

accès
libre

Premiers pas
en **CSS 3 &**
HTML 5



EYROLLES

L'informatique libre à la portée de tous !

L'auteur

Ingénieur EFREI, Francis Draillard a exercé dans l'industrie, la recherche, la formation continue et l'enseignement supérieur. Il intervient comme professeur associé à l'EIGSI, école d'ingénieurs à La Rochelle. Concepteur web indépendant et conseil en entreprise, il contribue au site Framasoft dédié aux logiciels libres et à l'évolution de PluXml, système de gestion de contenu libre pour créer des sites modifiables en ligne.

Des bases HTML 5 et CSS 3 aux fonctions de styles avancées, utilisez les meilleurs outils pour créer un site performant et l'adapter aux mobiles !

- **Bases** : l'essentiel du HTML 5 et la structure d'une page, imbrication et hierarchie des balises
- **Multimedia** : images, sons et vidéos dans vos pages
- **HTML 5 avance** : éléments spécifiques, formulaires de contact et contrôle des informations saisies
- **Principe des CSS** : règles CSS et feuilles de styles, présentation homogène et facilement modifiable
- **Mise en forme** : présentation du texte et positionnement des blocs, images et couleurs, listes et tableaux
- **CSS 3 avancées** : ombres et degradés, répartition du contenu, transformations et animations
- **Synthèse** : modèles et exemples complets de pages, adaptation aux tablettes et smartphones

Cette septième édition prend en compte les dernières évolutions des standards HTML 5 et CSS 3.

Annexes : Choix et codage des principales couleurs • Spécificités des navigateurs • Aide-mémoire des propriétés CSS 2 et 3 • Références web et bibliographiques.

www.editions-eyrolles.com

Premiers pas en CSS 3 & HTML 5

DANS LA MÊME COLLECTION

M. DUPONT DE DINECHIN. – **Blender pour l'architecture.**
N°14310, 2^e édition, 2016, 330 pages.

V. KOVALSKY, C. VILLENEUVE. – **Drupal avancé.**
N°14011, 2015, 308 pages.

X. DELENGAIGNE. – **Organiser sa veille sur Internet.**
N°13945, 2^e édition, 2014, 300 pages.

SUR LE MÊME THÈME

R. GOETTER. – **Mémento CSS 3.**
N°67432, 4^e édition, 2017, 14 pages.

D. CEDERHOLM. – **CSS 3 pour les web designers.**
N°11765, 2^e édition, 2016, 152 pages.

H. GIRAUDEL, R. GOETTER. – **CSS 3 - Pratique du design web.**
N°14023, 2015, 354 pages.

S. POLLET-VILLARD. – **Créer un seul site pour toutes les plates-formes.**
N°13986, 2014, 144 pages.

E. MARCOTTE. – **Responsive web design.**
N°67361, 2^e édition, 2017, 168 pages.

Retrouvez nos bundles (livres papier + e-book) et livres numériques sur
<http://izibook.eyrolles.com>

Francis Draillard



Premiers pas
en CSS 3 &
HTML 5

→ 7^e édition

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© Groupe Eyrolles, 2006, 2008, 2010, 2012, 2013, 2015, 2017, ISBN : 978-2-212-67430-9

Avant-propos

Partout dans notre vie, des pages Internet ! Que d'activités professionnelles ou personnelles sont concernées par ce sujet ! Mais la conception d'un site est-elle compliquée ? Qu'y a-t-il derrière ces textes, images et animations dont l'assemblage nous paraît si naturel ? Comment créer des pages à la fois esthétiques et faciles à mettre à jour ?

Cet ouvrage vous guidera pas à pas dans la découverte des techniques de base qu'il faut connaître pour la création d'un site : le langage HTML pour la structure des pages et les feuilles de styles CSS, *Cascading Style Sheets*, pour la mise en forme.

Vous verrez que le langage HTML est assez intuitif et facile à appréhender. Quant aux CSS, il est essentiel de comprendre d'abord leur « philosophie », c'est-à-dire leurs principes de fonctionnement, expliqués ici en douceur. Ensuite, il nous suffira d'explorer ensemble les différentes propriétés de mise en forme, pour savoir comment en tirer parti. Celles-ci sont classées par catégories, en commençant par les plus courantes, bien entendu.

La démarche de ce livre est une approche pratique et progressive de ces techniques, toujours accompagnées d'exemples et de nombreuses illustrations, pour une meilleure compréhension. Une fois les bases acquises, vous trouverez des méthodes plus avancées (mais pas nécessairement plus compliquées) qui exploitent les dernières nouveautés de la norme CSS, toujours expliquées à l'aide d'exemples et présentées par l'image.

En guise de synthèse, deux chapitres détaillent des exemples concrets et complets, toujours amenés de façon progressive et illustrée, qui sont une application des techniques les plus courantes et une occasion de plus de bien les assimiler.

Enfin, l'ensemble du livre est truffé d'astuces, de conseils pratiques et de bonnes adresses : les références de programmes utiles, généralement

sélectionnés dans le meilleur des logiciels libres, et celles de sites qui proposent des outils en ligne pour faciliter le développement d'un site Internet.

Outre la compréhension de la technique pure, l'objectif est ici de présenter les « bonnes pratiques » de la programmation web. Ce n'est pas un qualificatif qui leur a été attribué par un organisme de normalisation ; elles ont acquis cette réputation parce qu'elles apportent à la fois simplicité, précision et souplesse dans la création de nos pages.

En effet, au-delà de la présentation d'un site, il faut penser à sa maintenance et à son évolution. Sera-t-il facile d'en effectuer la mise à jour régulière, de changer sa charte graphique, sa mise en page ? Pourrons-nous nous dispenser de réécrire le code, en réutilisant la structure et l'essentiel du travail de mise en forme ? Bien sûr ! Et vous avez de la chance, tous ces points sont justement l'objet de l'ouvrage qui se trouve entre vos mains !

Les avantages cumulés du HTML et des CSS sont disponibles, profitons-en ! Et découvrons ensemble dans ce livre tout le bénéfice que l'utilisation rationnelle de ces normes nous apporte : une méthode simple et efficace pour créer des sites Internet. Il suffit de s'y aventurer pour être conquis... Bonne lecture !

Structure de l'ouvrage

Le **premier chapitre** est une introduction qui nous présente le principe général du langage HTML, des feuilles de styles et d'une bonne écriture HTML/CSS.

Le **deuxième chapitre** explique de façon concrète les bases du HTML. À partir d'exemples, il détaille l'utilisation des principales balises HTML, présente leur classement par types de balise et leur hiérarchie dans une page web. Il fournit les quelques mots de vocabulaire utilisés par la suite pour expliquer la conception d'une feuille de styles.

Des techniques plus spécialisées apportées par la norme HTML 5 sont traitées dans le **troisième chapitre**. Ces nouveaux éléments pourront être utilisés au fur et à mesure de leur prise en compte par les navigateurs. Vous trouverez dans ce même chapitre tout ce qui concerne les formulaires : éléments de base, envoi en ligne et contrôle de la saisie.

Au **quatrième chapitre** apparaissent les feuilles de styles. À partir d'exemples, nous verrons de quelle façon et à quel endroit les écrire, quelles unités de mesure utiliser, etc.

Les propriétés de style définies pour la norme CSS 2 constituent toujours la base des CSS 3 ; elles sont détaillées dans les **cinquième et sixième chapitres**, qui expliquent respectivement les propriétés de mise en forme et celles liées au positionnement des éléments dans la page. Des exemples illustrent chaque propriété, dont toutes les valeurs possibles sont détaillées. Les principales nouveautés apportées par la norme CSS 3 sont expliquées de la même façon, dans le **septième chapitre**.

Le **huitième chapitre** est une synthèse des précédents : à partir de nos connaissances en HTML et CSS, nous construirons ensemble les trames de sites complets, d'abord avec des blocs de couleur unie, puis avec des images de fond pour créer des sites réellement finis.

Ensuite, nous nous rendrons dans le domaine des pages web pour smartphones et tablettes. Ce **neuvième chapitre** explique dans un premier temps le principe d'adaptation d'une page au Web mobile. Puis, il reprend le dernier exemple complet du chapitre précédent, pour en proposer une version qui s'adapte automatiquement à la taille de l'écran.

Le **dixième chapitre** nous parle des autres médias pour lesquels des propriétés de style existent, et plus particulièrement des pages destinées à l'impression et la diffusion sonore des pages web.

En **annexe** se trouvent des informations sur les choix des couleurs et le récapitulatif de celles qui sont nommées, un tableau de synthèse sur le comportement des principaux navigateurs. Suivent un index des propriétés, en guise de formulaire, ainsi qu'une liste de références bibliographiques et de sites web utiles.

Les fichiers et images qui servent d'exemples dans le livre peuvent être téléchargés à l'adresse <http://livre.antevox.fr>.

Crédits des photographies et illustrations

Tous droits réservés pour toutes les photographies et illustrations publiées dans cet ouvrage.

Les crédits qui ne figurent pas dans les légendes des illustrations sont mentionnés ci-après.

Les pages d'ouverture des chapitres 1, 3, 6 et 7 sont des extraits du site Zen-Garden (<http://www.csszengarden.com/tr/francais/>), il s'agit respectivement des versions suivantes :

- « Tranquille » par Dave Shea (<http://www.mezzoblue.com>);
- « Japanese garden » par Masanori Kawachi (<http://www.jugglinglife.org>);
- « Like the Sea » par Lars Daum (<http://www.redrotate.de/>);
- « Kyoto Forest » par John Politowski (<http://www.rpmdesignfactory.com>).

Figures 4–4 à 4–11, pages d'ouverture des chapitres 4, 5 et 10, ainsi que des annexes A, C et D : copyright 2017 Francis Draillard, Micro Application et ses concédants.

Figure 4-3 : photo d'Éric Pouhier (décembre 2005). Cette figure 4-3 et la photo de la page d'ouverture du chapitre 9 proviennent du site Wikimedia : <https://commons.wikimedia.org/?uselang=fr>.

Page d'ouverture du chapitre 2 : dessin d'Alice Draillard.

Pages d'ouverture du chapitre 8 et de l'annexe B, toutes les figures dont le crédit n'est pas mentionné en légende et qui ne figurent pas dans la liste cidessus : Francis Draillard.

Remerciements

Je tiens à remercier Muriel Shan Sei Fan, ancienne éditrice informatique des éditions Eyrolles. C'est grâce à elle que la publication de ce livre a été possible, et ses conseils m'ont été précieux pour la rédaction finale. Merci aussi à Dimitri Robert : auteur d'un excellent livre sur le logiciel Gimp, dans cette même collection, il m'a aiguillé vers Muriel pour lui proposer mon manuscrit. Je remercie également Alexandre Habian, éditeur informatique qui a repris le poste de Muriel aux éditions Eyrolles et gère donc maintenant un catalogue de grande ampleur. Tous mes remerciements vont aussi à Sophie Hincelin, qui, au-delà de la collection informatique, assure la coordination de nombre d'ouvrages dont celui-ci, en tant que Responsable de production éditoriale chez Eyrolles.

Merci encore à celles et ceux qui ont participé aux différents stades de la conception de ce livre :

- pour leurs conseils judicieux et la coordination de ce livre, à Muriel Shan Sei Fan jusqu'à la cinquième édition, ainsi qu'à Sandrine Paniel pour les troisième et quatrième éditions, et à Sophie Hincelin depuis la sixième édition ;
- pour la relecture à Eliza Gapenne et Anne Bougnoux (première édition), à Jean-Baka Domelevo Entfellner (deuxième édition), à Sandrine Paniel (troisième et quatrième éditions), à Laurène Gibaud (cinquième édition), à nouveau à Eliza Gapenne (sixième édition), ainsi qu'à Sophie Hincelin et son équipe pour cette septième édition ;
- pour la mise en pages à Gaël Thomas et Jean-Marie Thomas lors de la première édition, et à nouveau à Gaël Thomas qui a œuvré seul pour les éditions suivantes.

Je remercie beaucoup pour leur contribution :

- les auteurs du site <http://www.w3.org>, source extrêmement riche de renseignements qui explique dans tous leurs détails les normes du *World Wide Web Consortium* (W3C), ainsi que Jean-Jacques Solari, qui a traduit en français bon nombre de ces documents sur le site <http://www.yoyodesign.org>, dont est notamment extraite une partie des

- tableaux récapitulant les propriétés CSS, qui se trouvent en annexe ;
- le site www.w3schools.com, dont le contenu et les exemples sont un complément utile aux normes éditées par le W3C ;
 - Alain Beyrand (<http://www.pressibus.org>) : son classement des couleurs est très intéressant. Il est publié en annexe (mais en moins bien, car sans les couleurs !).

Je suis reconnaissant également à mes étudiants de l'EIGSI, École d'ingénieurs à La Rochelle. Qu'ils me pardonnent : je me suis servi de leurs erreurs et de leurs difficultés pour rendre ce livre plus clair et plus pédagogique.

Enfin, c'est de tout mon cœur que je remercie mon épouse et ma fille, pour leur soutien et leur participation.

Francis Draillard

contact@antevox.fr

<http://www.antevox.fr>

Table des matières

1. INTRODUCTION AU **HTML** ET AUX FEUILLES DE STYLES **CSS**

Signification de HTML et CSS

Principes de base pour une page web

Choix sensé des balises HTML

Adaptation aux navigateurs

Accessibilité

L'apparence, fonction du thème et du public concerné

Polices de caractères

En résumé, quelques sentiments liés aux couleurs

Homogénéité du site

Principes d'une bonne écriture HTML/CSS : donner du sens au codage

Titre de page

Mise en gras ou en italique

Liste de liens hypertextes (menu)

Intérêt des feuilles de styles

2. L'ESSENTIEL DU **HTML**

Principe des balises

Évolution de la norme HTML

Succession des normes HTML

La philosophie du HTML 5

Évolution et compatibilité

Pragmatisme et tolérance

Premières règles d'écriture HTML

Règles pour les noms des fichiers

Règles d'écriture des balises en HTML

Structure d'une page HTML

Espaces, sauts de ligne et commentaires invisibles

Principales balises HTML

Un exemple pour commencer

Les deux premières balises

En-tête

Corps de la page

Paragraphes et titres

Mise en forme commune à une partie du texte

Principales mises en forme

Italique et gras

Exposant et indice

Annotations en petite taille de caractères

Citation avec retrait

Trait de séparation horizontal

Liens hypertextes

Les listes

Listes à puces ou numérotées

Listes de définitions

Tableaux

Création d'un tableau HTML

Fusionner des cellules

Insertion d'images

La balise image

Dimensionner une image

Une icône sur l'onglet

Contenus audio et vidéo

Exemples d'utilisation

« Pistes complémentaires » pour les contenus audio et vidéo

Des blocs pour structurer les pages

Sections de base d'une page

Sous-sections de type <div>

Deux catégories d'éléments

Éléments en ligne

Éléments de type bloc

Hiérarchie des éléments : l'héritage

Hiérarchie des blocs imbriqués et juxtaposés

Termes hiérarchiques utilisés en HTML/CSS

Héritage des propriétés de style

Validation du code HTML

3. NOUVEAUTÉS DU HTML 5

Davantage de signification pour le texte

- Balises spécifiques pour le texte
 - Surlignage de mots
 - Mesures comprises entre deux bornes
 - Dates et heures
 - Texte barré
 - Coupe des mots trop longs

- Autres balises de texte

Des éléments modifiables

- Éléments déplaçables dans la page
- Contenus modifiables

Blocs spécifiques

- Une page dans un cadre
- Des détails sur demande

Images dynamiques et légendes

- Affichage dynamique (<canvas>)
- Indicateur de progression (<progress>)
- Images, photos et figures (<figure>)

Formulaires de contact

- Balise formulaire
- Regroupement de parties de formulaire
- Les étiquettes
- Zones de texte simples
- Zones de texte sur plusieurs lignes
- Boutons radio, à choix unique
- Cases à cocher
- Listes déroulantes
- Boutons d'effacement et d'envoi
- Fichier d'envoi du formulaire
- Fonctionnement du fichier envoi.php

Contrôle plus précis des formulaires

- Balise form
- Contrôle des balises input
 - Nouveaux types d'entrées
 - Attributs des balises input
 - Balises input utilisées pour la saisie
 - Balises input en forme de bouton
- Nouvelles balises de formulaire

- Listes d'options modifiables
- Affichage du résultat d'un calcul
- Nouveaux attributs pour les éléments de formulaire
 - Rattachement à un autre élément
 - Zones de saisie à plusieurs lignes

À utiliser progressivement

4. ÉCRITURE DES FEUILLES DE STYLES

Définition d'une règle de style

- Principe
- Exemple de règle de style
- Commentaires
- Emplacement des styles

Feuille de styles interne

Feuille de styles externe

Styles en ligne

Sélecteurs de styles

- Comme au théâtre
- Sélecteur simple
- Classe
 - Une catégorie de balises
 - Une même classe pour plusieurs types de balises

Identifiant

- Identifiant sans nom de balise
- Différence entre classe et identifiant

Pseudo-classes

- Pseudo-classes pour les liens hypertextes
- Autres pseudo-classes

Pseudo-éléments

Règle associée à plusieurs sélecteurs

Regroupement de propriétés à l'aide de « raccourcis »

Hiérarchie des sélecteurs

Hiérarchie précise des sélecteurs

- Imbrication directe
- Juxtaposition

Sélecteur d'attribut [...]

Sélecteur universel *

Ordre de priorité des styles

- Règle de style prioritaire
- Degré de priorité d'une règle de style
- Application

Valeurs, tailles et couleurs

- Héritage de propriété
- Unités de taille
 - Unités de taille courantes
 - Autres unités de taille
- Codage des couleurs
 - Noms de couleurs
 - Code RVB
 - Toutes les couleurs sont « sûres »
 - Des outils en ligne pour les couleurs

Exemple de page avec feuille de styles interne

5. PROPRIÉTÉS DE MISE EN FORME

Mise en forme des caractères

- Choix des polices
- Taille de police
- Couleur du texte
- Texte en gras
- Italique
- Soulignement et autres « décorations »
- Majuscules et minuscules
- Petites majuscules
- Surlignage de lettres
- Décalage vers le haut ou le bas
- Raccourci pour la mise en forme de caractères

Paragraphes et blocs de texte

- Alignement horizontal du texte
- Retrait de première ligne
- Interligne minimum
- Espacement entre les lettres
- Espacement entre les mots
- Conservation des espaces et sauts de ligne saisis
- Modification du curseur de la souris
- Affichage automatique d'un contenu
- Guillemets à utiliser

Réinitialisation d'un compteur
Incrémantation d'un compteur
Sens de l'écriture
Écriture bidirectionnelle

Bordures

Style de bordure
Styles de bordure pour chaque côté
Épaisseur de bordure
Épaisseur de bordure pour chaque côté
Couleur de bordure
Couleur de bordure pour chaque côté
Raccourci pour toutes les propriétés de bordure
Raccourci des propriétés de bordure pour chaque côté
Contour superposé à un élément

Images et couleurs d'arrière-plan

Couleur d'arrière-plan
Image d'arrière-plan
Répétition ou non de l'image d'arrière-plan
Alignement de l'image d'arrière-plan
Fixation de l'image d'arrière-plan
Raccourcis pour les arrière-plans

Listes à puces ou numérotées

Type de puce ou de numérotation
Utilisation d'une image comme puce
Position de la puce
Raccourci pour toutes les propriétés de liste

Les tableaux

Largeur fixe ou variable des colonnes ou du tableau
Recouvrement des bordures
Espace entre les bordures de cellules
Contour des cellules vides
Position du titre du tableau
Alignement vertical des cellules

6. POSITIONNEMENT DES BLOCS

Marges et dimensions d'un bloc

Marges externes autour d'un bloc

- Raccourci pour les marges externes
- Marges internes d'un bloc
- Raccourci pour les marges internes
- Largeur fixe pour un bloc ou une image
- Hauteur fixe pour un bloc ou une image
- Largeur et hauteur totales d'un bloc
- Largeur et hauteur minimales
- Largeur et hauteur maximales

Position des éléments

- Flux normal des éléments
- Principe du positionnement des blocs
- Types de position possibles
 - Position normale
 - Position relative, absolue ou fixe
 - Position flottante

- Utilisation des différents types de positionnement
 - Élément dans le flux (position normale)
 - Position relative
 - Position absolue
 - Position fixe
 - Élément flottant

- Type de positionnement d'un bloc
 - Décalages indiquant la position d'un bloc
 - Niveau d'empilement des blocs
 - Transformation en bloc flottant
 - Pas d'éléments flottants sur le côté
 - Affichage ou non d'un élément
 - Affichage des débordements
 - Zone visible d'un élément
 - Changement de type d'élément

Délimitation des blocs

Exemples de positionnement

- Image du haut (nuages)
- Image de l'arbre en position absolue
- Sous-titre « En images » en position relative
- Centrage horizontal du titre
- Titre latéral fixé sur l'écran
- Position de la galerie d'images

Images côte à côté

Centrage d'éléments à l'intérieur des blocs

Centrage horizontal

Centrage horizontal d'éléments en ligne

Centrage horizontal de blocs

Centrage vertical

Centrage vertical d'éléments en ligne

Centrage vertical de blocs

Exemple de centrage vertical

7. PRINCIPALES NOUVEAUTÉS DES CSS 3

Du relief pour nos pages

Codes couleurs et niveaux de transparence

Niveau d'opacité

Codage RGBA des couleurs

Codages HSL et HSLA des couleurs

De nouveaux effets pour le texte

Style d'écriture

Couleur, forme et type de soulignement

Ajustement de la taille des caractères

Une police originale avec @font-face

Ombrage du texte

Présentation et ergonomie

Apparence d'un élément

Marqueur de ligne tronquée

Coupure des mots trop longs

Des bordures plus variées

Des images en guise de bordures

Ces fameux coins arrondis !

Des ombres pour nos boîtes

Espacement pour encadrement double

Dimensions des blocs

Dimensions globales des blocs

Blocs de dimensions modifiables

Couleurs et images de fond

Plusieurs images d'arrière-plan

Placement et étendue des images d'arrière-plan

Fixation de l'image d'arrière-plan

Dégradés de couleurs

 Dégradé linéaire

 Dégradé radial

 Dégradés répétitifs

Multicolonnage

Nombre et largeur des colonnes

Espacement des colonnes

Trait de séparation des colonnes

Équilibrage des colonnes

Titre sur plusieurs colonnes

La flexbox pour répartir des blocs

Un bloc conteneur de type flex

Propriétés flexbox appliquées au conteneur

 Type flexbox pour un élément

 Direction et axe principal des blocs

 Retour à la ligne des blocs

 Raccourci flex-flow pour la direction et les débordements

 Alignement sur l'axe principal

 Alignement sur l'axe perpendiculaire

 Répartition sur l'axe perpendiculaire

Propriétés flexbox appliquées aux blocs « items flex »

 Alignement spécifique d'un bloc

 Agrandissement automatique des blocs

 Réduction automatique des blocs

 Dimensions de base avant agrandissement ou réduction des blocs

 Raccourci flex à privilégier

 Ordre des blocs

 Deux utilisations pratiques de la flexbox

Transformations géométriques

Propriété de transformation

Fonctions de transformation 2D

Fonctions de transformation 3D

Le Web s'anime en CSS 3

Les transitions

Les animations

Nouveaux types de sélecteurs

Sélecteur de voisinage

Sélection sur les attributs

 Attribut existant ou ayant une valeur donnée

- Attribut sélectionné sur une partie de son contenu
- Attribut commençant par...
- Attribut se terminant par...
- Attribut contenant...

Pseudo-classes

- Élément ou attribut différent de...
- Pseudo-classes pour les éléments de formulaire
- Distinction des éléments inclus dans un bloc

Pseudo-élément

Les CSS 3 : déjà utilisées et toujours en évolution

8. EXEMPLES DE SITES WEB

Structure d'une page web

Code HTML de base

Créer des pages de base à menu horizontal ou vertical

- Page de base avec menu horizontal
- Page de base avec menu vertical

Exemples concrets avec images de fond et dégradés CSS 3

- Une grande image pour toute la page
- Site sur deux colonnes, avec image de fond et dégradés CSS

9. UN SITE WEB POUR LES MOBILES

Les contraintes du Web mobile

Adaptation de la mise en page

- Le sélecteur @media

Responsive et media queries CSS 3

- Un exemple pour commencer
- Syntaxe des media queries
- Application aux navigateurs mobiles

Adaptation pratique d'un site pour le Web mobile

- Ajouter une version pour mobile à un site existant
- Adaptation d'une page aux mobiles

10. DIFFÉRENTS TYPES DE MÉDIAS

Types de médias

Média paginé : styles CSS 2 et CSS 3 pour l'impression

- Gestion des veuves
- Gestion des orphelines

- Saut de page avant
- Saut de page après
- Coupure par un saut de page
- Dimensions d'une page
- Sélecteur de page
- Référence à un type de page
- Rotation d'image

Média sonore : fonctions audio CSS 3

A. COULEURS

Choix et harmonisation des couleurs

Les 16 couleurs de base

Liste de toutes les couleurs nommées

B. COMPORTEMENT DES PRINCIPAUX NAVIGATEURS

Test des pages sur plusieurs navigateurs

Interprétation du HTML et des propriétés CSS

Normes et navigateurs

- Comportement des navigateurs courants

- Tester un navigateur

C. RÉSUMÉ DES PROPRIÉTÉS CSS

Propriétés communes aux normes CSS 2 et CSS 3

- Propriétés d'affichage

Principales propriétés spécifiques aux CSS 3

Styles pour les médias paginés et sonores

- Média paginé

- Média sonore

Propriétés classées par catégories

- Propriétés communes aux CSS 2 et CSS 3

- Propriétés spécifiques aux CSS 3

- Médias paginés et sonores

D. RÉFÉRENCES BIBLIOGRAPHIQUES ET SITES WEB

Bibliographie

Sites web utiles

INDEX

chapitre 1

Introduction au HTML et aux feuilles de styles CSS



The screenshot shows the homepage of the Jardin Zen CSS website. The header features a background image of a pond with lily pads and a red torii gate. On the left, there's a vertical column with Japanese-style text and a purple lotus flower. The main title "Jardin Zen CSS" is prominently displayed. Below the title, there are several sections of text in French, including a demonstration of CSS selectors, a call to action to choose a design, and a sidebar listing various CSS designs by different authors.

Une démonstration de ce qu'on peut accomplir lorsqu'on utilise les CSS pour la conception web.

Sélectionnez n'importe quelle feuille de style listée pour charger le résultat sur cette page.

Téléchargez les fichiers d'exemple [html](#) et [css](#)

Le chemin vers l'édification

Les reliques passées des sélecteurs spécifiques aux navigateurs, des DOMs incompatibles, et du manque de support des CSS encombrent un long chemin sombre et môme.

Aujourd'hui, nous devons nous clarifier l'esprit et nous débarasser des pratiques passées. La révélation de la véritable nature du Web est maintenant possible, grâce aux efforts infatigables des gens du W3C, du W3SP et des créateurs des principaux navigateurs.

Le Jardin Zen CSS vous invite à vous relaxer et à méditer sur les leçons importantes des maîtres. Commencez à voir clairement. Apprenez à utiliser ces techniques (bientôt consacrées par l'usage) de manière neuve et revigorante. Ne faites qu'un avec le Web.

Alors, de quoi s'agit-il?

Il y a clairement un besoin pour les graphistes de prendre les CSS au sérieux. Le Jardin Zen vise à exciter, inspirer, et encourager la participation. Pour commencer, voyez quelques concepts choisis dans la liste. Cliquez sur n'importe lequel pour le charger sur cette page. Le code HTML demeure le même, et seule la feuille de style extérieure change. Oui, vraiment.

Les CSS permettent un contrôle complet et total du style d'un document hypertexte. La seule manière de vraiment démontrer cela d'une manière qui excite les gens est de démontrer ce qui peut vraiment être, une fois que les personnes ont été placées dans les mains de ceux capables de créer la beauté basée sur la forme. Jusqu'à maintenant, les exemples de trouvailles et montages intéressants ont été fournis par des programmeurs et des structuralistes. Les concepteurs ont encore à faire leurs preuves. Cela doit changer.

La beauté de la conception CSS

choisissez une conception:

- Dazzling Beauty by Deny Sri Supriyono
- Dark Rose by Rose Fu
- Lesgo My Egg by Jon Tan
- LuCaZee by Viallon Pierre-Antoine
- The Diary by Alexander Shabuniewicz
- Lonely Flower by Nitja Rubic
- Mozart by Andrew Brundle
- Organica Creativa by Eduardo Cesario

archives

Conceptions suivantes »

Voir toutes les conceptions

Que signifient HTML et CSS ? Quels avantages les feuilles de styles nous apportent-elles ? Comment se partagent-elles le « travail » de mise en page avec le code HTML ?

SOMMAIRE

- ▶ **Signification de HTML et CSS**
- ▶ **Principes de base pour une page web**
- ▶ **Principes d'une bonne écriture HTML/CSS**
- ▶ **Intérêt des feuilles de styles**

Cette introduction nous emmène à la découverte de quelques notions fondamentales à propos du HTML et des feuilles de styles CSS. C'est aussi l'occasion, à partir d'exemples, de poser les principes d'une bonne écriture des pages web.

Signification de HTML et CSS

Le sigle HTML correspond aux initiales d'*HyperText Markup Language*, c'est-à-dire « langage de marquage hypertexte ».

Cela signifie que la mise en place d'une page web (titres, paragraphes, images...) utilisera des caractères pour *marquer* d'une certaine façon les différentes parties du texte : <p> représentera un paragraphe, <audio> un fichier son, etc.

Parmi ces caractères de marquage, certains correspondront à des liens vers d'autres pages web : ce sont des liens *hypertextes*, représentés par la balise <a>, pour *anchor* en anglais, c'est-à-dire un lien vers une ancre qui est attachée à un endroit donné, en général une page web.

Le HTML 4 a évolué vers l'appellation XHTML 1, avant de revenir à son nom initial HTML pour la version 5. Le « X » de XHTML vient de XML, soit *eXtensible Markup Language*, langage plus complexe et plus strict que le HTML. C'est lui qui a inspiré la transition du HTML vers la forme plus rigoureuse qu'est le XHTML. Son successeur, le HTML 5, accepte à nouveau une syntaxe plus souple, tout en conservant les bases et les acquis de la version XHTML 1.

Quant à CSS, cela signifie *Cascading Style Sheets*, ce qui se traduit en français par *feuilles de styles en cascade*. À la version courante CSS 2 vient s'ajouter la dernière norme CSS 3, qui reprend l'existant et ajoute de nouvelles propriétés très intéressantes.

La feuille de styles fournit la mise en forme des éléments de la page, qui auront été écrits en HTML. Elle s'applique à une ou plusieurs page(s) du site.

L'expression « en cascade » indique que la mise en forme d'une page peut faire appel à plusieurs feuilles de styles. Les différentes propriétés affectées à un même élément s'ajoutent alors pour lui donner sa mise en forme finale. Il arrive parfois qu'une propriété en contredise une autre qui aura été définie auparavant : dans ce cas, des règles de priorité s'appliquent, et c'est généralement le dernier style défini qui est pris en compte.

Principes de base pour une page web

Voici les principales qualités demandées à une page web : qu'elle soit claire dans sa conception, accessible à tous et que son esthétique s'accorde bien à son contenu.

Choix sensé des balises HTML

En HTML, chaque élément doit être porteur de sens. Par exemple :

- pour un titre de page, utiliser un titre de niveau 1 `<h1>` plutôt qu'un paragraphe quelconque `<p>` ;
- pour un menu (liste de liens), choisir une liste sans numérotation ``.

L'utilisation de balises qui donnent du sens présente plusieurs intérêts :

- le code sera plus clair pour le développeur et la maintenance future du site en sera facilitée ;
- les moteurs de recherche indexeront mieux les pages, car ils y retrouveront plus facilement les mots-clés essentiels ;
- l'accessibilité sera améliorée pour les personnes en situation de handicap visuel.

Adaptation aux navigateurs

Il s'agit de couvrir, autant que possible, une large gamme de navigateurs :

- différents logiciels du marché ;
- divers systèmes d'exploitation ;
- d'autres médias que le PC, comme les téléphones mobiles de type smartphone ou les tablettes numériques...

De plus, les pages web doivent rester lisibles lorsque la feuille de styles n'est pas prise en compte :

- lecture en mode texte ;
- lecture vocale ou en braille ;
- anciens navigateurs qui ne reconnaissent pas complètement les styles.



FIGURE 1–1 Nos pages doivent pouvoir s'afficher sur différents types d'écrans (simulation sur <http://ami.responsivedesign.is>) : il s'agit ici du site AbulÉdu (<http://www.abuledu.org>), qui propose des outils numériques et des logiciels libres éducatifs pour les enseignants.

Accessibilité

L'accès aux personnes handicapées (que le handicap soit visuel, auditif, moteur) doit être facilité :

- proposez une navigation alternative lorsque sont utilisés des menus graphiques ou reposant sur des scripts (des modules complémentaires, appelés *plug-ins* en anglais, sont nécessaires pour l'affichage d'éléments Flash, Java...) ;
- évitez les structures de page reposant sur des cadres (*frames*) ou des tableaux, réservez les tableaux à la présentation de données alignées ;
- ne vous basez pas uniquement sur les couleurs, permettez également l'augmentation de la taille du texte ;
- proposez une alternative aux contenus purement visuels (images) ou auditifs, facilitez la lecture des liens hypertextes...

L'apparence, fonction du thème et du public concerné

Le choix des couleurs et des polices de caractères est fonction du style à donner aux pages web, donc de leur thème et du public visé.

Police de caractères

- Pour le Web, il est courant d'utiliser des polices sans serif (Arial, Helvetica, Trebuchet, Verdana...).
- Réservez aux titres les autres polices et surtout les polices fantaisie.
- N'abusez pas de l'italique et du gras, à réserver à quelques mots ou remarques.
- Évitez les caractères trop petits pour des paragraphes entiers.
- Limitez à deux ou trois le nombre de polices différentes dans une même page.



FIGURE 1–2 Aérer la présentation, proposer un menu clair et ergonomique, harmoniser les couleurs et en limiter le nombre pour le texte : tous ces points ont été oubliés sur le site <http://www.arngren.net>, qui a acquis une certaine célébrité grâce aux classements des pires pages Internet !

En résumé, quelques sentiments liés aux couleurs

Les différentes couleurs correspondent à des sentiments, des impressions, des atmosphères. La connaissance de ces relations peut nous aider à choisir le graphisme du site à créer, en fonction de son sujet et de la catégorie d'internautes à laquelle il est destiné. Voici les valeurs communément associées aux couleurs les plus courantes.

- Les couleurs chaudes, telles que le jaune, l'orange et le rouge, représentent la chaleur et le dynamisme, ainsi que les impulsions.
- Les couleurs froides, comme le bleu, le vert et le violet, indiquent la fraîcheur, le calme et aussi le raisonnement (sciences).
- Les couleurs vives sont associées à l'action.
- Les couleurs pastel font penser à la poésie et donnent une impression de sensibilité.
- Enfin, le gris et le blanc sont des couleurs passe-partout.

Homogénéité du site

Les différentes pages d'un site doivent présenter un minimum d'homogénéité entre elles. Elles proposeront par exemple des variations autour d'un graphisme commun.

Il est donc important de définir une « charte graphique » (polices, couleurs, logos...) à partir de laquelle les pages seront construites (figure 1-3).

Principes d'une bonne écriture HTML/CSS : donner du sens au codage

L'essentiel est de séparer le contenu (codé en HTML) de la mise en forme (feuilles de styles CSS). Cette méthode présente plusieurs avantages, notamment la clarté du code et la possibilité de définir des styles communs à plusieurs pages.

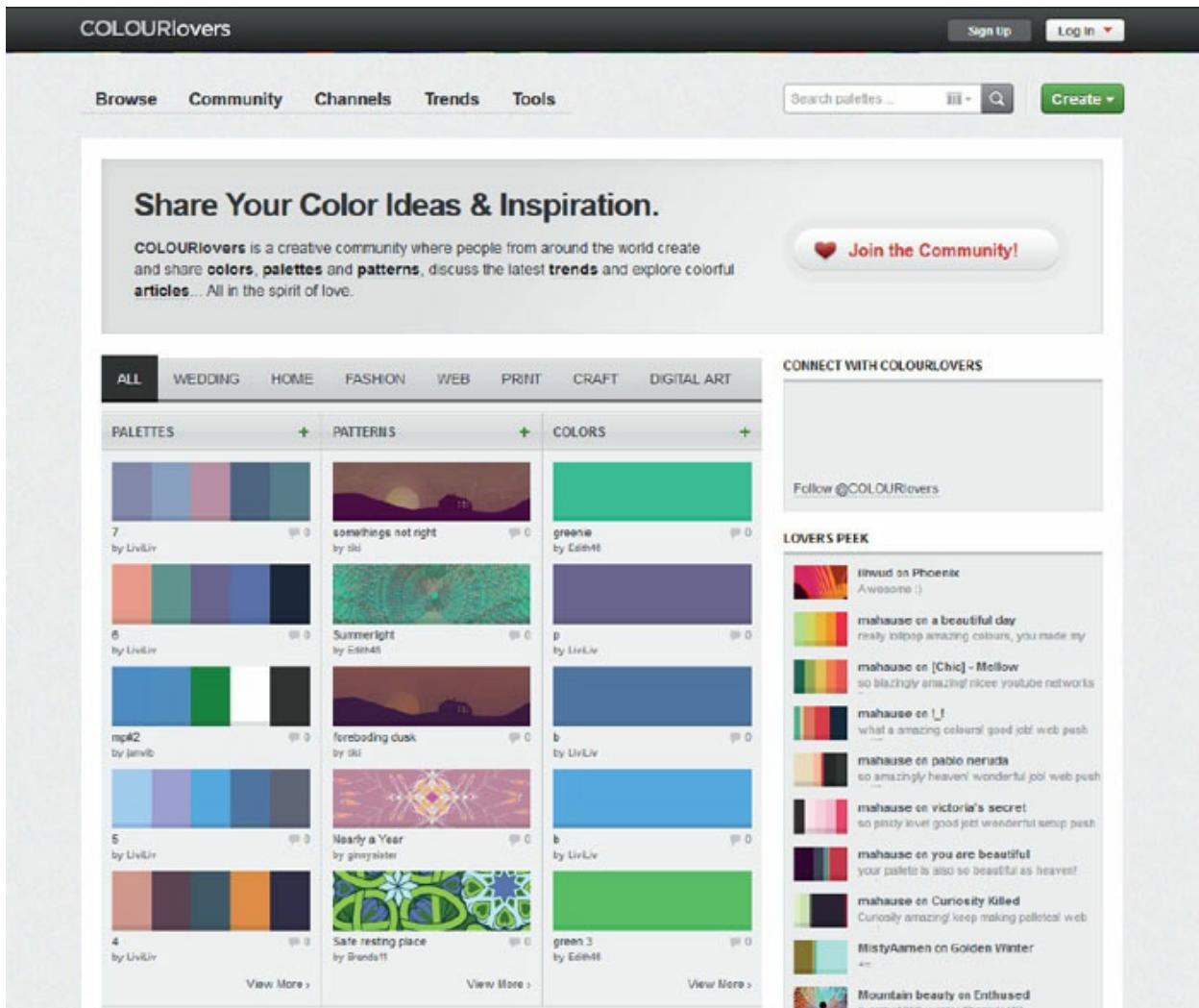


FIGURE 1–3 Bien choisir les couleurs d'une page : extrait du site <http://www.colourlovers.com/>, qui propose un grand nombre de palettes de couleurs assorties et de motifs pour nous aider à réussir un site harmonieux.

Voici quelques exemples de mise en forme qui montrent de quelle façon les

éléments du codage sont liés au sens au texte. Ils utilisent le principe des balises, que nous n'avons pas encore détaillé, car c'est l'objet du chapitre qui suit. Cependant, c'est une première approche intéressante, avant de se jeter dans le grand bain du HTML et des CSS !

Titre de page

Au lieu d'écrire le titre dans un paragraphe normal <p> et de l'affubler d'une tonne d'éléments de mise en forme (grande taille, gras, espaceMENT au-dessus et en dessous), codez-le plutôt comme « titre de niveau 1 » avec la balise <h1>. Si besoin, sa mise en forme pourra être complétée à l'aide d'une règle de style CSS :

- dans le code HTML : <h1>Ici un titre</h1>
- et dans la feuille de styles : h1 { ... mise en forme ... }

Mise en gras ou en italique

Pour l'accentuation de mots, en gras ou en italique, la norme HTML 4 proposait autrefois les balises `` (gras, qui se dit *bold* en anglais) et `<i>` (italique). Tant en XHTML 1 qu'en HTML 5, ces deux mises en relief s'effectuent à l'aide des balises `` et ``.

Ce petit rappel historique n'est pas là par hasard, car il correspond bien à la philosophie du HTML depuis plusieurs années, qui consiste à « donner du sens au codage ».

- Les nouvelles balises `` et `` indiquent une mise en relief plus ou moins prononcée sans dire par quel moyen elle s'effectuera, notamment sans préciser qu'il s'agit du gras ou de l'italique.
- Ainsi par exemple, le concepteur de la page peut effectuer cette accentuation par un changement de couleur du texte, sans recourir ni au gras ni à l'italique. D'où l'intérêt de ne pas indiquer le type de mise en forme dans la balise elle-même.
- Si les balises `` (bold pour gras) et `<i>` (italique) font à nouveau partie de la norme HTML 5, elles ne doivent servir qu'à appliquer une mise en forme, sans aucune notion de mise en relief du texte concerné.

Liste de liens hypertextes (menu)

Pour écrire un menu, évitez d'avoir recours à une succession de paragraphes `<p>`. Préférez-leur une structure de liste en délimitant l'ensemble par la balise `` (liste non numérotée), et chaque ligne par une balise ``. Nous reviendrons bientôt sur l'utilisation de ces balises.

Ainsi, cette partie se différenciera du texte et constituera un ensemble homogène, avec une fonction bien précise : celle d'un menu. En revanche, la suite de la page contient des paragraphes `<p>` incluant aussi des liens hypertextes, ce qui permettra de les identifier en tant que texte sur la page.

À NOTER Menu en début de page

Puisque nous parlons de menu, il faut signaler que sa place de choix dans le code se trouve au début de la page. Celle-ci sera alors mieux comprise par les navigateurs qui lisent la page en mode texte, d'où une accessibilité améliorée pour les personnes handicapées et un meilleur référencement par les moteurs de recherche.

The screenshot shows the homepage of the FramaSoft website. At the top left is the FramaSoft logo. To its right is a tagline: "La route est longue mais la voie est libre...". On the far right are icons for social media sharing. The main content area contains several paragraphs of text about the organization's mission and values. To the right of the text is a sidebar titled "Libres logiciels" which lists links to various resources like Framalibre, Framakey, etc. Below that is another sidebar titled "Libres cultures" with links to Framablog, Framabook, etc. At the bottom left is a search bar with a magnifying glass icon and the word "Rechercher".

FIGURE 1–4 Cet extrait de page contient en haut un titre de niveau 1 (nom du site), à droite une liste de liens (menu général) et sur la gauche un paragraphe de texte. Il provient de la page d'accueil du site <https://framasoft.org>, portail de la communauté francophone des logiciels et solutions libres.

Intérêt des feuilles de styles

L'utilisation des feuilles de styles n'a pas pour seul but de répondre aux normes et de faire plaisir au W3C (consortium qui définit les règles de codage des pages web : <https://www.w3.org>). Un bénéfice réel et concret découle de cette façon de travailler.

La dissociation du contenu (HTML) et de la mise en forme (feuille de styles) permet :

- de retrouver et corriger plus facilement le texte des pages ;
- d'utiliser une feuille de styles externe, commune aux différentes pages d'un site. Il en résulte une meilleure unité graphique entre ces pages et aussi des mises à jour plus simples par la suite. Une modification dans la feuille de styles externe se répercute d'un seul coup sur toutes les pages du site.

La mise en page est beaucoup plus légère, car elle ne nécessite plus, comme dans un lointain passé, l'utilisation de tableaux. Les CSS permettent en effet de positionner les différentes parties d'une page web :

- soit de façon rigoureuse : blocs fixes dont les coordonnées sont choisies ;
- soit d'une manière souple : blocs « flottants » ou placés en ligne, qui se juxtaposent ;
- soit encore d'une manière adaptative : les blocs s'alignent au choix en ligne ou en colonne, leurs dimensions s'adaptent automatiquement en fonction de l'espace disponible.

Nous aurons l'occasion, au cours des chapitres sur les feuilles de styles, d'examiner de façon plus précise toutes les manières d'aligner ou de positionner les éléments dans une page.

Cette présentation était l'occasion de connaître quelques principes de base auxquels nous ferons référence plus tard. Le chapitre qui suit va nous donner des outils concrets pour écrire une page HTML et nous préparer à la mise en place d'une feuille de styles.

The screenshot shows the homepage of the Léa-Linux website. At the top left is a cartoon penguin icon. To its right, the site's name "Léa-Linux" is displayed in a large, stylized font where the letters are partially blue and red. A horizontal menu bar above the main content area includes links for "Nouvelles", "Documentation", "Forums", and "Mailing-List". Below the menu, there are four main sections: "Nouvelles" (with a note icon), "Documentation" (with a document icon), "Forums" (with a green checkmark and orange exclamation mark icon), and "Mailing-List" (with a colorful sun-like character icon). The central content area features a news article titled "LINUX: UBUNTU ABANDONNE UNITY ET LES TÉLÉPHONES" posted by "Léa" on April 6, 2017. The article discusses Canonical's decision to abandon Unity in favor of GNOME Shell. It also mentions the development of Wayland and Mir. On the right side of the page is a login form titled "Identification!" with fields for "Identifiant" and "Mot de passe", and buttons for "Connexion!", "Enregistrez-vous", and "Réinitialiser".

Léa-Linux à voir : LinuxFF CCU-Squad Zarb.Org GNU

Léa-Linux

Nouvelles
L'actualité du monde des Logiciels Libres

Documentations
Découvrez toutes la documentation Léa-Linux, ainsi que les fiches

Forums
Besoin d'aide ? Trouvez des informations sur notre forum

Mailing-List
La communauté d'aide.
Participez et posez vos questions

LINUX: UBUNTU ABANDONNE UNITY ET LES TÉLÉPHONES

posté par Léa, le 6 avril 2017

Par la voix de son président, Canonical, la société éditrice de la distribution grand public Ubuntu, a [annoncé](#) hier qu'Unity serait abandonné.

Unity est une interface graphique pour l'environnement de bureau [GNOME](#). Ubuntu l'utilise depuis Ubuntu 10.10. La majorité des autres distributions, telles que Fedora, Debian ou Mageia, en revanche, utilise l'interface par défaut de GNOME 3, GNOME Shell. L'abandon d'Unity dans la version 18.04 d'Ubuntu est donc une bonne nouvelle, puisque cela permet à la communauté d'être unie pour construire la meilleure interface possible pour les utilisateurs.

Évidemment, l'abandon d'Unity ne vient pas forcément d'un regret de la dispersion des bonnes volontés. C'est aussi une décision logique, car Unity avait été développée dans le but de fonctionner aussi bien sur les ordinateurs que sur les tablettes et les smartphones. Or, Canonical a annoncé dans le même temps qu'elle désirait abandonner le développement d'un système pour smartphone (un an après Mozilla), pour se concentrer sur les marchés juteux de l'[Internet des objets](#) et du [cloud](#).

Enfin, un corollaire de cette annonce est très probablement l'abandon de Mir, que Canonical développait en concurrence au projet communautaire [Wayland](#).

Identification !

Identifiant

Mot de passe

FIGURE 1–5 Une page web est constituée de blocs contenant du texte et des images (extrait de la page <http://lea-linux.org>, site francophone d'entraide pour les utilisateurs de Linux).

chapitre 2

L'essentiel du HTML



Qu'est-ce qu'une page web et quelle en est la structure ? Pourquoi les éléments qu'elle contient doivent-ils être entourés de balises et comment celles-ci sont-elles imbriquées ? Partons à la découverte du langage HTML 5 ; nous y trouverons les principes de base pour afficher du texte et des images dans nos pages.

SOMMAIRE

- ▶ **Principe des balises**
- ▶ **Évolution de la norme HTML**
- ▶ **Premières règles d'écriture HTML**
- ▶ **Principales balises HTML**
- ▶ **Des blocs pour structurer les pages**
- ▶ **Deux catégories d'éléments**
- ▶ **Hiérarchie des éléments : l'héritage**
- ▶ **Validation du code HTML**

Dans le contenu d'une page web, chaque élément (titre, paragraphe, image, etc.) voit son type défini par une « balise HTML », par exemple <p> pour définir un paragraphe. Après avoir étudié ces balises, nous nous intéresserons à leur imbrication : un élément est toujours inclus à l'intérieur d'un autre, ne serait-ce que dans celui qui délimite l'ensemble de la page. La hiérarchie entre ces éléments est importante à comprendre pour bien utiliser les feuilles de styles par la suite.

Principe des balises

HISTOIRE Naissance du HTML

Le langage HTML fut inventé en 1990 par Tim Berners-Lee lorsqu'il travaillait au CERN de Genève (Centre Européen de Recherche Nucléaire). L'objectif était alors d'afficher des pages d'information ayant les propriétés suivantes :

- Les pages étaient reliées entre elles par des liens hypertextes (ces liens sur lesquels, aujourd'hui, nous n'arrêtons pas de cliquer !).
- Ces documents devaient être lisibles sur tous les ordinateurs, quel que soit leur système d'exploitation : Windows, Unix, macOS, etc.

Tim Berners-Lee a poursuivi ce travail en fondant le W3C ou World Wide Web Consortium. Cet organisme regroupe les principaux acteurs du Web ; il standardise et fait évoluer les différentes normes du monde Internet.

C'est pour constituer un ensemble de pages reliées entre elles, accessibles de n'importe quelle machine sur le réseau, que le HTML fut créé, sur le principe du codage de portions de texte à l'aide de balises. Cette méthode reste bien sûr d'actualité, les pages du Web étant exclusivement codées avec des caractères.

Les différentes parties de texte sont donc délimitées par des « balises », qui donnent des indications de mise en forme ou de structuration, l'une d'elles (`<a>` comme *ancre*) servant à relier les pages entre elles par les fameux liens hypertextes.

Par exemple, la mise en italique du mot « bonjour » s'écrit :

```
<em>bonjour</em>
```

Vous avez remarqué la présence d'une balise de fin, ici ``. Elle reprend le nom de la balise de début, précédé de la barre oblique (/) appelée également *slash*. Voici un autre exemple d'élément qui est ici un titre de niveau 1, avec la lettre « h » comme *header* en anglais :

■ <h1>Premier chapitre</h1>

Les mots ainsi encadrés par la balise <h1> seront automatiquement écrits en caractères gras et de grande taille. Ce titre bénéficiera également d'un espacement vertical, au-dessus et en dessous, ce qui améliorera sa visibilité au milieu du texte.

Évolution de la norme HTML

Succession des normes HTML

Le HTML est devenu vraiment populaire à partir de sa version 4. La version suivante a voulu être une norme plus rigoureuse, pour clarifier le code et simplifier la maintenance des pages web ; elle a été appelée XHTML 1, de façon à bien marquer cette orientation.

Son successeur, le HTML 5, a repris la dénomination initiale, car il est redevenu plus tolérant pour le codage, bien qu'il soit fortement conseillé de conserver les bonnes habitudes introduites par le XHTML.

Ces pratiques éclairées n'ont rien de compliqué et consistent pour une bonne partie à utiliser les feuilles de styles CSS (*Cascading Style Sheets*), de façon à ne pas mélanger le contenu du texte et sa mise en forme. Nous allons étudier tout cela de façon progressive : après avoir posé la base que constitue le code HTML, nous passerons à l'étude des CSS.

NORMES Méli-mélo de numéros pour les normes HTML/XHTML

Entre HTML et XHTML, les numérotations de versions ont été disparates, car au HTML 4 a succédé le XHTML 1, puis le HTML 5. Voici l'explication de ce cheminement sinueux.

- Apparu en 1997, le HTML version 4 a fait place en 2000 au XHTML 1, plus rigoureux dans ses règles d'écriture, la lettre « X » correspondant à un rapprochement avec le langage XML. La version suivante aurait dû être le XHTML 2 et induire une rupture technologique, mais la compatibilité nécessaire avec les sites et navigateurs existants rendait ce cap difficile à maintenir.
- C'est pourquoi le W3C a adopté en 2007 les propositions du WHATWG (*Web Hypertext Application Technology Working Group*) pour la création de la norme HTML 5. L'appellation HTML est reprise, car cette version 5 réintroduit quelques éléments et tolérances dont ont besoin les éditeurs *Wysiwyg* (*What you see is what you get*). Ces logiciels mélangent le contenu et la mise en

forme, celle-ci s'effectuant à l'aide de la souris et sans écrire de code.

- Il existe également une version XHTML 5 dont nous n'aurons pas l'utilité dans cet ouvrage : hormis deux lignes d'en-tête, elle partage les mêmes balises que le HTML5, mais donne accès à des données au format XML (*eXtensible Markup Language*) utilisant des balises personnalisables. Quant à la version XHTML 2, son développement a officiellement été abandonné en 2009.

La philosophie du HTML 5

Évolution et compatibilité

Au fur et à mesure de l'évolution des normes, un des objectifs essentiels du W3C a consisté à donner davantage de sens au codage. On a notamment l'introduction en HTML 5 de balises spécialisées, ce qui permet de bien repérer les fonctions des différentes parties qui composent une page.

Plusieurs bénéfices en découlent :

- une meilleure accessibilité pour les personnes handicapées, en particulier pour celles qui lisent les sites en mode texte ;
- une vue plus claire du code, qui permet au développeur de s'y retrouver plus facilement pour la mise à jour des sites web ;
- la possibilité d'affiner le référencement des pages, pour les moteurs de recherche qui exploitent les informations données par des balises qui ont un sens bien précis.

Pragmatisme et tolérance

Une autre orientation du HTML 5 concerne la compatibilité et la tolérance avec les versions précédentes du HTML. Ainsi, par exemple, il sera possible d'écrire les balises en minuscules ou en majuscules.

Enfin, la philosophie du HTML 5 est pleine de pragmatisme : si la norme fait évoluer le codage, ajoutant de nouveaux éléments et en délaissant d'autres, son principe est que les navigateurs doivent toujours prendre en compte l'existant, c'est-à-dire toutes les formes de codage qui auront été utilisées dans le passé. C'est la garantie, pour nos pages Internet, de conserver une mise en forme immuable dans le temps, quelles que soient les évolutions futures.

Premières règles d'écriture HTML

Familiarisons-nous avec les principes de base qui valent pour l'écriture d'un fichier HTML : son nom d'abord, puis la façon d'écrire son contenu.

Règles pour les noms des fichiers

Chaque page web est un fichier dont le nom peut comprendre des lettres, des chiffres et des tirets. À éviter : les espaces, les caractères accentués et le « ç ». Son extension est généralement « .html », bien que la variante plus rare .htm soit acceptée également, car autrefois certains systèmes limitaient les extensions à trois caractères.

La première page du site et de chacun de ses sous-dossiers doit être nommée `index.html` (ou `index.htm`). En effet, c'est cette page de nom `index` qui s'affichera par défaut si l'internaute tape l'adresse de votre site sans préciser le nom du fichier, comme dans `www.votresite.com`. S'il n'existe pas de page nommée `index`, l'internaute ne verra alors qu'une liste de fichiers et de sous-dossiers, façon explorateur de fichiers..., comme le montre la figure 2-1.

REMARQUE Arborescence du site

Lorsque le site comprend des sous-niveaux, notez qu'il faut saisir dans la barre d'adresse le séparateur `/`, habituel également dans le monde Unix/Linux, tandis que l'explorateur de fichiers sous Windows utilise la barre oblique inverse ou *antislash* `\`...

Index of /tests

Name	Last modified	Size	Description
 Parent Directory		-	
 accueil.html	06-Jun-2018 11:51	3.3K	
 contact.html	06-Jun-2018 11:52	1.6K	
 images/	06-Jun-2018 11:34	-	
 presentation.html	06-Jun-2018 11:52	4.8K	

FIGURE 2-1 Type d'écran obtenu lorsqu'une adresse Internet correspond à un dossier ne contenant pas de page `index.html`. Si la page d'accueil s'appelait ici `index.html` au lieu de `accueil.html`, elle s'affichera directement.

Règles d'écriture des balises en HTML

Depuis la transition par la norme XHTML, entre les versions 4 et 5 du HTML, une certaine rigueur d'écriture a été adoptée, pour davantage de logique et de clarté dans le codage de nos pages web. Ces règles sont simples à énoncer :

- les balises s'écrivent toujours en minuscules ;
- chaque balise doit être refermée.

Le HTML 5 est redevenu plus tolérant, considérant par exemple que le début d'un nouveau paragraphe entraîne automatiquement la fermeture du précédent, même si celle-ci n'a pas été précisée. Cependant, cette notation rigoureuse de la norme précédente est vivement conseillée : non seulement cette écriture reste valide en HTML 5, mais elle ne peut que faciliter la lecture du code et éviter des erreurs.

NORMES Fermeture facultative pour les balises simples

Dans la norme précédente XHTML, toutes les balises devaient être fermées, y compris celles qui, n'entourant pas de contenu, ne s'écrivent pas par paires. Leur barre de fermeture était alors intégrée dans la balise elle-même :

- saut de ligne : `
` au lieu de `
`
- tracé d'une ligne horizontale : `<hr />` au lieu de `<hr>`
- image : `` au lieu de ``

Cette barre de fermeture interne est facultative en HTML 5, qui accepte donc les deux formes d'écriture. Elle tend d'ailleurs à disparaître, car son absence ne compromet pas la clarté du code.

Quant aux attributs des balises, ils sont à écrire entre guillemets simples ou doubles. Par exemple, la balise qui affiche l'image fournie par le fichier `logo.gif` s'écrit :

```

```

Par ailleurs, lorsqu'il y a imbrication de balises dans le code, leur ordre de fermeture doit être l'inverse de celui d'ouverture :

| <p>......</p>

Dans l'exemple précédent, si la balise de fermeture `` était placée après celle du paragraphe `</p>`, il n'y aurait pas d'imbrication réelle des balises `...` dans les balises `<p>...</p>` et le code ne serait pas correct par conséquent.

Structure d'une page HTML

Une page HTML s'écrit de la façon suivante :

- sur la première ligne, la balise `<!DOCTYPE ...>` indique que ce document est de type HTML ;
- le reste de la page est encadré par des balises `<html>` et `</html>` qui signifient *début* et *fin* de HTML ;
- entre ces deux balises se trouvent deux parties : l'en-tête de la page entre `<head>` et `</head>` et le contenu (le corps) de la page entre `<body>` et `</body>`.

Structure générale d'une page HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title> Titre affiché dans la barre du navigateur
    </title>
    ...
  </head>
  <body>
    ... Contenu de la page ...
  </body>
</html>
```

Notez que la balise d'introduction `DOCTYPE` est la seule à s'écrire en majuscules, toutes les autres sont en minuscules.

Délimité par les balises `<head>` et `</head>`, l'*en-tête* donne des informations qui ne seront pas visibles dans la page web, sauf la balise `<title>` qui fournit le titre de la page affiché dans la barre de titre, tout en haut de la fenêtre du navigateur. Les autres balises de l'en-tête indiquent la langue et le codage utilisés, les styles (feuilles de styles CSS), etc. Nous les détaillerons plus loin également.

Tout le contenu visible dans le navigateur, le texte comme les liens vers les images, se trouve dans le *corps de la page*, entre les balises `<body>` et `</body>`.

ATTENTION Ne pas confondre titre et nom du fichier

Le contenu de cette balise `<title>` n'a rien à voir avec le nom du

fichier, ce dernier étant défini lors de l'enregistrement de la page. En revanche, ce sera le nom proposé par défaut à l'internaute qui voudra mémoriser l'adresse de cette page parmi ses marque-pages ou favoris. Il sera également pris en compte par les moteurs de recherche ; c'est donc un des endroits où il est utile de placer les mots-clés qui participeront au référencement du site.

Espaces, sauts de ligne et commentaires invisibles

Dans votre fichier d'édition, vous pouvez sauter des lignes et aérer le code HTML à votre guise : quel que soit le nombre d'espaces ou de sauts de lignes se succédant entre deux mots, le navigateur n'affichera qu'un seul espace.

Le texte suivant écrit dans le code HTML :

```
<p>Voici           un  
                   exemple.</p>
```



s'affichera de cette façon :

```
Voici un exemple.
```

FIGURE 2–2 Les suites d'espaces et les sauts de ligne tapés dans le code HTML sont ignorés lors de l'affichage.

Retour à la ligne

Pour créer un saut de ligne qui sera effectivement affiché, il faut utiliser la balise `
` (comme *break* en anglais).

Nous verrons plus tard qu'au lieu de multiplier les sauts de lignes avec des balises `
`, il sera plus clair de définir des marges dans la feuille de styles.

Espace insécable

Pour forcer l'affichage de plusieurs espaces successifs sur une ligne, il faut utiliser le caractère spécial `&nbsp` pour *non breakable space*, ou *espace insécable* en français.

L'espace insécable `&nbsp` a pour propriété d'empêcher la coupure en fin de ligne, s'il est placé entre deux mots ou caractères. Exemple courant :

```
<p> Nous sommes ouverts tous les jours de 9&ampnbspheures à 18&ampnbspheures</p>
```

Les expressions « 9 heures » et « 18 heures » ne doivent pas être coupées par un retour à la ligne entre le nombre (9 ou 18) et le mot « heures ». Il en est de

même lorsqu'un prix est suivi du symbole « € ». L'espace insécable est indispensable dans ces cas de figure.

SYNTAXE Caractères spéciaux

Attention, tous les caractères spéciaux (dont l'espace insécable) commencent par l'esperluette (&) et se terminent par un point-virgule, qu'il ne faut pas oublier.

Commentaires pour le code HTML

Par ailleurs, il est toujours utile de placer des *commentaires* explicatifs dans le code HTML, pour s'y retrouver plus tard. Ils ne seront pas affichés par le navigateur, mais ils constituent une aide pour celui qui écrit ou lit le code source de la page. Ils peuvent être écrits sur une ou plusieurs ligne(s) et sont délimités par les balises <!-- et -->.

Exemples de commentaires

```
<!-- Ici un commentaire pour le créateur de la page -->  
<!--  
Et là un autre exemple de commentaire,  
sur plusieurs lignes cette fois,  
toujours à l'attention du programmeur.  
-->
```

Principales balises HTML

L'expérience montre que dans l'utilisation d'un logiciel de traitement de texte, par exemple, ce sont toujours les mêmes fonctions de base qui sont utilisées et qui permettent de satisfaire la majorité des besoins.

Il en sera de même en HTML : la connaissance de quelques balises simples nous suffira pour une première approche et dans la plupart des utilisations courantes.

Un exemple pour commencer

Avant de détailler ces balises essentielles utilisées en HTML, voici un exemple concret et illustré qui nous permettra d'en découvrir quelquesunes en avant-première.

Exemple de page codée en HTML

```
<!DOCTYPE html> 1
<html> 2

<head>
  <meta charset="utf-8"> 3
  <title>Blog de Vincent THYME</title> 4
</head>
<body>
  <h1>Bienvenue chez Vincent THYME</h1> 5

  <h2>Mon blog à quatre sous</h2> 6
  <p> 7
    Voici quelques petites lignes à l'attention de
    mes visiteurs. Je voudrais partager avec vous
    <em>mes passions, mes idées, mes projets</em>...
  </p>

  <h2>Mes activités préférées</h2> 8
  <ul> 9
    <li>Le surf</li>
    <li>La plongée sous-marine</li>
    <li>L'informatique</li>
  </ul>
</body>

</html>
```

La figure 2-3 montre le résultat de cette page intitulée *Blog de Vincent Thyme*. Nous allons étudier et commenter les balises qui la composent.

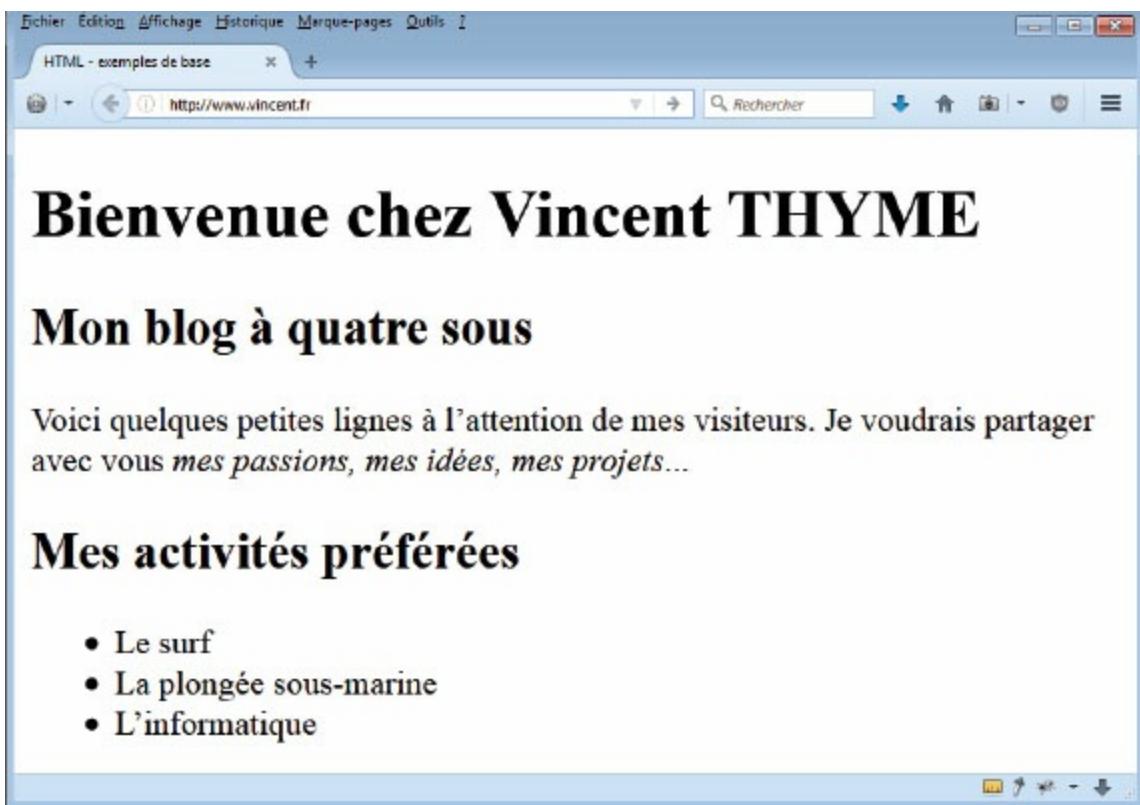


FIGURE 2–3 Le blog de Vincent : une première page toute simple en HTML

Les deux premières balises

- ① La balise de début `<!DOCTYPE html>` nous indique que cette page est codée en HTML. Cette ligne doit être la toute première de notre fichier.
- ② La deuxième balise `<html>` indique le début du code HTML. Il est possible d'y spécifier la langue utilisée, en écrivant par exemple :

```
<html lang="fr">.
```

La toute dernière ligne de notre fichier sera la balise de fermeture associée, qui s'écrit simplement `</html>`.

PRÉCISION Pas de numéro de version dans la balise DOCTYPE

Depuis la version 5 du HTML, la balise `DOCTYPE` n'indique aucun numéro de version. Cela correspond à la philosophie des initiateurs du HTML 5 : l'objectif est de toujours prendre en compte l'existant, les navigateurs devant reconnaître toutes les façons de coder les pages web, les anciennes comme les nouvelles, ce qui est effectivement le cas dans la pratique.

En-tête

③ La première ligne de l'en-tête est nécessairement la balise `<meta ...>` qui indique le *type de codage utilisé*, ici `utf-8`. Si le fichier HTML est bien enregistré suivant ce codage `utf-8`, vous pourrez taper tels quels les accents, le « ç » et le symbole « € », sans avoir à les coder avec des *entités HTML*, comme `ç` ou `€`. Pour choisir le codage de vos fichiers, il vous suffit de rechercher l'option correspondante dans le logiciel avec lequel vous les créez : menu **Format** dans PsPad, menu **Encodage** dans Notepad++, par exemple.

NORMES Principaux types d'encodage

Les types d'encodage les plus connus sont les suivants :

- `utf-8 (charset="utf-8")` : codage universel Unicode sur un ou deux octet(s), à privilégier étant donné que c'est de loin la solution la plus utilisée et la plus évolutive ; elle permet l'écriture des caractères de toutes les langues ;
- `iso-8859-1` ou `Latin-1 (charset="iso-8859-1")` : codage occidental classique, souvent utilisé dans le passé ;
- `iso-8859-15` ou `Latin-9 (charset="iso-8859-15")` : codage occidental `iso-8859-1` plus quelques caractères, dont €, ©, œ et Ù ;
- `Windows-1252` ou `ANSI (charset="windows-1252")` : codage provenant du `iso-8859-1`, comprenant également €, ©, œ, et Ù.

④ La balise `<title>` permet d'écrire un *titre* qui apparaîtra dans la barre de titre du navigateur, tout en haut de la fenêtre.

ANCIENNES NORMES Premières balises d'une page en XHTML 1

La norme HTML 5 a nettement simplifié l'écriture du début du code. En effet, les quatre premières lignes du code HMTL 5 :

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
```

s'écrivaient, en XHTML Strict, version 1.0 :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

En XHTML, le `DOCTYPE` fournissait des références de version très précises ; la balise `<html>` donne un lien vers la liste des balises qui sont valides dans cette version. Cette comparaison suffit pour comprendre que le passage du XHTML au HTML 5 a été animé par une volonté de simplification : le copier-coller n'est plus indispensable pour écrire le début d'une page !

La liste des `<!DOCTYPE ...>` officiels, pour les différentes versions de HTML ou XHTML, se trouve sur le site du W3C, et ils sont également expliqués sur le site [pompage.net](http://www.pompage.net/traduction/le-doctype-qu-il-vous-faut) :

- <http://www.w3.org/QA/2002/04/valid-dtd-list.html>
- <http://www.pompage.net/traduction/le-doctype-qu-il-vous-faut>

En revanche, la toute dernière ligne du fichier s'est toujours écrite `</html>`, quelle que soit la version de HTML ou de XHTML utilisée.

COMPLÉMENTS Autres balises d'en-tête

Rarement utilisée, la balise `<base>` sert à définir un dossier par défaut pour l'ensemble des liens de la page vers un fichier, une image ou un contenu multimédia. Par exemple, si l'en-tête contient la balise :

```
<base href="http://monsite.com/fichiers/">
```

alors la balise :

```

```

(placée dans le corps de la page) affichera l'image `logo.png` située dans le dossier

```
http://monsite.com/fichiers/images/.
```

D'autres balises `<meta ...>` permettent de fournir des informations complémentaires : mots-clés, résumé de la page, nom de son auteur...

Corps de la page

- ⑤ La balise `<h1>` met en forme le *titre de niveau 1* : grande taille, gras,

espacement important au-dessus et en dessous.

6/8 Les balises `<h2>` correspondent à des *sous-titres* : taille un peu moins grande, un peu moins d'espacement autour.

7 Le *paragraphe* `<p>` permet d'écrire une phrase. Quelques mots sont en *italique* à la fin de celle-ci, grâce à la balise `` imbriquée à l'intérieur des balises `<p> ... </p>`.

9 Enfin, la page se termine par une *liste à puces* indiquée par la balise ``, chaque *ligne* étant repérée par une balise ``, ce qui signifie retour à la ligne et nouvelle puce.

Quittons maintenant ce premier exemple pour présenter en détail les balises HTML les plus utiles.

Paragraphes et titres

Chaque *paragraphe* est encadré par les balises `<p> ... </p>`. Un paragraphe ne contient jamais d'autres paragraphes, mais peut inclure des balises de mise en forme, comme le gras ou l'italique, des liens hypertextes et des images.

Les *titres* sont des balises commençant par **h** comme *header*, c'est-à-dire *entête* en anglais. `<h1>`, `<h2>`, `<h3>`,... correspondent à différents niveaux de titre : `<h1>` pour un titre de niveau 1, `<h2>` pour un sous-titre de niveau 2, etc.

À NOTER Niveaux de titre courants

Il est rare d'utiliser des niveaux de titre au-delà de 3 ou 4. Si ces balises existent jusqu'à `<h6>`, le titre `<h4>` correspond à un texte de taille normale mis en gras, `<h5>` et `<h6>` donnant des caractères plus petits. La figure 2-4 donne un aperçu des tailles associées aux différents niveaux de titre.

Titre de niveau 1

Titre de niveau 2

Titre de niveau 3

Titre de niveau 4

Titre de niveau 5

Titre de niveau 6

Paragraph de texte ordinaire

FIGURE 2-4 Différences de taille pour les niveaux de titre 1 à 6, encadrés par les balises `<h1> ... </h1>` jusqu'à `<h6> ... </h6>`

Mise en forme commune à une partie du texte

Pour *regrouper en un seul bloc* un ensemble de paragraphes, de titres, etc., il suffit de les encadrer au moyen des balises `<div> ... </div>` (*div* comme *division* ou partie d'une page). Le bloc ainsi constitué peut être encadré, mis en forme ou positionné dans la page, à l'aide des styles CSS qui seront abordés dans les chapitres suivants.

Les balises ` ... ` permettent de *regrouper plusieurs mots* d'un paragraphe et de leur donner une mise en forme commune, par exemple pour la taille, la couleur ou la police de caractères. Le mot anglais *span* signifiant *portée*, cela revient à sélectionner une *partie du texte* à mettre en forme, quelques mots ou quelques lettres.

Notez bien la différence entre ces deux balises.

- `<div>` représente un bloc de texte auquel il est possible d'affecter une position définie dans la page ; par défaut, les blocs occupent toute la largeur disponible et se placent les uns sous les autres.
- `` encadre un groupe de mots ou de lettres qui conserve sa place à l'intérieur d'un paragraphe, et qui ne provoque pas de retour à la ligne par conséquent.

L'utilisation de ces blocs servira notamment à définir une mise en forme spécifique à certains éléments de la page. Nous verrons plus loin la possibilité de les distinguer les uns des autres, en leur attribuant des noms, qui seront appelés *classes* ou *identifiants*.

À NOTER Balises génériques

Les balises `<div>` et `` sont appelées *balises génériques*, parce qu'elles n'ont pas de sens en elles-mêmes. Pour respecter l'esprit du HTML, il est préférable de ne pas en abuser et d'utiliser autant que possible des balises qui ont une signification, donnant une indication sur l'importance ou le rôle du contenu :

- `` ou `` sera plus approprié que `` pour une mise en relief du texte, même si ce n'est pas du gras ou de l'italique ;
- `<h1>, <h2>,...` pour un titre et `<p>` pour un paragraphe unique

remplaceront avantageusement un `<div>` s'il peut être évité.

En ce qui concerne les blocs `<div>`, nous verront plus loin qu'ils peuvent être également remplacés par des balises spécifiques, suivant leur fonction dans la page : menu de navigation, colonne latérale, etc.

Principales mises en forme

Italique et gras

L'*italique* s'obtient avec les balises ` ... ` signifiant en anglais *emphasis*, c'est-à-dire *accent* ou *insistance*. Elles n'entraînent pas de retour à la ligne, ce n'est pas un « bloc » de texte.

Pour mettre un terme en **gras**, il suffit de l'encadrer à l'aide de la balise ` ... `, comme *stronger emphasis*, pour encore plus de relief qu'avec la balise ``.

Notez qu'il existe deux autres balises pour ces mises en forme :

- ` ... ` pour mettre des mots ou des caractères en gras ;
- `<i> ... </i>` pour écrire en italique.

Attention cependant : elles ne sont pas interchangeables avec les précédentes, même si elles produisent le même effet. Il s'agit d'une question de sens à donner au texte concerné.

- Les balises `` et `` correspondent à une *mise en relief* des mots encadrés, concrétisée par le gras et l'italique (bien que cette mise en évidence puisse s'effectuer d'une autre manière, comme indiqué en remarque).
- Les balises `` et `<i>` conviennent à une *simple mise en forme typographique* avec le gras et l'italique, sans aucune intention de mettre en avant les mots qu'elles entourent.

À NOTER Modifier la mise en forme par défaut

Les balises `` et `` correspondent par défaut à l'italique et au gras. Mais grâce aux styles CSS, vous pouvez décider de remplacer ces mises en forme. Par exemple :

- l'italique fera place à un texte en vert, police Arial ;
- le gras sera remplacé par du blanc sur fond bleu.

Dans les deux cas, l'idée de mettre le texte en relief est respectée, mais

avec d'autres apparences que le gras et l'italique. Cette modification de la mise en forme par défaut sera d'ailleurs possible pour toutes les balises HTML.

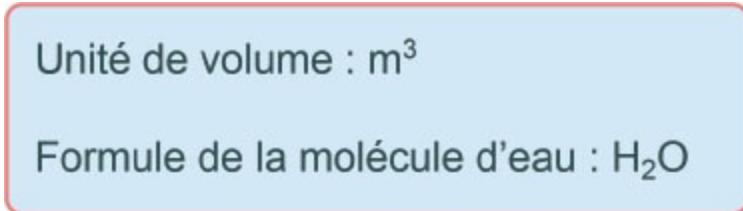
Dans la catégorie des mises en forme de base, notons l'existence de la balise toute simple `<s> ... </s>` qui permet de rayer un texte (voir dans le chapitre 3 la section « Texte barré »).

Exposant et indice

Les balises `^{...}` et `_{...}` permettent respectivement de placer un caractère en *exposant* ou en *indice*, comme dans l'exemple suivant :

```
<p>
    Unité de volume : m<sup>3</sup>
    <br>
    Formule de la molécule d'eau : H<sub>2</sub>O
</p>
```

La taille du texte entre ces balises `sup` et `sub` est diminuée, avec un décalage vers le haut pour `sup`, vers le bas avec `sub`.



Unité de volume : m³

Formule de la molécule d'eau : H₂O

FIGURE 2–5 Nombres en exposant et en indice, à l'aide des balises `<sup>` et `<sub>`

Annotations en petite taille de caractères

Pour afficher un texte en plus petite taille, il est pratique d'utiliser les balises `<small> ... </small>`, qui réduisent la taille du texte pour afficher un commentaire, une annotation ou une référence.

Cependant, l'emploi de cette balise pour une simple mise en forme est à éviter, car le texte qu'elle encadre doit vraiment correspondre à un contenu accessoire et indépendant du contenu principal. Pour la simple réduction de la taille des caractères, nous utiliserons les feuilles de styles.

Voici un exemple de code utilisant la balise `<small>` et dont vous pouvez voir

le résultat sur la figure 2-6.

```
<p>Grande opération de déstockage avant inventaire</p>
<p>Promotions de folie sur tout le magasin !</p>
<p><small>Création CertiPub</small></p>
```

Grande opération de déstockage après inventaire

Promotions de folie sur tout le magasin !

Création CertiPub

FIGURE 2–6 Les trois lignes ont la même mise en forme. La troisième, dont le contenu est accessoire par rapport aux deux autres, est encadrée par des balises `<small>...</small>`.

Notez qu'il existait autrefois la balise `<big>`, pendant de la balise `<small>`, pour écrire un texte en plus grande taille. Elle est à présent obsolète en HTML 5 : il faut la remplacer par un niveau de titre ou une mise en forme dans une feuille de styles.

Citation avec retrait

Les balises `<blockquote> ... </blockquote>` permettent de faire figurer une citation dans une page, avec un effet de mise en page qui consiste en une mise en retrait du texte.

Cependant, si le seul but est d'augmenter la marge de gauche du texte sans qu'il y ait de notion de citation, il est préférable là encore d'effectuer cette mise en forme à l'aide d'une feuille de styles.

Cette balise n'étant pas un conteneur, le texte qu'elle contient doit être placé à l'intérieur d'une balise `<p>` ou d'un bloc `<div>`, comme dans cet exemple, qui correspond à la figure 2-7 :

```
<h4>La devise des Trois Mousquetaires</h4>
<blockquote>
  <p>Un pour tous, tous pour un !</p>
</blockquote>
```

La devise des Trois Mousquetaires

Un pour tous, tous pour un !

FIGURE 2–7 La citation des Trois Mousquetaires, qui est ici insérée dans une balise `<blockquote>`, a été automatiquement mise en retrait. À l'aide d'une feuille de styles, il sera possible de mieux particulariser encore le contenu d'une telle balise.

PRÉCISION Attribut cite pour la balise `<blockquote>`

Il existe un attribut `cite` qui peut être inclus dans la balise

`<blockquote>` pour donner la référence de la citation. Cela peut être par exemple une adresse web :

```
<blockquote cite="https://www.w3.org/TR/html5/">
```

ou encore un numéro ISBN pour un livre :

```
<blockquote cite="ISBN:978-2-212-12724-9">
```

ou bien toute autre référence universelle, appelée URI, pour *Uniform Resource Identifier*.

Trait de séparation horizontal

La balise `<hr>`, comme *horizontal rule* en anglais, trace un trait sur toute la largeur disponible (celle de la page ou de l'élément qui la contient). Elle peut également s'écrire `<hr />`, avec une barre de fermeture intégrée. Plus tard, une feuille de styles nous permettra de personnaliser son épaisseur, sa couleur, sa largeur, etc.

Liens hypertextes

Les balises `<a>...` permettent d'écrire un *lien hypertexte* : le curseur prend la forme d'une main au passage de la souris sur ce lien, et un clic affiche la page qui est référencée dans cette balise. Cette balise est dite *en ligne*, car elle ne produit pas de retour à la ligne.

Exemples simples de liens

```
<a href="http://livre.antevox.fr"> 1  
    Exemples du livre à télécharger  
</a>  
  
<a href="index.html">Retour à la page d'accueil 2  
<a href="documents/liste.html">Documents à télécharger 3
```

L'attribut `href` (*hypertext reference*) est obligatoire, puisqu'il indique l'adresse de la page à afficher lors d'un clic sur ce lien.

Cela peut être *une adresse internet* 1 ou *un nom de fichier seul* 2, lorsque la page à atteindre se trouve située dans le même dossier que la page en cours d'affichage.

Si le fichier est dans un sous-dossier, il faut taper le nom de ce dossier suivi d'un séparateur : / comme sous Unix ou Linux, et non \ comme dans Windows. Le dernier exemple 3 affiche le fichier `liste.html` qui se trouve dans le sous-dossier `documents` du dossier courant, c'est-à-dire du dossier qui contient la page affichée.

Lien avec attributs accesskey et title

```
<a href="http://www.antevox.fr" accesskey="a" title="Site de l'auteur">  
    Visitez le site de l'auteur  
</a>
```

Il est très utile de faire figurer l'attribut `accesskey` : les personnes handicapées pourront activer le lien par un appui simultané sur la touche **Alt** et la lettre indiquée entre guillemets (à condition que ce raccourci ne soit pas déjà utilisé par le navigateur).

L'attribut `title` permet d'afficher un texte automatiquement dans une bulle au passage de la souris sur le lien, comme le montre la figure 2-8.



FIGURE 2–8 Lien hypertexte (par défaut en bleu et souligné, cet élément de mise en forme étant modifiable en CSS) : au passage de la souris, le curseur prend la forme d'une main et le contenu de l'attribut title s'affiche dans une bulle.

Choix de la fenêtre de destination

```
| <a href="http://www.unautresite.com" target="_blank">
```

Associé à la balise de lien hypertexte `<a>`, l'attribut `target` sert à définir la fenêtre dans laquelle s'ouvrira la page sur laquelle pointe le lien. Ici, la valeur `_blank` indique que le lien sera ouvert dans un nouvel onglet.

L'attribut `target` sera également utilisé pour gérer les balises `<iframe>`, que nous examinerons dans le chapitre suivant et qui servent à inclure une page web à l'intérieur d'une autre, la fenêtre principale étant appelée fenêtre parente. Ces balises `<iframe>` sont toutefois déconseillées, car elles compliquent la navigation et entravent l'accès des personnes ayant un handicap visuel.

Les valeurs possibles pour cet attribut `target` sont les suivantes.

- `target="_blank"` ouvre le lien dans une nouvelle fenêtre ou plus souvent dans un nouvel onglet, suivant le paramétrage du navigateur.
- `target="_parent"` affiche la page dans la fenêtre parente, lorsque le lien se trouve dans une fenêtre `iframe`.
- `target="_self"` correspond au comportement habituel d'un lien, la page s'affichant dans la même fenêtre que celle qui contient le lien (c'est la valeur par défaut, utilisée si l'attribut `target` est absent).
- `target="_top"` permet de revenir à la fenêtre parente de plus haut niveau, en cas de fenêtres imbriquées.

L'affichage d'une page à l'intérieur d'une autre étant à éviter, c'est surtout la première valeur `_blank` qui retiendra notre attention.

Lien hypertexte vers un endroit de la page

```
| <a href="#poissons"
```

```
accesskey="p" title="Les habitants de la mer">
    Accès au paragraphe "Les habitants de la mer"
</a>
...
...
<h2 id="poissons">Les habitants de la mer</h2>
```

Pour mettre en place un lien vers un endroit précis de la page courante, il suffit d'ajouter un identifiant en tant qu'attribut de la balise *destination*, par exemple `id="toto"`, ce qui permet de l'atteindre directement grâce au lien `...`.

Si le texte à relier au lien en question n'est pas encadré par des balises, il est possible d'utiliser la balise `<a>` comme ancre simple, uniquement pour attribuer un identifiant à cette partie du texte :

```
<a id="toto">Texte à relier au lien interne</a>
```

À NOTER Dièse # - pas dièse

Le lien `<a href="#toto" ... ` contient un *dièse #* avant le nom de l'identifiant, alors que la balise destination contient un attribut `id="toto"` sans dièse. Bien que cette méthode d'adressage soit très simple, au sujet du dièse ce bémol s'imposait !

Cette méthode permet également de créer un lien vers un endroit précis d'une autre page que celle affichée.

Par exemple, le lien `` affichera la page `oiseaux.html` et placera le curseur de la fenêtre sur la balise d'identifiant `id="rossignol"`.

Par ailleurs, la balise `<a>` permet d'autres types de liens que ceux vers une page web, en particulier des *liens de contact* vers une adresse de messagerie, comme le montre l'exemple ci-dessous.

Lien vers une adresse de messagerie

```
<a href="mailto:contact@antevox.fr"
accesskey="m"
title="Envoi d'un courriel à l'auteur">
    Pour contacter l'auteur...
</a>
```

Dans cet exemple, l'attribut `href` contient `mailto:` suivi de l'adresse électronique vers laquelle sera envoyé le courriel.

Un clic sur ce lien ouvre le *logiciel de messagerie par défaut* sur l'ordinateur du visiteur, et remplit la rubrique *Destinataire* avec l'adresse fournie.

Il est possible d'ajouter un sujet et d'autres destinataires, avec le séparateur `?` après l'adresse e-mail, puis avec le séparateur `&` avant chaque nouveau paramètre.

```
<a href="mailto:toto@laposte.net?subject=Essai de  
message&cc=titi@gmail.com">  
    Envoyer un message  
</a>
```

Dans l'exemple précédent, le *destinataire* du message est `toto@laposte.net`, le *titre* du message est « Essai de message », et il y a un destinataire en *copie* (`cc` comme copie carbone) : `titi@gmail.com`.

ATTENTION Protégez du spam vos adresses de courriel !

Les adresses ainsi inscrites dans une page web peuvent être lues par des logiciels automatiques, qui balaien les pages pour remplir des bases de données d'adresses électroniques à l'usage des *spammeurs* : ceux qui envoient du *spam*, ces messages publicitaires indésirables également appelés *pourriels*.

Il faut donc protéger les adresses :

- soit en codant la balise en *JavaScript*, programme que ces logiciels n'interprètent pas mais qui est normalement pris en compte par les navigateurs ; une recherche sur Internet vous fournira rapidement différentes solutions, comme celle proposée par ce site autour de la messagerie électronique :
 - ▶ https://www.arobase.org/spam/se_proteger/prevenir-webmaster.htm
- soit *au minimum* en modifiant l'adresse, que l'internaute rectifiera à la main dans la fenêtre de sa messagerie, par exemple sous cette forme :

```
<a href="mailto:titi@pasdespamlaposte.net">
```

ou encore celle-ci :

```
<a href="mailto:titi-chez-gmail.com">.
```

Les listes

Différents types de listes sont utilisables en HTML : d'une part, les listes à puces et les listes numérotées telles que celles proposées par les logiciels de traitement de texte, et d'autre part les listes de définitions constituées d'une succession de termes et de définitions associées, comme dans un lexique.

Listes à puces ou numérotées

Une énumération gagne en clarté lorsque chacun des points est repéré par une puce ou un numéro. De telles listes sont délimitées par les balises :

- `...` comme *unordered list* pour une *liste à puces* ;
- `...` comme *ordered list* pour une *liste numérotée*.

Dans les deux cas, *chaque ligne* est repérée par `...` à l'intérieur de ces balises.

Exemples de listes

```
<h3>Liste à puces</h3>
<ul>
  <li>Premièrement</li>
  <li>Deuxièmement</li>
  <li>Troisièmement</li>
</ul>

<h3>Liste numérotée</h3>
<ol>
  <li>Premièrement</li>
  <li>Deuxièmement</li>
  <li>Troisièmement</li>
</ol>
```

L'affichage correspondant à ce code HTML est donné par la figure 2-9.

Liste à puces	Liste numérotée
<ul style="list-style-type: none"> • Premièrement • Deuxièmement • Troisièmement 	<ol style="list-style-type: none"> 1. Premièrement 2. Deuxièmement 3. Troisièmement

FIGURE 2–9 Deux types de listes : liste à puces et liste numérotée

Les types de puces ou de numérotation pourront être personnalisés, grâce à l’emploi des feuilles de styles CSS. En particulier, les puces pourront être remplacées par des images. Les listes numérotées, qui seront elles aussi modifiables en CSS, peuvent également être paramétrées à l’aide de quelques attributs, comme indiqué en remarque.

VARIANTES Personnalisation des listes numérotées

La balise `` crée automatiquement une liste numérotée, attribuant un nouveau numéro à chacune des lignes `` qu’elle contient. Deux attributs complètent cette balise.

- `start="..."` sert à indiquer un numéro de départ pour la liste, sous la forme d’un nombre entier.
- `reversed="reversed"` entraîne une numérotation décroissante.
- `type="..."`, appliqué suivant l’exemple `<ol type="A">`, permet de changer le style de numérotation. Les types possibles sont "1" (1, 2, 3... qui est le type par défaut), "A" (A, B, C...), "a" (a, b, c...), "I" (I, II, III...) et "i" (i, ii, iii...).

Exemples de listes numérotées personnalisées (résultat sur la figure 2–10) :

```

<h3>Nombres en anglais à partir de 10</h3>
<ol start="10">
  <li>Number ten</li>
  <li>Number eleven</li>
  <li>Number twelve</li>
</ol>

<h3>Notre classement</h3>
<ol reversed="reversed">
  <li>Le troisième est Pierre.</li>

```

```

<li>Le deuxième est Paul.</li>
<li>Le premier est Jacques.</li>
</ol>

```

Quant aux balises `` d'une liste numérotée ``, elles acceptent l'attribut `value`, qui permet de spécifier un numéro donné pour une ligne, comme dans l'exemple suivant :

```

<ol>
  <li>...</li>
  ...
  <li value="888">Numéro chinois porte-bonheur</li>
  ...
</ol>

```

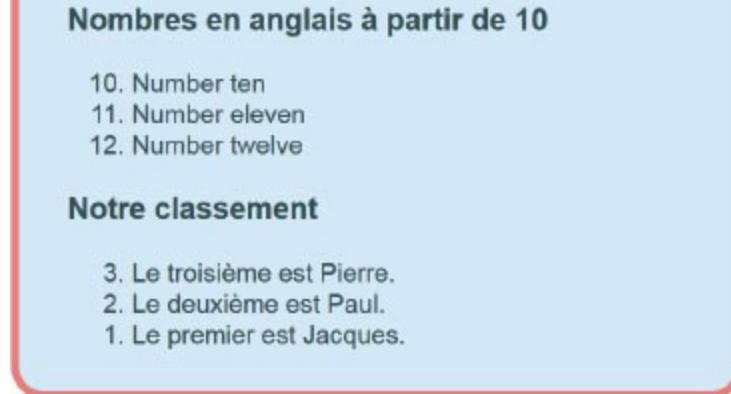


FIGURE 2–10 Exemples de listes numérotées personnalisées : la première avec un numéro de départ différent de 1, la seconde présentée dans l'ordre décroissant

Listes de définitions

Plus rarement employées que les listes à puces, les listes de définitions servent à présenter une suite de termes accompagnés de leur définition ou d'une explication à leur propos.

Le navigateur met en décalage les termes et les définitions, qui utilisent les balises suivantes :

dl	<i>definition list</i> = liste de définitions (encadre la liste)
dt	<i>definition list term</i> = un terme de la liste de définitions
dd	<i>definition list definition</i> = une définition de la liste de définitions

(associée à un terme)

Exemple de liste de définitions

```
<dl>
  <dt>amener</dt>
    <dd>Conduire quelqu'un avec soi.</dd>
  <dt>apporter</dt>
    <dd>Porter quelque chose avec soi.</dd>
</dl>
```

Ces listes de définitions sont également utilisées pour afficher des dialogues, la balise `<dt>` mentionnant le nom du personnage et la balise `<dd>` encadrant sa réplique. Les noms de ces deux balises peuvent alors être compris comme *dialog talker* pour `<dt>` (le personnage qui parle) et *dialog discourse* pour `<dd>` (la réplique de ce personnage).

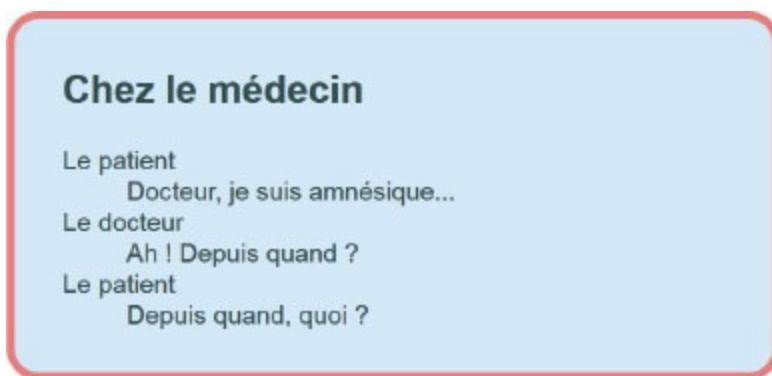


FIGURE 2–11 Exemple d'utilisation des balises `<dl>`, `<dd>` et `<dt>` pour la mise en forme d'un dialogue

Tableaux

En HTML, l'utilisation des tableaux est à réserver à la présentation de données numériques ou d'informations synthétiques. Lorsqu'il s'agira de placer côté à côté les différentes parties d'une page, nous utiliserons des blocs `<div>` en les positionnant à l'aide d'une feuille de styles.

Création d'un tableau HTML

Un tableau est délimité par les balises HTML `<table> ... </table>`.

Initialement, les bordures du tableau sont invisibles ; il faut donc indiquer l'épaisseur de la bordure en pixels à l'aide d'une des deux méthodes suivantes :

- en associant l'attribut `border` à la balise `<table>`, en écrivant par exemple
`<table border="1">...</table>` ;
- en définissant cette bordure de tableau à l'intérieur d'une feuille de styles, comme nous le verrons au chapitre 5.

Deux anciens attributs de la balise `<table>`, `cellspacing` et `cellpadding`, permettaient de préciser respectivement en pixels *l'espacement entre cellules* (traits d'encadrement simples si 0, doubles sinon) et *les marges intérieures* des cellules. Encore acceptés jusqu'au XHTML 1, ils sont tombés en désuétude depuis la norme HTML 5 et remplacés maintenant par des éléments de mise en forme dans la feuille de styles.

Le tableau se construit ligne par ligne, avec les balises `<tr> ... </tr>`, pour *table row* en anglais, c'est-à-dire *rangée du tableau*.

Les cellules du tableau sont définies à l'intérieur de chaque ligne, grâce aux balises `<td> ... </td>` comme *table data*, soit *donnée du tableau*. Lorsqu'il s'agit d'une cellule de titre, il est possible d'utiliser à la place les balises `<th> ... </th>` comme *table header* ou *en-tête du tableau* : le texte est alors mis en gras et centré.

Les balises `<caption> ... </caption>` sont facultatives : placées sous la balise de début de tableau `<table>`, elles permettent de définir un titre associé au tableau, qui s'écrira par défaut au-dessus de celui-ci.

```

<table>
  <tr> <th>...</th> <th>...</th> </tr>
  <tr> <td>...</td> <td>...</td> </tr>
  <tr> <td>...</td> <td>...</td> </tr>
</table>

```

FIGURE 2–12 Structure d'un tableau en HTML

Exemple de tableau

```


||
||
||
||


```

Cet exemple affiche un tableau à trois lignes (trois balises `<tr>`) et deux colonnes (deux balises `<td>` ou `<th>` par ligne). La première ligne ① contient deux cellules d'en-tête, les deux autres lignes ② et ③ contiennent les données du tableau. La figure 2-13 montre le résultat obtenu avec ce code.

Pays	Capitale
France	Paris
Espagne	Madrid

FIGURE 2–13 Premier exemple de tableau : une ligne de titres (balises `<th>`) et deux lignes de données (balises `<td>`)

COMPLÉMENTS En-tête et pied de tableau, mise en forme de colonnes

Une ou plusieurs ligne(s) peuvent être englobées dans un en-tête de tableau, entre les balises `<thead>` et `</thead>`, ou dans un pied de tableau, elles sont alors placées entre `<tfoot>` et `</tfoot>`. Les balises `<thead>` et `<tfoot>` sont à écrire *avant* le corps du tableau, qui devra alors être entouré des balises `<tbody>` et `</tbody>`. Le navigateur affichera l'en-tête au début du tableau et le pied de tableau à la fin, bien que ce dernier soit écrit dans le code avant le corps du tableau. Pour définir une mise en forme spécifique aux différentes colonnes d'un tableau, il suffit de les repérer en utilisant autant de balises `<col ...>` qu'il y a de colonnes, réunies à l'intérieur d'une balise `<colgroup>`. Par exemple, un tableau comprenant deux colonnes aux caractéristiques différentes (largeur, couleur de fond, etc.) peut s'écrire avec la structure suivante :

```
<table>
  <colgroup>
    <col class="colonne1">
    <col class="colonne2">
  </colgroup>
  ... contenu du tableau ...
</table>
```

La mise en forme de chaque colonne (largeur, couleur de fond, etc.) est alors effectuée dans une feuille de styles, à partir des classes associées à chacune des colonnes.

Fusionner des cellules

L'attribut `colspan` permet de fusionner horizontalement les cellules de plusieurs colonnes. Par exemple, la balise `<td colspan="3">...</td>` sera équivalente à trois cellules `<td>...</td>` à l'intérieur d'une ligne.

De même, l'attribut `rowspan` sert à fusionner verticalement les cellules de plusieurs lignes. Une balise `<td rowspan="4">...</td>` sera équivalente à quatre cellules dans le sens vertical : cela signifie que les trois lignes qui suivent auront une cellule `<td>...</td>` en moins.

L'exemple suivant montre le codage d'un tableau qui comprend des cellules fusionnées, horizontalement et verticalement.

```

<table border="1">
<tr> ①
  <th colspan="2">Infos pays</th>
</tr>

<tr>
  <th>Pays</th>
  <th>Langue</th>
</tr>

<tr>
  <td>Andorre</td>
  <td>catalan</td>
</tr>

<tr> ②
  <td rowspan="2">Canada</td>
  <td>anglais</td>
</tr>

<tr> ③
  <td>français</td>
</tr>
</table>

```

Comme le montre la figure 2-14, la première ligne ① de ce tableau contient une *fusion horizontale* de cellules, *sur deux colonnes* : une seule balise `<th>` au lieu de deux sur cette ligne.

L'avant-dernière ligne ② contient une *fusion verticale* de cellules, *sur deux lignes* : la ligne suivante ③ ne contient donc qu'une seule balise `<td>` au lieu de deux.

The diagram illustrates a table structure with the following data:

Infos pays	
Pays	Langue
Andorre	catalan
Canada	anglais
	français

The first row has a colspan="2" attribute, indicated by a red circle labeled ①. The second row has a rowspan="2" attribute on the first cell, indicated by a red circle labeled ②. The third row contains two empty cells, indicated by a red circle labeled ③.

FIGURE 2–14 Deuxième exemple de tableau : il comprend des cellules fusionnées, horizontalement et verticalement.

NORMES Éviter les tableaux pour la mise en page

Rappelons qu'en HTML, les tableaux servent uniquement à présenter

des données. En effet, ils sont tout à fait déconseillés lorsqu'il s'agit de placer des contenus côte à côté : nous étudierons plus loin comment positionner des blocs de texte dans une page, en utilisant les propriétés CSS adéquates. Les mises en page effectuées à l'aide de tableaux sont complexes, rigides, difficiles à mettre à jour et empêchent un référencement correct par les moteurs de recherche.

Insertion d'images

La création d'une page web contenant texte et images nécessite plusieurs fichiers : le fichier HTML contient le texte et chacune des images est enregistrée dans un fichier séparé.

Pour ne pas alourdir les pages, il est important de n'utiliser que des formats d'image compressés :

- JPEG (16 millions de couleurs, extension de fichier .jpg) pour les photos ou les dessins dotés de nombreuses nuances de couleurs, avec un taux de compression de 70 à 80 % pour ne pas trop perdre en qualité (90 % pour les images de toute petite taille) ;
- GIF (256 couleurs) pour les dessins au trait sans dégradés de couleurs, les images avec une couleur de fond transparente, les images animées (un seul fichier contenant une succession d'images dans le temps) ;
- PNG (16 millions de couleurs) pour les photos ou images munies de nombreuses nuances de couleurs, permettant plusieurs niveaux de transparence (transparence progressive utilisée pour créer des fonds translucides, par exemple).

Il faut éviter les images non compressées, telles que celles d'extension .bmp, car leur taille peut être multipliée par dix par rapport aux formats précédents.

La balise image

En HTML, insérer une image revient à placer un *lien* vers le fichier qui la contient, avec la balise ``. Elle contient obligatoirement l'attribut `src`, qui indique le nom du fichier image à afficher.

Exemple de balise image

```

```

Si l'attribut `src` ne contient qu'un nom de fichier, cela signifie que ce fichier image se trouve dans le même dossier que la page web qui l'utilise.

Si cette image se trouvait dans le sous-dossier *images*, la balise `img` s'écrirait :

```

```

NORMES Fermeture facultative de la balise

Vous trouverez parfois une barre de fermeture intégrée, à la fin de la balise , comme dans cet exemple :

```

```

C'était une obligation pour la norme XHTML, où toute balise devait être fermée, même si elle était seule. Le HTML 5 accepte cette forme, mais n'a plus cette exigence ; c'est pourquoi cette barre devenue facultative tend à disparaître, de la même façon que pour les balises
 (saut de ligne) et <hr> (trait horizontal).

Il est important de renseigner l'attribut `alt` d'une balise `image` : c'est un *texte de remplacement* pour ceux qui naviguent en mode texte, notamment les personnes ayant un handicap visuel. C'est aussi un moyen de renseigner les moteurs de recherche.

NORMES Contenu de l'attribut alt

La balise `alt` doit indiquer, de façon concise, le *contenu* de l'image et sa *fonction* si elle en a une (par exemple, si l'image est un bouton d'action). Il est superflu d'y écrire « image de... », « graphisme de... » ou « photo de... ».

Pour une image décorative qui n'a pas de sens en elle-même, ou si une légende est associée à l'image, l'attribut `alt` peut être vide mais reste obligatoire : `alt=""`.

Quant à l'attribut `title`, il permet d'afficher un *texte dans une bulle* au passage de la souris sur l'image : c'est un renseignement complémentaire, dont le but n'est pas de décrire l'image, contrairement à la balise `alt`.

À NOTER L'attribut title est facultatif

Lorsque la balise `title` est absente, il n'y a pas de bulle d'information au survol de la souris. Seuls les anciens navigateurs Internet Explorer, jusqu'à la version 7, utilisent le texte de l'attribut `alt` de la balise en tant que bulle dans ce cas de figure.

Dimensionner une image

Il est possible de préciser les dimensions que doit prendre l'image, en utilisant les attributs `width` et `height`, qui donnent respectivement sa *largeur* et sa *hauteur*. Le plus simple consiste à ne définir qu'un seul de ces attributs, car l'autre sera calculé automatiquement pour que les proportions soient respectées.

Cependant, mieux vaut retailler préalablement l'image aux dimensions souhaitées : l'image affichée sera de meilleure qualité et cela n'obligera pas les internautes à télécharger un gros fichier pour afficher seulement une petite image ! Ce redimensionnement du fichier image peut s'effectuer très simplement, en utilisant par exemple un logiciel gratuit comme *Gimp*, *PhoXo*, *PhotoFiltre*, *IrfanView*, *XnView* ou *Rphoto*.

Une icône sur l'onglet

Nous aurons l'utilité d'insérer une petite image à un endroit bien particulier : à gauche de l'onglet qui affiche le titre de la page, tout en haut de la fenêtre du navigateur. Il s'agit souvent du logo du site ou d'une icône qui correspond bien au site Internet.

Dans ce cas, ce n'est pas la balise `image` qui est utilisée, mais une balise `<link>` à écrire dans l'en-tête de la page, entre `<head>` et `</head>` (et non dans la section `<body>`). Ce qui est logique, puisque cette icône va côtoyer le titre de la page, fourni par la balise `<title>` qui se trouve également dans l'en-tête.

La syntaxe est alors la suivante :

```
| <link rel="icon" href="favicon.png">
```

Ce code indique qu'il s'agit d'un lien relatif à une icône, et l'attribut `href` a exactement la même utilité que dans la balise `` : c'est le nom du fichier qui contient l'image servant d'icône. Si elle est souvent appelée `favicon`, ce qui signifie « icône favorite » puisqu'elle représente le site, ce nom n'est pas du tout obligatoire.

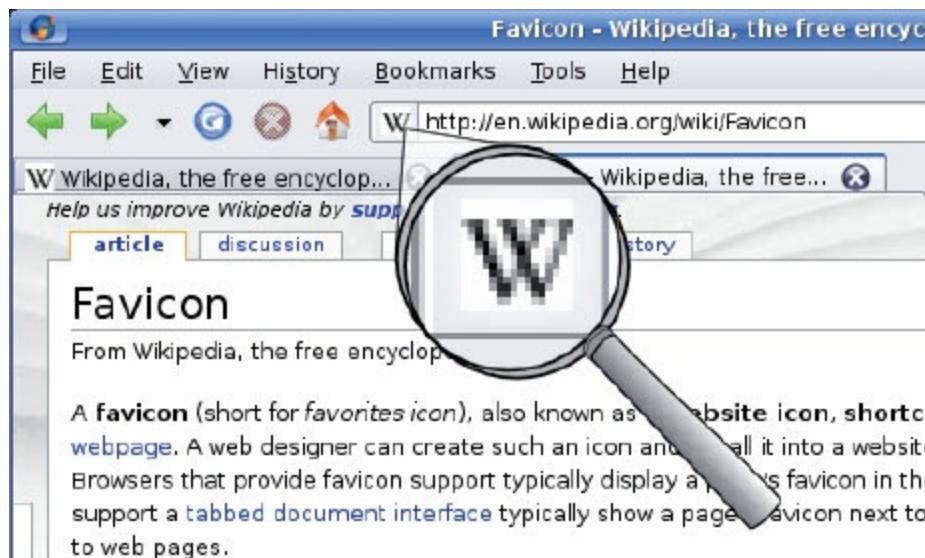


FIGURE 2–15 Wikipédia a choisi comme icône un W noir sur fond blanc qui est devenu célèbre (source : Wikipédia).

Le type et l'extension d'une telle icône sont ceux d'une image, soit généralement .png, .jpg ou .gif (cela peut être un gif animé). Il existe également le format .ico, qui était le plus courant autrefois et qui se fait doucement oublier à présent.

Le lien qui définit l'icône peut être relatif ou absolu, comme pour une image :

- si seul le nom du fichier est fourni, celui-ci doit se trouver dans le même dossier que notre page ;
- si cette icône se trouve dans un sous-dossier `images` par exemple, il faudra écrire `href="images/favicon.png"` ;
- il est également possible d'écrire un lien absolu, commençant par « `http://` », mais il est rare d'aller chercher cette icône sur un site extérieur.

IMPORTANT Taille de l'icône

Pour créer une icône d'onglet, il faut bien garder à l'esprit qu'elle sera affichée en petite taille. Par conséquent, il ne faut pas hésiter à tester plusieurs versions d'images et à observer leur allure une fois réduites. Dans tous les cas, cette icône est une image carrée. Sa taille standard est de 16 ou 32 pixels de côté, mais elle peut atteindre jusqu'à 196 pixels, suivant le système et le navigateur utilisés.

Il est possible d'utiliser l'attribut `sizes` (attention à ne pas oublier le « `s` » du pluriel) qui indique dans l'ordre la hauteur et la largeur de l'icône

| séparées par la lettre x, comme dans cet exemple :

```
<link rel="icon" href="favicon.png" sizes="16x16">
```

Contenus audio et vidéo

Depuis la version 5 du HTML, il existe des balises spécifiques `<audio>` et `<video>`, pour proposer un élément sonore ou une vidéo. Outre leurs noms significatifs, l'avantage de ces balises est d'utiliser les ressources du navigateur et de ne pas nécessiter l'appel à un logiciel externe pour leur lecture.

Leur syntaxe de base est très simple :

```
<audio src="mamusique.mp3"></audio>
<video src="monfilm.mp4"></video>
```

L'utilisation de la balise de fin `</audio>` ou `</video>` permet d'encadrer un contenu alternatif à destination des navigateurs anciens qui n'interpréteraient pas ces balises :

```
<audio src="mamusique.mp3">
    Votre navigateur ne reconnaît pas ce contenu audio.
</audio>
<video src="monfilm.mp4">
    Votre navigateur ne reconnaît pas ce contenu vidéo.
</video>
```

Outre l'attribut essentiel `src` qui indique la source du contenu, c'est-à-dire le nom du fichier audio ou vidéo à lire, ces balises autorisent les attributs suivants.

- `controls` affiche les boutons de contrôle proposés par l'interface du navigateur, comme volume, pause et reprise, retour au début, avance ou retour rapide, arrêt (un exemple est donné par la figure 2-16).
- `autoplay` entraîne un démarrage automatique du son ou du film, dès l'affichage de la page.
- `preload` permet d'anticiper ou non le chargement du contenu à l'ouverture de la page, de façon à en préparer la diffusion. Les valeurs possibles sont `preload="none"` (le fichier ne se télécharge qu'à la lecture), `preload="metadata"` (seules quelques données associées au fichier sont téléchargées : taille, durée, liste de lecture, première image pour une vidéo, etc.) et `preload="auto"` (chargement automatique du fichier à lire).

- `loop` sert à reboucler en permanence la diffusion.
- `mediagroup="..."` permet d'associer différents fichiers audio ou vidéo, en les reliant au même nom de groupe, écrit entre guillemets.

De plus, la balise `<video>` accepte les attributs `width` et `height`, qui définissent les dimensions en pixels de l'objet, ainsi que l'attribut `poster` pour indiquer l'adresse d'une image qui s'affichera dans l'attente de la diffusion de la vidéo.

Exemples d'utilisation

Les extraits de code suivants montrent, plus clairement qu'un long discours, la façon d'utiliser ces balises `<audio>` et `<video>`.

Insertion simple de contenus audio et vidéo

```
<audio src="mamusique.mp3" loop controls>
  Votre navigateur ne peut pas afficher
  ce contenu audio.
</audio>
<video src="monfilm.webm" poster="affiche.jpg" autoplay>
  Votre navigateur ne peut pas afficher
  ce contenu vidéo.
</video>
```



FIGURE 2–16 Interfaces affichées par le navigateur Firefox lorsque les balises `<audio>` et

`<video>` sont utilisées avec l'attribut `controls`

Une deuxième syntaxe existe, utilisant une ou plusieurs balise(s) `<source>` qui proposeront soit un type de décodeur pour la lecture du contenu, soit plusieurs formats du même contenu à diffuser, de façon à ce que le navigateur en reconnaisse au moins un. Les deux exemples suivants montrent ces possibilités d'utilisation et peuvent s'appliquer chacun aux deux balises `<audio>` et `<video>`.

Contenu vidéo proposé dans plusieurs types de codage

```
<video controls="controls">
  <source src="monfilm.webm" type="video/webm">
  <source src="monfilm.mp4">
    Votre navigateur ne peut pas afficher
    ce contenu vidéo.
</video>
```

Contenu audio avec définition d'un codec

```
<audio>
  <source src="ma_musique.ogg" type="audio/ogg; codecs=vorbis">
    Votre navigateur ne peut pas afficher
    ce contenu audio.
</audio>
```

NORMES Formats audio et vidéo

Le W3C ne préconise pas de normes pour l'audio et la vidéo, laissant les navigateurs prendre en compte les formats les plus courants.

- Les fichiers audio sont en général codés en MP3 ou dans le format libre Ogg Vorbis, qui fournit des fichiers plus compacts pour une qualité identique. Le format Opus, libre et performant du point de vue de la compression des fichiers, a vu le jour en 2012 ; il commence à être pris en compte par les navigateurs, tout d'abord par Firefox et Chrome dans leurs versions récentes.
- La vidéo pourra être codée avec le format libre et ouvert WebM, soutenu par de nombreux acteurs de l'informatique. Ce format est en concurrence avec le MPEG-4 (appelé aussi MP4) et sa norme propriétaire H.264, d'usage commun mais soumise à licence. Le MPEG-4 fonctionne aussi avec les normes DivX (propriétaire) et

XDiv (libre).

