
04 – Les formulaires

Sommaire

1. Page html vierge	4
2. Balise formulaire.....	4
3. Ajouter des éléments dans le formulaire.....	5
4. Bulles d'aide.....	5
5. Les derniers champs de formulaire	6
6. Ajout du placeholder	7
7. CSS pour placeholder	8
8. CSS de focus	8
9. Toute la mise en page CSS	9
10. Police et taille du texte	9
11. Style de la liste.....	10
12. Lignes horizontales	11
13. Entête du formulaire	11
14. Taille et position des éléments du formulaire.....	12
15. Style des éléments du formulaire.....	13
16. Un peu d'animation lors du focus	14

17. Gestion des champs obligatoires en HTML5.....	15
18. Style pour les champs obligatoires.....	15
19. Validation des champs en HTML5	16
20. Les champs valides et invalides	17
21. Message d'erreur permanent	17
22. Les types de champs	19
23. Select	20
24. textarea	20
25. button	21
26. input	21
27. datalist.....	24
28. fieldset.....	24
29. Compatibilité HTML5	24
30. Exercices.....	25

Les formulaires

Dans cette partie, nous allons créer un formulaire *HTML5* et *CSS3*, en explicitant chaque étape de la conception. Afin de bien intégrer ces notions, il est demandé de dérouler toutes les étapes et de les tester. Il s'agit d'un formulaire de contact comprenant un champ email, un champ nom, un champ site web et une zone de message. Une validation des données sera aussi implémentée.

1. Page html vierge

Ces quelques lignes n'ont plus de secret pour vous.

2. Balise formulaire

```
<form class='contact_form'
      action='http://www.action-
creation.fr/cnam/HTML_CSS/chapitre4/validateForm.php'
      method='post'
      name='contact_form'
      target='_blank'>
</form>
```

La balise formulaire intégrera ensuite les différents éléments du formulaire. Elle admet des paramètres :

- **id, class et name** : comme toutes les balises
- **action** : adresse du script –ou de la page- auquel envoyer les données lors de la validation du formulaire.
- **target** : destination (iframe) utilisée pour la validation du formulaire
- **enctype** : endocage des données du formulaire, particulièrement utile pour les formulaires intégrant des envois de fichiers
- **method** : méthode d'envoi des données
 - *post* : les données sont envoyées dans l'entête de la requête.
 - *get* : les données sont envoyées dans l'url. Ainsi, il est possible de simuler les informations envoyées par un formulaire get en ajoutant les données dans la ligne de commande de la manière suivante (n'hésitez pas à tester) :

http://www.action-creation.fr/cnam/HTML_CSS/chapitre4/validateForm.php?nom=RENARD&prenom=Arnaud&ville=Reims

Les informations *nom=valeur* sont envoyées précédées d'un **?** (point d'interrogation) et séparées d'un **&** (esperluette)

3. Ajouter des éléments dans le formulaire

Pour créer un formulaire structuré, nous choisissons d'intégrer ses éléments (label, input, ...) dans une liste, à commencer par un titre et un premier champ :

```
<ul>
  <li>
    <h2>Contactez-nous</h2>
    <span class="champ_obligatoire">*
Information obligatoire</span>
  </li>
  <li>
    <label for="l_nom">Nom : </label>
    <input type="text" name="nom"
id="l_nom" >
  </li>
</ul>
```



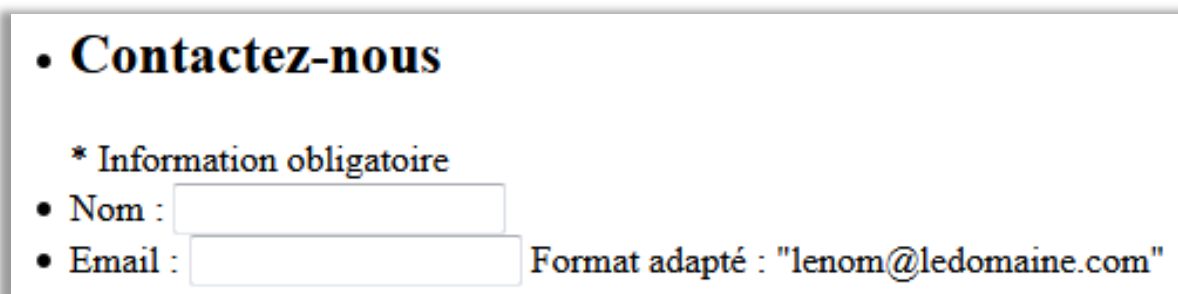
Pour chaque champ de formulaire input, il convient d'indiquer le type, ici du texte (*type='text'*), et de lui donner un nom (attribut *name*). C'est ce nom qui sera associée à la valeur envoyée au serveur.

