

# HTML5

---



---

# HTML Tutorial

## INTRODUCTION

# Introduction

---

- Ce cours a pour but d'explorer les différentes fonctionnalités du HTML et du CSS
- C'est un cours de premier niveau.
- Le HTML et le CSS sont deux véritables standards et n'ont donc, à ce titre, pas de concurrent comme cela peut être le cas pour le langage PHP par exemple
- Ils sont à la base de tout projet de développement web. Qu'il s'agisse de créer un site e-commerce, un blog, une application mobile ou quoique ce soit d'autre, on est obligé de passer par les langages HTML et CSS.

# Quelques a priori dangereux

---

Pas besoin de connaître HTML et CSS, car on a à sa disposition:

- 1) Des frameworks (Drupal, Wordpress), qui font tout à notre place
- 2) Des éditeurs WYSIWIG (Dreamweaver), où il suffit de dessiner pour que le code soit généré automatiquement
- 3) Des agences spécialisées (utilisant souvent des freelances) à qui l'on peut simplement passer commande (pour quelques multiples de dizaines de Keuros, cependant ...)

Des idées bien dangereuses ...

# HTML, ou HyperText Markup Language

---

- **HTML est l'abréviation de HyperText Markup Language**, soit en français « langage de balisage hypertexte ».
- Ce langage a été créé en 1991 et a pour fonction de structurer et de donner du sens à du contenu.
- Il existe de très nombreuses et souvent excellentes introductions pour comprendre le rôle d'HTML et de ses divers composants associés (CSS, JAVASCRIPT) mais ce qui est essentiel, c'est de comprendre avant tout qu'il s'agit d'un langage d'affichage de page indissolublement lié à l'architecture client-serveur que le Web a adopté autour du protocole HTTP.

# HTML, ou HyperText Markup Language

---

- Grâce au HTML, on va par exemple pouvoir indiquer au navigateur que tel texte doit être considéré comme un simple paragraphe ou que tel autre est un titre.
- Le code HTML ci-dessous par exemple indique que l'on souhaite créer un grand titre (grâce à l'élément h1) et un paragraphe (grâce à l'élément p).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mon premier code HTML !</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Ceci est un gros titre</h1>
    <p>Ceci est un paragraphe</p>
  </body>
</html>
```

- Le HTML va permettre d'insérer différents types d'éléments dans nos pages web : du texte, des liens, des images, etc.

# Le CSS, ou Cascading StyleSheets

---

- **CSS est le diminutif de Cascading StyleSheets**, ou feuilles de styles en cascade.
- Le CSS a été créé en 1996 et a pour rôle de mettre en forme du contenu en lui appliquant ce qu'on appelle des styles.
- Le CSS va permettre par exemple de définir la taille, la couleur
- Le code ci-dessous par exemple indique que les titres h1 écrits en HTML doivent avoir une couleur rouge et une taille de 14px tandis que les paragraphes doivent être bleus et soulignés. ou l'alignement d'un texte.

