R1.02 : Développement d'interfaces web

C. Rozé

Organisation

- CM : 1 séance
- TD : 3 séances d'1h30
- TP : 2 séances de 3h + 1 TP contrôle d'1h30
- DS : commun avec la partie Analyse-Conception

Programme National

Descriptif détaillé

Objectif
L'objectif de cette ressource est d'apprendre les techniques de création de documents numériques sur le web en réponse à des besoins client. Cette ressource est une base pour réaliser un développement d'application tout en appréhendant les besoins du client et de l'utilisateur.
Savoirs de référence étudiés
– Spécifications d'interfaces utilisateur, maquettage (sketch, scénarios, persona…)
– Technologies d'affichage du Web (par ex. : Hypertext Markup Language (`HTML`), Cascading Style Sheet (`CSS`)…)
– Test de la conformité des sites Web aux standards d'accessibilité World Wide Web Consortium (W3C) / Web Accessibility Initiative (WAI)
Prolongements suggérés
– Génération de documents numériques

interaction humain-machine (IHM) | Front web | Maquettage

Programme National

Descriptif détaillé

Objectif

L'objectif de cette ressource est d'apprendre les techniques de création de documents numériques sur le web en réponse à des besoins client. Cette ressource est une base pour réaliser un développement d'application tout en appréhendant les besoins du client et de l'utilisateur.

Savoirs de référence étudiés

- Spécifications d'interfaces utilisateur, maquettage (sketch, scénarios, persona…)
- Technologies d'affichage du Web (par ex. : Hypertext Markup Language (`HTML`), Cascading Style Sheet (`CSS`)…)
- Test de la conformité des sites Web aux standards d'accessibilité World Wide Web Consortium (W3C) / Web Accessibility Initiative (WAI)

Prolongements suggérés

- Génération de documents numériques

interaction humain-machine (IHM) | Front web | Maquettage

Programme National

Descriptif détaillé

Objectif
L'objectif de cette ressource est d'apprendre les techniques de création de documents numériques sur le web en réponse à des besoins client. Cette ressource est une base pour réaliser un développement d'application tout en appréhendant les besoins du client et de l'utilisateur.

Savoirs de référence étudiés
– Spécifications d'interfaces utilisateur, maquettage (sketch, scénarios, persona…)
– Technologies d'affichage du Web (par ex. : Hypertext Markup Language (HTML), Cascading Style Sheet (CSS)…)
– Test de la conformité des sites Web aux standards d'accessibilité World Wide Web Consortium (W3C) / Web Accessibility Initiative (WAI)

Prolongements suggérés
– Génération de documents numériques

interaction humain-machine (IHM) | Front web | Maquettage

Ce que vous ne verrez pas dans cette ressource :

Web côté serveur (PHP...) --> S3

Web côté client (JS, AJAX...) --> S4

Technologie d'affichage du Web

- **Web ?**

  Le Web (ou World Wide Web, www) désigne l'*ensemble des sites Web visitables depuis un navigateur* grâce à un système d'adresses URL (Uniform Ressource Locator).

  Les pages Web sont des documents écrits en **HTML** (langage de balisage hypertexte - **H**yper **T**ext **M**arkup **L**anguage), qui peuvent contenir des liens vers d'autres pages Web.

- Fonctionnement :

  `http://www.uphf.fr/fr/formations/but-informatique.html`

Technologie d'affichage du Web

- **Web ?**

  Le Web (ou World Wide Web, www) désigne l'*ensemble des sites Web visitables depuis un navigateur* grâce à un système d'adresses URL (Uniform Ressource Locator).

  Les pages Web sont des documents écrits en **HTML** (langage de balisage hypertexte - **H**yper **T**ext **M**arkup **L**anguage), qui peuvent contenir des liens vers d'autres pages Web.

- Fonctionnement :

Evolution du Web

L'évolution du web peut être expliquée en trois grandes étapes :

Web 1.0 (années 1990) : C'était un web **statique**, principalement constitué de pages HTML simples, avec peu d'interaction. Les utilisateurs consommaient principalement du contenu, qui était souvent créé par des entreprises ou des webmasters.

Web 2.0 (années 2000) : Cette phase marque l'arrivée du web **interactif** et **collaboratif**. Les plateformes comme les réseaux sociaux (Facebook, YouTube, etc.) et les blogs ont permis aux utilisateurs de créer, partager et commenter du contenu.

Web 3.0 (en émergence) : Connu comme le web **sémantique** ou **décentralisé**, il s'appuie sur l'**intelligence artificielle** pour rendre les données plus intelligibles et accessibles aux machines.

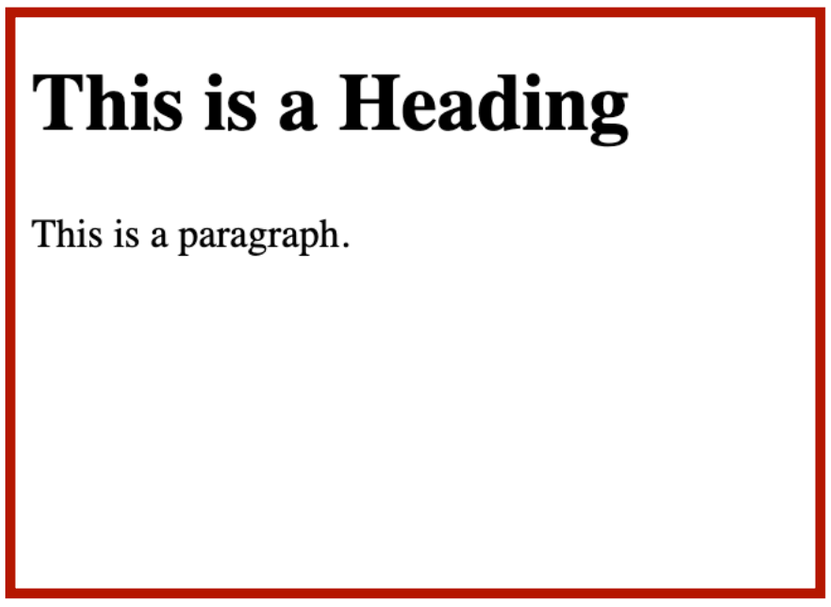
Langage HTML

- Une page HTML est écrite en texte brut et comporte du contenu textuel accompagné de balises.

