Programmation Web

Premiers pas en programmation Web

Aline Gérard

Telecom Physique Strasbourg
http://www.telecom-physique.fr/

2019-12-02

Outline

Topic

CSS Donner du style

CSS : les couleurs

CSS : unité de taille

CSS : gestion des marges

CSS : typographie

CSS : exemple en 4 minutes

CSS : mise en page

Principes de bases

- Le CSS permet d'habiller un site web
- On définit des régles d'affichage dans un fichier spécifique
- Les régles sont associées aux éléments HTML

Principes de bases

- Le CSS permet d'habiller un site web
- On définit des régles d'affichage dans un fichier spécifique
- Les régles sont associées aux éléments HTML

Principes de bases

- Le CSS permet d'habiller un site web
- On définit des régles d'affichage dans un fichier spécifique
- Les régles sont associées aux éléments HTML

Définition

- Défini dans un fichier séparé avec pour extension .css
- Un même fichier pourra être utilisé dans plusieurs fichiers HTML
- Plusieurs fichiers CSS peuvent être utilisés
- Défini dans le fichier HTML dans le conteneur <head> via la balise suivante :

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

Définition

- Défini dans un fichier séparé avec pour extension .css
- Un même fichier pourra être utilisé dans plusieurs fichiers HTML
- Plusieurs fichiers CSS peuvent être utilisés
- Défini dans le fichier HTML dans le conteneur <head> via la balise suivante :

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

Définition

- Défini dans un fichier séparé avec pour extension .css
- Un même fichier pourra être utilisé dans plusieurs fichiers HTML
- Plusieurs fichiers CSS peuvent être utilisés
- Défini dans le fichier HTML dans le conteneur <head> via la balise suivante :

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

Définition

- Défini dans un fichier séparé avec pour extension .css
- Un même fichier pourra être utilisé dans plusieurs fichiers HTML
- Plusieurs fichiers CSS peuvent être utilisés
- Défini dans le fichier HTML dans le conteneur <head> via la balise suivante :

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

Syntaxe

la syntaxe est composée de 3 éléments :

le sélecteur élément sur lequel on applique les propriétés (balise HTML, id, classe, etc.)

la propriété est l'effet sur lequel on veut jouer (ex couleur de texte, positionnement, couleur de fond, etc.)

la valeur de la propriété CSS (rouge, 10px, etc.)

syntaxe de base :

selecteur { propriete: valeur; propriete: valeur; ... }

Les commentaires

```
/* Commentaires très utiles */

/*
Les fichiers CSS deviennent vites volumineux,
mettre des commentaires utiles est souvent nécessaire.
*/
```

Règles d'écritures (1/2)

Appliquer un style à une balise

balise { propriété: valeur; }

p { color: red; }

Appliquer à une classe

.class { propriété: valeur; }

Règles d'écritures (2/2)

Appliquer à un id

#id { propriété: valeur; }

Appliquer à une classe relative à un élément

balise.class { propriété: valeur; }

Sélection (1/2)

```
<p class="exemple">Hello World!</p>
<ul id="firstMenu">
  <li></li>
  <li></li>
</ul>

/* tout les éléments paragraphe */
p { }
/* tout les élemnts avec une classe exemple */
.exemple { }
/* tout les paragraphes ayant une class exemple */
p.exemple { }
```

Sélection (1/2)

```
<p class="exemple">Hello World!</p>
<ul id="firstMenu">
  <li></li>
  <li></li>
</ul>

/*
l'unique élément avec l'id firstMenu
ul#firstMenu est inutile sur cette page
mais peut être utile si le css implique plusieurs pages
*/
#firstMenu { }
```

Hiérarchie

Cascade, élément a contenu dans élément nav

```
nav a { color: red; }
```

Même propriété pour tout les éléments (ici h1 et h2)

```
h1, h2 { color: blue; }
```

Priorités des prises en compte

la dernière déclaration est celle qui est prise en compte

id -> class -> balise

L'id est prioritaire sur la classe qui est prioritaire sur la balise

Exemple

```
<p class="message" id="introduction">
Introduction au CSS
</p>

#introduction { color: red; }
.message { color: green; }
p { color: blue; }
```

Résumé

CSS	HTML ciblé
p { }	...
.maClasse { }	...
#minId { }	...
p.maClasse { }	...
p .maClasse { }	...
p, div { }	... ...