Pour chaque champ il est aussi possible d'associer un *label* : l'attribut *for* du *label* a la même valeur que l'attribut *id* de l'*input*. Ainsi le label constitue l'intitulé du champ de formulaire. Certains navigateurs en font un usage spécifique.

4. Bulles d'aide

Pour le champ email, nous choisissons comme prévu d'ajouter une zone destinée à gérer le texte d'aide.

```
<li>
  <label for="l_email">Email : </label>
  <input type="text" name="email" id="l_email">
  <span class="form_aide">Format adapté :
"lenom@ledomaine.com"</span>
</li>
```



5. Les derniers champs de formulaire

Il est maintenant temps d'ajouter les champs manquants à notre formulaire, toujours dans des éléments de la liste.

```
<li>
    <label for="l_siteweb">Site Web :</label>
    <input type="text" name="siteweb" id="l_siteweb">
    <span class="form_aide">Format adapté :
"http://www.ledomaine.fr"</span>
</li>
<li>
    <label for="l_message">Message : </label>
    <textarea name="message" cols="45" rows="5"
id="l_message"></textarea>
</li>
<li>
    <button class="validationButton" type="submit">Envoyer</button>
</li>
```

• Contactez-nous

*** Information obligatoire**

- Nom :
- Email : Format adapté : "lenom@ledomaine.com"
- Site Web : Format adapté : "http://www.ledomaine.fr"

• Message :

•

Dès à présent, le formulaire est valide, vous pouvez remplir les champs et appuyer sur le bouton valider pour vérifier le fonctionnement du formulaire.

6. Ajout du placeholder

L'attribut *placeholder* permet de placer un texte par défaut quand la zone de texte est vide.

```
<li>
    <label for="l_nom">Nom et Prénom : </label>
    <input type="text" name="nom" id="l_nom" placeholder="Arnaud
RENARD" >
</li>
<li>
    <label for="l_email">Email : </label>
    <input type="text" name="email" id="l_email"
placeholder="arenard@lecnam.net" >
    <span class="form_aide">Format adapté :
"lenom@ledomaine.com"</span>
</li>
<li>
    <label for="l_siteweb">Site Web :</label>
    <input type="text" name="siteweb" id="l_siteweb"
placeholder="www.renard.com/arnaud" >
    <span class="form_aide">Format adapté :
"http://www.ledomaine.fr"</span>
</li>
```

• Contactez-nous

* Information obligatoire

- Nom et Prénom :
- Email : Format adapté : "lenom@ledomaine.com"
- Site Web : Format adapté : "http://www.ledomaine.fr"

- Message :

-

7. CSS pour placeholder

Le style du placeholder peut bien sûr être spécifié. La syntaxe est spécifique pour chaque navigateur le supportant.

```
::-moz-placeholder {  
    color: blue;  
}  
::-webkit-input-placeholder {  
    color: blue;  
}
```

The screenshot shows a contact form with the following elements:

- Contactez-nous**
- * Information obligatoire**
- Nom et Prénom :**
- Email :** **Format adapté :** "lenom@ledomaine.com"
- Site Web :** **Format adapté :** "http://www.ledomaine.fr"
- Message :**
- Envoyer** (button)

8. CSS de focus

Webkit (le moteur utilisé entre autres par Chrome et Safari) ajoute un style autour des éléments qui sont actuellement sélectionnés (qui ont le focus). Afin d'éliminer ce comportement et ainsi uniformiser l'affichage du formulaire sur les différents navigateurs. Voici le rendu normal :

• Email : Format adapté : "lenom@ledomaine.com"

Et celui en modifiant la feuille de style :

```
*:focus {outline: none;}
```

• Email : Format adapté : "lenom@ledomaine.com"

9. Toute la mise en page CSS

Il est maintenant temps de finaliser la mise en page de notre formulaire. En plus de l'exemple, la méthodologie est à prendre en exemple : on s'attache à mettre en forme prioritairement la structure puis on tend vers le détail petit à petit.

