le cnam
Grand Est

Développement web (2)

Sites dynamiques
et développement côté serveur

NFA017 (4 ECTS)

Séance 01
Rappels
2022

Le Cnam en Grand Est - Centre régional
4 avenue du Docteur Heydenreich CS 65228 F 54052 Nancy Cedex
tél +33 (0)3 83 85 49 00 www.cnam-grandest.fr

Le plus grand soin a été apporté à la réalisation de ce support pédagogique afin de vous fournir une information complète et fiable. Cependant, le Cnam Grand-Est n'assume de responsabilités, ni pour son utilisation, ni pour les contrefaçons de brevets ou atteintes aux droits de tierces personnes qui pourraient résulter de cette utilisation.

Les exemples ou programmes présents dans cet ouvrage sont fournis pour illustrer les descriptions théoriques. Ils ne sont en aucun cas destinés à une utilisation commerciale ou professionnelle.

Le Cnam ne pourra en aucun cas être tenu pour responsable des préjudices ou dommages de quelque nature que ce soit pouvant résulter de l'utilisation de ces exemples ou programmes.

Tous les noms de produits ou autres marques cités dans ce support sont des marques déposées par leurs propriétaires respectifs.

Ce support pédagogique a été rédigé par Simon MAHIEUX, enseignant au Cnam Grand-Est.

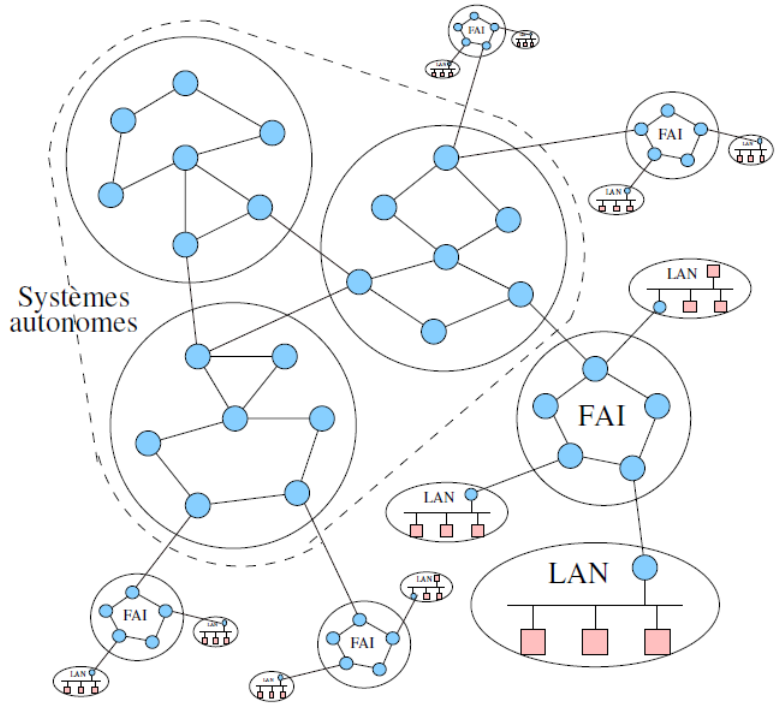
Table des matières

1. Introduction

Cette première séance a pour objectif de réactiver les connaissances acquises lors des précédentes unités. Nous allons ainsi rappeler brièvement les concepts d'Internet et du Web, puis réviser le HTML, le XML et le CSS.

2. Internet

Internet est un réseau informatique mondial constitué d'un ensemble de réseaux nationaux, régionaux et privés interconnectés entres eux.



2.1 ARPAnet

Le projet ARPAnet « Advanced Research Projects Agency Network » est en quelque sorte l'ancêtre d'Internet. Ce projet a été développé par les États-Unis et a vu le jour en 1969. L'objectif du projet ARPAnet était de créer un réseau à transfert de paquets avec l'idée de base de rendre possible l'interaction entre différents protocoles.

Ce projet était novateur sur plusieurs points :

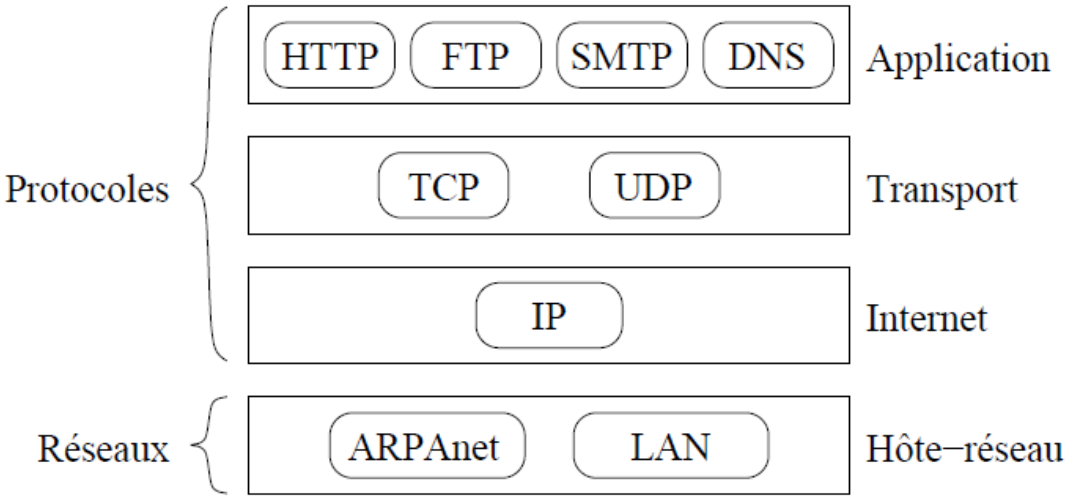
- Un réseau décentralisé : Sortir du modèle d'un réseau centralisé sur lequel les machines se connectent à un centre unique.
- Un réseau hétérogène : rendre possible une connexion entre des machines de différents constructeurs.
- L'utilisation de la commutation de paquets : diviser les informations dans plusieurs paquets. Chaque paquet fonctionne indépendamment des autres, cela signifie que les paquets peuvent prendre des routes différentes puis sont réassemblés lorsqu'ils sont tous arrivés à destination.

2.2 TCP/IP

TCP/IP (*Transmission Control Protocol / Internet Protocol*) est un modèle regroupant les protocoles TCP et IP.

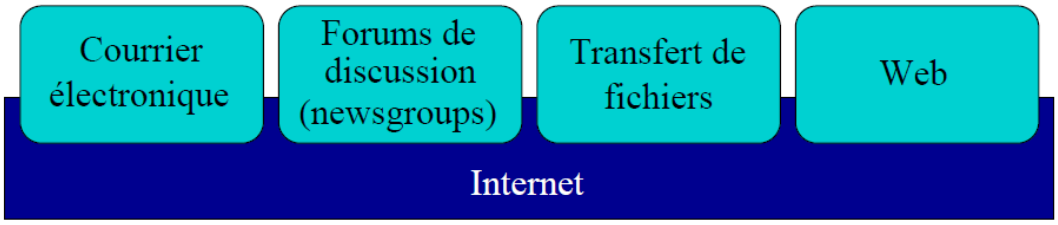
Le modèle TCP/IP est décomposé en plusieurs modules qui sont exécutés dans un ordre précis et qui réalise une tâche spécifique. C'est pour cette raison que nous parlons d'une architecture réseau en couches.

Les quatre couches du modèle TCP/IP sont :
- La couche Réseau
- La couche Internet
- La couche Transport
- La couche Application



2.3 Les applications d'Internet

Les applications utilisant le réseau Internet sont nombreuses :
- le courrier électronique : l'envoi de messages à distance
- Forums de discussion (newsgroups)
- la messagerie instantanée
- la téléphonie
- l'échange de fichiers : transfert de fichier d'une machine à une autre (Protocole PDF)
- le Web : mise en ligne de pages avec des liens et du contenu multimédia accessibles via navigateur



C'est l'apparition du web qui a popularisé et considérablement développé l'utilisation d'internet.

