Cours (html et css) : thème 2 - Question 3

Question 3 : La résolution de tous les problèmes de gestion est-elle automatisable ?

Notions abordées :
- introduction aux langage HTML et CSS ;
- notions HTML : structure d'une page web, ressources externes, balises ouvrantes, balises fermantes et auto-fermantes, listes, tableaux, liens, images, sous-titres, formulaires, champs, boutons ;
- notions CSS : sélecteurs, propriétés, mise en forme de textes, mise en forme des fonds, dimensionnement et positionnement, etc.

1. Préliminaire

Le présent cours est un préalable au cours « Cours - Q3 - web et php » au travers duquel nous aborderons une partie centrale du programme de terminale STMG. Ce cours consistera en une introduction à la programmation PHP. Cette partie sera centrale en ce qu'elle est des plus nécessaires afin que vous puissiez mener à bien vos projets de terminale. Elle est également centrale en ce qu'elle fait écho à diverses notions que nous avons étudiées et continuerons d'étudier tout au long de l'année.

Cependant, avant de pouvoir aborder la programmation web, il vous faut acquérir quelques bases pour décrire et mettre en forme le contenu de pages web.

Remarques ! On tiendra bien compte des deux remarques suivantes :
- Le présent cours ne vise pas à former des experts en HTML/CSS. Il s'agit bien d'une introduction ! Ainsi, on omet volontairement toutes les bonnes pratiques du web. Autrement dit, les pages web réalisées par des développeurs chevronnés (que vous serez sans doute dans quelques années) font intervenir de nombreuses balises et des notions que nous n'étudions pas ici.
- Au travers de ce cours, l'on ne traite que de pages web « locales » qui peuvent être immédiatement ouvertes au moyen d'un navigateur. Il n'est pas ici question de « serveur web ». Les différences rencontrées seront entre autres traitées au travers du cours « Cours - Q3 - web et php ».

2. Créer une page HTML

2.1. La structure d'un page HTML

Le langage HTML est un langage de description de contenu. Il n'est donc pas un langage de programmation. Il s'agit d'un format XML permettant spécifiquement de décrire le contenu d'une page web.

Comme le langage XML, le langage HTML est constitué de balises :
- Balises ouvrantes. Exemple : <html>
- Balises fermantes. Exemple : </html>
- Balises « auto-fermantes ». Exemple : <input type="text" placeholder="[Votre nom]" />
- Attributs. Une balise peut avoir des attributs. La balise précédente nous fournit un exemple. En effet,

la balise « input » précédente comporte deux attributs : type et placeholder. Un attribut a un nom et une valeur. La valeur de l'attribut type est en l'occurrence "text".

Toute page ou document HTML « bien formé » contient *a minima* la code figurant ci-dessous.

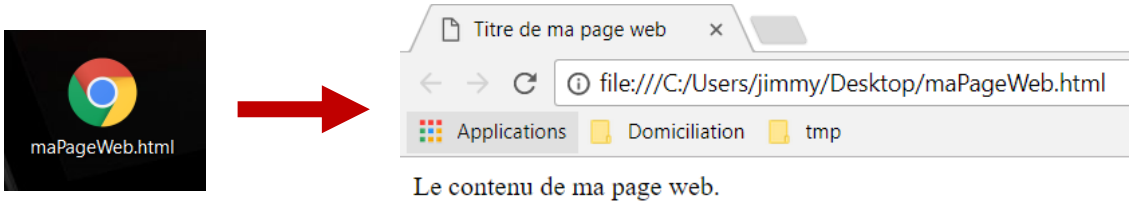
Structure minimale d'une page web	
<!DOCTYPE html> <html> <!-- en-têtes de la page HTML --> <head> <title>Titre de ma page web</title> </head> <!-- contenus de la page HTML --> <body> Le contenu de ma page web. </body> </html>	**DOCTYPE :** permet de préciser au navigateur la version HTML utilisée, en l'occurrence HTML5. **HTML :** permet de préciser le début et la fin de la page ; **HEAD :** marque le début et la fin des en-têtes de la page. Les en-têtes permettent de préciser le titre de la page, de définir des métadonnées ou encore d'importer des ressources telles que feuilles de styles CSS ou script JavaScript ; **BODY :** marque le début et la fin du corps de la page web, à savoir du contenu visible de la page web : textes, images, vidéos, etc.

Afin de prévenir quelques problèmes d'encodage de caractères, on pensera à ajouter les lignes suivantes dans l'en-tête de la page HTML, à savoir entre les balises <head> et </head> :

```
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Ces deux lignes permettent d'indiquer à l'ordinateur le jeu de caractères (charset en anglais) utilisé, c'est-à-dire la manière dont sont stockées en binaire les caractères (ici, en UTF-8).

Pour l'instant, notre page web est disons assez vide…



Ci-dessus, on constatera qu'un des rôles du navigateur est de produire l'affichage correspondant au contenu HTML d'une page web.

2.2. L'utilisation de ressources externes

Une page web peut signifier qu'elle a besoin de ressources pour fonctionner pleinement. On pourra typiquement ajouter les balises suivantes dans l'en-tête d'une page HTML :

- Ajout d'une feuille de styles CSS :

```
<link rel="stylesheet" href="./maMiseEnForme.css" />
```

La ligne qui précède permet de demander au navigateur de charger le fichier « maMiseEnForme.css » en tant que feuille de styles CSS. Le « ./ » permet de préciser que le fichier se trouve dans le même dossier que la page web.

- Ajout d'un script JavaScript :

