification spéciale et qui vont servir à décrire un ensemble de chaines de caractères possibles. Les expressions régulières sont utilisées conjointement avec de nombreux langages informatiques pour comparer des données envoyées avec le motif attendu et les refuser si elles ne correspondent pas. Bien évidemment, l'objet de cette leçon n'est pas de faire un cours sur les expressions régulières.

Reprenons notre formulaire précédent en se concentrant sur certains champs en particulier et ajoutons quelques contraintes sur les données que l'on souhaite recevoir :

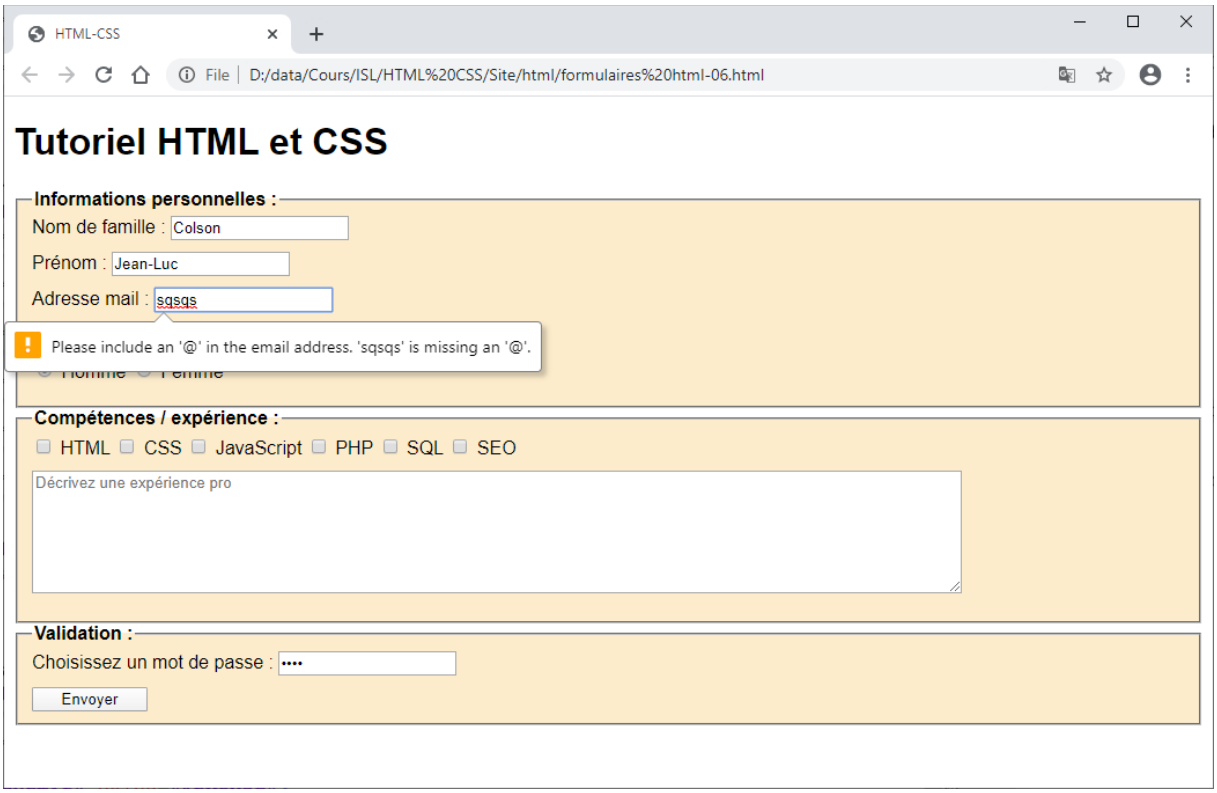
- Les nom, prénom, adresse mail et mot de passe sont désormais obligatoires
- Les nom et prénom doivent faire au moins 2 caractères et maximum 25 caractères ;
- L'âge doit être compris entre 16 et 99 ans ;
- La description de l'expérience professionnelle ne doit pas excéder 500 caractères ;
- Le mot de passe doit contenir au moins 8 caractères.



Suite:


Tutoriel
HTML et CSS
Informations personnelles
Nom de famille :
Prénom :
Adresse mail :
Age :
Homme Femme
Compétences / expérienc
HTML CSS
JavaScript PHP
SQL SEO
Validation :
Choisissez un mot de passe :
Envoyer

Résultat:


HTML-CSS
File D:/data/Cours/ISL/HTML%20CSS/Site/html/formulaires%20html-06.html
Tutoriel HTML et CSS
Informations personnelles :
Nom de famille : Colson
Prénom : Jean-Luc
Adresse mail : sqsqs
Please include an '@' in the email address. 'sqsqs' is missing an '@'.
Compétences / expérience :
HTML CSS JavaScript PHP SQL SEO
Décrivez une expérience pro
Validation :
Choisissez un mot de passe :
Envoyer

Essayez d'envoyer le formulaire ci-dessus en remplissant mal les différents champs : le navigateur va vous indiquer les différentes erreurs et va bloquer l'envoi du formulaire tant que vous ne les aurez pas corrigées.

Vous pouvez noter que l'attribut required possède une valeur muette : il n'est pas nécessaire de la fournir (car elle est unique) et il suffit donc de préciser le nom de l'attribut pour rendre le remplissage du champ obligatoire.