```
h1{  
  color: red;  
  font-size: 14px;  
}  
  
p{  
  color: blue;  
  text-decoration: underline;  
}
```

# HTML et CSS

---

- Bien que l'on puisse remarquer que certaines balises HTML apporte une certaine dimension de mise en forme (titres h1 par ex..), ce n'est absolument pas sa vocation.
- Le langage HTML a une vocation avant tout sémantique (surtout depuis HTML5), il faut lui laisser la gestion du fond, tandis que le CSS s'occupe de la gestion de la forme.
- HTML est un langage qui a été créé pour structurer des pages et pour donner du sens au contenu.



# HTML et CSS

---

- Depuis sa naissance en 1991, le langage HTML a connu de nombreuses versions. Les plus notables sont la version HTML4 récente, et surtout la nouvelle donne apportée par la version HTML5 (plus qu'une évolution, presque une révolution)
- De même, la plus récente des versions du CSS, CSS3 est plus riche que les versions CSS1 et CSS2 qui l'ont précédée réunies.
- Les versions de référence de ce cours seront – sauf indication contraire - HTML5 et CSS3

# Edition de texte et Tests

---

- Pour programmer en HTML et CSS, il faut bien sûr disposer d'un éditeur de texte adapté (coloration syntaxique, etc ...).
- Parmi un choix aujourd'hui considérable, on distingue particulièrement:
  - Notepad ++ (windows)
  - Sublime Text (idem)
  - Visual Studio Code (nouvel outil de Microsoft)
  - Komodo Edit(tout OS)
- Et pour les tests, n'importe quel navigateur bien sûr mais aussi l'excellent **jsbin** ([jsbin.com](http://jsbin.com))

---

# **HTML Tutorial**

## **LES BASES EN HTML**

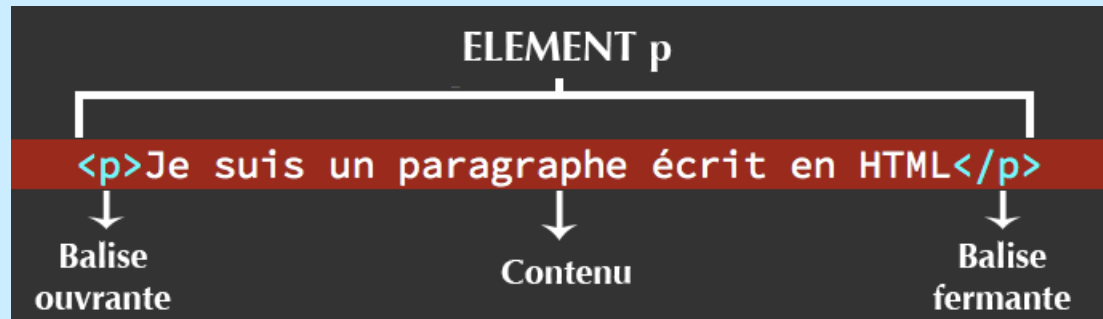
---

# HTML Tutorial

ELEMENTS, BALISES, ATTRIBUTS  
EN HTML

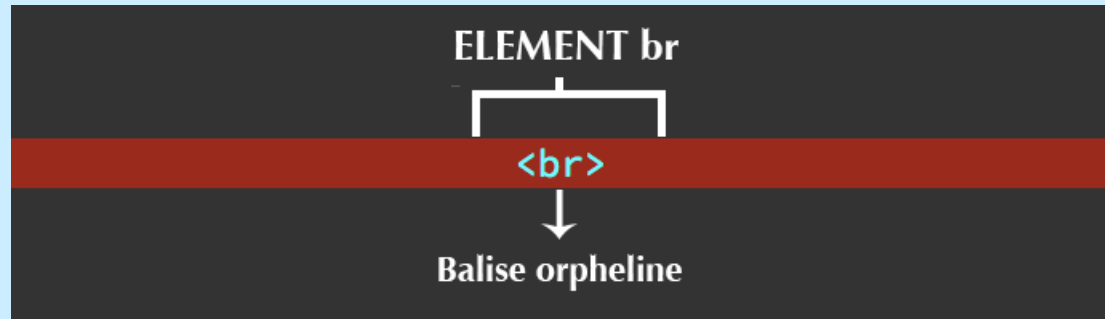
# Les balises en HTML

- Un élément HTML peut être soit constitué d'une **paire de balises et d'un contenu**, soit (plus rarement) d'une **balise unique qu'on dit alors orpheline**.
- L'élément p ci-dessous est constitué d'une balise ouvrante, d'une balise fermante (notez la présence du slash), et d'un contenu (textuel) entre les balises.



# Les balises en HTML

- L'élément br ci-dessous (servant à créer un retour à la ligne) n'est lui constitué que d'une balise orpheline.



- Sur le web, vous trouverez peut être des éléments br écrits avec un slash après le nom de l'élément, comme ceci : <br/>. Les deux syntaxes sont acceptées en HTML, la seule différence est que la syntaxe utilisant le slash est également reconnue par le langage XML.

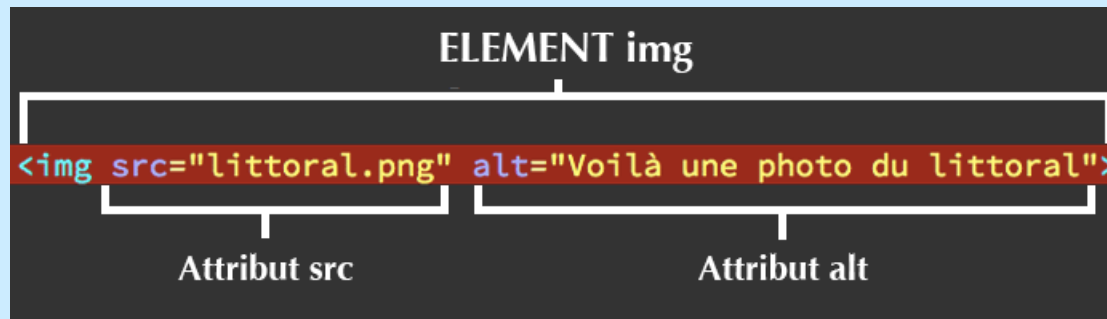
# Les balises en HTML

---

- Finalement, **la balise ouvrante d'un élément HTML peut contenir des attributs**, qui sont parfois même obligatoires.
- Les attributs vont venir compléter les éléments en les définissant plus précisément ou en leur apportant des informations supplémentaires.
- Un attribut contient toujours une valeur, qu'on peut cependant parfois omettre dans le cas des attributs ne possédant qu'une seule valeur (la valeur est considérée comme évidente).

# Les balises en HTML

- Par exemple, l'élément a (pour "anchor") servant à créer des liens vers d'autres sites ou d'autres pages, va avoir besoin d'un attribut href ("hypertexte reference") qui va prendre comme valeur l'adresse (relative ou absolue) de la page vers laquelle on souhaite faire un lien.
- L'élément img, servant à insérer une image dans une page HTML, va lui demander deux attributs : src et alt.





---

# HTML Tutorial

## STRUCTURE D'UNE PAGE HTML

# Structure minimale d'une page en HTML

---

- Pour qu'une **page HTML soit déclarée valide**, elle doit obligatoirement comporter certains éléments et suivre un schéma préétabli.
- Voici un code minimum pour créer une page HTML valide.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

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

# Le doctype

---

- Tout d'abord, on doit toujours démarrer une page HTML en précisant le doctype de notre document.
- Comme son nom l'indique, le doctype sert à préciser le type du document.
- La balise représentant le doctype commence par un point d'exclamation. Ceci est un cas unique.

# L'élément html

---

- Cet élément est composé de deux balises `<html>` et `</html>` et va représenter la page HTML en soi.

# Les éléments head et body

---

- A l'intérieur de l'élément html, on doit indiquer deux éléments : head et body.
- L'élément head va contenir des meta informations relatives à la page, c'est-à-dire des informations générales dont la page va avoir besoin pour fonctionner, comme le titre de la page ou encore le type d'encodage utilisé.
- L'élément body va lui contenir tout le contenu « visible » de la page : les textes, images, liens, vidéos, etc.

# Les éléments title et meta

---

- On doit indiquer au moins deux autres éléments à l'intérieur de l'élément head : l'élément title, qui va contenir le titre de la page et un élément meta avec son attribut charset qui va permettre de définir l'encodage de la page.
- L'encodage le plus universel aujourd'hui est celui de valeur utf-8.
- Notez que vous devrez également régler l'encodage de votre éditeur de texte sur utf-8 afin que tous les caractères s'affichent bien par la suite dans votre navigateur.

# Observations sur la syntaxe

- Il est interdit d'imbriquer des éléments HTML
- En d'autres termes, le premier élément déclaré doit toujours être le dernier refermé.
- Il est important d'autre part de mettre en relief les éléments contenus dans d'autres et ceux qui sont au même niveau hiérarchique. Exemple de situation valide :

