Polycopié de cours R102 : Interfaces Web

Benjamin Hellouin de Menibus, IUT d'Orsay

Version du 9 septembre 2021

Avant-propos

Ce polycopié est un support de cours pour le cours R102 : Interfaces Web, destiné aux étudiants de premier semestre du BUT informatique de l'IUT d'Orsay. Il a bénéficié des commentaires et corrections de Mathilde Mousset et Anaïs Artiges.

Les corrections d'erreurs ou suggestions sont acceptées, avec gratitude, à l'adresse suivante : `benjamin.hellouin-de-menibus@u-psud.fr`. Si vos corrections sont utiles et significatives, votre nom sera immortalisé pour les années à venir dans ce polycopié.

Table des matières

1. https ://creativecommons.org/licenses/by/4.0/

1 Fonctionnement du cours

1.1 Organisation et matériel

Tout le matériel du cours (polycopié, sujets et TP, rendus) peut-être trouvé sur la plateforme Moodle : `https://cours.iut-orsay.fr`. Pour les TP, le sujets seront toujours à l'adresse `https://webdev.iut-orsay.fr/web`. Des sujets papiers seront fournis sur demande.

En cas de question / problème, votre premier interlocuteur est votre chargé de TP. En cas de désaccord ou d'urgence, vous pouvez m'envoyer un mail à `benjamin.hellouin-de-menibus@universite-paris-saclay.fr`.

1.2 Planning

Cours 7 séances de 1h

TP 7 séances de 2h ; trois ou quatre rendus notés

Interrogation sur le contenu du cours.

Le projet de Web et communication (CWAC) a lieu en parallèle. Il donne lieu à une note séparée. Une des séances de TP servira à faire le logo pour votre projet. Les chargés de TP ne sont pas responsables du projet, mais vous avez le droit de leur demander conseil ponctuellement.

1.3 Assiduité & notation

La présence aux TP est **obligatoire** (comme pour tous les cours). Les chargés de TP feront l'appel.

Si vous avez une absence ou un retard qui n'est pas de votre fait, il est important de le justifier **auprès du secrétariat**. Pour l'interrogation en particulier, nous devons mettre un zéro si l'absence est injustifiée !

La note du cours se compose des notes suivantes :

50% Trois ou quatre rendus de TP (le TP sur trois séances compte triple)

15% Quiz sur le contenu du cours

35% La note de Web du projet CWAC

Des malus seront donnés pour les défauts d'assiduité : absences, rendus en retard, non-respect des consignes, séances le nez dans le téléphone, etc. Votre chargé de TP précisera les règles.

2 Fichiers et documents

2.1 Encodage de texte

La mémoire d'un ordinateur enregistre l'information sous forme de 0 et de 1 (bits). Afin de pouvoir enregistrer du texte il convient de déterminer la fonction qui associe des caractères à des groupes de bits : l'**encodage**.

Par exemple, un des plus anciens encodages est l'**ASCII** qui associe un caractère à un octet (8 bits), mais qui ne peut représenter que peu de caractères. Exemple d'ecodage ASCII : $\begin{array}{l} 00100000\ (64) \rightarrow \text{␣ (espace)} \\ 01000001\ (65) \rightarrow \text{A} \end{array}$.

Aujourd'hui, on trouve surtout l'encodage **UTF-8** qui utilise de 1 à 4 octets et qui permet de représenter tous les alphabets, et bien plus - environ 138000 caractères. Quand le choix vous est offert, utilisez UTF-8.

Quelques encodages concurrents aux noms tels que ISO-8859-1 ou Windows-1251 persistent. Quand vous rencontrez des caractères comme Ã_ Ã© Ã® ou encore xE0 xEE, votre éditeur de texte lit le texte dans le mauvais encodage. Il faut changer le mode de votre éditeur ou convertir le texte.

2.2 Format et extensions de fichiers

Le **format** d'un fichier décrit le type d'information qu'il contient (texte, musique, programme...) ainsi que la manière dont elle est "rangée" - et, par conséquent, quels logiciels peuvent l'ouvrir. Le format d'un fichier peut être reconnu à son **extension**.

Le **texte brut** est le format textuel le plus basique. il ne contient que des caractères sans mise en forme. Il peut être lu et modifé sur tous les systèmes (une fois le bon encodage choisi). Il est utilisé pour la communication simple (mails, fichiers README, etc.) et pour écrire du code, des fichiers de configurations, et plus généralement tous le contenu textuel manipulé par des programmes.

Exemples : `.txt` indique du contenu textuel. De très nombreux autres formats sont écrits en texte brut avec des conventions supplémentaires, comme le HTML et le CSS.

Le **texte mis en forme** (ou document de traitement de texte) contient du texte brut, mais également des informations supplémentaires d'apparence : mise en forme, couleur, images, etc. Concrètement, il s'agit le plus souvent d'une archive (compressée) qui comprend le contenu textuel et différents fichiers pour les autres informations. C'est le format adapté pour créer et modifier des documents, mais pas pour les afficher.

Exemples : `.docx` pour un document Microsoft Word (récent), `.odt` pour un document OpenOffice, etc.

Le **format PDF** et les formats similaires servent à décrire un document mis en page (texte ou présentation) de telle sorte que le document apparaisse exactement de la même manière sur tous les systèmes et à l'impression. En contrepartie, le fichier est plus difficile à modifier.

Exemples : `.pdf`, `.xps`, `.ps`, etc.

Le format **HTML** est un langage qui décrit la structure du document à l'aide de balises. Le fichier lui-même est écrit en texte brut. L'affichage n'est pas prévu pour une taille fixée mais change suivant la taille de l'écran. Par conséquent, il n'est pas en général adapté à l'impression, sauf s'il a été explicitement prévu pour.

Exemples : `.html` pour toutes les pages Web, mais également des mails « enrichis ».

2.3 Utilisation des différents formats

1. Utilisez toujours du texte brut pour des données manipulées par des programmes (code source, entrées/sorties).
2. Pour transmettre des documents de traitement de texte ou des présentations, utilisez un format d'affichage comme le PDF pour vous assurer [2] qu'on puisse l'ouvrir partout avec son apparence finale. Une seule exception à cette règle : si le document doit rester modifiable.
3. Pour un site Web, il vaut mieux en général mettre le maximum de contenu au format HTML (accessible plus directement). Transmettre certains documents au format pdf est acceptable, surtout s'il est probable qu'on veuille les imprimer.

Concernant le point 2, il faut éviter de transmettre des informations dans des documents scannés, qui posent de gros problèmes d'accessibilité (personnes malvoyantes en particulier). Les pdf générés par les logiciels de traitement de texte n'ont pas ce problème.