Pseudo-class (1/3)

Définition d'une pseudo-class

- liée à l'état ou la position d'un élément

Pseudo-class (2/3)

État d'un lien <a>

```
/* lien non visité */
a:link {
    color: #FF0000;
}

/* lien visité */
a:visited {
    color: #00FF00;
}

/*  lors du survole avec la souris */
a:hover {
    color: #FF00FF;
}
```

Pseudo-class (4/3)

n-iéme élément

```
/* premier élément */
p:first-child { ... }

/* tout les éléments paires */
p:nth-child(2n) { ... }
```

Topic

CSS Donner du style

CSS : les couleurs

CSS : unité de taille

CSS : gestion des marges

CSS : typographie

CSS : exemple en 4 minutes

CSS : mise en page

Notations

Plusieurs notations (exemple avec le blanc) :

hexadécimale	#ffffff
hexadécimale	notation courte doublée pour obtenir la version longue : #fff
RGB	rgb(255,255,255)
RGBA	notion de transparence : rgba(255,255,255,1)
mot clé	white

Propriétés utilisant des couleurs

Notamment utilisé avec les propriétés

- color
- backgroud, background-color
- border

Topic

CSS Donner du style

CSS : les couleurs

CSS : unité de taille

CSS : gestion des marges

CSS : typographie

CSS : exemple en 4 minutes

CSS : mise en page

Utilisation

Utilisées pour les propriétés telles que :

- font-size : taille du texte
- margin : marge entre deux éléments
- padding : espace entre le contenu et sa bordure

Les unités courantes

px	pixels, unité fixe
pt	point
cm, mm	centimètre, millimètre
%	pourcentage, unité fluide
em	cadratin, unité fluide
rem	root em,
vh, vw	viewport units

Le pixel : px

- unité fixe, historique et "rassurante"
- produite par les logiciels graphiques
- taille figée = source de problèmes d'accessibilités
- essentielle pour les éléments tels que les images, les vidéos

Le cadratin : em

- unité fluide
- taille relative à la taille de police du parent direct
- se répercute aux enfants et enfants des enfants (cascade)
- varie en fonction de la taille de police de l'utilisateur

Le cadratin : em, exemple

```
<article>
  <aside>
    <p>Paragraphe dans aside</p>
    <div><p>Paragraphe dans une div</p></div>
    <div>
      <div><p>Paragraphe dans deux div</p></div>
    </div>
  </aside>
</article>

article { font-size: 20px; }
aside { font-size: 1.5em; }
div, p { font-size: 2em; }
```