```
<balise ouvrante élément A>  
    <balise ouvrante élément B>  
    </balise fermante élément B>  
    <balise ouvrante élément C>  
    </balise fermante élément C>  
    <balise orpheline élément D>  
</balise fermante élément A>
```

# HTML Tutorial

## CREATION D'UNE PREMIERE PAGE HTML



# Première page HTML

---

- Le formateur vous guide pour créer, enregistrer et tester une première page HTML dont voici le modèle :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Je viens d'écrire un titre en HTML !</h1>
    <p>Et voilà mon premier paragraphe :)</p>
  </body>
</html>
```

- Pour les paresseux :  
<http://jsbin.com/sipujup/3/edit?html,output>

---

# HTML Tutorial

## INDENTATION ET COMMENTAIRES

# Bonnes pratiques

## L'indentation :

- **Indenter va permettre d'avoir un code plus propre et plus lisible, donc plus compréhensible.**
- Indenter permet également de plus facilement détecter les erreurs potentielles dans un code.

- Avec indentation :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Je viens d'écrire un titre en HTML !</h1>
    <p>Et voilà mon premier paragraphe :)</p>
  </body>
</html>
```

- Sans indentation :

```
<!DOCTYPE html>
<html>
<head>
<title>Première page HTML</title>
<meta charset= "utf-8">
</head>

<body>
<h1>Je viens d'écrire un titre en HTML !</h1>
<p>Et voilà mon premier paragraphe :)</p>
</body>
</html>
```

# Bonnes pratiques

## Les Commentaires :

- Les lignes de commentaires sont des lignes de texte que l'on écrit au milieu du code, afin de donner des indications sur ce que fait le code en question.
- Les commentaires sont précieux dans deux situations :
  - Dans le cas d'un gros / long projet, afin de se repérer dans le code ;
  - Si l'on souhaite distribuer son code, pour permettre aux autres développeurs de comprendre rapidement le code distribué.
- Les commentaires peuvent être mono-ligne ou multi-lignes. La syntaxe est la suivante:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!-- Je suis un commentaire, je ne serai pas affiché-->
    <h1>Je viens d'écrire un titre en HTML !</h1>

    <!--Ci-dessous, vous allez trouver
    mon premier paragraphe-->
    <p>Et voilà mon premier paragraphe :)</p>
  </body>
</html>
```

# Observation

---

- Ne pas oublier qu'avec l'outil d'inspection dont disposent aujourd'hui tous les navigateurs, n'importe qui peut lire les commentaires que vous avez introduit dans votre page.
- Il ne faut donc jamais y placer des informations sensibles (mots de passe, etc..)

---

# HTML Tutorial

## LES TITRES ET LES PARAGRAPHES EN HTML

# Définir des titres et des paragraphes

---

**Le HTML est un langage de balisage.**

- Son seul et unique rôle est de permettre de créer un document structuré en différenciant d'un point de vue sémantique les différents éléments d'une page.
- En ce sens, il est essentiel de différencier les titres des paragraphes au sein de la page.

# Les titres en HTML

- Il existe six niveaux hiérarchiques de titres ("heading" en anglais) définis par les éléments h1, h2, h3, h4, h5 et h6.  
L'élément h1 représente un titre très important, h2 un titre important, h3 un titre d'importance moyenne, etc.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!--Un titre très important-->
    <h1>Mon titre principal</h1>

    <!--Un titre important-->
    <h2>Je suis un titre important</h2>

    <!--Un autre titre important-->
    <h2>Moi aussi, je suis un titre important</h2>

    <!--Un titre un peu moins important-->
    <h3>Je suis un titre d'important moyenne</h3>

    <!--Etc, etc.-->
    <h4>Un titre pas très important</h4>
    <h5>Un titre peu important</h5>
    <h6>Un titre vraiment peu important</h6>
  </body>
</html>
```

- Pour tester ce code : <http://jsbin.com/sipujup/5/edit?html,output>



# Les éléments title et h\*

---

- Attention à ne pas confondre les éléments h1, h2, etc. et l'élément title, qui sont tout à fait différents.
- Les éléments h1, h2, etc. servent à définir des titres à l'intérieur de la page, tandis que l'élément title sert à définir le titre de la page, qui va s'afficher dans la barre haute du navigateur.

# Les paragraphes en HTML

- Pour **créer des paragraphes en HTML**, on utilise l'élément `p`.
- On peut créer autant de paragraphes que l'on souhaite dans une page. A chaque nouveau paragraphe, il faut utiliser un nouvel élément `p`.
- Pour chaque nouveau paragraphe, un retour à la ligne est créé automatiquement et affiché par le navigateur (exactement comme c'était le cas avec les titres).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!--Deux titres h1 et h2-->
    <h1>Mon titre principal</h1>
    <h2>Je suis un titre important</h2>

    <!--Deux paragraphes différents-->
    <p>Voici mon premier paragraphe.</p>
    <p>Et en voilà un second !</p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/sipujup/6/edit?html,output>

---

# HTML Tutorial

## LES ESPACES ET LES RETOURS À LA LIGNE EN HTML

# Problématique

- On peut ajouter autant d'espaces que l'on veut au sein d'un paragraphe ou d'un titre ou effectuer des retours à la ligne dans le code, ceux-ci ne seront jamais affichés visuellement dans le navigateur.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!--Les espaces ne seront pas affichés-->
    <h1>Mon titre                principal</h1>
    <h2>Je suis          un          titre important</h2>

    <!--Les retours à la ligne non plus-->
    <p>Voici
mon
premier
paragraphe.</p>
    <p>Et en voilà un second !</p>
  </body>
</html>
```

