The preceding examples use some of the most common elements used in (X)HTML documents, including:

• The <!DOCTYPE> statement, which indicates the particular version of HTML or XHTML being used in the document. The first example uses the strict 4.01 specification, the second uses a reduced form for HTML5 the meaning of which will be explained a bit later on, and the final example uses the XHTML 1.0 strict specification.

• The <html>, <head>, and <body> tag pairs are used to specify the general structure of the document. The required inclusion of the xmlns attribute in the <html> tag is a small difference required by XHTML.

• The <meta> tag used in the examples indicates the MIME type of the document and the character set in use. Notice that in the XHTML example, the element has a trailing slash to indicate that it is an empty element.

• The <title> and </title> tag pair specifies the title of the document, which generally appears in the title bar of the Web browser.

• A comment is specified by <!-- -->, allowing page authors to provide notes for future reference.

The <h1> and </h1> header tag pair indicates a headline specifying some important information.

• The <hr> tag, which has a self-identifying end tag (<hr />) under XHTML, inserts a horizontal rule, or bar, across the screen.

• The <p> and </p> paragraph tag pair indicates a paragraph of text.

• A special character is inserted using a named entity (&hearts;), which in this case inserts a heart dingbat character into the text.

• The <em> and </em> tag pair surrounds a small piece of text to emphasize which a browser typically renders in italics.

There are numerous other markup elements that may be employed, all of which will be explored throughout the book, but for now this sampling is enough to get our first example rendered in a browser.

NOTE: Examples in the book will generally be presented in HTML5. Syntax specific to particular browsers, older HTML variants, or XHTML will always be noted when used.

[Intermediate HTML and CSS](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css)

**INTRODUCTION**

**Introduction :**

Dans ce cours, nous allons approfondir à la fois HTML et CSS. Nos leçons de base, par conception, n'ont fait qu'effleurer la surface de ce qui est possible. Nous voulions vous donner tout le strict nécessaire pour commencer afin que vous puissiez passer à autre chose et être quelque peu productif le plus rapidement possible, mais il est maintenant temps de ralentir et de creuser !

Vous avez probablement déjà réalisé qu'il y a beaucoup plus d'éléments HTML que nous n'en avons mentionnés dans notre contenu de base. Tout au long de ce cours, nous aborderons le reste des éléments importants tels que les formulaires, les tableaux et les éléments sémantiques tels que les en-têtes, les pieds de page, les sections et les apartés.

Il y a aussi beaucoup plus que vous pouvez faire avec CSS que vous apprendrez tout au long de ce cours, comme les variables et les fonctions, les ombres, l'animation et bien sûr les dispositions de grille ! Alors accrochez-vous ! À la fin de ce cours, vous serez en mesure de recréer à peu près n'importe quelle conception Web que vous pouvez trouver sur Internet… ce qui est une compétence importante à poursuivre. Même si vous ne visez pas pour un travail spécifique au front-end, il est important de pouvoir rendre les éléments de votre portefeuille beaux pour vous démarquer.

**Mission**

Lisez cette référence d'éléments HTML pour avoir un aperçu des autres éléments HTML disponibles. Inutile de mémoriser cela, car vous apprendrez les parties importantes au fur et à mesure, mais un coup d'œil maintenant aidera le contenu à rester plus tard.

Jetez un coup d'œil à une feuille de triche CSS quelque peu écrasante. Encore une fois, nous n'avons pas besoin que vous appreniez quoi que ce soit de spécifique ou que vous mémorisiez quoi que ce soit de cette liste, utilisez-la simplement pour avoir une idée de ce qu'il vous reste à apprendre !

**L'élément de formulaire**

L'élément form est un élément conteneur comme l'élément div que nous avons appris plus tôt dans le programme. L'élément de formulaire encapsule toutes les entrées avec lesquelles un utilisateur interagira sur un formulaire.

L'élément form accepte deux attributs essentiels ; le premier est l'attribut action qui prend une valeur d'URL qui indique au formulaire où il doit envoyer ses données à traiter. Plus tard dans le programme, nous apprendrons à raccorder les systèmes backend aux formulaires frontaux à l'aide de cet attribut. Pour l'instant, il est seulement essentiel de savoir à quoi sert l'attribut action.

Le second est l'attribut method qui indique au navigateur quelle méthode de requête HTTP il doit utiliser pour soumettre le formulaire. Les méthodes de requête GET et POST sont les deux que vous utiliserez le plus.