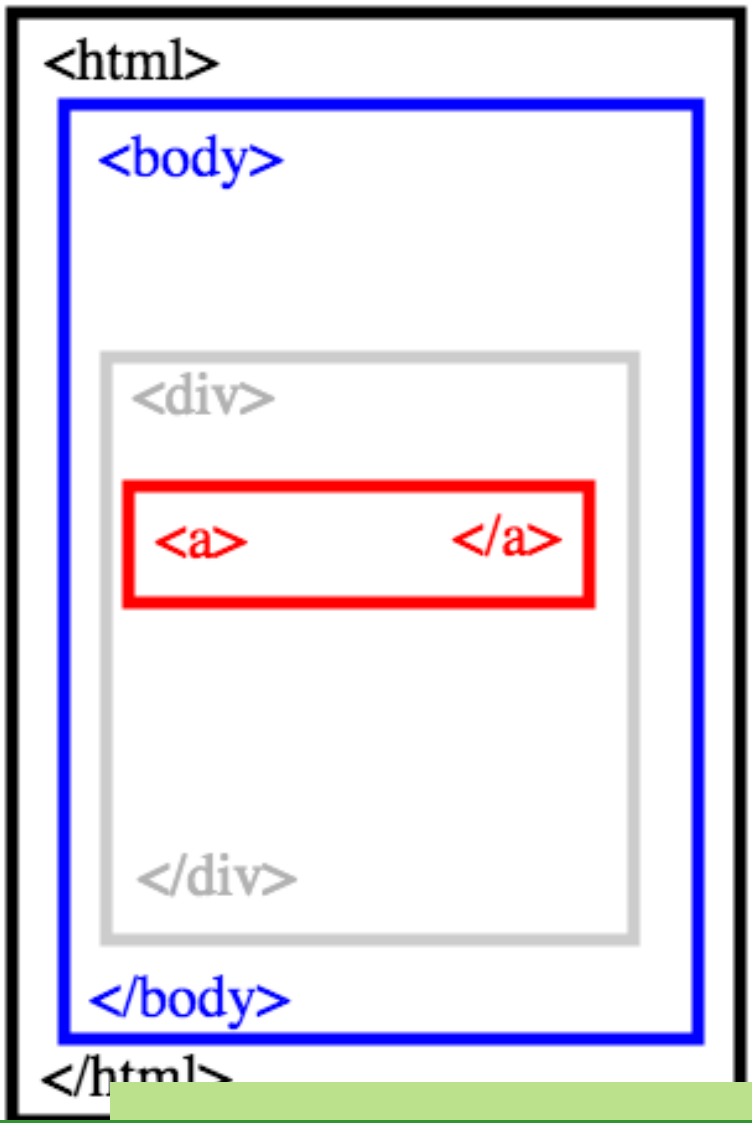
Le pourcentage : %

- unité fluide
- taille relative à la taille de police du parent
- se répercute aux enfants, et enfants des enfants (cascade)
- varie en fonction de la taille de police de l'utilisateur
- varie en fonction de la taille de la fenêtre du navigateur (viewport)

Le pourcentage : %, exemple

```
<article>
  <aside>
    <p>Paragraphe dans aside</p>
    <div><p>Paragraphe dans une div</p></div>
    <div>
      <div><p>Paragraphe dans deux div</p></div>
    </div>
  </aside>
</article>
article { border: 2px solid black; width: 80%; }
div { border: 2px solid red; width: 50%; }
p { border: 2px solid green; width: 50%; }
```

Le pourcentage : %, donner la largeur de l'élément A

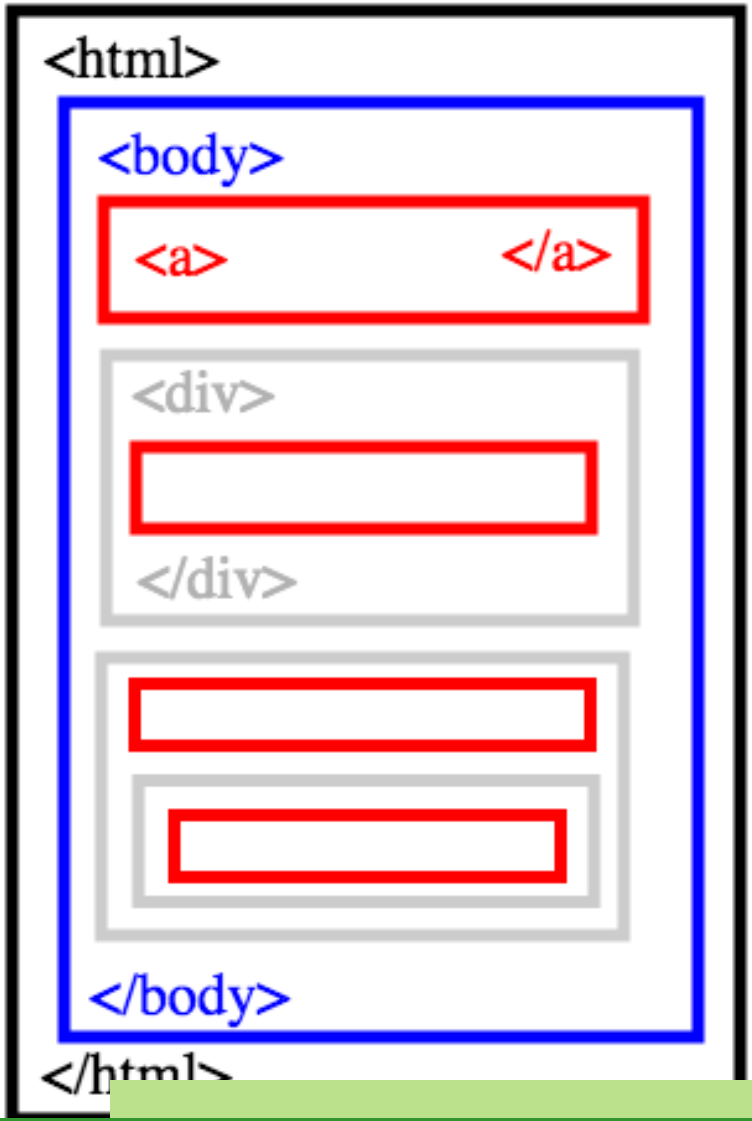



```
body {
    width: 80%;
}
div {
    width: 50%;
}
 a {
    width: 50%;
}
```

