**Synthèse Développement Web – Théorie**

**Introduction**

**Deep web** : tout site web qui nécessite une autentification

**Dark web** : nécessite un navigateur spécifique comme tor

**L'ergonomie** d'un site web est l'optimisation des interfaces web et leur adaptation à leur public

**Le référencement web** : positionnement des sites web dans les résultats des moteurs de recherche

**Le développement web** nécessite la connaissance de nombreux langages et technologies comme les langages HTML, CSS, JavaScript, PHP, XML, les bases de données, les navigateurs, les normes d'accessibilité WCAG, l'ergonomie web, le référencement, le responsive design, les solutions d'hébergement, etc.

**Technologies web**

**Front end (coté client)**

Ce que le client peut voir en faisant ctrl + U

* HTML (HyperText Markup Language) : utiliser pour la structure des sites web, il a beaucoup évolué depuis sa création et est passé par plusieurs versions différentes. L'HTML5 est la version actuel et restera la dernière version.

* XHTML : évolution de l'HTML4 adapté aux normes XML. Il subsiste encore de nombreux sites codés en XHTML actuellement

* CSS (Cascading Style Sheet) : utiliser pour la mise en page des web. Actuellement à la version 3. Les sites comportant uniquement les langages CSS et HTML sont dit "statiques". Les pages web dynamiques sont générées par un langage dynamique coté serveur et dont le contenu est variable

* W3C (World Wide Web Consortium) + WCAG (accessibilité) : La promotion de la compatibilité des technologies web HTML5, HTML, XHTML, XML, XSL, CSS, PNG et SVG

* JavaScript : calcul qui effectuerais des variables. Langage de programmation objet pour le web. Avec le PHP, il s’agit des deux langages de programmation dynamiques pour le web les plus répandus. Contrairement au PHP, le JavaScript est interprété côté client, par le navigateur. Le JavaScript est également un langage évènementiel : il permet de détecter les actions des visiteurs (le clic, le survol, le focus, le scroll, etc.) et d’y attribuer des actions.
  + Ajax : allie JavaScript et XML
  + JSON (JavaScript Object Notation) : structure les informations
  + Frameworks JS : jQuery, Angularr, React, Vue.js

**Back end (coté serveur)**

**La partie que le client ne voit pas et ne peut pas voire**

* PHP (Personal Home Page Hypertext Preprocessor) : langage dynamique qui a la particularité d’être interprété côté serveur ce qui garantit la sécurité du code

* XML : langage balisé pour stocker des données de manière semi-structurée

**Parmi les autres langages côté serveur, on retrouve Java, ColdFusion, Node.js, Perl, Ruby et Python**

**Système de gestion du contenu**

* CMS (Content Management System) : permet de créer des sites destinés à gérer de grandes quantités de données

**Les navigateurs web**

World Wide Web --> Mosaic --> Netscape --> Microsoft Internet Explorer --> Mozilla --> (fin de netscape) --> Firefox --> Safari --> UC browser --> Chrome --> Edge

Une image contenant texte, diagramme, ligne, Tracé

Description générée automatiquement

* Navigateurs : chrome, safari, firefox, edge, …
* Graphisme web
* Ergonomie UI / UX : UI = UX = questionnement globale pour arriver a une fin qui est souvent un achat

SEO : Référencement, moteurs de recherche

**Les langages web**

**HTML**

* HTML = HyperText Markup Language
* Langage de description, interprété et banalisé
* Pour coder **la structure, le squelette** des sites web
* HTML 5 = version définitive

**CSS**

* CSS = Cascading Style Sheet
* Langage interprété
* Pour coder la présentation des sites web
* Version actuelle : CSS3 ==> version définitive et non terminer

**PHP**

* PHP = (Personal Home Page) HyperText Preprocessor
* Crée en 1994 par Rasmus Lerford
* Langage dynamique interprété côté serveur
* Pour les **pré-traitements**
* Version 8

==> le PHP n'arrive jamais chez le client

**Chapitre 1 : HTML**

**1.1 Introduction au HTML**

**Rôle du langage HTML**

**HTML est le langage universel de structure des informations sur le Web.**

Au HTML, viennent se greffer des instructions CSS définissant la mise en page du site

C’est en HTML que vous indiquez la présence d’un titre « courses » et d’une liste d’aliments « pommes, carottes, patates, etc. », mais c’est en CSS que vous définissez la couleur et la police du texte.

L'HTML contient quelques fonctions de mise en page car au début du Web le CSS n'existait pas encore mais il est préférable de séparer la structure de la présentation.

**Non maitrise de l'affichage**

**L'affichage des sites varie en fonction du périphérique, du système d'exploitation, des polices disponibles, du navigateur et de sa version et de la définition de l'écran ==> privilégié l'accessibilité des contenus**

**Programmes nécessaires**

Pour développer une page HTML il y a deux solutions :

* La solution assistée : Les éditeurs WYSIWYG (What You See Is What You Get), ou en français les éditeurs « tel-tel » (tel écran, tel écrit)
* La solution des professionnels : les éditeurs de code tels que Notepad++, Sublime Text, VSC, …

**Visualisation de votre page HTML**

Il faut tester chaque site sur plusieurs navigateurs différents pour s'assurer de la compatibilité avec chaque navigateur

Commandes importantes :

* Ctrl + R ou F5 : actualiser l'onglet actif.
* Ctrl + F5 : actualiser l'onglet actif et forcer le rafraichissement du cache.
* Ctrl + + : zoomer la page.
* Ctrl + - : dézoomer la page.
* Ctrl + 0 : revenir au niveau de zoom par défaut.
* Ctrl + U : voir le code source de la page.
* F11 : plein écran.
* F12 : ouvrir les outils de développement.

**1.2 Les bases du HTML**

**Les éléments**

Langage balisé car ses instructions s’écrivent sous forme de balises.

Le code HTML contient les textes de la page web, accompagnés d’éléments HTML. Le plus souvent, ces éléments sont composés de 2 balises :



Balise ouvrante Balise fermante

Les balises HTML sont des instructions réservées au navigateur. Elles se différencient du texte par les caractères < et > appelés chevrons.

Chaque élément HTML a une signification : une image <img>, un titre <h1> ou <h2>, un paragraphe <p>, un pied de page <footer>, une citation <cite>, etc.

le choix d’un élément se fait toujours en fonction de sa sémantique et non pas de son apparence. La sémantique est le sens apporté par un élément. L’élément , par exemple, indique que son contenu est une citation et provoque la mise en italique de ce contenu

L’écriture minuscule est de mise en HTML, bien qu’elle ne soit pas obligatoire en HTML5.