Nous utilisons GET lorsque nous voulons récupérer quelque chose d'un serveur. Par exemple, Google utilise une requête GET lorsque vous effectuez une recherche car il obtient les résultats de la recherche.

POST est utilisé lorsque nous voulons modifier quelque chose sur le serveur, par exemple lorsqu'un utilisateur crée un compte ou effectue un paiement sur un site Web.

Le balisage pour la création d'un élément de formulaire ressemble à ceci :

<form action="example.com/path" method="post">

</form>

[**Form Controls**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-basics#form-controls)

Pour commencer à collecter des données utilisateur, nous devons utiliser des éléments de contrôle de formulaire. Ce sont tous les éléments avec lesquels les utilisateurs interagiront sur le formulaire, tels que les zones de texte, les listes déroulantes, les cases à cocher et les boutons. Dans les quelques sections suivantes, nous explorerons certains des contrôles de formulaire que vous utiliserez le plus souvent.

[**The Input Element**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-basics#the-input-element)

L'élément d'entrée est le plus polyvalent de tous les éléments de contrôle de formulaire. Il accepte un attribut de type qui indique au navigateur quel type de données il doit attendre et comment il doit rendre l'élément d'entrée.

Une entrée de texte ressemble à ceci :

<form action="example.com/path" method="post">

<input type="text">

</form>

Les entrées de texte acceptent n'importe quelle entrée de texte. Par exemple, vous l'utiliseriez pour collecter des éléments tels que les noms et prénoms des utilisateurs.

**Labels**

Une entrée en elle-même n'est pas très utile car l'utilisateur ne saura pas quel type de données il est censé fournir. Au lieu de cela, nous pouvons donner à nos entrées une étiquette pour informer les utilisateurs du type de données qu'ils sont censés saisir.

Pour créer une étiquette, nous utilisons l'élément <label>. Le texte que nous voulons afficher dans l'étiquette ira entre ses balises d'ouverture et de fermeture :

<form action="example.com/path" method="post">

<label for="firstName">First Name:</label>

<input type="text" id="firstName">

</form>

Les labels acceptent un attribut for, qui l'associe à une entrée particulière. L'entrée que nous voulons associer à une étiquette nécessite un attribut id avec la même valeur que l'attribut for de label.

Lorsqu'une étiquette est associée à une entrée et que l'on clique dessus, le curseur se concentre sur cette entrée, prêt pour que l'utilisateur saisisse des données. Cela contribue à rendre nos formulaires plus accessibles aux utilisateurs qui s'appuient sur des technologies d'assistance.

**Placeholder Attribute**

Pour guider les utilisateurs sur ce qu'il faut saisir dans les éléments de formulaire, nous pouvons inclure un texte d'espace réservé dans les champs de saisie.

Cela se fait en ajoutant un placeholder réservé à une entrée. La valeur sera le texte d'espace réservé que nous voulons afficher dans l'entrée :

<label for="first\_name">First Name:</label>

<input type="text" id="first\_name" placeholder="Bob...">

Utilisez un placeholder réservé pour montrer comment le texte doit être saisi et formaté.

**The Name Attribute**

Nous devons utiliser des étiquettes pour que les utilisateurs comprennent ce que représenteront les données saisies dans un champ de saisie. Juste comme ça, nous devons également laisser le backend où nous envoyons nos données savoir ce que représente chaque donnée.

Nous faisons cela en ajoutant un attribut name à nos entrées :

<label for="first\_name">First Name:</label>

<input type="text" id="first\_name" name="first\_name">

L'attribut name sert de référence aux données saisies dans un contrôle de formulaire après l'avoir soumis. Vous pouvez le considérer comme un nom de variable pour l'entrée. L'entrée de formulaire doit toujours avoir un attribut name ; sinon, il sera ignoré lors de la soumission du formulaire.

Pour mieux comprendre à quoi cela ressemble, nous pouvons soumettre un formulaire à httpbin. Ce service renverra une réponse qui nous permettra de voir quelles données ont été soumises. Remplissez le formulaire ci-dessous et cliquez sur soumettre:

<form action="http://httpbin.org/post" method="post">

<div>

<label for="first\_name">First Name:</label>

<input type="text" name="first\_name" id="first\_name">

</div>

<div>

<label for="last\_name">Last Name:</label>

<input type="text" name="last\_name" id="last\_name">

</div>

<div>

<label for="age">Age:</label>

<input type="number" name="age" id="age">

</div>

<button type="submit">Submit</button>

</form>

La sortie qui nous intéresse à partir de la réponse est l'objet "form". Ça devrait ressembler a quelque chose comme ca:"form": {

"age": "33",

"first\_name": "John",

"last\_name": "Doe"

},

Essayez de modifier les attributs name de certains des champs de saisie du formulaire, puis soumettez-les à nouveau pour voir comment les données du formulaire dans la réponse changent.