La version anglaise de Wikipédia fournit des tableaux indiquant la prise en compte, par les différentes versions de navigateurs, des formats audio et vidéo existants :

- › https://en.wikipedia.org/wiki/HTML5_audio
- › https://en.wikipedia.org/wiki/HTML5_video

« Pistes complémentaires » pour les contenus audio et vidéo

Une chanson ou un film proposé par une balise `<audio>` ou `<video>` est parfois accompagné d'une ou plusieurs piste(s) secondaire(s), constituée(s) en principe de textes : sous-titres, références d'auteurs, descriptions complémentaires...

Ces contenus sont alors fournis par la balise `<track>`, incluse entre le début et la fin de la balise `<audio>` ou `<video>` associée. Cette balise `<track>` possède obligatoirement les deux attributs `src="..."` (fichier contenant le texte de cette piste) et `label="..."` (titre de la piste), ainsi que d'autres attributs spécifiques éventuellement :

- `kind="..."` indique le type de contenu fourni par cette piste, avec comme valeurs possibles `subtitles` (sous-titres, retranscription des paroles), `captions` (titres donnant des informations ponctuelles), `chapters` (titres des chapitres du film ou des morceaux de musique) et `metadata` (données à traiter par programme) ;
- `srclang="..."` renseigne la langue de cette piste ;
- `default="default"` s'applique lorsque la piste associée doit être automatiquement diffusée par le lecteur, sauf lorsque l'utilisateur a paramétré son navigateur pour choisir par défaut un autre type de piste, à partir d'une valeur possible de l'attribut `kind`.

Une balise `<audio>` ou `<video>` peut contenir zéro, une ou plusieurs balise(s) `<track>`, suivant le nombre de pistes complémentaires disponibles. Cependant, cette technique ne reste qu'une possibilité théorique, car il faudra attendre son interprétation par les différents navigateurs.

Des blocs pour structurer les pages

Les pages web regorgent de blocs de texte encadrés par les balises `<div>...</div>`, souvent imbriqués les uns dans les autres. Dans bien des cas, leur nombre pourrait être diminué, par un emploi judicieux du couple HTML/CSS.

C'est pourquoi des balises ont été introduites en HTML 5 pour fournir davantage de précisions, notamment sur la structure des pages et le contenu des blocs qui la composent. Les balises de section qui suivent pourront donc avantageusement remplacer une partie de nos bonnes vieilles `<div>`.

Sections de base d'une page

Ces balises ont pour objet de délimiter certaines zones de texte en fonction de leur sens dans la page.

main	Délimite le contenu principal de la page. Cette balise n'est pas obligatoire, mais elle ne doit pas être utilisée plus d'une fois dans la page. Elle ne peut pas non plus être incluse dans une balise <code>article</code> , <code>aside</code> , <code>nav</code> , <code>header</code> ou <code>footer</code> .
section	Regroupe un contenu associé à un thème donné. Bien qu'elle paraisse presque aussi vague que la balise <code><div></code> , elle est cependant moins générique : elle délimitera les différentes parties, soit d'une page, soit d'un bloc indépendant comme ceux indiqués ci-dessous.
article	Il s'agit d'une partie de la page qui peut se comprendre indépendamment du reste, de la même façon que l'article d'un journal par rapport aux autres.
aside	C'est également une partie indépendante de texte qui se suffit à elle-même, à ceci près qu'elle peut n'avoir qu'un rapport éloigné, voire aucun rapport, avec le contenu principal de la page. Il s'agit typiquement d'un encadré, d'une zone grisée à l'écart ou d'un bloc placé sur un côté de la page (d'où son nom), comme une publicité, une annonce interne ou un complément d'information.
nav	Menu de navigation à l'intérieur des pages du site. Il contient normalement une liste de liens, éventuellement précédée d'un titre. Il n'est pas nécessaire d'inclure tous les menus de la page dans cette balise, seuls les principaux ont vocation à l'utiliser.

Une particularité est à noter : si une balise `article` peut comprendre plusieurs balises `section`, une page peut aussi être divisée en sections plus générales. Dans ce cas, chaque balise `section` est susceptible de contenir plusieurs balises `article`. Il est donc prévu que les balises `section` puissent se retrouver à plusieurs niveaux, à l'intérieur d'une page.

Sous-sections de type <div>

Ces blocs peuvent également remplacer certaines balises <div>, mais ils n'ont pas le statut de section. Ils ont vocation à regrouper, à l'intérieur d'une page ou d'une section, des éléments liés entre eux et qui ensemble constituent soit un en-tête, soit un pied de page ou un « pied de section ». Ces blocs sont les suivants.

header	Cette balise a pour but de délimiter l'en-tête de la page ou d'une de ses sections. Elle peut contenir des titres et un paragraphe d'introduction.
footer	Comme son nom l'indique, cette partie regroupe les éléments constituant le pied de page, soit d'une page, soit d'une section.

Dans la même catégorie, il serait possible d'ajouter l'élément <address>, qui, bien sûr, sert à afficher une adresse, mais qui existait déjà depuis la version HTML 4.

Toutes ces balises peuvent contenir des titres du type <h1>, des paragraphes <p>, des listes à puces, etc.

Les balises qui ont le statut de section pourront en outre contenir des entêtes <header> et des pieds de page <footer>.

Exemple utilisant plusieurs sections

```
<article>
  <header> ①
    
    <h1>Fleurs</h1>
    <h2>Les couleurs de la nature</h2>
    <p>Quelques fleurs pour orner votre écran.</p>
  </header>
  <section> ②
    <h2>Les roses</h2>
    
    <p>Du rouge au jaune, elles sont inimitables.</p>
  </section>
  <section> ③
    <h2>Les œillets</h2>
    
    <p>Leur parfum et leur finesse vous séduira.</p>
```

```
</section>
<footer> ④
  <p>Admirez les fleurs qui vous entourent</p>
</footer>
</article>
<aside> ⑤
  <p>Notre almanach est paru !</p>
<aside>
```

Cet exemple concerne une partie de page web, comprenant un article `<article>` et un encadré `<aside>`. Une fois sa mise en forme effectuée à l'aide de la feuille de styles, son aspect sera celui de la figure 2-17.

Il possède un en-tête ①, deux sections ② et ③, ainsi qu'un pied de page ④. Il est suivi d'une section `<aside>` ⑤, qui est un encadré sur le côté de la page.

L'en-tête `<header>` ① aurait pu rassembler les deux titres `<h1>` et `<h2>` à l'intérieur d'un groupe de titres `<hgroup>` si la mise en page le nécessitait, pour positionner ces deux titres à un endroit précis par exemple.

Dans le cas d'un article plus étoffé, l'en-tête `<header>` ① peut inclure un bloc de navigation interne `<nav>`, ce dernier comportant par exemple un titre `<h1>` et une liste non numérotée `` contenant les liens.

Si les sections ② et ③ avaient été plus développées, elles auraient pu ellesmêmes contenir un en-tête `<header>` et un pied de page `<footer>` spécifiques. Cette remarque vaut aussi pour la section `<aside>` ⑤.

Fleurs

Notre almanach
est paru !

Les couleurs de la nature

Quelques fleurs pour orner votre écran...

Les roses



Du rouge au jaune,
elles sont inimitables.

Les œillets



Leur parfum et leur
finesse vous séduira.

Admirez les fleurs qui vous entourent

FIGURE 2–17 Exemple utilisant plusieurs sections HTML 5

COMPLÉMENT Balise <menu> à ne pas confondre avec <nav>

Le HTML 5 réintroduit également les balises `<menu>...</menu>` (disparues dès le HTML 4) pour encadrer une liste de liens correspondant à un menu secondaire ou à un ensemble de boutons de commande. Son emploi est donc distinct de la balise `<nav>`, qui est réservée à un menu de navigation parmi des pages. La balise `<menu>` peut contenir deux attributs :

- `label="..."` pour donner au menu un titre qui sera affiché ;
- `type="..."` qui indique de quel menu il s'agit, avec les valeurs possibles `list` (valeur par défaut, pour une liste de liens secondaires), `context` (lorsqu'il s'agit d'un menu contextuel) et `toolbar` (pour encadrer un ensemble de boutons constituant une barre d'outils).

Cette balise `<menu>` peut contenir des boutons associés à des actions écrites en JavaScript et représentés par la nouvelle balise `<command>`,

utilisable avec les attributs suivants :

- `type` pour définir la forme prise par cette balise, soit `type="command"` (valeur par défaut, c'est un texte qui sert de commande, comme un lien de menu), `type="checkbox"` (case à cocher) ou `type="radio"` (bouton radio) ;
- `label="..."` qui relie à cette commande un titre qui sera affiché ;
- `icon="mon_image.gif"` pour associer une image à cette commande ;
- `checked="checked"` pour activer cette option par défaut (réservé aux types checkbox et radio) ;
- `radiogroup="..."` (pour le type `radio`), qui permet d'associer cette commande à un choix exclusif parmi les autres commandes qui ont le même nom pour cet attribut `radiogroup` ;
- `disabled="disabled"` pour désactiver cette commande, dans les cas où elle n'est pas disponible (cet attribut sera généralement mis en place par un programme PHP ou JavaScript, en fonction du contexte).

La balise `<command>` n'est utilisable qu'à l'intérieur d'une balise `<menu>` et sera ignorée si elle se trouve en dehors de celle-ci.

Deux catégories d'éléments

Dans le premier exemple HTML que nous avons étudié, vous avez remarqué que certaines balises comme `<p>` provoquaient un retour à la ligne, alors que d'autres comme `` laissaient le texte concerné à sa place, à la suite des mots précédents.

Certaines propriétés de mise en forme par les feuilles de styles s'appliqueront dans un cas et pas dans l'autre. C'est pourquoi il est important de bien les distinguer.

Le HTML définit donc pour les balises deux types d'éléments.

- Certains se suivent sur une même ligne de texte : ce sont des *éléments en ligne*, parfois appelés aussi *éléments de niveau texte*.
- Les autres se succèdent verticalement, séparés par un retour à la ligne automatique : ils sont de type *bloc*.

Éléments en ligne

Ils s'écrivent les uns à la suite des autres, dans le texte de la page.

Exemples d'éléments en ligne

```
| <strong>...</strong> <!-- mise en relief -->
| <em>....</em> <!-- emphase -->
```

Les éléments en ligne se répartissent eux-mêmes en deux catégories :

- les « éléments **remplacés** » dont les dimensions (largeur et hauteur) peuvent être définies : images, zones de saisie d'un formulaire...
- les « éléments **non remplacés** » dont la taille est fonction du contenu : éléments `, , , ancre <a>, ...`

Certaines propriétés liées aux blocs peuvent être appliquées aux éléments en ligne de type « remplacés ».

Les principaux éléments HTML de niveau texte, c'est-à-dire de type « en ligne », sont les suivants :

- élément `` (qui sert à délimiter une partie de texte ayant une mise en forme commune) ;
- ancre `<a>` ;
- image `` et objet multimédia `<object>` ;
- texte mis en relief avec `` (italique) ou encore plus en évidence avec `` (en gras) ;
- extraits de citation `<q>` (apparaît entre guillemets) et `<cite>` (italique) ;
- extrait de programme `<code>` ou de texte à entrer au clavier (police de type Courier) ;
- exemple `<samp>` (police Courier), variable `<var>` (italique) ;
- abréviation `<abbr>` ;
- texte inséré `<ins>` (apparaît souligné) et texte supprimé `` (apparaît barré).

Ces éléments en ligne peuvent être imbriqués, mais ils ne peuvent pas contenir d'élément de type bloc.

Éléments de type bloc

Les blocs se placent automatiquement les uns sous les autres. L'utilisation des styles permet de les positionner de façon précise.

Exemples d'éléments de type bloc

```
<h1>....</h1> <!-- titre de niveau 1 -->  
<p>...</p> <!-- paragraphe -->
```

Tout élément de type bloc peut contenir d'autres blocs et, bien sûr, des éléments en ligne, à la notable exception des `<p>` et `<h1>, <h2>,...<h6>` qui ne peuvent inclure d'autres blocs.

ATTENTION

Les paragraphes et les titres ne peuvent inclure de blocs.

À NOTER Marges par défaut

Tous les blocs (sauf `<div>`) possèdent des marges intérieures (`padding`) et extérieures (`margin`) par défaut, qu'il faut préciser ou mettre à zéro dans la feuille de styles.

Voici les principaux éléments HTML de type « bloc » :

- élément `<div>`, qui sert de boîte ou de « conteneur » et dans lequel seront placés d'autres blocs ;
- titres `<h1>` à `<h6>` ;
- paragraphe `<p>` ;
- liste et élément de liste ``, `` et ``, liste de définition `<dl>` ainsi que ses éléments `<dd>` et `<dt>` ;
- citation `<blockquote>` (apparaît en retrait) ;
- texte préformaté `<pre>` (affichage fidèle de tous les espaces et retours à la ligne) ;
- adresse `<address>` (s'affiche en italique, avec espacement vertical).

À ces éléments de base s'ajoutent d'autres balises de type bloc, dont les sections de page notamment, introduites par la norme HTML 5 et qui ont été

détaillées précédemment : `<main>`, `<section>`, `<article>`, `<aside>`, `<nav>`, `<header>` et `<footer>`.

Hiérarchie des éléments : l'héritage

Les éléments qui composent une page HTML, c'est-à-dire les balises avec leur contenu, sont *juxtaposés* ou *imbriqués*. Il en découle une *hiérarchie*, qui sera à prendre en compte dans le choix des propriétés de style.

En effet, certaines propriétés de style ont la faculté d'être héritées et se transmettront donc aux blocs imbriqués, alors que ce ne sera pas le cas pour d'autres propriétés.

Hiérarchie des blocs imbriqués et juxtaposés

Les blocs qui constituent une page forment en quelque sorte une famille ; ils sont désignés les uns par rapport aux autres dans cet esprit.

- Lorsque des blocs sont contenus dans un autre bloc, ils sont les *enfants* de ce dernier.
- Entre eux, ces blocs imbriqués sont des *frères*.
- Le bloc conteneur est leur *père*.
- Le *premier fils* d'un bloc conteneur est le premier des blocs imbriqués qu'il contient.

Voici un exemple qui nous transporte dans l'atmosphère calme et sereine de la campagne et qui produit l'affichage montré par la figure 2-18.

Exemple de code contenant des blocs imbriqués et juxtaposés

```
<body id="ferme"> ①
  <div id="basse-cour">À l'ombre du noisetier... ②
    <p id="poule">Cot ! Cot ! Cot!</p>
    <p id="canard">Coin ! Coin ! Coin !</p>
    <p id="chien">Ouah ! Ouah !
      <em id="puce">une puce pique le chien</em>
    </p>
  </div>
  <div id="enclos">Dans une prairie verte... ③
    <p id="vache">Meuh ! Meuh !</p>
    <p id="cochon">Groin ! Groin !</p>
  </div>
</body>
```

En français, l'histoire se raconte ainsi :

- la ferme contient la basse-cour et l'enclos ;
- la basse-cour contient la poule, le canard et le chien ;
- le chien contient la puce ;
- l'enclos contient la vache et le cochon.

Les éléments HTML de cet exemple sont imbriqués de la même façon, en suivant la logique de cette histoire.

A l'ombre du noisetier...

Cot ! Cot ! Cot!

Coin ! Coin ! Coin !

Ouah ! Ouah ! *une puce pique le chien*

Dans une prairie verte...

Meuh ! Meuh !

Groin ! Groin !

FIGURE 2–18 Exemple de blocs imbriqués et juxtaposés



FIGURE 2–19 À la ferme...

Termes hiérarchiques utilisés en HTML/CSS

Le bloc `<body id="ferme">` ① est l'*ancêtre* commun à tous les autres blocs.

Il est le *père* des deux éléments `<div id="basse-cour">` ② et `<div id="enclos">` ③ qui sont ses deux descendants directs, appelés ses *fils* ou ses *enfants*. Ces deux blocs sont donc *frères*. Le *premier fils* de l'élément `<body id="ferme">` ① est `<div id="basse-cour">` ②.

Le bloc `<div id="basse-cour">` ② a trois fils (paragraphes *poule*, *canard*, *chien*, qui sont frères) et un petit-fils (le texte *puce*, qui est le fils du paragraphe *chien*).

Le bloc `<div id="enclos">` ③ a deux fils (les paragraphes *vache* et *cochon*).

Héritage des propriétés de style

Comme nous l'avons vu, certaines des propriétés définies dans les feuilles de styles sont *héritées*, c'est-à-dire qu'elles sont transmises aux éléments imbriqués ; d'autres ne peuvent pas l'être.

Si une propriété *héritée* est appliquée à un élément, elle s'appliquera en même temps à tous les éléments que celui-ci contient.

Si, en revanche, elle n'est *pas héritée*, elle ne se retrouvera pas sur les éléments imbriqués.

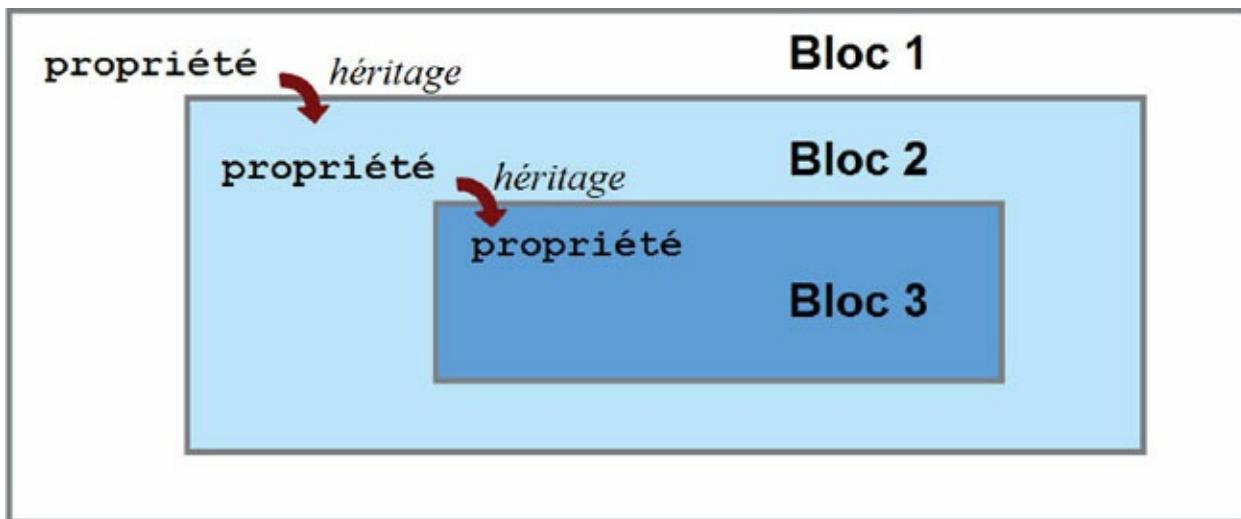


FIGURE 2–20 Principe de l'héritage d'une propriété, illustré ici pour des blocs imbriqués

Reprendons l'exemple de la ferme pour illustrer le comportement de deux propriétés dans les blocs imbriqués.

La propriété `font-family` (police d'écriture) est *héritée*. Supposons, par exemple, que la propriété `font-family: Arial;` soit appliquée aux blocs `<div>` ② et ③ seulement (basse-cour et enclos). Grâce à l'héritage de cette propriété et comme le montre la figure 2–21, tous leurs descendants directs ou indirects seront également écrits en Arial (ces descendants sont les blocs `<p>`, qui sont leurs enfants, et l'élément ``, qui est un petit-enfant).

A l'ombre du noisetier...

Cot ! Cot ! Cot!

Coin ! Coin ! Coin !

Ouah ! Ouah ! *une puce pique le chien*

Dans une prairie verte...

Meuh ! Meuh !

Groin ! Groin !

FIGURE 2–21 La propriété *font-family* est héritée.

La propriété *border* (type de bordure) n'est *pas héritée*. La figure 2-22 donne le résultat affiché lorsqu'une bordure est attribuée aux seuls blocs `<div>` ② et ③ (basse-cour et enclos). Leurs descendants ne seront pas encadrés, car ils conserveront la valeur par défaut « aucune bordure ».

A l'ombre du noisetier...

Cot ! Cot ! Cot!

Coin ! Coin ! Coin !

Ouah ! Ouah ! *une puce pique le chien*

Dans une prairie verte...

Meuh ! Meuh !

Groin ! Groin !

FIGURE 2–22 La propriété border (type de bordure) n'est pas héritée.

Lorsque nous utiliserons des feuilles de styles CSS pour mettre en forme les différents éléments de nos pages, nous nous inquiéterons des facultés d'héritage de chacune des propriétés utilisées :

- lorsqu'il s'agit d'une propriété héritée, il est inutile de la réécrire pour chaque bloc imbriqué ; il suffira de l'appliquer une fois au bloc conteneur ;
- en revanche, si la propriété de style concernée n'est pas héritée, elle devra être répétée pour chacun des blocs imbriqués auxquels il faudra affecter ce style.

Nous allons voir à présent comment vérifier la validité de notre code, pour conclure ce chapitre.

Validation du code HTML

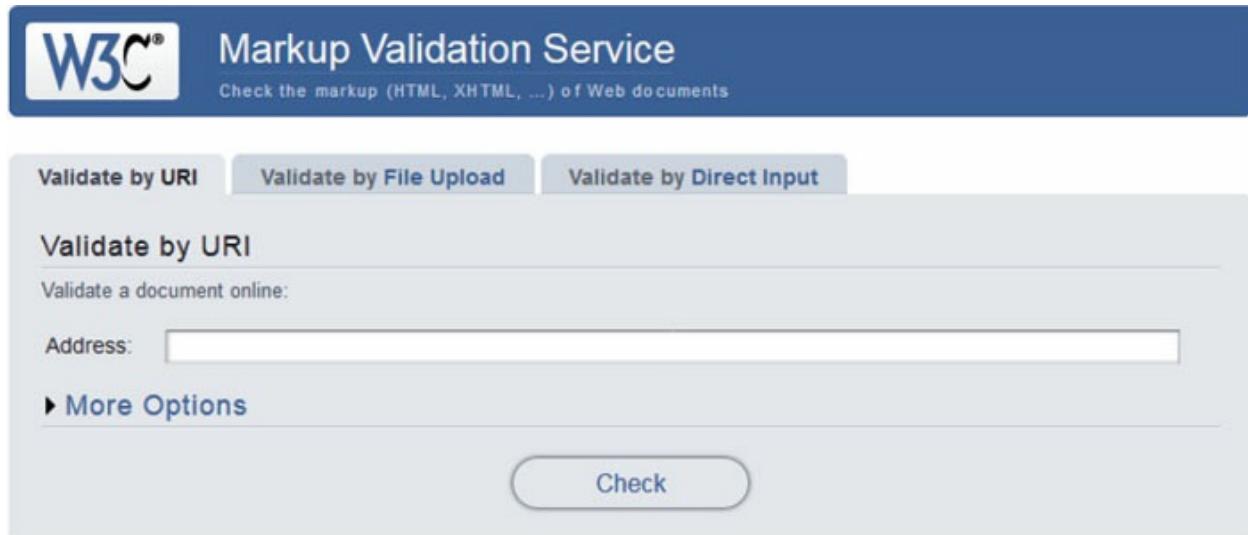
Pour vérifier si le code d'une page est valide, suivant la norme indiquée dans la balise `<!DOCTYPE ...>`, il suffit de mettre cette page en ligne et d'indiquer son url au validateur du W3C, à l'adresse suivante : <https://validator.w3.org>.

Cette page propose également la validation d'une page par un copiercoller de son contenu dans un cadre. Une option propose même la vérification d'un extrait de page, pour lequel il faudra alors choisir la version de HTML correspondant à cette partie de code.

À NOTER La validation n'est qu'une indication

Le validateur vérifie uniquement la *syntaxe* de la page (en quelque sorte, sa « grammaire »), mais pas la logique dans l'emploi des balises, à savoir si le sens associé à telle ou telle balise correspond bien au texte qu'elle encadre.

Vous voilà familiarisé avec le HTML : à présent, la construction d'une page élémentaire avec quelques balises simples est tout à fait dans vos cordes. De plus, nous sommes sensibilisés à des notions telles que l'imbrication des blocs et l'héritage des propriétés de style, qui nous seront utiles pour mettre en forme notre page.



The image shows the W3C Markup Validation Service interface. At the top, there is a blue header bar with the W3C logo on the left and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents" on the right. Below the header, there are three tabs: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" tab is selected. Under this tab, there is a section titled "Validate by URI" with the sub-instruction "Validate a document online:". Below this is a "Address:" label followed by a text input field. To the right of the input field is a "Check" button. Below the input field, there is a link "More Options" preceded by a small arrow icon.

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-based validator](#).

FIGURE 2-23 Page de validation du HTML, proposée par le W3C à l'adresse <https://validator.w3.org>

Nous venons d'étudier les éléments de base de la norme HTML 5. Le chapitre qui suit va nous permettre d'aller plus loin en découvrant des balises et attributs plus spécifiques, et en apprenant à créer un formulaire de contact. Si les balises courantes que nous venons d'étudier vous paraissent suffisantes et si vous êtes pressé d'aborder les feuilles de styles, vous pourrez passer ce chapitre.

chapitre 3

Nouveautés du HTML 5

thebeautyincssdesign
CSS Zen Garden

Select a Design:

- Under the Seal par Eric Stoltz
- Make 'em Proud par Michael McAghon and Scotty Reifsnyder
- Orchid Beauty par Kevin Addison
- Oceanscape par Justin Gray
- CSS Co., Ltd. par Benjamin Klemm
- Sakura par Tatsuya Uchida
- Kyoto Forest par John Politowski
- A Walk in the Garden par Simon Van Hauwermeiren

Archives:

- Conceptions suivantes »
- Voir toutes les conceptions

Resources:

- Voir le CSS pour cette conception
- Ressources CSS
- FAQ
- Soumettez une conception
- Traductions

Une démonstration de ce qu'on peut accomplir lorsqu'on utilise les CSS pour la conception web. Sélectionnez n'importe quelle feuille de style listée pour charger le résultat sur cette page.

Téléchargez les fichiers d'exemple [html](#) et [css](#)

悟りへの道
The Road to Enlightenment

Les reliques passées des sélecteurs spécifique aux navigateurs, des DOMs incompatibles, et du manque de support des CSS encombrent un long chemin sombre et morne.

Aujourd'hui, nous devons nous clarifier l'esprit et nous débarassez des pratiques passées. La révélation de la véritable nature du Web est maintenant possible, grâce aux efforts infatigables des gens du W3C, du WaSP et des créateurs des principaux navigateurs.

Le Jardin Zen css vous invite à vous relaxer et à méditer sur les leçons importantes des maîtres. Commencez à voir clairement. Apprenez à utiliser ces techniques (bientôt consacrées par l'usage) de manière neuve et revigorante. Ne faites qu'un avec le Web.

何のためにあるの？
So What is This About?

Il y a clairement un besoin pour les graphistes de prendre les CSS au sérieux. Le Jardin Zen vise à exciter, inspirer, et encourager la participation. Pour commencer, voyez quelques concepts choisis dans la liste. Cliquez sur n'importe lequel pour le charger sur cette page. Le code HTML demeure le même, et seule la feuille de style extérieure change. Oui, vraiment. Les CSS permettent un contrôle complet et total du style d'un document hypertexte. La seule manière de vraiment démontrer cela d'une manière qui excite les gens est de démontrer ce qui peut vraiment être, une fois que les personnes ont été placées dans les mains de ceux capables de créer la beauté basée sur la forme.

Nous allons maintenant étudier les nouvelles fonctions

de la norme HTML 5 : balises et attributs spécifiques, ainsi que création d'un formulaire de contact avec le contrôle des informations saisies.

SOMMAIRE

- ▶ **Davantage de signification pour le texte**
- ▶ **Éléments modifiables**
- ▶ **Des blocs spécifiques**
- ▶ **Images dynamiques et légendes**
- ▶ **Créer un formulaire de contact**
- ▶ **À utiliser progressivement**

Davantage de signification pour le texte

Le HTML 5 propose des balises spécifiques pour repérer des parties de phrases, préciser ce qu'elles représentent et apporter de nouvelles fonctions.

Balises spécifiques pour le texte

À l'intérieur d'une phrase, la balise générale `` entoure les mots, groupes de mots ou valeurs qui doivent avoir une mise en forme particulière. Le HTML 5 propose d'autres balises pour repérer certaines significations courantes de contenu.

Surlignage de mots

La balise `<mark>` a pour objectif le surlignage de mots, d'expressions ou de phrases pour montrer qu'ils sont plus précisément en rapport avec le sujet de la page. En général, elle servira à mettre en évidence un mot-clé dans une page qui affiche le résultat d'une recherche effectuée à partir de ce mot.

Exemple donnant les résultats d'une recherche sur le mot « riz »

```
<h2>Le <mark>riz</mark> dans nos repas</h2>
<ul>
  <li><mark>Riz</mark> soufflé le matin ;</li>
  <li><mark>Riz</mark> cantonais à midi ;</li>
  <li>Gâteau de <mark>riz</mark> le soir.</li>
</ul>
```



FIGURE 3–1 Repérage d'un mot dans un résultat de recherche, grâce à la balise `<mark>`

Mesures comprises entre deux bornes

Si le HTML 5 a introduit la balise `<meter>`, ce n'est pas pour une utilisation avec tous les nombres qui constituent des mesures, mais uniquement pour ceux qui correspondent à une valeur placée entre un minimum et un maximum connus.

Six attributs peuvent être utilisés avec cette balise `<meter>` :

- `min` et `max` indiquent les valeurs minimale et maximale possibles, qui sont par défaut `0` et `1` lorsque ces attributs ne sont pas précisés ;
- `low` et `high` définissent les limites en deçà ou au-delà desquelles la valeur sera considérée comme basse ou haute ;
- `optimum` donne la valeur optimale, qui n'est pas nécessairement la valeur moyenne, car c'est parfois la valeur maximale ou minimale ;
- `value` est la valeur à afficher (alternative à l'écriture d'une valeur entre `<meter>` et `</meter>`, elle permet l'utilisation du JavaScript pour modifier cette valeur).

Exemples d'utilisation de la balise `<meter>`

```
<h1>Résultats des rencontres sportives</h1>
<ul>
  <li>Danse : <meter min="0" max="10">9,5</meter></li>
  <li>Tirs au but<meter max="5">4</meter> sur 5</li>
  <li>Taux de qualification : <meter>75 %</meter></li>
</ul>
```

Dates et heures

À partir d'une date ou d'une heure écrite dans un format standard, la balise `<time>` permettra au navigateur de l'afficher au format « régional », défini par l'internaute sur son ordinateur et qui correspond généralement à celui utilisé dans son pays.

Exemple affichant date et heure avec la balise `<time>`

```
<p>Le <time>1789-07-14</time>, la prise de la Bastille s'est terminée à
<time>17:00</time>.</p>
```

Dans le rendu de cet exemple sur un navigateur, la date se présentera sous la forme habituelle du pays de notre visiteur. De même, l'heure pourra s'afficher sous la forme **17h00** ou **5pm**, suivant le cas.

Deux attributs sont utilisables avec la balise `<time>`.

- `datetime` pour écrire à l'intérieur de la balise la date ou l'heure indiquée (ou bien les deux). Le format complet associant date et heure est le

suivant : `datetime="YYYY-MM-DDThh:mm:ss"`, YYYY représentant l'année, MM le mois, DD le jour, hh l'heure et ss les secondes. Cette valeur peut éventuellement être complétée par un point et deux chiffres, pour afficher les centièmes de seconde.

- `pubdate` sert à préciser que la date fournie est celle de la publication de l'article qui contient la balise `<time>` ou du document entier, sa syntaxe étant `pubdate="pubdate"`.

Texte barré

Le HTML 5 a repris quelques balises courantes du HTML 4 dont l'emploi était déconseillé en XHTML 1 : ``, `<i>` et `<s>` pour *gras*, *italique* et *barré*. Cependant, il nuance un peu leur sens.

Nous avons déjà étudié les balises `` et `<i>`, qui correspondent à la mise en gras et en italique, mais uniquement lorsqu'il ne s'agit pas de mettre en relief le texte concerné.

La balise `<s>` (pour *strike* en anglais) permet de rayer un texte qui est incorrect ou qui n'est plus approprié. Le plus simple consiste à le supprimer, mais il est parfois utile d'en laisser la trace tout en montrant bien qu'il n'est plus valable. Un exemple simple utilisant `<s>` est celui du *prix barré* qui fait place à une promotion :

```
<h3>Promotion sur les voyages dans l'espace !</h3>
<p>Prix normal : <s>20 000 000 €</s></p>
<p>Cette semaine : <strong>19 999 900 €</strong></p>
```



FIGURE 3–2 Utilisation de la balise `<s>` pour rayer un contenu qui n'est plus d'actualité

Coupure des mots trop longs

Lorsqu'un mot ou un groupe de caractères attachés est très long, cela peut poser des problèmes de mise en page, comme le montre la figure 3-3. La

justification du texte allonge exagérément les espaces sur la ligne qui précède le long mot en question, et ce dernier constituant un tout, il peut déborder du bloc dont il fait partie.

La solution, présentée par cette même figure, consiste à autoriser des coupures de lignes à l'intérieur de ces longs mots à certains endroits précis, qui seront repérés par la balise `<wbr>` comme *word break*, c'est-à-dire *coupure de mot*.

Sans <code><wbr></code>	Avec <code><wbr></code>
<p>Mes recettes sont sur le site www.jaimelesbananesetlespoiresavecduchocolat.com</p> <p>Elles sont faciles, variées et amusantes, essayez-les !</p>	<p>Vous aussi, partagez avec nous une sélection de vos meilleures recettes.</p> <p>Chaque mois, les visiteurs sélectionnent leurs préférées et le gagnant reçoit le badge officiel du site.</p>

FIGURE 3-3 Dans les mots trop longs, la balise `<wbr>` permet de placer, à certains endroits précis, des autorisations de retour à la ligne.

Dans notre exemple, le mot problématique est une gigantesque adresse web :

www.jaimelesbananesetlespoiresavecduchocolat.com

Écrite telle quelle, elle sort allègrement de sa colonne. Il faut donc y autoriser

des coupures, en y insérant quelques balises `<wbr>` :

www.jaime<wbr>lesbananes<wbr>etlespoires<wbr>avecduchocolat.com

Le navigateur permettra de revenir à la ligne (sans tiret de coupure de mot) lorsqu'apparaît la balise `<wbr>`. Dans l'exemple présenté par la figure 3-3, il utilise la première et la troisième balises `<wbr>` pour insérer un retour à la ligne, tandis qu'il ignore la deuxième, dont il n'a pas besoin.

Autres balises de texte

Une liste de quelques autres balises, moins courantes d'utilisation dans une première approche du HTML, complétera notre énumération.

NORME Liste complète des balises HTML sur le site du W3C

La liste détaillée des balises HTML est disponible sur de nombreux sites, comme celui-ci (en anglais) qui détaille de façon claire et précise les normes du W3C :

► <https://www.w3schools.com/tags>

TABLEAU 3–1 Quelques balises HTML d'utilisation moins courante

Balise	Action
<address>	crée un nouveau paragraphe constitué par une adresse, écrite en italique
<q>	<q> comme <i>quote</i> indique une courte citation dans une phrase, le texte étant placé entre guillemets
<cite>	sert également à afficher une citation au cours d'une phrase, le texte étant alors écrit en italique
<dfn>	définition d'un mot, en cours de phrase (en italique)
<abbr>	abréviation (pas de mise en forme)
 <ins>	texte supprimé (barré grâce à la balise comme delete) et remplacé par un autre (à insérer en utilisant la balise <ins>)
<var>	nom d'une variable (en italique)
<code>	extrait de code informatique (police Courier)
<samp>	exemple de code informatique (police Courier)
<kbd>	saisie au clavier (police Courier)
<pre>	préformaté (police Courier, espaces et sauts de lignes affichés tels qu'ils sont notés)

Des éléments modifiables

Certains attributs permettent au visiteur de personnaliser nos pages ; il pourra par exemple déplacer des blocs ou modifier certains contenus.

Éléments déplaçables dans la page

De plus en plus de sites nous proposent des pages sur mesure, en nous donnant par exemple la possibilité de déplacer certains de leurs éléments. L'attribut `draggable` pourra déterminer si un élément est déplaçable ou non, avec les valeurs suivantes :

- `draggable="true"` si l'élément peut être déplacé par l'utilisateur ;
- `draggable="false"` pour interdire le glisser-déposer ;
- `draggable="auto"` pour utiliser l'option par défaut du navigateur (déplacement des objets autorisé ou non), définie dans le logiciel à l'aide d'un menu du type *Options* ou *Péférences*.

Glisser-déposer en HTML 5

Elément 1 : <div id="div1" draggable="true";>
Elément 2 : <div id="div2" draggable="false";>

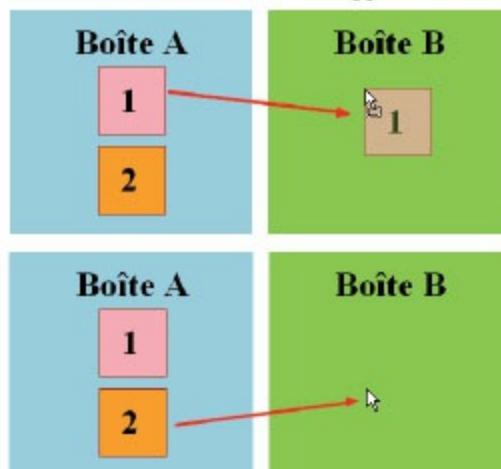


FIGURE 3–4 Le glisser-déposer d'éléments, géré par ailleurs en JavaScript, peut être autorisé ou interdit à l'aide de l'attribut `draggable`.

COMPLÉMENT Principe du glisser-déposer

La gestion du glisser-déposer, en anglais *drag and drop*, s'effectue en

JavaScript à partir des événements suivants :

- `ondragstart/ondragstop` : début/fin du déplacement de l'objet ;
- `ondragenter/ondragexit` : la souris entre dans un bloc de dépose éventuel/sort de ce bloc ;
- `ondragover` : l'objet bouge au-dessus d'un bloc de dépose éventuel ;
- `ondragdrop` : l'objet concerné est lâché dans une zone de dépose.

Ces événements sont à traiter à l'intérieur de plusieurs balises : l'élément à déplacer, le bloc qui le contient, ainsi que le ou les bloc(s) susceptible(s) de le recevoir. Par exemple, si à un bloc est associé l'événement `ondragover="return false;"`, le placement de l'objet mobile y sera interdit. La page suivante contient une démonstration de code JavaScript qui a inspiré l'exemple de la figure 3-4 :

► <http://ljouanneau.com/lab/html5/demodragdrop.html>

Contenus modifiables

L'attribut `contenteditable` peut être inséré dans n'importe quelle balise HTML, pour permettre à l'internaute d'en modifier le contenu. Ses valeurs possibles sont :

- `contenteditable=""` ou `contenteditable="true"`, si le contenu de la balise doit être modifiable ;
- `contenteditable="false"` pour que le texte affiché reste fixe ;
- `contenteditable="inherit"` (valeur par défaut) lorsque la valeur de cet attribut est héritée, donc définie par celle attribuée à l'élément parent, qui contient la balise concernée.

Bien évidemment, les modifications ainsi effectuées n'apparaissent que sur l'écran du visiteur et ne viennent pas modifier le contenu du site. Le chargement de la page, par la commande **Actualiser** du navigateur ou la touche **F5** du clavier, permet de retrouver la version initiale du texte.

Blocs spécifiques

Une page dans un cadre

En utilisant la balise <iframe>, il est possible d'inclure, à l'intérieur d'une page web, un cadre contenant une autre page HTML, cette dernière pouvant appartenir à votre site ou provenir d'un autre.

ATTENTION Problèmes d'accessibilité

La balise <iframe> est mentionnée ici, car elle fait partie du « paysage web » et de la norme HTML 5, mais son emploi n'est pas conseillé : elle pose des problèmes d'accessibilité pour les personnes handicapées et peut aussi compliquer la navigation dans votre site.

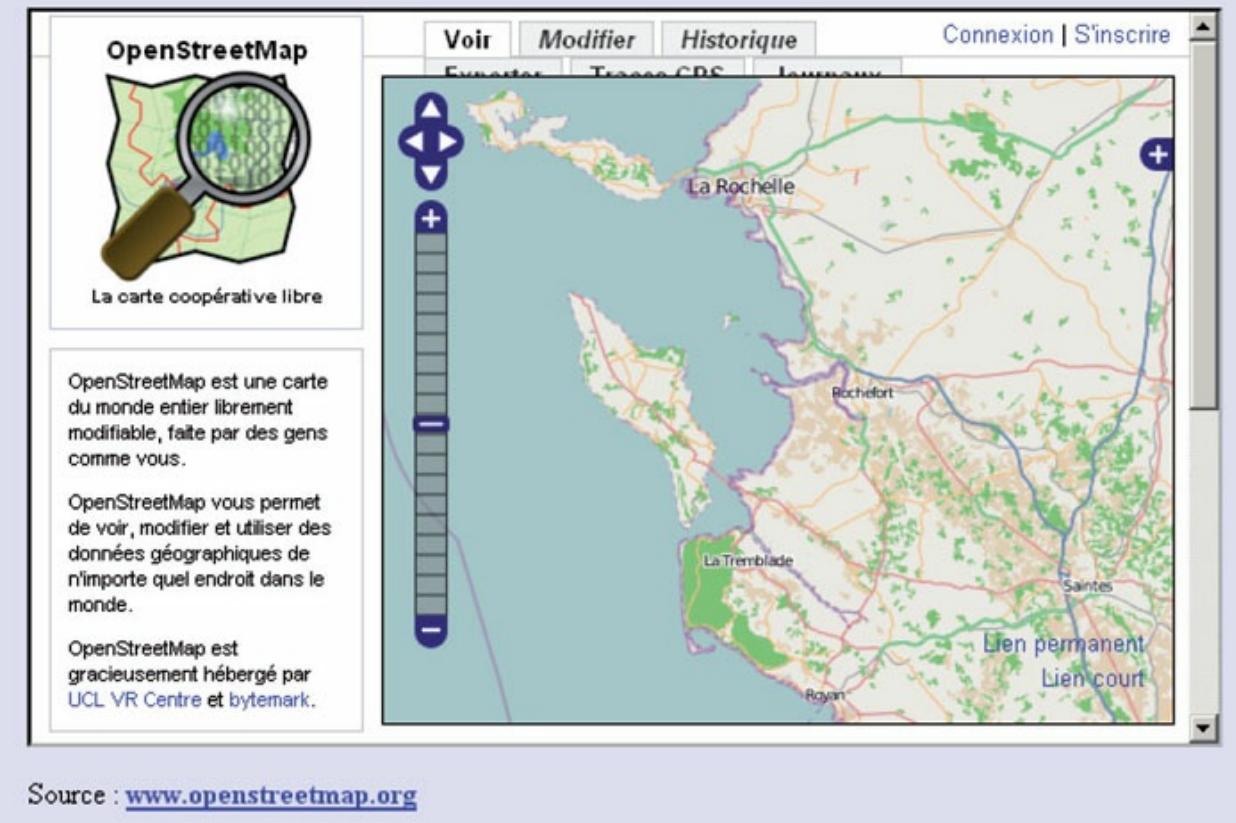
Même si elle n'est pas du tout recommandée en matière d'accessibilité, l'existence de la balise <iframe> est à connaître, car vous pourrez la rencontrer en surfant sur le Web. Elle peut être intéressante dans certains cas très particuliers, comme dans l'exemple suivant qui affiche une carte dynamique du site de cartographie coopérative libre OpenStreetMap (www.openstreetmap.org).

Balise <iframe> affichant la page www.openstreetmap.org

```
<h2>OpenStreetMap, la cartographie libre</h2>
<iframe width="420" height="370" src="http://cartosm.eu/ map?
lon=-0.84&lat=45.89">
  <p>Votre navigateur ne reconnaît pas les iframes.</p>
</iframe>
<p>Source : <a href="http://www.openstreetmap.org">
  www.openstreetmap.org</a> -
  Service : <a href="http://cartosm.eu">Cartosm</a></p>
```

Les dimensions du cadre pourront bien sûr être modifiées ; le résultat sera du type de celui présenté par la figure 3-5. Entre les deux balises <iframe> et </iframe> se trouve un message de remplacement : il sera affiché par les navigateurs qui ne comprennent pas ce type de cadre.

OpenStreetMap, la cartographie coopérative libre



Source : www.openstreetmap.org

FIGURE 3–5 Une fenêtre affiche le contenu du site OpenStreetMap.

IMPORTANT Droit et éthique

Si la balise `<iframe>` permet d'inclure dans votre page une autre page provenant d'un site différent, il est important de mentionner clairement sa source, pour des raisons d'éthique. N'oubliez pas de vérifier qu'un tel affichage est autorisé. En particulier, ce n'est pas le cas pour le site de cartographie privé GoogleMaps.

Des détails sur demande

La balise `<details>` donne la possibilité d'afficher sur demande des précisions sur un texte. Elles sont initialement masquées, et il suffira à l'internaute de cliquer sur le titre associé, affiché à l'aide de la balise `<summary>`.



FIGURE 3–6 Le navigateur Chrome reconnaît les balises `<details>` et `<summary>`. Un clic sur le titre fait apparaître ou disparaître les détails et modifie le sens de la flèche, affichée automatiquement.

Il est possible d'afficher directement les précisions en question, en ajoutant l'attribut `open="true"` à la balise `<details>`. Si cet attribut est absent ou s'il a la valeur `open="false"`, les précisions sont masquées jusqu'au clic sur le titre associé.

Exemple de détail d'information masqué

```
<details>
  <summary>Le W3C</summary>
  <p>W3C signifie World Wide Web Consortium.</p>
  <p>Il a été créé par Tim Berners-Lee en octobre 1994.</p>
</details>
```

Dans cet exemple, seul le titre, encadré par `<summary>`, est affiché. Cliquer dessus fera apparaître les détails qui lui sont associés, c'est-à-dire les deux paragraphes qui le suivent. Un deuxième clic sur ce titre masque à nouveau ces détails.

Images dynamiques et légendes

Affichage dynamique (<canvas>)

La nouvelle balise `<canvas>` affiche dynamiquement une image mise à jour à l'aide d'un code JavaScript. C'est en quelque sorte une balise image dont le contenu viendrait d'ailleurs, c'est-à-dire du script en question. Quant au texte qui se trouve entre `<canvas>` et `</canvas>`, il est ignoré et sert d'alternative pour les navigateurs qui ne comprennent pas cette balise.

L'exemple qui suit, inspiré de la page www.scriptol.fr/html5/canvas/premierspas.php, montre son utilisation de façon schématique. La figure 3-10 indique le résultat obtenu avec ces quelques lignes de code.

Principe d'utilisation de la balise `<canvas>` : tracé d'un rectangle vert

```
<script type="text/javascript">
function fonction_canvas() ①
{
    var balise_canvas = document.getElementById('image1'); ②
    if (balise_canvas.getContext) { ③
        var context2D = balise_canvas.getContext('2d'); ④
        context2D.fillStyle = "rgb(0,220,0)"; ⑤
        context2D.fillRect(50,10,150,100); ⑥
    }
}
window.onload = fonction_canvas; ⑥
</script>

<h3>Voici un rectangle vert</h3>
<canvas id="image1"> ⑧
    Votre navigateur ne peut pas afficher cet élément. ⑨
</canvas>
```

Le fonctionnement de l'exemple ci-dessus est le suivant.

- La fonction JavaScript, appelée ici `fonction_canvas` ①, est exécutée au chargement de la page, grâce à la ligne ⑦.
- Elle repère la balise `<canvas>` concernée ⑧ par son identifiant ② puis

s'assure que la méthode `getContext` est bien disponible ③, celle-ci permettant d'utiliser les fonctions de dessin.

- Dans ce contexte de dessin en deux dimensions ④, deux éléments sont créés : un remplissage vert ⑤ et le tracé d'un rectangle plein ⑥ avec cette couleur. Le rectangle est défini par les coordonnées (x,y) de son coin supérieur gauche et de son coin inférieur droit.
- La balise `<canvas>` ⑧ affiche le dessin ainsi défini dans son « contexte à deux dimensions ». Les navigateurs qui ne connaissent pas cette balise montreront le texte de la ligne ⑨.



FIGURE 3–7 Résultat obtenu avec notre exemple tout simple, qui affiche un rectangle vert

Il est évident que l'utilisation réelle de la balise `<canvas>` ne se limitera pas à un exemple aussi simple. La structure étant la même, le code JavaScript sera beaucoup plus sophistiqué et modifiera automatiquement l'image affichée, pour l'actualiser en fonction de nouvelles données ou pour répondre à une action de l'utilisateur.

Une application très utile consiste à afficher une carte dynamique à l'écran, comme celles fournies par le service Google Maps. Nous ne nous étendrons pas sur les quelques pages de code JavaScript nécessaires à sa mise en œuvre ! Cependant, il suffit d'observer la figure 3-8 pour constater le gain considérable de rapidité qu'elle permet d'obtenir.

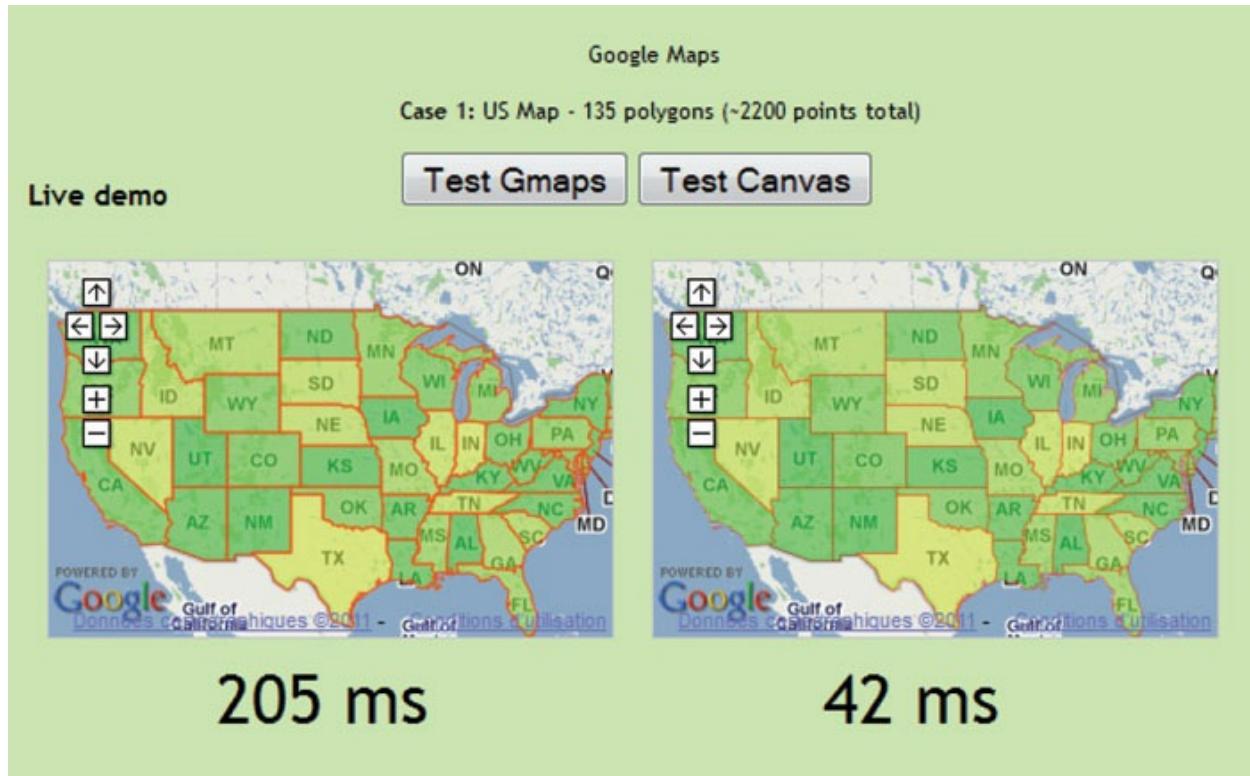


FIGURE 3–8 Grâce à la balise `<canvas>`, l'affichage d'une carte dynamique est trois à cinq fois plus rapide (durées indiquées en millisecondes). Cet exemple a été créé par Ernest Delgado (www.ernestdelgado.com).

La balise `<canvas>` accepte les attributs courants `width` et `height` qui définissent respectivement sa largeur et sa hauteur en pixels.

Indicateur de progression (`<progress>`)

Par le même type de rafraîchissement dynamique que celui évoqué précédemment, la balise `<progress>` permet de montrer graphiquement une progression (chargement d'un contenu, évolution d'un projet, etc.).

Elle possède deux attributs : `value` qui donne sa valeur actuelle et `max` pour définir sa valeur maximale (égale à 1 par défaut, si `max` n'est pas utilisé). La valeur de cette balise pourra être soit définie dynamiquement en Java-Script, soit rectifiée manuellement s'il s'agit d'une progression longue dans le temps, comme dans le cas de l'avancement d'un projet.

Exemple d'écriture de la balise `<progress>`

```
<p>Mise à jour de la documentation<br/><progress value="80" max="100">80</progress>
```

|| </p>

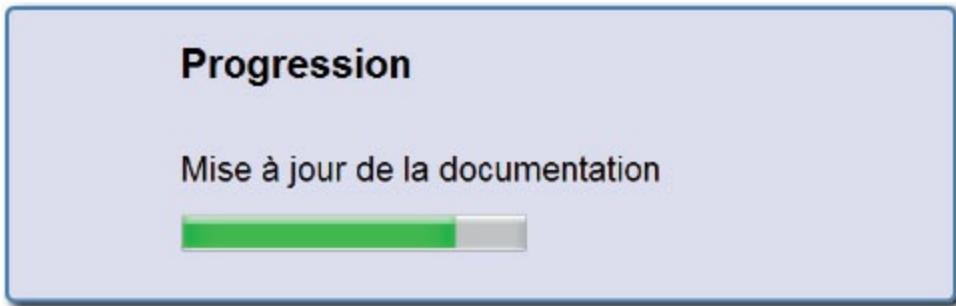


FIGURE 3–9 Effet créé par la balise `<progress>`, ici avec Firefox

Images, photos et figures (`<figure>`)

La balise `<figure>` est une sorte de minisection regroupant tout ce qui est en rapport avec l'image qu'elle contient. Une balise `<figcaption>` peut y être incluse, au début ou à la fin : elle permet d'indiquer un titre pour cette figure.

Utilisation de la balise `<figure>`

```
<figure>
<figcaption>Tux en mousquetaire !</figcaption>

<p>Par André Pascual de Linuxgraphic.org pour les GNUsquetaires.
</p>
</figure>
```

Regrouper à l'intérieur de la balise `<figure>` tout ce qui concerne une image donnée apporte du sens à la page, facilitant l'accessibilité aux personnes ayant un handicap visuel et donnant des informations aux moteurs de recherche. Cette balise ne correspond pas à une mise en forme spécifique de la part du navigateur, mais nous pourrons lui en attribuer une à l'aide d'une feuille de styles.

Formulaires de contact

Pour inviter le visiteur à prendre contact avec nous, un formulaire à remplir est plus simple et plus direct à utiliser qu'une adresse de messagerie. La figure 3-10 montre les principaux éléments qu'il pourra contenir, que nous allons étudier à partir des numéros repères.

Les zones de saisie d'un formulaire sont parfois appelées *champs*, terme utilisé pour les bases de données.

Balise formulaire

L'ensemble du formulaire ① se trouve entre les balises `<form>` et `</form>`. La balise de début doit contenir au minimum deux attributs.

- La *méthode d'envoi des données*, en principe `method="post"`, permet l'envoi masqué des données.
- Le fichier qui effectuera l'*action d'envoi du message* et le confirmera au visiteur, tel que `action="envoi.php"`. Il s'agit obligatoirement d'un fichier écrit en langage PHP : nous en examinerons un exemple très simple plus loin.

ALTERNATIVE Envoi visible des données

La technique `method="get"` est peu employée, car elle affiche ces données dans la barre d'adresse du navigateur.

The screenshot shows a web form titled "Merci de nous donner votre avis sur le site". The form is enclosed in a large orange border. A vertical brace on the left side groups the first two items (name and password) under reference ①, and the next two items (message and navigation) under reference ②. To the right of the form, three dropdown menus are shown in boxes:

- ⑤: A dropdown menu for gender with options: Homme, Adulte, Homme (highlighted), Femme, Jeune, Garçon, Fille.
- ⑨: A dropdown menu for how they discovered the site with options: Presse, Presse (highlighted), Internet, Amis ou autres.
- ⑩: A dropdown menu for navigation with options: Très simple, Correcte (highlighted), Compliquée.

Below the form are two buttons: "Effacer" and "Envoyer".

FIGURE 3–10 Exemple de formulaire

Tout formulaire sera donc encadré par des balises de ce type, avec ici l'appel d'un fichier `envoi.php`, qu'il faudra écrire pour transmettre le message :

```
<form method="post" action="envoi.php">
```

... toutes les balises du formulaire seront ici ...

</form>

Regroupement de parties de formulaire

Pour améliorer la présentation, il est possible de lier visuellement certaines parties de formulaire, en les plaçant entre des balises `<fieldset>` et `</fieldset>`. Nous retrouvons ici le terme de *champ*, soit *field* en anglais. En principe, le navigateur encadre d'un filet les parties ② ainsi constituées ; la figure 3-10 en contient deux.

Les étiquettes

Elles ne sont pas numérotées sur notre exemple, car elles apparaissent partout ! Ce sont les textes qui présentent les zones de saisie, comme *Nom ou pseudo*, *Mot de passe*, *Votre message*, etc.

Chacune d'entre elles est encadrée par les balises `<label>` et `</label>`, le mot *label* signifiant *étiquette* en anglais.

Zones de texte simples

La balise `input` correspond à une *entrée* ou *saisie* d'informations. Elle peut prendre différentes formes, notamment celle d'une zone de saisie à une seule ligne : elle est idéale pour indiquer un nom, un prénom, un numéro de téléphone, etc. Comme c'est une balise qui s'utilise seule, elle peut se terminer par la barre de fermeture intégrée, qui est facultative : `<input ... />`.

Elle possède comme attributs principaux :

- son nom `name="..."`, qui sert à identifier la donnée lors de l'envoi du formulaire ; ce nom sera sans accents ni espaces. Toutes les balises de saisie d'un formulaire auront ainsi un nom ;
- le type d'entrée, qui sera `type="text"` pour un texte en clair ③ et `type="password"` pour un mot de passe ④, lorsque les caractères saisis doivent être masqués par des points ou des étoiles ;
- éventuellement, un texte prédéfini à afficher dans la zone de saisie, qui sera indiqué par l'attribut `value="..."`.

Les zones de texte ③ et ④ s'écrivent ainsi, accompagnées de leurs étiquettes :

```
<label>Nom ou pseudo</label>
<input type="text" name="nom" /> ③
<label>Mot de passe</label>
<input type="password" name="motdepasse" /> ④
```

Zones de texte sur plusieurs lignes

Lorsque le contenu à saisir dans une zone de texte est susceptible de comporter plusieurs lignes, il faut utiliser la balise `<textarea>`, qui s'écrit pour la zone ⑥ de notre exemple :

```
<label>Votre message</label>
<textarea name="message">Tapez ici votre message
</textarea>
```

Contrairement à la balise `<input ...>` qui est seule et intègre sa barre de fermeture, il existe une balise de fermeture`</textarea>`.

Le texte initialement affiché est facultatif et s'écrit ici entre les balises `<textarea ...>` et `</textarea>`.

Boutons radio, à choix unique

Les boutons ronds, appelés *boutons radio*, sont utilisés lorsqu'il s'agit d'effectuer un choix exclusif dans une liste d'options. C'est la balise `<input>` qui est à nouveau utilisée, ici avec l'attribut `type="radio"`. L'exemple 7 de la figure 2-12 s'écrit ainsi :

```
<input type="radio" name="navig" value="très simple" />
<label>Très simple</label>
<input type="radio" name="navig" value="correcte" checked="checked" />
<label>Correcte</label>
<input type="radio" name="navig" value="compliquée" />
<label>Compliquée</label>
```

Penchons-nous sur le fonctionnement des boutons radio.

- Les balises `<input>` d'un même groupe de choix ont toutes le même attribut `name`. La valeur renvoyée sera l'attribut `value` du bouton qui aura été sélectionné.
- Pour proposer dans le formulaire une autre liste de choix à réponse unique, il suffira d'écrire des balises `<input>` de type `radio` avec un autre attribut `name`.
- Il est possible de cocher initialement une des options, en lui ajoutant l'attribut `checked="checked"`. L'attribut `checked` tout seul (sans le signe = ni la valeur `checked`) est admis en HTML 5.

Cases à cocher

Contrairement aux boutons radio, les cases à cocher permettent plusieurs choix ou aucun. C'est toujours la balise `<input>` qui est de service, cette fois avec l'attribut `type="checkbox"`. Les cases à cocher de notre exemple 8 s'obtiennent avec le code suivant :

```
<input name="instructif" type="checkbox" value="instructif" />
<label>Instructif</label>
<input name="esthetique" type="checkbox" value="esthétique" />
<label>Esthétique</label>
<input name="clair" type="checkbox" value="clair" />
<label>Clair</label>
<input name="pratique" type="checkbox" value="pratique" />
<label>Pratique</label>
```

Alors que le choix unique d'un bouton radio ne représentait qu'une seule donnée, les balises `<input>` de type `checkbox` possèdent des attributs `name="..."` différents : l'envoi du formulaire devra fournir l'état de chaque case à cocher.

Si la case a été cochée, la valeur renvoyée sera le contenu de l'attribut `value`. En l'absence de cet attribut, la valeur transmise est `on`, équivalente à `true` et qui vaut 1. Si la case n'est pas cochée, c'est la valeur `off` qui sera renvoyée, équivalente à `false` et qui vaut 0.

Listes déroulantes

Une liste déroulante permet un choix unique comme dans les exemples 5 et 9 et s'écrit à l'aide des balises `select` et `option`. L'ensemble de la liste est encadré par les balises `<select name="...>` et `</select>`, chaque ligne étant définie entre les balises `<option value="...>` et `</option>`.

Voici le code correspondant à la liste 9 de la figure 3-10 :

```
<label>Comment avez-vous connu ce site</label>
<select name="connaissance">
  <option value="presse">Presse</option>
  <option value="internet">Internet</option>
  <option value="autre">Amis ou autres</option>
</select>
```

COMPLÉMENT Liste à choix multiples

La balise `<select name="..." multiple>` autorise l'utilisateur à choisir plusieurs occurrences dans la liste, en appuyant sur la touche **Ctrl** du clavier. Dans ce cas, la liste n'est plus déroulante et affiche directement toutes les options les unes sous les autres. L'attribut `multiple` modifie donc à la fois les possibilités de choix et l'apparence de la balise `<select>`.

La liste de choix est envoyée par le formulaire sous la forme d'un tableau PHP, c'est pourquoi l'attribut `name` de la balise `<select>` doit être suivi de deux crochets, comme le montre l'exemple suivant, illustré par la figure 3-11 :

```
<select name="choixcouleurs[]" multiple size="5">
  <option value="bleu">le bleu</option>
  <option value="vert">le vert</option>
  <option value="rouge">le rouge</option>
  <option value="jaune">le jaune</option>
  <option value="blanc">le blanc</option>
</select>
```

L'attribut `size` permet de définir le nombre de lignes à afficher simultanément. Si la liste est plus grande que sa taille, une barre de défilement s'affiche.

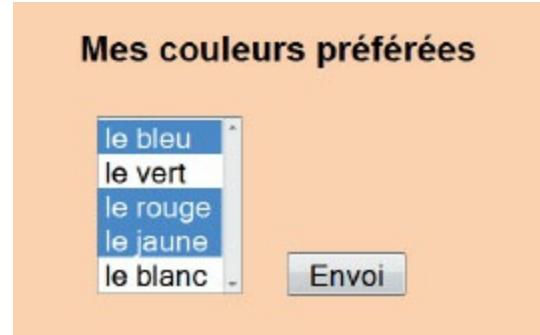


FIGURE 3-11 Choix multiple dans une liste, à l'aide d'une balise `<select>` possédant l'attribut `multiple`

Il est possible de regrouper par catégories les options d'une liste déroulante, comme c'est le cas dans l'exemple ⑤. Les options d'une catégorie sont regroupées entre deux balises `<optgroup label="...>` et `</optgroup>`; le titre de la catégorie est indiqué dans l'attribut `label` et ne peut pas être sélectionné.

L'exemple ⑤ de la figure 3-10 comprend deux groupes et s'écrit :

```
<label>Vous êtes</label>
<select name="qui">
  <optgroup label="Adulte">
    <option value="homme">Homme</option>
    <option value="femme">Femme</option>
  </optgroup>
  <optgroup label="Jeune">
    <option value="garçon">Garçon</option>
    <option value="fille">Fille</option>
  </optgroup>
</select>
```

Boutons d'effacement et d'envoi

Le *bouton d'effacement* réinitialise le formulaire à son affichage initial (les données saisies sont effacées, les textes et choix par défaut sont repris en compte).

Le *bouton d'envoi* transmet les données à la page dont le nom est indiqué dans l'attribut `action` de la balise de début de formulaire :

```
| <form method="post" action="envoi.php" > .
```

Ces deux boutons ⑩ terminent en général le formulaire. Ils s'écrivent avec la balise `input` et l'attribut `type="reset"` pour l'effacement, `type="submit"` pour l'envoi :

```
| <input type="reset" value="Effacer" />
| <input type="submit" value="Envoyer" />
```

L'attribut `value` a ici une fonction très particulière : il sert à définir le texte qui sera affiché sur le bouton.

UTILE Bouton Envoyer et touche Entrée

Le bouton d'envoi, tel qu'il est écrit ici, présente un inconvénient : l'appui sur la touche **Entrée** provoquera l'envoi du message (sauf si le curseur est situé dans une zone de texte à plusieurs lignes). Cet envoi risque d'être prématuré aux yeux du visiteur, lui qui pensait peut-être valider une ligne de texte.

Pour éviter ce genre de désagrément, il est possible d'utiliser un bouton général, grâce à une balise `<input type="button" ...>`, l'envoi s'effectuant par une action écrite en JavaScript. Ce bouton d'envoi amélioré s'écrit :

```
<input type="button" value="Envoyer" onclick="submit();"
```

Fichier d'envoi du formulaire

Nous avons fait le tour des balises de formulaire, jusqu'au bouton d'envoi. Ce dernier fait appel à une page écrite en PHP, mentionnée dans la balise de début du formulaire :

```
<form method="post" action="envoi.php">
```

Nous allons adopter la forme la plus simple possible de ce fichier `envoi.php`, pour n'utiliser que des notions très accessibles du PHP, langage qui s'exécute sur le serveur hébergeant le site Internet.

Ce fichier doit effectuer les opérations suivantes.

1. Récupérer les données envoyées par le formulaire par la méthode `post`.
2. Les associer à des titres pour constituer un message lisible.
3. Envoyer le message.
4. Afficher un texte de confirmation pour le visiteur, si l'envoi s'est passé normalement, un message d'erreur dans le cas contraire.

ATTENTION Conditions techniques nécessaires

L'envoi d'un formulaire par un fichier PHP ne peut s'effectuer qu'à deux conditions.

- L'hébergement sur lequel vous avez placé votre site doit être capable d'exécuter un programme en PHP.
- La fonction `mail` du PHP, qui servira à l'envoi, doit être autorisée.

La plupart des hébergements proposent ces fonctions, mais certaines offres gratuites ou très basiques peuvent être limitées.

Pour ne pas avoir à écrire trop de lignes identiques et clarifier notre fichier, nous allons étudier l'envoi de la première partie du formulaire de la figure 3-10, soit les balises ayant pour attributs `name="nom"` (saisie de **Votre nom** ou **Pseudo**), `name="qui"` (liste déroulante **Vous êtes**) et `name="motdepasse"` (saisie intitulée **Mot de passe**).

Contenu du fichier `envoi.php`

```

<?php
    // Récupération des données du formulaire ①
    $nom = $_POST["nom"];
    $qui = $_POST["qui"];
    $passe = $_POST["motdepasse"];

    // Texte à envoyer ②
    $texte = "Nom : $nom\n";
    $texte = $texte . "Qui êtes-vous : $qui\n";
    $texte = $texte . "Mot de passe :\n" . $passe;
    $texte = stripslashes($texte);

    // Destinataire et objet du message ③
    $destinataire = "toto@net.fr";
    $objet = "Formulaire du site Internet";

    // En-tête masqué (codage des caractères)
    $headers = "Content-type: text/plain; charset=utf-8"; ④

    // Envoi du message, puis confirmation sur la page
    $envoi_bon = mail($destinataire, $objet, $texte, $headers); ⑤
    if ($envoi_bon) { echo "Envoi OK"; }
    else { echo "Erreur"; } ⑥
?

```

Fonctionnement du fichier envoi.php

Il suffit de placer ce fichier texte, enregistré sous le nom `envoi.php`, dans le même dossier que la page contenant le formulaire, pour correspondre à l'attribut `action="envoi.php"` de la balise `<form>`.

Quelques points de repère :

- le code PHP commence par `<?php` et se termine par `?>` : ce sont ici la première et la dernière lignes de notre fichier ;
- les lignes de commentaires commencent par `//` et sont bien sûr facultatives ;
- chaque instruction se termine par un point-virgule ;
- tous les noms de variables commencent par le symbole `$` ;
- dans la constitution des textes du message, le caractère `\n` correspond à un saut de ligne (`\n\n` pour deux sauts de ligne), et le point sert à « coller » (*concaténer* en langage informatique) deux morceaux de texte ou un texte avec une variable contenant du texte.

La récupération des données ① s'effectue selon une syntaxe bien précise, par exemple `$_POST["nom"]` pour la balise d'attribut `name="nom"`. Ces données sont stockées dans des variables `$nom`, `$qui` et `$passe` (il est souvent plus simple de leur donner les mêmes noms que dans le formulaire, mais ce n'est pas obligatoire).

La variable `$texte` sert à mémoriser le message à envoyer. Pour plus de clarté, sa construction ② s'effectue sur plusieurs lignes, à l'aide du point pour ajouter du texte à cette variable. La fonction `stripslashes($texte)` sert à enlever le signe `\` qui est ajouté par le PHP avant chaque apostrophe ou guillemet saisi dans le formulaire (par exemple, si le nom saisi est « d'Alembert », la variable `$nom` contient « d\Alembert »).

Les variables `$destinataire` et `$objet` ③ sont utilisées, comme leur nom l'indique, pour mémoriser à la fois l'adresse de messagerie à laquelle sera envoyée le message et le titre de celui-ci.

La variable d'en-tête `$headers` ④ peut contenir beaucoup plus d'informations. Il faut au minimum indiquer le type de codage de la page du formulaire, de façon à ce que les lettres accentuées s'affichent correctement. Le codage universel UTF-8 est conseillé, car il est utilisable pour toutes les langues.

La fonction `mail` ⑤ utilise les différents éléments que nous avons préparés pour envoyer le message. Le résultat est ici stocké dans la variable `$envoi_bon` qui vaudra `true` (vrai) si l'envoi a réussi, `false` (faux) sinon.

Un test ⑥ sur la variable `$envoi_bon` permet d'afficher un message à l'aide de la fonction `echo "texte à afficher";`.

Pour envoyer le reste du formulaire, il suffit d'ajouter des lignes aux étapes ① et ②. De même, en adaptant ces deux étapes à votre formulaire et aux noms que vous avez utilisés, vous pourrez créer votre propre fichier d'envoi.

Contrôle plus précis des formulaires

Après avoir découvert les éléments de base qui constituent un formulaire, utilisons les atouts du HTML 5, qui introduit de nombreux attributs supplémentaires pour mieux définir les formulaires et vérifier les types de données saisies.

Balise form

Pour la balise `<form>`, l'attribut `target` permet de définir dans quelle fenêtre s'ouvrira la page définie dans l'attribut `action` du formulaire, lors de sa validation. Ses valeurs possibles sont les mêmes que celles décrites pour la balise `<a>` :

- `_blank` (nouvelle fenêtre) ;
- `_self` (même fenêtre) ;
- `_parent` et `_top` (fenêtre parente de niveau précédent ou de plus haut niveau, lorsque le formulaire se trouve dans une fenêtre de type `iframe`).

Elle accepte également l'attribut `autocomplete`, qui permet d'autoriser la complétion automatique par le navigateur de certains champs de formulaire, en fonction des options de mémorisation choisies dans le logiciel : `autocomplete="on"` (valeur par défaut) pour autoriser ce fonctionnement, `autocomplete="off"` pour le désactiver.

Enfin, il est possible de désactiver la vérification des champs de saisie lors de l'envoi du formulaire, grâce à l'attribut `novalidate`. Cela concerne la vérification des balises `<input>` lorsqu'elles sont associées à certains types de données ; nous en parlerons un peu plus loin.

Contrôle des balises input

Dans un formulaire, la balise `<input>` peut prendre plusieurs formes en fonction de son type : ligne de saisie, case à cocher ou d'option, bouton d'action. Le HTML 5 ajoute de nombreux types plus précis, notamment pour les zones de saisie ; il introduit aussi des attributs supplémentaires. Le tout permettant une vérification plus fine de la saisie.

Nouveaux types d'entrées

L'attribut `type="..."` détermine à la fois l'apparence et la fonction de la balise `<input>`. Les valeurs courantes sont conservées.

- `text` correspond à une ligne de saisie, `password` à un mot de passe (caractères masqués par des étoiles ou des points), `hidden` sert à masquer cet élément.
- `file` permet à l'utilisateur de fournir le nom d'un fichier, en s'aidant du bouton **Parcourir** que le navigateur affiche automatiquement.
- `checkbox` correspond à une case à cocher et `radio` à un choix exclusif, entre des boutons radio ayant le même attribut `name`.
- D'autres valeurs servent à l'affichage de boutons, comme `submit` (envoi du formulaire), `reset` (remise à zéro), `button` (affichage d'un bouton dont l'action sera programmée en JavaScript) et `image` pour créer un bouton dont l'image est fournie par l'attribut `src`.

De nouveaux types permettent de préciser le contenu d'une zone de saisie qui aurait été auparavant de type texte, et dont la validité sera ainsi vérifiée avant l'envoi du formulaire (sauf si la balise `<form>` contient l'attribut `novalidate="novalidate"`) :

- `email` pour une adresse courriel, `tel` si c'est un numéro de téléphone ;
- `number` pour entrer un nombre, ainsi que `range` qui limite ce nombre à une plage de valeurs délimitées par les attributs `min` et `max` ;
- `url` servant à la saisie d'une adresse web, `color` permettant de fournir un code de couleur hexadécimal comme « #a6b8ff », par un clic dans une palette polychrome proposée par le navigateur ;
- `date` dont la valeur pourra être choisie par un clic dans un calendrier

qu'affichera le navigateur ou fournie suivant le format *année –mois – jour* `aaaa-mm-jj` (par exemple `1515-09-14` pour la célèbre victoire de Marignan) ;

- `time` pour donner une heure (heures, minutes, secondes et centièmes de secondes éventuels sous la forme `hh:mm:ss.ss`, comme `8:30` pour `8 h 30` par exemple) ;
- `datetime` OU `datetime-local` si jour et heure sont fournis, suivant le format `aaaa-mm-jjT hh:mm:ss.ss`, la lettre `T` sans espace séparant le format de date de celui de l'heure, l'ensemble étant suivi pour `datetime` du décalage par rapport à l'heure universelle UTC (soit la lettre majuscule `z` pour zéro ou un décalage comme « `+01:00` » ou « `- 08:00` », par exemple `1789-07-14T17:00Z`) ;
- `month` correspondant à un numéro de mois (entre 1 et 12) et `week` à un numéro de semaine (compris entre 1 et 52 ou 53, suivant les années) ;
- `search` pour afficher une zone de texte destinée à entrer un ou plusieurs mots-clé(s) à rechercher.

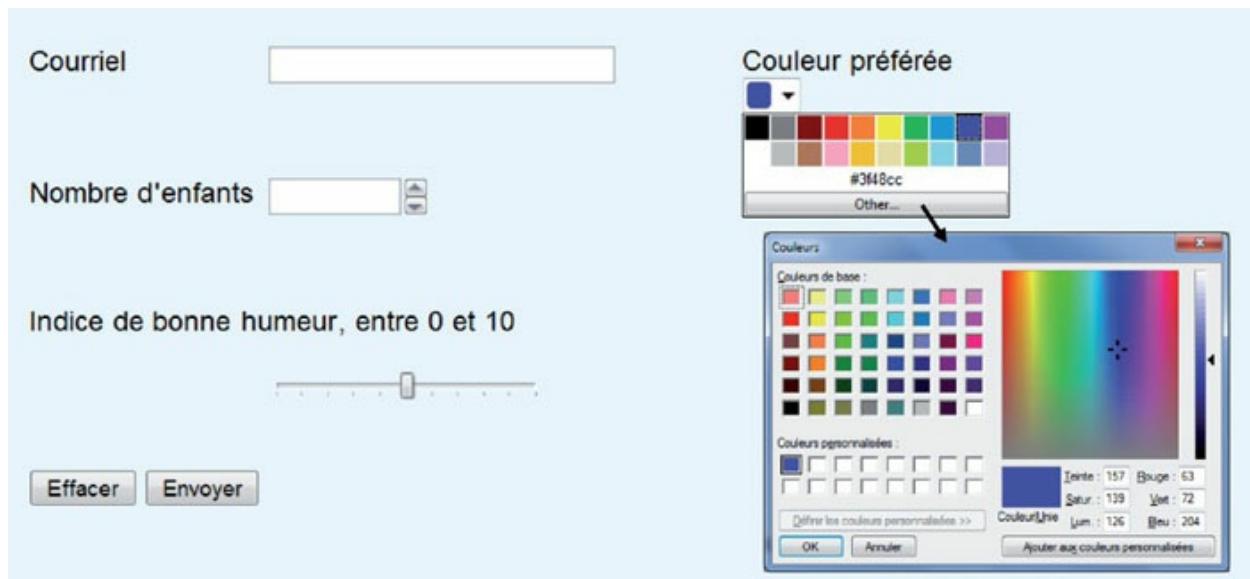


FIGURE 3–12 Affichage par le navigateur Opera des balises `<input>` de type `number` (nombre d'enfants), `range` pour une plage de valeurs possibles (indice de bonne humeur) et `color` pour un code de couleur défini par un choix dans une palette.

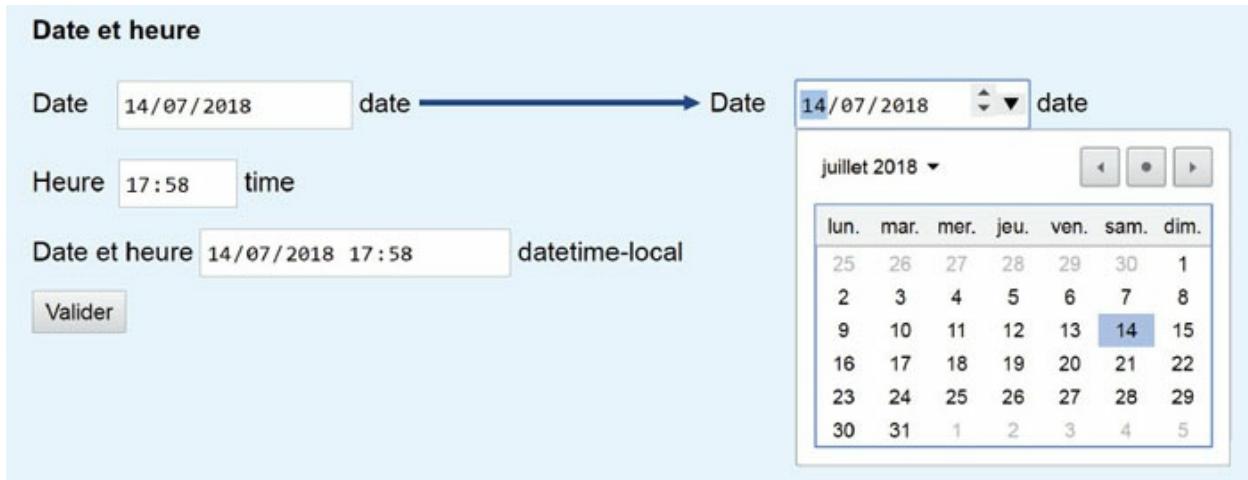


FIGURE 3-13 Lorsqu'il affiche les balises `<input>` de types `date` et `heure`, le navigateur Chrome propose un calendrier pour choisir la date et des flèches pour régler l'heure.

Attributs des balises input

Les balises `input` conservent leurs attributs classiques comme `name` (nom de la donnée transmise par le formulaire), `value` (valeur à afficher dans la zone de texte ou sur le bouton), `disabled` ou `readonly` (désactivé mais apparent – souvent grisé – ou en *lecture seule*), `maxlength` (nombre maximum de caractères pouvant être saisis) et `checked` (case à cocher ou bouton radio activé). Toutes les balises `<input>` sont concernées par deux nouveaux attributs.

- `form="..."` permet de rattacher cette balise à un formulaire donné en utilisant son identifiant, ce qui donne la possibilité de la placer en dehors des balises `<form id="...">` et `</form>` qui normalement délimitent le formulaire.
- `autofocus` sert à sélectionner l'élément concerné dès le chargement de la page (le curseur se place dans la zone de saisie ou, s'il s'agit d'un bouton, il peut être actionné directement par la touche **Entrée**).

Les autres attributs concernant `<input>` étant nombreux, nous allons les étudier par catégories, puisque cette balise `<input>` peut prendre des formes très différentes.

Balises input utilisées pour la saisie

Les balises `<input>` servant à la saisie de données pourront utiliser les

nouveaux attributs suivants.

- `required="required"` rend la saisie obligatoire.
- `size` définit la taille de la zone de saisie affichée, exprimée en nombre de caractères.
- `placeholder` fournit un texte d'aide, qui s'affiche en gris dans la zone de saisie et qui disparaît dès que l'utilisateur accède à cette zone (cet attribut remplace le classique texte d'aide affecté par l'attribut `value` et effacé par un programme écrit en JavaScript).
- `autocomplete="on"` permet la complétion automatique de la saisie par le navigateur, en fonction des options de mémorisation choisies dans ce logiciel ; ce mode est désactivé par `autocomplete="off"`.
- `pattern` autorise une plage de valeurs possibles pour la saisie, comme `pattern="[0-100]"` (nombre entre 0 et 100), `pattern="[A-Z]{3}"` (trois lettres en majuscules), `pattern="[A-z]{5}"` (cinq lettres en majuscules ou minuscules).

Trois attributs définissent des restrictions pour la saisie de *nombres* (lorsque l'attribut `type` est égal à `number`, à `range` ou à une valeur de la catégorie `dates` et `heures`) :

- `min` et `max` définissent des valeurs minimale et maximale pour la valeur à fournir.
- `step` précise si le nombre à saisir doit être multiple d'un nombre donné, par exemple `step="5"` pour n'autoriser que des nombres positifs ou négatifs se terminant par 0 ou 5.

Par ailleurs, lorsque le type de balise `<input>` est `file` (fichier) ou `email`, l'attribut `multiple="multiple"` autorise la sélection de plusieurs fichiers ou adresses e-mail en une seule fois.

Notons que la balise `<input type="file">` conserve l'attribut `accept` qui permet de filtrer la liste des fichiers proposés par le bouton **Parcourir**, par exemple `accept="image/*"` pour n'afficher que les images, `accept="audio/*"` et `accept="video/*"` pour sélectionner respectivement les fichiers de types son et vidéo.

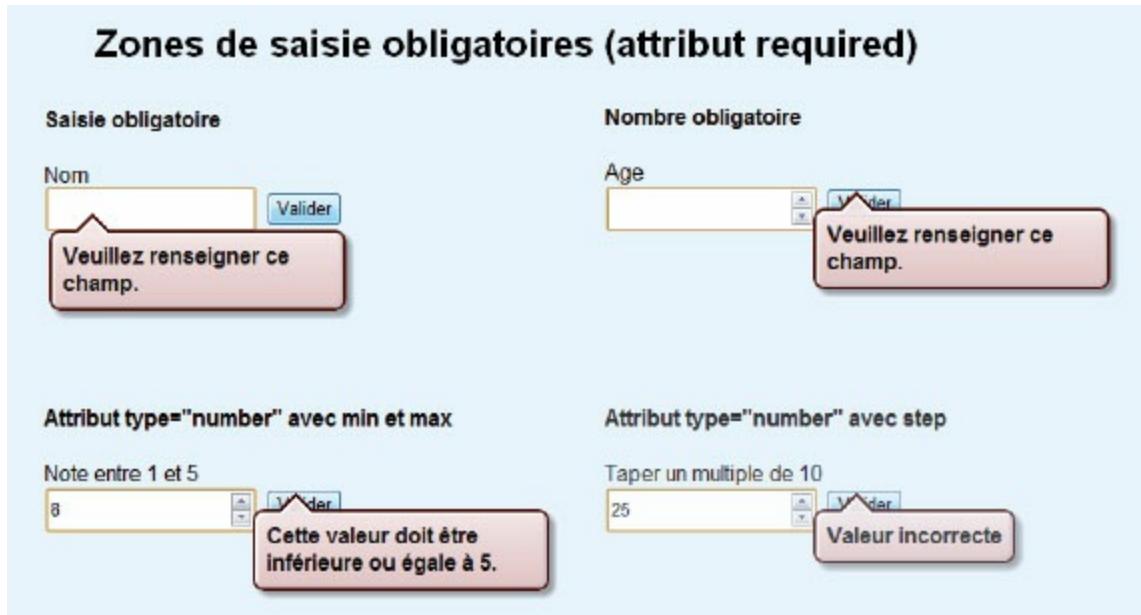


FIGURE 3-14 Messages du navigateur Chrome indiquant, dans différents cas, l'erreur de saisie qui empêche l'envoi du formulaire. Cela signifie que ces éléments sont obligatoires et qu'ils contiennent donc l'attribut required.

Balises input en forme de bouton

Une balise `<input>` prend la forme d'un bouton quand elle est de type `submit`, `reset`, `button` ou `image`. Lorsqu'ils servent à l'envoi du formulaire (type `submit`, types `button` et `image` s'ils sont utilisés pour l'envoi des données), ces boutons acceptent de nouveaux attributs pour modifier la façon dont le formulaire est transmis, initialement définie par la balise `<form>`.

- `formaction` permet de modifier l'action initialement prévue lors de la validation du formulaire, c'est-à-dire la page appelée par l'attribut `action` de la balise `<form>` .
- `formmethod` sert à redéfinir la méthode de transmission des données, `get` pour inscrire celles-ci dans la barre d'adresse ou `post` pour un envoi masqué.
- `formenctype`, moins utilisé, donne le type d'encodage des données, indiqué de façon facultative dans la balise `<form>` (par défaut `application/x-www-form-urlencoded` si cet attribut n'est pas spécifié, `multipart/form-data` quand il y a un fichier à envoyer, `text/plain` pour ne conserver que le texte et les espaces qui sont convertis en signes +).
- `formnovalidate="formnovalidate"` désactive la validation des données par le

formulaire, si l'attribut `novalidate` de la balise `<form>` n'a pas été précisé.

- `formtarget` indique dans quelle fenêtre doit s'ouvrir la page d'action (appelée par le formulaire lors de sa validation) et correspond à l'attribut `target` de la balise `<form>`, les valeurs possibles étant les mêmes, soit `_self` (même fenêtre) `_blank` (nouvelle fenêtre), `_parent` ou `_top` (fenêtre parente immédiate ou celle de plus haut niveau, en cas de fenêtres imbriquées).

À NOTER Bouton créé avec la balise `<button>`

Pour créer un bouton dont l'action sera déterminée par une commande JavaScript, la balise `<button>` est une alternative à la balise `<input type="button">`. Elle admet trois valeurs possibles pour l'attribut `type` : `button`, `reset` et `submit`, avec la même signification que pour la balise `<input>`. Elle accepte également les mêmes attributs que la balise `<input>` de type `button`, y compris les nouveautés apportées par le HTML 5.

La différence d'écriture entre cette balise et `<input>` réside dans le fait qu'elle possède une balise de fermeture : ainsi, le texte affiché sur le bouton n'est pas écrit dans un attribut `value`, mais entre les balises `<button>` et `</button>`. Cette balise `<button>` peut également être utilisée en dehors d'un formulaire.

Restent quelques attributs qui concernent uniquement les balises `<input>` de type `image`.

- Sont conservés `src`, qui fournit le nom de l'image, et `alt`, qui propose un texte alternatif à l'image.
- Deux nouveaux attributs permettent de définir les dimensions des images, `width` pour la largeur et `height` pour la hauteur, exprimées par des nombres en pixels comme `150`.

Le W3C propose (en anglais) deux tableaux récapitulatifs qui donnent la liste des différents types de balises `<input>` et les attributs qui sont utilisables pour chacune d'entre elles :

► <https://www.w3.org/wiki/HTML/Elements/input>

Nouvelles balises de formulaire

Bien que l'essentiel des apports du HTML 5 pour les formulaires soit un ensemble d'attributs supplémentaires, quelques nouvelles balises de formulaire viennent enrichir le panel existant.

Listes d'options modifiables

Les listes déroulantes `<select>` n'acceptent qu'une seule valeur parmi les options proposées, qui constituent alors les seules entrées possibles. Il manquait la liberté de choisir une des options prévues ou de saisir une donnée différente.

C'est ce que propose la balise `<datalist>`, associée à une balise `<input>` qui délimite la saisie et à des balises `<option>` fournissant les choix proposés. Son utilisation se comprend mieux à partir de l'exemple suivant, dont le résultat se trouve sur la figure 3-15 :

```
<label>Animal préféré</label>
<input list="animal" /> ①
<datalist id="animal"> ②
  <option value="âne">
  <option value="chat">
  <option value="cheval">
  <option value="chèvre" disabled="disabled">
  <option value="chien">
  <option value="lapin">
</datalist>
```

La zone de saisie `<input>` ① possède un attribut `list="..."`, dont la valeur est l'identifiant de la balise `<datalist>` ② à laquelle elle est reliée. Entre les balises `<datalist>` et `</datalist>` se trouvent les options qui fournissent, grâce à leur attribut `value`, les différents choix.

L'attribut `disabled="disabled"` appliqué à une balise `<option>` revient à la faire momentanément disparaître de la liste.

Lorsque l'utilisateur saisit quelques lettres dans la zone de texte, les options commençant par ces lettres sont affichées, comme le montre la figure 3-15. Il peut alors choisir une ligne de cette liste ou taper un texte qui n'y figure pas.

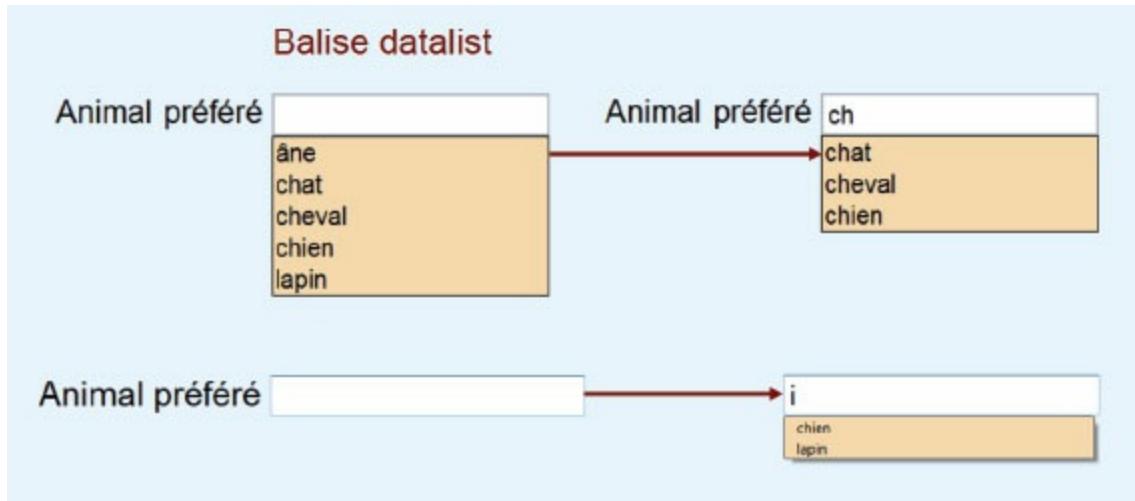


FIGURE 3–15 Différentes interprétations de la balise `datalist`, par Opera en haut et par Firefox en bas. Dans les deux cas, une liste de termes est suggérée à partir des lettres saisies, mais le choix affiché n'est pas limitatif.

À NOTER Pas de balise de fermeture `</option>`

Notons que la balise de fermeture `</option>` n'est pas employée avec `<datalist>`. La syntaxe diffère donc de celle employée avec la balise `<select>` pour la création de listes déroulantes :

```
<option value="chat">chat</option>
```

En effet, l'utilisation avec `<datalist>` ne prend pas en compte le texte situé entre les balises `<option>` et `</option>` : pour l'affichage des termes suggérés, seule est utilisée la valeur de l'attribut `value` inclus dans la balise `<option>`.

Certains navigateurs, tel Firefox, élargissent le concept et proposent toutes les occurrences qui contiennent la suite de lettres saisies, même si cela ne correspond pas au début du mot.

Affichage du résultat d'un calcul

Dans un formulaire, certaines zones affichent le résultat d'un calcul lié aux autres éléments saisis. La balise spécifique `<output>` a été créée en HTML 5 pour assurer cette fonction, le calcul étant toujours assuré par une fonction écrite en JavaScript. Voici un exemple simple, correspondant à la figure 3-16, qui multiplie deux nombres.

```
<h4>Balise output</h4>
<label>Nombre 1 </label>
<input type="number" name="nb1" />
<label> multiplié par Nombre 2 </label>
<input type="number" name="nb2" />
<label> est égal à </label>
<output name="calcul" onforminput="value=nb1.value * nb2.value;">
</output>
```

Les deux nombres à multiplier sont repérés par leur attribut `name`, ici respectivement "nb1" et "nb2". La balise `<output>` utilise l'attribut `onforminput` pour que le calcul en JavaScript de sa valeur se déclenche dès qu'une modification est effectuée dans le formulaire.

Table de multiplication

Le nombre multiplié par donne : **48**

FIGURE 3–16 Le résultat d'un calcul, affiché sur Opera à l'aide de la balise `<output>` associée à un calcul en JavaScript.

Si un texte figure entre les balises d'ouverture `<output>` et de fermeture `</output>`, il apparaît lors du chargement de la page, puis il est remplacé par le résultat du calcul dès que l'utilisateur commence à entrer des données dans le formulaire, même s'il n'a pas atteint les zones de saisie concernées par le calcul.

Nouveaux attributs pour les éléments de formulaire

Après avoir énuméré les nombreux aménagements apportés à la balise `<input>`, étudions les attributs associés aux autres balises de formulaire.

Il existe des attributs communs aux principales balises de formulaire : certains existaient déjà en XHTML 1, comme `name` pour repérer les données envoyées et `disabled` pour désactiver un élément tout en le laissant apparent, tandis que d'autres sont nouveaux en HTML 5.

- `form="..."` sert à rattacher l'élément concerné à un formulaire donné, en fournissant son identifiant. Cela permet de placer cet élément à l'extérieur des balises `<form id="..."> ... </form>` qui délimitent le formulaire en question.
- `autofocus="autofocus"` permet, lors du chargement de la page, de placer le curseur sur l'élément qui possède cet attribut.
- `required="required"` rend obligatoire la saisie de la donnée avant l'envoi du formulaire.

Le tableau qui suit résume l'utilisation des principaux attributs HTML 5 pour les différentes balises de formulaire.

TABLEAU 3–2 Attributs associés aux balises de formulaire

Attributs	form	name	autofocus	required	disabled
<code><label></code>	X				
<code><input></code>	X	X	X	X	X
<code><textarea></code>		X	X	X	X
<code><select></code>	X	X	X	X	X
<code><option></code>					X
<code><optgroup></code>					X
<code><fieldset></code>	X	X			X
<code><output></code>	X	X			X
<code><button></code>	X	X	X		X

D'autres attributs HTML 5 se rapportent plus précisément à un ou deux types de balise de formulaire, nous allons maintenant en faire le tour.

Rattachement à un autre élément

L’attribut `for="..."`, appliqué à une étiquette `<label>` ou à une valeur calculée `<output>`, permet de créer un lien logique avec une balise de formulaire quelconque, en utilisant comme valeur l’identifiant de celle-ci.

Exemple d’utilisation de l’attribut `for`

```
| <label for="nom">Nom de famille : </label>
| <input type="text" name="nom" id="nom" />
```

Bien que cela n’apporte pas de modification à la mise en forme, c’est un atout pour l’accessibilité des pages aux personnes souffrant d’un handicap visuel et ne pouvant pas distinguer les liens entre éléments et qui sont généralement mis en évidence dans la mise en page. Cette fonction facilite également la maintenance pour le développeur.

Zones de saisie à plusieurs lignes

Outre les attributs communs anciens et nouveaux déjà évoqués, la balise `<textarea>`, qui permet de saisir plusieurs lignes de texte, conserve ses attributs initiaux : `readonly` (lecture seule, pas de modification possible de son contenu), ainsi que `rows` et `cols` (taille de la zone de saisie en nombre de lignes et de colonnes, à définir de préférence dans une feuille de styles). Viennent s’y adjoindre quelques autres attributs en HTML 5 :

- `placeholder="..."` sert à inscrire en grisé un texte d’aide, disparaissant quand l’utilisateur accède à cette zone ;
- `wrap` peut prendre deux valeurs, `wrap="soft"` (valeur par défaut) si les sauts de ligne ne sont pas pris en compte ou `wrap="hard"` si ces derniers doivent être transmis par le formulaire (la présence de l’attribut `cols` étant obligatoire dans ce dernier cas) ;
- `dirname` sera rarement utilisé, donnant le nom (attribut `name`) d’une balise `<input>` contenant le sens d’écriture dans la balise `<textarea>` sous forme de texte (valeur par défaut `ltr` pour *left to right*, de gauche à droite, ou `rtl` pour l’autre sens) ;
- l’attribut `maxlength`, déjà existant pour la balise `<input>`, est nouveau pour la balise `<textarea>` et fixe le nombre maximum de caractères qui pourront être saisis dans cette zone de texte.

Attribut placeholder

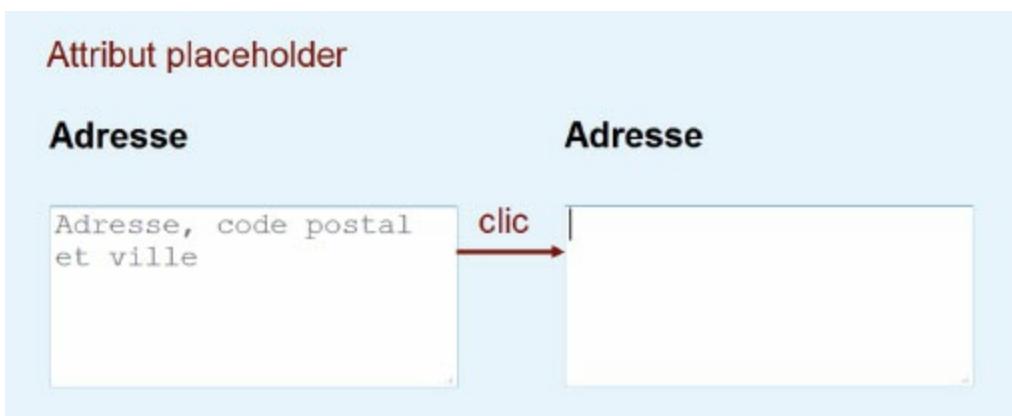


FIGURE 3–17 Dans la zone de texte, l'attribut *placeholder* permet d'écrire en gris un texte d'aide, qui disparaît automatiquement dès le début de la saisie.

À utiliser progressivement

Toutes ces nouvelles propriétés et ces attributs supplémentaires, introduits par le HTML 5, vont clarifier nos pages web et en faciliter la programmation. La liste complète des balises, qui constitue la norme officielle HTML 5, est disponible sur le site du W3C, à cette adresse :

- ▶ <https://www.w3.org/wiki/HTML/Elements>

À NOTER Tests et validation du HTML5

Certains sites proposent des exemples utilisant les principales nouvelles fonctions du HTML 5, en particulier :

- ▶ <https://html5demos.com>

Ces démonstrations permettent de tester l'évolution des navigateurs dans la prise en compte de cette nouvelle norme. Par ailleurs, le W3C propose une validation des différentes normes HTML à l'adresse suivante :

- ▶ <https://validator.w3.org>

L'introduction dans nos pages des nouveautés HTML 5 un peu spécifiques, comme certaines que nous venons de parcourir, devra être progressive, avec toujours une alternative pour les anciens navigateurs, de façon à ce que le site reste lisible et utilisable pour l'ensemble des visiteurs qui le consulteront.

Il nous reste à apprendre comment mettre en forme le contenu de nos pages. Il s'agira de savoir, dans un premier temps, où et comment écrire des règles de styles CSS, puis de quelle façon celles-ci pourront être appliquées aux différents éléments de la page. Le chapitre qui suit va nous permettre de comprendre ces notions.

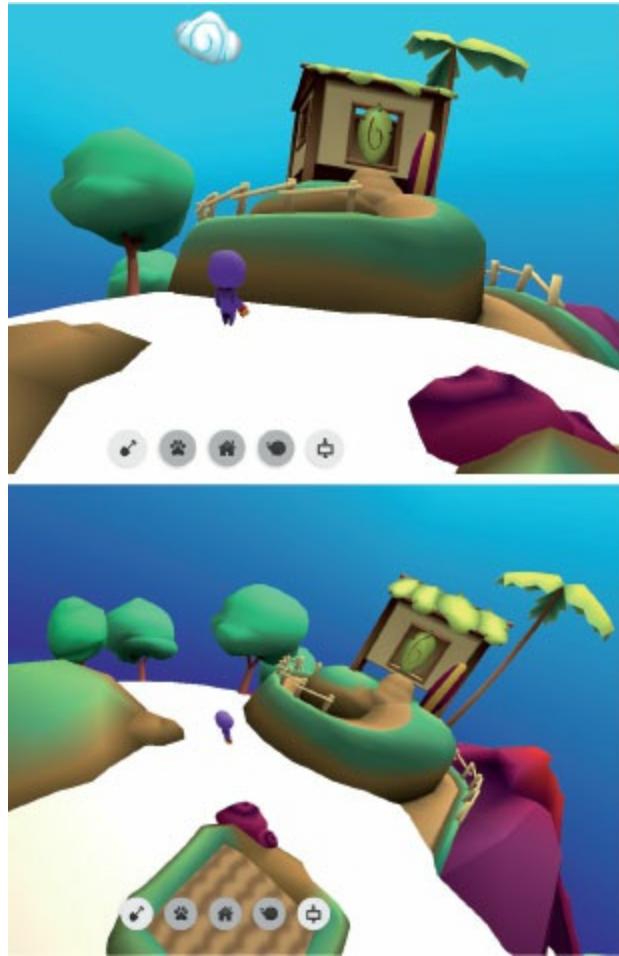
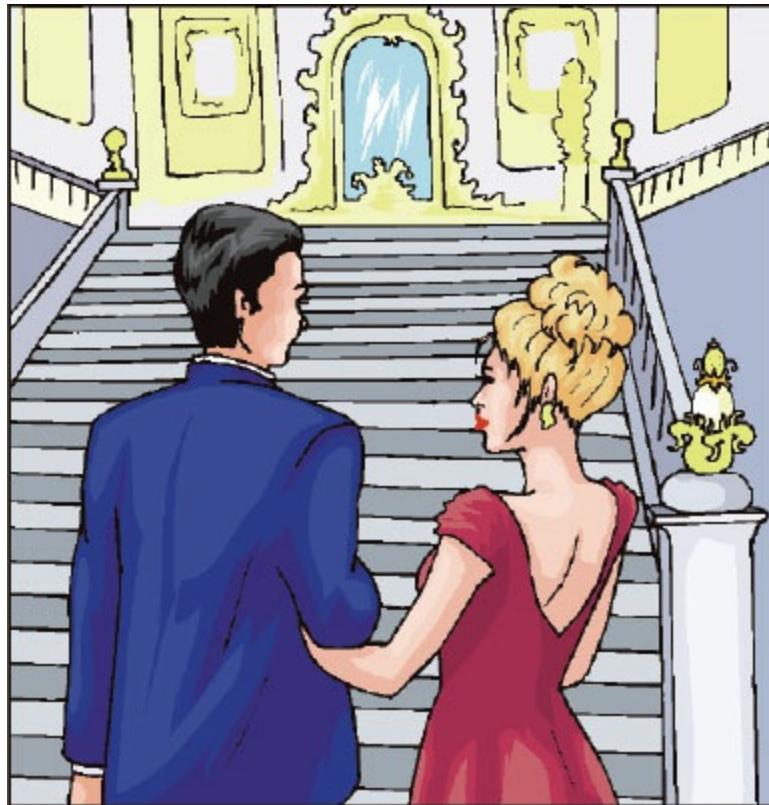


FIGURE 3–18 Navigation interactive à l'intérieur d'un dessin en 3D, grâce à la balise canvas (démonstration disponible sur cette page : <https://collinhover.github.io/kaiopua/>)

chapitre
4

Écriture des feuilles de styles



Comment écrire des règles de style CSS ? Où et dans quel ordre écrire ces règles ? De quelle façon attribuer une propriété à un élément donné de la page web ?

SOMMAIRE

- ▶ **Définition d'une règle de style**
- ▶ **Feuille de styles interne**
- ▶ **Feuille de styles externe**
- ▶ **Styles en ligne**
- ▶ **Sélecteurs de style**
- ▶ **Ordre de priorité des styles**
- ▶ **Valeurs, tailles et couleurs**
- ▶ **Exemple de page avec feuille de styles interne**

Nous voilà prêts à aborder les feuilles de styles proprement dites. Organisons notre démarche, en commençant par nous poser les questions de base : comment, où et dans quel ordre écrire ces définitions qui, une fois réunies, formeront une feuille de styles ?

Pour chaque règle de mise en forme, il faudra d'abord définir les éléments concernés dans la page web, puis les propriétés à leur attribuer. Les principes que nous allons découvrir ici s'appliquent à toutes les normes actuelles, aux CSS 2 comme aux CSS 3.

Définition d'une règle de style

Voici comment sélectionner un élément de la page et lui attribuer une propriété de mise en forme.

Principe

Une règle de style comprend :

- un **sélecteur** : il s'agit des balises concernées par cette règle ;
- un **bloc de déclarations** : il indique les propriétés qui seront attribuées à ces balises.

Chaque déclaration est du type `propriété: valeur;.`

Exemple de règle de style

La règle de la figure 4–1 indique que les titres de niveau 3 (encadrés par `<h3>...</h3>`) s’afficheront en *italique* et en *Arial* (ou dans une police générique *sans-serif* si la police Arial est absente).

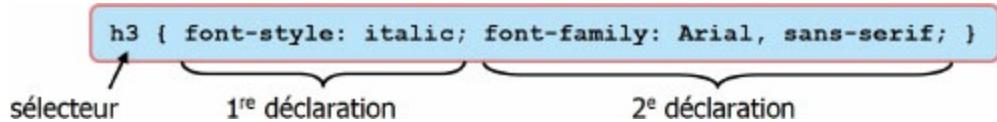


FIGURE 4–1 Exemple de règle de style

Cette règle comprend :

- le sélecteur (`h3`);
- deux déclarations, donc deux propriétés à attribuer aux titres de niveau 3 de la page.

À NOTER Écriture d’une règle de style

- Chaque déclaration se termine par un point-virgule.
- Une règle peut s’écrire sur plusieurs lignes :

```
h3 {  
    font-style: italic;  
    font-family: Arial, sans-serif;  
}
```

Commentaires

Il est utile de commenter abondamment les feuilles de styles, pour s'y retrouver plus tard lorsqu'il s'agira d'apporter des modifications. Il suffit de placer les commentaires entre les signes /* et */ :

```
/* Voici un commentaire */  
/* Et puis un autre, mais sur plusieurs lignes */
```

Emplacement des styles

Les règles de styles peuvent se trouver :

- dans le code HTML, comme attributs de balises : ce sont des styles en ligne (utilisation déconseillée - voir plus loin) ;
- dans l'en-tête de la page web : feuille de styles interne ;
- ou dans un fichier distinct : feuille de styles externe, à appeler dans l'en-tête de la page web.

Feuille de styles interne

Lorsque les règles de styles sont regroupées dans l'en-tête de la page web, elles constituent une feuille de styles interne.

Les styles sont écrits entre les balises `<head>` et `</head>`, à l'intérieur d'une balise `<style>` :

```
<head>
...
<style>
...règles de styles ici...
</style>
...
</head>
```

À NOTER Ancienne déclaration d'une feuille de styles interne

- La norme XHTML utilisait un attribut `type` dans la balise `<style>`, qui n'est plus nécessaire en HTML 5 :

```
<style type="text/css">.
```

- De même, vous rencontrerez peut-être des balises de commentaires HTML imbriquées dans la balise `<style>`. Elles servaient à masquer les CSS aux très anciens navigateurs qui ne les reconnaissaient pas, mais elles n'ont plus d'utilité à présent :

```
<style type="text/css">
<!--
... règles de styles ici ...
-->
</style>
```

Feuille de styles externe

Lorsque des règles de styles sont applicables à plusieurs pages web, il est intéressant de les écrire dans un fichier à part. Cette feuille de styles externe est appelée par chacune des pages concernées. Elle garantit l'unité graphique du site et facilite les modifications.

Une feuille de styles externe est un fichier texte d'extension .css contenant l'ensemble des règles définies, sans la balise <style>.

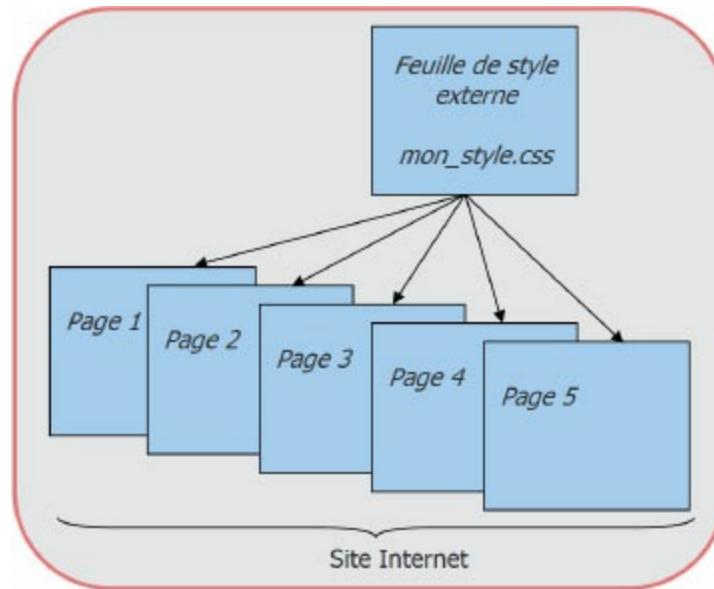


FIGURE 4–2 Une feuille de styles externe garantit une mise en forme homogène pour l'ensemble du site web.

Pour que cette feuille de styles soit prise en compte dans une page web, il suffit de l'appeler dans l'en-tête, en utilisant une des deux méthodes suivantes (dans l'exemple qui suit, la feuille de styles s'appelle `mon_style.css` et se trouve dans le même dossier que la page web) :

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

ou

```
<head>
...
<style>
@import url("mon_style.css");
</style>
...
</head>
```

La balise `<link>` était autrefois utilisée avec un attribut `type`, indiquant que le fichier appelé était de type texte et contenait des styles CSS. Vous rencontrerez donc parfois cette forme, un peu plus ancienne mais qui fonctionne également :

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

À NOTER Méthode `@import`

La règle `@import` peut être utilisée dans une feuille de styles externe, ce qui permet d'importer une première feuille de styles à l'intérieur d'une deuxième. Dans ce cas, elle doit figurer au tout début de la feuille.

Styles en ligne

Dans le corps de la page HTML (entre `<body>` et `</body>`), il est possible de préciser des styles qui prévaudront sur ceux précédemment déclarés dans la feuille de styles.

Exemple d'un titre de niveau 2 qui doit être centré et écrit en rouge :

```
<h2 style="text-align: center; color: red;">  
    ...Titre...  
</h2>
```

Cette méthode est à éviter autant que possible, car elle revient à mélanger à nouveau le contenu et la mise en forme, comme aux débuts d'Internet :

- ces ajouts alourdissent le code, nous privant de la clarté et de l'homogénéité apportées par les feuilles de styles ;
- la maintenance des pages devient plus délicate.

Il est préférable d'identifier la balise concernée à l'aide d'un nom (ce sera une *classe* ou un *identifiant*, nous les aborderons d'ici peu), ce qui permettra de lui attribuer une règle de style spécifique.

Sélecteurs de styles

Dans une règle de style, le choix du sélecteur est extrêmement important : il indique les balises concernées par la mise en forme qui suit.

Si des balises de même type doivent avoir différentes mises en forme, elles seront identifiées par des noms, lesquels seront repris dans les sélecteurs.