- Les **balises** servent à définir la structure du document et le positionnement des éléments, ajouter des liens, insérer d'autres ressources (images, vidéos, scripts), etc.

- Le navigateur Web affiche un document HTML en "**interprétant**" visuellement le code HTML (ou source).

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

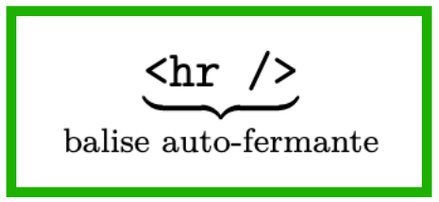


Balise HTML

- Un mot entre les symboles **<** et **>** est une balise. Tout le reste est du contenu textuel.

- Les balises forment des éléments de différents types.

Elément h1



Elément hr

- Les éléments peuvent s’imbriquer les uns dans les autres.

```
<body> <h1> Titre </h1> </body>
```

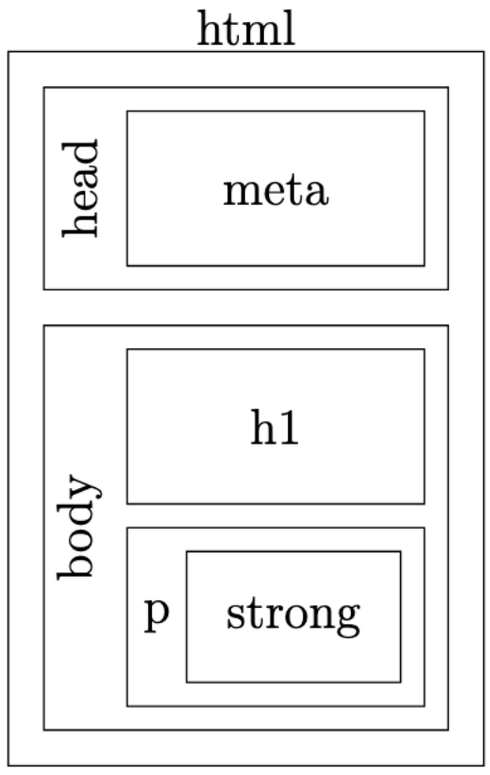
- Balise "commentaire" :

```
<!--
Ce texte est un commentaire.
-->
```

Structure d'une page HTML

- Une page HTML est composée de deux parties :
  - L'élément **<head>** : informations générales (ou métadonnées) sur la page qui ne sont pas affichées. Ce peut être l'encodage de caractères, la langue, des feuilles de style CSS, etc.
  - L'élément **<body>** : tout ce qui s'affiche dans le navigateur.

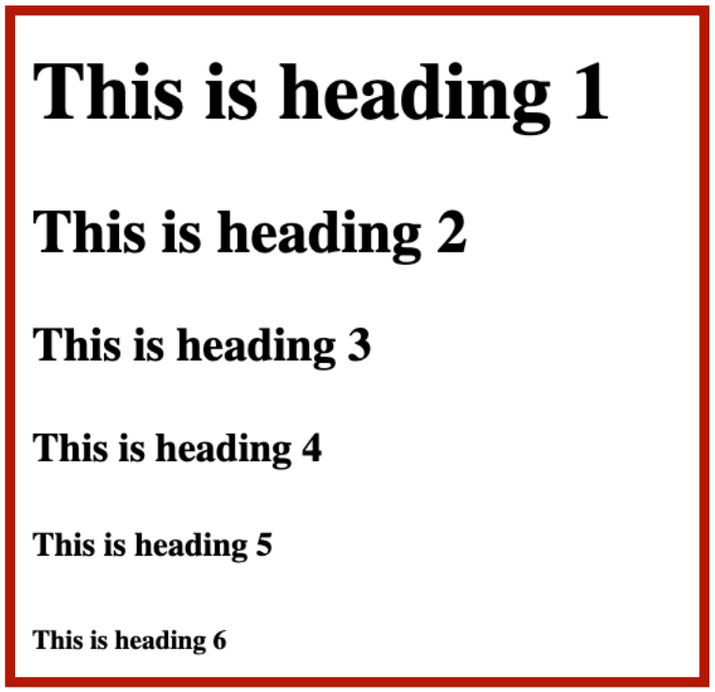
```
<!DOCTYPE html>
 <html>
   <head>
     <meta charset="utf-8"/>
   </head>
   <body>
     <h1> Titre de niveau 1 </h1>
     <p>
      Un <strong>paragraphe !</strong>
     </p>
   </body>
 </html>
```



Exemples de balises

- **<h1>** - **<h6>** sont les différents niveaux de titre

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```



https://www.w3schools.com/html/

Exemples de balises

- **<h1> - <h6>** sont les différents niveaux de titre

- **<section>** qui indique une partie de la page (imaginez une séparation partie 1 / partie 2)

```
<section>
  <h2>WWF History</h2>
  <p>The World Wide Fund for Nature (WWF) is an international organization
working on issues regarding the conservation, research and restoration of
the environment, formerly named the World Wildlife Fund. WWF was founded
in 1961.</p>
</section>

<section>
  <h2>WWF's Symbol</h2>
  <p>The Panda has become the symbol of WWF. The well-known panda logo of
WWF originated from a panda named Chi Chi that was transferred from the
Beijing Zoo to the London Zoo in the same year of the establishment of
WWF.</p>
</section>
```

https://www.w3sc

WWF History

The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.

WWF's Symbol

The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.