Par exemple, vous voudrez peut-être avoir une entrée qui obtient des données d'un utilisateur et les afficher ailleurs sur la page avec Javascript :

<div>

<label for="username">What's your name?</label>

<input type="text"id="username">

<button class="greet-btn">Greet Me</button>

</div>

<h1 class="greeting"></h1>

Nous aurons besoin de manipuler un peu les données des contrôles de formulaire comme celui-ci lorsque nous arriverons aux projets du cours Fullstack JavaScript.

**Email Input**

Les entrées d'e-mail sont des entrées de texte spécialisées uniquement pour les adresses e-mail. Ils sont différents des entrées de texte en ce sens qu'ils affichent un clavier différent qui inclura le symbole @ sur les appareils mobiles pour faciliter la saisie des adresses e-mail.

Ils valident également que l'utilisateur a entré une adresse e-mail correctement formatée, mais il y aura plus sur les validations plus tard.

Pour créer une entrée de courrier électronique, nous utilisons un élément d'entrée avec l'attribut de type "email":

<label for="user\_email">Email Address:</label>

<input type="email" id="user\_email" name="email" placeholder="you@example.com">

**Password Input**

Les entrées de mot de passe sont une autre entrée de texte spécialisée. Ils diffèrent des saisies de texte ordinaires en ce qu'ils masquent les données saisies avec des astérisques (\*) pour empêcher quiconque de voir ce qui a été saisi.

Une entrée de mot de passe peut être créée à l'aide d'un élément d'entrée de type « mot de passe » :

<label for="user\_password">Password:</label>

<input type="password" id="user\_password" name="password">

**Number Input**

L'entrée numérique accepte uniquement les valeurs numériques et ignore tout autre caractère que l'utilisateur essaie d'entrer.

Nous créons une entrée de nombre à l'aide de l'élément input avec un attribut type de "nombre":

<label for="amount">Amount:</label>

<input type="number" id="amount" name="amount">

**Date Input**

Pour collecter les dates d'un utilisateur, nous pouvons utiliser une entrée de date. Cette entrée est unique car elle offre une meilleure expérience utilisateur pour le choix des dates en affichant un simple calendrier de sélection de dates.

Pour créer une entrée de date, nous utilisons l'élément input avec un attribut type de "date":

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob">

**Text Area**

L'élément de zone de texte fournit une zone de saisie qui peut accepter du texte qui s'étend sur plusieurs lignes, comme les commentaires et les critiques des utilisateurs. Il peut également être redimensionné en cliquant et en faisant glisser le coin inférieur droit pour l'agrandir ou le réduire.

Pour créer une zone de texte, nous utilisons l'élément <textarea> :

<textarea></textarea>

Contrairement aux éléments d'entrée, les éléments Textarea ont une balise de fermeture. Cela vous permet d'envelopper le contenu initial que vous souhaitez que la zone de texte affiche :

<textarea>Some initial content</textarea>

Les éléments de zone de texte acceptent quelques attributs uniques que les autres contrôles de formulaire n'acceptent pas. Ce sont les attributs rows et cols. Ils vous permettent de contrôler la hauteur initiale (rows) et la largeur (cols) de la zone de texte :

**Selection Elements**

Parfois, vous souhaiterez que les utilisateurs sélectionnent une valeur dans une liste prédéfinie. C'est là que les éléments sélectionnés seront utiles.

**Select Dropdown**

L'élément select affiche une liste déroulante dans laquelle les utilisateurs peuvent sélectionner une option. Syntaxiquement, les éléments select ont un balisage similaire aux listes non ordonnées. L'élément select enveloppe les éléments option qui sont les options pouvant être sélectionnées.

Pour créer une liste déroulante de sélection, nous utilisons l'élément <select>. Toutes les options que nous voulons afficher dans l'élément select sont définies à l'aide des éléments <option> :

<select name="Car">

<option value="mercedes">Mercedes</option>

<option value="tesla">Tesla</option>

<option value="volvo">Volvo</option>

<option value="bmw">BMW</option>

<option value="mini">Mini</option>

<option value="ford">Ford</option>

</select>

Tous les éléments d'option doivent avoir un attribut value. Cette valeur sera envoyée au serveur lors de la soumission du formulaire.