10. Police et taille du texte

```
body {font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans Unicode"}
.contact_form h2, .contact_form label {font-family:Georgia, Times, "Times New Roman";}
.form_aide, .champ_obligatoire {font-size: 11px;}
```

- **Contactez-nous**
 - * Information obligatoire
 - Nom et Prénom :
 - Email : Format adapté : "lenom@ledomaine.com"
 - Site Web : Format adapté : "http://www.ledomaine.fr"
 -
 - Message :
 -

11. Style de la liste

L'application de ces quelques lignes de CSS permet de donner un peu de structure à la liste.

```
.contact_form ul {  
    width:750px;  
    list-style-type:none;  
    list-style-position:outside;  
    margin:0px;  
    padding:0px;  
}  
.contact_form li{  
    padding:12px;  
    border-bottom:1px solid #eee;  
    position:relative;  
}
```

Contactez-nous

* Information obligatoire

Nom et Prénom :

Email : Format adapté : "lenom@ledomaine.com"

Site Web : Format adapté : "http://www.ledomaine.fr"

Message :

dd

12. Lignes horizontales

Ajoutons une bordure inférieure sur le premier et sur le dernier élément de la liste.

```
.contact_form li:first-child, .contact_form li:last-child {  
    border-bottom:1px solid #777;  
}
```

Contactez-nous

* Information obligatoire

Nom et Prénom :

Email : Format adapté : "lenom@ledomaine.com"

Site Web : Format adapté : "http://www.ledomaine.fr"

Message :

13. Entête du formulaire

L'entête du formulaire consiste en un titre et une coloration de la légende indiquant que certains champs sont obligatoires.

```
.contact_form h2 { margin:0; display: inline; }  
.champ_obligatoire { color:#d45252; margin:5px 0 0 0; display:inline;  
float:right; }
```

Contactez-nous * Information obligatoire

Nom et Prénom :

Email : Format adapté : "lenom@ledomaine.com"

Site Web : Format adapté : "http://www.ledomaine.fr"

Message :

14. Taille et position des éléments du formulaire

La mise en forme de la structure du formulaire permet d'aligner les champs et de rendre l'ensemble harmonieux.

```
.contact_form label {  
    width:150px;  
    margin-top: 3px;  
    display:inline-block;  
    float:left;  
    padding:3px;  
}  
.contact_form input {  
    height:20px;  
    width:220px;  
    padding:5px 8px;  
}  
.contact_form textarea {  
    padding:8px;  
    width:300px;  
}  
.contact_form button {  
    margin-left:156px;  
}
```

Contactez-nous * Information obligatoire

Nom et Prénom :

Amaud RENARD

Email :

arenard@lecnam.net

Format adapté : "lenom@ledomaine.com"

Site Web :

www.renard.com/arnaud

Format adapté : "http://www.ledomaine.fr"

Message :

dd

Envoyer

15. Style des éléments du formulaire

Il est maintenant temps d'ajouter des styles de finition pour un effet moderne : ombres, arrondis, dégradés, background.

```
/* style CSS3 des inputs */
.contact_form input, .contact_form textarea {
    border:1px solid #aaa;
    box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
    border-radius:2px;
}
.contact_form input:focus, .contact_form textarea:focus {
    background: #fff;
    border:1px solid #555;
    box-shadow: 0 0 3px #aaa;
}
/* style CSS3 des Button */
button.validationButton {
    background-color: #68b12f;
    background: -webkit-gradient(linear, left top, left bottom,
from(#68b12f), to(#50911e));
    background: -webkit-linear-gradient(top, #68b12f, #50911e);
    background: -moz-linear-gradient(top, #68b12f, #50911e);
    background: -ms-linear-gradient(top, #68b12f, #50911e);
    background: -o-linear-gradient(top, #68b12f, #50911e);
    background: linear-gradient(top, #68b12f, #50911e);
    border: 1px solid #509111;
    border-bottom: 1px solid #5b992b;
    border-radius: 3px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    -ms-border-radius: 3px;
    -o-border-radius: 3px;
    box-shadow: inset 0 1px 0 0 #9fd574;
    -webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;
    -moz-box-shadow: 0 1px 0 0 #9fd574 inset;
    -ms-box-shadow: 0 1px 0 0 #9fd574 inset;
    -o-box-shadow: 0 1px 0 0 #9fd574 inset;
    color: white;
    font-weight: bold;
    padding: 6px 20px;
    text-align: center;
    text-shadow: 0 -1px 0 #396715;
}
button.validationButton:hover {
    opacity:.85;
    cursor: pointer;
}
button.validationButton:active {
    border: 1px solid #20911e;
    box-shadow: 0 0 10px 5px #356b0b inset;
    -webkit-box-shadow:0 0 10px 5px #356b0b inset;
    -moz-box-shadow: 0 0 10px 5px #356b0b inset;
    -ms-box-shadow: 0 0 10px 5px #356b0b inset;
    -o-box-shadow: 0 0 10px 5px #356b0b inset;
}
```