**Classification du langage HTML**

HTML est également un langage interprété

Bien qu’HTML soit un langage informatique, on ne peut pas le qualifier de langage de programmation car son rôle se limite à la représentation de données et non pas à l’écriture d’algorithmes

**Les attributs**

Les attributs permettent de personnaliser les éléments en en modifiant certains critères.

attribut 
<body 
balise ouvrante 
élément HTML 
balise fermante 

body est le nom de l’élément et bgcolor qui est un attribut de cet élément

Autre exemple avec une balise auto-fermante

a tt ri buts 
<img bison dans la prairie"' 
balise auto-fermante 
élément HTML 

* Chaque élément HTML possède ses propres attributs.
* Chaque attribut possède une valeur par défaut, par exemple "white" pour bgcolor qui permet de spécifier la couleur de fond. Il est donc superflu de coder bgcolor="white".
* Les attributs se codent toujours dans la balise ouvrante, jamais dans la balise fermante.
* Il peut y avoir plusieurs attributs codés dans une même balise et ce, dans n’importe quel ordre.
* Un même attribut ne peut être codé qu’une seule fois par élément.
* La valeur de l’attribut doit toujours être entre guillemets.
* Certains attributs sont obligatoires car l’élément n’aurait pas de sens sans (par exemple une image sans sa source src) ou car imposé par le W3C (par exemple une image sans texte alternatif alt).

**Les balises auto-fermantes**

<br> : retour à la ligne

<wbr> : permission de retour à la ligne si nécessaire, par exemple dans un long mot

<hr> : barre de séparation horizontale

<img> : image

<input> : champ de formulaire

<meta> : informations pour le navigateur et les moteurs de recherche

<link> : inclusion de fichier externe

Attributs de <hr> : size (épaisseur), width (largeur), align (alignement)

**Les commentaires**

Les développeurs utilisent les commentaires pour annuler des parties de code sans les effacer



**La page HTML**

Toute page HTML5 contient au minimum ces éléments :

< ! DOCTYPE html> e—version HTML 
<html 
<meta charset="utf-8"> 
< body > 
Bonjour ! 
< / html > 
format d•encodage 

Le résultat affiché ici par votre navigateur sera une page blanche avec le texte « Hello World ! » et ayant pour titre de document « Code Minimal ».

Tout document HTML5 commence par la balise qui précise aux navigateurs que la version HTML choisie est HTML5.

Nous retrouvons ensuite 3 éléments fondamentaux :

* <html>, l’élément racine de la page, qui est unique et qui contient tout le reste du code ;
* <head> qui est unique et qui contient toutes les informations non visibles ;
* <body> qui est unique et qui contient toutes les informations visibles

On retrouve aussi <title> qui contient le titre du document, visible dans l’onglet.

Et enfin <meta charset="utf-8"> qui précise le format d'encodage.

Ne jamais croiser les flux ! <b><i>…</i></b> est correct alors que <b><i>…</b></i> est erroné et risque de poser des problèmes.

**Traitement des espaces par le navigateur**

Le navigateur ne tient compte que d'un seul espace entre les mots. Ainsi :

s'affiche de façon identique à. 

Le navigateur ne tient pas compte des retours à la ligne dans le code. Pour forcer un retour à la ligne dans la page web, il faut utiliser la baliser <br>. Ainsi :

texte</strong» 
s'affiche de façon identique à : 
< st ronpmon 

**Le flux**

L'ordre dans lequel sont codés les éléments HTML correspond à l'ordre dans lequel ils s’affichent. Cet ordre est appelé le flux et suit grosso modo le sens de lecture d’un livre.

**L'arborescence**

Les éléments <body>< et <head> sont les enfants de l’élément <html>. <html> est le parent de <head> et <body>. L’élément <html> est appelé l’**élément racine** car il est l’ancêtre de tous les autres éléments du document

head> 
< html> 
< head> 
< 'body > 

Racine : <html> est appelé l’élément racine car il contient tous les autres.

Ancêtre : élément qui se trouve sur le chemin entre un élément et l’élément racine. Dans l’exemple : <body> et <html> sont les ancêtres de title.

Parent : ancêtre direct d’un élément. Dans l’exemple : <head> est le parent de <title>

Enfant : inverse d'un parent. Descendant direct d'un élément. Dans l'exemple : <h1> et <p> sont des enfants de <body>

Descendant : inverse de l'ancêtre. Dans l'exemple : <h1>, <p>, <em> et <a> sont les descendants de <body>

Frère : deux enfants ayant le même parent sont des frères. Dans l'exemple : <h1> et <p> sont frère

**1.3 les formats d'encodage**

Pour enregistrer un texte ou une suite de caractères en mémoire, il faut passer par une table de correspondance entre caractères et valeurs numériques. Ces tables sont des **formats d’encodage.**

L’ASCII est la première véritable norme de codage de caractères. L’ASCII est une table de correspondance dans laquelle chaque caractère correspond à une valeur numérique : le caractère « A » vaut 65 ou 1000001 en binaire, valeur qui est enregistrée en mémoire pour représenter la lettre « A ».

Les fichiers HTML ne dérogent pas aux formats d’encodage : leur lecture et leur écriture se fait selon un format d’encodage à définir :

* Lors de l’édition du code HTML, le format d’encodage est paramétrable dans l’éditeur de code.
* Lors de l’interprétation de la page, le navigateur se fie à la méta-balise charset.

**De l'ASCII à l'UTF-8**

ASCII : American Standard Code for Information Interchange (Code standard américain pour l’échange d’informations).

L’ASCII est codé sur 7 bits (128 caractères), complétés par un huitième bit, redondant, pour effectuer un contrôle de parité : grâce à ce système de codage, tout caractère comportant un nombre impair de 1 est détecté et une erreur est signalée.

ASCII étendu ANSI : Comme le code ASCII, mis au point pour la langue anglaise, ne contient pas de caractères accentués, plusieurs extensions de l’ASCII ont vu le jour mais il existe un format d’encodage plus universel : l'UNICODE.

Unicode : L’Unicode (1991) est un système de codage des caractères sur plusieurs octets. Ce format d’encodage regroupe la quasi-totalité des alphabets de la planète mais aussi le braille, le phonétique, les symboles monétaires, des symboles techniques divers, des formes géométriques, des flèches, les pièces du jeu d’échec, des petits bonhommes, les signes du zodiaque, des symboles religieux et astrologiques, etc.