3. Le Web

Le Web (World Wide Web) est un système qui permet de consulter des pages contenant des liens hypertexte avec un navigateur.

Le Web repose sur un service d'Internet : le protocole HTTP (*HyperText Transfer Protocol*). Il faut garder à l'esprit que tous les échanges sur le Web exploitent le protocole HTTP.

De nombreux langages et formats sont liés au Web, nous pouvons ainsi lister :
- Le HTML
- Le CSS
- Le Javascript
- Le JSON
- Etc…

Tous ces langages et formats seront abordés dans le cadre de notre unité.

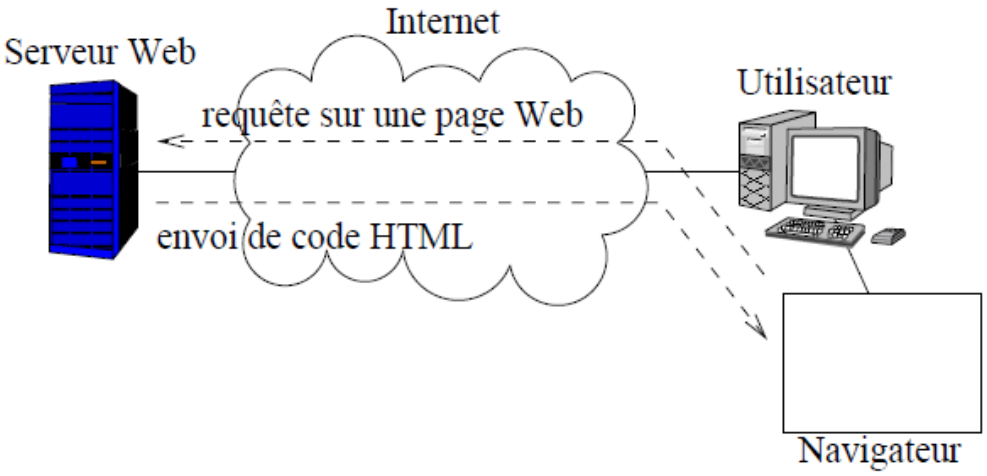
3.1 Le protocole HTTP

Le protocole HTTP (*HyperText Transfer Protocol*) fait partie de la couche applicative d'Internet.

Il est basé sur le modèle de communication client/serveur :

- Le client envoie des requêtes, reçoit les données et affiche des objets Web (par exemple : un navigateur Web)

- Le serveur attend des requêtes et y répond (serveur Web)



Le protocole HTTP est « sans état » (*stateless*), cela signifie qu'il ne conserve pas d'information entre deux transactions. Il faut donc tout reprendre à zéro lors de la prochaine transaction.

3.1.1 Les requêtes

Une requête est un message au format ASCII.

Le format est le suivant :
- Le verbe (commandes GET, POST, HEAD) en première ligne
- Les champs d'en-tête (un champ par ligne)
- Les données

Il existe de nombreux champs d'en-tête, voici une liste non-exhaustive des champs les plus utilisés pour une requête HTTP :

Nom	Explications
Accept	Types de contenu acceptés par le client
Accept-Charset	Jeux de caractères acceptés par le client
Accept-Encoding	Algorithme de compression acceptés par le client
Accept-Langage	Langues acceptées par le client
Connection	Utilisation d'une connexion persistante ou non
Host	Nom de domaine du serveur
Keep-Alive	Gestion du timeout et de la durée maximum de la connexion
User-Agent	Chaine permettant d'identifier l'application, sa version et le système d'exploitation

Voici un exemple d'une requête HTTP :

```
GET /index.php / HTTP/1.1
Host: mon-site.com
User-agent: Mozilla/5.0
Accept: text/html, image/gif, image/jpeg
Accept-Language: fr, fr-FR, en, en-US
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

En analysant cette requête, nous comprenons que :
- La page /index.php est appelée en GET
- Le nom de domaine (host) est : mon-site.com
- Le navigateur utilisé (User-agent) est : Mozilla
- Le client accepte les types de contenu (Accept) : HTML, Images GIF, Image JPEG
- Le client accepte les types de compression (Accept-Encoding) : gzip et deflate
- Le client accepte les jeux de caractères (Accept-Charset) : ISO-8859-1 et UTF-8
- La connexion est persistante (Connection) pendant 300 secondes (Keep-Alive)

3.1.2 Les réponses

Une réponse à une requête HTTP est également un message au format ASCII.

Le format est le suivant :
- Le protocole et l'état en première ligne (protocole, code d'état et message d'état)
- Les champs d'en-tête (un champ par ligne)
- Le corps : les données

La liste ci-dessous présente les champs les plus utilisés pour une réponse HTTP :

Nom	Explications
Server	Jeux de caractères acceptés par le client
Content-Type	Type MIME de la ressource
Content-Length	Taille en octets du corps de la réponse
Content-Language	Langue du corps de la réponse
Content-Encoding	Compression utilisée dans le corps de la réponse
Date	Date et heure d'origine du message
Referer	Adresse de la page web précédente

Par exemple, le résultat de la requête donnée en 3.1.1 est le suivant :