```
<script type="text/javascript" src="./monProgramme.js"></script>
```

La ligne qui précède permet de demander au navigateur de charger le fichier « monProgramme.js » en tant que script JavaScript. Cela demande également au navigateur d'interpréter le programme JavaScript. En effet, un navigateur sait exécuter un programme rédigé à l'aide du langage de programmation JavaScript. Nous ne l'étudierons pas ici. Sachez qu'il est très utilisé et qu'il est souvent utilisé en outre grâce à « l'outil » (framework) appelé jQuery. En d'autres termes, un navigateur fait également office de « compilateur », plus exactement d'interpréteur, ce qui en fait déjà un programme très complexe.

2.3. L'éditeur Notepad++

Pour faciliter le développement de vos pages web, il vous est vivement recommandé d'utiliser *a minima* ce qu'on appelle un colorateur syntaxique. Un tel outil colore vos codes sources ce qui facilite grandement la lecture et l'écriture de vos codes HTML, CSS, PHP, etc.

Nous utiliserons le logiciel gratuit « Notepad++ ». Pensez à le télécharger sur vos ordinateurs personnels !

2.4. La structurer d'un projet web

En terminale, lorsque vous codez, pensez à vous organiser. A cet égard, si vous avez un projet web, un ensemble d'exercices ou de tests à réaliser, je vous recommande vivement de créer un dossier dans lequel conserver tous les éléments. Je vous recommande de vous organiser comme suit :

- **monprojet** : répertoire contenant tous les fichiers de votre projet, de vos exercices ou de vos tests
  - **doc** : répertoire contenant vos documents connexes (maquettes, planning, etc.)
  - **html** : répertoire contenant vos pages HTML (ou PHP)
  - **css** : répertoire contenant vos feuilles de styles CSS
  - **js** : répertoire contenant vos programmes JavaScript (si vous en avez)
  - **img** : répertoire contenant vos images

3. Utiliser une feuille de styles CSS

Si le HTML est le langage permettant de décrire le contenu d'une page web, le CSS est le langage permettant quant à lui de décrire la mise en forme des contenus HTML. Une fois encore, c'est le navigateur qui se charge de produire l'affichage correspondant, c'est-à-dire d'appliquer les règles de mises en forme. En effet, l'un des rôles principaux du navigateur est celui de moteur de rendu. Autrement dit, un navigateur est en outre une « usine à affichage » de pages web. En cela, un navigateur est un programme particulièrement complexe.

Nous avons vu précédemment (*voir 2.2. L'utilisation de ressources externes*) comment ajouter une feuille de styles CSS. Il convient désormais d'en faire quelque chose… Tout d'abord, il faut comprendre la notion de sélecteur CSS.

Important ! Sous Word, lorsque vous voulez mettre en forme quelque chose, vous sélectionnez la ou les zones à mettre en forme puis vous appliquez la mise en forme désirée. C'est le même principe en CSS ! On sélectionne la ou les zones à mettre en forme puis on précise la mise en forme à appliquer à cette ou ces zones.

Donnons-nous tout à la fois une page HTML un peu plus consistante que la précédente et une feuille de styles dans laquelle nous introduisons un peu de mise en forme :

Contenu HTML de « exemple-1.html »

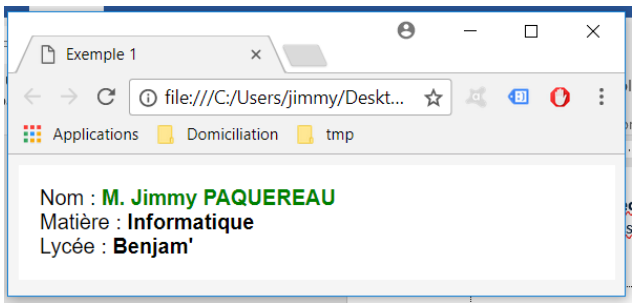
```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemple 1</title>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="../css/exemple-1.css" />
    </head>
    <body>
        <div id="encart">
            <div>
                <span class="libelle">Nom :</span>
                <span class="valeur vert">Jimmy PAQUEREAU</span>
            </div>
            <div>
                <span class="libelle">Matière :</span>
                <span class="valeur">Informatique</span>
            </div>
            <div>
                <span class="libelle">Lycée :</span>
                <span class="valeur">Benjam'</span>
            </div>
        </div>
    </body>
</html>
```

Styles CSS de « exemple-1.css »

```
body{
    background: #f5f5f5;
    font-family: arial;
}

#encart{
    background: #ffffff;
    padding: 1em;
}

.valeur{
    font-weight: bold;
}

.vert{
    color: rgb(0,128,0);
}
```

Quelques constats :

- On constate La présence de nouvelles balises : « div » et « span ». Il s'agit de balises structurantes. Elles permettent de structurer une page, c'est-à-dire de définir des zones.
- On constate également la présence de deux nouveaux attributs, qu'on peut appliquer à presque toutes les balises : « id » et « class ».



L'attribut « id » permet d'identifier (normalement de façon unique) un élément* d'une page HTML. L'attribut « class » permet d'identifier des éléments* ayant un « comportement » similaire, typiquement ayant une mise en forme commune. Un élément peut avoir plusieurs classes : "valeur vert".

** En HTML, on utilise les termes « balises », « marqueurs », « nœuds » ou encore « éléments ». Ces mots sont synonymes.*

Quant à la feuille de styles CSS, on observe qu'elle est constituée de blocs :

- Un bloc est précédé par un sélecteur CSS qui permet de sélectionner l'élément à mettre en forme ;
- Un bloc est délimité par une accolade ouvrante et une accolade fermante, à savoir { et } ;
- Un bloc contient la liste des règles de mises en forme à appliquer à l'élément sélectionné. Une règle est constituée d'une propriété (exemple : « background », arrière-plan en français) et d'une valeur (exemple : « #f5f5f5 » est un code couleur) séparées par un caractère « : ». Il convient de ne pas oublier le caractère « ; » (point-virgule) à la fin de la règle de mise en forme.

On remarque ainsi qu'on peut sélectionner un ou plusieurs éléments HTML :

- En fonction de leur marqueur, c'est-à-dire en fonction du nom de sa balise.
  Exemple : body{ … }
- En fonction de leur id, c'est-à-dire en fonction de la valeur de son attribut id.
  Exemple : #encart{ … }
- En fonction de leur(s) classe(s) de styles, c'est-à-dire en fonction de son attribut class.
  Exemple : .valeur{ … }

En somme, les sélecteurs CSS les plus simples sont les suivants :

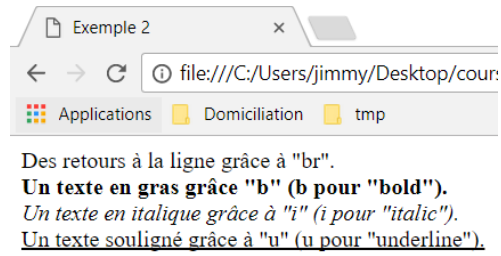
Sélecteur CSS	Description
nomElement{ … } *Exemple : body{ … }*	Permet de sélectionner et d'appliquer des mises en forme à tous les éléments dont le nom de balise est « nomElement ».
#idElement{ … } *Exemple : #encart{ … }*	Permet de sélectionner et d'appliquer des mises en forme à l'élément ayant l'id « idElement ».
.classeElement{ … } *Exemple : .valeur{ … }*	Permet de sélectionner et d'appliquer des mises en forme à tous les éléments ayant la classe de styles « classeElement ».
selecteur sousSelecteur{ … } Exemple : #encart span{ … }	Permet de sélectionner et d'appliquer des mises en forme à tous les éléments « sousSelecteur » se trouvant en-dessous du ou des éléments « selecteur ». *Dans l'exemple (à gauche), on sélectionne tous les « span » situés dans « #encart ».*
selecteur:hover{ … } Exemple : .vert:hover{ … }	Permet de sélectionner et d'appliquer des mises en forme au survol de tout élément « selecteur ». *L'exemple (à gauche) permet d'appliquer une mise en forme au survol de l'élément dont la classe est « vert ».*

4. Créer et mettre en forme des textes

4.1. Quelques balises HTML

Voici quelques balises bien utiles qui permettent directement de mettre en forme du texte :

