Web : principes de base (HTML/CSS)

DA1

Henallux

Module 4
Arborescence HTML

Au programme

- **L'arborescence HTML**
  - *Structure d'arbres*
  - *Vocabulaire de la théorie des graphes*
  - *L'arborescence HTML*
- **CSS (approfondissement)**
  - *C = Cascading… pourquoi ?*
  - *Layers CSS et héritage*
- **Sélecteurs avancés**
  - *Combinaison de sélecteurs*
  - *Sélecteurs basés sur l'arborescence*
  - *Sélecteurs basés sur les attributs*
  - *Pseudo-classes*
  - *Pseudo-éléments*

L'arborescence HTML

Au programme de ce chapitre…

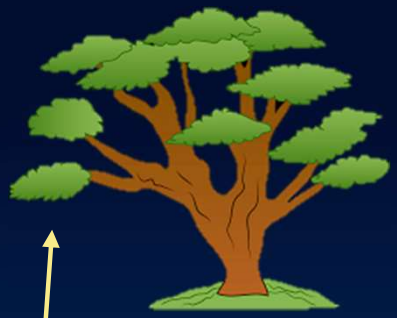
- **Structure d'arbre**
  - *Arbre = notion utilisée dans de nombreux domaines en informatique*
  - *Qu'est-ce qu'un arbre exactement ?*
- **Vocabulaire de la théorie des graphes**
  - *Vocabulaire commun à de nombreuses applications, dont HTML*
- **L'arborescence HTML**
  - *En quoi un document HTML est-il un arbre ?*

Ensuite : *CSS (approfondissement)*

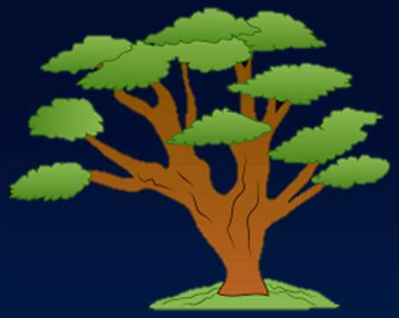
Structure d'arbre

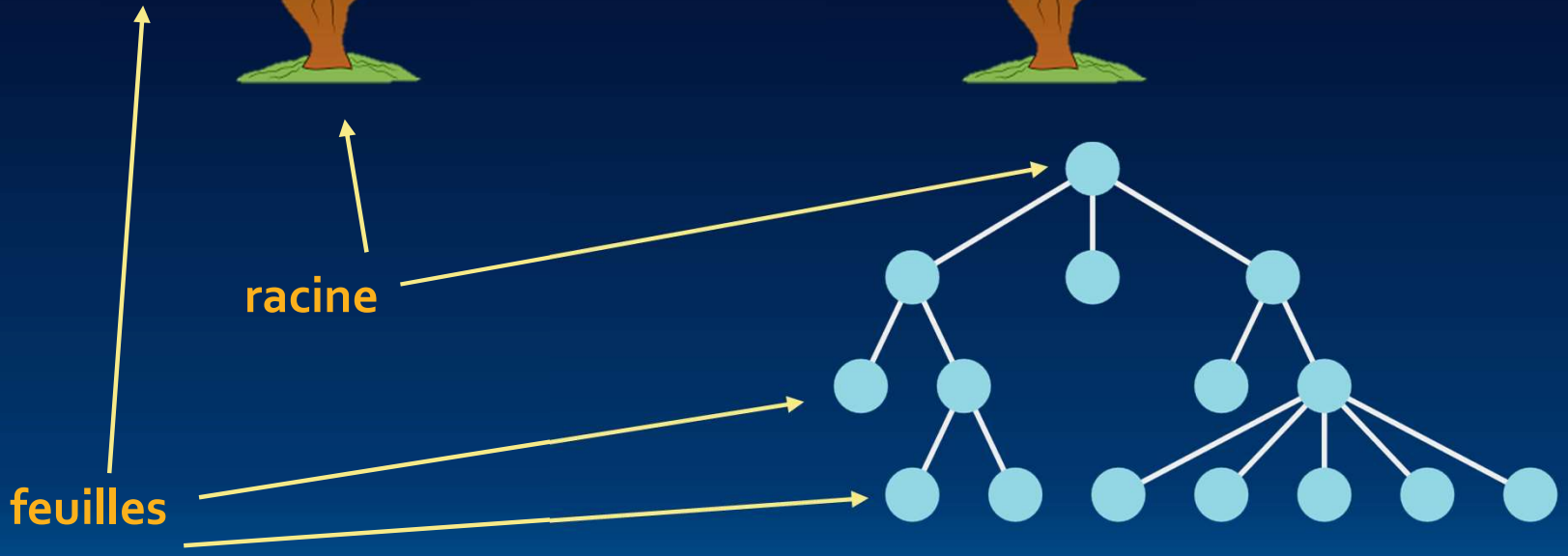
Qu'est-ce qu'un **arbre** ?