Exemples de balises

- **<h1>** - **<h6>** sont les différents niveaux de titre
- **<section>** qui indique une partie de la page (imaginez une séparation partie 1 / partie 2)
- **<p>** indique un paragraphe
- **** et ** ** sont des éléments “neutres”. On utilise ces balises pour appliquer des changements d’apparence à des morceaux qui ne correspondent pas à un élément préexistant.
- ** ** : pour créer des listes d'éléments
- **<table> <tr> <td>** : pour créer des tableaux

https://www.w3schools.com/html/

Exemples de balises

- **`<h1>`** - **`<h6>`** sont les différents niveaux de titre
- **`<section>`** qui indique une partie de la page (imaginez une séparation partie 1 / partie 2)
- **`<p>`** indique un paragraphe
- **`<span>`** et **`<div>`** sont des éléments “neutres”. On utilise ces balises pour appliquer des changements d’apparence à des morceaux qui ne correspondent pas à un élément préexistant.

```
<p>My mother has <span style="color:blue;font-weight:bold">blue</span>
eyes and my father has <span style="color:darkolivegreen;font-
weight:bold">dark green</span> eyes.</p>
```

My mother has **blue** eyes and my father has **dark green** eyes.

https://www.w3schools.com/html/

Exemples de balises

- **<h1>** - **<h6>** sont les différents niveaux de titre
- **<section>** qui indique une partie de la page (imaginez une séparation partie 1 / partie 2)
- **<p>** indique un paragraphe
- **** et ** ** sont des éléments “neutres”. On utilise ces balises pour appliquer des changements d’apparence à des morceaux qui ne correspondent pas à un élément préexistant.
- ** ** : pour créer des listes d'éléments

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Tea
- Milk

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>

<ol start="50">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

1. Coffee
2. Tea
3. Milk

50. Coffee
51. Tea
52. Milk

https://www.w3schools.com/html/

Exemples de balises

- **<table> <tr> <td>** : pour créer des tableaux

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
  <tr>
    <td>March</td>
    <td>$90</td>
  </tr>
</table>
```

Month	Savings
January	$100
February	$80
March	$90

https://www.w3schools.com/html/

Attributs

- En plus du nom de la balise et du contenu, on peut ajouter des **attributs** à un élément pour modifier son comportement.

- Ces attributs sont indiqués *dans la balise d'ouverture uniquement*.

- Certains éléments ont des attributs obligatoires. Par exemple, un lien hypertexte <a> qui utilise l'attribut obligatoire href pour indiquer la cible.

```
<a href= "http://www.uphf.fr"> Lien vers l'uphf </a>
```

- **`id`** et **`class`** : peuvent s'appliquer à tous les éléments (ils sont génériques). Ils servent à donner un « nom » à un élément unique (id) ou à un groupe d'éléments (class)

- Table : colspan et rowspan

Attributs

- En plus du nom de la balise et du contenu, on peut ajouter des **attributs** à un élément pour modifier son comportement.

- Ces attributs sont indiqués *dans la balise d'ouverture uniquement*.

- Certains éléments ont des attributs obligat… …n lien hypertexte <a> qui utilise l'attribut obli… …quer la cible.

```
<a href= "http://www.uphf.fr">                                    /a>
```

- **id** et **class** : peuvent s'appliquer à tous … génériques). Ils servent à donner un « nom … …e (id) ou à un groupe d'éléments (class)

- Table : colspan et rowspan

Month	Savings
January	$100
February	$80
March	$90
Exemple1	
April	$80
May	

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
  <tr>
    <td>March</td>
    <td>$90</td>
  </tr>
  <tr>
  <td colspan=2>Exemple1</td>
  </tr>
  <tr >
    <td >April</td>
    <td rowspan=2>$80</td>
  </tr>
  <tr>
    <td>May</td>
  </tr>
</table>
```

Base du CSS

- Le code CSS se présente sous la forme d'une suite de blocs :

```
sélecteur {propriété: valeur;}
```

Exemple : `p {color : red;}`

- Feuille de style CSS (extension .css) : fichier dans lequel vous écrirez toutes les instructions css.
- Lien entre html et css : on place un élément <link /> dans le <head> :

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

Sélecteurs

- Le sélecteur sélectionne les éléments HTML qui seront concernés par le bloc.

- Les sélecteurs les plus utilisés sont :

  `nom{ ... }` : sélectionne tous les éléments <nom>.

  `.nom{ ... }` : sélectionne tous les éléments de classe (class) nom.

  `#nom{ ... }` : sélectionne tous les éléments d'identifiant (id) nom.

  `nom1 nom2{ ... }` : sélectionne tous les éléments <nom2> qui sont à l'intérieur d'un élément <nom1>.

  `nom1, nom2` sélectionne les élements <nom1> et les éléments <nom2>.

  `nom1.nom2` sélectionne tous les élements <nom1> de classe nom2.

Propriétés

- A l'intérieur d'un bloc de CSS, on trouve une série de propriétés et de valeurs avec la syntaxe suivante :