- Pour **effectuer un retour à la ligne en HTML** on utilise l'élément **br** qui est représenté par une unique balise orpheline

# Les changements de thématique

- Lorsque le retour à la ligne ne suffit pas pour indiquer suffisamment un changement de thématique, on peut utiliser l'élément `hr` :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!--Les espaces ne seront pas affichés-->
    <h1>Mon titre principal</h1>
    <h2>Je suis un titre important</h2>

    <!--hr marque un changement de thématique-->
    <p>HTML : le HTML est un langage utilisé pour marquer
    sémantiquement un contenu.</p>
    <hr>
    <p>CSS : le CSS sert à mettre en forme des éléments en
    leur appliquant des styles.</p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/sipujup/8/edit?html,output>

# Gestion des espaces

- Il n'existe pas à proprement parler d'**élément permettant de gérer les espaces en HTML**.
- Cependant, on peut utiliser l'élément `pre` qui permet de préformater un texte.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <!--Les espaces ne seront pas affichés-->
    <h1>Mon titre          principal</h1>
    <h2>Je suis          un          titre important</h2>

    <!--L'élément pre conserve les espaces et les retours à la ligne-->
    <pre>
      HTML : le HTML est un langage utilisé pour marquer
             sémantiquement un contenu.

      CSS : le CSS sert à mettre en forme des éléments en
             leur appliquant des styles.
    </pre>
  </body>
</html>
```

- Il est conseillé de ne pas abuser de cet élément qui ne comporte après tout aucune signification d'un point de vue sémantique.
- Pour tester le code : <http://jsbin.com/sipujup/9/edit?html,output>

---

# HTML Tutorial

## DEFINIR L'IMPORTANCE DES TEXTES

# L'importance des textes

---

- Il est des cas où l'on souhaite signifier à un moteur de recherche (et au navigateur) que telles ou telles parties d'un texte doivent être considérées comme très importantes ou juste importantes
- Pour cela, on dispose des éléments **strong** et **em**.



# L'élément strong

- L'élément HTML **strong** est utilisé pour signifier qu'un contenu est **très important** et doit être considéré comme tel par les moteurs de recherche.
- En résultat, le navigateur affichera le contenu à l'intérieur de l'élément strong en gras.
- Il ne faut pas pour autant utiliser cet élément dans l'unique but de mettre un morceau de texte en gras. C'est le rôle du CSS de gérer le poids d'un texte.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Le langage HTML</h1>

    <!--Strong indique un texte très important-->
    <p>
      HTML signifie <strong>HyperText Markup Language</strong>. Ce
      langage est utilisé pour marquer sémantiquement un contenu.
    </p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/sipujup/10/edit?html,output>

# L'élément em

- L'élément HTML **em** (pour **emphasis** ; « **emphase** » en français) sert à indiquer qu'un texte doit être considéré comme **relativement important**, mais avec moins d'importance qu'un texte dans un élément **strong**.
- Le résultat visuel par défaut de l'emphase est la mise en italique du texte contenu dans l'élément.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Première page HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Le langage HTML</h1>

    <!--Strong indique un texte très important-->
    <p>
      HTML signifie <strong>HyperText Markup Language</strong>. Ce
      <!--Em indique un texte d'importance relative-->
      langage est utilisé pour <em>marquer sémantiquement un contenu</em>.
    </p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/sipujup/11/edit?html,output>

---

# HTML Tutorial

## LES LISTES

# Définition d'une liste

---

- Les listes sont utiles pour apporter de la clarté et de l'ordre aux documents, ainsi que pour créer des menus de navigation
- Il existe **deux grands types de listes en HTML : les listes ordonnées et les listes non-ordonnées.**
- Il existe également un troisième type de liste un peu particulier et moins utilisé : **les listes de définitions.**

# Les listes non-ordonnées

- Les listes non-ordonnées vont être utiles pour lister des éléments sans hiérarchie ni ordre logique.
- Pour créer une liste non-ordonnée, on fait appel à un élément `ul` (pour « unordered list », ou « liste non-ordonnée » en français) qui va représenter la liste en soi ainsi qu'à des éléments `li` ("list items" = "éléments de liste") représentant chaque élément de la liste.
- Visuellement, des puces (les points noirs) apparaissent automatiquement devant chaque élément d'une liste non-ordonnée.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les listes en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les listes</h1>

    <!--Un exemple de liste non-ordonnée-->
    <ul>
      <li>Pomme</li>
      <li>Vélo</li>
      <li>Guitare</li>
    </ul>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/18/edit?html,output>

# Les listes ordonnées

- On utilise les listes ordonnées lorsqu'il y aura une notion d'ordre ou de progression logique entre les éléments de notre liste.
- Pour une liste ordonnée, on remplace l'élément `ul` par l'élément `ol`.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les listes en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les listes</h1>

    <!--Deux exemples de listes ordonnées-->
    <p>Liste n°1 :</p>
    <ol>
      <li>On naît</li>
      <li>On grandit</li>
      <li>On meurt</li>
    </ol>

    <p>Liste n°2 :</p>
    <ol>
      <li>Introduction</li>
      <li>Partie I</li>
      <li>Partie II</li>
      <li>Conclusion</li>
    </ol>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/19/edit?html,output>

# Les listes ordonnées

---

- Pour une liste ordonnée, des numéros sont affichés devant chaque élément de la liste par défaut.
- On peut changer ce comportement et afficher plutôt des chiffres romains ou des lettres grâce à l'attribut type des listes ordonnées.