3 Conception de documents

3.1 Quelques conseils de rédaction

3.1.1 Demander de l'aide - écrire un mail

Par mail, sur Discord, sur un chat… il vous arrivera de demander de l'aide (ou autre). La règle générale est d'**économiser le temps et le travail de la personne à qui vous écrivez**, en lui donnant les infos dont elle a besoin.

Erreurs courantes à ne pas commettre :

- Ne demandez pas la permission ! Posez votre question, et je n'y répondrai pas si je ne veux pas.
- Envoyez-moi votre site entier. Les captures d'écran peuvent aider mais ne remplacent pas.
- Evitez les questions vagues « Est-ce que c'est normal ? ». Que voulez-vous ?
  - Dites-moi où se situe le problème (quelle page, quel bloc HTML).
  - Si vous voulez **comprendre**, dites-moi le comportement que vous attendez.

2. dans la mesure du possible…

Jean-Loup Zeur (10h02)
> Bonjour Monsieur, est-ce que je peux vous demander de l'aide ?
> C'est super important parce que le projet doit être rendu demain
Benjamin Hellouin (10h25)
> Dis-moi
Jean-Loup Zeur (10h28)
> Ben le site marche pas quand je clique sur le bouton
Benjamin Hellouin (10h45)
> Où puis-je voir ton code ? Quelle page ? Quel bouton ? Qu'est-ce qui ne marche pas ?
Jean-Loup Zeur (10h49)
... en train d'écrire ...

FIGURE 1 – Conversation Discord à peine inventée : ça m'arrive une fois par semaine

— Si vous voulez **faire quelque chose**, dites-moi quoi.

Il faut pouvoir comprendre facilement qui vous êtes (attention aux pseudos Discord) et il faut indiquer **la raison** de votre mail aussitôt que possible. Une erreur courante est, par peur d'être impoli, de donner de nombreux détails du contexte avant d'en venir au fait. Faites des phrases courtes et simples, et sautez une ligne quand vous changez de sujet.

L'orthographe et la grammaire sont des sources de difficultés fréquentes. Pour des mails professionnels, il est important de bien se relire avant l'envoi. Pour les mails importants, par exemple destinés à l'extérieur de votre entreprise, n'hésitez pas à demander à un camarade ou un responsable hiérarchique de vous relire. C'est de cette manière qu'on apprend.

Enfin, votre mail sera plus agréable et facile à traiter si vous incluez toutes les informations nécessaires. N'utilisez pas de vocabulaire technique si la personne risque de ne pas le connaître, ou expliquez-le. Si vous faites référence à un document, mettez-le en lien ou en pièce jointe.

⚠ Les enseignants reçoivent des dizaines de mails par jour. Si votre demande nécessite une réponse immédiate, n'hésitez pas à placer un "Urgent" dans l'objet. N'en abusez pas.

3.1.2 Documents texte

Au début d'un document texte, on doit trouver :

- Un **titre** qui indique le sujet du document et le but dans lequel il a été écrit.
- Un **auteur**, éventuellement avec son rôle (groupe de TP, entreprise où vous travaillez, etc.).
- Une **table des matières** si le document dépasse 4 ou 5 pages. Tous les logiciels de traitement de texte permettent de faire une table des matières automatique avec des liens hypertexte (essayez sur ce polycopié).

de : ke.jie@u-psud.fr
à : benjamin.hellouin-de-menibus@u-psud.fr
objet : [CODIN] Note du TP3

Monsieur Hellouin,
Je suis dans le groupe TP1E et vous êtes mon enseignant en CODIN. Je vous contacte pour discuter de ma note du TP3. Ma note est de 13 alors que le total de mes points fait 15, comme vous pouvez voir : ⟨lien⟩. Pourriez-vous vérifier ?

Cordialement,
Ke Jie.

FIGURE 2 – Un exemple de mail bien rédigé

Pour différencier le sujet et le but du document on peut avoir un titre et un sous-titre. Par exemple :

Développement d'un plug-in Thunderbird
d'aide à la rédaction de mails
Rapport de stage S4 effectué chez Mozilla

Le titre est souvent trop court pour mettre beaucoup d'informations. C'est le rôle que jouera l'**introduction**. L'introduction :

- donne plus en détail le sujet et le but du document dès les premières lignes, comme dans un mail (n'ayez pas peur de répéter votre titre) ;
- fournit les éléments de contexte nécessaires à comprendre le document (qu'est-ce qu'un plug-in ? Thunderbird ? Mozilla ?) ;
- annonce le contenu du document avec ses différentes parties.

Votre document doit donc (sauf s'il est vraiment très court) être séparé en sections, sous-sections... qui séparent le contenu dans ses différentes parties. Il est important d'utiliser les outils adaptés pour que la navigation entre ces sections soit facile (table des matières automatique, liens).

Un défaut de rédaction habituel, surtout en début de carrière, est de faire des textes trop compliqués ou trop techniques. Il est très important de se demander à qui votre document est destiné.

- Est-ce que la personne connaît tous vos termes techniques ou acronymes ? Sinon, expliquez-les la première fois, éventuellement avec un lien vers le site officiel. Si le lecteur doit faire une recherche Web pour vous suivre, c'est un échec.
- Par rapport à l'oral, rédiger professionnellement demande d'utiliser des **phrases courtes**. Une erreur courante consiste à faire plusieurs phrases séparées par des virgules, c'est ce que je fais dans cette phrase, le texte devient désagréable à lire.

Enfin, il est recommandé d'avoir :

- en haut de la page (**en-tête**), le titre (éventuellement raccourci) du document, l'auteur, éventuellement le titre de la partie actuelle pour les documents longs.
- en bas de la page (**pied de page**), le numéro de la page actuelle.

Ces éléments aident, surtout quand le document est imprimé, à retrouver facilement là où le lecteur en est.

3.2 Lisibilité et accessibilité

Un document est fait pour être lu ou vu, et il faut donc que vous vous assuriez que ce ne soit pas trop difficile pour votre lecteur. Ce n'est pas suffisant que vous réussissiez à le lire sur votre machine : vous êtes jeunes, proche de l'écran, sans lumière dans les yeux... Quelques règles à respecter :

Règle 1. Le texte doit être assez grand : 12 points pour un document texte, 22 points pour une présentation. On peut faire des exceptions pour des détails moins importants. Pour les présentations, rappelez-vous que vous ne savez pas la taille de l'écran sur lequel vous allez projeter.

Règle 2. Garder un constraste fort entre le texte et le fond. Cela signifie du texte foncé (noir ou presque) sur un fond clair, ou du texte clair (blanc ou presque) sur un fond foncé. Méfiez-vous particulièrement des imprimantes et des vidéoprojecteurs qui peuvent mal afficher certaines couleurs : augmenter le contraste aide toujours.

Règle 3. Restez sobres. Les polices extravagantes et les couleurs sont à utiliser avec parcimonie : pour les titres par exemple, en aucun cas pour la majorité du texte. De même pour les grands aplats de couleurs vives.