```
Des retours à la ligne grâce "br".
<br/>
<b>Un texte en gras grâce à "b" (b pour "bold").</b>
<br/>
<i>Un texte en italique grâce à "i" (i pour "italic").</i>
<br/>
<u>Un texte souligné grâce à "u" (u pour "underline").</u>
```



En voici d'autres permettant de créer proprement des sous-titres sur plusieurs niveaux. Bien entendu, vous pouvez en modifier la mise en forme au moyen de CSS :

```
<h1>Un sous-titre de niveau 1</h1>
<h2>Un sous-titre de niveau 2</h2>
<h3>Un sous-titre de niveau 3</h3>
<h4>Un sous-titre de niveau 4</h4>
<h5>Un sous-titre de niveau 5</h5>
<h6>Un sous-titre de niveau 6</h6>
```

4.2. Quelques propriétés CSS

Voici quelques propriétés CSS usuelles permettant entre autres choses de mettre en forme des zones contenant du texte :

Propriété CSS	Description
font-size : 16px ; font-size : 1.5em ;	Permet de préciser la taille du texte en pixels ou en « points ». Il est préférable d'utiliser une taille exprimée en « points ».
font-family : Arial, Verdana ;	Permet de préciser la police d'écriture à utiliser (par ordre de préférences). Pour que la police d'écriture « fonctionne », il faut typiquement que celle-ci soit installée sur l'ordinateur.
color : #FCA800 ; color : rgb(252, 168, 0) ; color : rgba(252, 168, 0, 0.5) ;	Permet de préciser la couleur du texte. On peut utiliser différents formats de codes couleurs.
font-weight : bold ;	Permet de préciser que le texte doit être affiché « en gras ».

font-style : italic ;	Permet de préciser que le texte doit être affiché « en italique ».
text-decoration : underline ;	Permet de préciser que le texte doit être affiché « en souligné ».
text-align : left ;	Permet d'aligner le texte. Les valeurs possibles de la propriété sont : left, right, center, justify.

Une petite explication à titre indicatif concernant les codes couleurs :

- **#FCA800** : un tel code couleur est dit « hexadécimal » (en base 16). Il est la forme « **RRVVBB** ». Les deux premiers caractères représentent la quantité de rouge, les deux suivants la quantité de vert et les deux derniers la quantité de bleu.
  Si on note les nombres de 0 à 16 comme ceci : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10), B (11), C (12), D (13), E (14) et F (15). Alors :
  - **FC** vaut $C \times 1 + F \times 16 = 12 \times 1 + 15 \times 16 = 252$
  - **A8** vaut $8 \times 1 + A \times 16 = 8 \times 1 + 10 \times 16 = 168$
  - **00** vaut $0 \times 1 + 0 \times 16 = 0 \times 1 + 0 \times 16 = 0$
  - Finalement, le code couleur #FCA800 est équivalent à rgb(252, 168, 0).
- **rgb(252, 168, 0)** : un tel code couleur est écrit en « décimal » (en base 10) sous la forme **rgb(R, V, B)**, où chacun des nombres, compris entre 0 et 255, représente respectivement la quantité de rouge, de vert et de bleu. D'où les lettres r (*red*), g (*green*) et b (*blue*).
- **rgba(252, 168, 0, 0.5)** : un tel code couleur est tout à fait similaire au précédent. On a juste ajouté un quatrième nombre, compris entre 0 et 1. Ce nombre représente la transparence, ou plus exactement l'opacité. C'est ce qu'on appelle le canal « alpha », d'où le « a ». Si on indique 1, le fond est 100% opaque. Si on indique 0, le fond est 0% opaque, c'est-à-dire complètement transparent. On peut faire varier l'opacité. En l'occurrence, l'opacité a été fixée à 0.5, c'est-à-dire 50%.

Voici un exemple faisant intervenir certaines des propriétés ci-avant décrites :

Extrait du contenu HTML de « exemple-4.html »

```
…
<body>
    <h1>Le célèbre Lorem Ipsum</h1>
    <div class="texte">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Pellentesque malesuada ligula odio, dictum dignissim
        magna volutpat rutrum. Etiam eu varius leo. Ut a cursus
        ex. Nulla rutrum at libero non fringilla. Phasellus
        ultrices accumsan diam, vitae dignissim enim sodales
        vitae. Sed varius, nulla quis pharetra egestas, mauris
        mi tempor enim, et vehicula sapien purus et quam.
        Suspendisse tincidunt tincidunt metus, in lobortis metus
        malesuada a.
    </div>
</body>
…
```

Styles CSS de « exemple-4.css »