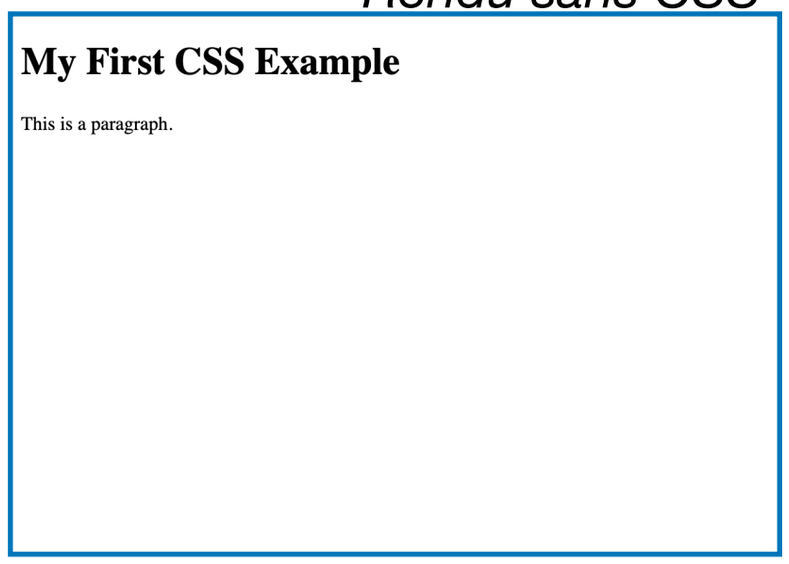
```
selecteur{
propriété: valeur;
. . .
propriété: valeur;
}
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css"/>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

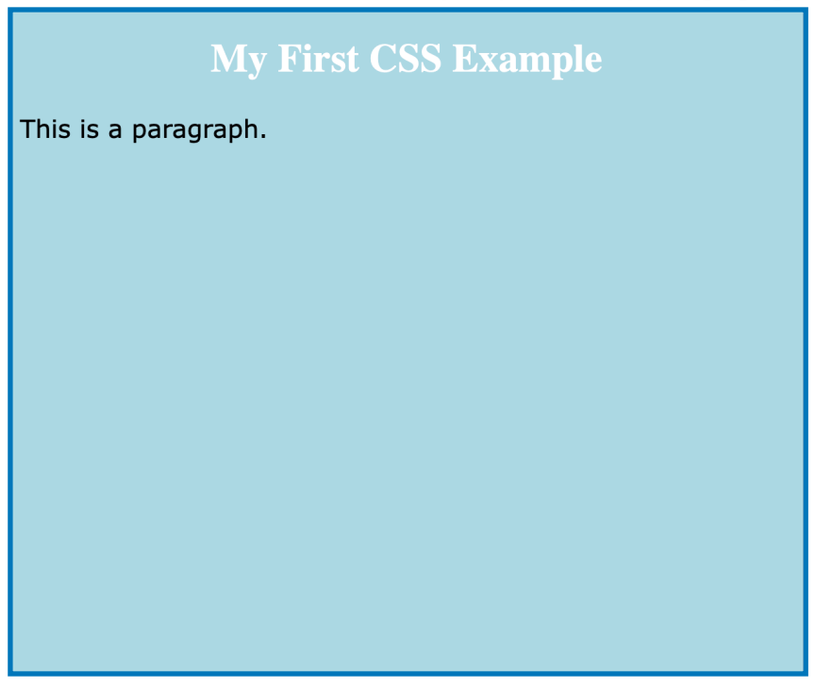
style.css

```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

Rendu sans CSS



Rendu avec CSS



Flot HTML

- Par défaut, les éléments viennent se placer les uns après les autres dans l'ordre du fichier, de gauche à droite et de haut en bas, comme dans un traitement de texte : on appelle ce comportement par défaut **le flot HTML**.

- On peut modifier ce flot ou sortir des éléments du flot pour faire des pages beaucoup plus complexes.

Comportement des éléments

- **Le mode inline** : Tous les éléments inline se comportent comme du texte et occupe uniquement l'espace nécessaire de son contenu.

  Par exemple : <a> (lien hypertexte), <code> (fragment de code), (élément neutre)...

- **Le mode block** : Par défaut, les éléments block sont placés seuls sur leur ligne (ils prennent toute la place disponible horizontalement).

  Par exemple : <h1> - <h6> (titres), <p> (paragraphe), <img/> (image), (bloc neutre)...

```
<p style="border:1px solid red">This is a paragraph.</p>
<span style="border:1px solid red">This is a paragraph.</span>
```

This is a paragraph.

This is a paragraph.

- Pour changer le mode d'un élément : propriété CSS **`display`**.

  `display:inline;`

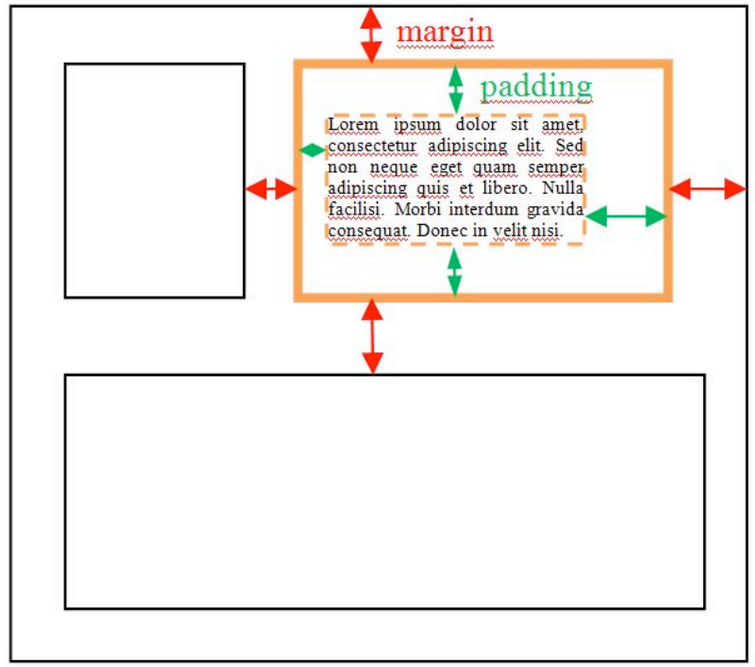
  `display: block;`

  `display:none;`

  `display: inline-block;` (éléments côte à côte pour lesquels ont peut modifier la hauteur)

Positionnement CSS

Le positionnement CSS consiste à utiliser certaines propriétés CSS pour changer le comportement par défaut et que le bloc sorte de son emplacement attendu.



- Propriétés **`margin`** et **`padding`** : servent à ajouter de l'espace entre les éléments.
  margin l'ajoute autour du bloc
  padding l'ajoute dans le bloc.

  Exemple : margin: 1cm ou margin-left: 1cm.

- Propriété **`float`** : (valeurs left ou right) permet d'envoyer un bloc à gauche ou à droite de la page (on dit qu'il devient flottant). Le reste du contenu reste positionné normalement dans l'espace restant.
  (https://www.w3schools.com/cssref/pr_class_float.php)

Positionnement CSS

- Propriété **`position`** : permet de placer des éléments de manière mieux contrôlée. Quand vous modifiez la propriété position, vous devez préciser là où vous placez le bloc avec les propriétés left, right, top et bottom.

- *Les valeurs de position sont :*

    - static est la valeur par défaut. L'élément reste à sa position normale dans le flux.

    - relative part de la position normale de l'élément et le déplace des valeurs left/right/top/bottom.

    - absolute place l'élément par rapport à son ancêtre le plus proche qui n'a pas static. L'élément sort du flux normal → ne prend plus de place et peut chevaucher d'autres éléments.

    - fixed place l'élément à une position fixée sur la fenêtre du navigateur (donnée par left/right/top/bottom) et reste immobile même quand la page défile.