Nous pouvons définir l'une des options comme élément sélectionné par défaut lorsque le navigateur affiche le formulaire pour la première fois en donnant à l'une des options l'attribut selected :

<select name="Car">

<option value="mercedes">Mercedes</option>

<option value="tesla">Tesla</option>

<option value="volvo" selected>Volvo</option>

<option value="bmw">BMW</option>

<option value="mini">Mini</option>

<option value="ford">Ford</option>

</select>

Nous pouvons également diviser la liste des options en groupes à l'aide de l'élément <optgroup>. L'élément optgroup prend un attribut label que le navigateur utilise comme étiquette pour chaque groupe :

<select name="fashion">

<optgroup label="Clothing">

<option value="t\_shirt">T-Shirts</option>

<option value="sweater">Sweaters</option>

<option value="coats">Coats</option>

</optgroup>

<optgroup label="Foot Wear">

<option value="sneakers">Sneakers</option>

<option value="boots">Boots</option>

<option value="sandals">Sandals</option>

</optgroup>

</select>

**Radio Buttons**

Les listes déroulantes de sélection sont idéales pour économiser de l'espace sur la page lorsque nous avons une longue liste d'options parmi lesquelles nous souhaitons que les utilisateurs puissent choisir. Cependant, lorsque nous avons une liste plus petite de 5 options ou moins parmi lesquelles choisir, il est souvent préférable pour l'utilisateur de les afficher sur la page au lieu de les cacher derrière une liste déroulante.

Dans ce cas, nous pouvons utiliser des boutons radio. Les boutons radio nous permettent de créer plusieurs options parmi lesquelles l'utilisateur peut choisir. Pour créer des boutons radio, nous utilisons à nouveau l'élément d'entrée toujours adaptable avec un attribut de type "radio":

h1>Ticket Type</h1>

<div>

<input type="radio" id="child" name="ticket\_type" value="child">

<label for="child">Child</label>

</div>

<div>

<input type="radio" id="adult" name="ticket\_type" value="adult">

<label for="adult">Adult</label>

</div>

<div>

<input type="radio" id="senior" name="ticket\_type" value="senior">

<label for="senior">Senior</label>

</div>

Lorsque nous sélectionnons l'un des boutons radio, puis en sélectionnons un autre, il désélectionne le premier. Les boutons radio savent le faire car ils ont le même attribut name. C'est ainsi que le navigateur sait que ces éléments font partie du même groupe d'options.

Nous pouvons définir le bouton radio sélectionné par défaut en y ajoutant l'attribut checked :

<h1>Ticket Type</h1>

<div>

<input type="radio" id="child" name="ticket\_type" value="child">

<label for="child">Child</label>

</div>

<div>

<input type="radio" id="adult" name="ticket\_type" value="adult" checked>

<label for="adult">Adult</label>

</div>

<div>

<input type="radio" id="senior" name="ticket\_type" value="senior">

<label for="senior">Senior</label>

</div>

**Checkboxes**

Les cases à cocher sont similaires aux boutons radio en ce sens qu'elles permettent aux utilisateurs de choisir parmi un ensemble d'options prédéfinies. Mais contrairement aux boutons radio, ils permettent de sélectionner plusieurs options à la fois.

Pour créer une case à cocher, nous utilisons l'élément input avec un attribut type de "checkbox":

<h1>Pizza Toppings</h1>

<div>

<input type="checkbox" id="sausage" name="topping" value="sausage">

<label for="sausage">Sausage</label>

</div>

<div>

<input type="checkbox" id="onions" name="topping" value="onions">

<label for="onions">Onions</label>

</div>

<div>

<input type="checkbox" id="pepperoni" name="topping" value="pepperoni">

<label for="pepperoni">Pepperoni</label>

</div>

<div>

<input type="checkbox" id="mushrooms" name="topping" value="mushrooms">

<label for="mushrooms">Mushrooms</label>

</div>

Nous pouvons également avoir une seule case à cocher lorsque nous voulons que les utilisateurs basculent s'ils veulent que quelque chose soit vrai ou faux. Comme s'inscrire à une newsletter lors de la création d'un compte par exemple :

<div>

<input type="checkbox" id="newsletter" name="news\_letter">

<label for="newsletter">Send me the news letter</label>

</div>

Nous pouvons définir une case à cocher pour qu'elle soit cochée par défaut au chargement de la page en leur donnant un attribut checked :

<div>

<input type="checkbox" id="newsletter" name="news\_letter" checked>

<label for="newsletter">Send me the news letter</label>

</div>

**Buttons**

L'élément bouton crée des boutons cliquables avec lesquels l'utilisateur peut interagir pour soumettre des formulaires et déclencher d'autres actions.