```
HTTP/1.x 200 OK
Connection: close
Date: Wed, 03 Feb 2020 06:09:02 GMT
Server: Apache
Content-Length: 6821
Content-Type: text/html
données données données
```

En analysant cette réponse, nous comprenons que :
- Le protocole est http et le code de réponse est 200 OK (indique la réussite de la requête)
- La date (Date) d'origine est 03/02/2020 à 06:09 (GMT)
- Le serveur (Server) est Apache
- Le contenu a une taille (Content-Length) de 6821 octets
- Le contenu (Content-Type) est un fichier HTML (cf 3.2)

3.1.3 Codes de réponse

Le code HTTP est un code d'état permettant de déterminer le résultat d'une requête et de l'indiquer au client.

Le code est indiqué sous la forme de 3 chiffres, dont le premier indique la catégorie de réponse. Il en existe 5 en tout : informations (1xx), succès (2xx), redirection (3xx), erreur client (4xx) et erreur serveur (5xx).

Les codes les plus courants sont listés dans le tableau ci-après par catégorie.

Note : la catégorie « Informations » étant surtout utilisée pour une « extension » du HTTP nommée WebDAV, elle ne figure pas dans ce tableau.

- **2xx - Succès**

Code	Message	Explication
200	OK	Requête traitée avec succès
204	No Content	Requête traitée avec succès (Aucune information à renvoyer))

- **3xx - Redirection**

Code	Message	Explication
301	Moved Permanently	Document déplacé de façon permanente
302	Found	Document déplacé de façon temporaire
304	Not Modified	Document non modifié depuis la dernière requête

- **4xx – Erreur du client**

Code	Message	Explication
400	Bad Request	Syntaxe incorrecte de la requête
401	Unauthorized	Authentification nécessaire pour accéder au document
402	Payment Required	Paiement requis pour accéder au document
403	Forbidden	
404	Not Found	Document non trouvé

- **5xx – Erreur du serveur**

Code	Message	Explication
500	Internal Server Error	Erreur interne du serveur
501	Not Implemented	Fonctionnalité non supportée par le serveur
503	Service Unabailable	Service temporairement indisponible (ou en maintenance)

3.2 Le type MIME

Le MIME (*Multimedia Mail Extension*) définit un ensemble de types permettant de déchiffrer les données.

Il existe de nombreux types MIME, vous trouverez ci-dessous une liste non exhaustive :

Groupe	Type MIME	Explication
application	application/pdf	Fichier PDF
	application/json	Fichier JSON
	application/vnd.ms-excel	Fichier Microsoft Excel
audio	audio/mpeg (ou audio/mp3)	Fichier MP3
	audio/x-wav (ou audio/wav)	Fichier Wav
image	image/gif	Fichier GIF
	image/jpeg	Fichier JPEG
	image/png	Fichier PNG
text	text/css	Fichier CSS
	text/csv	Fichier CSV
	text/html	Fichier HTML
	text/plain	Fichier texte brut
	text/xml	Fichier XML
video	video/mpeg	Fichier MPEG
	video/mp4	Fichier MP4

On retrouve le type MIME dans le champ HTTP « Content-Type ».

3.3 Le serveur Web

Un serveur HTTP est une application capable de répondre à des requêtes HTTP. Il écoute par défaut sur le port 80.

Les trois principaux serveurs Web utilisés sont :
- Apache : Logiciel libre créé et maintenu par la fondation Apache
- IIS : Logiciel propriétaire développé par Microsoft
- Nginx : Logiciel libre créé par Igo Sysoev

3.4 La programmation Web côté client

La programmation Web côté client regroupe les langages et technologies qui sont utilisés par le navigateur.

En effet, l'affichage d'une page web utilise :
- Le HTML pour définir la structure
- Le CSS pour définir la mise en page
- Le Javascript pour rendre plus dynamique la page (scripts exécutés sur le navigateur)
- Mais aussi des images, du son, de la vidéo etc…

Ces langages et technologies permettent de créer un site statique (comme par exemple un site vitrine permettant d'afficher les services proposés par une entreprise, ou bien un CV en ligne etc…).

Les pages d'un site statique sont stockées sur le serveur web et sont en HTML/CSS/JS pur : elles sont servies au client sans aucune modification.

Les sites statiques sont limités : il n'est pas possible de gérer les actions de l'utilisateur, ni de stocker et lire des données depuis une base de données.

Si des fonctionnalités supplémentaires doivent être implémentés (création de pages dynamique à partir d'une base de données pour un blog, création d'un espace sécurisé pour un utilisateur reconnu, etc…), il faut alors utiliser un langage spécifique qui sera exécuté côté serveur.

3.5 La programmation Web côté serveur

Un langage exécuté côté serveur (par exemple : PHP, ASP, J2EE, Python, etc…) permet de construire les pages web (HTML/CSS/JS) dynamiquement sur le serveur avant de les envoyer au client. C'est pour cette raison que nous parlons de sites « dynamiques ».

De plus, l'utilisation d'un langage exécuté côté serveur est obligatoire pour utiliser une base de données.

Nous aborderons dans cette unité le langage PHP.

4. Le HTML

Le HTML (*HyperText Markup Language*) est le langage utilisé pour décrire la structure d'une page WEB. Ce n'est pas un langage de programmation, mais un langage de balisage (utilisation de balises).

La norme complète est accessible sur le site du W3C (www.w3c.org).

4.1 Structure générale d'un document HTML

L'ensemble d'un document HTML est inclus dans une balise racine <html>. Le corps du document est séparé en deux sections <head> et <body>.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <!-- En-tête : encodage, format, titre, etc. -->
    <meta charset="utf-8">
    <!-- Insertion des documents CSS -->
</head>
<body>
    <!-- Éléments de la page -->
    <!-- Insertion des scripts Javascript -->
</body>
</html>
```

La première section (<head>), l'en-tête, contient des informations générales sur le document ainsi que des métadonnées (lien vers fichier CSS, …). Les informations indiquées dans cette partie ne sont pas affichées à l'écran.

Les balises « meta » sont des balises qui fournissent des métadonnées au navigateur. Elles doivent être ajoutées dans la partie <head>. Par exemple : le titre du document, la langue utilisée, l'encodage, etc…

La seconde section (<body>), le corps, contient le contenu du document.

Pour terminer, dans l'exemple ci-dessus, on remarque également des balises commençant par "<!-- « et finissant par "-->", ce sont des commentaires. Tout le texte placé à l'intérieur de cette balise est ignoré par le navigateur. **Attention :** il reste présent dans le code-source de la page !

4.2 Les principales balises

4.2.1 En-tête

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <meta charset="utf-8"/>
   <title>Titre de mon document</title>
   <meta name="description" content="Description du contenu de la page"/>
   <link rel="stylesheet" type="text/css" href="/assets/styles.css"/>
</head>
<body>
</body>
</html>
```

4.2.1.1 Le titre

La balise <title></title> permet d'indiquer le titre d'un document HTML.

```
<title>Titre de mon document</title>
```

4.2.1.2 L'encodage

La balise <meta /> avec l'attribut "charset" permet d'indiquer au navigateur que le contenu du document utilise l'encodage UTF-8. En développement web, il est fortement conseillé d'utiliser l'encodage l'UTF-8 pour la création et le contenu des fichiers.

```
<meta charset="utf-8"/>
```

4.2.1.3 La description

La balise <meta /> avec l'attribut "name" et sa valeur "description" permet d'indiquer une description courte du contenu de la page web. Cette balise est notamment utilisée par les moteurs de recherche ou les réseaux sociaux pour afficher une description à côté du lien partagé.

```
<meta name="description" content="Description du contenu de la page"/>
```

4.2.1.4 La liaison avec un fichier

La balise <link/> définit la relation entre le document et une ressource externe. Cette balise est utilisée principalement pour lier les fichiers CSS.

```
<link rel="stylesheet" type="text/css" href="/assets/styles.css" />
```

Cette balise est également utilisée pour afficher une « *favicon* ». Un favicon est une icône mise à disposition par un site Web pour être affichée à côté du titre de la page dans l'onglet du navigateur.

```
<link rel="icon" type="image/png" href="logo.png" />
```

4.2.2 Général

4.2.2.1 Les titres

Pour réaliser différents niveaux de titres, on dispose des balises <h1> à <h6>. Ces dernières permettent de structurer les titres d'un document sur 7 niveaux.

```
<h1>Titre principale</h1>
<h2>Titre secondaire</h2>
<p>Contenu</p>
```

4.2.2.2 Les paragraphes et saut de ligne

La balise représente un paragraphe de texte et la balise
 permet de réaliser un retour chariot.

```
<p>Bonjour, ceci est une première ligne.<br/>Ceci est une deuxième ligne</p>
```

4.2.2.3 Les liens

Un lien hypertexte est une zone d'un document HTML sensible aux clics.

En cliquant sur un lien, l'utilisateur peut être redirigé vers une autre partie du document ou vers un autre document que celui-ci soit dans le même site ou n'importe où sur le Web.

Pour réaliser la redirection, il est nécessaire de disposer de l'URL (*Uniform Ressource Locator*) du document à atteindre. Cette dernière précise exactement la localisation du document à afficher. Une URL peut être absolue (chemin complet : nom de domaine + chemin + fichier) ou seulement relative (chemin relatif + nom de fichier).

```
<a href="https://www.cnam-grandest.fr/">Site du Cnam</a>
```

4.2.2.4 Les images

Dans un document HTML plusieurs formats d'images peuvent être insérées. Il est possible d'utiliser les formats JPEG, GIF, PNG et SVG.

Pour insérer une image, il faut utiliser la balise <img/>. Cette balise comporte plusieurs paramètres qui permettent de modifier l'aspect de l'image.

L'attribut src permet d'indiquer le chemin du fichier contenant l'image. Cet attribut peut contenir une URL relative ou absolue.