```
body{
    font-family: Rockwell ;
    background: rgb(240,255,240) ;
}

h1{
    text-align: center ;
    color: rgb(0,128,0) ;
}

.texte{
    text-align: justify ;
    color: rgb(0,64,0) ;
}
```

5. Créer et mettre en forme des zones

5.1. Arrière-plan, bordures, dimensionnement et centrage

Propriété CSS	Description
background : #ff0000 ; background : rgb(255,0,0) ; background : rgba(255,0,0,0.5) ; background : url(../img/fond.jpg) no-repeat center center / cover;	Permet de définir un arrière-plan à partir d'un code couleur ou d'une image de fond. Dans le cas d'une image de fond : • url(…) permet de préciser le chemin permettant d'accéder à l'image (fichier) image servant d'arrière-plan ; • « no-repeat » permet d'éviter la répétition du motif ; • « center center » permet de centrer l'image ; • « cover » permet de faire de sorte que l'image couvre tout l'élément (une autre valeur possible est « contain »).
border : solid 1px #ff0000 ; border-top : solid 1px #ff0000 ; border-left : solid 1px #ff0000 ; border-bottom : solid 1px #ff0000 ; border-right : solid 1px #ff0000 ;	Permet de définir une bordure autour d'un élément : • « solid » représente ici le type de bordure. D'autres valeurs sont possibles : dotted, dashed, etc. • « 1px » représente ici l'épaisseur de la bordure en pixels. • « #ff0000 » représente ici la couleur de la bordure. Il est possible de ne définir qu'une bordure « en haut » (*top*), en bas (*bottom*), à gauche (*left*) ou à droite (*right*).
height : 250px ; width : 400px ;	Permet qu'un élément ait une taille fixe exprimée en pixels (px) ou en points (em). On précise la hauteur (*height*) et/ou la largeur (*width*) de l'élément à dimensionner.
height : 100% ; width : 75% ;	Permet de faire en sorte qu'un élément est une taille relative, à savoir une taille exprimée en pourcentages de l'élément parent, c'est-à-dire pourcentage de l'élément « au-dessus ».
margin : auto ;	Permet de centrer un élément horizontalement.

Un exemple pour illustrer ces propriétés CSS :

	Code	Résultat
Extrait du contenu HTML	… <body> Que la force du codeur soit avec toi ! </body> …	

Style CSS	.zone{
	background: url(../img/fond.jpg)
	no-repeat center center / cover ;
	height: 20em ; width: 20em ;
	color: rgb(255,255,255) ;
	margin: auto ;
	}
	.texte{
	background: rgb(0,0,0) ;
	text-align: center ;
	padding: 1em ;
	}	

Remarque ! Une fois encore, les balises « div » comme les balises « span » s'avèrent très utiles afin de structurer le contenu d'une page web.

5.2. Marges internes et externes

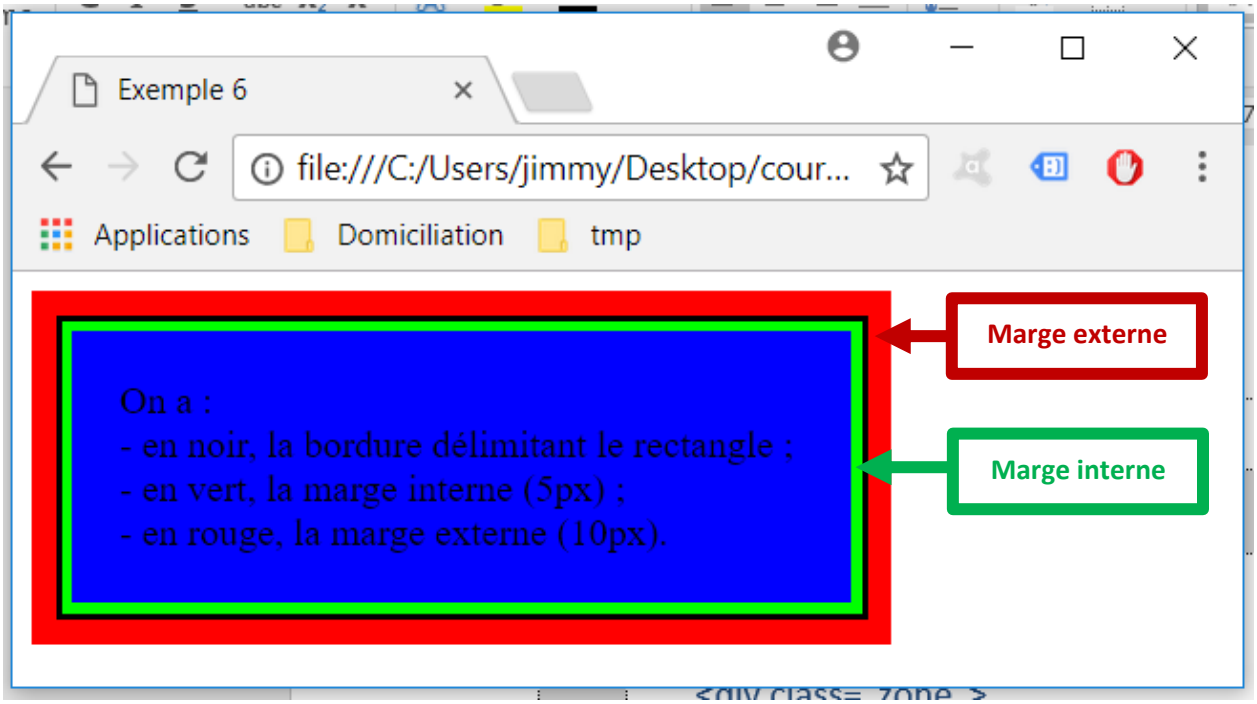
Les notions de marges internes et externes sont également assez essentielles afin de pouvoir mettre en forme convenablement des contenus HTML. Les propriétés à utiliser sont les suivantes :

Propriété CSS	Description
margin : 5px ; margin-top : 10px ; margin-left : 10px; …	Permet de préciser les marges externes de l'élément. Les marges externes peuvent être exprimées en pixels (px), points (em) ou pourcentages (%). On pourra aussi préciser individuellement chacune des marges externes de l'élément.
padding : 5px ; padding-top : 10px ; padding -left : 10px; …	Permet de préciser les marges internes de l'élément. Les marges internes peuvent être exprimées en pixels (px), points (em) ou pourcentages (%). On pourra aussi préciser individuellement chacune des marges internes de l'élément.

Illustration des notions de marges internes et externes :

Extrait du contenu HTML de « exemple-6.html »	**Styles CSS de « exemple-6.css »**

…
<body>
 	
On a :
 	
- en noir, la bordure délimitant le rectangle ;
 	
- en vert, la marge interne (5px) ;
 	
- en rouge, la marge externe (10px).
 	
</body>
…
	.zone{
	display: inline-block ;
	background: rgb(255,0,0) ;
	}
	.rectangle{
	display: inline-block ;
	border: solid 2px rgb(0,0,0) ;
	background: rgb(0,255,0) ;
	margin: 10px ; padding: 5px ;
	}
	.texte{
	background: #0000ff ;
	padding: 20px ;
	}



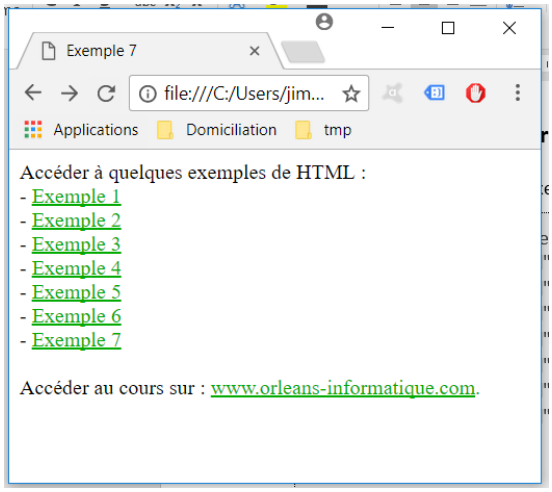
Important ! Le code CSS est interprété par le navigateur de manière séquentielle… Qu'est-ce que cela veut dire ? Les règles de mise en forme sont appliquées dont l'ordre où vous les avez écrites. Autrement dit, si une règle de mise en forme vient contredire une règle qui a été écrite avant, c'est la dernière règle de mise en forme écrite qui a raison !

6. Créer et mettre en forme des liens et des images

6.1. Liens hypertextes

Une fois de plus, quelques exemples vaudront mieux qu'un long discours. Observons…

