Web

CSS

Bruno BEAUFILS

2024/2025

Rappels sur la création documentaire

- 2 aspects importants
  - **fond** : informations, données
  - **forme** : présentation

Web conçu pour être un système **universel** d'échange de documents

- 2 outils pour décrire les documents
  - HTML : description des informations *fond*
  - **CSS** : description de la mise en forme *forme*
- 1 langage de programmation pour manipuler le tout
  - Javascript

Rappels sur la création documentaire

- 2 aspects importants
  - **fond** : informations, données
  - **forme** : présentation

Web conçu pour être un système **universel** d'échange de documents

- 2 outils pour décrire les documents
  - HTML : description des informations *fond*
  - **CSS** : description de la mise en forme *forme*
- 1 langage de programmation pour manipuler le tout
  - Javascript

Montrer des données

CSS

Cascading Style Sheet = Feuilles de styles en cascade

Langage de description de mise en forme

- description de la **mise forme** pas du contenu (fond)
  1. choisir des noeuds de l'arbre HTML — **repérer du contenu**
  2. définir comment montrer le contenu de ces noeuds — définir un style
- intérêt
  - structuration avec séparation fond/forme des pages web
    - sortir la forme des documents
  - permet de définir une *charte* applicable à toutes les pages d'un site
    - **CSS Zengarden**

Règles

- liste des règles pour
  1. **sélectionner des noeuds dans l'arbre HTML**
  2. fixer les *propriétés* gérées par le logiciel de présentation — *moteur de rendu*
     - application de la mise en forme sur les noeuds sélectionnés
     - différenciation des media de présentation
- combinaison des propriétés de différents sources (règles)
  - choix via cascade, héritage et spécialisation

Montrer des données

CSS

Cascading Style Sheet = Feuilles de styles en cascade

Langage de description de mise en forme

- description de la **mise forme** pas du contenu (fond)
  1. choisir des noeuds de l'arbre HTML — **repérer du contenu**
  2. définir comment montrer le contenu de ces noeuds — définir un style
- intérêt
  - structuration avec séparation fond/forme des pages web
    - sortir la forme des documents
  - permet de définir une *charte* applicable à toutes les pages d'un site
    - **CSS Zengarden**

Règles

- liste des règles pour
  1. **sélectionner des noeuds dans l'arbre HTML**
  2. fixer les *propriétés* gérées par le logiciel de présentation — *moteur de rendu*
     - application de la mise en forme sur les noeuds sélectionnés
     - différenciation des media de présentation
- combinaison des propriétés de différents sources (règles)
  - choix via cascade, héritage et spécialisation

Montrer des données

CSS

Cascading Style Sheet = Feuilles de styles en cascade

Langage de description de mise en forme

- description de la **mise forme** pas du contenu (fond)
  1. choisir des noeuds de l'arbre HTML — **repérer du contenu**
  2. définir comment montrer le contenu de ces noeuds — définir un style
- intérêt
  - structuration avec séparation fond/forme des pages web
    - sortir la forme des documents
  - permet de définir une *charte* applicable à toutes les pages d'un site
    - **CSS Zengarden**

Règles

- liste des règles pour
  1. **sélectionner des noeuds dans l'arbre HTML**
  2. fixer les *propriétés* gérées par le logiciel de présentation — *moteur de rendu*
     - application de la mise en forme sur les noeuds sélectionnés
     - différenciation des media de présentation
- combinaison des propriétés de différents sources (règles)
  - choix via cascade, héritage et spécialisation

Liaison des styles avec HTML

3 manières différentes

1. via une liaison vers un fichier dans l'entête (`<head>`) du fichier HTML

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

utilisation d'un élément vide `<link/>` établissant une **relation** entre fichiers

exemple : src/gotlib-css-link.html

2. via un élément `<style>` dans l'entête (`<head>`) du fichier HTML

```
<style>
 REGLES
</style>
```

exemple : src/gotlib-css-element-style.html

3. via l'attribut `style` d'un élément HTML (pas de sélecteur)

```
<balise style="DECLARATIONS"> ... </balise>
```