Contactez-nous

* Information obligatoire

Nom et Prénom :

Email :

Format adapté : "lenom@ledomaine.com"

Site Web :

Format adapté : "http://www.ledomaine.fr"

Message :

Envoyer

16. Un peu d'animation lors du focus

Afin d'ajouter un peu d'interactivité, on va créer un *padding* de 70 px (`padding-right:70px;`) sur la droite des champs sélectionnés (`:focus`), c'est ce que l'on fait en modifiant le style existant. A vous de tester.

```
.contact_form input:focus, .contact_form textarea:focus {
    background: #fff;
    border:1px solid #555;
    box-shadow: 0 0 3px #aaa;
    padding-right:70px;
}
```

En complément, l'ajout des 4 lignes au style ci-dessous permet de transformer les modifications de padding en animations de 0,25 secondes :

```
/* style CSS3 des inputs, avec transition quand un champ prend le focus
*/
.contact_form input, .contact_form textarea {
    border:1px solid #aaa;
    box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
    border-radius:2px;
    -moz-transition: padding .25s;
    -webkit-transition: padding .25s;
    -o-transition: padding .25s;
    transition: padding .25s;
}
```


19. Validation des champs en HTML5

La validation des éléments s'appuie sur l'attribut `type` du champ. Depuis le début de notre démonstration, les `input` sont des texte (`type='text'`), mais il en existe beaucoup, `email` et `url` sont aussi des types acceptés. En combinant le type à l'attribut `required`, le navigateur est en mesure de vérifier le contenu des champs de formulaire.

Il convient donc de modifier les inputs de l'email et de l'url :

```
<input type="email" name="email" id="l_email" placeholder="..." required >  
<input type="url" name="siteweb" id="l_siteweb" placeholder="..." required >
```

De plus, en fonction du type, un smartphone est en mesure de vous proposer le clavier tactile adapté. Par exemple, s'il s'agit d'une adresse mail, l'arobase (@) sera accessible directement.



Si toutefois vous souhaitez conserver le type (et donc le clavier) sans la validation, il est possible d'ajouter l'attribut `novalidate` (sans valeur) dans le formulaire.

20. Les champs valides et invalides

La première étape est d'ajouter un style spécifique aux *input* en mode *focus*, en fonction du cas :

- Si le champ est valide, une ombre verte et un check vert en background
- Si le champ n'est pas valide, une ombre rouge et une croix rouge en background

```
/* champ focus et invalide */
.contact_form input:focus:invalid, .contact_form textarea:focus:invalid {
    background: #fff url(invalide.png) no-repeat 98% center;
    box-shadow: 0 0 5px #d45252;
    border-color: #b03535;
}
```

Nom et Prénom :

Arnaud RENARD



```
/* champ focus et valide */
.contact_form input:required:valid, .contact_form textarea:required:valid {
    background: #fff url(valide.png) no-repeat 98% center;
    box-shadow: 0 0 5px #5cd053;
    border-color: #28921f;
}
```

Nom et Prénom :

Sébastien PATRICK



21. Message d'erreur permanent

Il reste encore à gérer les messages d'erreur présents à côté des champs email et url. En effet, outre sa mise en forme, il faudrait aussi pouvoir le faire apparaître uniquement en cas de focus (ie, quand ils sont frères est en focus).

- La première étape consiste à mettre le texte dans un bloc, avec de la couleur et des coins arrondis.

```
.form_aide {
    background: #d45252;
    border-radius: 3px 3px 3px 3px;
    color: white;
    margin-left: 8px;
    padding: 1px 6px;
    z-index: 999; /* mettre au premier plan */
    position: absolute; /* pour gérer simplement les messages sur
plusieurs lignes */
}
```

Email :

arenard@lecnam.net



Format adapté : "lenom@ledomaine.com"

- CSS permet de définir le style d'un élément inexistant situé juste avant un autre, à l'aide de la notation `::before`

```
.form_aide::before {
  content: "\25C0"; /* triangle plein gauche au format unicode échappé */
  color: #d45252;
  position: absolute;
  top: 1px;
  left: -6px;
}
```

La propriété `content` permet d'ajouter du texte, et même un caractère spécial au format unicode : http://en.wikipedia.org/wiki/List_of_Unicode_characters