Comme au théâtre

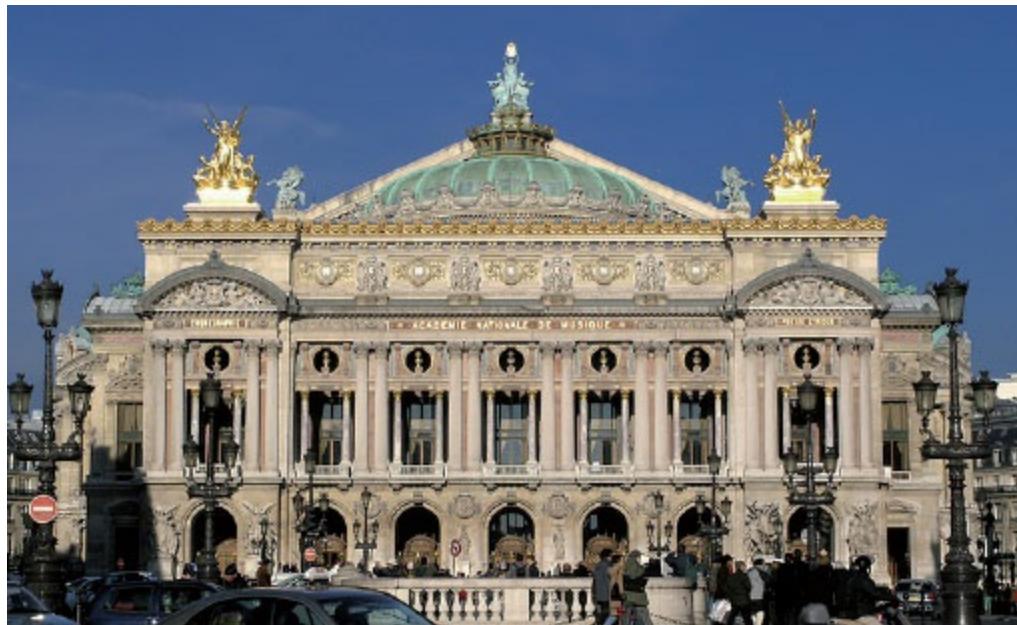


FIGURE 4–3 L’Opéra Garnier

Avant d’aller plus loin, transportons-nous au théâtre pour un petit instant. Tiens ! On y monte une pièce... Le metteur en scène distribue les rôles, les costumes, et il place les acteurs qui joueront la pièce.

De la même façon, le concepteur web va placer les balises HTML de la page. Le costume qu’il leur attribuera sera fait de couleurs, de polices de caractères, de bordures...



FIGURE 4–4 *Le metteur en scène monte une pièce, comme le concepteur web met en place son site : il aura un certain nombre d'éléments à mettre en forme.*

Sélecteur simple

Au théâtre, le metteur en scène dit : « Vous les hommes, vous serez habillés en bleu. Et vous, les femmes, en rouge. »

Les acteurs représentant les balises de notre page web, les hommes et les femmes sont des types de balises ; on aura par exemple, `<div>` et `<p>`.

Voici la règle qui vient d'être énoncée :

```
homme { couleur: bleu; }
femme { couleur: rouge; }
```

Elle devient dans notre feuille de styles :

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

La première règle s'applique à toutes les balises `<div>` de la page web, la deuxième à toutes les balises `<p>`.



FIGURE 4–5 « Les hommes seront habillés en bleu, les femmes en rouge. » Hommes et femmes représentent ici des « balises » distinctes, auxquelles sont attribuées des valeurs différentes pour la propriété « couleur ».

Classe

Une catégorie de balises

Au théâtre, le metteur en scène demande aux femmes qui ont un chapeau de venir au centre de la scène. Là, il vient de définir une sous-catégorie à l'intérieur de celle des femmes. La consigne ne s'adresse plus à toutes les femmes, mais seulement aux femmes qui ont un chapeau.

Dans nos pages web, supposons que parmi les paragraphes (délimités par les balises `<p>...</p>`), nous souhaitions centrer uniquement ceux qui constituent un chapeau dans le texte (un chapeau est une phrase d'introduction qui surplombe plusieurs colonnes).

Il suffit de donner un *nom de classe* à ces paragraphes, en tant qu'attribut de la balise dans le code HTML :

```
| <p class="chapeau" >...</p>
```

et d'écrire la règle suivante dans la feuille de styles :

```
| p.chapeau { text-align: center; }
```

Les balises qui ne sont pas de la classe « chapeau » ne seront pas concernées par cette règle, ni les balises autres que `<p>`, même si elles appartiennent à la classe `chapeau`. En particulier, la règle précédente ne s'applique pas à l'élément `<div class="chapeau"><p> ... </p></div>`.



FIGURE 4–6 « *Les femmes qui ont un chapeau, venez au centre de la scène !* » Parmi les femmes, ou désigne la « classe » de celles qui ont un chapeau.

Une même classe pour plusieurs types de balises

Notre metteur en scène prie alors toutes les personnes qui ont un chapeau de se pencher pour saluer le public. Une catégorie est définie, là encore, mais elle n'est pas associée à un type d'acteur : les hommes comme les femmes sont concernés.

De la même façon, il est possible d'appliquer des propriétés à plusieurs éléments de la page web, quelles que soient les balises qui les entourent. Il suffit de leur attribuer une classe commune, par exemple `<p class="menu">`, `<h1 class="menu">`, `` et d'écrire une règle du type :

```
.menu { font-style: italic; }
```

Le type de balise n'est pas précisé, seule la classe est indiquée, toujours précédée d'un point dans la feuille de styles. Cette règle fera donc apparaître en italique le contenu de n'importe quelle balise de classe `menu`.

À NOTER Plusieurs classes pour une même balise

Une balise peut être associée à plusieurs classes. Par exemple, l'élément `<p class="intro menu">...</p>` sera mis en forme par les règles du type `p.intro { ... }` et `p.menu { ... }`.

Il est d'ailleurs possible d'écrire comme sélecteur `p.intro.menu` (sans aucun espace dans cette expression), donc de préciser l'appartenance simultanée d'une balise à deux classes.

Identifiant

Retour au théâtre : après les directives générales, le metteur en scène donne des instructions plus précises, à chaque acteur en particulier. « Toi, Thomas, passe à droite ! Toi, Marion, assieds-toi dans le fauteuil ! » Chacune de ces consignes ne concerne qu'une personne. Il n'y a qu'un seul Thomas et une seule Marion dans la troupe des Joyeux Cabotins, et au début de la saison, lors des présentations, chacun a donné son prénom.

C'est ainsi qu'en HTML, une alternative aux classes est utilisée pour repérer un élément *unique* dans une page web. Il s'agit de donner à la balise un *identifiant* :

```
<div id="Thomas">....</div>
<p id="Marion">...</p>
```

ce qui permet de préciser cet identifiant dans la feuille de styles, à l'aide du caractère dièse # :

```
div#Thomas { text-align: right; }
p#Marion { vertical-align: -50%; }
```

Comme vous le voyez, le sélecteur précise l'identifiant comme une classe, simplement avec un dièse (#) à la place du point. Évidemment, les identifiants seront rarement des prénoms, mais plutôt des repères liés à la fonction de l'élément dans la page. Par exemple, s'il faut écrire en gris ce paragraphe précis :

```
<p id="auteur" >...</p>
```

nous pourrons lui attribuer une propriété de style comme :

```
p#auteur { color: gray; }
```

À NOTER Utilisation des identifiants

- Un même identifiant ne peut pas être utilisé pour deux éléments distincts de la même page. Lorsqu'il s'agira de repérer deux éléments en utilisant un même nom, nous utiliserons une classe.

- Dans une règle, il est possible d'utiliser une classe et un identifiant en même temps : `p.menu#auteur { ... }`.



FIGURE 4–7 « Marion, assieds-toi dans le fauteuil ! » Le nom de cette actrice, c'est son « identifiant ».

Identifiant sans nom de balise

De la même manière que pour les classes, le nom de la balise peut être omis dans le sélecteur :

```
#auteur { color: gray; }
```

Cette règle s'applique à la balise d'identifiant « auteur » `<... id="auteur">` (il ne peut y en avoir qu'une seule dans la page).

PRÉCISION Avec ou sans nom de balise ?

Les sélecteurs `p#auteur` et `#auteur` sont équivalents, puisqu'il ne doit y avoir qu'une seule balise d'identifiant « auteur » dans la page.

En revanche, s'agissant des classes, les sélecteurs `p.chapeau` et `.chapeau` n'ont pas le même sens : la balise `<div class="chapeau">` est concernée par le deuxième sélecteur, mais pas par le premier.

Différence entre classe et identifiant

Le tableau 4-1 résume la façon d'utiliser une classe et un identifiant, dans le code HTML et dans la feuille de styles.

TABLEAU 4-1 Comparaison entre classe et identifiant

	Classe	Identifiant
Balise	<p class="toto">	<p id="toto">
Règle de style	p.toto { ... } .toto { ... }	p#toto { ... } #toto { ... }
Éléments concernés	une seule balise ou plusieurs, identiques ou différentes	une seule balise dans toute la page

Les classes sont plus souvent utilisées que les identifiants, car elles permettent davantage de souplesse d'utilisation, en particulier lorsque la cascade des CSS entraîne la redéfinition d'un style. Plus loin dans ce chapitre, nous verrons pourquoi un identifiant est prioritaire sur une classe, lorsque deux styles successifs se contredisent.

Pseudo-classes

Notre metteur en scène précise et coordonne les actions des acteurs. « Toi, lorsque le maître de maison arrive, tu te retournes. Toi, après le passage du docteur, tu es guéri ! ».

En CSS, il existe des *pseudo-classes* qui, accolées à une balise, apportent des précisions aux sélecteurs. Cette méthode permet d'écrire des propriétés à utiliser uniquement dans certains cas de figure. La pseudo-classe la plus utilisée s'écrit :`hover` ; elle indique que la règle de style n'est à appliquer qu'au passage de la souris.

Par exemple, pour mettre en rouge le texte d'une balise `<a>` au moment de son survol par le curseur de la souris, il suffit d'écrire dans la feuille de styles :

```
a:hover { color: red; }
```

La règle suivante, elle, met le texte en rouge au passage de la souris, mais uniquement sur les balises ``, les autres balises `<a>` n'étant pas concernées :

```
a.menu:hover { color: red; }
```



FIGURE 4–8 « Après le passage du docteur, tu es guéri ! » L'action (rapide !) du docteur a eu un effet sur l'acteur, comme un clic de souris sur un lien transforme le lien en lien « visité ».

Pseudo-classes pour les liens hypertextes

- `:link` : lien hypertexte qui n'a pas été visité
- `:visited` : lien visité (et encore présent dans l'historique du navigateur)
- `:hover` : élément survolé par la souris
- `:active` : élément activé (la souris pointant ce lien, son bouton est enfoncé)

Si des règles de style concernant un même élément utilisent successivement plusieurs de ces quatre pseudo-classes, il faut respecter un ordre précis pour qu'elles soient bien prises en compte : `:link`, puis `:visited`, puis `:hover`, puis `:active`. Il existe un moyen mnémotechnique pour mémoriser leur ordre à partir de leurs initiales : LoVe HAte, soit *aimer* et *détester* en anglais, curieux mélange qui nous servira à retenir cet ordre.

Autres pseudo-classes

Trois autres pseudo-classes sont utilisables en CSS.

- `:first-child` : premier enfant d'une balise quelconque (premier des éléments imbriqués dans cette balise), comme `p:first-child` pour désigner tous les paragraphes `<p>` qui sont chacun le premier enfant de leur conteneur.
- `:focus` : qui possède le focus (c'est le cas pour une zone de saisie de formulaire, lorsqu'elle est sélectionnée et que le curseur clignote dedans, par exemple).
- `:lang(fr)` : balise qui possède un attribut `lang="fr"` ou qui est incluse dans un élément ayant cet attribut ; cette pseudo-classe peut être utilisée avec n'importe quel code de langue.

La norme CSS 3 a introduit un certain nombre de pseudo-classes supplémentaires, qui apportent plus de finesse et de souplesse dans l'écriture des sélecteurs. Vous les retrouverez dans le chapitre 7.

Pseudo-éléments

Ressemblant aux pseudo-classes, les pseudo-éléments apportent d'autres types de précision. Auparavant, ils commençaient eux aussi par le caractère « deux-points » (:), mais les nouvelles normes préconisent à présent deux « deux-points » (::). Dans la pratique, les deux formes sont reconnues pour les pseudo-éléments :

- ::first-letter (ou :first-letter) = première lettre du bloc ;
- ::first-line (ou :first-line) = première ligne du bloc ;
- ::before (ou :before) = avant la balise spécifiée ;
- ::after (ou :after) = après la balise spécifiée.

L'exemple suivant agrandit la taille de la première lettre pour chaque paragraphe <p> :

```
p::first-letter { font-size: 150%; }
```

UTILISATION Pseudo-éléments ::before et ::after

Ces deux pseudo-éléments permettent d'insérer un texte ou une image avant ou après une balise donnée. La propriété `content` permet d'ajouter un contenu avant ou après l'élément indiqué dans le sélecteur. Si ce contenu est une image, il faut utiliser la syntaxe `url('http://...')` avec un lien absolu vers l'image (chemin à partir de la racine).

Exemples :

```
p.note::before { content: "Note : " }  
blockquote::after { content: url('http://titi.fr/logo.png'); }
```

Règle associée à plusieurs sélecteurs

Si le metteur en scène dit : « Tous les hommes avec un chapeau et toutes les femmes, venez au centre de la scène ! », il donne une seule consigne qui s'adresse à plusieurs catégories d'acteurs. Sont concernés ici tous ceux des hommes qui ont un chapeau et toutes les femmes sans exception.

En CSS, cela donnerait quelque chose comme :

```
div.chapeau, p { text-align: center; }
```

Cette règle s'applique à toutes les balises `<p>` et aux balises `<div class="chapeau">`.

Voici un autre exemple de règle qui s'applique au contenu des balises `<h1>`, `<h2>` et `<h3 class="sommaire">` :

```
h1, h2, h3.sommaire { text-align: center; }
```



FIGURE 4–9 En donnant une consigne unique pour les hommes avec chapeau et pour les femmes, le metteur en scène attribue, en une seule fois, une « propriété » à une balise avec la classe « chapeau », ainsi qu'à une autre balise, sans précision de classe.

Regroupement de propriétés à l'aide de « raccourcis »

La règle suivante définit, pour les titres de niveau 1, le type de bordure (épaisseur, style et couleur de l'encadrement) :

```
h1 {  
    border-style: solid;  
    border-width: 2px;  
    border-color: blue;  
}
```

Ces trois propriétés peuvent être remplacées par une seule propriété `border` qui prend en compte les trois valeurs associées :

```
h1 { border: solid 2px blue; }
```

Lorsque nous étudierons en détail les propriétés de style, ces raccourcis seront mentionnés chaque fois qu'ils existent.

Hiérarchie des sélecteurs

Pour appliquer un style à un élément lorsqu'il est inclus dans un autre élément donné, il suffit d'écrire un sélecteur avec les deux noms de balise (ou de classe, d'identifiant...) séparés par un espace, comme le montrent les exemples ci-après.

Si nous voulons justifier seulement le texte des paragraphes `<p>` qui sont inclus dans un bloc `<div>`, nous écrirons la règle suivante :

```
div p { text-align: justify; }
```

Pour mettre en gris uniquement les liens contenus dans l'élément d'identifiant « sommaire », la règle à utiliser est :

```
#sommaire a { color: gray; }
```

À NOTER Plusieurs niveaux d'imbrication

Ces règles s'appliquent aussi aux balises qui sont séparées par d'autres niveaux d'imbrication. Il suffit que la deuxième balise soit un « descendant », direct ou éloigné, de la première.

La règle du dernier exemple s'appliquera notamment dans le cas suivant :

```
<div #sommaire>...<p>...<a>...</a>...</p>...</div>
```

Hiérarchie précise des sélecteurs

Les caractères > et + reflètent une hiérarchie plus précise entre les balises : imbrication directe pour >, juxtaposition des balises avec +.

Imbrication directe

```
| div > h1 { font-style: italic; }
```

Cette règle s'applique aux balises `<h1>` qui sont au premier niveau d'imbrication à l'intérieur d'une balise `<div>` (enfant direct), mais elle n'influera pas sur celles imbriquées plus « profondément » à partir du `<div>` (descendants au-delà de la première génération).

Juxtaposition

```
| h1 + h2 { margin-top: 10px; }
```

Cette règle s'applique à chaque balise `<h2>` qui suit une balise de fermeture `</h1>` (c'est le frère suivant de `<h1>`). Entre ces deux balises peut se trouver du texte, mais pas une autre balise.

Sélecteur d'attribut [...]

Cette méthode intéressante sélectionne les balises utilisant un attribut donné. Il est possible de tester la valeur affectée à cet attribut, ou simplement sa présence dans la balise.

Voici un exemple de règle qui applique une couleur de fond jaune aux boutons de formulaire autres que ceux spécifiques à l'envoi et à l'effacement. Cette règle concerne les balises `<input>` qui ont un attribut `type="button"`, donc les balises `<input type="button" ...>` :

```
| input[type="button"] { background-color: yellow; }
```

Voici un autre exemple, qui met en vert le texte des liens pour lesquels un raccourci d'accessibilité est défini par l'attribut `accesskey` (``), quelle que soit ici la valeur de cet attribut :

```
| a[accesskey] { color: green; }
```

Sélecteur universel *

Retournons une dernière fois au théâtre, si vous le voulez bien. Le metteur en scène, très pointilleux, explique : « À la fin de la pièce, tous les acteurs viennent au centre pour saluer. » (et les acteurs haussent les épaules, comme pour dire : « Mais il nous prend pour des débutants, ce rigolo ? »). L'expression « tous les acteurs » englobe tous les éléments de notre joyeuse troupe, sans exception.

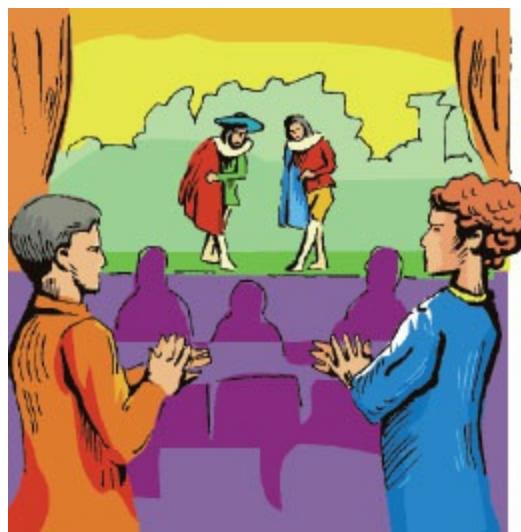


FIGURE 4-10 «À la fin de la pièce, tous les acteurs viennent au centre pour saluer ». Toutes les balises sont concernées par la propriété.

Dans une règle de style, c'est l'étoile (*) qui est utilisée comme sélecteur universel ; elle signifie : « n'importe quelle balise ». La consigne précédente devient alors :

```
* { text-align: center; }
```

Voici en particulier une propriété qui est souvent utilisée pour toutes les balises de la page, afin d'annuler les marges extérieures :

```
* { margin: 0; }
```

À NOTER Règle pour toutes les balises d'un bloc

Si une règle concerne toutes les balises qui sont incluses dans un bloc donné, il faut écrire par exemple :

```
div * { color: blue; }
```

Dans ce cas précis, toutes les balises qui sont incluses dans un bloc `<div>` quelconque verront leur texte écrit en bleu.

Ordre de priorité des styles

Si une règle de style vient contredire une règle précédente, c'est *en général le dernier style défini qui s'applique.*

Règle de style prioritaire

Pour qu'un style ne soit pas modifié par un autre, il faut écrire `!important` avant le point-virgule qui termine la propriété. Exemple :

```
body { background-color: white !important; }
```



FIGURE 4–11 Comme les véhicules de secours sur la route, les propriétés marquées par `!important` auront la priorité.

Degré de priorité d'une règle de style

AVIS Lecteur découvrant les CSS

Débutant en CSS, vous pouvez dans un premier temps ignorer ce paragraphe un peu pointu et y revenir plus tard.

La règle exacte de priorité, pour les styles en cascade, est la suivante :

Si deux règles de style sont contradictoires, la deuxième remplace la première, sauf si cette première règle a un degré de priorité (c'est-à-dire de spécificité) supérieur à la deuxième.

Le degré de priorité d'une règle de style est fonction d'un nombre de quatre chiffres $x_4 x_3 x_2 x_1$, calculé à partir du sélecteur de cette règle.

- chiffre des *milliers* (x_4) :
1 si style **prioritaire** (style en ligne, ou `!important`), 0 sinon
- chiffre des *centaines* (x_3) :
nombre d'identifiants (#xxx) dans le sélecteur
- chiffre des *dizaines* (x_2) :
nombre de classes (.xxx) qui interviennent dans le sélecteur
- chiffre des *unités* (x_1) :
nombre d'éléments séparés par des espaces dans le sélecteur

Le tableau suivant permet de comprendre ce calcul de spécificités, donc de priorités, classées ici par ordre croissant. Ce document provient de la page https://www.openweb.eu.org/articles/cascade_css/, à consulter pour plus d'informations.

TABLEAU 4–2 Exemples de calculs de priorité

Style	Style local ou !important	Nombre d'identifiants	Nombre de classes	Nombre d'éléments	Priorité
<code>* { ... }</code>	0	0	0	0	0000
<code>p { ... }</code>	0	0	0	1	0001
<code>div p { ... }</code>	0	0	0	2	0002

TABLEAU 4–2 Exemples de calculs de priorité (suite)

Style	Style local ou <code>!important</code>	Nombre d'identifiants	Nombre de classes	Nombre d'éléments	Priorité
<code>.class {...}</code>	0	0	1	0	0010
<code>p.class {...}</code>	0	0	1	1	0011
<code>div p.class {...}</code>	0	0	1	2	0012
<code>#id {...}</code>	0	1	0	0	0100
<code>p#id {...}</code>	0	1	0	1	0101
<code>div p#id {...}</code>	0	1	0	2	0102
<code>.class #id {...}</code>	0	1	1	0	0110
<code>.class p#id {...}</code>	0	1	1	1	0111
<code>div.class p#id {...}</code>	0	1	1	2	0112
<code><p style="..."></code>	1	0	0	0	1000
<code>...{...!important;}</code>	1	0	0	0	1000

Application

Voici un exemple où une propriété est redéfinie à l'aide d'une deuxième règle qui a une priorité plus faible que la première :

```
div p { color: blue; }
p { color: green; }
```

Dans ce cas, les paragraphes `<p>` seront écrits en vert, sauf ceux inclus dans un bloc `<div>`, qui resteront en bleu.

Cet exemple n'est toutefois pas un modèle d'écriture à suivre : la logique et la facilité de compréhension du code voudraient que l'ordre de ces deux règles soit inversé, pour aller du plus général vers le plus spécifique.

À NOTER Priorités et combinateurs d'éléments

Les combinateurs d'éléments tels que `>` (enfants directs) ou `+` (éléments adjacents) n'ont pas d'influence sur les priorités.

Valeurs, tailles et couleurs

Avant d'aborder le détail des propriétés, il est important de définir les codes et unités à adopter pour les valeurs qui leur seront attribuées.

Héritage de propriété

Toutes les propriétés peuvent prendre la valeur `inherit` : cela crée un héritage pour des propriétés qui normalement ne sont pas héritées.

Unités de taille

Les unités de taille sont utilisées pour les propriétés telles que la taille des caractères bien sûr, mais aussi pour les largeurs et hauteurs de blocs et d'images, leurs marges, leurs bordures, le positionnement des éléments dans la page, etc.

À NOTER Valeurs décimales

Si rien n'oblige les valeurs de taille à être entières, il faut cependant penser à utiliser le point, et non la virgule, comme séparateur décimal.

Unités de taille courantes

Les unités de taille relatives sont fonction de la taille du texte environnant ou du nombre de points sur l'écran. C'est dans cette catégorie que nous retrouvons les plus utilisées :

- `rem`, largeur d'une majuscule comme M dans l'élément racine `<html>` ;
- `em`, largeur d'une majuscule comme M dans l'élément parent ;
- `px` pour pixel, soit un point de l'écran (en théorie il y a 72 ou 96 pixels par pouce) ;
- % pour les pourcentages, 100 % correspondant à `1em` pour la taille des caractères `font-size`.

Dans un paragraphe standard, la taille par défaut des caractères est de `16px`, soit `1rem` ou `1em` (ou encore `12pt`, ce qui correspond à la taille 12 dans un traitement de texte).

Pourquoi préférer l'unité `rem` à `em`

Les différences entre `rem` et `em` sont les suivantes :

- `1rem` équivaut à 100 % de la taille de base de l'élément *racine*, c'est-à-dire de la balise `<html>` (« `rem` » est l'acronyme de *root em*, *root* signifiant « racine » en anglais) ;
- `1em` correspond à 100 % de la taille de base dans l'élément *parent*.

Avantage et inconvénient s'expliquent simplement avec un exemple. Prenons

le morceau de code HTML suivant :

```
| <p> texte 1 <span> texte 2 </span></p>
```

auquel nous appliquons cette règle de style :

```
| p, span { font-size: 2em; }
```

Dans ce cas, `texte 2` sera écrit deux fois plus gros que `texte 1`, puisque les caractères de la balise `` seront doubles de ceux de son conteneur `<p>`, la taille `em` faisant référence à celle de l'élément parent.

En revanche, si nous utilisons l'unité `rem` dans cette règle de style, ce problème n'existe plus :

```
| p, span { font-size: 2rem; }
```

Ici, les caractères des balises `<p>` et `` ont deux fois la taille de base de l'élément racine `<html>`. Par conséquent, lorsqu'elle est exprimée en `rem`, la dimension des lettres est la même dans ces deux balises.

ASTUCE Transformer facilement des pixels en rem

L'unité `rem` est à privilégier, car elle est à la fois souple et pratique. En effet, elle permet d'exprimer les tailles des caractères en fonction de la taille de base figurant dans la balise racine `html`. Par conséquent, il suffit de changer cette taille de base pour obtenir une modification proportionnelle de la taille des polices dans l'ensemble des balises de la page.

Cependant, le pixel est une unité très couramment utilisée (par exemple, pour la taille des images) et jongler entre *pixels* et *rem* peut être délicat. Jonathan Snook propose sur son site <https://snook.ca> une règle de style qui permet de créer un lien direct et évident entre `rem` et `pixels` :

```
html { font-size: 62.5%; }
```

La taille standard des caractères, qui est de 16 pixels, vaut maintenant $16 \times 62.5 / 100$ soit 10 pixels. Il en résulte une correspondance très simple, puisque `1rem` représente `10px` et que `2rem` valent `20px`. En bref, l'unité `rem` est devenue la dizaine de pixels.

Si, par exemple, nous écrivons les règles suivantes :

```
body { font-size: 1.6rem; }
h1 { font-size: 2.5rem; }
```

la taille de base des caractères (dans la balise `body`) redevient 16 pixels, tandis que les titres de niveau 1 prennent une taille de 25 pixels.

Autres unités de taille

Unités de taille relative

Les autres unités de taille relative, plus rarement utilisées, sont les suivantes :

- `ex`, hauteur d'une minuscule comme `x`, souvent arrondie à `0.5em` et en théorie fonction de la police utilisée, donc à éviter ;
- `ch`, unité égale à la largeur du chiffre « 0 » (zéro) ;
- `vw`, taille égale à 1 % de la largeur de la fenêtre (« `v` » pour `viewport` et « `w` » pour `width`) ;
- `vh`, taille valant 1 % de la hauteur de la fenêtre (« `v` » pour `viewport` et « `h` » pour `height`) ;
- `vmin`, qui est la plus petite valeur de `vw` et `vh`, soit 1 % de la dimension minimale du `viewport` (c'est-à-dire 1 % de la largeur ou de la hauteur de la fenêtre du navigateur, suivant le cas) ;
- `vmax`, qui est la plus grande valeur de `vw` et `vh`, donc 1 % de la dimension maximale du `viewport`, la fenêtre du navigateur.

En particulier, les unités de mesure `vw` et `vh` pourront être utiles pour définir des largeurs et hauteurs de blocs dans une page, de façon à adapter la présentation aux dimensions de la fenêtre du navigateur.

Unités de taille fixe (déconseillées)

Les valeurs utilisables pour des tailles fixes sont :

- `pt` (1 point = 0,35 mm) ;
- `pc` (1 pica = 12 pt = 4,22 mm) ;
- `cm`, `mm`, `in` (1 inch ou 1 pouce = 2,54 cm).

Il est préférable d'éviter ces tailles fixes, qui ne tiennent pas compte de la

taille de l'écran et empêchent aussi la personnalisation de l'affichage dans le navigateur. Leur utilisation concerneira seulement des documents spécifiques destinés à l'impression.

Tailles définies par mots-clés (déconseillées)

Les tailles peuvent également être indiquées à l'aide de mots-clés, qui ressemblent aux dimensions d'un vêtement. Ces définitions sont moins précises, car interprétées de façons diverses par les différents navigateurs.

De la plus petite à la plus grande, les tailles disponibles sont :

■ `xx-small`, `x-small`, `small`, `medium` (taille standard), `large`, `x-large`, `xx-large`

ATTENTION Différence entre taille des caractères et taille perçue

Pour une taille de caractères donnée, nous obtiendrons des résultats très différents en fonction de la police utilisée, comme le montre nettement la figure 4-12.

En effet, la taille des caractères définit la hauteur des majuscules, qui n'est qu'un des critères de dimension pour les lettres : il faut également tenir compte de leur largeur, de l'épaisseur de leur trait (*la graisse*), ainsi que de la hauteur des minuscules comme le « a » ou le « o » (qui s'appelle la *hauteur d'œil* en typographie).

C'est pourquoi un changement de police pour les caractères nous amènera parfois à revoir leur taille, si nous souhaitons que le lecteur perçoive globalement la même dimension d'écriture.

Texte écrit en taille 20px - police Arial

Texte écrit en taille 20px - police Tahoma

Texte écrit en taille 20px - police Verdana

Texte écrit en taille 20px - police Georgia

Texte écrit en taille 20px - police Times

FIGURE 4–12 Pour une même taille de caractères, la largeur des mots et la taille globale perçue seront très variables suivant la police utilisée.

Si les exemples précédents avaient souvent trait à la taille des caractères, n'oublions pas pour autant que les unités de mesure servent également à attribuer aux éléments HTML des dimensions (largeur et hauteur), des marges et espacements, des valeurs de positionnement, des épaisseurs de bordure, etc. Dans les chapitres qui suivent, les occasions de rencontrer ces différentes applications ne nous manqueront pas.

Codage des couleurs

Les couleurs sont définies à l'aide de noms ou de codes numériques.

Noms de couleurs

À certaines couleurs « standards » ont été attribués des mots réservés : `blue`, `white`, `red`... La liste de ces mots-clés est donnée en annexe.

Code RVB

Le codage rouge-vert-bleu (RVB en français, RGB en anglais) consiste à préciser la quantité de chacune de ces couleurs, exprimée :

- en **décimal** : `rgb(255, 0, 0)` = rouge ;
- en **pourcentage** : `rgb(0, 100%, 0)` = vert ;
- en **hexadécimal** : `#0000ff` = bleu (deux chiffres hexadécimaux pour chaque couleur RVB).

REMARQUE Notation hexadécimale raccourcie

Il est possible d'utiliser une notation hexadécimale raccourcie, dans laquelle chaque chiffre hexadécimal doit être doublé pour obtenir le code réel de la couleur. Par exemple, `#00f` est équivalent à `#0000ff` et représente le bleu.

À NOTER Noir et blanc

Il existe donc plusieurs façons d'écrire le code du noir et celui du blanc :

- pour le noir
`rgb(0, 0, 0)` = `#000` = `#000000` = `black`
- pour le blanc
`rgb(255, 255, 255)` = `rgb(100%, 100%, 100%)` = `#fff` = `#ffffff` = `white`

Toutes les couleurs sont « sûres »

Si vous apercevez à l'occasion une liste de 216 couleurs RVB appelées « couleurs sûres », ce sont celles dont autrefois l'affichage à l'écran était

initialement garanti sur toutes les configurations matérielles et logicielles.

En hexadécimal, chacune des composantes RVB de ces couleurs « sûres » vaut 00, 33, 66, 99, cc ou ff. Cette parenthèse historique vise à préciser que tous les systèmes affichent plus de 16 millions de couleurs en RVB depuis fort longtemps, et qu'il n'est plus nécessaire, par conséquent, de se restreindre à ces couleurs sûres.

En revanche, suivant la qualité de l'écran et la personnalisation des réglages effectués par l'utilisateur, il se peut que le rendu des couleurs soit différent ; mais là, c'est une autre histoire...

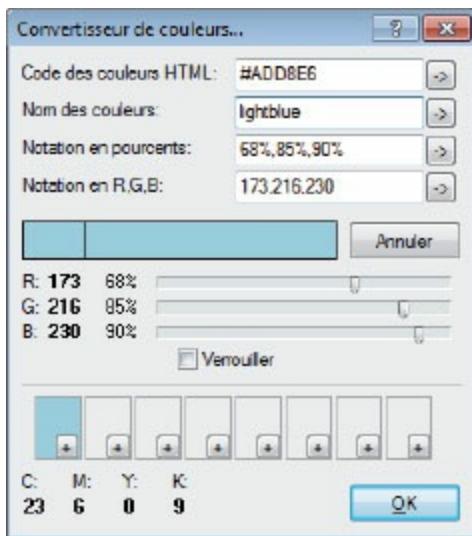


FIGURE 4–13 Le logiciel PsPad propose un convertisseur de couleurs, entre code hexadécimal, nom de couleur et notation RGB en pourcentage ou en décimal. Quelques couleurs peuvent être mémorisées dans les cadres du bas.

Des outils en ligne pour les couleurs

Un certain nombre des sites Internet proposent une aide dans le choix des couleurs, pour assortir des teintes différentes ou pour trouver une nuance précise. Dans tous les cas, ces sites fournissent les codes HTML associés :

- *Adobe Color CC* (<https://color.adobe.com>) affiche des assortiments et des nuances, à partir d'une couleur choisie (la rubrique **Explorer**, en haut de la fenêtre, donne accès à une variété de palettes) ;
- « *0 to 255* » (<http://www.0to255.com>) fournit des gammes de couleurs monochromes ;
- *Colour Lovers* (<http://www.colourlovers.com>) est une communauté où

sont partagées par les internautes des palettes de couleurs, ainsi que des motifs fantaisie.

Exemple de page avec feuille de styles interne

Voici une page HTML simplifiée, avec sa feuille de styles interne.

La lecture du code et de ses commentaires nous montre une mise en pratique des notions vues jusqu'ici.

Certes, les propriétés et leur utilisation n'ont pas encore été détaillées, mais celles qui figurent ici sont simples à comprendre.

```
<!DOCTYPE html>
<html>
<head>

<meta charset="utf-8" />
<title>Garage des Tacots - Page d'accueil</title>
<style>
/* Pour toute la page : texte centré, fond gris clair */
body { text-align: center; background-color: silver; }
/* Tous les titres h1 sont en marron, en taille 250% et sur fond blanc */
h1 { color: brown; font-size: 250%;
    background-color: white; }
/* Tous les titres h2 sont en bleu, avec une marge de 30 pixels autour */
h2 { color: blue; margin: 30px; }
/* Toutes les balises de classe "titre" sont en vert */
.titre { color: green; }
/* Les titres h1 de classe "titre" sont en Arial, en taille 280% et encadrés
d'un trait plein */
h1.titre { font-family: Arial, sans-serif;
    font-size: 280%; border: solid; }
/* Les titres h2 de classe "titre" sont en italique et en taille 150% */
h2.titre { font-style: italic; font-size: 150%; }
</style>

</head>

<body>

<h1 class="titre">Garage des Tacots</h1>
<h2 class="titre">Voitures anciennes</h2>
<br />

<h1>Nos services</h1>
```

```
<h2>Peinture et retouches</h2>
<h2>Pièces sur mesure</h2>
<h2>Pneumatiques toutes dimensions</h2>

</body>

</html>
```

Vous pouvez recopier ce code dans le bloc-notes ou un éditeur HTML, enregistrer ce fichier texte avec l'extension .html et l'afficher dans votre navigateur web en double-cliquant sur son nom dans l'explorateur de fichiers.



FIGURE 4–14 Affichage de la page « Garage des tacots » dans un navigateur web

À partir de cet exemple, n'hésitez pas à modifier les propriétés, les niveaux de titre et les classes, pour mieux comprendre leur fonctionnement.

Une présentation très différente de cette page peut être obtenue en modifiant seulement *les valeurs* de certaines propriétés. La figure 4-15 en donne une illustration, à partir de la feuille de styles suivante :

```
<style>
/* Pour toute la page : texte aligné à gauche, fond bleu clair "lavande" */
body {text-align: left; background-color: lavender;}
/* Titres h1 : en blanc, taille 200%, fond gris */
h1 { color: white; font-size: 200%; background-color: silver; }
/* Titres h2 : en noir, marges de 10 pixels */
h2 { color: black; margin: 10px; }
/* Balises de classe "titre" : en bleu */
.titre { color: blue; }
/* Titres h1 de classe "titre" : en Courier New, taille 250% et encadré avec
des tirets */
h1.titre { font-family: "Courier New", monospace; font-size: 250%; border:
dashed; }
/* Titres h2 de classe "titre" : en italique, taille 190% */
h2.titre { font-style: italic; font-size: 190%; }
</style>
```

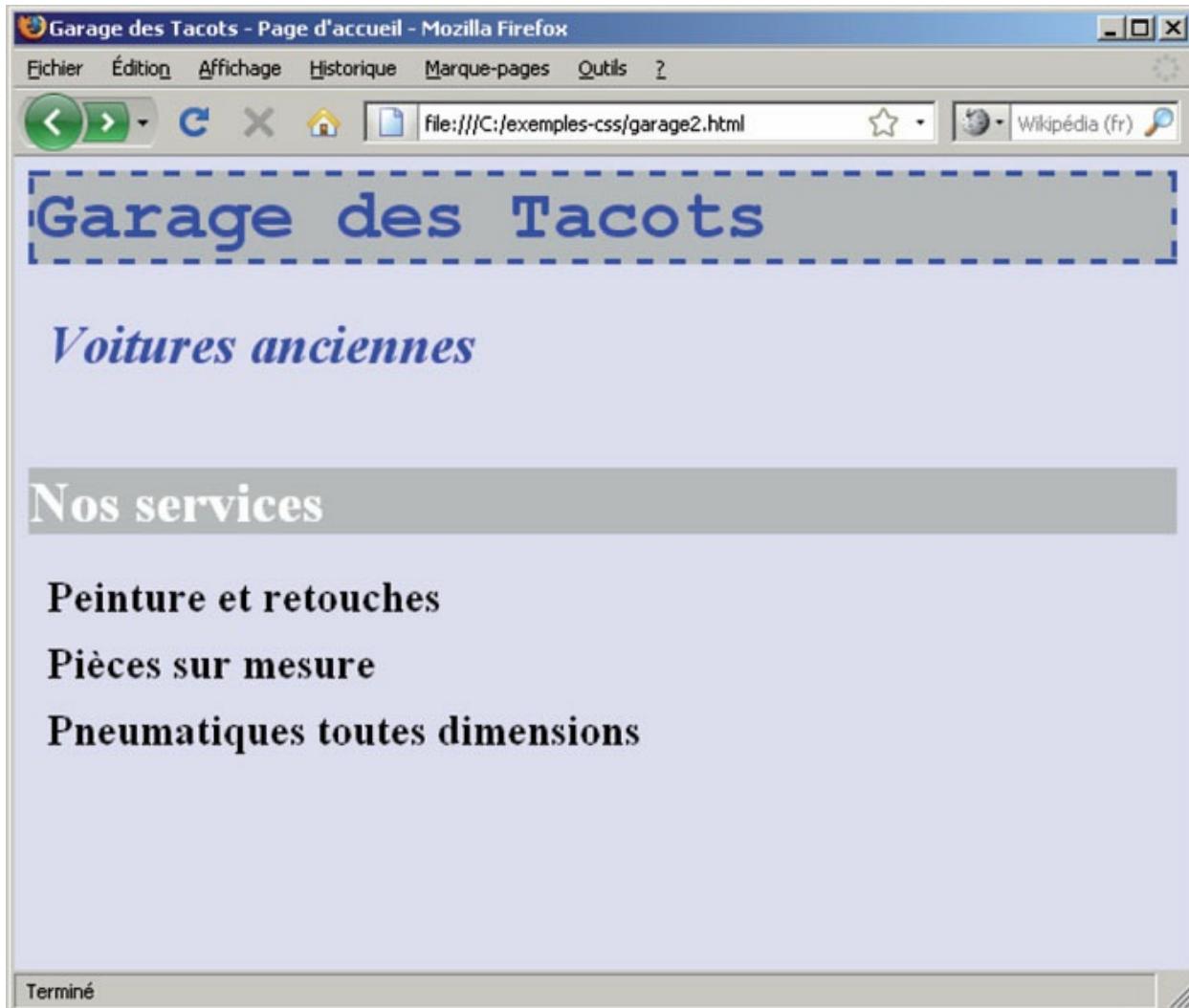


FIGURE 4–15 Une autre version de la page précédente, où seules ont été modifiées les valeurs de certaines propriétés dans la feuille de styles

Nous avons maintenant compris la structure d'une feuille de styles CSS et son utilisation en liaison avec le code HTML. Les exemples précédents nous ont d'ailleurs permis de connaître quelques-unes des propriétés et certaines de leurs valeurs possibles.

Le chapitre qui suit nous emmène vers la découverte méthodique des propriétés CSS qui serviront à mettre en forme le texte de nos pages.

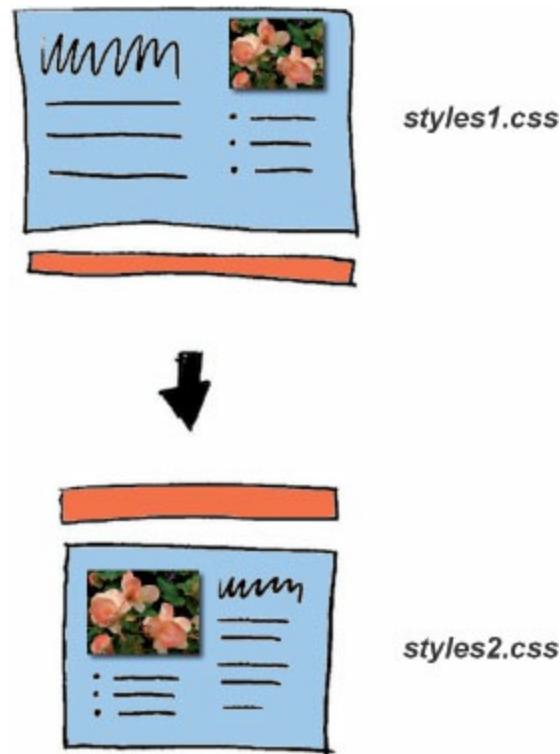
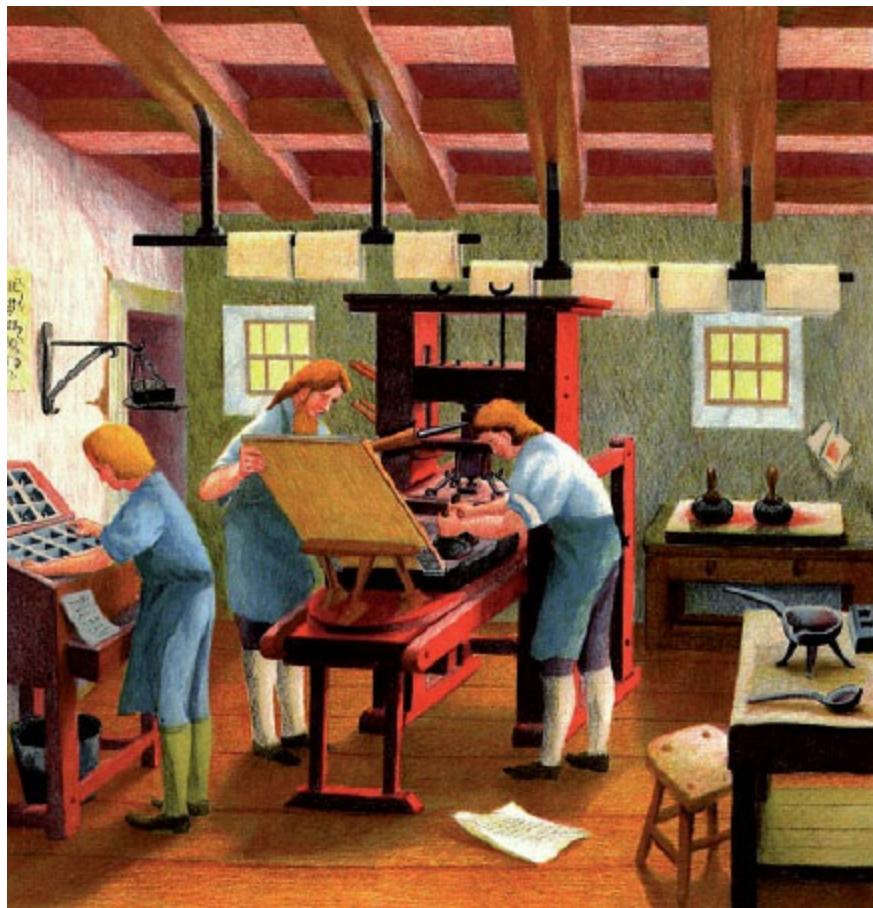


FIGURE 4–16 À partir d'un même contenu HTML, la seule modification de la feuille de styles peut transformer complètement la présentation d'une page. D'après un dessin du blog de l'agence Zurb (<http://zurb.com/blog>) et une photo Wikimédia.

chapitre 5

Propriétés de mise en forme



Examinons en détail les différentes propriétés de mise en forme proposées par les normes CSS 2 et CSS 3 : nom, syntaxe, valeurs possibles, héritage. Des exemples illustrent chacune de ces propriétés.

SOMMAIRE

- ▶ **Mise en forme des caractères**
- ▶ **Paragraphes et blocs de texte**
- ▶ **Bordures**
- ▶ **Images et couleurs d'arrière-plan**
- ▶ **Listes à puces ou numérotées**
- ▶ **Tableaux**

Après avoir découvert le principe des feuilles de styles (avec leur écriture), nous entrons dans le vif du sujet : les propriétés de style permettent la mise en forme de nos pages. Celles présentées dans ce chapitre sont communes aux deux normes CSS 2 et CSS 3. Sans avoir à les apprendre par cœur, il est quand même utile d'en connaître l'existence, pour penser à les utiliser.

Mise en forme des caractères

Donnez du style à vos textes ! Vous allez pouvoir leur conférer tantôt une allure qui tranche, tantôt un aspect discret, bref tout ce qu'il faut pour enjoliver votre prose (ou vos poèmes !) à la manière d'un traitement de texte.

Sont regroupées ici toutes les propriétés qui peuvent s'appliquer à un seul ou plusieurs caractère(s). Toutefois, elles sont généralement utilisées pour mettre en forme des mots ou des paragraphes entiers.

Choix des polices

Le choix d'une police de caractères s'effectue à l'aide de la propriété `font-family`.

TABLEAU 5–1 Propriété font-family

Propriété	<code>font-family</code>
Exemples	<code>p {font-family: Arial, Verdana, sans-serif;}</code> <code>p {font-family: "Times New Roman", serif;}</code> <code>p {font-family: "Courier New" , monospace;}</code>
Valeurs possibles	Noms de polices de caractères, séparés par des virgules, ou type de police générique : <code>serif</code> , <code>sans-serif</code> , <code>monospace</code> . Les noms en plusieurs mots sont à placer entre guillemets.
Héritage	Propriété <i>héritée</i> : elle se transmet dans les balises imbriquées.

IMPORTANT Choix des polices

En CSS 2, une police de caractères ne s'affichera que si elle est installée sur l'ordinateur utilisé pour consulter la page web. Il vaut donc mieux procéder ainsi.

- Éviter les polices « personnelles », qui ne sont pas installées par défaut par Windows, Linux et macOS, mais qui ont été ajoutées sur votre ordinateur (par exemple, l'installation de certains logiciels ajoute des polices automatiquement).
- Proposer plusieurs polices de caractères : si le navigateur ne trouve pas la première police sur l'ordinateur utilisé, il prendra la deuxième ; s'il ne la trouve pas non plus, il prendra la troisième, et ainsi de suite...
- Terminer la liste par une police « générique » : `serif`, `sans-serif`, `monospace` (police à chasse fixe, du type Courier) ; il existe aussi les types `cursive` et `fantasy`, très peu utilisés.

Nous verrons plus tard une méthode spécifique aux CSS 3, qui permet de transmettre une police de caractères associée à une page web.

À NOTER Polices courantes

Les polices « standards », qu'on a toutes les chances de trouver sur un PC ou un Mac, sont les suivantes : Arial, Comic, Courier, Garamond, Georgia, Times, Trebuchet, Verdana.

D'autres polices sont assez souvent présentes, tant sur les PC que sur les Mac, pour être utilisables à condition de ne pas oublier de mentionner une police de remplacement dans la propriété `font-family` : *Arial Black, Arial Narrow, Century Gothic, Helvetica, Impact, Palatino et Tahoma*.

L'arrivée des CSS 3 nous permettra d'utiliser des polices plus originales : elles pourront être téléchargées en même temps que la page à afficher, de la même façon que les fichiers images.

Taille de police

La propriété `font-size` permet de préciser la taille des caractères.

ATTENTION Tailles relatives

Mises à part celles exprimées en pixels ou en `rem`, les tailles relatives sont exprimées par rapport à la taille de police de la balise qui contient l'élément. Ainsi, prenons l'exemple suivant :

```
<p>Bonjour <span>tout le monde</span></p>
```

associé à la règle de style :

```
p, span { font-size: 80%; }
```

La taille du texte de la balise `` représente 64 % de celle de la taille initiale. En effet, puisque la règle s'applique à chacune des balises, les caractères de la balise imbriquée `` ont une taille égale à 80 % de 80 % de la taille de base.

TABLEAU 5–2 Propriété `font-size`

Propriété	<code>font-size</code>
Exemples	<code>h1 { font-size: 150%; }</code> <code>p { font-size: 15px; }</code>
Valeurs possibles	<i>taille relative</i> (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... ou <i>taille fixe</i> en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code> ou <i>mot-clé</i> <code>xx-small</code> , <code>x-small</code> , <code>small</code> , <code>medium</code> (= standard), <code>large</code> , <code>x-large</code> , <code>xx-large</code>
Pourcentages	% de la taille de la police dans l'élément parent
Héritage	Propriété <i>héritée</i>

Couleur du texte

Les valeurs attribuées à la propriété `color`, pour la couleur du texte, peuvent s'exprimer soit à l'aide d'un mot-clé, soit avec un code numérique hexadécimal, soit par la fonction `rgb` associée à des nombres entiers ou des pourcentages.

TABLEAU 5–3 Propriété `color`

Propriété	<code>color</code>
Exemples	<pre>body { color: #0000ff; } h1.menu { color: #6e05c3; } .utile { color: rgb(255,0,0); } a { color: rgb(30%,80%,30%); } em { color: green; }</pre>
Valeurs possibles	nom de couleur prédéfini ou code RVB
Héritage	Propriété héritée

Il faut noter qu'un cas de figure très courant met en défaut l'héritage de la propriété `color` : si une couleur est affectée à un élément HTML, tout son contenu s'écrit dans la couleur choisie... sauf les liens. En effet, les navigateurs conservent leur couleur par défaut pour les liens hypertextes (en général le bleu), de façon à les rendre plus visibles.

Par conséquent, la couleur des liens devra être précisée explicitement ; il ne faut pas compter sur l'héritage de `color` pour ce type d'élément. Si les liens sont à afficher de la même couleur que le texte, il faudra écrire par exemple :

```
div.contenu, div.contenu a { color: dimgray; }
```

Grâce à cette règle de style améliorée, tout le texte inclus dans le bloc `div` de classe `contenu` est écrit en gris foncé, y compris les liens hypertextes. Cette couleur s'applique également aux liens visités (souvent affichés en violet), le sélecteur général `a` incluant la pseudo-classe `a:visited`. Le soulignement automatique des liens reste ici actif ; ceux-ci restent donc repérables dans la page, ce qui est important.

ATTENTION Nom de la propriété `color`

La propriété qui change la couleur du texte est bien `color`. Une erreur courante consiste à écrire `font-color`, alors que ce nom de propriété n'existe pas, bien qu'il eût été logique pour indiquer la couleur de police.

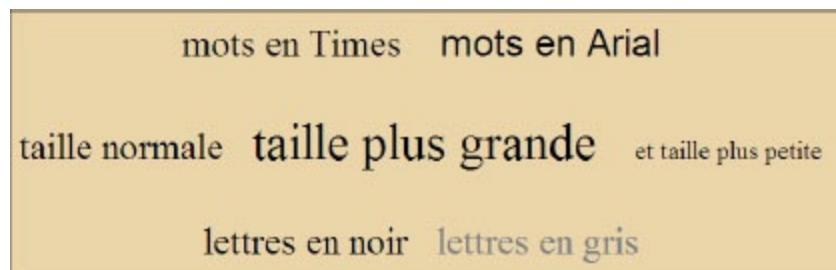


FIGURE 5–1 Utilisation des propriétés `font-family` (type de police), `font-size` (taille des caractères) et `color` (couleur d'écriture)

Texte en gras

Avec la propriété `font-weight`, il s'agit de préciser le degré d'épaisseur de l'écriture, c'est-à-dire la graisse de la police.

TABLEAU 5–4 Propriété `font-weight`

Propriété	<code>font-weight</code>
Exemple	<code>.principal { font-weight: bold; }</code>
Valeurs possibles	<code>normal</code> (valeur par défaut), <code>bold</code> : gras, <code>lighter</code> : moins gras que le style en cours, <code>bolder</code> : plus gras que le style en cours, ou un nombre (<code>100</code> , <code>200</code> ... <code>900</code>) qui définit le niveau de gras
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

À NOTER Valeurs numériques de gras

Dans l'échelle des valeurs numériques qui définissent le niveau de gras, `400` correspond à `normal` et `700` à `bold`.

Toutefois, l'utilisation de nombres est déconseillée avec `fontweight`, car ceux-ci ne sont pas bien pris en compte par les différents navigateurs. Dans la pratique, il vaut mieux s'en tenir aux valeurs de base `normal` et `bold`.

Italique

Pour mettre des mots en italique, c'est la propriété `font-style` qui nous sera utile.

TABLEAU 5–5 Propriété font-style

Propriété	<code>font-style</code>
Exemple	<code>.remarque { font-style: italic; }</code>
Valeurs possibles	<code>normal</code> : écriture droite (valeur par défaut), <code>italic</code> : italique, <code>oblique</code> : police de type « oblique » (rare).
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Soulignement et autres « décos »

La propriété `text-decoration` permet d'obtenir différents effets : trait au-dessus ou en dessous des mots, texte barré ou clignotant.

TABLEAU 5–6 Propriété `text-decoration`

Propriété	<code>text-decoration</code>
Exemples	<pre>a:hover { text-decoration: none; } (suppression du soulignement des liens au passage de la souris) h1 { text-decoration: underline overline; } (titres de niveau 1 soulignés et surmontés)</pre>
Valeurs possibles	<code>none</code> : supprime tout soulignement et toute autre valeur attribuée à <code>text-decoration</code> (valeur par défaut) <code>underline</code> : souligné <code>overline</code> : surmonté (trait au-dessus) <code>line-through</code> : barré <code>blink</code> : clignotement du texte
Héritage	Cette propriété n'est <i>pas héritée</i> , contrairement aux autres propriétés liées aux caractères.

À NOTER Utilisation de `text-decoration`

- La prise en compte par les navigateurs de `blink` (clignotement du texte) est facultative.
- Il est possible de spécifier plusieurs valeurs pour `text-decoration`. Le deuxième exemple du tableau permet d'obtenir des titres de niveau 1 qui sont à la fois soulignés et surmontés d'un trait horizontal :

```
h1 { text-decoration: underline overline; }
```
- Dans la norme CSS 3, `text-decoration` devient un raccourci qui permet de définir, outre le type de « décoration » abordé ici, la couleur et le type de trait à utiliser (continu, tirets, pointillés). Ces propriétés sont détaillées dans le chapitre dédié aux CSS 3.

texte normal **texte en gras**

écriture droite *écriture en italique*

lettres normales lettres soulignées

FIGURE 5–2 Mise en œuvre des propriétés *font-weight* (*gras*), *font-style* (*italique*) et *text-decoration* (*soulignement*)

Majuscules et minuscules

Il est parfois utile de demander au navigateur d'afficher une partie du texte en minuscules ou en majuscules, quelle que soit la façon dont celui-ci aura été écrit initialement. C'est la propriété `text-transform` qui se charge de cette opération.

TABLEAU 5–7 Propriété `text-transform`

Propriété	<code>text-transform</code>
Exemple	<code>.pays { text-transform: uppercase; }</code>
Valeurs possibles	<code>capitalize</code> : une majuscule pour la première lettre de chaque mot <code>lowercase</code> : tout en minuscules <code>uppercase</code> : tout en majuscules <code>none</code> : écriture standard (valeur par défaut)
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>none</code> pour annuler l'héritage.

Petites majuscules

Pour mettre un texte en majuscules sans qu'il ne soit trop voyant, la propriété `font-variant` nous propose de l'écrire en petites majuscules : ces majuscules ont à peu près la taille des minuscules. Les lettres déjà écrites en majuscules conservent leur taille.

TABLEAU 5–8 Propriété `font-variant`

Propriété	<code>font-variant</code>
Exemple	<code>.ville { font-variant: small-caps; }</code>
Valeurs possibles	<code>normal</code> : texte normal (valeur par défaut) <code>small-caps</code> : tout en petites majuscules
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Surlignage de lettres

Grâce à la propriété `background-color`, il est possible d'attribuer un fond de couleur à certaines lettres ou à certains mots, à la manière du surlieur sur papier. Cette propriété servira aussi à choisir la couleur de fond d'un paragraphe ou d'un bloc, comme nous le verrons plus loin.

TABLEAU 5–9 Propriété `background-color` pour le texte

Propriété	<code>background-color</code>
Exemples	<code>strong { background-color: red; }</code> <code>p.remarque { background-color: #0000ff; }</code>
Valeurs possibles	nom de couleur prédéfini ou code RVB <code>transparent</code> (valeur par défaut)
Héritage	Cette propriété n'est <i>pas héritée</i> , mais la valeur par défaut <code>transparent</code> laisse voir la couleur de l'élément conteneur ou qui se trouve en dessous.

Décalage vers le haut ou le bas

Le décalage vertical de lettres ou de mots avec la propriété `vertical-align` sert dans deux cas de figure :

- pour l'affichage d'éléments en *indice* ou en *exposant* (attention : il s'agit d'un simple décalage en hauteur, il faudra y ajouter une réduction de la taille des lettres) ;
- pour le *centrage vertical d'une image* sur une ligne.

Cette propriété `vertical-align` est également applicable aux *cellules d'un tableau*, avec les valeurs suivantes : `baseline`, `top` (en haut), `middle` (au milieu) et `bottom` (en bas).

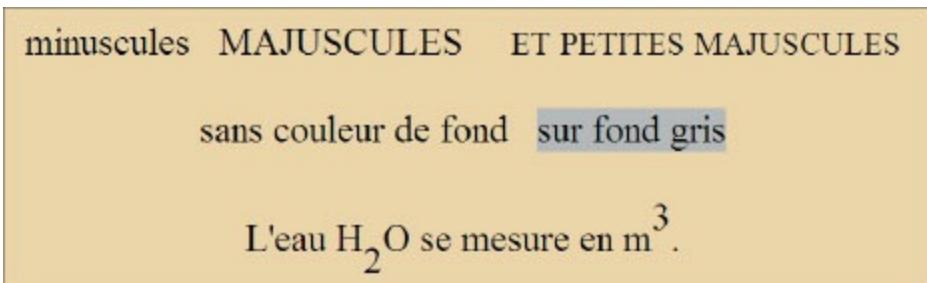


FIGURE 5–3 Application des propriétés `text-transform` (*majuscules*), `font-variant` (*petites majuscules*), `background-color` (*couleur de fond = surlignage*) et `vertical-align` (*indice/exposant*)

TABLEAU 5–10 Propriété `vertical-align`

Propriété	<code>vertical-align</code>
Exemples	<code>.exposant { vertical-align: super; }</code> <code>.indice { vertical-align: -50%; }</code>
Valeurs possibles	<code>baseline</code> : sur la base de la ligne (valeur par défaut) <code>sub</code> : indice <code>super</code> : exposant <code>middle</code> : au milieu de la ligne (centrage vertical sur la ligne) <code>top</code> ou <code>bottom</code> : alignement avec le haut ou le bas de la boîte parente <code>text-top</code> ou <code>text-bottom</code> : alignement avec le haut ou le bas du texte de la boîte parente

Héritage

valeur ou pourcentage : valeur *positive* pour un décalage vers le *haut* ; valeur *négative* pour un décalage vers le *bas*

Cette propriété n'est *pas héritée*.

Raccourci pour la mise en forme de caractères

Le raccourci `font` permet d'indiquer, en une seule propriété, les mises en forme qui concernent l'italique, les petites majuscules, la graisse, la taille et la police de caractères.

TABLEAU 5–11 Propriété raccourcie `font`

Propriété	<code>font</code>
Exemples	<pre>h3 { font: bold 1.2rem Verdana, sans-serif; } #note1 { font: italic 80% Garamond, serif; } .ville {font: bold small-caps 2rem Times, serif;}</pre>
Valeurs possibles	Dans cet ordre, valeurs des propriétés <code>font-style</code> , <code>font-variant</code> , <code>font-weight</code> , <code>font-size</code> , <code>line-height</code> (hauteur de ligne, voir plus loin) et <code>font-family</code> en dernier
Héritage	Propriété héritée

IMPORTANT Utilisation du raccourci `font`

- La propriété `font-family` est **obligatoire** et à placer à la fin, les autres sont facultatives.
- Les propriétés qui ne sont pas fournies sont réinitialisées à `normal`.
- La hauteur de ligne s'écrit après la taille des caractères et précédée d'une barre oblique. Par exemple : `18px/22px` pour une taille de caractères de 18 pixels et une hauteur de ligne de 22 pixels.
- Ce raccourci `font` n'inclut pas les propriétés `color`, `text-decoration`, `text-transform`, `background-color` et `vertical-align`.

Paragraphes et blocs de texte

Nos mots étant mis en forme, penchons-nous à présent sur les propriétés qui s'appliquent à des paragraphes ou à des blocs de texte tout entiers.

Alignment horizontal du texte

La propriété `text-align` modifie l'alignement horizontal comme le ferait un traitement de texte : paragraphe aligné à gauche, centré, aligné à droite ou justifié.

TABLEAU 5–12 Propriété `text-align`

Propriété	<code>text-align</code>
Exemples	<pre>p { text-align: justify; } .auteur { text-align: right; }</pre>
Valeurs possibles	<code>left</code> : aligné à gauche (par défaut), <code>right</code> : aligné à droite, <code>center</code> : centré, <code>justify</code> : justifié
Héritage	Propriété héritée. Pour retrouver la valeur initiale, utiliser <code>left</code> .

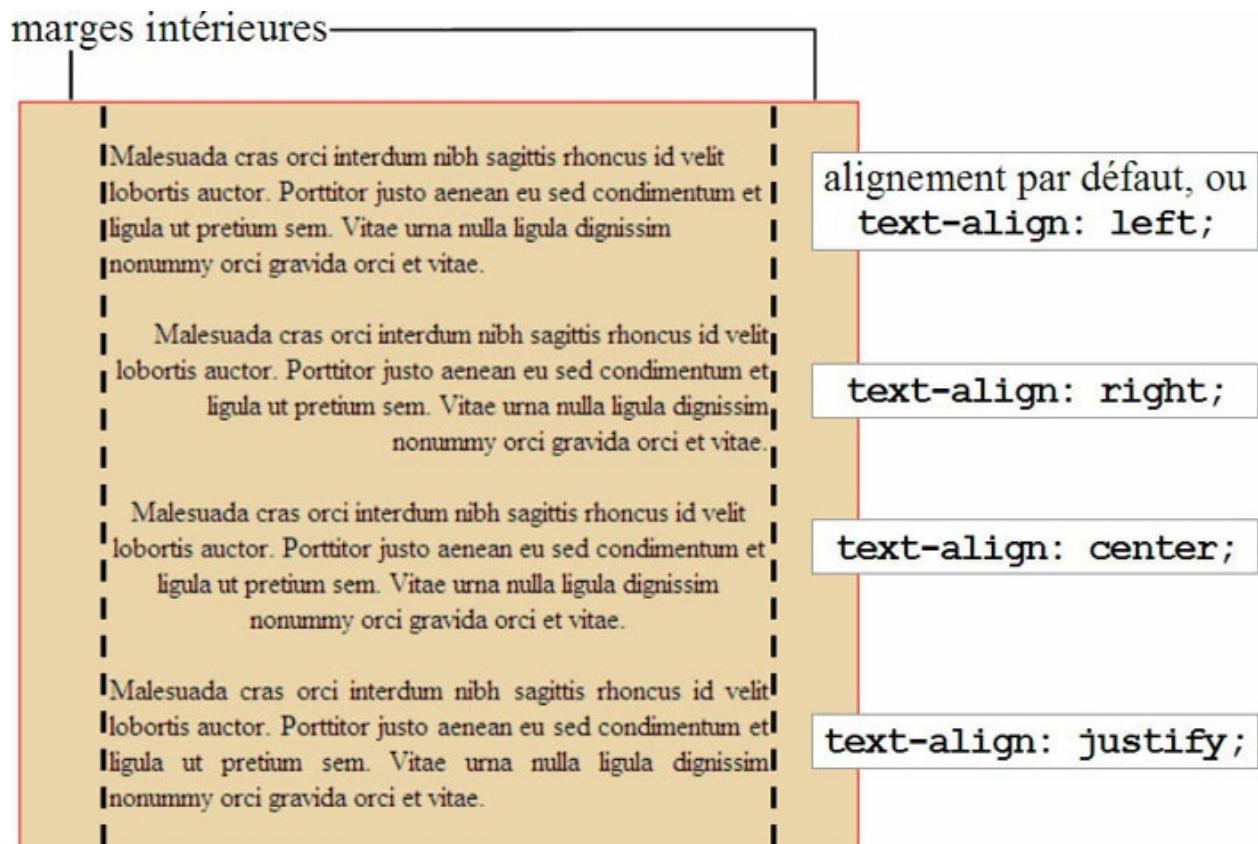


FIGURE 5–4 Effet de la propriété `text-align` (alignement horizontal du texte), lorsqu'elle prend successivement les valeurs `left`, `right`, `center` et `justify`.

Retrait de première ligne

Il s'agit de créer, avec la propriété `text-indent`, un retrait à gauche qui ne s'applique qu'à la première ligne de chacun des paragraphes concernés. Les autres lignes débutent sur la marge de gauche.

TABLEAU 5–13 Propriété `text-indent`

Propriété	<code>text-indent</code>
Exemple	<code>p { text-indent: 5rem; }</code>
Valeurs possibles	valeur positive ou négative , pour un retrait respectivement vers la droite ou vers la gauche de la première ligne ; valeur par défaut : 0 Mêmes unités que les tailles de polices de caractères, % inclus
Pourcentages	% de la largeur du bloc conteneur
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>0</code> pour annuler cet héritage.

ASTUCE Retrait négatif de première ligne

Pour obtenir un « retrait négatif de première ligne », c'est-à-dire tout le paragraphe en retrait sauf la première ligne, il faudra augmenter la marge interne de la même valeur, pour compenser ce retrait négatif. Comme nous le verrons dans le chapitre suivant, c'est la propriété `padding-left` qui règle la marge interne de gauche. Par exemple, si la marge interne de notre document est initialement de 20 pixels, un retrait négatif de 10 pixels pour la première ligne peut s'écrire :

```
p { text-indent: -10px; padding-left: 30px; }
```

La première ligne reste alors à sa place habituelle, et le reste du paragraphe est en retrait de 10 pixels.

marge
intérieure
gauche

retrait positif de la première ligne

Malesuada cras orci interdum nibh sagittis rhoncus id velit lobortis auctor. Porttitor justo aenean eu sed condimentum et ligula ut pretium sem. Vitae urna nulla ligula dignissim nonummy orci gravida orci et vitae.

Malesuada cras orci interdum nibh sagittis rhoncus id velit lobortis auctor. Porttitor justo aenean eu sed condimentum et ligula ut pretium sem. Vitae urna nulla ligula dignissim nonummy orci gravida orci et vitae.

retrait négatif de la première ligne

FIGURE 5–5 Retraits positif et négatif de la première ligne d'un paragraphe, avec la propriété `text-indent`

Interligne minimum

La hauteur d'une ligne de texte peut être modifiée à l'aide de la propriété `line-height`. Il en résulte un centrage vertical automatique du texte sur la hauteur ainsi spécifiée.

TABLEAU 5–14 Propriété `line-height`

Propriété	<code>line-height</code>
Exemple	<code>a.menu { line-height: 2rem; }</code>
Valeurs possibles	<code>normal</code> (valeur standard) ou valeur positive pour régler l'espacement entre les lignes Mêmes unités que les tailles de polices, ou % de cette taille, ou encore une valeur entière ou décimale qui est le nombre de fois l'interlignage : 1 (interlignage minimal, resserré), 1.25 (interligne standard), 2.5 (interligne double), etc.
Pourcentage	% de la taille de police utilisée
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

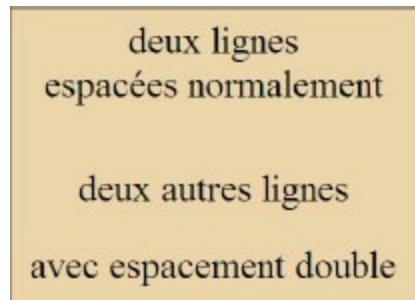


FIGURE 5–6 Modification de l'interlignage avec la propriété `line-height`

Espacement entre les lettres

La propriété `letter-spacing` fixe l'espacement entre les lettres d'un mot, aussi appelé *crénage* ou encore *interlettrage*.

TABLEAU 5–15 Propriété letter-spacing

Propriété	<code>letter-spacing</code>
Exemples	<code>.pluslarge { letter-spacing: 0.5rem; }</code> <code>.moinslarge { letter-spacing: -1px; }</code>
Valeurs possibles	<code>normal</code> (par défaut), valeur positive ou négative , respectivement pour augmenter ou diminuer l'interlettrage Mêmes unités que les tailles de polices, sauf les pourcentages
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Espacement entre les mots

Comme son nom l'indique, la propriété `word-spacing` ajuste l'espacement entre les mots.

TABLEAU 5–16 Propriété `word-spacing`

Propriété	<code>word-spacing</code>
Exemples	<code>.grandsespaces { word-spacing: 0.5rem; }</code> <code>.petitsespaces { word-spacing: -2px; }</code>
Valeurs possibles	<code>normal</code> (par défaut), valeur positive ou négative , respectivement pour augmenter ou diminuer l'espace entre les mots Mêmes unités que les tailles de polices, sauf %
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

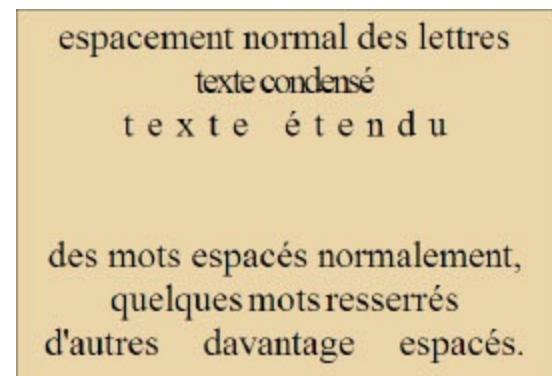


FIGURE 5–7 Application des propriétés `letter-spacing` pour condenser ou étirer le texte et `word-spacing` pour écarter ou resserrer les mots

Conservation des espaces et sauts de ligne saisis

Appliquée à un paragraphe ou un bloc de texte, la propriété `white-space` indique s'il faut ou non conserver tous les espaces et retours à la ligne tapés dans le code HTML. En même temps, elle permet d'accepter ou non le retour à la ligne automatique, qui se produit normalement lorsque le texte arrive sur le bord droit du bloc.

TABLEAU 5–17 Propriété `white-space`

Propriété	<code>white-space</code>
Exemple	<code>p.extraits { white-space: pre; }</code>
Valeurs possibles	<p><code>normal</code> (par défaut) : affichage d'un seul espace à la place de tous les espaces et retours à la ligne qui séparent deux mots ; le retour à la ligne s'effectue avec la balise <code>
</code> et est automatique en fin de ligne du bloc de texte.</p> <p><code>nowrap</code> : espaces successifs fusionnés en un seul et retours à la ligne ignorés, comme pour la valeur <code>normal</code> ; en revanche, il n'y a pas de retour automatique en fin de ligne, seul <code>
</code> peut produire un retour à la ligne.</p> <p><code>pre</code> : conservation de tous les espaces et de tous les retours à la ligne saisis dans le code source, mais il n'y a pas de retour automatique en fin de ligne du bloc de texte.</p> <p><code>pre-line</code> : espaces successifs fusionnés en un seul, conservation des retours à la ligne du code source ; le retour automatique en fin de ligne est activé.</p> <p><code>pre-wrap</code> : tous les espaces du code source sont conservés, mais les retours à la ligne qui ne sont pas indiqués par <code>
</code> sont ignorés ; le retour automatique en fin de ligne est activé.</p>
Héritage	Propriété héritée. Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Modification du curseur de la souris

La propriété `cursor` n'a pas d'effet sur l'affichage de la page en elle-même, mais sur l'apparence du curseur de la souris lorsqu'il passe sur le texte concerné.

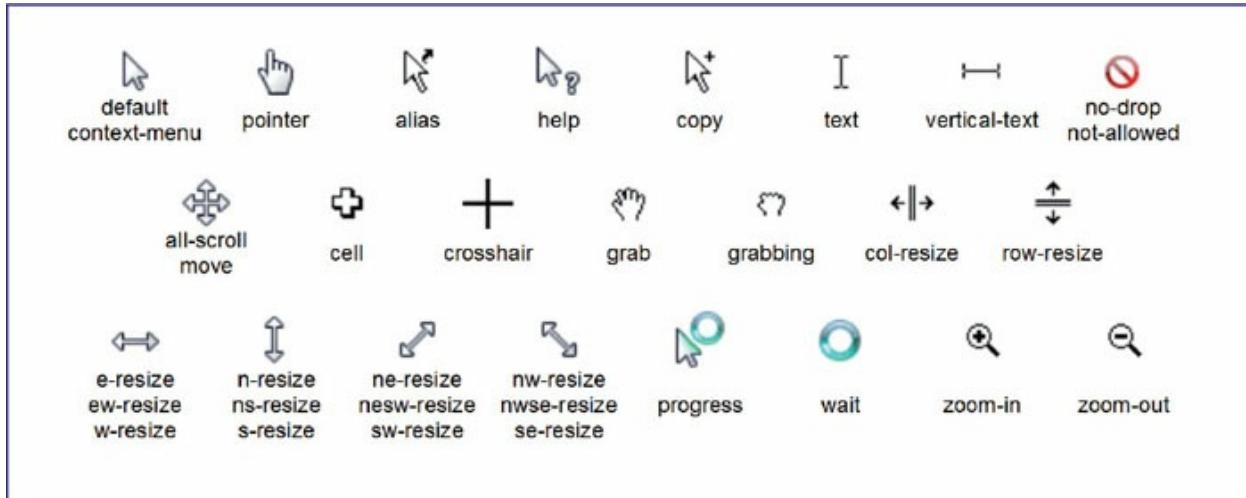


FIGURE 5–8 Différents types de curseur possibles avec la propriété `cursor`, dont les doubles flèches de redimensionnement qui sont orientées suivant les directions Nord (n), Sud (s), Est (e) ou Ouest (w).

TABLEAU 5–18 Propriété cursor

Propriété	<code>cursor</code>
Exemple	<code>.aide { cursor: help; }</code>
Valeurs possibles	auto (valeur par défaut) : la forme est fonction du contexte. <code>default</code> : généralement une flèche blanche, ou comme pour de nombreuses valeurs prédéfinies (voir la liste et les formes de curseur associées sur la figure 5–8)
Héritage	Propriété héritée. Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Affichage automatique d'un contenu

La propriété `content` s'utilise avec les pseudo-éléments `::before` et `::after`. Elle permet d'afficher automatiquement un contenu, avant ou après l'élément concerné : cela peut être un texte, un numéro, des guillemets, une image, etc.

TABLEAU 5–19 Propriété `content`

Propriété	<code>content</code>
Exemples	<pre>p.note::before { content: "Nota bene : "; } p.remarque::before {content: url(crayon.gif)" ";} li::before { content: "[" counter(chapitre,lower-roman) "]";} .citation::before { content: open-quote; } .citation::after { content: close-quote; } img::after { content: attr(title); }</pre>
Valeurs possibles	<p>chaîne de caractères : à écrire entre guillemets (valeur par défaut : la chaîne vide "")</p> <p><code>url(fichier)</code> : fichier à utiliser (image à afficher, son à diffuser, etc.)</p> <p><code>counter(nom)</code> OU <code>counter(nom, style)</code> : nom de compteur (voir plus loin <code>counter-reset</code>) et style facultatif, à choisir parmi les valeurs possibles de <code>list-style-type</code> (style par défaut : décimal, <code>none</code> pour ne rien afficher)</p> <p><code>counters(nom, chaîne)</code> OU <code>counters(nom, chaîne, style)</code> : avec chaîne séparatrice (par exemple ".." pour obtenir §3 puis §3.1, 3.2, ...).</p> <p><code>open-quote</code>, <code>close-quote</code> : guillemets de début et de fin définis par la propriété <code>quote</code></p> <p><code>no-open-quote</code>, <code>no-close-quote</code> : pas de guillemets, mais un niveau d'imbrication de guillemets qui est décompté pour les prochains guillemets qui seront affichés par <code>content</code></p> <p><code>attr(propriété)</code> : valeur de l'attribut indiqué de l'élément concerné par la propriété <code>content</code>, chaîne vide si l'attribut est absent</p> <p>Chacune de ces valeurs peut être suivie d'une chaîne, par exemple " " pour un espace de séparation ou "-" pour un tiret.</p>
Héritage	<code>Non</code>

Guillemets à utiliser

La propriété quotes sert à définir les types de guillemets qui serviront aux différents niveaux d'imbrication.

Ces guillemets seront utilisables avec la balise <q> ou la propriété content lorsqu'elle affiche des guillemets ouvrants ou fermants, à l'aide des valeurs content: open-quote; OU content: close-quote;.

TABLEAU 5–20 Propriété quotes

Propriété	<code>quotes</code>
Exemples	<pre>q { quotes: "'''"'"'"'; } q.guill2 { quotes: "«" "»" "<" ">"; }</pre>
Valeurs possibles	Sous forme de chaînes de caractères : <ul style="list-style-type: none">• guillemets d'ouverture puis de fermeture, pour le premier niveau de guillemets ;• puis éventuellement guillemets d'ouverture, puis de fermeture pour le deuxième niveau (guillemets imbriqués) ;• et ainsi de suite selon le nombre de niveaux d'imbrication souhaités.
Héritage	Propriété héritée

Réinitialisation d'un compteur

Il s'agit, avec la propriété counter-reset, de définir un compteur en le nommant, tout en l'initialisant à zéro ou à une valeur donnée.

Ce compteur sera utilisable grâce à la propriété content.

ATTENTION Réinitialisation de plusieurs compteurs

Si deux compteurs doivent être réinitialisés pour le même élément, il faut réunir ces deux réinitialisations, en n'écrivant qu'une seule fois la propriété counter-reset. Exemple :

```
h1 { counter-reset: page 2 section 1; }
```

TABLEAU 5–21 Propriété counter-reset

Propriété	counter-reset
Exemples	<pre>h1 { counter-reset: chapitre; } h1.nouveau { counter-reset: numpage -1; }</pre>
Valeurs possibles	Nom du compteur , puis éventuellement valeur initiale (si elle est différente de 0) : <i>nombre entier</i> positif ou négatif, ou <i>none</i> = pas de compteur (c'est la valeur par défaut).
Héritage	Non

Incrémantion d'un compteur

Chaque fois qu'un compteur est utilisé à l'aide de la propriété `content`, il s'incrémentera d'une valeur donnée, qui peut être définie par la propriété `counter-increment`.

TABLEAU 5–22 Propriété `counter-increment`

Propriété	<code>counter-increment</code>
Exemples	<pre>h2 { counter-increment: chapitre; } .instruction { counter-increment: numligne 10; }</pre>
Valeurs possibles	Nom du compteur , puis éventuellement la valeur d'incrémantion (si elle est différente de 1) : nombre entier positif ou négatif, ou <code>none</code> = pas d'incrémantion du compteur (valeur par défaut)
Héritage	<i>Non</i>

À NOTER Ordre des opérations

L'incrémantion du compteur s'effectue **avant** son utilisation.

Sens de l'écriture

Certaines langues qui s'écrivent de droite à gauche nécessitent l'emploi de la propriété `direction` pour préciser le sens de lecture.

TABLEAU 5–23 Propriété `direction`

Propriété	<code>direction</code>
Exemples	<code>body { direction: ltr; } .yiddish { direction: rtl; }</code>
Valeurs possibles	<code>ltr</code> : de gauche à droite (<i>left to right</i>) - valeur par défaut <code>rtl</code> : de droite à gauche (<i>right to left</i>)
Héritage	Propriété héritée

À NOTER Caractères Unicode

Le sens de lecture des différents encodages *Unicode*, dont fait partie le codage `utf-8` préconisé dès le chapitre 2, est reconnu automatiquement. Leur utilisation nous dispense donc de spécifier cette propriété `direction`.

Écriture bidirectionnelle

Cette propriété `unicode-bidi` (texte Unicode bidirectionnel) est rarement utilisée ; elle permet d'utiliser plusieurs sens de lecture dans un même bloc.

Elle peut servir aux amateurs d'exotisme, pour inclure des citations en arabe, farsi, hébreu ou urdu à l'intérieur d'un paragraphe en français...

TABLEAU 5–24 Propriété `unicode-bidi`

Propriété	<code>unicode-bidi</code>
Exemples	<pre>span.citation { direction: rtl; unicode-bidi: embed; } span.sens2 { direction: rtl; unicode-bidi: bidi-override; }</pre>
Valeurs possibles	<p><code>normal</code> (par défaut) : à l'intérieur de chaque groupe de mots homogène, les caractères Unicode s'écrivent dans leur sens d'écriture naturel, fonction de la langue et reconnu automatiquement.</p> <p><code>embed</code> : les caractères s'écrivent dans leur sens naturel, les groupes de mots homogènes (composés de caractères qui s'écrivent dans le même sens) sont placés dans l'ordre défini par la propriété <code>direction</code> (voir la figure 5–9).</p> <p><code>bidi-override</code> : tous les caractères sont écrits les uns après les autres, dans le sens indiqué par la propriété <code>direction</code>.</p>
Héritage	<code>Non</code>

EXEMPLE Lignes comprenant des textes en français et en hébreu

Cet exemple est inspiré du test de conformité CSS 2 de Daniel Glazman, disponible sur le site des Éditions Eyrolles (modèle de formatage, groupe de tests 6) à l'adresse suivante :

➤ <http://www.editions-eyrolles.com/auteurs/Glazman/tests/vfm/vfm14.htm>

Le texte saisi contient les mots français *un*, *deux*, *trois* mélangés aux caractères hébreux *aleph* ♪ (U+05d0), *beth* ♩ (U+05d1) et *tav* ♪ (U+05ea).

Ordre de la saisie

♪ un deux ♩ ♪ trois

soit dans le code : א un deux ב ת trois

Affichage à l'écran

- Sans *direction*: *rtl*; ni *unicode-bidi*: ♪ un deux ♩ ♪ trois
- Avec *direction: rtl*; sans *unicode-bidi*: ♪ un deux ♩ ♪ trois
- Avec *direction: rtl*; et les valeurs suivantes pour *unicode-bidi*:
 - *normal*: ♪ un deux ♩ ♪ trois
 - *embed*: trois ♩ ♪ un deux ♪
 - *bidi-override*: siort ♩ ♪ xued nu ♪

Noter que les caractères *beth* ♩ et *tav* ♪ sont toujours inversés, que les propriétés *direction* et *unicode-bidi*. soient spécifiées ou non.

FIGURE 5–9 Utilisation de la propriété *unicode-bidi*

Bordures

Les propriétés de bordure s'appliquent aux blocs de texte et aux éléments remplacés, qui possèdent des dimensions comme les images.

Style de bordure

La propriété `border-style` précise le type des traits de contour à afficher autour des blocs de texte concernés.

ATTENTION Propriété obligatoire pour afficher une bordure

La valeur par défaut de `border-style` étant `none`, il est indispensable de préciser un style de bordure pour que celle-ci soit visible. Tant qu'un style de bordure n'a pas été défini, donner une épaisseur et une couleur de bordure ne change rien à l'affichage.

TABLEAU 5–25 Propriété `border-style`

Propriété	<code>border-style</code>
Exemples	<code>h2 { border-style: solid; } p.note { border-style: double; }</code>
Valeurs possibles	<code>none</code> (valeur par défaut) ou <code>hidden</code> : aucune bordure, <code>solid</code> : trait plein, <code>dotted</code> : pointillés, <code>dashed</code> : tirets, <code>double</code> : trait plein double, <code>groove</code> : en creux, <code>ridge</code> : en relief, <code>inset</code> : creux ombré, <code>outset</code> : relief ombré L'aspect de ces bordures est donné par la figure 5–10.
Héritage	<code>Non</code>

PRÉCISION Différence entre `none` et `hidden`

C'est dans les tableaux qu'il existe une différence entre les valeurs `none` et `hidden` pour la propriété `border-style` :

- `none` = aucune bordure, sauf si une cellule voisine en possède une ;
- `hidden` = aucune bordure, dans tous les cas.

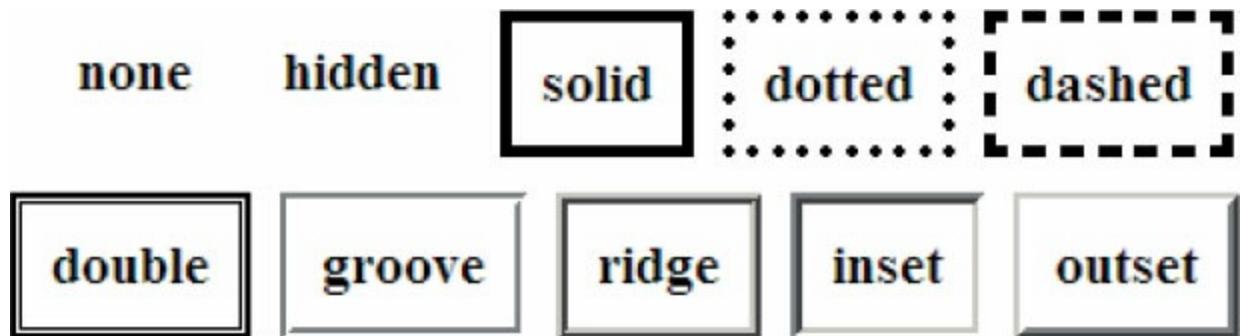


FIGURE 5–10 Différents types de bordure

Styles de bordure pour chaque côté

Il existe quatre propriétés distinctes pour définir un style de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 5–26 Style de bordure pour chaque côté

Propriété	border-top-style : style de la bordure du haut border-right-style : style de la bordure de droite border-bottom-style : style de la bordure du bas border-left-style : style de la bordure de gauche
-----------	---

RACCOURCI Utilisation de border-style pour singulariser chaque côté

La propriété générale `border-style` peut être utilisée pour préciser le style de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche

```
p { border-style: solid double; }
```
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas

```
p { border-style: dashed solid dotted; }
```
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche

```
p { border-style: dashed dotted solid double; }
```

Épaisseur de bordure

L'épaisseur du trait de contour est indiquée par la propriété `border-width`.

TABLEAU 5–27 Propriété border-width

Propriété	<code>border-width</code>
Exemple	<code>p.note { border-width: 2px; }</code>
Valeurs possibles	<code>thin</code> = bordure fine, <code>medium</code> = bordure moyenne, <code>thick</code> = bordure épaisse ou valeur numérique en rem, px... (mais pas en %)
Héritage	<i>Non</i>

Épaisseur de bordure pour chaque côté

Cette épaisseur de bordure peut être précisée pour chacun des quatre côtés de l'élément concerné, à l'aide de quatre propriétés distinctes.

TABLEAU 5–28 Épaisseur de bordure pour chaque côté

Propriété	border-top-width : épaisseur de la bordure du haut border-right-width : épaisseur de la bordure de droite border-bottom-width : épaisseur de la bordure du bas border-left-width : épaisseur de la bordure de gauche
-----------	---

RACCOURCI Utilisation de border-width pour singulariser chaque côté

La propriété générale `border-width` peut être utilisée pour préciser l'épaisseur de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche

```
p { border-width: 1rem 2rem; }
```
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas

```
p { border-width: thin medium thick; }
```
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche

```
p { border-width: 1px 3px 3px 1px; }
```

Couleur de bordure

Par défaut, le contour d'un bloc est de la même couleur que le texte (valeur de la propriété `color`, si elle a été définie pour cet élément, sinon noir par défaut). Pour modifier cette couleur de bordure, il faut utiliser la propriété `border-color`.

TABLEAU 5–29 Propriété `border-color`

Propriété	<code>border-color</code>
Exemples	<pre>div.remarque { border-color: gray; } p.utile { border-color: #ff0088; }</pre>
Valeurs possibles	nom de couleur prédéfini ou code RVB <code>transparent</code> = bordure invisible
Héritage	<i>Non</i>

Couleur de bordure pour chaque côté

Quatre propriétés distinctes permettent de définir la couleur de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 5–30 Couleur de bordure pour chaque côté

Propriété	
	<code>border-top-color</code> : couleur de la bordure du haut
	<code>border-right-color</code> : couleur de la bordure de droite
	<code>border-bottom-color</code> : couleur de la bordure du bas
	<code>border-left-color</code> : couleur de la bordure de gauche

RACCOURCI Utilisation de `border-color` pour singulariser chaque côté

La propriété générale `border-color` peut être utilisée pour préciser la couleur de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche
`p { border-color: blue red; }`
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas
`p { border-color: blue gray green; }`
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche
`p { border-color: blue gray gray blue; }`

Raccourci pour toutes les propriétés de bordure

L'ensemble des propriétés qui définissent les bordures (épaisseur, style et couleur) peut être déclaré à l'aide du raccourci `border`.

TABLEAU 5–31 Propriété raccourcie `border`

Propriété	<code>border</code>
Exemple	<code>h2.chapitre { border: 5px gray groove; }</code>
Valeurs possibles	Toutes les valeurs des propriétés <code>border-width</code> (facultative), <code>border-style</code> (obligatoire) et <code>border-color</code> (facultative)
Héritage	<code>Non</code>

IMPORTANT Propriété `border` : le style de bordure est obligatoire

Lorsqu'une des propriétés (épaisseur, style ou couleur de bordure) n'est pas précisée dans ce raccourci, elle est initialisée à sa valeur par défaut.

Il en résulte que *le style de bordure est obligatoire*, sa valeur par défaut étant `none` (pas de bordure, quelles que soient l'épaisseur et la couleur choisies).

Raccourci des propriétés de bordure pour chaque côté

Il existe quatre raccourcis distincts pour définir les propriétés de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 5–32 Raccourcis des propriétés de bordure pour chaque côté

Propriété	<code>border-top</code> : propriétés de la bordure du haut <code>border-right</code> : propriétés de la bordure de droite <code>border-bottom</code> : propriétés de la bordure du bas <code>border-left</code> : propriétés de la bordure de gauche
Exemple	<code>.titre { border-top: 3px solid red; border-right: 1px dotted red; }</code>
Valeurs possibles	Toutes les valeurs des propriétés <code>border-width</code> (facultative), <code>border-style</code> (obligatoire) et <code>border-color</code> (facultative).
Héritage	<i>Non</i>

Contour superposé à un élément

La propriété `outline` affiche une bordure qui se superpose à l'élément, sans augmenter ses dimensions.

Les propriétés utilisables sont les suivantes :

- `outline-style` : mêmes valeurs que `border-style`, sauf `hidden` ;
- `outline-width` : mêmes valeurs que `border-width` ;
- `outline-color` : mêmes valeurs que `border-color`, plus la valeur `invert` (couleur inverse de la couleur de fond) ;
- `outline` : raccourci pour `outline-width`, `outline-style` et `outline-color` ;
- `outline-offset` : dimension donnant l'écart entre la bordure définie par `outline` et celle produite par `border` (valeur par défaut : 0).

Ces propriétés `outline`, `outline-width`, `outline-style`, `outline-color` et `outline-offset` ne sont *pas héritées*.

Images et couleurs d'arrière-plan

Les propriétés suivantes s'appliquent aux blocs de texte et aux éléments remplacés, comme les images ou les boutons de formulaire.

Elles servent à agrémenter l'arrière-plan de l'élément concerné, soit d'une image, soit d'une couleur unie. Attention cependant : il faut éviter une cacophonie de couleurs !

Couleur d'arrière-plan

Il est possible de définir une couleur d'arrière-plan sur l'ensemble d'un bloc, à l'aide de la propriété `background-color`.

TABLEAU 5–33 Propriété `background-color`

Propriété	<code>background-color</code>
Exemple	<code>p.relief { background-color: yellow; }</code>
Valeurs possibles	nom de couleur prédéfini ou code RVB <code>transparent</code> (valeur par défaut)
Héritage	Cette propriété n'est <i>pas héritée</i> , mais la valeur par défaut <code>transparent</code> laisse voir la couleur de l'élément conteneur ou se trouvant en dessous.

Image d'arrière-plan

La propriété `background-image` permet d'afficher une image en arrièreplan d'un bloc. Sa valeur est l'adresse de l'image, qui peut être relative (nom du fichier image situé par rapport au dossier qui contient la page web) ou absolue (URL se terminant par un nom de fichier image).

TABLEAU 5–34 Propriété `background-image`

Propriété	<code>background-image</code>
Exemples	<code>body {background-image: url(images/maison.png);}</code> <code>.pub {background-image: url(http://www.sncf.com/logo.gif);}</code>
Valeurs possibles	<code>url</code> (nom d'image avec chemin relatif ou absolu) ou <code>none</code> : aucune image (valeur par défaut)
Héritage	<i>Non</i>

À NOTER **Guillemets facultatifs autour des noms de fichiers images**

Grâce aux parenthèses de `url(...)`, les guillemets ou apostrophes – qui logiquement entourent le nom du fichier image ou l'URL – sont *facultatifs*.

Répétition ou non de l'image d'arrière-plan

Lorsque l'image est en arrière-plan d'un bloc, mais possède des dimensions inférieures à celles de ce bloc, elle est automatiquement répétée, horizontalement et verticalement. L'annulation de l'une de ces répétitions ou des deux s'effectue à l'aide de la propriété `background-repeat`.

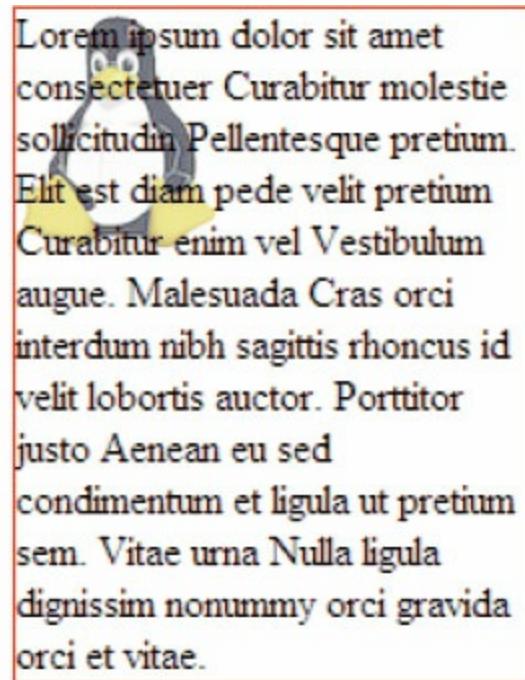
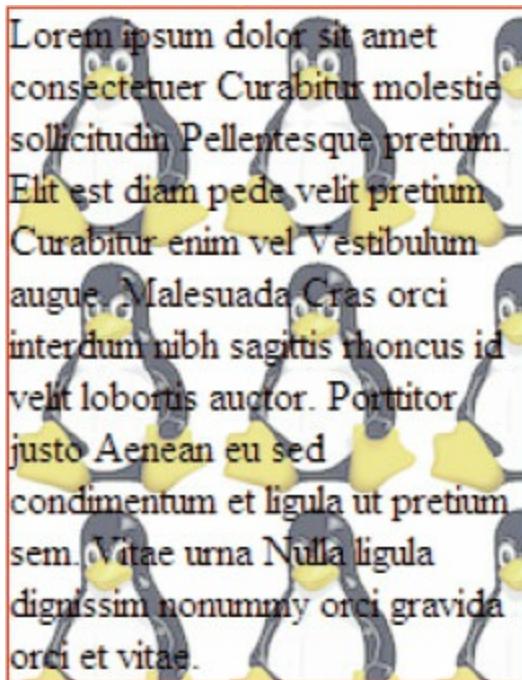


FIGURE 5–11 Image d'arrière-plan plus petite que le bloc, avec répétition (valeur par défaut), puis avec la propriété `background-repeat: no-repeat`

TABLEAU 5–35 Propriété `background-repeat`

Propriété	<code>background-repeat</code>
Exemples	<pre>body { background-repeat: repeat-y; } .pub { background-repeat: no-repeat; }</pre>
Valeurs possibles	<p><code>repeat</code> : répétition horizontale et verticale (valeur par défaut)</p> <p><code>repeat-x</code> : répétition horizontale seulement</p> <p><code>repeat-y</code> : répétition verticale seulement</p> <p><code>no-repeat</code> : pas de répétition</p>
Héritage	<code>Non</code>

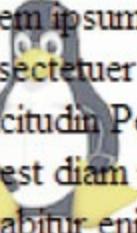
Alignment de l'image d'arrière-plan

Avec la propriété `background-position`, il est possible de préciser la position horizontale et la position verticale de l'image d'arrière-plan, à l'intérieur du bloc dont elle constitue le fond.

Cette position est exprimée par rapport aux bords de l'élément.

TABLEAU 5–36 Propriété `background-position`

Propriété	<code>background-position</code>
Exemples	<code>body { background-position: center top; }</code> <code>.pub { background-position: left center; }</code>
Valeurs possibles	Une valeur, ou deux valeurs données par des noms ou des nombres (dimension relative ou absolue, souvent exprimée en %) Première valeur pour l'alignement horizontal : <code>left</code> (valeur par défaut), <code>center</code> , <code>right</code> ou nombre en % ou px (<code>0% = left</code> , <code>100% = right</code>). Deuxième valeur pour l'alignement vertical : <code>top</code> (valeur par défaut), <code>center</code> , <code>bottom</code> ou nombre en % ou px (<code>0% = top</code> , <code>100% = bottom</code>).
Pourcentage	% de la taille de la boîte elle-même
Héritage	<i>Non</i>

Lorem ipsum dolor sit amet
consectetuer Curabitur molestie
sollicitudin Pellentesque pretium.
Elit est diam pede velit pretium
Curabitur enim vel Vestibulum
augue. Malesuada Cras orci
interdum nibh sagittis rhoncus id
velit lobortis auctor. Porttitor
justo Aenean eu sed
condimentum et ligula ut pretium
sem. Vitae urna Nulla ligula
dignissim nonummy orci gravida
orci et vitae.

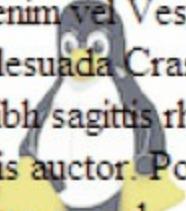
Lorem ipsum dolor sit amet
consectetuer Curabitur molestie
sollicitudin Pellentesque pretium.
Elit est diam pede velit pretium
Curabitur enim vel Vestibulum
augue. Malesuada Cras orci
interdum nibh sagittis rhoncus id
velit lobortis auctor. Porttitor
justo Aenean eu sed
condimentum et ligula ut pretium
sem. Vitae urna Nulla ligula
dignissim nonummy orci gravida
orci et vitae.

FIGURE 5–12 Image d’arrière-plan dans sa position par défaut (coin supérieur gauche), puis placée au centre du bloc

À NOTER Utilisation de nombres pour background-position

Un conseil et une astuce pour l’emploi de la propriété background-position avec des nombres :

- évitez de mélanger valeurs fixes (par exemple 10px) et relatives (comme 20%) ;
- il est possible d’indiquer des nombres négatifs, pour « rogner » l’image.

Fixation de l'image d'arrière-plan

Lorsque l'internaute fait défiler une page contenant une image d'arrièreplan, celle-ci se déplace sur l'écran en même temps que le texte, car elle est attachée au coin supérieur gauche du bloc auquel elle est associée. Si cette image doit rester fixe lorsque le texte défile sur l'écran, il faut le signaler avec la propriété `background-attachment`.

TABLEAU 5–37 Propriété `background-attachment`

Propriété	<code>background-attachment</code>
Exemples	<pre>body { background-attachment: scroll; } .pub { background-attachment: fixed; }</pre>
Valeurs possibles	<code>scroll</code> : l'image défile avec le contenu (valeur par défaut). <code>fixed</code> : l'image reste fixe lors du défilement, seul défile le contenu qui est au premier plan.
Héritage	<i>Non</i>

Raccourcis pour les arrière-plans

Comme pour les bordures, il existe une propriété raccourcie pour l'image d'arrière-plan et ses caractéristiques.

TABLEAU 5–38 Raccourci background

Propriété	<code>background</code>
Exemple	<code>h1 { background: blue url(logo.png) 50% repeat-x fixed; }</code>
Valeurs possibles	Valeurs de <code>background-color</code> , <code>background-image</code> , <code>background-repeat</code> , <code>background-attachment</code> et <code>background-position</code> dans un ordre quelconque
Héritage	<i>Non</i>

Listes à puces ou numérotées

Nous allons étudier à présent les propriétés associées aux listes. Elles permettent de choisir les types de puces ou de numéros, ou encore de remplacer les puces par des images.

Type de puce ou de numérotation

La propriété `list-style-type` indique quel type générique de puce ou quel mode de numérotation doit utiliser la liste concernée.

TABLEAU 5–39 Propriété `list-style-type`

Propriété	<code>list-style-type</code>
Exemples	<code>ul { list-style-type: square; }</code> <code>ol { list-style-type: upper-roman; }</code>
Valeurs possibles	<p><i>Liste à puces :</i></p> <p><code>disc</code> (cercle plein - valeur par défaut) ; <code>circle</code> (cercle vide) ; <code>square</code> (carré plein).</p> <p><i>Liste numérotée :</i></p> <p><code>decimal</code> (1, 2...- valeur par défaut) ; <code>decimal-leading-zero</code> (01, 02...) ; <code>lower-roman</code> (i,ii...), <code>upper-roman</code> (I,II...) ; <code>lower-latin</code> = <code>lower-alpha</code> (a, b, c...) ; <code>upper-latin</code> = <code>upper-alpha</code> (A, B, C...) ; <code>lower-greek</code> (α, β, γ...) ; et de nombreuses notations étrangères comme <code>armenian</code>, <code>georgian</code>, <code>hebrew</code> (hébreux), <code>cjk-ideographic</code> (chinois), <code>hiragana</code>, <code>hiragana-iroha</code>, <code>katakana</code>, <code>katakana-iroha</code> (quatre notations japonaises), etc.</p> <p><i>Pas de puce ni de numéro :</i> <code>none</code></p>
Héritage	Propriété héritée. Pour retrouver la valeur initiale, utiliser <code>disc</code> pour les listes à puces et <code>decimal</code> pour les listes numérotées.

La liste complète des notations étrangères prédéfinies pour `list-style-type` est disponible sur cette page du W3C :

<https://www.w3.org/TR/css-counter-styles-3/#predefined-counters>

Utilisation d'une image comme puce

Grâce à la propriété `list-style-image`, n'importe quelle image peut être utilisée comme puce. Il faudra évidemment s'assurer que la taille, la forme et la couleur de cette image conviennent à un tel usage.

TABLEAU 5–40 Propriété `list-style-image`

Propriété	<code>list-style-image</code>
Exemples	<code>ul { list-style-image: url(image/puce.gif); } li { list-style-image: url(http://www.top.org/logo.gif); }</code>
Valeurs possibles	<code>url</code> (nom d'image avec chemin relatif ou absolu) ou <code>none</code> : aucune image (valeur par défaut)
Héritage	Propriété héritée. Pour retrouver la valeur initiale, utiliser <code>none</code> .

À NOTER Guillemets facultatifs

Comme pour la propriété `background-image`, les guillemets ou apostrophes autour du nom de fichier image ou d'une URL sont *facultatifs*, en raison de la présence des parenthèses dans l'expression `url(...)`.

• Vestibulum nec nisi id augue malesuada congue.	▪ Vestibulum nec nisi id augue malesuada congue.	 Vestibulum nec nisi id augue malesuada congue.
• Mauris consequat pharetra ligula.	▪ Mauris consequat pharetra ligula.	 Mauris consequat pharetra ligula.
• Etiam posuere faucibus lorem.	▪ Etiam posuere faucibus lorem.	 Etiam posuere faucibus lorem.

FIGURE 5–13 Listes avec différentes puces : disque plein (par défaut), carré plein (`list-style-type: square`) et image (avec la propriété `list-style-image`)

Position de la puce

La puce, l'image qui la remplace ou le numéro pourront être placés, suivant la valeur de la propriété `list-style-position` et comme le montre la figure 5–14 :

- soit à gauche du paragraphe (c'est la configuration standard) ;
- soit à l'intérieur du paragraphe, avec un décalage de la première ligne pour chaque item de la liste.

TABLEAU 5–41 Propriété `list-style-position`

Propriété	<code>list-style-position</code>
Exemples	<code>ul { list-style-position: outside; }</code> <code>ol { list-style-position: inside; }</code>
Valeurs possibles	<code>outside</code> : la puce est dans la marge (valeur par défaut) <code>inside</code> : la puce fait partie de la première ligne du paragraphe
Héritage	Propriété héritée. Retour à la valeur initiale avec <code>outside</code> .

- Sed lectus. Nunc vehicula, arcu in consectetur et sodales, sed lacinia elit arcu eget augue.
- Aliquam et lacus. Nunc faucibus consectetur et leo. Sed ante ut magna dignissim elementum.

outside

- Sed lectus. Nunc vehicula, arcu in consectetur et sodales, sed lacinia elit arcu eget augue.
- Aliquam et lacus. Nunc faucibus consectetur et leo. Sed ante ut magna dignissim elementum.

inside

FIGURE 5–14 Puces à l'intérieur ou à l'extérieur des paragraphes

Raccourci pour toutes les propriétés de liste

L’ensemble des propriétés qui permettent de paramétriser les listes (type de puce ou de numérotation, fichier image remplaçant les puces, position des puces ou numéros) peut être défini à l’aide du raccourci `list-style`.

TABLEAU 5–42 Propriété raccourcie `list-style`

Propriété	<code>list-style</code>
Exemple	<code>li { list-style: circle inside; }</code>
Valeurs possibles	Toutes les valeurs (facultatives) de <code>list-style-type</code> , <code>list-style-image</code> et <code>list-style-position</code>
Héritage	Cette propriété est <i>héritée</i> , comme chacune des propriétés dont elle est un raccourci.

Les tableaux

Les propriétés de style qui suivent permettent de préciser la mise en forme des tableaux. Rappelons à cette occasion qu'en HTML moderne, les tableaux ne sont pas utilisés pour la mise en page. Il est préférable d'employer d'autres techniques, par exemple lorsqu'il s'agit de placer des contenus côté à côté : nous aurons l'occasion d'en parler plus en détail dans le chapitre qui suit, à propos du positionnement des blocs de texte dans la page.

Largeur fixe ou variable des colonnes ou du tableau

Par défaut, les largeurs de colonne d'un tableau sont automatiques : elles s'adaptent à leur contenu. Pour obtenir des colonnes de largeur fixe, il faut utiliser la propriété `table-layout`.

TABLEAU 5–43 Propriété `table-layout`

Propriété	<code>table-layout</code>
Exemple	<code>table { table-layout: fixed; }</code>
Valeurs possibles	<code>auto</code> : largeur automatique (valeur par défaut) <code>ou fixed</code> : largeur fixe
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utiliser <code>auto</code> .

À NOTER Dimension du tableau avec la valeur `fixed`

Avec la propriété `table-layout: fixed`, le tableau prend toute la largeur disponible (sur la page ou dans son bloc conteneur), sauf si une dimension est précisée pour ce tableau avec la propriété `width`.

Recouvrement des bordures

La propriété `border-collapse` sert à indiquer s'il y aura fusion ou non des bordures qui se touchent dans le tableau (voir la figure 5–15). Cela concerne :

- les bordures contiguës de deux cellules voisines ;
- la bordure du tableau et celle d'une cellule qui se trouve en bord de tableau.

TABLEAU 5–44 Propriété border-collapse

Propriété	<code>border-collapse</code>
Exemple	<code>table, td { border: solid 1px red; border-collapse: collapse; }</code>
Valeurs possibles	<code>collapse</code> : fusion des bordures <code>separate</code> : séparation des bordures (voir la figure 5–15)
Héritage	Propriété héritée

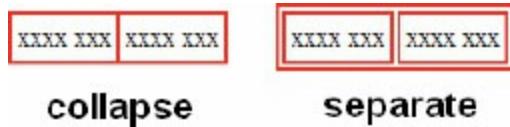


FIGURE 5–15 Fusion ou séparation des bordures avec border-collapse

Espacement entre les bordures de cellules

La taille de l'espace qui se trouve entre les bordures de deux cellules adjacentes est réglable à l'aide de la propriété `border-spacing`. Bien sûr, cela ne vaut que pour les cellules dont les bordures sont distinctes, c'est-à-dire pour lesquelles la propriété `border-collapse` a pour valeur `separate`.

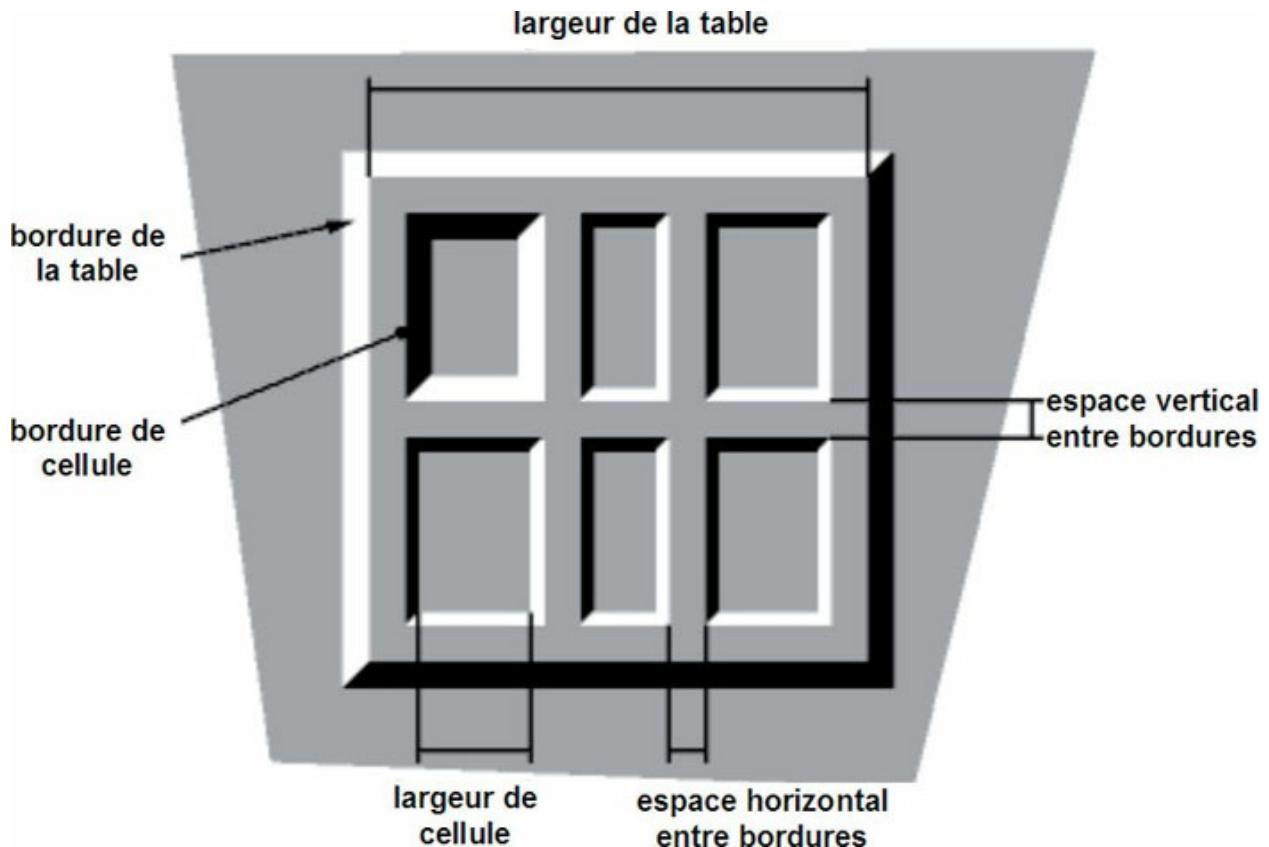


FIGURE 5–16 Schéma d'un tableau d'après la traduction des normes CSS 2 du W3C : <http://www.yoyodesign.org/doc/w3c/css2/tables.html#x22>

TABLEAU 5–45 Propriété border-spacing

Propriété	<code>border-spacing</code>
Exemples	<code>table { border-collapse: separate; border-spacing: 5px; }</code> <code>table { border-collapse: separate; border-spacing: 2px 5px; }</code>
Valeurs possibles	Un ou deux nombres positifs , dans les mêmes unités que les tailles de police (px, rem, em,...) sauf % Une valeur : espace pour toutes les bordures Deux valeurs : espace <i>horizontal</i> et espace <i>vertical</i>

Héritage

Propriété *héritée*

Contour des cellules vides

Afficher ou masquer le contour des cellules vides, voilà ce que va paramétriser la propriété `empty-cells`, uniquement dans le cas où les bordures sont distinctes (`border-collapse: separate;`).