```
Accéder à quelques exemples de HTML : <br/>
- <a href="./exemple-1.html">Exemple 1</a> <br/>
- <a href="./exemple-2.html">Exemple 2</a> <br/>
- <a href="./exemple-3.html">Exemple 3</a> <br/>
- <a href="./exemple-4.html">Exemple 4</a> <br/>
- <a href="./exemple-5.html">Exemple 5</a> <br/>
- <a href="./exemple-6.html">Exemple 6</a> <br/>
- <a href="./exemple-7.html">Exemple 7</a> <br/>
<br/>
Accéder au cours sur :
<a
    href="https://www.orleans-informatique.com/"
    target="_blank"
>www.orleans-informatique.com<a>.
```



Au regard de cet exemple, on constate que :

- Le marqueur « a » permet d'ajouter un lien hypertexte vers une ressource web interne ou externe ;
- Le contenu du marqueur « a » correspond au lien tel qu'il est affiché ;
- L'attribut href permet de préciser le chemin vers la ressource web à ouvrir lorsque l'on clique sur le lien hypertexte ;
- L'attribut target, lorsqu'il vaut "_blank", permet d'ouvrir la ressource web dans un nouvel onglet.

On a par ailleurs utilisé un soupçon de CSS :

```
a{ color: #00AA00 ; }
a:hover{ color: #00EE00 ; }
```

Le sélecteur CSS « a:hover » permet de préciser la mise en forme au survol des liens hypertextes.

6.2. Images

Extrait du contenu HTML de « exemple-8.html »	**Styles CSS de « exemple-8.css »**

…
<body>

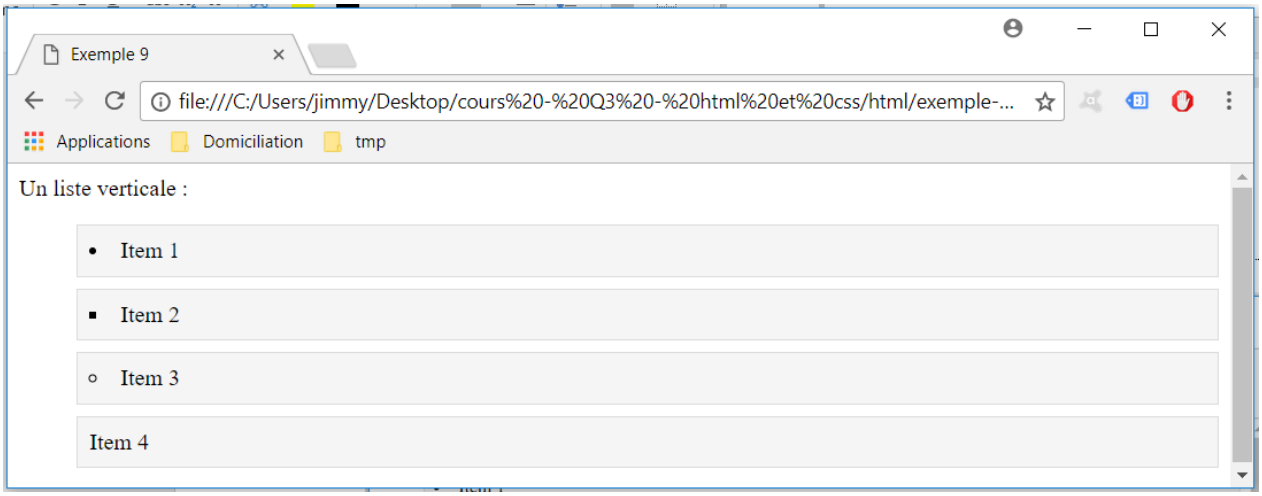
</body>
…
	#pika-slim{
	  height: 15em; width: 10em;
	}
	#pika-chu{
	  height: 15em;
	}
	#pika-fat{
	  height: 15em; width: 30em;
	}

7. Créer une liste

7.1. Créer une liste verticale

Extrait du contenu HTML de « exemple-9.html »	Styles CSS de « exemple-9.css »

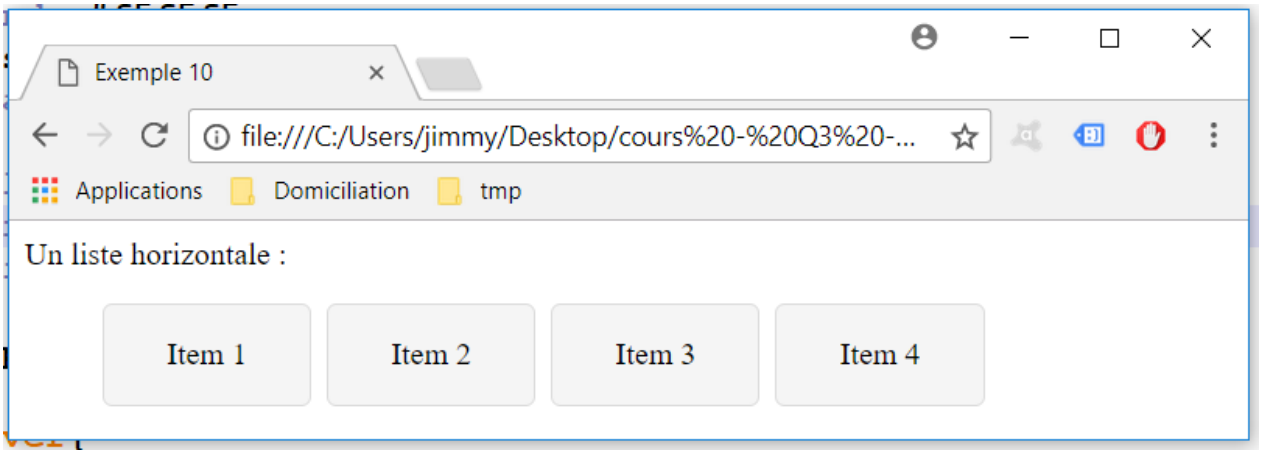
…
Un liste verticale :
<ul class="liste">
<li class="disc">Item 1
<li class="square">Item 2
<li class="circle">Item 3
<li class="none">Item 4
 	
…
	.liste li{
	border: solid 1px #dcdcdc ;
	background: #f5f5f5 ;
	padding: .5em ;
	margin-bottom: .5em ;
	}
	.liste .disc{ list-style: disc inside ; }
	.liste .circle{ list-style: circle inside ; }
	.liste .square{ list-style: square inside ; }
	.liste .none{ list-style: none ; }



7.2. Créer une liste horizontale

Extrait du contenu HTML de « exemple-10.html »	Styles CSS de « exemple-10.css »

…
<body>