Le root em : rem

- unité fluide
- taille relative à la taille de l'élément racine <html>
- aucune répercution sur les enfants
- varie en fonction de la taille de police de l'utilisateur

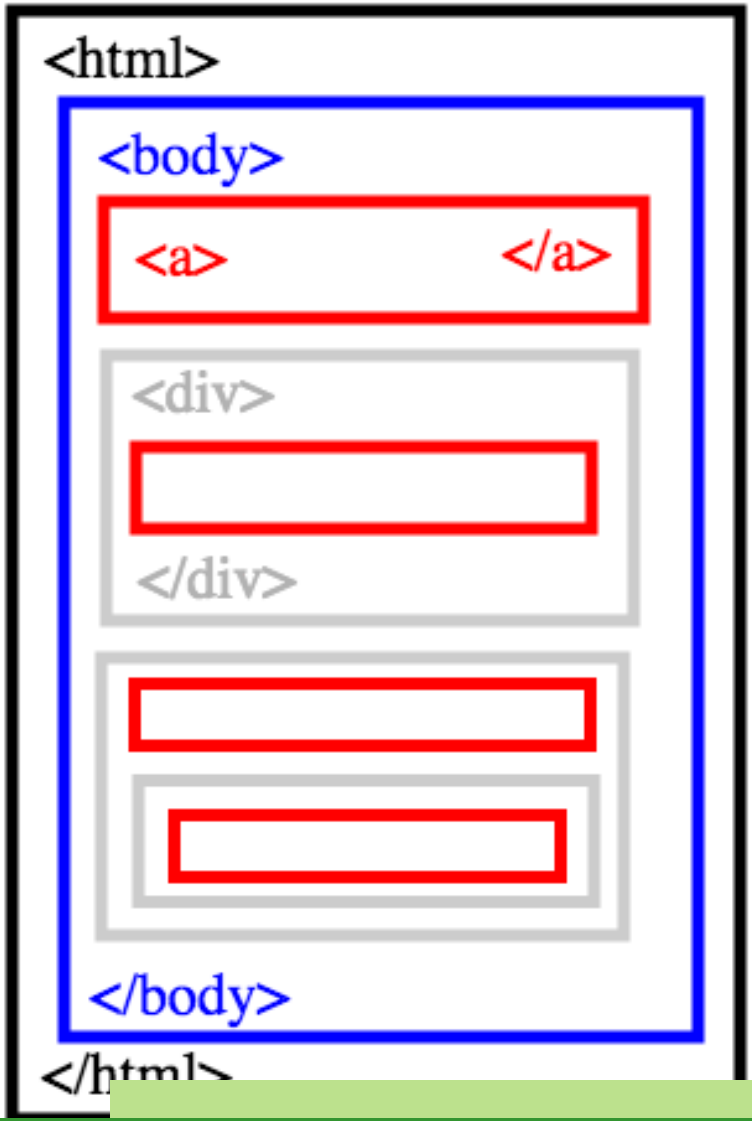
em, exemple, donne la taille de la police des éléments A




```
html {
   font-size: 20px;
}

body {
  font-size: 1.5em;
}
div {
   font-size: 2em;
}
 a {
    font-size: 2em;
}
```