```
<img src="logo.png" alt="Logo"/>
```

4.2.2.5 Les tableaux

La balise <table></table> délimite les données qui doivent être contenues dans le tableau.

Les balises <thead></thead> et <tfoot></tfoot> permettent respectivement de définir un entête et un pied au tableau.
C'est particulièrement utile en cas d'impression de la page car ils se répéterons sur chaque page. Certains navigateurs répètent aussi les entêtes en fonction de la taille de l'écran, améliorant ainsi l'accessibilité.

Les balises <tr></tr> permettent de créer une nouvelle ligne.

Les balises <td></td> permettent de créer une cellule dans une ligne. Cette balise dispose de plusieurs paramètres :

- L'attribut colspan permet d'étendre la cellule en cours sur les cellules voisines de droite. Ce paramètre permet de déterminer le nombre de cellules occupées, en comptant la cellule courante.
- L'attribut rowspan permet d'étendre la cellule en cours sur les cellules voisines du dessous. Ce paramètre permet de déterminer le nombre de cellules occupées, en comptant la cellule courante.

Les balises <th> permettent de définir des cellules de titre pour une colonne. Le fonctionnement de ces balises est similaire aux balises <td>. Le contenu est juste mis en valeur (généralement en gras).

Voici un exemple complet :

```
<table>
   <thead>
      <tr><th colspan="3">Titre du tableau</th></tr>
   </thead>
   <tr>
      <td>case 1</td>
      <td>case 2</td>
      <td>case 3</td>
   </tr>
   <tr>
      <td>case 4</td>
      <td>case 5</td>
      <td>case 6</td>
   </tr>
   <tr>
      <td>case 7</td>
      <td>case 8</td>
      <td>case 9</td>
   </tr>
</table>
```

Titre du tableau		
case 1	case 2	case 3
case 4	case 5	case 6
case 7	case 8	case 9

4.2.3 Format du texte

4.2.3.1 Texte en gras

Les balises permettent de mettre en gras le texte.

Les balises permettent d'indiquer que le texte est important (il sera également mis en gras) Il y a une notion de sémantique sur l'élément .

```
<p><b>Partie A :</b> Texte <strong>important</strong></p>
```

4.2.3.2 Texte en italique

Les balises permettent de mettre en italique du texte.

```
<p>Test <i>Test</i> Test</p>
```

Les balises sont utilisées pour marquer un texte sur lequel on veut insister. Il y a également une notion de sémantique sur cet élément.

```
<p>Bonjour et bienvenue sur mon site sur la <em>programmation</em></p>
```

4.2.3.3 Texte en indice ou en exposant

Les balises permettent de mettre en indice un texte.

```
<p>Test <sub>Test</sub> Test</p>
```

Les balises permettent de mettre en exposant un texte.

```
<p>Test <sup>Test</sup> Test</p>
```

4.2.4 Éléments particuliers

4.2.4.1 Les listes à puces

Les balises permettent de créer des listes à puces qui seront non numérotées (*Unsorted List*). Chaque élément de la liste, associé avec la balise <li>, sera affiché avec une puce à sa gauche.

```
<ul>
   <li>Elément A</li>
   <li>Elément B</li>
   <li>Elément C</li>
</ul>
```

4.2.4.2 Les listes ordonnées

Les balises permettent de créer des listes numérotés (*Ordered List*). Comme pour les listes à puces, il faut utiliser la balise <li> pour ajouter un élément.

```
<ol>
   <li>Elément A</li>
   <li>Elément B</li>
   <li>Elément C</li>
</ol>
```

Notez qu'il est possible d'imbriquer des listes.

4.2.4.3 Éléments block et éléments inline

Il existe deux catégories d'éléments HTML : les éléments block (div) et les éléments inline (span).

Le comportement n'est pas le même entre ses deux catégories :

- Élément block : il est affiché en début de ligne et prend toute la largeur de l'écran
- Élément inline : il est affiché à partir de sa position

Certains éléments HTML se comportent comme un :

-
- <table>
- <h1>
- … etc.

Et d'autres comme un :

- <a>
- <b>
-
- … etc.

4.2.5 Formulaires

Un formulaire est un ensemble de champs permettant de saisir des données. Lorsque tous les champs sont remplis, les données peuvent être envoyées à un script chargé de les traiter (Nous reviendrons sur cette partie côté serveur dans un cours spécifique de cette unité.)

Pour créer un formulaire, il faut utiliser les balises <form></form>. C'est à l'intérieur de ces balises que seront placés les champs du formulaire.

Cet élément accepte différents paramètres :

- L'attribut "Method" : il indique la méthode qui sera utilisée pour envoyer les données au serveur (GET ou POST)
- L'attribut "Action" : il permet d'indiquer l'url du script qui doit recevoir les données

```
<form method="post" action="script.php">
   <p>Nom : </p>
   <input type="text" name="nom" />
   <p>Ville : </p>
   <input type="text" name="ville" value="Reims"/>
</form>
```

4.2.5.1 La balise input

La majorité des champs sont mis en place avec la balise <input/>. Cette balise dispose de différents paramètres :

- L'attribut name qui permet de donner un nom au champ. C'est obligatoire pour pouvoir récupérer la valeur d'un champ.
- L'attribut value qui définit la valeur d'un champ à l'initialisation. Pour certains champs, cette valeur peut être modifiée par l'utilisateur.
- L'attribut type qui identifie le type de champ. Nous allons voir chacun des types existants.
- D'autres paramètres peuvent être utilisés en fonction du type de champ.

Vous trouverez ci-dessous une liste des attributs les plus utilisés sur la balise <input/> :

Attribut "Type"	Explication
text	Champ texte
checkbox	Sélection d'un choix multiple
radio	Sélection d'un choix unique
date	Date
password	Mot de passe (caractères cachés)
file	Permet le choix d'un fichier (upload)
submit	Affichage d'un bouton dont le clique déclenchera l'envoi des données vers le serveur

4.2.5.2 Les boutons

La balise <button></button> permet d'afficher un bouton dans le formulaire. Le comportement associé au bouton est majoritairement géré avec du Javascript (voir de l'ajax).

```
<button type="button">Rafraichir la page</button>
```

4.2.5.3 Les listes de sélection

La balise <select></select> représente une liste d'options parmi lesquelles l'utilisateur pourra effectuer son choix. Elle est associée à la balise <option></option> qui permet d'afficher les options de la liste.

```
<form method="post" action="script.php">
   <span>Ville :</span>
   <select>
      <option value="1">Reims</option>
      <option value="2">Tinqueux</option>
      <option value="3">Cormontreuil</option>
   <select>
   <input type="submit" name="Envoyer" value="ok"/>
</form>
```

4.2.5.4 La zone de texte

La dernière balise à se rappeler est la balise <textarea></textarea> qui permet d'éditer du texte sur plusieurs lignes.

```
<form method="post" action="script.php">
   <span>Adresse</span>
   <textarea name="adresse"></textarea>
   <input type="submit" name="Envoyer" value="ok"/>
</form>
```

5. Le XML

Le XML (*Extensible Markup Language*) est un langage informatique de balisage qui a pour mission de formaliser des données textuelles dans le but de les afficher ou de les transmettre à un destinataire (serveur, application, etc…).

Dans le langage HTML, les balises sont utilisées pour définir la structure des données. Dans le langage XML, les balises définissent la structure mais également la signification de vos données. Le document ainsi produit sera à la fois lisible par un être humain et exploitable par un ordinateur.

XML dérive du SGML d'où son apparente similitude syntaxique (les balises) avec le HTML (qui dérive lui aussi du SGML).

XML repose sur des balises (comme HTML) mais impose un ensemble de règles plus strictes (par exemple chaque balise doit obligatoirement être fermée). Notons que le XHTML est une version d'HTML qui suit les règles XML.

Nous allons nous concentrer dans ce rappel sur l'utilisation de XML comme moyen de transmettre des informations et non comme moyen de les afficher (en utilisant XSLT).

5.1 Structure d'un document XML

La première ligne d'un document XML est une déclaration, elle commence par < ? et se termine par ?>. Elle permet d'indiquer que le contenu du document est en XML et que son encodage est UTF-8.