  Un liste horizontale :
  <ul class="liste">
   - Item 1
   - Item 2
   - Item 3
   - Item 4
 	
</body>
…
	.liste li{
	  background: #f5f5f5; border: solid 1px #dcdcdc;
	  border-radius: 5px; padding: 1em;
	  padding-left: 2em; padding-right: 2em;
	  margin-right: .25em;
	  display: inline-block;
	  cursor: pointer;
	}
	.liste li:hover{
	  background: #ffcc00; border: solid 1px #dcba00;
	}



8. Créer un tableau

Un tableau permet de construire des mises en forme évoluées. Cependant, pour créer un tableau, il va falloir un certain nombre de balises, typiquement les suivantes :

- Les balises <table> et </table> : elles permettent d'insérer un tableau ;
- Les balises <thead> et </thead> (pour « table header ») : au sein d'un marqueur « table », elles permettent d'insérer l'en-tête du tableau ;
- Les balises <tbody> et </tbody> (pour « table body ») : au sein d'un marqueur « table », elles permettent d'insérer le corps du tableau ;
- Les balises <tr> et </tr> (pour « table row ») : au sein d'un marqueur « thead » ou « tbody », elles permettent d'insérer une ligne ;
- Les balises <td> et </td> : au sein d'un marqueur « tr », elles permettent d'ajouter une cellule (une colonne) à une ligne ;
- Les balises <th> et </th> : au sein d'un marqueur « tr », elles permettent d'ajouter une cellule (une colonne) d'en-tête.

Voici un exemple de tableau :

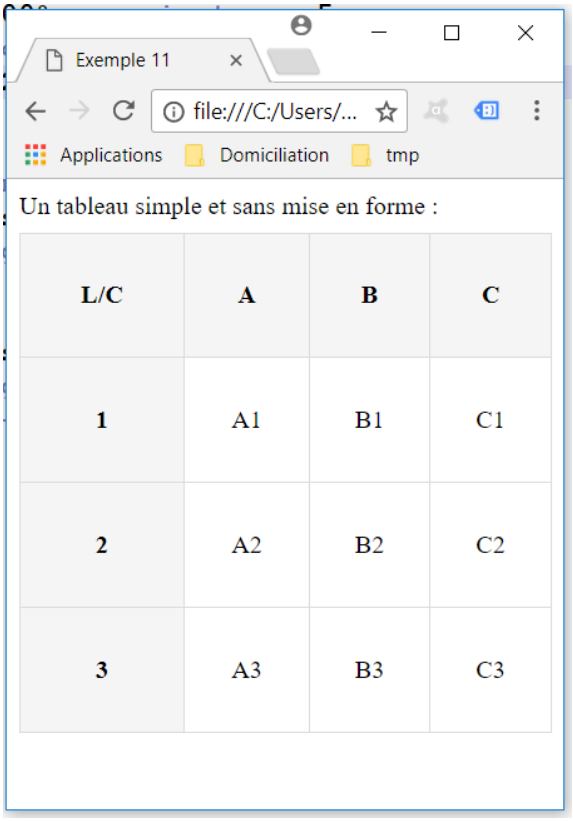
Extrait du contenu HTML de « exemple-11.html »

```
…
Un tableau simple et sans mise en forme :
<body>
    <table>
        <tbody>
            <tr>
                <th>L/C</th>
                <th>A</th>
                <th>B</th>
                <th>C</th>
            </tr>
            <tr>
                <th>1</th>
                <td>A1</td>
                <td>B1</td>
                <td>C1</td>
            </tr>
            <tr>
                <th>2</th>
                <td>A2</td>
                <td>B2</td>
                <td>C2</td>
            </tr>
            <tr>
                <th>3</th>
                <td>A3</td>
                <td>B3</td>
                <td>C3</td>
            </tr>
        </tbody>
    </table>
</body>
…
```

Styles CSS de « exemple-10.css »