exemple : src/gotlib-css-attribut-style.html

La « **bonne** » manière est la première

Liaison des styles avec HTML

3 manières différentes

1. via une liaison vers un fichier dans l'entête (`<head>`) du fichier HTML

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

utilisation d'un élément vide `<link/>` établissant une **relation** entre fichiers

exemple : src/gotlib-css-link.html

2. via un élément `<style>` dans l'entête (`<head>`) du fichier HTML

```
<style>
 REGLES
</style>
```

exemple : src/gotlib-css-element-style.html

3. via l'attribut `style` d'un élément HTML (pas de sélecteur)

```
<balise style="DECLARATIONS"> ... </balise>
```

exemple : src/gotlib-css-attribut-style.html

La « **bonne** » manière est la première

Liaison des styles avec HTML

3 manières différentes

1. via une liaison vers un fichier dans l'entête (`<head>`) du fichier HTML

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

utilisation d'un élément vide `<link/>` établissant une **relation** entre fichiers

exemple : src/gotlib-css-link.html

2. via un élément `<style>` dans l'entête (`<head>`) du fichier HTML

```
<style>
 REGLES
</style>
```

exemple : src/gotlib-css-element-style.html

3. via l'attribut `style` d'un élément HTML (pas de sélecteur)

```
<balise style="DECLARATIONS"> ... </balise>
```

exemple : src/gotlib-css-attribut-style.html

La « **bonne** » manière est la première

Liaison des styles avec HTML

3 manières différentes

1. via une liaison vers un fichier dans l'entête (`<head>`) du fichier HTML

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

utilisation d'un élément vide `<link/>` établissant une **relation** entre fichiers

exemple : src/gotlib-css-link.html

2. via un élément `<style>` dans l'entête (`<head>`) du fichier HTML

```
<style>
 REGLES
</style>
```

exemple : src/gotlib-css-element-style.html

3. via l'attribut `style` d'un élément HTML (pas de sélecteur)

```
<balise style="DECLARATIONS"> ... </balise>
```

exemple : src/gotlib-css-attribut-style.html

La « **bonne** » manière est la première

Un (autre) langage textuel simple

- Représentation des règles par du **texte simple**
  - même outils nécessaires que pour HTML : **éditeur de textes**
- Contenu
  - un ensemble **ordonné** de **règles**
  - groupement possible des règles en fonction du medium de **rendu**
- 2 syntaxes différentes en une seule
  - une pour choisir les noeuds — ***sélecteur***
  - une pour décrire la forme — ***style***

Exercice : un premier style CSS

1. Copiez le fichier `premier.html` en `second.html`
2. Regardez votre fichier avec Firefox
3. Créez le fichier `second.css` avec le contenu suivant

```css
body {
  background: red;
  color: white;
}

p {
  margin: 2em;
  font-family: sans-serif;
  background: blue;
}
```

4. Ajoutez dans l'entête (`<head>`) de `second.html` la ligne

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

5. Regardez votre fichier avec Firefox

Composants syntaxiques

- **Règle**
  - le choix des noeuds
    - la définition d'un style

```
selecteurs { declarations }
```

- **Déclaration**
  - la définition d'une partie de style

```
propriete : valeur ;
```

- **Commentaire**

```
/* Ceci est commentaire sur une règle de style */
```

- Exemple

```
p { background: red; }
```

  - `p` est le sélecteur *(tous les éléments de type `p`)*
  - `background` est la propriété *(couleur de l'arrière-plan)*
  - `red` est la valeur *(rouge)*

Composants syntaxiques

- **Règle**
  - le choix des noeuds
    - la définition d'un style

```
selecteurs { declarations }
```

- **Déclaration**
  - la définition d'une partie de style

```
propriete : valeur ;
```

- **Commentaire**

```
/* Ceci est commentaire sur une règle de style */
```

- Exemple

```
p { background: red; }
```

  - `p` est le sélecteur *(tous les éléments de type `p`)*
  - `background` est la propriété *(couleur de l'arrière-plan)*
  - `red` est la valeur *(rouge)*

Composants syntaxiques

- **Règle**
  - le choix des noeuds
    - la définition d'un style

```
selecteurs { declarations }
```

- **Déclaration**
  - la définition d'une partie de style

```
propriete : valeur ;
```