rem, exemple, donne la taille de la police de l'élément A



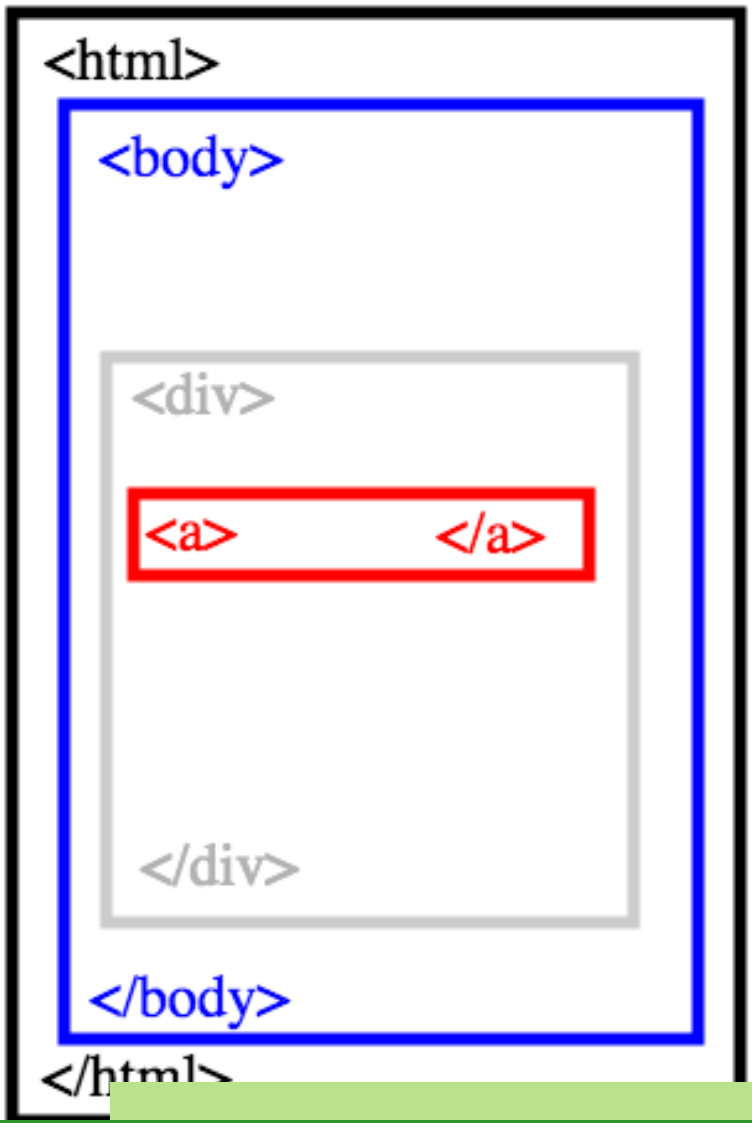

```
html {
    font-size: 20px;
}

body {
    font-size: 1.5em;
}
div {
    font-size: 2em;
}
 a {
    font-size: 2rem;
}
```

Unité viewport : vw, vh

- unité fluide
- viewport = fenêtre du navigateur
- 100vh : hauteur du viewport
- 100vw : largeur du viewport

Viewport: exemple




```
body {
        width: 80%;
}
div {
        width: 50%;
}
 a {
     width: 50vw;
}
```

Topic

Occupation de l'espace

- 2 types d'occupation : bloc ou en ligne
- par défaut les éléments de type block prennent toute la largeur de leur parent, et sont suivis d'un retour à la ligne
  - ex: body, html, p, div, h1
- les éléments de type inline prennent la largeur de leur contenu, et ne sont pas suivis d'un retour à la ligne
  - ex: span, a, strong, img

Largeur et hauteur d'un élément : width et height

- width (largeur) et height (hauteur) sont applicables uniquement sur des éléments de bloc

valeurs possibles

auto par défaut, toute la taille du parent

valeur numérique en px, % , em ex : body{ width: 800px;}

Largeur et hauteur avec max en min

on peut utiliser max et min pour déterminer des tailles maximums et minimums

- min-width
- min-height
- max-width
- max-height

permet notamment de gérer un débordement d'image :