Même si l'UNICODE est très complet, il reste peu utilisé par rapport à l'ASCII à cause de son problème de surpoids : en Unicode chaque caractère prend 2, 3, 4 octets ou plus

UTF-8 : L’UTF-8 est la solution au problème de poids de l’UNICODE et aux limites de l’ASCII. En UTF-8, les caractères ASCII sont codés sur 1 octet ; les caractères plus rares signalés par le 8e bit de l’ASCII (celui qui servait au contrôle de parité) codés en UNICODE (2 octets ou +). Par exemple, pour le texte "Bienvenue chez Sébastien !", seul le "é" ne fait pas partie du code ASCII et est codé sur 2 octets.

**Les entités de caractères**

Les premières versions du langage HTML travaillaient en ASCII (comme le langage C) et nous privaient donc de certains caractères spéciaux ou accentués comme le « é » pour lesquels il fallait passer par des codes particuliers appelés entités de caractères

Exemples :

**&nbsp;** ou son code numérique **&#160;** représente un espace insécable.

**&euro;** ou **&#128;** représente le caractère €.

**&amp;** ou **&#38;** représente le caractère &

**1.4 Les familles d'éléments**

Les éléments HTML se classent en 2 grandes familles : les éléments **lignes** et les éléments **blocs**. Une règle d’or en HTML :

**Les lignes ne peuvent pas contenir de blocs !**

**Les éléments lignes**

La taille des éléments lignes est déterminée par leur contenu. Les éléments lignes se disposent côte à côte, de gauche à droite, en ne provoquant un retour à la ligne que si l’espace est insuffisant.

Trois éléments lignes font exception à la règle car ils peuvent contenir des blocs depuis l’HTML5 : <ins> et <del> qui sont autorisées à contenir des blocs depuis HTML5

Une image contenant Rectangle, ligne, capture d’écran, carré

Description générée automatiquement

**Les éléments blocs**

Les éléments blocs ont pour but de structurer la page en zones rectangulaires. Ils s’affichent les uns sous les autres, en s’étendant sur toute la largeur, avec une hauteur dépendant de leur contenu.

Certains éléments blocs font exceptions à la règle car ils ne peuvent pas contenir d’autres blocs :

<h1>, <h2>, <h3>, <h4>, <h5>, <h6> et <p>

**1.5 Les principaux éléments HTML**

**Les titres et paragraphes**

Les titres se codent à l’aide des éléments <h1> à <h6>

Les paragraphes de texte se codent avec <p>.

Par page, le titre <h1> est sensé être unique, exception faite de certaines structures de pages décomposées en sections

Un titre <h4> a, par défaut, la même taille que le texte. Les titres de niveau 5 et 6 étant plus petits que le texte de la page, ils sont considérés comme moins importants que le texte par les moteurs de recherche

Ces 7 éléments présentent une particularité : ce sont des blocs, mais ils ne peuvent pas contenir d’autres blocs. Ils peuvent cependant contenir des éléments lignes comme <i> ou <span>*.*

L’élément <p> présente une autre particularité : on peut omettre de le fermer.

Le navigateur considère l’élément fermé dès qu’il rencontre la fermeture de son parent ou qu’il rencontre le début d’un nouveau <p>.

Eléments d'enrichissement du texte, de structure de page, interdits et spéciaux page 30-32 du syllabus.

**1.6 Les couleurs**

En HTML, il existe deux syntaxes pour coder les couleurs :

* Par leurs noms
* Par code hexadécimal

**Coder des couleurs par leurs noms**

Les noms de couleurs en anglais sont plus explicites que les codes hexadécimaux, mais ils sont parfois incompatibles et sont surtout beaucoup moins nombreux

<body bgcolor="black" text="grey" link="yellow"…

**Coder des couleurs par code hexadécimal**

C’est le système de la synthèse additive RVB : partant du noir (absence de couleur), des composantes de couleur primaire sont ajoutées pour tendre vers le blanc. Les couleurs primaires sont dans ce cas le rouge, le vert et le bleu

Le code hexadécimal commence par un croisillon « # » et est composé de 6 chiffres allant de 0 à F. (0 étant le plus foncé et F le plus clair)

Les 2 premiers chiffres représentent la composante rouge, les 2 chiffres du milieu la composante verte et les 2 chiffres de la fin la composante bleue

Chaque composante a une valeur comprise entre 00 à FF, couvrant ainsi 256 possibilités (FF = 16x15 + 15 = 255). Le nombre de couleurs possibles est de 256 x 256 x 256 = 16 777 216.

#888888, #808080, #333333 = gris car les 3 parties ont la même valeur

Pour éclaircir une couleur : augmenter les 2 autres parties avec une valeur égal. (ex : #FF8888 )

3 dimensions

**Synthèse soustractive**

Procédé consistant à combiner l'absorption d'au moins 3 colorants pour obtenir les nuances d'une gamme. Le terme soustractif vient du fait qu'un objet coloré absorbe une partie de la lumière incidente. Il soustrait donc de l'énergie de celui-ci.

==> Tend vers le noir

**Synthèse additive**

Procédé consistant à combiner les lumières de plusieurs sources colorées dans le but d'obtenir une lumière colorée quelconque dans un gamut déterminé.

==> utilisé dans l'Informatique

==> Tend vers le blanc

**1.7 Les listes**

**Listes ordonnées et liste non-ordonnées**

Liste ordonnée <ol>…</ol> (Ordered List) Liste dont les éléments sont numérotés. Liste non-ordonnée

Liste non-ordonnée <ul>…</ul> (Unordered List) Liste dont les éléments sont pucés (disque, cercle ou carré).

Élément de liste <li>…</li> (List Item) Chaque élément de la liste

Fermeture des <li> non obligatoire (comme <P>)

<li> toujours enfant de <ul> ou <ol>

La dénomination « ordonné », ne signifie pas que la liste a été triée, mais que les éléments de la liste sont numérotés

**Les attributs**

Attributs des Listes ordonnées <ol> :

* Type : apparence de la puce: A, a, I, i ou 1
* Start : valeur de départ de la numérotation
* Reversed : inverse la numérotation (attribut à utiliser sans valeur)

Attributs des listes non-ordonnées <ul> :

* Type : apparence des puces (disc, circle ou square)

Attributs des éléments de listes <li> :

* Value : définit le numéro d'un élément de liste (par exemple pour sauter de l'élément 5 à l'élément 8)

**Listes de définitions ou d'associations**

L’élément <dl> (definition list) permet de coder une liste d’une ou plusieurs définitions. Chaque mot à définir est contenu dans un élément <dt> (definition term) et chaque définition par dans un élément <dd> (definition definition).