    - sticky laisse l'élément à sa position normale, et passe en mode fixed quand le défilement arrive à son niveau.

(https://www.w3schools.com/css/css_positioning.asp)

Programme National

Descriptif détaillé

Objectif

L'objectif de cette ressource est d'apprendre les techniques de création de documents numériques sur le web en réponse à des besoins client. Cette ressource est une base pour réaliser un développement d'application tout en appréhendant les besoins du client et de l'utilisateur.

Savoirs de référence étudiés

- Spécifications d'interfaces utilisateur, maquettage (sketch, scénarios, persona…)
- Technologies d'affichage du Web (par ex. : Hypertext Markup Language (HTML), Cascading Style Sheet (CSS)…)
- Test de la conformité des sites Web aux standards d'accessibilité World Wide Web Consortium (W3C) / Web Accessibility Initiative (WAI)

Prolongements suggérés

- Génération de documents numériques

interaction humain-machine (IHM) | Front web | Maquettage

W3C

- Le **W3C** (pour « World Wide Web Consortium ») est l'organisme qui définit les standards du Web (comment on écrit du HTML & CSS et comment les navigateurs l'affichent).

- C'est grâce à ce standard que (en théorie) tous les sites du monde s'affichent correctement dans votre navigateur.

- Les standards sont essentiels pour assurer un web **accessible et interopérable :**

  - **Interopérabilité** : Les standards permettent aux sites web de fonctionner correctement sur différents navigateurs (Chrome, Firefox, Safari, etc.) et plateformes (ordinateurs, smartphones, tablettes) sans avoir besoin d'adaptations spécifiques.

  - **Accessibilité** : Le W3C établit des normes pour rendre le web accessible à tous, y compris aux personnes en situation de handicap. Cela inclut des directives pour le contenu, la navigation, et les interactions, afin que le web soit inclusif.

Validation

- La plupart des navigateurs ont une approche « laxiste » : ils tentent de corriger et d'afficher au mieux le HTML invalide. Pour les utilisateurs, cela évite de rencontrer des messages d'erreur sur des sites anciens ou mal codés.

- Vous devez vous méfier : même si une page HTML s'affiche bien sur un navigateur, elle peut tout de même contenir des erreurs pas directement visibles.

- Pour s'assurer que notre code respecte les standards, il faut donc ***utiliser un validateur*** qui, comme un compilateur en programmation, nous indique les endroits où notre code ne respecte pas le bon format et fournit des suggestions pour le corriger.

- *Voici les validateurs fournis par le W3C :*

  HTML : http://validator.w3.org/

  CSS : http://jigsaw.w3.org/css/

Conception adaptative

- Aux origines du Web, la plupart des terminaux utilisés pour se connecter avaient des tailles d'écran proches.

- Plus tard, de nombreuses pages Web bien adaptées à un écran d'ordinateur se sont révélées difficiles d'utilisation avec un téléphone, une tablette, etc.

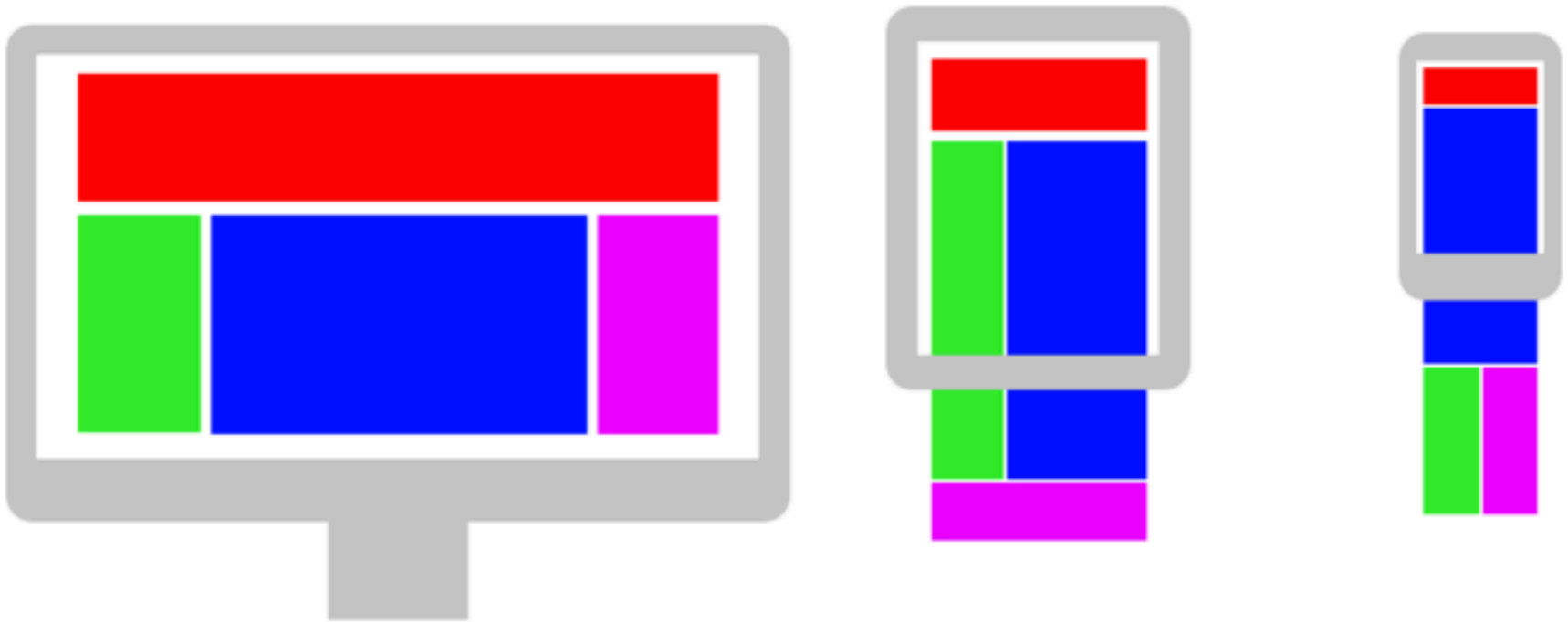
- Face à ce problème, on peut distinguer deux approches :
  - Faire des pages Web différentes en fonction de la taille de l'écran du visiteur (ou faire une application pour téléphone) ;
  - Faire une seule page Web qui reste utilisable quelle que soit la taille de l'écran du visiteur.