```
<?xml version="1.0" encoding="UTF-8"?>
<personnes>
   <personne id="1">
      <nom>Dupont</nom>
      <prenom>Robert</prenom>
      <poids unite="kg">75</poids>
   </personne>
   <personne id="2">
      <nom>Dumaire</nom>
      <prenom>Jean</prenom>
      <poids unite="kg">80</poids>
   </personne>
</personnes>
```

Le contenu du document se situe juste en dessous de cette première ligne.
Comme nous le constatons, un document XML reprends la syntaxe des balises HTML mais permets d'utiliser et de créer un nombre illimité de balises.

Il est également possible d'utiliser des attributs. Dans l'exemple ci-dessus, on remarque l'attribut id sur l'élément *personne* et l'attribut unite sur l'élément *poids.*

5.2 Les nœuds

Chaque élément d'un document XML s'appelle un nœud. Chaque nœud peut avoir 0 ou 1 parent et 0 ou plusieurs enfants.

Dans l'exemple du point 5.1, le nœud "personnes" n'a pas de parent mais contient des enfants : les nœuds "personne" (sans S).

Nous pouvons représenter notre exemple sous la forme d'un arbre :

5.3 Validation d'un document XML

Pour être analysé, un document XML doit respecter les règles suivantes :

- Le document doit contenir au moins une balise
- Le document ne doit posséder qu'un seul nœud racine. Dans l'exemple précédent, nous ne pourrions pas avoir deux éléments *personnes*.
- Chaque balise doit respecter les règles suivantes :
  - Le nom ne doit pas commencer par un chiffre ou un signe de ponctuation
  - Le nom ne doit pas commencer par xml
  - Le nom ne doit pas contenir d'espace
  - Le nom doit être écrit en minuscule
- Chaque balise ouvrante doit avoir une balise fermante
- Les balises ne peuvent pas se chevaucher (Exemple interdit : <personne><prenom></personne></prenom>)
- Les valeurs des attributs doivent être obligatoirement encadrées avec des simples ou doubles quotes
- Les caractères "<", ">" et "&" doivent être remplacés par leur entités HTML

5.3.1 La DTD

La DTD (*Document Type Definition*) permet de contrôler la validité du document en fixant des règles (la grammaire) concernant sa structure.

Par exemple : le nœud <personne> doit obligatoirement avoir un nœud <poids> avec un attribut "unite".

La DTD doit être créée dans un fichier séparé du fichier XML.

Par exemple, pour valider notre document du point 5.1, la DTD correspondante est :

```
<?xml encoding="UTF-8"?>
<!ELEMENT personnes (personne)+>
<!ELEMENT personne (nom,prenom,poids)>
<!ATTLIST personne id NMTOKEN #REQUIRED>
<!ATTLIST poids unite NMTOKEN #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT poids (#PCDATA)>
```

La DTD permet de définir les règles suivantes :

- Plusieurs nœuds personne dans le nœud racine personnes
- Une personne doit contenir un nom, un prenom et un poids
- Le nœud personne doit obligatoirement avoir un attribut "id"
- Le nœud poids doit obligatoirement avoir un attribut "unite"

Remarque : nous nous arrêtons ici dans le cadre de ce rappel concernant le XML / DTD.

5.3.2 Les schémas XML (XSD)

Les schémas XML, appelés également XSD ont été mis en place par le W3C pour ajouter des contrôles qui ne sont pas faits par les DTD. Le W3C conseille maintenant d'utiliser XSD au lieu d'une DTD.

Ce langage repose sur une syntaxe XML donc il est plus verbeux et moins lisible qu'une DTD.

Par exemple, pour valider notre document du point 5.1, le XSD correspondant est :

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="personnes">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="personne">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nom" type="xs:string" />
              <xs:element name="prenom" type="xs:string" />
              <xs:element name="poids">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:unsignedByte">
                      <xs:attribute name="unite" type="xs:string" use="required" />
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:unsignedByte" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

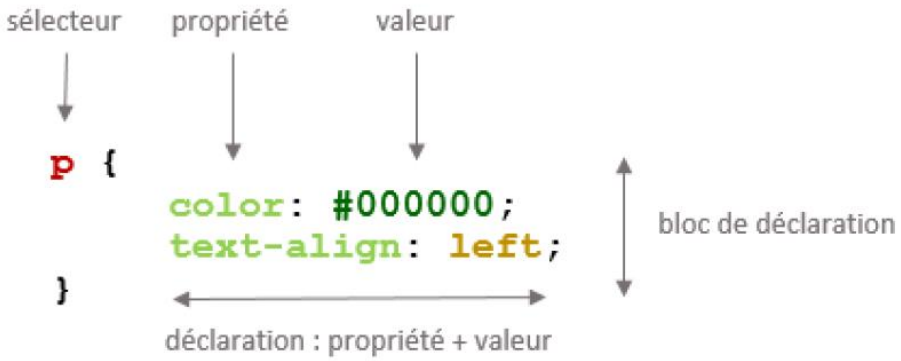
On remarque rapidement que le XSD est beaucoup plus verbeux !

Remarque : nous nous arrêtons ici dans le cadre de ce rappel concernant le XML / XSD.

6. Le CSS

Le CSS (*Cascading Style Sheets*) est un langage qui décrit la présentation de documents HTML. Il s'occupe ainsi de la mise en forme des pages. Son principal intérêt est de séparer la structure d'un document (HTML) de sa présentation (CSS), ce qui est intéressant pour la maintenabilité d'un site.

6.1 Terminologies CSS



La structure ci-dessus est un ensemble de règles CSS appliqué sur l'élément HTML <p>.

Regardons en détail les différentes parties :

- **Sélecteur**

C'est le nom de l'élément HTML situé au début de l'ensemble de règles. Comme son nom l'indique, il permet de sélectionner les éléments sur lesquels appliquer le style souhaité.

- **Déclaration**

La déclaration contient une ou plusieurs propriétés.

- **Propriétés**

Les propriétés permettent de mettre en forme un élément HTML. Dans l'exemple ci-dessus, nous utilisons la propriété *color*, qui permet de définir la couleur du texte, et la propriété *text-align*, qui permet de définir l'alignement du texte.

- **Valeur de la propriété**

À droite de la propriété, après les deux points, nous avons la valeur de la propriété.

En plus de ces 4 parties, nous pouvons également noter que :

- Chaque ensemble de règles (sans le sélecteur), doit être entre accolades
- La propriété est séparée de ses valeurs par le symbole deux points (:)
- Il faut utiliser un point-virgule (;) entre chaque déclaration.

6.2 Emplacements des styles

Il existe trois façons de définir un style CSS :
- Directement sur la balise HTML
- Dans la partie <head> de la page
- Dans un fichier externe

Il est fortement conseillé de gérer un fichier externe pour définir les styles CSS.

6.2.1 Définition d'un style au niveau d'une balise

Pour définir un style directement sur une balise HTML, il faut utiliser l'attribut *style.*
Un style définit de cette manière est appliqué sur la balise contenant la définition (et seulement cette balise).

```
<p style="color:red;text-align:center;">
   Voici un paragraphe de couleur rouge et dont le texte est centré.
</p>
```

6.2.2 Définition d'un style au niveau d'une page

Pour définir un style au niveau de la page, il faut utiliser les balises HTML <style></style> dans la partie <head> du document HTML.

```
   <head>
      <style>
         p {
            color:red;
            text-align: center;
         }
      </style>
   </head>
```

Dans cet exemple, le style va être appliqué sur toutes les balises HTML du document.

6.2.3 Définition d'un style dans un fichier externe

La dernière possibilité consiste à créer un fichier avec l'extension .css et qui va contenir l'ensemble des styles CSS pour la page (voir le même le site entier !).

Pour lier le fichier au document HTML, il faut utiliser la balise <link> dans la partie <head> du document HTML.

```
<link rel="stylesheet" type="text/css" href="styles.css"/>
```

6.3 Notion de cascade

Il existe un ordre de priorité dans la définition des styles CSS (le terme *Cascading* dans l'acronyme CSS).

Voici un exemple qui va nous permettre de comprendre ces règles.