Email : ★ Format adapté : "lenom@ledomaine.com"

- Par défaut, il ne faut pas que ce message soit affiché. On ajoute ajouter un élément visuel pour mettre en `display: none`;
- Si l'on veut que le message soit affiché uniquement dans le champ correspondant est en focus, on ajoute la règle suivante :

```
.contact_form input:focus + .form_hint {display: inline;}
```

Email : ✓ Format adapté : "lenom@ledomaine.com"

- Pour aller plus loin, il est intéressant que le message s'affiche sur fond vert quand le champ est valide, sans oublier la flèche du before.

```
.contact_form input:required:valid + .form_aide {background: #28921f;}
.contact_form input:required:valid + .form_aide::before {color: #28921f;}
```

Email : ✓ Format adapté : "lenom@ledomaine.com"

Email : ! Format adapté : "lenom@ledomaine.com"

22. Les types de champs

Après avoir déroulé cet exemple explicatif, il est temps d'avoir une liste exhaustive des champs. Tous ne seront pas explicités en détail. Voici la liste des champs :

- *select*, la liste de choix, avec des *option* et des *optgroup*
- *textarea*, la zone de texte sur plusieurs lignes
- *button*, les boutons
- *input*, avec les types suivants : *text*, *password*, *file*, *radio*, *checkbox*, *submit*, *image*, *hidden*, *reset*, *button*, *email*, *url*, *tel*, *search*, *date*, *time*, *datetime*, *month*, *week*, *number*, *range*, *color*
- *keygen* : permet de générer des clés pour le cryptage des données de formulaire
- *output* : pour les résultats du formulaire, dans le cadre d'un calcul par exemple
- *datalist* : zone de texte libre mais s'appuyant sur une liste avec complétion

Pour chacun de ces champs, les attributs *name*, *class*, et *id* sont possibles. Les champs peuvent être désactivés (et visuellement grisés) avec l'attribut *disable*. Il est possible d'empêcher la modification d'un champ avec l'attribut *readonly*.

23. Select

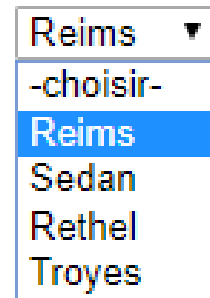
Les listes de choix sont définies par la balise *select*.

- **size** : le nombre de ligne à afficher, 1 par défaut
- **multiple** : attribut sans valeur autorisant à sélectionner plusieurs éléments dans la liste

La balise *select* contient des *option* :

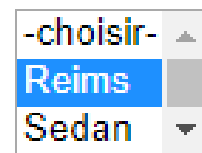
- **value** : la valeur de l'option, qui sera envoyée à la validation du formulaire
- **selected** : pour choisir quelle est l'option sélectionnée

```
<select name="ville">
  <option value="">-choisir-</option>
  <option value="0" selected>Reims</option>
  <option value="1">Sedan</option>
  <option value="2">Rethel</option>
  <option value="3">Troyes</option>
</select>
```



A single-select dropdown menu. The current selection is 'Reims'. The options are: -choisir-, Reims, Sedan, Rethel, Troyes.

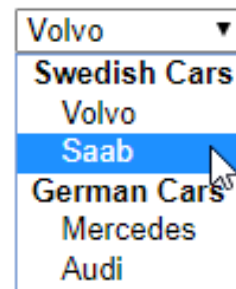
```
<select name="ville" size="3" multiple>
  <option value="">-choisir-</option>
  <option value="0">Reims</option>
  <option value="1">Sedan</option>
  <option value="2">Rethel</option>
  <option value="3">Troyes</option>
</select>
```



A multi-select dropdown menu with size="3". The current selections are 'Reims' and 'Sedan'. The options are: -choisir-, Reims, Sedan.

La version évoluée permet des groupes grâce à l'ajout des *optgroup* :

```
<select>
  <optgroup label="Swedish Cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </optgroup>
  <optgroup label="German Cars">
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
```



A grouped multi-select dropdown menu. The current selection is 'Saab'. The options are grouped into 'Swedish Cars' (Volvo, Saab) and 'German Cars' (Mercedes, Audi).

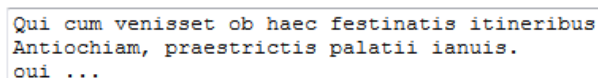
24. textarea

textarea définit une zone de texte multi-lignes.

