3IN017 – TECHNOLOGIES DU WEB

CSS

21 janvier 2025

Gilles Chagnon

Plan

Origines et principes

Origine

Premiers temps du Web : mélange fond/forme, par exemple...

```
<body bgcolor="yellow">
        <bgsound src="mouettes.wav" loop="infinite">
        <center><h1 font="Comic Sans MS">Grand titre</h1></center>
        <p><font color="blue">Un texte bleu</font><p>
(...)
</body>
```

Nombreux problèmes :

- Maintenabilité et passage à l'échelle
- Limitations artistiques
- Adaptabilité aux outils de consultation
- Accessibilité

La solution : les *Cascading StyleSheets* – CSS

D'un côté le HTML structuré, de l'autre sa mise en forme par CSS.

- permet de entraliser les informations de style propres à un site, et d'en assurer la cohérence visuelle
- rend possible la séparation du fond et de la forme (voir par exemple le site CSS Zen Garden `https://www.csszengarden.com/`)

Dernière version à date, CSS Niveau 3, 2023
`https://www.w3.org/TR/css-2023/`

Insérer un style

- Attribut style :
  `<p style="...">(...)</p>`
- Feuille de style interne, dans le `head` de la page :
  `<style>(...)</style>`
- Feuille de style externe, dans le `head` de la page :
  `<link rel="stylesheet" href="blabla.css">`

Ordre de priorité : style>interne>externe

Syntaxe et sélecteurs

Syntaxe de base

SCIENCES SORBONNE UNIVERSITÉ

```
Sélecteur {
propriété1: valeur;
propriété2: valeur;
}
```

Exemples :

```
p {
        color:blue;
}
cite {
        font-variant:small-caps;
        }
.auteur, #editeur {
        font-style:italic
}
article p:forst-of-type::first-letter {
        font-size: 130%
}
```

Avec attribut HTML `style` : `style="color:blue"`

Attributs HTML `id` et `class`

SCIENCES SORBONNE UNIVERSITÉ

`id`
- identifiant unique dans le document
- utile pour le HTML lui-même (labels, liens internes) et la sélection *via* CSS

class
- assignable à plusieurs éléments
- un élément peut avoir plusieurs classes
- utile seulement pour la sélection *via* CSS

Exemples :

```
<div id="connectedUsers">
        <h3>Utilisateurs connectés</h3>
        <ul>
                <li class="userName">Alice</li>
                <li class="userName admin">Bob</li>
                <li class="userName">Charlie</li>
                <li class="userName">Daisy</li>
        </ul>
</div>
```

Sélecteurs principaux

SCIENCES SORBONNE UNIVERSITÉ

Titre	Syntaxe	Exemple
Universel	*	*
Type d'élément	<tag>	a
Identifiant	#<id>	#searchInput
Classe	.<class>	.message
Élément avec classe	<tag>.<class>	span.date
Descendant	<ancestor> <child>	.message img
Enfant direct	<ancestor> > <child>	.message > img
Même parent	<child1> - <child2>	table - h3
Suivant	<child1> + <child2>	.message + separator
Attribut	[<attr><operator><value>]	span[lang=en]

Voir la référence sur le MDN :
`https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_selectors`

Sélecteurs : pseudo-classes (:)

SCIENCES SORBONNE UNIVERSITÉ

Ciblage d'un élément en fonction de son état

- État dans le DOM : `:first-child`, `:first-of-type`, `:nth-child`... Par exemple `tr:nth-child(2n): {background-color: gray}`
- États d'un lien : `:hover`, `:active`, `:visited`
- États de champs de formulaires : `:focus`, `:disabled`, `:invalid`, `:checked`... (voir `https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes#input_pseudo-classes`)
- Également opérateur `:not`

Sélecteurs : pseudo-éléments (: :)

SCIENCES SORBONNE UNIVERSITÉ

Ciblage d'une sous-partie d'un élément

- `::first-letter`, `::first-line`

```
p::first-letter{
        font-size: 1.5em;
        margin-left: 1em
}
```