TABLEAU 5–46 Propriété `empty-cells`

Propriété	<code>empty-cells</code>
Exemple	<code>table { border-collapse: separate; empty-cells: show; }</code>
Valeurs possibles	<code>show</code> : afficher le contour des cellules vides (valeur par défaut) <code>hide</code> : masquer cette bordure
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utiliser <code>show</code> .

Position du titre du tableau

Placée entre balises `<table>...</table>`, la balise `<caption>` donne un titre au tableau, qui se trouve initialement au-dessus de celui-ci. Il est cependant possible de le placer en dessous, en donnant la valeur adéquate à la propriété `caption-side`.

TABLEAU 5–47 Propriété `caption-side`

Propriété	<code>caption-side</code>
Exemple	<code>caption { caption-side: bottom; }</code>
Valeurs possibles	<code>top</code> : titre au-dessus du tableau (valeur par défaut) <code>bottom</code> : titre sous le tableau
Héritage	<i>Non</i>

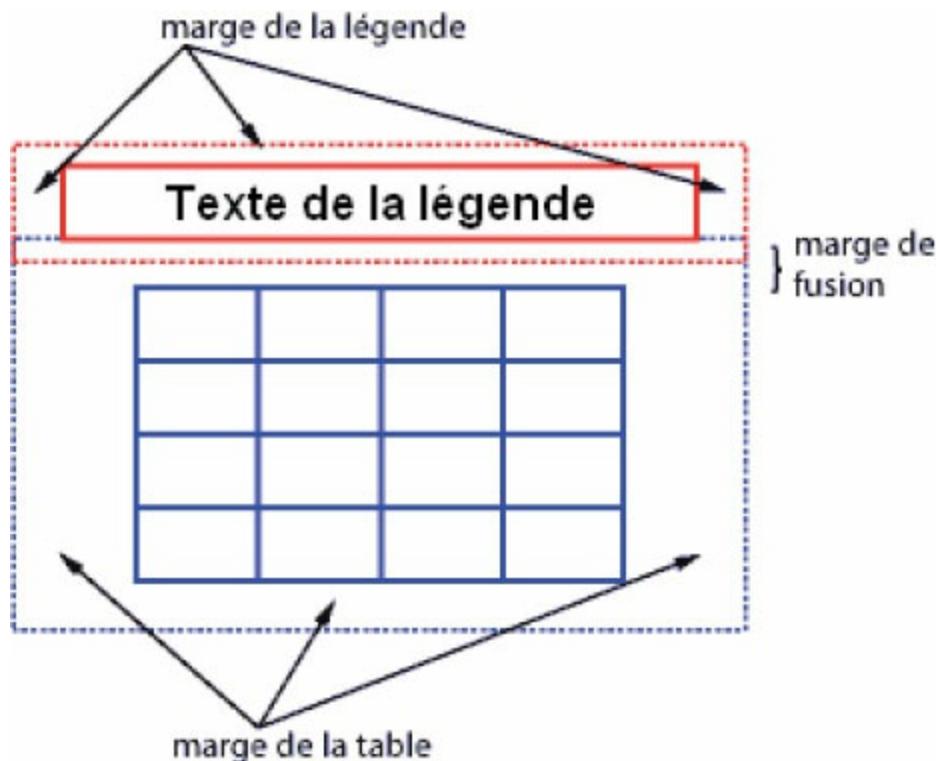


FIGURE 5–17 Un tableau et sa légende, d'après les normes du W3C traduites : <http://www.yoyodesign.org/doc/w3c/css2/tables.html#q5>

Alignement vertical des cellules

La propriété `vertical-align` permet d'indiquer quel doit être l'alignement d'une cellule par rapport à la rangée de cellules dont elle fait partie, lorsque la taille de cette rangée est supérieure à celle de la cellule.

TABLEAU 5–48 Propriété `vertical-align`

Propriété	<code>vertical-align</code>
Exemples	<pre>.commentaire { vertical-align: top; } .titre { vertical-align: middle; }</pre>
Valeurs possibles	<p><code>baseline</code> : alignement normal sur la première ligne (valeur par défaut)</p> <p><code>top</code> : alignement sur le haut de la rangée de cellules, <code>middle</code> : alignement au milieu de la rangée</p> <p><code>bottom</code> : alignement sur le bas de la rangée</p> <p><code>text-top</code> ou <code>text-bottom</code> : alignement sur le haut ou le bas des caractères du conteneur</p> <p><code>super</code> ou <code>sub</code> : alignement sur les exposants ou sur les indices, <i>dimension</i> (en px, rem...) ou <i>pourcentage</i> (100 % = valeur de <code>line-height</code>) positif ou négatif pour un décalage vers le haut ou vers le bas</p>
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utiliser <code>baseline</code> .

La ligne de base d'une rangée de cellules (sur laquelle se « poseront » les lettres de la première ligne) est la même pour toutes les cellules d'une même ligne. Dans le tableau de la figure 5-18, la première ligne de la boîte de la cellule 2 est la plus haute des premières lignes de cellules : c'est donc elle qui déterminera la « ligne de base » de la rangée.

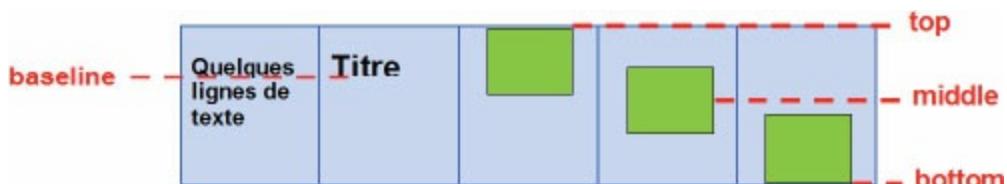


FIGURE 5–18 Alignement vertical de cellules dans une rangée de tableau

Nous voilà maintenant en possession des propriétés de base, qui permettent –

tant en CSS 2 qu'en CSS 3 – la mise en forme de nos pages web. Cependant, nous ne les avons pas encore toutes détaillées : il nous reste à étudier les propriétés liées à la position des blocs, ainsi qu'à leurs dimensions et à leurs marges. Ce sujet étant suffisamment vaste, il fera l'objet du chapitre suivant...



FIGURE 5–19 Extrait du site <https://framazic.org> sur la musique libre, lié au portail du libre <https://framasoft.org> : un choix judicieux de polices, couleurs et images de fond pour une page à la fois sobre et très gaie

chapitre 6

Positionnement des blocs

The screenshot shows the homepage of the CSS Zen Garden. At the top, there's a banner with a scallop shell background. The main title "CSS Zen Garden" is written in a large, brown, stylized font. Below it, the subtitle "The Beauty of CSS Design" is in a smaller, cursive font. A text block on the left says: "Une démonstration de ce qu'on peut accomplir lorsqu'on utilise les CSS pour la conception web. Sélectionnez n'importe quelle feuille de style listée pour charger le résultat sur cette page." On the right, there's a link "Téléchargez les fichiers d'exemple [html](#) et [css](#)". On the left side, there's a sidebar titled "Select a design" with a list of design names and their creators:

- [Love Is In The Air](#) by Nele Goetz
- [Greece Remembrance](#) by Pierre-Léo Bourbonnais
- [Hengarden](#) by Mr. Khmerang
- [Hoops - Tournament Edition](#) by David Marshall Jr.
- [Obsequience](#) by Pierce Gleeson
- [Red Paper](#) by Rob Soule
- [Chien](#) by Alex Miller

In the center, there's a box titled "The Road to Enlightenment" containing text about the history of CSS and its evolution. Another box below it is titled "So What is This About?"

En CSS 2 comme en CSS 3, la mise en page consiste à placer les blocs de texte et les images comme des boîtes imbriquées, juxtaposées ou superposées dans la page.

SOMMAIRE

- ▶ **Marges et dimensions d'un bloc**
- ▶ **Position des éléments**
- ▶ **Délimitation des blocs**
- ▶ **Exemples de positionnement**
- ▶ **Centrage d'éléments à l'intérieur des blocs**

À l'époque lointaine du HTML 4, la mise en page utilisait des tableaux inclus dans d'autres tableaux, avec parfois plusieurs niveaux d'imbrication : imaginez la complexité du code ! Heureusement, les temps ont changé et les feuilles de styles nous proposent une méthode claire pour la mise en page : elle consiste à positionner les éléments dans la page ou dans leurs blocs conteneurs.

Toutefois, il faudra au préalable définir les dimensions et les marges, intérieures et extérieures, de chaque élément. Toutes les propriétés abordées dans ce chapitre font à la fois partie des normes CSS 2 et CSS 3.

Marges et dimensions d'un bloc

Il est important de comprendre comment sont calculées les dimensions des « boîtes », c'est-à-dire des blocs contenant texte et images.

Les CSS vont nous permettre de fixer les dimensions d'une boîte, ainsi que ses marges intérieures (à l'intérieur des bordures) et ses marges extérieures (à l'extérieur des bordures).

Marges externes autour d'un bloc

Les marges externes d'un bloc sont situées au-delà de ses bordures. Elles servent à espacer les blocs entre eux. Elles sont définies sur chacun des côtés à l'aide des propriétés `margin-top` en haut, `margin-right` à droite, `margin-bottom` en bas et `margin-left` à gauche, ou globalement par la propriété raccourcie `margin`.

TABLEAU 6–1 Propriétés définissant les marges extérieures

Propriétés	<code>margin-left</code> , <code>margin-right</code> , <code>margin-top</code> , <code>margin-bottom</code>
Exemples	<code>p { margin-left: 4rem; margin-right: 3rem; }</code> <code>p { margin-top: 5px; margin-bottom: 6px; }</code>
Valeurs possibles	<code>auto</code> , taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... ou taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la largeur du bloc conteneur, même pour <code>margin-top</code> et <code>margin-bottom</code>
Héritage	<i>Non</i>

À NOTER Utilisation des marges externes

- Pour les blocs juxtaposés ou imbriqués, les marges mitoyennes sont confondues : par exemple, la marge inférieure d'un bloc et la marge supérieure du bloc suivant sont fusionnées. Cependant, ce n'est pas le cas pour les blocs flottants ou positionnés.
- La valeur `auto` s'emploie avec `margin-left` et `margin-right` : elle signifie que ces deux marges doivent être égales, ce qui revient à centrer l'élément concerné dans son bloc conteneur.
- En utilisant des valeurs négatives pour les marges externes, il est possible de superposer des blocs.
- Les éléments en ligne n'ont pas de marge supérieure, ni inférieure ; pour leurs marges de gauche et de droite, la valeur `auto` correspond à 0.
- Certaines feuilles de styles commencent par la règle `* { margin: 0; }`, qui met à zéro les marges externes de tous les éléments de la page web. Cela nous affranchit des différences d'interprétation possibles

| de la part des navigateurs, pour leur valeur par défaut.

Raccourci pour les marges externes

La propriété `margin` simplifie la définition des marges externes, en remplaçant les quatre propriétés précédentes.

TABLEAU 6–2 Propriété raccourcie `margin`

Propriété	<code>margin</code>
Exemples	<pre>p { margin: 0; } p { margin: 3rem 5rem; } p { margin: 5% 10% 8%; } p { margin: 15px 10px 20px 15px; }</pre>
Valeurs possibles	<p>Une valeur : définit toutes les marges extérieures.</p> <p>Deux valeurs : ① marges du haut et du bas égales, ② marges de gauche et de droite égales</p> <p>Trois valeurs : ① marge du haut, ② marges de gauche et de droite égales, ③ marge du bas</p> <p>Quatre valeurs : ① marge du haut, ② marge de droite, ③ marge du bas, ④ marge de gauche</p>
Pourcentages	% de la largeur du bloc conteneur, même pour les marges du haut et du bas
Héritage	<i>Non</i>

ASTUCE Ordre des propriétés

Pour les marges extérieures ou intérieures, comme pour les bordures, il est facile de se rappeler l'ordre dans lequel elles sont définies, car il suffit de commencer par le haut et de tourner dans le sens des aiguilles d'une montre : haut - droite - bas - gauche.

Marges internes d'un bloc

Les marges intérieures d'un bloc se trouvent à l'intérieur de ses bordures. Leur présence évite que le texte ne soit collé au cadre du bloc qui le contient. Elles sont définies ainsi soit sur chaque côté à l'aide des propriétés `padding-top` en haut, `padding-right` à droite, `padding-bottom` en bas et `padding-left` à gauche, soit globalement avec la propriété raccourcie `padding`.

TABLEAU 6–3 Propriétés définissant les marges intérieures

Propriétés	<code>padding-left</code> , <code>padding-right</code> , <code>padding-top</code> , <code>padding-bottom</code>
Exemples	<code>p { padding-left: 20px; padding-right: 15px; }</code> <code>p { padding-top: 5%; padding-bottom: 10%; }</code>
Valeurs possibles	Valeur de taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... ou de taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la largeur du bloc conteneur, même pour <code>padding-top</code> et <code>padding-bottom</code>
Héritage	<i>Non</i>

À NOTER Utilisation des marges internes

- La valeur par défaut des marges internes est 0.
- Les marges internes ne peuvent pas être négatives.
- La valeur `auto` n'existe pas pour les marges intérieures.
- Les éléments en ligne n'ont ni marge supérieure, ni marge inférieure.

Raccourci pour les marges internes

La propriété `padding` simplifie la définition des marges internes, en remplaçant les quatre propriétés précédentes.

TABLEAU 6–4 Propriété raccourcie `padding`

Propriété	<code>padding</code>
Exemples	<code>p { padding: 5rem; }</code> <code>p { padding: 10px 0; }</code> <code>p { padding: 2rem 1rem 3rem; }</code> <code>p { padding: 5% 8% 6% 10%; }</code>
Valeurs possibles	Une valeur : définit toutes les marges intérieures Deux valeurs : ① marges du haut et du bas égales, ② marges de gauche et de droite égales Trois valeurs : ① marge du haut, ② marges de gauche et de droite égales, ③ marge du bas Quatre valeurs : ① marge du haut, ② marge de droite, ③ marge du bas, ④ marge de gauche
Pourcentages	% de la largeur du bloc conteneur, même pour les marges du haut et du bas
Héritage	<i>Non</i>

Largeur fixe pour un bloc ou une image

Il est possible de choisir une largeur fixe pour le contenu d'un bloc de texte ou pour une image, en utilisant la propriété `width`.

ATTENTION Utilisation de `width`

- Les valeurs de `width` ne peuvent pas être négatives.
- Cette propriété `width` correspond uniquement à la largeur disponible pour le contenu. Elle ne comprend pas les marges internes et externes, ni l'épaisseur de la bordure éventuelle. Si nous voulons changer les marges tout en conservant une même taille globale pour le bloc, il nous faudra modifier la propriété `width` pour compenser le changement des marges.

TABLEAU 6–5 Propriété `width`

Propriété	<code>width</code>
Exemples	<code>div { width: 300px; }</code> <code>.menu { width: 20%; }</code>
Valeurs possibles	<code>auto</code> (par défaut), taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... ou taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la largeur du bloc conteneur
Héritage	<i>Non</i>

Hauteur fixe pour un bloc ou une image

La propriété `height` permet de définir une hauteur fixe pour le contenu d'un bloc de texte ou pour une image.

ATTENTION Utilisation de `height`

- Les valeurs de `height` ne peuvent pas être négatives.
- Dans le même esprit que la propriété `width`, la valeur attribuée à `height` correspond à la hauteur disponible pour le contenu, n'incluant pas les marges (externes comme internes) ni l'épaisseur de bordure si elle existe. Un changement de marges à hauteur globale constante implique donc une modification de la propriété `height`.

TABLEAU 6–6 Propriété `height`

Propriété	<code>height</code>
Exemples	<code>div { height: 50%; }</code> <code>img#logo { height: 10rem; }</code>
Valeurs possibles	<code>auto</code> (par défaut), taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... ou taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la hauteur du bloc conteneur si celle-ci est fixée, sinon c'est la valeur <code>auto</code> qui est appliquée.
Héritage	<code>Non</code>

Largeur et hauteur totales d'un bloc

Le modèle de boîte de la figure 6–1 montre comment calculer la largeur et la hauteur totales d'un bloc. Nous verrons au chapitre suivant la propriété CSS 3 `box-sizing` qui permet d'inclure les marges internes et éventuellement les bordures dans ces dimensions. Mais lorsque cette propriété n'est pas utilisée, le calcul s'effectue de cette façon.

Pour obtenir la **largeur totale** d'un bloc, il faut additionner :

- les marges extérieures de gauche et de droite :
`margin-left + margin-right ;`
- deux fois l'épaisseur de la bordure : $2 \times \text{border-width} ;$
- les marges intérieures de gauche et de droite :
`padding-left + padding-right ;`
- la largeur du contenu : `width`.

Pour obtenir la **hauteur totale** d'un bloc, il faut additionner :

- les marges extérieures du haut et du bas :
`margin-top + margin-bottom ;`
- deux fois l'épaisseur de la bordure : $2 \times \text{border-width} ;$
- les marges intérieures du haut et du bas :
`padding-top + padding-bottom ;`
- la hauteur du contenu : `height`.

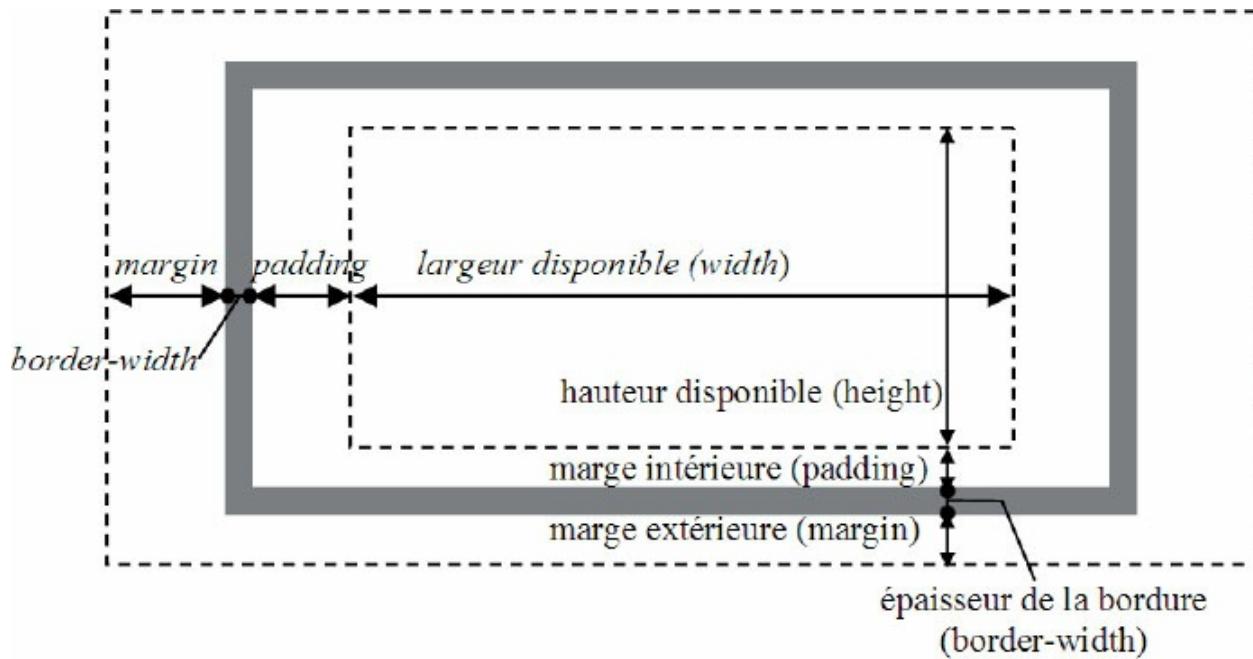


FIGURE 6–1 Dimensions des boîtes

Lorsque les marges horizontales sont égales, le calcul de la largeur totale est plus simple :

$$\boxed{\text{Largeur totale} = 2 \times (\text{margin} + \text{border-width} + \text{padding}) + \text{width}}$$

Il en est de même pour le calcul de la hauteur totale, lorsque les marges verticales sont égales :

$$\boxed{\text{Hauteur totale} = 2 \times (\text{margin} + \text{border-width} + \text{padding}) + \text{height}}$$

À NOTER Prise en compte des bordures

Les calculs qui précèdent supposent que le bloc concerné est encadré par une bordure uniforme, avec la même épaisseur d'encadrement à gauche et à droite pour le calcul de la largeur, ou bien en haut et en bas pour celui de la hauteur. Dans le cas contraire :

- si le bloc ne comprend pas de bordure, il suffit de soustraire `border-width`, cela simplifie la formule ;
- si les bordures ne sont pas uniformes, il faut ajouter séparément l'épaisseur de chacune des bordures concernées.

Largeur et hauteur minimales

Les propriétés `min-width` et `min-height` définissent respectivement la largeur et la hauteur minimum d'un bloc.

TABLEAU 6–7 Propriétés `min-width` et `min-height`

Propriétés	<code>min-width</code> , <code>min-height</code>
Exemples	<code>h1 { min-width: 50%; }</code> <code>div.remarque { min-height: 50rem; }</code>
Valeurs possibles	<code>none</code> = 0 (valeur par défaut) ou dimension (nombre positif) : taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la largeur du bloc conteneur pour <code>min-width</code> % de la hauteur du bloc conteneur pour <code>min-height</code>
Héritage	<i>Non</i>

Largeur et hauteur maximales

Les propriétés `max-width` et `max-height` limitent respectivement la largeur et la hauteur d'un bloc.

TABLEAU 6–8 Propriétés `max-width` et `max-height`

Propriétés	<code>max-width</code> , <code>max-height</code>
Exemples	<code>p { max-width: 80%; }</code> <code>#extraits { max-height: 200px; }</code>
Valeurs possibles	<code>none</code> : pas de limite (valeur par défaut) ou dimension (positive) : taille relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... taille fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Pourcentages	% de la largeur du bloc conteneur pour <code>max-width</code> % de la hauteur du bloc conteneur pour <code>max-height</code>
Héritage	<i>Non</i>

Position des éléments

Chaque bloc peut être placé de différentes façons à l'intérieur de la page web : par rapport à d'autres blocs, ou bien à un endroit précis du bloc qui le contient, ou encore à un emplacement fixe sur la page.

Flux normal des éléments

À l'intérieur de chaque bloc, les éléments se placent au fur et à mesure, suivant le flux normal.

- les uns à la suite des autres pour les éléments en ligne :
`, , , , ...`
- les uns en dessous des autres pour les éléments de type bloc :
`<p>, <div>, <h1>, <h2>, ...`

Dans le flux normal, les dimensions d'un bloc sont les suivantes :

- **Largeur par défaut** = largeur disponible dans le bloc conteneur ;
- **Hauteur par défaut** = celle du contenu, 0 si le bloc ne contient rien.

Les blocs qui se succèdent dans le flux normal sont séparés les uns des autres par leurs marges extérieures, ces marges étant fusionnées entre deux blocs consécutifs.

Principe du positionnement des blocs

Seuls peuvent être positionnés les blocs de texte et les éléments « remplacés », comme les images. Pour positionner des éléments en ligne, il faut les transformer en blocs à l'aide de la propriété `display: block;`.

Par positionnement, les blocs peuvent être :

- juxtaposés ;
- fixés par rapport à la position de leur bloc conteneur ;
- ou encore fixés par rapport à la page.

Le positionnement nous permet de superposer les blocs, comme des *calques*. D'ailleurs, ce nom de « calque » est parfois donné à la balise `<div>`, qui est souvent utilisée comme bloc conteneur.

Si les blocs qui sont au premier plan ont un fond transparent (valeur par défaut de la couleur de fond), les blocs situés en dessous restent visibles.

Il est possible de choisir la position des blocs et de modifier leur ordre vertical de superposition, à l'aide de la propriété `z-index` que nous étudierons plus loin.

Types de position possibles

Un bloc peut être positionné de façon **normale**, **relative**, **absolue**, **fixe** ou **flottante**. Nous allons examiner à quoi correspondent ces différents types de positionnement.

Position normale

Lorsque sa position n'est pas précisée, un bloc se place dans le *flux normal* de la page web.

Position relative, absolue ou fixe

Il est possible de placer un élément en indiquant un *décalage* (en haut, en bas, à gauche, à droite) :

- par rapport à sa position dans le flux normal : c'est la « position relative » (propriété `position: relative;`) ;
- par rapport au bloc conteneur : c'est la « position absolue » (propriété `position: absolute;`) ;
- par rapport à l'écran : c'est la « position fixe » (propriété `position: fixed;`).

Dans chacun de ces trois cas, il faut indiquer un ou deux décalage(s) :

- un premier décalage à partir du haut (par exemple : `top: 2px;`) ou du bas (par exemple : `bottom: 10%;`) ;
- un deuxième à partir de la gauche (par exemple : `left: 5rem;`) ou de la droite (par exemple : `right: 50px;`).

Position flottante

Un élément peut enfin être déclaré « flottant » à gauche ou à droite, avec la propriété `float` qui s'écrit alors respectivement `float: left;` ou `float: right;`.

Le bloc est placé le plus à gauche ou le plus à droite possible, tout en gardant sa position verticale dans la boîte de son conteneur.

Le contenu qui suit encadre alors cette boîte flottante, comme le montre la

figure 6–2. S'il y a plusieurs éléments flottants, ils s'alignent côté à côté, avec retour à la ligne automatique lorsque le bord du bloc conteneur est atteint.

Tux à toute heure. As-tu ton Tux ? Tout est dans Tux. Tux est tentant. Tout est dit.		Consectetuer neque dui habitant Nulla justo. Cursus massa fermentum porttitor euismod pretium justo in iaculis est condimentum. Quam fringilla mollis wisi congue mauris laoreet Sed nulla id Praesent. Proin laoreet vel auctor ante nibh congue tellus ut id Pellentesque. Sit est pede a Vestibulum nec ac commodo consequat a at. Platea tempor lacinia at ut orci. Condimentum egestas velit morbi Nam quis at vel a volutpat sagittis. Id at consequat Nunc porttitor tincidunt Morbi risus rutrum elit semper. Nec et natoque tellus Curabitur Quisque id est ac augue Quisque. Ut sed pretium nec mattis ipsum facilisi et laoreet orci nunc. Odio massa venenatis habitant euismod felis hendrerit id laoreet.
--	---	--

FIGURE 6–2 Le bloc de texte sur fond gris et l'image sont flottants à gauche. Le texte qui suit habille ces deux blocs.

IMPORTANT Hauteur d'un bloc qui contient des éléments flottants

La dimension d'un bloc conteneur ne prend pas en compte celle des éléments flottants qu'il contient : les blocs flottants débordent de leur conteneur.

Le bloc qui suivra risque donc de se superposer aux éléments flottants ou de se trouver à côté d'eux, comme le montre la figure 6–3. Pour éviter cela, il suffit d'attribuer au deuxième bloc la propriété `clear: both;` qui interdit la présence d'éléments flottants sur le côté (nous étudierons cette propriété plus loin).

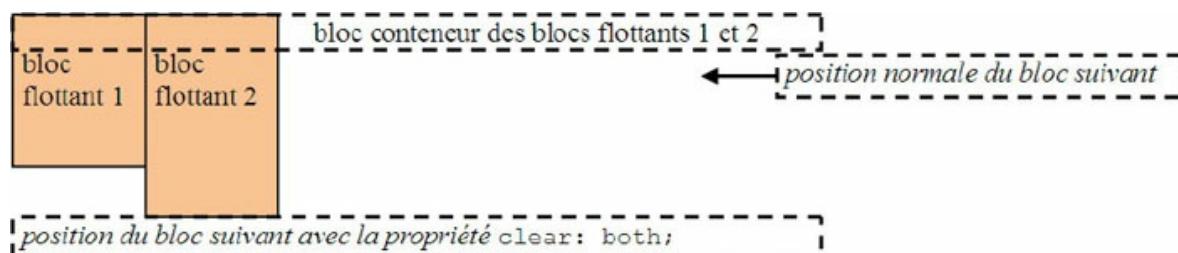


FIGURE 6–3 Les blocs flottants peuvent déborder de leur conteneur.

Utilisation des différents types de positionnement

Voici en résumé la façon d'utiliser les différents positionnements disponibles en CSS.

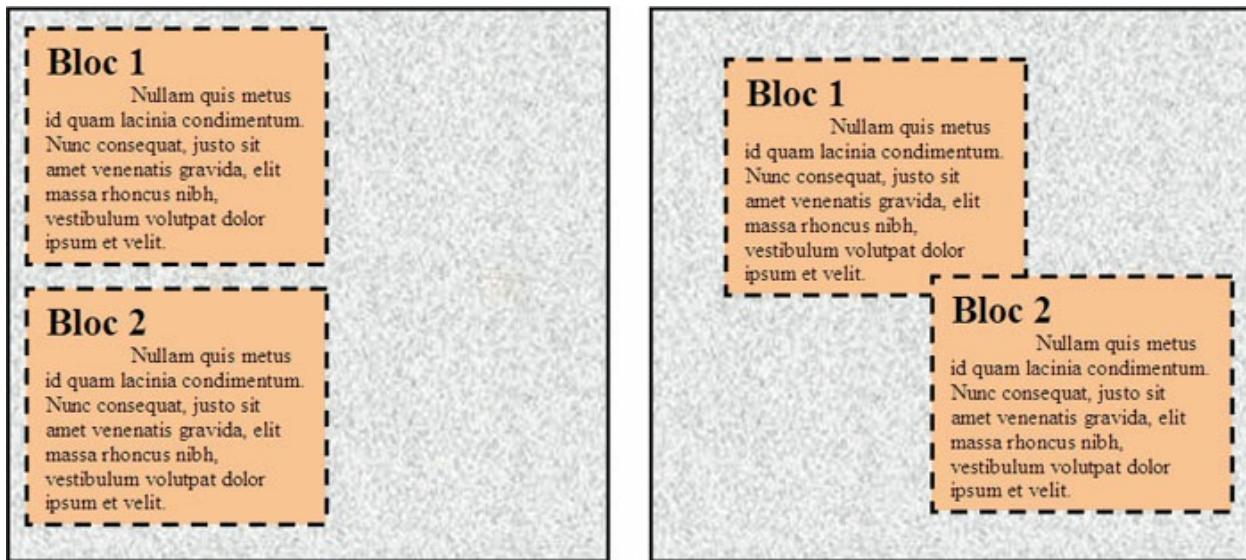


FIGURE 6-4 À gauche, les blocs 1 et 2 sont dans le flux normal de la page ; à droite, ces deux blocs sont placés en positionnement absolu à l'intérieur de la page.

Élément dans le flux (position normale)

- Soit aucune propriété de positionnement, soit `position: static;`
- Positionnement à utiliser aussi souvent que possible
- Blocs affichés les uns sous les autres, éléments en ligne placés côte à côte

Position relative

- Propriété `position: relative;`
- Pour décaler ou superposer un élément par rapport à ses « frères »

Position absolue

- Propriété `position: absolute;`
- Pour découper la page en zones, sans utiliser de tableaux

- Pour disposer à un endroit précis un menu, un encadré, une image...

Position fixe

- Propriété `position: fixed;`
- Pour conserver en permanence un élément à l'écran (qui ne bouge pas lors du défilement)
- Pour éviter l'utilisation de cadres (balise `<iframe>`)

ATTENTION Les blocs positionnés en absolu, en fixe ou en flottant sortent du flux

Mis à part les éléments positionnés en relatif, pour lesquels il n'y a qu'un décalage par rapport à leur position initiale, tous les éléments positionnés d'une autre manière – donc en absolu, en fixe ou en flottant – « sortent du flux ».

Cela entraîne ce que nous avons noté précédemment dans le cas des blocs flottants : si un bloc ne contient que des éléments positionnés, sa hauteur est nulle ; la suite de la page remonte donc comme si le bloc précédent n'existe pas.

La solution consiste à préciser une hauteur pour le bloc conteneur, ce qui nécessite de connaître la hauteur du ou des bloc(s) qu'il contient et qui sera(seront) positionné(s).

Lors de la conception de nos pages, gardons ce point à l'esprit pour éviter de voir, après le positionnement de nos blocs, des éléments se superposer ou d'autres disparaître !

Élément flottant

- Propriété `float: left;` ou `float: right;`
- Pour placer des éléments côté à côté, en fonction de la place disponible dans la fenêtre d'affichage ou le bloc conteneur
- Pour habiller une image de texte, pour une galerie d'images ou une suite de menus...

IMPORTANT Remarques pour la position absolue

Un bloc positionné se place par rapport au bloc qui le contient (son conteneur), mais **seulement si celui-ci est lui-même positionné**, que ce soit en position relative ou en position absolue.

Si ce n'est pas le cas (conteneur dans le flux ou flottant), le bloc à positionner remonte de parent en parent jusqu'au premier bloc positionné (jusqu'à `<body>` s'il n'y en a pas) et se place par rapport à lui.

Pour placer en position absolue un bloc dans un autre, il est donc important de vérifier si le conteneur est lui-même positionné. Si ce n'est pas le cas, il faut donner à ce bloc conteneur une position relative avec un décalage nul, ce qui ne modifiera pas sa position.

Type de positionnement d'un bloc

Au fil des exemples précédents, nous avons rencontré plusieurs fois la propriété `position` qui, comme son nom l'indique, définit un type de position pour l'élément concerné.

TABLEAU 6–9 Propriété position

Propriété	<code>position</code>
Exemples	<pre>p.note {position: relative; left: -5px;} #menu {position: absolute; top: 0; right: 10%;}</pre>
Valeurs possibles	<code>static</code> : positionnement dans le flux normal (valeur par défaut) <code>relative</code> : décalage par rapport à la position dans le flux <code>absolute</code> : positionnement par rapport au bloc conteneur <code>fixed</code> : positionnement par rapport à l'écran
Héritage	<i>Non</i>

Décalages indiquant la position d'un bloc

Le positionnement utilise les décalages `top` (haut), `bottom` (bas), `left` (gauche) et `right` (droite).

TABLEAU 6–10 Propriétés `top`, `bottom`, `left` et `right`

Propriétés	<code>top</code> , <code>bottom</code> , <code>left</code> , <code>right</code>
Exemples	<pre>p.note { position: relative; top: 5px; left: 10px; } div.menu { position: absolute; top: 30%; right: 20%; } #remarque {position: relative; top: 2rem;}</pre>
Valeurs possibles	<code>none</code> : pas de décalage (valeur par défaut) dimension relative (conseillée) en <code>rem</code> , <code>em</code> , <code>%</code> , <code>px</code> , ... dimension fixe en <code>pt</code> , <code>pc</code> , <code>cm</code> , <code>mm</code> , <code>in</code>
Héritage	<i>Non</i>

À NOTER Décalages `top`, `bottom`, `left` et `right`

- Les valeurs négatives sont possibles pour `top`, `bottom`, `left` et `right`.
- Si `top` et `bottom` sont spécifiés simultanément, seul `top` est pris en compte.
- Si `left` et `right` sont spécifiés simultanément, seul `left` est pris en compte.

Niveau d'empilement des blocs

Lorsque plusieurs blocs sont superposés, ils s'empilent suivant l'ordre de leur arrivée dans le code HTML. Cet arrangement peut toutefois être modifié, grâce à la propriété `z-index`.

TABLEAU 6–11 Propriété `z-index`

Propriété	<code>z-index</code>
Exemples	<pre>ul.menu { position: relative; z-index: 10; } #logo {position: absolute; top: 0; z-index: -5;}</pre>
Valeurs possibles	<code>auto</code> : même niveau d'empilement que la boîte parent (valeur par défaut) ou nombre entier positif, nul ou négatif
Héritage	<i>Non</i>

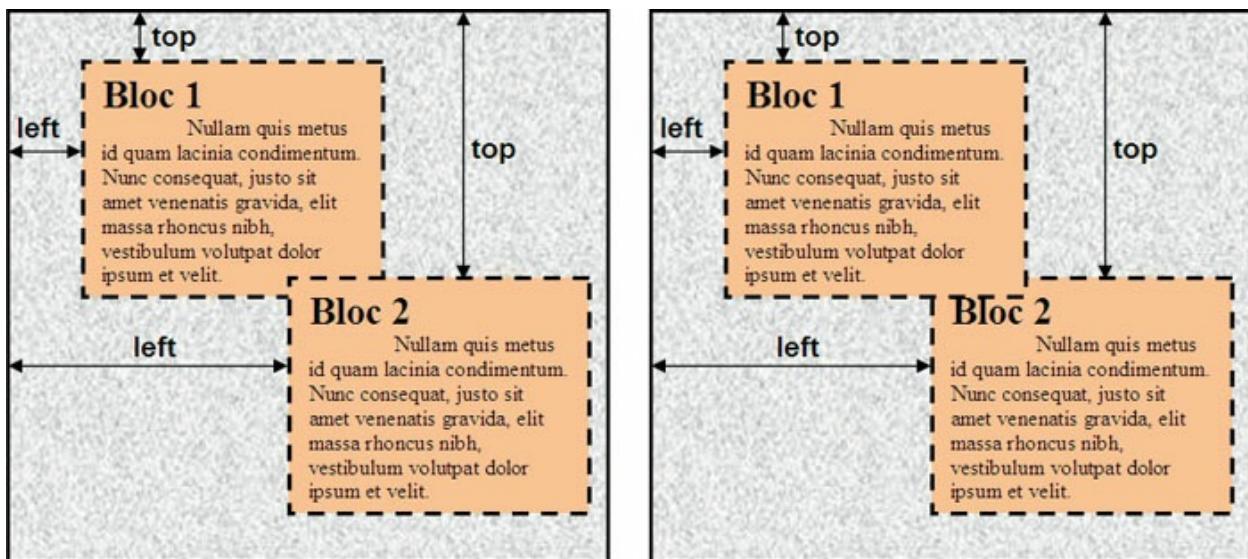


FIGURE 6–5 Les blocs 1 et 2 sont positionnés ici par rapport aux limites haute et gauche de leur bloc conteneur ; à droite, le bloc 1 a un niveau d'empilement `z-index` plus élevé que le bloc 2.

IMPORTANT Utilisation de la propriété `zindex`

- La propriété `z-index` n'est prise en compte que lorsqu'elle s'applique à un bloc qui est positionné. Si ce n'est pas le cas, le bloc concerné peut être positionné tout en conservant son emplacement : il suffit de lui donner une position relative, sans décalage.

- Plus la valeur de `z-index` est élevée, plus le bloc se trouve en haut dans la superposition des blocs.
- La transparence du fond (valeur par défaut de `background-color`) permet de voir le contenu des boîtes situées plus bas dans la superposition.

Transformation en bloc flottant

La propriété `position` ne permet pas d'effectuer le positionnement flottant d'un bloc ; c'est la propriété spécifique `float` qui est utilisée pour cela.

TABLEAU 6–12 Propriété float

Propriété	<code>float</code>
Exemples	<code>img.vignettes { float: left; }</code> <code>div.infos { float: right; }</code>
Valeurs possibles	<code>none</code> : pas de flottement (valeur par défaut) <code>left</code> : élément flottant calé à gauche <code>right</code> : élément flottant calé à droite
Héritage	<i>Non</i>

Pas d'éléments flottants sur le côté

Lorsqu'un élément sera flottant, le contenu qui le suit dans le code HTML se trouvera à côté de celui-ci, si la place disponible le permet : c'est ainsi qu'une image flottante peut être « habillée » par du texte.

Pour obliger le navigateur à afficher un contenu à la ligne, sous le plus bas des éléments flottants, il faut le lui préciser en attribuant à cet élément la propriété `clear`.

TABLEAU 6–13 Propriété clear

Propriété	<code>clear</code>
Exemples	<code>h1 { clear: both; }</code> <code>.remarque { clear: left; }</code>
Valeurs possibles	<code>none</code> : éléments flottants autorisés à gauche et à droite (valeur par défaut) <code>left</code> : pas d'élément flottant sur la gauche <code>right</code> : pas d'élément flottant sur la droite <code>both</code> : aucun élément flottant, ni à gauche, ni à droite
Héritage	<i>Non</i>

Affichage ou non d'un élément

La propriété `visibility` permet de masquer occasionnellement un élément, tout en réservant dans la page la place qu'il occuperait normalement – sans perturber la mise en page, par conséquent.

TABLEAU 6–14 Propriété `visibility`

Propriété	<code>visibility</code>
Exemple	<code>.note { visibility: hidden; }</code>
Valeurs possibles	<code>visible</code> : l'élément est visible (valeur par défaut) <code>hidden</code> : l'élément est masqué, mais il occupe toujours le même espace dans la page <code>collapse = hidden</code> , sauf dans les tableaux, où l'espace est libéré lorsqu'il s'agit d'une ligne entière ou d'une colonne entière
Héritage	<i>Non</i>

Affichage des débordements

Lorsqu'un contenu déborde de son bloc conteneur (si les dimensions de ce bloc sont fixées), il peut être visible, masqué ou accessible grâce à une barre de défilement, suivant la valeur attribuée à la propriété `overflow`. Les propriétés `overflow-x` et `overflow-y` ont la même utilité, mais concernent respectivement les dimensions horizontale et verticale du bloc.

TABLEAU 6–15 Propriétés `overflow`, `overflow-x` et `overflow-y`

Propriété	<code>overflow</code> , <code>overflow-x</code> , <code>overflow-y</code>
Exemples	<pre>p { overflow: scroll; } #cadre1 { overflow: hidden; } div.texte { overflow-y: auto ; }</pre>
Valeurs possibles	<p><code>visible</code> : le débordement est visible (valeur par défaut)</p> <p><code>hidden</code> : le débordement est masqué</p> <p><code>scroll</code> : affichage dans tous les cas d'une barre de défilement ; elle permettra l'accès à un débordement éventuel.</p> <p><code>auto</code> : une barre de défilement apparaît, mais seulement en cas de débordement.</p>
Héritage	<i>Non</i>

Zone visible d'un élément

Il existait en CSS 2 une propriété `clip` qui permettait de restreindre à un rectangle donné la zone visible d'un élément. Elle a été abandonnée au profit d'une nouvelle propriété CSS 3 `clip-path` qui nous donne accès à des formes plus variées ; elle n'est pas encore reconnue par tous les navigateurs cependant.

Appliquée à une image ou à un bloc, la propriété `clip-path` permet d'y découper une zone géométrique qui sera la zone visible. Cette forme sera déterminée soit par un nom (*inset* pour un rectangle, *circle*, *ellipse*, *polygon*) associé à des coordonnées, soit par une URL renvoyant vers une image SVG placée dans le code HTML et définissant le découpage.

TABLEAU 6–16 Propriété `clip-path`

Propriété	<code>clip-path</code>
Exemples	<pre>.img1 { clip-path: inset(20% 50% 20% 10%); } .img2 { clip-path: circle(35% at 50% 50%); } .img3 { clip-path: ellipse(80px 60px at 100px 60px); } .img4 { clip-path: polygon(5% 5%, 100% 5%, 95% 80%, 0% 80%); } .img5 { clip-path: url(#masque); }</pre>
Valeurs possibles	<p>none : pas de rognage (valeur par défaut) une forme géométrique :</p> <p><code>inset(r1 r2 r3 r4)</code>, rectangle défini par les rognages en haut, à droite, en bas et à gauche (deux valeurs si haut = bas et gauche = droite, une valeur pour un rognage homogène) ;</p> <p><code>circle(r at x y)</code>, cercle défini par son rayon r et son centre (x,y) ;</p> <p><code>ellipse(r1 r2 at x y)</code>, ellipse avec ses deux rayons (r1,r2) et son centre (x,y)</p> <p><code>polygon(x1 y1, x2 y2, ..., xn yn)</code>, forme donnée par un ensemble de points.</p> <p>ou bien un lien vers une image de découpage en <code>svg</code> : <code>url(#masque1)</code></p> <p>avec par exemple dans le code HTML :</p> <pre><svg width="200" height="100"> <defs></pre>

```
<clipPath id="masque1">
  <rect x="50" y="20" width="100" height="60">
</clipPath>
</defs>
</svg>
```

Héritage

Non

Changement de type d'élément

Chaque élément de la page fait partie d'une catégorie, comme les éléments en ligne, les blocs, etc. Cependant, il est parfois nécessaire de changer le type d'un élément.

Par exemple, pour appliquer à un élément en ligne tel qu'un lien une propriété liée aux blocs, il faut d'abord le transformer en bloc. C'est ce que permet la propriété `display`.

TABLEAU 6–17 Propriété display

Propriété	<code>display</code>
Exemples	<pre>p.secret { display: none; } .cellule { display: table-cell; } span.bloc { display: block; }</pre>
Valeurs possibles	<code>inline</code> : élément en ligne (valeur par défaut) <code>block</code> : bloc <code>list-item</code> : élément de liste <code>inline-block</code> : élément en ligne remplacé (avec dimensions et marges verticales) <code>run-in</code> : bloc ou élément en ligne, suivant le contexte <code>table</code> : tableau, <code>inline-table</code> : tableau en ligne <code>table-cell</code> : cellule de tableau, <code>table-row</code> : ligne de tableau <code>table-column</code> : colonne de tableau <code>table-caption</code> : titre de tableau <code>table-row-group</code> : groupe de lignes de tableau <code>table-column-group</code> : groupe de colonnes de tableau <code>table-header-group</code> : groupe d'en-têtes de tableau <code>table-footer-group</code> : groupe de pieds de tableau <code>flex</code> (ou <code>inline-flex</code>) : type <code>block</code> (ou <code>inline-block</code>) dont les blocs qu'il contient pourront être automatiquement alignés, ajustés ou répartis, à l'aide de propriétés spécifiques qui seront détaillées dans le chapitre suivant <code>none</code> : l'élément est invisible et ses dimensions sont nulles.
Héritage	<code>Non</code>

À NOTER Différence entre `visibility: hidden` ; et `display: none` ;

Ces deux règles permettent de masquer un objet dans la page.

- Avec `visibility: hidden;`, l'objet est masqué, mais occupe toujours la même place qu'auparavant dans la page.
- Avec `display: none;`, l'objet devient invisible également, mais sa place n'est plus réservée sur la page (aucune boîte n'est générée).

Délimitation des blocs

Même en voyant la page web s'afficher à l'écran, il n'est pas toujours facile d'appréhender la position et les dimensions des blocs qui la composent.

Il existe une méthode toute simple – mais qui rend d'immenses services – pour voir ces blocs à l'écran : il suffit de les délimiter au moyen d'une bordure, en leur appliquant temporairement la propriété suivante :

```
border: 1px solid red;
```

Une fois ces blocs encadrés, comme le montre la figure 6–6, il est beaucoup plus facile de voir comment ils sont organisés entre eux, comment sont prises en compte les dimensions et les marges, quelles sont les différences d'interprétation entre deux navigateurs, etc.

Toutefois, il faut savoir que cette opération augmente de deux pixels les dimensions horizontale et verticale de chaque bloc. Par conséquent, elle est pratique pour avoir une idée de l'emplacement et de la taille de ces éléments, mais elle ne peut pas être utilisée pour un positionnement précis au pixel près. Elle peut aussi modifier la mise en page, le retour à la ligne d'un bloc flottant dû à l'ajout de ces pixels, par exemple.

Une variante de cette méthode consiste à utiliser différentes couleurs d'encadrement, de façon à mieux repérer les blocs.

A l'ombre du noisetier...

Cot ! Cot ! Cot!

Coin ! Coin ! Coin !

Ouah ! Ouah !

une puce pique le chien

Dans une prairie verte...

Meuh ! Meuh !

Groin ! Groin !

FIGURE 6-6 Vous souvenez-vous du passage sur l'héritage, au début de cet ouvrage ? Une fois les blocs encadrés, leur imbrication et leurs marges sont bien visibles.

ASTUCE Informations sur les éléments d'une page web

L'extension *Web Developer* (figure 6-7), disponible pour Firefox et Chrome, est une aide précieuse pour le concepteur web. Une de ses nombreuses fonctions permet d'entourer un élément de la page, par un simple clic sur celui-ci, en affichant des informations sur le code HTML et CSS associé à cet élément : menu *Infos - Afficher les informations de l'élément*. Cette extension (multilingue) et sa documentation (en anglais) sont disponibles à l'adresse suivante :

- ▶ <http://chrispederick.com/work/web-developer>

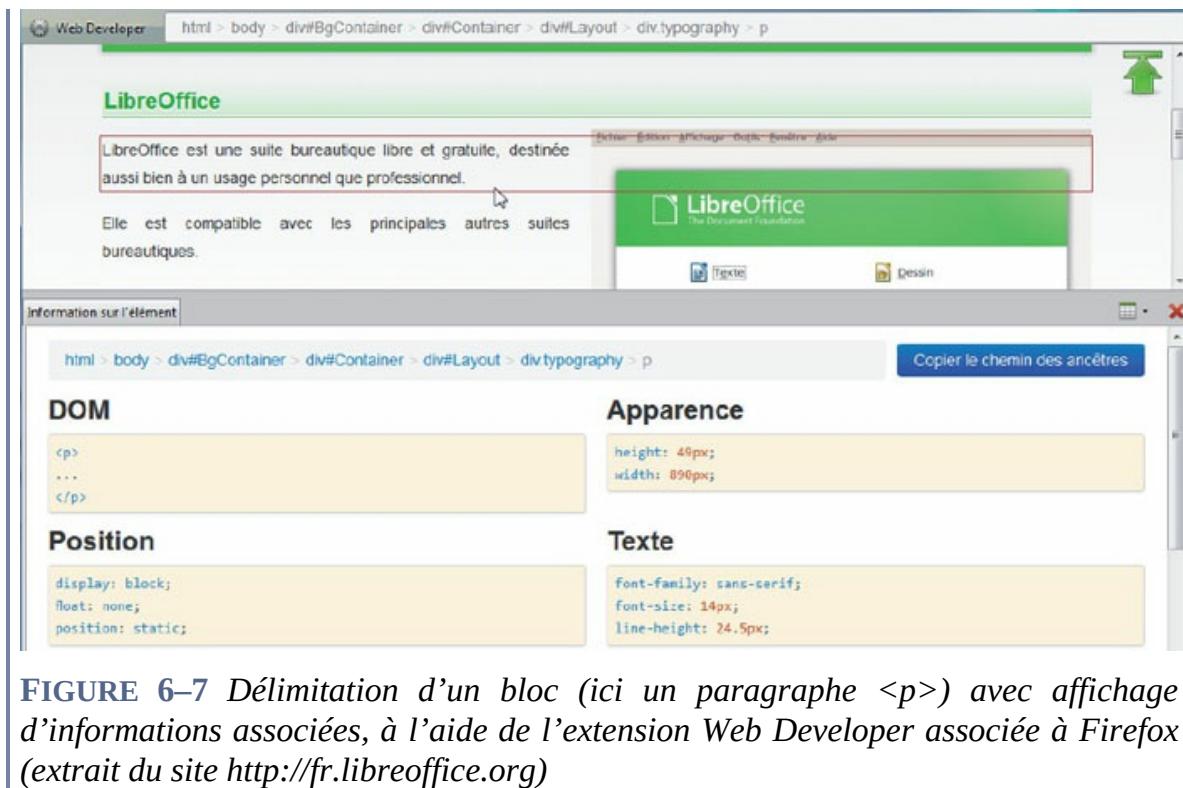


FIGURE 6–7 Délimitation d'un bloc (ici un paragraphe `<p>`) avec affichage d'informations associées, à l'aide de l'extension Web Developer associée à Firefox (extrait du site <http://fr.libreoffice.org>)

Exemples de positionnement

Pour comprendre les différents types de positionnement et leur utilisation, rien de tel qu'un exemple.

Il s'agit d'une page web complète, qui parle de nature et utilise tous les types de positionnement : un peu de fraîcheur et beaucoup d'explications !

Voici le code de cette page, dont la feuille de styles sera expliquée en détail par la suite et qui produit le résultat montré par la figure 6-8.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">

    <title>La nature : Fleurs et plantes</title>
    <style>
        * { margin: 0; padding: 0; }

        img#frise { display: block; ① width: 100%; height: 30px; }

        div#titre { background-color: skyblue; height: 60px; position: relative; } ②

        #titre img { position: absolute; top: 5px; left: 25%; ③ height: 50px; width: 40px; }

        #titre h1, #titre h2 { margin: 0 auto; ④ width: 200px; color: green; }
        #titre h1 { background-color: khaki; } ⑤

        #titre h2 { position: relative; top: -10px; left: 50px; } ⑥
        h2#titre_fixe { position: fixed; top: 200px; left: 0; ⑦ width: 80px; padding: 20px; background-color: lightgreen; text-align: center; }

        div#galerie { margin-left: 20px; } ⑧
        #galerie img { margin: 20px; width: 200px; height: 230px; border: solid 1px; display: inline-block; } ⑨

    </style>
</head>

<body>
     ⑩
    <div id="titre"> ⑪
         ⑫
        <h1>LA NATURE</h1> ⑬
        <h2>EN IMAGES</h2> ⑭
    </div>
    <h2 id="titre_fixe">Fleurs<br />et<br />plantes</h2> ⑮
</body>
```

```

<div id="galerie"> 16
  
  
  
  
  
  
</div>
</body>

</html>

```



FIGURE 6–8 Affichage de notre code exemple. Les illustrations proviennent du site www.wikipedia.org.

Image du haut (nuages)

Cette image ⑩ a été insérée avant le bloc `<div id="titre">` (bandeau uni contenant le titre principal) pour différencier, du point de vue du positionnement, le haut de l'élément `<body>` (haut de la page) et le haut de ce bloc `<div>`. Cela nous permettra de faire apparaître, pour le positionnement de l'arbre, un détail à ne pas oublier.

À NOTER Image transformée en bloc

Une image doit normalement se trouver dans un bloc, par exemple entre deux balises `<p>`. Il est toutefois possible de l'insérer directement, en la déclarant comme bloc : `display: block` ⑪. Cela évite également l'apparition d'un espace vertical sous l'image.

Image de l'arbre en position absolue

La figure 6–9 montre les modifications apportées à la position du logo en forme d'arbre, qui correspond à l'image ⑫.

1. Initialement, l'arbre est positionné dans le flux. Il se place en haut de son bloc conteneur (dont le contenu est centré par une propriété de style). Le bloc suivant (titre de niveau 1 « La nature » ⑬) est placé en dessous.
2. Cette image d'arbre est positionnée en absolu, par la règle ⑭ :

```
#titre img { position: absolute; top: 5px; left: 25%; }
```

Tant que le bloc parent `<div id="titre">` n'est pas lui-même positionné, la position de l'arbre est calculée à partir du bloc `<body>`, c'est-à-dire en partant du début de la page.
3. Il nous faut donc ajouter la règle `div#titre { position: relative; } ⑮` pour positionner ce bloc parent. La position de l'arbre est alors calculée à partir des limites du bloc `<div id="titre">` ⑯ qui le contient.

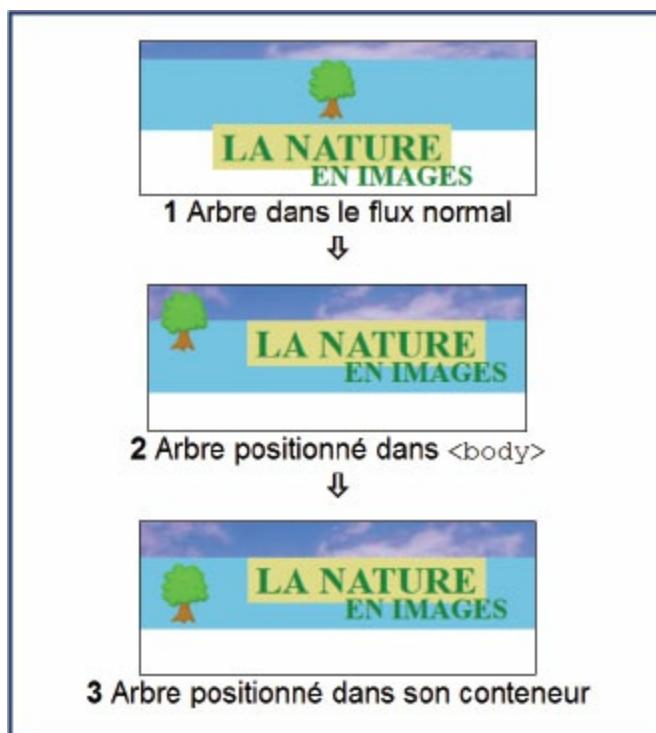


FIGURE 6–9 Les différentes étapes du positionnement de l'arbre

Sous-titre « En images » en position relative

Sur la figure 6–10 apparaissent le titre « La nature » ⑬ et le sous-titre « En images » ⑭ de la page.

Dans la première image, ils sont affichés dans le flux normal. Le sous-titre se place en dessous du titre principal, centré comme lui.

Grâce à un positionnement relatif du sous-titre « En images », il est possible de décaler celui-ci vers le haut et vers la droite, en utilisant la règle de style :
`#titre h2 { position: relative; top: -10px; left: 50px; } ⑮` qui produit :

- un décalage négatif à partir du haut, pour que le sous-titre chevauche le bloc contenant le titre « La nature » ;
- un décalage positif à partir de la gauche, pour décaler le sous-titre vers la droite.

Le résultat obtenu est visible sur la deuxième image de la figure 6–10.

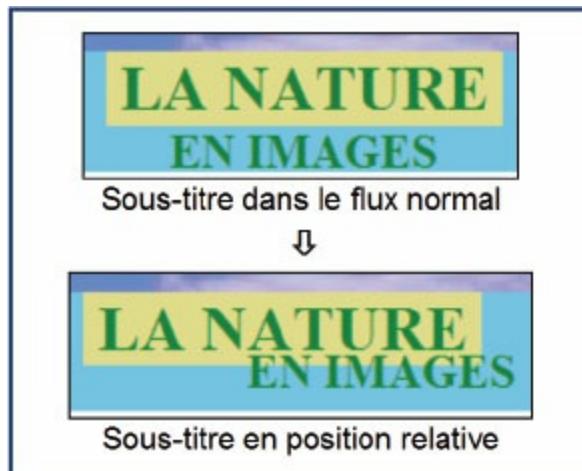


FIGURE 6–10 Décalage relatif du sous-titre par rapport à sa position normale

Centrage horizontal du titre

Le centrage horizontal des blocs `<h1>` 13 et `<h2>` 14 qui contiennent le titre « La nature » et le sous-titre « En images » s'effectue à l'aide de la règle :

```
#titre h1, #titre h2 { margin: 0 auto; } 4
```

Titre latéral fixé sur l'écran

Le titre « Fleurs et plantes » ⑯, qui se trouve à gauche de la page, doit rester visible et à la même position sur l'écran, même lorsque la page défile. La figure 6–11 illustre ce mécanisme.

Nous utiliserons pour cela la règle suivante :

```
h2#titre_fixe { position: fixed; top: 200px; left: 0; } ⑰
```

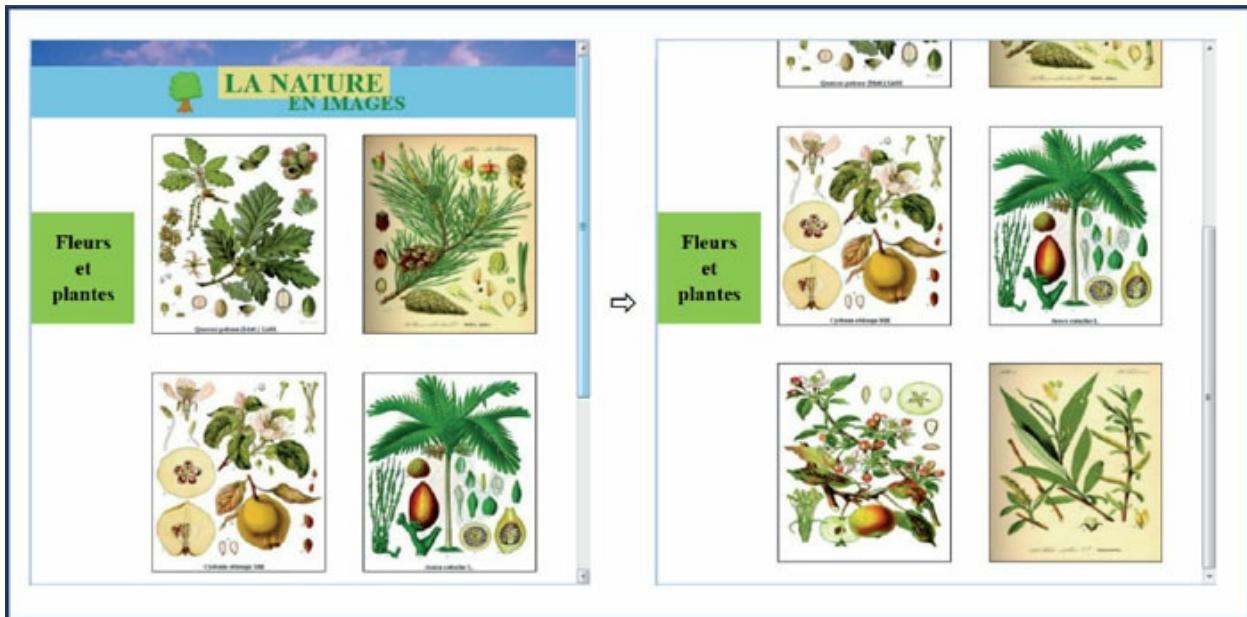


FIGURE 6–11 Le titre « Fleurs et plantes » reste fixe lors du défilement de la page.

Position de la galerie d'images

Le bloc ⑯ qui contient toutes les images doit se juxtaposer au titre latéral « Fleurs et plantes » ⑮. Il suffit de le décaler vers la droite avec une marge de gauche de 120 pixels, puisque c'est la largeur de ce titre :

```
div#galerie { margin-left: 120px; }
```

Le résultat se trouve sur la figure 6–12, où le bloc qui contient la galerie d'images est encadré par des pointillés.



FIGURE 6–12 Le bloc qui contient la galerie d'images est en retrait de 120 pixels, grâce à la marge externe de gauche.

Images côté à côté

Les images qui constituent le corps de la page sont alignées côté à côté, avec retour à la ligne automatique en fonction de la largeur de la fenêtre. Il suffit de leur attribuer le type « bloc en ligne » (`inline-block`) pour qu'elles se placent comme des caractères sur une ligne, avec un retour automatique en fin de ligne :

```
#galerie img { display: inline-block; } 9
```

La figure 6–13 montre bien que le nombre d'images par ligne s'adapte à la largeur de la fenêtre. Si les images de la galerie ont des dimensions différentes, il sera utile de les centrer verticalement en ajoutant la propriété suivante :

```
vertical-align: middle;
```

PRÉCISION Quelques efforts de présentation

Sans autre mise en forme, les images transformées en blocs en ligne vont s'agglutiner les unes aux autres de façon peu esthétique. Pour une présentation harmonieuse, il faudra :

- centrer verticalement les images avec `vertical-align` ;
- régler les marges avec `margin` ou `padding`, pour espacer les images entre elles.

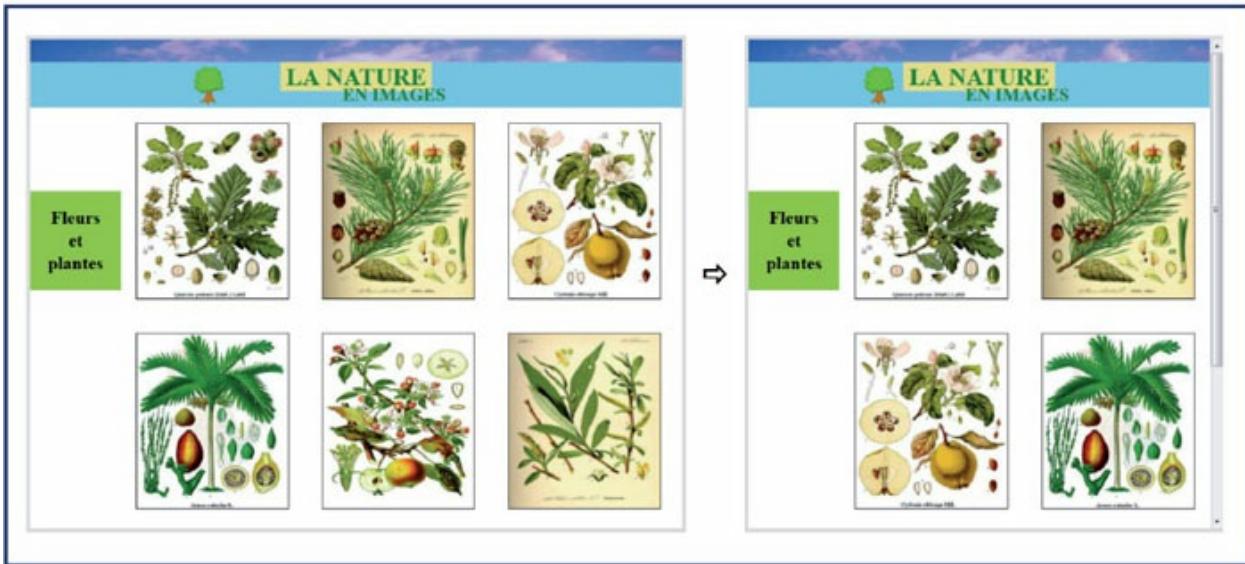


FIGURE 6-13 Grâce au type « *inline-block* », le nombre d’images sur chaque ligne s’adapte à la largeur de la fenêtre.

À NOTER Habillage d’une image

Le flottement à gauche des images (`float: left;`) est une alternative courante pour créer une galerie d’illustrations placées côte à côté. C’était la seule solution lorsque le type *inline-block* n’était pas reconnu par tous les navigateurs.

Le positionnement flottant permet aussi d’habiller une image avec du texte, comme le montre la figure 6-14 : l’image du pingouin Tux est flottante à droite ; elle est donc habillée par le texte qui la suit dans le code de la page.

Lorem ipsum dolor sit amet consectetuer neque dui habitant Nulla justo.
 Cursus massa fermentum porttitor euismod pretium justo in iaculis est
 condimentum. Quam fringilla mollis wisi congue mauris laoreet Sed nulla id
 Praesent. Proin laoreet vel auctor ante nibh congue tellus ut id
 Pellentesque. Sit est pede a Vestibulum nec ac commodo consequat a at.
 Platea tempor lacinia at ut orci. Condimentum egestas velit morbi Nam quis at vel a
 volutpat sagittis. Id at consequat Nunc porttitor tincidunt Morbi risus rutrum elit
 semper. Nec et natoque tellus Curabitur Quisque id est ac augue Quisque. Ut sed pretium
 nec mattis ipsum facilisi et laoreet orci nunc. Odio massa venenatis habitant euismod
 felis hendrerit id laoreet.



FIGURE 6-14 Image flottante à droite, habillée par le texte qui la suit

Centrage d'éléments à l'intérieur des blocs

Il est souvent nécessaire de centrer textes, images ou blocs, soit horizontalement, soit verticalement. Voici, de façon simple, comment procéder avec les feuilles de styles.

Dans chaque cas, il faudra distinguer :

- le centrage des *éléments en ligne* (texte, images...), qui s'affichent les uns à la suite des autres ;
- le centrage des *blocs* (`<div>`, `<p>`, `<h1>`, ...), éléments qui se placent les uns sous les autres et qui n'utilisent pas les mêmes propriétés.

Centrage horizontal

Mieux vaut éviter la très vieille balise `<center>` (abandonnée depuis la version 4 du HTML !) ou les très anciens attributs `align="center"` à l'intérieur des balises : bien qu'elles fonctionnent, ces méthodes sont à la fois obsolètes et déconseillées, car elles mélangent mise en forme et contenu de la page.

Centrage horizontal d'éléments en ligne

Pour centrer horizontalement un *texte* ou un *élément en ligne*, il suffit d'attribuer à son *bloc conteneur* la propriété suivante :

```
text-align: center;
```

Centrage horizontal de blocs

Le centrage horizontal d'un *bloc* à l'intérieur de son conteneur consiste à lui attribuer des marges identiques à gauche et à droite, en utilisant les propriétés suivantes :

```
margin-left: auto; margin-right: auto;
```

ou avec une seule propriété, comme dans cet exemple où les marges du haut et du bas sont nulles (marges verticales nulles et centrage horizontal) :

```
margin: 0 auto;
```

À NOTER Largeur du bloc à centrer

Pour que le centrage horizontal change quelque chose à l'affichage, il faut s'assurer que le bloc à centrer soit plus étroit que son bloc conteneur, en définissant sa largeur `width`.

Centrage vertical

Là, il n'y a aucun risque d'utiliser une balise périmée, puisqu'il n'en existe pas en HTML pour centrer verticalement ! Il faudra utiliser plusieurs propriétés pour placer l'élément au milieu de son conteneur.

Centrage vertical d'éléments en ligne

Pour centrer verticalement un élément en ligne, il faut déclarer une *hauteur de ligne* égale à la hauteur de l'élément. Exemple :

```
p.milieu { height: 150px; line-height: 150px; }
```

À NOTER Centrage d'éléments comprenant plusieurs lignes

La technique précédente ne fonctionne que si les éléments à centrer tiennent sur une seule ligne.

Pour un élément à centrer verticalement et qui comporte deux ou plusieurs lignes, la méthode suivante est applicable :

Code HTML

```
<span>texte sur une ligne</span>  
<span class="sur2lignes">texte plus long,<br /> sur deux lignes</span>
```

Code CSS

```
span { height: 2rem; line-height: 2rem; }  
span.sur2lignes { line-height: 1rem; }
```

Cette technique est généralisable à un nombre quelconque de lignes, avec `line-height = height/nombre de lignes`.

Centrage vertical de blocs

Les blocs à centrer verticalement doivent évidemment avoir une hauteur inférieure à celle de son conteneur.

Le centrage vertical d'un bloc utilise le principe suivant, qui est illustré par la figure 6–15.

- Les blocs à centrer verticalement sont affichés en tant qu'éléments de

type `inline-block`; il s'agit de blocs posés sur une ligne. Il faut ajouter un centrage vertical sur la ligne.

- Le bloc conteneur possède une hauteur de ligne `line-height` égale à sa hauteur `height`.
- La propriété `line-height` étant héritée, il ne faut pas oublier de rétablir une hauteur de ligne normale (`1.3rem`) pour tous les éléments inclus dans le conteneur, avec un sélecteur du type : `div.conteneur *`.

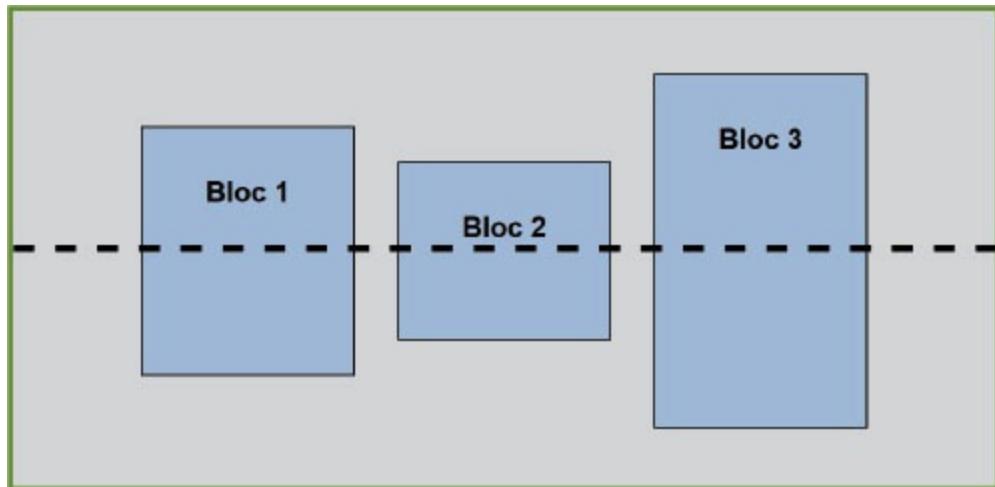


FIGURE 6–15 Centrage vertical de trois blocs à l'intérieur d'un conteneur

Exemple de centrage vertical

Pour le conteneur

```
height: 250px;  
line-height: 250px; /* line-height = height */
```

Pour les éléments inclus dans le conteneur

```
line-height: 1.3rem;
```

Pour le bloc à centrer

```
display: inline-block;  
vertical-align: middle;
```

Avec ces propriétés liées aux blocs et au positionnement d'éléments, nous avons à présent fini le tour des propriétés de style de base – communes aux CSS 2 et aux CSS 3 – destinées à mettre en forme les pages web affichées à

l'écran.

Peut-être auriez-vous aimé trouver ici certaines autres fonctions pour embellir vos pages... Ne vous inquiétez pas, le chapitre suivant vous propose d'aller plus loin en découvrant de nouvelles possibilités offertes par la norme CSS 3.

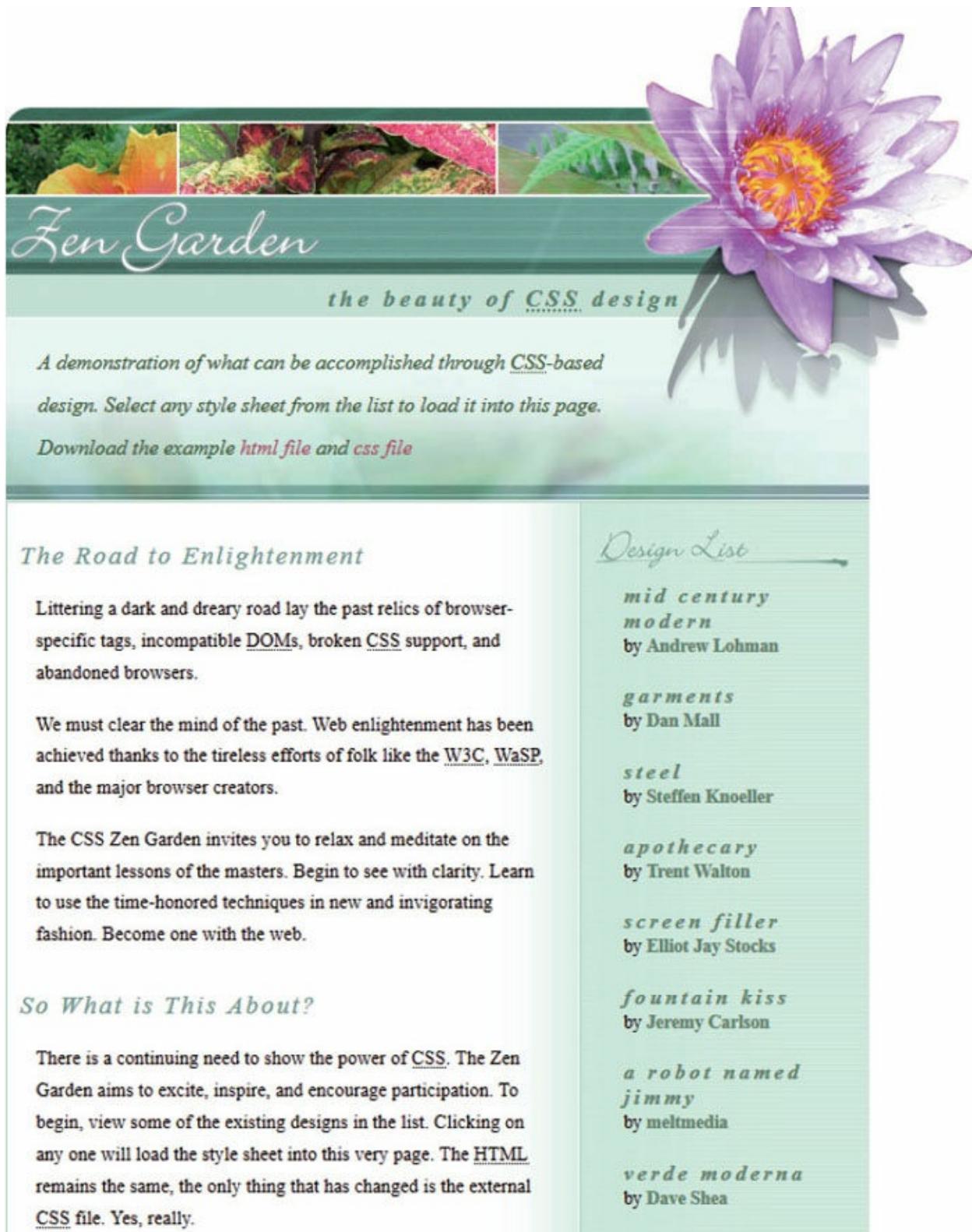


FIGURE 6–16 Les pages web sont constituées de blocs. Ceux-ci sont bien visibles dans cet exemple, extrait du site <http://www.csszengarden.com/tr/francais> (version « 15 Petals », par Eric Meyer et Dave Shea, deux grands spécialistes des CSS).

chapitre 7

Principales nouveautés des CSS 3



Depuis l'arrivée des CSS 3 dans l'univers du Web, il est beaucoup plus facile de donner de l'allure à nos pages. Nous pouvons notamment créer des effets de mise en forme et d'animation très intéressants, sans

avoir à utiliser d'images, ni de programmes en JavaScript.

SOMMAIRE

- ▶ **Du relief pour nos pages**
- ▶ **Codes couleurs et niveaux de transparence**
- ▶ **De nouveaux effets pour le texte**
- ▶ **Des bordures plus variées**
- ▶ **Couleurs et images de fond**
- ▶ **Multicolonnage**
- ▶ **La flexbox pour répartir des blocs**
- ▶ **Transformations géométriques**
- ▶ **Le Web s'anime en CSS 3**
- ▶ **Nouveaux types de sélecteurs**
- ▶ **Les CSS 3 : bientôt et déjà !**

Les bases des feuilles de styles que nous avons étudiées nous ont permis de mettre en forme nos pages en utilisant un codage clair et facile à mettre à jour. Il nous reste à progresser vers des styles graphiques plus évolués : cette possibilité nous est offerte très simplement, grâce aux nouveautés des CSS 3.

ATTENTION Normes en cours de préparation

Si alléchantes soient-elles, quelques-unes de ces normes CSS 3 sont encore en cours de finalisation, voire sur certains points à l'état de brouillon au moment de la parution de cet ouvrage. Il faut donc être patient et les utiliser petit à petit, en tenant compte de leur possible évolution et, bien sûr, de leur prise en compte progressive par les différents navigateurs. N'oublions pas non plus de conserver une compatibilité ascendante, pour que les internautes ne disposant pas d'un butineur dernier cri puissent néanmoins consulter nos pages web.

Parfois, les navigateurs implantent certaines propriétés avec un préfixe qui leur est propre, en attendant leur publication ou la stabilisation de leur syntaxe. Ces préfixes sont `-moz-` pour Mozilla Firefox, `-webkit-` pour les navigateurs utilisant ce moteur de rendu (Safari, Konqueror, Android) ou sa version optimisée *Blink* (Chrome, Chromium, Opera à partir de sa version 15), `-ms-` pour Edge et Internet Explorer.

Par exemple, si la propriété créant des coins arrondis (que nous découvrirons plus loin) est maintenant bien connue des navigateurs et ne nécessite plus de préfixe, il y a quelques années, nous aurions pu la répéter suivant plusieurs variantes, de façon à englober plusieurs versions de navigateur :

```
-moz-border-radius: 20px;  
-webkit-border-radius: 20px;  
border-radius: 20px;
```

Les propriétés utilisant un préfixe correspondent à des versions provisoires et peuvent présenter quelques différences avec la version définitive, celle qui n'a pas de préfixe. C'est pourquoi il est préférable d'écrire d'abord les propriétés accompagnées d'un préfixe, puis en dernier celle qui sera devenue « officielle » et qui n'a pas de préfixe.

Une alternative à cette répétition consiste à utiliser « `-prefix-free` », un programme JavaScript qui nous dispense de cette écriture multiple, et

qui est disponible à cette adresse :

▶ **<http://leaverou.github.io/prefixfree/>**

Du relief pour nos pages

Une présentation plate n'est plus tellement au goût du jour. Il nous faut du relief, des ombres, des dégradés... pour que notre site ait de l'allure !

Lorsque ces effets sont créés sous forme d'images, la durée de conception de nos beaux écrans est considérablement allongée : il faut dessiner la forme, lui donner la bonne dimension, transférer le fichier et l'intégrer au contenu. À la balise image correspondante, il faut généralement appliquer une technique de positionnement dans la feuille de styles. Lorsqu'il s'agit d'en modifier les couleurs ou les dimensions, par exemple, il ne nous reste plus qu'à retourner dans notre logiciel de traitement d'image pour modifier l'image, voire la recréer...



FIGURE 7-1 Des cadres arrondis, obtenus à l'aide de blocs imbriqués et de plusieurs images (extrait du site www.claffitte.fr).

Les effets d'animation, quant à eux, utilisent souvent la programmation en JavaScript, liée à des bibliothèques qu'il faut inclure dans les pages, ou bien la technologie propriétaire Flash, ce qui nécessite alors un logiciel de création spécifique.

Les nouvelles propriétés CSS 3 apportent une solution aux problèmes les plus courants de ce type. Cela va nous éviter l'utilisation de bien des images et autres scripts : il suffira d'écrire une balise avec son contenu sous forme de texte, puis de lui associer une ou deux propriété(s), et le tour sera joué !

Voici donc les principales innovations que les CSS 3 nous offrent. Elles seront utilisables au fur et à mesure de leur prise en compte par les navigateurs, mais les plus modernes d'entre eux les reconnaissent déjà. Il faudra toutefois s'assurer que l'affichage des pages reste acceptable sur les anciennes versions de navigateurs.

Codes couleurs et niveaux de transparence

Les évolutions en matière de gestion des couleurs, et d'opacité en particulier, constituent un grand bond en avant pour l'esthétique de nos pages : nous allons enfin pouvoir, très simplement, leur attribuer un degré de transparence.

Niveau d'opacité

En l'absence de normes universelles, l'utilisation du critère d'opacité nécessitait jusqu'ici des codes spécifiques à certains navigateurs. La nouvelle propriété `opacity` va permettre des effets de semi-transparence très intéressants, à partir d'éléments superposés.

TABLEAU 7–1 Propriété `opacity`

Propriété	<code>opacity</code>
Exemple	<code>div.pub { background-color: red; opacity: 0.5; }</code>
Valeurs possibles	Nombre compris entre <code>0</code> (transparence totale) et <code>1</code> (aucune transparence – valeur par défaut) Attention : le séparateur décimal est le point.
Héritage	<i>Non</i>

Codage RGBA des couleurs

Si le codage RGB des couleurs (*Red Green Blue*, soit en français RVB, pour *Rouge Vert Bleu*) est toujours utilisé, il existe à présent une version améliorée intitulée RGBA. Le *A* désigne la couche *Alpha*, c'est-à-dire le niveau de transparence. Équivalent à l'opacité, il est donné de la même façon par une valeur comprise entre 0 (transparence complète) et 1 (opacité totale).

Utilisation du codage RGBA avec la propriété color

```
| h1 { color: rgba(100%, 0, 0, 1); }  
| h2 { color: rgba(0, 0, 100%, 0.5); }
```

Dans cet exemple, les grands titres `<h1>` sont rouges et opaques, alors que les sous-titres `<h2>` sont bleus et translucides.

RAPPEL Valeurs possibles pour les couleurs

Les trois couleurs, les trois premiers paramètres à indiquer pour `rgba(...)` par conséquent, peuvent être codées de deux façons :

- soit par des pourcentages compris entre 0 % et 100 % (sans espace entre le nombre et le signe %) ;
- soit par des valeurs numériques allant de 0 à 255.

Ces trois couleurs représentent respectivement la quantité de rouge (*red*), de vert (*green*) et de bleu (*blue*).

Pour bénéficier de cette couche Alpha tout en conservant une compatibilité avec les anciens navigateurs, il est utile d'effectuer deux déclarations de couleur, en utilisant d'abord `rgb(...)`, puis `rgba(...)`, comme dans l'exemple suivant :

```
| div { background-color: rgb(183, 232, 255); }  
| div { background-color: rgba(183, 232, 255, 0.7); }
```

Ainsi, les navigateurs qui ne comprennent pas la deuxième ligne l'ignoreront, tandis que les navigateurs modernes prendront en compte cette seconde déclaration, qui remplacera alors la première.

Pour transformer en décimal un code couleur écrit en hexadécimal, des outils sont à notre disposition. Nous avons par exemple :

- un convertisseur inclus dans nombre de logiciels, comme *PsPad* ;
- un utilitaire en ligne comme celui disponible à la page <http://lab.darklg.me/GiveMeRGBA>, qui génère automatiquement la règle `background-color` et affiche en fond la couleur obtenue.

Pour le bloc contenant la montre et le texte :

`background-color: white; opacity: 0.4;`



Pour le bloc contenant la montre et le texte :

`background-color: rgba(100%, 100%, 100%, 0.4);`



FIGURE 7–2 Bien qu'elle ne soit pas héritée, la propriété `opacity` agit sur tout le contenu du bloc concerné (image du haut). En revanche, une propriété `background-color` intégrant un niveau de transparence n'affecte que la couleur de fond de ce bloc (image du bas).

Codages HSL et HSLA des couleurs

Une nouvelle méthode pour coder les couleurs apparaît en CSS 3 : il s'agit du HSL (*Hue, Saturation, Lightness* – soit en français TSL, comme Teinte, Saturation, Luminosité).

PRÉCISION Codage des couleurs en TSL (HSL en CSS 3)

Le codage RVB (RGB pour les CSS) est courant en informatique, tandis que le codage TSL (HSL pour le codage web) se rapproche de la perception humaine des couleurs.

- La teinte représente la couleur, indiquée par un angle (en degrés) dans le cercle chromatique des couleurs : 0 = rouge, 60 = jaune, 120 = vert, 180 = cyan, 240 = bleu et 360 à nouveau rouge. Le symbole « degré » ne sera pas mentionné dans le code HSL.
- La saturation est un pourcentage qui donne la pureté de la couleur choisie, du gris pour 0 % à la couleur la plus vive avec 100 %, et plus ou moins de mélange avec le gris entre ces deux valeurs.
- La luminosité est également un pourcentage, où 0 % correspond au noir et 100 % au blanc.

Le cercle chromatique des couleurs est décrit dans l'article suivant :

▶ https://fr.wikipedia.org/wiki/Cercle_chromatique

Ce codage HSL s'accompagne de sa variante HSLA, dans laquelle le *A* représente toujours la couche Alpha, donc le degré d'opacité compris entre 0 et 1.

Exemples de codages HSL et HSLA

```
p.remarque { color: hsl(320,100%,48%); }
ul.menu { background-color: hsl(251,73%,72%); }

h2 { background-color: hsla(39,47%,77%,0.5); }
```

Les deux premiers exemples correspondent respectivement aux couleurs fuchsia et lavande. Pour le troisième, c'est une couleur sable avec un effet translucide, puisqu'il utilise un codage sous la forme `hsla(...)` au lieu de

`hs1(...)` pour les deux premiers.

Une liste de couleurs, avec leur codage sous différentes normes, est disponible à l'adresse suivante :
https://fr.wikipedia.org/wiki/Liste_de_noms_de_couleur.

De nouveaux effets pour le texte

Il était temps que nous puissions mettre un peu de fantaisie dans nos textes ! Nous n'aurons plus besoin d'inclure sous forme d'images des contenus écrits dans une police particulière ou avec un peu de relief.

Style d'écriture

De nouveaux styles et un choix plus large de polices nous sont proposés en CSS 3. Cela nous permettra une variété plus grande dans la conception de nos pages, tout en conservant la souplesse apportée par un contenu modifiable, plus pratique qu'une image.

Couleur, forme et type de soulignement

La propriété de soulignement du texte `text-decoration` a été étendue en CSS 3, pour que nous puissions choisir la couleur, la forme et le type de soulignement. Ces variantes sont récentes ; il faudra donc vérifier leur prise en compte par les navigateurs et éventuellement utiliser des préfixes.

Couleur de soulignement

Initialement de la même couleur que le texte, le soulignement peut à présent être d'une couleur différente, grâce à la propriété `text-decoration-color`.

TABLEAU 7–2 Propriété `text-decoration-color`

Propriété	<code>text-decoration-color</code>
Exemple	<pre>.ref { color: black; text-decoration: underline; text-decoration-color: green; }</pre>
Valeurs possibles	Nom ou code de couleur. Par défaut, la couleur de soulignement est celle du texte.
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>initial</code> pour annuler l'héritage et retrouver la couleur par défaut, à savoir celle de texte.

Forme du soulignement

Pour modifier le soulignement par un trait continu, la propriété CSS 3 `text-decoration-style` nous propose différentes variantes, dont le trait ondulé, similaire à celui qui indique les fautes d'orthographe dans un logiciel de traitement de texte.

TABLEAU 7–3 Propriété `text-decoration-style`

Propriété	<code>text-decoration-style</code>
Exemple	<code>a { text-decoration: underline; text-decoration-style: dashed; }</code>
Valeurs possibles	<code>solid</code> (trait continu simple, valeur par défaut), <code>double</code> (trait continu double), <code>dotted</code> (suite de points), <code>dashed</code> (suite de tirets) et <code>wavy</code> (trait ondulé)
Héritage	<i>Non</i>

Position du trait

La propriété `text-decoration-line` est dédiée à la position du trait : en dessous ou au-dessus du texte, ou encore à travers les lettres pour rayer un ou plusieurs mots.

TABLEAU 7–4 Propriété `text-decoration-line`

Propriété	<code>text-decoration-line</code>
Exemple	<code>h3 { text-decoration-line: overline; }</code>
Valeurs possibles	<code>none</code> (aucun trait, valeur par défaut), <code>underline</code> (trait sous le texte), <code>overline</code> (trait au-dessus du texte) et <code>line-through</code> (trait sur les lettres pour rayer un texte)
Héritage	<i>Non</i>

À première vue, la propriété `text-decoration-line` semble faire doublon avec la propriété CSS 2 `text-decoration`, puisqu'elle propose les mêmes valeurs. Son apparition est due au fait qu'en CSS 3, `text-decoration` est considérée comme un raccourci des trois propriétés `text-decoration-line`, `text-decoration-style` et `text-decoration-color`, bien que son utilisation sous cette forme ne soit à l'heure actuelle pas encore comprise par les navigateurs.

La position du trait étant obligatoire et les deux autres caractéristiques (type et couleur du trait) facultatives, la nouvelle définition en CSS 3 de la propriété `text-decoration` est bien compatible avec son utilisation précédente, telle qu'elle a été définie en CSS 2.

Ajustement de la taille des caractères

L'ajustement des lettres bénéficie de deux nouvelles propriétés qui vont nous

permettre d'une part d'augmenter ou diminuer la largeur des lettres et, d'autre part, d'homogénéiser la taille des caractères.

Étirement des caractères

La propriété `font-stretch` permet de diminuer ou augmenter la largeur des lettres, avec comme valeurs principales `normal`, `condensed` (lettres étroites) et `expanded` (lettres élargies).

TABLEAU 7–5 Propriété `font-stretch`

Propriété	<code>font-stretch</code>
Exemples	<code>h1 { font-stretch: extended; }</code> <code>p.note { font-stretch: condensed; }</code>
Valeurs possibles	<code>normal</code> (valeur par défaut) ; Plusieurs niveaux de largeur de lettre, du plus étroit au plus large dans l'ordre : <code>ultra-condensed</code> , <code>extra-condensed</code> , <code>condensed</code> , <code>semi-condensed</code> , <code>normal</code> , <code>semi-expanded</code> , <code>expanded</code> , <code>extra-expanded</code> , <code>ultra-expanded</code>
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>normal</code> pour annuler l'héritage.

Adaptation de la taille de la police

Lorsque plusieurs polices sont proposées à la suite par la propriété `fontfamily`, le navigateur recherche si la première est disponible sur sa machine, sinon il passe à la deuxième, et ainsi de suite. Il en résulte qu'une même page risque d'être affichée avec des polices distinctes sur différents appareils.

La taille de police définie par `font-size` reste bien sûr la même, mais elle correspond à la taille des majuscules, tandis que les minuscules peuvent présenter des disparités importantes.

En effet, pour une même taille, les polices peuvent avoir chacune un « œil » différent : il s'agit de la dimension d'une lettre minuscule comme un « o ». La lisibilité en est affectée, car elle est liée de près à la taille des minuscules, et la mise en page peut aussi en souffrir, si la largeur totale du texte augmente ou diminue.

La nouvelle propriété `font-size-adjust` essaie de résoudre ce problème en imposant un rapport constant entre la taille des minuscules et celle des

majuscules, quelle que soit la police affichée. Elle est encore expérimentale ; avant de l'utiliser, il sera donc utile de consulter par exemple le site www.caniuse.com, pour savoir si elle est bien reconnue par les navigateurs.

TABLEAU 7–6 Propriété font-size-adjust

Propriété	<code>font-size-adjust</code>
Exemple	<code>p { font-family: Tahoma, Arial, sans-serif; font-size: 1rem; font-size-adjust: 0.5; }</code>
Valeurs possibles	<code>none</code> (pas d'ajustement, valeur par défaut), ou un nombre qui correspond au rapport entre la taille d'un « x » minuscule et celle d'un « X » majuscule (il s'agit donc une valeur inférieure à 1).
Héritage	Propriété <i>héritée</i> . Utiliser la valeur <code>none</code> ou <code>initial</code> pour annuler l'héritage.

Cette égalisation de la taille des lettres nous donne envie de diversifier les polices de caractères dans nos pages, et nous allons voir de quelle façon nous pourrons élargir notre palette.

Citations de Jean-Claude Vandamme

Sans font-size-adjust

Pas besoin de flash pour photographier un lapin qui a déjà les yeux rouges...
Pas besoin de flash pour photographier un lapin qui a déjà les yeux rouges...

Avec font-size-adjust

Pas besoin de flash pour photographier un lapin qui a déjà les yeux rouges...
Pas besoin de flash pour photographier un lapin qui a déjà les yeux rouges...

FIGURE 7–3 Le même texte de taille identique 1rem (soit 16px), affiché en police Verdana, puis en Calibri : différence avant et après utilisation de la propriété `font-size-adjust`. Le résultat est très bon dans ce cas, mais l'efficacité de cette méthode est variable et dépend des polices employées.

Une police originale avec @font-face

Comme nous l'avons vu précédemment, les polices de caractères qui s'affichent à l'écran sont celles présentes sur l'ordinateur de l'internaute qui consulte notre site. Cela limite notre choix à une liste de polices extrêmement réduite, celles que possèdent toutes les configurations, même les plus basiques.

Pour égayer nos pages, la solution consistait jusqu'ici à créer des images, jusqu'à ce que la règle `@font-face` nous permette d'envoyer aux visiteurs la police de caractères utilisée. Il suffit de lui attribuer un nom et d'indiquer l'adresse où elle sera automatiquement téléchargée.

Exemple de définition de police

```
@font-face {  
    font-family: Bellepolice;  
    src: url(http://www.monsite.fr/pol/mapolice.ttf);  
}  
  
p { font-family: Bellepolice, serif; }
```

NORME Reconnaissance des standards

La règle `@font-face` existait déjà dans la norme CSS 2, puis elle a été retirée en CSS 2.1 avant de revenir avec les CSS 3. Le manque de standardisation vient principalement du fait que tous les navigateurs ne reconnaissent pas les mêmes types de fichiers de police. En particulier, Internet Explorer jusqu'à sa version 8 ne reconnaît qu'un format spécifique .eot (*Embedded Open Type*) et il est le seul à l'utiliser, alors que les formats .ttf (*True Type Font*), .otf (*Open Type Font*) et .woff (*Web Open Font Format*) sont plus largement répandus parmi les autres navigateurs. Il sera donc parfois utile de déclarer, pour une police donnée, plusieurs fichiers sources séparés par des virgules comme dans cet exemple :

```
src: url(mapolice.ttf), url(mapolice.eot)
```

Si le navigateur ne peut pas lire le format du premier fichier, il passera au deuxième.

Le nom de la police est au choix ; il s'agit ici de *Bellepolice*. Il est à placer

entre guillemets s'il comporte un ou plusieurs espaces. L'attribut `src` indique l'adresse Internet du fichier `mapolice.ttf` contenant les éléments de cette police.

La dernière ligne de cet exemple reprend ce nouveau nom de police dans la propriété `font-family`, pour l'appliquer à tous les paragraphes `<p>`.

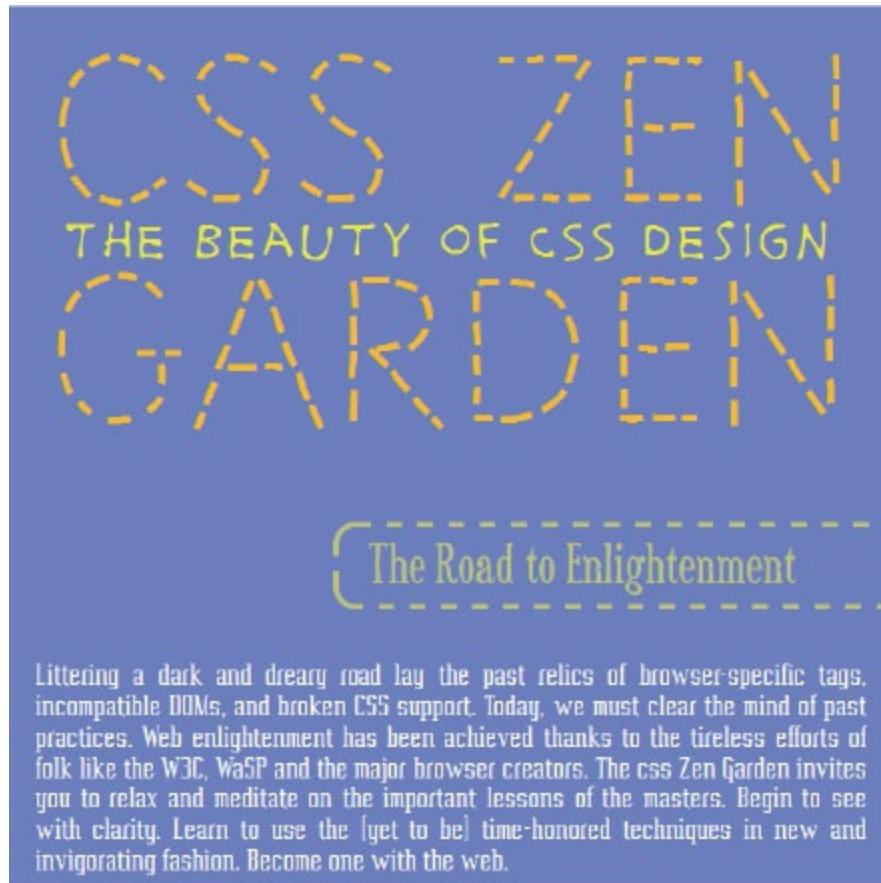


FIGURE 7–4 La fantaisie arrive dans les polices de caractères ! (extrait de la page <https://alistapart.com/article/cssatten>)

Il faut noter qu'une police ainsi définie ne concerne qu'un seul type de mise en forme. Par exemple, il faudra trois autres fichiers pour pouvoir l'utiliser en gras, en italique, ou en gras et italique.

Déclaration complète d'une police de caractères

```
@font-face {  
    font-family: Bellepolice;  
    font-weight: normal;  
    font-style: normal;  
    src: url(/polices/mapolice.otf);
```

```

}
@font-face {
    font-family: Bellepolice;
    font-weight: bold;
    font-style: normal;
    src: url(/polices/mapolice-bold.otf);
}
@font-face {
    font-family: Bellepolice;
    font-weight: normal;
    font-style: italic;
    src: url(/polices/mapolice-italic.otf);
}
@font-face {
    font-family: Bellepolice;
    font-weight: bold;
    font-style: italic;
    src: url(/polices/mapolice-bolditalic.otf);
}

```

Les propriétés utilisables dans une règle `@font-face` sont les suivantes.

- `font-family` indique le nom donné à la police, nom qui sera employé dans la feuille de styles.
- `src:local(...), url(...)` `format("...")` fournit le nom de la police locale, en l'absence de laquelle le navigateur prend le fichier défini par `url(...)` dont le format peut éventuellement être indiqué, avec comme valeurs possibles "truetype" (extension .ttf), "opentype" (extension .ttf ou .otf), "woff" (*Web Open Font Format*, extension .woff), "embedded-opentype" (extension .eot pour Internet Explorer seul) ou "svg" (extension .svg ou .svgz).
- On trouve `font-weight`, `font-style` et `font-stretch` avec les valeurs de ces propriétés CSS, principalement `normal` et `bold` pour la première, `normal` et `italic` pour la deuxième, `normal`, `condensed` et `expanded` pour la troisième.
- Quant à la propriété `unicode-range`, elle sera peu utilisée, donnant la panoplie de caractères disponibles sous forme de plage de codes, par défaut "`U+0-10FFFF`". Cela permet d'indiquer, par exemple, si la police en question prend en charge des langues comme l'hébreu, le japonais, et autres caractères spéciaux.

@font-face pour une police locale

```
@font-face {  
    font-family: Courier;  
    src: local("Courier New");  
}  
p.exemple { font-family: Courier, monospace; }
```

Dans cet exemple, l'emploi de la forme `src:local(...)` sert à désigner une police qui existe sur l'ordinateur de notre visiteur, en lui donnant un nouveau nom : ici `Courier` au lieu de `"Courier New"`, pour éviter l'emploi de guillemets en appelant cette police avec un seul mot.

ATTENTION Inconvénients des fichiers de polices

Si cette technique est intéressante pour rompre la monotonie de l'écriture un peu standardisée de nos pages, elle présente quelques inconvénients.

- Si la taille d'un fichier de police est en général de quelques dizaines de kilooctets, elle peut atteindre 200 à 300 Ko. Cette taille est à multiplier par quatre pour obtenir la version complète, incluant gras, italique et gras plus italique ensemble.
- Suivant le système d'exploitation et le navigateur, une même police pourra s'afficher de façon assez différente, et le lissage des grandes tailles de caractère sera de qualité variable. Il conviendra donc de tester plusieurs configurations avant de l'adopter.
- La grande liberté de choix apportée par cette technique ne nous dispense pas d'être attentifs à la lisibilité de nos sites, ni de conserver une certaine homogénéité de nos pages : 2 à 3 polices au maximum pour éviter de perdre le lecteur au milieu d'une trop grande variété d'écritures, chaque style étant supposé apporter un sens à son contenu.

Police à télécharger seulement si elle est inconnue