Les attributs sont suivants :

- **cols** : définit le nombre de colonnes (la largeur)
- **rows** : définit le nombre de ligne (la hauteur)

```
<textarea rows="4" cols="50">Qui cum venisset ob haec festinatis
itineribus Antiochiam, praestricis palatii ianuis.
oui ...
</textarea>
```



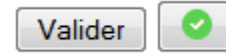
Qui cum venisset ob haec festinatis itineribus Antiochiam, praestricis palatii ianuis. oui ...

25. button

La balise bouton permet de déclencher une action dans un formulaire, et même en dehors.

Elle accepte un attribut type :

- **button** : bouton générique
- **reset** : permet de remettre à zéro tous les champs
- **submit** : permet de valider le formulaire



```
<button type='submit' name='validate'>Valider</button>
<button type='button' name='validate'>
  <img src='valide.png' alt='Valider'>
</button>
```

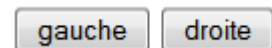
26. input

Le *input* est le champ de base en HTML. Il comprend toutefois un attribut *type* qui permet de créer différents champs. Les *input* intègrent aussi l'attribut *value* pour spécifier la valeur initiale de l'*input*.

type='button' : il s'agit d'un bouton sans action particulière. L'action sera définie en javascript.

type='submit' : permet de créer un bouton de validation de formulaire. L'intérêt de l'*input* par rapport au *button* est de permettre d'associer une variable (nom = valeur) à chaque bouton. Seul la variable correspondant au bouton cliqué est envoyé par le navigateur :

```
<input type='button' name='envoi' value='gauche'>
<input type='button' name='envoi' value='droite'>
```



type='reset' : il s'agit d'un bouton qui va réinitialiser le formulaire pour lui redonner les valeurs par défaut.

type='hidden' : c'est un champ caché, que l'internaute ne voit pas et ne peut pas modifier. Il est utilisé par le développeur de la page HTML pour ajouter des informations à destination du serveur qui va traiter les informations envoyées par le formulaire.

```
<input type='hidden' name='origine' value='arnaud'>
```

type='file' : ce champ permet de sélectionner un fichier à uploader sur le serveur en même temps que l'envoi des données de formulaire.

```
<input type="file" name="fichier1" >
```

Parcourir...

Aucun fichier sélectionné.

Attention, pour être en mesure d'envoyer un fichier, le formulaire doit utiliser la méthode *post* et se voir attribuer une option d'encodage :

```
<form ... enctype='multipart/form-data' >
```

type='checkbox' : c'est une case à cocher qui prend la valeur vraie ou fausse. L'attribut *checked* permet de la cocher par défaut.

```
<input type="checkbox" name="majeur" checked>Majeur  
<input type="checkbox" name="etudiant">Etudiant
```

☒ Majeur ☐ Etudiant

type='radio' . Les radio-buttons fonctionnent par groupe –avec un nom commun- ou seul un bouton est sélectionné. L'attribut *checked* permet de choisir l'input sélectionné par défaut. La valeur envoyée par le formulaire est celle du bouton enfoncé. A défaut, aucune valeur n'est envoyée.

```
<input type="radio" name="nrj" value="essence" checked>essence -  
<input type="radio" name="nrj" value="diesel">diesel<br>  
<input type="radio" name="nrj" value="electricite">electricité
```

L'illustration ci-dessous montre un autoradio du siècle précédent, où l'on sélectionne une station en appuyant sur le bouton correspondant. C'est cet appareil qui a donné son nom au champ HTML.



☒ essence - ☐ diesel
☐ electricité

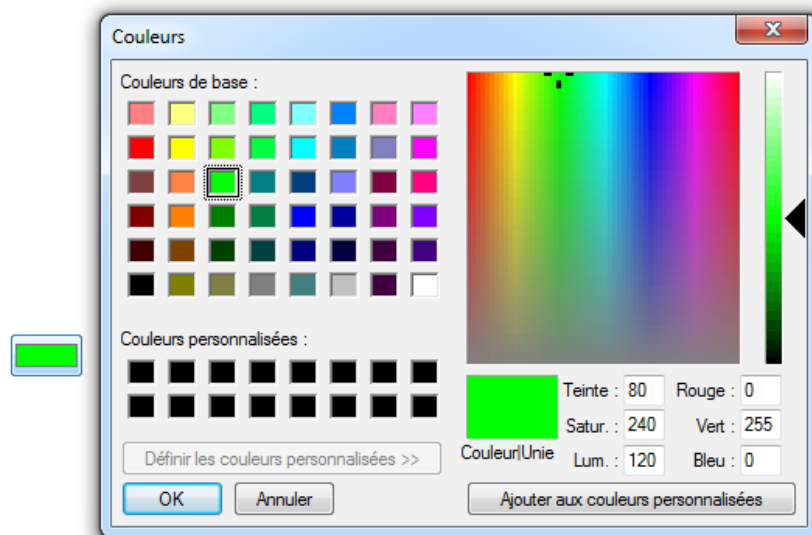
Les autres *input* offrent à l'internaute de modifier le contenu du champ, sur le modèle de `type='text'`, mais avec la propriété de vérification au moment de la validation. Il s'agit des types *text*, *color*, *email*, *url*, *tel*, *search*, *date*, *time*, *datetime*, *month*, *week*, *number* et *range*