- `::before`, `::after` : injection de contenu (attention à l'accessibilité)

```
p:last-of-type::after{
        content: "The end";
        font-style: italic;
        display: block;
        text-align: right
}
```

Couleurs et unités

Couleurs

Deux principales manières :

- Par le nom de la couleur. 16 couleurs de base : aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white et yellow. Environ 150 couleurs supplémentaires consultables sur `https://developer.mozilla.org/en-US/docs/Web/CSS/named-color`
- Par son « code RGB » : #RRGGBB (niveaux de rouge, vert et bleu entre 0 et 255 codés en hexadécimal).

Il en existe d'autres (voir `https://developer.mozilla.org/fr/docs/Web/CSS/color_value`)

Unités

Les valeurs de taille peuvent s'exprimer de très nombreuses façons :

Absolues

- cm, mm, pt, *etc.* : pas recommandées sur écran
- px : normalement relative car dépend de la taille du pixel physique sur l'écran

Relatives

- % : pourcentage (requiert une valeur de référence dans les parents)
- em : largeur d'un M typographique, relatif au `font-size` parent
- ex : hauteur d'un x typographique, relatif au `font-size` parent
- rem (*root em*) : comme `em`, mais relatif au font-size de l'élément racine
- vw / vh / vmin / vmax : pourcentages de hauteur/largeur d'affichage (*viewport*)

Propriétés

Aperçu

Le standard définit :

- quelles propriétés sont permises pour quel élément
- quels sont les types et valeurs possibles pour les propriétés
- quelles propriétés sont héritées (exemple : `font`) ou non (exemple : `border`)

Tous les aspects visuels sont couverts :

- Typographie (`font-size`, `font-style`, `font-family`, `text-transform`, `text-decoration`...)
- Couleurs, images, rendu (`color`, `background-color`, `background-image`, `filter`...)
- Marges internes/externes, bordures (`padding`, `margin`, `border`...) : voir plus loin
- Dimensions (`width`/`height`, `min-width` / `max-height`...)
- Positionnement (`display`, `float`, `position`)
- Transformation, animations (`transition`, `animation`...)
- *etc.*

Propriétés raccourcies (1)

Certaines propriétés permettent de déclarer plusieurs variables à la fois. C'est le cas de `font`, `background`, `margin`, `padding`, `border`, `flex`...

```
.example {
        background-color: yellow;
        background-image: url("/assets/tweetie.png");
        background-repeat: no-repeat;
        background-position: top left;
}
/* ... peut s'écrire... */
.example {
        background: yellow url("/assets/tweetie.png") top left no-repeat;
}
```

Propriétés raccourcies (2)

Il est aussi parfois possible de ne pas avoir à répéter certaines dimensions quand il y a des symétries :

```
.example {
        margin-top: 10px;
        margin-right: 5px;
        margin-bottom: 10px;
        margin-left: 5x
}
/* équivaut à */
.example {
        margin: 10px 5px 10px 5px;
}
/* mais aussi à */
.example {
        margin: 10px 5px;
}
```