**1.8 Les tableaux**

L'usage des tableaux en HTML sert à aligner des données selon un repaire à 2 dimensions (lignes / colonnes). Les tableaux HTML ne doivent surtout pas être utilisés pour créer la structure d’une page web, mais seulement pour structurer des données au sein d’une page.

En HTML, comme les éléments ne peuvent pas se croiser, nous ne parlerons pas de lignes et de colonnes, mais de lignes et de cellules : les tableaux sont divisés en lignes et chaque ligne est divisée en cellules.

Les cellules d’un tableau peuvent contenir n’importe quel élément HTML : du texte, des images, des liens et même d’autres tableaux.

**Fusion de cellules**

Une fusion horizontale s’obtient avec ***colspan***. Une fusion verticale s’obtient avec ***rowspan***

Pour fusionner des cellules, conservez la première de ces cellules, codez-y l’attribut colspan ou rowspan avec pour valeur le nombre de cellules fusionnant. Puis, supprimez les autres cellules à fusionner.

**1.9 Les liens**

**L'hypertexte**

L’hypertexte est un concept d’agencement des informations de façon à les lier entre elles par des hyperliens, en opposition avec la linéarité des livres, des discours ou des vidéos.

HTML est un langage hypertexte qui permet de cliquer sur un mot ou une image afin d’accéder à de nouvelles informations

**Les liens**

La balise HTML permettant la création de ces liens hypertextes est la balise <a>

Attributs de l'élément <a> :

* **href** (pour Hypertext REFerence) contient l’adresse (URL) du document ciblé
* **Title** (facultatif) : contient une explication sur la cible du lien. Cette explication apparaîtra lors du survol prolongé du lien. Il est vivement déconseillé de coder un attribut title identique à l’intitulé du lien
* **Target** (facultatif) précise la cible du lien et peut prendre les valeurs :
  + "\_blank" qui indique au navigateur qu'il doit créer une nouvelle fenêtre afin d'y afficher le fichier. Cette valeur doit faire intervenir un attribut rel="noopener" pour raison de sécurité.
  + "\_self" qui indique que le fichier sera chargé dans la même fenêtre que celle dans laquelle se trouve le lien.
  + "\_top" qui implique l'affichage du fichier sur toute la surface de la fenêtre du browser.
  + Dans le cas des frames, l’attribut target peut également prendre comme valeur le nom d’un frame.

**Les protocoles HTTP et HTTPS**

Lorsque vous chargez une page web, le navigateur envoie une requête sur le réseau, en suivant scrupuleusement un protocole appelé HTTP ou HTTPS pour sa version plus sécurisée

En HTTP, les données transitent « en clair » : quiconque intercepterait les messages transitant sur le réseau, peut en lire le contenu. Ceci est évidemment problématique lorsque ces messages contiennent des mots de passe ou d’autres données personnelles.

En HTTPS, la connexion débute par un échange de clés entre le serveur et le navigateur. Les clés servent ensuite à crypter les messages avant leur envoi, puis les décrypter lors de la réception. Les sites web ne supportent pas tous ce protocole, car il nécessite une configuration spécifique sur le serveur et un certificat SSL.

**Erreurs**

Lors du chargement d’une ressource, différentes erreurs peuvent survenir. La plus répandue est l’erreur 404 signalant une ressource manquante

Chaque requête http comporte un code d’état :

* 200 : succès de la requête
* 301 ou 302 : redirections
* 401 : utilisateur non authentifié
* 403 : accès refusé
* 404 : fichier non trouvé
* 500, 503 et 504 : erreurs serveur.

**Les URL**

Chaque ressource disponible sur le Web possède une adresse unique appelée URL (Uniform Resource Locator)

Certains caractères ne sont pas tolérés dans les URL et doivent être remplacés par des codes. Par exemple les espaces sont remplacés par %20, les guillemets par %22 et les pourcents par %25

Il existe 2 façons de coder les URL : **URL absolues et URL relatives**

**Les URL absolues**

Les URL absolues décrivent l’entièreté du chemin : du protocole jusqu’au nom du fichier et son extension, en passant par le nom du serveur et les noms des répertoires. Dans la pratique, les URL absolues sont utilisées pour les ressources **externes**, provenant d’autres sites.

Chaque URL absolue se compose :

* du protocole d’échange de données (HTTP, HTTPS, FTP, SMTP, etc.)
* du nom de domaine
* des répertoires d’accès du document
* du nom du document et de son extension

**Les URL relatives**

Les URL relatives sont plus simples à écrire car elles ne décrivent que le chemin entre le dossier où l’on se trouve et le fichier de destination.

Exemples de liens avec URL relatives pour accéder aux fichiers suivants :

C:/doc/page1.html

C:/doc/page2.html

C:/doc/dossier/page3.html

C:/page4.html

qa Lien vers page2 
ca Lien vers page3 
ga href=". Lien vers page4 

Les URL relatives sont utilisées pour désigner tous les fichiers internes au site (sur un même domaine) et ce, pour deux raisons : cela simplifie le code (plus léger et plus lisible) et cela permet de transporter le site sans briser les liens internes. Effectivement, si vous changez votre site de dossier ou de domaine, les URL absolues des ressources internes s’en retrouvent modifiées

**Les ancres**

Une ancre est un lien pointant vers un endroit précis du document HTML. Les ancres sont par exemple utilisées pour envoyer le visiteur en bas ou en haut de la page, ou au commencement d'un chapitre

La cible, appelée aussi le point d’ancrage, est désignée par la valeur de son attribut ID. Cette valeur peut s’utiliser en fin d’URL, précédée du symbole croisillon « # »

du chapitre 
«a le chapitre 3</ay 

**1.10 Les images**

JPEG : format d’images matricielles, applique une compression à taux variable et ajustable, impliquant une perte de qualité souvent indiscernable à l’œil nu

PNG (Portable Network Graphic) : format d’images matricielles, contrairement au JPEG, PNG applique une compression sans perte

Il existe 8 formats de PNG dont les 2 plus courants sont :

* **PNG 8 bits** : 256 couleurs
* **PNG 24 bits** : 16 777 216 couleurs disponibles : pour les images riches en couleur ou pour les photos

GIF : format d’image qui permet la transparence et les animations (GIFs animés), basé sur un algorithme de compression sans perte. Malgré un débat tendu sur la prononciation francophone de « gif », il s’avère finalement que la prononciation officielle est « jif »

WEBP : format d’image développé par Google et s’annonçant très prometteur : léger, avec ou sans perte lors de la compression (au choix), permet la transparence et les animations. Seules ombres au tableau : son incompatibilité avec IE et avec Safari pour les versions inférieures à 14.