```
@font-face {  
    font-family: Mistral;  
    src: local(Mistral), url(/fonts/Mistral.ttf), url(/fonts/Mistral.otf)  
        format("opentype"), url(/fonts/Mistral.ttf) format ("truetype");  
}
```

Si la police à utiliser n'est pas universelle mais assez connue, pourquoi obliger l'internaute à la télécharger ? Il l'a peut-être déjà sur son ordinateur ! C'est pourquoi la syntaxe de l'exemple ci-dessus permet de rechercher une police locale, tout en proposant le téléchargement lorsqu'elle n'est pas présente. Si plusieurs formats de police sont proposés comme ici, le premier qui est connu du navigateur sera pris en compte.

Notez que dans cet exemple, le fichier contenant la police se trouve sur le site lui-même, à une adresse donnée à partir de sa racine.

Ombrage du texte

Voici l'exemple typique d'un effet qui nécessitait jusqu'ici un logiciel de traitement d'images tel que Gimp, PhotoFiltre, PhoXo ou Inskcape : donner du relief à un texte, grâce à une ombre portée qui apparaît en arrière-plan. La propriété `text-shadow` va ombrer notre texte. Il suffira de lui indiquer le décalage horizontal et vertical nécessaire entre les lettres et leur ombre, ainsi que l'étendue du flou et la couleur de cette ombre.

TABLEAU 7–7 Propriété `text-shadow`

Propriété	<code>text-shadow</code>
Exemples	<pre>h2 { text-shadow: 2px 2px 2px blue; } h1 { text-shadow: 3px -3px 3px yellow, 3px -6px 3px red; } p { text-shadow: none; }</pre>
Valeurs possibles	<p>none (valeur par défaut, aucun ombrage) ou quatre valeurs :</p> <ul style="list-style-type: none">- 1^{re} valeur : décalage en x de l'ombre (positif = vers la droite)- 2^e valeur : décalage en y de l'ombre (positif = vers le bas)- 3^e valeur : étendue du flou, aucun flou si cette valeur est absente ou égale à 0- couleur : couleur de l'ombre (code couleur ou mot clé) <p>Plusieurs ombres peuvent être définies à la suite dans une même propriété <code>text-shadow</code>, séparées par des virgules : dans ce cas, les différents effets d'ombre se superposent. Les décalages sont des dimensions positives ou négatives. La couleur peut éventuellement être placée en première position.</p>
Héritage	Cette propriété est <i>héritée</i> . Pour annuler l'ombrage, utilisez <code>none</code> .

Bienvenue sur notre site

L'été sera chaud !

Contactez-nous...

FIGURE 7-5 Apportez une touche de relief avec un text-shadow simple. Ou bien « allumez le feu » avec un text-shadow multiple : à réserver aux titres !

Présentation et ergonomie

Les quelques propriétés qui suivent nous permettent de modifier l'aspect d'un élément, de faciliter la navigation, ainsi que d'affiner la façon dont s'effectuent les coupures de lignes et de mots dans les paragraphes.

ERGONOMIE Propriétés facilitant la navigation

Des propriétés sont prévues pour faciliter la navigation à l'aide des flèches du clavier, améliorant ainsi l'accessibilité de nos pages. Elles ne sont pas encore comprises par la majorité des navigateurs, mais cela ne saurait tarder, espérons-le !

Le nom de ces propriétés commence toujours par `nav-` comme `navigation`.

- `nav-index` remplacera l'attribut `tabindex` pour indiquer l'ordre de tabulation, nombre entier supérieur ou égal à 1. La valeur 1 correspond au premier élément, le nombre 2 sera le suivant, etc. La valeur par défaut est `auto`, l'ordre utilisé pour les éléments correspondant à leur place dans le contenu de la page.
- Quatre propriétés gèrent les déplacements dans la page avec les quatre flèches du clavier, à partir de l'élément sélectionné. Ces propriétés `nav-up`, `nav-down`, `nav-left` et `nav-right` prennent chacune pour valeur un texte qui est l'identifiant d'un autre élément de la page. Lorsqu'est sélectionné l'élément ayant ces propriétés, appuyer sur les touches ***haut, bas, gauche et droite*** placera le focus sur l'élément repéré par l'identifiant indiqué.

L'utilisation de ces propriétés concernera principalement les formulaires et les liens hypertextes d'une page ou d'un menu.

Apparence d'un élément

Les CSS 3 nous donnent la possibilité de changer l'apparence d'un élément, en particulier dans les formulaires, un paragraphe pouvant alors devenir un bouton, par exemple. Plus généralement, un élément HTML quelconque pourra prendre la forme d'un autre élément, grâce à la propriété `appearance`.

TABLEAU 7–8 Propriété appearance

Propriété	appearance
Exemples	.valider { appearance: push-button; } .choix { appearance: radio-button; }
Valeurs possibles	Les valeurs principales sont <code>normal</code> (valeur par défaut, apparence standard), type d'apparence comme : <code>icon</code> (petite image), <code>window</code> (fenêtre), <code>button</code> (bouton), <code>menu</code> (liste d'options) et <code>field</code> (zone de saisie de type <code><input></code>).
Héritage	Non

Aux valeurs courantes de la propriété `appearance` (tableau 7-8), s'en ajoutent d'autres qui fournissent davantage de précisions :

- éléments de type `window` : `document` (document avec titre ou liste de dossiers), `desktop` (ensemble pouvant contenir plusieurs fenêtres), `workspace` (projet avec un titre et ses fenêtres), `tooltip` (infobulle), `dialog` (boîte de message, appelée aussi *message-box*) ;
- éléments de type `button` : `push-button` (bouton 3D), `hyperlink` (lien hypertexte), `radio-button` (bouton radio), `checkbox` (case à cocher) ;
- éléments de type `menu` : `menubar` (barre de menus), `pull-down-menu` (menu déroulant), `pop-up-menu` (affichage permanent du sous-menu choisi), `list-menu` (liste d'options cumulables), `radio-group` (groupe de boutons radio), `checkbox-group` (groupe de cases à cocher), `outline-tree` (menu arborescent avec des symboles pour l'affichage des sous-menus, par exemple des flèches).

Notez que cette propriété `appearance` est encore à l'état de brouillon et qu'avant de pouvoir l'employer, il sera préférable d'attendre qu'elle soit finalisée et bien reconnue par les navigateurs.

Marqueur de ligne tronquée

Lorsqu'un texte dépasse la dimension prévue dans le bloc qui le contient, il est coupé brusquement si les dépassemens de bloc sont masqués (donc si la propriété `overflow` a une valeur différente de celle par défaut qui est `visible`). Cela se produit notamment lorsqu'un paragraphe doit tenir sur une seule ligne

(propriété `white-space: nowrap;`). La propriété `textoverflow` permet d'adoucir cette coupure, en affichant automatiquement dans ce cas des points de suspension ou d'autres caractères.

TABLEAU 7–9 Propriété text-overflow

Propriété	<code>text-overflow</code>
Exemple	<pre>.titre { width: 250px; overflow: hidden; white-space: nowrap; text-overflow: ellipsis; }</pre>
Valeurs possibles	<code>clip</code> (valeur par défaut) : coupure brute du texte, <code>ellipsis</code> : affichage de caractères indiquant la coupure du texte (points de suspension par défaut) avec coupure autorisée en milieu de mot, ou bien d'une chaîne de caractères placée entre guillemets, qui sera affichée à la place des points de suspension classiques.
Héritage	<code>Non</code>

Coupure des mots trop longs

Lorsqu'un mot particulier possède une longueur plus grande que la largeur du bloc qui le contient, il déborde du cadre prévu s'il ne contient pas de tiret permettant une coupure. La propriété `word-break` autorise la coupure des mots trop longs, pour éviter ce cas de figure inesthétique.

TABLEAU 7–10 Propriété word-break

Propriété	<code>word-break</code>
Exemples	<pre>body { word-break: break-all; } .expressions { word-break: hyphenate; }</pre>
Valeurs possibles	<code>normal</code> (valeur par défaut) : coupure des mots autorisée uniquement sur les espaces et les tirets <code>break-all</code> : coupure possible à n'importe quel endroit, pour les mots qui dépassent la largeur de leur conteneur <code>keep-all</code> : proche de la valeur <code>normal</code> , ne permet que la coupure de mots sur les caractères clairement définis (en français, les tirets) et ne coupe pas les mots en chinois, japonais ou coréen
Héritage	Propriété héritée. Utiliser <code>normal</code> pour retrouver la valeur par

défaut.

Des bordures plus variées

La mise en forme des blocs de texte n'a pas été oubliée par les CSS 3 : bordures à partir d'images, ombres et arrondis sont de la partie !

Des images en guise de bordures

La nouvelle propriété `border-image` apporte un peu d'originalité dans nos encadrements, en utilisant une image de notre choix dont elle répétera des parties pour créer l'entourage.

L'image modèle est divisée en neuf zones (figures 7-6 et 7-7) en fonction des mesures fournies pour le découpage à partir de chaque bord. Ces éléments d'image serviront à encadrer le bloc concerné de la façon suivante :

- les coins sont reportés dans ceux du bloc à encadrer et si nécessaire ajustés à leurs dimensions ;
- les zones intermédiaires (bordures horizontales et verticales) seront répétées ou étirées au choix, pour s'adapter aux dimensions du bloc à encadrer ;
- sur demande, le motif de la zone centrale pourra être utilisé pour remplir l'intérieur du bloc ; il sera alors répété ou étiré suivant les indications des bordures horizontales et verticales ;
- si l'épaisseur de bordure du bloc à encadrer est inférieure à celle prélevée dans l'image de référence, il peut y avoir ou non un débordement, au choix, et sur une distance à préciser.

Dans l'exemple du tableau 7-11, une zone de 30 pixels sur les bords de l'image modèle *moncadre.png* est utilisée pour encadrer chaque titre `h1`, où un contour de 20 pixels est réservé à la bordure, avec un dépassement autorisé de 10 pixels à l'extérieur du bloc. L'addition des deux nous donne bien la place d'afficher la bande de 30 pixels prélevée sur le modèle. Les bordures du haut et du bas sont répétées (avec un nombre entier de motifs, légèrement élargis si nécessaire pour obtenir un compte rond), tandis que celles de gauche et de droite sont étirées.

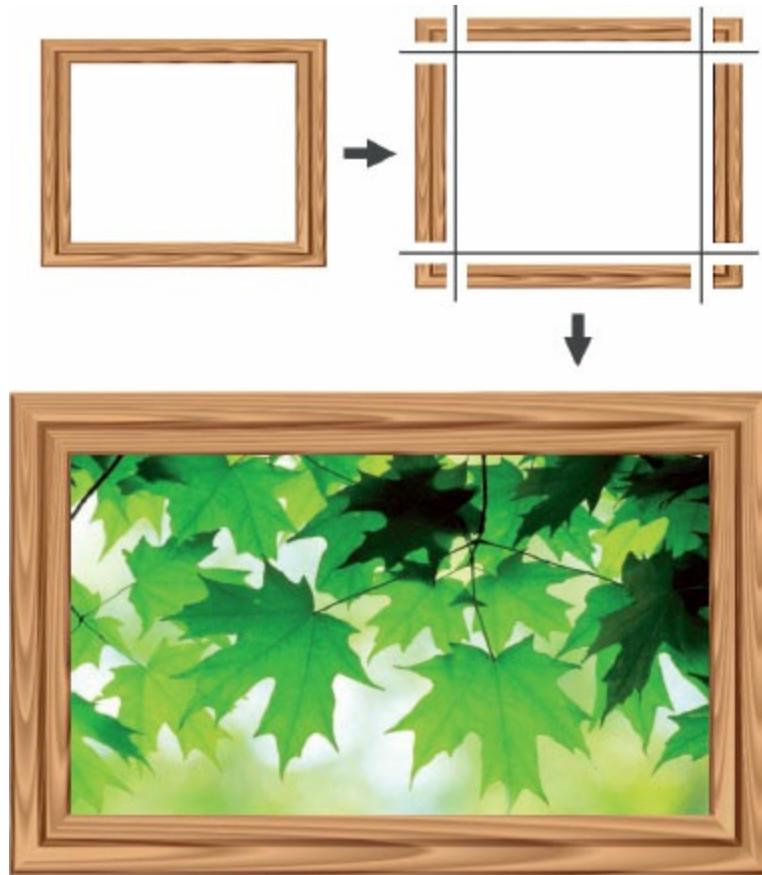


FIGURE 7–6 Encadrement d'un bloc à partir d'une image modèle, d'après la page <http://codeitdown.com/css/border-image-slice/> de Jose Vargas

TABLEAU 7–11 Propriété border-image

Propriété	<code>border-image</code>
Exemple	<code>h1 { border-image: url(moncadre.png) 30 / 20px / 10px round stretch; }</code>
Valeurs possibles	<p>Soit <code>none</code> (valeur par défaut - pas de bordure image), soit plusieurs paramètres qui sont eux-mêmes des propriétés :</p> <ol style="list-style-type: none"> 1) <code>border-image-source</code> : définition par la syntaxe <code>url(..)</code> du fichier contenant l'<i>image d'un cadre</i>, valeur <code>none</code> par défaut 2) <code>border-image-slice</code> (facultatif) : <i>un à quatre nombre(s) ou pourcentage(s)</i> pour définir la partie de l'image à conserver à partir de ses bords (figure 7-7), suivant le principe et l'ordre habituels (une à quatre valeurs, haut droite bas gauche). Ces dimensions sont données soit par des pourcentages de la hauteur ou de la largeur de l'image (la valeur par défaut étant

`100%`), soit par des nombres seuls sans unité, qui correspondent à des pixels (si l'image est définie point par point) ou à des coordonnées de vecteurs (pour une image vectorisée).

Si le mot `fill` est ajouté à la suite, séparé par un espace, la partie centrale de l'image (partie 5 sur la figure 7-7) sera utilisée pour remplir le milieu de l'élément encadré. Ce motif sera répété ou étiré dans chaque sens, suivant les choix effectués pour les bordures horizontales et verticales avec `border-image-repeat` (voir plus loin).

3) Facultative et précédée d'une barre oblique (`/`), la propriété `border-image-width` correspond à `border-width` et redéfinit l'épaisseur réservée pour la bordure, autour du bloc à encadrer. Il s'agit d'une à quatre valeurs (haut droite bas gauche), exprimées en pourcentage ou en pixels avec l'unité `px`, ou bien avec la valeur `auto` pour une adaptation automatique à la partie d'image utilisée (dans le cas où l'image n'est pas vectorielle).

La valeur par défaut est `1px`, appliquée seulement si la propriété `border-width` n'a pas non plus été définie.

4) Facultative et précédée d'une deuxième barre oblique (`/`), la propriété `border-image-outset` sert à définir les dimensions des débordements autorisés sur chaque bord, si la partie de l'image utilisée est supérieure à la largeur de la bordure (une à quatre valeurs, haut droite bas gauche, en pourcentage ou en pixels avec l'unité `px`). La valeur par défaut est `0`, ce qui signifie qu'il ne doit pas y avoir de débordement.

5) `border-image-repeat` est utilisée pour calibrer la partie de l'image utilisée : `stretch` pour l'étirer, `repeat` pour une répétition simple (valeur par défaut, le dernier motif peut être tronqué), `round` pour répéter un nombre entier de fois cette partie d'image (éventuellement agrandie pour une adaptation à la dimension du bloc), `space` pour un même nombre entier de motifs sans déformation (l'ajustement s'effectuant à l'aide d'espaces entre les motifs répétés).

Si deux valeurs sont indiquées, la première s'applique aux bordures horizontales (haut et bas), la deuxième aux bordures verticales (gauche et droite).

Héritage

Non

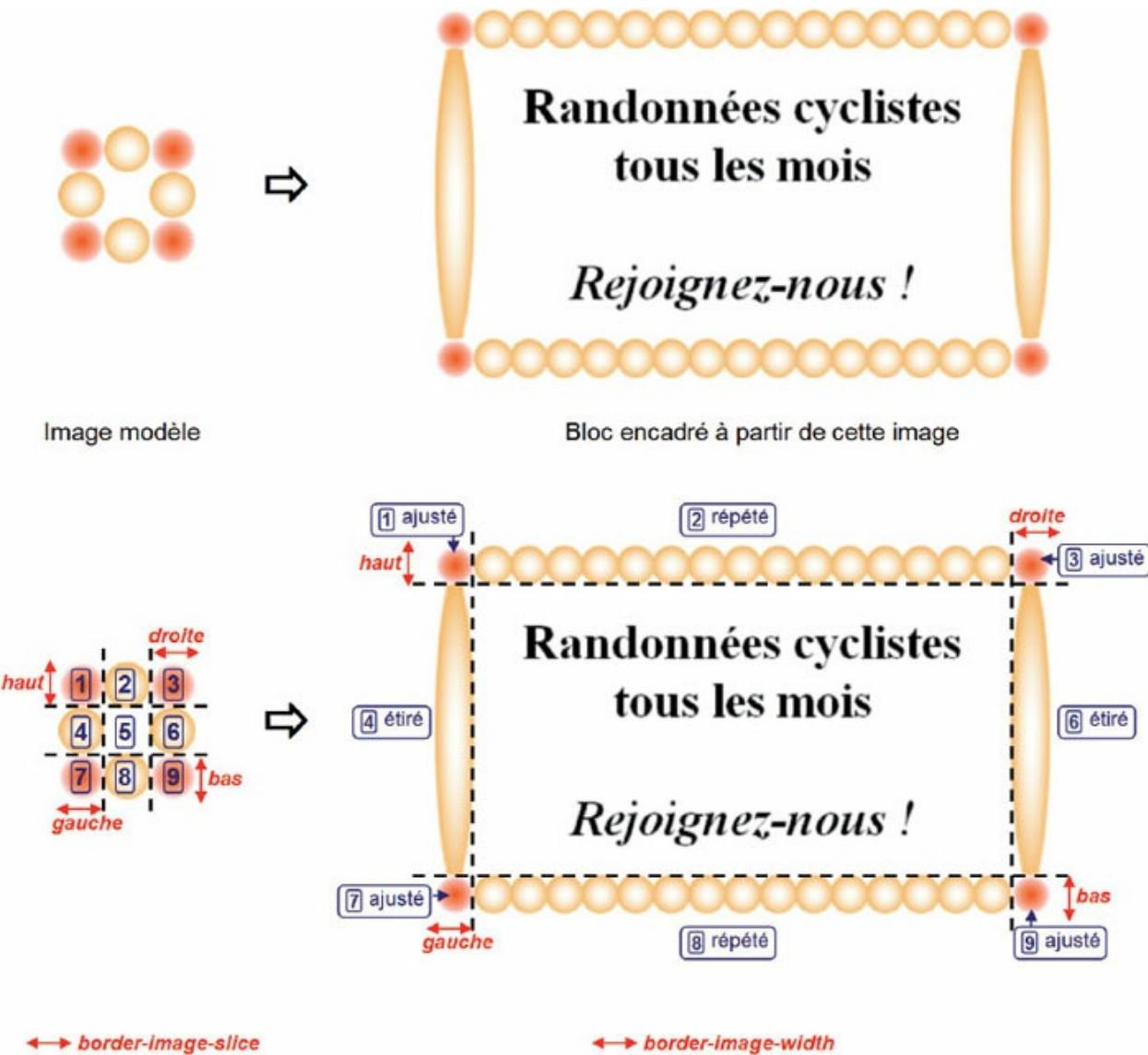


FIGURE 7-7 Encadrement d'un bloc à partir d'une image (extrait du site <https://www.w3.org>) : le motif est divisé en neuf parties ; les coins sont ajustés à ceux du bloc, les côtés sont répétés ou étirés.

Notez qu'il vaut mieux utiliser la propriété raccourcie `border-image`, car pour l'instant elle est mieux reconnue par les navigateurs que chacune des propriétés séparées qu'elle regroupe : `border-image-source`, `border-image-slice`, `border-image-width`, `border-image-outset` et `border-image-repeat`.

Pour que cette propriété fonctionne correctement, il ne faudra pas oublier de donner une épaisseur suffisante aux bordures, soit à l'aide de la propriété `border-width` ou de ses déclinaisons spécifiques à chaque côté, soit en

renseignant le paramètre `border-image-width` de la propriété `border-image`, avec une à quatre valeurs numériques ou le mot-clé `auto`.

Ces fameux coins arrondis !

Pour obtenir des encadrés aux coins ronds, il fallait autrefois imbriquer plusieurs blocs et utiliser des morceaux d'image qui représentaient les coins. Le bloc rectangulaire à encadrer se trouvait au milieu, avec une couleur de fond identique aux coins.

L'arrivée de la propriété `border-radius`, qui arrondit n'importe quel bloc en une seule ligne, a bien changé la vie des développeurs !

Concours de photos

FIGURE 7–8 La couleur de fond suit l'arrondi, de même que la bordure, qu'il suffira de définir avec les propriétés habituelles.

TABLEAU 7–12 Propriété border-radius

Propriété	<code>border-radius</code>
Exemples	<pre>p.annonce { border-radius: 30px / 10px; background-color: yellow; } div.pub { border-radius: 5px 20px; } h2 { border-radius: 25px 15px / 50px 15px; }</pre>
Valeurs possibles	Soit <code>none</code> (valeur par défaut, pas d'arrondi), soit une seule valeur, soit deux valeurs de longueur séparées par une barre oblique (/) : rayon vertical et rayon horizontal . Si une seule valeur est fournie, ces deux rayons sont égaux. Pour détailler séparément les arrondis des différents coins, du coin supérieur gauche au coin inférieur gauche en tournant dans le sens horaire, il suffit d'écrire les quatre valeurs d'arrondis, ou bien les quatre valeurs de rayons verticaux, la barre oblique (/), puis les quatre valeurs de rayons horizontaux.
Héritage	<code>Non</code>

L'arrondi se décline pour chaque coin du rectangle à arrondir, qui pourra être traité séparément grâce à des propriétés distinctes utilisant la même syntaxe ; `border-radius` est un raccourci de ces propriétés qui sont dans l'ordre :

- border-top-left-radius : coin supérieur gauche ;
- border-top-right-radius : coin supérieur droit ;
- border-bottom-right-radius : coin inférieur droit ;
- border-bottom-left-radius : coin inférieur gauche.

Quelques exemples

```
border-radius: 10px 20px 30px 40px / 5px 15px 25px 35px; ①
border-radius: 10px 20px 30px 40px; ②
border-radius: 25px 15px / 55px 15px; ③
```

Dans l'exemple ①, tous les coins sont gérés séparément, dans l'ordre indiqué précédemment : les rayons verticaux des quatre coins sont définis, puis après la barre oblique (/) sont indiqués les quatre rayons horizontaux associés. L'exemple ② correspond à la définition de ces quatre coins, mais avec des rayons verticaux et horizontaux identiques.

Dans l'exemple ③, il n'y a que deux valeurs pour chaque type de rayon, horizontal et vertical. La première correspond dans chaque cas aux coins supérieur gauche et inférieur droit (border-top-left-radius et border-bottom-right-radius), tandis que la deuxième s'applique aux coins supérieur droit et inférieur gauche (border-top-right-radius et border-bottom-left-radius).

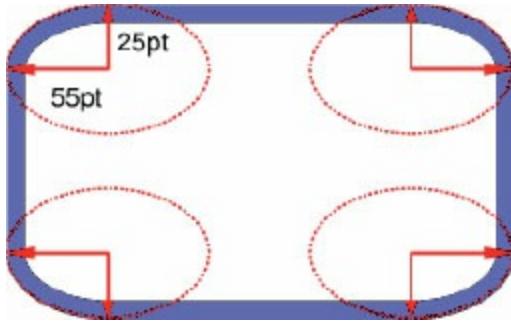


FIGURE 7–9 Les rayons qu'il faut fournir à la propriété border-radius sont en fait le demi-axe vertical et le demi-axe horizontal de l'ellipse qui définit l'arrondi (extrait du site www.w3.org).

CAS PARTICULIER Bordures arrondies pour un tableau

Dans un tableau, la suppression des espacements entre bordures contiguës avec la règle border-collapse: collapse; entre en conflit avec la propriété border-radius et l'empêche d'arrondir les coins.

Ce défaut est expliqué sur le site <http://yidille.free.fr/>, qui propose une

astuce consistant à définir des bordures séparées avec un espacement nul. Les espacements entre les cellules sont donc bien supprimés, mais les coins du tableau peuvent alors s'arrondir, comme dans cet exemple :

```
table { border-collapse: separate; border-spacing: 0; border-radius: 10px; }
```

Cela suppose que les lignes `tr` et les cellules `td` ou `th` ne soient pas encadrées, sinon leurs bordures rectangulaires viendraient se superposer aux coins arrondis du tableau.

Pour quadriller un tableau aux coins arrondis, une solution consiste à attribuer des classes à la première et à la dernière lignes du tableau, et dans ces lignes à la première et à la dernière cellules, de façon à gérer dans chaque cas l'arrondi des coins. Une autre méthode plus légère fait appel aux pseudo-classes `:first-child` et `:last-child` (premier et dernier éléments du type donné), ce qui évite l'ajout de classes sur chaque ligne.

Des ombres pour nos boîtes

Encore un effet bien utile qui, avant les CSS 3, n'était possible que par la création d'une image ou par une superposition de blocs : l'effet d'ombre apporte une touche esthétique autour des cadres à faire ressortir.

La propriété `box-shadow` est bien pratique, puisqu'elle va créer cette ombre automatiquement. Les valeurs à lui attribuer sont les mêmes que celles de la propriété `text-shadow` vue précédemment (ombrage des caractères) : le décalage horizontal et vertical à créer entre le cadre et son ombre, puis l'étendue du flou de l'ombrage et sa couleur.

Écouter les autres, c'est
encore la meilleure façon
d'entendre ce qu'ils disent.

Pierre Dac

FIGURE 7–10 Quand les CSS 3 créent automatiquement des ombres, l'avenir s'éclaire pour les concepteurs de pages web !

TABLEAU 7–13 Propriété box-shadow

Propriété	<code>box-shadow</code>
Exemples	<pre>ul { box-shadow: 2px 2px 2px green; } h1 { box-shadow: 3px -3px 3px violet, 3px -6px 3px blue; }</pre>
Valeurs possibles	<p>none (valeur par défaut, aucun ombrage) ou deux à quatre dimensions suivies de la couleur de l'ombre et d'une option de type d'ombre</p> <ul style="list-style-type: none">• 1^{re} valeur : décalage en x de l'ombre (positif vers la droite, négatif vers la gauche)• 2^e valeur : décalage en y de l'ombre (positif vers le bas, négatif vers le haut)• 3^e valeur : étendue du flou, aucun flou si cette valeur est absente ou égale à 0

- 4^e valeur : **étendue de l'ombre**, qui définit l'épaisseur d'ombre de couleur unie avant l'application du flou, dimension comptée de chaque côté après le décalage et égale à 0 si absente
- couleur : **couleur de l'ombre** (code couleur ou mot clé)
- le mot-clé optionnel `inset` qui affiche l'ombre à l'intérieur du bloc (à partir du haut et de la gauche), son absence correspondant à l'ombrage habituel situé à l'extérieur

Plusieurs ombres peuvent être définies à la suite dans une même propriété `box-shadow`, séparées par des virgules : dans ce cas, les différents effets d'ombre se superposent. Les décalages en x et y sont des dimensions positives ou négatives. La couleur peut éventuellement être placée en première position.

Héritage

Non

Espacement pour encadrement double

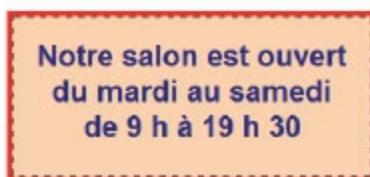
La propriété `outline`, définie depuis la norme CSS 2, permet de superposer une bordure à un élément, sans augmenter ses dimensions. Son utilisation restait assez confidentielle, car elle était proche de la propriété `border`.

La nouvelle propriété CSS 3 `outline-offset` associe ces deux premières propriétés de manière intéressante, car elle définit un espacement entre la bordure créée par `border` et celle affichée par `outline`. Ainsi, il devient possible de créer un double encadrement personnalisé, dans lequel les deux bordures peuvent avoir des épaisseurs, des couleurs et des styles différents.

TABLEAU 7–14 Propriété `outline-offset`

Propriété	<code>outline-offset</code>
Exemple	<pre>div.titre { border: 2px dashed dimgray; outline: 4px solid red; outline-offset: 5px; }</pre>
Valeurs possibles	Dimension indiquant l'espacement entre la bordure créée par <code>border</code> et celle affichée par <code>outline</code> (valeur 0 par défaut)
Héritage	<i>Non</i>

Sans `outline-offset`



Avec `outline-offset`



FIGURE 7–11 Utilisation de la propriété `outline-offset`, combinée à `outline` et `border` pour personnaliser un double encadrement.

Dimensions des blocs

Dimensions globales des blocs

Les dimensions totales d'un bloc, d'une bordure à celle opposée, peuvent varier. En effet, lorsque la largeur `width` et la hauteur `height` d'un bloc sont définies, il faut ajouter les marges et l'épaisseur de la bordure pour savoir quelle sera sa taille finale. Par exemple, toute modification de la marge interne d'un bloc affecte ses dimensions finales.

C'est pourquoi la propriété `box-sizing` propose de fixer la largeur et la hauteur tout compris du bloc et de les définir avec `width` et `height`, les marges internes et les bordures étant alors déduites de ces dimensions pour calculer la partie destinée au contenu. Dans ce cas, une modification de marge interne, par exemple, ne change pas la taille du bloc : c'est l'espace réservé au contenu qui varie.

TABLEAU 7-15 Propriété `box-sizing`

Propriété	<code>box-sizing</code>
Exemples	<pre>div.chapeau { box-sizing: border-box; width: 500px; height: 250px; } p { box-sizing: content-box; }</pre>
Valeurs possibles	<p><code>content-box</code> (valeur par défaut) : les dimensions <code>width</code> et <code>height</code> du bloc ne comprennent pas les marges internes <code>padding</code> ni l'épaisseur des bordures <code>border-width</code>.</p> <p><code>padding-box</code> : les valeurs fournies par les propriétés <code>width</code> et <code>height</code> indiquent la dimension du bloc en incluant les marges internes <code>padding</code>, mais pas les bordures <code>border-width</code>, ni les marges externes <code>margin</code>.</p> <p><code>border-box</code> : les valeurs fournies par les propriétés <code>width</code> et <code>height</code> indiquent la dimension totale du bloc jusqu'aux bordures incluses, incluant <code>padding</code> et <code>border-width</code>, mais pas les marges externes <code>margin</code>.</p>
Héritage	<code>Non</code>

| ASTUCE Essayer `box-sizing`, c'est l'adopter !

La règle `box-sizing: border-box;` est de plus en plus utilisée, car elle nous facilite grandement la tâche. En particulier lorsqu'il s'agit de partager une largeur entre plusieurs blocs, leurs dimensions peuvent être données en pourcentages : il suffit que le total soit égal à 100 %.

Il n'y a plus à se poser de question, par exemple, sur le pourcentage à réservé à des bordures exprimées en pixels.

Si les marges internes de ces blocs viennent à être modifiées, il ne sera pas davantage nécessaire de revenir sur leurs dimensions globales, en largeur comme en hauteur.

Blocs de dimensions modifiables

La propriété `resize` autorise le redimensionnement avec la souris d'un bloc, lorsque sa propriété `overflow` a été fixée (gestion des débordements du texte).

En pratique, le navigateur affiche un coin strié en bas à droite du bloc, sur lequel le curseur de la souris peut agir pour modifier les dimensions du rectangle, comme le montre la figure 7-12. Il est possible d'autoriser le redimensionnement en largeur, en hauteur ou les deux.

TABLEAU 7–16 Propriété `resize`

Propriété	<code>resize</code>
Exemple	<code>div { border: solid 1px; overflow: auto; resize: both; }</code>
Valeurs possibles	<code>none</code> (dimensions fixes, valeur par défaut), <code>both</code> (largeur et hauteur modifiables), <code>horizontal</code> (largeur seule modifiable), <code>vertical</code> (hauteur seule modifiable)
Héritage	<code>Non</code>

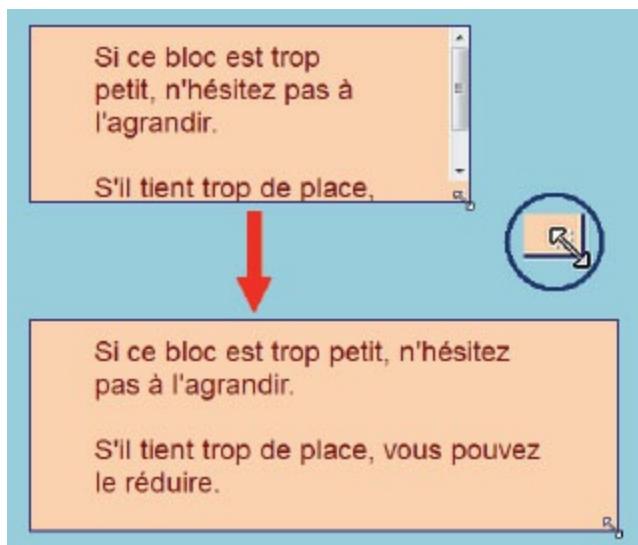


FIGURE 7–12 Modification des dimensions d'un bloc avec la souris, grâce à la propriété `resize`

Couleurs et images de fond

Certaines nouvelles propriétés CSS 3 apportent quelques fonctions supplémentaires pour améliorer le rendu des couleurs et l'utilisation des images de fond.

Plusieurs images d'arrière-plan

Tout d'abord, il est possible en CSS 3 de déclarer plusieurs images de fond à l'aide de la propriété `background-image` ou du raccourci `background`, comme dans ces deux règles qui sont équivalentes :

```
background-image: url(logo.png), url(texture.jpg);  
background: url(logo.png), url(texture.jpg);
```

La première image déclarée se trouvera « au premier plan de cet arrièreplan », la deuxième en dessous, etc.

PRÉCISION Conséquence de l'utilisation de plusieurs images de fond

Toutes les propriétés associées aux images de fond pourront voir leurs valeurs répétées autant de fois qu'il y a d'images ainsi déclarées, ces valeurs ou groupes de valeurs étant séparé(e)s par des virgules. Par exemple :

```
p {  
    background-image: url(logo.png), url(texture.jpg);  
    background-repeat: no-repeat, repeat;  
    background-position: center top, left top;  
}
```

L'ordre des valeurs ou groupes de valeurs correspond à l'ordre de déclaration des images de fond. Dans cet exemple, notons que les trois propriétés utilisées peuvent être résumées en une seule, grâce au raccourci `background` :

```
p { background: url(logo.png) no-repeat center top,  
    url(texture.jpg) repeat left top; }
```

Placement et étendue des images d'arrière-plan

Trois nouvelles propriétés sont associées à l'arrière-plan d'un bloc.

- `background-origin` sert à préciser à partir de quel point la position de cette image de fond sera prise en compte, pour que le décalage fourni par la propriété `background-position` soit décompté à partir de cette limite : bordure, marge interne ou début du contenu.
- `background-clip` indique si l'arrière-plan doit se poursuivre ou non sous l'espace réservé à la bordure du bloc.
- `background-size` permet de modifier les dimensions de l'image utilisée comme arrière-plan.

TABLEAU 7–17 Propriété `background-origin`

Propriété	<code>background-origin</code>
Exemple	<pre>div.titre { background-image: url(fond.png); padding: 20px; border: 2px solid; background-origin: content-box; }</pre>
Valeurs possibles	<p><code>padding-box</code> : l'arrière-plan est placé par rapport au début de la marge interne du bloc (valeur par défaut).</p> <p><code>border-box</code> : il est positionné par rapport aux limites externes de la bordure.</p> <p><code>content-box</code> : l'arrière-plan se place par rapport au début du contenu du bloc, après la marge interne.</p> <p>La base de placement ainsi fournie permettra de centrer l'arrièreplan ou de l'aligner sur un côté, avec <code>background-position</code>.</p>
Héritage	<code>Non</code>

Tant pour `background-origin` que pour `background-clip`, il est possible de fournir plusieurs valeurs séparées par des virgules : elles correspondent respectivement aux différentes images de fond, lorsque plusieurs ont été définies pour un même bloc.

TABLEAU 7–18 Propriété `background-clip`

Propriété	<code>background-clip</code>
------------------	------------------------------

Exemple	<pre>p.note { background-image: url(texture.png); padding: 10px; border: 3px solid gray; background-clip: padding-box; }</pre>
Valeurs possibles	<p><code>border-box</code> : l'arrière-plan se prolonge jusque sous la bordure (valeur par défaut).</p> <p><code>padding-box</code> : l'arrière-plan s'étend sous la marge interne, mais pas sous la bordure.</p> <p><code>content-box</code> : l'arrière-plan n'apparaît que sous le contenu, mais pas sous les marges internes ni sous la bordure.</p>
Héritage	<i>Non</i>

TABLEAU 7–19 Propriété `background-size`

Propriété	<code>background-size</code>
Exemples	<pre>#pub { background-image: url(plage.jpg); background-size: 100px 300px; } #haut { background-image: url(nature.png); background-size: cover; }</pre>
Valeurs possibles	<p><i>largeur</i> de l'image de fond, en unité de mesure ou en % de la largeur du bloc (hauteur calculée automatiquement pour ne pas déformer l'image)</p> <p><i>largeur et hauteur</i> de l'image, séparées par un espace, en unités de mesure ou en %, respectivement de la largeur et de la hauteur du bloc (par exemple 100% 100% pour couvrir tout le bloc), ce qui peut déformer l'image</p> <p><code>auto</code> : dimensions réelles de l'image de fond (valeur par défaut)</p> <p><code>contain</code> : dimensions calculées pour que l'image atteigne juste le bord du bloc, sans le dépasser (l'image conserve ses proportions, donc elle ne couvrira pas nécessairement tout le bloc)</p> <p><code>cover</code> : dimensions calculées pour que l'image recouvre entièrement le bloc (l'image conservant ses proportions, il se peut qu'elle ne soit pas visible entièrement)</p>
Héritage	<i>Non</i>

Fixation de l'image d'arrière-plan

La propriété connue `background-attachment` permet d'indiquer si l'image de fond reste fixe ou non, lorsque l'utilisateur utilise une barre de défilement. Elle se voit attribuer une nouvelle valeur `local`, en plus des valeurs `scroll` (valeur par défaut) et `fixed`.

background-attachment: `local` ;

Le tableau suivant compare l'effet de la propriété `background-attachment` attribuée à un bloc qui possède une image de fond, lorsqu'il se produit un défilement, soit dans le bloc lui-même (barres de défilement interne à ce bloc), soit dans la page entière.

TABLEAU 7–20 Effet de la propriété `background-attachment` sur un bloc

Valeur de la propriété	Défilement dans le bloc (barres de défilement internes)	Défilement de la page
<code>scroll</code>	Le contenu du bloc glisse sur l'image, qui reste fixe.	L'image se déplace avec le bloc, restant fixe par rapport à son contenu.
<code>fixed</code>	Le contenu du bloc glisse sur l'image, qui reste fixe.	Le bloc se déplace, mais son contenu glisse sur l'image, car elle est attachée à la page.
<code>local</code>	L'image et son contenu se déplacent ensemble.	L'image et le contenu du bloc se déplacent ensemble.

Les barres de défilement internes apparaissent dans un bloc lorsque sa propriété `overflow` vaut `scroll` (barre de défilement permanente) ou `auto` (barre de défilement si nécessaire).

À NOTER Propriétés `background-...` et images de fond multiples

À toutes les propriétés du type `background-...` peuvent être attribuées plusieurs valeurs séparées par des virgules : ces paramètres correspondent respectivement aux images de fond successives qui ont été définies pour un même bloc.

Dégradés de couleurs

Pour les propriétés qui acceptent une image de fond, comme `background`, `background-image` ou `list-style-image`, il est possible de remplacer celle-ci par un dégradé de couleurs, linéaire ou radial, créé en CSS.

Ce fond est créé à l'aide des propriétés `linear-gradient(...)` et `radial-gradient(...)` pour un dégradé simple, `repeating-lineargradient(...)` et `repeating-radial-gradient(...)` pour des dégradés répétés.

Dégradé linéaire

Il existe différentes façons de définir un dégradé linéaire, comme le montrent les exemples suivants :

```
p { background-image: linear-gradient(white, blue); }
div { background: linear-gradient(to top right, yellow, green); }
body { background: linear-gradient(45deg, silver, gray); }
```

Chaque dégradé s'effectue de la première couleur indiquée vers la seconde. Les couleurs sont indiquées par leurs noms ou un code de couleur.

Pour donner la direction du dégradé, il faudra indiquer le mot `to` (c'est-à-dire « vers » en anglais) suivi de la localisation verticale, `top` ou `bottom`, puis horizontale, `left` ou `right`. Un dégradé vertical utilisera la valeur `to top` ou `to bottom` ; un dégradé horizontal s'écrira avec `to left` ou `to right`. Par défaut, le dégradé s'effectue du haut vers le bas, ce qui correspond à la direction `to bottom` utilisée seule.

Une valeur d'angle peut remplacer ces indications de direction : l'angle `0deg` correspond à l'aiguille d'un réveil désignant midi, les valeurs positives d'angles représentant une rotation dans le sens des aiguilles d'une montre. L'angle `0deg` est donc l'équivalent de `to top`.

Un dégradé peut comprendre plusieurs couleurs, comme ci-dessous :

```
div#menu { background: linear-gradient(white, blue 30%, green) }
```

Dans ce cas, le dégradé (de haut en bas par défaut) part du blanc, arrive au bleu à 30 % de la hauteur, puis passe du bleu au vert sur les 70 % de hauteur qui restent.

Les couleurs peuvent être définies comme d'habitude par leur nom, par un code hexadécimal précédé d'un dièse (#), un code *rgb* ou encore un code *rgba* pour obtenir un effet de transparence.

IMPORTANT Préfixes et syntaxes pour les dégradés linéaires

1) Pour utiliser un préfixe tel que `-moz-` ou `-webkit-` dans une propriété définissant l'image d'un dégradé, il faut le placer avant la fonction `linear-gradient` et répéter le nom de la propriété pour chaque préfixe. Il en sera de même avec les fonctions cousines `radial-gradient`, `repeating-linear-gradient` et `repeating-radial-gradient`.

Par exemple, nous pourrons écrire :

```
background: -moz-linear-gradient(white, blue);
background: -webkit-linear-gradient(white, blue);
background: -ms-linear-gradient(white, blue);
background: linear-gradient(white, blue);
```

2) Pour la fonction `linear-gradient` utilisée avec un angle, les premières spécifications partaient de la direction **Est** (soit l'aiguille d'une montre sur trois heures) pour l'angle `0deg`, avec un sens positif **antihoraire**. Depuis, la norme a changé sur ces deux aspects (point de départ des angles à midi et sens horaire), mais les syntaxes avec préfixe utilisent cet ancien fonctionnement, qu'il faudra donc prendre en compte avec `-moz-`, `-webkit-` ou `-ms-`.

Comme d'habitude, il ne faudra pas oublier d'écrire *à la fin* la propriété sans préfixe et avec la nouvelle syntaxe, comme dans cet exemple :

```
background: -moz-linear-gradient(120deg, red, blue);
background: -webkit-linear-gradient(120deg\, red, blue);
background: -ms-linear-gradient(120deg, red, blue);
background: linear-gradient(-30deg, red, blue);
```

Dégradé radial

La propriété `radial-gradient` permet de créer un dégradé radial, partant d'un point central avec une couleur pour atteindre l'autre couleur, ce changement s'effectuant suivant une forme elliptique (valeur `ellipse` par défaut, si aucune dimension n'est fournie) ou circulaire (avec la valeur `circle`).

Comme pour le dégradé linéaire, il est possible d'utiliser plusieurs couleurs pour obtenir un dégradé multicolore. Voici quelques exemples de cette

fonction :

```
div { background: radial-gradient(white, blue); }
body { background: radial-gradient(circle, yellow, green); }
p { background: radial-gradient(circle, blue, white, red); }
```

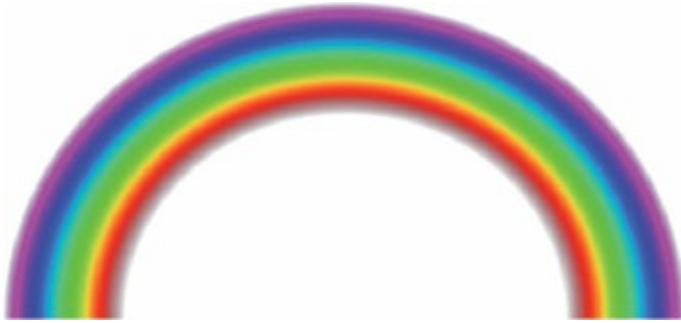


FIGURE 7–13 Un arc-en-ciel créé en CSS 3, à l'aide de la fonction de dégradé radialgradient et en utilisant plusieurs couleurs

La forme complète d'un dégradé radial utilise quatre arguments séparés par des virgules. Il s'écrira `radial-gradient(forme et taille at origine du dégradé, couleur initiale, couleur finale)`.

- La **forme** peut être `circle` (cercle) ou `ellipse`, la valeur par défaut étant le cercle si une seule dimension est fournie, une ellipse sinon.
- Séparée par un espace et placée avant ou après la forme, la **taille** qui définit la fin du dégradé est soit pour un cercle *une longueur*, soit pour une ellipse *deux longueurs* ou *deux pourcentages* (des dimensions de la boîte du dégradé), soit dans tous les cas un *mot-clé* qui se rapporte aux limites du bloc : `closest-side` (côté le plus proche), `farthest-side` (côté le plus éloigné), `closest-corner` (coin le plus proche) ou `farthest-corner` (coin le plus éloigné ; c'est la valeur par défaut lorsque ce paramètre facultatif est absent).
- L'**origine** du dégradé se compose du mot `at` (séparé des deux premières valeurs par un espace et non par une virgule), suivi d'un positionnement comme dans la propriété `background-position` : position horizontale donnée par `left`, `center`, `right` ou un nombre (dimension ou pourcentage, 0 % correspondant à `left` et 100 % à `right`), une position verticale fournie par `top`, `center`, `bottom` ou un nombre (0 % correspondant à `top` et 100 % à `bottom`). La valeur par défaut est `at center` (ce qui pourrait s'écrire `50% 50%`) ; c'est donc le centre du bloc qui sert de point de départ si ce paramètre facultatif est omis.

- Séparées par des virgules, les **couleurs** initiale et finale s'écrivent comme d'habitude au moyen de codes ou de noms de couleurs. Il est possible d'écrire plusieurs codes ou noms de couleurs séparés par des virgules : la première est la couleur de début du dégradé, la dernière celle de fin, et les autres sont des couleurs intermédiaires.

Voici un exemple de dégradé radial, dans sa forme plus complexe :

```
background: radial-gradient(circle farthest-side at 80% 60%, red, green, blue);
```

ATTENTION Anciennes normes pour radial-gradient avec préfixe

Comme pour `linear-gradient`, la norme de la propriété `radial-gradient` est à présent stabilisée, mais sous une forme différente de celle envisagée initialement et utilisée par les versions avec préfixes sur les navigateurs. Si nous utilisons ces formes avec leur préfixe, il faudra les écrire avec cette **ancienne syntaxe** :

-préfixe-radial-gradient(origine du dégradé, forme et taille, couleur initiale, couleur finale)

Bien sûr, la version sans préfixe sera à écrire *à la fin* et avec sa nouvelle syntaxe, comme dans cet exemple :

```
background: -moz-radial-gradient(60% 60%, circle closest-side, red, blue);
background: -webkit-radial-gradient(60% 60%, circle closest-side, red, blue);
background: -ms-radial-gradient(60% 60%, circle closest-side, red, blue);
background: radial-gradient(circle closest-side at 60% 60%, red, blue);
```

Un autre type de dégradé, proche du style radial, est à l'étude pour la norme CSS 4 : c'est le dégradé conique `conic-gradient`. Il effectue un dégradé circulaire, suivant le balayage d'un rayon tournant dans son cercle. Le point central semble être en avant par rapport au reste de l'image, créant ainsi une forme de cône. Mais nous sommes déjà suffisamment occupés par la découverte des propriétés CSS 3, laissons ce concept pour une prochaine fois !

Dégradés répétitifs

Des dégradés répétitifs, linéaires ou radiaux peuvent être obtenus grâce aux propriétés `repeating-linear-gradient` et `repeating-radialgradient`. Il faut alors préciser la dimension de chaque couleur à répéter, comme dans les exemples suivants :

```
p { background: repeating-linear-gradient(blue, red 30px, blue 50px); }
div { background: repeating-radial-gradient(red, blue 10px, white 30px); }
```

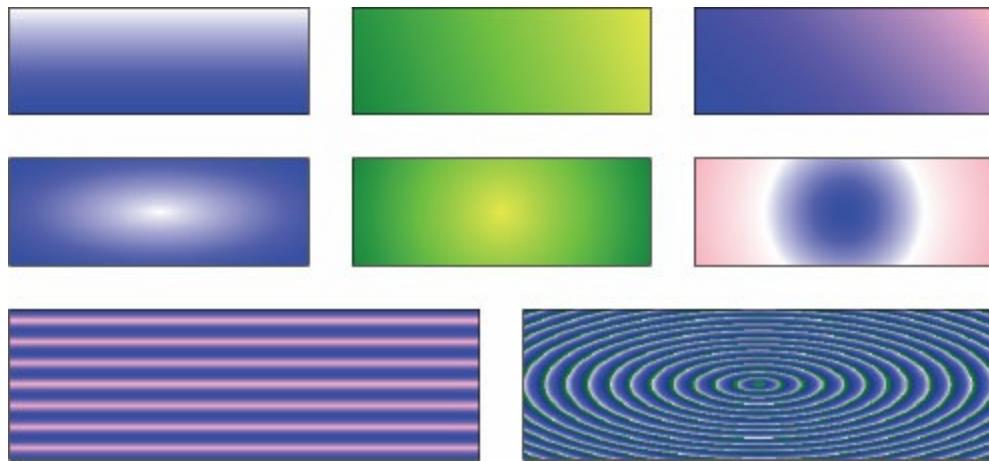


FIGURE 7–14 Quelques exemples de dégradés linéaires (première ligne), radiaux (deuxième ligne) et répétés (troisième ligne), affichés par le navigateur Chrome

Multicolumnage

C'est encore une fonction qui, avant son apparition, a généré bien du code JavaScript : la possibilité de répartir automatiquement le texte d'une page ou d'un bloc en plusieurs colonnes. À présent, un ensemble de propriétés nous permet de gérer aisément le nombre de colonnes, leur largeur et l'espace qui les sépare.

Nombre et largeur des colonnes

Si le nombre de colonnes doit être fixe, il faut utiliser la propriété `column-count`.

TABLEAU 7–21 Propriété column-count

Propriété	<code>column-count</code>
Exemple	<code>div.contenu { column-count: 3; }</code>
Valeurs possibles	Soit un nombre entier de colonnes (nombre de colonnes fixé), soit <code>auto</code> (valeur par défaut) où le nombre de colonnes est fonction d'autres paramètres, comme la largeur <code>column-width</code> .
Héritage	<i>Non</i>

La largeur des colonnes peut être définie en attribuant une dimension à la propriété `column-width`.

TABLEAU 7–22 Propriété column-width

Propriété	<code>column-width</code>
Exemple	<code>div.contenu { column-width: 15rem; }</code>
Valeurs possibles	Soit une dimension indiquant la largeur des colonnes, soit <code>auto</code> (valeur par défaut) où la largeur des colonnes est fonction d'autres paramètres, la dimension de la fenêtre et le nombre de colonnes <code>column-count</code>
Héritage	<i>Non</i>

Il existe un raccourci `columns` qui permet d'écrire en une seule ligne les deux propriétés précédentes.

TABLEAU 7–23 Propriété columns

Propriété	<code>columns</code>
Exemples	<code>div { columns: auto; } div.contenu { columns: 3 15rem; } article { columns: 2; } section.chapeau { columns: 12rem; }</code>

Valeurs possibles	Une ou deux valeur(s) : celle qui n'a pas d'unité indique le nombre de colonnes ou vaut auto, l'autre est une dimension qui donne la largeur des colonnes ou vaut auto. Si un des deux paramètres est absent, il prend la valeur auto.
Héritage	<i>Non</i>

Le Parthénon

Le Parthénon (en grec ancien Παρθενών - de παρθένος, jeune fille) est un édifice sisué sur l'acropole d'Athènes, consacré à la déesse Athéna. C'est le plus connu des monuments grecs classiques.

Il a été construit de -449 à -438 par l'architecte Ictinos et décoré par le sculpteur Phidias, sur l'initiative de Périclès, à l'emplacement de deux édifices précédents.

Le Parthénon primitif (début du VIe siècle av. J.-C.) fut remplacé par le pré-Parthénon, commencé vers -500 et détruit lors du sac de l'Acropole en -480.

Le Parthénon actuel fut construit par Ictinos et Callicratès, Phidias supervisant les sculptures.

Vous partez, non ?

La Grèce, en grec ancien Ἑλλάς (Hellas) et en grec démotique Ελλάδα (Elláda), est située dans l'extrême Sud

des Balkans. Sa capitale est Athènes.

Elle comprend 131 957 km² pour un peu moins de dix millions d'habitants. Elle est bordée par la mer Ionienne à l'ouest et la mer Égée à l'est. Le cinquième de son territoire est constitué de plus de 9 000 îles et îlots dont près de 200 sont habitées.

La plus grande partie de la Grèce est constituée de montagnes, dont la plus haute est le mont Olympe qui culmine à 2 917 mètres.

FIGURE 7–15 Moins typiques que celles du Parthénon, les colonnes générées automatiquement en CSS 3 nous rendront cependant de grands services (texte issu de Wikipédia).

Espacement des colonnes

L’espacement entre les colonnes peut être choisi à l’aide de la propriété column-gap.

TABLEAU 7–24 Propriété column-gap

Propriété	column-gap
Exemples	<pre>div { column-gap: normal; } div.fixe { column-gap: 10px; }</pre>
Valeurs possibles	Dimension positive indiquant l’espacement entre les colonnes ou <code>normal</code> (valeur par défaut) qui vaut en principe <code>1em</code> .
Héritage	<i>Non</i>

Trait de séparation des colonnes

Il est possible de séparer les colonnes par une ligne qui se définira de la même façon que les bordures d'un bloc :

- style : column-rule-style ;
- épaisseur : column-rule-width ;
- couleur : column-rule-color.

Ce trait de séparation se placera au milieu de l'espace qui sépare les colonnes.

TABLEAU 7–25 Propriété column-rule-style

Propriété	column-rule-style
Exemples	<pre>div.fixe { column-rule-style: solid; } div.contrat { column-rule-style: double; }</pre>
Valeurs possibles	Toutes les valeurs acceptées par la propriété border-style : none (valeur par défaut), hidden, solid, dotted, dashed, double, groove, ridge, inset, outset
Héritage	Non

TABLEAU 7–26 Propriété column-rule-width

Propriété	column-rule-width
Exemples	<pre>div.fixe { column-rule-width: 2px; } div.colonnes { column-rule-width: thin; }</pre>
Valeurs possibles	Toutes les valeurs acceptées par la propriété border-width : soit une dimension, soit un des mots-clés thin, medium (valeur par défaut) ou thick
Héritage	Non

TABLEAU 7–27 Propriété column-rule-color

Propriété	column-rule-color
Exemples	<pre>div.fixe { column-rule-color: blue; } div.pub { column-rule-color: #ff0000; }</pre>
Valeurs	Toutes les valeurs de couleur : mot-clé ou code couleur Par

possibles	défaut, c'est la couleur du texte (propriété color).
Héritage	<i>Non</i>

Là encore, un raccourci `column-rule` permet de définir en une seule fois les trois propriétés précédentes.

TABLEAU 7–28 Propriété `column-rule`

Propriété	<code>column-rule</code>
Exemples	<pre>div.fixe { column-rule: solid 2px blue; } div.pub { column-rule: solid #ff0000; }</pre>
Valeurs possibles	Une à trois valeur(s), correspondant à <code>column-rule-style</code> , <code>column-rule-width</code> et <code>column-rule-color</code> Les valeurs absentes sont celles par défaut, associées aux propriétés correspondantes.
Héritage	<i>Non</i>

Le Parthénon	Le Parthénon primitif (début du VIIe siècle av. J.-C.) fut remplacé par le pré-Parthénon, commencé vers -500 et détruit lors du sac de l'Acropole en -480. Le Parthénon actuel fut construit par Ictinos et Callicrates. Phidias supervisait les sculptures.	Sud des Balkans. Sa capitale est Athènes. Elle comprend 131 957 km ² pour un peu moins de dix millions d'habitants. Elle est bordée par la mer Ionienne à l'ouest et la mer Égée à l'est. Le cinquième de son territoire est constitué de plus de 9 000 îles et îlots dont près de 200 sont habitées. La plus grande partie de la Grèce est constituée de montagnes, dont la plus haute est le mont Olympe qui culmine à 2 917 mètres.
---------------------	---	---

FIGURE 7–16 Exemple de multicolonnage avec espacement des colonnes et traits de séparation (texte issu de Wikipédia)

Équilibrage des colonnes

Dans le cas particulier où la hauteur des colonnes est fixe (lorsque la hauteur du bloc conteneur est définie), la question de l'équilibrage des colonnes se pose : faut-il remplir les premières colonnes en partant de la gauche ou répartir le texte entre toutes les colonnes ? La propriété `column-fill` indique s'il faut ou non équilibrer les colonnes.

TABLEAU 7–29 Propriété `column-fill`

Propriété	<code>column-fill</code>
Exemples	<code>div.fixe { column-fill: auto; }</code> <code>div.pub { column-fill: balance; }</code>
Valeurs possibles	<code>auto</code> : remplissage séquentiel des colonnes à partir de la gauche <code>balance</code> : répartition homogène du texte dans les colonnes (valeur par défaut)
Héritage	<i>Non</i>

Ce problème de répartition du texte entre les colonnes se pose plus souvent lors d'une mise en forme pour le média « impression » que pour le média « écran » qui nous préoccupe généralement.

IMPRESSION Coupures de page

Il existe également trois propriétés qui gèrent les coupures de page contenant du texte sur plusieurs colonnes : `break-before`, `break-after` et `break-inside`. Elles ne seront utilisées que pour une mise en forme destinée à l'impression.

Titre sur plusieurs colonnes

Un titre doit pouvoir s'étaler sur l'ensemble des colonnes. Il faut alors lui appliquer la propriété `column-span`.

Lorsqu'un titre est ainsi étalé sur plusieurs colonnes, il crée automatiquement deux zones de multicolonnage séparées : l'une au-dessus du titre et l'autre en dessous, pour le contenu qui précède ce titre et celui qui le suit.

TABLEAU 7–30 Propriété `column-span`

Propriété	<code>column-span</code>
Exemples	<code>h2.grandtitre { column-span: all; } h3.soustitre { column-span: none; }</code>
Valeurs possibles	<code>none</code> : texte sur une seule colonne (valeur par défaut) <code>all</code> : texte réparti sur l'ensemble des colonnes
Héritage	<code>Non</code>

Le Parthénon

Le Parthénon — en grec ancien Παρθενών / Parthenón (de παρθένος, nom féminin, « jeune fille »), spécialement ici « la demeure d'Athéna Parthenos », est un édifice situé sur l'Acropole d'Athènes.

Le Parthénon était consacré à la déesse Athéna, protectrice de la cité et déesse de la guerre et de la sagesse. Il a été conçu tout à la fois pour abriter la statue de la déesse Athéna Parthenos, œuvre de Phidias, et pour abriter l'argent de la cité.

C'est probablement le plus connu des monuments grecs classiques. Il a été construit de -447 à -438 par l'architecte Ictinos et décoré par le sculpteur Phidias, sur l'initiative de Périclès.

Le plus connu des monuments grecs classiques

Edification du bâtiment

Le Parthénon sur l'emplacement de deux autres édifices qui l'ont précédé.

Le premier *Parthénon primitif* a été bâti vers le début du VI^e siècle av. J.-C., puis remplacé par le *pré-Parthénon*, commencé vers 500 av. J.-C. et détruit lors du sac de l'Acropole en 480 av. J.-C.

Le troisième bâtiment, qui est le Parthénon actuel, a pour architectes Ictinos et Callicrates, Phidias supervisant l'ensemble des sculptures.

Vous partez, non ?

La Grèce, en grec ancien Ἑλλάς (Hellás) et en grec démotique Ελλάδα (Elláda), est située dans l'extrême Sud des Balkans. Sa capitale est Athènes.

Elle comprend 131 957 km² pour un peu moins de dix millions d'habitants. Elle est bordée par la mer Ionienne à l'ouest et la mer Égée à l'est, parties de la mer Méditerranée. Le cinquième de son territoire est constitué de plus de 9 000 îles et îlots dont près de 200 sont habitées.

La plus grande partie de la Grèce est constituée de montagnes, dont la plus haute est le mont Olympe qui culmine à 2 917 mètres.

FIGURE 7–17 La propriété `column-span` permet d'étaler un titre sur toutes les colonnes (texte issu de Wikipédia).

Chacune de ces deux parties de contenu se répartit alors sur l'ensemble des colonnes.

Après la répartition du texte en colonnes, nous allons à présent nous intéresser à l'agencement des blocs, lorsqu'il s'agit de les juxtaposer et de gérer leur disposition.

La flexbox pour répartir des blocs

Plusieurs techniques ont été successivement adoptées pour organiser des blocs à l'intérieur d'un conteneur. Aux débuts du Web, la seule solution était d'utiliser des tableaux, parfois imbriqués, d'où un code terriblement compliqué. Depuis, la juxtaposition de blocs s'effectue avec le flottement à gauche ou bien par l'utilisation du type `inline-block`.

Le W3C a réfléchi à la mise en place d'une solution rationnelle qui réponde à des besoins courants : juxtaposer, répartir, aligner et distribuer des blocs de contenu, en largeur ou en hauteur. C'est ainsi qu'est né le principe de la **flexbox**, qui regroupe différentes possibilités de mise en page.

Un bloc conteneur de type flex

La base de cette technique consiste à déclarer que le conteneur est une flexbox. Prenons pour exemple le code HTML suivant, composé d'une section dans laquelle trois blocs `<div>` sont à répartir :

```
| <section>
|   <div class="bloc b1">Bloc 1</div>
|   <div class="bloc b2">Bloc 2</div>
|   <div class="bloc b3">Bloc 3</div>
| </section>
```

Afin de clarifier les explications, ce code présente une section contenant des blocs `<div>`, mais le conteneur pourrait aussi être un bloc `<div>` incluant d'autres blocs `<div>`, ou toute autre balise de type bloc incluant des éléments à répartir.

Pour que notre section devienne une flexbox, il faut lui donner le type `flex`, ce qui ne lui enlève pas pour autant son statut de type bloc. Éventuellement, s'il est nécessaire de transformer cette section en élément en ligne, nous pourrons utiliser le type `inline-flex`.

La règle de style correspondante s'écrit alors :

```
| section { display: flex; }
```

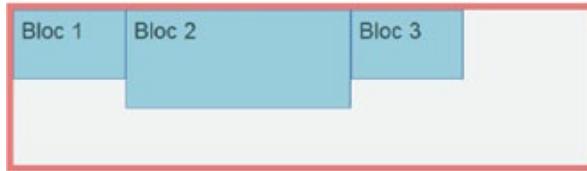
ou dans la seconde configuration, beaucoup moins courante :

```
| section { display: inline-flex; }
```

Dans les deux cas, les éléments directs inclus dans le conteneur deviennent des « items flex ». Dans notre exemple, ce sont les blocs `<div>` inclus dans la section et qui initialement vont s'afficher côte à côte, comme le montre la figure 7-18. Par ailleurs, si la hauteur de ces blocs est automatique, elle s'ajustera à celle du conteneur.

Nous verrons plus loin la propriété `flex-direction` qui nous proposera une variante organisant en colonne les blocs inclus dans la flexbox. Lorsque ces blocs ont une largeur automatique, ils remplissent toute la largeur du conteneur.

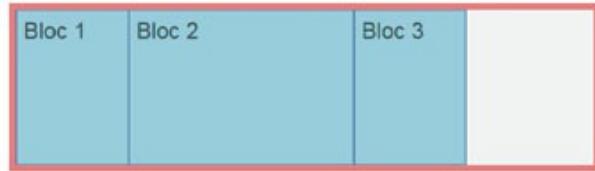
Conteneur ► `display: flex;` ou `display: inline-flex;`



Conteneur ► `display: flex;` ou `display: inline-flex;` (mode colonne)



Conteneur ► `display: flex;` Blocs ► `height: auto;`



Conteneur ► `display: flex;` (mode colonne)
Blocs ► `width: auto;`

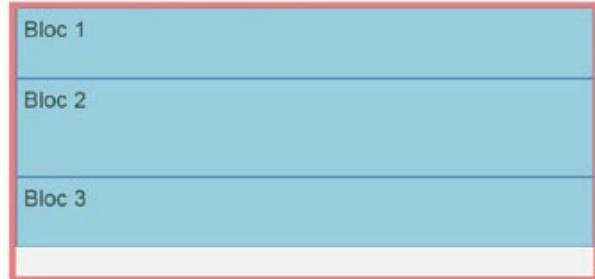


FIGURE 7–18 Dans un conteneur de type flexbox, les blocs sont naturellement juxtaposés en ligne (par défaut) ou en colonne. Si leur hauteur (ou leur largeur en mode colonne) n'est pas définie (valeur `auto` par défaut), les blocs rempliront toute la hauteur du conteneur (ou toute sa largeur en mode colonne).

Si, en mode ligne, les blocs items flex sont par défaut juxtaposés horizontalement, ils ne sont pas pour autant de type `inline-block` : leur position ne sera pas affectée par un centrage du texte ou un multicolonnage, ni par les propriétés d'alignement vertical ou de position flottante.

En revanche, un ensemble de propriétés, spécifiques à la technique flexbox, va nous permettre de modifier la position et la distribution de ces blocs dans leur conteneur. Voici un résumé des choix possibles, pour une suite de blocs de type item flex à l'intérieur d'un conteneur :

- direction (horizontale ou verticale) et sens du placement des blocs ;
- retour à la ligne des blocs sur l'extrême du conteneur ;
- alignement et distribution de la suite de blocs, calés sur un bord du conteneur ou uniformément répartis ;
- placement vertical de l'ensemble des blocs ou d'un bloc en particulier, par rapport à la ligne de base ;
- accroissement ou réduction automatique de chaque bloc, de façon à ce que l'ensemble soit ajusté à la dimension du conteneur ;
- priorité des blocs dans l'ordre d'affichage.

Certaines de ces propriétés s'appliquent au conteneur, lorsqu'elles concernent

l'ensemble des blocs qu'il contient. D'autres sont définies pour chacun de ces blocs items flex, leur attribuant ainsi des spécificités individuelles.

Propriétés flexbox appliquées au conteneur

La plupart des propriétés flexbox s'appliquent à l'élément conteneur, car elles concernent la suite de blocs inclus, les items flex.

Type flexbox pour un élément

Nous avons déjà étudié la propriété `display`, mais comme il s'agit de l'opération de base, l'attribution au conteneur du type `flex` ou `inlineflex` sera l'objet de notre premier tableau.

TABLEAU 7–31 Valeurs de type flexbox pour la propriété `display`

Propriété	<code>display</code>
Exemples	<code>section { display: flex; }</code> <code>div.enligne { display: inline-flex; }</code>
Valeurs possibles	En plus des valeurs classiques pour la propriété <code>display</code> (voir le chapitre précédent), deux valeurs confèrent le type flexbox à l'élément, donc le type item flex aux blocs qu'il contient : <code>flex</code> : élément flexbox de type bloc ; <code>inline-flex</code> : élément flexbox de type « bloc en ligne » comme pour la valeur <code>inline-block</code> .
Héritage	<code>Non</code>

Direction et axe principal des blocs

Si dans leur conteneur flexbox, les blocs items flex s'alignent initialement de la gauche vers la droite, cette orientation est modifiable à l'aide de la propriété `flex-direction` : alignement horizontal ou vertical, dans un sens ou dans l'autre.

En anglais, *l'axe principal* défini par cette propriété s'appelle *main axis*, tandis que *l'axe perpendiculaire* est nommé *cross axis*.

TABLEAU 7–32 Propriété `flex-direction`

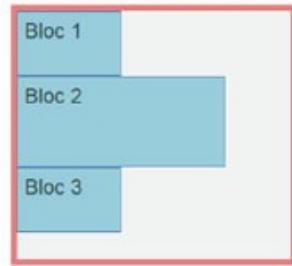
Propriété	<code>flex-direction</code>
	<code>.r2 { display: flex; flex-direction: row-reverse; }</code>

Exemples	.v1 { display: flex; flex-direction: column; }
Valeurs possibles	Choix de l'alignement ordonné des blocs items flex : row : de la gauche vers la droite (valeur par défaut) ; row-reverse : de la droite vers la gauche ; column : du haut vers le bas ; column-reverse : du bas vers le haut.
Héritage	<i>Non</i>

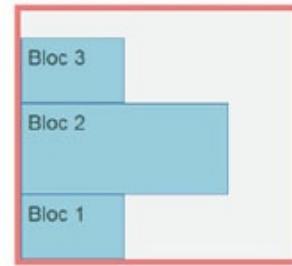
flex-direction: row;



flex-direction: column;



flex-direction: column-reverse;



flex-direction: row-reverse;

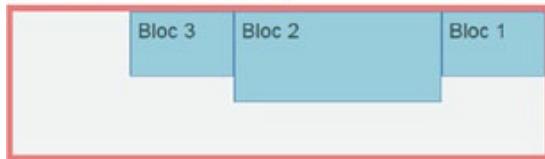


FIGURE 7-19 Effet de la propriété `flex-direction` appliquée au conteneur, avec respectivement les valeurs `row` (valeur par défaut), `row-reverse`, `column` et `column-reverse`

À NOTER Sens par défaut lié à la direction du texte

En général, le sens initial de l'alignement horizontal des blocs inclus dans le conteneur flex va de la gauche vers la droite, parce que le sens d'écriture par défaut est classiquement `ltr` (*left to right*, de la gauche vers la droite). Cependant, si la propriété `direction` vaut `rtl` (*right to left*, de la droite vers la gauche), ce sens initial d'écriture s'inverse. Dans ce cas, la règle `flex-direction: row-reverse` signifie que les blocs seront alignés de la gauche vers la droite.

Par conséquent, si l'écriture d'un document est définie de la droite vers la gauche, il faudra raisonner en inversant les mots gauche et droite pour l'alignement des items flex, dans tout ce qui précède et suit à propos de la technique flexbox.

Retour à la ligne des blocs

En mode ligne, lorsque la largeur totale des blocs à répartir dépasse celle du

conteneur, la largeur de chacun des blocs est automatiquement réduite (en mode colonne, les hauteurs de colonnes sont réduites si la hauteur du conteneur est dépassée), pour que l'ensemble ne dépasse pas la taille de la flexbox. Du moins dans une certaine mesure, car il existe une limite de réduction liée au contenu et au-delà de laquelle l'ensemble des blocs restent alignés, mais dépassent des bords des conteneurs.

Ces désagréments peuvent être évités en utilisant la propriété `flex-wrap` qui autorise le retour à la ligne automatique des blocs items flex, ou bien un changement de colonne s'ils sont alignés verticalement.

TABLEAU 7–33 Propriété `flex-wrap`

Propriété	<code>flex-wrap</code>
Exemples	<pre>.r1 {display: flex; flex-wrap: wrap;} .v2 { display: flex; flex-wrap: wrap-reverse; }</pre>
Valeurs possibles	Comportement des blocs items flex qui débordent de leur conteneur : <code>no-wrap</code> : blocs alignés sur la même ligne ou la même colonne, avec compression et débordement possible (valeur par défaut) ; <code>wrap</code> : passage automatique à la ligne du bas ou à la colonne de droite (blocs calés à gauche et en haut de leur conteneur) ; <code>wrap-reverse</code> : passage à la ligne du haut (blocs calés en bas de la flexbox) ou à la colonne de gauche (blocs calés vers la droite).
Héritage	<i>Non</i>

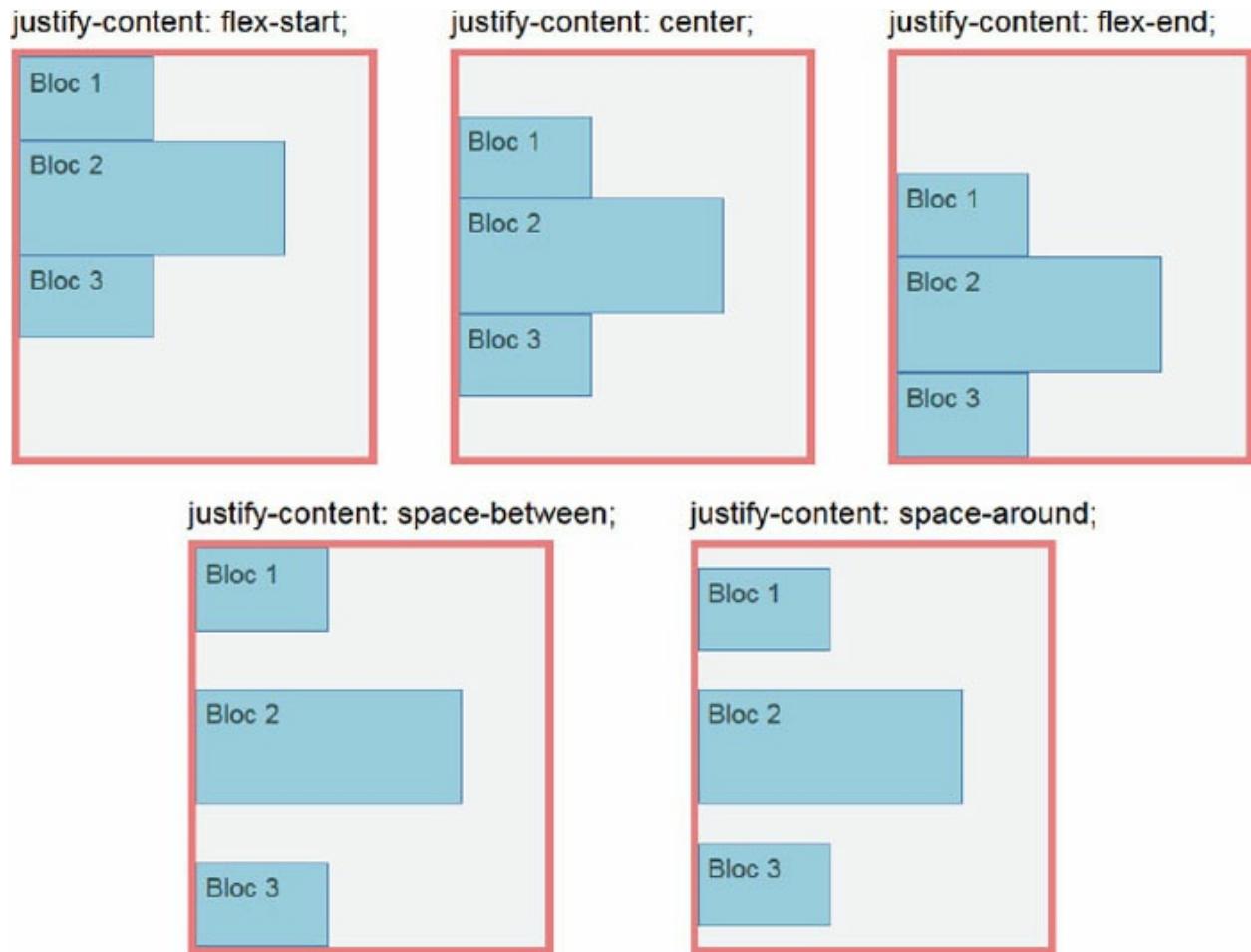


FIGURE 7–20 Gestion des dépassemens avec la propriété `flex-wrap` appliquée au conteneur : respectivement `no-wrap` (valeur par défaut), `wrap` et `wrap-reverse`, en mode ligne, puis en mode colonne

Raccourci `flex-flow` pour la direction et les débordements

Toujours appliqué au conteneur, le raccourci `flex-flow` résume en une seule propriété les caractéristiques de direction `flex-direction` et de débordement `flex-wrap` qui seront appliquées à la flexbox.

TABLEAU 7–34 Propriété `flex-flow`

Propriété	<code>flex-flow</code>
Exemples	<pre>.r2 {display: flex; flex-flow: row wrap-reverse;} .v1 { display: flex; flex-flow: column wrap; }</pre>

Valeurs possibles	Deux valeurs correspondent aux propriétés <code>flex-direction</code> et <code>flex-wrap</code> : 1) <code>row</code> , <code>row-reverse</code> , <code>column</code> , <code>column-reverse</code> 2) <code>no-wrap</code> , <code>wrap</code> , <code>wrap-reverse</code> Si une des deux données est absente, elle prend la valeur par défaut : <code>row</code> pour la direction et <code>no-wrap</code> pour les débordements.
Héritage	<code>Non</code>

Alignement sur l'axe principal

Il s'agit ici de paramétriser l'alignement des blocs suivant l'axe principal, c'est-à-dire l'alignement horizontal en mode ligne et vertical en mode colonne.

De la même façon qu'un paragraphe dans un document texte, l'ensemble des blocs inclus dans une flexbox peut être aligné à gauche, à droite, ou bien voir ses blocs espacés pour occuper toute sa largeur (toute sa hauteur en mode colonne). Ce type de choix s'effectue grâce à la propriété `justify-content`.

TABLEAU 7–35 Propriété `justify-content`

Propriété	<code>justify-content</code>
Exemples	<pre>.r1 { display: flex; justify-content: center; } .r2 { display: flex; justify-content: flex-end; } .v1 { display: flex; flex-direction: column; justify-content: space-between; }</pre>
Valeurs possibles	<p>Type de répartition des items flex dans leur conteneur, horizontalement en mode ligne ou verticalement en mode colonne :</p> <ul style="list-style-type: none"> <code>flex-start</code> : blocs calés à gauche (en haut en mode colonne), c'est la valeur par défaut ; <code>center</code> : ensemble des blocs centré horizontalement (verticalement en mode colonne) ; <code>flex-end</code> : blocs calés à droite (en bas en mode colonne) ; <code>space-between</code> : blocs espacés et répartis sur toute la largeur du conteneur (sur sa hauteur en mode colonne), les blocs extrêmes étant calés sur les bords de la flexbox ; <code>space-around</code> : blocs espacés et répartis comme avec <code>space-between</code>, mais avec des espaces avant le premier et après

le dernier bloc.

Héritage

Non

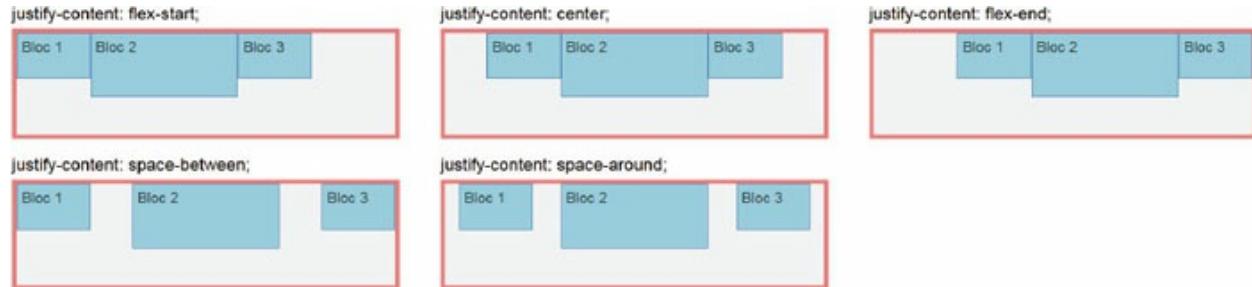


FIGURE 7–21 Alignement horizontal des blocs à l'intérieur d'une flexbox en mode ligne, avec la propriété justify-content

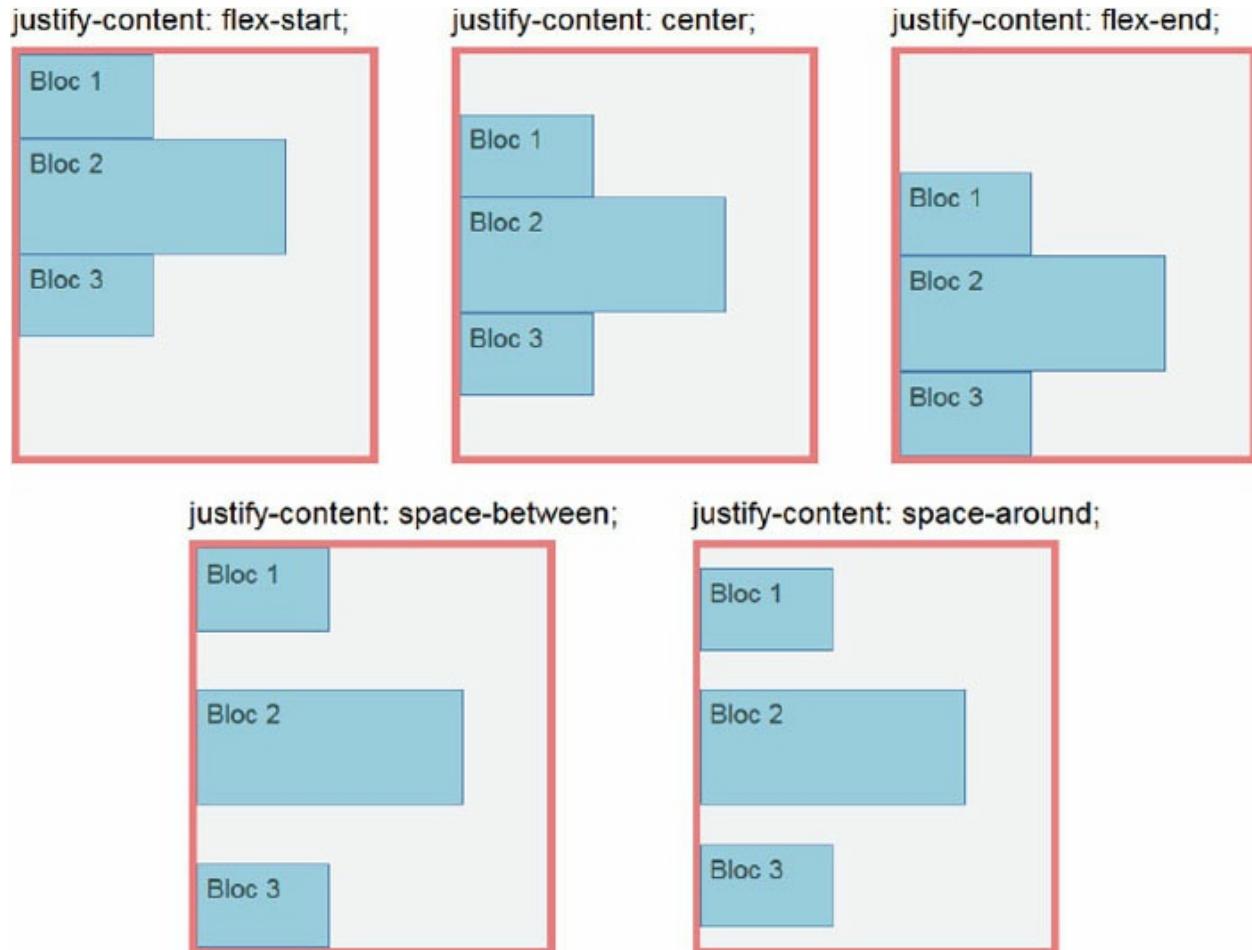


FIGURE 7–22 Alignement vertical des blocs à l'intérieur d'une flexbox en mode colonne, toujours avec la propriété justify-content

Alignement sur l'axe perpendiculaire

L'ensemble des blocs d'une flexbox pourra être placé de différentes façons sur l'axe perpendiculaire, suivant le choix qui sera effectué pour l'alignement vertical des lignes et horizontal des colonnes.

Par défaut , les blocs items flex sont calés vers le haut en mode ligne et vers la gauche en mode colonne. La propriété `align-items` nous propose plusieurs autres sortes d'alignement.

En particulier en mode ligne, la valeur par défaut `align-items:stretch;` agrandit les blocs lorsqu'ils ont une *hauteur automatique* : ainsi, les lignes de blocs se partagent et remplissent toute la hauteur de leur conteneur. En mode colonne, les colonnes de blocs occupent toute la largeur du conteneur, si la largeur de ces blocs n'a pas été spécifiée.

TABLEAU 7–36 Propriété align-items

Propriété	<code>align-items</code>
Exemples	<pre>.r1 { display: flex; align-items: baseline; } .v1 { display: flex; flex-direction: column; align-items: flex-end; }</pre>
Valeurs possibles	Type d'alignement des items flex dans leur conteneur, verticalement en mode ligne ou horizontalement en mode colonne : <code>stretch</code> : c'est la valeur par défaut, qui laisse les lignes de blocs calées vers le haut et étire sur toute la hauteur du conteneur les blocs de hauteur automatique (en mode colonne, les blocs sont calés à gauche et s'étirent sur la largeur de la flexbox si leur largeur est automatique) ; <code>flex-start</code> : blocs calés en haut (à gauche en mode colonne) ; <code>center</code> : ensemble des blocs centrés verticalement en mode ligne (horizontalement en mode colonne) ; <code>flex-end</code> : blocs calés en bas (à droite en mode colonne) ; <code>baseline</code> : blocs alignés en hauteur de telle façon que les premières lignes de texte de chaque bloc se trouvent sur la ligne de base du conteneur (en mode colonne, cette valeur n'a pas d'utilité et équivaut à <code>flex-start</code>).
Héritage	<i>Non</i>

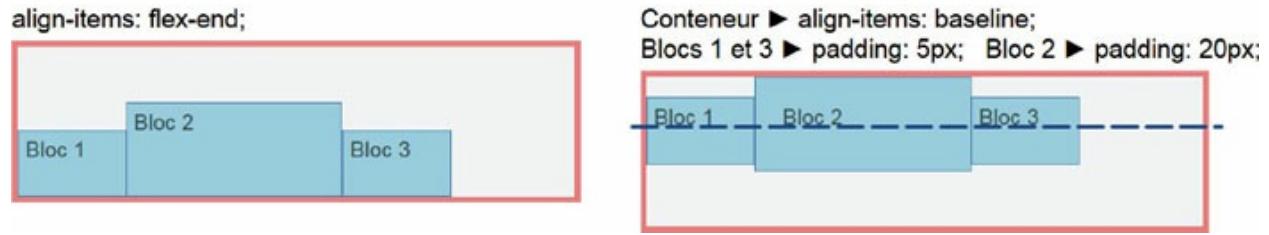
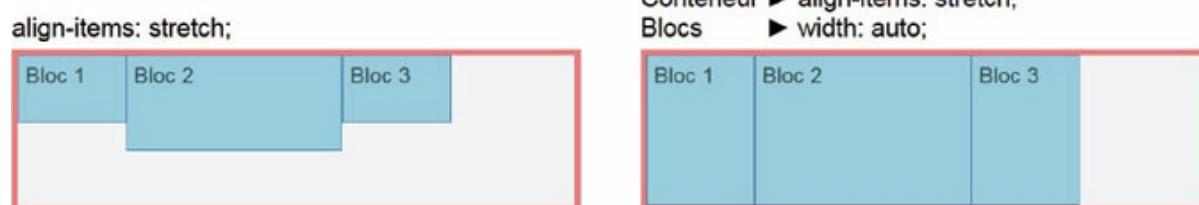


FIGURE 7–23 Alignement vertical des blocs à l'intérieur d'une flexbox en mode ligne, suivant les valeurs de la propriété align-items

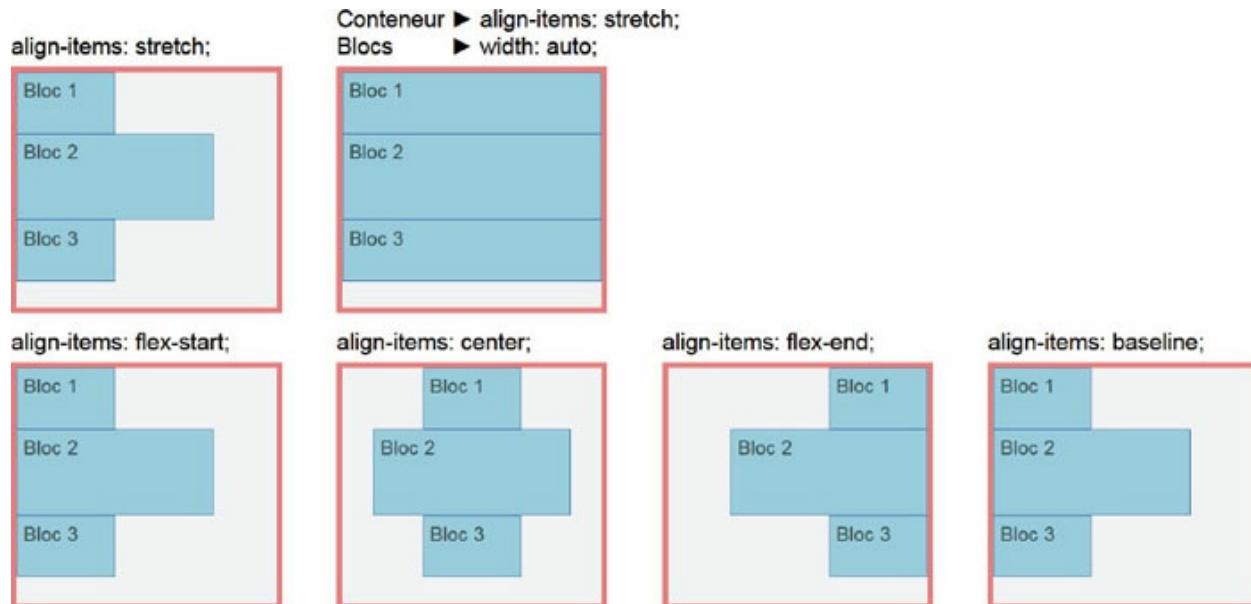


FIGURE 7–24 Alignement horizontal des blocs dans leur conteneur flexbox en mode colonne, suivant la valeur de la propriété align-items

À NOTER Deux cas particuliers pour la propriété align-items

- Si les blocs items flex possèdent des hauteurs définies en mode ligne (ou des largeurs définies en mode colonne), ils ne s'étirent pas sur toute la dimension du conteneur : dans ce cas, la valeur par défaut *stretch* équivaut à *flex-start*.
- Lorsque la position de la première ligne de texte est la même pour tous les blocs (mêmes marges internes, même interlignage), la valeur *baseline* a le même effet que *flex-start*.

Répartition sur l'axe perpendiculaire

Après avoir étudié l'alignement de chaque ligne ou colonne de blocs items flex, allons plus loin en abordant la propriété *align-content* : elle indique la façon dont les lignes ou colonnes de blocs seront réparties dans la flexbox, verticalement en mode ligne et horizontalement en mode colonne.

Cela suppose l'utilisation de la propriété *flex-wrap* avec la valeur *wrap* ou *wrap-reverse*, pour autoriser l'affichage des blocs sur plusieurs lignes ou plusieurs colonnes. En effet, la propriété *align-content* n'a pas d'utilité s'il n'y a qu'une seule ligne ou une seule colonne d'items.

Pour mieux comprendre le tableau qui suit, n'hésitez pas à consulter les figures 7-25 et 7-26 qui présentent, en ligne et en colonne, le résultat de chacune des valeurs possibles pour *align-content*.

TABLEAU 7-37 Propriété align-content

Propriété	align-content
Exemples	<pre>.r1 { display: flex; flex-wrap: wrap; align-content: flex-start; } .v1 { display: flex; flex-flow: column wrap; align-content: center; } .v2 { display: flex; flex-flow: column wrap; align-content: space-around; }</pre>
Valeurs possibles	<p>Voici les modes de répartition des lignes ou colonnes de blocs items flex dans leur conteneur.</p> <p><i>stretch</i> : valeur par défaut, pour laquelle les lignes se partagent toute la hauteur du conteneur (toute sa largeur en mode colonne).</p> <p>Ceux des blocs qui ont une hauteur automatique remplissent</p>

toute la hauteur de leur ligne. En mode colonne, le raisonnement est le même pour la largeur des colonnes et des blocs, par rapport à la largeur de la flexbox.

`flex-start` : les lignes de blocs sont regroupées et calées verticalement en haut de la flexbox (horizontalement à gauche, en mode colonne).

`center` : l'ensemble des lignes de blocs est regroupé et centré verticalement dans le conteneur (centrage horizontal en mode colonne).

`flex-end` : les lignes de blocs sont regroupées et calées verticalement en bas de la flexbox (horizontalement à droite, en mode colonne).

`space-between` : les lignes de blocs sont réparties sur toute la hauteur du conteneur (ou les colonnes sont réparties sur sa largeur), les lignes ou colonnes extrêmes étant calées sur ses bords.

`space-around` : blocs répartis comme avec `space-between`, mais avec un espacement à gauche de la première ligne et à droite de la dernière ligne de blocs (ou au-dessus de la première colonne et en dessous de la dernière colonne de blocs), cet espacement initial et final étant égal à la moitié de celui qui sépare deux lignes (ou deux colonnes) de blocs.

Héritage

Non

Nous venons de faire le tour des propriétés de style qui s'appliquent au conteneur flexbox, ce dernier s'adressant aux items flex dans leur ensemble. Il nous reste à voir quelques propriétés qui distingueront ces blocs les uns des autres et qui les concerteront donc individuellement.

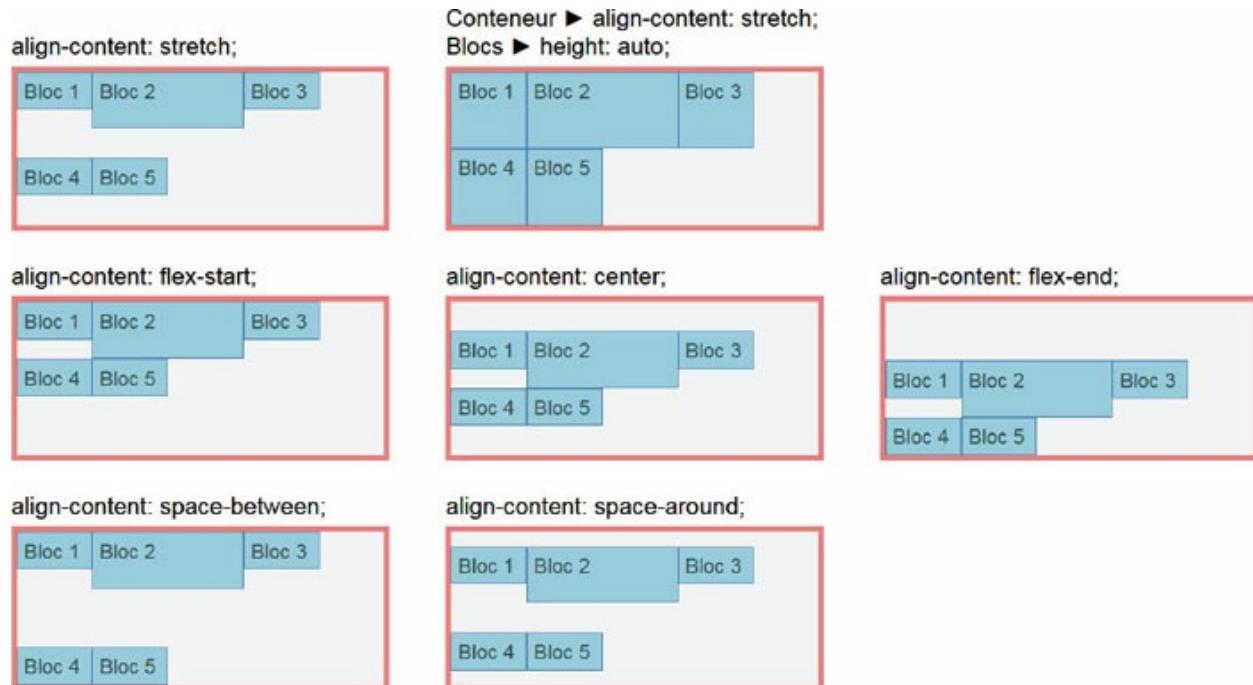


FIGURE 7–25 Répartition des lignes blocs à l'intérieur d'une flexbox avec la propriété align-content

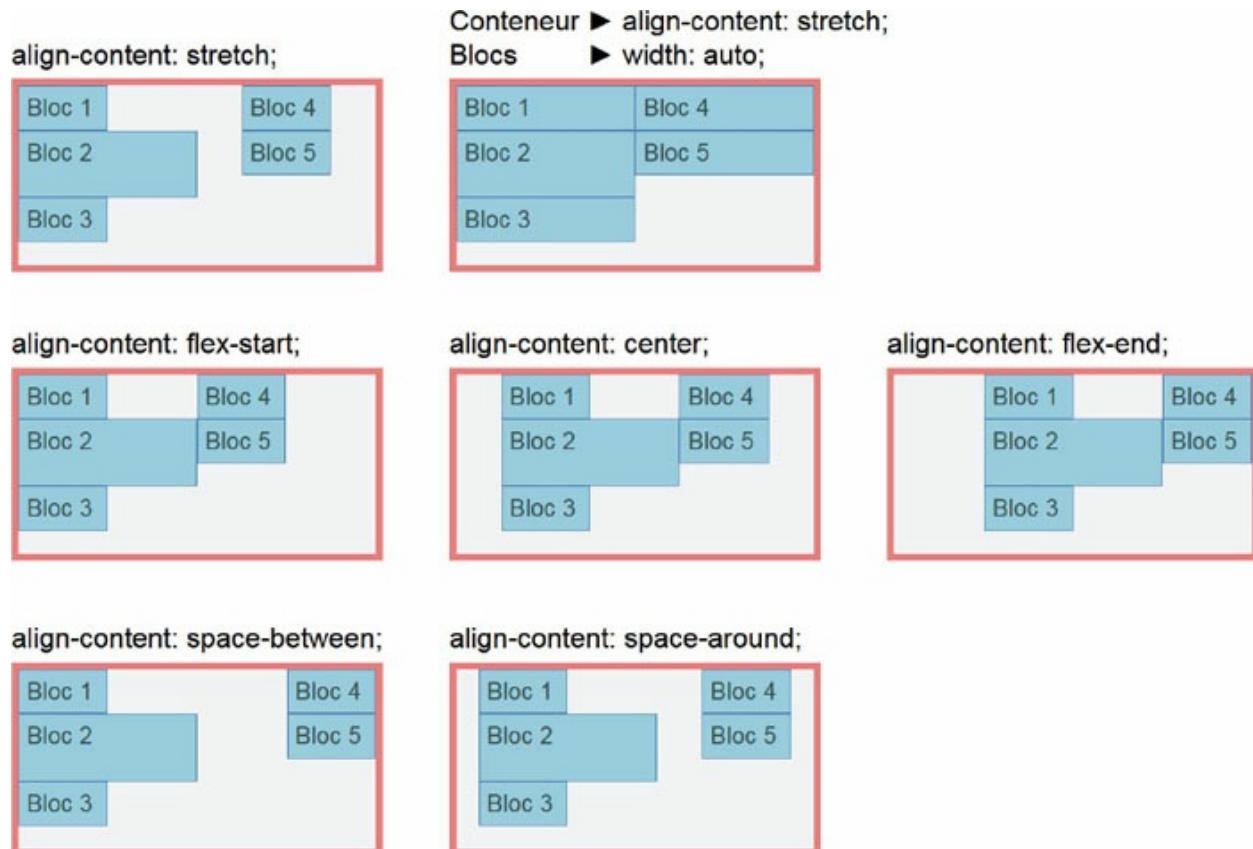


FIGURE 7–26 Répartition des colonnes de blocs à l'intérieur d'une flexbox avec la

propriété align-content

Propriétés flexbox appliquées aux blocs « items flex »

Pour affiner l'alignement ou la répartition des lignes ou colonnes de blocs, il est possible de paramétrier pour chacun de ces blocs items flex :

- son alignement vertical en mode ligne ou vertical en mode colonne, s'il diffère de celui adopté pour l'ensemble des blocs ;
- un coefficient d'accroissement ou de réduction permettant de gérer l'importance de chaque bloc, lors de l'ajustement en largeur d'une ligne ou en hauteur d'une colonne ;
- la dimension de base de chaque bloc : soit sa largeur en mode ligne, soit sa hauteur en mode colonne, autour de laquelle s'effectuera l'ajustement ;
- la priorité de chaque bloc, qui définira sa position d'affichage parmi les autres blocs.

Alignment spécifique d'un bloc

Vous vous souvenez qu'avec la propriété `align-items`, nous avons pu définir la position de l'ensemble des blocs sur l'axe perpendiculaire : alignement vertical en mode ligne, horizontal en mode colonne. La propriété `align-self` donne la possibilité à un bloc de déroger à cette règle commune et de se distinguer des autres par un alignement spécifique.

Les valeurs proposées sont les mêmes que celle utilisables avec `align-items`, plus la valeur `auto` qui revient à ne pas particulariser le bloc. Rappelons-nous à cette occasion que dans une flexbox, les propriétés `float`, `clear` et `vertical-align` n'ont pas d'effet sur les blocs items flex.

TABLEAU 7–38 Propriété `align-self`

Propriété	<code>align-self</code>
Exemple	<pre>.r1 { display: flex; align-items: flex-start; } .r1 div.bloc1 { align-self: flex-end; }</pre>
Valeurs possibles	Type d'alignement d'un bloc donné dans son conteneur, verticalement en mode ligne ou horizontalement en mode colonne : <code>auto</code> : valeur par défaut, qui reprend l'alignement défini par

`align-items` dans la flexbox (qui est la valeur par défaut `stretch` si `align-items` n'est pas utilisé) ;
`stretch` : bloc calé vers le haut et, s'il a une hauteur automatique, étiré sur toute la hauteur du conteneur (en mode colonne, le bloc est calé à gauche et si sa largeur est automatique, il s'étire sur la largeur de la flexbox) ;
`flex-start` : bloc calé en haut (à gauche en mode colonne) ;
`center` : bloc centré verticalement dans son conteneur (horizontalement en mode colonne) ;
`flex-end` : bloc calé en bas (à droite en mode colonne) ;
`baseline` : bloc aligné en hauteur de telle façon que sa première ligne de texte se trouve sur la ligne de base du conteneur (en mode colonne, cette valeur n'a pas d'utilité et équivaut à `flex-start`).

Héritage

Non

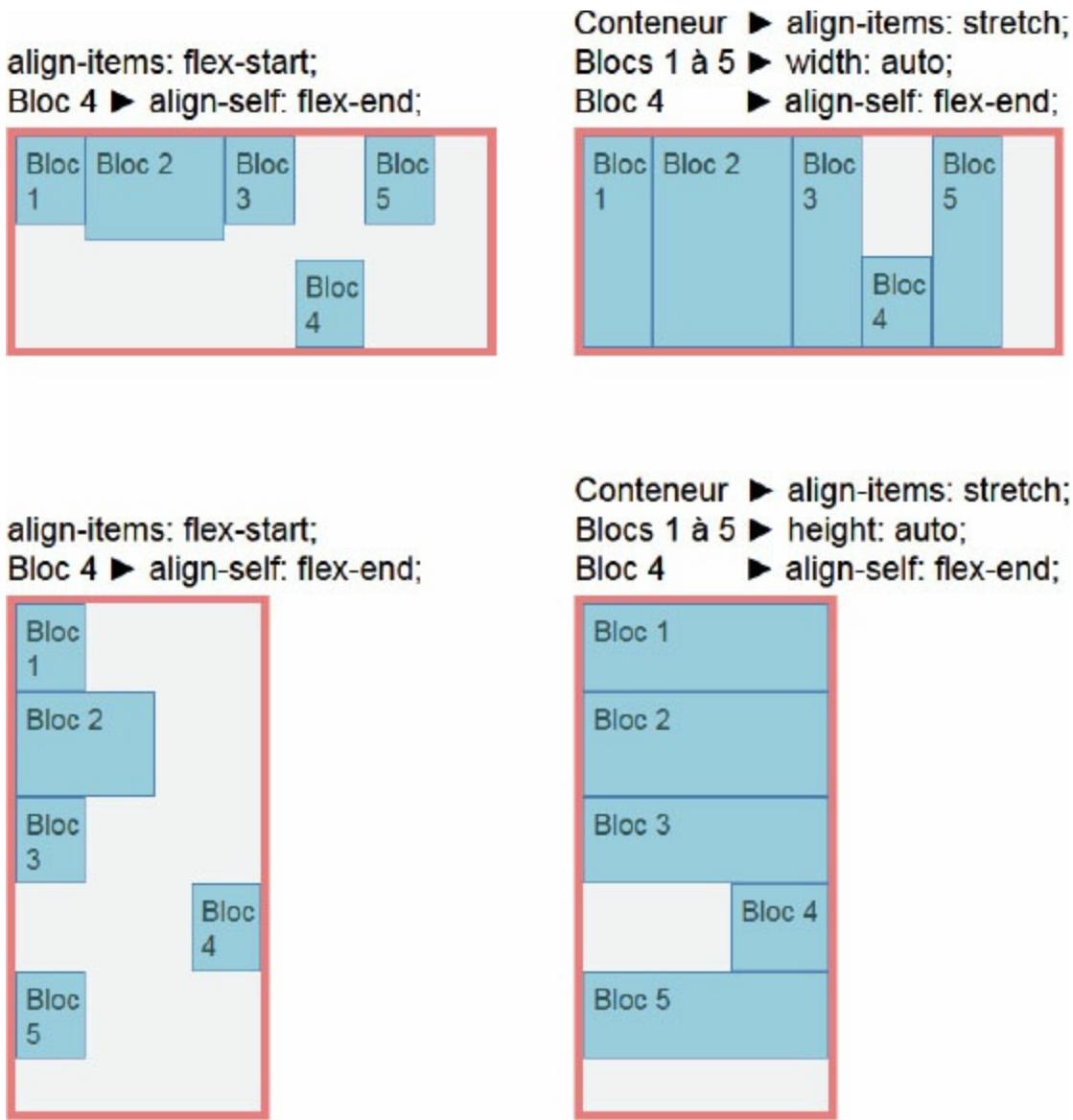


FIGURE 7–27 La propriété `align-self` attribue un alignement spécifique à un bloc.

Agrandissement automatique des blocs

Nous avons vu que dans une flexbox où les dimensions des blocs n'ont pas été définies, les lignes de blocs se partagent toute la hauteur de leur conteneur, tandis que les colonnes de blocs remplissent toute sa largeur.

Comment effectuer alors un ajustement suivant l'axe principal, c'est-à-dire adapter une ligne de blocs à la largeur du conteneur, ou une colonne de blocs à sa hauteur ? Grâce à la propriété `flex-grow` qui va autoriser l'agrandissement automatique des items flex. Nous pourrons ainsi, par exemple, répartir une ligne de menus sur toute la largeur de la barre de navigation.

Si `flex-grow` vaut `0` (valeur par défaut), il n'y aura pas d'agrandissement pour le bloc concerné. S'il a une valeur positive, son ajustement s'effectuera en fonction des valeurs `flex-grow` des autres blocs : ce sont des coefficients d'agrandissement relatifs, comme le montre la figure 7-28.

TABLEAU 7–39 Propriété `flex-grow`

Propriété	<code>flex-grow</code>
Exemple	<pre>.menu { display: flex; } .menu li { width: auto; flex-grow: 1; }</pre>
Valeurs possibles	Coefficient d'agrandissement d'un bloc donné pour ajuster la ligne de blocs à la largeur de son conteneur, ou la colonne de blocs à la hauteur de la flexbox : <code>0</code> : valeur par défaut, pas d'agrandissement autorisé pour ce bloc ; <i>nombre positif, entier ou décimal</i> : coefficient d'accroissement du bloc, relatif aux valeurs <code>flex-grow</code> des autres blocs.
Héritage	<i>Non</i>

Calcul de l'agrandissement des blocs

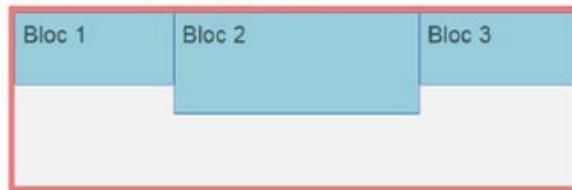
Nous allons découvrir les quelques calculs simples qui expliquent le fonctionnement exact de la propriété `flex-grow`. Mais il est courant d'utiliser celle-ci de façon pragmatique, au jugé. Par conséquent, si vous n'êtes pas passionné par ces détails, vous pouvez les passer et aborder la propriété suivante.

Blocs de largeur et hauteur définies

Conteneur ► `display: flex;`
Blocs de largeur et hauteur définies



Conteneur ► `display: flex;`
Blocs ► `flex-grow: 1;`



Conteneur ► `display: flex;`
Blocs ► valeurs respectives de `flex-grow: 1, 1 et 3`

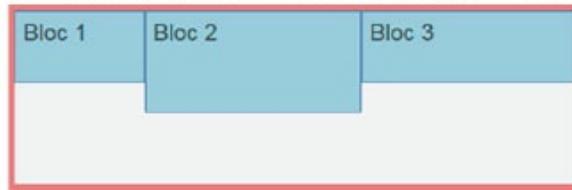
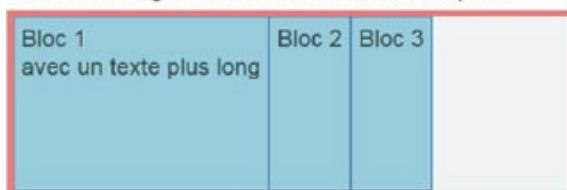


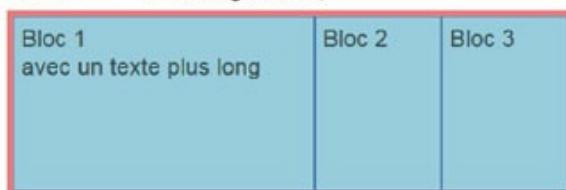
FIGURE 7–28 Agrandissement d'une ligne de blocs, ajustée à la largeur de la flexbox avec la propriété `flex-grow`

Blocs de largeur et hauteur automatiques

Conteneur ► `display: flex;`
Blocs de largeur et hauteur automatiques



Conteneur ► `display: flex;`
Blocs ► `flex-grow: 1;`



Conteneur ► `display: flex;`
Blocs ► valeurs respectives de `flex-grow: 1, 1 et 3`

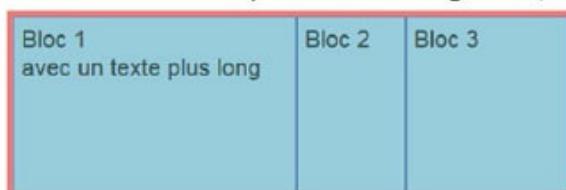


FIGURE 7–29 Agrandissement par `flex-grow` d'une ligne de blocs dont les dimensions sont automatiques

Rappelons-nous le code HTML correspondant à la figure 7-28 :

```
<section>
  <div class="bloc b1">Bloc 1</div>
  <div class="bloc b2">Bloc 2</div>
  <div class="bloc b3">Bloc 3</div>
```

```
</section>
```

Voici un extrait du code CSS associé à ces éléments :

```
section { width: 390px; box-sizing: content-box; }
div.b1, div.b3 { width: 60px; flex-grow: 1; }
div.b2 { width: 120px; flex-grow: 1; }
```

Calcul de l'espace vide dans la flexbox :

- largeur totale des blocs `b1`, `b2` et `b3` : $60 + 120 + 60 = 240$ pixels ;
- largeur de la section : 390 pixels ;
- espace disponible à répartir en largeur : $390 - 240 = 150$ pixels.

Les trois blocs `<div>` ayant la même valeur pour `flex-grow`, cette largeur disponible sera partagée en trois pour être répartie uniformément. Le calcul est le suivant :

- nombre de blocs (de même coefficient `flex-grow`) : 3 ;
- espace à ajouter à chaque bloc : $150 / 3 = 50$ pixels ;
- nouvelles dimensions des blocs `b1`, `b2` et `b3` : respectivement 110, 170 et 110 pixels.

Le total est donc de $110 + 170 + 110 = 390$ pixels ; l'ajustement est bien effectué (deuxième image de la figure 7-28).

Observons à présent un autre cas de figure, dans lequel nos trois blocs ont toujours les mêmes dimensions, mais prennent des valeurs différentes pour `flex-grow` :

```
div.b1 { width: 60px; flex-grow: 1; }
div.b2 { width: 120px; flex-grow: 1; }
div.b3 { width: 60px; flex-grow: 3; }
```

Les largeurs des blocs étant les mêmes, l'espace qui reste à partager vaut toujours 150 pixels. Pour effectuer la répartition, le navigateur va utiliser une moyenne pondérée :

- calcul du total des valeurs `flex-grow` : $1+1+3 = 5$;
- division de l'espace disponible par ce total : $150 / 5 = 30$ pixels ;
- pour chaque bloc, ajout à sa largeur de ces 30 pixels multipliés par son

coefficient `flex-grow`.

Par conséquent, la largeur finale des blocs sera de :

- $60 + 30 \times 1 = 90$ pixels pour le premier bloc `<div>` ;
- $120 + 30 \times 1 = 150$ pixels pour le deuxième ;
- $60 + 30 \times 3 = 150$ pixels pour le troisième, car son `flex-grow` vaut 3.

Le total est de $90 + 150 + 150 = 390$ pixels, l'ajustement étant à nouveau réalisé (troisième image de la figure 7-28).

En résumé, le calcul des largeurs de blocs pour l'ajustement est le suivant :

- largeur disponible = largeur de la flexbox – somme des largeurs des blocs ;
- largeur modifiée d'un bloc = largeur initiale du bloc + `flex-grow` du bloc \times (largeur disponible / total des `flex-grow`).

PRÉCISION Agrandissement des blocs de largeur automatique

Lorsque les blocs items flex ont une largeur automatique, le navigateur calcule d'abord la dimension de chacun de ces éléments, en fonction de leur contenu (première image de la figure 7-29).

À partir de ces dimensions, il effectue les mêmes opérations que pour des blocs de dimensions définies, répartissant l'espace disponible en fonction des valeurs de `flex-grow` (deuxième et troisième images de la figure 7-29).

Les explications qui précèdent et les illustrations associées concernent le mode ligne de la flexbox. Ces raisonnements restent les mêmes en mode colonne, en remplaçant *lignes de blocs* par *colonnes de blocs*, *largeur* par *hauteur* et la propriété `width` par `height`.

En mode colonne, le calcul de chaque hauteur ajustée de bloc est donc :

- hauteur disponible = hauteur de la flexbox – somme des hauteurs des blocs ;
- hauteur modifiée d'un bloc = hauteur initiale du bloc + `flex-grow` du bloc \times (hauteur disponible / total des `flex-grow`).

Réduction automatique des blocs

Le cas précédent consistait à agrandir les blocs items flex dans un conteneur trop grand. Que se passe-t-il lorsque, au contraire, la flexbox est trop petite pour contenir une ligne ou une colonne de blocs ?

C'est là qu'intervient la propriété `flex-shrink`, qui autorise la diminution de la largeur des blocs en ligne, ou de leur hauteur en mode colonne. Cette propriété est presque symétrique à `flex-grow` que nous venons d'étudier, à deux nuances près toutefois.

- Pour réduire des blocs qui devraient déborder de leur flexbox, il n'est pas obligatoire d'utiliser `flex-shrink`, car sa valeur par défaut est `1`, ce qui autorise un ajustement automatique et uniforme. Ce n'était pas le cas pour l'agrandissement, la valeur par défaut de `flex-grow` étant `0`.
- Plus le coefficient de réduction `flex-shrink` d'un bloc est grand, plus sa dimension sera affectée par l'ajustement, comme c'est le cas également avec `flex-grow`. Cependant, le calcul est plus complexe, car il fait appel ici à une double pondération, sur la valeur relative du coefficient `flex-shrink` bien sûr, mais aussi sur la dimension initiale du bloc.

TABLEAU 7–40 Propriété `flex-shrink`

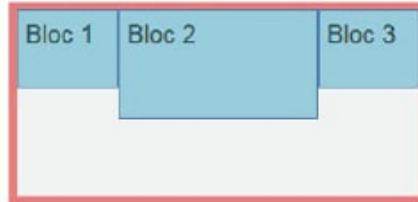
Propriété	<code>flex-shrink</code>
Exemples	<pre>.menu { display: flex; } .menu li.accueil { flex-shrink: 1; } .menu li.infos { flex-shrink: 2; }</pre>
Valeurs possibles	Coefficient de réduction d'un bloc donné pour éviter les débordements sur l'axe principal (ajustement de la ligne de blocs à la largeur de son conteneur, ou de la colonne de blocs à la hauteur de la flexbox) : <code>0</code> : pas de réduction autorisée pour ce bloc ; <code>1</code> : valeur par défaut, réduction autorisée pour ce bloc ; <i>nombre positif, entier ou décimal</i> : coefficient de réduction du bloc, relatif aux valeurs <code>flex-shrink</code> des autres blocs.
Héritage	<code>Non</code>

Blocs de largeur et hauteur définies

Conteneur ► `display: flex;`
Blocs ► `flex-shrink: 0;`



Conteneur ► `display: flex;`
Blocs ► `flex-shrink: 1;` (ou `flex-shrink` non défini)



Conteneur ► `display: flex;`
Blocs ► valeurs respectives de `flex-shrink`: 1, 1 et 3

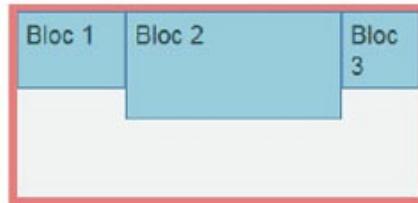
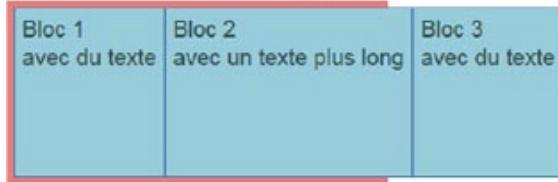


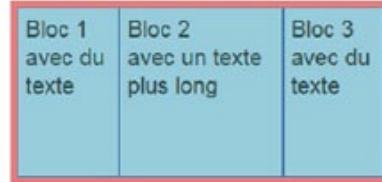
FIGURE 7–30 Réduction d'une ligne de blocs trop grande pour son conteneur, ajustée à la largeur de la flexbox avec la propriété `flex-shrink`

Blocs de largeur et hauteur automatiques

Conteneur ► `display: flex;`
Blocs ► `flex-shrink: 0;`



Conteneur ► `display: flex;`
Blocs ► `flex-shrink: 1;` (ou `flex-shrink` non défini)



Conteneur ► `display: flex;`
Blocs ► valeurs respectives de `flex-shrink`: 1, 1 et 3

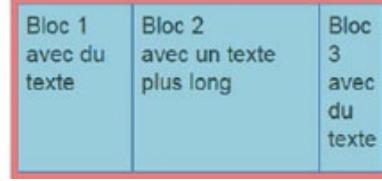


FIGURE 7–31 Réduction par `flex-shrink` d'une ligne de blocs dont les dimensions sont automatiques

Calcul de la réduction des blocs

La technique de réduction des blocs avec `flex-shrink` est un peu plus complexe que celle de l'agrandissement avec `flex-grow`. Elle est donnée ici

pour information sur le fonctionnement de cette propriété.

Dans la pratique, chacun réglera ces paramétrages de façon intuitive. Par conséquent, si vous n'êtes pas adepte de calculs ou si vous êtes pressé de découvrir les propriétés suivantes, vous pouvez passer cette partie sans hésitation.

Nous reprenons toujours le même code HTML, associé à la figure 7-30 :

```
<section>
  <div class="bloc b1">Bloc 1</div>
  <div class="bloc b2">Bloc 2</div>
  <div class="bloc b3">Bloc 3</div>
</section>
```

Avec le code CSS suivant, nous allons voir que les blocs débordent de la flexbox :

```
section { width: 260px; box-sizing: content-box; }
div.b1, div.b3 { width: 80px; flex-shrink: 0; }
div.b2 { width: 160px; flex-shrink: 0; }
```

Calcul du débordement en largeur, hors de la flexbox :

- largeur totale des blocs `b1`, `b2` et `b3` : $80 + 160 + 80 = 320$ pixels ;
- largeur de la section : 260 pixels ;
- débordement des blocs en largeur : $320 - 260 = 60$ pixels.

Puisqu'ici tous les blocs ont leur propriété `flex-shrink` à 0, ils ne se réduisent pas et l'ensemble déborde du conteneur (première image de la figure 7-30).

Pour éviter trop de calculs similaires et comme la deuxième image de la figure 7-30 est un cas particulier, nous allons étudier directement le cas général présenté par la troisième image de cette figure 7-30, avec différentes valeurs différentes de `flex-shrink` pour les blocs, d'où le code CSS :