Mise en page

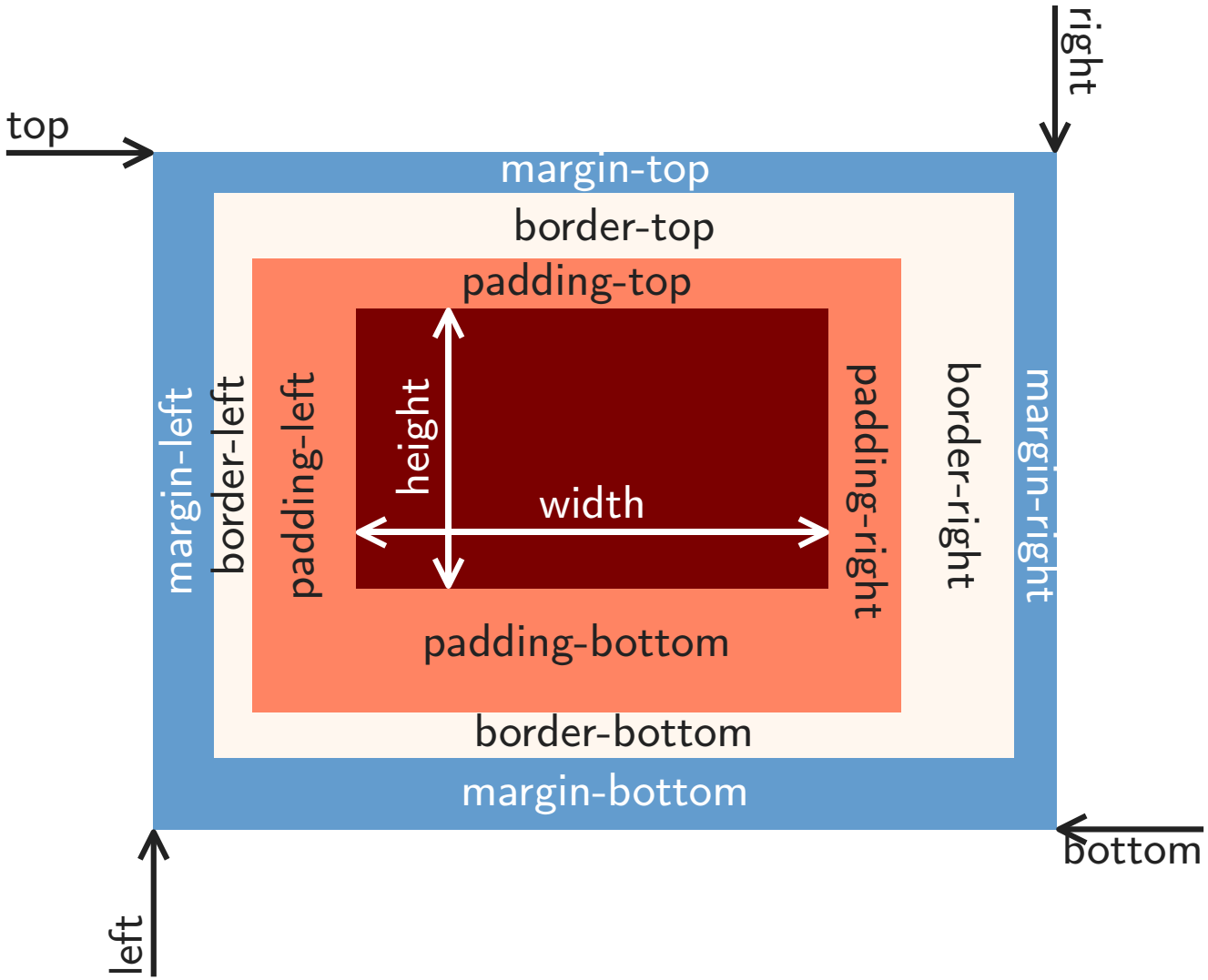
Types *block* et *inline*

Il existe principalement deux grands types d'éléments en HTML/CSS :

- Éléments de type block :
  - occupent 100% de la largeur
  - hauteur définie par son contenu
  - par exemple `p`, `ul`, `article`...
- Éléments de type inline :
  - largeur définie par son contenu
  - rendu dans le flux du document
  - par exemple `span`, `a`, `img`, `input`...

En CSS : propriété `display` (qui peut valoir `inline-block`, `table-column`...)
voir `https://developer.mozilla.org/fr/docs/Web/CSS/display`)

Modèle de boîte CSS



Original : Günther M. Apsel Vecteur : Valtlai, CC0, via Wikimedia Commons

Layout

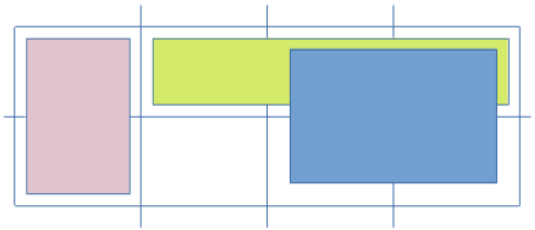
Maintenant, des solutions matures pour gérer des arrangements complexes :