Pour créer un bouton, nous utilisons l'élément <button>. Le contenu ou le texte que nous voulons afficher à l'intérieur du bouton ira dans les balises d'ouverture et de fermeture :

<button>Click Me</button>

L'élément Le bouton accepte également un attribut de type qui indique au navigateur à quel type de bouton il a affaire. Trois types de boutons sont à notre disposition.

**Submit Buttons**

Une fois qu'un utilisateur a fini de remplir un formulaire, il a besoin d'un moyen de le soumettre. Il existe un bouton spécialisé pour cela; le bouton Soumettre. Lorsqu'un bouton de soumission est cliqué, il soumettra le formulaire dans lequel il se trouve. L'attribut type a la valeur de submit par défaut, c'est-à-dire si le type n'est pas spécifié ou si la valeur fournie n'est pas valide.

Pour créer un bouton d'envoi, nous utilisons l'élément button avec un attribut de type "submit":

**<button type="submit">Submit</button>**

**Reset Button**

Un bouton de réinitialisation efface toutes les données que l'utilisateur a saisies dans le formulaire et remet toutes les entrées du formulaire à ce qu'elles étaient initialement.

Pour créer un bouton de réinitialisation, nous utilisons l'élément button avec un attribut de type "reset":

<button type="reset">Reset</button>

**Generic Button**

Le troisième et dernier type de bouton est simplement un bouton générique qui peut être utilisé pour n'importe quoi. Il est couramment utilisé avec JS pour créer des interfaces utilisateur interactives.

Pour créer un bouton générique, nous utilisons l'élément button avec un attribut type de « button » :

<button type="button">Click to Toggle</button>

**Remarque** : Il est important de se rappeler qu'un bouton dans un formulaire avec la valeur de type submit (qui se trouve être la valeur par défaut) essaiera toujours de faire une nouvelle demande et de renvoyer les données au serveur. Par conséquent, pour les boutons utilisés dans un formulaire à des fins autres que la soumission des données, l'attribut type doit toujours être spécifié pour éviter les effets indésirables de la soumission d'un formulaire.

[**Organizing Form Elements**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-basics#organizing-form-elements)

L'utilisation des entrées correctes pour les données que nous voulons que les utilisateurs saisissent contribue grandement à rendre nos formulaires conviviaux. Cependant, dans les formulaires plus volumineux, les utilisateurs peuvent facilement être submergés et découragés s'il y a de nombreuses entrées à remplir.

Heureusement, HTML fournit quelques éléments qui nous aident à organiser les formulaires en sections visuellement distinctes et gérables pour digérer les sections en un clin d'œil.

**Fieldset Element**

L'élément fieldset est un élément conteneur qui nous permet de regrouper les entrées de formulaire associées dans une unité logique.

Pour créer un fieldset, nous utilisons l'élément <fieldset>. Quelles que soient les entrées de formulaire que nous voulons regrouper, allez dans les balises fieldset d'ouverture et de fermeture :

<fieldset>

<label for="first\_name">First Name</label>

<input type="text" id="first\_name" name="first\_name">

<label for="last\_name">Last Name</label>

<input type="text" id="last\_name" name="last\_name">

</fieldset>

**Legend**

L'élément de légende est utilisé pour donner aux ensembles de champs un en-tête ou une légende afin que l'utilisateur puisse voir à quoi sert un groupement d'entrées.

Pour créer une légende, nous utilisons l'élément <legend> avec le texte que nous voulons afficher dans ses balises d'ouverture et de fermeture. Une légende doit toujours venir juste après une balise d'ouverture de champs :

<fieldset>

<legend>Contact Details</legend>

<label for="name">Name:</label>

<input type="text" id="name" name="name">

<label for="phone\_number">Phone Number:</label>

<input type="tel" id="phone\_number" name="phone\_number">

<label for="email">Email:</label>

<input type="email" id="email" name="email">

</fieldset>

<fieldset>

<legend>Delivery Details</legend>

<label for="street\_address">Street Address:</label>

<input type="text" id="street\_address" name="street\_address">

<label for="city">City:</label>

<input type="text" id="city" name="city">

<label for="zip\_code">Zip Code:</label>

<input type="text" id="zip\_code" name="zip\_code">

</fieldset>

Un cas d'utilisation courant pour ces éléments consiste à utiliser un ensemble de champs pour regrouper les boutons radio et à utiliser une légende pour communiquer à l'utilisateur à quoi sert chacune des options :