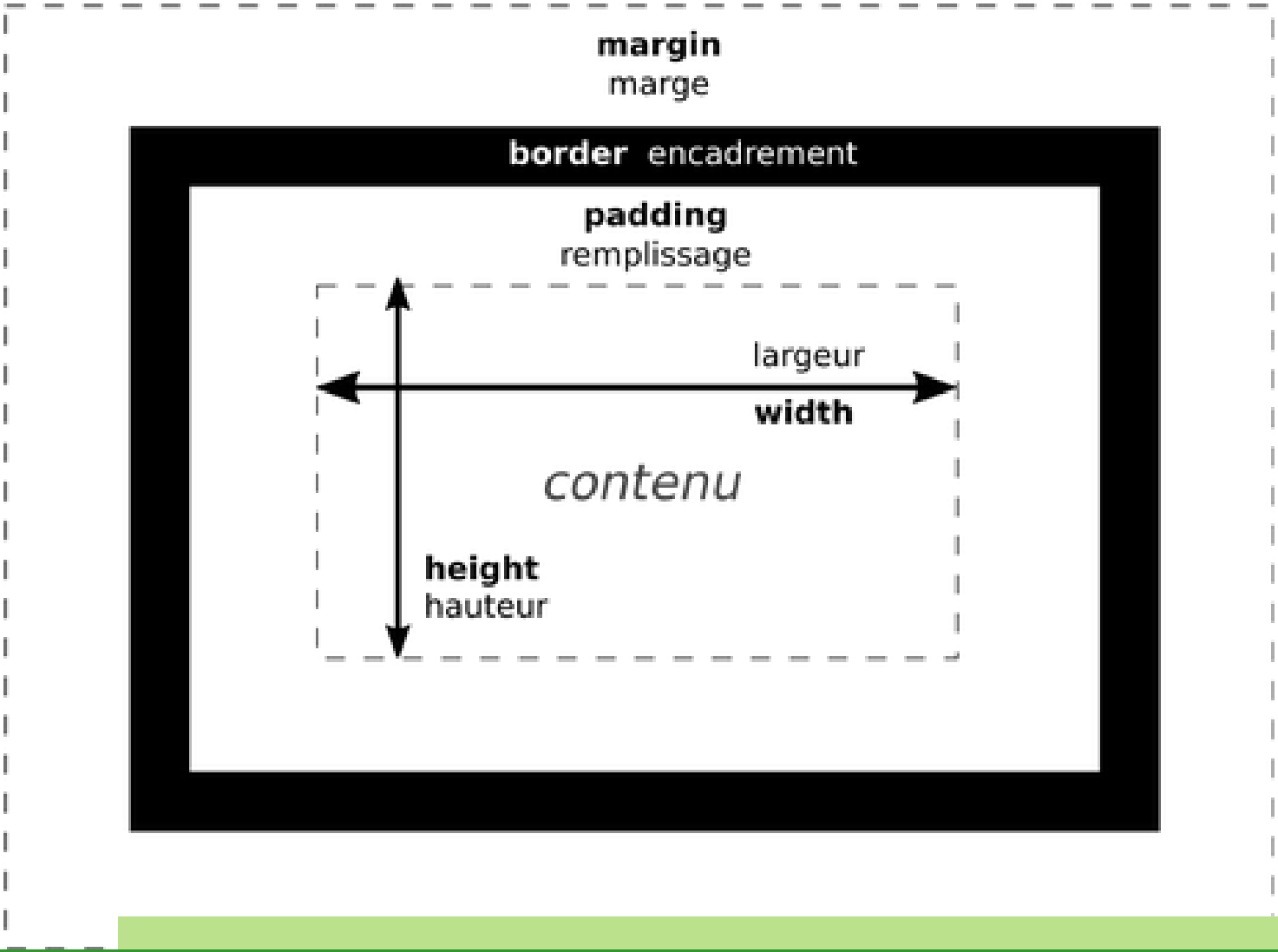
- img { max-width: 100%; }
- les images sont limitées à la taille de leur parent

Gestion des marges

Différentes marges

margin	marge extérieure
padding	marge intérieure
border	cadre entre la mage extérieure et intérieure

CSS BOX


margin
marge
border encadrement
padding
remplissage
largeur
width
contenu
height
hauteur

Définition des marges

définition globale

margin : 5px; applique une marge de 5px aux 4 coins

margin : 0 5px; applique une marge de 0 en haut et bas et 5px sur les côtés

margin : 1px 2px 3px 4px; définition des 4 marges dans l'ordre suivant : haut, droite, bas, gauche

définition individuelle :

- margin-top : 5px;
- margin-bottom : 5px;
- margin-right : 5px;
- margin-left : 5px;

Utilisation des marges

Centrer horizontalement

- center un élément de type block

```
body {
    /* donner une largeur avec width */
    width: 800px;
    /* marge automatique sur les côtés */
    margin 5px auto;
}
```