| `<input type="text" name="var1" >`

Chrome, par exemple, apporte un support particulier à ces champs :

`type='color'` : sélection de couleur.

| `<input type='color' name='couleur' value=''>`



`<input type="search" name="var1" >`

`<input type="date" name="var2" >`

`<input type="time" name="var3" >`

`<input type="month" name="var4" >`

`<input type="week" name="var5" >`

`<input type="number" name="var6" >`

`<input type="range" name="var7" >`

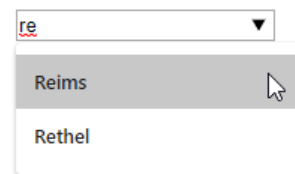
27. datalist

datalist permet de définir une liste, de cette manière :

```
<datalist id="villes">
  <option value="Reims">
  <option value="Sedan">
  <option value="Rethel">
  <option value="Troyes">
</datalist>
```

Ainsi, il est possible d'associer un champ input à cette liste (l'attribut *list* de l'input est l'id de la *datalist*) afin de permettre une complétion sur la base de cette liste.

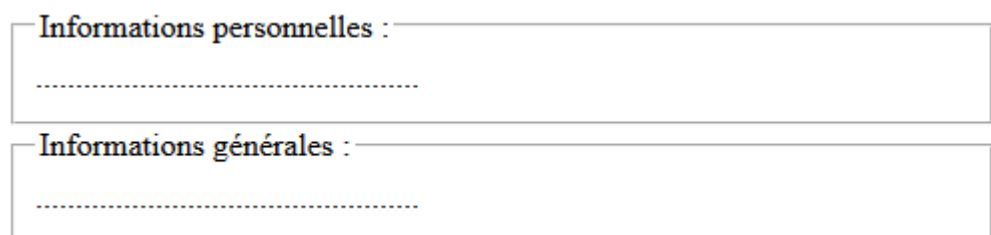
```
<input list="villes" type="text" >
```



28. fieldset

Le fieldset permet de regrouper les champs visuellement.

```
<fieldset>
  <legend>Informations personnelles : </legend>
  .....
</fieldset>
<fieldset>
  <legend>Informations générales : </legend>
  .....
</fieldset>
```



29. Compatibilité HTML5

Les formulaires html5 seront affichés comme prévu sur les navigateurs compatibles. Sur les navigateurs plus anciens, ce ne sera pas le cas, mais il est possible de rendre le formulaire fonctionnel, modulo l'utilisation, entre autre, de quelques lignes de code *javascript* / *jquery*.

Ce lien donne un aperçu de la technique.

<http://41mag.fr/tutoriel-html5-comment-faire-un-formulaire-complet.html>

30. Exercices

L'ensemble des formulaires enverront leurs données à la page

http://www.action-creation.fr/cnam/HTML_CSS/chapitre4/validateForm.php

Exercice 1 :

Reproduire le formulaire suivant. Les données sont envoyées en mode *get* et apparaissent donc dans l'URL

Valeur 1:

Opérateur:

Valeur 2:

Exercice 2 :

Modifier l'exercice précédent pour qu'il puisse envoyer les données en mode *post*.

Une fois le formulaire validé, essayez de recharger la page (*Ctrl+r*). Que se passe-t-il ? Pourquoi ?

Exercice 3 :

Ecrire le formulaire suivant, en mode *get*. Pour obtenir un résultat personnalisé, les champs doivent être nommés *val1*, *opérateur* et *val2*.