- Cet attribut peut prendre cinq valeurs différentes :
  - « 1 » : valeur par défaut. Des chiffres apparaîtront devant chaque élément de la liste ;
  - « I » : Des chiffres romains majuscules apparaîtront devant chaque élément de la liste ;
  - « i » : Des chiffres romains minuscules apparaîtront devant chaque élément de la liste ;
  - « A » : Des lettres majuscules apparaîtront devant chaque élément de la liste ;
  - « a » : Des lettres minuscules apparaîtront devant chaque élément de la liste ;

# Les listes ordonnées

- Illustration :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les listes en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les listes</h1>

    <!--Deux exemples de listes ordonnées-->
    <p>Liste n°1 :</p>
    <ol type="A">
      <li>On naît</li>
      <li>On grandit</li>
      <li>On meurt</li>
    </ol>

    <p>Liste n°2 :</p>
    <ol type="i">
      <li>Introduction</li>
      <li>Partie I</li>
      <li>Partie II</li>
      <li>Conclusion</li>
    </ol>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/20/edit?html,output>



# Les listes de définitions

- Les listes de définitions, encore appelées « listes de descriptions » permettent de lister des termes et d'ajouter des définitions ou descriptions pour chacun de ces termes.
- Pour créer une liste de définitions, on utilise l'élément **dl** signifiant « description list » ou « liste de description / définition » en français pour définir la liste en soi, puis des éléments **dt** (description term) pour chaque élément à décrire et enfin l'élément **dd** pour la définition / description en soi.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les listes en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les listes</h1>

    <!--Une liste de définition-->
    <dl>
      <dt>HTML</dt>
      <dd>HTML signifie HyperText Markup Language.</dd>
    </dl>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/21/edit?html,output>

# Les listes imbriquées

- Il est tout-à-fait possible d’imbriquer une liste dans une autre en suivant quelques règles simples.
- Pour imbriquer une liste dans une autre, il suffit de définir une nouvelle liste à l’intérieur de l’un des éléments d’une autre liste, juste avant la balise fermante de cet élément
- Il est clair que dans ce cas, une bonne indentation est indispensable.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les listes en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les listes</h1>

    <!--Listes imbriquées-->
    <ol>
      <li>Introduction</li>
      <li>Partie I
        <!--On imbrique une liste non-ordonnée dans une liste ordonnée-->
        <ul>
          <li>Définitions</li>
          <li>Auteurs</li>
          <li>Exemples</li>
        </ul>
      </li>
      <li>Partie II</li>
      <li>Conclusion</li>
    </ol>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/22/edit?html,output>

---

# HTML Tutorial

## LIENS INTERNES ET EXTERNES EN HTML

# Définition et types de liens

---

- Pour **créer des liens en HTML**, on utilise l'élément **a** accompagné de son attribut **href** (hypertext reference) qui prend comme valeur la cible du lien.
- La cible d'un lien est l'adresse de la page de destination du lien.
- Il y a deux grands types de liens : les liens permettant de se déplacer d'une page vers une autre à travers un même site et les liens permettant de se rendre sur d'autres sites.

# Les liens externes en HTML

- Les liens externes vont être des liens ramenant vers des pages d'autres sites.
- Pour créer un lien externe, on indique l'URL complète de destination du lien.
- Lorsque l'attribut href prend une URL complète en valeur, on parle de valeur absolue (car celle-ci est fixe et ne dépend de rien). Les liens externes utilisent toujours des valeurs absolues.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les liens</h1>

    <p>Cliquez sur <a href="http://wikipedia.org">ce lien </a>pour
    aller sur Wikipédia.</p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/23/edit?html,output>

# Les liens internes en HTML

---

- Les liens internes vont être des liens ramenant vers d'autres pages au sein d'un même site.
- Dans ce cas, on renseigne une valeur relative pour l'attribut **href**.
- La valeur est relative car on indique l'adresse de la page de destination relativement à l'adresse de la page de départ (c'est-à-dire celle à partir de laquelle on fait notre lien) Pour créer des liens internes, il faut donc distinguer trois cas :
  - Le cas où la page de départ et celle de destination sont dans le même dossier ;
  - Le cas où la page de destination est dans un sous-dossier par rapport à la page de départ ;
  - Le cas où la page de destination est dans un dossier parent par rapport à la page de départ.

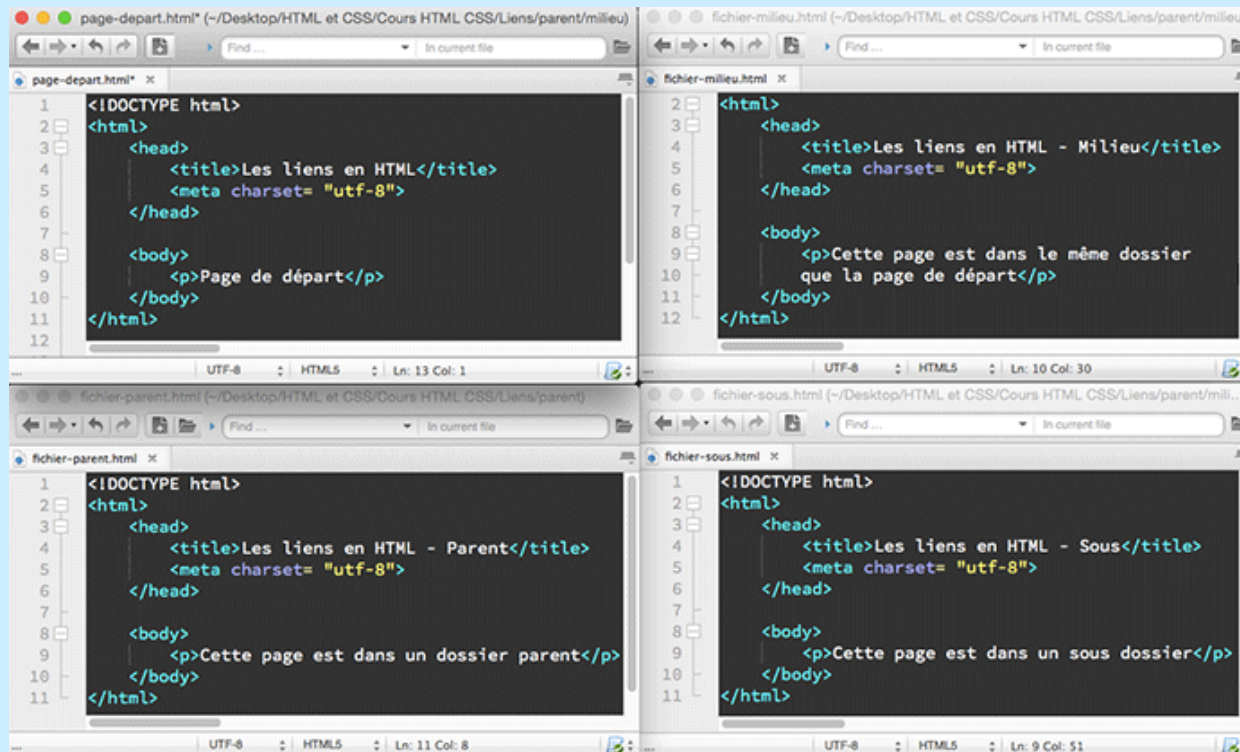
# Les liens internes en HTML

---

- Pour construire un exemple, prenons pour hypothèse que l'on dispose de trois dossiers distincts et quatre fichiers HTML selon :
- Un dossier nommé « **parent** », contenant un fichier HTML nommé **fichier-parent.html** ainsi qu'un dossier « **milieu** » ;
- Un dossier nommé « milieu », contenant les fichiers **page-depart.html** et **fichier-milieu.html** ainsi qu'un dossier nommé « **sous** » ;
- Un dossier nommé « sous », comportant un fichier nommé **fichier-sous.html**.

# Les liens internes en HTML

- Chaque fichier HTML est construit suivant une structure minimale de page HTML afin que nos fichiers soient valides ainsi qu'une phrase de texte pour les différencier les uns des autres :



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Les liens en HTML</title>
5     <meta charset= "utf-8">
6   </head>
7
8   <body>
9     <p>Page de départ</p>
10  </body>
11 </html>
```

```
2 <html>
3   <head>
4     <title>Les liens en HTML - Milieu</title>
5     <meta charset= "utf-8">
6   </head>
7
8   <body>
9     <p>Cette page est dans le même dossier
10    que la page de départ</p>
11  </body>
12 </html>
```