```
<html>
   <head>
      <style type="text/css">
         p {
            color: red;
         }
      </style>
   </head>
   <body>
      <p style="color: green">Bonjour !</p>
   </body>
</html>
```

En testant ce code dans un navigateur, nous remarquons que le texte « Bonjour ! » est écrit en vert.

En effet c'est toujours la spécification la plus localisée qui est prioritaire en CSS.

En résumé, si un style est défini aux trois emplacements :
- Dans une feuille de style externe
- Dans la feuille de style dans l'en-tête
- Dans le style de la balise

Alors c'est le style définit au niveau de la balise qui sera appliqué.

6.4 Les sélecteurs

Pour associer le style aux différents éléments du document HTML, on utilise la notion de sélecteur.

6.4.1 Les sélecteurs simples

6.4.1.1 Les sélecteurs de type

Ce sélecteur permet de cibler les éléments qui correspondent au nom de la balise HTML.

```
p {
   color:red;
}
```

Dans cet exemple, nous appliquons le style sur toutes les balises HTML .

6.4.1.2 Les sélecteurs de classe

Ce sélecteur permet de cibler les éléments en fonction de leur classe (Attribut *class* sur l'élément HTML). Il faut utiliser le point (.) pour cibler une classe.

CSS :

```
.rouge {
   color:red;
}
```

HTML :

```
<p class="rouge">Ce texte est en rouge</p>
<p>Ce texte n'est pas en rouge</p>
```

Dans cet exemple, le style sera appliqué sur les éléments HTML ayant la classe « rouge ».

6.4.1.3 Les sélecteurs d'identifiant

Ce sélecteur permet de cibler les éléments en fonction de leur identifiant (Attribut *id* sur l'élément HTML). Il faut utiliser le dièse (#) pour cibler un identifiant.

CSS :

```
#exemple {
   color:red;
}
```

HTML :

```
<p id="exemple">Ce texte est en rouge</p>
<p>Ce texte n'est pas en rouge</p>
```

Dans cet exemple, le style sera appliqué sur l'élément HTML ayant l'identifiant « rouge ».
Attention : l'identifiant doit être unique dans tout le document !

6.4.1.4 Le sélecteur universel

Ce sélecteur est le sélecteur universel, il permet de cibler tous les nœuds d'un document.

```
* {
   font-family: Verdana;
}
```

6.4.1.5 Les sélecteurs d'attribut

Ce sélecteur permet de cibler des éléments en fonction de la valeur d'un de leurs attributs.

Voici une liste des sélecteurs d'attribut les plus utilisés :

Syntaxe	Explications	Exemple
[attr]	Permet de cibler un élément qui possède l'attribut	`a[title]`
[attr=valeur]	Permet de cibler un élément qui possède l'attribut avec la valeur indiquée	`div[lang="fr"]`
[attr^=valeur]	Permet de cibler un élément qui possède l'attribut dont la valeur commence par la valeur indiquée	a[href^="https://]
[attr$=valeur]	Permet de cibler un élément qui possède l'attribut dont la valeur termine par la valeur indiquée	a[href$=".fr"]

6.4.2 Les combinateurs

6.4.2.1 Les sélecteurs de voisin direct

Le combinateur **+** permet de sélectionner les nœuds qui suivent immédiatement un élément donné.

```
p + span  {
   color:red;
}
```

Dans cet exemple, nous appliquons le style sur la balise HTML qui suit immédiatement la balise .

6.4.2.2 Les sélecteurs de voisins

Le combinateur **~** permet de sélectionner les nœuds qui suivent un élément et qui ont le même parent.

```
p ~ span  {
   color:red;
}
```

Dans cet exemple, nous appliquons le style sur toutes les balises HTML qui suivent la balise et qui ont le même élément parent.

6.4.2.3 Les sélecteurs d'éléments fils

Le combinateur **>** permet de sélectionner les nœuds qui sont des fils directs d'un élément donné.

```
p > span  {
   color:red;
}
```

Dans cet exemple, nous appliquons le style sur toutes les balises HTML qui sont situées dans une balise .

6.4.2.4 Les sélecteurs d'éléments descendants

L'espace est le combinateur qui permet de sélectionner les nœuds qui sont des descendants (fils directs ou non) d'un élément donné.