SVG (Scalable Vector Graphics) : format d’image vectoriel basé sur le méta langage XML. Très léger, permet la translucidité et la transparence. Permet également des animations. Grâce à son format XML, les images SVG peuvent être modifiées par des scripts en temps réel.

En conclusion, le format JPEG est un bon choix pour les photos et le format PNG 8 bits pour les images « synthétiques ». Le SVG reste un choix intéressant pour les logos, les images géométriques ou pour relever certains défit techniques

Les formats de nouvelle génération comme WEBP sont la garantie de fichiers beaucoup plus légers. Mais ces formats entrainent malheureusement des problèmes de compatibilité encore à l’heure actuelle

**Résolutions d'image**

Les fichiers images, comme les JPEG ou les PNG, sont constitués de points appelés « pixels ». Le nombre de pixels d’une image est appelé **définition** de l’image

Lors de l’impression d’une image sur une certaine surface, on peut en mesurer le nombre de pixels par pouce (1 pouce = 2.54cm). Cette mesure s’appelle la **résolution de l’image** et est exprimée en DPI (Dots Per Inch), ou **PPP** en français (Points Par Pouce).

**Les images : <img>**

Pour afficher une **image** :

<img src="url de l'image" alt="texte alternatif">

Alt : Texte alternatif : description de l’image. Cet attribut est obligatoire, même s’il est laissé vide pour certaines images de décoration, n’illustrant pas le sujet

Src : Source : URL de l’image. Cet attribut est obligatoire. Forcément

Height / width : Dimensions : Hauteur et largeur. En règle générale, les dimensions des éléments HTML sont codées en CSS, mais une exception est prévue ici car préciser les dimensions d’une image permet au navigateur de réserver l’espace nécessaire avant leur chargement. En HTML, les dimensions en pixels s’écrivent sans l’unité : width="400"

Title : Bulle d'informations lorsque l'on survole l'image sans bouger

Wifth : En HTML, les dimensions en pixels s’écrivent sans l’unité : width="400". Autre dimension est adapté avec le navigateur

Usemap : référence a une carte

**Légendes de médias**

<figure> et <figcaption> permettent d’associer une légende à un contenu de type image, vidéo, tableau, etc.

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

**Les sons et vidéos**

Les attributs :

Controls ou controls="controls" : équipe le lecteur de boutons de contrôle (play, pause, etc.)

Preload="auto" : permet de spécifier au navigateur de débuter le téléchargement de la vidéo tout de suite, en anticipant le fait que l'utilisateur lira la vidéo

Autoplay ou autoplay="true" : permet de lancer la lecture automatiquement. certains navigateurs bloquent ce paramètre s’il n’est pas accompagné de muted.

Muted ou muted"true" : coupe le son de la vidéo

Loop ou loop"true" : provoque une lecture en boucle de la vidéo

**Balise audio = balise vidéo sauf qu'on met audio a la place de vidéo**

**1.11 Les frame et iframe**

**<frame>**

Technologie obsolète. Elles servaient a séparer les pages en zones. Chaque zone doit être codée séparément

Dans la page principale, la balise frameset remplace la balise body. Elle sert à séparer la page en différents frames et définit les positions et répartitions de ces frames.

**<iframe>**

Si les balises et sont dorénavant dépréciées, la balise est encore utilisée pour importer des contenus HTML externes au sein d’une page. Elle est, par exemple, souvent utilisée pour importer des contenus de YouTube, de Facebook, etc.

**1.12 Les formulaires**

La structure du formulaire (les différents champs à remplir) se fait en HTML. Mais le traitement des informations (leur affichage, leur enregistrement dans une base de données, etc.) nécessite un langage dynamique comme le PHP ou le JavaScript.

**Formulaire : <from>**

<from action="index.php" method="post"> … </from>

Il peut y avoir plusieurs formulaires dans une même page

Le formulaire est délimité dans le code HTML par : <from> </from>. Tous les champs d’un même formulaire doivent être contenus entre ces 2 balises.

* Action : Indique la destination des données. Peut contenir l’URL du programme qui traite les données ou l’adresse email de destination (exemple : action="mailto:ivan.miller@heh.be").
* Method : Deux valeurs possibles : get ou post. Avec get, les données apparaissent dans l'URL de la page, laissant la possibilité à l’utilisateur de les modifier. Cette méthode convient aux formulaires de recherche.

Avec post, les données sont contenues dans l'en-tête de la page de manière invisible pour l’utilisateur. Cette méthode convient davantage aux formulaires de connexion, de commandes, etc

* Autocomplete : Autorise le navigateur à autocompléter les champs. Peut valoir « on » (par défaut) ou « off ». Peut être utilisé dans l’élément ou dans d’autres éléments de formulaire (<input>, <select>, <textarea>, etc.)
* Enctype : Format d’envoi des données (uniquement utile si le formulaire permet l’envoi de fichiers).

**Conteneur : <fieldset>**

L'élément <fieldset> est un bloc dont l’usage est conseillé dans le cas des formulaires afin de contenir les champs. L'élément , enfant de , permet d'ajouter une légende apparaissant par défaut en haut à gauche à mi-hauteur de la bordure

**Champs de saisie : <input>**

Les formulaires comportent de nombreux champs de saisie dont la nature peut varier : champ texte, champ mot de passe, bouton, case à cocher, etc. Tous ces champs de saisie se codent avec l’élément <input>, dont l’attribut *type* définit la nature du champ. Si le navigateur ne comprend pas la valeur de l’attribut *type*, il affiche un champ texte.

L'élément input propose de nombreux attributs listés à la page 56 du syllabus

**Zone cliquable étendue : <label>**

Les champs de formulaires sont cliquables, parfois pour provoquer le focus, parfois pour cocher un bouton radio ou checkbox. De nombreux utilisateurs ont l’habitude de cliquer sur l’intitulé du champ plutôt que sur le champ lui-même. Cette fonctionnalité n’est présente que si vous avez étendu la zone cliquable en codant un élément <label> .Il est conseillé de toujours coder un <label> pour les champs de saisie suivant : text, checkbox, radio, file et password

Le lien entre un et son champ de formulaire est exprimé par l’attribut for qui reprend la valeur de l’id du champ.