- **Commentaire**

```
/* Ceci est commentaire sur une règle de style */
```

- Exemple

```
p { background: red; }
```

  - `p` est le sélecteur *(tous les éléments de type `p`)*
  - `background` est la propriété *(couleur de l'arrière-plan)*
  - `red` est la valeur *(rouge)*

Composants syntaxiques

- **Règle**
  - le choix des noeuds
    - la définition d'un style

```
selecteurs { declarations }
```

- **Déclaration**
  - la définition d'une partie de style

```
propriete : valeur ;
```

- **Commentaire**

```
/* Ceci est commentaire sur une règle de style */
```

- Exemple

```
p { background: red; }
```

  - `p` est le sélecteur *(tous les éléments de type `p`)*
  - `background` est la propriété *(couleur de l'arrière-plan)*
  - `red` est la valeur *(rouge)*

Sélecteurs

- Syntaxe particulière
  - permet de préciser comment choisir les noeuds sur lesquels appliquer un style
  - **sélectionne** des noeuds de l'arbre HTML (**du contenu**)
- Factorisation des styles
  - on peut spécifier plusieurs sélecteurs en les séparant par une **virgule**
    - si `S1` et `S2` sont deux sélecteurs
    - alors `S1 , S2` signifie tous les éléments sélectionnés par `S1` **ou** par `S2`
  - exemple :

```
p, h1 { background: red; }
```

l'arrière-plan doit être rouge pour tous les éléments `p` et `h1`

Sélecteurs (base)

- tous les éléments `*`
- sélection d'éléments par leur **type**
  - tous les exemplaires d'un élément de type `ELT` `ELT`
    - `ELT` doit être défini dans HTML
- sélection d'éléments par leurs **attributs**
  - ceux avec un attribut **monattribut** `[monattribut]`
    - **monattribut** doit être défini dans HTML
    - valeur de l'attribut quelconque
    - exemple :

```
<p monattribut="XXX">
```

  - ceux avec un attribut **monattribut** valant **mavaleur** `[monattribut=mavaleur]`
    - **monattribut** doit être défini dans HTML
    - exemple :

```
<p monattribut="mavaleur">
```

Sélecteurs (base)

- tous les éléments `*`
- sélection d'éléments par leur **type**
  - tous les exemplaires d'un élément de type `ELT` `ELT`
    - `ELT` doit être défini dans HTML
- sélection d'éléments par leurs **attributs**
  - ceux avec un attribut **monattribut** `[monattribut]`
    - **monattribut** doit être défini dans HTML
    - valeur de l'attribut quelconque
    - exemple :

```
<p monattribut="XXX">
```

  - ceux avec un attribut **monattribut** valant **mavaleur** `[monattribut=mavaleur]`
    - **monattribut** doit être défini dans HTML
    - exemple :

```
<p monattribut="mavaleur">
```

Sélecteurs (base)

- tous les éléments `*`
- sélection d'éléments par leur **type**
  - tous les exemplaires d'un élément de type `ELT` `ELT`
    - `ELT` doit être défini dans HTML
- sélection d'éléments par leurs **attributs**
  - ceux avec un attribut **monattribut** `[monattribut]`
    - **monattribut** doit être défini dans HTML
    - valeur de l'attribut quelconque
    - exemple :

```
<p monattribut="XXX">
```

  - ceux avec un attribut **monattribut** valant **mavaleur** `[monattribut=mavaleur]`
    - **monattribut** doit être défini dans HTML
    - exemple :

```
<p monattribut="mavaleur">
```

Sélecteurs (base)

- tous les éléments `*`
- sélection d'éléments par leur **type**
  - tous les exemplaires d'un élément de type `ELT` `ELT`
    - `ELT` doit être défini dans HTML
- sélection d'éléments par leurs **attributs**
  - ceux avec un attribut **monattribut** `[monattribut]`
    - **monattribut** doit être défini dans HTML
    - valeur de l'attribut quelconque
    - exemple :

```
<p monattribut="XXX">
```

  - ceux avec un attribut **monattribut** valant **mavaleur** `[monattribut=mavaleur]`
    - **monattribut** doit être défini dans HTML
    - exemple :

```
<p monattribut="mavaleur">
```