```
table{
    width: 100%; margin-top: .5em;
    border-collapse: collapse; height: 20em;
}
th{
    background: #f5f5f5;
    border: solid 1px #dcdcdc; font-weight: bold;
}
td{
    border: solid 1px #dcdcdc;
    text-align: center; vertical-align: middle;
}
```

Résultat



Quelques remarques concernant l'exemple ci-dessus :

- On a autant de ligne que de balises « tr », 4 en l'occurrence ;
- Pour chaque ligne, on a autant de balises « td » et « th » que de colonnes, 4 en l'occurrence ;
- On a utilisé des balises « th » pour pouvoir bien distinguer les cellules d'en-tête des autres ;
- La règle CSS « vertical-align : middle ; » permet de centrer les textes verticalement.

De l'exemple qui précède, on tire quelques propriétés CSS remarquables :

Propriété CSS	Description
vertical-align : top ; vertical-align : middle ; vertical-align : bottom ;	Permet le centrage vertical du contenu des cellules « td » ou « th » d'un tableau. On peut également centrer verticalement les éléments ayant la propriété CSS « display: table-cell ; » s'ils sont inclus dans un élément ayant la propriété CSS « display : table ; ». Un exemple est fourni ci-après.
border-collapse : separate ; border-collapse : collapse ;	Permet de préciser si les cellules d'un tableau doivent être « espacées » (*separate*) ou, au contraire, si toutes les bordures doivent être fusionnées/côte-à-côte (*collapse*).
border-spacing : 10px ; border-spacing : 5px 10px ;	Permet de préciser l'espacement entre cellules d'un tableau. Cette propriété fonctionne si les cellules sont « *separate* ».

Remarque ! A titre d'information, les attributs « rowspan » (exemple : rowspan="3") et « colspan » (exemple : colspan="3") permettent de fusionner des cellules. Plus exactement, ils permettent de préciser qu'une cellule s'étale respectivement sur plusieurs ligne (*rowspan*) et/ou plusieurs colonnes (*colspan*).

Par ailleurs, grâce à la propriété CSS display et à ses valeurs table et table-cell, il est possible de créer un tableau sans utiliser les balises ci-avant décrites. Pour quoi faire me direz-vous ? Pour plusieurs raisons, en voici au moins deux :

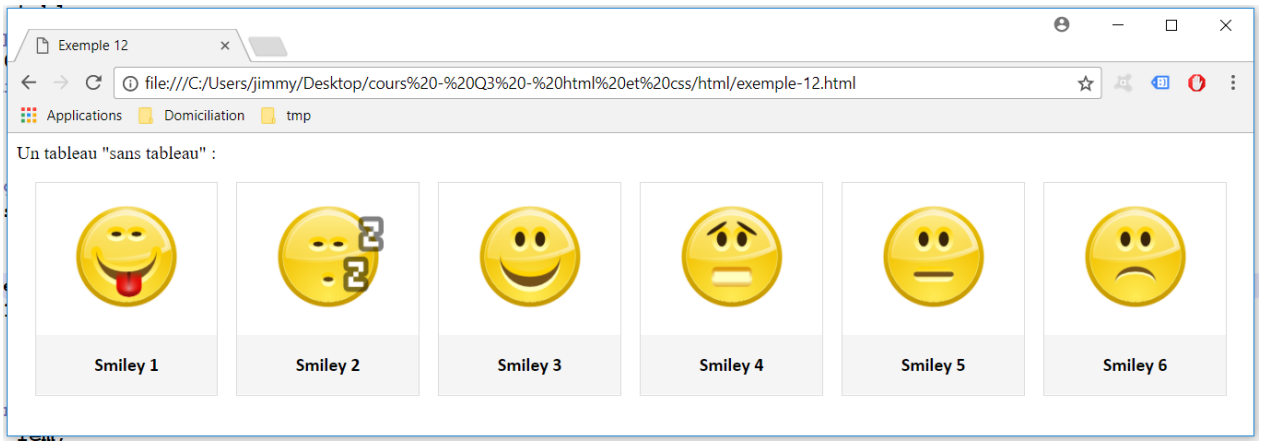
- Les cellules de tableau offrent un moyen simple d'effectuer un centrage vertical ;
- Les cellules de tableau peuvent être parfaitement réparties sur une ligne. Cela permet de faire des « jolies » listes horizontales bien équilibrées.

Extrait du contenu HTML de « exemple-12.html »

```
...
Un tableau "sans tableau" :
<div class="liste">
    <div class="item">
        <img src="../img/smiley-1.png" />
        <span>Smiley 1</span>
    </div>
    <div class="item">
        <img src="../img/smiley-2.png" />
        <span>Smiley 2</span>
    </div>
    ...
    <div class="item">
        <img src="../img/smiley-6.png" />
        <span>Smiley 6</span>
    </div>
</div>
...
```

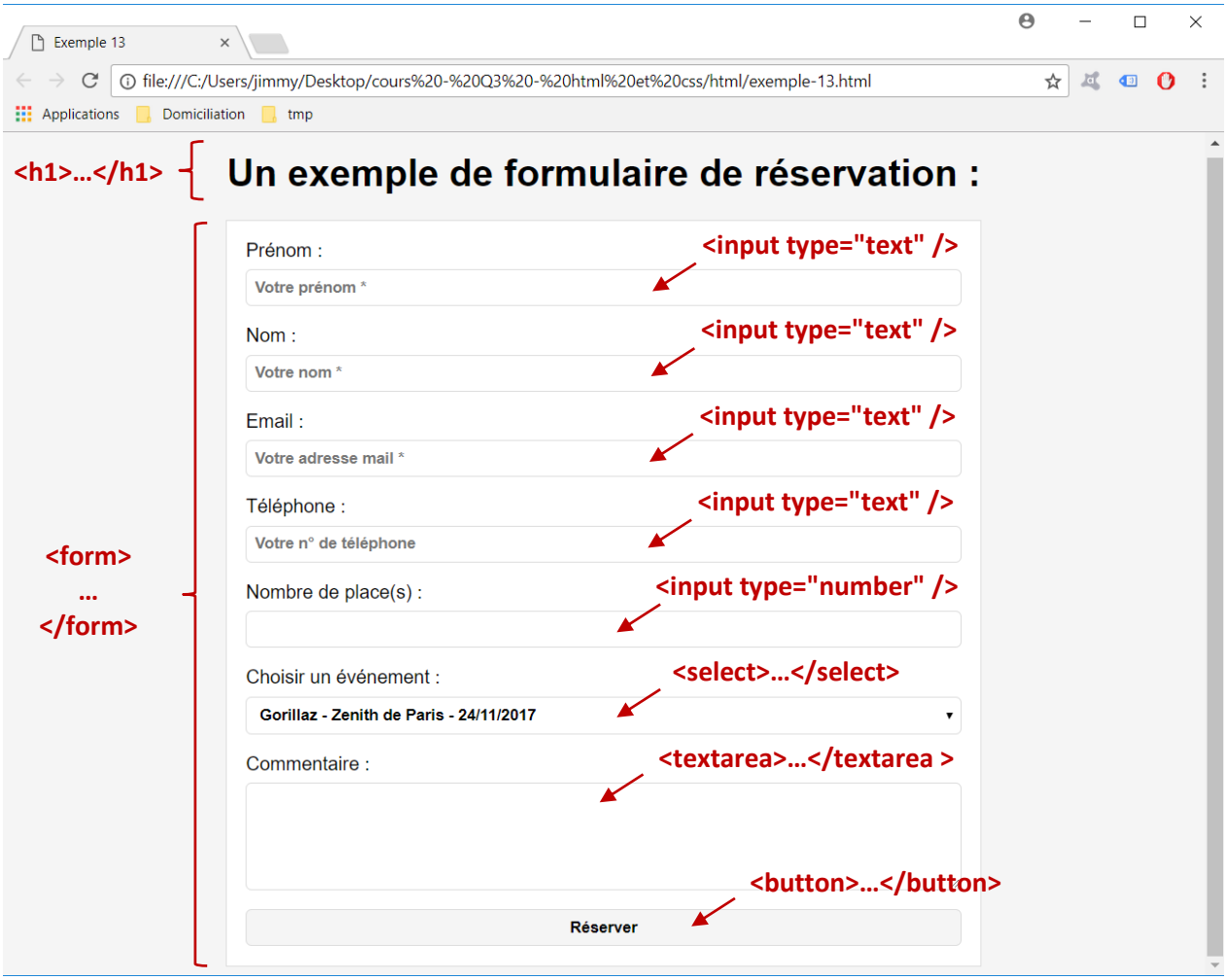
Styles CSS de « exemple-12.css »