Une situation classique est de chercher à mettre du texte sur une image, ce qui devient facilement illisible. Si on ne parvient pas à avoir un bon contraste, il existe d'autres méthodes : mettre un contour aux lettres, ou appliquer un fond semi-transparent derrière le texte.

Pour les documents devant être diffusés (y compris les pages Web), vous devez fournir un **texte alternatif** à vos images. Ce texte alternatif sera donné aux utilisateurs ne pouvant ou ne voulant pas accéder aux images (handicap visuel, connexion limitée...). Tous les logiciels et langages que nous utilisons ont une option dédiée. Pour la même raison, il ne faut pas qu'une information soit transmise uniquement par la couleur.

Enfin, il est recommandé de ne pas utiliser la justification du texte  qui crée des problèmes aux personnes ayant des difficultés de lecture.

3.3 Séparation structure - présentation

3.3.1 Définitions

Quelques définitions :

Contenu L'information transmise par le document. Il s'agit souvent surtout de texte, mais également d'images ou de vidéos (si elles ne sont pas utilisées purement pour la décoration)

Structure Le rôle joué par les différents éléments au sein du document. Par exemple : un titre général, un titre de section, un paragraphe, une légende, un élément de liste...

Présentation L'apparence que l'on choisit de donner aux différents éléments. Par exemple : gras, 20pt, bleu, fond noir, centré...

Le rôle primordial de la présentation est, avant de faire joli, de faire comprendre la structure du document. La règle d'or est donc : **les éléments ayant le même rôle doivent avoir la même apparence** et, dans la plupart des cas, une apparence différente pour des rôles différents.

Les logiciels de traitement de texte, de présentation, etc. ne sont pas des machines à écrire. Ils proposent beaucoup d'outils pour automatiser des tâches. Pour le problème qui intéresse (uniformiser l'apparence des éléments de même rôle), imaginons que vous voulez que vos titres de section soit en taille 20pt. Il y a deux solutions :

Approche « machine à écrire » Appliquer la taille de police 20pt à chaque titre de section.

Approche « traitement de texte » Appliquer le style "Titre 1" à chaque titre de section, puis modifier le style pour que la taille soit 20pt.

On voit que cette deuxième approche sépare deux étapes : indiquer quel est le rôle de l'élément (**structure**), puis indiquer l'apparence qu'ont tous les éléments du même rôle (**présentation**). C'est une illustration de la **séparation structure-présentation**.

Cette approche a de multiples avantages :

- Moins d'erreurs dû à la répétition des tâches (couleurs ou tailles légèrement différentes) ;
- Modifications futures plus faciles (ajout de nouveaux titres, changement de style général) ;
- Navigation facilitée (table des matières automatique, etc.) ;
- Document plus accessible (une personne aveugle peut savoir qu'il s'agit d'un titre).

3.3.2 Mise en pratique

Voici quelques exemples. Imaginons que vous voulez :

- Sauter une page ou ajouter de l'espace après les titres.

  Ne pas faire : touche entrée (saut de ligne) à répétition.

  Mais plutôt : trouver l'option de saut de page, modifier l'espacement dans le style correspondant.

- Centrer un objet.

  Ne pas faire : utiliser la barre espace à répétition ou des tabulations.

  Mais plutôt : utiliser l'option de centrage, éventuellement en groupant les objets.

– Mettre un fond de couleur à vos diapositives.

Ne pas faire : le faire à la main sur chaque diapositive, ou le faire sur la première et la copier.

Mais plutôt : utiliser le masque des diapositives.

De manière générale, ces conseils reviennent à **trouver les outils corrects pour chaque tâche, et les utiliser** (avec l'aide du Web ou de vos enseignants). Par exemple, il est important de ne jamais faire de table des matières ou d'alignement à la main.

3.4 Éléments de chartes graphiques

3.4.1 Identité visuelle

La grande majorité des entités (entreprises, associations, institutions...) ont une **identité visuelle** bien définie, autrement dit, un style cohérent entre tous les documents qu'elle produit. Par exemple, tous les courriers envoyés par votre banque ont une même apparence générale (logo, couleurs, polices...) qui vous permettent de l'identifier facilement.

Jusqu'ici, nous avons surtout insisté sur la cohérence dans l'apparence au sein d'un document. Des problèmes similaires se posent entre différents documents : comment faire pour que tous mes rapports aient le même style ? Comment faire pour que ce style soit cohérent avec celui du site Web ? Et si j'utilise un éditeur de texte différent ?

Cela peut être fait de manière informelle (typiquement, les collègues ont un ensemble de documents-types qu'ils modifient) mais les grandes entités définissent une **charte graphique** formelle pour tous types de documents. Une charte graphique fournit des indications sur l'apparence des différents éléments en fonction de leur rôle : titres, textes, page de garde, etc.

Idéalement, les indications d'une charte graphique permettent de réaliser un document conforme sans forcer l'usage d'un logiciel particulier. Cela demande de décrire certains éléments, comme les couleurs et les polices, dans un langage universel et indépedant du logiciel.

3.4.2 Couleurs

Il est presque impossible de décider « à l'oeil » si deux couleurs sur des documents différents sont vraiment identiques. Pour définir la couleur d'un objet, il est beaucoup plus sûr (et plus facile) d'utiliser un modèle de description des couleurs comme le modèle RVB (rouge, vert, bleu - RGB en anglais). Une couleur est donnée par trois valeurs, respectivement l'intensité de rouge, de vert et de bleu.

Suivant le contexte plusieurs conventions sont utilisées :

Pourcentages (100%, 50%, 0%)

Décimal 0-255 (255, 127, 0)

Hexadécimal #FF7F00

Il existe d'autres modèles : RVBA (RGBA en anglais) qui permet la transparence, CMYK, etc.

3.4.3 Polices

Une police d'écriture (*font family* en anglais) décrit la forme des lettres et des caractères utilisés pour écrire un texte, dans toutes leurs variations (différentes tailles, gras, italique, etc.). Chaque police porte un nom tel que *Times New Roman*, *Arial*, *Comic Sans*.

Le format le plus utilisé pour des polices informatiques est un ensemble de fichiers `.ttf`. Ces fichiers permet d'utiliser la même police dans des documents et présentations générés par des logiciels différents, ainsi que sur des documents HTML. Vous pouvez facilement trouver sur Internet des sites qui proposent une grande variété de polices gratuites ou payantes.

4 Bases du Web et du HTML

4.1 Internet, Web, HTML : définitions

Internet désigne un réseau informatique mondial. L'information y est transmise par un ensemble de protocoles (la suite TCP/IP, qui contient par exemple le protocole HTTP [3]. qui permettent divers usages tels que le courrier électronique, la voix sur IP (Skype, WhatsApp, Discord), le Web, etc.

le Web (ou World Wide Web, www) désigne l'ensemble des sites Web visitables depuis un navigateur grâce à un système d'adresses URL [4]. On y trouve des ressources de tout type, mais les pages Web à proprement parler sont des documents écrits en HTML, qui peuvent contenir des liens vers d'autres pages Web.

le langage HTML désigne le langage de balisage dans lequel les pages Web sont écrites.