Sélecteurs (attribut particulier)

- tous les éléments ayant la classe `monstyle` `.monstyle`
  - `.monstyle` ≈ `[class=monstyle]`
  - un élément a la classe `monstyle` si
    - il a un attribut `class`...
    - ...dont la valeur **contient** `monstyle`
  - exemple :

```
<p class="monstyle dallas">
```

- l'élément identifié par `monnom` `#monnom`
  - `#monnom` = `[id=monnom]`
  - un élément est identifié par `monnom` si
    - il a un attribut `id`...
    - ...dont la valeur est `monnom`
  - l'identifiant doit être **unique** dans le document
  - exemple :

```
<p id="monnom">
```

Sélecteurs (attribut particulier)

- tous les éléments ayant la classe `monstyle` `.monstyle`
  - `.monstyle` ≈ `[class=monstyle]`
  - un élément a la classe `monstyle` si
    - il a un attribut `class`...
    - ...dont la valeur **contient** `monstyle`
  - exemple :

```
<p class="monstyle dallas">
```

- l'élément identifié par `monnom` `#monnom`
  - `#monnom` = `[id=monnom]`
  - un élément est identifié par `monnom` si
    - il a un attribut `id`...
    - ...dont la valeur est `monnom`
  - l'identifiant doit être **unique** dans le document
  - exemple :

```
<p id="monnom">
```

Sélecteurs (combinaison)

Précision

On accolle **sans espace** des sélecteurs de base

- exemple :
  - les élements sélectionnés par *S* **ayant la classe** `monstyle` `S.monstyle`
  - les élements sélectionnés par *S* **ayant un attribut** `monattr` `S[monattr]`
- cumulable à loisir `S.monstyle[monattribut]`

Filiation

On accolle **avec des espaces** des sélecteurs

- en profondeur
  - les éléments sélectionnés par *S* **contenus quelque part dans** *T* `T S`
    - *descendance* de T
  - les éléments sélectionnés par *S* **contenus directement dans** *T* `T > S`
    - *enfant* de T
- en largeur
  - les éléments sélectionnés par *S* **avec le même parent** *T* `T ~ S`
    - *soeur* ou *frère* de T
  - les éléments sélectionnés par *S* **suivant directement** *T* `T + S`
    - *soeur* ou *frère* **suivant** de T

Sélecteurs (combinaison)

Précision

On accolle **sans espace** des sélecteurs de base

- exemple :
  - les élements sélectionnés par *S* **ayant la classe** `monstyle` `S.monstyle`
  - les élements sélectionnés par *S* **ayant un attribut** `monattr` `S[monattr]`
- cumulable à loisir `S.monstyle[monattribut]`

Filiation

On accolle **avec des espaces** des sélecteurs

- en profondeur
  - les éléments sélectionnés par *S* **contenus quelque part dans** *T* `T S`
    - *descendance* de T
  - les éléments sélectionnés par *S* **contenus directement dans** *T* `T > S`
    - *enfant* de T
- en largeur
  - les éléments sélectionnés par *S* **avec le même parent** *T* `T ~ S`
    - *soeur* ou *frère* de T
  - les éléments sélectionnés par *S* **suivant directement** *T* `T + S`
    - *soeur* ou *frère* **suivant** de T

Pseudo-sélecteurs

- s'accole directement à un autre sélecteur (sans espace)
  - commencent par deux-points (`:`)
- concernent
  - l'état d'un élément — ***pseudo-classes*** *(préfixé par `:`)*
  - partie d'un élément — ***pseudo-éléments*** *(préfixé par `::`)*
- exemple
  - `:hover` — l'élément sous la souris
  - `:visited` — un lien déjà visité
  - `:first-child` — un élément **premier enfant** de son parent
  - `:last-child` — un élément **dernier enfant** de son parent
  - `:nth-child(N)` — un élément **Nième** enfant de son parent
    - `N` peut être `odd`, `even` ou un nombre

Exercice

1. Récupérez le fichier http://m2.ape-cee.fr/src/gotlib-css-exo.html
2. Modifiez vos styles pour appliquer les propriétés suivantes
   - **2.1** à toutes les images la déclaration

```
{ float: right; }
```

