

HTML5 / CSS

Master **IMSD**
IFRISSE

Pierre Claver OUEDRAOGO

Tables de matières

Aperçu du cours

Objectifs du cours

Evaluations

Unité 0 : Définitions et usages du HTML5 et du CSS

Unité 1 : Les bases du HTML5

Unité 2 : Les tableaux HTML5

Unité 3 : Les Formulaires HTML5

Unité 4 : Les bases du CSS

Conclusion

Aperçu du cours

Il existe aujourd'hui des dizaines et des dizaines de langages de programmation différents : HTML, CSS, JavaScript, PHP, Python, Ruby on Rails, C, C#, C++, Java, etc. pour ne citer qu'eux.

Parmi ces langages, on note ceux là qu'on appelle communément les langages du web. Ce sont entre autres : HTML, CSS, JavaScript, PHP, SQL, etc.

Comment fonctionne le web ?

le Web est un système d'échange entre un client et un serveur.

Le client c'est nous, c'est notre navigateur Web, c'est lui qui va nous permettre de voir le Web depuis notre ordinateur, smartphone ou tablette. C'est celui que vous connaissez sans doute sous le nom de Chrome, Firefox, Safari, Internet Explorer, etc.

Et le serveur, c'est en fait un ordinateur puissant qui stocke et héberge des sites Web. C'est sur cet ordinateur que se trouvent les pages Web, c'est à dire tous les fichiers du site internet auquel on veut accéder.

Le but du serveur web est de servir des clients, d'où le nom « serveur ».

Aperçu du cours

On note ainsi les langages dits clients et les langages dits serveurs.

Les langages clients correspondent aux langages nécessaires pour afficher une page web sur un écran d'ordinateur. On retrouve : le HTML5, le CSS3 et JavaScript.

Les langages sont exécutés côté serveur, car un ordinateur n'est pas capable de les lire. Le rôle du serveur est donc de traduire et de restituer ce code afin qu'il soit lisible par un ordinateur. On retrouve : Le PHP, le Java, etc.

Dans ce cours, nous allons nous intéresser aux premiers types de langages: les langages Clients et plus précisément le HTML et le CSS.

Objectifs du cours

À la fin de ce cours, vous devrez être capable de :

- Utiliser du code HTML
- Structurer une page web en HTML
- Intégrer des formulaires dans une page web
- Utiliser du code CSS
- Mettre en forme une page web avec du CSS

Unité 0 : Définitions et usages du HTML et du CSS

Nous allons dans cette leçon poser une première définition du HTML et du CSS et comprendre plus précisément quel va être leur rôle dans la création d'un site Internet. On va ainsi expliquer ce qu'on peut faire et ce qu'on ne peut pas faire avec chacun de ces deux langages.

Le HTML : langage de balisage

Le HTML est un langage qui a été créé en 1991. Les sigles « HTML » sont l'abréviation de « HyperText Markup Language » ou « langage de balisage hypertexte » en français.

Le HTML est donc un langage de balisage, c'est-à-dire un langage qui va nous permettre de définir les différents contenus d'une page.

Comment accède-t-on à une page d'un site internet ? Imaginons que l'on souhaite accéder à la page d'accueil de mon site. Pour cela, on va taper `www.monsite.net` dans la barre de notre navigateur.

Lorsqu'on demande à accéder à une page d'un site internet, nous sommes ce qu'on appelle des « clients ».

Unité 0 : Définitions et usages du HTML et du CSS

Notre navigateur va contacter le serveur sur lequel est hébergée la page à laquelle on souhaite accéder et lui demander de renvoyer les différents éléments de la page : la page sous forme de code HTML et potentiellement les différents médias intégrés dans la page.

Le navigateur va donc recevoir ces différents éléments et les afficher. Cependant, comment fait-il pour savoir ce qu'il doit afficher ? Il va bien évidemment utiliser le code HTML. En effet, le navigateur comprend bien les différentes balises HTML (le HTML utilise ce qu'on appelle des « balises » pour définir les contenus) et va donc « comprendre » de quoi est constituée notre page et ce qu'il doit afficher.

Le rôle du HTML est donc crucial puisqu'il va être notre langage privilégié pour indiquer aux navigateurs ce qui est constituée chaque page et ce qu'ils doivent afficher. Grâce au HTML, on va par exemple pouvoir indiquer que tel contenu est un texte qui n'est qu'un paragraphe, que tel autre contenu est un texte qui est un titre de niveau 1 dans notre page, que tel autre contenu est une liste, un lien, etc.

Unité 0 : Définitions et usages du HTML et du CSS

En plus de cela, le HTML va également nous permettre d'insérer différents médias (images, vidéos, etc.) dans nos pages web en indiquant au navigateur « à cette place-là dans ma page, je veux que s'affiche cette image ». Notez que dans ce cas précis, pour que le navigateur affiche la bonne image, on va lui fournir l'adresse de l'image dans le code HTML.

Le CSS : langage de styles

Le CSS a été créé en 1996, soit 5 ans après le HTML. Les sigles « CSS » sont l'abréviation de « Cascading StyleSheets » ou « feuilles de styles en cascade » en français.

Le CSS vient résoudre un problème bien différent du HTML : en effet, le HTML sert à définir les différents éléments d'une page, à leur donner du sens. Le CSS, lui, va servir à mettre en forme les différents contenus définis par le HTML en leur appliquant des styles.

Unité 0 : Définitions et usages du HTML et du CSS

Le HTML va donc créer la structure des pages tandis que le CSS va nous permettre de modifier l'apparence des contenus de la page. On va ainsi par exemple pouvoir définir la taille, la couleur ou l'alignement de certains contenus HTML et notamment en l'occurrence de certains textes dans notre page.

Pour styliser le contenu lié à un élément HTML en CSS, nous allons pouvoir le cibler en nous basant sur l'élément HTML en question ou en utilisant d'autres procédés que nous verrons plus tard dans ce cours.

A retenir : n'utilisez pas le HTML pour mettre en forme vos contenus !

Unité 0 : Définitions et usages du HTML et du CSS

Au fil du temps, les langages HTML et CSS ont beaucoup évolué. Dans la toute première version de HTML (HTML 1.0) il n'était même pas possible d'afficher des images !

Voici un très bref historique de ces langages pour votre culture générale.

Les versions de HTML

HTML 1 : c'est la toute première version créée par Tim Berners-Lee en 1991.

HTML 2 : la deuxième version du HTML apparaît en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML.

HTML 3 : apparue en 1996, cette nouvelle version du HTML rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.

Unité 0 : Définitions et usages du HTML et du CSS

HTML 4 : cette version aura été utilisée un long moment durant les années 2000. Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style, notre fameux CSS !

HTML 5 : c'est LA dernière version. De plus en plus répandue, elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc. C'est cette version que nous allons découvrir ensemble.

Unité 1 : Définitions et usages du HTML et du CSS

Les versions de CSS

CSS 1 : dès 1996, on dispose de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.

CSS 2 : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut désormais utiliser des techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.

CSS 3 : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc.

Unité 1 : Les bases du HTML

Structure minimale d'une page HTML

Le W3C(World Wide Web Consortium) abrégé par le sigle W3C, est un organisme de standardisation, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, XSL, PNG, SVG et SOAP. En effet aujourd'hui, tous les navigateurs sérieux suivent les standards proposés par le W3C, ce qui n'était pas forcément le cas dans le passé.

Cela nous permet donc d'être plus ou moins certain qu'un même code va produire le même résultat quel que soit le navigateur utilisé par les visiteurs qui consultent la page.

Le schéma de base d'une page HTML va donc toujours être le même. C'est ce que nous appellerons la « structure minimale d'une page HTML valide ».

Voici ci-dessous le code minimum pour créer une page HTML valide.