Flexbox

`display: flex` :

- arrangements sur un axe
- centrages et alignements
- designs fluides

Grid



`display: grid` :

- arrangements assimilables à une grille
- facilement réorganisable
- superposition possible

- *A complete guide to flexbox* :
  `https://css-tricks.com/snippets/css/a-guide-to-flexbox/`
- *A complete guide to grid* :
  `https://css-tricks.com/snippets/css/complete-guide-grid/`
- Flexbox ou grid ($\approx$ 4 minutes) :
  `https://www.youtube.com/watch?v=hs3piaN4b5I`

Media queries : @media

Application de sélecteurs selon les caractéristiques et fonctionnalités du client :

```
@media print {
        video {
                display: none;
        }
}
@media (prefers-color-scheme: dark) {
        body {
                background-color: black;
                color: white;
        }
}
@media (prefers-reduced-motion) { ... }
@media (pointer: none) { ... }
```

Responsive Web design

Une des applications les plus répandues des *media queries*. Principe :

- servir un même contenu HTML quel que soit l'outil de consultation
- en fonction des caractéristiques du navigateur, adapter le contenu (déplacer/redimensionner/masquer/...) via CSS
- adaptable à largeur/hauteur d'écran, mais aussi orientation (portrait/paysage), finesse d'affichage, *etc.*

```
body {
        font-size: 14px;
}
/* "breakpoint" */
@media (min-width: 600px) {
        body {
                font-size: 16px;
        }
}

@media (min-width: 1000px) {
        body {
                margin: 0 auto;
                max-width: 800px;
        }
}
```

Difficultés et pratiques

Problématique de CSS sur les projets d'envergure

SCIENCES SORBONNE UNIVERSITÉ

- **Standard très vivant, en évolution constante**
  - En pratique, dépend du support par les navigateurs (voir `https://caniuse.com/`)
  - Comment continuer à supporter les vieilles versions ?
- **Complexité / performance**
  - Comment s'adapter à tous les supports ?
  - Comment ne pas envoyer du style inutile ?
  - Comment garantir la cohérence à l'échelle du projet entier ?
- **Conflits de règles**
  - Pratiques de namespacing des sélecteurs (OOCSS, BEM, etc. . .)

Pour les gros projets, il est rare d'écrire du CSS brut et *from scratch*.

Pratiques et outils actuels

- Préprocesseurs ou transpileurs (ex : Sass, PostCSS)
  - Surcouche au langage, transformé en vrai CSS pour le navigateur
  - Support syntaxique plus large (ex : boucles, nesting de sélecteurs)
  - Application de fonctions et automatisation à la compilation (ex : préfixage de propriétés)
- *Frameworks* et *design systems* clef-en-main
  - Ex : Bootstrap, ensemble de classes utilitaires de haut niveau `<button class="btn btn-lg btn-primary">`
  - Ex : Tailwind, Tachyons, ensemble de classes utilitaires "atomiques" + compilation

    ```
    <button class="m-10 bg-red-300 text-white px-4 text-xs shadow-lg">
    ```
- Lien avec le code JavaScript, approches "CSS-in-JS" (ex : Styled components, Emotion)
  - *namespacing* automatique *via* le code JavaScript

    ```
    <button class="sc-bkzYnD sc-dmlqKv lgOOoH gNtXWi">
    ```

Conclusion

À retenir

Ce cours est une introduction très très rapide !

HTML et CSS sont trop complexes pour tout connaître d'un coup (les subtilités de la notion de cascade CSS, les « variables » CSS...)

Plus important :

- Connaître les grands principes des langages
- Être sensible aux bonnes pratiques (standards, accessibilité, *etc.*)
- Savoir chercher l'information
- Se tenir un minimum au courant des évolutions
- Technologie toujours « bidouillable » $\Rightarrow$ bricoler !