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Les liens en HTML - Parent</title>
5     <meta charset= "utf-8">
6   </head>
7
8   <body>
9     <p>Cette page est dans un dossier parent</p>
10  </body>
11 </html>
```

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Les liens en HTML - Sous</title>
5     <meta charset= "utf-8">
6   </head>
7
8   <body>
9     <p>Cette page est dans un sous dossier</p>
10  </body>
11 </html>
```



# Les liens internes en HTML

- La syntaxe d'accès aux différents fichiers est illustrée par le code suivant:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <p>Page de départ</p>

    <!--On crée un lien vers une page dans un même dossier-->
    <p>Cliquez <a href="fichier-milieu.html">ici</a> pour accéder à
    la page "fichier-milieu.html"</p>

    <!--On crée un lien vers une page dans un dossier parent-->
    <p>Cliquez <a href="../fichier-parent.html">ici</a> pour accéder à
    la page "fichier-parent.html"</p>

    <!--On crée un lien vers une page dans un sous dossier-->
    <p>Cliquez <a href="sous/fichier-sous.html">ici</a> pour accéder à
    la page "fichier-sous.html"</p>
  </body>
</html>
```

# Les liens internes en HTML

- Et si l'on souhaitait faire un lien de la page **fichier-sous.html** vers la page **fichier-parent.html** située deux dossiers plus haut :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML - Sous</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <p>Cette page est dans un sous dossier</p>
    <p>Clique <a href="../..fichier-parent.html">ici</a>
    vers la page "fichier-parent.html"</p>
  </body>
</html>
```

# L'attribut target ("cible")

- L'attribut **target** permet de choisir où doit s'ouvrir la page de destination.
- En pratique, on utilise le plus souvent la valeur « **\_blank** » qui spécifie que la nouvelle page doit s'ouvrir dans un nouvel onglet ou dans une nouvelle fenêtre selon le navigateur utilisé par vos visiteurs.
- Il existe aussi la valeur « **\_self** » où dans ce cas, la page recherchée remplace la page affichée.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <p>Page de départ</p>

    <!--Ce lien va s'ouvrir dans un nouvel onglet ou fenêtre-->
    <p>Vers <a href="http://wikipedia.org" target="_blank">Wikipédia</a></p>
  </body>
</html>
```

- Pour tester le code : <http://jsbin.com/gopega/24/edit?html,output>

---

# HTML Tutorial

## AUTRES TYPES DE LIENS EN HTML

## D'autres utilisations de l'élément HTML a

---

- On peut également utiliser l'élément a pour créer d'autres types de liens, qui ne vont pas ramener vers d'autres pages mais permettre par exemple de se déplacer dans une même page, d'afficher ou de télécharger un fichier PDF ou encore d'envoyer un mail.

# Créer une ancre en HTML

Pour créer un lien de type ancre, il faut une ancre (!).

- On crée cette ancre en rajoutant un attribut **id** à l'intérieur de la balise ouvrante de l'élément où l'on veut envoyer le visiteur
- Une fois l'ancre créée, pour créer le lien ramenant à l'ancre, on utilise une nouvelle fois un élément **a** avec son attribut **href** dans les conditions suivantes: on indique la valeur donnée à l'attribut **id** comme valeur pour l'attribut **href**, en faisant précéder cette valeur d'un **#**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Les ancrs</h1>
    <p>Accès direct à <a href="#ancree1">mon ancre n°1</a></p>
    <h2 id="ancree1">Une première ancre</h2>
  </body>
</html>
```

# Un lien pour permettre l'envoi d'un mail en HTML

---

On peut également utiliser l'élément **a** pour permettre à nos utilisateurs de nous envoyer simplement un mail.