<fieldset>

<legend>What would you like to drink?</legend>

<div>

<input type="radio" name="drink" id="coffee" value="coffee">

<label for="coffee">Coffee</label>

</div>

<div>

<input type="radio" name="drink" id="tea" value="tea">

<label for="tea">Tea</label>

</div>

<div>

<input type="radio" name="drink" id="soda" value="soda">

<label for="soda">Soda</label>

</div>

</fieldset>

[**A Note on Styling Forms**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-basics#a-note-on-styling-forms)

Nous fournirons des ressources qui approfondissent les formulaires de style dans la section d'affectation qui vient ensuite. Cependant, avant d'en arriver à la tâche, nous devrions parler de certains des défis liés au style des formulaires HTML et comment nous pouvons les contourner :

**Styles de navigateur par défaut :** Chaque navigateur a ses propres styles par défaut pour les contrôles de formulaire, ce qui rend vos formulaires visuellement différents pour les utilisateurs en fonction du navigateur qu'ils utilisent. Pour avoir une conception cohérente entre tous les navigateurs, nous devons remplacer ces styles par défaut et les styliser nous-mêmes.

Délicat et carrément impossible de styliser les contrôles de formulaire

Les contrôles de formulaire textuels tels que le texte, l'e-mail, le mot de passe et les zones de texte sont relativement simples à styliser. Ils fonctionnent comme n'importe quel autre élément HTML et la plupart des propriétés CSS peuvent être utilisées dessus. Les choses deviennent plus délicates lors de la création de styles personnalisés pour les boutons radio et les cases à cocher. Mais il existe de nombreux guides que vous pouvez utiliser pour réaliser le design souhaité. De nouvelles propriétés CSS ont également été mises à disposition ces derniers temps pour faciliter le style des boutons radio et des cases à cocher.

Certains aspects d'autres éléments sont carrément impossibles à styliser, par exemple, le calendrier ou les sélecteurs de date. Si nous voulons des styles personnalisés pour ceux-ci, nous devrons créer des contrôles de formulaire personnalisés avec JavaScript ou utiliser l'une des nombreuses bibliothèques JavaScript qui nous fournissent des solutions prêtes à l'emploi.

## FORM VALIDATION

Les validations nous permettent de définir des contraintes ou des règles spécifiques qui déterminent les données que les utilisateurs peuvent entrer dans une entrée. Lorsqu'un utilisateur saisit des données qui enfreignent les règles, un message apparaîtra, fournissant des commentaires sur ce qui n'allait pas avec les données saisies et comment y remédier.

Les validations sont un ingrédient essentiel des formulaires bien conçus. Ils aident à protéger nos systèmes principaux contre la réception de données incorrectes, et ils aident à rendre l'expérience d'interaction avec notre formulaire aussi simple que possible pour nos utilisateurs.

Cette leçon explorera certaines des validations intégrées que vous pouvez utiliser avec les formulaires HTML. Nous nous plongerons également dans les validations de style avec CSS.

[**Required Validation**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-validation#required-validation)

Nous voudrons souvent nous assurer que des champs spécifiques ont été remplis avant de soumettre le formulaire, par exemple, l'e-mail et le mot de passe dans un formulaire de connexion. Pour rendre un champ obligatoire, nous lui ajoutons simplement l'attribut required :

<form action="#" method="get">

<div>

<label for="user\_email">\*Email:</label>

<input type="email" id="user\_email" name="user\_email" required>

</div>

<br>

<div>

<label for="user\_password">\*Password:</label>

<input type="password" id="user\_password" name="user\_password" required>

</div>

<button type="submit">Login</button>

</form>

Pour garantir une bonne expérience utilisateur et respecter les directives d'accessibilité, nous devons toujours indiquer les champs obligatoires. Cela sera souvent fait en ajoutant un astérisque (\*) à l'étiquette du champ requis comme nous l'avons fait dans l'exemple.

**Text Length Validation**

Parfois, nous voudrons que les utilisateurs entrent un minimum ou un maximum de texte dans un champ. Des exemples concrets d'utilisation de ces validations seraient l'ancienne limite de 140 caractères que Twitter avait dans son champ d'état ou des contraintes de longueur minimale et maximale sur un champ de nom d'utilisateur.