Voilà un résumé (simplifié) de ce qui se passe quand on se connecte à un serveur Web [5].

1. Vous rentrez une URL dans votre navigateur (ou vous cliquez un lien vers cette URL), par exemple :

   `http` (protocole) `://` `www.iut-orsay.u-psud.fr` (nom de domaine) `/` `fr/vie_étudiante` (répertoire) `/` `associations.html` (page ou fichier demandé)

2. le navigateur se connecte au serveur Web correspondant au nom de domaine (la partie finissant souvent par `.com`, `.fr`...) et télécharge la ressource à l'emplacement indiqué (répertoire + nom de fichier, comme pour un fichier local). Cette partie utilise le protocole HTTP.

3. HTTP (*HyperText Transfer Protocol*) est un protocole qui définit la manière dont votre navigateur et le serveur contenant la page communiquent ensemble

4. *Uniform Ressource Locator* ; autrement dit, adresse qui permet de trouver une ressource.

5. Cette description est très simplifiée et ne décrit qu'une page Web **statique** : la page Web à un emplacement donnée est fixée. Une bonne partie du Web moderne est dynamique, ce qui signifie que chaque utilisateur peut recevoir une page Web différente même en suivant la même URL. Ce cours ne concerne que les pages Web statiques, le Web dynamique étant l'objet de futurs cours de programmation Web.

3. si le fichier est au format HTML, le moteur de rendu du navigateur interprète visuellement le code HTML et affiche le résultat à l'écran. Cela peut demander de télécharger d'autres ressources (images, feuille de style CSS...)

Quand une URL ne précise pas le nom du fichier, le navigateur cherche automatiquement à ouvrir la page `index.html` (souvent la page d'accueil) à l'emplacement indiqué. Si la page n'existe pas, le serveur (suivant la manière dont il est configuré) renvoie une erreur ou afficher une liste des fichiers disponibles à cet emplacement.

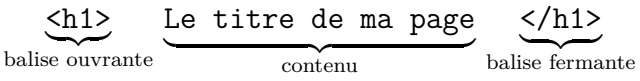
Une URL peut pointer sur n'importe quel type de fichier. Les navigateurs actuels sont capables d'afficher quelques types de fichier en plus du HTML (texte brut, fichiers pdf, vidéos, musique) ; si l'URL pointe vers un fichier d'un autre type, il propose le plus souvent de la télécharger.

HTML (HyperText Markup Language) est le langage dans lequel sont écrites les pages Web. Un document HTML (page Web) est écrit en texte brut et comporte du contenu textuel accompagné de **balises**. Les balises servent à définir la structure du document et le positionnement des élements, ajouter des liens, insérer d'autres ressources (images, vidéos, scripts), etc.

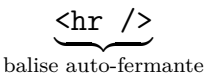
Le navigateur Web affiche un document HTML en "interprétant" visuellement le code HTML (ou source). Il s'agit bien du même document affiché de deux différentes manières. Sur la plupart des navigateurs, on peut afficher le code source d'une page Web après un clic droit sur la page.

4.2 Balises et éléments

Un document HTML est écrit en texte brut (extension `.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 de la manière suivante :



Un élément normal `h1`



Un élément vide `hr`

À gauche, l'élément `h1` (pour *heading of level 1*) correspond au titre principal de la page. Les deux balises indiquent le début et la fin du titre. À droite, l'élément `hr` (pour *horizontal rule*) correspond à une ligne horizontale qui représente un changement de sujet. Cet élément n'a pas besoin de contenu, l'unique balise est donc à la fois le début et la fin.

⚠ Chaque élément est soit normal, soit vide ; ce n'est pas un choix de votre part.

4.3 Structure d'une page HTML

Les éléments peuvent s'imbriquer les uns dans les autres, comme illustré Figure 3. Attention à l'ordre des balises :

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

Bonne imbrication
(`h1` est contenu dans `body`)

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

Erreur de syntaxe

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

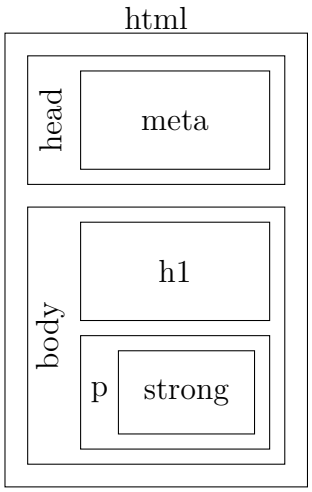


FIGURE 3 – Exemple de structure d'une page HTML. Chaque élément est représenté par un bloc.

Dans la Figure 3, l'élément `<h1>` est contenu directement dans `<body>` : on dit que `<body>` est son **élément parent**. En revanche, `<h1>` et `<p>` sont côte à côte. Expliquons quelques autres balises :

- `<!DOCTYPE html>` est placé au début pour indiquer qu'il s'agit d'un document HTML.
- L'élément `<html>` indique les limites de la page. Tous ce qui est en dehors est ignoré (excepté le doctype).
- L'élément `<head>` contient des 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 (voir plus bas), etc.
- L'élément `<body>` contient tout ce qui s'affiche vraiment dans le navigateur.

4.4 Comportement par défaut des éléments

Les balises HTML sont utilisés pour décrire la structure de la page. Ainsi

`<h1> - <h6>` sont les différents niveaux de titre ;

`<p>` indique un paragraphe ;

`<strong>` indique un élément mis en valeur, etc.

Certains éléments modifient l'apparence de leur contenu[6]. Ainsi, deux titres `<h1>` et `<h2>` seront affichés en gras et 100% / 50% plus larges que la taille de police de base, respectivement. Cependant, vous ne devez pas choisir la balise à appliquer en fonction de l'apparence par défaut : les balises HTML sont une indication de structure, la présentation sera gérée par le CSS.

Voici donc la manière correcte de procéder :

Structure Ce titre est le titre principal de la page, je le met donc entre des balises `<h1>` `</h1>`.

Présentation Je veux que le titre principal ait une certaine apparence, je fais cela en CSS.

Certains éléments n'ont pas d'impact initial sur l'affichage, comme `<section>` qui indique une partie de la page (imaginez une séparation partie 1 / partie 2). Bien structurer la page apporte des bénéfices importants en lisibilité, accessibilité (personnes malvoyantes), et en référencement par moteurs de recherche. De plus, on pourra toujours lui appliquer une apparence en CSS.