Unité 1 : Les bases du HTML

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

  </body>
</html>
```

Le DOCTYPE

Tout d'abord, nous devons toujours démarrer une page HTML en précisant le doctype de notre document. Comme son nom l'indique, le doctype sert à indiquer le type du document.

Faites bien attention à l'écriture du doctype : vous pouvez remarquer que la balise représentant le doctype commence par un point d'exclamation. Ceci est un cas unique.

Dans la balise de l'élément doctype, on va préciser le langage utilisé, à savoir le HTML dans notre cas.

Unité 1 : Les bases du HTML

L'élément HTML

Après avoir renseigné le type du document, nous devons utiliser un élément html. Cet élément est composé d'une paire de balises ouvrante `<html>` et fermante `</html>`.

L'élément html va représenter notre page en soi. On va insérer tout le contenu de notre page (et donc les autres éléments) à l'intérieur de celui-ci.

Les éléments head et body

A l'intérieur de l'élément html, nous devons à nouveau indiquer obligatoirement deux éléments qui sont les éléments **head** et **body** et qui vont avoir des rôles très différents.

L'élément **head** est un élément d'en-tête. Il va contenir des éléments qui vont servir à fournir des informations sur la page au navigateur, comme le titre de la page ou encore le type d'encodage utilisé pour que celui-ci puisse afficher les caractères de texte correctement.

L'élément **body** va lui contenir tous les éléments définissant les contenus « visibles » de la page, c'est-à-dire les contenus à destination de l'utilisateur et notamment les différents textes présents dans la page, les images, etc.

Unité 1 : Les bases du HTML

Les éléments title et meta

Au sein de l'élément head, nous allons devoir à minima indiquer deux éléments qui vont permettre de donner des informations essentielles sur la page au navigateur : les éléments title et meta.

L'élément title va nous permettre d'indiquer le titre de la page en soi, qui ne doit pas être confondu avec les différents textes définis comme des titres dans la page. Ce titre de page est le texte visible sur le haut des onglets de votre navigateur.

L'élément meta sert lui à transmettre des meta informations sur la page au navigateur. Cet élément possède de nombreux attributs différents. Le type d'informations transmis va dépendre de l'attribut que l'on va préciser.

Ici, ce qui nous intéresse va être de préciser le type d'encodage utilisé dans nos pages. Cela va permettre aux navigateurs d'afficher correctement nos différents textes avec les accents, les cédilles, etc.

Pour faire cela, nous allons utiliser l'attribut charset pour « characters set » c'est-à-dire « ensemble de caractères » de l'élément meta et lui fournir la valeur utf-8.

Unité 1 : Les bases du HTML

La valeur utf-8 est de loin la valeur la plus utilisée sur le web et est la valeur de référence pour tous les alphabets latins. Cela va permettre à chacun de nos caractères de s'afficher correctement dans le navigateur.

PS :

- Lorsque vous travaillez en local vous devrez, avec certains éditeurs, renseigner également l'encodage de la page dans l'éditeur afin que le contenu de celle-ci s'affiche bien et notamment si vous utilisez un outil de prévisualisation de page fourni par votre éditeur.
- Il faut aussi noter que nous n'avons absolument pas le droit de « croiser » les balises des éléments ou, pour le dire plus clairement : le premier élément déclaré doit toujours être le dernier refermé, tandis que le dernier ouvert doit toujours être le premier fermé.

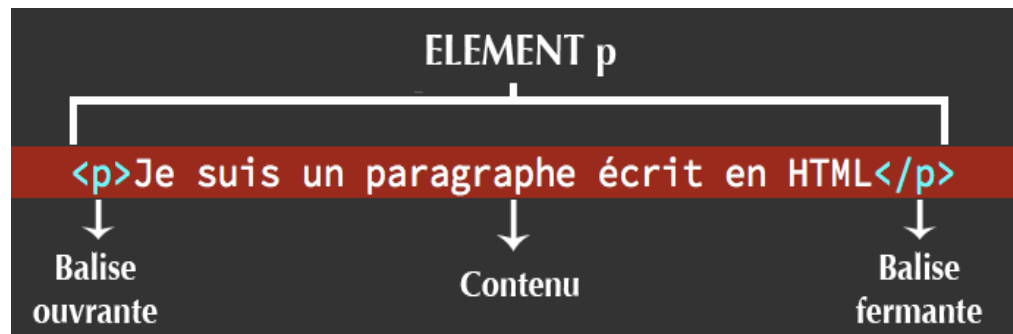
Unité 0 : Les bases du HTML

Les balises HTML

Un élément HTML peut être soit constitué d'une paire de balises (ouvrante et fermante) et d'un contenu, soit d'une balise unique qu'on dit alors « orpheline ».

L'élément p (qui sert à définir un paragraphe) est par exemple constitué d'une balise ouvrante, d'une balise fermante et d'un contenu textuel entre les balises. L'idée ici est que le texte contenu entre les deux balises va être le texte considéré par le navigateur comme étant un paragraphe.

Voici comment on va écrire cela :



Notez bien ici la différence entre la balise ouvrante et la balise fermante de notre élément p : la balise fermante contient un slash avant le nom de l'élément.

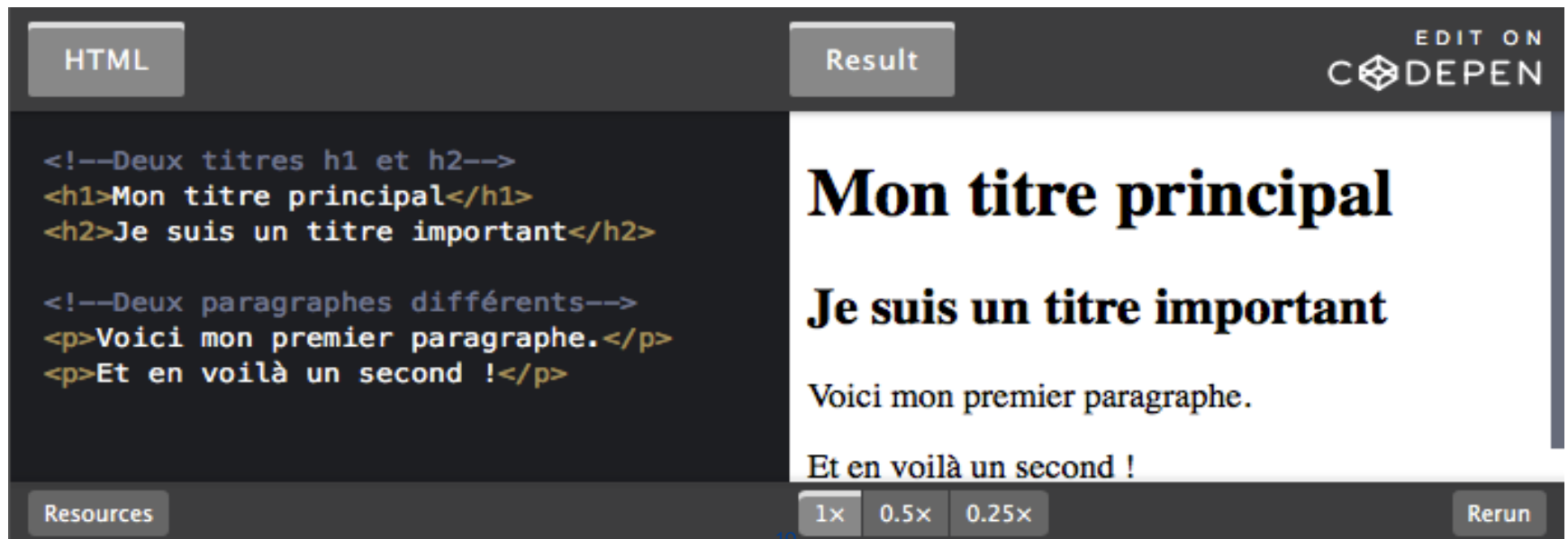
Unité 1 : Les bases du HTML

Définition de paragraphes en HTML

Donc pour créer des paragraphes en HTML, nous allons utiliser 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 va être créé automatiquement et affiché par votre navigateur (exactement comme c'était le cas avec les titres).



The screenshot shows a web editor interface with two main panels: 'HTML' on the left and 'Result' on the right. The 'HTML' panel contains the following code:

```
<!--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>
```

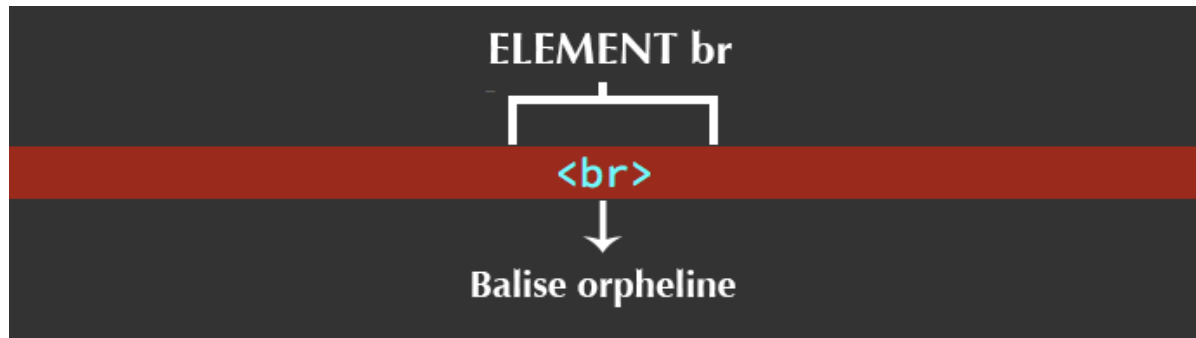
The 'Result' panel shows the rendered output of this code. It features a dark header with 'EDIT ON CODEPEN' on the right. The main content area displays:

- A large heading: **Mon titre principal**
- A slightly smaller heading: **Je suis un titre important**
- A paragraph: Voici mon premier paragraphe.
- Another paragraph: Et en voilà un second !

At the bottom of the editor, there are controls for zooming (1x, 0.5x, 0.25x) and a 'Rerun' button.

Unité 1 : Les bases du HTML

Certains éléments en HTML ne vont être constitués que d'une balise qu'on appelle alors orpheline. Cela va être le cas pour certains éléments qui ne possèdent pas de contenu textuel comme l'élément **br** par exemple qui sert simplement à créer un retour à la ligne en HTML et qui va s'écrire comme ceci :



Notez ici qu'il est possible que vous rencontriez un jour une syntaxe un peu différente pour les balises orphelines utilisant un slash après le nom de l'élément comme ceci : `
`.

Cette syntaxe est une syntaxe qui était acceptée il y a quelques années mais qui est aujourd'hui dépréciée en HTML. Elle provient en fait d'un autre langage qui est le XML.

Unité 1 : Les bases du HTML

Les attributs HTML

Les éléments vont également pouvoir contenir des attributs qu'on va alors placer au sein de la balise ouvrante de ceux-ci. Pour certains éléments, les attributs vont être facultatifs tandis que pour d'autres ils vont être obligatoires pour garantir le bon fonctionnement du code HTML.

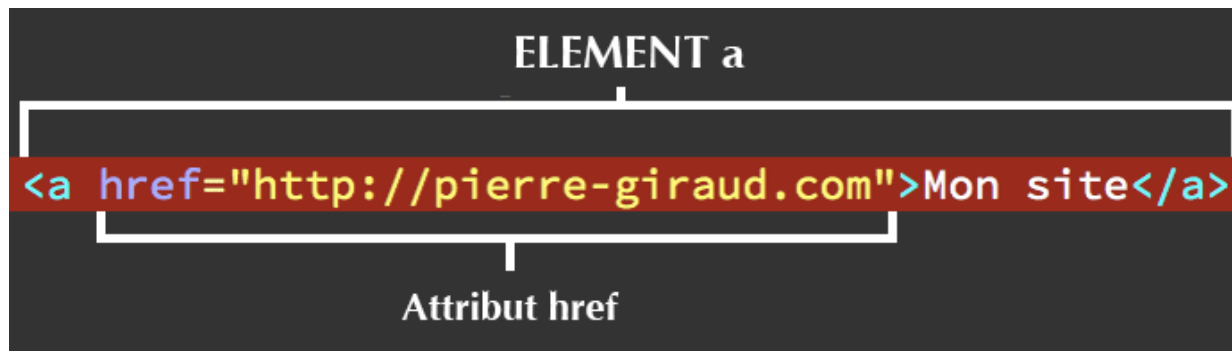
Un attribut contient toujours une valeur, qu'on peut cependant parfois omettre dans le cas des attributs ne possédant qu'une seule valeur (car la valeur est alors considérée comme évidente).

Prenons ici l'exemple de l'élément **a** qui est l'abréviation de « anchor » ou « ancre » en français. Cet élément va principalement nous servir à créer des liens vers d'autres pages.

Pour le faire fonctionner correctement, nous allons devoir lui ajouter un attribut **href** pour « hypertexte référence » ou « référence hypertexte » en français.

Unité 1 : Les bases du HTML

En effet, c'est l'attribut **href** qui va nous permettre de préciser la cible du lien, c'est-à-dire la page de destination du lien en lui passant l'adresse de la page en question en valeur.



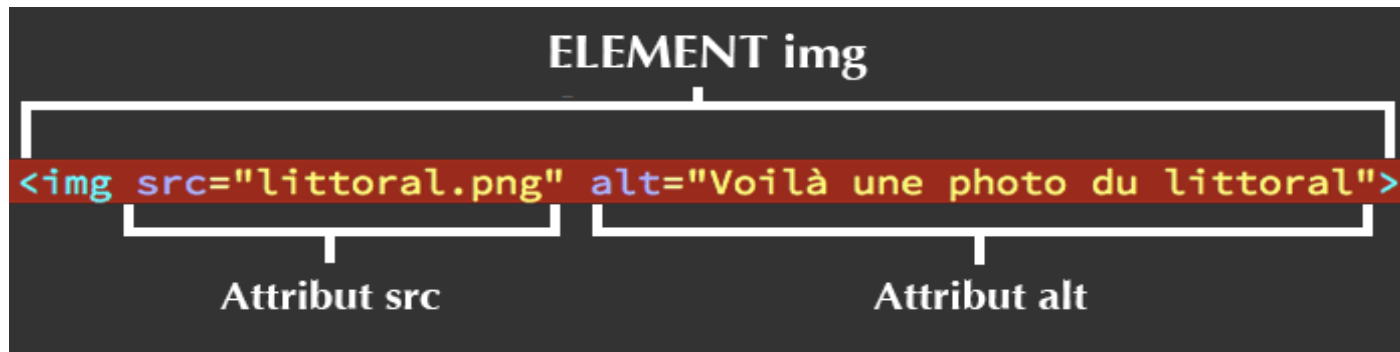
Un autre exemple : l'élément **img**, servant à insérer une image dans une page HTML, va lui nécessiter deux attributs qui sont les attributs **src** et **alt**.

L'attribut **src** (pour « source ») va prendre comme valeur l'adresse de l'image tandis que l'attribut **alt** (pour « alternative ») va nous permettre de renseigner une description textuelle de l'image qui sera affichée dans les cas où l'image ne serait pas disponible pour une raison ou une autre : image introuvable, impossible à charger, etc.

Unité 1 : Les bases du HTML

L'attribut **alt** va également se révéler indispensable pour rendre notre site accessible aux non-voyants ou aux mal voyants et pour leur fournir une bonne expérience de navigation puisqu'ils sont généralement équipés de lecteur spéciaux qui vont pouvoir lire la valeur de l'attribut **alt** et donc leur permettre de se faire une représentation du contenu de l'image.

Notez au passage que l'élément **img** n'est constitué que d'une seule balise orpheline, tout comme l'élément **br** vu précédemment. On place dans ce cas les attributs au sein de la balise orpheline.



Unité 1 : Les bases du HTML

- Les commentaires en HTML vont prendre la forme d'une balise orpheline très particulière, avec un chevron ouvrant suivi d'un point d'exclamation suivi de deux tirets au début, du commentaire en soi puis à nouveau de deux tirets et d'un chevron fermant (sans point d'exclamation cette fois, attention !). Par exemple :

```
<!-- Mon commentaire -- >
```

Définition de titres en HTML

Il existe six niveaux hiérarchiques de titres (“heading” en anglais) définis par les éléments **h1, h2, h3, h4, h5 et h6** qui vont nous permettre d'organiser le contenu dans nos pages.

Bon à savoir : « h » signifie « heading », soit l'équivalent du mot « titre » en français. Les éléments en HTML portent souvent l'initiale de ce qu'ils représentent, en anglais.

L'élément h1 représente un titre principal dans notre page ou dans une section de page et, à ce titre, nous n'allons pouvoir utiliser qu'un seul élément h1 par page (ou par section de page)

Unité 1 : Les bases du HTML

HTML

```
<!--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>
```

Result

EDIT ON
CODEPEN

Mon titre principal

Je suis un titre important

**Moi aussi, je suis un titre
important**

Je suis un titre d'important moyenne

Un titre pas très important

Un titre peu important

Un titre vraiment peu important

Resources

1x

0.5x

0.25x

Rerun

Unité 1 : Les bases du HTML

Les entités HTML

Une entité HTML est une suite de caractère qui est utilisée pour afficher un caractère réservé ou un caractère invisible (comme un espace) en HTML.

Qu'est-ce qu'un caractère réservé ? C'est un caractère qui possède déjà une signification particulière en HTML. Par exemple, imaginions que l'on souhaite afficher le caractère < dans un texte.

On ne va pas pouvoir mentionner ce caractère tel quel dans notre éditeur car le navigateur va interpréter cela comme l'ouverture d'une balise d'un élément. Il va donc falloir indiquer au navigateur qu'on souhaite afficher le caractère < en tant que tel et non pas ouvrir une balise.

Pour cela, il va falloir échapper le sens de ce caractère, et c'est ce à quoi vont nous servir les entités HTML.

Unité 1 : Les bases du HTML

Exemples :

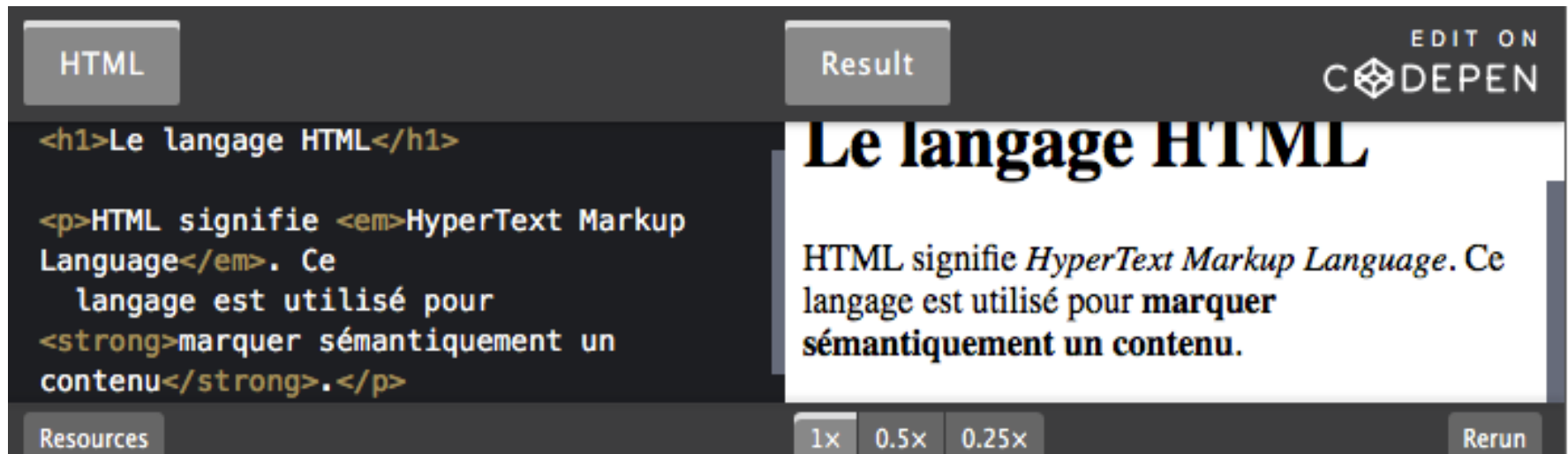
- L'entité HTML ` ` (« non breaking space ») va nous permettre d'ajouter une espace simple dit espace « insécable » ;
- L'entité HTML ` ` (« en space ») va nous permettre de créer une espace double ;
- L'entité HTML ` ` (« em space ») va nous permettre de créer une espace quadruple ;
- L'entité HTML ` ` (« thin space ») va nous permettre de créer un espace très fin (demi-espace).
- `&` `&` Interprété comme le début d'une référence d'entité ou de caractère.
- `<` `<` Interprété comme le début d'une balise
- `>` `>` Interprété comme la fin d'une balise
- `"` `"` Interprété comme le début et la fin d'une valeur d'attributs

Unité 1 : Les bases du HTML

L'élément HTML **strong** va être utilisé pour signifier qu'un contenu est très important et doit être considéré comme tel par les moteurs de recherche (et les navigateurs).

L'élément HTML **em** (pour emphasis ; « emphase » ou « accentuation » en français) va être utilisé pour mettre des termes en emphase.

Exemple :



The screenshot shows a live HTML editor interface. On the left, a dark-themed code editor displays the following HTML code:

```
<h1>Le langage HTML</h1>

<p>HTML signifie <em>HyperText Markup Language</em>. Ce langage est utilisé pour <strong>marquer sémantiquement un contenu</strong>.</p>
```

On the right, the rendered output is shown in a light-themed preview window. It features a large heading "Le langage HTML" and a paragraph: "HTML signifie *HyperText Markup Language*. Ce langage est utilisé pour **marquer sémantiquement un contenu**." The text "marquer sémantiquement un contenu" is bolded, and "HyperText Markup Language" is italicized. The interface includes a "Result" tab, a "Resources" tab, and a "Rerun" button. The top right corner has "EDIT ON CODEPEN" and the bottom right corner has zoom controls (1x, 0.5x, 0.25x).

Unité 1 : Les bases du HTML

Les listes HTML

Il existe deux grands types de listes en HTML : les listes ordonnées et les listes non-ordonnées.

L'**élément li** nous permet de créer des listes en HTML. Selon qu'elle soit ordonnée ou non ordonnée, on rajoutera les éléments **ol** pour ordonnée et **ul** pour non ordonnée.

Par exemple :

Liste non ordonnée ()



The screenshot shows a CodePen editor interface. On the left, the 'HTML' tab is active, displaying the following code:

```
<h1>Les listes</h1>

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

On the right, the 'Result' tab shows the rendered output:

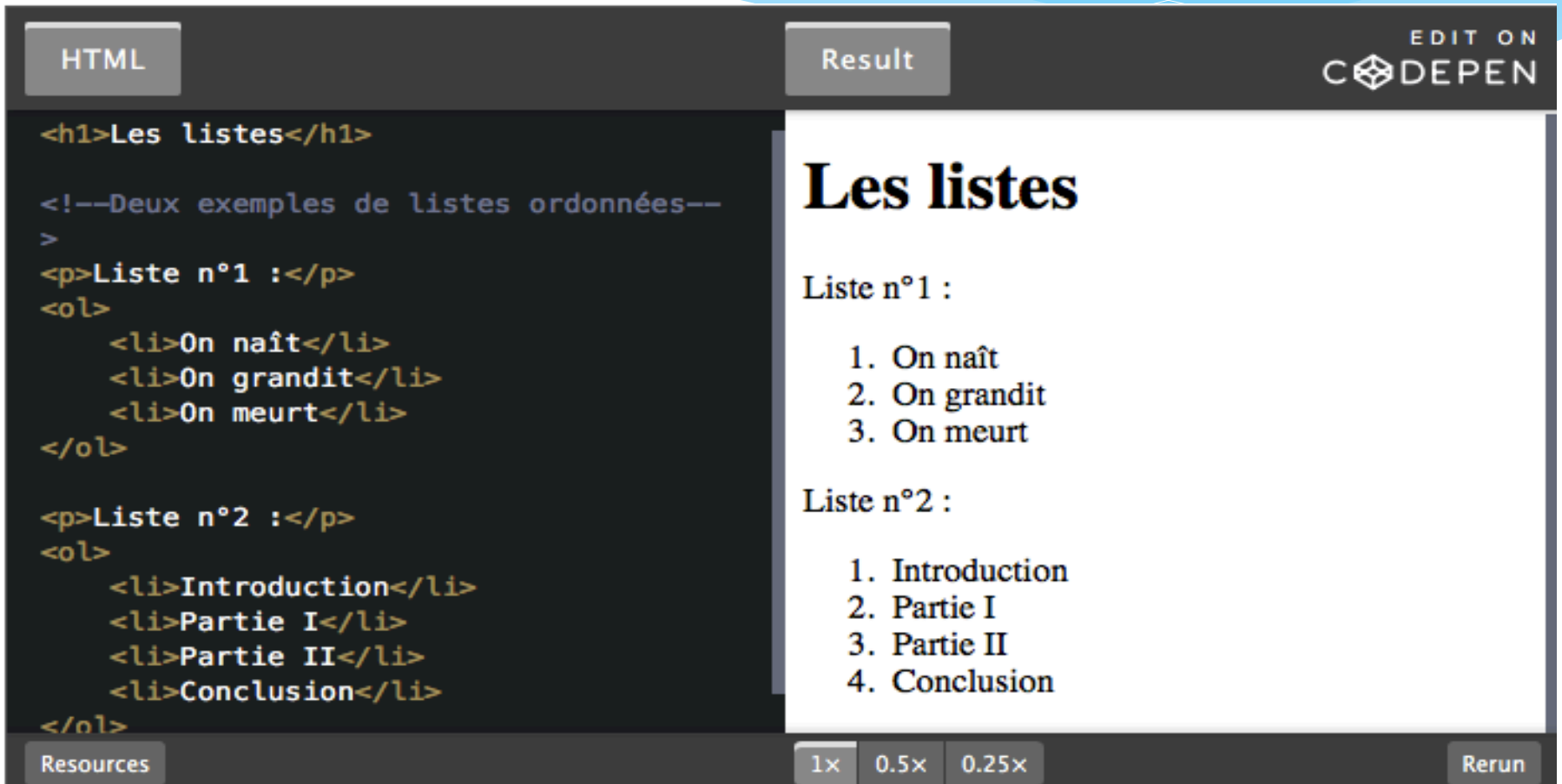
Les listes

- Pomme
- Vélo
- Guitare

The interface also includes a 'Resources' button at the bottom left, zoom controls (1x, 0.5x, 0.25x) at the bottom center, and a 'Rerun' button at the bottom right. The CodePen logo and 'EDIT ON CODEPEN' text are visible in the top right corner.

Unité 1 : Les bases du HTML

Liste ordonnée ()



The screenshot shows a web editor interface with two main panels: 'HTML' on the left and 'Result' on the right. The 'HTML' panel contains the following code:

```
<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>
```

The 'Result' panel displays the rendered output:

Les listes

Liste n°1 :

1. On naît
2. On grandit
3. On meurt

Liste n°2 :

1. Introduction
2. Partie I
3. Partie II
4. Conclusion

The interface also includes a 'Resources' button at the bottom left, zoom controls (1x, 0.5x, 0.25x) at the bottom center, and a 'Rerun' button at the bottom right. The top right corner features the text 'EDIT ON CODEPEN'.