   - **2.2** à tous les éléments portant la classe `inverse` la déclaration

```
{ background: black; color: white; }
```

   - **2.3** à l'élément d'identification `nom` la déclaration

```
{ font-size: 200%; }
```

   - **2.4** à tous les éléments portant la classe `rouge` dans un tableau la déclaration

```
{ background: #ff0000; }
```

Exercice (suite)

2.5 à tous les paragraphes, ancres ou cellules de tableau de listes survolés par la souris la déclaration

```
{ background: #00ff00; }
```

1. Principes de CSS

2. Syntaxe CSS

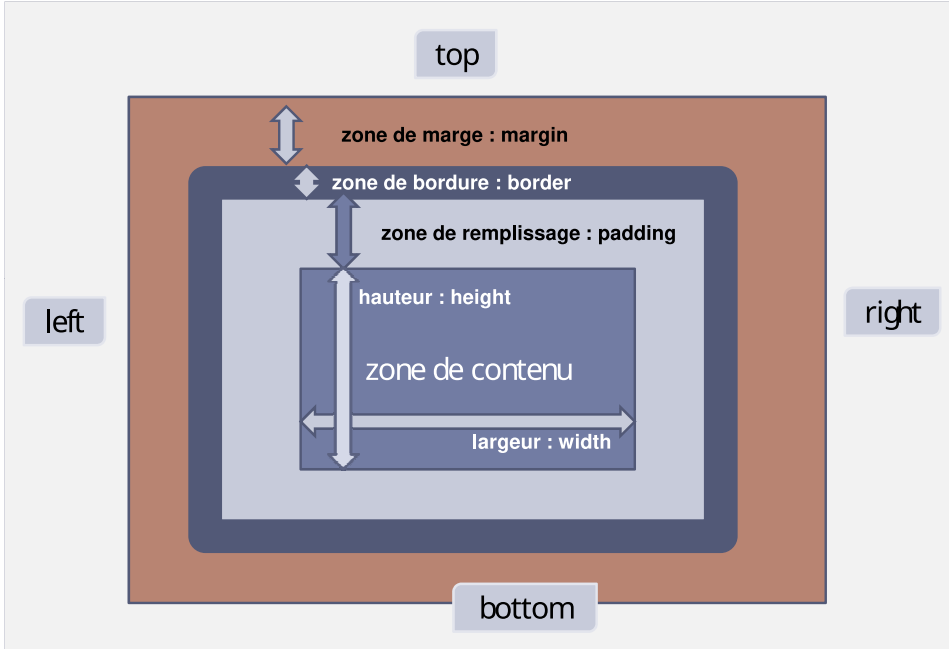
3. Sélection

4. Propriétés

Propriétés pour media graphiques

Disposition de l'affichage

- Modèles de boîtes `box-sizing`



  - marge, bordure, remplissage, contenu
    - `margin-top`, `margin-right`, `margin-bottom`, `margin-left`
    - etc.

- Disposition et mise en page `display`

Propriétés pour media graphiques (suite)

Apparence de l'affichage

- Couleurs — `color`, `background`
- Polices — `font-family`, `font-size`, `font-style`
- Textes — `text-align`, `text-transform`, `text-decoration`

Valeurs

Le domaine des valeurs possibles dépendent de la propriété

- notion de **type**
- se référer à la définition de la propriété

Quelques types

- unités de longueur
  - absolue `px`
  - relative `em`, `%`
- modèle d'affichage
  - en boîte `block`
  - en ligne `inline`
  - pas affiché `none`
  - beaucoup d'autres valeurs
- couleurs
  - prédéfinies `red`, `green`, `blue`, etc.
  - modèle RGB `#rrggbb`
    - `rr`, `gg`, et `bb` valeurs hexadécimales
  - plein de *Color Picker* en ligne :
    - Mozilla Developper Network
    - DuckDuckGo

Références

- Mozilla Developper Network (CSS)
  - **Les bases de CSS**
  - **Les sélecteurs CSS**
  - Introduction à CSS
  - Références CSS
- DevDocs CSS
- Démonstration CSS
  - Beauté de la conception CSS