Pour la plupart des gens…



En math / info :





Structure d'arbre/arborescence

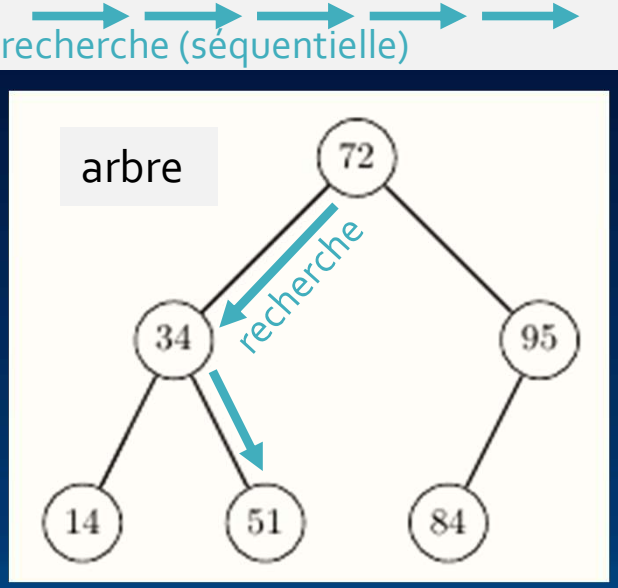
Utilisée dans de nombreux domaines :

- Organisation de données (voir aussi cours d'OED)
  - Arbres binaires de recherche (efficacité des recherches)
  - Arbres à préfixe pour le stockage de mots/séquences

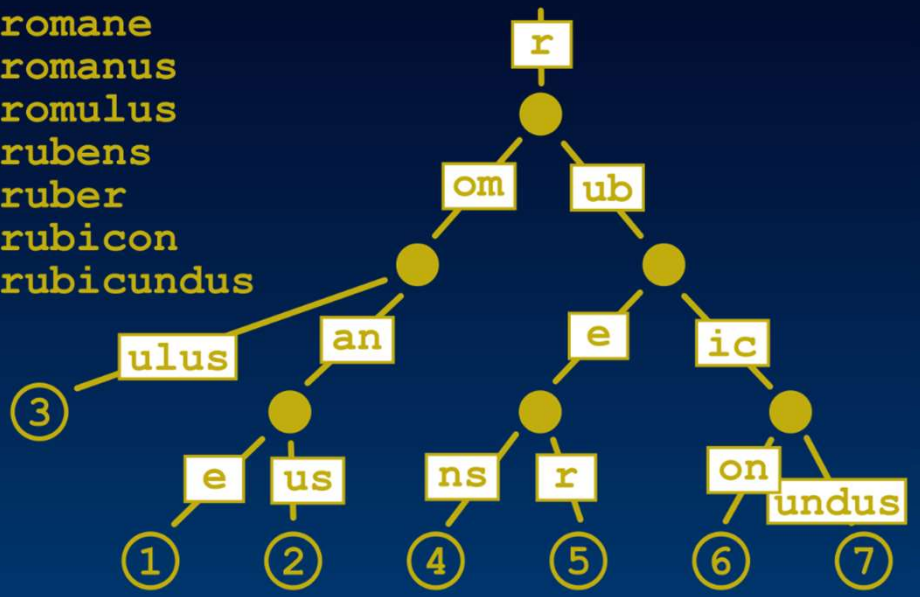
tableau

[14, 34, 51, 72, 84, 95]

recherche (séquentielle)