Occupation de l'espace

- le padding se défini sous les mêmes formes que margin
- un élément occupe l'espace de sa largeur + son padding + sa bordure
- on utilisera box-sizing : border-box; pour la bordure et le padding soient calculés à l'intérieur de l'élément

CSS Reset

- supprime les spécificités des navigateurs
- à placer avant toute définition de style
- Reset CSS

Topic

Déclaration

police(s) à utiliser

font-family: Helvetica, Arial, sans-serif;

- indique l'ordre dans lequel les polices sont recherchées
- recherche si la police est présente sur l'ordinateur de l'utilisateur
- en dernier on précise une famille "générique"

Fonts

- défini dans l'élément body pour tout le document
- 6 familles génériques:
  - serif
  - sans-serif
  - monospace
  - cursive
  - fantasy
  - system-ui

Les polices courantes sur le Web

Police	Type	Desc.
Arial	sans-serif	équivalente Helevetica
Trebuchet	MS	serif moins répandue sur les mobiles
Courier	New	monospace équivalente Courier

Importer une font

Formats	Définition	Desc.
.ttf	TrueTypeFont	le plus répandu,non compressé
.eot	EmbedOpenType	spécifique IE
.woff	WebOpenFormatFont	format compressé adapté au web
.svg	ScalableVectorGraphic	format libre du W3C

```
@font-face {
  font-family: "Open Sans";
  src: url("/fonts/OpenSans-Regular-webfont.woff2") format("w
       url("/fonts/OpenSans-Regular-webfont.woff") format("wof
}
```

Jouer sur la taille

Taille de police

font-size permet de définir la taille de la police (hauteur)

unité px, em, %

Interlignage

line-height

unité px, em % ou sans unité

Mise en valeur

font-weight : bold; mise en gras

font-style : italic; italique

text-align left, right, center, justify

Topic

Démonstration

- `https://jgthms.com/web-design-in-4-minutes/`

Topic

Flexbox

Flexbox

`https://jonibologna.com/content/flexboxsheet.pdf`

`./exemple/flexbox/flextest01.html`

Grid layout

- positionnement en grille
- découpage en ligne et colonne de la page
- déclaration via la déclaration display: grid dans le conteneur principal

les propriétés (1/2)

- grid, inline-grid : déclaration d'un contexte de grille (création d'un "grid-container")
- grid-template-areas : déclaration d'un "canevas" de cellules nommées (optionnel)
- grid-template-rows, grid-template-columns : déclaration des dimensions de lignes et colonnes

les propriétés (2/2)

- grid-row, grid-column : placement d'un élément "grid-item" dans une ligne ou une colonne
- grid-gap, grid-column-gap, grid-row-gap : espaces inter-colonnes ou inter-rangées (gouttières)
- align-items, justify-items : alignement horizontal ou vertical
- align-self, justify-self : alignement horizontal ou vertical d'éléments distincts

Affichage de deux blocs sur une ligne

```
<body>
  <nav>nav</nav>
  <section>section</section>
</body>

body {
  display: grid;
  grid-template-columns: 200px 400px;
}
nav {
  grid-column: 1; /* placement en colonne 1 */
}
section {
  grid-column: 2; /* placement en colonne 2 */
}
```

Grille de 4 emplacements

```
<body>
  <nav>nav</nav>
  <section>section</section>
  <article>article</article>
  <aside>aside</aside>
</body>
body {
  display: grid;
  grid-template-columns: 250px 400px;
  grid-template-rows: 100px 300px;
}
nav { grid-column: 1; grid-row: 1; }
section { grid-column: 2; grid-row: 1; }
article { grid-column: 1; grid-row: 2; }
aside { grid-column: 2; grid-row: 2; }
```

Template

```
#inGrid {
  display: grid;
  grid-template-areas: "h h"
       "n c"
       "f f";
}
nav {
/* placement de <nav> dans l'emplacement "n" */
  grid-area: n;
}
```