```
p span  {
   color:red;
}
```

Dans cet exemple, nous appliquons le style sur toutes les balises HTML qui sont situées dans une balise .

6.4.3 Les pseudo-classes

Les pseudo-classes permettent de cibler des éléments selon un état. La syntaxe utilise le deux points (:).
Voici une liste des pseudo-classes plus utilisées :

Syntaxe	Explications	Exemple
:link	Cible les liens à l'intérieur d'éléments	`a:link`
:visited	Cible un lien déjà visité par l'utilisateur (cette pseudo-classe s'applique seulement sur la balise <a>)	`a:visited`
:hover	Cible un élément survolé par le pointeur de la souris	a:hover
:active	Cible un élément activé par l'utilisateur	a:active

6.4.4 Les pseudo-éléments

Un pseudo-élément est un mot-clé qui permet de mettre en forme certaines parties de l'élément ciblé. La syntaxe utilise un double deux points (::).

La plupart des pseudo-éléments sont encore en cours de normalisation, mais vous trouverez ci-dessous quelques pseudo-éléments validés.

Syntaxe	Explications	Exemple
::first-line	Permet de sélectionner la première ligne d'un élément	`p::first-line`
::first-letter	Permet de sélectionner la première ligne d'un élément	`p::first-letter`

::before	Créé un pseudo-élément qui sera le premier enfant de l'élément sélectionné. Il est souvent utilisé avec la propriété *content.*	`a:before {`
		`  content: "♥";`
		`}`

::after	Créé un pseudo-élément qui sera le dernier enfant de l'élément sélectionné. Il est souvent utilisé avec la propriété *content.*	`a:after {`
		`  content: "→";`
		`}`

6.5 Les propriétés CSS

6.5.1 Les mesures

Avant d'entrer dans la liste des propriétés, nous allons d'abord parler des types de valeurs.

Les valeurs de dimensions peuvent être exprimée en :

- **px** : représente un pixel
- **em** : la taille de la police. Permet de s'adapter au redimensionnement du texte
- **mm** : millimètres
- **%** : pourcentage de la taille de l'élément

Les couleurs peuvent être exprimée par leur nom, par leur code Hex, RGB, RGBA, HSL ou HSLA. RGB représente les couleurs avec leur pourcentage de rouge, de vert et de bleu. Le A de RGBA et HSLA représente l'opacité de la couleur. HSL est une notation de couleur demandant la coloration, la saturation et l'éclairage de la couleur.

- **red**: la couleur rouge
- **rgb(255,0,0)** ou rgb**(100%,0,0)** : rouge en code RGB
- **#ff0000** : le rouge en hexadécimal (ff signifie le maximum de rouge, 00 pas de vert et 00 pas de bleu)
- **rbg(255,0,0,0.2)** : du rouge presque transparent (20 % d'opacité)
- **hsl(0,100%,50%)** : du rouge pur en code HSL

6.5.2 Positionnement

Propriété	Explications
position : static	Le bloc aura sa position naturelle.
position : relative	Le bloc sera positionné par rapport à sa position naturelle au sein de la page.
position : absolute	Le bloc sera positionné par rapport à la « position zéro » de la page web. Cette position est représentée par le coin supérieur gauche de la page web.
position : fixed	Le bloc sera positionné par rapport au bord de l'écran du navigateur. Il sera donc « fixe » sur l'écran du visiteur

6.5.2.1 top, bottom, left, right et z-index

Propriété	Explications
top : 10px	Le bloc sera à 10 pixels du haut de son référentiel de position
bottom : 1em	Le bloc sera à 1 ligne du bas de son référentiel de position
left : 3mm	Le bloc sera à 3 millimètres de la gauche de son référentiel de position
right : 50%	Le bloc sera positionné à la moitié de la taille du référentiel en partant de la droite
z-index : 18	Le bloc sera placé par-dessus les autres blocs qui ont un z-index inférieur.

6.5.3 Display

Propriété	Explications
display : none	Ne plus afficher l'élément. Il sera ignoré dans la construction de la page
display : inline	Afficher l'élément comme un élément en ligne
display : block	Afficher l'élément comme un élément bloc
display : inline-block	Afficher comme un bloc sans retours de ligne, ni avant ni après.

6.5.4 Couleurs et images d'arrière-plan

Propriétés	Explications	Exemple

color	Permet de fixer la couleur d'avant-plan	`color:red`
		`color:rgb(120,0,255)`
		`color:#FF00FF`

background	Permet de fixer la couleur d'arrière-plan.	`background: red`
		`background: rgb(255,0,0)`
		`background: #FF0000`

background-image	Permet de fixer une image d'arrière-plan. Une image d'arrière-plan prévaut sur la couleur d'arrière-plan.	`background-image: url('http://host/reps/fichier.html')`

6.5.5 Polices de caractères

Propriétés	Explications	Exemple
font-family	Cet attribut permet de spécifier la police de caractère à utiliser. Au cas où la police n'existerait pas, il est possible de spécifier une liste de choix. La première police trouvée à partir de la gauche est appliquée.	`font-family : verdana, arial`
font-weight	Permet de fixer le poids de la police de caractères.	`font-weight: bold`
font-decoration	Permet de décorer la police de caractères	`font-decoration: underline` `font-decoration: strike`

6.5.6 Mise en forme de blocs de caractères

Propriétés	Descriptif	Exemple

text-align	Cet attribut spécifie l'alignement du bloc texte.	`text-align: center`
		`text-align: left`
		`text-align: right`
		`text-align: justify`

text-indent	Permet de définir le retrait de la première
	ligne du bloc de texte.
		`text-indent: 10pt`
		`text-indent: 15px`
		`text-indent: 5%`

6.5.7 Gestion des boîtes

Propriétés	Descriptif	Exemple

width
min-width
max-width
	Permet de spécifier la largeur de l'élément, taille de bordure comprise.	`width: 80pt`
		`width: 100px`
		`width:80%`

height **min-height** **max-height**	Permet de spécifier la hauteur de l'élément, taille de bordure comprise.	`height: 25%`
padding **padding-top** **padding-bottom** **padding-left** **padding-right**	Le padding correspond à l'espace interne d'une cellule (entre le contenu et la bordure). Plutôt que de spécifier le padding pour les quatre côtés, on peut le spécifier individuellement coté par coté. Cette marge interne prend comme couleur, celle de l'arrière-plan de l'élément considéré.	`padding: 5px` `padding-bottom: 10px`

margin
margin-top
margin-bottom
margin-left
margin-right
	La marge externe n'est pas prise en compte dans la taille de l'objet. Elle est de plus transparente. Vous pouvez aussi la spécifier individuellement pour chaque côté.	`margin: 10px`
		`margin-left:10%`
		`margin-right: 10%`

border
border-style
border-width
border-color
border-top-style
	Ces attributs permettent de spécifier les caractéristiques de la bordure (son style de bordure, sa taille et sa couleur)	`border: 2px solid #FF00FF`
		`border-style: solid`
		`border-width: 2px`
		`border-color: #FF00FF`
		`border-top-style: ridge`

7. Le Web responsive

L'objectif du Responsive Web Design est de s'assurer que les pages aient un bon visuel sur différents terminaux (écran de PC, tablette, smartphone etc…).

Cette notion n'utilise pas de nouvelle technologie mais repose sur HTML et CSS.

En effet, le CSS propose le module media queries qui permet de définir des règles CSS en fonction de conditions comme le media (écran, synthèse vocale, imprimante), la largeur de l'écran, l'orientation (portrait, paysage), la résolution etc…

Un site web responsive est un site qui s'affiche proprement sur différents terminaux avec la même base HTML et avec un CSS permettant d'ajuster l'affichage suivant le terminal.

7.1 Règles pour les media queries en CSS

Pour définir une règle, il faut utiliser la syntaxe suivante :

```
@media screen and (max-width: 1280px)
{
   /* Écrivez vos styles ici */
}
```

Dans cet exemple, nous utilisons deux règles :
- media
- max-width

Cette règle permet ainsi de cibler les écrans « classique » (ordinateur, tablette, etc…) avec une largeur maximum de 1280 px.

Vous trouverez ci-dessous quelques règles permettant de construire des media queries.

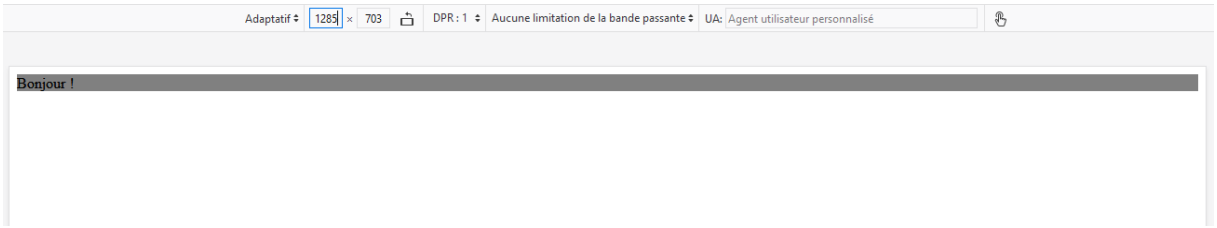
Nom de la règle	Explications
media	Type d'écran de sortie. La valeur peut être : • all : Tous les écrans • handheld : Mobile • print : Impression • projection : Projecteur • screen : Écran classique • tv : Télévision
width **min-width** **max-width**	Largeur de la zone d'affichage
orientation	Orientation de terminal (portrait/paysage)

7.2 Exemple complet

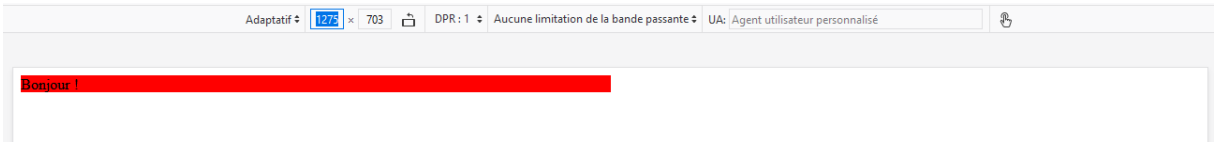
Voici un exemple afin de bien comprendre le fonctionnement :