Unité 1 : Les bases du HTML

L'attribut target de l'élément a

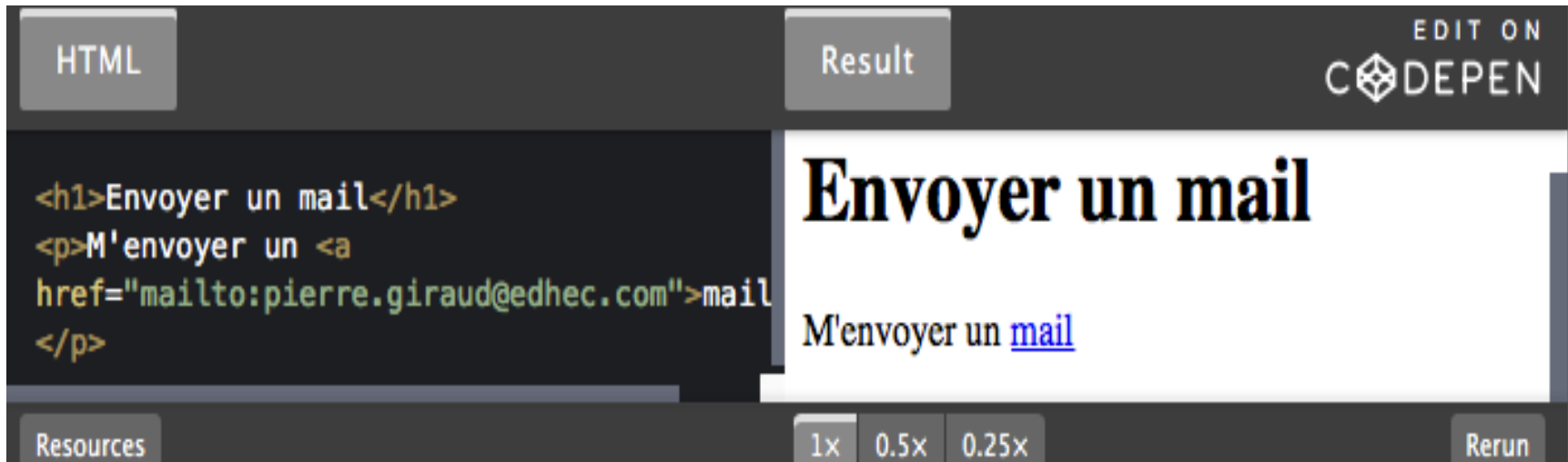
Pour rappel , l'élément a nous permet de créer des liens en HTML avec l'attribut **href**. L'attribut **target** vient s'ajouter à la liste des attributs de l'élément a et en fonction de sa valeur, il nous permet de :

Valeur de target	Comportement
_self	Valeur par défaut : la page cible s'ouvre dans le même emplacement (cadre ou « frame ») que là où l'utilisateur a cliqué
_blank	La page cible s'ouvre dans un nouvel onglet ou dans une nouvelle fenêtre
_parent	La page cible s'ouvre dans la cadre (frame) de niveau immédiatement supérieur par rapport à l'emplacement du lien
_top	La page cible s'ouvre dans la fenêtre hôte (par dessus le frameset)
Nom du cadre (frame)	Ouverture de la page cible dans le cadre portant le nom cité (en valeur de l'attribut name)

Unité 1 : Les bases du HTML

On peut utiliser l'élément `a` pour transmettre notre adresse mail à nos utilisateurs et leur permettre de nous envoyer simplement un mail.

Pour permettre l'envoi d'un mail en HTML, on va placer en valeur de l'attribut `href` de notre élément `a` la valeur `mailto` : suivie de notre adresse email.



The screenshot shows a web editor interface with two main panels: 'HTML' on the left and 'Result' on the right. The 'HTML' panel contains the following code:

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

The 'Result' panel shows the rendered output: a large heading 'Envoyer un mail' and a paragraph 'M'envoyer un [mail](mailto:pierre.giraud@edhec.com)'. The interface also includes a 'Resources' button at the bottom left, zoom controls (1x, 0.5x, 0.25x) at the bottom center, and a 'Rerun' button at the bottom right. The top right corner features the text 'EDIT ON CODEPEN'.

Unité 1 : Les bases du HTML

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 ce faire, il faut utiliser l'attribut **download** dans l'élément `a` tout en indiquant l'adresse du fichier en question en valeur de l'attribut **href**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première page HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Cours HTML</h1>
    <!--On imagine qu'on possède une image nommée "logo.png" située dans le
    même dossier que notre page HTML-->
    <p>Mon <a href="logo.png" download="pg-logo">logo</a> à télécharger</p>
  </body>
</html>
```

Unité 1 : Les bases du HTML

Pour aller plus loin avec les éléments et balises html , vous pouvez retrouver une liste plus ou moins complète via le lien suivant :

<https://jaetheme.com/balises-html5/>

Unité 1 : Les bases du HTML

Exercice 1

Ecrire un cv sous la forme d'une page web dans laquelle on trouve:

- Votre Etat civil;
- Votre (une) photo avec une option de téléchargement;
- Une section Formation dans laquelle on trouvera
 - Une sous section qui liste vos compétences (par exemple Informatique, Médecine, etc.)
 - Pour chaque compétences faire une liste ordonnée et une liste non ordonnée qui détails les différentes compétences.

Unité 2 : Les tableaux HTML

Les tableaux en HTML vont nous permettre de présenter des données de manière organisée et sous une certaine forme pour les structurer et les rendre compréhensibles pour les navigateurs, moteurs de recherche et utilisateurs.

Un tableau est un ensemble de lignes et de colonnes. L'intersection entre une ligne et une colonne est une cellule de tableau.

Pour créer un tableau fonctionnel en HTML, nous avons besoin à minima de 3 éléments :

Un élément table (« tableau » en français) qui va définir le tableau en soi ;

Des éléments tr, pour « table row » ou « ligne de tableau » en français qui vont nous permettre d'ajouter des lignes dans notre tableau ;

Des éléments td, pour « table data » ou « donnée de tableau » en français qui vont nous permettre d'ajouter des cellules dans nos lignes et ainsi de créer automatiquement de nouvelles colonnes.

Unité 2 : Les tableaux HTML

L'élément HTML **table** va représenter le tableau en soi. Cet élément est composé d'une paire de balises ouvrante **<table>** et fermante **</table>** au sein desquelles nous allons placer les différents autres éléments de notre tableau. Les éléments **tr** et **td** sont également représentés sous la forme d'une paire de balises avec leur contenu entre les balises.

Retenez qu'il est considéré comme une mauvaise pratique de créer des lignes de tableau avec un nombre de cellules différent dans chacune. En pratique, nous attribuerons toujours un même nombre de td à nos différentes lignes. Si on souhaite laisser une cellule vide, nous nous contenterons d'écrire `<td> </td>` sans rien écrire entre les balises.

Par exemple, pour créer un tableau en HTML contenant 3 lignes contenant chacune 4 cellules (c'est-à-dire un tableau de 3 lignes, 4 colonnes), nous utiliserons notre élément `table` qui va contenir 3 éléments `tr` et chaque élément `tr` contiendra 4 éléments `td` comme ceci:

Unité 2 : Les tableaux HTML

HTML CSS Result EDIT ON CODEPEN