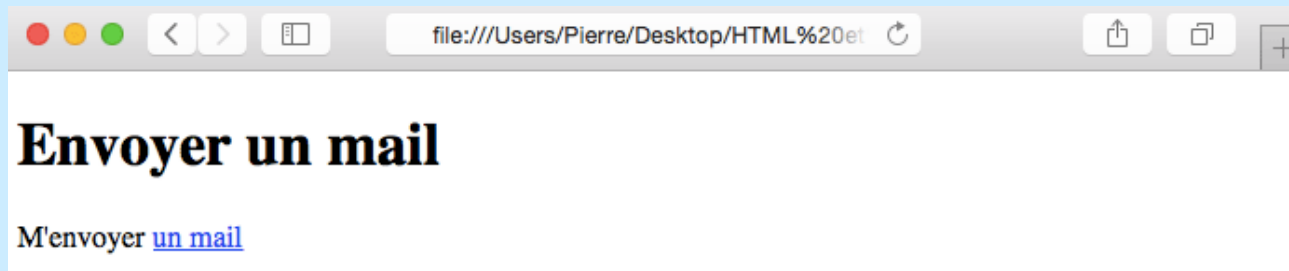
- Pour cela, on va placer la valeur « **mailto :** » suivie de notre adresse email en valeur de l'attribut **href**.
- Lorsqu'un visiteur va cliquer sur notre lien, sa messagerie par défaut va automatiquement s'ouvrir. Par exemple, si vous avez un Mac, ce sera certainement l'application « Mail » qui va s'ouvrir.
- De plus, le champ destinataire sera automatiquement rempli avec notre adresse email.
- Note : si vous travaillez sous Windows, il est possible que rien ne se passe si vous n'avez configuré aucune messagerie par défaut.

# Un lien pour permettre l'envoi d'un mail en HTML

- Exemple:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les liens en HTML</title>
    <meta charset= "utf-8">
  </head>

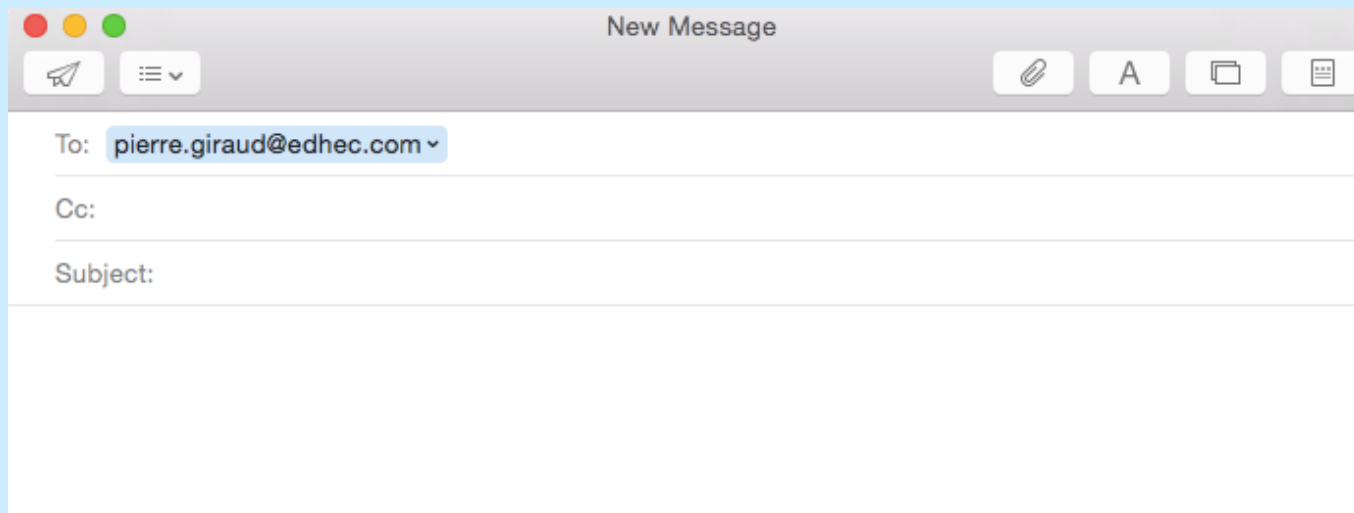
  <body>
    <h1>Envoyer un mail</h1>
    <p>M'envoyer <a href="mailto:pierre.giraud@edhec.com">un mail</a></p>
  </body>
</html>
```





# Un lien pour permettre l'envoi d'un mail en HTML

- Dès qu'un visiteur clique sur le texte de notre lien, sa messagerie par défaut s'ouvre s'il en a configuré une :



# Un lien pour télécharger un fichier

---

On peut encore utiliser l'élément `a` pour permettre à vos visiteurs de télécharger certains types de fichiers, comme des fichiers PDF ou Word par exemple.

- Pour faire télécharger un fichier, on a deux solutions:
- La première solution consiste à utiliser l'attribut **download** dans l'élément `a` grâce auquel vous pouvez théoriquement faire télécharger n'importe quel type de fichier à vos visiteurs.
- Le problème est que cet attribut a été créé très récemment et ne fonctionne pas avec certains navigateurs majeurs, dont Safari entre autres.

# Un lien pour télécharger un fichier

---

- La deuxième solution est de préciser tout simplement l'adresse du fichier avec son extension sous forme de lien relatif en valeur de l'attribut href.
- L'inconvénient de cette deuxième méthode est que vous ne pouvez pas faire télécharger un fichier HTML par exemple à vos visiteurs car le fichier va être ouvert et donc interprété par le navigateur si ceux-ci cliquent directement sur votre lien.

# Un lien pour télécharger un fichier

- Voici un exemple utilisant la deuxième méthode ci-dessous.



- Lorsqu'un utilisateur clique sur le lien, le fichier « cours.pdf » s'ouvre et l'utilisateur peut alors le télécharger.

---

# HTML Tutorial

## INSERER DES IMAGES EN HTML

# Les différents formats d'image

---

- Les formats les plus utilisés sur Internet sont :
- Le JPG ou JPEG ;
- Le PNG ;
- Le GIF ;
- Le BITMAP.

Les autres formats, tels que TIFF, PSD, sont considérés comme plus spécifiques.

# Les différents formats d'image

---

- Le format JPEG (pour Joint Photographic Expert Group), ou plus communément JPG est un format qui permet de compresser le poids d'une image par dix tout en conservant une bonne qualité.
- Le PNG (Portable Network Graphic) est un format qui a été créé à l'origine pour remplacer le format GIF (Apple). Ce format gère la transparence et présente un très bon taux de compression tout en conservant une bonne qualité d'image
- Le GIF (Graphic Interchange Format) est un vieux format d'images dont l'utilisation n'est plus recommandée aujourd'hui, sauf dans le cas d'images animées..
- Le BITMAP (ou BMP) est un format qui possède une très bonne prise en charge par tous les navigateurs et éditeurs, mais la compression des images y est assez mauvaise, ce qui donne au final des images très lourdes et donc longues à charger.

# Le poids des images

---

- Toute image possède un poids généralement exprimé de kilo-octets. Plus une image est lourde, plus elle va demander de ressources de la part du serveur et du navigateur pour être chargée.
- Cette problématique est d'autant plus à considérer aujourd'hui avec l'essor de l'Internet mobile car les connexions sont beaucoup moins puissantes sur mobile.
- Il est donc plus essentiel que jamais aujourd'hui d'enregistrer les images au bon format et de les recadrer à la bonne taille avant de les envoyer sur le serveur.



# Insérer des images

---

- L'insertion d'images en HTML se fait au moyen de l'élément HTML **img**. Cet élément est représenté par une balise orpheline.
- Au sein de l'élément **img**, il faut obligatoirement préciser deux attributs : les attributs **src** et **alt**.
- L'attribut **src** (pour source) prend comme valeur l'adresse de l'image (adresse relative ou absolue) tandis que l'attribut **alt** (pour alternative) contient un texte alternatif décrivant l'image. Ce texte sera affiché si l'image ne peut pas l'être pour une raison ou pour une autre.

# Insérer des images

- Exemple pour une image « sunset.png » stockée dans le même dossier que la page HTML d'une largeur et une hauteur de 150 pixels. (ce qui détermine sa dimension d'affichage)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Insérer des images en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Un joli coucher de soleil</h1>
    
  </body>
</html>
```