**Champ multi-lignes : <textarea>**

L’élément <textarea> permet au visiteur d’introduire un texte long, de plusieurs lignes

Attributs :

* name : le nom du champ.
* rows : le nombre de lignes.
* cols : le nombre de colonnes (nombre de caractères par lignes).
* readonly : le champ se trouve alors en mode lecture seule, c'est-à-dire que le texte qui s'y trouve ne peut pas être effacé ou modifié

**Listes déroulantes : <select>**

par exemple pour le choix du pays ou de la langue, mieux vaut utiliser les listes déroulantes <select> dont les attributs sont :

* name : le nom de la liste.
* size : (facultatif) dont la valeur unité (par défaut) signifie qu'on a affaire à une liste déroulante sinon c'est une liste normale mais avec barre de défilement.
* multiple : (facultatif) qui signifie qu'il est possible de sélectionner plusieurs valeurs. Si multiple n'est pas spécifié, il ne sera possible de faire qu'un seul et unique choix.

Le conteneur <select> contient autant d’éléments <option> que de choix possibles. Chaque <option> comprend un attribut *value*. C’est la valeur de l’option sélectionnée qui est envoyée au serveur lors de la soumission du formulaire

**Liste d'autocomplétion : <datalist>**

<datalist> permet de lier un champ de formulaire à une liste prédéfinie, pour que le navigateur suggère les valeurs de la liste.

<datagrid> est une sorte de <datalist> plus complexe permettant de structurer les informations sous formes d’arbres (hiérarchie).

**1.13 Les balises meta**

Les balises meta sont toujours codées dans le <head> car elles contiennent des informations (ou méta-données) qui ne sont pas affichées. Ces métadonnées sont utiles aux moteurs de recherche, aux navigateurs et aux visiteurs curieux.

Démarche à suivre :

1. Spécifier le format d'encodage
2. Spécifier l'auteur du site
3. Spécifier le viewport
4. Spécifier une description
5. Spécifier des mots-clés
6. Spécifier les outils de développement du site
7. Choisir le comportement des robots
8. Le rafraîchissement et la redirection

En conclusion, à part la méta-balise précisant le format d’encodage des caractères qui est obligatoire, il revient au développeur de choisir les informations qu’il veut stipuler dans ses pages

**Viewport, device-width et initial-scale**

Spécifier le viewport (pour permettre un affichage adapté aux mobiles) :

<meta name="viewport" content="width=device-width, initail-scale=1.0">

Sur mobiles, les pixels physiques (surface physique) diffèrent souvent des pixels CSS (surface utilisable ou device-width).

La taille du viewport, elle, ne dépend d’aucune des deux surfaces précédentes, mais dépend du navigateur. Souvent, il s’agit d’une dimension élevée (souvent entre 800px et 1024px) afin de pouvoir y afficher des sites web en entier mais dézoomés

Le niveau de zoom initial (initial-scale) vaut device-width / viewport.

En forçant un zoom initial de 1.0 dans le viewport, le navigateur affichera le site sur 320px et pas sur 980px et ce sera au CSS d’adapter le site à cette largeur de 320px. C’est l’utilité de la méta-balise ci-dessus.

**Chapitre 2 : CSS**

**2.1 Introduction aux feuilles de style CSS**

Le langage CSS (Cascading Style Sheets ou feuilles de style en cascade) est utilisé pour définir la mise en page des documents HTML.

Les avantages apportés par le CSS sont multiples :

* Simplifie le code HTML, donc le rend plus léger et plus lisible.
* De facto, le chargement des pages en est accéléré.
* Facilite la maintenance du code.
* Garantit l’homogénéité de la mise en page.
* Améliore le positionnement des sites dans les moteurs de recherche.
* Permet un affichage différent selon les périphériques de sortie.
* Permet de nouvelles possibilités de mise en page comme le positionnement des blocs, les ombres, les transitions, les transformations, les animations, les filtres, etc.

**2.2 Comment utiliser les CSS**

Le principe d'une feuille de style est d'associer des propriétés visuelles à des éléments HTML.

**3 types de feuilles de style**

1. Style incorporés : sous la forme d’un attribut HTML « style ». Cette écriture est tolérée mais va tout de même à l’encontre de la séparation entre structure et présentation



1. Feuilles de style incorporées : sous la forme d’un élément . Cette écriture est utile par exemple dans le cas d’une page sensiblement différente des autres, qui mériterait quelques lignes de CSS exclusives. ==> la plus déconseillée car très lourd

< styl 
< / s tyle > 

1. Feuilles de style externes : sous la forme d’un fichier lié aux pages HTML par l’élément , toujours situé dans le . Ces feuilles externes sont les plus utiles : une seule feuille de style peut être liées à toutes les pages du site et garantir un maximum d’avantages



**Comment écrire une feuille de style**

P10Piété 
A font-family: Verdana; } 
Sélecfeul- 
Déclal-ation 

* Sélecteur : définit quels éléments sont concernés par les déclarations qui suivent.
* Déclaration : le bloc de déclarations compris entre { } peut contenir plusieurs déclarations séparées par des « ; ». Chaque déclaration est une paire propriété / valeur.
* Propriété : décrit un aspect de la présentation d’un élément : comme sa taille ou sa couleur.
* Valeur : définit la valeur de la propriété. Ci-dessus, la propriété « font-family » (la police d’écriture) est « Verdana ».
* Règle : le tout s’appelle une règle CSS

**C comme Cascade**

CSS = Cascading Style Sheets

Le terme "cascading" souligne les caractéristiques de présentation qui se propagent "en cascade" d'un élément à ses descendants

Certaines propriétés CSS ne s’héritent pas : les positions, les marges, les dimensions (largeur et hauteur), les bordures, etc.

**Commentaire en CSS**

Il est possible de coder des commentaires presque n'importe où dans une feuille de style CSS en les entourant de /\* et \*/, comme en langage C.

**Les médias**

Il est possible de tester le périphérique de sortie, afin d’appliquer ou non des règles CSS, par exemple pour corriger l’apparence du site lors de son affichage sur un mobile ou lors de son impression.

Les périphériques (ou médias) que l’on peut cibler sont : **screen** (écrans), **handheld** (Périphériques mobiles ou de petite taille), **print** (impression), **aural** ou **speech** (synthétiseurs vocaux), **braille** (plages braille), **embossed** (imprimantes braille), **projection** (projecteurs), **tty** (terminal avec police monoespacée), **tv** (téléviseur), **all** (tout).