`<span>` et `<div>` sont des éléments "neutres", c'est-à-dire, qui n'ont pas de rôle particulier. On utilise ces balises pour appliquer des changements d'apparence à des morceaux qui ne correspondent pas à un élément préexistant. On verra la différence entre ces deux balises en Section 6.1.

Il existe une au total centaine d'éléments différents. Savoir quel élément correspond à ce que vous voulez faire vient avec l'expérience, les recherches sur Internet et les conseils de vos enseignants. Quelques références sont proposées en section 10.

4.5 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.

Prenons l'exemple d'un lien hypertexte `<a>` qui utilise l'attribut obligatoire `href` pour indiquer la cible. L'élément s'écrira de cette manière :

```
Le cours est sur <a href="http://cours.iut-orsay.fr">Moodle</a>.
```

Le cours est sur Moodle.

Je recommande de s'habituer à toujours mettre des apostrophes (simples ou doubles) autour des valeurs pour éviter des problèmes d'espace. Une bonne documentation indiquera pour chaque élément les attributs fréquemment utilisés.

Comme pour les éléments, savoir quel attribut il faut indiquer et quel attribut correspond à ce que vous voulez faire vient avec l'expérience et en consultant des références.

6. Techniquement, cela dépend du navigateur, mais les différences sont minimes.

Deux attributs importants, `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`) On comprendra mieux leur usage dans la section CSS plus bas.

`id` est également utilisé pour créer une cible d'URL. Par exemple, l'URL suivante :

`https://fr.wikipedia.org/wiki/Université#France`

redirige vers l'élément dont l'`id` vaut "France" dans la page demandée.

5 Bases du CSS

5.1 Introduction

Le code CSS se présente sous la forme d'une suite de blocs qui ont l'apparence suivante :

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

`p` est le **sélecteur** : il indique que ce qui suit s'applique aux paragraphes (éléments `p`).
`color` est la **propriété** : elle indique que nous voulons changer la couleur du texte.
`red` est la **valeur** donnée à la propriété. En l'occurence, le texte est mis en rouge.

Pour que le code CSS ait un effet, il faut l'inclure dans le document HTML concerné. La manière normale est d'écrire son code CSS dans une **feuille de style CSS** (extension `.css`), puis de mettre un lien vers cette feuille de style dans le document. On place pour cela un élément `<link />` dans le `<head>` :

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

où les étoiles indiquent le chemin vers la feuille CSS.

5.2 Sélecteurs

Comme son nom l'indique, 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** `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>`.

Les sélecteurs peuvent se combiner : `nom1, nom2` sélectionne les élements `<nom1>` **et** les éléments `<nom2>`.`nom1.nom2` sélectionne tous les élements `<nom1>` de classe `nom2`.

Notez qu'on peut sélectionner un élément individuel (par classe ou identifiant), mais la syntaxe facilite le fait d'appliquer les mêmes modifications à tous les éléments d'un certain type.

⚠ Il existe d'autres sélecteurs plus avancés, que vous pouvez trouver dans votre documentation préférée.