Valeur 1:

Opérateur:

Valeur 2:

Exercice 4 :

Ecrire le formulaire suivant (les vérifications pourraient être faites en *javascript* dans un chapitre prochain, elles ne sont pas à faire pour cet exercice) :

Vos coordonnées

Nom :	<input type="text"/>
Prenom :	<input type="text"/>
Adresse e-mail :	<input type="text"/>
Verification adresse e-mail :	<input type="text"/>
Mot de passe :	<input type="password"/>
Verification mot de passe :	<input type="password"/>

Votre réservation

Type de chambre :	<input type="text" value="Deux lits king size"/>
Vue :	<input type="radio"/> Parking • <input type="radio"/> Jardin
Option :	<input type="checkbox"/> Parking • <input type="checkbox"/> Petit déjeuner • <input type="checkbox"/> Accès à la piscine
Demandes spéciales :	<div><input type="text"/></div>

Exercice 5 :

On considère le formulaire suivant :



Formulaire de forum

Nom :

Prénom :

Adresse e-mail :

Message :

Envoyer

Les 3 premiers points précisent l'aperçu ci-dessus

1. Ecrire le formulaire puis modifiez-le, étape par étape :
2. Les champs se nomment *nom*, *prenom*, *email* et *message*
3. Le formulaire se valide en mode *post*.

Nous allons ajouter des éléments dans le formulaire

4. Remplacer le bouton « envoyer » par deux boutons ayant le même nom (name) *envoi*, l'un vaut (value) *normal* et l'autre *important*.
5. Un champ caché nommé *specialMode* valant *forum* est ajouté au formulaire.
6. Pensez à poster avec vos noms et prénom.
7. Ajouter un champ nommé *image* permettant d'uploader une image en plus de votre message.
8. Ajoutez le style nécessaire et intégrer ce formulaire dans une page web. Le formulaire se validant dans une *iframe*. (target = *name* de l'iframe)

31. QCM :

1. On peut insérer un formulaire dans un autre formulaire
 - a. Oui
 - b. Non
2. Pour créer une zone de saisie d'un mot de passe, on utilisera :
 - a. Un password-area
 - b. Un input de type hidden
 - c. Un input type password
 - d. Un input type text avec l'option format='password'
3. Pour créer une zone de texte multi-lignes, on utilisera :
 - a. Un input type='multiline'
 - b. Un textarea
 - c. Plusieurs inputs
 - d. Un multiline
4. Si on donne le même nom (valeur de l'attribut *name*) à plusieurs boutons radio :
 - a. Ça ne marchera pas
 - b. On pourra tous les sélectionner
 - c. On pourra en sélectionner un seul
 - d. On pourra en sélectionner deux
5. On peut créer un `<fieldset>` en incluant une `<legend>`
 - a. Juste avant la balise d'ouverture `<fieldset>`
 - b. Juste en dessous de la balise d'ouverture `<fieldset>`
 - c. Juste au-dessus de la balise d'ouverture `<fieldset>`
6. La balise `fieldset` permet
 - a. De grouper différents éléments d'un formulaire
 - b. De définir une légende pour un groupe d'élément
 - c. De sélectionner les champs de formulaire à envoyer
7. L'attribut `target`
 - a. Permet de définir l'URL à laquelle envoyer les données du formulaire
 - b. Permet de définir la cible (onglet, page de navigateur, cadre ...) dans laquelle le traitement du formulaire va s'opérer
 - c. Permet de choisir avec quelle méthode de chiffrement les données seront envoyées au serveur.
8. L'attribut `action`
 - a. Permet de définir l'URL à laquelle envoyer les données du formulaire
 - b. Permet de définir la cible (onglet, page de navigateur, cadre ...) dans laquelle le traitement du formulaire va s'opérer
 - c. Permet de choisir l'action `javascript` qui sera déclenchée pour traiter le formulaire
9. La méthode `get`
 - a. Envoie les données dans l'URL
 - b. Envoie les données à part de l'URL, dans les options de la requête http
 - c. Peut-être simulée avec une URL contenant des caractères `?` `&` et `=`
 - d. Est plus forte que la méthode post

10. Quelles sont les allégations vraies :

- a. Les champs *password* sont automatiquement chiffrés
- b. Le serveur peut faire toute confiance dans les données transmises par le formulaire
- c. Les données des formulaires sont chiffrées avant la transmission au serveur
- d. Elles sont chiffrées uniquement pour les pages https (le cadenas à côté de la barre d'adresse)

32. Conclusion :

A l'intersection de HTML et de CSS, ce chapitre sur les formulaires avait vocation à vous montrer l'ensemble des possibilités offertes pour envoyer des informations au serveur et de clore l'introduction à ces deux technologies. Le prochain chapitre permettra d'appliquer ces notions à un cas concret et d'aller plus loin dans l'accessibilité des sites web pour le rendre responsive.