➡ C'est cette seconde approche qu'on appelle **« Conception adaptative de sites Web »** ou **« Responsive Web Design »**.

Conception adaptative

- En conception adaptative de sites Web, les blocs principaux changent de position en fonction de la taille de l'écran.

- Le but est que la page soit lisible et utilisable par tous, et d'utiliser au mieux l'espace disponible.



Conception adaptative

- **Unités relatives** : Les unités relatives sont celle dont la valeur dépend de l'appareil de l'utilisateur, par opposition aux unités absolues (centimètres, points, pixels. . .).

  Exemple : `width: 50%; height:80vh;`

  L'unité **`vh`** est une unité de mesure relative en CSS qui signifie **"Viewport Height"** (hauteur de la fenêtre d'affichage) : `1vh` = 1% de la hauteur du viewport.

- Propriétés `min-width` et `max-width` (idem avec height) pour empêcher un bloc de devenir trop petit ou trop grand.

- **Requête media** : permet d'appliquer une partie du CSS uniquement pour certains appareils.

  Exemple : `@media (min-width: 640px and max-width: 1280px){bloc CSS}` ne s'appliquera que quand la zone d'affichage du navigateur a une largeur comprise entre ces deux valeurs.

- Tester vos codes avec un outil de vue adaptative. La vue adaptative vous permet de tester des largeurs précises et de reproduire le comportement d'une variété d'appareil.

Utilisation de Flexbox

Pour faire de la mise en page avec Flexbox, il faut :

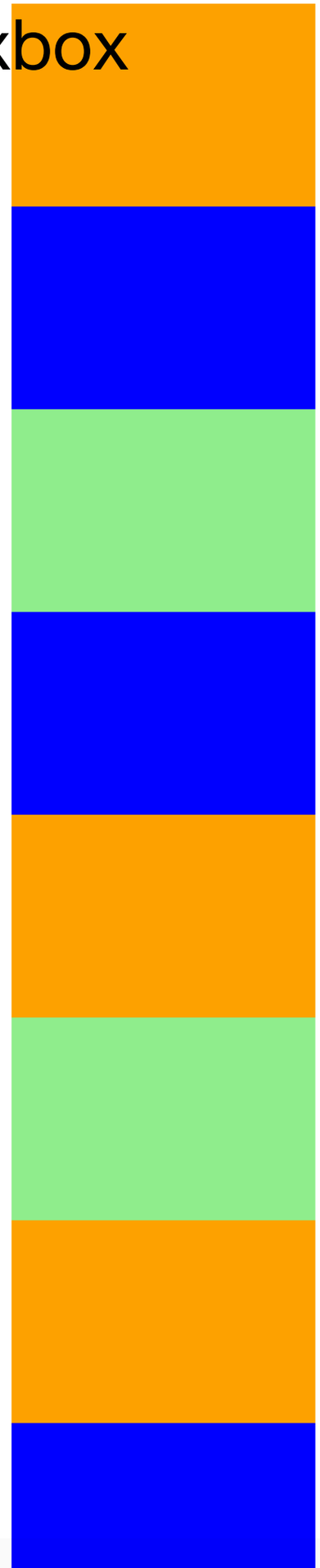
1. Définir un conteneur.
2. Et placer à l'intérieur plusieurs éléments.

Un conteneur (*container* en anglais) est une balise qui peut renfermer d'autres balises.

```
<div class="container">
  <div class="element element1"></div>
  <div class="element element2"></div>
  <div class="element element3"></div>
  <div class="element element2"></div>
  <div class="element element1"></div>
  <div class="element element3"></div>
  <div class="element element1"></div>
  <div class="element element2"></div>
  <div class="element element3"></div>
  <div class="element element2"></div>
  <div class="element element1"></div>
  <div class="element element3"></div>
</div>
```

```
.element {
  width: 150px;
  min-height: 100px;
}

.element1 {
  background-color : orange;
}
.element2 {
  background-color : blue;
}
.element3 {
  background-color : lightgreen;
}
```

Sans flexbox



Utilisation de Flexbox

La propriété display:flex va placer les blocs côte à côte.

```
<div class="container">
  <div class="element element1"></div>
  <div class="element element2"></div>
  <div class="element element3"></div>
  <div class="element element2"></div>
  <div class="element element1"></div>
  <div class="element element3"></div>
  <div class="element element1"></div>
  <div class="element element2"></div>
  <div class="element element3"></div>
  <div class="element element2"></div>
  <div class="element element1"></div>
  <div class="element element3"></div>
</div>
```

```
.container {
    display: flex;
}
```

Avec flexbox

Utilisation de Flexbox

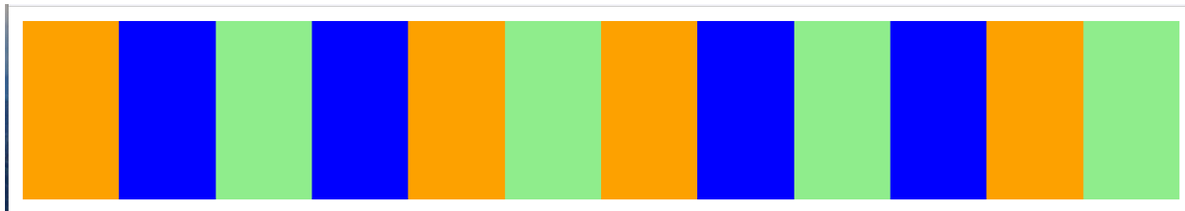
Flexbox permet d'agencer ces éléments dans le sens que l'on veut. Avec **flex-direction**, on peut les positionner verticalement ou encore les inverser.

- row : organisés sur une ligne (par défaut) ;
- column : organisés sur une colonne ;
- row-reverse : organisés sur une ligne, mais en ordre inversé ;
- column-reverse : organisés sur une colonne, mais en ordre inversé.