```
section { width: 260px; box-sizing: content-box; }
div.b1 { width: 80px; flex-shrink: 1; }
div.b2 { width: 160px; flex-shrink: 1; }
div.b3 { width: 80px; flex-shrink: 3; }
```

Il n'y a pas de changement pour les dimensions ; le débordement théorique est toujours de 60 pixels.

Pour chaque bloc, il s'agit d'abord de calculer sa *largeur pondérée* : c'est sa largeur initiale multipliée par son coefficient `flex-shrink`.

Le coefficient de réduction de chaque bloc sera égal à la division de sa largeur pondérée par le total des largeurs pondérées (ici 480). Ce coefficient est donc inférieur à 1 : c'est la proportion du débordement qu'il faudra enlever à chacun des blocs.

Bloc	Largeur initiale	flex-shrink	Largeur pondérée	Coef. de réduction
Bloc 1	80 px	1	$80 \times 1 = 80$	$80 / 480 = 0,167$
Bloc 2	160 px	1	$160 \times 1 = 160$	$160 / 480 = 0,333$
Bloc 3	80 px	3	$80 \times 3 = 240$	$240 / 480 = 0,5$
Total			480	1

Pour obtenir la nouvelle largeur de chaque bloc, il faut enlever à sa largeur initiale cette proportion du débordement, soit les 60 pixels multipliés par le coefficient de réduction du bloc. D'où finalement les nouvelles largeurs de blocs que voici :

- bloc b_1 : $80 - 60 \times 0,167 = 70$ pixels ;
- bloc b_2 : $160 - 60 \times 0,333 = 140$ pixels ;
- bloc b_3 : $80 - 60 \times 0,5 = 50$ pixels.

Le total de ces largeurs est de $70 + 140 + 50 = 260$ pixels, l'ajustement à la flexbox étant réussi.

La formule générale de calcul pour les largeurs réduites tient donc en trois étapes :

- débordement = somme des largeurs initiales des blocs – largeur de la flexbox ;
- largeur pondérée de chaque bloc = largeur initiale \times `flex-shrink` du bloc ;
- largeur modifiée = largeur initiale – débordement \times (largeur pondérée / somme des largeurs pondérées).

PRÉCISION Réduction des blocs de largeur automatique

Pour réduire les blocs de largeur automatique avec `flex-shrink`, la démarche est la même que pour l'agrandissement avec `flexgrow` : le

navigateur va d'abord calculer la dimension de chaque élément suivant son contenu (première image de la figure 7-31).

À partir de ces dimensions, le processus de réduction reste le même qu'avec les dimensions initiales (deuxième et troisième images de la figure 7-31).

En ce qui concerne le mode colonne, cette méthode reste bien sûr valable, en changeant *lignes de blocs* par *colonnes de blocs*, *largeur* par *hauteur* et la propriété `width` par `height` :

- débordement = somme des hauteurs initiales des blocs – hauteur de la flexbox ;
- hauteur pondérée de chaque bloc = hauteur initiale \times `flex-shrink` du bloc ;
- hauteur modifiée = hauteur initiale – débordement \times (hauteur pondérée / somme des hauteurs pondérées).

Rassurez-vous à double titre : d'une part, nous venons de traiter avec `flex-shrink` les calculs les plus compliqués de cette partie, et d'autre part cette propriété s'utilise généralement en attribuant des « poids » relatifs à chaque bloc, sans effectuer tous ces calculs.

Dimensions de base avant agrandissement ou réduction des blocs

Il existe une propriété `flex-basis` qui permet de définir pour chaque bloc une dimension de base, à partir de laquelle s'effectuera l'ajustement à la flexbox, si nécessaire.

Cette dimension est donc une largeur en mode ligne, ou une hauteur en mode colonne, qui sera utilisée pour les calculs d'agrandissement ou de réduction, avec `flex-grow` ou `flex-shrink`. L'ajustement, qu'il soit dans un sens ou dans l'autre, s'effectue sur l'espace disponible autour de cette dimension de base.

TABLEAU 7–41 Propriété `flex-basis`

Propriété	<code>flex-basis</code>
Exemples	<code>.menu { display: flex; }</code>

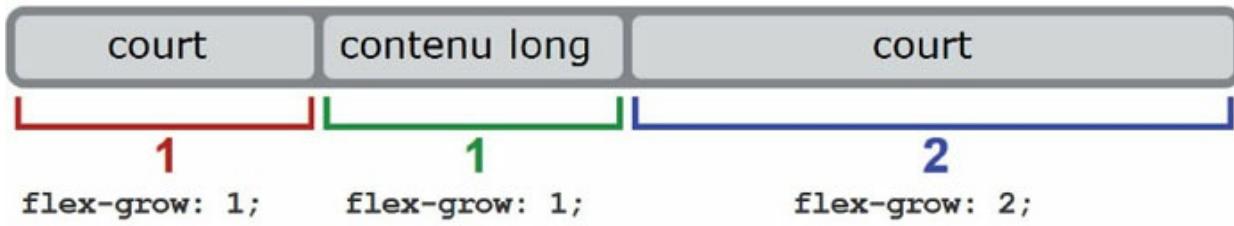
	<pre>.menu li { flex-grow: 1; } .menu li.accueil { flex-basis: 250px; } .menu li.infos { flex-basis: 200px; }</pre>
Valeurs possibles	<p>Mot clé ou dimension initiale du bloc à utiliser pour les calculs d'agrandissement (par <code>flex-grow</code>) ou de réduction (par <code>flex-shrink</code>):</p> <p><code>auto</code> : valeur par défaut, indiquant que les ajustements doivent utiliser les dimensions courantes du bloc (propriété <code>width</code> en mode ligne, <code>height</code> en mode colonne) ;</p> <p><i>dimension (en px, rem, %...)</i> : dimension de base (remplaçant <code>width</code> en mode ligne et <code>height</code> en mode colonne), autour de laquelle s'effectuent les ajustements automatiques (agrandissement ou réduction) à l'intérieur de la flexbox.</p>
Héritage	<i>Non</i>

Pour mieux comprendre cette propriété `flex-basis`, examinons la figure 7-32 qui présente deux exemples d'ajustement dans une flexbox :

- si `flex-basis` vaut `0%`, la dimension de base de chaque bloc n'est pas prise en compte et les opérations d'agrandissement (ou réduction) partagent la largeur entière des blocs, en fonction des coefficients d'ajustement fournis pour chacun (ici, `flex-grow`) ;
- lorsque `flex-basis` prend la valeur `auto`, c'est l'espace disponible autour du contenu qui est partagé, toujours proportionnellement aux poids relatifs des coefficients d'ajustement.

Si une dimension autre que `auto` ou `0%` est fournie pour la propriété `flexbasis`, elle servira de base pour définir le contenu de chaque bloc, l'ajustement s'effectuant sur l'espacement supplémentaire dans chaque bloc, toujours pondéré par les coefficients d'agrandissement ou de réduction.

Blocs ► **flex-basis: 0%;** ⇒ tout le contenu est réparti



Blocs ► **flex-basis: auto;** ⇒ répartition de l'espace disponible

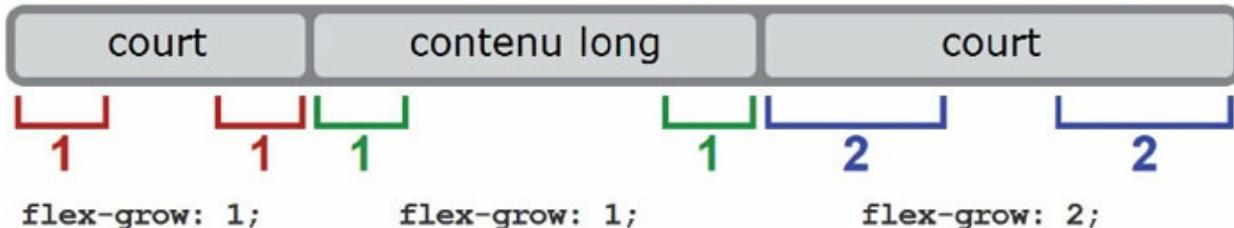


FIGURE 7–32 Différences d'ajustement dans une flexbox lorsque la propriété `flex-basis` prend la valeur `0%` ou `auto`, d'après un schéma du W3C : <https://www.w3.org/TR/css-flexbox-1/>

PRÉCISIONS Cas particuliers pour `flex-basis`

Quelques points sont à noter à propos de la propriété `flexbasis`.

- Si `flex-basis` vaut `auto` et si la dimension associée (`width` en mode ligne, `height` en mode colonne) est absente ou prend elle-même la valeur `auto`, la dimension du bloc sera fonction de son contenu.
- Si une dimension est spécifiée pour `flex-basis`, elle remplacera `width` (ou `height`) dans les calculs, mais elle sera néanmoins limitée par les valeurs `min-width` et `max-width` (ou par `min-height` et `max-height` en mode colonne).
- La propriété `flex-basis` étant une dimension, lorsqu'il s'agit de lui attribuer la valeur `0`, il est habituel de noter plutôt `0%` ou `0px`, de façon à éviter tout comportement inattendu de la part d'un navigateur.

Raccourci `flex` à privilégier

La propriété `flex` est un raccourci qui définit trois valeurs : `flex-grow`, `flex-`

`shrink` et `flex-basis`. Le W3C recommande l'utilisation de ce raccourci `flex`, qui permet d'initialiser en une seule fois ces trois propriétés, d'où une plus grande sûreté d'utilisation.

TABLEAU 7–42 Propriété raccourcie flex

Propriété	<code>flex</code>
Exemples	<pre>.menu { display: flex; } .menu li { flex: 1 1 auto; } .menu li.infos { flex: 1 1 200px; }</pre>
Valeurs possibles	<p>Suite de trois valeurs correspondant respectivement aux propriétés <code>flex-grow</code> (coeffcient d'agrandissement), <code>flex-shrink</code> (coeffcient de réduction) et <code>flex-basis</code> (dimension de base) :</p> <ul style="list-style-type: none"> <code>initial</code> : valeur par défaut équivalente à <code>0 1 auto</code>, soit les valeurs par défaut de ces trois propriétés (rétrécissement seul autorisé) ; <code>auto</code> : équivaut à <code>1 1 auto</code>, soit un ajustement automatique (agrandissement ou réduction) à partir de la dimension courante du bloc (<code>width</code> ou <code>height</code>) ; <code>none</code> : équivalent à <code>0 0 auto</code> (interdit donc tout redimensionnement du bloc).
Héritage	<code>Non</code>

PRATIQUE Valeurs raccourcies équivalentes pour flex

L'écriture de la propriété `flex` peut être raccourcie. Voici donc, pour éviter tout doute sur les trois valeurs qu'elle fournit, quelques versions condensées avec leurs équivalents.

- `flex: initial;` correspond aux valeurs par défaut, donc à `flex: 0 1 auto;` ou encore à `flex: 0 auto;` (pas d'agrandissement, réduction autorisée).
- `flex: auto;` représente `flex: 1 1 auto;`, soit agrandissement et réduction autorisés à partir de la dimension initiale du bloc.
- `flex: none;` interdit tout(e) agrandissement ou réduction, c'est une autre écriture de `flex: 0 0 auto;`.
- `flex: 1;` ou `flex: nombre;` donne le coefficient d'agrandissement, d'où les formes respectives équivalentes et complètes `flex: 1 1`

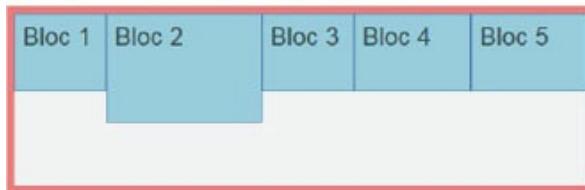
```
auto; et flex: nombre 1 auto;.
```

Ordre des blocs

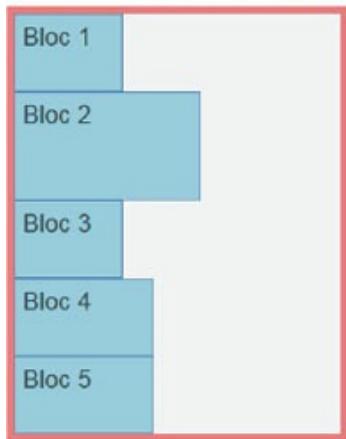
Dans la suite de blocs items flex d'une flexbox, la propriété `order` donne à chaque bloc un ordre de priorité, qui va définir son placement parmi les autres. Les règles de fonctionnement de cette propriété sont très simples :

- plus ce numéro d'ordre est petit, plus le bloc est prioritaire ;
- la valeur attribuée à `order` peut être un nombre entier positif, nul (valeur par défaut) ou négatif ;
- si plusieurs blocs ont le même numéro d'ordre, c'est leur position dans le code HTML qui les départage.

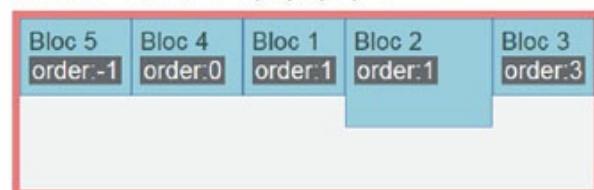
Conteneur ► `display: flex;`
Blocs ► ordre non défini



Conteneur ► `display: flex;`
Blocs ► ordre non défini



Conteneur ► `display: flex;`
Blocs ► `order = 1, 1, 3, 0, -1`



Conteneur ► `display: flex;`
Blocs ► `order = 1, 1, 3, 0, -1`

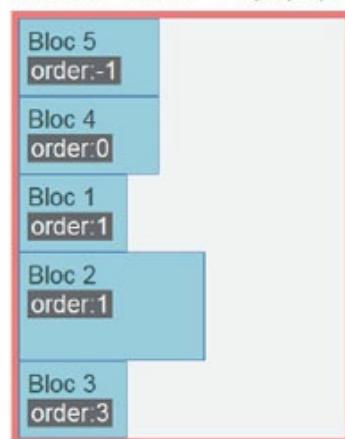


FIGURE 7–33 Modification de l'ordre d'affichage des blocs avec la propriété `order`, dans une flexbox en mode ligne et en mode colonne

TABLEAU 7–43 Propriété `order`

Propriété	<code>order</code>
-----------	--------------------

Exemples	.menu { display: flex; } .menu li.contact { flex: 1 1 auto; order: 5; } .menu li.infos { flex: 1 1 200px; order: -1; }
Valeurs possibles	Numéro d'ordre d'affichage positif, négatif ou nul (valeur 0 par défaut). Les numéros les plus petits sont prioritaires, les nombres doivent être entiers.
Héritage	<i>Non</i>

ATTENTION Ordre fonction de la direction choisie

Si en mode ligne la propriété `flex-direction` a pour valeur `row-reverse`, les blocs prioritaires seront affichés à droite.

En mode colonne, la sens change de la même façon lorsque `flex-direction` vaut `column-reverse`, cas où les « premiers » blocs se trouveront en bas.

Enfin, si la direction d'écriture va de la droite vers la gauche, tout s'inverse en mode ligne.

Ce choix de priorité pourrait ne pas exister : il suffirait de changer l'ordre des éléments en HTML pour obtenir le même résultat. Cependant, cette propriété se révélera bien utile dans plusieurs cas : par exemple, pour mettre en avant un encadré de façon provisoire, ou encore pour gérer la position des blocs en JavaScript.

Nous avons fait le tour de la méthode flexbox et des propriétés qui l'accompagnent, qu'elles concernent le conteneur (la flexbox) ou les blocs qu'il inclut (les items flex).

Il n'y a pas de quoi s'affoler si certaines de ces fonctions paraissent complexes. Les possibilités sont variées et les champs d'action étendus, mais dans la pratique nous utiliserons les propriétés de base.

Deux utilisations pratiques de la flexbox

Pour terminer cette partie sur la flexbox, voici deux exemples d'application qui sont presque déroutants de simplicité, après le parcours des nombreuses fonctions associées à cette technique. Les menus deviennent faciles à mettre en place ; la structure d'une page s'écrit en quelques lignes.

Un menu sur toute la largeur

Ainsi, de façon très simple, nous pourrons ajuster un menu horizontal à la largeur de l'écran ou de la barre de navigation, comme le montre la figure 7-34. Les propriétés flexbox utilisées ici sont pour le conteneur :

```
display: flex;
```

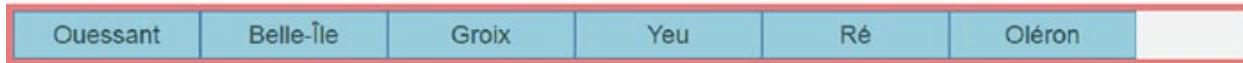
et pour les éléments du menu :

```
flex: 1 1 auto;
```

Le conteneur a une hauteur de 40 pixels et, suivant les cas, possède différentes largeurs qui sont notées sur la figure. Chaque élément du menu a une hauteur automatique et une largeur de 120 pixels : celle-ci est automatiquement réduite ou augmentée en fonction de la largeur totale réservée au menu.

Îles de l'Atlantique

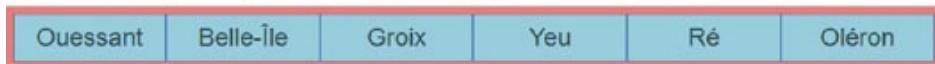
Sans flexbox - largeur du menu : 800px



Avec flexbox - largeur du menu : 350px



Avec flexbox - largeur du menu : 600px



Avec flexbox - largeur du menu : 800px

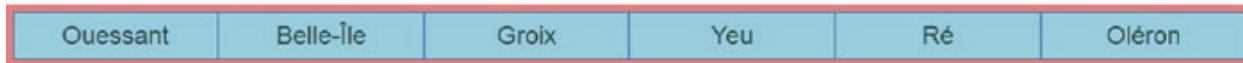


FIGURE 7-34 Menu de navigation sans flexbox, puis avec la méthode flexbox pour différentes largeurs

Structure d'une page à plusieurs colonnes

La flexbox a été inventée pour faciliter la mise en place de blocs dans une page, avec des dimensions adaptables à l'écran. C'est ainsi que nous allons pouvoir définir en quelques lignes la structure générale d'une page comprenant un en-tête, un menu et un contenu sur plusieurs colonnes, ainsi

qu'un pied de page.

L'objectif est d'obtenir la présentation montrée par la figure 7-35. Cet exemple est une version simplifiée d'une page plus complète intitulée « Full-Screen Layout using Flexbox », proposée par John Schulz à l'adresse suivante : <http://bl.ocks.org/jfsiii/5380802>.

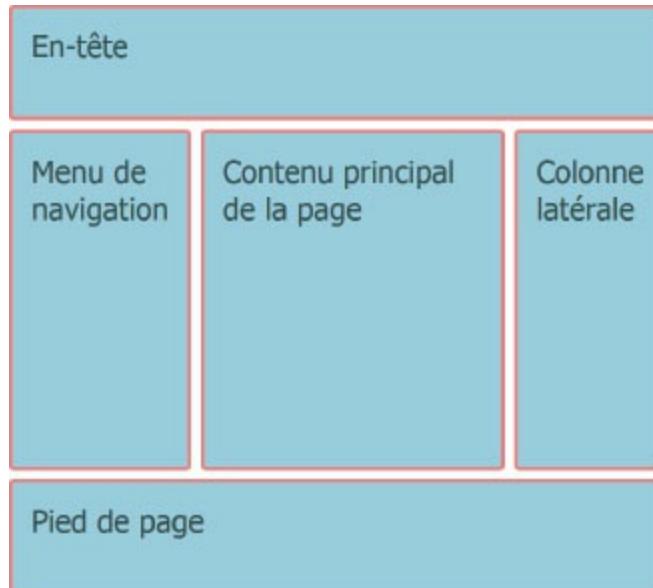


FIGURE 7–35 Une structure flexible à trois colonnes, grâce à la technique de la flexbox

Le code HTML de notre exemple est le suivant, pour le corps de la page :

```
<body>

  <header>En-tête</header>

  <section>
    <nav>Menu de navigation</nav>
    <article>Contenu principal de la page</article>
    <aside>Colonne latérale</aside>
  </section>

  <footer>Pied de page</footer>

</body>
```

Les différentes parties de la page sont aisément reconnaissables par les noms de balises utilisés : `header` pour l'en-tête, un bloc `section` contenant les trois colonnes centrales (`nav` désignant le menu de navigation, `article` le contenu principal et `aside` la colonne latérale), puis la balise `footer` pour le pied de

page.

De façon à simplifier la feuille de styles, nous allons utiliser le modèle flexbox pour la partie centrale, il nous aidera à mettre en place les trois colonnes juxtaposées. Ce code CSS comprend tout d'abord des mises en forme classiques :

```
header, footer, nav, article, aside {  
    padding: 1rem; margin: 0.25rem; min-height: 60px;  
    border: solid 3px lightcoral; border-radius: 0.25rem;  
    background-color: lightblue; color: darkslategray;  
    font: 1.6rem Tahoma, Arial, sans-serif; }  
  
article { min-height: 250px; }
```

Rien que de très classique là-dedans, pour nos différents éléments : des marges internes et externes, des bordures avec les coins arrondis, des couleurs pour le fond et le texte. À cela s'ajoute une hauteur minimale, pour donner un peu d'ampleur à nos blocs, car ils contiennent peu de texte ici : elle est de 60 pixels pour tous les blocs, puis redéfinie à 250 pixels pour la colonne principale.

Nous entrons maintenant dans le vif du sujet : la mise en place des trois colonnes. Celle-ci est tellement simple avec le modèle flexbox qu'elle tient seulement en trois lignes :

```
section { display: flex; }  
article { flex: 3 1 60%; }  
nav, aside { flex: 1 1 20%; }
```

La balise `section`, qui est le conteneur des trois colonnes, est de type `flex`. Comme la direction n'est pas indiquée, les items qu'elle contient se positionneront en ligne, de gauche à droite.

Les largeurs de base sont de 20 % pour chacune des deux colonnes latérales (`nav` et `aside`) et de 60 % pour le contenu principal (`article`). Elles seront associées à des coefficients différents pour l'agrandissement : 3 pour l'article, 1 pour les colonnes situées sur les côtés.

Ainsi, la colonne centrale conserve une largeur plus grande que le menu de gauche et l'encadré de droite, comme nous pouvons le vérifier pour trois différentes largeurs d'écran sur la figure 7-36.

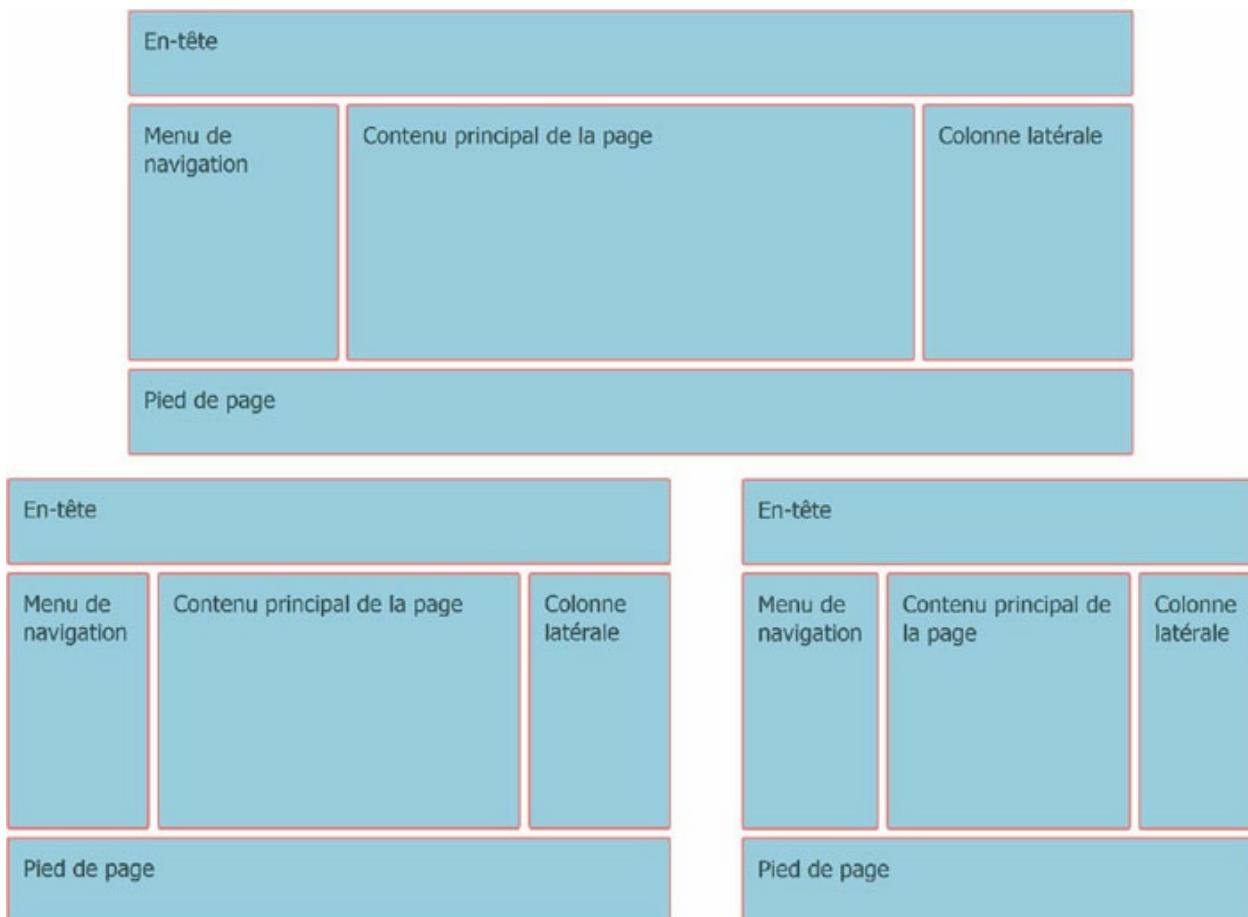


FIGURE 7–36 Répartition automatique du contenu pour différentes largeurs de fenêtre, en fonction des valeurs de la propriété flex pour chacune des colonnes

Partons maintenant à la découverte de fonctions d'un autre domaine, celles qui vont modifier, déplacer, faire tourner ou déformer géométriquement des images, du texte ou toute une partie de contenu à l'intérieur d'une page web.

Transformations géométriques

La norme CSS 3 propose maintenant des transformations géométriques en deux et trois dimensions. Elles s'appliquent aux blocs comme aux éléments en ligne et permettent déplacements, changements d'échelle, rotations et déformations.

Propriété de transformation

C'est la propriété `transform` qui va nous permettre toutes ces possibilités. Elle utilise une palette de fonctions que nous détaillerons plus loin.

Ces transformations s'effectuent à partir d'un point donné de l'élément concerné. Ce point, qui sera l'origine de l'opération géométrique de transformation, est fixé par la propriété `transform-origin`.

TABLEAU 7–44 Propriété transform

Propriété	<code>transform</code>
Exemples	<code>img.logo { transform: rotate(45deg); }</code> <code>a:visited { transform: translate(30px, 5px); }</code>
Valeurs possibles	<code>none</code> : aucune transformation (valeur par défaut), ou fonction : -translations : <code>translateX</code> , <code>translateY</code> , <code>translate</code> -changements d'échelle : <code>scaleX</code> , <code>scaleY</code> , <code>scale</code> -rotation ou déformation : <code>rotate</code> , <code>skewX</code> , <code>skewY</code> , <code>skew</code> Voir plus loin la syntaxe de chacune de ces fonctions.
Héritage	<code>Non</code>

TABLEAU 7–45 Propriété transform-origin

Propriété	<code>transform-origin</code>
Exemples	<code>img.logo { transform-origin: 0 0; }</code> <code>a:visited { transform-origin: 50% 50%; }</code>
Valeurs possibles	Deux valeurs correspondant au couple de coordonnées définissant l'origine géométrique (x,y) de la transformation. Chaque valeur peut être : -soit un nombre, pourcentage ou longueur (unité de dimension) ; -soit un mot-clé, combinant position horizontale (<code>left</code> , <code>center</code> , <code>right</code>) et position verticale (<code>top</code> , <code>center</code> , <code>bottom</code>). Pour les transformations 3D, trois valeurs numériques peuvent être fournies, correspondant aux coordonnées

dans l'espace (x,y,z) de l'origine géométrique de la transformation.

Les valeurs par défaut sont 50% pour x et y (ce qui équivaut aux mots-clés center center) et 0 pour z.

Héritage

Non

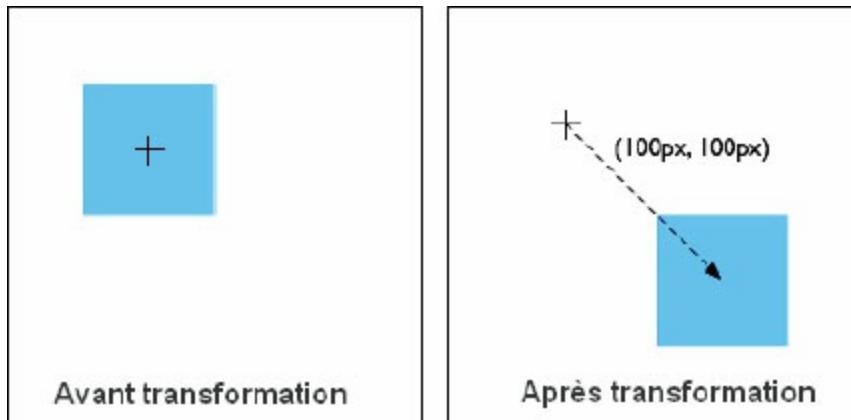


FIGURE 7–37 Transformation géométrique simple : déplacement utilisant la fonction translate (extrait du site www.w3.org).

Fonctions de transformation 2D

Voici l'utilisation des fonctions qu'il est possible d'appeler à l'aide de la propriété `transform`, pour les transformations à deux dimensions. Les figures 7-37 et 7-38, extraites du site du W3C (<https://www.w3.org>), donnent deux exemples de transformations géométriques effectuées à l'aide de ces fonctions.

Pour des transformations sous forme de décalage :

- `matrix(a, b, c, d, e, f)` : transformation matricielle à partir des six nombres fournis ;
- `translateX(nombre)` : décalage horizontal de la valeur fournie ;
- `translateY(nombre)` : décalage vertical de la valeur fournie ;
- `translate(x, y)` : décalage horizontal de `x` et vertical de `y` (valeur `y` facultative, égale à zéro si absente).

Pour des changements d'échelle :

- `scaleX(nombre)` : changement d'échelle horizontale ;
- `scaleY(nombre)` : changement de l'échelle verticale ;
- `scale(x, y)` : changement des échelles horizontale et verticale (valeur `y` facultative, égale à `x` si absente).

Pour une rotation ou déformation :

- `rotate(angle)` : rotation de l'angle indiqué, dans le sens des aiguilles d'une montre ;
- `skewX(angle)` : déformation le long de l'axe horizontal ;
- `skewY(angle)` : déformation le long de l'axe vertical ;
- `skew(angle1, angle2)` : déformation le long de l'axe horizontal et de l'axe vertical (`angle2` facultatif, égal à zéro si absent).

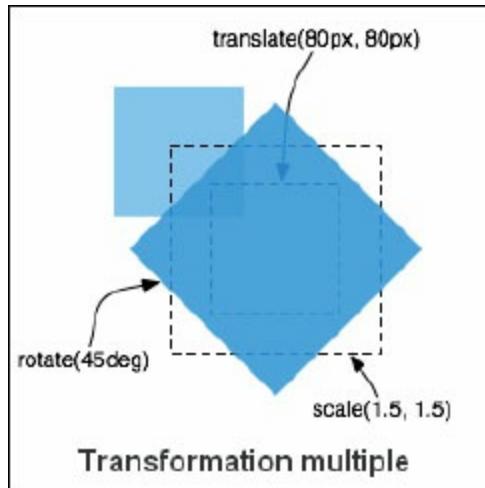


FIGURE 7–38 Transformation géométrique complexe utilisant successivement les fonctions `translate` (déplacement), `scale` (changement d'échelle) et `rotate` (rotation) (extrait du site www.w3.org)

Fonctions de transformation 3D

La propriété `transform` permet également d'utiliser des fonctions de transformation en trois dimensions, que les navigateurs modernes commencent à prendre en compte.

Transformations 3D sous forme de décalage :

- `matrix(a,b,c,d,e,f,g,h,i,j,k,l,m,o,p)` : en trois dimensions, la matrice de transformation a une dimension de 4 par 4, donc elle comprend 16 nombres ;
- `translateX(nombre)` : décalage horizontal de la valeur fournie ;
- `translateY(nombre)` : décalage vertical de la valeur fournie ;
- `translateZ(nombre)` : décalage en profondeur de la valeur fournie ;
- `translate3d(x,y,z)` : décalage horizontal de x, vertical de y (valeur y facultative, égale à zéro si absente) et en profondeur de z (si cette valeur est absente, elle vaut également zéro).

Changements d'échelle :

- `scaleX(nombre)` : changement d'échelle horizontale ;
- `scaleY(nombre)` : changement de l'échelle verticale ;
- `scaleZ(nombre)` : changement de l'échelle dans le sens de la profondeur ;
- `scale3d(x,y,z)` : changement des échelles horizontale, verticale (valeur y facultative, égale à x si absente) et en profondeur (valeur z facultative, égale à 1 si absente, ce qui revient alors à un changement d'échelle à deux dimensions).

Rotations et perspective :

- `rotateX(angle)` : rotation de l'angle indiqué autour de l'axe x ;
- `rotateY(angle)` : rotation de l'angle indiqué autour de l'axe y ;
- `rotateZ(angle)` : rotation de l'angle indiqué autour de l'axe z ;
- `rotate3d(x,y,z,angle)` : rotation de l'angle indiqué, autour du vecteur dont les composantes sont x, y et z, plusieurs triplets de valeurs étant possibles pour définir le même vecteur, car il part de l'origine et passe par le point de coordonnées (x,y,z) ;
- `perspective(profondeur)` : place en perspective le *contenu* du bloc

concerné, suivant la profondeur indiquée (celle-ci doit être positive ou nulle et contenir une unité de mesure, par exemple `500px`), la valeur 0 correspondant à l'absence de perspective.

En ce qui concerne les rotations, l'angle fourni peut être positif ou négatif ; une valeur positive d'angle correspond à une rotation dans le sens des aiguilles d'une montre.

Par ailleurs, il existe des propriétés spécifiques aux transformations en trois dimensions.

- **transform-style:** `valeur`; précise, pour un élément qui a été transformé, si les éléments qu'il contient doivent être aplatis avant la transformation 3D (valeur par défaut, `flat`), ou si ce contenu doit rester en trois dimensions (valeur `preserve-3d`).
- **perspective:** `distance`; place en perspective, avec la profondeur indiquée, les éléments contenus dans le bloc concerné, mais uniquement s'ils ont été positionnés ou transformés (distance à fournir avec une unité, comme `300px`). La valeur par défaut est `none`, qui équivaut à la valeur `0`, soit aucune perspective.
- **perspective-origin:** `x y`; définit les coordonnées de l'origine de la perspective, à partir de valeurs exprimées en pixels, pourcentages, etc. ou par un mot-clé (`left`, `center` OU `right` pour `x`, `top`, `center` OU `bottom` pour `y`). La valeur par défaut est `50% 50%`, ce qui équivaut à `center center`.
- **backface-visibility:** `valeur`; précise si la face arrière d'un objet doit être visible (valeur par défaut, `visible`) ou cachée (valeur `hidden`).

Le Web s'anime en CSS 3

Jusqu'alors, les animations de nos pages étaient créées soit en JavaScript, par exemple à l'aide des célèbres bibliothèques *jQuery* et *MooTools*, soit en *Flash*, avec l'obligation d'utiliser un logiciel externe pour toute modification (*Flash*, *Swish*, *E-anim...*).

Un débat existait d'ailleurs pour décider si les CSS devaient ou non se limiter à la seule mise en forme des pages. À présent, le pas est franchi ; les CSS 3 nous proposent donc quelques propriétés qui simplifient énormément les animations de page les plus courantes.

Les transitions

Les CSS permettent depuis longtemps de modifier l'apparence d'un bloc ou d'un texte dans certaines circonstances : au survol de la souris, lorsque l'élément est sélectionné ou reçoit le focus, ou encore lorsqu'il change de classe par l'intervention d'une fonction JavaScript.

Le but des transitions en CSS 3 est d'effectuer en douceur le passage d'un aspect à l'autre, que ce soit un changement de taille, de position, de couleur, de forme, etc. La propriété `transition`, appliquée à l'objet initial, permettra cet effet d'animation lorsqu'un changement interviendra.

Abordons cette fonction à travers un exemple simple où des liens hypertextes, initialement de couleur verte, deviennent rouges au passage de la souris et changent en même temps de taille :

```
a { color: green; font-size: 16px; transition: color 2s; }
a:hover { color: red; font-size: 20px; }
```

Dans cet exemple, le fonctionnement de la propriété `transition` est le suivant.

- Dans la première règle de style, la propriété `transition` indique que les changements de couleur pour les liens doivent s'effectuer en deux secondes.
- La deuxième programme l'écriture des liens en rouge, au survol de la souris, ainsi qu'un agrandissement de leur taille.
- Au lieu d'être instantané, le changement de couleur s'effectuera progressivement, ici en deux secondes. Cette progressivité vaut aussi pour le retour à la couleur verte, lorsque la souris s'éloigne du lien.
- Cependant, l'agrandissement des caractères sera instantané, car la transition ne concerne ici que la propriété `color`.

Pour que la progressivité s'applique aux deux transformations, nous aurions dû écrire la propriété `transition` comme ceci :

```
transition: color 2s, font-size 2s;
```

Notons que dans cette dernière écriture, il devient possible d'attribuer une durée de transformation spécifique à chacune des propriétés de mise en

forme.

Une écriture simplifiée permet d'attribuer une progressivité à l'ensemble des modifications :

```
transition: all 2s;  
transition: 2s;
```

Les deux propriétés ci-dessus sont équivalentes, car la valeur par défaut est `all`, signifiant que toutes les propriétés sont concernées.

Aux deux paramètres que nous venons d'utiliser, propriété concernée et durée de la transition, il est possible d'ajouter deux autres précisions : la variation de vitesse dans la transition, que nous examinerons en détail un peu plus loin, et le délai avant l'application de la transition. Ces quatre paramètres constituent des propriétés à part, que nous allons étudier.

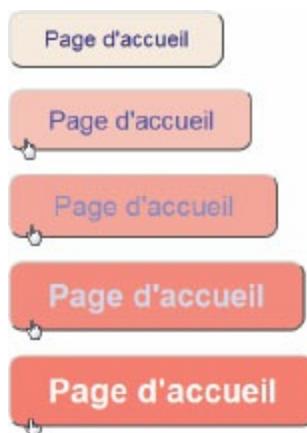


FIGURE 7–39 Exemple de transitions, appliquées ici à la taille du bloc et sa couleur de fond, ainsi qu'à la taille, à la graisse et à la couleur des caractères.

Propriétés concernées par la transition

La propriété `transition-property` définit quelle(s) propriété(s) concernée(s) par la transition.

TABLEAU 7–46 Propriété transition-property

Propriété	<code>transition-property</code>
Exemples	<code>a.menu { transition-property: all; } div { transition-property: width, height; }</code>
Valeurs	<code>none</code> : aucune propriété concernée par les transitions

possibles	a11 (valeur par défaut - toutes les propriétés sont concernées) un <i>nom de propriété</i> ou plusieurs, séparés par des virgules
Héritage	Non

Si plusieurs propriétés sont indiquées, il sera possible de leur associer plusieurs durées, vitesses de progression et délais, en respectant l'ordre indiqué dans `transition-property`.

À NOTER Propriétés concernées par les transitions

Les transitions peuvent être appliquées à une liste définie de propriétés, qui seront donc les valeurs utilisables avec `transition-property`.

- Les dimensions, les marges et certains alignements : `width` et `height` (avec leurs variantes `min-` et `max-`), `margin` et `padding` (ainsi que leurs variantes `-top`, `-bottom`, `-left` et `-right`), `line-height`, `vertical-align`, `text-indent`.
- Les décalages et options de positionnement : `top`, `bottom`, `left`, `right`, `z-index`.
- Les couleurs, encadrements et options de visibilité : `background` et ses souspropriétés (couleur, position, taille), `border` et ses multiples variantes (couleur, épaisseur, espacement, arrondis de bordure) de même que `outline` (couleur, épaisseur, décalage), `box-shadow`, `opacity`, `clip-path` et `visibility`.
- La mise en forme des mots et des lettres : `color`, `font`, `font-size`, `font-weight`, `font-stretch`, `font-size-adjust`, `text-shadow`, `text-decoration-color`, `letter-spacing` et `word-spacing`.
- Le multicolonnage avec le raccourci `columns` et les propriétés associées `column-...` (nombre, largeur et espacement des colonnes, traits de séparation avec leur épaisseur et leur couleur).
- Les transformations 2D et 3D: `transform`, `transform-origin`, `perspective` et `perspective-origin`.

Durée de la transition

La durée d'une transition est exprimée en secondes à l'aide de la propriété `transition-duration`.

TABLEAU 7–47 Propriété transition-duration

Propriété	<code>transitionduration</code>
Exemples	<code>a.menu { transition-duration: 1.5s; } div { transition-duration: 2s, 4s; }</code>
Valeurs possibles	<code>0</code> : (valeur par défaut, pas de transition) une durée indiquée en secondes et suivie sans espace par le symbole « s », (le séparateur décimal étant le point, comme <code>0.5s</code> pour une demi-seconde) plusieurs durées séparées par des virgules si elles varient suivant les propriétés
Héritage	<code>Non</code>

Progression de la transition dans le temps

Une transition peut voir sa progression varier dans le temps. La propriété `transition-timing-function` définit la manière de réaliser la transition, pour une durée totale inchangée : de façon régulière ou saccadée, avec ou sans accélération au début (démarrage progressif) et décélération à la fin (arrivée en douceur).

Par défaut, la transition démarre de façon progressive mais assez rapide, puis sa vitesse reste constante et ralentit nettement sur la fin. Ce ralentissement final se voit bien sur la taille du bloc dans la figure 7-39, dont les images ont été capturées à intervalles de temps réguliers.

TABLEAU 7–48 Propriété transition-timing-function

Propriété	<code>transition-timing-function</code>
Exemples	<code>a.menu { transition-timing-function: ease-out; } div { transition-timing-function: ease, linear; }</code>
Valeurs possibles	<code>linear</code> (linéaire, vitesse constante) <code>ease</code> (valeur par défaut, démarrage progressif mais rapide avec un ralentissement à la fin) <code>ease-in</code> (lent au début, rapide ensuite) <code>ease-out</code> (rapide dès le début et ralenti à la fin) <code>ease-in-out</code> (démarrage et fin en douceur, vitesse plus rapide entre les deux) <code>cubic-bezier(x1,y1,x2,y2)</code> (paramétrage de la courbe à partir des

coordonnées de deux points intermédiaires, indiqués plus loin sur la figure 7-43)

`step-start` ou `step-end` (pas de transition, valeur finale attribuée instantanément au début ou à la fin de la durée de la transition)

`steps(n, start)` ou `steps(n, end)` (transitions par `n` sauts successifs, avec pour `start` le premier saut dès le début de la transition et pour `end` le dernier saut à la fin de la transition, comme le montre la figure 7-41)

Plusieurs valeurs, séparées par des virgules, pourront correspondre à des transitions pour différentes propriétés.

Héritage

Non

La figure 7-40 montre les différences entre les valeurs proposées pour la progression de la transition. Plus la pente de la courbe est raide, plus la transition est rapide. Une pente douce correspond à un ralentissement de l'animation.

Généralement, il sera plus simple d'utiliser une des valeurs prédéfinies qui nous fournissent une panoplie suffisante. Cependant, la page <https://matthewlein.com/ceaser/> propose de définir graphiquement l'évolution de la transition dans le temps, de tester l'effet obtenu sur plusieurs exemples de transition ; elle nous fournit également les paramètres correspondants de la fonction `cubic-bezier`. De façon à mieux sentir la différence entre les différents types de progression dans le temps, il est préférable d'opter pour une durée de 1 000 à 2 000 millisecondes (soit une à deux seconde(s)).

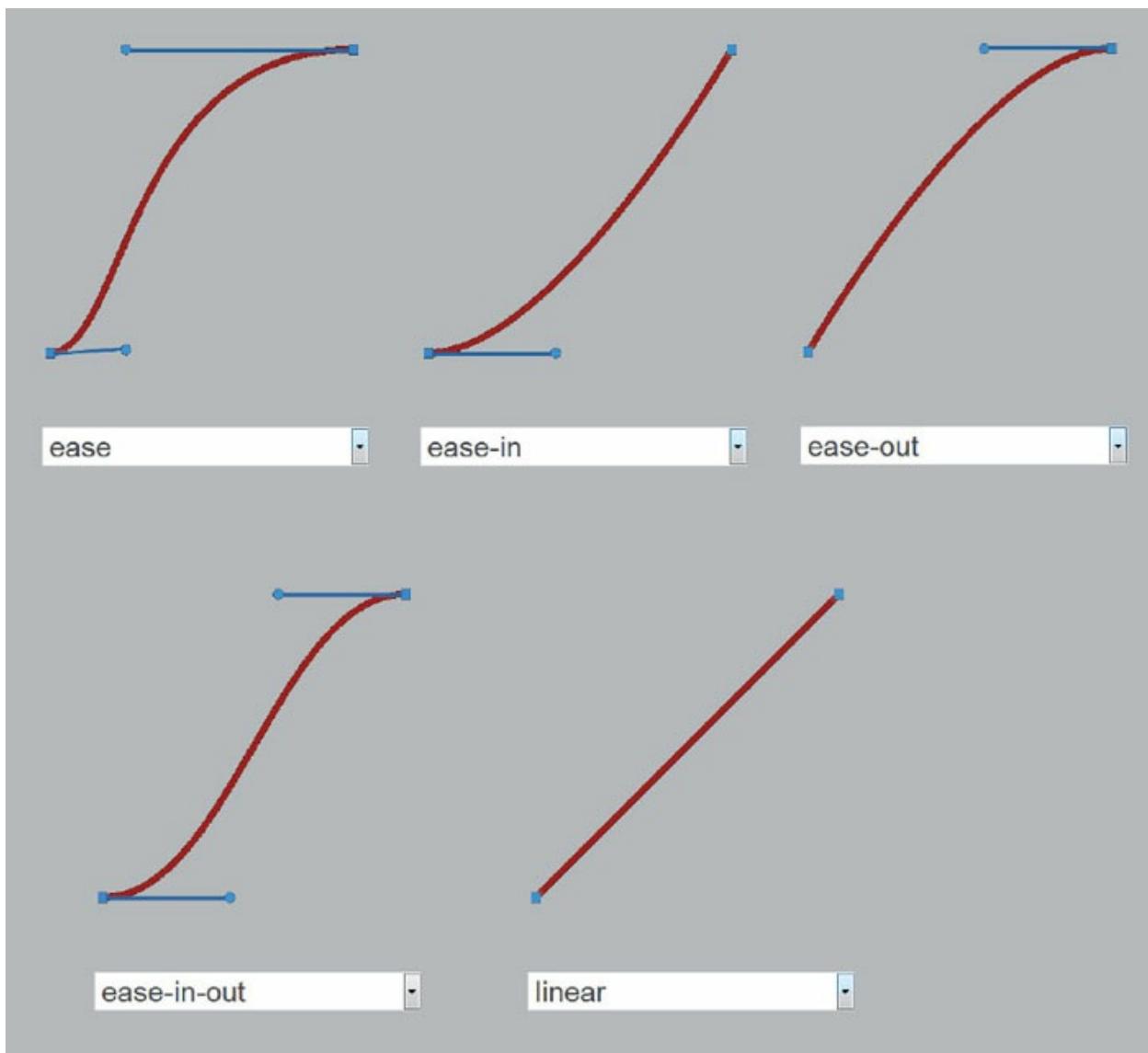


FIGURE 7-40 Différences de progression correspondant aux valeurs prédéfinies pour les transitions. Les pentes raides sont des passages rapides, les pentes douces sont des ralentissements.

La figure 7-41, extraite du site www.w3.org, explique la progression par à coups créée par la valeur `steps(n, start)` ou `steps(n, end)`, n étant le nombre de sauts. La valeur `step-start` correspond à `steps(1, start)` tandis que la valeur `step-end` est équivalente à `steps(1, end)`.

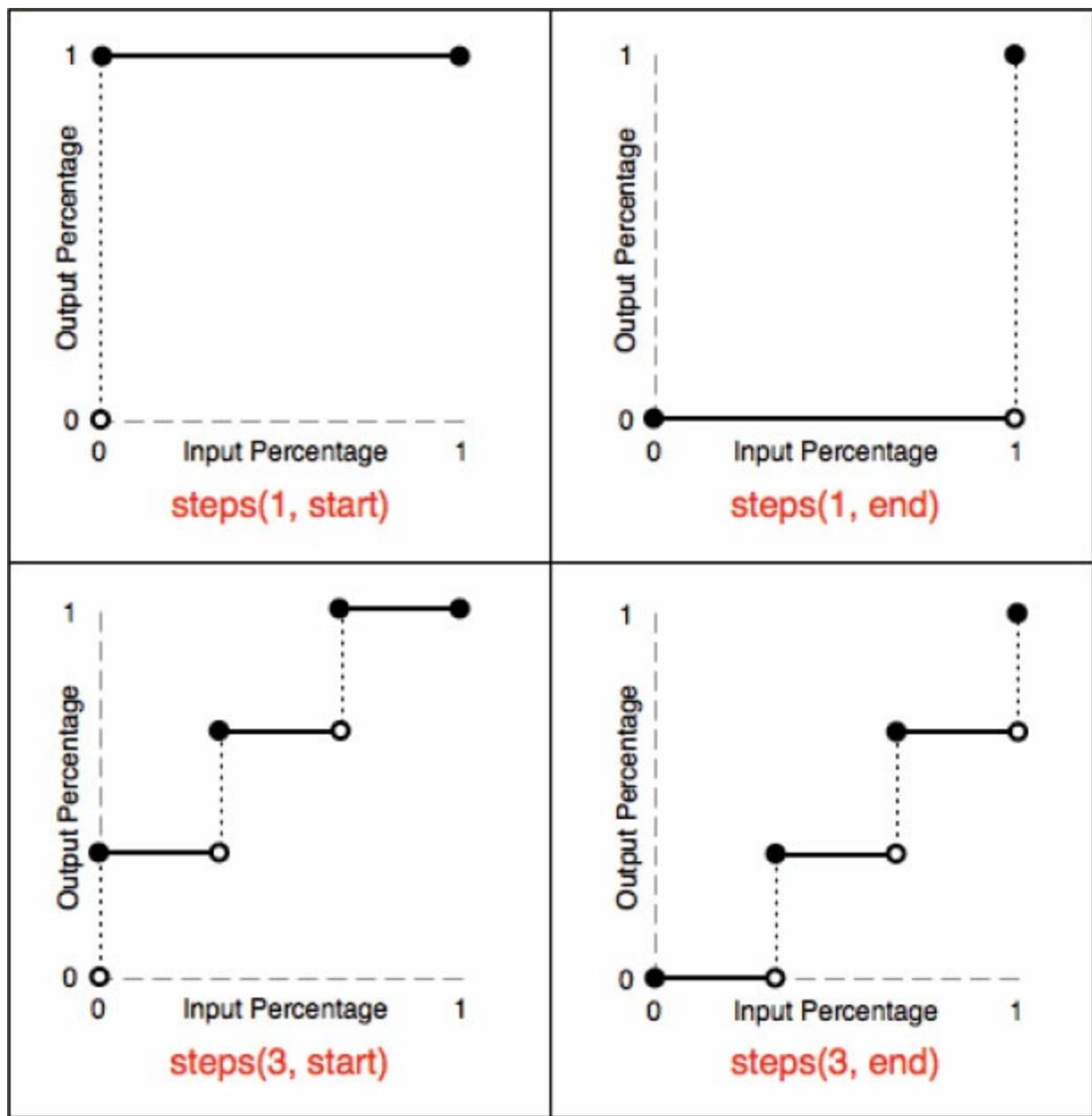


FIGURE 7–41 La transition par paliers, à l'aide de la fonction `steps(n, start)` ou `steps(n, end)` attribuée à la propriété `transition-timing-function` (extrait du site www.w3.org)

Délai avant la transition

Lorsqu'une deuxième valeur en secondes est indiquée dans la propriété `transition`, il s'agit du délai d'attente avant le démarrage de la transition. Cette durée peut être définie séparément, à l'aide de la propriété `transition-delay`.

ATTENTION Délais et durées des transitions

De nombreuses possibilités nous sont offertes, tant pour le choix des transitions que pour leur déroulement dans le temps. Cependant, au moment de fixer la durée d'une transition et le délai éventuel avant son démarrage, nous devrons veiller à ne pas ralentir la consultation des pages : ces effets d'animation ont pour seul but de rendre plus harmonieuses les transitions ; ils ne doivent pas devenir une gêne pour le visiteur.

TABLEAU 7–49 Propriété transition-delay

Propriété	transition-delay
Exemples	a.menu { transition-delay: 0; } div { transition-delay: 0.5s, 1s; }
Valeurs possibles	0 (valeur par défaut, pas de retard au démarrage) un délai indiqué en secondes et suivi par le symbole « s » sans espace avant (le séparateur décimal éventuel est le point), plusieurs délais séparés par des virgules si elles varient en fonction des propriétés
Héritage	Non

Raccourci pour les transitions

Nous terminons par le raccourci transition, celui par lequel nous avons commencé pour aborder ce sujet et que nous utiliserons le plus fréquemment. Il définit à la fois la ou les propriété(s) concernée(s) par la transition, sa durée, son déroulement et son délai éventuel avant le démarrage.

TABLEAU 7–50 Propriété raccourcie transition

Propriété	transition
Exemples	a.menu { transition: 1.5s ease-out; } div { transition: width 2s ease 0.5s, height 4s linear 1s; }
Valeurs possibles	Valeurs successives, séparées par des espaces, des propriétés : transition-property, transition-duration, transition-timing-function et transition-delay Les propriétés qui ne sont pas indiquées prennent leurs valeurs par défaut et seule la

	deuxième durée correspond au délai d'attente. Plusieurs groupes de valeurs, séparés par des virgules, permettent de personnaliser les transitions pour différentes propriétés.
Héritage	<i>Non</i>

RAPPEL Emplacement de la propriété transition

Souvenons-nous que la propriété transition, ainsi que ses sous-propriétés, doivent concerner *l'élément initial*, par exemple `a.menu`, et non la balise correspondant au passage de la souris comme `a.menu:hover`.

En effet, bien que l'action de cette propriété n'apparaisse que sur un changement comme le survol par le curseur, il existe d'autres situations qui peuvent entraîner des modifications d'aspect, comme la réception du focus ou un changement de classe par une action écrite en JavaScript.

C'est pourquoi il est plus simple de noter ces transitions une seule fois, en tant que style appliqué à l'élément initial.

Nous voici prêts à animer nos menus et les changements de mise en forme qui interviennent dans nos pages. Comme d'habitude, nous utiliserons ces effets avec modération, en gardant à l'esprit qu'il faut toujours privilégier la facilité de lecture et d'accès à l'information.

Pour des effets animés qui ne rentrent pas dans le cadre des transitions, les CSS 3 nous proposent, dans le même esprit, la mise en place d'animations.

Les animations

Une règle `@keyframes` permet de créer en CSS 3 des animations qui s'afficheront lors du chargement de la page, en leur attribuant un nom qui permettra de les utiliser dans la feuille de styles avec la propriété `animation`. Nous retrouvons ici le principe de la règle `@font-face`, qui définit un nouveau nom de police utilisable avec `font-family`.

Règle `@keyframes`

Pour créer une nouvelle animation, il suffit de définir un nom et des styles qui correspondront au minimum à deux états, le départ et l'arrivée, comme dans cet exemple :

```
@keyframes descente {  
    from { margin-top: 50px; }  
    to { margin-top: 200px; }  
}
```

Le nom attribué à l'animation est ici `descente`. La situation de départ est une marge du haut de 50 pixels, dont la valeur devient 200 pixels à l'arrivée. Pour obtenir ce mouvement sur un bloc `<div id="titre">` par exemple, il suffira d'utiliser la propriété `animation` comme ci-dessous :

```
div#titre { animation: descente 3s; }
```

L'animation définie par `@keyframes` ne se limite pas à cette forme très simple.

- Elle peut utiliser plusieurs propriétés, chacune ayant des valeurs différentes au départ et à l'arrivée.
- Il est possible de définir des étapes intermédiaires, en les repérant par des pourcentages du temps écoulé.

Voici un exemple d'animation comportant ces deux possibilités, puis son utilisation par une image :

```
@keyframes apparition {  
    0% { width: 0; height: 0; opacity: 0; }  
    20% { width: 50px; height: 70px; opacity: 0.2; }  
    50% { width: 100px; height: 140px; opacity: 0.3; }
```

```

80% { width: 220px; height: 310px; opacity: 0.5; }
100% { width: 250px; height: 350px; opacity: 1; }
}
img.logo { animation: apparition 5s linear; }

```

Dans ce dernier cas, l'image de classe `logo` apparaît progressivement, inexisteante au départ et arrivant finalement à une taille de 250 par 350 pixels, tout en devenant de plus en plus opaque au fur et à mesure de son agrandissement.

Dans la propriété `animation`, la valeur `linear` a été ajoutée ici pour éviter que les différentes étapes ne soient marquées par des ralentissements, lors des phases intermédiaires. En effet, les animations proposent par défaut un démarrage progressif et un final adouci, de la même façon que les transitions.

Nous avons utilisé la propriété `animation` sans l'avoir vraiment étudiée. Il s'agit en fait d'un raccourci de plusieurs propriétés qui nous sembleront familières, car elles ressemblent à celles utilisées pour les transitions.



FIGURE 7-42 Les différentes étapes de l'animation présentée comme exemple, utilisée ici avec « *Tux GNUsquetaire* » (image créée par André Pascual et provenant du site <https://abul.org>)

ATTENTION Préfixes des navigateurs avec @keyframes

Pour les navigateurs qui ont une implémentation provisoire de ces propriétés d'animation, il ne faudra pas oublier d'ajouter les préfixes :

- d'une part, pour la propriété `animation` :

-moz-animation, -webkit-animation, ... ;
- d'autre part pour la règle `@keyframes`, qui devra être répétée entièrement avec chaque préfixe placé entre l'arobase et le mot `keyframes`, soit :

`@-moz-keyframes { ... }`

```
@-webkit-keyframes { ... }  
etc.
```

Définition du nom de l'animation

La propriété `animation-name` permet de choisir le nom de l'animation à utiliser, qui aura bien sûr été défini au préalable, à l'aide de la fonction `@keyframes`.

TABLEAU 7–51 Propriété `animation-name`

Propriété	<code>animation-name</code>
Exemple	<code>img.logo { animation-name: apparition; }</code>
Valeurs possibles	none (aucune animation, valeur par défaut) ou le nom d'une animation qui a été définie par <code>@keyframes</code> . Éventuellement, plusieurs noms d'animation peuvent être indiqués, séparés par des virgules.
Héritage	<code>Non</code>

Plusieurs animations peuvent être choisies en même temps ; il suffit de noter leurs noms, séparés par des virgules, dans la propriété `animationname`. Dans ce cas, les autres propriétés d'animation auront elles aussi plusieurs valeurs séparées par des virgules, qui correspondront dans l'ordre aux noms d'animations ainsi définis.

Durée de l'animation

La propriété `animation-duration` définit la durée totale de l'animation.

TABLEAU 7–52 Propriété `animation-duration`

Propriété	<code>animation-duration</code>
Exemple	<code>img.logo { animation-duration: 5s; }</code>
Valeurs possibles	0 (valeur par défaut, aucune animation) ou une durée en secondes suivie sans espace de la lettre « s ». Les valeurs décimales utilisent le point, comme <code>3.5s</code> pour 3,5 secondes. S'il y a plusieurs animations, plusieurs durées respectives peuvent leur correspondre, séparées par des virgules.

Héritage	<i>Non</i>
-----------------	------------

Progression de l'animation

Nous retrouvons la progression dans le temps : c'est exactement la même que pour les transitions, et elle est ici déterminée par la propriété `animation-timing-function`.

TABLEAU 7–53 Propriété `animation-timing-function`

Propriété	<code>animation-timing-function</code>
Exemple	<code>img.logo { animation-timing-function: linear; }</code>
Valeurs possibles	<p><code>linear</code> (linéaire, vitesse constante) <code>ease</code> (valeur par défaut, démarrage progressif mais rapide avec un ralentissement à la fin) <code>ease-in</code> (lent au début, rapide ensuite) <code>ease-out</code> (rapide dès le début et ralenti à la fin) <code>ease-in-out</code> (démarrage et fin en douceur, vitesse augmentée entre les deux) <code>cubic-bezier(x1,y1,x2,y2)</code> (paramétrage de la courbe à partir des coordonnées de deux points intermédiaires, indiqués sur la figure 7-43) <code>step-start</code> ou <code>step-end</code> (pas de transition, valeur finale attribuée instantanément au début ou à la fin de la durée de la transition) <code>steps(n,start)</code> ou <code>steps(n,end)</code> (transitions par <code>n</code> sauts successifs, avec <code>start</code> le premier saut dès le début de la transition et <code>end</code> le dernier saut à la fin de la transition, comme le montre la figure 7-41) Plusieurs valeurs, séparées par des virgules, pourront correspondre à différentes animations choisies simultanément.</p>
Héritage	<i>Non</i>

La figure 7-43 explique les fameuses courbes de Bézier, du nom de l'ingénieur français Pierre Bézier qui les a décrites : les points d'arrivée et de départ ayant pour coordonnées respectives (0,0) et (1,1), la fonction `cubic-bezier(x1,y1,x2,y2)` doit contenir les coordonnées des deux points intermédiaires, qui définissent les vitesses de départ et de fin de la transition.

Là encore, les valeurs prédefinies comme `ease-out` ou `linear` nous suffiront pour une utilisation courante.

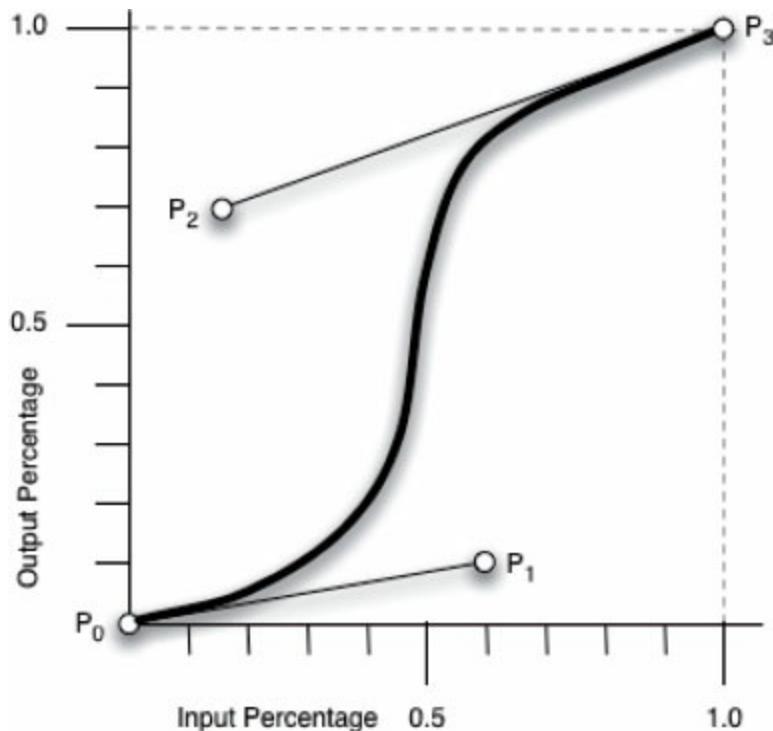


FIGURE 7–43 Extraite du site www.w3.org, l'explication de la courbe de Bézier, avec les deux points P_1 et P_2 dont il faut fournir les coordonnées. Dans cet exemple, la vitesse est un peu moins élevée au début et à la fin par rapport au reste de l'animation.

Délai avant le démarrage de l'animation

Un délai d'attente avant le démarrage de l'animation peut être paramétré à l'aide de la propriété `animation-delay`.

TABLEAU 7–54 Propriété `animation-delay`

Propriété	<code>animation-delay</code>
Exemple	<code>div#bandeau { animation-delay: 2.5s; }</code>
Valeurs possibles	0 (valeur par défaut, pas de retard au démarrage) un délai indiqué en secondes et suivi par le symbole « s » sans espace avant (si nécessaire, le séparateur décimal est le point) Si plusieurs délais sont séparés par des virgules, ils correspondent respectivement aux différentes animations choisies.

Héritage	<i>Non</i>
-----------------	------------

Après ces quelques propriétés qui ont pour nous un air de déjà-vu tant elles ressemblent à celles qui gèrent les transitions, nous allons en découvrir d'autres qui sont spécifiques.

Répétition de l'animation

L'animation se déroule normalement une seule fois, sauf si un nombre de répétitions est fixé par la propriété `animation-iteration-count`. L'option `infinite` permet même de ne jamais arrêter l'animation.

TABLEAU 7–55 Propriété `animation-iteration-count`

Propriété	<code>animation-iteration-count</code>
Exemples	<code>img.logo { animation-iteration-count: 3; }</code> <code>#bandeau { animation-iteration-count: infinite; }</code>
Valeurs possibles	1 (valeur par défaut, l'animation ne se produit qu'une seule fois) un nombre quelconque d'itérations (s'il est décimal, l'animation s'arrête en cours de route, puis passe directement à l'aspect final) <code>infinite</code> pour une répétition permanente Lorsque plusieurs animations ont été sélectionnées, il est possible de spécifier autant de nombres de répétitions différents, séparés par des virgules.
Héritage	<i>Non</i>

Sens de l'animation

Lorsque l'animation est répétée, elle se déroule en principe dans l'ordre indiqué par la définition `@keyframes`, sauf si la propriété `animation-direction` précise qu'il faut changer de sens une fois sur deux.

Cela revient alors à effectuer des « allers et retours » dans l'animation, chaque aller et chaque retour étant compté comme une itération. Si la répétition se termine par un retour, l'élément repasse instantanément à sa forme finale à la fin de l'animation.

TABLEAU 7–56 Propriété `animation-direction`

Propriété	<code>animation-direction</code>
------------------	----------------------------------

Exemples	<code>img.logo { animation-direction: normal; } #bandeau { animation-direction: alternate; }</code>
Valeurs possibles	<p>normal (valeur par défaut) si l'animation doit toujours recommencer à partir de sa forme initiale</p> <p>reverse pour une animation dans le sens inverse, par rapport à celui défini initialement par la règle @keyframes</p> <p>alternate si les itérations paires doivent s'effectuer dans le sens contraire de celui défini par @keyframes</p> <p>alternate-reverse si ce sont les itérations impaires qui doivent s'effectuer dans le sens contraire de celui défini par @keyframes</p> <p>Plusieurs directions peuvent être indiquées, séparées par des virgules, lorsque différentes animations ont été choisies.</p>
Héritage	Non



FIGURE 7-44 Animation d'un bloc de texte en CSS 3, d'après la page https://www.w3schools.com/css/css3_animations.asp

Forme initiale et forme finale de l'élément animé

Nous allons affiner notre découverte des animations, en nous demandant ce qui se passe *avant* et *après* l'animation. En effet, l'élément qui est animé peut, en même temps, être mis en forme par une feuille de styles et posséder des propriétés dont certaines sont modifiées par l'animation.

Dans l'exemple qui correspond à la figure 7-42, l'animation modifie la largeur de l'image, qui passe de 0 à 250 pixels. Que se passe-t-il si, par ailleurs, cette image possède la propriété `width: 50px`? Par défaut, l'affichage est le suivant.

1. Avant l'animation, l'image possède la taille indiquée dans la feuille de styles, soit une largeur de 50 pixels. Cet état ne sera visible que si un

délai est appliqué avant le début de l'animation.

2. Pendant l'animation, sa largeur passe de 0 à 250 pixels ; l'image disparaît donc au début, si la présence d'un délai d'attente nous a permis de la voir avec sa largeur initiale de 50 pixels.
3. Après l'animation, l'image voit sa largeur passer instantanément de 250 à 50 pixels, pour s'accorder avec le style fixe qui a été défini.

La propriété `animation-fill-mode` permet de modifier les états initial et final de l'objet. Dans notre exemple :

- `animation-fill-mode="forwards"` conservera une largeur d'image de 250 pixels à la fin de l'animation ;
- `animation-fill-mode="backwards"` masquera l'image (en lui attribuant la largeur du début de l'animation, soit 0 pixels) dès l'affichage de la page, même si l'animation ne doit démarrer qu'après un délai d'attente ;
- `animation-fill-mode="both"` combinera les deux modes précédents ; la largeur de 50 pixels ne sera donc plus prise en compte, et elle sera de 0 pixels au début et de 250 pixels à la fin.

TABLEAU 7–57 Propriété `animation-fill-mode`

Propriété	<code>animation-fill-mode</code>
Exemples	<pre>img.logo { animation-fill-mode: both; } #bandeau { animation-fill-mode: forwards; }</pre>
Valeurs possibles	<p><code>none</code> (valeur par défaut) : avant et après l'animation, l'élément animé conserve les propriétés qu'il avait dans la feuille de styles, si elles sont en contradiction avec celles de l'animation (les propriétés de début et fin de la règle <code>@keyframes</code> ne sont prises en compte que si elles n'ont pas été définies par ailleurs).</p> <p><code>forwards</code> : après l'animation, l'élément conserve les propriétés finales qui lui ont été attribuées par la règle <code>@keyframes</code>, dans l'étape <code>to {...} ou 100% {...}</code>.</p> <p><code>backwards</code> : avant le début de l'animation (si un délai d'attente a été défini), l'objet prend les propriétés définies pour l'instant initial dans la règle <code>@keyframes</code>, à l'étape <code>from {...} ou 0% {...}</code>.</p> <p><code>both</code> : l'élément utilise toujours les propriétés initiales et finales qui lui ont été attribuées par la règle <code>@keyframes</code>, même</p>

	si une règle de style les contredit par ailleurs. Comme d'habitude, lorsque différentes animations ont été sélectionnées, plusieurs valeurs peuvent être indiquées, séparées par des virgules.
Héritage	<i>Non</i>

Suspension de l'animation

Il existe une propriété `animation-play-state` qui permet de mettre en pause l'animation, par exemple à l'aide d'un bouton auquel nous aurons affecté une fonction JavaScript modifiant cette valeur.

TABLEAU 7–58 Propriété `animation-play-state`

Propriété	<code>animation-play-state</code>
Exemples	<code>img.logo { animation-play-state: running; }</code> <code>#bandeau { animation-play-state: paused; }</code>
Valeurs possibles	<code>running</code> (valeur par défaut) si l'animation doit continuer à s'exécuter <code>paused</code> pour interrompre momentanément l'animation Plusieurs états peuvent être indiqués, séparés par des virgules, pour contrôler le cas échéant les différentes animations choisies.
Héritage	<i>Non</i>

Raccourci pour les animations

Nous terminons par la propriété raccourcie `animation`, avec laquelle nous avions commencé à travers quelques exemples de présentation. C'est celle que nous utiliserons le plus souvent, puisqu'elle permet d'appliquer une animation en écrivant une seule ligne. Elle reprend en effet toutes les propriétés précédentes, sauf `animation-play-state` dont l'usage sera moins courant.

TABLEAU 7–59 Propriété raccourcie `animation`

Propriété	<code>animation</code>
Exemple	<code>img.logo { animation: apparition 5s linear 0.5s 3 alternate both; }</code>

Valeurs possibles	Valeurs successives, séparées par des espaces, des propriétés : animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction et animation-fill-mode Les propriétés qui ne sont pas indiquées prennent leur valeur par défaut et seule la deuxième durée correspond au délai d'attente. La valeur qui n'est pas suivie par la lettre « s », comme secondes, est le nombre d'itérations (nombre de répétitions de l'animation). Plusieurs groupes de valeurs peuvent se succéder, séparés par des virgules, pour utiliser simultanément différentes animations.
Héritage	<i>Non</i>

RÉFLEXION Utilisation des animations pour le Web

Comme nous l'avons vu, un des dangers des transitions et des animations est de ralentir la consultation des pages, mais ce n'est pas le seul. Si l'emploi de ces propriétés est un atout esthétique incontestable, il nous faut les utiliser avec suffisamment de finesse pour éviter de tomber dans les excès qui nous guettent : les pages qui bougent dans tous les sens risquent de manquer de clarté. Pire encore, le message véhiculé par le site peut manquer de crédibilité si trop d'effets spéciaux lui donnent un aspect « amateur ».

Par conséquent, tout comme dans l'emploi des polices, des couleurs, des images de fond et des ombres, ce sont la sobriété et la discrétion des effets qui donneront de l'allure à nos sites.

Bien sûr, il ne faudra pas oublier de déclarer l'animation utilisée, en lui attribuant un nom et un fonctionnement à l'aide de la règle `@keyframes` que nous avons étudiée au début de notre découverte des animations.

Nouveaux types de sélecteurs

Après avoir exploré les propriétés avancées de mise en forme, parcourons les nouveautés dans l'emploi des sélecteurs. Leur effet est moins spectaculaire que les transformations graphiques, mais ce sont des outils très utiles. Les CSS 2 permettaient déjà de sélectionner des éléments en fonction du contenu d'un de leurs attributs ou de leur imbrication dans un autre élément. Ces choix sont complétés et affinés, grâce à l'apparition de nouvelles conventions d'écriture.

Sélecteur de voisinage

Le tilde (~) indique le voisinage entre deux balises : cela signifie qu'elles ont le même bloc parent. Par exemple, pour ajouter une marge aux images qui côtoient une liste , il suffira d'écrire :

```
ul~img { margin: 15px; }
```

Sélection sur les attributs

Il est à présent possible de sélectionner des éléments en fonction d'une partie du contenu d'un de leurs attributs.

Attribut existant ou ayant une valeur donnée

Existence d'un attribut

Le simple nom d'un attribut, placé entre crochets, permet de tester son existence pour la balise HTML indiquée. Voici par exemple une règle de style qui écrit en vert tous les liens hypertextes du type ``, mais pas les ancrées `` qui servent à repérer un paragraphe dans une page :

```
a[href] { color: green; }
```

Une des valeurs d'un attribut égale à la valeur proposée

L'attribut d'une balise peut être égal à un ensemble de mots séparés par des espaces, par exemple `class="note bulle"` ou `title="fleur jaune"`. Il est alors possible de tester la présence d'un mot parmi ceux qui composent l'attribut désigné. Ce test est différent de celui d'une valeur entière d'attribut, comme `img[src="logo.png"]` pour désigner toutes les images affichant le fichier `logo.png`. La syntaxe suit la même structure, le signe égal étant simplement précédé d'un tilde (`~`), comme dans la règle qui suit :

```
p[title~="sport"] { font-style: italic; }
```

Seront ici écrits en italique les paragraphes possédant un attribut `title` dont la valeur contient le mot fourni, comme `<p title="sport">` et `<p title="sport nautique">`.

Valeur d'attribut exacte ou suivie d'un tiret

La syntaxe précédente, reprise cette fois avec une barre verticale précédant le signe égal (`|=`) sert à sélectionner les balises dont l'attribut est :

- soit exactement égal à la valeur donnée ;
- soit égal à cette valeur suivie d'un tiret (-) puis d'un contenu

quelconque.

La mise en forme suivante :

```
| p[class+="fr"] { color: blue; }
```

écrira en bleu les paragraphes :

```
| <p class="fr"> ... </p>
| <p class="fr-France"> ... </p>
| <p class="fr-Canada"> ... </p>
```

mais n'affectera pas ces deux paragraphes :

```
| <p class="français"> ... </p>
| <p class="fr France"> ... </p>
```

Attribut sélectionné sur une partie de son contenu

Attribut commençant par...

Le caractère chapeau (^) signifie « commence par ». Ainsi, la règle de style suivante :

```
| a[href^="piscine"] { font-weight: bold; }
```

mettra en gras les balises `<a>` dont l'attribut `href` commence par le texte *piscine*, comme les deux suivantes :

```
| <a href="piscine-ronde.html">
| <a href="piscines-de-nage.html">
```

mais pas cette troisième :

```
| <a href="petite-piscine.html">
```

Attribut se terminant par...

Le caractère dollar (\$) veut dire « se termine par », comme dans la règle de style :

```
| h3[title$="plage"] { color: blue; }
```

qui écrira en bleu les titres `<h3>` possédant un attribut `title` dont la valeur se termine par *plage*, comme dans cet exemple :

```
| <h3 title="drap de plage">  
| <h3 title="jeu de plage">
```

mais pas celui-ci :

```
| <h3 title="une plage tranquille">
```

Attribut contenant...

L'étoile (*) permet de rechercher une chaîne de caractères à l'intérieur de l'attribut spécifié. C'est pourquoi la règle de style suivante :

```
| p[class*="rose"] { font-size="120%"; }
```

grossira les caractères des paragraphes `<p>` auxquels est attribuée une classe contenant le terme *rose*, comme les deux suivants :

```
| <p class="plante rose promotion">  
| <p class="plante roseau">
```

mais pas ce dernier :

```
| <p class="fleur tulipe jaune">
```

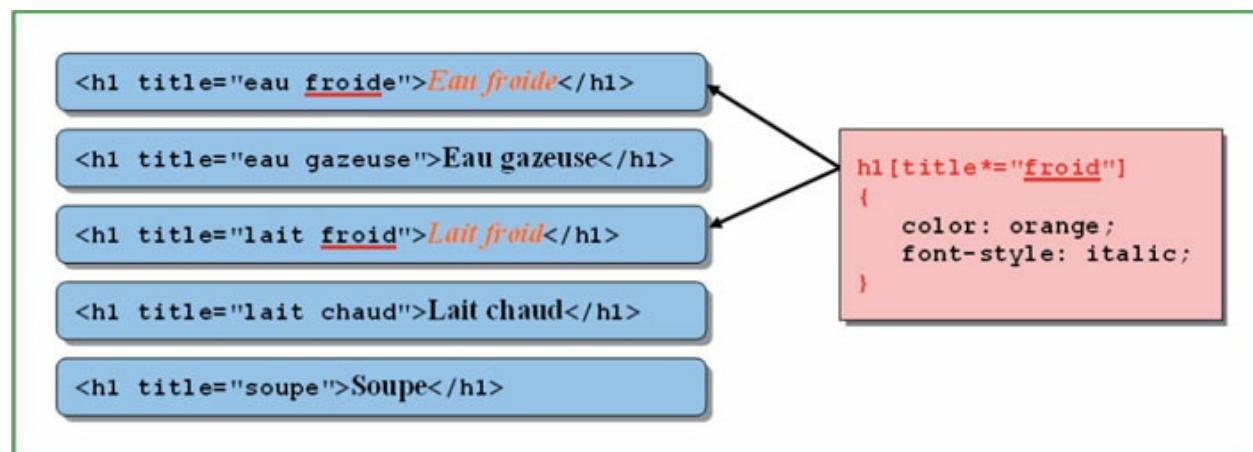


FIGURE 7–45 En CSS 3, nous pouvons sélectionner des balises en fonction d'une partie d'un attribut. Cela permet d'utiliser moins de classes, et le code s'en trouve allégé.

Pseudo-classes

Les pseudo-classes servent à englober un certain nombre d'éléments qui ont un point commun. Nous connaissons déjà celles qui sont souvent associées aux liens hypertextes : `:link` (lien ordinaire), `:visited` (lien déjà visité), `:active` (lien actif) et `:hover` (élément survolé par le curseur de la souris).

Pour nous permettre de repérer plus aisément nos balises, sans avoir à leur attribuer des classes en surnombre, la norme CSS 3 ajoute quelques nouvelles pseudo-classes.

Élément ou attribut différent de...

La pseudo-classe `:not([attribut="..."])` sert à écarter les éléments qui n'ont pas l'attribut indiqué. Ainsi, les règles de styles :

```
p:not([class="note"]) { font-family: Arial, sans-serif; }
a:not([class*="menu"]) { text-decoration: underline; }
```

écriront en police Arial tous les paragraphes `<p>`, sauf ceux de classe *note* `<p class="note">` et souligneront tous les liens `<a>` dont la classe ne contient pas le texte *menu*.

D'une manière plus générale, la pseudo-classe `:not(...)` peut être utilisée pour indiquer le contraire de tout sélecteur CSS. Par exemple, pour mettre sur fond blanc tous les éléments de la page sauf les titres `<h1>`, il faudra écrire :

```
:not(h1) { background-color: white; }
```

Ce type de règle peut parfois être formulé de différentes façons. Les deux exemples qui suivent permettent d'écrire en gras toutes les ancre `<a>` servant de points de repère (du type ``) et ne correspondant pas à un lien hypertexte (donc différentes du type ``) :

```
a:not([href]) { font-weight: bold; }
:not(a[href]) { font-weight: bold; }
```

Pseudo-classes pour les éléments de formulaire

En ce qui concerne les formulaires, les CSS 2 acceptent déjà la pseudoclasse

`:focus` permettant de repérer l'élément actif ou qui reçoit le curseur, et qui peut être combiné avec `:hover` (élément sélectionné et sur lequel passe le curseur de la souris) :

```
select:focus { background-color: yellow; }
select:focus:hover { background-color: orange; }
```

Les CSS 3 ajoutent quelques autres possibilités pour repérer certains états d'éléments de formulaire.

Éléments cochés

La pseudo-classe `:checked` permet de mettre en forme des cases à cocher et des boutons radio, une fois qu'ils sont validés par l'utilisateur. Par exemple, la règle de style qui suit les encadre en rouge dans ce cas :

```
input:checked { border: 1px solid red; }
```

Éléments activés ou désactivés

Les pseudo-classes `:enabled` et `:disabled` servent à repérer les éléments qui sont respectivement activés (état par défaut) ou désactivés (possédant l'attribut `disabled`). En voici deux applications pour des zones de texte à plusieurs lignes, avec un fond blanc pour celles qui sont activées et un fond gris lorsqu'elles sont désactivées :

```
textarea:enabled { background-color: white; }
textarea:disabled { background-color: gray; }
```

Distinction des éléments inclus dans un bloc

Dernier enfant d'un bloc

La pseudo-classe `:last-child` s'utilise comme celles existant déjà, `:first-child` (premier enfant d'un bloc) ou `p:first-child` (tout paragraphe `<p>` qui est le premier enfant d'un conteneur). Comme son nom l'indique, la différence est qu'il s'agit du dernier enfant au lieu du premier. Par exemple, la mise en italique du dernier élément de chaque liste à puces `` et l'encadrement du dernier paragraphe `<p>` de chaque bloc `<div>` s'écrivent respectivement :

```
ul :last-child { font-style: italic; }
div p:last-child { border: solid 1px; }
```

Notez que cette dernière règle de style encadrera aussi tous les paragraphes `<p>` qui sont chacun le dernier élément d'une balise quelconque imbriquée dans un bloc `<div>`. Pour que ce soit le dernier paragraphe enfant *direct* de chaque bloc `<div>`, il faudrait écrire :

```
div > p:last-child { ... }
```

Premier ou dernier enfant d'un type donné

Pour repérer un élément qui est, à l'intérieur de son conteneur, le premier ou le dernier enfant d'un type donné, il existe deux pseudo-classes. Ce sont `:first-of-type` et `:last-of-type`.

Pour mettre en gras respectivement le premier et le dernier paragraphe `<p>` du bloc `<div class="menu">`, nous écrirons :

```
div.menu p:first-of-type { font-weight: bold; }
div.menu p:last-of-type { font-weight: bold; }
```

Dans le premier cas, il se peut que le paragraphe en question ne soit pas le premier enfant de ce bloc `<div>` ; le premier enfant sera un paragraphe `<p>`. De même pour l'autre exemple : le dernier enfant de type `<p>` n'est pas nécessairement le dernier enfant tout court.

Unique enfant dans un bloc

Deux pseudo-classes servent à repérer un enfant unique dans un conteneur : `:only-child` si c'est absolument le seul enfant de ce conteneur, `:only-of-type` si c'est le seul enfant qui est du type indiqué.

Par exemple, les règles de style suivantes donnent un fond gris à un paragraphe `<p>` contenu dans un bloc `<div>` :

```
div p:only-child { background-color: gray; }
div p:only-of-type { background-color: gray; }
```

La première règle concerne tout paragraphe `<p>` inclus dans un bloc `<div>` qui ne contient pas d'autre enfant. La deuxième ligne attribuera au paragraphe ce fond gris si son conteneur `<div>` ne contient pas d'autre paragraphe `<p>`, ce bloc `<div>` pouvant néanmoins avoir d'autres enfants, comme une liste `` ou des images par exemple.

Éléments enfants comptés à partir du début

Deux pseudo-classes permettent de sélectionner un ou plusieurs élément(s) enfant(s) d'un conteneur, en les comptant à partir du début, `:nth-child(...)` et `:nth-of-type(...)`. La première compte toutes les balises enfants du conteneur, alors que la deuxième ne dénombre que celles qui sont du type indiqué. Un exemple est plus simple à comprendre :

```
div#page p:nth-child(3) { color: green; }
form input:nth-of-type(5) { border: solid red 1px; }
```

La première ligne écrit en vert la troisième balise enfant du bloc `<div>` d'identifiant `page`, s'il s'agit d'un paragraphe, les deux balises précédentes pouvant être d'un autre type que `<p>`. Si ce troisième enfant n'est pas un paragraphe `<p>`, cette règle n'a aucun effet.

En revanche, la deuxième règle encadre en rouge la cinquième des balises de type `<input>` contenues dans un formulaire `<form>` ; les balises d'un autre type que `<input>` ne seront pas comptées.

Éléments enfants comptés à partir de la fin

Les pseudo-classes `:nth-last-child(...)` et `:nth-last-of-type(...)` fonctionnent de la même manière que les deux précédentes `:nth-child(...)` et `:nth-of-type(...)`, mais en comptant les balises à partir de la fin.

Exemple de comptage à partir de la fin :

```
div p:nth-last-child(3) { background-color: gray; }
div :nth-last-child(3) { background-color: gray; }
ul#menu li:nth-last-of-type(2) { font-style: italic; }
```

La première règle de style affiche sur fond gris la troisième balise du bloc `<div>` en partant de la fin, si c'est un paragraphe `<p>`. La deuxième ligne effectue le même traitement, mais dans tous les cas, y compris si la troisième balise en partant de la fin du `<div>` est d'un autre type que `<p>` (par exemple, cela peut être une balise `<h3>`).

La troisième règle écrit en italique l'avant-dernière des lignes `` d'une liste à puces `` d'identifiant `menu`.

Balises repérées par leur contenu ou leur fonction

Les CSS 3 proposent quelques pseudo-classes qui permettent de sélectionner des éléments en fonction de leur contenu ou de leur particularité dans la page.

Balises vides

Il s'agit de repérer les balises qui n'ont aucun contenu, à l'aide de la pseudoclasse `:empty`. Ainsi, la ligne suivante permettra d'annuler les marges des blocs `<div>` qui sont vides :

```
div:empty { margin: 0; padding: 0; }
```

GÉNÉRALISATION Valeurs et expressions calculées pour le comptage des balises

Les pseudo-classes qui comptent le nombre de balises enfants comme `:nth-child(...)`, `:nth-of-type(...)`, `:nth-last-child(...)` et `:nth-last-of-type(...)` acceptent différents types de valeurs entre les parenthèses.

- Un nombre, pour indiquer le numéro d'enfant à sélectionner, en partant de 1.
- Un mot-clé comme `odd` ou `even`, qui signifient respectivement *impair* et *pair*.
- Une expression calculée du type `an+b`, où `a` et `b` sont des nombres entiers et la lettre `n` s'écrit telle quelle : par exemple, `:nth-child(2n)` représente les enfants pairs (équivalent de `even`), `:nth-child(2n+1)` les enfants impairs (équivalent de `odd`), `:nth-child(10n)` les enfants des dizaines entières (numéros 10, 20, 30, etc.). Les nombres `a` et `b` sont des entiers qui peuvent être positifs, négatifs ou nuls.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

FIGURE 7–46 Mise en forme à partir du comptage d’éléments : les nombres impairs de ce tableau ont un fond de couleur, ceux correspondant aux dizaines entières sont en gras, écrits en rouge et de taille plus grande.

Élément racine

Nous aurons peu l’occasion d’utiliser la pseudo-classe :root qui correspond à la racine des balises, car celle-ci est tout simplement l’élément <html>.

Ancre ou cible d’un lien

Un lien interne s’écrit suivant l’exemple , qui sert à accéder directement à un bloc ayant l’identifiant indiqué, comme un titre <h2 id="sommaire"> ou une ancre . Le lien peut également s’effectuer à partir d’une autre page, avec une forme comme celle-ci :

```
<a href="rapport.html#sommaire">Sommaire du rapport</a>
```

Dans les deux cas, le dièse (#) s’affiche dans la barre d’adresse du navigateur et l’élément en question se place en haut de l’écran (sauf s’il se trouve vers la fin de la page).

Pour repérer cet élément pointé par le symbole dièse (#) de la barre d’adresse, la pseudo-classe :target nous permet de lui donner une mise en forme spécifique :

```
| h2:target { color: red; }
```

Dans cet exemple, le titre `<h2>` qui est la cible de l'adresse vient s'écrire en rouge.

Un autre exemple permet d'afficher l'image d'une flèche à gauche de ce titre :

```
| h2:target::before { content: url(images/fleche.png); }
```

Cela suppose bien sûr qu'un fichier nommé `fleche.png` se trouve placé dans le sous-dossier `image`. Vous avez noté que la règle de style possède à présent une nouvelle écriture, avec deux caractères « deux-points » avant `before` au lieu d'un seul, l'ancienne forme restant néanmoins prise en compte :

```
| h2:target:before { content: url(images/fleche.png); }
```

Pseudo-élément

Ce titre est au singulier, car il n'existe qu'un seul nouveau pseudo-élément en CSS 3, et c'est celui qui prend en compte la partie sélectionnée d'un texte. Il s'agit du pseudo-élément `:selection`, qui s'écrit à présent `::selection` avec la nouvelle notation utilisant deux caractères « deuxpoints ». Dans la pratique, les deux écritures sont reconnues, soit par exemple :

```
p::selection { color: red; background-color: yellow; }
p:selection { color: red; background-color: yellow; }
```

ASTUCE Nouveaux sélecteurs CSS3 et anciens navigateurs

Nombre de ces sélecteurs introduits en CSS 3 ne sont pas compris par d'anciens navigateurs, en particulier par les versions 8 et précédentes d'Internet Explorer. Pour qu'ils soient reconnus par ces versions, il faudra ajouter un script tel que Selectivizr, associé par exemple à la bibliothèque jQuery, ou mieux au moteur JavaScript NWMatcher, spécialisé dans les sélecteurs. Ces deux utilitaires sont respectivement disponibles aux adresses suivantes :

- ▶ <http://selectivizr.com>
- ▶ <http://javascript.nwbox.com/NWMatcher>

Les CSS 3 : déjà utilisées et toujours en évolution

Comme indiqué en remarque au tout début de ce chapitre, ces fonctions proposées par la norme CSS 3 sont prises en compte progressivement par les navigateurs. Nombre d'entre elles sont à présent reconnues sous leur forme standard ; mais pour certaines, les préfixes sont nécessaires. Suivant la propriété, il faudra parfois l'écrire plusieurs fois comme indiqué au début de ce chapitre, d'abord avec les préfixes `-moz-`, `-webkit-` et `-ms-`, puis selon sa syntaxe initiale, ou utiliser l'alternative JavaScript « prefixfree » qui ajoutera pour nous ces différents préfixes.

En ce qui concerne la compatibilité de nos pages, une bonne partie des propriétés que nous venons de parcourir pourra heureusement s'utiliser de telle sorte que nos pages puissent être affichées correctement par tous les butineurs, les règles de styles inconnues étant simplement ignorées. Dans ce cas, le site ne s'affichera pas exactement de la même façon pour toutes les versions de tous les logiciels : cela s'appelle une « dégradation élégante » sur les anciennes versions, et c'est une méthode couramment admise à présent.

Les feuilles de styles CSS 3 peuvent être validées de façon très simple, en copiant-collant des règles à tester dans une zone de saisie disponible à l'adresse suivante : http://jigsaw.w3.org/css-validator/#validate_by_input.



FIGURE 7–47 Une galerie d’images en pur CSS, créée à l’aide de transformations 3D, extraite du site de Sandro Paganotti : démonstration sur <http://sandropaganotti.com/wp-content/goodies/demos/cards/> et explications des techniques via le lien « tech info » de cette page



FIGURE 7–48 Cette autre galerie d’images en 3D utilise les transformations CSS 3 pour plaquer les photos sur les faces d’un cube et découper celui-ci en trois parties afin d’assurer les transitions (démonstration sur la page <https://tympanus.net/Development/Slicebox/index2.html>).

chapitre 8

Exemples de sites web



En guise de synthèse de ce que nous avons vu précédemment, nous allons réaliser des pages web complètes, en partant d'une base commune et en utilisant différentes variantes pour la mise en place des blocs qui les constituent : menu horizontal ou vertical, éléments de couleur unie ou dégradée, associés ou non à une image de fond.

SOMMAIRE

- ▶ **Structure d'une page web**
- ▶ **Code HTML de base**
- ▶ **Créer des pages de base à menu horizontal ou vertical**
- ▶ **Exemples concrets avec images de fond et dégradés CSS 3**

Avec les techniques que nous avons étudiées, balises HTML et feuilles de styles CSS, nous avons en main tous les éléments nécessaires à la construction d'un site. Il nous reste à les assembler pour obtenir une page qui ressemble à un site Internet. Nous allons donc étudier quelques cas pratiques, qui vous serviront peut-être de bases pour vos futures pages web.

Structure d'une page web

La présentation d'une page web peut être très différente d'un site à l'autre, suivant sa complexité et son originalité. Cependant, il existe des éléments qui se retrouvent dans toutes les pages, comme le montre la figure 8-1 :

- un en-tête, comprenant souvent un logo et le titre du site ;
- un menu qui peut être horizontal ou vertical, les deux étant parfois utilisés en même temps ;
- le contenu spécifique à la page, placé sur une ou plusieurs colonne(s) ;
- un pied de page regroupant des indications ou menus complémentaires, des mentions légales, un rappel d'adresse, etc.

Pour délimiter ces différentes parties, nous utiliserons les balises adaptées du HTML 5 : `header`, `nav`, `section` et `footer`.

CHOIX Menu horizontal ou vertical ?

Ces deux types de menu ont chacun des avantages et des inconvénients.

- Un menu horizontal est assez visible, mais limité à la place disponible en largeur. Cependant, il est possible d'adopter des listes déroulantes, la liste horizontale représentant alors la structure générale du site. Le contenu de la page, situé au-dessous, peut alors occuper toute la largeur disponible et être présenté sur plusieurs colonnes.
- Le menu vertical a pour avantage d'être extensible. Par ailleurs, si le contenu de la page est écrit sur une seule colonne, celle-ci s'en trouve réduite en largeur, ce qui facilite la lecture en évitant des lignes trop longues. Si ce menu comporte beaucoup de liens, il est préférable de les regrouper par catégories, de façon à conserver une bonne lisibilité.
- Les sites sophistiqués peuvent proposer les deux à la fois : soit pour un accès suivant deux classements différents, soit un menu horizontal pour les grands thèmes et un autre vertical pour les détails du thème choisi. Cette dernière utilisation est une

alternative aux menus déroulants.



FIGURE 8–1 Les grandes parties qui constituent généralement une page web (extrait du site <http://www.w3c-schools.com/>)

Nous allons étudier la mise en place de ces deux méthodes de navigation ; vous n'aurez plus qu'à choisir celle qui vous convient. Nous nous hasarderons ensuite à utiliser des images de fond pour embellir notre site.

Code HTML de base

Nos modèles de base comprendront toujours la même structure HTML, avec les éléments essentiels que nous venons d'évoquer : un en-tête, un menu, un contenu et un pied de page, le tout formant une page de largeur fixe qui sera centrée au milieu de l'écran.

Examinons tout d'abord le code HTML de la page de base. Grâce au potentiel des feuilles de styles, il servira à la fois pour les versions avec menu horizontal et avec menu vertical.

IMPORTANT Nom de la page d'accueil

Rappelons que notre page d'accueil doit obligatoirement s'appeler `index.html`. Le nom des autres pages est libre ; il est simplement conseillé de se limiter aux lettres, chiffres et tirets, sans espaces, ni accents, ni caractères particuliers comme le ç.

Si une page nommée par exemple `accueil.html` peut s'afficher correctement, il faut se souvenir que lorsqu'un visiteur tape le nom d'un site dans son navigateur, il ne précise pas le nom du fichier : c'est donc la page `index` qui apparaît. Si elle n'est pas présente, c'est la liste brute des fichiers du répertoire qui s'affiche à l'écran.

Notre feuille de styles est appelée ici, de façon très banale, `style.css`. Tant que la page n'a pas été définie, son affichage présente l'aspect austère et très rudimentaire que vous pouvez voir sur la figure 8-2.

Logo et en-tête de la page.

- [Page d'accueil](#)
- [Lien 2](#)
- [Lien 3](#)
- [Lien 4](#)
- [Lien 5](#)

Titre général

Premier sous-titre

Paragraphe de texte. ...

Paragraphe de texte. ...

Deuxième sous-titre

Paragraphe de texte. ...

Contenu du pied de page

FIGURE 8–2 L'aspect brut de notre page de base, sans feuille de styles

Code HTML de la trame avec menu horizontal ou vertical

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Modèle de page web</title>
    <link rel="stylesheet" href="style.css"> ①
</head>

<body>
<div class="global"> ② <!-- Conteneur général -->

    <header class="entete"> ③ <!-- En-tête -->
        <p>Logo et en-tête de la page</p>
    </header> <!-- Fin d'en-tête -->
```

```

<nav class="blocmenu"> ④ <!-- Menu de navigation -->
  <ul id="menu"> ⑤
    <li><a href="index.html">Page d'accueil</a></li>
    <li><a href="page2.html">Lien 2</a></li>
    <li><a href="page3.html">Lien 3</a></li>
    <li><a href="page4.html">Lien 4</a></li>
    <li><a href="page5.html">Lien 5</a></li>
  </ul>
</nav> <!-- Fin du menu de navigation -->

<section class="contenu"> ⑥ <!-- Contenu de la page -->
  <h1>Titre général</h1>

  <h2>Premier sous-titre</h2>
  <p>Paragraphe de texte... </p>
  <p>Paragraphe de texte... </p>

  <h2>Deuxième sous-titre</h2>
  <p>Paragraphe de texte... </p>
</section> <!-- Fin du contenu de la page -->

<footer class="pied"> ⑦ <!-- Pied de page -->
  <p>Contenu du pied de page</p>
</footer> <!-- Fin du pied de page -->
</div> <!-- Fin du conteneur général -->
</body>
</html>

```

Nous ne nous étendrons pas sur l'en-tête, dont l'écriture a été détaillée dans le chapitre 2. Seule la balise `<link>` ① retiendra notre attention, puisqu'elle donne le nom de la feuille de styles utilisée, ici `style.css`. Ce nom n'est pas obligatoire, seule l'extension `.css` est indispensable. Une fois que nous aurons compris la structure de ce code HTML, c'est principalement ce fichier `style.css` que nous étudierons et que nous modifierons pour créer différentes variantes.

IMPORTANT Emplacement des fichiers

La balise `<link>` ne fournit que le seul nom de fichier `style.css` : cela signifie que ce fichier se trouve dans le même dossier que notre page d'accueil `index.html`.

Si notre feuille de styles se trouvait dans un sous-dossier nommé `css`, il faudrait écrire :

```
<link href="css/style.css" ...>
```

N'oublions pas que la barre de séparation des dossiers s'écrit / comme sous Linux, et non \ comme sous Windows.

Pour aller plus loin dans la gestion des dossiers, sachez que deux points successifs et une barre « ../ » permettent de remonter dans le dossier « parent » de l'arborescence.

Les blocs qui constituent la structure de notre page sont les suivants.

- Le bloc ② `<div class="global">` comprend toute la page. Il permettra d'en définir la largeur et de la centrer au milieu de l'écran (ce bloc `<div>` sera centré horizontalement dans la balise `<body>`).
- Le bloc ③ `<header class="entete">` correspond au bandeau situé en haut de page, où se trouvent en évidence le logo, le thème du site et éventuellement d'autres éléments (slogan, numéro de téléphone, etc.).
- Le bloc ④ `<nav class="blocmenu">` inclut la liste des liens constituant le menu du site ⑤ `<ul id="menu">`. Si la présence de ce bloc `<nav>` n'est pas indispensable, puisque la liste `` est elle-même un bloc, elle permet cependant d'ajouter un contenu à droite du menu horizontal ou sous le menu vertical, suivant la configuration adoptée.
- Dans le bloc ⑥ `<section class="contenu">` se trouvera le contenu spécifique de chacune des pages.
- La page se termine par le bloc ⑦ `<footer class="pied">`, c'est notre pied de page.

MÉTHODE Classes ou identifiants ?

Notre code HTML utilise plusieurs classes qui n'apparaissent qu'une seule fois. Pourquoi ne pas avoir choisi plutôt des identifiants ? Pour répondre à cette question, prenons un paragraphe HTML possédant à la fois un identifiant et une classe :

```
<p id="rappel" class="rouge">Pensez à recycler vos papiers !</p>
```

Supposons que l'écriture de cette phrase soit initialement prévue en vert, puis que la classe rouge soit ajoutée à cette balise `<p>` dans un second temps, à l'occasion d'une mise à jour ou dynamiquement par programme, en JavaScript par exemple. Cette classe correspond à un style général qui attribue au texte la couleur rouge. Si les styles suivants lui sont appliqués, notre phrase ne sera pas écrite en rouge :
`#rappel { color: green; }`

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

Comme nous l'avons vu dans le chapitre 4, l'identifiant est prioritaire sur la classe, la phrase reste donc ici en vert. La solution à ce problème est la suivante : nous allons garder cette règle CSS, en conservant la classe `rouge` seule pour pouvoir l'utiliser ailleurs dans la page, mais en ajoutant un deuxième sélecteur pour traiter ce cas précis (sans espace entre l'identifiant `#rappel` et la classe `.rouge`):

```
#rappel { color: green; }
.rouge, #rappel.rouge { color: red; }
```

C'est pourquoi il est plus simple d'attribuer à notre paragraphe une classe `rappel` plutôt qu'un identifiant du même nom, en écrivant le code HTML de la façon suivante :

```
<p class="rappel rouge">Pensez à recycler vos papiers !</p>
```

Dans ce cas, le code CSS initial fonctionne alors très bien :

- `.rappel { color: green; }`
- `.rouge { color: red; }`

La règle de la classe `rouge` s'applique dans notre exemple, car elle est au même niveau de priorité que celle de la classe `rappel`. Les classes sont donc plus souvent utilisées que les identifiants, parce qu'elles simplifient la définition des styles.

Mais une autre question se pose alors : pourquoi utiliser un identifiant dans la balise `<ul id="menu">` ? C'est le chapitre suivant qui nous apportera la réponse : pour rendre notre site *responsive* et modifier dynamiquement le menu en fonction de la taille de l'écran, il nous faudra accéder en JavaScript à cette balise `ul`, d'où l'identifiant employé dès maintenant en prévision de ce besoin.

L'ossature et les blocs de code que nous venons d'étudier se retrouveront dans toutes les pages de notre site. En général, seul le bloc associé au contenu de la page sera modifié, les autres étant permanents d'une page à l'autre. Il ne s'agit pas d'une règle absolue, mais cela permet de donner une certaine unité au site.

Le plus important, c'est que le visiteur trouve toujours le même menu au même endroit ; sinon, la navigation dans le site devient compliquée. C'est d'ailleurs en nous basant sur la position du menu que nous allons définir

différents types de mise en forme.

Créer des pages de base à menu horizontal ou vertical

À partir du code HTML que nous venons d'étudier, nous allons écrire deux feuilles de styles, de façon à obtenir une page de base avec un menu horizontal, puis avec un menu vertical.

Page de base avec menu horizontal

Il s'agit de positionner les différents éléments du code HTML précédent pour placer le menu de façon horizontale, en haut de la page. Notre feuille de styles sera la plus simple possible pour obtenir ce résultat. L'affichage obtenu, bien que très sobre, sera tout à fait présentable comme le montre la figure 8-3.

Cette feuille de styles à écrire est un fichier texte, à enregistrer sous le nom de fichier `style.css`. N'oublions pas de le placer dans le même dossier que notre page d'accueil `index.html`. Nous allons observer son contenu pour le détailler ensuite.

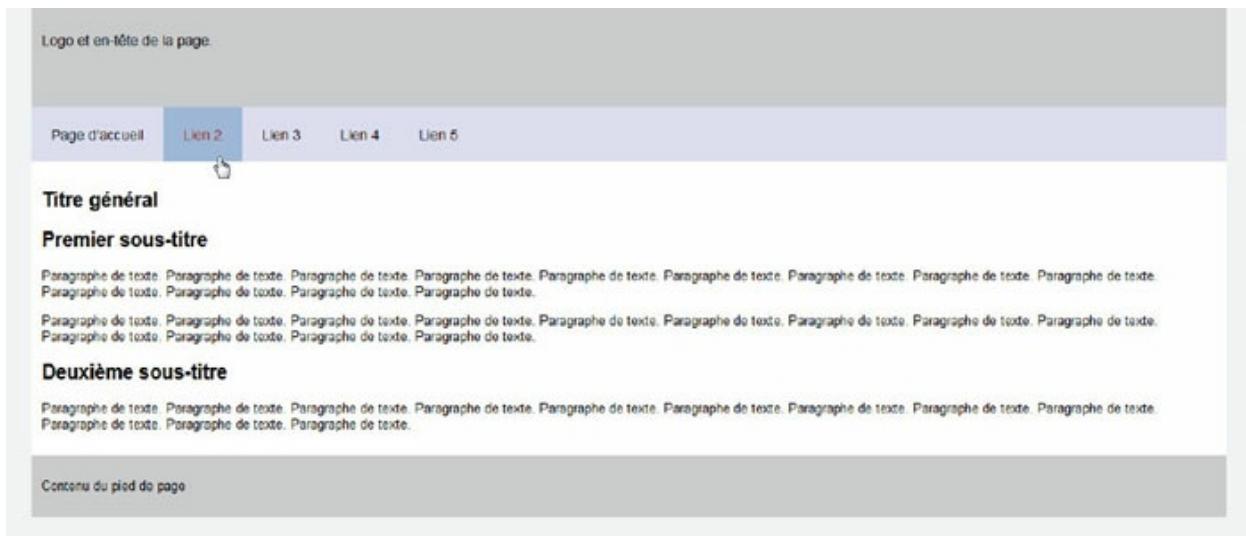


FIGURE 8–3 Notre trame de base avec un menu horizontal, après sa mise en forme par la feuille de styles

Feuille de styles `style.css` pour un menu horizontal

```
body, div, ul { margin: 0; padding: 0; } ①  
  
body {  
    background-color: #F5F5F5; /* gris clair */ ②  
    font-family: Arial, sans-serif; font-size: 13px;  
}  
/* L'ensemble de la page est fixé et centré ③ */  
.global { width: 1200px; margin: 0 auto; }  
  
/* Bandeau d'en-tête ④ */  
.entete { height: 80px; padding: 10px; background-color: lightgray; font-
```

```

size: 14px; }

/* Menu de navigation 5 */
.blocmenu { background-color: lavender; }

/* Contenu de la page 6 */
.contenu { padding: 10px; background-color: white; }

/* Pied de page 7 */
.pied { padding: 10px; background-color: lightgray; }

/* Détails pour le menu de navigation */
#menu { display: flex; list-style-type: none; }
#menu li { flex: 0 1 auto; } 8

#menu li a, #menu li a:visited {
  display: block; height: 35px; line-height: 35px;
  padding: 10px 20px; font-size: 14px;
  text-decoration: none; color: black; } 9

#menu li a:hover { background-color: lightsteelblue; color: brown; } 10

```

Étudions ensemble cette feuille de styles qui permet notamment, grâce aux propriétés de positionnement utilisées, de placer les différents blocs de notre code HTML comme indiqué par la figure 8-4.

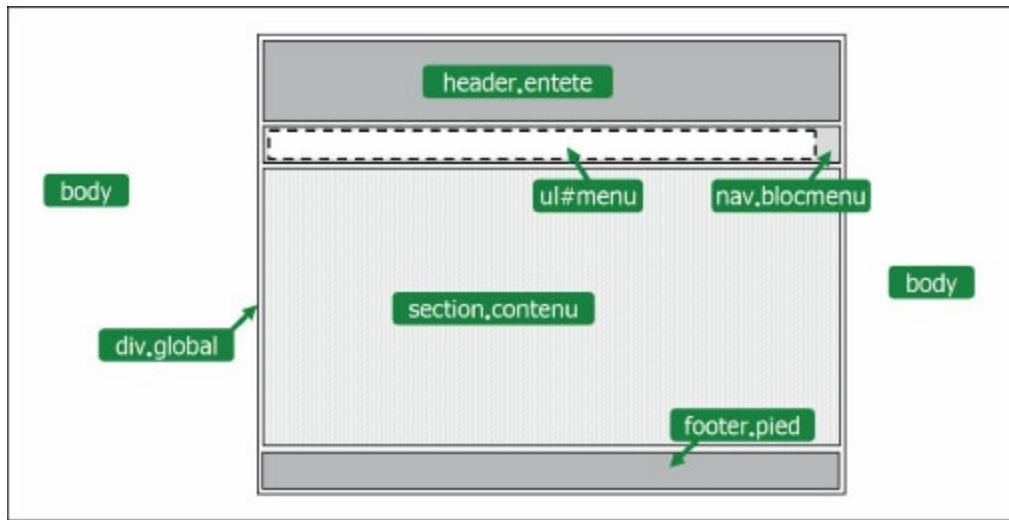


FIGURE 8-4 L'emplacement des différents blocs présents dans le code HTML. Les classes et identifiants sont repérés par les signes . et #, comme dans les feuilles de styles.

Principaux éléments de la page

La ligne ① sert à annuler les marges intérieures et extérieures des éléments

<body>, <div> et . Par la suite, nous pourrons redéfinir ponctuellement certaines d'entre elles, en fonction des besoins. Cette « remise à zéro » n'est pas nécessaire pour les balises introduites en HTML 5 (nav, header, section, footer...) : elles sont reconnues par les navigateurs modernes qui appliquent les standards du W3C, dont les marges initiales nulles pour ces blocs.

Pour la balise <body>, un fond gris ② est défini : il sera visible à gauche et à droite de la page centrée horizontalement. La police et la taille des caractères étant des propriétés héritées, celles données ici s'appliquent à tous les éléments de la page, sauf si elles sont redéfinies pour certaines balises.

Le contenu effectif de la page, placé dans le bloc <div class="global"> ③, aura une largeur de 1 200 pixels. Sa largeur étant fixe, ce bloc peut être centré dans la page avec la valeur auto attribuée aux marges extérieures de gauche et de droite.

À NOTER Largeur de base de la page

La largeur choisie pour le contenu de la page est ici de 1 200 pixels. Cela permet d'adapter le site à un large public, y compris aux personnes ayant un écran de petite taille, comme une tablette numérique en mode paysage ou un petit ordinateur portable de type *netbook*. La largeur réelle disponible sera peut-être légèrement supérieure à cette dimension, mais il faut prévoir la place nécessaire pour la barre de défilement et une barre d'outils éventuelle, d'où les 1 200 pixels.

Le bandeau d'en-tête <head class="entete"> ④ a ici une hauteur définie de 80 pixels, ainsi qu'une marge intérieure pour espacer du bord les textes et logos qu'il contiendra. Il possède sa propre couleur de fond et la taille du texte y est plus grande que dans les paragraphes de la page.

Le bloc <nav class="blocmenu"> ⑤ contient le menu de navigation. Sa hauteur sera automatique, fonction de la hauteur des liens (propriété height des balises <a> du menu). La propriété de couleur de fond qui lui est appliquée pourrait être directement appliquée à la liste <ul id="menu">, ce bloc nav qui l'entoure étant facultatif.

Le contenu de la page ⑥ est espacé du bord par une marge intérieure padding de 10 pixels sur chaque côté ; sa couleur de fond est blanche.

Le pied de page ⑦ possède des marges intérieures de 10 pixels et s'affichera sur un fond gris.

Détails du menu

La liste `<ul id="menu">`, dont les puces standards sont supprimées, est de type `flex` sans précision, donc en mode ligne : les blocs qu'elle contient se placeront côté à côté. Les lignes du menu ❸ sont des items flex possédant la propriété `flex: 0 1 auto;`, ce qui signifie : pas d'ajustement à la largeur totale (valeur `0`), largeur réduite automatiquement s'il y a débordement (valeur `1`) et largeur de base automatique, donc calculée en fonction du texte de chaque lien.

Les liens `<a>` ❹ du menu sont définis en tant que blocs, ce qui permettra de régler leur hauteur. Ils ont une marge interne de 10 pixels en haut et en bas, de 20 pixels à gauche et à droite, une hauteur ainsi qu'une hauteur de ligne de 35 pixels, une taille de caractère de 14 pixels. Par ailleurs, les liens étant par défaut soulignés et écrits en bleu (parfois en violet pour les liens déjà visités), il faut redéfinir deux propriétés, `text-decoration` pour enlever le soulignement et `color` pour changer la couleur.

Au passage de la souris ❽, les couleurs du fond et des caractères changent. Grâce aux valeurs identiques fournies pour la hauteur et la hauteur de ligne de ces liens, toute la hauteur du menu change de couleur.

Pour personnaliser votre feuille de styles, vous pouvez changer à loisir les couleurs de fond et des caractères, les tailles des différents blocs, etc. Si vous préférez un menu vertical, la solution se trouve dans les lignes qui suivent.

Page de base avec menu vertical

Le même code HTML que celui utilisé précédemment nous permet de construire un modèle de page avec un *menu vertical*. Il suffit de modifier quelques points de notre feuille de styles, pour obtenir la page donnée par la figure 8-5.

Nous allons examiner le nouveau fichier `style.css` (toujours à placer dans le même dossier que notre page HTML) et détailler ensuite les principaux changements effectués par rapport à la feuille de styles précédente.



FIGURE 8–5 Notre trame de base avec un menu vertical, après sa mise en forme par la feuille de styles adéquate

Feuille de styles `style.css` pour un menu vertical

```
body, div, ul { margin: 0; padding: 0; } ①

body {
    background-color: #F5F5F5; /* gris clair */ ②
    font-family: Arial, sans-serif; font-size: 13px;
}
/* Page centrée contenant des blocs en ligne ③ */
.global { width: 1200px; margin: 0 auto;
    background-color: lavender;
    letter-spacing: -0.31rem; }
.global * { letter-spacing: normal; }

/* Bandeau d'en-tête ④ */
.entete { height: 80px; padding: 10px; background-color: lightgray; font-size: 14px; }
```

```

/* Colonnes de gauche (menu) et de droite (contenu) */
.blocmenu, .contenu { box-sizing: border-box; display: inline-block;
vertical-align: top; } ⑤
.blocmenu { width: 200px; } ⑥
.contenu { width: 1000px; } ⑦
padding: 10px; padding-left: 30px;
background-color: white;

/* Pied de page */
.pied { padding: 10px; background-color: lightgray; } ⑧

/* Détails pour le menu de navigation */
#menu { list-style-type: none; } ⑨

#menu li a, #menu li a:visited { ⑩
display: block; padding: 15px 20px; font-size: 14px;
text-decoration: none; color: black; }

#menu li a:hover { background-color: lightsteelblue; color: brown; } ⑪

```

Pour cette trame à menu vertical, le placement des différents blocs HTML ressemble beaucoup à celui de la trame à menu horizontal : seuls le menu et le contenu de la page sont disposés différemment, comme le montre la figure 8-6.

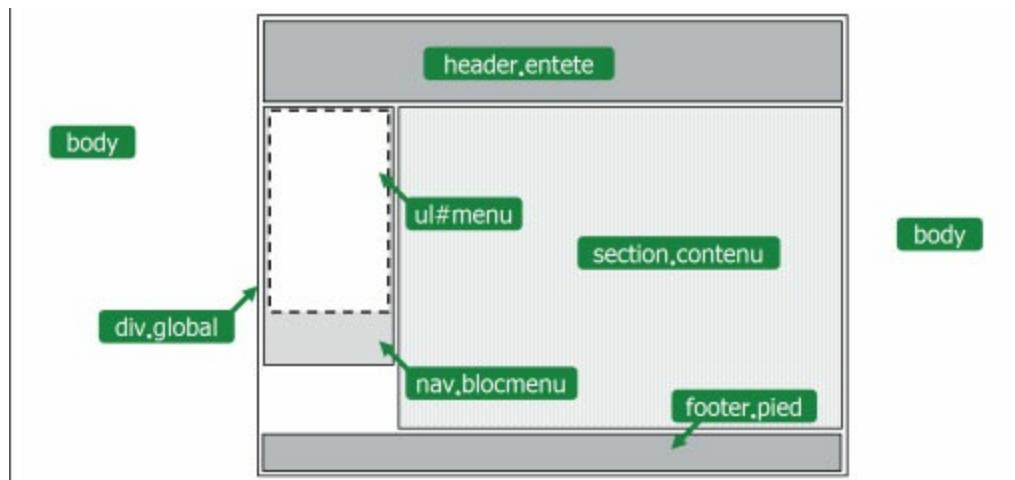


FIGURE 8–6 L'emplacement des différents blocs HTML dans la trame à menu vertical, leurs classes et identifiants étant repérés par les signes . et #. La hauteur du bloc nav.blocmenu est fonction de son contenu.