5.3 Propriétés et valeurs

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;
}
```

Il existe plus de 500 propriétés CSS, mais la grande majorité sont très spécifiques et ne s'appliquent qu'à quelques éléments. Comme pour les balises HTML, ce n'est pas utile de les apprendre toutes par coeur.

6 Flot HTML et Positionnement CSS

Historiquement, on pensait une page HTML comme à un document textuel. 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 sur flot pour faire des pages beaucoup plus complexes.

6.1 Comportements block et inline, propriété `display`

On peut séparer les éléments HTML en deux catégories suivant la manière dont ils se disposent sur la page.

Le mode **inline** correspond à de simples modifications du texte comme la mise en gras. Tous les éléments **inline** se comportent comme du texte et ont exactement la largeur de leur contenu.

Le mode **block** correspond à des objets comme des images, des menus, des titres... Les blocs ont ensuite beaucoup d'options pour décider où se placer. Par défaut, ils sont placés seuls sur leur ligne (ils prennent toute la place disponible horizontalement).

`<a>` (lien hypertexte), `<strong>` (texte mis en valeur), `<code>` (fragment de code), `<span>` (élément neutre) sont des éléments **inline**. `<h1>` - `<h6>` (titres), `<p>` (paragraphe), `<img/>` (image), `<div>` (bloc neutre) sont des éléments **block**.

Pour changer le mode d'un élément, on utilisera la propriété CSS `display`. `display: inline;` et `display: block;` donnent à l'élément le comportement correspondant, et `display:none;` fait simplement disparaître l'élément. Pour pouvoir mettre des éléments côte à côte (comportement inline) mais pouvoir contrôler leur hauteur et leur largeur (comportement block), on utilisera la valeur `display: inline-block;`. Elle est utile, par exemple, pour faire une barre de menu avec plusieurs boutons côte à côté.

6.2 Positionnement CSS

Le positionnement des blocs par défaut ne suffit pas à faire tous les comportements dont on a besoin. Le positionnement CSS consiste à utiliser certaines propriétés CSS pour changer ce comportement et que le block sorte de son emplacement attendu.

Deux propriétés CSS très utiles pour mettre de l'espace dans votre page sont les propriétés `margin` et `padding`. Toutes les deux servent à ajouter de l'espace entre les éléments, mais `margin` en ajoute autour du block alors que `padding` l'ajoute dans le bloc. On peut ainsi utiliser `margin: 1cm` ou `margin-left: 1cm`.

La 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. C'est très utilisé pour un menu de navigation.

La propriété `position` permet de placer des éléments de manière mieux contrôlée. On la combine avec les propriétés `left, right, top` et `bottom` (abbrévié `lrtb` ci-dessous). 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 `lrtb`.

`absolute` place l'élément à une position fixée sur la page (donnée par `lrtb`).

`fixed` place l'élément à une position fixée sur l'écran (donnée par `lrtb`) et reste immobile même quand la page défile.

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

Quand vous modifiez la propriété `position`, vous **devez** préciser là où vous placez le bloc avec `lrtb`.

En pratique, `relative` est utilisé rarement et pour des déplacements minimes. `absolute` est utilisé quand on veut que plusieurs éléments se chevauchent. `fixed` et `sticky` sont souvent utilisés quand on veut qu'un menu reste accessible quand la page défile.

Astuce étrange : pour les propriétés `absolute`, `fixed` et `sticky`, le positionnement (avec `ltrb`) se fait par rapport à la page entière. Parfois, vous voulez positionner à l'intérieur d'un autre bloc. Pour faire cela, il faut donner une propriété `position` au bloc parent (magie noire). En général, on met `position:relative` tout seul pour ne pas déplacer le bloc.

6.3 Progresser en CSS

Tout ça est assez compliqué et la page ne s'affiche peut-être pas comme vous le souhaitez. Dans ce cas, il faut apprendre à utiliser les outils de développeurs pour comprendre le problème. Faites `Clic droit` → `Inspecter` ou `Examiner l'élément` dans votre page Web.

L'aspect le plus utile est le fait de cliquer sur les éléments dans le code HTML et de regarder sa position sur la page. L'élément est en bleu, ses marges sont en jaune, et le padding est en violet. En bas à droite, on peut vérifier que le CSS qui s'applique à

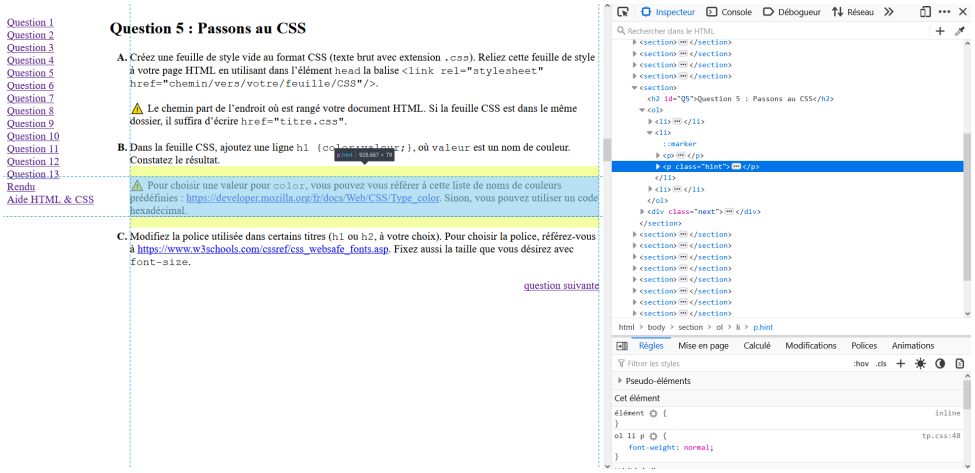


FIGURE 4 – A gauche : votre page. En haut à droite : le code HTML. En bas à droite : le CSS qui s'applique à l'élément choisi (il y a d'autres options)

l'élément est bien le bon. On peut ainsi facilement voir si la position des éléments et trouver nos erreurs.

Il est ainsi possible de modifier le HTML ou le CSS directement dans le navigateur. Cela permet de faire des tests facilement, mais les fichiers HTML et CSS ne seront pas modifiés.

7 Conseils et bonnes pratiques

7.1 Dossiers et chemins

Il est recommandé de créer un dossier spécifique pour chaque site Web. En effet, chaque document HTML vient avec des ressources liées (images, feuille de style CSS, etc.). Pour vos rendus de TP ou pour mettre votre site en ligne sur un serveur, il est beaucoup plus facile de transférer un dossier entier (ou une archive) sans avoir des fichiers externes qui traînent.

Vous mettrez souvent dans vos pages HTML un lien vers une autre ressource locale. Supposons que vous avez un fichier `exemple.html` dans le dossier `siteweb` et que vous voulez mettre un lien vers la page `cible.html` dans le même dossier. En ouvrant cette dernière page dans votre navigateur, la barre d'adresse indique quelque chose de ce genre :

`file:///C:/Users/bhellou/Documents/CODIN/siteweb/cible.html`

Il s'agit du **chemin absolu** du fichier `cible.html`. Notez le protocole `file://` indiquant qu'il s'agit d'un fichier local. Son **chemin relatif** depuis `exemple.html` est simplement `cible.html`, puisqu'ils sont dans le même dossier.

Dans vos fichiers HTML, n'utilisez que des chemins relatifs. La raison est que tout déplacement du dossier `siteweb` modifie les chemins absolus en conséquence.

Les liens utilisant des chemins absolus seront donc inutilisables dès que vous transmettrez votre document à quelqu'un d'autre, par exemple en le mettant sur un serveur Web. Vous pouvez facilement tester vos sites en déplaçant le dossier et vérifier si les liens et les images fonctionnent.

7.2 Lisibilité du code

Un code HTML + CSS n'est pas lu que par des navigateurs : d'autres informaticiens (des collègues, des enseignants, vous dans le futur) auront besoin de comprendre et modifier votre code. C'est pour cela qu'il faut prendre des bonnes habitudes afin de rendre le code facilement lisible et modifiable. Voici quelques conseils généraux et des exemples avec mon style préféré.

Quelques conseils généraux :

- Ne laissez pas traîner de code inutile. En particulier, si vous utilisez un bloc ou une propriété pour tester quelque chose, n'oubliez pas de l'enlever après. Cela mène à des bugs futurs particulièrement pénibles.
- Soyez cohérents.

Pour le HTML :

- Au minimum, sautez des lignes pour séparer les différentes parties de la page, et passez à la ligne entre les blocs importants. Les sauts de ligne ne changent rien au rendu du navigateur. On peut choisir de laisser sur la même ligne des petits blocs qui sont côte à côte sur la page.
- Utilisez des identations pour indiquer quels blocs sont à l'intérieur d'un autre.
- Ne faites pas de lignes trop longues. Certains styles de code recommandent de ne pas dépasser 80 caractères. En tout cas, il faut que la ligne apparaisse sur votre écran sans devoir défiler sur le côté.
- Utilisez des identifiants descriptifs (pour les attributs `id` et `class`). En voyant la page et le code CSS, on doit pouvoir deviner à quel(s) élément(s) correspond l'identifiant.

```
<article> <h2> Titre
 de l'article </h2> <p>

Lorem ipsum dolor sit amet,...
  </p>
    </article>
```

Code HTML peu lisible

```
<article>
  <h2> Titre de l'article </h2>

  <p>
    Lorem ipsum dolor sit amet,
    consectetur adipiscing elit.
  </p>
</article>
```

Code HTML lisible

Pour le CSS :

- N'hésitez pas à faire plusieurs fichiers CSS (un pour chaque page, un pour les éléments communs comme le menu).
- Rangez votre code CSS pour suivre la structure de la page. Rassemblez ensemble tout ce qui s'applique aux mêmes parties : par exemple, mettez au début ce qui concerne l'en-tête, etc. Ajoutez des commentaires (`\*... *\` en CSS) pour indiquer et séparer les parties.
- Évitez les répétititions. Si la même propriété est appliquée plusieurs fois, il est souvent possible d'éviter la répétition en utilisant un sélécteur plus approprié.
- Ajoutez une identation quand vous entrez à l'intérieur d'un bloc.