```
.liste{
    display: table; border-spacing: 1em;
    width: 100%; font-family: Calibri;
}
.item{
    display: table-cell; text-align: center;
    border: solid 1px #dcdcdc;
}
img{
    width: 6em; margin: 1em;
}
span{
    display: block; background: #f5f5f5;
    padding: 1em; font-weight: bold;
}
```



9. Créer un formulaire

Afin de conclure cette introduction aux langages HTML et CSS, nous abordons une dernière possibilité offerte par le langage HTML : la création de formulaires.



Extrait du contenu HTML de « exemple-13.html »

```
...
<h1>Un exemple de formulaire de réservation :</h1>
<form method="POST" action="./exemple-13.html">
    <label>
        Prénom :
        <input name="prenom" type="text"
            placeholder="Votre prénom *" />
    </label>
    <label>
        Nom :
        <input name="nom" type="text" placeholder="Votre nom *" />
    </label>
    <label>
        Email :
        <input name="email" type="text"
            placeholder="Votre adresse mail *" />
    </label>
    <label>
        Téléphone :
        <input name="telephone" type="text"
            placeholder="Votre n° de téléphone" />
    </label>
    <label>
        Nombre de place(s) :
        <input name="places" type="number" min="1" />
    </label>
    <label>
        Choisir un événement :
        <select name="evenement">
            <option value="EVT325">
                Gorillaz - Zenith de Paris - 24/11/2017
            </option>
            <option value="EVT327">
                Alonzo - Zenith de Paris - 26/11/2017
            </option>
            <option value="EVT328">
                Luis Fonsi - Zenith de Paris - 28/11/2017
            </option>
            <option value="EVT329">
                Benjamin Biolay - Zenith de Paris - 30/11/2017
            </option>
        </select>
    </label>
    <label>
        Commentaire :
        <textarea name="commentaire" rows="5"></textarea>
    </label>
    <button>Réserver</button>
</form>
...
```

Styles CSS de « exemple-13.css »

```
body{
    font-family: Arial ;
    background: #f5f5f5 ;
}
h1{
    text-align: center ;
}
*{
    box-sizing: border-box ;
}
form{
    padding: 1em ;
    background: #ffffff ;
    border: solid 1px #dcdcdc ;
    width: 40em ;
    margin: auto ;
}
label{
    display: block ;
    margin-bottom: 1em ;
    box-sizing: border-box ;
}
input, textarea, select{
    display: block ;
    margin-top: .5em ;
    padding: .5em ;
    border-radius: 5px ;
    width: 100% ;
    border: solid 1px #dcdcdc ;
    font-weight: bold ;
}
button{
    border-radius: 5px ;
    width: 100% ;
    padding: .5em ;
    background: #f5f5f5 ;
    border: solid 1px #dcdcdc ;
    font-weight: bold ;
}
```

Quelques explications s'imposent :

- La balise « form » permet d'insérer un formulaire. Elle délimite l'intégralité du formulaire. Elle a en outre deux attributs très utiles : « method » et « action ». De manière simpliste pour le moment, disons que l'attribut « action » permet d'indiquer la page/ressource web qui doit traiter le formulaire une fois que l'utilisateur l'a soumis. L'attribut « method » permet quant à lui de préciser la manière dont sont transmises les données que l'utilisateur à saisi lorsqu'il soumet le formulaire.
- La balise « button » permet d'insérer un bouton. Par défaut, un tel bouton, placé au sein d'un formulaire, permet à l'utilisateur de soumettre le formulaire. Il existe en fait 2 façons de créer un bouton de soumission de formulaire :
  - `<button>Réserver</button>`
  - `<input type="submit" value="Réserver" />`
- Les balises « input », « select » et « textarea » permettent de créer des champs de saisie :
  - « input » : permet de créer un champ. Il existe plusieurs types de champs (exemples : "text", "number", "date", "time", "password" voire "file"). Le type de champ est précisé grâce à l'attribut nommé « type ».
  - « select » : permet de créer une liste ou une liste déroulante. On y place des balises « option » afin de lister les éléments de la liste.
  - « textarea » : permet de créer une zone de texte.
- Les champs définis au moyen des balises « input », « select » ou encore « textarea » doivent être nommés en utilisant l'attribut « name ». Autant que faire se peut, il est préférable d'utiliser un nom pertinent. Nous verrons ultérieurement l'utilité de cet attribut.

Vous pouvez désormais envisager de créer des formulaires ! Malheureusement, à ce stade, vous ne pouvez aucunement traiter ce formulaire, à savoir que vous ne pouvez rien en faire… Au-delà du HTML et du CSS, nous étudierons ultérieurement la programmation PHP, laquelle nous permettra entre autres choses d'interagir avec l'utilisateur par l'intermédiaire de formulaires.

10. Limites

En conclusion, nous n'avons pour l'instant étudié que la partie visible des pages web : le contenu et la mise en forme. Les langages HTML et CSS ne nous permettent pas d'interagir avec l'utilisateur. Ils nous permettent uniquement de lui présenter un contenu de même que nous pourrions lui proposer un simple document Word. Autrement dit, nous avons fait abstraction de la partie « programmation » à proprement parler. La programmation PHP nous permettra de rendre nos pages web interactive et ainsi de pallier cet inconvénient. Elle nous permettra d'ailleurs bien plus encore !

Egalement, nos pages web ont été développées localement et nous sommes dans l'incapacité de les partager et/ou de les diffuser… C'est pourtant le propre des sites et applications web : le partage et la diffusion d'informations ! En d'autres termes, nous avons également fait abstraction des notions de « réseaux ». Nous constaterons prochainement que la notion « d'architecture client-serveur » permet de prévenir cette seconde lacune.