# Insérer des images

- Autre exemple avec une adresse absolue:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Insérer des images en HTML</title>
    <meta charset= "utf-8">
  </head>

  <body>
    <h1>Ratatouille !</h1>
    
  </body>
</html>
```



- Mais noter qu'il est généralement déconseillé d'utiliser des images provenant d'autres sites car si ces sites les suppriment, elles ne s'afficheront plus non plus sur le vôtre.

---

# HTML Tutorial

## VALIDATION ET COMPATIBILITE DU CODE HTML

# Les problèmes relatifs aux navigateurs

---

- Aujourd'hui, heureusement, le web a atteint une certaine maturité et possède désormais des règles bien définies. Cela n'a pas toujours été le cas, loin s'en faut !
- L'époque n'est pas si éloignée où, en l'absence de normes, les différents navigateurs se livraient une guerre sans merci et implémentaient de différentes façons certains codes avec leurs propres normes.
- Ainsi, certains éléments ou attributs HTML n'étaient pas supportés par certains navigateurs et on devait donc créer des codes différents afin que chaque navigateur affiche le résultat voulu.
- Il reste donc important de toujours tester son code sur les différents navigateurs les plus courants.

# Les entités HTML

---


- Le HTML possède des caractères réservés. Par exemple, on ne peut écrire les signes « < » et « > » tels quels dans une pages web, tout simplement car le navigateur pensera que l'on vient d'ouvrir une balise d'élément.
- Pour remédier à ce problème, on doit remplacer ces caractères réservés par des entités. Les entités sont des suites de caractères représentant un caractère réservé en HTML
- Trois entités à connaître:

Nom de l'entité	Résultat visuel
&lt;	<
&gt;	>
&amp;	&

# Les espaces

- Bien que le HTML ne prenne pas en compte les espaces, il est cependant possible d'utiliser l'entité **&nbsp;** (pour « non-breaking space ») pour ajouter des espaces en HTML.
- Exemple d'utilisation de ces diverses entités :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les entités</title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>L'élément HTML break s'écrit comme ceci : &lt;br&gt;</p>
    <p>Non &nbsp;&nbsp;&nbsp; breaking &nbsp;&nbsp;&nbsp; space</p>
  </body>
</html>
```



L'élément HTML break s'écrit comme ceci : <br>

Non    breaking    space

- Pour tester ce code : <http://jsbin.com/gopega/150/edit?html,output>

# Tester la validité de son code

---

- Il faut toujours s'efforcer d'écrire un code valide. Cela évitera des bugs potentiels et votre site sera au final mieux référencé sur les moteurs de recherche.
- Pour vérifier la validité d'un code HTML ou CSS, le w3c (World Wide Web Consortium), c'est-à-dire l'organisme qui gère l'évolution des langages comme le HTML et le CSS entre autres, a mis à disposition des validateurs de code, gratuits.
- Vous pouvez trouver les validateurs HTML et CSS aux adresses suivantes :
- HTML : <http://validator.w3.org/>
- CSS : <http://jigsaw.w3.org/css-validator/>



- X