```
<table>
  <tr>
    <td>Nom</td>
    <td>Prénom</td>
    <td>Age</td>
    <td>Mail</td>
  </tr>
  <tr>
    <td>Giraud</td>
    <td>Pierre</td>
    <td>28</td>
    <td>pierre.giraud@edhec.com</td>
  </tr>
  <tr>
    <td>Joly</td>
    <td>Pauline</td>
    <td>27</td>
    <td>pjl@gmail.com</td>
  </tr>
</table>
```

Nom	Prénom	Age	Mail
Giraud	Pierre	28	pierre.giraud@edhec.co
Joly	Pauline	27	pjl@gmail.com

Resources 1x 0.5x 0.25x Rerun

Unité 2 : Les tableaux HTML

Les attributs **colspan** et **rowspan**

Ce sont des attributs qui vont nous permettre de fusionner plusieurs cellules adjacentes d'une même ligne ou d'une même colonne.

Ces attributs vont prendre en valeur le nombre de colonnes ou de lignes qu'une cellule doit couvrir, c'est-à-dire le nombre de colonnes ou de lignes qu'une cellule doit occuper ou encore sur lesquelles elle doit s'étendre.

L'attribut **caption**

Pour ajouter une légende, nous allons utiliser l'élément HTML `caption`. Cet élément est très simple d'utilisation, mais il faut respecter une règle : il doit être inséré immédiatement après la balise ouvrante de l'élément `table`.

Unité 2 : Les tableaux HTML

Exercice 2

- Reprendre l'exercice 1 et rajouter un tableau de quatre lignes , 3 colonnes dans lequel vous insérerez vos loisirs.
- Nommez le tableau << Mes Loisirs >>

Unité 3 : Les Formulaires HTML

Les Formulaires HTML

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

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

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

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

Unité 3 : Les Formulaires HTML

L'élément form et ses attributs

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

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

Les valeurs get et post vont déterminer la méthode de transit des données du formulaire. En choisissant get, on indique que les données doivent transiter via l'URL (sous forme de paramètres) tandis qu'en choisissant la valeur post on indique qu'on veut envoyer les données du formulaire via transaction post HTTP.

L'attribut **action** va lui nous servir à préciser l'adresse de la page qui va traiter les données.

Unité 3 : Les Formulaires HTML

Création des champs de formulaire avec l'élément input

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

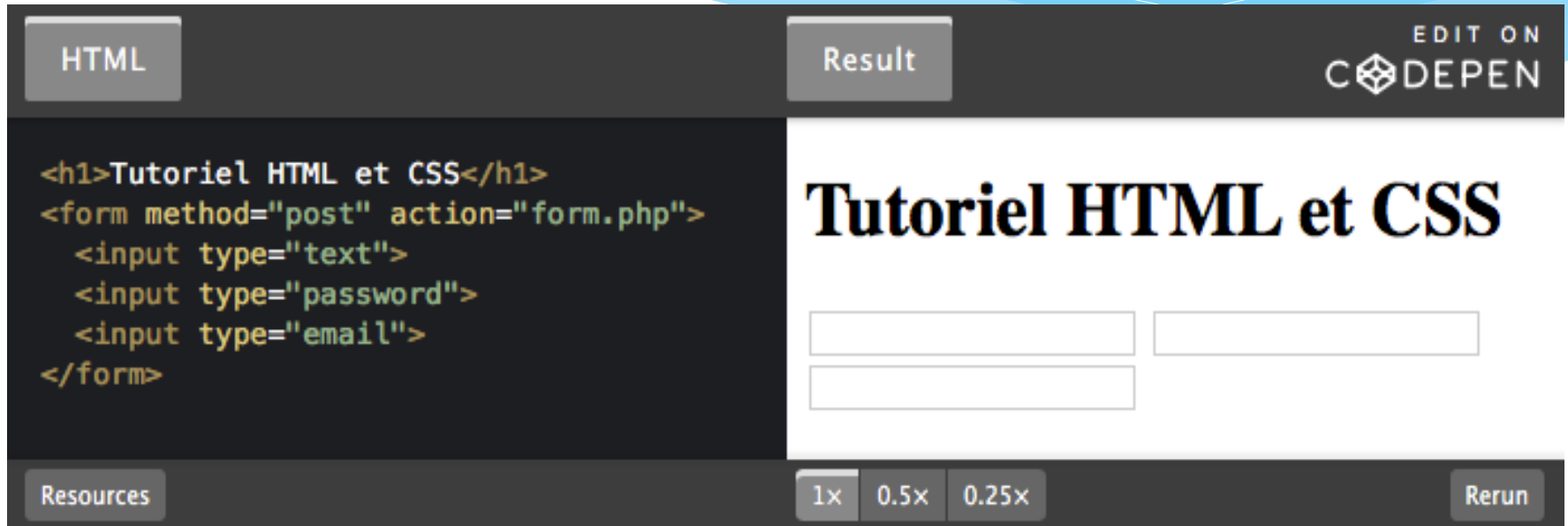
L'**élément HTML input** est un élément qui va permettre à l'utilisateur d'envoyer des données. Il se présente sous la forme d'une **balise orpheline** et va obligatoirement posséder un **attribut type** auquel on va pouvoir donner de nombreuses valeurs.

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

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

Unité 3 : Les Formulaires HTML

Notre formulaire ressemblera à ça:



The screenshot shows a CodePen editor interface. On the left, the 'HTML' tab is active, displaying the following code:

```
<h1>Tutoriel HTML et CSS</h1>
<form method="post" action="form.php">
  <input type="text">
  <input type="password">
  <input type="email">
</form>
```

On the right, the 'Result' tab shows the rendered output: a large heading 'Tutoriel HTML et CSS' followed by three empty input fields (one text, one password, one email) arranged in two rows. The CodePen logo and 'EDIT ON CODEPEN' are visible in the top right corner. At the bottom, there are buttons for 'Resources', zoom levels '1x', '0.5x', '0.25x', and a 'Rerun' button.

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

Unité 3 : Les Formulaires HTML

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

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

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

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

Mettons tout ça en pratique dans notre précédent formulaire. On aura donc

Unité 3 : Les Formulaires HTML

The screenshot shows a web editor interface. On the left, the HTML code is displayed in a dark theme. On the right, the rendered result is shown in a light theme. The code defines a form with three input fields, each wrapped in a `div` with the class `champ`. The rendered output shows the form with the title "Tutoriel HTML et CSS" and three input fields: "Entrez un pseudonyme :", "Entrez un mot de passe :", and "Entrez un mail :".

```
<h1>Tutoriel HTML et CSS</h1>
<form method="post" action="form.php">
  <div class="champ">
    <label for="pseudo">Entrez un
pseudonyme : </label>
    <input type="text" id="pseudo">
  </div>
  <div class="champ">
    <label for="pass">Entrez un mot de
passe :</label>
    <input type="password" id="pass">
  </div>
  <div class="champ">
    <label for="mail">Entrez un mail :
</label>
    <input type="email" id="mail">
  </div>
</form>
```

Resources 1x 0.5x 0.25x Rerun

Ici, les couples label + input ont été placé dans des div afin que chacun d'entre eux soit sur une ligne à eux .

Unité 3 : Les Formulaires HTML

Ajout du bouton d'envoi de données

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

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

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

Notre formulaire sera comme suit :

Unité 3 : Les Formulaires HTML

The screenshot shows a CodePen editor with the following components:

- HTML Tab:** Contains the following code:

```
<h1>Tutoriel HTML et CSS</h1>
<form method="post" action="form.php">
  <div class="champ">
    <label for="pseudo">Entrez un
pseudonyme : </label>
    <input type="text" id="pseudo"
name="pseudo">
  </div>
  <div class="champ">
    <label for="pass">Entrez un mot de
passe :</label>
    <input type="password" id="pass"
name="pass">
  </div>
  <div class="champ">
    <label for="mail">Entrez un mail :
</label>
    <input type="email" id="mail"
name="mail">
  </div>
  <div class="champ">
    <input type="submit"
value="Envoyer">
  </div>
```
- CSS Tab:** Empty.
- Result Tab:** Shows the rendered output:
 - Header: **Tutoriel HTML et CSS**
 - Form fields:
 - Entrez un pseudonyme :
 - Entrez un mot de passe :
 - Entrez un mail :
 - Submit button:
- Footer:** Includes "Resources", zoom controls (1x, 0.5x, 0.25x), and a "Rerun" button.

Unité 3 : Les Formulaires HTML

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

Exemple

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

Unité 3 : Les Formulaires HTML

L'attribut **type** peut prendre de nombreuses valeurs. Voici les valeurs les plus utiles et les plus courantes :

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

Unité 3 : Les Formulaires HTML

L'élément `textarea`

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

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

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

Les éléments `select`, `option` et `optgroup`

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

Unité 3 : Les Formulaires HTML

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

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

Les éléments **fieldset** et **legend**

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

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

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

Unité 3 : Les Formulaires HTML

Quelques attributs de l'élément input

Attribut	Définition
size	Permet de spécifier le nombre de caractères dans un champ
minlength	Permet de spécifier le nombre minimum de caractères dans un champ
maxlength	Permet de spécifier le nombre maximum de caractères dans un champ
min	Permet de spécifier une valeur minimale pour un champ de type number ou date
max	Permet de spécifier une valeur maximale pour un champ de type number ou date

Unité 3 : Les Formulaires HTML

Quelques attributs de l'élément input

step	Permet de définir un multiple de validité pour un champ acceptant des données de type nombre ou date. En indiquant <code>step="4"</code> , les nombres valides seront -8, -4, 0, 4, 8, etc.
autocomplete	Permet d'activer l'auto complétion pour un champ : si un utilisateur a déjà rempli un formulaire, des valeurs lui seront proposées automatiquement lorsqu'il va commencer à remplir le champ
required	Permet de forcer le remplissage d'un champ. Le formulaire ne pourra pas être envoyé si le champ est vide
pattern	Permet de préciser une expression régulière. La valeur du champ devra respecter la contrainte de la regex pour être valide

Unité 3 : Les Formulaires HTML

Exercices

Reprendre l'exercice 2 et rajouter un formulaire de contact dans lequel on pourra récupérer le nom, prénom, la date de naissance, l'adresse mail, un mot de passe et un commentaire.

Unité 4 : Les bases du CSS

Le CSS est un langage qui a été inventé pour styliser les contenus de nos pages en leur appliquant des styles.

Les sélecteurs CSS

Pour pouvoir appliquer un style à un contenu, il va déjà falloir le cibler, c'est-à-dire trouver un moyen d'indiquer qu'on souhaite appliquer tel style à un contenu en particulier.

Pour cela, nous allons utiliser des sélecteurs. Un sélecteur CSS est un mot-clef qui permet de désigner une catégorie d'éléments de la page éventuellement de nature différente ou une relation entre deux éléments.

Les propriétés CSS : définition

Les propriétés vont nous permettre de choisir quel(s) aspect(s) (ou "styles") d'un élément HTML on souhaite modifier.

Par exemple, nous allons pouvoir modifier la couleur d'un texte et lui appliquer la couleur que l'on souhaite grâce à la propriété **color** (« couleur », en français).

Une propriété va être accompagnée d'une ou plusieurs valeurs qui vont définir le comportement de cette propriété.

Unité 4 : Les bases du CSS

Par exemple, la propriété `color` peut prendre le nom d'une couleur (en anglais). Si l'on donne la valeur `red` (rouge) à notre propriété `color`, les textes au sein des éléments HTML auxquels on applique cette propriété s'afficheront en rouge.

Les déclarations CSS

Prenons immédiatement un premier exemple ensemble en expliquant bien à quoi correspond chaque élément du code afin d'illustrer ce que nous venons de dire et de bien voir comment le CSS fonctionne.