```
@media (min-width:640px){
.x{float: left;
 padding-right: 10px; float:left;
}
#y
{float: left;
  margin-right: 10px;
  }}
```

Code CSS peu lisible

```
/* Logos et portraits */

@media (min-width:640px){
  img#logo, img.portrait{
    float: left;
    margin-right: 10px;
  }
}
```

Code CSS lisible

7.3 Standards 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.

Il peut être difficile de comprendre à quel point ces standards ont été importants pour que le Web soit ouvert et uilisable par tous. On peut imaginer un monde alternatifs où existent quatre ou cinq formats différents de pages Web, dont certains formats fermés et lisibles seulement par des certains logiciels ou certains OS [7].

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.

Cependant, le concepteur doit se méfier : même si une page HTML s'affiche bien sur un navigateur, elle peut tout de même contenir des erreurs pas directement visibles. La bonne utilisation du HTML+CSS donne de grands bénéfices d'accessibilité, de navigation, et même de référencement sans effort important. Ces erreurs « invisibles » sont la raison principale de respecter les standards.

7. Une situation qu'on a vécu avec des technologies comme Flash qui venaient en plus du HTML.

7.4 Faire valider son code

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/`

⚠ Il existe de nombreux autres validateurs, mais soyez vigilants. Les standards changent (nous en sommes au HTML5), et de nombreux validateurs non officiels ne sont pas à jour.

Le validateur vous indiquera une série d'erreurs et d'avertissements, indiquant la ligne et la colonne concernée, et le problème. Un exemple est donné Figure 5.

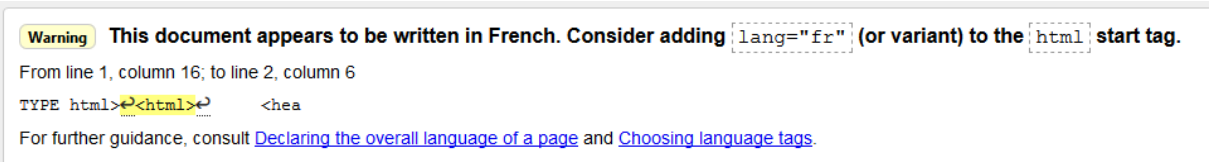


FIGURE 5 – Un avertissement du validateur HTML.

Si vous ne parvenez pas à comprendre une erreur, n'hésitez pas à vous référer à une documentation (voir section 10) sur l'élément incriminé, ou à demander à votre enseignant. Pour les TP et le projet, toute erreur ou avertissement donné par le validateur sera considéré comme une erreur dans votre code, sauf si votre enseignant vous indique le contraire.

8 Conception adaptative (ou *Responsive Web Design*)

8.1 Généralités

Aux origines du Web, la plupart des terminaux utilisés pour se connecter avaient des tailles d'écran proches. Ainsi, il suffisait de concevoir et tester une page Web sur un seul écran pour s'assurer qu'il s'affichait correctement pour la très grande majorité des visiteurs. Plus tard, de nombreuses pages Web bien adaptées à un écran d'ordinateur se sont révélées difficile d'utilisation avec un un téléphone, une tablette, une montre connectée, 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.

La première approche a pour défaut de dupliquer beaucoup de travail et d'outils pour mettre à jour les différentes pages / applications ; la seconde demande des outils spécifiques et un travail plus important de conception. C'est cette seconde approche qu'on appelle « Conception adaptative de sites Web » ou « *Responsive Web Design* ».

Voici les idées majeures qui sous-tendent la conception adaptative :

1. La position des blocs principaux de la page (menus, etc.) change avec la taille de l'écran : verticalement pour les petits écrans, horizontalement pour les grands écrans. Voir une illustration Figure 6.
2. Pour tous les objets de taille significative (images, blocs), leur taille évolue

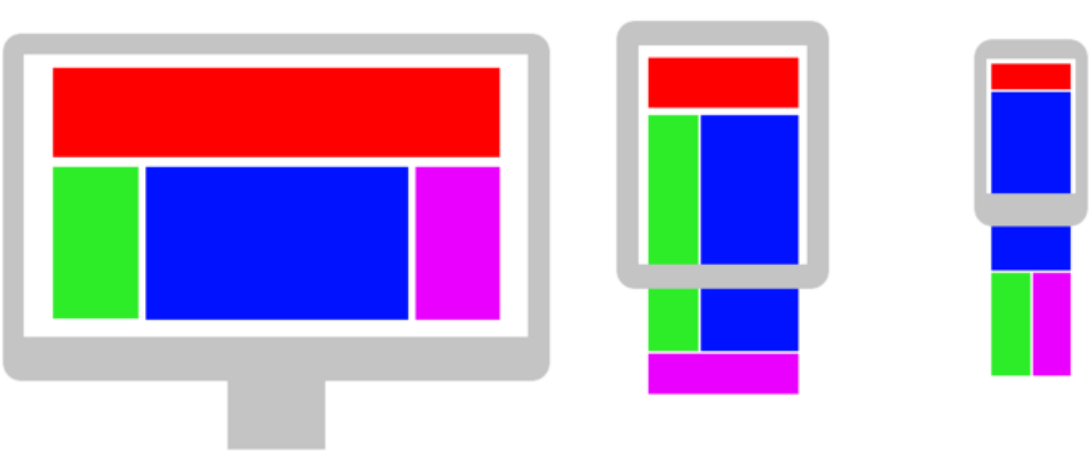


FIGURE 6 – En conception adaptative de sites Web, les blocs principaux changent de position en fonction de la taille de l'écran. (Tomáš Procházka, domaine public)

Il est important de se rappeler que le but est que la page soit lisible et utilisable par tous, mais aussi d'utiliser au mieux l'espace disponible. Si votre page est une colonne de 320px de largeur sur tous les écrans, ce n'est pas satisfaisant pour les écrans d'ordinateur de bureau.

8.2 Unités relatives

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

em 1em correspond à la taille de police de base. C'est utile pour avoir des titres proportionnellement plus grands que le texte normal, même quand la taille de base change.

% Le pourcentage a un sens différent suivant la propriété concernée.

vh (ou *viewport height*) correspond à la hauteur de l'écran.

Quelques exemples d'utilisation :

`width: 50%;` la largeur du bloc est la moitié de celle du bloc parent. Si le bloc parent est `body`, il s'agit de la moitié de la largeur de la page.

`height: 50%;` la hauteur du bloc est la moitié de celle du bloc parent. Attention, `body` n'a pas de hauteur ; le seul moyen de faire un bloc aussi grand que l'écran est d'utiliser `vh`.

`font-size: 120%;` ou `font-size: 1.2em;` correspond à une taille de police 20% plus grande que la taille de police de base. C'est utile pour définir la taille d'un titre en sachant que certains appareils imposent des tailles de police de base différentes.

En combinaison avec les pourcentages, on peut utiliser les propriétés `min-width` et `max-width` (idem avec `height`) pour empêcher un bloc de devenir trop petit ou trop grand.

8.3 Requêtes media CSS (ou *CSS media queries*)

Une requête media permet d'appliquer une partie du CSS uniquement pour certaines appareils.