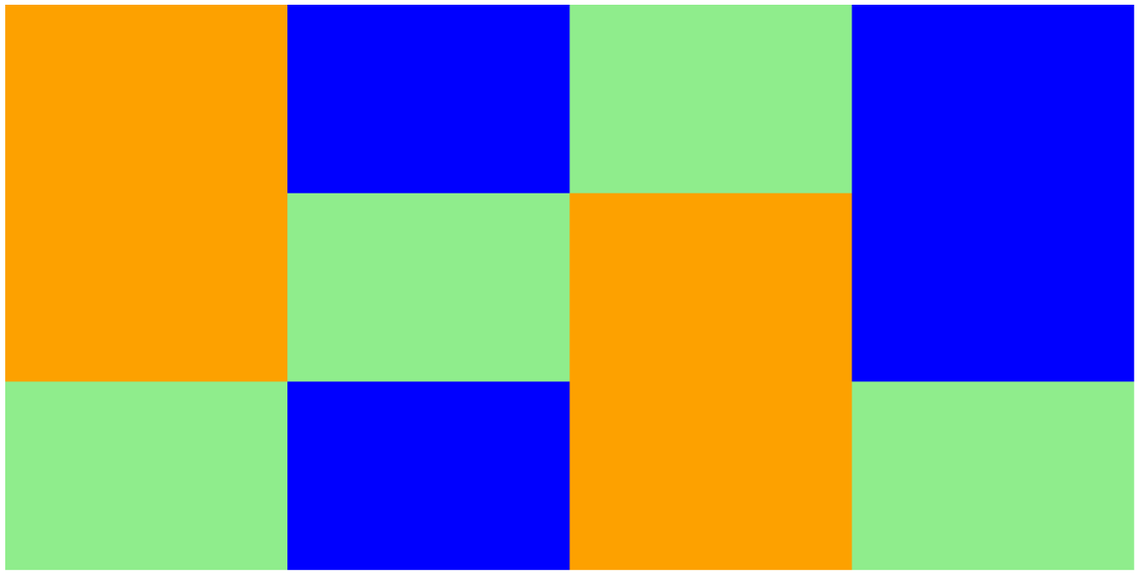
Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place. Vous pouvez demander à ce que les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec **flex-wrap**.

- nowrap : pas de retour à la ligne (par défaut) ;
- wrap : les éléments vont à la ligne lorsqu'il n'y a plus la place ;
- wrap-reverse : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

Utilisation de Flexbox



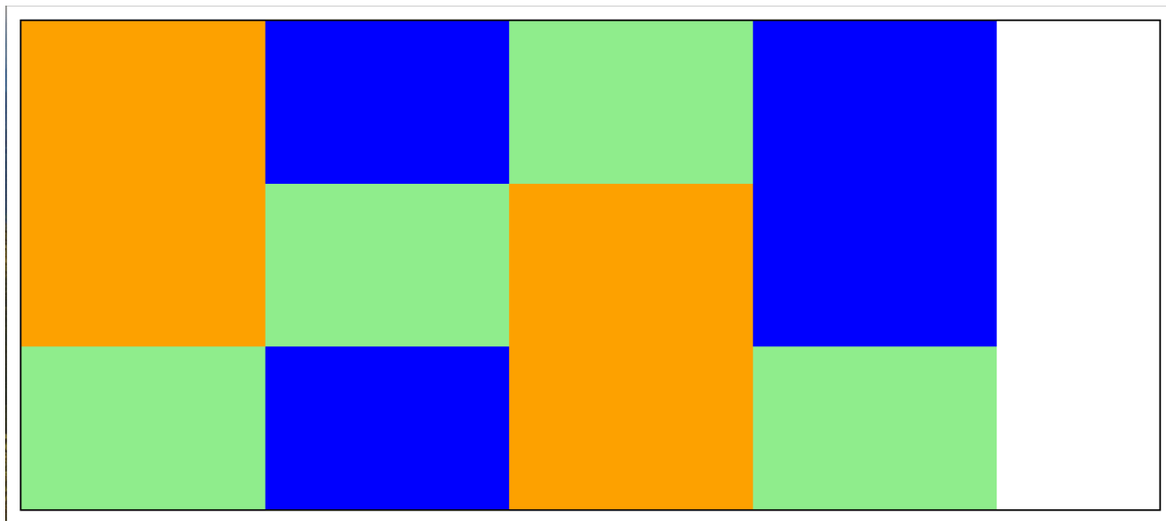
```
.container {
    display: flex;
    width: 650px;
}
```



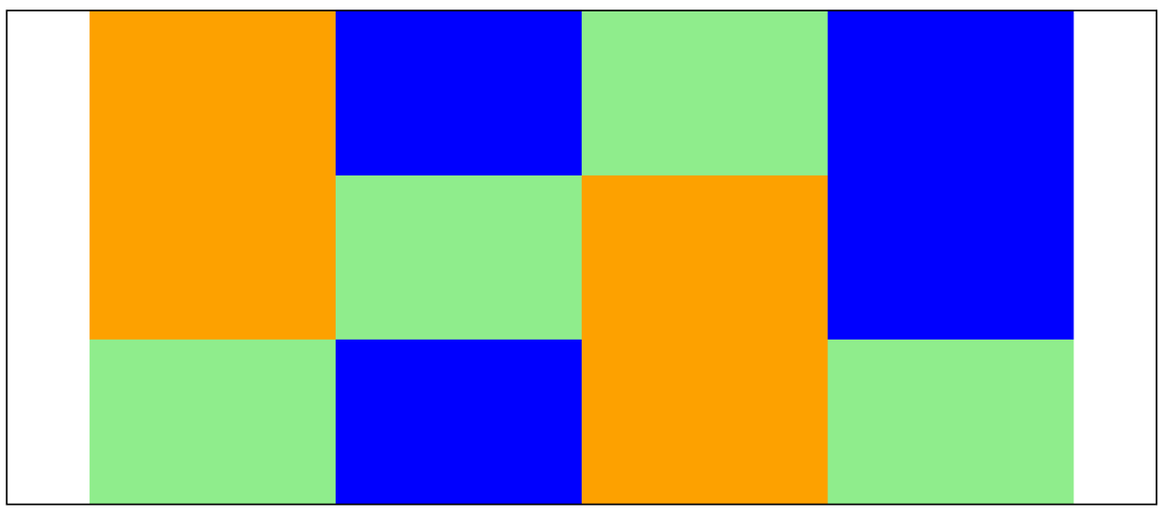
```
.container {
    display: flex;
    width: 650px;
    flex-wrap: wrap;
}
```