**Minimum Length Validation**

Pour ajouter la validation de longueur minimale, nous donnons au contrôle de formulaire un attribut minlength avec une valeur entière qui représente le nombre minimum de caractères que nous voulons autoriser dans le contrôle de formulaire :

<form action="#" method="get">

<div>

<textarea placeholder="What's happening?" minlength="3"></textarea>

</div>

<button type="submit">Tweet</button>

</form>

Essayez de saisir moins de trois caractères dans la zone de texte et cliquez sur le bouton tweet pour voir la validation en action.

**Maximum Length Validation**

Pour ajouter une validation de longueur maximale, nous donnons au contrôle de formulaire un attribut maxlength avec une valeur entière qui représente le nombre maximum de caractères que nous voulons autoriser dans le contrôle de formulaire :

<form action="#" method="get">

<div>

<textarea placeholder="What's happening?" maxlength="20"></textarea>

</div>

<button type="submit">Tweet</button>

</form>

Avec la validation de la longueur maximale, le navigateur empêchera les utilisateurs d'entrer plus de caractères que la valeur de l'attribut de longueur maximale. Essayez ceci par vous-même dans l'exemple ci-dessus.

**Combining Validations**

HTML permet d'appliquer autant de validations que l'on souhaite sur un champ de formulaire. Par exemple, nous pouvons donner à notre tweet textarea les validations minlength et maxlength :

<form action="#" method="get">

<div>

<textarea placeholder="What's happening?" minlength="5" maxlength="20"></textarea>

</div>

<button type="submit">Tweet</button>

</form>

Cela nous donne beaucoup plus de latitude pour contrôler ce que les utilisateurs saisissent.

[**Minimum and Maximum Validations**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-validation#minimum-and-maximum-validations)

Tout comme nous avons souvent besoin de contrôler la longueur des contrôles de formulaire basés sur du texte, il y aura de nombreuses situations où nous voudrons contrôler la plage de valeurs que les utilisateurs peuvent entrer dans les contrôles de formulaire basés sur des nombres.

Nous pouvons le faire avec les attributs min et max, ce qui nous permet de définir les limites inférieure et supérieure de la valeur saisie dans le contrôle du formulaire. Les attributs min et max ne fonctionnent qu'avec des contrôles de formulaire basés sur des nombres tels que les entrées de nombre, de dates et d'heure. Vous pouvez consulter la liste complète des éléments pris en charge sur la documentation de MDN.

Certains cas d'utilisation réels pour l'utilisation de ces validations seraient de limiter la quantité sur un formulaire de commande de produit ou de choisir le nombre de passagers sur un formulaire de réservation de vol.

**Min Validation**

Pour ajouter une validation de valeur minimale, nous donnons au contrôle de formulaire un attribut min avec une valeur entière qui représente le nombre minimum que nous voulons que le contrôle de formulaire accepte :

<form action="#" method="get">

<div>

<label for="quantity">Quantity</label>

</div>

<input type="number" id="quantity" name="quantity" min="1" value="0">

<div>

<button type="submit">Place Order</button>

</div>

</form>

Essayez de soumettre le formulaire avec une quantité de 0 pour voir la validation en action.

**Max Validation**

Pour ajouter une validation de valeur maximale, nous donnons au contrôle de formulaire un attribut max avec une valeur entière qui représente le nombre maximum que nous voulons que le contrôle de formulaire accepte :

<form action="#" method="get">

<div>

<label for="passengers">Passengers</label>

</div>

<input type="number" id="passengers" name="passengers" max="5" value="0">

<div>

<button type="submit">Book</button>

</div>

</form>

Essayez de soumettre le formulaire avec sept passagers pour voir la validation en action.

[**Pattern Validations**](https://www.theodinproject.com/paths/full-stack-javascript/courses/intermediate-html-and-css/lessons/form-validation#pattern-validations)

Pour nous assurer que nous obtenons les informations correctes des utilisateurs, nous voudrons souvent nous assurer que les données correspondent à un modèle particulier. Les applications du monde réel vérifieraient qu'un numéro de carte de crédit ou qu'un code postal est au bon format.