```
<html>
   <head>
      <style type="text/css">
         div {
            width: 100%;
            background-color: gray;
         }
         @media screen and (max-width: 1280px)
         {
            div {
               width: 50%;
               background-color: red;
            }
         }
      </style>
   </head>
   <body>
      <div>Bonjour !</div>
   </body>
</html>
```

En affichant cette page dans le navigateur, nous avons le rendu suivant :



Si nous modifions la largeur de la zone d'affichage (en utilisant la « Vue adaptive » dans Firefox par exemple), nous remarquons que l'affichage de la div change dès qu'on arrive en dessous de 1280 pixels.



En effet, le style par défaut pour la div indique une largeur de 100% et une couleur de fond *gray*.

La media queries indique que pour les écrans « classiques » et une largeur maximum de 1280px, il faut appliquer le style contenu dans la règle. Ainsi la div passe sur une largeur de 50% et une couleur de fond *red*.

7.3 Utilisation d'un framework

Le développement d'un site web responsive avec des media queries est puissant mais prend beaucoup de temps.

Pour éviter de tout faire manuellement, il est possible d'utiliser un framework. Un framework (framework CSS pour notre sujet) étend les possibilités d'un langage en ajoutant des fonctionnalités et des composants supplémentaires. Tout ce que fait un framework, peut également être écrit en en utilisant seulement le langage initial, mais un framework permet d'accélérer le développement.

Les avantages d'un framework CSS sont :

- Le support du Responsive Web Design
- La rapidité de développement grâce à un ensemble de classes CSS prédéfinies
- Des outils et modules supplémentaires permettant d'ajouter des fonctionnalités rapidement

Il existe tout de même quelques inconvénients :

- Il faut apprendre comment utiliser le framework et sa philosophie.
- Le chargement d'un framework peut être assez lourd. Il faut faire attention de ne pas utiliser un framework « usine à gaz » si c'est seulement pour afficher un joli bouton !

Nous pouvons citer par exemple : Pure.CSS, Semantic UI, Materialize, Bootstrap, etc…

7.4 Introduction à Bootstrap

Nous allons présenter dans ce cours le framework Bootstrap, créé par Twitter en 2011.

Il est très connu et il fournit, en plus de la partie purement responsive, de nombreux modules supplémentaires. Nous pouvons ainsi citer :

- Un système de grille
- Un module pour des menus
- Un carrousel
- Un module de gestion d'info-bulle,
- Etc…

7.4.1 Un premier exemple

Pour commencer, voici un premier exemple. Nous ajoutons Bootstrap à l'aide d'un CDN (*Content Delivery Network)*.

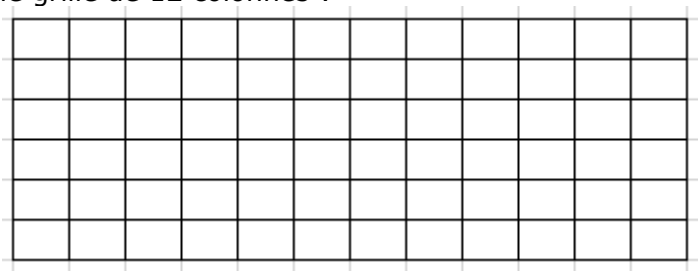
```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <!-- Meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

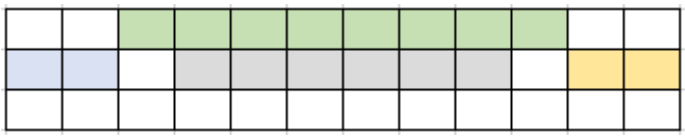
7.4.2 Un système de grille

Bootstrap propose une organisation spatiale de la page web basé sur un système de grille. La page est ainsi « découpée » en lignes et colonnes et permet de placer assez rapidement les différents éléments de la page (logo, bannière, menu, etc…)

Bootstrap utilise une grille de 12 colonnes :



Nous pouvons organiser le contenu en utilisant les cellules et les lignes :



Pour organiser la grille, il faut connaitre plusieurs classes :

- La classe **container** qui va « contenir » les lignes et les colonnes
- La classe **row** permet de représenter une ligne
- Les classes **col-x** permettent de représenter x colonnes. (par exemple : col-1 représente 1 colonne et col-4 en représente 4)

Avec ces classes nous pouvons commencer à écrire l'organisation de la page :

```
<!DOCTYPE html>
<html lang="fr">
   <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
      <title>Une grille</title>
   </head>
      <body>
         <div class="container">
            <div class="row">
               <div class="col-2"></div>
               <div class="col-8 bg-success">A</div>
               <div class="col-2"></div>
            </div>
            <div class="row">
               <div class="col-3 bg-info">B</div>
               <div class="col-6 bg-secondary">C</div>
               <div class="col-3 bg-warning">D</div>
            </div>
         </div>
      </body>
</html>
```

Avec notre code, les colonnes seront toujours affichées, mêmes sur les petits écrans :

Bootstrap étant responsive, il propose également des préfixes de classe qui vont activer le comportement de la classe suivant la largeur de l'écran.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Nous modifions notre code pour indiquer :

- Par défaut, chaque cellule fera 12 colonnes (donc la ligne entière)
- Dès que l'écran fait plus de 960px de large -préfixe **lg**) les colonnes reprennent le format défini sur notre grille

```
<div class="container">
   <div class="row">
      <div class="col-12 col-lg-2"></div>
      <div class="col-12 col-lg-8 bg-success">A</div>
      <div class="col-12 col-lg-2"></div>
   </div>
   <div class="row">
      <div class="col-12"></div>
   </div>
   <div class="row">
      <div class="col-12 col-lg-3 bg-info">B</div>
      <div class="col-12 col-lg-6 bg-secondary">C</div>
      <div class="col-12 col-lg-3 bg-warning">D</div>
   </div>
</div>
```

Nous avons maintenant une disposition qui change en fonction de la largeur de l'écran.
Si la largeur de l'écran est inférieure à 960px, l'affichage est le suivant :

Si la largeur de l'écran est supérieure à 960px, la disposition change dynamiquement :

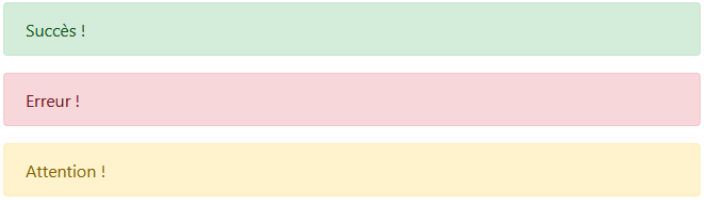
7.4.3 De nombreux composants

Bootstrap inclut de nombreux composants, nous allons en présenter quelques-uns.

- **Des styles pour les retours sémantiques**

La classe *alert* permet d'afficher un message de retour contextuels pour des actions de l'utilisateur. La classe *alert* est combinée avec une classe permettant de choisir la couleur.

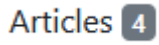
```
<div class="alert alert-success" role="alert">
 Succès !
</div>
<div class="alert alert-danger" role="alert">
  Erreur !
</div>
<div class="alert alert-warning" role="alert">
  Attention !
</div>
```



- **Des styles pour afficher des badges ou compteurs**

La classe *badge* permet de créer un badge pouvant contenir une information (un état par exemple) ou un compteur.

```
<p>Articles <span class="badge badge-secondary">4</span></p>
```



- **Des styles pour afficher les boutons**

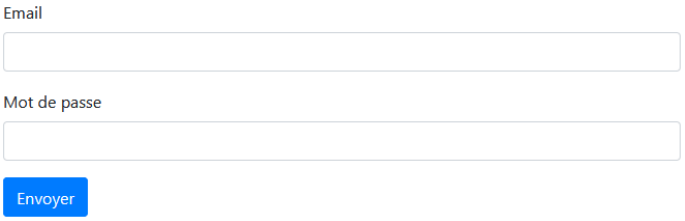
La classe *btn* permet d'afficher un bouton. La classe est combinée avec une classe permettant de choisir la couleur.

```
<button type="button" class="btn btn-success">Activer</button>
<button type="button" class="btn btn-warning">Désactiver</button>
<button type="button" class="btn btn-danger">Supprimer</button>
```

- **Des styles pour les formulaires**

Les classes *form-xxx* permettent de styliser les champs de formulaires.

```
<form>
   <div class="form-group">
      <label for="mail">Email</label>
      <input type="email" class="form-control" id="mail">
   </div>
   <div class="form-group">
      <label for="password">Mot de passe</label>
      <input type="password" class="form-control" id="password">
   </div>
   <button type="submit" class="btn btn-primary">Envoyer</button>
</form>
```



7.4.4 Pour aller plus loin

Nous venons seulement de survoler Bootstrap, il y a de nombreuses fonctionnalités qui restent à découvrir comme par exemple un module pour gérer un carrousel, un autre pour afficher des menus, un autre pour afficher des fenêtres modales, etc…

Le site internet de Bootstrap (https://getbootstrap.com/) contient une documentation complète ainsi que des exemples et tutoriels.