Pour changer l'alignement de l'axe principal, on utilise **justify-content** :

- **flex-start**: alignés au début (par défaut) ;
- **flex-end**: alignés à la fin ;
- **center**: alignés au centre ;
- **space-between**: les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux) ;
- **space-around**: idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités.



```css
.container {
    display: flex;
    width: 700px;
    flex-wrap: wrap;
    border: 1px solid #000;
}
```

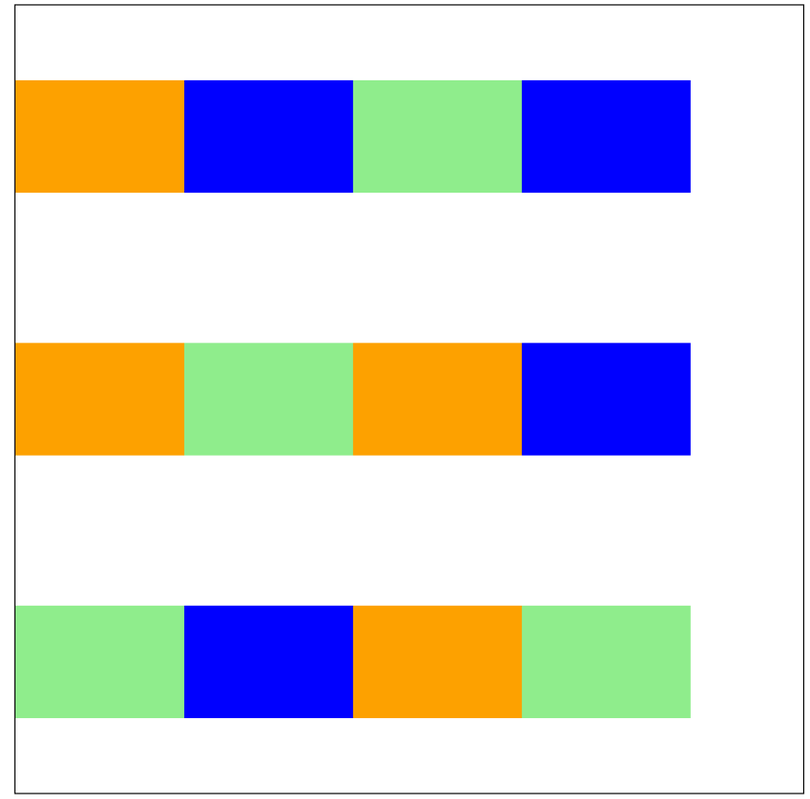


```css
.container {
    display: flex;
    width: 700px;
    flex-wrap: wrap;
    justify-content:center;
    border: 1px solid #000;
}
```

Pour changer l'alignement de l'axe secondaire, on utilise :

1. align-items

- **S'applique aux éléments individuels sur chaque ligne**.
- Gère l'alignement des éléments sur **l'axe secondaire**, **dans leur propre ligne**.



```css
.container {
    display: flex;
    width: 700px;
    height:700px;
    flex-wrap: wrap;
    border: 1px solid #000;
    align-items:center;
}
```

`flex-start`: alignés au début (par défaut) ;
`flex-end`: alignés à la fin ;
`center`: alignés au centre ;

2. align-content

- **S'applique aux lignes elles-mêmes dans l'espace global du conteneur**.
- Gère l'alignement des **lignes flexibles** sur l'axe transversal.


```css
.container {
    display: flex;
    width: 700px;
    height:700px;
    flex-wrap: wrap;
    border: 1px solid #000;
    align-content:center;
}
```

Evolution du langage HTML

HTML 1.0 (1993) : Première version officielle du langage, définie par Tim Berners-Lee.

Elle permettait une structuration basique des documents en ajoutant des titres, des paragraphes, des listes, et des liens hypertextes.
Version très limitée : il n'y avait ni tableaux, ni styles, ni fonctionnalités multimédia avancées.

HTML 2.0 (1995) : Standardisé par le W3C pour officialiser les bases du web.

Introduit des éléments comme les **formulaires**, permettant l'interaction des utilisateurs avec les sites web.
Il reprend les éléments de HTML 1.0 tout en les consolidant.

HTML 3.2 (1997) :

Il introduit des fonctionnalités comme : les tableaux, le support des styles (attributs bgcolor, font...)
Il commence à s'éloigner de la simple structuration de texte pour permettre des mises en page plus élaborées.

HTML 4.01 (1999) : Séparation claire entre la structure (HTML) et la présentation (à travers les feuilles de style CSS).

Introduction de la notion d'**accessibilité** (balises comme <label>, <legend>).
Support des **scripts** via JavaScript et d'autres fonctionnalités interactives (DOM).

XHTML 1.0 (2000) : Version stricte de HTML conforme aux règles du XML.

XHTML impose une syntaxe rigide (balises toujours fermées, attributs entre guillemets, respect de la casse).
Objectif : rendre les documents web plus facilement **interopérables** avec d'autres technologies XML, mais la syntaxe stricte n'a pas été adoptée universellement.

HTML5 (2014) : Standardisé en 2014

Support **natif** de la **vidéo** (<video>) et de l'**audio** (<audio>) sans besoin de plugins externes (comme Flash).
Introduction des **API web** modernes (Canvas pour les graphiques, WebGL pour le rendu 3D, API géolocalisation, API WebSocket, etc.).
Nouvelles balises sémantiques comme <article>, <section>, <header>, <footer>, qui permettent une meilleure structuration et lisibilité des documents.
Abandon de la syntaxe rigide de XHTML tout en encourageant une structure claire.
Renforcement de la prise en compte des **appareils mobiles** et des interfaces tactiles, avec une conception plus adaptée au **responsive design**.