Modifications des principaux éléments de la page

Cette feuille de styles comporte de nombreux points communs avec la

précédente ; nous allons nous intéresser aux changements effectués.

Les règles de style ①, ② et ④ ne sont pas modifiées. La troisième ③ voit deux propriétés s'ajouter pour le bloc `<div class="global">` : une couleur de fond, qui sera celle du bloc contenant le menu, et une annulation de l'espacement entre les lettres (valeur `-0.31rem`). Cette astuce nous évitera un espace entre les *blocs en ligne* qu'il contiendra, car ces éléments `blocmenu` et `contenu` seront posés comme des caractères sur une ligne.

La propriété `letter-spacing` étant héritée, elle sera rétablie à sa valeur normale pour tous les éléments inclus dans ce conteneur (sélecteur `.global *`, où l'étoile désigne n'importe quelle balise incluse dans `global`).

ASTUCE Emplacement de la couleur de fond du menu

Pourquoi la couleur de fond du menu est-elle indiquée dans le bloc `global` ? Parce que le menu vertical ayant une taille limitée et fonction du nombre de liens, il est généralement moins haut que le contenu de la page, ce qui correspond au schéma de la figure 8-6. Si ce menu portait sa couleur de fond, celle-ci s'arrêterait après le dernier lien, et notre colonne de gauche ne serait ni homogène, ni esthétique, comme le montre la figure 8-7.

L'astuce consiste donc à :

- laisser le bloc menu transparent ;
- attribuer la couleur de fond de ce menu au bloc `global` ;
- définir une couleur de fond pour chacun des autres éléments, enfin.

Ainsi, la colonne de gauche donne l'impression que le menu se prolonge jusqu'à la fin de la page. La figure 8-7 présente l'aspect du menu, avant et après l'application de cette astuce.

Ce procédé suppose cependant que le contenu de la page ait une hauteur supérieure ou égale à celle du menu. Si ce n'est pas le cas, il suffira d'ajouter quelques sauts de ligne ou une marge supplémentaire à la fin de ce contenu.

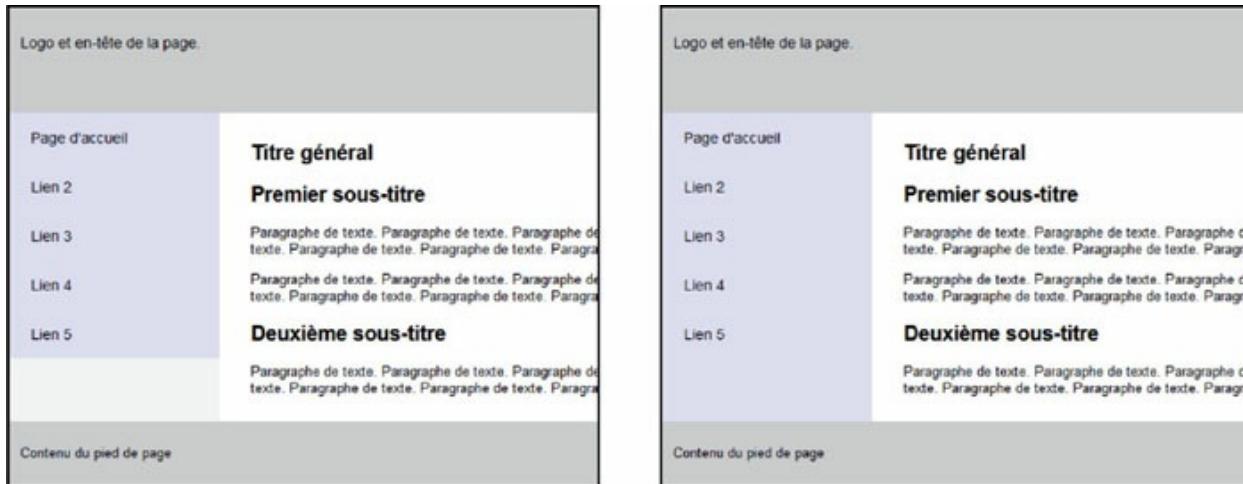


FIGURE 8-7 Le menu étant plus court que la page, le grisé qui lui est attribué en couleur de fond ne remplit pas la colonne de gauche (image de gauche). Si le fond du bloc menu est transparent, c'est le bloc `<div class="global">` qui assure la couleur de fond du menu (image de droite).

Les deux blocs `<div>` de classes `blocmenu` et `contenu` ⑤ (menu vertical et contenu de la page) seront placés côté à côté, grâce à leur type `inlineblock`, et alignés en haut de la page. La propriété `box-sizing` prend la valeur `border-box` : ainsi, la largeur de chacun comprendra leurs marges internes et leurs bordures ; leur somme correspondra donc à la largeur totale disponible. Cette méthode est conseillée, car elle permet un calcul plus sûr et plus rapide des dimensions.

Pour le bloc contenant le menu `<nav class="blocmenu">` ⑥, il suffit de définir sa largeur. Il ne possède plus de couleur de fond : comme nous l'avons vu, ce bloc est transparent et le fond du menu sera celui du bloc `<div class="global">`.

Le contenu de la page `<section class="contenu">` ⑦ conserve son fond blanc et ses marges intérieures de 10 pixels, mais celle de gauche passe à 30 pixels, de façon à écarter un peu le texte qu'il contiendra du menu de gauche. Sa largeur doit ici être définie : elle est de 1 000 pixels, qui, ajoutés aux 200 pixels du menu, remplissent bien notre gabarit de 1 200 pixels.

Le pied de page ⑧ conserve la même mise en forme. En ce qui concerne le menu ⑨, dont les puces sont toujours supprimées, il n'est plus nécessaire d'adopter le modèle flexbox.

Les liens du menu ⑩ sont affichés en tant que blocs, de façon à constituer pour chacun un rectangle aussi large que la colonne du menu : c'est plus esthétique et d'un meilleur confort pour la navigation. Il en résulte que leur

hauteur et leur hauteur de ligne n'ont plus à être définies. Mis à part un ajustement des marges internes verticales (15px au lieu de 10px, pour un meilleur confort de navigation), les autres styles de ces liens sont inchangés.

Au passage de la souris [11](#), la couleur de fond et celle du texte des liens sont modifiées de la même façon que pour le menu horizontal.

Voici donc le tour de force que permettent les CSS : en utilisant exactement le même code HTML et en apportant une douzaine de modifications à la feuille de styles, nous avons transformé notre page à menu horizontal en une autre à menu vertical.

Nous avons commencé de façon simple, avec ces deux trames de site comprenant uniquement des couleurs de fond pour assurer le minimum d'esthétique. Il est temps d'ajouter quelques images ; nous allons nous en préoccuper à présent.

Exemples concrets avec images de fond et dégradés CSS 3

Pour améliorer nos réalisations, rien de tel que des images qui vont agrémenter les différentes parties de la page. À partir de la même structure de code HTML que nous venons d'utiliser, nous allons étudier deux types de modèles. Le premier, à menu vertical, utilise une grande photo qui constitue l'essentiel de la page. Le deuxième, à menu horizontal, associe un fond différent à chacun des principaux blocs : couleur unie, image de fond ou dégradé CSS.

RESSOURCES

Toutes les images nécessaires sont téléchargeables à l'adresse suivante :

► <http://livre.antevox.fr>

Nous adressons tous nos remerciements pour son aide au graphiste réputé Kar Kui Lam, qui a aimablement apporté son concours pour le graphisme, les photos et l'harmonie des couleurs. Vous pourrez trouver ses références sur LinkedIn :

► <https://www.linkedin.com/in/kar-kui-lam-24958150>

Une grande image pour toute la page

Les images et les couleurs d'un site sont à choisir en fonction de son thème. Notre premier site en images parle des mers et océans dans le monde. Vous pouvez voir son allure sur la figure 8-8. Sous le nom du site, en blanc étiré sur un fond de ciel bleu, les titres sont en vert foncé (titre de niveau 1) et en bleu foncé (titres de niveau 2) : ce sont des couleurs reposantes, qui nous donnent un avant-goût des vacances !



FIGURE 8–8 Un air de vacances pour notre première page avec des images de fond

Comme le montre la figure 8-9, peu d'images composent la structure de ce site, puisque l'essentiel de la page est constitué d'une grande image de fond. Cette image se termine en bas par une couleur unie sur toute sa largeur : c'est important, car nous choisirons la même couleur pour le fond du bloc global. Ainsi, il n'y aura pas de trait de séparation entre les deux lorsque nous agrandirons notre page en y plaçant du contenu.

Les autres images sont :

- un fond répétitif qui se verra de part et d'autre de la page centrée – il s'agit ici d'une texture de sable ;
- un bouton de menu, avec une coquille Saint-Jacques sur la gauche ;
- un bouton de menu actif : au passage de la souris, la coquille pivote vers le menu pointé.

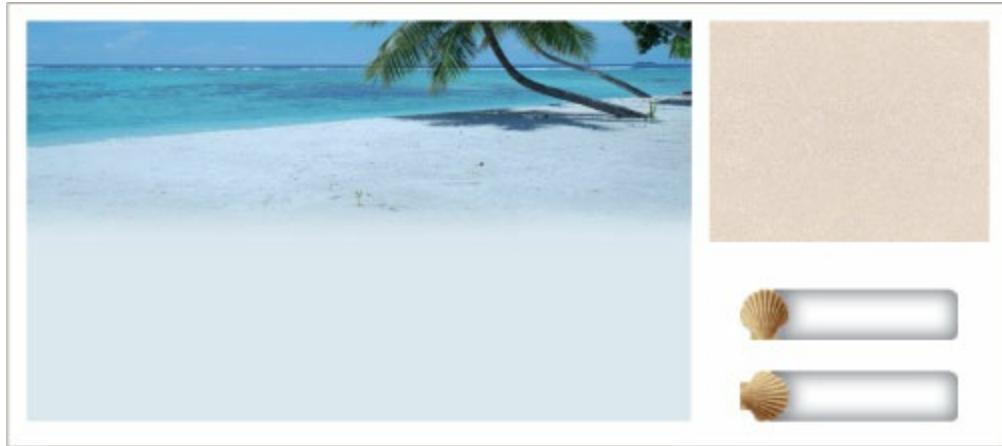


FIGURE 8–9 Les quatre images qui constituent la structure du site : une grande image posée sur une texture de sable (répétée sur tout le fond de l'écran) et les images qui agrémenteront le menu, pour les liens ordinaires et pour le lien signalant le passage de la souris

Code HTML

La structure du code HTML sera identique à celle que nous avions étudiée précédemment pour les modèles de base. Le début de la page (jusqu'à la balise `</head>`) n'est pas modifié, sauf pour la balise de titre qu'il faudra rectifier :

```
| <title>Mers et Océans</title>
```

Voici le contenu de la balise `<body>` : les blocs de base sont les mêmes et conservent leur nom, seul leur contenu change.

Corps de la page HTML pour le site Mers et Océans

```
| <body>
| <div class="global">
|
|   <header class="entete">
|     <h1>Mers et Océans</h1> ①
|   </header>
```

```

<nav class="blocmenu">
  <ul id="menu"> ②
    <li><a href="index.html">Accueil</a></li>
    <li><a href="oceans.html">Les Océans</a></li>
    <li><a href="mers.html">Les Mers</a></li>
    <li><a href="courants.html">Les courants</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>

<section class="contenu">
  <h1>La Planète Bleue</h1> ③

  <h2>Source de vie</h2> ④
  <p>Paragraphe de texte...</p>
  <p class="centre">
     ⑤
  </p>
  <p>Paragraphe de texte...</p>

  <h2>Une grande diversité</h2> ④
  <p>
     ⑥
    Paragraphe de texte...
  </p>
  <p>Paragraphe de texte...</p>

</section> <!-- Fin du bloc section.contenu -->
<footer class="pied">
  <p>Mers et Océans, le site de l'eau sous toutes ses formes</p> ⑦
</footer>

</div> <!-- Fin du bloc div.global -->
</body>

```

Il ne faudra pas oublier de terminer la page par la balise `</html>`.

Les modifications par rapport à notre modèle de base sont les suivantes.

- L'en-tête comprend un grand titre « Mers et Océans » ① écrit sous forme de texte.
- Le texte du menu ② et des liens associés a ici été précisé, de même que le titre général ③ de la page, les sous-titres ④ et le texte du pied de page ⑦. Le texte réel des paragraphes n'a pas été reproduit, pour conférer le maximum de clarté au code HTML.

- L'image **5** est centrée, grâce à la classe `centre` attribuée au paragraphe qui la contient. La feuille de styles devra donc indiquer que le texte est centré pour les éléments possédant cette classe.
- L'image **6** doit être calée à gauche et habillée sur la droite par le texte qui suit. C'est la classe `gauche` qui permettra ce positionnement, à définir dans la feuille de styles : l'élément sera « flottant à gauche » et possédera une marge à droite, pour espacer le texte de l'image.

ASTUCE Des commentaires bien utiles

Vous noterez les petits commentaires qui accompagnent notre code HTML : ils sont bien pratiques pour repérer les balises qui ferment un bloc dont l'ouverture est un peu lointaine. C'est le cas en particulier pour le bloc de classe `global` et pour la section qui encadre le contenu de la page.

Il ne faut pas hésiter à écrire d'autres commentaires pour séparer les différentes parties du contenu de la page, lorsque celle-ci prend de la consistance. Ces annotations permettent aussi d'expliquer une méthode de codage utilisée, la présence de telle ou telle balise, etc. Cette bonne habitude est également valable pour la feuille de styles, tout comme dans la programmation en général.

Cela peut paraître fastidieux au moment de la création de la page, alors que tout est si clair dans notre tête. Mais s'il faut reprendre et modifier le site un an plus tard, quel bonheur de trouver ces points de repère !

Il nous reste à écrire la feuille de styles. Là encore, nous allons reprendre notre version de base, celle correspondant au menu à gauche, pour y apporter quelques améliorations.

Feuille de styles

Si nous conservons l'en-tête de la version de base, notre feuille de styles sera un fichier texte, appelé `style.css` et placé dans le même dossier que notre page HTML. Il s'agit d'une adaptation de la feuille de styles du modèle précédent, la trame de base avec le menu à gauche. Les images de fond y sont insérées, les marges sont ajustées et quelques styles de présentation sont modifiés ou ajoutés en fonction du contenu de la page.

Feuille de styles CSS pour le site Mers et Océans

```
body, div, ul { margin: 0; padding: 0; }

p { margin: 10px 0; } ①

body { background-image: url(texture_sable.jpg);
      font-family: Times, serif; font-size: 18px; } ②

/* Page centrée, fond bleu clair, image au-dessus */ ③
.global { width: 1200px; margin: 0 auto;
           background: #e6eef1 url(fond_global.jpg) no-repeat; letter-spacing:
-0.31rem; }
.global * { letter-spacing: normal; }

.entete { height: 80px; padding: 20px; } ④
#entete h1 { margin: 3px 30px; color: white; font-size: 60px; font-weight:
normal; }

.blocmenu, .contenu { box-sizing: border-box; ⑤
                      display: inline-block; vertical-align: top;
                      margin-top: 20px; }

.blocmenu { width: 225px; } ⑥

.contenu { width: 975px;
           padding: 0 10px 10px 40px; } ⑦
.pied { margin-bottom: 10px; padding: 20px;
        background-color: white; color: #005f36;
        font-size: 18px; font-style: italic; } ⑧

/* Détails pour le menu de navigation */
#menu { margin-top: 30px; margin-left: 10px; ⑨
         list-style-type: none; }

#menu li a, #menu li a:visited { ⑩
                                     background: url(bouton-menu.png);
                                     display: block; line-height: 30px;
                                     margin: 25px 0; padding: 10px 5px 10px 60px;
                                     text-decoration: none;
                                     font-size: 20px; font-weight: bold;
                                     font-style: italic; color: #003060; }

#menu li a:hover { ⑪
                     background-image: url(bouton-menu-actif.png);
                     color: #ac0000; }

/* Contenu de la page */
h1 { font-style: italic; font-family: Times, serif;
```

```
font-size: 40px; color: #005f36; } 12  
h2 { font-family: Arial, sans-serif; font-size: 26px;  
    font-weight: normal; color: #003060; } 13  
  
/* Classes pour l'alignement */  
.centre { text-align: center; } 14  
.gauche { float: left; margin-right: 10px; } 15
```

Intéressons-nous aux changements et ajouts effectués à partir du style basique pour un menu vertical, celui que nous avions précédemment étudié. Tous les paragraphes `<p>` ont une marge extérieure de 10 pixels en haut et en bas ①, aucune à gauche ni à droite, les marges intérieures des blocs conteneurs étant suffisantes. Rappelons que les marges extérieures de deux éléments consécutifs sont fusionnées.

Pour la balise `<body>`, outre une police en Times 18 pixels valable pour toute la page, une image de fond est définie ②: c'est la texture qui va remplir de sable tout l'écran et qui sera visible en dehors de la partie centrale.

Si le bloc `<div class="global">` ③ conserve sa largeur et ses marges pour un centrage horizontal, il a maintenant pour fond notre belle et grande image de plage, cette image n'étant bien sûr pas répétée. Sa couleur de fond est un bleu clair absolument identique à la couleur unie qui termine le bas de l'image. Ainsi, la transition sera invisible lorsque la page s'agrandira. Notez que l'image s'applique par dessus la couleur de fond, quel que soit l'ordre d'écriture des déclarations de style.

Nous retrouvons l'astuce qui annule l'espacement entre caractères (valeur `-0.31rem` pour `letter-spacing`), pour éviter un espace de séparation entre les blocs du menu et du contenu (étant de type `inline-block`, ils seront posés comme des lettres sur une ligne). La valeur normale de cet espacement est réattribuée aux contenus des balises incluses dans le bloc `global`.

La hauteur de l'en-tête ④ n'a pas changé (80 pixels), mais ses marges internes augmentent un peu (20 pixels). La taille de police est inutile ici, et il faut enlever la couleur de fond pour ne pas masquer notre belle image. Une règle est ensuite ajoutée pour déterminer la position et l'écriture du titre `<h1>` contenu dans cet en-tête.

Les blocs du menu et du contenu ⑤ sont paramétrés comme auparavant : leurs dimensions incluront les marges internes (ainsi que les bordures éventuelles, inutilisées ici), et ils seront toujours de type `inline-block` ; ils se

trouveront côté à côté, comme les lettres d'un mot, mais alignés vers le haut. Une marge externe commune de 20 pixels est ajoutée en haut, ce qui aérera le début de la page.

La largeur du `<nav class="blocmenu">` 6 est de 225 pixels au lieu de 200. Par conséquent, le bloc `<section class="contenu">` 7, où se trouvera le texte de la page, voit sa largeur passer de 1 000 à 975 pixels, et nous retrouvons bien un total de 1 200 pixels. Ses marges internes sont modifiées : celle du haut est supprimée, elles restent à 10 pixels à droite et en bas, celle de gauche est agrandie à 40 pixels. Bien entendu, la couleur de fond du contenu de la page est supprimée, sinon elle recouvrirait la plage et ses palmiers.

Le pied de page 8 voit ses marges et sa couleur de fond modifiées, ainsi que les attributs du texte qu'il contient : du même vert foncé que le titre principal de la page, en taille 18 pixels et en italique.

En ce qui concerne le menu de navigation, dont un aperçu est donné par la figure 8-10, sa mise en forme est la suivante.

- Des marges externes sont appliquées au bloc `<ul id="menu">` 9.
- Les liens ordinaires ou déjà visités 10 ont un bouton pour image de fond. Des marges et une hauteur de ligne adaptées permettent d'afficher entièrement les boutons, des les espacer et de caler le texte à l'intérieur. Le texte du menu est du même bleu foncé que les sous-titres de la page ; sa mise en forme est ajustée.
- Lorsque la souris passe sur un lien 11, l'image du bouton change, ainsi que la couleur du texte qui devient rouge-marron.



FIGURE 8–10 Détail du fonctionnement du menu, pour la page Mers et Océans : seule la coquille Saint-Jacques semble changer de sens au passage de la souris, mais il s'agit bien de deux images de bouton différentes.

Il ne reste plus qu'à s'occuper du contenu de la page.

- Les titres `<h1>` 12 sont en Times italique, écrits en vert foncé, taille 40 pixels.
- Les titres `<h2>` 13 sont en Arial et en bleu foncé, taille 26 pixels ; le gras (défini par défaut pour toutes les balises `<h1>` à `<h6>`) a été supprimé pour alléger ces sous-titres.
- La classe `centre` 14 sert, comme son nom l'indique, à centrer un contenu. Notre code HTML l'utilise pour centrer une image dans un paragraphe.
- La classe `gauche` 15 permet de faire flotter des images à gauche, de façon à ce que le texte qui suit habille ces images sur la droite. Une marge de droite est ajoutée pour créer un espace esthétique entre le texte et l'image. S'il y avait des images à placer à droite, nous ajouterions une classe `droite`, avec flottement à droite et marge à gauche.

En résumé, deux images de fond peuvent suffire pour agrémenter un site : une grande pour le bloc global et une autre pour les boutons du menu. L'image du bouton actif et la texture de fond autour de la page sont facultatives. Il est important que le bas de l'image principale se termine par une couleur unie, éventuellement à l'aide d'un fondu pour atteindre cette couleur fixe.

Il faut bien sûr adapter chaque fois les dimensions et les marges au graphisme choisi, et trouver une présentation appropriée aux textes de chacune des parties (type de police, taille, couleur, gras, italique), le tout en harmonie. Nous n'allons pas entrer dans les détails de la présentation ; il s'agit d'un autre métier, celui de graphiste.

IMPORTANT **Lisibilité du texte**

Le choix de la couleur du texte de la page est important, car il faut s'assurer d'une bonne lisibilité. En particulier, le texte de la page doit pouvoir se détacher nettement sur tous les endroits de l'image utilisée, sauf pour la partie qui constitue le haut de la page.

Si l'image comporte à la fois des zones très sombres et des zones très claires, c'est pratiquement mission impossible. Le mieux est alors de changer d'image ou bien de l'estomper en utilisant un logiciel de retouche d'image, comme Gimp, PhotoFiltre ou PhoXo.

Les sites créés ainsi avec une grande image principale ne sont pas les plus courants. Souvent, une image ou un fond distinct est utilisé pour chacun des éléments de la page : le bandeau, les boutons du menu, le pied de page, etc. Nous allons donc étudier ce cas de figure et en profiter pour changer la position du menu.

Site sur deux colonnes, avec image de fond et dégradés CSS

Ce deuxième exemple nous emmène en voyage, plus précisément à Hong Kong. La page à réaliser est celle que vous voyez sur la figure 8-11. Elle contient un menu horizontal et se trouve constituée de nos éléments habituels, ils sont dotés pour la plupart soit d'une image de fond, soit d'un dégradé en CSS 3.

Le titre principal et les sous-titres de la page sont respectivement en rouge et orange. Il s'agit de couleurs chaudes, qui correspondent bien à l'activité intense de cette ville, et le rouge rappelle en même temps le drapeau de Hong Kong. Le titre du bandeau fait partie de l'image ; il est en gris très clair, presque blanc, pour mieux ressortir sur la photo de la ville au crépuscule.

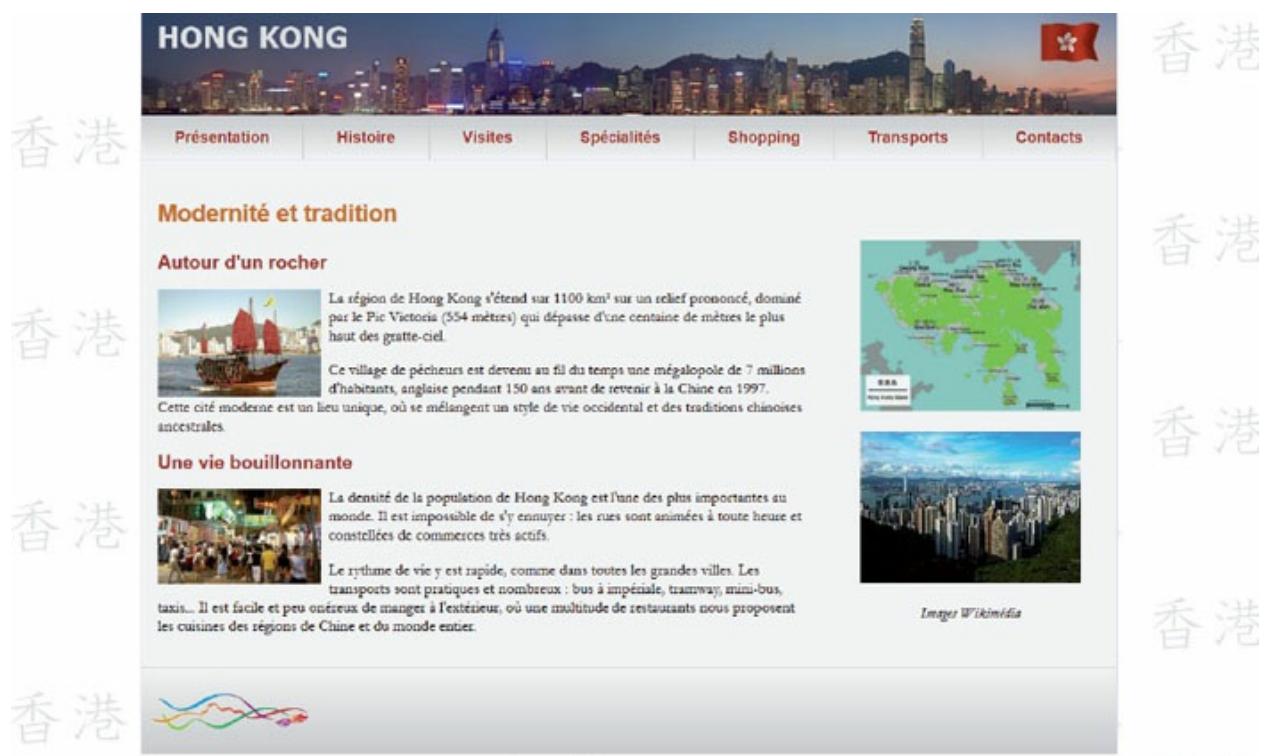


FIGURE 8-11 Notre invitation au voyage à Hong Kong utilise des images de fond et des dégradés en CSS 3, associés aux différents blocs de la page.

Code HTML

Nous allons conserver la même structure de code HTML, le contenu étant

spécifique au site, et nous baser sur la feuille de styles déjà vue pour le modèle à menu vertical.

Corps de la page HTML pour le site Hong Kong

```
<body>
<div class="global">

    <header class="entete"> 1
        
    </header>

    <nav class="blocmenu">
        <ul id="menu"> 2
            <li><a href="index.html">Présentation</a></li>
            <li><a href="histoire.html">Histoire</a></li>
            <li><a href="visites.html">Visites</a></li>
            <li><a href="specialites.html">Spécialités</a></li>
            <li><a href="shopping.html">Shopping</a></li>
            <li><a href="transports.html">Transports</a></li>
            <li><a href="contacts.html">Contacts</a></li>
        </ul>
    </nav>

    <section class="contenu">

        <article>
            <h1>Modernité et tradition</h1> 3
            <h2>Autour d'un rocher</h2> 4
            <p>
                 5
                Paragraphe de texte...
            </p>
            <p>Paragraphe de texte... </p>
            <h2>Une vie bouillonnante</h2> 6
            <p>
                 7
                Paragraphe de texte...
            </p>
            <p>Paragraphe de texte...</p>
        </article>

        <aside> 8
            <p></p>
            <p></p>
    
```

```
<p><em>Images Wikimédia</em></p>
</aside>

</section> <!-- Fin du bloc section.contenu -->

<footer class="pied"> 9
  <p></p>
</footer>

</div> <!-- Fin du bloc div.global -->
</body>
```

Il ne faudra pas oublier de terminer la page par la balise `</html>` et de préciser le thème du site, à l'aide de la balise `<title>` à placer dans l'en-tête :

```
<title>Hong Kong</title>
```

Dans ce code, l'en-tête ① est constitué par une image. Pour l'adaptation aux mobiles, qui fera l'objet du chapitre suivant, elle pourra être limitée à 100 % de la largeur avec la propriété `max-width`. Notez que cette balise `` aurait aussi pu être remplacée par une image de fond du bloc `header`, adaptable aux petits écrans grâce à la nouvelle propriété CSS 3 `background-size` qui ajuste les images de fond.

Le menu ② contient des liens vers d'autres pages, pour l'instant fictives mais que vous pourrez vous amuser à composer.

Le contenu se trouve dans une balise `section` comprenant deux colonnes, encadrées par les balises `article` et `aside`. Dans la colonne principale (balise `article`), le titre ③, ainsi que les sous-titres ④ et ⑥, donnent la structure du contenu de la page. L'utilisation d'intertitres et de paragraphes courts améliore la lisibilité d'un texte.

Les images ⑤ et ⑦ sont habillées par le texte qui les suit : celui-ci vient s'écrire à côté de l'illustration. La classe `gauche` sera définie dans la feuille de styles, pour rendre flottantes à gauche les images concernées. En cas de besoin, une classe `droite` aurait pu être ajoutée pour les images qui doivent être placées à droite, avec habillement par le texte sur la gauche.

La balise `aside` ⑧ sert à afficher la colonne latérale, constituée de deux images et d'une petite légende.

Enfin, le contenu du pied de page ⑨ est l'image stylisée d'un dragon, extraite du site du Hong Kong Tourism Board : www.discoverhongkong.com.

Feuille de styles

Notre référence sera cette fois la feuille de styles de la trame de base avec menu horizontal. Il suffira d'y placer l'image de fond, présentée sur la figure 8-12, puis d'adapter et compléter le reste du code CSS. Ces règles de style constitueront un fichier texte appelé `style.css` et placé dans le même dossier que notre fichier HTML, ce qui nous évitera de modifier la balise `<link>` de l'en-tête.

Feuille de styles CSS pour le site Hong Kong

```
body, div, ul { margin: 0; padding: 0; }

body { background-image: url(hk-body.jpg); ①
      font-family: 1.2rem Garamond, Times, serif; }

.global { margin: 0 auto; width: 1200px;
          background-color: whitesmoke;
          border: solid 1px lavender; border-top: none; } ②

.entete img { display: block; } ③

#menu { background: linear-gradient(lightgray, whitesmoke);
         border-bottom: solid 1px lavender; } ④

.contenu { letter-spacing: -0.31rem; } ⑤
.contenu * { letter-spacing: normal; }

article, aside { box-sizing: border-box; ⑥
                 display: inline-block; vertical-align: top; padding: 20px; }
article { width: 70%; } ⑦
aside { width: 30%; padding-top: 80px; text-align: center; } ⑧

.pied { padding: 10px; ⑨
        background: linear-gradient(whitesmoke, lightgray);
        border-top: solid 1px lightgray; }

/* Détails pour le menu de navigation */
#menu { display: flex; list-style-type: none; } ⑩
#menu li { flex: 1 1 auto; } ⑪

#menu li a, #menu li a:visited { ⑫
      display: block; text-align: center;
      padding: 15px 0; border-right: solid lightgray 1px;
      font-family: Arial, sans-serif;
      font-size: 1rem; font-weight: bold;
```

```

color: brown; text-shadow: white 2px 0 0;
text-decoration: none; }

#menu li a:hover { ⑬
background: linear-gradient(silver, dimgray);
color: white; text-shadow: none; }

/* Contenu de la page */
h1 { font-family: Arial, sans-serif;
font-size: 1.5rem; color: #d06b19; } ⑭
h2 { font-family: Arial, sans-serif;
font-size: 1.2rem; color: brown;} ⑮

/* Classes pour l'alignement */ ⑯
.droite { float: right; margin-left: 10px; }
.gauche { float: left; margin-right: 10px; }

```

Examinons ensemble les principaux changements de cette feuille de styles par rapport à celle de notre modèle de base avec menu à gauche.

L’ensemble de l’écran ① (balise `<body>`) contient une image de fond qui sera répétée : elle est constituée du mot Hong Kong (c’est-à-dire « port parfumé ») écrit en chinois. De taille `1.2rem` (environ 19 pixels), la police choisie pour le contenu de la page est le Garamond ; elle sera remplacée par du Times si elle est absente de l’ordinateur du visiteur. Notez qu’en CSS, `Times` est équivalent à `Times New Roman`.



FIGURE 8–12 Présentée sur fond gris, l’image de fond utilisée pour la balise `body` du site sur Hong Kong. Les autres images sont insérées sous forme de balises `img`, tandis que certains fonds sont créés à partir d’une couleur unie (contenu de la page) ou d’un dégradé CSS 3 (menu et pied de page).

Le bloc `<div class="global">` ②, centré horizontalement et de largeur `1200px`, aura un fond uni de couleur gris très clair `whitesmoke` et sera doté d’une bordure gris-bleu `lavender`, en bas et sur les côtés, mais pas en haut.

Puisque l'en-tête `<header class="entete">` ③ contient une seule image (qui définit ses dimensions) mais pas de texte, il n'est plus nécessaire de lui attribuer de styles : hauteur, couleur de fond et taille du texte sont inutiles, les marges internes restant nulles pour que l'image remplisse entièrement ce bloc. En revanche, il est nécessaire de donner le type `block` à cette image pour éviter la présence d'un espace non désiré en dessous.

Le conteneur du menu `<nav class="blocmenu">` n'a plus besoin de styles, la mise en forme étant effectuée sur la balise `<ul id="menu">` ④ (il s'agit d'une couleur de fond créée par un dégradé linéaire). Une bordure du bas termine ce menu. Les règles CSS associées aux liens du menu se trouvent plus loin dans cette feuille de styles, ce qui permet de rassembler au début les propriétés des blocs principaux.

Le contenu de la page `<section class="contenu">` ⑤ n'a plus besoin que d'une seule mise en forme : c'est l'astuce qui annule l'espacement entre caractères (`-0.31rem` pour `letter-spacing`). Les « caractères » qu'il faut coller sont en fait les deux blocs `article` et `aside`, qui constitueront le contenu de la page. Ces éléments seront de type `inline-block`, donc posés comme des caractères sur une ligne, à l'intérieur de la section.

Bien sûr, l'espacement entre lettres doit revenir à la normale pour l'écriture dans les blocs inclus dans la section, d'où la règle appliquée au sélecteur `section *`, soit pour tous les éléments imbriqués dans la balise `<section>`.

Notre page sera donc organisée en deux colonnes, une partie principale dans la balise `article`, un contenu latéral dans l'élément `aside`. À ces deux balises correspond une mise en forme commune ⑥ : en particulier, elles seront toutes les deux de type `inline-block` et alignées verticalement sur le haut. Pour faciliter le calcul de leur largeur, la propriété `boxsizing` vaut `border-box` ; leurs dimensions incluant donc marges internes et bordure, il suffira de partager en deux la largeur totale disponible. Une marge interne de 20 pixels de chaque côté complète ces propriétés communes.

L'article ⑦ correspond à la partie principale de la page, d'où une largeur de 70 %. Par conséquent, il reste une largeur de 30 % pour la colonne latérale `aside` ⑧. Celle-ci possède une marge interne du haut plus importante (80 pixels), et les textes et images qu'elle contient seront centrés.

Le pied de page `<footer class="pied">` ⑨ possède des marges internes de 10 pixels, une bordure du haut grise, ainsi qu'une image de fond, créée ici en

CSS 3 par un dégradé linéaire.

Le menu `<ul id="menu">` 10, dont les puces sont supprimées, est de type flexbox. Aux lignes du menu 11 est attribuée la propriété `flex: 1 1 auto;`, qui aurait pu être raccourcie en `flex: auto;` pour que l'ensemble des lignes soit ajusté à la largeur du menu `ul`. Si besoin, chaque élément du menu sera agrandi (première valeur 1) ou réduit (deuxième valeur 1) pour réaliser cette adaptation ; l'ajustement s'effectuera en augmentant ou diminuant l'espace disponible autour du contenu (valeur `auto`).

Ces liens du menu 12, standards ou déjà visités, sont des blocs avec des marges internes qui ajustent leur hauteur et avec un centrage horizontal du texte, ainsi qu'une bordure de droite pour marquer une séparation entre eux. Leur mise en forme concerne l'écriture (Arial et gras en taille 1 rem, soit 16 pixels), la suppression du soulignement automatique des liens, ainsi qu'une couleur `brown` agrémentée d'une ombre blanche sur le texte. Cette ombre comporte uniquement un décalage horizontal de 2 pixels, sans décalage vertical ni effet de flou.

Au passage de la souris sur ces liens 13, c'est un dégradé linéaire gris foncé qui sert de fond, et le texte du lien apparaît en blanc – donc sans ombre blanche –, sinon la finesse de l'écriture en serait affectée. La figure 8-13 montre en détail le fonctionnement de ce menu.



FIGURE 8-13 Un zoom sur le menu de notre site : au passage de la souris, un dégradé gris foncé vient mettre en relief le lien, dont le texte s'écrit alors en blanc pour davantage de lisibilité.

Dans le contenu de la page, les titres de niveau 1 `<h1>` 14 sont en Arial de couleur orange et en taille 1.5 rem (soit 150 % ou 24 pixels), tandis que les sous-titres `<h2>` 15 sont en Arial de couleur `brown`, soit rouge foncé, et en taille 1.2 rem (soit 120 %, environ 19 pixels). La couleur de ce deuxième niveau de titre étant la même que celle du menu, cela donne une certaine unité au graphisme.

Enfin, deux classes `16` sont définies, `gauche` et `droite`, pour placer les images sur la gauche ou sur la droite, bien que la classe `droite` ne soit pas utilisée dans notre page d'exemple. Une illustration flottante est automatiquement « habillée » sur le côté par le texte qui suit. Chaque fois, une marge est définie du côté opposé au flottement, pour obtenir un espacement entre le texte et le bord de l'image.

Notre page est maintenant terminée, avec la mise en place de cette feuille de styles. Les autres pages du site sur Hong Kong pourront reprendre la même base de contenu et de mise en forme ; seul le contenu effectif de la page (délimité par la balise `section` de classe `contenu`) sera à modifier.

Comme tous ceux présentés dans ce chapitre, cet exemple pourra vous servir de base pour créer votre propre site. Il ne vous reste plus qu'à faire travailler votre imagination, votre sens créatif, votre clavier et votre souris !

Cependant, il y a encore moyen de progresser, si vous êtes partant ! En effet, vous aurez certainement envie de sophistiquer votre site, pour qu'il s'adapte automatiquement aux petits écrans des tablettes et smartphones : le chapitre qui suit vous donnera la marche à suivre.



FIGURE 8–14 Exemples de sites utilisant soit un menu horizontal (<http://www.phoxo.com/en> et <https://portableapps.com>), soit un menu vertical (<https://fr.wikipedia.org> et <http://www.freac.org>)

chapitre 9

Un site web pour les mobiles



Toutes les mises en page que nous avons examinées jusqu'ici sont destinées à des écrans d'ordinateur. Qu'en est-il de l'affichage sur des appareils mobiles, smartphones et tablettes numériques de toutes dimensions ? Il faudra leur affecter des styles spécifiques et changer certains éléments de contenu.

SOMMAIRE

- ▶ **Les contraintes du Web mobile**
- ▶ **Adaptation de la mise en page**
- ▶ **Responsive et media queries CSS 3**
- ▶ **Adaptation pratique d'un site pour le Web mobile**

Les contraintes du Web mobile

Lorsque nos visiteurs ne sont pas devant leur ordinateur, ils se servent d'un appareil mobile pour se connecter à Internet. La consultation de sites web sur de petits écrans est de plus en plus fréquente, à partir d'un smartphone ou d'une tablette numérique.

Il nous faudra donc prévoir une adaptation de notre site, pour qu'il soit lisible et d'une navigation agréable à partir de ces appareils.

- Les pages web doivent se charger rapidement, pour tenir compte d'un débit de connexion parfois limité : exit donc les images de très grande taille, comme certaines images de fond. De même, les fichiers multimédias, son et vidéo, ne seront à fournir que sur demande.
- Toujours pour la même raison, il vaut mieux éviter les animations trop gourmandes en ressources, dont celles conçues en Flash. Notez que cette technologie, qui n'est pas reconnue sur certains modèles et de moins en moins prévue par défaut sur les navigateurs, tend à devenir obsolète. Elle sera avantageusement remplacée par des techniques d'animation en HTML 5 et CSS 3, ou bien en JavaScript avec la bibliothèque *jQuery*.
- Pour que le texte reste lisible, il faut tenir compte du contraste des couleurs et se méfier des images de fond.
- Si le contenu est disposé sur plusieurs colonnes, celles-ci peuvent soit conserver leurs proportions, grâce à des largeurs données en pourcentages, soit être organisées différemment dans la fenêtre ou bien occuper toute la largeur de l'écran, étant alors placées les unes sous les autres.
- Le menu doit s'adapter à l'affichage et être présenté de façon plus compacte sur les petits écrans : sur plusieurs lignes par exemple, ou alors remplacé par un bouton qui fait apparaître ou disparaître ce menu (pour réaliser ce comportement, nous serons amenés à écrire une petite fonction JavaScript, qui sera appelée par un clic sur ce bouton).

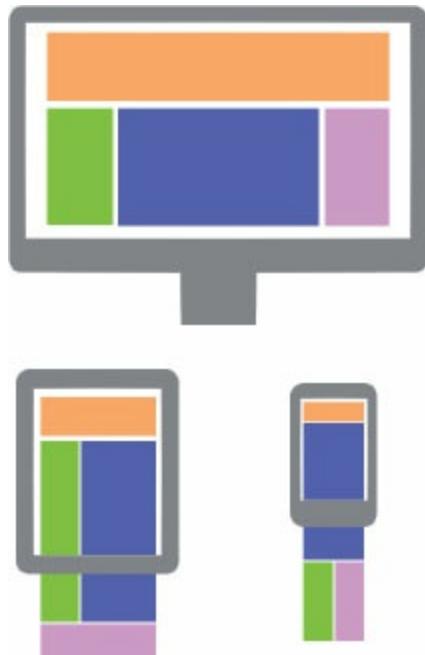


FIGURE 9–1 Adaptation d'un site à différentes tailles d'écrans, par la modification de la position des blocs (d'après une image de Wikimédia)

Adaptation de la mise en page

Pour adapter un site aux écrans mobiles, il suffit d'ajouter, dans la feuille de styles initiale, des règles spécifiques pour les petites largeurs d'écran. Une légère adaptation du code HTML est nécessaire, avec une fonction JavaScript minimale pour l'appel du menu.

Avant d'entrer dans le détail de cette technique, nous allons l'aborder en douceur, en examinant comment il est possible de créer une feuille de styles séparée, destinée à certains appareils spécifiques. Même si la méthode simplifiée qui suit est peu utilisée dans la pratique, nous allons la survoler, pour nous familiariser avec la syntaxe de base que nous développerons par la suite.

Le sélecteur @media

La règle `@media screen { ... }` est prévue pour réserver un ensemble de styles à la lecture sur écran. C'est de très loin l'utilisation la plus courante, mais certaines pages sont destinées à l'impression ou à la lecture par synthèse vocale. La règle de sélection `@media` s'écrit :

- `@media screen` pour la lecture sur écran ;
- `@media print` pour les imprimantes ;
- `@media speech` pour la synthèse vocale ;
- `@media all` pour tous les appareils, sans distinction.

Autrefois, le mot-clé `handheld` était destiné à sélectionner les appareils mobiles. Mais son sens était trop vague, il a donc été peu adopté par les fabricants et, finalement, il ne fait plus partie des normes du W3C. Dans le chapitre suivant, nous parlerons des propriétés CSS spécifiques à l'impression et à la lecture par synthèse vocale des pages, ce qui correspond à ces deux mots-clés `print` et `speech`.

Règles de styles pour différents types d'écran avec @media

```
<style>
  @media all {
    styles communs, applicables dans tous les cas
  }
  @media screen {
    styles pour écrans d'ordinateur
  }
  @media print {
    styles adaptés aux imprimantes
  }
  @media speech {
    styles adaptés à la synthèse vocale
  }
</style>
```

D'une manière similaire, le mot-clé `media` est utilisable en tant qu'attribut, associé à la balise `<link>` dans l'en-tête de la page, pour sélectionner la feuille de styles qui convient à chaque type d'appareil.

Balises `<link>` pour différents terminaux

```
<link rel="stylesheet" href="styles-communs.css" media="all">
<link rel="stylesheet" href="styles-ecran.css" media="screen">
<link rel="stylesheet" href="styles-imprimante.css" media="print">
<link rel="stylesheet" href="styles-sonores.css" media="speech">
```

La règle `@import` peut également tirer parti de cette possibilité. Le mot `media` n'est pas utilisé, seul le type de terminal est mentionné :

```
@import "styles-communs.css" all;
@import "styles-ecran.css" screen;
@import "styles-imprimante.css" print;
@import "styles-sonores.css" speech;
```

Il ne s'agit ici que d'une première approche de la sélection des styles en fonction du type d'appareil. Les CSS 3 ont repris et amélioré la méthode précédente pour nous permettre une sélection plus détaillée, notamment en fonction des largeurs d'écran : c'est ce que nous allons découvrir à présent.



FIGURE 9–2 Entre ordinateur, tablette et smartphone, une grande diversité de largeurs d'écran (source : Wikimédia)

Responsive et media queries CSS 3

Deux mots barbares dans un seul titre, n'est-ce pas un peu trop ? Rassurezvous, ils ne sont là que pour attiser votre curiosité ! Vous serez certainement friands de quelques explications au sujet de ces techniques, dont vous avez peut-être déjà entendu parler et qui sont apparues avec la diversité des écrans.

En CSS 3, il est possible d'affecter des styles spécifiques en fonction des caractéristiques des terminaux utilisés. Cette sélection de médias, appelée en anglais *media queries*, permet notamment d'utiliser une feuille de styles adaptée aux dimensions de l'écran utilisé.

Quant à l'expression très à la mode *responsive web design* (en français, *conception adaptative*), elle signifie que notre site sera lisible sur différentes tailles d'écrans, sa mise en page et son contenu étant adaptés grâce à ces fameux media queries...

Un exemple pour commencer

Les media queries vont nous aider à sélectionner les terminaux en fonction d'un ou plusieurs critère(s), pour leur appliquer des styles adaptés. Sans le savoir, nous les avons déjà utilisés dans les exemples précédents, avec les mots-clés `screen`, `print`, `speech` et `all`.

Le principe de leur écriture restera le même dans les différents cas que nous avons survolés : à l'intérieur d'une feuille de styles, dans la balise `<link>` ou pour la règle `@import`. Nos mots-clés désignant le type d'écran sont toujours utilisables et d'autres sont apparus, qui nous serviront pour un test plus précis des caractéristiques du terminal.

Voici un premier exemple qui nous donne un aperçu de cette méthode :

```
| @media screen and (max-width: 800px) {  
|   ... (règles de styles ici) ...  
| }
```

Il s'agit ici d'appliquer des styles aux médias de type écran (mot-clé `screen`) et dont la largeur de fenêtre ne dépasse pas 800 pixels (condition `max-width: 800px`). Le mot-clé `and` indique que les deux conditions doivent être réunies.

Cette règle s'écrira de la même façon lorsque nous voudrons réserver une feuille de styles spécifique `styles2.css` à ces terminaux, avec la balise `<link>` :

```
| <link rel="stylesheet" href="styles2.css" media="screen and (max-width:  
|   800px)">
```

ou bien avec la règle `@import` :

```
| @import "styles2.css" screen and (max-width: 800px);
```

Syntaxe des media queries

Nous sommes prêts à aborder la syntaxe générale des media queries. Voici donc la liste des mots-clés que nous utiliserons pour élaborer les conditions correspondant aux sélections recherchées.

Le préfixe `min-` ou `max-`, placé devant le nom de la caractéristique, signale une valeur minimale ou maximale. Par exemple, la condition `width` sur la largeur d'écran se décline en `min-width` et `max-width`.

TABLEAU 9–1 Conditions utilisées dans les media queries CSS 3

Mots-clés	Signification	Utilisation
<code>all</code> , <code>screen</code> , <code>print</code> , <code>speech</code> , <code>only screen</code> , <code>only print</code> , <code>only speech</code>	Tous les terminaux (<code>all</code>), écran (<code>screen</code>), impression (<code>print</code>), synthétiseur vocal pour la lecture sonore des pages (<code>speech</code>)	L'habitude est d'utiliser <code>only screen</code> (spécifique aux CSS 3) au lieu de <code>screen</code> écrit seul, qui existait en CSS 2 et présentait le risque d'une reconnaissance partielle des conditions par d'anciennes versions de navigateurs.

TABLEAU 9–1 Conditions utilisées dans les media queries CSS 3 (suite)

Mots-clés	Signification	Utilisation
<code>width: ...</code> <code>height : ...</code>	Largeur (<code>width</code>) ou hauteur (<code>height</code>) en pixels de la fenêtre (y compris pour une fenêtre réduite sur un ordinateur)	En général, c'est la largeur qui est utilisée, avec <code>min-</code> ou <code>max-</code> . Exemple : <code>min-width: 900px</code> Attention : voir plus loin la remarque sur les navigateurs mobiles qui simulent une dimension supérieure à la réalité,

		puis rétrécissent la page obtenue
orientation: portrait orientation: landscape	Type d'orientation de l'écran, en mode portrait (<code>portrait</code>) ou paysage (<code>landscape</code>)	Peu utilisable, car mal pris en compte par les navigateurs mobiles
aspect-ratio	Rapport largeur / hauteur de la fenêtre	Rapport du type 4/3, 16/9, 1280/720... Exemple : <code>aspect-ratio: 16/9</code>
color	Nombre de bits utilisés pour coder chaque couleur d'un pixel (<code>color: 0</code> pour un écran en noir et blanc)	Exemple : <code>min-color: 8</code>
color-index	Nombre de couleurs possibles (« nombre d'entrées dans la table des couleurs », égal à 0 si aucune table de couleurs n'est utilisée)	Exemple : <code>min-color-index : 256</code>
monochrome	Nombre de bits pour coder un pixel en niveaux de gris (autant de niveaux que 2 élevé à la puissance de ce nombre de bits)	Par exemple, pour 128 niveaux de gris ou plus : <code>min -monochrome: 7</code>
resolution	Précision de l'impression (média <code>print</code>), en points par pouce (<code>dpi</code> , dot per inch) ou en points par centimètre (<code>dpcm</code> , dot per cm).	Exemple : <code>print and (min-resolution: 300dpi)</code>
scan: progressive scan: interlace	Pour un média de type	Exemple :

	<code>tv</code> : soit progressive pour un affichage progressif des images, de la première à la dernière ligne, soit <code>interlace</code> pour l'entrelaçage des tubes cathodiques, avec affichage des lignes impaires puis des lignes paires.	<code>tv and (scan: interlace)</code>
<code>(grid)</code> ou <code>grid: 0</code> <code>grid: 1</code>	une grille de caractères donc une seule police (<code>grid</code> ou <code>grid: 0</code>) ou affichage plus courant point par point donc multipolices (<code>grid: 1</code>)	Exemple : <code>print and (grid)</code>

Les préfixes `min-` et `max-` peuvent s'appliquer à toutes ces conditions pour élargir leur champ d'action, exceptées les caractéristiques `orientation`, `scan` et `grid`.

Lorsque plusieurs de ces conditions sont utilisées, elles peuvent être groupées à l'intérieur de parenthèses et reliées entre elles par des opérateurs logiques :

- `and` pour le « et », lorsque les deux conditions sont nécessaires ;
- une virgule (,) pour le « ou », si au moins une des conditions est requise ;
- le mot `not` pour le « non », de façon à obtenir le contraire de la condition initiale (ce `not` étant à placer *avant* la parenthèse de la condition).

Dans l'exemple suivant, extrait d'une feuille de styles, les images sont désactivées sur les écrans dont la largeur n'est pas au minimum de 480 pixels :

```
@media only screen and not (min-width: 480px)
  { img { display: none; }
}
```

La première condition `only screen` est à peu près équivalente à `screen`, pour désigner l'affichage sur écran. C'est une habitude qui a été prise, parce que `only screen` est apparu en CSS 3, alors que le mot-clé `screen` existait déjà en CSS 2. Les anciens navigateurs, limités aux CSS 2, auraient alors pu prendre en compte cette règle, mais ne pas savoir en interpréter correctement la suite.

Vous retrouverez tous les détails des media queries sur le site du W3C, à l'adresse suivante : <https://www.w3.org/TR/css3-mediaqueries>.

ATTENTION Les mobiles simulent des largeurs d'écran fictives !

Sur un smartphone, le test de la largeur de fenêtre avec `width` ou `min-width` peut ne pas fonctionner tel quel.

En effet, le navigateur mobile donne souvent des valeurs bien supérieures à celles de l'appareil, lorsqu'il fournit le *viewport*, c'est-à-dire les dimensions de la fenêtre. Par exemple, la largeur indiquée peut être de 960 pixels, alors qu'elle est en réalité de 480 pixels. Ayant récupéré la page web à cette dimension fictive, le navigateur réduit le contenu obtenu avant de l'afficher.

La solution consiste à régler le zoom à 1 et à étendre la largeur de la fenêtre à la dimension obtenue. C'est la page HTML qui effectuera cette demande, par l'insertion dans son en-tête (entre les balises `<head>` et `</head>`) de la balise `<meta>` suivante :

```
<meta name="viewport" content="initial-scale=1.0">
```

Dans tous les cas, il sera utile d'effectuer des tests avec différentes configurations, pour vérifier la bonne prise en compte de nos sélections par media queries.

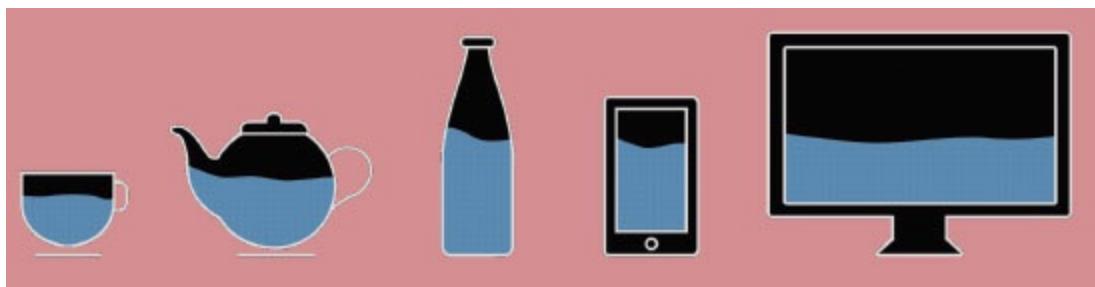


FIGURE 9–3 Tout comme l'eau, le contenu d'une page web prend la forme du terminal qui l'affiche. D'après une illustration de Stéphanie Walter (source : Wikimédia) et une analogie de Josh Clark, sur une citation originale de Bruce Lee : « Si vous mettez de l'eau dans une tasse, elle devient la tasse ».

Application aux navigateurs mobiles

Avec la possibilité de tester la largeur d'affichage disponible, nous voici équipés pour attribuer des styles spécifiques aux navigateurs mobiles, sur Android, iOS ou autres.

Pour affecter la feuille de styles `style-mobile.css` aux fenêtres dont la largeur est inférieure ou égale à 480 pixels, nous pourrons utiliser les media queries de la façon suivante, ici avec la balise `link` :

```
<link rel="stylesheet" href="style-mobile.css" media="only screen and (max-width:480px)">
```

La même syntaxe pourra bien sûr s'appliquer aux règles équivalentes commençant par `@import` ou par `@media`.

Notez qu'il est courant d'effectuer ces adaptations à l'intérieur de la feuille de styles globale, qui comprend alors les styles de base suivis de plusieurs règles `@media`, ce qui correspond aux adaptations du site à différentes gammes de largeurs de fenêtre. Dans tous les cas, nous devrons garder à l'esprit les limitations des appareils mobiles : taille de l'écran réduite, navigation tactile, débit parfois limité.

Le marché de l'Internet mobile étant en perpétuelle évolution et les téléphones mobiles renouvelés plus souvent que les ordinateurs, nous pouvons considérer que cette norme *media queries* est comprise dans sa version CSS 3 par la plupart des navigateurs installés sur ces appareils.



FIGURE 9-4 Le site <http://www.april.org> (association APRIL pour la promotion des logiciels libres) affiché sur différents types d'écrans

Simplicité de l'affichage et de la navigation, légèreté du contenu : voilà qui nous change des habitudes qui tendent à rendre les sites de plus en plus sophistiqués. Mais n'est-ce pas un exercice attrayant que de construire des pages légères, tout en leur donnant belle allure ? Nous allons d'ailleurs appliquer cette technique dès à présent, à partir de l'exemple qui suit.

Adaptation pratique d'un site pour le Web mobile

Comment partir d'un site existant et l'adapter aux écrans de petite taille ? Nous allons reprendre notre site sur Hong Kong, avec lequel nous avons terminé le chapitre précédent, et le modifier pour qu'il s'adapte au Web mobile.

Ajouter une version pour mobile à un site existant

Notre site de référence sur Hong Kong possède une largeur fixe qui n'est pas pensée pour les petits écrans. C'est pourquoi, avant de nous lancer dans un codage spécifique, il nous faut réfléchir sur la méthode à suivre.

Il n'est pas question ici de créer un deuxième site pour les mobiles, car cela doublerait le travail ultérieur de mise à jour des pages. Ce sont bien les mêmes pages qui s'afficheront sur tous les types d'écrans, mais de façon différente et au moyen de quelques aménagements, et ce grâce à la performance des feuilles de styles CSS 3.

Tout d'abord, reprenons les points à ne pas oublier pour réussir une adaptation au Web mobile.

- Toutes nos images sont légères : nous pourrons donc les conserver, sauf l'image de fond de la balise `body`, puisqu'elle sera invisible sur des affichages de petite largeur.
- Les fonds constitués de couleurs unies ou de dégradés ne gênent pas la lecture du texte, qui restera bien lisible sur les smartphones.
- Le contenu effectif de la page possède une largeur de 1 200 pixels, qu'il faudra transformer en 100 % pour des fenêtres de dimensions inférieures.
- Les deux colonnes de texte conserveront leurs proportions, puisque leurs largeurs sont déjà données en pourcentages, ceux-ci pouvant d'ailleurs être ajustés au cas par cas. Pour une taille d'écran réduite, ces colonnes s'afficheront l'une sous l'autre, en pleine largeur.
- Enfin, le menu pourra rester horizontal et tenir sur une seule ligne, si nous diminuons les marges de gauche et de droite pour chaque lien, ainsi que la taille des caractères. Lorsque la largeur disponible deviendra trop petite, ce menu deviendra déroulant et sera affiché par un clic sur un lien permanent **Menu**, puis masqué de la même manière.

Test du site mobile

Notre réalisation pourra être testée de trois façons : en réduisant la fenêtre de notre navigateur, en utilisant un simulateur ou à l'aide d'un téléphone réel.



FIGURE 9–5 Le site <https://framasoft.org> affiché de façon dynamique sur différentes tailles d'écrans, simulations proposées en ligne par le site « Am I responsive? » (<http://ami.responsivedesign.is/>)

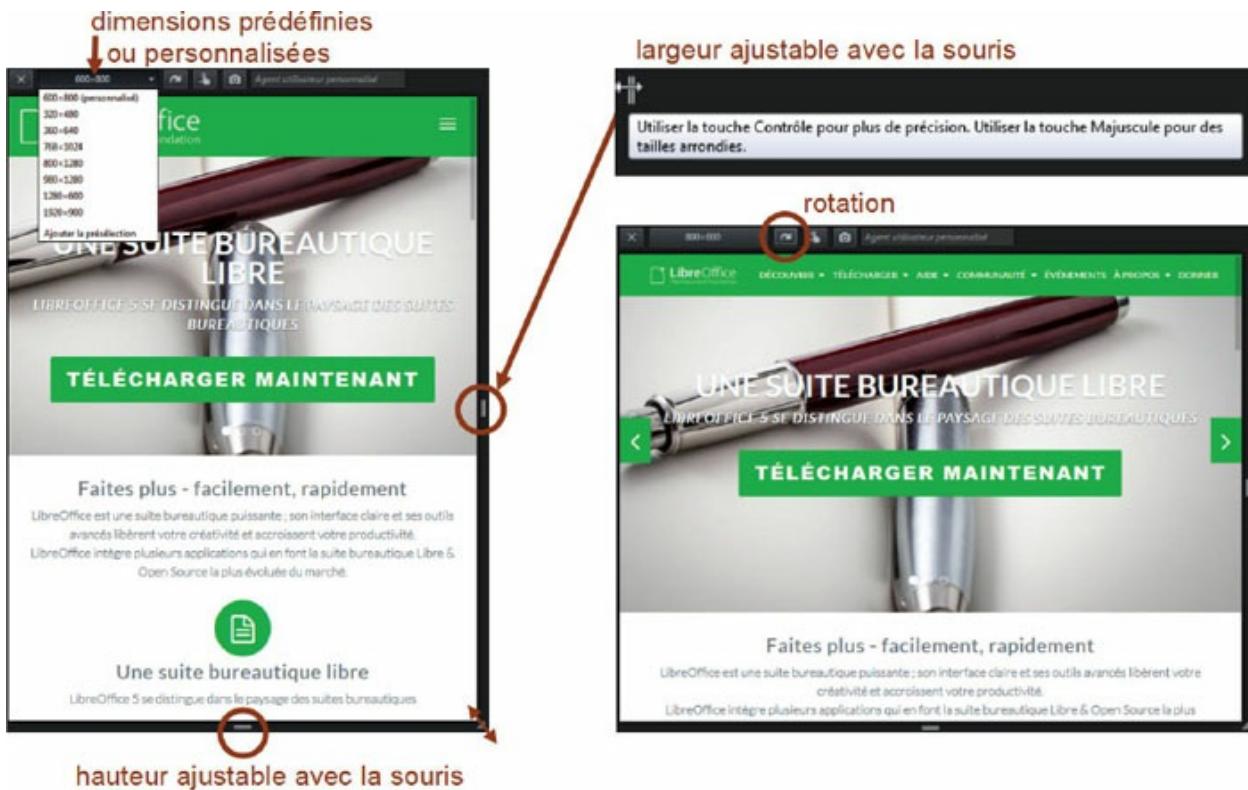


FIGURE 9–6 Le site <http://fr.libreoffice.org> affiché en mode portrait et paysage, avec les fonctions de simulateur d'écran mobile proposées par Firefox avec son raccourci *Ctrl + Maj + M*

SIMULATION Test d'un site pour les mobiles

Pour obtenir la bonne résolution et simuler un téléphone portable sur ordinateur, plusieurs solutions sont possibles.

- Il est facile de quitter le mode « plein écran » et de régler la taille de la fenêtre avec la souris.
- Certains navigateurs proposent des fonctions de test sur mobile : la fenêtre se réduit et la largeur d'écran est affichée directement ; les dimensions peuvent être ajustées, soit par une liste déroulante (valeurs prédefinies), soit à l'aide de la souris.

Ces possibilités sont offertes, entre autres, par Firefox (*Ctrl + Maj + M*, puis retour à l'affichage normal par la même combinaison de touches) et par Chrome (raccourcis à bascule *Ctrl + Maj + I* pour afficher les outils de développement, puis *Ctrl + Maj + M*).

En complément de ces méthodes, l'extension *MeasureIt* nous fournit

un outil de mesure en pixels affichant les dimensions d'une zone sélectionnée avec la souris. Il s'agit d'un module complémentaire qui peut être ajouté à Firefox, Chrome et Safari. Très pratique pour mesurer les dimensions de certains éléments, il est disponible à l'adresse suivante : <http://frayd.us/extensions/measureit>.

Par ailleurs, plusieurs sites permettent de tester l'affichage de pages web sur différents types d'écrans mobiles, pour une vue plus précise du résultat et à condition bien sûr que le site soit en ligne, par exemple :

- le site Responsivepx (<http://responsivepx.com/>) est à la fois simple d'emploi et évolutif : il suffit de saisir l'adresse du site à tester et de choisir les dimensions du terminal, largeur et hauteur ;
- la page Screenfly (<http://quirktools.com/screenfly/>) fonctionne de façon semblable, à condition d'activer les options **Custom screen size** et **Allow scrolling** pour choisir les dimensions du simulateur et autoriser le défilement vertical ;
- le site « Am I responsive? » (<http://ami.responsivedesign.is/>) propose un aperçu dynamique du site choisi pour différentes catégories d'écrans (ordinateur de bureau, portable, smartphone, tablette), le résultat pouvant être agrandi par un zoom sur le navigateur (touche **Ctrl** appuyée, utiliser la molette de la souris) ;
- d'autres services en ligne émulent certains modèles de smartphones, à choisir dans une liste qui, hélas, devient rapidement obsolète : Emulateur mobile (<http://www.emulateurmobil.com>), MobileTest (<http://mobiletest.me/>), mobiReady (<https://ready.mobi/>), Piressive (<http://app-en.orson.io/>), etc.

Adaptation d'une page aux mobiles

De façon à adapter automatiquement notre page sur Hong Kong aux petits écrans, nous allons modifier un tout petit peu son code HTML et surtout ajouter des règles de styles spécifiques qui changeront la présentation de la page en fonction de la taille de la fenêtre.

Code HTML

La partie de l'en-tête comporte deux ajouts : la mise à l'échelle réelle de 100 % et une fonction permettant d'afficher ou masquer le menu, pour les petits écrans.

Début du code HTML et en-tête de la page

```
<!DOCTYPE html>
<html>
<head>

    <meta charset="utf-8">
    <title>Hong Kong</title>

    <meta name="viewport" content="initial-scale=1.0"> ①
    <link rel="stylesheet" href="style.css" />

    <script> ②
        function menumobile() {
            bloc = document.getElementById("menu");
            if (bloc.className=="afficher")
                { bloc.className = "masquer"; }
            else { bloc.className = "afficher"; }
        }
    </script>

</head>
```

La balise `meta` ① définit une échelle de 1 (réglage du zoom à 100 % du *viewport*) ; le navigateur connaîtra ainsi la largeur réelle de fenêtre d'affichage. Comme nous l'avions vu, les écrans mobiles indiquent parfois une largeur d'écran supérieure à la réalité, pour capter l'ensemble d'une page web et la réduire avec un zoom arrière.

Sur les écrans de petite taille, la fonction JavaScript `menumobile()` ② générera

l'accès au menu, qui sera affiché ou masqué par un clic sur un bouton **Menu**. Ce clic appellera la fonction `menumobile()` qui récupère l'élément d'identifiant `menu` (donc la balise `ul` contenant le menu) pour lui attribuer la classe `masquer`, s'il est affiché, ou la classe `afficher` dans le cas contraire. C'est la feuille de styles qui se chargera de rendre le menu visible ou non, en fonction de sa classe.

EN DÉTAIL Explication de la fonction JavaScript `menumobile()`

Cette fonction `menumobile()` effectue les opérations suivantes.

- La variable `bloc` est la balise `<ul id="menu">` : c'est l'élément de la page repéré par l'identifiant `menu`, fourni par la fonction `getElementById('menu')`, qui cherche dans la page un élément d'identifiant donné, ici `menu`. Elle va servir pour le test et le changement de la classe CSS de ce bloc menu.
- Si cet élément a pour classe `afficher`, alors le menu est déjà déroulé, et c'est donc la classe `masquer` qui lui est attribuée.
- Dans le cas contraire, le menu est masqué, car sa classe est déjà égale à `masquer` ou bien parce qu'il n'a pas encore de classe (état initial, lorsqu'on n'a jamais cliqué sur le bouton **Menu**) ; c'est donc la classe `afficher` qui est attribuée à cet élément.

Le contenu initial de la balise `<body>` est presque inchangé par rapport à sa version initiale, celle du chapitre précédent. Seule est ajoutée une balise `<a>` pour créer le bouton **Menu** sur les écrans de petite largeur.

Fin du code HTML et corps de la page

```
<body>
<div class="global"> ③

    <header class="entete">
         ④
    </header>

    <nav class="blocmenu">
        <a href="#" id="menutitre" class="mobile" onclick="menumobile(); return
false;">Menu</a> ⑤
        <ul id="menu"> ⑥
            <li><a href="index.html">Présentation</a></li>
            <li><a href="histoire.html">Histoire</a></li>
```

```

<li><a href="visites.html">Visites</a></li>
<li><a href="specialites.html">Spécialités</a></li>
<li><a href="shopping.html">Shopping</a></li>
<li><a href="transports.html">Transports</a></li>
<li><a href="contacts.html">Contacts</a></li>
</ul>
</nav>

<section class="contenu"> 7

<article> 8
  <h1>Modernité et tradition</h1>

  <h2>Autour d'un rocher</h2>
  <p>
    
    Paragraphe de texte...
  </p>
  <p>Paragraphe de texte... </p>

  <h2>Une vie bouillonnante</h2>
  <p>
    
    Paragraphe de texte...
  </p>
  <p>Paragraphe de texte...</p>
</article>

<aside> 9
  <p></p>
  <p></p>
  <p><em>Images Wikimédia</em></p>
</aside>

</section> <!-- Fin du bloc section.contenu -->
<footer class="pied">
  <p></p>
</footer>

</div> <!-- Fin du bloc div.global -->
</body>
</html>

```

Nous avons donc ajouté un lien 5 avant la balise `` qui contient le menu. Initialement masquée, cette balise `a` sera transformée en un bouton **Menu** sur les écrans de petites tailles, pour afficher ou masquer le menu de navigation.

Sa classe `mobile` nous permettra en CSS de ne montrer ce bouton que sur les fenêtres de largeur inférieure à 640 pixels.

Un clic sur ce bouton appelle la fonction JavaScript `menumobile()` qui était venue compléter l'en-tête HTML. L'instruction `return false;` permet de désactiver le lien `href="#"` qui renvoie en haut de la page.

Dans la feuille de styles qui suit, nous adapterons, entre autres, la largeur du conteneur global ③ et de l'image d'en-tête ④, ainsi que les lignes du menu ⑥. Dans la section du contenu ⑦, les colonnes `article` ⑧ et `aside` ⑨ s'adapteront à la fenêtre, soit en restant côté à côté, soit en se plaçant l'une sous l'autre. Par sécurité, la taille des images sera limitée à la largeur disponible dans leur colonne.

Feuille de styles

Dans un premier temps, l'adaptation de la feuille de styles consistera à définir des dimensions maximales au lieu de dimensions fixes :

- la largeur du conteneur global ne sera plus fixée avec la propriété `width`, mais seulement limitée pour les grands écrans avec `max-width` ; elle pourra donc se réduire et s'adapter aux plus petites tailles d'affichage ;
- cette même propriété `max-width` permettra de limiter la taille des images à 100 % de leur conteneur, ce qui leur évitera de déborder de leur colonne (une hauteur automatique sera également attribuée à ces images, pour éviter toute distorsion) ;
- l'image d'en-tête, qui s'affichait auparavant dans ses dimensions natives, remplira 100 % de la largeur du conteneur global, pour pouvoir s'adapter à un rétrécissement éventuel de cette largeur utile ; dans une grande fenêtre, elle n'aura pas pour autant à s'afficher au-delà de sa taille initiale, puisque ce conteneur est limité en largeur.

Ensuite, la mise en forme de trois classes (`.mobile`, `.afficher`, `.masquer`) servira à gérer les éléments spécifiques aux petits écrans, notamment le menu : le bouton **Menu** ne doit apparaître que sur les mobiles, les deux dernières classes permettant d'afficher ou masquer le menu complet suivant l'action du visiteur sur ce bouton.

Enfin, et grâce aux media queries, nous pourrons régler les propriétés du menu (ses marges, la taille des caractères) et des colonnes (leur répartition

dans la page) pour les adapter aux différentes tailles d'écrans mobiles, en définissant trois plages de largeurs.

Par rapport à la feuille de styles initiale, les parties modifiées sont celles mises en gras, ainsi que les trois règles media queries situées à la fin. Ces quelques ajouts et transformations suffiront pour adapter notre page à tout type d'écran.

Feuille de styles CSS adaptative pour le site Hong Kong

```
.mobile { display: none; } ①
body, div, ul { margin: 0; padding: 0; }

body { background-image: url(hk-body.jpg);
      font-family: 1.2rem Garamond, Times, serif; }

.global { margin: 0 auto; max-width: 1200px; ②
          background-color: whitesmoke;
          border: solid 1px lavender; border-top: none; }

.entete img { display: block; width: 100%; } ③

#menu { background: linear-gradient(lightgray, whitesmoke);
         border-bottom: solid 1px lavender; }

.contenu { letter-spacing: -0.31rem; }
.contenu * { letter-spacing: normal; }
.contenu img { max-width: 100%; height: auto; } ④
article, aside { box-sizing: border-box;
                 display: inline-block; vertical-align: top; padding: 20px; }
article { width: 70%; }
aside { width: 30%; padding-top: 80px;
        text-align: center; }

.pied { padding: 10px;
        background: linear-gradient(whitesmoke, lightgray);
        border-top: solid 1px lightgray; }

/* Détails pour le menu de navigation */
#menu { display: flex; list-style-type: none; }
#menu li { flex: 1 1 auto; }

#menu li a, #menu li a:visited {
    display: block; text-align: center;
    padding: 15px 0; border-right: solid lightgray 1px;
    font-family: Arial, sans-serif;
```

```

    font-size: 1rem; font-weight: bold;
    color: brown; text-shadow: white 2px 0 0;
    text-decoration: none; }

#menu li a:hover {
    background: linear-gradient(silver, dimgray);
    color: white; text-shadow: none; }

/* Contenu de la page */
h1 { font-family: Arial, sans-serif;
    font-size: 1.5rem; color: #d06b19; }
h2 { font-family: Arial, sans-serif;
    font-size: 1.2rem; color: brown; }

.droite { float: right; margin-left: 10px; }
.gauche { float: left; margin-right: 10px; }

/* RESPONSIVE */

@media only screen and (min-width: 640px)
    and (max-width: 799px) { 5
body { background-image: none; }
#menu li a, #menu li a:visited {
    padding: 10px 0; font-size: 0.9rem; }
article { width: 60%; }
aside { width: 40%; }
}

@media only screen and (max-width: 639px) { 6
body { background-image: none; }
.mobile { display: block; } 7

#menu, #menu.masquer { display: block; height: 0; } 8
#menu.afficher { height: 427px; } 9

#menu { transition: height 1s; overflow: hidden; 10
    border-bottom: none; background: none; }
#menu li { line-height: 30px; 11
    background: linear-gradient(lightgray, whitesmoke);
    border-bottom: solid 1px lavender; }
#menu li a, #menu li a:visited {
    text-align: left; padding-left: 10%; } 12

#menutitre { display: block; line-height: 3rem; 13
    background: linear-gradient(silver, gray);
    text-align: center;
    font: bold 1.1rem/2.5rem Arial, sans-serif;
    color: white; text-decoration: none; }

```

Nous allons tout d'abord faire le tour des changements qui concernent la première partie, qui va du début du code CSS jusqu'aux règles @media.

Les éléments de classe `mobile` ① sont masqués dans la feuille de styles standard : cela ne concerne ici que le bouton **Menu** destiné aux smartphones, créé à partir d'une balise `a` de classe `mobile`.

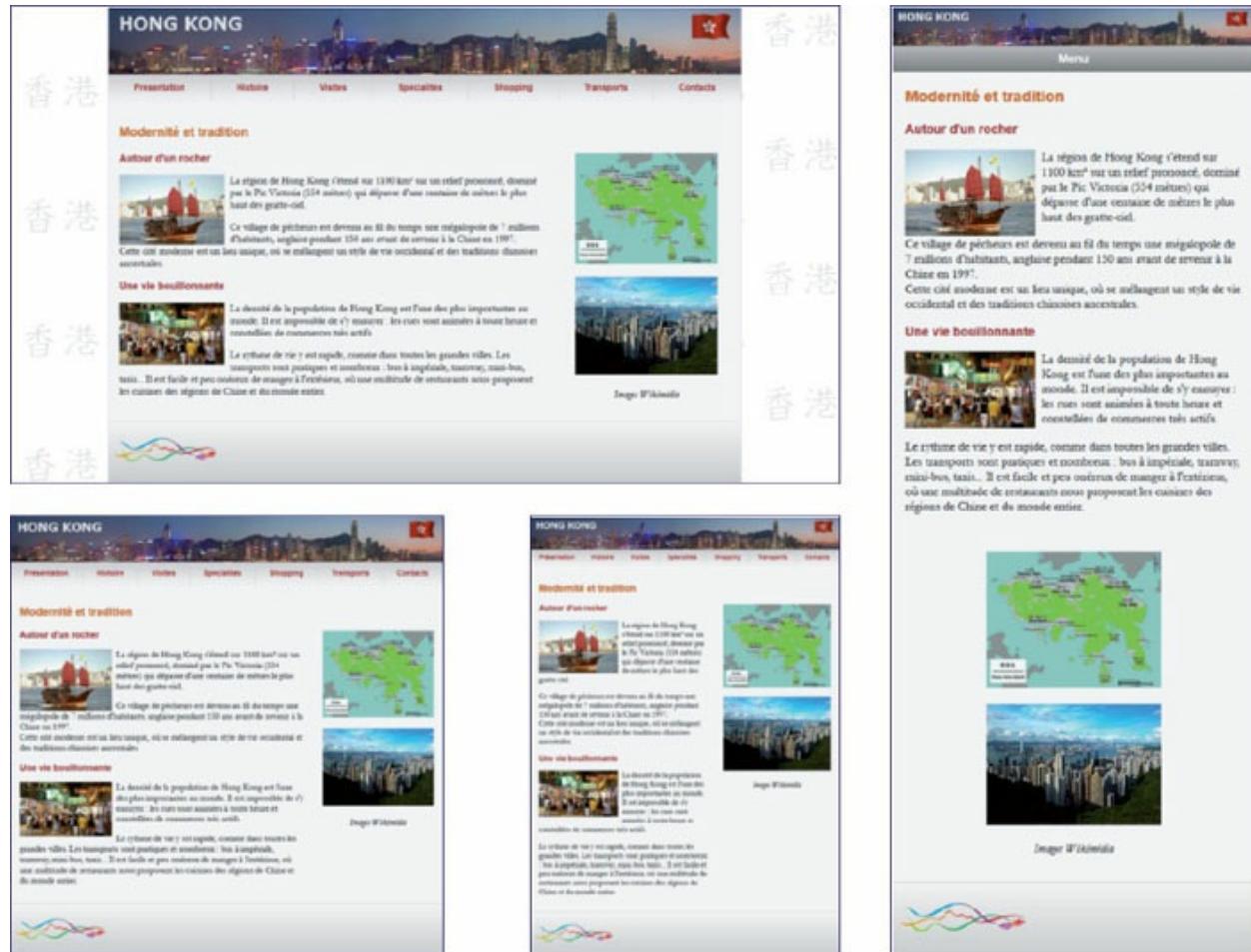


FIGURE 9-7 Le site sur Hong Kong dans sa version adaptative, affiché avec différentes tailles d'écrans : l'en-tête et le menu s'adaptent à la largeur disponible. Sur les petites dimensions d'écrans, le menu devient déroulant à partir d'un bouton **Menu** et les deux colonnes s'affichent l'une sous l'autre en pleine largeur.

La largeur du conteneur global n'est plus fixée à 1 200 pixels avec la propriété `width` : il s'agit maintenant d'une *largeur maximale* fournie par la propriété `max-width` ②. Cela ne change rien pour les écrans d'ordinateurs, mais

ce conteneur s'adaptera maintenant à la largeur des écrans, si elle est inférieure à cette dimension.

Pour la même raison, l'image d'en-tête voit sa largeur ③ ajustée à 100 % de celle de son conteneur, qui est la balise `header`, elle-même ayant la largeur du bloc `div` de classe `global`. Sur un ordinateur, cette image conserve donc une largeur de 1 200 pixels, puis elle s'adapte à tous types d'écrans. Ainsi, elle reste entièrement visible dans tous les cas.

La règle ④ concerne toutes les balises `img` placées dans la section de classe `contenu`, c'est-à-dire dans la partie principale de la page. C'est une des bases de la conception adaptative (*responsive design* en anglais) : toute image ne doit pas dépasser la largeur de la fenêtre, si petite soit-elle. Une largeur maximale `max-width` de 100 % ne change pas la dimension des images sur un écran d'ordinateur et résout ce problème de façon très simple !

En revanche, les images ainsi réduites peuvent se trouver déformées si leur hauteur `height` a été définie, par exemple en tant qu'attribut dans la balise ``. C'est pourquoi cette règle ④ est complétée par `height: auto`; pour que les images conservent leurs proportions dans tous les cas.

Les styles qui suivent sont identiques à ceux de la version initiale de notre site, ils sont expliqués en détail dans le chapitre précédent.

Pour compléter l'adaptation du site, il nous reste à ajouter des styles spécifiques pour différentes largeurs d'écran, classées par catégories à partir de valeurs en pixels appelées « points de rupture ». Ceux-ci sont déterminés de façon pratique pour chaque site, avec souvent des valeurs classiques liées aux tailles d'écrans courantes, qui sont données par le tableau 9-2, mais qui n'ont rien de contraignant.

TABLEAU 9–2 Quelques largeurs courantes d'écrans

Type de terminal	Largeurs courantes en pixels
Smartphones (mode portrait ou paysage)	480, 640, 720, 854, 1 080, 1 280, 1 440, 1 920, ...
Tablettes (mode portrait ou paysage)	600, 768, 800, 1024, 1 200, 1 280, 1 600, 1 920, ...
Netbooks	1 024, 1 366, 1 920

Ordinateurs portables ou fixes

1 280, 1 366, 1 440, 1 600, 1 920, 2 048, 2 560, ...

Dans la pratique, nous pouvons trouver les points de rupture à utiliser en réduisant progressivement la fenêtre du navigateur et en notant les dimensions à partir desquelles il faut modifier l'affichage. Ces valeurs s'affichent directement dans un navigateur qui propose un aperçu de type « écran mobile » (par exemple, Firefox avec le raccourci *Ctrl + Maj + M*).

Dans notre page sur Hong Kong, compte tenu de la présence possible d'une barre de défilement verticale, nous allons utiliser comme points de rupture les largeurs 800 et 640 pixels. Dans ces deux cas, l'image de fond de la balise `body` est supprimée parce qu'elle n'est plus visible, le contenu prenant toute la largeur de l'écran. Les autres modifications de styles sont fonction des dimensions d'écrans.

De 640 à 799 pixels ⑤, l'espace occupé par les liens du menu est diminué de deux façons : les marges internes du haut et du bas passent de 15 à 10 pixels, tandis que la taille des caractères du menu diminue et vaut `0.9rem`, soit 14,4 pixels. Par ailleurs, les largeurs des colonnes sont légèrement modifiées, à 60 % et 40 % au lieu de 70 % et 30 %, pour éviter que la colonne latérale ne devienne trop étroite.

Pour une largeur de fenêtre qui ne dépasse pas 639 pixels ⑥, nous voici dans la configuration « petit écran », qui nécessite davantage de modifications. En particulier, le menu ne s'affichera que sur commande, à l'aide d'un bouton à bascule **Menu**, qui attribuera la classe `masquer` ou `afficher` à la liste de liens.

- Le menu devient donc déroulant à partir d'un bouton **Menu** de classe `mobile`, qui n'est plus masqué mais affiché comme bloc ⑦.
- L'élément `<ul id="menu">` ⑧ redevient un bloc standard (il était de type `flex` auparavant). Il est rendu invisible par une hauteur nulle, lorsqu'il n'a aucune classe (affichage initial) ou quand il a la classe `masquer` (après deux clics sur le bouton **Menu**, par exemple). Attention : il est important de ne pas insérer d'espace dans le sélecteur `#menu.masquer`, car c'est ici le même élément `ul` qui possède l'identifiant `menu` et la classe `masquer`.
- Lorsqu'il possède la classe `afficher` (après un clic sur le bouton **Menu**), ce bloc `` d'identifiant `menu` prend une hauteur de 427 pixels ⑨, correspondant à 7 lignes de menu de 61 pixels chacune (une hauteur de ligne de 30 pixels, des marges internes de 15 pixels en haut et en bas,

plus la bordure de 1 pixel qui sera ajoutée en bas de chaque ligne). Comme précédemment, cette règle concerne une même balise `` qui aura à la fois l'identifiant `menu` et la classe `afficher`, il n'y a donc pas d'espace dans le sélecteur `#menu.afficher`.

- Toujours à ce même bloc `` d'identifiant `menu` sont attribuées les propriétés suivantes [10](#) : *transition* d'une seconde pour la propriété `height`, de façon à animer le déroulement du menu, débordement `overflow` masqué par la valeur `hidden` (sinon, une hauteur nulle de `` n'empêcherait pas les lignes de menu de s'afficher), pas de bordure en bas et aucune de couleur de fond.
- En ce qui concerne les lignes `` [11](#) de ce menu, les changements sont les suivants : une hauteur de ligne définie à 30 pixels, un fond sous forme de *dégradé* en CSS 3 et une *bordure* de séparation en bas de chaque ligne.
- Les liens [12](#) sont à présent alignés à gauche au lieu d'être centrés, avec une marge interne de gauche de 10 % pour une meilleure esthétique. Au passage de la souris sur les liens, le dégradé prévu pour les grands écrans est conservé.
- Le bouton **Menu**, créé avec la balise `<a>` d'identifiant `menutitre` [13](#), n'avait pas encore de mise en forme : c'est un bloc qui s'étend par défaut sur toute la largeur de l'écran, sa hauteur de ligne est de `3rem` et il possède un *fond dégradé* en CSS 3. Le texte est *centré*, écrit en police *Arial* et en *blanc, sans soulignement* pour le lien. Sa taille est de `1.1rem` et son interligne de `2.5rem`, de façon à obtenir un centrage vertical (dans la propriété `font`, les deux dimensions séparées par une barre représentent respectivement la taille de police et la valeur de l'interligne).
- Enfin, les deux colonnes de contenu, balises `<article>` et `<aside>` [14](#) ne seront plus côté à côté mais l'une sous l'autre, puisqu'elles occupent chacune toute la largeur de l'écran. La valeur `0` pour la propriété `padding-top` sert à annuler la marge interne du haut du bloc `<aside>`.



FIGURE 9–8 Détail du menu lorsque la largeur d'écran se réduit : adaptation automatique grâce à la flexbox, accompagnée ensuite d'une réduction de la taille des caractères (avec des marges verticales plus petites), puis disparition du menu qui devient déroulant et s'affiche ou se masque par un clic sur le bouton *Menu*

CHOIX Méthodes pour afficher ou masquer le menu

Pour les plus petits écrans, c'est la propriété `height` qui est appliquée pour masquer ou afficher le menu, passant de 0 à une valeur fixe en pixels. D'autres méthodes auraient pu être utilisées, mais sans le même effet d'animation.

- Il est possible de faire disparaître ou apparaître la balise `` d'identifiant `menu` en utilisant simplement la propriété `display` avec les valeurs `none` et `block`. Cependant, il n'y a plus d'animation dans ce cas, car la propriété CSS `transition` ne s'applique pas à `display`.
- Une autre solution consiste à attribuer à la hauteur `height` de la balise `` la valeur `auto`, lorsque le menu doit être affiché. Celui-ci s'agrandit alors automatiquement en fonction du nombre de lignes, la hauteur de chaque ligne étant fixée. Cette solution a l'avantage d'être plus souple et de ne pas dépendre du nombre de lignes dans le menu. Hélas, la transition ne s'applique pas à la valeur `auto` de la propriété `height` ; elle ne vaut que pour le passage d'une valeur numérique à une autre. Cette technique nous prive donc là encore de l'animation du menu déroulant.
- Si nous reportons la transition et la propriété `height` sur la hauteur des lignes `` (passage de 0 à une valeur donnée), le conteneur `` pourra conserver sa hauteur automatique, et il ne sera plus

nécessaire de la calculer. En revanche, si une animation apparaît effectivement, ce n'est pas le déroulement habituel d'une liste déroulante : toutes les lignes apparaissent et s'agrandissent en même temps, ce qui n'est pas l'effet escompté.

- Il existe une astuce qui permet de conserver la valeur `auto` pour la hauteur de la liste ``, tout en préservant le déroulement animé du menu : il suffit d'utiliser la propriété `max-width` pour la balise `` et d'y appliquer l'effet de transition. La valeur initiale de cette propriété est toujours `0` et la valeur finale est arbitraire, suffisamment grande pour ne pas tronquer la liste en cas d'ajout de lignes supplémentaires. Mais l'effet d'enroulement animé risque d'être retardé lors de la réduction de la liste, car la réduction commence audelà de la limite du menu, dans l'espace fictif compris entre la valeur `maxwidth` choisie et la hauteur réelle du menu.

Animation du menu

Dans cette dernière version pour les plus petits écrans, un clic sur le bouton **Menu** active la fonction JavaScript qui attribue à la balise `` la classe `afficher`, puis un autre clic change cette classe en `masquer`. C'est ainsi que la hauteur du menu passe de zéro à une valeur fixée, puis de nouveau à zéro.

La transition CSS d'une seconde appliquée sur la propriété `height` entraîne alors un déroulement progressif de ce bloc, puis une disparition animée dans le sens inverse.

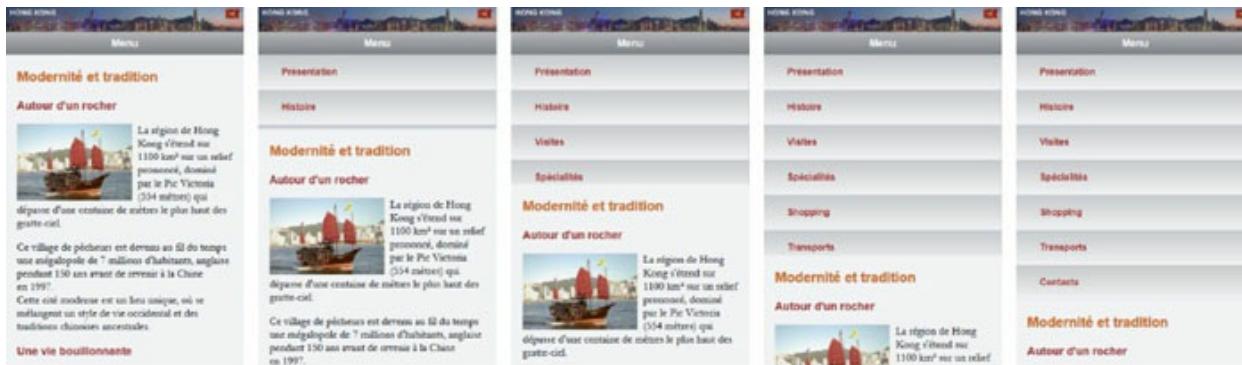


FIGURE 9–9 Animation du menu en mode « smartphone », grâce à la propriété CSS `transition`. Un clic sur le bouton *Menu* permet d'afficher ou de masquer le menu, avec la transition animée dans les deux sens.

Nous avons à présent un site adapté à toutes les tailles d'écrans d'ordinateurs, de tablettes et de smartphones. Grâce aux media queries qui sélectionnent automatiquement les styles en fonction de la taille de la fenêtre, les mêmes pages serviront aux différentes versions et il n'y aura qu'un seul fichier HTML à mettre à jour si le contenu doit être modifié.

Vous connaissez maintenant les méthodes pour adapter un site à tous types d'écrans, ce sera (presque) un jeu d'enfant de l'appliquer à votre site. Le code CSS devient évidemment un peu plus complexe et la mise à jour des pages doit tenir compte de la version mobile, notamment en ce qui concerne l'insertion d'images de grande taille et autres éléments qui augmentent le poids des pages.

Il existe d'autres cas de figure où des styles particuliers sont nécessaires, notamment pour les documents destinés à être imprimés, ou encore lorsqu'il s'agit de gérer la lecture orale des pages. Les propriétés spécifiques à ces pages sont décrites dans le chapitre qui suit.

Bonnes Pratiques du Web Mobile



W3C® **Mobile Web**
Initiative

mobiliser en 10 points

- Créer** pour un seul Web
- Respecter** les standards Web
- Eviter** les risques connus
- Connaitre** les limitations des terminaux
- Optimiser** la navigation
- Vérifier** graphiques & couleurs
- Penser** "petit"
- Utiliser** le réseau avec parcimonie
- Aider & guider** les utilisateurs
- Penser** aux utilisateurs en déplacement

W3C® **Mobile Web**
Initiative



Supported by MobiWeb2.0
FP7 EU Project



FIGURE 9–10 Le principe de la conception adaptative ou responsive design, d'après un schéma d'Alex Gaslang (<http://designbeep.com>)



FIGURE 9–11 Les bonnes pratiques pour le Web mobile sont résumées sur le site du W3C, à l'adresse https://www.w3.org/2007/02/mwbp_flip_cards.html.fr.

chapitre 10

Différents types de médias



Le principal média utilisé est l'affichage sur un écran, mais il en existe d'autres, notamment l'impression et le son.

SOMMAIRE

- ▶ **Types de médias**
- ▶ **Média paginé : styles CSS 2 et CSS 3 pour l'impression**
- ▶ **Média sonore : fonctions audio CSS 3**

Jusqu'alors, nous nous sommes intéressés à un seul type de média : celui qui produit un *affichage sur un écran*, que ce soit avec un ordinateur ou sur un terminal mobile.

Les feuilles de styles permettent cependant de gérer d'autres médias. En particulier, en dehors de l'affichage sur écran, les CSS proposent des propriétés pour l'impression et la diffusion sonore des pages web.

Types de médias

À l'intérieur d'une feuille de styles interne ou externe, il est possible de déclarer le type de média concerné par un groupe de propriétés. Cette déclaration est facultative ; elle s'effectue sous la forme :

```
| @media xxx { /* xxx= type de média concerné */
|   ... Propriétés pour ce type de média ...
| }
```

Nous avons déjà étudié cette notation dans le chapitre précédent, notamment pour les types de médias `screen` (écran d'ordinateur) et `all` (tout type d'appareil).

Exemple

```
<style>
  @media screen {
    * { font-style: Arial, sans-serif; }
  }
  @media print {
    * { font-style: "Times New Roman", serif; }
  }
  @media all {
    * { background-color: white; }
  }
</style>
```

Le tableau suivant nous rappelle quels sont les types de médias possibles.

TABLEAU 10–1 Les médias à notre disposition

Règle <code>@media</code>	Type de médias concernés
<code>all</code>	Tous les médias sont concernés par les propriétés indiquées.
<code>braille</code>	Écran tactile pour les non-voyants
<code>print</code>	Impression papier
<code>screen</code>	Écran couleur (affichage classique sur un ordinateur)

`speech`

Lecture sonore (par synthèse vocale ou diffusion de son)

La norme CSS 2 distinguait les types de médias `aural` (diffusion de son) et `speech` (lecture par synthèse vocale). En CSS 2.1 comme en CSS 3, le premier de ces deux types a disparu, et seule la règle `@media speech` est utilisée pour l'ensemble des propriétés sonores.

À NOTER Emplacement de la déclaration du type de média

Le type de média peut être spécifié :

- à l'intérieur d'une feuille de styles interne, comme dans l'exemple précédent
- dans l'appel à une feuille de styles externe, comme ci-après :

```
<link rel="stylesheet" media="print" href="impression.css">
```

Nous allons étudier plus en détail les propriétés associées à la règle `print`, qui s'applique à la fois aux pages imprimées et à toute visualisation sur écran en mode Aperçu, dont les livres électroniques affichés sur des liseuses, tablettes numériques destinées à la lecture d'ouvrages.

Média paginé : styles CSS 2 et CSS 3 pour l'impression

Les propriétés associées au média `print` sont liées à l'impression d'un document ou à son affichage sous forme paginée.

- L'aperçu avant impression d'une page web, de même que son résultat imprimé, peut ainsi être adapté au papier et adopter une présentation différente de celle du site consulté sur l'écran.
- Ces propriétés sont également utiles pour la génération automatique d'un document PDF sur un site web, généralement à l'aide d'un programme écrit en PHP.
- Les liseuses affichant des livres au format électronique (*eBook* en anglais) et disposant d'un accès Internet devraient en principe interpréter la feuille de styles de type `print`, ainsi que les logiciels permettant la conversion d'une page web vers le format ePub spécifique aux liseuses. Cependant, la lecture des pages web « comme sur un ordinateur » est un argument commercial qui pousse souvent constructeurs et éditeurs à utiliser le type de média `screen` au lieu de `print`.

Pour le document concerné, il s'agit principalement de définir sa taille, les sauts de page, ainsi que la gestion des veuves et des orphelines.

Vous ne connaissez pas l'histoire de la veuve, ni celle de l'orpheline ? Préparez un mouchoir, je vous la raconte (rassurez-vous, elle finit bien !). Dans les deux cas, il s'agit d'un paragraphe dont une ligne est toute seule, perdue sur une autre page, ce qui est fâcheux d'un point de vue esthétique.

Pour la veuve, c'est une pauvre petite ligne qui se trouve seule en haut d'une page, alors que tout le reste du paragraphe auquel elle appartient se trouve en bas de la page précédente. Dans ce cas, il pourra être demandé qu'une ligne se détache du paragraphe pour lui tenir compagnie.

Quant à l'orpheline, elle est isolée en bas d'une page, tandis que toute sa famille, c'est-à-dire le reste du paragraphe, se trouve en haut de la page suivante. Alors, grâce à la bonté du logiciel, elle pourra le rejoindre sur cette nouvelle page.



FIGURE 10–1 *Veuve et orpheline*

PARENTHÈSE L'effet «veuve et orpheline» dans un traitement de texte

En général, les traitements de texte gèrent les veuves et les orphelines. C'est pourquoi, dans certains cas, l'ajout d'un saut de ligne dans une page entraîne le passage de deux lignes sur la page suivante, ou parfois la suppression d'une ligne de la page courante fait revenir deux lignes de la page suivante.

Voici les propriétés liées à l'impression, qui seront donc en général placées entre accolades, dans la partie `@media print{ }` de la feuille de styles. Le détail de ces normes se trouve sur le site du W3C, à l'adresse www.w3.org/TR/css3-page.

Gestion des veuves

La propriété `widows` permet de définir le nombre *minimum* de lignes d'un même paragraphe à afficher ensemble en haut d'une page. En dessous de ce nombre, la suppression des « veuves » est activée : une ou plusieurs ligne(s) du même paragraphe, se trouvant au bas de la page précédente, est(sont) transférée(s) en haut de la nouvelle page.

TABLEAU 10–2 Propriété `widows`

Propriété	<code>widows</code>
Exemple	<code>body { widows: 3; }</code>
Valeurs possibles	Nombre entier (valeur par défaut : 2) indiquant le nombre minimum de lignes qui peuvent rester en haut d'une page
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, indiquez 2.

Gestion des orphelines

La propriété `orphans` indique le nombre minimum de lignes d'un même paragraphe à afficher en bas d'une page. Si ce nombre n'est pas atteint, ces lignes sont considérées comme « orphelines » et sont donc transférées en haut de la page suivante.

TABLEAU 10–3 Propriété `orphans`

Propriété	<code>orphans</code>
Exemple	<code>body { orphans: 3; }</code>
Valeurs possibles	Nombre entier (valeur par défaut : 2) indiquant le nombre minimum de lignes qui peuvent rester en bas d'une page
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, indiquez 2.

Saut de page avant

Associée à un élément de la page, la propriété `page-break-before` permet d'indiquer s'il doit être imprimé en haut d'une nouvelle page.

TABLEAU 10–4 Propriété `page-break-before`

Propriété	<code>page-break-before</code>
Exemple	<code>.chapitre { page-break-before: always; }</code>
Valeurs possibles	<code>auto</code> : saut de page s'effectuant automatiquement lorsque le bas de la page est atteint (valeur par défaut) <code>always</code> : saut de page obligatoire avant <code>avoid</code> : pas de saut de page avant <code>left</code> : un ou deux saut(s) de page avant, pour que l'élément <i>qui suit</i> l'élément concerné se trouve dans une page de gauche <code>right</code> : un ou deux saut(s) de page avant, pour que l'élément <i>qui suit</i> l'élément concerné se trouve dans une page de droite
Héritage	<i>Non</i>

Saut de page après

La propriété `page-break-after` sert à préciser si un élément de la page doit être ou non suivi d'un saut de page.

TABLEAU 10–5 Propriété `page-break-after`

Propriété	<code>page-break-after</code>
Exemple	<code>.conclusion { page-break-after: left; }</code>
Valeurs possibles	<code>auto</code> : saut de page s'effectuant automatiquement lorsque le bas de la page est atteint (valeur par défaut) <code>always</code> : saut de page obligatoire après <code>avoid</code> : pas de saut de page après <code>left</code> : un ou deux saut(s) de page après, pour que l'élément <i>qui suit</i> l'élément concerné se trouve dans une page de gauche <code>right</code> : un ou deux saut(s) de page après, pour que l'élément <i>qui suit</i> l'élément concerné se trouve dans une page de droite
Héritage	<i>Non</i>

Coupure par un saut de page

Certaines parties de texte perdent leur sens ou sont beaucoup moins lisibles si elles se trouvent coupées par un saut de page. Cette coupure peut être évitée grâce à la propriété `page-break-inside`.

TABLEAU 10–6 Propriété `page-break-inside`

Propriété	<code>page-break-inside</code>
Exemple	<code>table { page-break-inside: avoid; }</code>
Valeurs possibles	<code>auto</code> : saut de page s'effectuant automatiquement lorsque le bas de la page est atteint, l'élément concerné pouvant donc se trouver sur deux pages (valeur par défaut) <code>avoid</code> : pas de saut de page <i>à l'intérieur</i> de l'élément concerné : il doit se trouver entier sur une seule page (si ce contenu est plus grand que la page, la partie qui déborde est coupée)
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utilisez <code>auto</code> .

Dimensions d'une page

Les dimensions et l'orientation des pages sont spécifiées à l'aide de la propriété `size`. Le sélecteur peut être par exemple `@page` pour indiquer « toutes les pages » ; nous examinerons plus loin les types de sélecteur possibles.

TABLEAU 10–7 Propriété `size`

Propriété	<code>size</code>
Exemple	<code>@page { size: A4 landscape; }</code>
Valeurs possibles	<p>Les valeurs possibles, pour la taille et l'orientation de la page, sont les suivantes :</p> <p><code>auto</code> : dimensions et orientation de la page standards (valeurs par défaut définies dans le navigateur) ;</p> <p>deux valeurs <code>largeur</code> et <code>hauteur</code> séparées par un espace (généralement exprimées en cm ou en in - pas de %).</p> <p>Taille standardisée (<code>A5, A4, A3; B5, B4, letter, legal, ledger</code>)</p> <p>Si une taille standardisée est utilisée pour la page, elle peut être accompagnée de son orientation : <code>landscape</code> (orientation « paysage ») ou <code>portrait</code> (orientation « portrait »).</p>
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utilisez <code>auto</code> .

Sélecteur de page

Le sélecteur `@page` permet de spécifier les caractéristiques de pagination (taille, orientation portrait ou paysage, marges...). Il peut être complété par `:first`, `:left` ou `:right` pour n'attribuer les propriétés en question qu'à une partie du document.

Pour toutes les pages

```
| @page { margin: 1.5cm; }
```

Pour la première page

```
| @page :first { margin-top: 5cm; }
```

Pour toutes les pages de gauche

```
| @page :left { margin-right: 2cm; }
```

Pour toutes les pages de droite

```
| @page :right { margin-left: 2cm; }
```

Pages nommées

Il est possible de donner un nom à un type de page et de lui associer des propriétés de mise en forme, en utilisant la syntaxe suivante :

```
| @page nom_de_page_choisi { propriétés_associées}
```

Exemples :

```
| @page paysage { size: A4 landscape; }
| @page formatA5 { size: 10cm 15cm; }
```

Par la suite, un élément peut faire référence à ce nom de page pour en prendre toutes les caractéristiques. Il faut pour cela utiliser la propriété `page`, décrite ci-après.

Référence à un type de page

Lorsqu'un règle @page définit des propriétés en leur associant un nom, comme dans les deux exemples précédents, ce nom peut être affecté à un élément à l'aide de la propriété page. Toutes les propriétés d'impression ainsi définies (dimensions, marges, etc.) seront alors attribuées à cet élément.

ATTENTION Saut de page automatique

Si deux éléments qui se suivent sont associés à des types de page différents, alors un saut de page est automatiquement inséré entre eux.

La même règle est d'ailleurs appliquée par les traitements de texte : il ne peut pas y avoir deux mises en forme différentes pour la même page. Par exemple, dans un texte en mode portrait, si un paragraphe est associé au mode paysage, un saut de page sera automatiquement inséré avant et après.

TABLEAU 10–8 Propriété page

Propriété	page
Exemples	<pre>div.large { page: paysage; } #notice { page: formatA5; }</pre> <p>Les types de page nommés paysage et formatA5 auront été définis précédemment. On aura par exemple :</p> <pre>@page paysage { size: landscape; } @page formatA5 { size: 10cm 15cm; }</pre>
Valeurs possibles	auto : pas de nom de page associée (valeur par défaut), ou nom d'une page défini par @page xxx { ... }
Héritage	Propriété <i>héritée</i> . Pour retrouver la valeur initiale, utilisez auto.

Rotation d'image

Si l'image fournie par un appareil photo ou un scanner est encodée séparément, il n'est pas certain que le matériel utilisé possédera la ressource logicielle nécessaire pour adapter les images à l'orientation de la page à imprimer.

La propriété `image-orientation` intervient alors, permettant de faire pivoter d'un angle donné les images concernées, de 90 degrés lorsque la page est imprimée en mode paysage par exemple.

TABLEAU 10–9 Propriété `image-orientation`

Propriété	<code>image-orientation</code>
Exemple	<code>img { image-orientation: 90deg; }</code>
Valeurs possibles	<code>auto</code> : rotation automatique en fonction de l'orientation de la page (valeur par défaut), ou angle en degrés, positif pour le sens des aiguilles d'une montre, négatif pour le sens inverse (la norme demande aux navigateurs d'interpréter au moins les valeurs <code>0deg</code> , <code>90deg</code> , <code>180deg</code> et <code>270deg</code> , en positif et en négatif).
Héritage	<code>Non</code>

NORMES Propriétés spécifiques à la version CSS 3

Les fonctions liées à l'impression que nous venons d'étudier ont été introduites en CSS 2 et reprises dans la version CSS 3. Seule la propriété `image-orientation` est spécifique à la norme CSS 3.

Nous disposons donc d'une panoplie de propriétés qui facilitent la présentation de données sous une forme paginée. Cependant, avant de les utiliser, il faudra s'assurer de leur interprétation correcte par le logiciel destinataire, car ces fonctions sont encore assez peu utilisées. Il en est de même pour les propriétés audio que nous allons découvrir à présent.

Média sonore : fonctions audio CSS 3

Les propriétés audio, appelées « Aural CSS » ou ACSS, permettent en CSS 3 le paramétrage des sons, pour l'accompagnement du site par des extraits sonores, ainsi que pour la lecture orale des pages écrites, principalement à destination des personnes ayant un handicap visuel. Dans la feuille de styles, elles seront placées à l'intérieur des accolades de la règle `@media speech { ... }`.

Si une utilisation essentielle de ces techniques est la synthèse vocale des pages web pour les personnes malvoyantes, d'autres applications existent, comme l'apprentissage des langues étrangères. Malheureusement, ces propriétés CSS sont peu reconnues par les navigateurs et par les logiciels de lecture à l'écran, que ce soient des logiciels spécifiques ou des extensions de navigateur. Ces normes sont donc très peu utilisées dans la pratique.

Voici un résumé de ces propriétés sonores, dont les valeurs possibles sont indiquées dans l'annexe C. Pour tout détail complémentaire à leur sujet, vous pouvez consulter la page correspondante sur le site officiel du W3C : www.w3.org/TR/css3-speech.

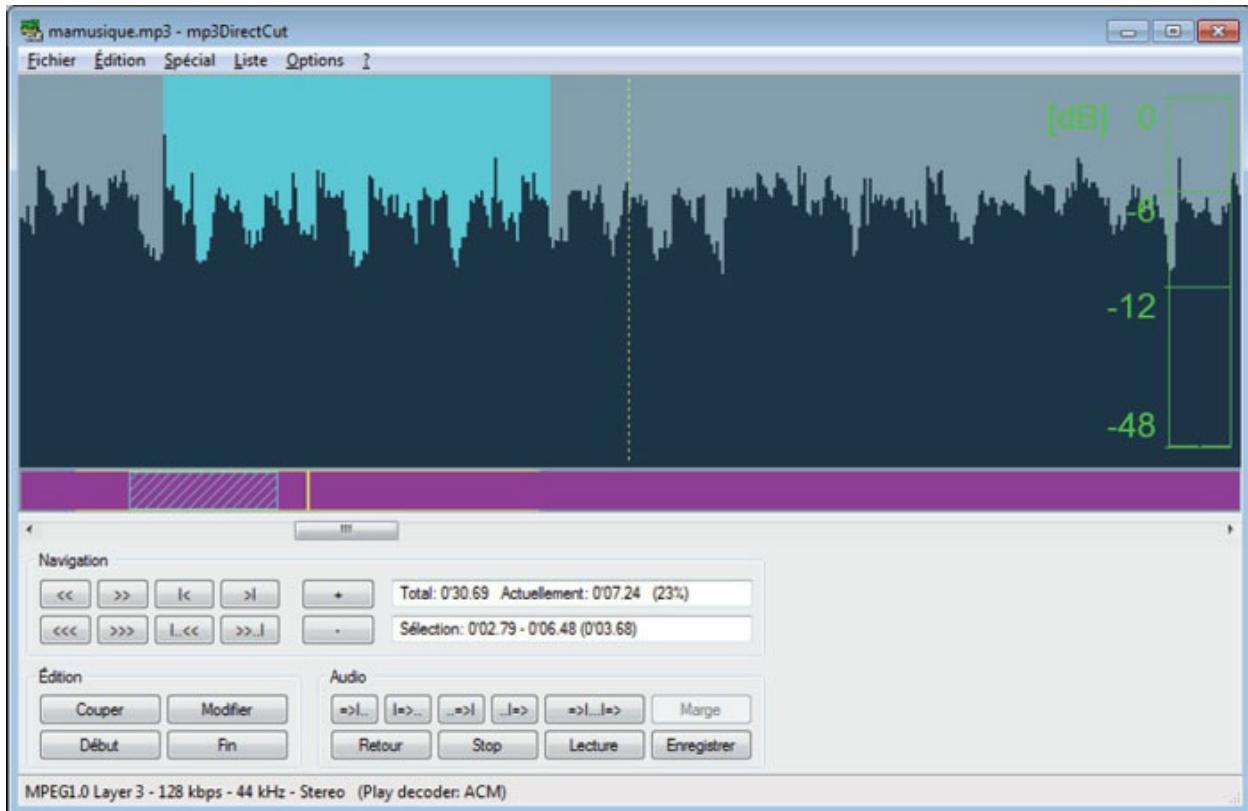


FIGURE 10–2 Copie d'écran du logiciel gratuit mp3DirectCut, qui permet d'éditer directement des fichiers MP3. Il est disponible à l'adresse <http://www.mpesch3.de1.cc/mp3dc.html>.

Les propriétés sonores de type speech proposées en CSS 3 sont ici classées en deux catégories :

- les propriétés de réglage du son et de la voix ;
- les ponctuations audio autour d'un élément de texte lu, pauses et encadrements sonores.

TABLEAU 10–10 Paramétrages du son et de la voix

Propriété	Fonction assurée
<code>voice-volume</code>	Volume de la voix ou des éléments sonores diffusés
<code>voice-balance</code>	Balance gauche/droite du son
<code>speak</code>	Indique si le texte doit être lu ou non par le lecteur vocal.

<code>speak as</code>	Précise si le lecteur vocal doit tenir compte ou non des ponctuations, les interpréter, les épeler ou les ignorer.
<code>voice-family</code>	Types de voix à utiliser (analogue à <code>font-family</code> pour les polices de caractères)
<code>voice-rate</code>	Vitesse de lecture du texte
<code>voice-pitch</code>	Fréquence moyenne à utiliser pour la voix (grave, aigüe...)
<code>voice-range</code>	Étendue des variations de tonalité, pour le type de voix choisi
<code>voice-stress</code>	Importance de l'accentuation de la voix, pour les mots mis en relief dans le texte
<code>voice-duration</code>	Durée de lecture de la totalité de l'élément concerné, incluant tous ses éléments enfants, les icônes sonores de début et de fin, ainsi que les pauses

Si la propriété `voice-duration` est affectée à un élément, sa propriété `voice-rate` est ignorée, ainsi que les propriétés `voice-duration` et `voice-rate` de tous ses éléments enfants, puisque la durée de lecture indiquée concerne la lecture de l'élément tout entier.