```
p{  
  color: blue;  
  border: 2px solid orange;  
  padding: 5px;  
}
```

Détaillons le code CSS ci-dessus. Ici, nous utilisons le sélecteur CSS simple `p` pour cibler tous les paragraphes de nos pages HTML. Ensuite, nous ouvrons une paire d'accolades. Entre ces accolades, nous allons préciser les différents styles que l'on souhaite appliquer à nos éléments `p`.

Unité 4 : Les bases du CSS

En l'occurrence, on définit une couleur, bordure et une marge interne personnalisées pour tous nos paragraphes grâce aux propriétés CSS `color`, `border` et `padding`.

Le texte de nos paragraphes va donc s'afficher en bleu et nos paragraphes auront des bordures solides oranges de 2px d'épaisseur et des marges internes de 5px.

Le couple « propriété : valeur » est appelée « déclaration » en CSS. Chaque déclaration doit se terminer par un point-virgule.

On va pouvoir écrire autant de déclarations que l'on souhaite à l'intérieur du couple d'accolades qui suit un sélecteur en CSS et ainsi pouvoir définir le comportement de plusieurs propriétés facilement.

Unité 4 : Les bases du CSS

Où écrire le code CSS ?

Avant d'étudier les mécanismes du CSS en soi, il convient de comprendre où placer le code CSS afin qu'il s'applique bien à un fichier HTML.

Dans le cas présent, nous avons notre code HTML d'un côté et nous aimerions lui appliquer des styles en CSS. Cependant, il va falloir d'une manière ou d'une autre « lier » notre code CSS à notre code HTML afin que les éléments de nos pages HTML tiennent bien compte des styles qu'on a voulu leur appliquer en CSS.

Pour faire cela, nous allons pouvoir écrire le code CSS à trois endroits différents. Chaque méthode va présenter des avantages et des inconvénients selon une situation donnée et c'est le sujet que nous allons aborder dans cette leçon.

Méthode n°1 : écrire le CSS au sein du fichier HTML, dans un élément style

La première façon d'écrire du code CSS va être à l'intérieur même de notre page HTML, au sein d'un élément style.

En plaçant le CSS de cette façon, le code CSS ne s'appliquera qu'aux éléments de la page HTML dans laquelle il a été écrit.

Unité 4 : Les bases du CSS

Cette première méthode d'écriture du CSS n'est pas recommandée, pour des raisons de maintenance et d'organisation du code en général. Cependant, elle peut s'avérer utile pour modifier rapidement les styles d'une page HTML ou si vous n'avez pas facilement accès aux fichiers de style de votre site. Ci-dessous un exemple :



The screenshot shows a web editor interface with two main panels: 'HTML' on the left and 'Result' on the right. The 'HTML' panel contains the following code:

```
<head>
  <title>Où écrire le CSS ?</title>
  <meta charset= "utf-8">

  <style>
    body{
      background-color: orange;
    }
    p{
      color: blue;
      font-size: 16px;
    }
  </style>
</head>

<body>
  <h1>Un titre de niveau 1</h1>
  <p>Un paragraphe</p>
  <p>Un deuxième paragraphe</p>
</body>
```

The 'Result' panel shows the rendered output: a page with an orange background, a large black heading 'Un titre de niveau 1', and two blue paragraphs. The editor interface includes a 'Resources' tab at the bottom left, zoom controls (1x, 0.5x, 0.25x) at the bottom center, and a 'Rerun' button at the bottom right. The top right corner of the editor has the text 'EDIT ON CODEPEN'.

Unité 4 : Les bases du CSS

Méthode n°2 : déclarer le CSS au sein du fichier HTML, dans des attributs style

Nous pouvons également écrire notre code CSS au sein d'attributs style qu'on va ajouter à l'intérieur de la balise ouvrante des éléments HTML pour lesquels on souhaite modifier les styles.

Nous allons passer en valeurs des attributs style des déclarations CSS pour modifier certains styles précis de l'élément HTML en question. En effet, en utilisant cette méthode, les styles déclarés dans un attribut style ne vont s'appliquer qu'à l'élément dans lequel ils sont écrits, et c'est la raison pour laquelle nous n'allons pas avoir besoin de préciser de sélecteur ici.



The screenshot shows a web editor interface with two main panels: 'HTML' on the left and 'Result' on the right. The 'HTML' panel contains the following code:

```
<body style="background-color:orange;">
  <h1>Un titre de niveau 1</h1>
  <p style="color: blue;font-size: 20px;">Un paragraphe</p>
  <p>Un deuxième paragraphe</p>
</body>
```

The 'Result' panel shows the rendered output: a yellow background, a large bold black heading 'Un titre de niveau 1', a blue paragraph 'Un paragraphe', and a smaller black paragraph 'Un deuxième paragraphe'. The editor includes a 'Resources' button at the bottom left, zoom controls (1x, 0.5x, 0.25x) at the bottom center, and a 'Rerun' button at the bottom right. The top right corner has the text 'EDIT ON CODEPEN'.

Unité 4 : Les bases du CSS

Cette deuxième méthode d'écriture du CSS, bien qu'elle puisse sembler pratique à priori puisqu'elle permet de n'appliquer des styles qu'à un élément en particulier plutôt qu'à tous les éléments d'un même type n'est également pas recommandée et est à éviter tant que possible pour des raisons de maintenabilité et de performance du code.

Méthode n°3 : écrire le CSS dans un fichier séparé

Finalement, nous pouvons écrire notre code CSS dans un fichier séparé portant l'extension « .css ». C'est la méthode recommandée, qui sera utilisée autant que possible.

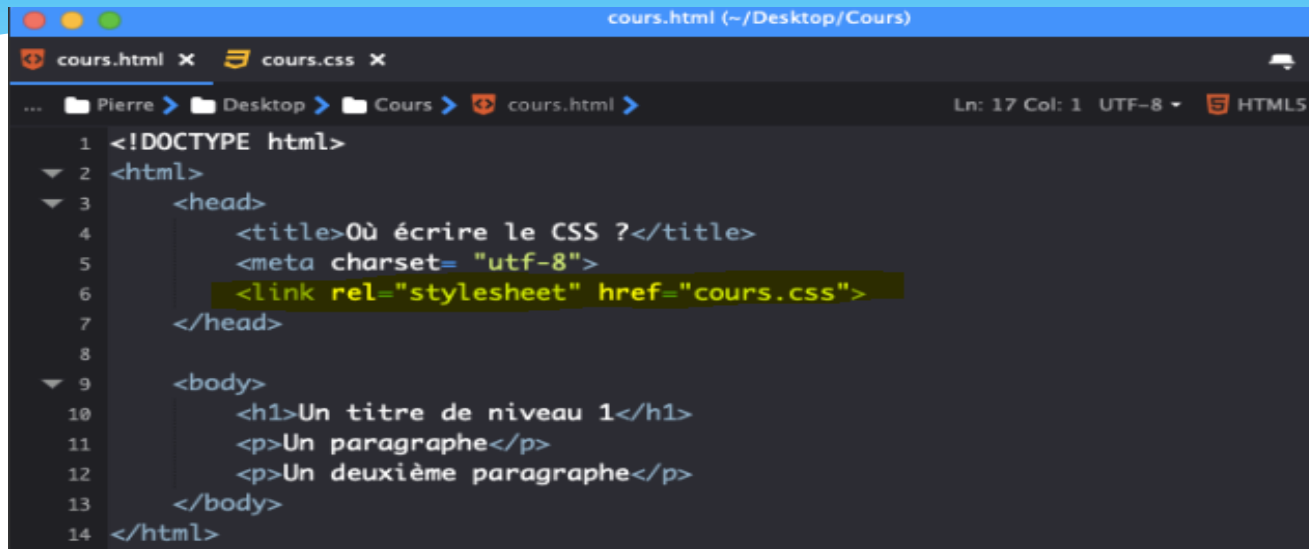
Cette méthode comporte de nombreux avantages, notamment une meilleure maintenabilité du code grâce à la séparation des différents langages, ainsi qu'une meilleure lisibilité. Le plus gros avantage de cette méthode est qu'on va pouvoir appliquer des styles à plusieurs pages HTML en même temps, d'un seul coup. Pour cela, il faut créer un nouveau fichier dans notre éditeur (**cours.css** par exemple). Nous allons enregistrer ce fichier et le placer dans le même dossier que notre page HTML pour plus de simplicité.

Unité 4 : Les bases du CSS

Ensuite, dans notre fichier html, nous allons utiliser un nouvel élément HTML : l'élément **link** (« lien », en français). On va placer l'élément **link** au sein de l'élément **head** de notre fichier HTML. Cet élément se présente sous la forme d'une balise orpheline et va avoir besoin de deux attributs pour fonctionner correctement :

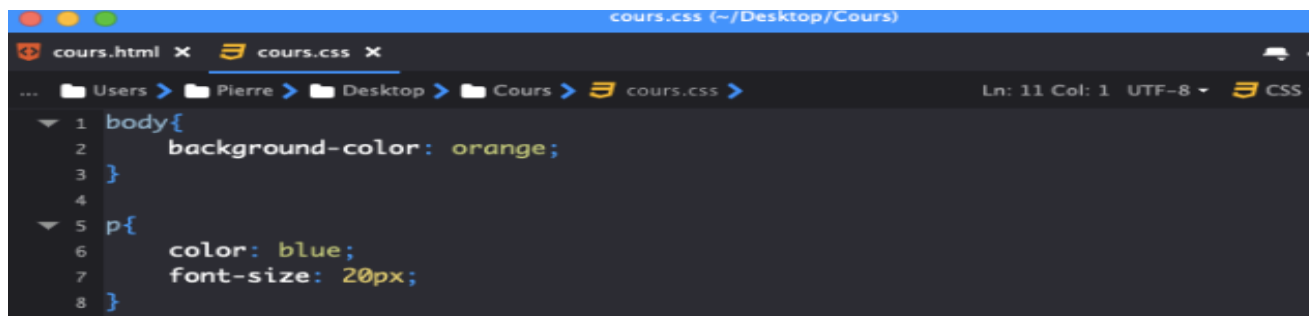
- ❑ Un attribut **rel** qui va nous servir à préciser le type de ressource que l'on souhaite lier à notre fichier HTML. Dans notre cas, nous indiquerons la valeur stylesheet pour « feuille de style » ;
- ❑ Un attribut **href** qui va indiquer l'adresse relative de la ressource que l'on souhaite lier par rapport à l'emplacement de notre fichier HTML. Ici, comme nous avons enregistré nos deux fichiers dans le même dossier, il suffira d'indiquer le nom de notre fichier CSS en valeur de href.

Unité 4 : Les bases du CSS