==> utiliser avec la commande @media

Les ***media-queries*** sont composées d’expressions booléennes (toujours entre parenthèses) et de médias (screen, all, ...) reliés par des opérateurs logiques : and (et), only (uniquement), not (non). L’opération « ou » s’obtient en séparant des expressions par des virgules

**2.3 Les sélecteurs**

* Le sélecteur de type : sélectionne tous les éléments de même type

==> x { }

line -height : 2em; 

* Le sélecteur d'ID (#) : sélectionne tous les élément dont l'ID est *nom*

==> #nom { }

#ciel { 
color:white • 
background- color: blue • 

* Le sélecteur d'ID et de type (.) : sélectionne tous les éléments de type x dont l'ID est *nom*

==> x#nom { }

* Le sélecteur de classe : sélectionne les éléments dont la classe est *nom*

==> .nom { }

color : red • 

* Sélecteur de classe et de type : sélectionne tous les éléments x dont la classe est *nom*

==> x.nom { }

* Sélecteur multiple : sélectionne tous les éléments x et tous les éléments y

==> x, y { }

div, *nav { 
color : gold ; 

* Sélecteur descendant : sélectionne tous les éléments y contenus dans une élément x (descendants)

==> x y { }

div p { 
font-weight: bold; 

* Sélecteur d'enfant (>) : sélectionne tous les éléments y directement contenus dans un élément x (=les enfants y de x)

==> x>y { }

div>span { 
color: blue; 

* Sélecteur universel (\*) : sélectionne tous les éléments

==> \* { }

div • Span { 
color : blue; 

* Sélecteur universel : sélectionne tous les éléments y s'ouvrant après la fermeture de x et ayant le même parent

==> x \* y { }

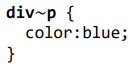
* Sélecteur de frères adjacents(+) : sélectionne l’élément y s’ouvrant juste après la fermeture d’un élément x et ayant le même parent

==> x+y { }

div+p { 
font-weight : bold; 

* Sélecteur de petits frères(~) : sélectionne tous les éléments y s’ouvrant après la fermeture de x et ayant le même parent

==> x~y { }



* Sélecteur de pseudo-classes : sélectionne des éléments selon leur état.

==> x:z { }

{ text-decoration: none; 
a 
a:link { } 
a:visited { } 
a: hover, a:focus { text-decoration: underline; color 
a:active { } 

**2.4 Les déclarations et leurs priorités**

**Déclarations, héritage et cascade**

Les déclarations CSS permettent de modifier l’aspect des éléments et sont toujours de la forme propriété:valeur;

Certaine déclaration sont héritées par les éléments descendant de l'élément sélectionné. Cet héritage peut toute fois être contredis par une nouvelle déclaration sur l'un des descendants.

Cette faculté d’héritage en cascade nous évite de coder les propriétés de mise en forme des textes pour chaque descendant.

**Les règles du CSS du navigateur**

Certains éléments disposent de leurs propres déclarations CSS par défaut. Ces règles CSS sont de la forme *a { color:blue; text-decoration:underline; }* et sont à prendre en compte comme toute autre règle CSS lors de l’héritage en cascade

**Imposer une déclaration prioritaire**

Il est possible de rendre une déclaration prioritaire sur d’autres déclarations concernant un même élément



**Priorité des déclarations**

Pour connaître la priorité d’une déclaration, une formule assez simple permet de donner une valeur à chaque déclaration : la valeur la plus haute l’emporte ; en cas d’égalité, la dernière déclaration dans le code l’emporte

Pour calculer la priorité des déclarations CSS il faut décomposer le sélecteur et additionner les valeur suivantes :

1. *!important* = 1000
2. Déclarations sous forme d'attribut *style* = 1000 mais passent après *!important*
3. Sélecteur d'ID = 100
4. Sélecteur de classe = 10
5. Sélecteur de type = 1
6. Sélecteur universel <<\*>> = 0

**Remarque** : dans ce calcul, il n’y a pas de report de rang, c’est-à-dire que 11 sélecteurs de classe ne valent pas plus qu’un sélecteur d’ID

**2.5 Les unités de mesure**

**Les dimensions**

Pixel : 320px

Pourcentage : 100%

**Les dimensions pour l'impression**

Centimètre : 2cm

Millimètre : 18mm

Pouce : 1in ( 1 inche = 2.54 cm )

Pica : 1pc ( 1 pica = 12pt )

Point : 12pt ( 1 point = 1/72 in ; une police de 12pt équivaut à 16px)

**Les cadratins**

Cadratin relatif à la largeur de la lettre « m » : 1em

Cadratin relatif à la hauteur de la lettre « x » : 1ex (peu utilisé)

Appliqué à la propriété font-size, le cadratin *em* agit comme un multiplicateur du font-size de l’élément parent. Appliqué à une autre propriété, il multiplie le font-size de l’élément lui-même

Root em (rem) : identique à *em*, le cadratin rem se base sur le font-size de la racine.

**Les angles**

Degrés : 90deg

Grades : -0.5grad

Radian : 1rad

Tour : 10turn

(180deg = 200grad = 0.5turn = 3.1416rad)

**Le temps et les fréquences**

Seconde : 3s

Milliseconde : 400ms

Hertz : 10000Hz

KiloHertz : 10kHz

**Calc()**

calc() permet un calcul utilisant les opérateurs + - \* / et portant sur des valeurs pouvant présenter des unités différentes

**Variables CSS**

Très pratiques pour s’assurer de la cohérence graphique d’un document, les variables CSS permettent de centraliser les valeurs récurrentes dans la feuille de style

: root { 
- -background: #333; 
- -text : #ddd ; 
body { 
ba ckground - color : var( - - background) ; 
color: var(- -text); 

**2.6 Les propriétés**

**Couleur**

Le CSS permet de modifier la couleur du texte (**color**), du fond (**background-color**), des bordures (**border-color**), la couleur d’accentuation des champs de formulaire (**accent-color**), etc.

**color: #D33 ;**

Le CSS supporte de nombreuses syntaxes différentes comme :

* #rrggbb
* #rgb
* #rrggbbaa (où aa est la transparence)
* #rgba (où a est la transparence)
* Rgb(r,g,b) ; où r, g et b sont des valeurs entières de 0 à 255
* Rgb(r,g,b) ; où r, g et b sont des pourcentages
* Rgba(r,g,b,a) ; ex. : 0.75 pour 75%
* Hsl(h,s,l) ; où h (hue) est la teinte, s la saturation et l (lightness) la clareté
* Hsla(h,s,l,a)
* Par un nom en anglais
* Par certain mots-clés (transparent, currentcolor (valeur de color), inherit, initial, unset)

**Opacité**

Attention à ne pas confondre l’opacité et la transparence (rgba, hsla). L’opacité s’applique à tous les descendants d’un élément et attend une valeur réelle entre 0 et 1. Il s’agit d’une propriété CSS3.