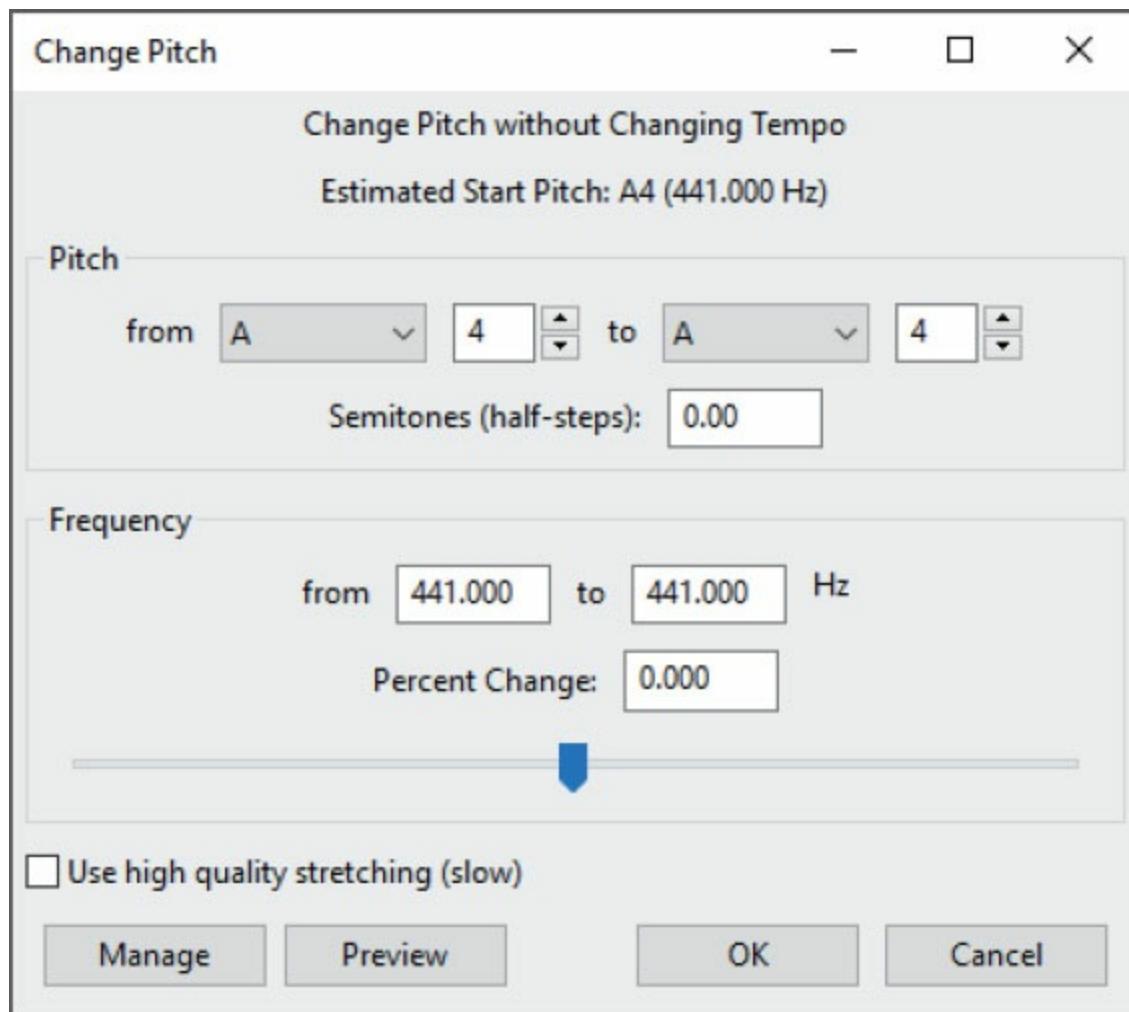


FIGURE 10–3 Le réglage pitch dans le logiciel libre Audacity de traitement du son (extrait de la page http://manual.audacityteam.org/man/change_pitch.html)

TABLEAU 10–11 Ponctuations sonores et pauses

Propriété	Fonction assurée
cue-before	Diffusion, avant l’élément concerné, d’une icône sonore désignée par le nom ou l’adresse d’un fichier son
cue-after	Diffusion, après l’élément concerné, d’une icône sonore désignée par le nom ou l’adresse d’un fichier son
cue	Raccourci pour cue-before et cue-after
	Durée de pause, avant la lecture

<code>pause-before</code>	d'une partie de texte et avant la lecture de l'éventuel élément sonore qui ponctue son début
<code>pause-after</code>	Durée de pause, après la lecture d'une partie de texte et après la lecture de l'éventuel élément sonore qui ponctue sa fin
<code>pause</code>	Raccourci pour <code>pause-before</code> et <code>pause-after</code>
<code>rest-before</code>	Durée de pause, avant la lecture d'une partie de texte mais après la lecture de l'éventuel élément sonore qui ponctue son début
<code>rest-after</code>	Durée de pause, après la lecture d'une partie de texte mais avant la lecture de l'éventuel élément sonore qui ponctue sa fin
<code>rest</code>	Raccourci pour <code>rest-before</code> et <code>rest-after</code>

Les icônes sonores de début et de fin d'un élément, définis par `cuebefore` et `cue-after`, jouent en quelque sorte le rôle de la bordure d'un bloc sur l'écran. En poursuivant cette analogie, les propriétés `rest` et `pause`, situées respectivement à l'intérieur et à l'extérieur par rapport à cet encadrement sonore, jouent le même rôle que les marges intérieures `padding` et extérieures `margin`. Cette correspondance est illustrée par la figure 10-4, qui est un exemple fourni par le site www.w3.org.

ATTENTION Prise en charge incomplète

Ces propriétés sonores étant peu utilisées, il ne faut pas compter sur une diffusion importante des informations sous cette forme. Seule leur interprétation convenable par les principaux navigateurs ou par des logiciels spécifiques permettra à ces styles sonores d'avoir un impact réel.

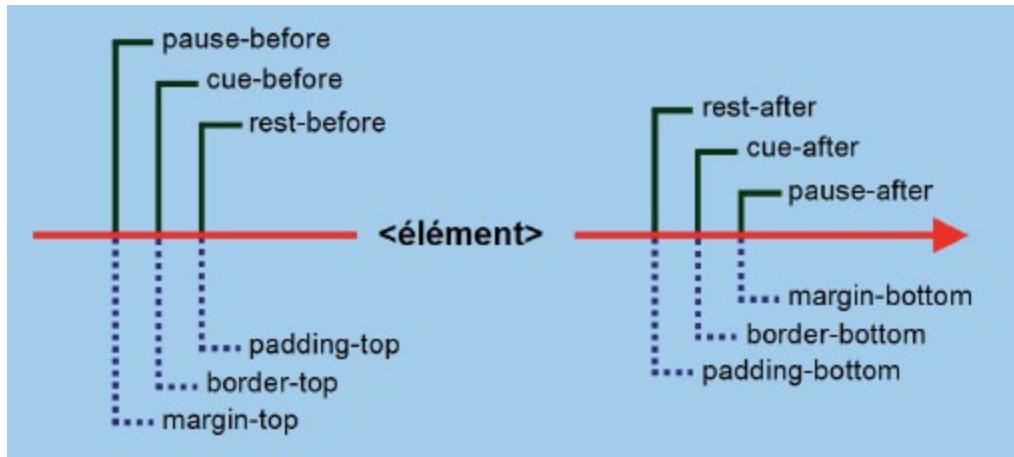


FIGURE 10-4 Analogie entre un bloc de texte, comprenant bordure et marges, et un élément lu par le navigateur, délimité par des icônes sonores espacées par des pauses avant et après

Ce livre se termine par des annexes où vous trouverez des informations sur les couleurs, la façon de tester la reconnaissances des nouvelles normes par les navigateurs, ainsi qu'un résumé des propriétés CSS 2 et CSS 3, puis des références web et bibliographiques.

annexe
A

Couleurs



Dans cette annexe sont répertoriés certains outils en ligne qui nous aideront à choisir et accorder les couleurs. Puis sont présentées les 16 couleurs de base du HTML, et ensuite toutes les couleurs nommées.

SOMMAIRE

- ▶ **Choix et harmonisation des couleurs**
- ▶ **Les 16 couleurs de base**
- ▶ **Liste de toutes les couleurs nommées**

« Des goûts et des couleurs, il ne faut point discuter »... Cependant, quels que soient les choix effectués, il faut ensuite les transcrire ! En général, les éditeurs HTML nous offrent la possibilité de choisir visuellement une couleur et affichent automatiquement le code correspondant.

Il existe par ailleurs un certain nombre d'outils en ligne qui nous aident à sélectionner une gamme de couleurs assorties pour les fonds, les bordures, les puces, les titres, etc. Nous en découvrirons une liste qui, sans être exhaustive, pourra nous rendre service.

Dans la foulée, nous ferons un tour de quelques couleurs connues, en commençant de façon très simple par le tableau des 16 couleurs de base du HTML. Du classique, mais du solide !

Ensuite, ceux qui sont allergiques aux codes numériques, décimaux ou hexadécimaux trouveront la liste complète des couleurs qui portent un nom.

Choix et harmonisation des couleurs

Le choix des couleurs et leur harmonisation est un vaste sujet. Pour nous aider à sélectionner des couleurs qui s'assemblent sans heurter le regard, plusieurs sites nous proposent des palettes assorties et autres informations sur les couleurs.

Voici donc quelques sites utiles autour des couleurs, parmi les plus connus.

- *Colour lovers*, comme son nom l'indique, rassemble les amateurs de couleurs à l'adresse <http://www.colourlovers.com/>. Vous y trouverez d'innombrables palettes de tous les tons, ainsi que des motifs, des formes et des propositions de teintes seules.
- *Paletton*, à l'adresse <http://paletton.com>, nous aide à choisir, dans le cercle chromatique, soit des nuances de couleurs monochromes, soit des assortiments de trois ou quatre couleurs.
- *Adobe Color CC* (qui s'appelait auparavant *Adobe Kuler*) fonctionne sur le même principe et permet d'associer différentes couleurs, toujours en tournant dans le cercle chromatique : <https://color.adobe.com>.
- Le site <http://www.code-couleur.com/> propose différents assortiments, mais donne aussi des informations sur la signification des couleurs.
- *0 to 255*, à l'adresse <http://www.0to255.com/>, nous offre toute une gamme de nuances pour une couleur donnée.
- Bien que sa traduction de l'anglais soit parfois approximative, le site <http://htmlcolorcodes.com/fr/> permet de créer des palettes avec un sélecteur de teinte et différents tableaux de couleurs.
- Enfin, il ne faut pas oublier l'outil « pipette » qui permet de récupérer le code couleur d'une zone donnée sur un site Internet. Il existe pour cela l'outil *ColorZilla*, qui est un plug-in adapté à Firefox et à Chrome. Disponible sur la page <http://www.colorzilla.com/>, il propose en outre un générateur automatique de codes CSS pour créer des dégradés de couleurs.

The screenshot shows two main sections of the ColourLovers website. On the left, a 'Browse Palettes' section displays five color palettes with their names, authors, and statistics (Comments, Favorites, Views, Loves). A mouse cursor is hovering over the 'Compatible' palette by joy_ct_summer. On the right, a 'COLORS' section shows a detailed view of the 'Compatible' palette, including its hex code (#33A999), RGB values (211,169,169), and a preview bar. Below it are other palettes like 'Sugar Hearts You', 'Party Confetti', 'Sugar Champagne', 'Bursts Of Euphoria', and 'Happy Balloons', each with their own hex codes, RGB values, and preview bars.

FIGURE A-1 Le site <http://www.colourlovers.com/> bénéficie d'une communauté qui partage des palettes de couleurs. Un clic sur l'une d'entre elles donne les détails de ses composantes, notamment les codes couleur HTML.

The screenshot shows the 'Ultimate CSS Gradient Generator' tool. At the top, it says 'A powerful Photoshop-like CSS gradient editor from ColorZilla.' and has tabs for 'For Firefox', 'For Chrome', and 'Gradient Generator'. The main interface includes:

- Presets:** A grid of color gradient thumbnails.
- Name:** A text input field containing 'Red Gloss' with a 'save' button.
- Preview:** A horizontal bar showing a red-to-orange gradient.
- Orientation:** A dropdown menu set to 'vertical'.
- Size:** A size input field set to '370 x 50'.
- IE:** A checkbox for Internet Explorer support.
- Stops:** A section for adding color stops. It includes an 'Opacity' slider, a 'Location' input field (e.g., 50%), and a 'delete' button.
- Color:** A color picker, a 'Location' input field (e.g., 50%), and a 'delete' button.
- CSS:** A large text area containing the generated CSS code for the gradient.
- Color format:** A dropdown menu set to 'hex'.
- Comments:** A checked checkbox.

FIGURE A-2 À l'adresse <http://www.colorzilla.com/gradient-editor/>, le site ColorZilla écrit pour nous le code CSS correspondant à un dégradé de couleurs.

Les 16 couleurs de base

Voici, leurs noms français classés par ordre alphabétique, les 16 couleurs de base du HTML.

TABLEAU A–1 Les 16 couleurs de base du HTML

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Blanc	white	#ffffff	rgb(255, 255, 255)
Bleu	blue	#0000ff	rgb(000, 000, 255)
Bleu foncé	navy	#000080	rgb(000, 000, 128)
Bleu-vert	teal	#008080	rgb(000, 128, 128)
Cyan	aqua	#00ffff	rgb(000, 255, 255)
Gris clair	silver	#c0c0c0	rgb(192, 192, 192)
Gris foncé	gray	#808080	rgb(128, 128, 128)
Jaune	yellow	#ffff00	rgb(255, 255, 000)
Marron	maroon	#800000	rgb(128, 000, 000)
Noir	black	#000000	rgb(000, 000, 000)
Rose	fuchsia	#ff00ff	rgb(255, 000, 255)
Rouge	red	#ff0000	rgb(255, 000, 000)
Vert	green	#008000	rgb(000, 128, 000)
Vert brillant	lime	#00ff00	rgb(000, 255, 000)
Vert olive	olive	#808000	rgb(128, 128, 000)
Violet	purple	#800080	rgb(128, 000, 128)

RAPPEL Code RVB

Le code RVB (Rouge - Vert - Bleu) ou RGB en anglais (Red - Green - Blue) consiste à fournir l'intensité de chacune de ces trois couleurs dans l'ordre, de trois façons possibles :

- soit en hexadécimal, chaque composante étant exprimée sur deux chiffres, compris entre `00` et `ff` ;
- soit en décimal, l'intensité de chaque couleur étant codée à l'aide de trois chiffres, compris entre `000` et `255`, avec la fonction `rgb(xx,xx,xx)` ;
- soit encore en pourcentage, puisque dans l'expression `rgb(xx,xx,xx)` le code `xx` de chaque couleur peut être aussi un pourcentage compris entre `0%` et `100%`.

La notation `rgba(xx,xx,xx,x)`, qui inclut un quatrième paramètre compris entre `0` et `1` correspondant à l'opacité, nécessite l'emploi des nombres décimaux compris entre `0` et `255` pour le choix de la couleur.

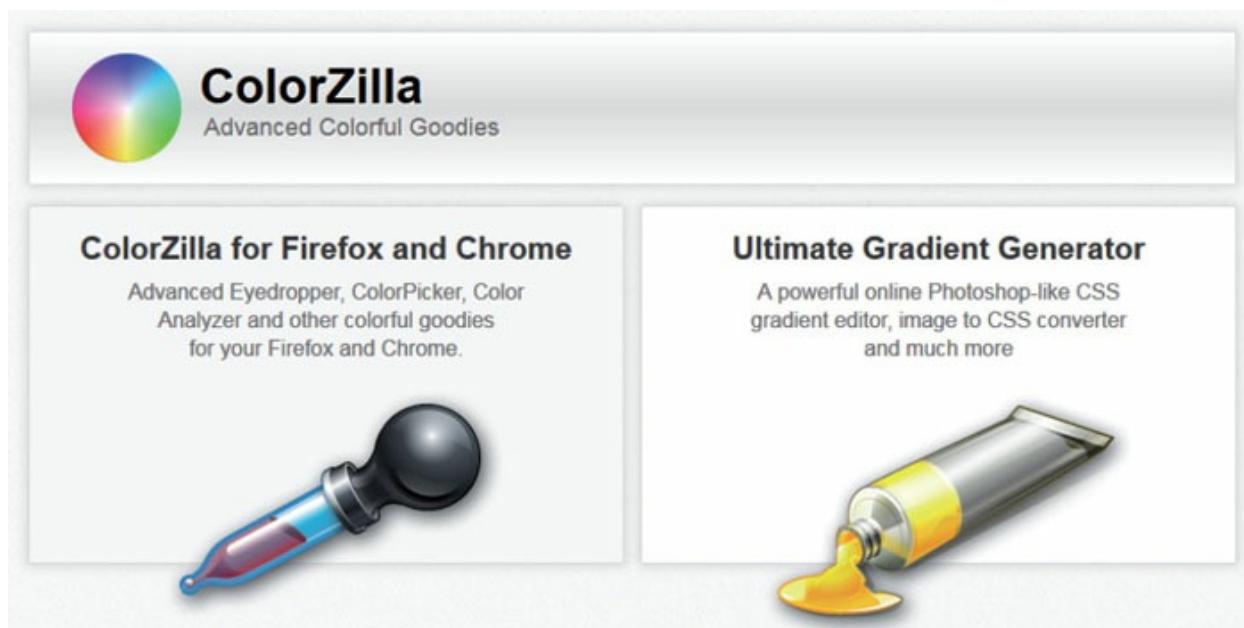


FIGURE A–3 La page d'accueil du site ColorZilla (<http://www.colorzilla.com>), qui nous propose l'outil « pipette » sous forme de module complémentaire pour Firefox ou Chrome, ainsi que son outil automatique de codage CSS pour créer un dégradé de couleurs.

Liste de toutes les couleurs nommées

Sauf pour les couleurs simples ou fréquemment utilisées, le code RVB « Rouge - Vert - Bleu » hexadécimal n'est pas très parlant : à quoi la couleur `#adff2f` ressemble-t-elle ? Même exprimée sous la forme `rgb(173, 255, 47)` ou encore `rgb(68%, 100%, 18%)`, cela ne nous dit pas grand-chose...

Une alternative plaisante consiste donc à utiliser les noms de couleurs prédéfinis, du moins pour celles qui en possèdent un. Pour reprendre l'exemple précédent, le nom `greenyellow` nous indique clairement qu'il s'agit d'un vert qui tire sur le jaune.

Le tableau suivant classe par teinte toutes les couleurs HTML nommées. Il provient du travail très intéressant d'Alain Beyrand, webmestre du site <http://www.pressibus.org>. La page des couleurs est disponible à l'adresse suivante : <http://www.pressibus.org/perso/html/frcouleurs.html>.

Pour voir les couleurs associées à ces noms, consultez ce site Internet ou essayez-les en HTML.

TABLEAU A–2 Couleurs nommées de ton BEIGE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Beige	beige	#f5f5dc	rgb(245, 245, 220)
Beige blanc antique	antiquewhite	#faebd7	rgb(250, 235, 215)
Beige blanc Dalmond	blanchedalmond	#ffebcd	rgb(255, 235, 205)
Beige bisque	bisque	#ffe4ba	rgb(255, 228, 186)
Beige citron-soie	lemonchiffon	#fffacd	rgb(255, 250, 205)
Beige crème de papaye	papayawhip	#ffefdd	rgb(255, 239, 213)
Beige mocassin	moccasin	#ffe4b5	rgb(255, 228, 181)
Beige pêche	peachpuff	#ffdab9	rgb(255, 218, 185)

TABLEAU A-3 Couleurs nommées de ton BLANC

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Blanc	white	#ffffff	rgb(255, 255, 255)
Blanc coquillage	seashell	#fff5ee	rgb(255, 245, 238)
Blanc dentelle ancienne	oldlace	#fdf5e6	rgb(253, 245, 230)
Blanc fantôme	ghostwhite	#f8f8ff	rgb(248, 248, 255)
Blanc floral	floralwhite	#ffffaf0	rgb(255, 250, 240)
Blanc ivoire	ivory	#fffff0	rgb(255, 255, 240)
Blanc fumée	whitesmoke	#f5f5f5	rgb(245, 245, 245)
Blanc lavande	lavenderblush	#fff0f5	rgb(255, 240, 245)
Blanc lin	linen	#faf0e6	rgb(250, 240, 230)
Blanc menthe	mintcream	#f5ffffa	rgb(245, 255, 250)
Blanc neige	snow	#ffffafa	rgb(255, 250, 250)

TABLEAU A-4 Couleurs nommées de ton BLEU

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Bleu	blue	#0000ff	rgb(000, 000, 255)
Bleu acier	steelblue	#4582b4	rgb(070, 130, 180)
Bleu acier clair	lightsteelblue	#b0c4de	rgb(176, 196, 222)
Bleu Alice	aliceblue	#f0f8ff	rgb(240, 248, 255)
Bleu ardoise	slateblue	#6a5acd	rgb(106, 090, 205)
Bleu ardoise foncé	darkslateblue	#483d88	rgb(072, 061, 139)
Bleu ardoise moyen	mediumslateblue	#7b68ee	rgb(123, 104, 238)
Bleu azur	azure	#f0ffff	rgb(240, 255, 255)

Bleu bleuet	cornflowerblue	#6495ed	rgb(100, 149, 237)
Bleu cadet	cadetblue	#5f9ea0	rgb(095, 158, 160)
Bleu ciel	skyblue	#87cdeb	rgb(135, 205, 235)
Bleu ciel clair	lightskyblue	#87cefa	rgb(135, 206, 250)
Bleu ciel profond	deepskyblue	#00bfff	rgb(000, 191, 255)
Bleu clair	lightblue	#add8e6	rgb(173, 216, 230)
Bleu foncé	darkblue	#00008b	rgb(000, 000, 139)
Bleu indigo	indigo	#4b0082	rgb(075, 000, 130)
Bleu lavande	lavender	#e6e6fa	rgb(230, 230, 250)
Bleu marin	navy	#000080	rgb(000, 000, 128)
Bleu de minuit	midnightblue	#191970	rgb(025, 025, 112)
Bleu moyen	mediumblue	#0000cd	rgb(000, 000, 205)
Bleu poudre	powderblue	#b0e0e6	rgb(176, 224, 230)
Bleu royal	royalblue	#4169e1	rgb(065, 105, 225)
Bleu toile	dodgerblue	#1e90ff	rgb(030, 144, 255)
Bleu violet	blueviolet	#262be2	rgb(250, 235, 215)

TABLEAU A–5 Couleurs nommées de ton BRUN

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Brun	brown	#a5292a	rgb(000, 255, 255)
Brun bois rustique	burlwood	#deb887	rgb(222, 184, 135)
Brun chocolat	chocolate	#d2691e	rgb(210, 105, 030)
Brun cuir	saddlebrown	#8b4513	rgb(139, 069, 019)
Brun kaki	khaki	#f0e68c	rgb(240, 230, 140)
Brun kaki foncé	darkkhaki	#bdb76b	rgb(189, 183, 107)
Brun marron	maroon	#800000	rgb(128, 000, 000)
Brun Pérou	peru	#cd8540	rgb(205, 133, 064)

Brun rosé	rosybrown	#bc8f8f	rgb(188, 143, 143)
Brun roux	tan	#d2b48c	rgb(210, 180, 140)
Brun sableux	sandybrown	#f4a460	rgb(244, 164, 096)
Brun terre de Sienne	sienna	#a0522d	rgb(160, 082, 045)

TABLEAU A–6 Couleurs nommées de tons CYAN – TURQUOISE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Cyan	cyan	#00ffff	rgb(000, 255, 255)
Cyan clair	lightcyan	#e0ffff	rgb(224, 255, 255)
Cyan foncé	darkcyan	#008b8b	rgb(000, 139, 139)
Turquoise	turquoise	#40e0d0	rgb(064, 224, 208)
Turquoise foncé	darkturquoise	#00ced1	rgb(000, 206, 209)
Turquoise moyen	mediumturquoise	#48d1cc	rgb(072, 209, 204)
Turquoise pâle	paleturquoise	#afEEEE	rgb(175, 238, 238)

TABLEAU A–7 Couleurs nommées de ton GRIS

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Gris	gray	#808080	rgb(128, 128, 128)
Gris ardoise	slategray	#708090	rgb(112, 128, 144)
Gris ardoise clair	lightslategray	#778899	rgb(119, 136, 153)
Gris ardoise foncé	darkslategray	#2f4f4f	rgb(047, 079, 079)
Gris argent	silver	#c0c0c0	rgb(192, 192, 192)
Gris clair	lightgrey	#d3d3d3	rgb(211, 211, 211)
Gris Gainsboro	gainsboro	#dcdcdc	rgb(220, 220, 220)
Gris mat	dimgray	#696969	rgb(105, 105, 105)

TABLEAU A–8 Couleur nommée NOIR et codes des NUANCES DE GRIS

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Noir	black	#000000	rgb(000, 000, 000)
(Gris très foncé)		#333333	rgb(051, 051, 051)
(Gris foncé)		#666666	rgb(102, 102, 102)
(Gris moyen)		#999999	rgb(153, 153, 153)
(Gris clair)		#cccccc	rgb(204, 204, 204)

TABLEAU A–9 Couleurs nommées de ton JAUNE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Jaune	yellow	#ffff00	rgb(255, 255, 000)
Jaune blanc Navajo	navajowhite	#ffflead	rgb(255, 222, 173)
Jaune blé	wheat	#f5deb3	rgb(245, 222, 179)
Jaune clair	lightyellow	#f4ffe0	rgb(244, 255, 224)
Jaune doré	goldenrod	#daa520	rgb(218, 165, 032)
Jaune doré clair yellow	lightgoldenrod yellow	#fafad2	rgb(250, 250, 210)
Jaune doré foncé	darkgoldenrod	#b8840b	rgb(184, 132, 011)
Jaune doré pâle	palegoldenrod	#eee8aa	rgb(238, 232, 170)
Jaune or	gold	#ffff00	rgb(255, 255, 000)

TABLEAU A–10 Couleurs nommées de tons ORANGE – CORAIL – SAUMON

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Orange	orange	#ffa500	rgb(255, 165, 000)
Orange foncé	darkorange	#ff8c00	rgb(255, 140, 000)
Orangé	orangered	#ff4500	rgb(255, 069, 000)

Corail	coral	#ff7f50	rgb(255, 127, 80)
Corail clair	lightcoral	#f08080	rgb(240, 128, 128)
Saumon	salmon	#fa8072	rgb(250, 120, 114)
Saumon clair	lightsalmon	#ffa07a	rgb(255, 160, 122)
Saumon foncé	darksalmon	#e9967a	rgb(233, 150, 122)

TABLEAU A–11 Couleurs nommées de ton ROUGE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Rouge	red	#ff0000	rgb(255, 000, 000)
Rouge brique	firebrick	#b22222	rgb(178, 034, 034)
Rouge cramoisi	crimson	#dc143c	rgb(220, 020, 060)
Rouge foncé	darkred	#8b0000	rgb(139, 000, 000)
Rouge indien	indianred	#cd5c5c	rgb(205, 092, 092)
Rouge tomate	tomato	#ff6347	rgb(255, 099, 071)

TABLEAU A–12 Couleurs nommées de ton ROSE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Rose	pink	#ffc0cb	rgb(255, 192, 203)
Rose brumeux	mistyrose	#ffe4ff	rgb(255, 228, 255)
Rose clair	lightpink	#ffb6c1	rgb(255, 182, 193)
Rose passion	hotpink	#ff69b4	rgb(255, 105, 180)
Rose profond	deeppink	#ff1493	rgb(255, 020, 147)

TABLEAU A–13 Couleurs nommées de tons VIOLET – POURPRE – MAGENTA

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Violet	violet	#ee82ee	rgb(238, 130, 238)

Violet bourbon	cornsilk	#ff30dc	rgb(255, 048, 220)
Violet chardon	thistle	#d8bfd8	rgb(216, 191, 216)
Violet foncé	darkviolet	#9400d3	rgb(148, 000, 211)
Violet fuchsia	fuchsia	#ff00ff	rgb(000, 206, 209)
Violet moyen	mediumvioletred	#c71585	rgb(199, 021, 133)
Violet orchidée	orchid	#da70d6	rgb(218, 112, 214)
Violet orchidée foncé	darkorchid	#9932cc	rgb(153, 050, 204)
Violet orchidée moyen	mediumorchid	#ba55d3	rgb(186, 085, 211)
Violet pâle	palevioletred	#db7093	rgb(219, 112, 147)
Violet prune	plum	#dda0dd	rgb(221, 160, 221)
Pourpre	purple	#800080	rgb(128, 000, 128)
Pourpre moyen	mediumpurple	#9370db	rgb(147, 112, 219)
Magenta	magenta	#ff00ff	rgb(255, 000, 255)
Magenta foncé	darkmagenta	#8b008b	rgb(139, 000, 139)

TABLEAU A–14 Couleurs nommées de ton VERT

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Vert	green	#008000	rgb(000, 128, 000)
Vert Chartreuse	chartreuse	#ffff00	rgb(127, 255, 000)
Vert clair	lightgreen	#90ee90	rgb(144, 238, 144)
Vert eau marine	aquamarine	#7ffffd4	rgb(127, 255, 212)
Vert eau marine moyen	mediumaquamarine	#66cdAA	rgb(102, 205, 170)
Vert forestier	forestgreen	#228b22	rgb(034, 139, 034)
Vert foncé	darkgreen	#006400	rgb(000, 100, 000)
Vert jaune	greenyellow	#adff2f	rgb(173, 255, 047)

Vert jauni	yellowgreen	#9acd32	rgb(154, 205, 050)
Vert marin	seagreen	#2e8b57	rgb(046, 139, 087)
Vert marin clair	lightseagreen	#20b2aa	rgb(032, 178, 170)
Vert marin foncé	darkseagreen	#8fbcbf	rgb(143, 188, 143)
Vert marin moyen	mediumseagreen	#3cb371	rgb(060, 179, 113)
Vert olive	olive	#808000	rgb(128, 128, 000)
Vert olive grise	olivedrab	#6b8e23	rgb(107, 142, 035)
Vert olive foncé	darkolivegreen	#556b2f	rgb(085, 107, 047)
Vert pâle	palegreen	#98fb98	rgb(152, 251, 152)
Vert pelouse	lawngreen	#7cfcc0	rgb(124, 252, 000)
Vert printanier	springgreen	#00ff7f	rgb(000, 255, 127)
Vert printanier moyen	mediumspring green	#00fa9a	rgb(000, 250, 154)
Vert sarcelle	teal	#008080	rgb(000, 128, 128)
Vert tilleul clair	lime	#00ffff	rgb(000, 255, 000)
Vert tilleul foncé	limegreen	#32cd32	rgb(050, 205, 050)

Bien sûr, il n'est absolument pas nécessaire de se limiter aux couleurs qui possèdent un nom ; ce serait bien trop restrictif. Votre éditeur HTML ou votre logiciel de retouche d'images vous proposera un nuancier associé aux codes hexadécimaux, et vous pourrez également consulter la liste de couleurs très complète qui se trouve à l'adresse https://fr.wikipedia.org/wiki/Liste_de_noms_de_couleur.



FIGURE A-4 Johannes Itten a eu l'idée d'associer le cercle chromatique de Newton au triangle des couleurs primaires (source : Wikipédia).

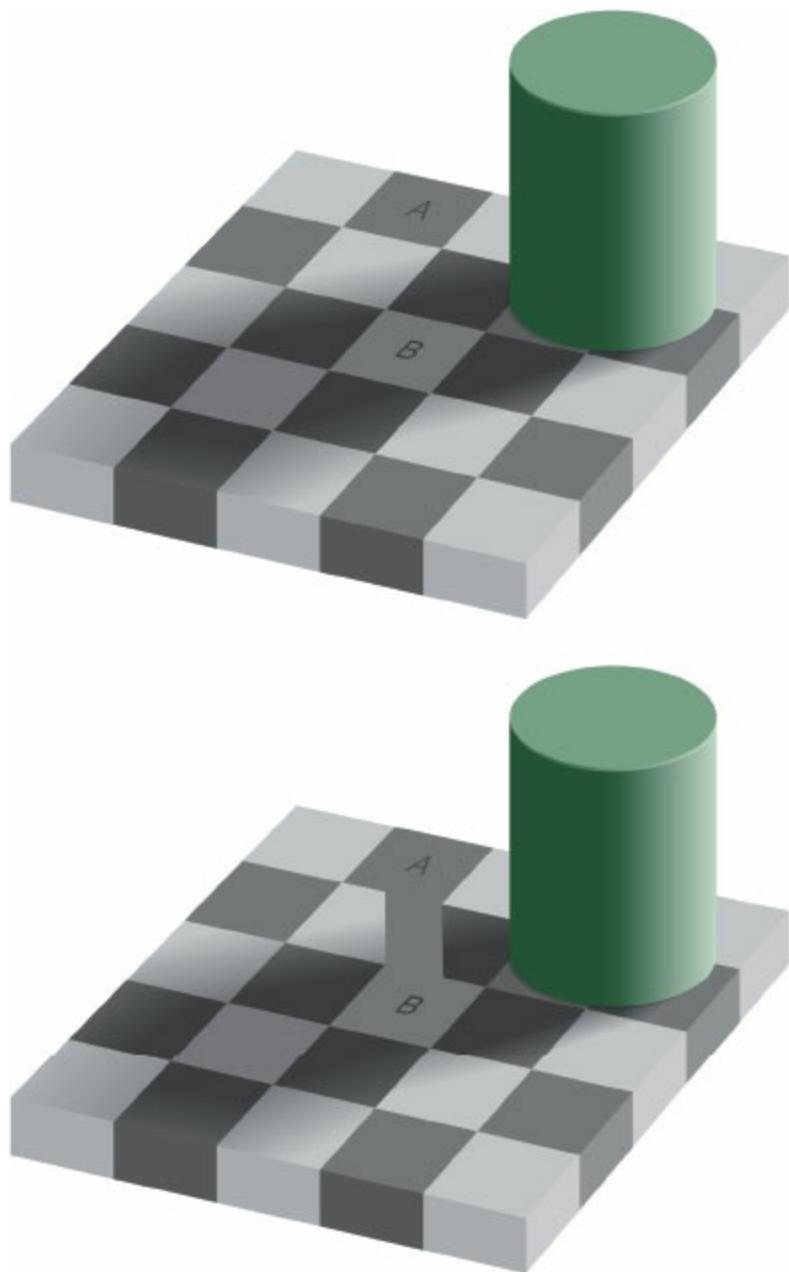


FIGURE A-5 L'échiquier d'Adelson est un effet d'optique : la teinte grise du carré A est la même que celle du carré B (source : Wikipédia).

annexe
B

Comportement des principaux navigateurs



Firefox



chrome



Safari



Opera

Il est important de tester un site sur plusieurs navigateurs, car les balises HTML et propriétés CSS peuvent donner lieu à quelques différences d'interprétation et d'affichage, suivant le logiciel utilisé et sa version. En particulier, certaines des nouvelles propriétés CSS 3 ne sont encore que partiellement intégrées dans les navigateurs actuels.

SOMMAIRE

- ▶ **Test des pages sur plusieurs navigateurs**
- ▶ **Interprétation du HTML et des CSS**
- ▶ **Normes et navigateurs**

Même si nous parlons correctement une langue étrangère, il reste des mots qui nous échappent. Cette liste de mots incompris sera plus ou moins longue, en fonction de l'étendue de nos connaissances dans cette langue.

Eh bien, pour les navigateurs web, c'est pareil ! Globalement, il comprennent ce qu'il leur est demandé d'afficher, mais suivant la version utilisée, ils peuvent avoir des lacunes sur certains détails...



FIGURE B-1 Parfois, des mots nous échappent ; de même, il y a des normes qui sont mal comprises par certains navigateurs web (illustration issue du site Wikipédia).

Test des pages sur plusieurs navigateurs

Lors de l'écriture d'une page, il est important de contrôler au fur et à mesure le résultat affiché. Certains éditeurs HTML proposent une fenêtre d'aperçu rapide qui simule le navigateur, mais il est utile de vérifier le rendu de la page dans les conditions réelles d'affichage.

Pour cela, nous ouvrirons deux fenêtres en parallèle, celle de l'éditeur utilisé pour écrire le code et une autre qui affichera la page dans un navigateur. Le test continu de la page s'effectue alors de la façon suivante :

- après chaque écriture ou modification d'une partie du code, il faut penser à enregistrer la page (raccourci ***Ctrl + S***, comme « Save ») ;
- ensuite, il suffit de basculer vers la fenêtre du navigateur (par exemple avec la combinaison des touches ***Alt*** et ***Tabulation***), puis d'actualiser l'affichage (c'est souvent la touche ***F5*** qui est affectée à cette fonction).

La vérification avec différentes versions de plusieurs navigateurs ne s'effectuera qu'une fois la page terminée, sans quoi le développement deviendrait beaucoup trop long.

UTILE Des extensions pour faciliter la conception

La plupart des navigateurs proposent des outils de développement, accessibles à partir du menu ***Outils*** ou d'un raccourci clavier. Il est également possible de leur adjoindre des extensions, qui sont des programmes additionnels apportant des fonctions supplémentaires à l'utilisateur ou au concepteur.

Par exemple, Firefox permet l'ajout de modules destinés à faciliter la création et le débogage des pages, tels que *Web Developer*, *User Agent Switcher* et *MeasureIt* (parmi les plus connus).

La liste de ces modules et leur description est disponible soit à partir du menu ***Outils*** du navigateur, soit directement à l'adresse <https://addons.mozilla.org/fr/firefox/>. Ces trois outils sont également disponibles pour le navigateur Chrome, sur la page <https://chrome.google.com/webstore/category/extensions>.

Il faudra néanmoins faire attention à ne pas trop installer d'extensions, car si elles sont nombreuses ou trop gourmandes en ressources, elles peuvent ralentir la navigation.

Une fois la page écrite et testée dans ces conditions, il faudra en vérifier l'affichage sur différents navigateurs pour ordinateur de bureau et pour mobile, les deux versions d'un logiciel donné n'ayant pas toujours le même comportement.

CHOIX Quels navigateurs et quelles versions faut-il viser ?

Évidemment, il ne sera pas possible de créer des pages qui s'afficheront parfaitement sur toutes les versions de tous les navigateurs en remontant jusqu'à Mathusalem ! D'ailleurs, à cette époque, les navigateurs étaient plutôt sur la mer que dans un ordinateur...

Il faudra donc sélectionner et ne tester que les principaux de ces « butineurs », appelés *browsers* en anglais. Aussi est-il intéressant de consulter les statistiques qui donnent le taux d'utilisation de chaque version des différents navigateurs. Elles sont accessibles à partir de la page suivante :

► <https://www.w3schools.com/browsers/>

Attention à l'interprétation de ces statistiques :

- d'une part, ces données sont des moyennes mondiales (les chiffres peuvent donc varier suivant les pays) ;
- d'autre part, elles peuvent être à pondérer en fonction du type de public visé. Par exemple, un public d'informaticiens utilisera plus souvent un navigateur récent interprétant les derniers standards du W3C.

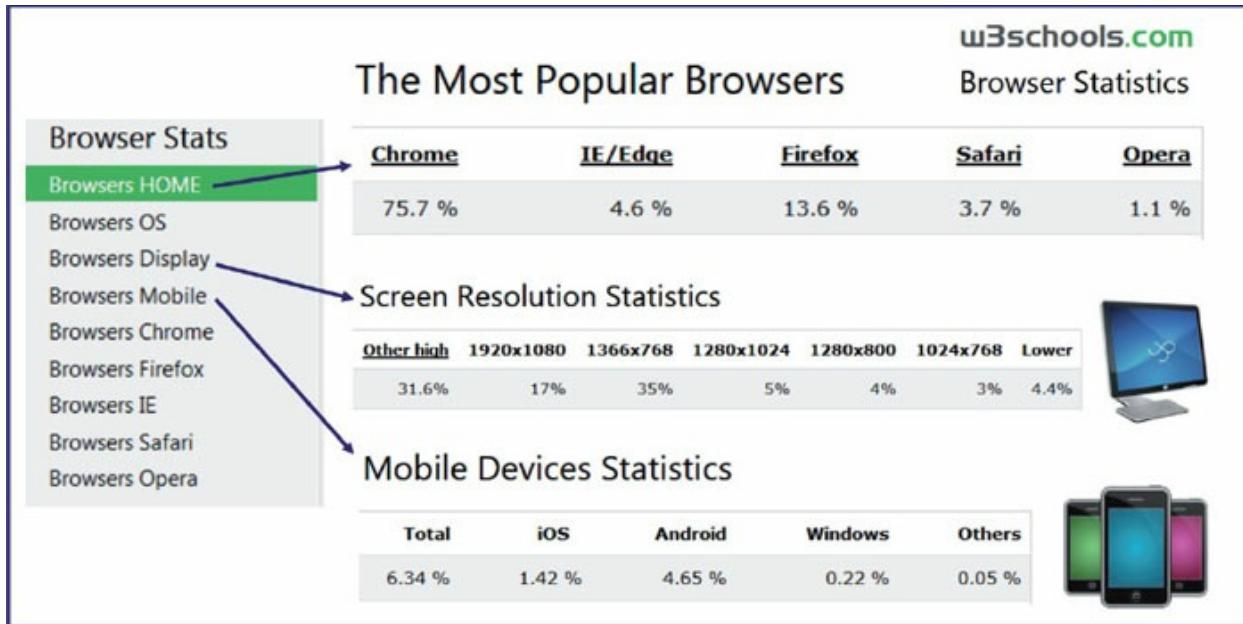


FIGURE B-2 Extraits des pages proposées à l'adresse <https://www.w3schools.com/browsers/> : accès aux statistiques sur l'utilisation des différents navigateurs Internet, des systèmes mobiles et des dimensions d'écrans

À NOTER Installer différentes versions d'un même logiciel

Pour afficher notre page dans plusieurs versions d'un même logiciel, il faudra installer celles-ci sur notre ordinateur. La cohabitation de ces versions nécessite une installation spécifique. Elle est même impossible pour Internet Explorer qui est intégré au système d'exploitation Windows, toute nouvelle version installée remplaçant la précédente.

Voici donc des astuces qui nous permettront d'utiliser tour à tour différentes versions d'un même navigateur :

- Firefox, Opera et Chrome existent en versions portables, qui ne nécessitent pas d'installation et sont utilisables à partir d'une clef USB ou d'un disque dur externe. Toutes les versions portables de ces logiciels, ainsi que bien d'autres, sont disponibles ici, sur l'intéressant site suivant : <https://portableapps.com>. Cependant, il ne sera peut-être pas possible de lancer simultanément deux versions d'un même navigateur.
- Pour Internet Explorer, des versions dites « standalone » non officielles pourront cohabiter avec celle intégrée au système

d'exploitation, par exemple les *IE Tester* (www.my-debugbar.com/wiki/IETester). Cependant, la solution la plus simple consiste à utiliser, avec Edge ou Internet Explorer, la touche de raccourci **F12** (équivalente au menu **Outils > Outils de développement**) : dans la fenêtre qui s'ouvre, l'onglet **Emulation** propose un menu déroulant permettant de changer instantanément le « mode de document », c'est-à-dire le moteur de rendu (la version d'Internet Explorer) pris en compte dans la fenêtre principale.

Interprétation du HTML et des propriétés CSS

Les navigateurs actuels interprètent correctement la plupart des balises HTML et les propriétés CSS 2.1, ainsi qu'une bonne partie de la norme CSS 3, sauf certaines fonctions très rarement utilisées, comme celles liées au son ou à la pagination.

Les tests en ligne Acid2 et Acid3 permettent de vérifier la bonne reconnaissance des standards de base par les navigateurs, et notamment quelques propriétés CSS 3 pour Acid3. Un test Acid4 est en préparation, avec pour objectif de mieux tester les normes CSS 3.

Il devient difficile de s'assurer du bon affichage d'un site sur l'ensemble des navigateurs. En effet, à leur variété s'ajoutent des mises à jours fréquentes, d'où la cohabitation de plusieurs versions d'un même logiciel. Par ailleurs, plusieurs de ces navigateurs sont déclinés en version mobile, laquelle peut présenter des différences avec la version de base. Heureusement, des solutions en ligne s'offrent à nous pour nous faciliter la tâche ; nous allons en examiner quelques-unes.



FIGURE B–3 Crées pour inciter les développeurs à mieux respecter les normes CSS 2 et 3, le test Acid2 (<http://acid2.acidtests.org>), puis le test Acid3 (<http://acid3.acidtests.org>) : si le navigateur passe ce dernier test, il affiche une animation qui se termine par l'image de droite.

Normes et navigateurs

Comme nous l'avons vu, certains points de la norme CSS 3 ne sont pas encore finalisés. Les nouveautés apportées sont progressivement prises en compte par les navigateurs, au fur et à mesure de leurs nouvelles versions. Il en est de même pour quelques balises ou attributs récemment introduits par le standard HTML 5.

Rappelons que lorsque des propriétés CSS sont en cours de développement, elles sont souvent implémentées de façon provisoire dans les navigateurs, avec un préfixe à écrire devant leur nom : `-moz-` pour Mozilla Firefox, `-webkit-` pour les navigateurs basés sur le moteur Webkit (Safari) ou son dérivé optimisé Blink (Chrome, Chromium et Opera), `-ms-` comme Microsoft pour Edge et Internet Explorer.

Comportement des navigateurs courants

Pour connaître le niveau d'interprétation de chacune des balises ou propriétés, il existe le site très utile « Can I use? », qui rend bien service aux développeurs. Il est accessible à l'adresse <http://www.caniuse.com>.

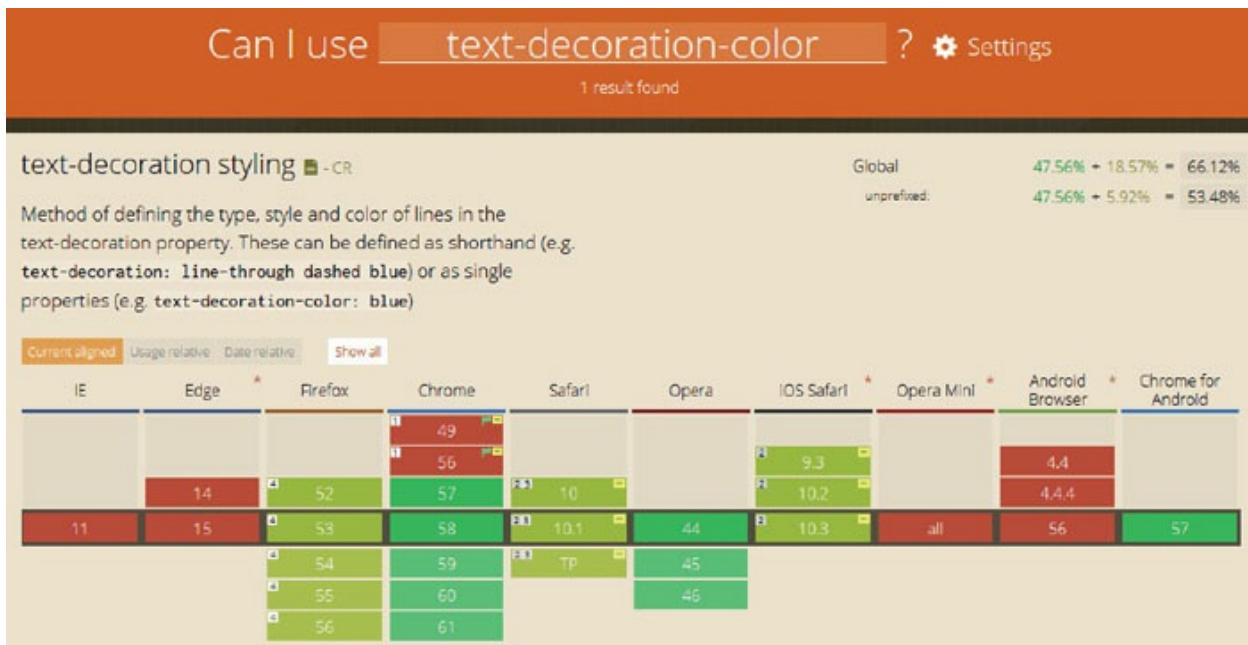


FIGURE B-4 Le site www.caniuse.com indique si la norme indiquée (ici la propriété CSS 3 `text-decoration-color`) est reconnue par les versions courantes des navigateurs (ligne centrale encadrée), ainsi que par les versions précédentes et par les versions à venir.

Comme le montre la figure B-4, il suffit d'indiquer un élément du standard W3C (balise HTML, propriété CSS, pseudo-classe, pseudo-élément, etc.) dans la zone de recherche pour savoir s'il est reconnu par les principaux navigateurs, grâce à un code de couleurs : vert pour une interprétation correcte, vert clair pour une prise en compte partielle, rouge si le standard est ignoré. La couleur grise peut apparaître lorsque l'information n'est pas disponible.

La présence d'un petit carré jaune, en haut à droite du rectangle associé à une version donnée, indique que le préfixe spécifique à ce navigateur devra être utilisé. S'il y a un petit drapeau vert, il correspond à une annotation. Celle-ci et d'autres précisions seront visibles dans une bulle qui s'affichera au passage de la souris sur le rectangle de couleur.

Pour avoir une idée de la popularité de chacune de ces versions de

navigateurs, il suffit de cliquer sur le bouton ***Usage relative*** (le choix par défaut étant ***Current aligned***, soit un alignement sur la version courante). La figure B-5 correspond à la même recherche que la figure précédente, mais avec une surface d'affichage proportionnelle au pourcentage d'utilisation de chaque version de navigateur.

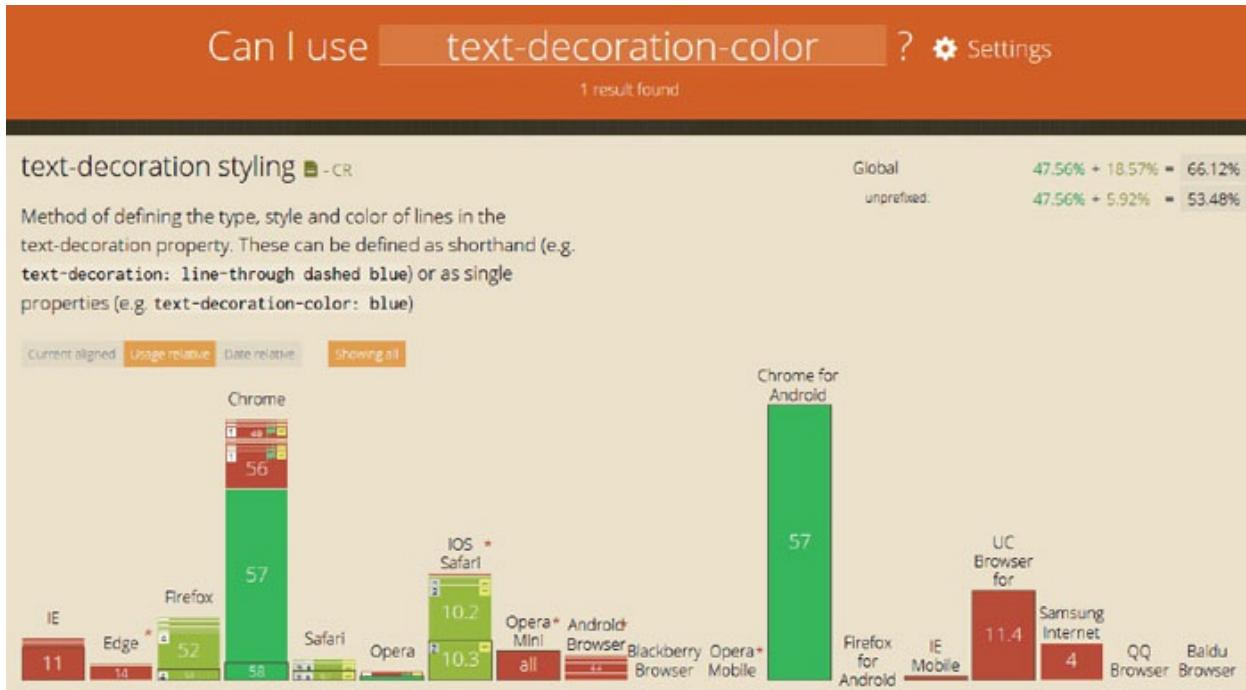


FIGURE B-5 Affichage proportionnel au taux d'utilisation sur « *Can I use?* », toujours avec les couleurs associées à la compréhension de la norme indiquée en zone de recherche

Une autre option, ***Date relative***, affiche sur une échelle des temps pour les versions des différents logiciels. Elle permet notamment de constater que le test du navigateur Opera Mini n'est pas à jour sur ce site, car ses dernières versions n'ont pas été prises en compte. Une note liée à ce navigateur, lisible par un clic sur l'étoile qui suit son nom, indique que les dernières versions se basent sur le moteur de rendu de iOS Safari. D'autres ressemblances pourront être trouvées, le comportement du navigateur Opera qui est lié à celui de Chrome par exemple, puisque ces deux logiciels utilisent le même moteur de rendu Blink.

UTILÉ Paramétrages du site « *Can I use?* »

D'autres réglages sont accessibles sur ce site, par un clic sur le lien ***Settings***, présent en haut à droite de la fenêtre. Sur la gauche s'affiche alors une barre de paramétrages, dont un des plus intéressants est

d'afficher les résultats pour un pays donné.

- Sous le titre **From geography**, saisir un ou plusieurs nom(s) de pays, en cliquant chaque fois sur le bouton **Import** situé à droite.
- La liste des pays ainsi ajoutés est alors disponible dans la liste déroulante placée au-dessus, sous le titre **Source**.
- Il suffit de choisir un pays dans cette liste pour voir s'adapter automatiquement les données fournies, comme le montre la figure B-6.

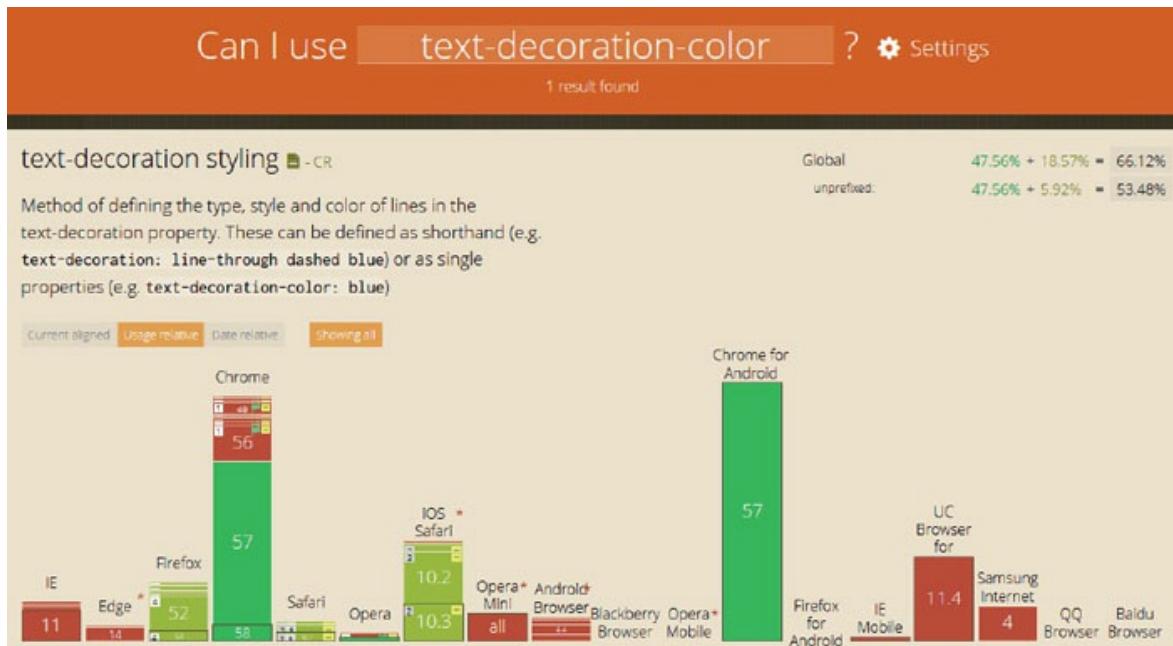


FIGURE B–6 Paramétrages du site « Can I use? », ici pour afficher les données spécifiques à la France

Tester un navigateur

Une autre approche de l'interprétation des standards consiste à les vérifier soi-même sur un navigateur. Un test général des propriétés CSS 3 est possible à partir du site <http://css3test.com/>. Celui-ci fournit un pourcentage global de reconnaissance de ce standard, qu'il détaille ensuite par catégories et par propriétés. Il faut toutefois se méfier de tels barèmes automatiques, qui peuvent donner une indication intéressante, mais restent à prendre avec précaution.

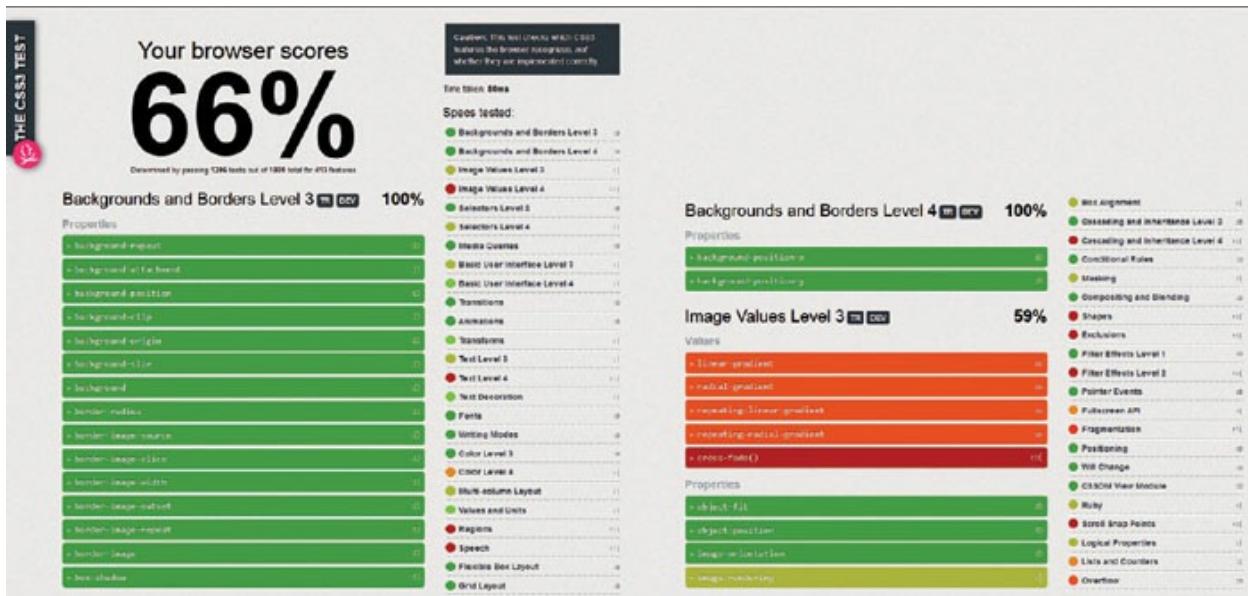


FIGURE B-7 Test du standard CSS 3 par le site <http://css3test.com>

Pour savoir comment notre navigateur prend en compte un élément tel qu'une balise, une propriété ou un sélecteur, la meilleure solution consiste à effectuer nous-même le test en écrivant un exemple de code. Cela prend évidemment beaucoup de temps, sans compter les risques d'erreurs dans l'écriture de notre code. Rassurez-vous, là encore il existe une solution magique !

Le fameux site www.w3schools.com, qui est une mine d'or pour l'apprentissage du HTML, des CSS, mais aussi de nombreuses autres techniques du Web, accompagne toujours d'exemples ses explications et tableaux de synthèse. La plupart du temps, il nous propose de les tester dans notre navigateur. Une nouvelle fenêtre s'ouvre et comprend alors deux parties : le code proposé et le résultat.

Il est possible de personnaliser ce code, car il est affiché dans un éditeur en

ligne. Après chaque modification, il suffit de cliquer sur le bouton *Run* pour observer le résultat dans la partie droite. Ce fonctionnement est résumé par la figure B-8.

Les normes évoluent sans cesse et les problèmes de compatibilité avec les nouveaux standards seront toujours un point à ne pas oublier. Avec un peu d'expérience, vous connaîtrez les balises et propriétés qui sont devenues courantes. Pour les autres et en cas de doute, les outils que nous venons d'explorer vous fourniront les informations nécessaires à la création de pages lisibles par le plus grand nombre.

CSS3 box-shadow Property

Example

Add a box-shadow to a <div> element:

```
div {  
    box-shadow: 10px 10px 5px #888888;  
}
```

[Try it Yourself »](#)

The screenshot shows a web-based code editor interface. At the top, there are icons for home, menu, and refresh, followed by a green 'Run' button. Below the toolbar is a code editor window containing an HTML and CSS snippet. The CSS part includes a box-shadow declaration. To the right of the editor is a preview area showing a purple rectangular box with a black shadow effect. A blue arrow points from the 'Try it Yourself' button in the example box up towards the preview area.

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div { font: 20px Arial, sans-serif;  
    color: dimgray;  
    width: 300px;  
    height: 100px;  
    padding: 20px;  
    border-radius: 20px;  
    background-color: lavender;  
    box-shadow: 10px 10px 5px #888888;  
}  
</style>  
</head>  
<body>  
|  
<div>Bloc ombré</div>  
  
</body>  
</html>
```

FIGURE B–8 Le site W3Schools nous invite à tester en ligne la fonction étudiée (ici, la propriété CSS box-shadow). Le code associé à cet exemple peut être modifié à volonté, ce qui est un excellent moyen d'apprentissage.

QUESTION Propriétés avec ou sans préfixes ?

En ce qui concerne les nouvelles propriétés CSS 3, l'usage est de les écrire avec les préfixes spécifiques aux navigateurs (-moz-, -webkit- et -ms-), en prenant soin d'écrire à la fin la forme standard non préfixée.

Si cela permet d'assurer l'affichage sur les anciennes versions des logiciels, le temps qui passe et la généralisation des mises à jour automatiques font que cette précaution devient de moins en moins nécessaire. Un petit tour sur le site « Can I use? » nous montrera depuis combien de versions les préfixes ne sont plus nécessaires. Nous en tirerons facilement des conclusions sur leur utilité dans chaque cas.

Puis une vérification à partir des exemples en ligne du site W3Schools nous confirmera le bon fonctionnement ou non des versions avec ou sans préfixes, que nous pourront tester sur les navigateurs les plus courants.

De temps en temps, vous vous poserez des questions sur la syntaxe à utiliser pour une propriété CSS donnée. Il suffira de la rechercher dans les chapitres précédents, ou bien de jeter un œil sur les tableaux de synthèse que vous trouverez dans le chapitre qui suit.

image initiale sans transformation



transform: translate(150px, 50px);



transform: skew(45deg,-15deg);



transform-origin: bottom left;

avec transform: rotate(45deg);



transform: scale(0.8, 1.5);



transform: rotate(45deg);



transform: matrix(0.5,3,1.5,1.2,3,5);



FIGURE B–9 Quelques transformations géométriques à deux dimensions en CSS 3, affichées par le navigateur Chrome

annexe
C

Résumé des propriétés CSS



Voici quelques pages qui pourront servir de référence, puisqu'elles résument les caractéristiques principales de chaque propriété.

SOMMAIRE

- ▶ **Propriétés communes aux normes CSS 2 et CSS 3**
- ▶ **Principales propriétés spécifiques aux CSS 3**
- ▶ **Styles CSS 2 et 3 pour les médias paginés et sonores**
- ▶ **Propriétés classées par catégories**

Une fois habitué à l'utilisation des feuilles de styles, vous aurez parfois besoin d'un petit rappel sur une propriété. Ces index seront alors bien pratiques pour vous rafraîchir la mémoire.

Propriétés communes aux normes CSS 2 et CSS 3

Ces tableaux ont été réalisés d'après une page du site [www.yoyodesign.org](http://www.yoyodesign.org/doc/w3c/css2/propidx.html), qui propose la traduction en français des normes du W3C, le World Wide Web Consortium : <http://www.yoyodesign.org/doc/w3c/css2/propidx.html>. La page web originale en anglais se trouve à l'adresse <http://www.w3.org/TR/CSS2/propidx.html>.

The screenshot shows the w3schools.com website with the URL <https://www.w3schools.com/cssref/> in the address bar. The page title is "CSS Reference". On the left, there's a sidebar with "CSS Reference" selected. The main content area starts with a note: "W3Schools' CSS reference is tested regularly with all major browsers." Below this is a section titled "CSS Properties" with a sub-section "CSS Property Groups". A list of property groups is provided, each preceded by a small icon. To the right of the list is a "COLOR PICKER" icon. At the bottom of the page, there's a table comparing properties across different CSS versions (1, 2, or 3).

Property	Description	css
<code>color</code>	Sets the color of text	1
<code>opacity</code>	Sets the opacity level for an element	3

FIGURE 3–1 Référence de la norme CSS sur la page <https://www.w3schools.com/cssref/>

D'autres ressources sont disponibles en ligne et rappellent de façon synthétique la syntaxe des propriétés et autres éléments de la norme CSS,

toujours accompagnée d'exemples :

- « W3Schools » (www.w3schools.com), qui est un excellent site pour l'apprentissage de toutes les techniques du Web, comprend bien sûr une importante section dédiée aux CSS (www.w3schools.com/cssref/);
- la référence CSS très complète de la communauté Codrops se trouve à l'adresse https://tympanus.net/codrops/css_reference/.

Propriétés d'affichage

Ces propriétés affectées aux écrans d'ordinateur, de tablettes ou de smartphones, de type @media screen par conséquent, font partie de la norme CSS 2 et constituent toujours les bases de la version CSS 3.

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
background	fond	[background-color background-image background-repeat background-attachment background-position] inherit	voir chaque propriété		admis pour background-position
background-attachment	défilement de l'image de fond	scroll fixed inherit	scroll		
background-color	couleur de fond	<couleur> transparent inherit	transparent		
background-image	image de fond	<uri> none inherit	none		
background-position	position de l'image de fond	[[<pourcentage> <longueur>] {1,2} [[top center bottom] [left center right]]] inherit	0% 0%	éléments de type bloc et éléments remplacés	% de la taille de la boîte elle-même
background-repeat	répétition de l'image de fond	repeat repeat-x repeat-y no-repeat inherit	repeat		

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
border	raccourci pour les bordures	[border-width border-style <couleur>] inherit	voir chaque propriété		
border-collapse	fusion des bordures	collapse separate inherit (H)	collapse	éléments avec 'display: table' ou 'display: inline-table'	
border-color	couleur des bordures	<couleur>{1,4} transparent inherit	valeur de la propriété 'color'		
border-spacing	espace entre les bordures	<longueur> <longueur>? inherit (H)	0	éléments avec 'display: table' ou 'display: inline-table'	
border-style	style de bordure	<bordure-style>{1,4} inherit	none		

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
border-top border-right border-bottom border-left	bordures sur les côtés	[border-top-width border-style <couleur>] inherit	voir chaque propriété		
border-top-color border-right-color border-bottom-color border-left-color	couleur des bordures sur les côtés	<couleur> inherit	valeur de la propriété 'color'		
border-top-style border-right-style border-bottom-style border-left-style	style de bordure sur les côtés	<bordure-style> inherit	none		
border-top-width border-right-width border-bottom-width border-left-width	épaisseur des bordures sur les côtés	<bordure-épaisseur> inherit	medium		
border-width	épaisseur des bordures	<bordure-épaisseur>{1,4} inherit	medium		

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcen- tage (si utilisé)
bottom	position par rapport au bas	<longueur> <pourcentage> auto inherit	auto	éléments positionnés	% de la hauteur du conteneur
caption-side	emplacement du titre (tableau)	top bottom left right inherit (H)	top	éléments avec 'display: table-caption'	
clear	pas de boîtes flottantes à côté	none left right both inherit	none	éléments de type bloc	
color	couleur de police	<couleur> inherit (H)	selon le navigateur		
content	contenu à ajouter	[<chaîne> <uri> <compteur> attr(X) open-quote close-quote no-open-quote no-close-quote]+ inherit	chaîne vide	pseudo-éléments ::before et ::after	
counter-increment	incrémentation de compteur	[<identifiant> <entier>?]+ none inherit	none		

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcen- tage (si utilisé)
counter-reset	remise à zéro du compteur	[<identifiant> <entier>?]+ none inherit	none		
cursor	type de curseur	[[<uri>,]* [auto default context-menu pointer alias help copy text vertical-text no-drop not-allowed all-scroll move cell crosshair grab grabbing col-resize row-resize e-resize ew-resize w-resize n-resize ns-resize s-resize ne-resize nesw-resize sw-resize nw-resize nwse-resize se-resize progress wait zoom-in zoom-out]] inherit (H)	auto	(médias interactifs)	
direction	sens de lecture	ltr rtl inherit (H)	ltr	tous les éléments	

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
display	mode d'affichage d'un élément	inline block list-item run-in inline-block flex inline-flex table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption none inherit	inline	(tous médias)	
empty-cells	bordure des cellules vides	show hide inherit (H)	show	éléments avec 'display: table-cell'	
float	transformation en bloc flottant	left right none inherit	none	tous les éléments, sauf ceux positionnés ou avec un contenu généré	

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
font	raccourci pour les propriétés font-...	[[font-style font-variant font-weight]? font-size [/ line-height]? font-family] caption icon menu message-box small-caption status-bar inherit (H)	voir chaque propriété		admis pour font-size et line-height
font-family	police(s) de caractères	[[<famille-nom> <famille-générique>],]* [<famille-nom> <famille-générique>] inherit (H)	selon le navigateur		
font-size	taille des caractères	<taille-absolue> <taille-relative> <longueur> <pourcentage> inherit (H)	medium		% de la taille de police du bloc parent
font-style	italique	normal italic oblique inherit (H)	normal		

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
font-variant	petites majuscules	normal small-caps inherit (H)	normal		
font-weight	graisse des caractères	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit (H)	normal		
height	hauteur	<longueur> <pourcentage> auto inherit	auto	éléments de type bloc ou remplacés, sauf colonnes de tableaux	voir explications
left	décalage à partir de la gauche	<longueur> <pourcentage> auto inherit	auto	éléments positionnés	% de la largeur du bloc conteneur
letter-spacing	espacement entre les lettres	normal <longueur> inherit (H)	normal		

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
line-height	hauteur de ligne	normal <nombre> <longueur> <pourcentage> inherit (H)	normal		% de la taille de la police de l'élément lui-même
list-style	raccourci pour les propriétés list-style-type, list-style-position et list-style-image	[list-style-type list-style-position list-style-image] inherit (H)	voir chaque propriété	éléments avec 'display: list-item'	
list-style-image	image à utiliser comme puce	<uri> none inherit (H)	none	éléments avec 'display: list-item'	
list-style-position	position de la puce	inside outside inherit (H)	outside	éléments avec 'display: list-item'	

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
list-style-type	type de puce ou de numérotation	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-alpha lower-latin upper-alpha upper-latin hebrew armenian georgian cjk-ideographic hiragana katakana hiragana-iroha katakana-iroha none inherit et nombreuses autres valeurs (H)	disc	éléments avec 'display: list-item'	
margin	raccourci pour les marges extérieures	<marge-largeur>{1,4} inherit	0		% de la largeur du bloc conteneur
margin-top margin-right margin-bottom margin-left	marges extérieures de chaque côté	<marge-largeur> inherit	0		% de la largeur du bloc conteneur

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
max-height	hauteur maximum	<longueur> <pourcentage> none inherit	none	éléments de type bloc ou remplacés, sauf éléments de tableaux	% de la largeur du bloc conteneur
max-width	largeur maximum	<longueur> <pourcentage> none inherit	none	éléments de type bloc ou remplacés, sauf éléments de tableaux	% de la largeur du bloc conteneur
min-height	hauteur minimum	<longueur> <pourcentage> inherit	0	éléments de type bloc ou remplacés, sauf éléments de tableaux	% de la largeur du bloc conteneur
min-width	largeur minimum	<longueur> <pourcentage> inherit	selon le navigateur	éléments de type bloc ou remplacés, sauf éléments de tableaux	% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
outline	raccourci pour les propriétés outline–...	[outline-color outline-style outline-width] inherit	voir chaque propriété		
outline-color	couleur du contour des boîtes	<couleur> invert inherit	invert		
outline-offset	écart entre les bordures outline et border	<bordure-épaisseur> inherit	0		
outline-style	style du contour des boîtes	<bordure-style> inherit	none		
outline-width	épaisseur du contour des boîtes	<bordure-épaisseur> inherit	medium		
overflow	affichage des débordements	visible hidden scroll auto inherit	visible	éléments de type bloc et éléments remplacés	

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
overflow-x	affichage du débordement horizontal	visible hidden scroll auto inherit	visible	éléments de type bloc et éléments remplacés	
overflow-y	affichage du débordement vertical	visible hidden scroll auto inherit	visible	éléments de type bloc et éléments remplacés	
padding	raccourci pour les propriétés padding–...	<espacement-largeur>{1,4} inherit	0		% de la largeur du bloc conteneur
padding-top padding-right padding-bottom padding-left	marges intérieures de chaque côté	<espacement-largeur> inherit	0		% de la largeur du bloc conteneur
position	type de positionnement	static relative absolute fixed inherit	static	tous les éléments, sauf ceux avec contenu généré	

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
quotes	caractères pour guillemets	[<chaîne> <chaîne>]+ none inherit (H)	selon le navigateur		
right	décalage à partir de la droite	<longueur> <pourcentage> auto inherit	auto	éléments positionnés	% de la largeur du bloc conteneur
table-layout	largeur des colonnes fixe ou variable	auto fixed inherit	auto	éléments avec 'display: table' ou 'display: inline-table'	
text-align	alignement horizontal du texte	left right center justify inherit (H)	selon le navigateur	éléments de type bloc	
text-decoration	souligné-barré-clignotant...	none [underline overline line-through blink] inherit	none		

TABLEAU C–1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
text-indent	retrait de la première ligne	<longueur> <pourcentage> inherit (H)	0	éléments de type bloc	% de la largeur du bloc conteneur
text-transform	majuscules-minuscules	capitalize uppercase lowercase none inherit (H)	none		
top	décalage à partir du haut	<longueur> <pourcentage> auto inherit	auto	éléments positionnés	% de la largeur du bloc conteneur
unicode-bidi	gestion du texte bidirectionnel	normal embed bidi-override inherit	normal		
vertical-align	alignement ou décalage vertical	baseline sub super top text-top middle bottom text-bottom <pourcentage> <longueur> inherit	baseline	éléments en ligne (décalage vertical) ou cellules de tableaux (alignement)	% de la valeur de line-height de l'élément lui-même

TABLEAU C-1 Index des propriétés d'affichage communes aux CSS 2 et CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
visibility	affichage de l'élément	visible hidden collapse inherit	inherit		
white-space	conservation des espaces et des sauts de ligne	normal pre nowrap inherit (H)	normal	éléments de type bloc	
width	largeur de l'élément	<longueur> <pourcentage> auto inherit	auto	tous les éléments, sauf éléments en ligne non remplacés et rangées de tableau	% de la largeur du bloc conteneur
word-spacing	espacement entre les mots	normal <longueur> inherit (H)	normal		
z-index	ordre de superposition	auto <entier> inherit	auto	éléments positionnés	

Principales propriétés spécifiques aux CSS

3

Suivant le même schéma que les tableaux précédents, voici un résumé des principales propriétés apportées par la norme CSS 3. Il faut cependant garder à l'esprit que cette norme n'est pas encore finalisée et que certaines caractéristiques figurant dans ce tableau peuvent évoluer au fil du temps.

Testing	Current	Upcoming	Notes	IO
CSS Image Values and Replaced Content Level 3	CR	CR		IO
CSS Speech	CR	CR		IO
CSS Flexible Box Layout	CR	PR		IO
CSS Text Decoration Level 3	CR	CR		IO
CSS Shapes Level 1	CR	CR		IO
CSS Masking Level 1	CR	CR		IO
CSS Fragmentation Level 3	CR	PR		IO
CSS Cascading Variables	CR	PR		IO
Compositing and Blending Level 1	CR	CR		IO
CSS Syntax Level 3	CR	CR		IO
CSS Grid Layout Level 1	CR	PR		IO
CSS Basic User Interface Level 3	CR	PR		IO
CSS Will Change Level 1	CR	PR		IO
Geometry Interfaces Level 1	CR	CR		IO
CSS Cascading and Inheritance Level 4	CR	PR		IO
CSS Scroll Snap Level 1	CR	PR		IO
Refining	Current	Upcoming	Notes	IO
CSS Animations	WD	WD		IO
Web Animations 1.0	WD	WD		IO
CSS Text Level 3	LC	CR		IO
CSS Transforms	WD	WD		IO
CSS Transitions	WD	CR		IO
CSS Box Alignment Level 3	WD	WD		IO
CSS Display Level 3	WD	WD		IO
Media Queries level 4	WD	WD		IO
Preview of CSS Level 2	FPWD	NOTE		IO
CSS Containment Level 1	WD	WD		IO
CSS Timing Functions Level 1	FPWD	WD		IO

FIGURE C–2 Le W3C tient à jour l'état d'avancement de la norme CSS 3 à l'adresse

<https://www.w3.org/Style/CSS/current-work>.

TABLEAU C–2 Index des propriétés d'affichage CSS 3

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
align-content	répartition des items flex sur l'axe perpendiculaire	stretch flex-start center flex-end space-between space-around initial inherit	stretch	blocs	
align-items	alignement des items flex suivant l'axe perpendiculaire	stretch flex-start center flex-end baseline initial inherit	stretch	blocs	
align-self	alignement d'un item flex sur l'axe perpendiculaire	auto stretch flex-start center flex-end baseline initial inherit	auto	blocs	
animation	animation au chargement de la page	[<animation-name> <animation-duration> <animation-timing-function> <animation-delay> <animation-iteration-count> <animation-direction> <animation-fill-mode>]	voir chaque propriété		

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
animation-delay	délai avant le début de l'animation	<durée> [, <durée>]+	0		
animation-direction	sens de l'animation	normal reverse alternate alternate-reverse [, normal reverse alternate alternate-reverse]+	normal		
animation-duration	durée de l'animation	<durée> [, <durée>]+	0		
animation-fill-mode	aspects initial et final de l'élément	none forwards backwards both [, ...]+	none		
animation-iteration-count	nombre de répétitions de l'animation	infinite <nombre> [, infinite <nombre>]	1		
animation-name	nom de l'animation	none <nom> [, none <nom>]+	none		
animation-play-state	suspension de l'animation	running paused [, running paused]+	running		

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
animation-timing-function	évolution de la vitesse de l'animation	ease linear ease-in ease-out ease-in-out step-start step-end steps(nombre[, start end]) cubic-bezier(x1,y1,x2,y2) [, ...]+	ease		
appearance	apparence de l'élément	normal <apparence> comme icon, window, menu, field inherit	normal		
background-clip	arrière-plan sous la bordure ou non	border-box padding-box content-box [, ...]+		blocs	
background-image	image de fond	none url(...) linear-gradient(...) radial-gradient(...) repeating-linear-gradient(...) repeating-radial-gradient(...) [, ...]+	none	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
background-origin	origine pour la position de l'image d'arrière-plan	padding-box border-box content-box [, ...] +	padding-box	blocs	
background-size	dimension de l'image d'arrière-plan	auto <longueur>{1,2} <pourcentage>{1,2} contain cover [, ...] +	auto	blocs	% des dimensions du bloc
border-bottom-left-radius	arrondi du coin inférieur gauche	<longueur> ou % {1,2}	0	blocs	
border-bottom-right-radius	arrondi du coin inférieur droit	<longueur> ou % {1,2}	0	blocs	
border-image	encadrement par une image	none <url> [<border-image-slice>] / [<border-image-width>] / [<border-image-outset>] / [<border-image-repeat>]	none	blocs	
border-image-outset	débordement autorisé	<longueur> ou % {1,4}	0	blocs	
border-image-repeat	motif répété ou étiré	<stretch repeat round space> {1,2}	repeat	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
border-image-slice	contour d'image à utiliser	<nombre> ou % {1,4} [fill]	100%	blocs	
border-image-source	fichier de l'image modèle	none <url>	none	blocs	
border-image-width	épaisseur de la bordure	auto <longueur> ou % {1,4}	<border-width> sinon 1px	blocs	
border-radius	arrondi des coins	none <longueur> ou % {1,4} [<longueur> ou % {1,4}]	0	blocs	
border-top-left-radius	arrondi du coin supérieur gauche	<longueur> ou % {1,2}	0	blocs	
border-top-right-radius	arrondi du coin supérieur droit	<longueur> ou % {1,2}	0	blocs	
box-shadow	ombrage d'un bloc	none x y flou épaisseur couleur [inset] [, ...]+	none	blocs	
box-sizing	dimensions incluant ou non padding et bordures	content-box padding-box border-box	content-box	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
clip-path	zone visible	<forme> <url> none inherit	none	éléments de type bloc et éléments remplacés	
columns	multicolonnage	[column-count column-width]	voir chaque propriété	blocs	
column-count	nombre de colonnes	auto <entier>	auto	blocs	
column-fill	équilibrage des colonnes	auto balance	balance	blocs	
column-gap	espacement entre colonnes	<largeur> <pourcentage> auto normal inherit	normal (1em)	blocs	% relatif au contenu
column-rule	trait de séparation des colonnes	[column-rule-style column-rule-width column-rule-color]	voir chaque propriété	blocs	
column-rule-color	couleur du trait de séparation des colonnes	<couleur>	valeur de la propriété 'color'	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
column-rule-style	type de trait de séparation des colonnes	<border-style>	none	blocs	
column-rule-width	épaisseur du trait de séparation des colonnes	<border-width>	medium	blocs	
column-span	étalement d'un titre sur plusieurs colonnes	1 all	1	blocs	
column-width	largeur des colonnes	auto <largeur>	auto	blocs	% de la largeur du contenu
flex	coefficients d'accroissement, de réduction et dimension de base d'un item flex	[flex-grow flex-shrink flex-basis]	voir chaque propriété	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
flex-basis	dimension de base d'un item flex pour son ajustement	<longueur> auto initial inherit	auto	blocs	% de la largeur du conteneur
flex-direction	sens et direction des items flex	row row-reverse column column-reverse initial inherit	row	blocs	
flex-flow	direction et retour à la ligne des items flex	[flex-direction flex-wrap]	voir chaque propriété	blocs	
flex-grow	coefficient relatif d'accroissement d'un item flex	<nombre> initial inherit	0	blocs	
flex-shrink	coefficient relatif de réduction d'un item flex	<nombre> initial inherit	1	blocs	
flex-wrap	retour à la ligne des items flex	nowrap wrap wrap-reverse initial inherit	nowrap	blocs	

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcen- tage (si utilisé)
font-size-adjust	rapport minuscules / majuscules	none <nombre> initial inherit	none		
font-stretch	largeur des lettres	ultra-condensed extra-condensed condensed semi-condensed normal semi-expanded expanded extra-expanded ultra-expanded inherit (H)	normal		
justify-content	répartition des items flex sur l'axe principal	flex-start center flex- end space-between space- around initial inherit	flex- start	blocs	
nav-index	ordre de tabulation dans un menu	auto <nombre> inherit	auto		
nav-up, nav-down, nav-left, nav-right	naviguation au clavier	auto <id> inherit	auto		
opacity	taux d'opacité	<nombre> (entre 0 et 1 - décimale=point) inherit	1 (opacité totale)		

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
order	priorité pour l'ordre d'affichage d'un item flex	<nombre> initial inherit	0	blocs	
outline-offset	espacement entre border et outline	<longueur> inherit	0		
resize	bloc redimensionnable	none both horizontal vertical inherit	none	blocs avec overflow, hidden ou scroll	
text-decoration (raccourci CSS 3)	soulignement surlignement texte barré	<text-decoration-line> [<text-decoration-style>] [<text-decoration-color>]	voir chaque propriété		
text-decoration-color	couleur du trait	<couleur> initial inherit	<color>		
text-decoration-line	position du trait	none underline overline line-through initial inherit	none		
text-decoration-style	type de trait	solid double dotted dashed wavy initial inherit	solid		

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
text-overflow	marqueur de coupure de ligne	clip ellipsis <chaîne> initial inherit	clip	blocs	
text-shadow	ombrage du texte	none x y flou couleur [,x y flou couleur ,...](H)	none		
transform	transformation par une fonction géométrique	none translateX(..) translateY(..) translate(..) scaleX(..) scaleY(..) scale(..) rotate(..) skewX(..) skewY(..) skew(..)	none		
transform-origin	origine de l'opération de transformation	[<longueur> % mots-clés] {1,2} avec mots-clés = combinaisons parmi top / center / bottom et left / center / right	50 % 50 %		
transition	changement progressif de propriétés	[<'transition-property'> <'transition-duration'> <'transition-timing-function'> <'transition-delay'> [, ...]+	voir chaque propriété		

TABLEAU C–2 Index des propriétés d'affichage CSS 3 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcen- tage (si utilisé)
transition-delay	délai avant le début de la transition	<durée> [, <durée>]+	0		
transition-duration	durée de la transition	<durée> [, <durée>]+	0		
transition-property	propriétés pour lesquelles il y aura transition	none all <propriété> [, <propriété>]+	all		
transition-timing- function	évolution de la vitesse de transition	ease linear ease-in ease-out ease-in-out step-start step-end steps(nombre[, start end]) cubic-bezier(x1,y1,x2,y2)[, ...]+	ease		
word-break	coupure des mots longs	normal break-all keep-all (H)	normal		

Pour compléter ce tableau résumant les nouvelles propriétés CSS 3, il ne faut pas oublier les nouvelles valeurs possibles en CSS 3 pour quelques propriétés déjà présentes dans la norme CSS 2 :

- background, background-attachment, background-image, backgroundposition, background-repeat : plusieurs valeurs possibles, pour le support des images de fond multiples ;
- background-attachment : valeur local, en plus de scroll (par défaut) et fixed.

Styles pour les médias paginés et sonores

Média paginé

La plupart de ces propriétés de type `@media print` sont communes aux normes CSS 2 et CSS 3 ; seules les propriétés `image-orientation`, `fit` et `fitposition` sont apparues avec les CSS 3.

TABLEAU C–3 Index des propriétés CSS pour les médias paginés

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à
<code>image-orientation</code>	rotation d'image	<code>auto</code> <code><angle></code>	<code>auto</code>	images
<code>orphans</code>	orphelines	<code><entier></code> <code>inherit</code> (H)	2	éléments de type bloc
<code>page</code>	choix de la page destination	<code><identifiant></code> <code>auto</code> (H)	<code>auto</code>	éléments de type bloc
<code>page-break-after</code>	saut de page après	<code>auto</code> <code>always</code> <code>avoid</code> <code>left</code> <code>right</code> <code>inherit</code>	<code>auto</code>	éléments de type bloc
<code>page-break-before</code>	saut de page avant	<code>auto</code> <code>always</code> <code>avoid</code> <code>left</code> <code>right</code> <code>inherit</code>	<code>auto</code>	éléments de type bloc

TABLEAU C–3 Index des propriétés CSS pour les médias paginés (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à
page-break-inside	autorisation de saut de page	avoid auto inherit (H)	auto	éléments de type bloc
size	portrait-paysage / taille	<longueur>{1,2} auto [<format-page> [portrait landscape]]	auto	contexte de page
widows	veuve	<entier> inherit (H)	2	éléments de type bloc

Média sonore

Les CSS 2 proposaient déjà des styles associés au type @media speech. Cependant, les noms de propriétés, leurs valeurs et la façon même d'aborder la lecture sonore ont été très largement remaniés par la norme CSS 3.

TABLEAU C-4 Index des propriétés CSS 3 pour les médias sonores

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
cue	raccourci pour cue-before et cue-after	<'cue-before'> <'cue-after'>?	voir chaque propriété	
cue-after	délimitation sonore après l'élément lu	<uri> <decibel>? none	none	
cue-before	délimitation sonore avant l'élément lu	<uri> <decibel>? none	none	
pause	raccourci pour pause-before et pause-after	[[<durée> <'pause-before'> <'pause-after'>?]	none	
pause-after	pause après élément et délimitation sonore	<durée> none x-weak weak none medium strong x-strong	none	
pause-before	pause avant élément et délimitation sonore	<durée> none x-weak weak none medium strong x-strong	none	
rest	raccourci pour rest-before et rest-after	<'rest-before'> <'rest-after'>?	none	

TABLEAU C-4 Index des propriétés CSS 3 pour les médias sonores (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
rest-after	pause après élément et avant délimitation sonore	<durée> none x-weak weak medium strong x-strong	none	
rest-before	pause avant élément et après délimitation sonore	<durée> none x-weak weak medium strong x-strong	none	
speak	texte à lire ou non	auto none normal (H)	auto	
speak-as	lecture des ponctuations	normal spell-out digits [literal-punctuation no-punctuation] (H)	normal	
voice-balance	balance gauche/droite du son	<number> left center right leftwards rightwards (H)	center	
voice-duration	durée de lecture	auto <durée>	auto	
voice-family	type de voix à utiliser	[[<nom> <voix-générique>],]* [<nom> <voix-générique>] preserve (H) où <voix-générique> = [<âge>? <genre> <nombre>?]	selon le navigateur	

TABLEAU C-4 Index des propriétés CSS 3 pour les médias sonores (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
voice-pitch	fréquence moyenne de la voix	<fréquence> && absolute [[x-low low medium high x-high] [<fréquence> <demi-ton> <pourcentage>]] (H)	medium	
voice-range	étendue des variations de tonalité	<fréquence> && absolute [[x-low low medium high x-high] [<fréquence> <demi-ton> <pourcentage>]] (H)	medium	
voice-rate	vitesse de lecture	[normal x-slow slow medium fast x-fast] <pourcentage> (H)	normal	
voice-stress	accentuation des mots en relief	normal strong moderate none reduced (H)	normal	
voice-volume	volume moyen du son	silent [[x-soft soft medium loud x-loud] <decibel>] (H)	medium	

Propriétés classées par catégories

Quelles sont les propriétés CSS qui peuvent être utilisées pour un paragraphe, un tableau ou une liste ?

L'index précédent classait les propriétés par ordre alphabétique.

Voici à présent les noms seuls des principales propriétés, avec cette fois un regroupement par catégories selon l'usage.

Propriétés communes aux CSS 2 et CSS 3

Caractères

```
background-color,  
color,  
font,  
font-family,  
font-size,  
font-style,  
font-variant,  
font-weight,  
text-decoration,  
text-transform,  
vertical-align
```

Mots, paragraphes et blocs de texte

```
background,  
background-attachment, background-color,  
background-image, background-position,  
background-repeat,  
  
border,  
border-top, border-right, border-bottom, border-left,  
border-color,  
border-top-color, border-right-color,  
border-bottom-color, border-left-color,  
border-spacing,  
border-style,  
border-top-style, border-right-style,  
border-bottom-style, border-left-style,  
border-width,
```

```
border-top-width, border-right-width,  
border-bottom-width, border-left-width,  
  
outline,  
outline-color, outline-style,  
outline-width, outline-offset,  
  
margin,  
margin-top, margin-right, margin-bottom, margin-left,  
  
height, width,  
max-height, max-width, min-height, min-width,  
  
padding,  
padding-top, padding-right,  
padding-bottom, padding-left,  
  
text-align, text-indent,  
line-height, letter-spacing, word-spacing,  
white-space,  
  
content, quotes,  
counter-increment, counter-reset,  
direction, unicode-bidi, cursor
```

Listes à puces ou à numéros

```
list-style,  
list-style-image,  
list-style-position,  
list-style-type
```

Tableaux

```
border-collapse,  
border-spacing,  
caption-side,  
empty-cells,  
table-layout,  
text-align,  
vertical-align
```

Positionnement

```
display, visibility,  
float, clear,  
position,
```

```
top, bottom,  
right, left,  
  
overflow,  
z-index
```

Propriétés spécifiques aux CSS 3

Mots, paragraphes et blocs de texte

```
@font-face, font-stretch, font-size-adjust,  
appearance, word-break,  
opacity, text-shadow, clip-path,  
text-decoration-line, text-decoration-style,  
text-decoration-color  
  
nav-index,  
nav-up, nav-down, nav-left, nav-right,  
  
box-shadow, outline-offset,  
border-image,  
border-image-source, border-image-slice,  
border-image-width, border-image-outset,  
border-image-repeat  
border-radius,  
border-top-left-radius, border-top-right-radius,  
border-bottom-left-radius, border-bottom-right-radius,  
  
box-sizing, resize,  
text-overflow
```

Images de fond et dégradés de couleurs

```
background-origin, background-clip,  
background-size, background-attachment,  
background: linear-gradient(...),  
background: radial-gradient(...),  
background: repeating-linear-gradient(...),  
background: repeating-radial-gradient(...)
```

Multicolonnage

```
columns,  
column-count, column-width,  
column-gap,  
column-rule,
```

```
column-rule-style, column-rule-width,  
column-rule-color  
column-span, column-fill
```

Modèle flexbox

```
Pour le conteneur flex-box  
flex-flow,  
flex-direction, flex-wrap,  
justify-content, align-items, align-content
```

```
Pour les items flex  
flex,  
flex-grow, flex-shrink, flex-basis,  
align-self, order
```

Transformations géométriques

```
transform: ...  
  
avec les fonctions 2D  
matrix(a,b,c,d,e,f),  
translateX(x), translateY(y), translate(x,y)  
scaleX(x), scaleY(y), scale(x,y)  
rotate(angle),  
skewX(angle), skewY(angle), skew(angle1, angle2)  
et avec les fonctions 3D  
matrix(a,b,c,d,e,f,g,h,i,j,k,l,m,o,p),  
translateX(x), translateY(y), translateZ(z),  
translate3d(x,y,z)  
scaleX(x), scaleY(y), scaleZ(z), scale(x,y,z)  
rotateX(angle), rotateY(angle), rotateZ(angle),  
rotate3d(x,y,z,angle),  
perspective(profondeur)  
  
transform-origin  
transform-style, backface-visibility  
perspective, perspective-origin
```

Transitions et animations

```
transition, transition-property, transition-duration,  
transition-timing-function, transition-delay  
@keyframes, animation,  
animation-name, animation-duration,  
animation-timing-function, animation-delay,  
animation-iteration-count, animation-direction,
```

```
animation-fill-mode, animation-play-state
```

Médias paginés et sonores

Médias paginés

```
page-break-before, page-break-after,  
page-break-inside  
size, page  
orphans, widows  
image-orientation
```

Médias sonores

```
voice-volume, voice-balance  
speak, speak as  
voice-family, voice-rate, voice-pitch  
voice-range, voice-stress  
voice-duration  
cue, cue-before, cue-after  
pause, pause-before, pause-after  
rest, rest-before, rest-after
```

annexe
D

Références bibliographiques et sites web



Voici maintenant quelques références et sites web intéressants pour compléter cet ouvrage et aller plus loin.

SOMMAIRE

- ▶ **Bibliographie**
- ▶ **Sites web utiles**

Cet ouvrage visait à présenter les techniques de base nécessaires pour créer des pages web, à savoir le HTML, les feuilles de styles CSS et leur mise en pratique. Une fois que vous aurez compris la philosophie de la programmation HTML/CSS, il pourra continuer à vous servir d'aide-mémoire.

Pour aller plus loin, n'hésitez pas à consulter d'autres livres qui vous permettront d'approfondir certains points et de découvrir des techniques complémentaires.

Bibliographie

Voici une liste d'ouvrages qui sont actuellement des références en matière de développement HTML/CSS :

- *Réalisez votre site web avec HTML5 et CSS 3*, de Mathieu Nebra, éditions Eyrolles
- *CSS 3 pour les web designers*, de Dan Cederholm, éditions Eyrolles
- *HTML 5 pour les web designers*, de Jeremy Keith et Rachel Andrew, éditions Eyrolles
- *HTML 5*, de Rodolphe Rimelé, éditions Eyrolles
- *Mémento CSS 2* et *Mémento CSS 3*, de Raphaël Goetter, éditions Eyrolles
- *CSS 3*, d'Hugo Giraudel et Raphaël Goetter, éditions Eyrolles
- *Responsive Web Design*, d'Ethan Marcotte, éditions Eyrolles
- *CSS - Techniques professionnelles pour une mise en page moderne*, d'Éric Meyer, éditions Pearson Campuspress

Sites web utiles

Les quelques sites web qui suivent sont très intéressants (attention à ne pas oublier le signe / qui termine certaines adresses). Vous pourrez y glaner d'autres informations, applications pratiques et astuces. Cette liste n'est évidemment pas exhaustive : ce n'est qu'un aperçu des trésors de la toile...

Les pages de spécifications des normes, complètes mais souvent austères, font évidemment partie des références de base.

- Le World Wide Web Consortium (W3C) met à notre disposition l'ensemble des normes web :
 - ▶ <http://www.w3.org>
- Le W3C propose notamment les spécifications officielles en anglais des différentes versions des normes CSS :
 - ▶ <http://www.w3.org/TR/CSS>
- Il existe une traduction en français de certaines spécifications officielles du W3C, dont les feuilles de styles CSS 2 :
 - ▶ <http://www.yoyodesign.org/doc/w3c/index.php>
- La communauté de propositions et d'aide WHATWG est à l'origine de la norme HTML 5 (en anglais) :
 - ▶ <https://www.whatwg.org/>

Il existe par ailleurs un grand nombre de sites pédagogiques et de tutoriels qui sont très accessibles, même lorsqu'ils sont en anglais.

- W3Schools est un site en anglais comprenant nombre de tutoriels autour des normes du Web, avec des exemples qui peuvent être testés et modifiés :
 - ▶ <https://www.w3schools.com>
- Des tutoriels CSS sont disponibles sur le site Mammouthland.net :
 - ▶ <http://css.mammouthland.net/tutoriels-css.php>
- Le site Pompage a pour sous-titre « Web design puisé à la source » et traduit en français une sélection d'articles parus à propos du Web :
 - ▶ <http://pompage.net/>
- Encore très connu sous son ancien nom « Le Site du Zéro », le site OpenClassrooms contient des tutoriels, des articles et des forums :

- ▶ <https://openclassrooms.com/>
- Le site CSS-Tricks (<https://css-tricks.com>) contient de nombreux articles et tutoriels, et en particulier la rubrique *Almanac* qui propose une référence très complète des propriétés CSS (en anglais), chacune étant accompagnée d'un exemple et d'une illustration :
 - ▶ <https://css-tricks.com/>
- La communauté Codrops propose une référence sur les CSS (menu *CSS Reference*) et nombre de tutoriels :
 - ▶ <https://tympanus.net/codrops/>
- Outre une sélection de scripts, le site ToutJavaScript.com propose des tutoriels pour le JavaScript :
 - ▶ <http://www.toutjavascript.com/savoir>
- Alsacreations est un lieu d'échange de tutoriels et d'astuces :
 - ▶ <https://www.alsacreations.com>
- Le forum d'Alsacreations est également une bonne source de renseignements sur des cas pratiques :
 - ▶ <https://forum.alsacreations.com>
- Framasoft est une ressource très utile qui, en plus de proposer un répertoire très important de logiciels libres et de documentations, rassemble nombre de projets logiciels et de services autour du « Libre », contribuant ainsi à l'indépendance des utilisateurs de l'informatique dans leur vie quotidienne :
 - ▶ <https://framasoft.org>
- CSS Zen Garden offre une démonstration très esthétique des possibilités apportées par les feuilles de styles :
 - ▶ <http://csszengarden.com/tr/francais>

Framasoft

La route est longue mais la voie est libre...

Un réseau dédié à la promotion du « LIBRE » en général et du LOGICIEL LIBRE en particulier.

De nombreux services et projets innovants mis librement à disposition du grand public.

Une communauté de bénévoles soutenue par une association d'intérêt général.

Une invitation à bâtir ensemble un monde de partage et de coopération.

Vous cherchez l'annuaire de logiciels libres ? C'est par ici !



Quoi de neuf sur Framasoft ?



FIGURE D-1 Extraits du site <https://framasoft.org>, réalisation de la communauté francophone du logiciel libre qui répertorie plus de 1 600 logiciels libres et milite pour le « Libre » en informatique, proposant des services en ligne et des projets innovants, pour que l'utilisateur conserve sa liberté de choix.

Libres logiciels

- Un annuaire > [Framalibre](#)
- Une clé USB > [Framakey](#)
- Un DVD > [Framadvd](#)
- Un distributeur > [Framapack](#)

Libres cultures

- Des Infos > [Framablog](#)
- Des livres > [Framabook](#)
- Des traductions > [Framalang](#)
- Des vidéos > [Framatube](#)
- De la musique > [Framazic](#)

Libres services

- Éditer > [Framapad](#)
- Organiser > [Framadate](#)
- Calculer > [Framacalc](#)
- Structurer > [Framindmap](#)
- Dessiner > [Framavectoriel](#)
- Partager > [Framasphère](#)

Lib'ren vrac

- Un laboratoire > [Framalab](#)
- Un forum > [Framagora](#)
- Une forge > [Framacode](#)
- Une boutique > [EnVenteLibre](#)

Index

Symboles

!important 129
@font-face 238
@import 385
@media 384, 412

A

abbr (balise) 75
absolute (position) 203, 215
accessibilité 4
aux personnes handicapées 34, 47
address (balise) 56, 74
align-content (propriété flexbox) 284
alignement
du texte 156
vertical (cellule de tableau) 185
align-items (propriété flexbox) 287
align-self (propriété flexbox) 287
animation 320
animation (raccourci) 330
animation-delay 325
animation-direction 326
animation-duration 323
animation-fill-mode 328
animation-iteration-count 326
animation-name 323
animation-play-state 329
animation-timing-function 324
appearance (propriété) 244
arrière-plan 173
arrondi (coin) 251
article (balise) 55
aside (balise) 55

audio (balise) 50
audio (média) 421

B

background 177
 background-attachment 177, 262
 background-clip 261
 background-color 153, 174
 background-image 174, 259
 background-origin 260
 background-position 176
 background-repeat 175
 background-size 261
balise
 changement de type d'élément 209
 classe 117
 de section 55
 identifiant 119
 imbrication 19
 principe 14
bloc 28
 affichage (visibility) 207
 centrage horizontal 221
 centrage vertical 222
 changement de type d'élément (display) 209
 débordements (overflow) 207
 délimitation 211
 dimension fixe 256
 hauteur fixe 194
 hauteur maximale 197
 hauteur minimale 197
 imbriqué 62
 largeur fixe 194
 largeur maximale 197
 largeur minimale 197
 marges externes (margin) 190
 marges internes (padding) 192
 modèle de boîte 195
 positionnement 201
 zone visible 208
blockquote (balise) 32
body (balise) 20
bordure

arrondi 251
border 172
border-collapse (tableaux) 182
border-color 170
border-image 247
border-radius 252
border-style 168
border-width 169
box-shadow 254
contour des cellules vides (tableaux) 184
espacement entre bordures (tableaux) 183
ombrage 254
outline-offset 255
bottom 203
box-sizing 256
button (balise) 100

C

cadre (balise iframe) 77
canvas (balise) 80
caption (balise) 42
caption-side 184
caractères
 adaptation de la taille 236
 background-color 153
 color 148
 décalage vertical 154
 étirement 236
 font 155
 font-family 146
 font-size 148
 font-size-adjust 236
 font-stretch 236
 font-style 151
 font-variant 153
 font-weight 150
 majuscules/minuscules 152
 mise en forme 146
 petites majuscules 153
 soulignement 151
 surlignage 153
 text-decoration 151
 text-transform 152

vertical-align 154
case à cocher 88
cellpadding (attribut de balise table) 42
cellspacing (attribut de balise table) 42
centrage
 horizontal 221
 vertical 222
cite (balise) 74
classe 121
clear 206
clip-path 208
codage de la page 25
code (balise) 75
coin arrondi 251
colonne de texte 268
 column 269
 column-count 268
 column-fill 272, 276
 column-gap 269
 column-rule 271
 column-rule-color 271
 column-rule-style 270
 column-rule-width 270
 column-span 273
 column-width 268
color 148
colspan (attribut de balise table) 44
commentaire
 CSS 111
 HTML 22
compteur automatique
 content 162
 counter-increment 165
 counter-reset 164
content 162
contenteditable (attribut) 77
corps de la page 27
couleur
 arrière-plan 174, 259
 choix et harmonisation des couleurs 428
codage
 RVB 137
 RVBA 230

TSL (HSL) 232
dégradé 263
du soulignement 234
du texte 148
les 16 couleurs de base 430
noms 137, 432
utilisation 6
coupure des mots 73, 246
curseur de la souris 161
cursor 161

D

datalist (balise) 101
date et heure 71
dd (balise) 40
décalage
transformations 2D 308
transformations 3D 310
vertical 154
dégradé de couleurs
linéaire 263
radial 264
répétitif 267
del (balise) 75
details (balise) 79
dfn (balise) 74
dimension *voir* bloc
direction (sens de l'écriture) 165
display 209
div (balise) 28, 55
dl (balise) 40
DOCTYPE 24
draggable (attribut) 75
dt (balise) 40

E

échelle (transformation) 309, 310
élément
bloc 59
en ligne 59
niveau texte 59
remplacé 60
em (balise emphase) 15, 29

- empty-cells 184
- en-tête de la page 25
- espacement
 - conservation des espacements 161
 - entre les lettres 159
 - entre les lignes 158
 - entre les mots 160
- exemple de site
 - adaptation au Web mobile 393
 - page complète (menu horizontal) 371
 - page complète (menu vertical) 362
 - structure de base (menu horizontal) 352
 - structure de base (menu vertical) 356
- exposant 30

F

- feuille de styles
 - commentaires 111
 - exemple 139
 - externe 112
 - hiérarchie 126
 - intérêt 9
 - interne 112
 - introduction 1
 - priorité des règles 129
 - règle de style 110
 - sélecteur 115
 - style en ligne 114
- fieldset (balise) 85
- figcaption (balise) 83
- figure (balise) 83
- fixed (position) 203, 217
- flexbox (technique) 274, 287, 302
 - align-content 284
 - align-self 287
 - flex-basis 297
 - flex-flow 279
 - flex-grow 289
 - flex-shrink 293
 - flex-wrap 278
 - justify-content 280
 - order 301
- float 205, 219

flux normal des éléments 198

font 155

font-family 146

font-size 148

font-size-adjust 237

font-stretch 236

font-style 151

font-variant 153

font-weight 150

footer (balise) 56

formulaire 84

attributs 103

autocomplete (attribut) 94

boutons radio 87

button (balise) 100

cases à cocher 88

contrôle de la saisie 94, 95

effacement 90

envoi 90, 91

étiquettes 86

form (balise) 84

liste déroulante 88

liste déroulante modifiable 101

regroupements (fieldset) 85

saisie sur plusieurs lignes 87, 105

target (attribut) 94

zone de texte 86

G

GIF (format d'image) 46

gras 29, 150

guillemets automatiques

affichage 162

choix 164

H

h1, h2, h3... (balises) 27

handicapés (accessibilité) 34, 47

hauteur

fixe 194

maximale 197

minimum 197

head (balise) 20

header (balise) 56
height 194
 attribut de balise 48
héritage 61
hiérarchie
 des éléments HTML 61
 des sélecteurs de styles 126
html (balise) 24

I

identifiant 121
iframe (balise) 35, 77
image
 alignement en arrière-plan 176
 arrière-plan 174, 259
 attachement arrière-plan 177
 balise img 46
 dynamique 80
 espace sous l'image 215
 figure (balise) 83
 formats JPEG, GIF, PNG 46
 height (attribut) 48
 raccourci background 177
 répétition 175
 width (attribut) 48
imbrication
 d'éléments 126
 des balises 19
 des blocs 62
indice 30
input (balise) 86, 95
ins (balise) 75
interligne 158
Internet Explorer
 modèle de boîte spécifique 195
italique 15, 29, 151

J

JPEG (format d'image) 46
justify-content (propriété flexbox) 280
juxtaposition d'éléments 127

K

kbd (balise) 75

L

label (balise) 86

largeur

fixe 194

maximale 197

minimum 197

left 203

letter-spacing 159

lien

hypertexte 33

mailto 36

linear-gradient (propriété) 263

line-height 158

liste

à puces 38

balises de liste HTML 37

définition 40

déroulante 88

déroulante modifiable 101

image comme puce 179

list-style 181

numérotée (personnaliser) 39

position de la puce 180

type de puce ou numérotation 178

M

mailto 36

majuscules/minuscules 152

marge

externe (margin) 190

interne (padding) 192

modèle de boîte 195

mark (balise) 70

max-height 197

max-width 197

média

affichage 412

impression 414

portable 382

sélecteur (media queries) 386

sonore 421

type 412
media queries 386
menu
 horizontal 352
 vertical 356
meta (balise) 25
meter (balise) 71
min-height 197
min-width 197
mobile (Web)
 adaptation d'un site 393
modèle de site
 avec menu horizontal 352
 avec menu vertical 356
 avec une seule image de fond 362
mobile 393
page complète (menu horizontal) 371
structure générale 346
multicolonnage 268
multimédia (objet) 50

N

nav (balise) 55
navigation au clavier 244
 nav-down (propriété) 244
 nav-index (propriété) 244
 nav-left (propriété) 244
 nav-right (propriété) 244
 nav-up (propriété) 244
numérotée (liste) 38

O

ol (balise) 38
ombrage
 bloc 254
 caractères 242
opacity 230
optgroup (balise) 90
option (balise) 89
order (propriété flexbox) 301
orphelines (orphans) 416
outline 173
outline-offset (propriété) 255

overflow 207

P

padding 192

page

- autorisation des sauts de page 417
- dimensions 418
- nom 419
- référence à un nom de page 419
- saut de page après 417
- saut de page avant 416
- sélecteur 418

paragraphe

- alignement horizontal 156
- background 177
- background-attachment 177
- background-color 174
- background-image 174
- background-position 176
- background-repeat 175
- balise p 27
- conservation des espacements 161
- cursor 161
- espace entre les lettres 159
- espace entre les mots 160
- interligne 158
- letter-spacing 159
- line-height 158
- mise en forme 156
- retrait de première ligne 157
- text-align 156
- text-indent 157
- white-space 161
- word-spacing 160

perspective (transformation 3D) 310

petite majuscule 153

PNG (format d'image) 46

police de caractères 146

- @font-face 238

- adaptation de la taille 236

- étirement des lettres 236

- font-size-adjust 236

- ombrage 242

position
absolue 203, 215
dans le flux normal 203
décalage 203
exemples 212
fixe 203, 217
flottante 205, 219
flux normal 198
position (propriété) 203
principe 198
relative 203, 216
superposition des blocs 204
types de positionnement 201
z-index 204
pre (balise) 75
préfixe navigateur 228
print (média) 414
progress (balise) 83
propriétés CSS
classées par catégories 494
héritage 132
raccourci 125
tableau de résumé 458
pseudo-classe 122, 334
pseudo-élément 124
puces (liste à) 38

Q

q (balise) 74
quotes 164

R

radial-gradient 264
règle de style
commentaires 111
définition 110
héritage 132
priorité 129
sélecteur 115
relative (position) 203, 216
repeating-linear-gradient 267
repeating-radial-gradient 267
responsive web design 386

retrait de première ligne 157
right 203
rotation (transformation) 309, 310
rowspan (attribut de balise table) 44

S

samp (balise) 75
section
 balise 55
 principe en HTML 5 55
select (balise) 88
sélecteur
 ancre ou cible d'un lien 340
 balise vide 339
 classe 117
 d'attribut 127, 331
 de voisinage 331
 différence entre classe et identifiant 121
 élément coché 336
 hiérarchie 126
 identifiant 119
 imbrication directe d'éléments 127
 juxtaposition d'éléments 127
 plusieurs sélecteurs 124
 pseudo-classe 122, 334
 pseudo-élément 124
 simple 116
 universel 128
sens de l'écriture 165
small (balise) 31
smartphone 382
sonore (média) 421
soulignement 151
 forme 235
 type 235
source (balise) 52
span (balise) 28
static (position) 203
strong (balise) 29
sub (balise) 30
summary (balise) 79
sup (balise) 30
surlignage 70, 153

T

tableau 181

- alignement vertical des cellules 185
- border-collapse 182
- border-spacing 183
- caption (balise) 42
- cellpadding 42
- cellspacing 42
- col (balise) 44
- colonne fixe ou variable 181
- colspan 44
- contour des cellules vides 184
- empty-cells 184
- espacement entre bordures 183
- fusion de cellules 44
- position du titre 184
- recouvrement des bordures 182
- rowspan 44
- table-layout 181
- taille des caractères 148
- target (attribut) 35, 94
- test des pages 444
- text-align 156
- textarea (balise) 87, 105
- text-decoration 151, 234
- text-decoration-color 234
- text-decoration-line 235
- text-decoration-style 235
- text-indent 157
- text-overflow (propriété) 245
- text-shadow 242
- text-transform 152
- time (balise) 71
- title
 - attribut 34, 48
 - balise 20, 25
- top 203
- transformation géométrique 307
 - fonction 308
 - fonctions 3D 310
 - transform 307
 - transform-origin 307
- transition 312

transition (raccourci) 319
transition-delay 319
transition-duration 315
transition-property 314
transition-timing-function 315
transparence (réglage) 230

U

ul (balise) 38
unicode-bidi 166
unité
 de couleurs 137
 de taille 132
 noms de couleurs 137
utf-8 25

V

validation 66
 code CSS 342
 code HTML 66
var (balise) 75
vertical-align 154, 185
veuves (widows) 415
video (balise) 50
visibility 207

W

wbr (balise) 73
Web mobile
 contraintes 382
 feuille de styles pour mobile 383
white-space 161
widows 415
width
 attribut de balise 48
 propriété 194
word-break (propriété) 246
word-spacing 160

X

XHTML, HTML
 choix des balises 3, 6
 classe 117

identifiant 119

Z

z-index 204

**Pour suivre toutes les nouveautés numériques du Groupe Eyrolles,
retrouvez-nous sur Twitter et Facebook**

 @ebookEyrolles

 EbooksEyrolles

Et retrouvez toutes les nouveautés papier sur

 @Eyrolles

 Eyrolles

Le dernier exemple _ affiche