```
@media(condition){bloc CSS}
```

Les conditions que nous utiliserons concernent la largeur de l'écran. Par exemple, un bloc de CSS entouré de `@media (min-width: 640px and max-width: 1280px)` ne s'appliquera que quand la zone d'affichage du navigateur a une largeur comprise entre ces deux valeurs.

Un usage fréquent consiste à modifier le positionnement CSS de certains blocs (`display`, `float`) pour changer la structure de la page, comme on l'avait illustré Figure 6. Il existe différentes écoles pour choisir les valeurs limites (ou **points de rupture**), que nous verrons dans la section suivante.

Si vous testez votre site sur mobile, il se peut que le résultat soit inattendu car les navigateurs mobiles peuvent "mentir" sur leur taille d'écran. Rajouter la ligne suivante dans le `head` de votre HTML résoudra ce problème :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

8.4 Concevoir et tester votre site

Si on ne veut pas acheter une dizaine d'écrans de différentes tailles, le plus simple est de redimensionner la fenêtre de son navigateur pour les tests rapides. Pour faire des tests plus propres et avoir des points ruptures précis, la plupart des navigateurs proposent un outil de **vue adaptative**.

Firefox `Ctrl+Maj+M` ou `Menu` → `Développement Web` → `Vue adaptative`.

Chrome `F12` ou `Menu` → `Plus d'outils` → `Outils de développement` → 

La vue adaptative vous permet de tester des largeurs précises, de reproduire le comportement d'une variété d'appareils (y compris pour tester si votre site est facilement utilisable tactilement), et même de voir s'il est accessible avec une bande passante réduite.

Pour déterminer les points de rupture, voici une approche possible :

1. Commencez à `320px`, modifiez le CSS pour que la page soit utilisable ;
2. Augmentez la largeur de la page en vous assurant que les blocs se redimensionnent bien (taille en pourcentages) ;
3. Au point où votre page devient moins utilisable, notez la largeur et ajoutez des requêtes media pour repositionner les blocs concernés ;
4. Continuez ainsi jusqu'à la largeur de votre écran (idéalement, $\sim$ 2000 pixels).

⚠ Pour les changements futurs, il est plus facile d'avoir quelques points de rupture importants que de nombreux points de rupture avec des changement mineurs.

9 Images, vidéos, ressources externes

9.1 Licences et sources

On lit parfois que « La compétence la plus importante d'un informaticien est de savoir faire des recherches sur Internet ». Il est certain que dans vos projets à l'IUT et dans votre vie future, vous utiliserez de nombreuses ressources externes : images, morceaux de code, vidéos, bibliothèques, etc.

Tout ce qui suit ne concerne pas la situation où vous faites un lien (hypertexte) vers une ressource présente sur un autre site.

Point de vue légal. Si votre projet va être diffusé, vous devez disposer d'une autorisation de l'auteur pour toutes les ressources externes que vous utilisez. Souvent, cette autorisation prend la forme d'une licence. La licence peut ajouter des contraintes supplémentaires : par exemple, interdire l'utilisation dans un projet commercial ou interdire toute modification. En l'absence de toute indication, il est supposé que vous n'avez pas le droit d'utiliser la ressource.

Supposons que vous voulez réutiliser un morceau de code de Stack Overflow, un site de questions-réponses de programmation. Une recherche dans l'aide vous amène à la page suivante :

`https://stackoverflow.com/help/licensing`

qui indique que vous pouvez réutiliser, modifier et rediffuser le morceau de code, du moment que votre projet est mis sous la même licence et que vous indiquez sa source.

Point de vue académique. Quoi que dise la licence, nous vous demandons d'indiquer ce que vous n'avez pas produit vous-même, et son origine. Il y a plusieurs manières

raisonnables de le faire : commentaire avant le morceau de code, petit texte sous l'image, section "Sources" dans le rapport, etc. Quelques précisions :

- Les moteurs de recherche (type Google Images) ne sont pas des sources. Vous devez trouver le site d'origine et, idéalement, trouver le nom d'auteur.
- Si vous avez adapté un morceau de code à votre usage, vous pouvez indiquer « adapté de (source) » ou « inspiré de (source) ».

9.2 Taille

Si vous ne faites pas attention aux images que vous utilisez, il est fréquent qu'un document ou une page Web dépasse plusieurs mégaoctets (Mo), dont une grande partie dûe aux images. Si vos images n'ont pas besoin d'une qualité photographique, cela représente un gâchis de ressources et pose plusieurs problèmes :

- limites aux tailles de pièces jointes dans les mails
- pages Web lentes à charger
- consommation de mémoire sur les serveurs ou les dépôts (comme Moodle), etc.

Le même problème peut arriver avec les vidéos, sauf si vous intégrez vos vidéos depuis un hébergeur qui s'en occupera lui-même. Quelques recommandations pour les images :

1. Réduisez la taille de l'image pour qu'elle soit proche de celle qu'elle aura sur un écran d'ordinateur. Si vous voulez une icône de 30×30 pixels, inutile de l'enregistrer en 640×640.
2. Pour les photos qui apparaissent en grand format (l'image de fond, par exemple), utilisez un format compressé comme le JPEG (`.jpg` ou `.jpeg`). Ce format permet de choisir un niveau de qualité (1-100). Vous pouvez faire des essais avec vos images mais un niveau de 75 est souvent un compromis acceptable.

Il existe de nombreux logiciels pour effectuer ces opérations. Je recommande GIMP (logiciel libre, tous systèmes) disponible à `https://www.gimp.org/`. La quantité d'options peut effrayer. Pour faire les opérations ci-dessus :

1. Après avoir ouvert l'image, faire `Image` → `Échelle et taille de l'image`, faites vos modifications puis `Mise à l'échelle` pour valider.
2. Faites `Exporter sous` et choisissez le format `.jpeg` (ou faites directement `Écraser *****.jpeg`). Vous pouvez choisir la qualité dans le menu suivant.

Attention, `Enregistrer (sous)` utilise un format spécifique à GIMP. Il faut toujours utiliser `Exporter` (ou `Écraser`) pour sauvegarder l'image dans un format habituel comme `.jpeg` ou `.png`.

10 Références

Documentation LibreOffice (`https://fr.libreoffice.org/get-help/documentation/`) : Wiki avec de nombreux outils expliqués et chapitres de livres expliquant LibreOffice de A à Z.

HTML & CSS

Mozilla Developer Network (`https://developer.mozilla.org`) : Tutoriel et documentation détaillés. Information très complète : rôle, usage et toutes les attributs disponibles sur chaque élément.

W3Shools (`https://www.w3schools.com/`) : Explications simples des éléments HTML et propriétés CSS de base avec exemple.

OpenClassRooms (`https://openclassrooms.com/fr`) : Tutoriels structurés pour différentes technologies (par exemple (« apprenez-a-creer-votre-site-web-avec-html5-et-css3 »)

StackOverflow (`https://stackoverflow.com`) : Site de questions-réponses de bonne qualité