**opacity: 0.75 ;**

**Visibilité**

La propriété visibility permet de rendre un élément invisible tout en conservant son emplacement. Cette propriété peut prendre les valeurs visible ou hidden.

**visibility: hidden ;**

**Police de caractères (fonte)**

Une fonte est une combinaison de police, de taille et de graisse

La propriété ***font-family*** autorise de lister plusieurs noms de police, ordonnés et séparés par des virgules

Les valeurs ***serif*** et ***sans-serif*** représentent les familles de polices avec et sans empattements, alors que la valeur ***monospace*** représente la famille de polices mono-espacées (chaque caractère a la même largeur), ***cursive*** la famille de polices simulant l’écriture à la main et ***fantasy*** la famille de polices décoratives.

**font-family: Trebuchet MS, Arial, sans-serif;**

Les polices rares ou exotiques, ont de faibles chances d’être installée sur les machines de vos visiteurs, vous devez donc l’incorporer à votre site.

**Mise en forme du texte**

L'essentiel des déclarations de police :

* **Color** : la couleur du texte

**color: #333 ;**

* **Font-family** : la police de caractères

**font-family: Trebuchet MS, Arial, sans-serif;**

* **Font-size** : la taille de la police

**font-size: 1.5rem;**

* **Font-style** : italique, oblique ou normal

**font-style: italic;**

* **Font-weight** : bold (gras), normal (non-gras), bolder et lighter (plus ou moins gras que l'élément parent)

**font-weight: bold;**

* **Line-height** : change la taille de l'interligne

**line-height: 1.5;**

* **Text-align** : alignement du texte (left, right, center et justify)

**text-align: center;**

* **Text-decoration** : modifie ou supprime la décoration du texte (**none**, **underline** (souligner), **overline** (surligner), **line-through** (barrer))

**text-decoration: none;**

3 sous-propriétés :

* **Text-decoration-line** (none, underline, overline, line-through)
* **Text-decoration-color** (une couleur)
* **Text-decoration-style** (solid, wavy, double, dotted, dashed)

**text-decoration: wavy purple underline ;**

* **Text-shadow** : applique une ombre au texte (**box-shadow** pour les blocs)

**text-shadow: 1px 1px 2px #999;**

* **Vertical-align** : modifie l'alignement vertical (**sub** (indice), **super** (exposant), **baseline** (sur la ligne), **top**, **middle**, **bottom**, **text-top**, **text-bottom**)

**vertical-align: middle;**

**Curseur**

La propriété **cursor** permet de modifier le curseur de la souris lorsque celui-ci survole un élément.

**cursor: pointer;**

I auto 
al I —scroll 
+ crosshair 
I inherit 
pointer 
progress 
I text 
vertical—text 
wait 
drop 
not-al loved 
e—resize 
•i 
row—eesize 

**Colonnes multiples**

Appliquées à un conteneur, les déclarations ci-dessous vont provoquer une disposition en 3 colonnes avec des goutières de 1em et des règles (barres de séparation) de 1px gris-clair continu.

**column-count: 3;**

**column-gap: 1em;**

**column-rule-style: solid;**

**column-rule-width: 1px;**

**column-rule-color: lightgrey;**

**Largeur et hauteur**

Width = largeur

Height = hauteur

La valeur auto demande la taille par défaut.

**width:80%; height:220px;**

Des valeurs minimales et maximales peuvent compléter ces 2 propriétés :

**min-width:100px;**

**max-width:960px;**

Tout élément bloc se compose d’un contenu enrobé de trois zones : le remplissage (padding), la bordure (border) et la marge (margin). Les propriétés width et height n’agissent que sur le contenu

margin 
border 
padding 

**Bordures et marges**

* **Marges (margin)** : seule l'étendue peut être définie. Pour cela, on utilise les propriétés **margin-top**, **margin-right**, **margin-bottom** et **margin-left.**

**margin-top : 120px ; /\* marge supérieure de 120px \*/**

**margin : 0 ; /\* 4 marges nulles \*/**

**margin : 80px auto ; /\*verticales de 80px ; horizontales automatiques\*/**

**margin : 10px 0 20px 2em ; /\* 4 valeurs : top, right, bottom, left \*/**

* **Remplissage (padding)** : Les marges intérieures (**padding**) se règlent exactement comme les marges extérieures (**margin**)

* **Bordure (border)** : La bordure se situe entre le remplissage et la marge. 3 caractéristiques : **width, color** et **style**

Les valeurs possibles pour border-style sont **none** (aucun, par défaut), **hidden** (invisible), **dotted** (pointillé), **dashed** (tirets), **solid** (continu), **double** (épais), **groove** (relief), **ridge** (bourrelet), **inset** (creux) et **outset** (relief).

**border : 2px #333 solid ;**  /\*une bordure continue anthracite de 2px\*/

**border-width : 5px 10px ;**  /\*épaisseurs horizontale et verticale\*/

**border-top : 3px dotted #f00 ;** /\*une bordure top rouge pointillée\*/

**border-bottom-color : red ;**  /\*une couleur rouge en bas\*/

Il est possible de charger une image de bordure avec border-image, puis de la régler avec **border-image-outset, -repeat, -slice, -source** et **-width**

* **Contour (outline)** : seconde bordure à l’extérieur de la première

**outline : 2px red solid ;** /\*utilisation similaire à border\*/

**outline-offset : 4px ;** /\* distance entre le contour et l’élément \*/

**Calcul de dimensions des blocs**

Ce bloc a une largeur apparente de 140px dont 100px pour disposer le contenu :

**div { box-sizing:content-box; width:100px; padding:20px; }**

Ce bloc a une largeur apparente de 100px dont 60px pour disposer le contenu :

**div { box-sizing:border-box; width:100px; padding:20px; }**

**Alignement : centrer un élément**

Pg 91

**2.7 Les positionnements**

**Positionnement absolu**

Les propriétés top, bottom, left et right définissent les éloignements d'un bloc absolu par rapport à la page

**Positionnement fixe**

Positionnement recouvrant le reste de la page et ne défilant pas quand on scroll

**Positionnement sticky**

Positionnement normal, puis fixe lorsque le scroll dépasse l'élément

**Positionnement relatif**

Positionnement relatif permet :

* De décaler un bloc par rapport à sa position
* De modifier l'origine (point(0,0) de référence pour les blocs absolus)

Combo relatif + absolu

Le positionnement relatif modifie l'origine

**Propriété z-index**

Pour modifier l'ordre de chevauchements