```
cours.html (~/Desktop/Cours)
cours.html x cours.css x
... Pierre > Desktop > Cours > cours.html > Ln: 17 Col: 1 UTF-8 HTML5
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Où écrire le CSS ?</title>
5     <meta charset= "utf-8">
6     <link rel="stylesheet" href="cours.css">
7   </head>
8
9   <body>
10    <h1>Un titre de niveau 1</h1>
11    <p>Un paragraphe</p>
12    <p>Un deuxième paragraphe</p>
13  </body>
14 </html>
```

Nos deux fichiers sont maintenant liés et les styles déclarés dans notre fichier CSS vont bien être appliqués aux éléments de notre page HTML.



```
cours.css (~/Desktop/Cours)
cours.html x cours.css x
... Users > Pierre > Desktop > Cours > cours.css > Ln: 11 Col: 1 UTF-8 CSS
1 body{
2   background-color: orange;
3 }
4
5 p{
6   color: blue;
7   font-size: 20px;
8 }
```


Unité 4 : Les bases du CSS

Tout comme nous avons vu qu'on pouvait écrire des commentaires en HTML, nous allons également pouvoir commenter notre code CSS. Tout comme nous avons vu qu'on pouvait écrire des commentaires en HTML, nous allons également pouvoir commenter notre code CSS.

Le CSS, tout comme le HTML et à la différence d'autres langages de développement ne possède qu'une seule syntaxe qui va nous permettre de créer à la fois des commentaires mono-ligne et multi-lignes.

Cette syntaxe est la suivante : **/*Un commentaire CSS*/**. Regardez plutôt l'exemple ci-dessous

```
/*Un premier commentaire CSS*/
body{
  background-color: orange;
}

/*Un deuxième commentaire
*sur plusieurs
*lignes*/
p{
  /*color: blue;*/
  font-size: 20px;
}
```

les étoiles en début de ligne pour mon commentaire multi-lignes ne sont absolument pas nécessaires (à part pour la première ligne, évidemment) : ce n'est que de la décoration afin de bien voir que l'on commente,

Unité 4 : Les bases du CSS

Attributs HTML class et id

Les attributs HTML **class** et **id** sont des attributs dits globaux car on va pouvoir les ajouter dans la balise ouvrante de n'importe quel élément HTML.

Ces deux attributs vont être principalement utilisés dans un but de mise en forme : ils vont nous servir à appliquer des styles CSS aux éléments qui vont les contenir.

Nous allons très facilement pouvoir nous resservir de ces deux attributs en CSS grâce aux sélecteurs associés **.class** et **#id**.

Pour cibler un id en particulier en CSS, on utilisera le symbole **dièse #** suivi de la valeur de l'id auquel on souhaite lier des styles.

Pour cibler une class en particulier en CSS, on utilisera le symbole **point .** suivi de la valeur de la class à laquelle on souhaite lier des styles.

Il existe une différence notable entre les deux attributs class et id : **chaque id** doit avoir une **valeur unique** dans une même page tandis que plusieurs attributs **class** peuvent partager la **même valeur**.

Les attributs class sont utilisés pour définir des styles généraux et communs à plusieurs éléments dans une même page.

En revanche, comme chaque id doit être unique dans une page, nous utiliserons ce sélecteur pour appliquer des styles très précis et pour être sûr de ne cibler qu'un élément HTML en CSS.

Unité 4 : Les bases du CSS

```
HTML CSS
<h1 id="orange">Un titre de niveau
1</h1>
<p class="bleu">Un premier
paragraphe</p>
<p class="bleu">Un autre paragraphe
avec un <a href="#">lien</a></p>
<ul>
  <li>Élément de liste 1</li>
  <li class="vert">Élément de liste
2</li>
</ul>
<p class="bleu" id="gros">Un troisième
paragraphe</p>
Resources
```

```
HTML CSS
#orange{
  color: orange;
}
#gros{
  font-size: 24px;
}
.bleu{
  color: blue;
}
.vert{
  color: green;
}
Resources
```

```
Result EDIT ON
CODEPEN
Un titre de niveau 1
Un premier paragraphe
Un autre paragraphe avec un lien
• Élément de liste 1
• Élément de liste 2
Un troisième paragraphe
67
1x 0.5x 0.25x Rerun
```

Unité 4 : Les bases du CSS

Le mécanisme de cascade CSS

Il est essentiel de bien comprendre comment le CSS va fonctionner pour déterminer quels styles devront être appliqués à tel élément. L'ordre de préférence et d'application d'un style va dépendre de trois grands facteurs qui vont être :

- ❑ La présence ou non du mot clef **!important** ;
 - ❑ La précision du sélecteur ;
 - ❑ L'ordre de déclaration dans le code ;
- La **mot clef !important** sert à forcer l'application d'une règle CSS. La règle en question sera alors considérée comme prioritaire sur toutes les autres déclarations et le style sera appliqué à l'élément concerné.

```
p{  
    color: red !important;  
    text-decoration: underline;  
}
```

- La « précision » désigne ici le fait d'identifier de manière plus ou moins unique un élément

Unité 4 : Les bases du CSS

Les sélecteurs peuvent être rangés dans l'ordre suivant (du plus précis au moins précis) :

- ❑ Un style défini dans un attribut HTML **style** sera toujours le plus précis et notamment plus précis qu'un style défini avec un sélecteur CSS ;
- ❑ Le sélecteur **#id** va être le sélecteur le plus précis mais sera moins précis qu'un style défini dans un attribut HTML style ;
- ❑ Un sélecteur **.class** ou un autre sélecteur d'attribut sera moins précis qu'un sélecteur #id ;

Unité 4 : Les bases du CSS

Attributs HTML div et span

Ces éléments sont très particuliers puisqu'ils ne servent pas à préciser la nature d'un contenu mais vont simplement nous servir de conteneurs génériques en HTML.

Les éléments HTML **div** et **span** ont été créés principalement pour simplifier la mise en page de nos pages HTML en CSS c'est-à-dire pour simplifier l'application de certains styles CSS.

L'élément div est utilisé comme conteneur pour différents éléments afin de pouvoir ensuite facilement appliquer les mêmes styles CSS à tous les éléments contenus dans notre div par héritage ou pour les mettre en forme en appliquant un style spécifique au div

L'élément span va lui servir de conteneur à un autre niveau : il va servir de conteneur interne à un élément plutôt que de conteneur pour plusieurs éléments.

Les éléments div et span peuvent être utilisés avec les attributs **id** et **class**

Unité 4 : Les bases du CSS

Les propriétés CSS

Il existe plusieurs propriétés possibles applicables aux éléments CSS. On a :

- Propriétés de formatage de texte (Police, taille, décorations, alignement,etc.);
- Propriétés de couleur et de fond (couleur, image de fond, etc);
- Propriétés des boîtes (Dimensions, Bordures, Marge,etc);
- Propriétés de positionnement et d'affichage (Affichage, positionnement, etc);
- Propriétés des listes;
- Propriétés des tableaux, etc.

A la suite de ce cours, vous aurez un document récapitulatif des propriétés CSS.

Unité 4 : Les bases du CSS

Exercices

Conclusion