Les unités de largeur et hauteur

px, %, em, . . .	unités courantes)
fr	fraction(s) de l'espace restant
min-content	se rapporte à la largeur (ou hauteur) de l'élément le plus petit
max-content	se rapporte à la largeur (ou hauteur) de l'élément le plus grand
minmax(min, max)	exemple minmax(min-content, 20%) correspond à largeur 20% (ou hauteur), mais au minimum largeur (ou hauteur) du contenu
auto	s'adapte à la largeur (ou hauteur) du contenu
fit-content	identique à auto et aussi à minmax(min-content, max-content)

illustration de l'unité "fr"

```
body {
  display: grid;
  /* 250px + "largeur restante" */
  grid-template-columns: 250px 1fr;
  /* 100px + "hauteur restante" */
  grid-template-rows: 100px 1fr;
}
nav { grid-column: 1; grid-row: 1; }
section { grid-column: 2; grid-row: 1; }
article { grid-column: 1; grid-row: 2; }
aside { grid-column: 2; grid-row: 2; }
```

Centrer les éléments 1/3

- par défaut, les éléments "grid-items" s'étirent pour occuper tout l'espace de leur cellule.
- Grid Layout permet également d'aligner les contenus verticalement ou horizontalement

Centrer les éléments 2/3

justify-items	alignement au sein d'une cellule (dans l'axe principal). S'applique au conteneur
justify-self	alignement d'un grid-item au sein de sa cellule. S'applique au grid-item
align-items	alignement au sein d'une cellule (dans l'axe secondaire). S'applique au conteneur
align-self	alignement d'un grid-item au sein de sa cellule. S'applique au grid-item

Centrer les éléments 3/3

Les valeurs de ces propriétés peuvent être les suivantes :

start	aligne l'élément au début de la cellule (gauche ou droite selon le sens de la lecture)
end	aligne l'élément à la fin de la cellule (gauche ou droite selon le sens de la lecture)
center	place l'élément au centre de la cellule
stretch	étire l'élément (ses marges) pour occuper tout l'espace dans la cellule

Multiples centrages

```
body {
  display: grid;
  grid-template-columns: 200px 200px;
  grid-template-rows: 200px 200px;
  grid-template-areas: "a b" "c d";
}
header { grid-area: a; justify-self: center; }
nav { grid-area: b; align-self: center;
}
article {
  grid-area: c;
  justify-self: center;
  align-self: center;
}
footer { grid-area: d; }
```

Occuper plusieurs emplacements

- donner la possibilité à un élément de s'étaler sur plusieurs emplacements, à la fois horizontalement et verticalement
- / indique le début et la fin de l'emplacement
- span indique le nombre de ligne ou colonne à occuper

```
.container {
  display: grid;
  grid-template-columns: 10em 1fr;
  grid-template-rows: min-content 1fr min-content;
  height: 300px;
}
header { grid-column: 1/span 2; grid-row: 1; }
nav { grid-column: 1; grid-row: 2; }
article { grid-column: 2; grid-row: 2; }
footer { grid-column: 1/span 2;  grid-row: 3; }
```

Les motifs de répétition (patterns)

- motifs de répétition (patterns) de colonnes ou de lignes à l'aide de la notation repeat()
- exemple, répéter le motif de colonnes (50px 1em) dix fois dans la grille : grid-template-columns: repeat(10, 50px 1em)

Patterns

```
.container {
  display: grid;
  grid-template-columns: 10px repeat(4, 1fr 10px);
}
header {
  grid-column: 2 ;
}
nav {
  grid-column: 4;
}
article {
  grid-column: 6;
}
footer { grid-column: 8; }
```

Répartition automatique

- par défaut les éléments grid-items se répartissent automatiquement dans le premier emplacement disponible au sein de la grille, dans le sens horizontal.
- il est possible de modifier ce comportement à l'aide de la propriété grid-auto-flow

  grid-auto-flow : row répartition automatique rangée par rangée (valeur par défaut)

  grid-auto-flow : column répartition automatique colonne par colonne

grid-auto-flow: column

```
.container {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-rows: 100px 100px 100px;
  grid-auto-flow: column;
}
```

Utilisation conjointe, grid et flex

https://rachelandrew.co.uk/css/cheatsheets/box-alignment

Utilisation conjointe, grid et flex

CSS Grid :

- est idéal pour construire l'image plus grande.
- permet de gérer facilement la disposition de la page entière
- pour les mises en page 2D (lignes et colonnes

FlexBox :

- est idéal pour aligner le contenu dans des éléments.
- permet de positionner des petits détails d'une conception
- mise en page une dimension (lignes ou colonnes)

`./grid-and-flex/index.html`