Pour ajouter une validation de modèle, nous donnons au contrôle de formulaire un attribut de modèle avec pattern comme valeur. Dans notre exemple, nous utilisons la validation de modèle pour nous assurer qu'un code postal américain est au format correct. 5 chiffres suivis d'un tiret facultatif et de 4 autres chiffres :

<form action="#" method="get">

<div>

<label for="zip\_code">Postal / Zip Code:</label>

</div>

<input type="text" id="zip\_code" name="zip\_code" pattern="(\d{5}([\-]\d{4})?)" required>

<div>

<button type="submit">Submit</button>

</div>

</form>

La saisie d'un code postal incorrect et la soumission du formulaire afficheront l'erreur de validation suivante dans le navigateur "Veuillez correspondre au format demandé". Ce n'est pas très utile car il ne communique pas comment résoudre le problème. Nous pouvons ajouter un message de validation plus descriptif en donnant à notre entrée un attribut title :

<form action="#" method="get">

<div>

<label for="zip\_code">Postal / Zip Code:</label>

</div>

<input type="text" id="zip\_code" name="zip\_code" pattern="(\d{5}([\-]\d{4})?)" title="Please enter a valid zip code, example: 65251" required>

<div>

<button type="submit">Submit</button>

</div>

</form>

Lorsque nous soumettons le formulaire avec un code postal incorrect, nous recevons un message plus utile nous indiquant exactement ce qui s'est passé et comment le résoudre. Il est également recommandé d'utiliser un attribut d'espace réservé pour montrer aux utilisateurs un exemple du modèle attendu qu'ils doivent saisir :

<form action="#" method="get">

<div>

<label for="zip\_code">Postal / Zip Code:</label>

</div>

<input type="text" id="zip\_code" name="zip\_code" pattern="(\d{5}([\-]\d{4})?)" title="Please enter a valid zip code, example: 65251" placeholder="65251" required>

<div>

<button type="submit">Submit</button>

</div>

</form>

L'attribut pattern ne peut être utilisé que sur les éléments <input>. Certains éléments d'entrée valident déjà des données qui correspondent à un certain modèle. Par exemple, le champ de saisie de l'e-mail qui s'assurera qu'un e-mail valide est entré et l'élément d'entrée de l'url qui vérifiera que l'url commence par http ou https :

<form action="#" method="get">

<div>

<label for="email">Email:</label>

</div>

<input type="email" id="email" name="email">

<div>

<label for="website">Website:</label>

</div>

<input type="url" id="website" name="website">

<div>

<button type="submit">Submit</button>

</div>

</form>

**Styling Validations**

Nous pouvons cibler les contrôles de formulaire qui ont réussi ou échoué les validations en utilisant les pseudo-classes :valid et :invalid. Pour voir cela en action, nous utiliserons notre exemple de courrier électronique et de site Web que nous avons examiné précédemment :

<form action="#" method="get">

<div>

<label for="email">\*Email</label>

</div>

<input type="email" id="email" name="email">

<div>

<label for="website">\*Website</label>

</div>

<input type="url" id="website" name="website">

<div>

<button type="submit">Submit</button>

</div>

</form>

Tout d'abord, nous ciblons toutes les entrées valides et leur donnons une bordure verte. Nos entrées d'e-mail et d'URL ont initialement une bordure verte car ce ne sont pas des champs obligatoires et elles sont valides. Lorsqu'un champ est invalide, nous lui donnons une bordure rouge à la place. Essayez de saisir une adresse e-mail et une URL non valides pour voir à quoi cela ressemble.

**Conclusion**

Les validations intégrées vous permettront de vous assurer que vos utilisateurs entrent les données correctes. Ils sont rapides et faciles à ajouter. Cependant, ils ont leurs limites.

Parfois, vous devrez inclure des validations que les validations intégrées ne pourront pas faire. Par exemple, valider qu'une entrée de mot de passe et une entrée de confirmation de mot de passe ont la même valeur ou valider qu'un nom d'utilisateur n'a pas déjà été pris. Nous sommes également limités dans ce que nous pouvons faire avec le style des messages de validation et le contenu qu'ils contiennent.

Dans ce cas, nous devrons faire preuve de créativité et effectuer des validations personnalisées en utilisant JavaScript et CSS. Nous verrons comment obtenir la validation via JavaScript dans une prochaine leçon.

Il convient également de noter que les validations côté client ne sont pas une solution miracle pour garantir que les utilisateurs saisissent les données correctes. Pour garantir l'intégrité de toutes les données utilisateur entrant dans nos systèmes, nous devons également mettre en place des validations côté serveur. Nous aborderons ce côté des validations plus tard dans le programme.