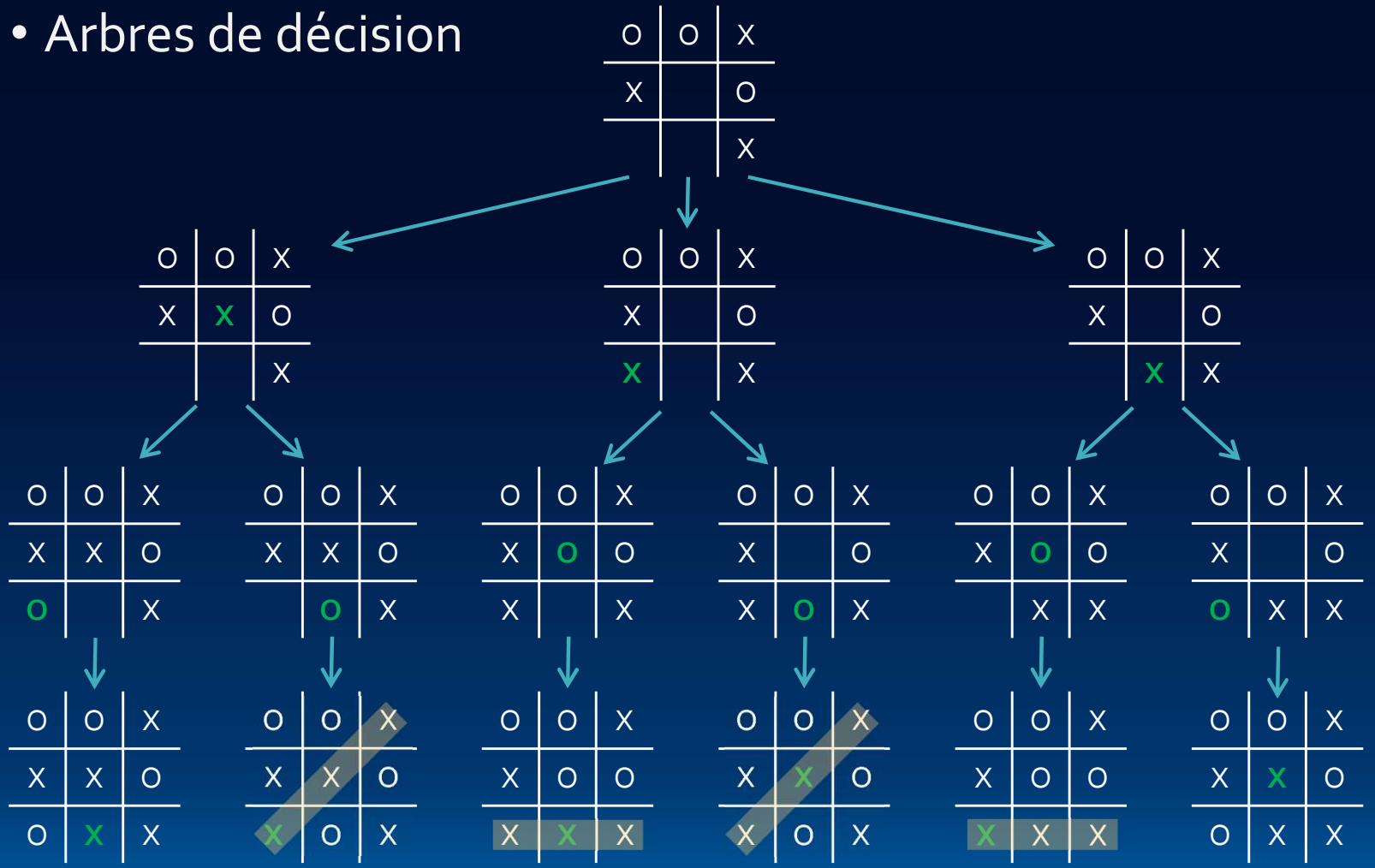
1 romane
2 romanus
3 romulus
4 rubens
5 ruber
6 rubicon
7 rubicundus



Structure d'arbre/arborescence

Utilisée dans de nombreux domaines :

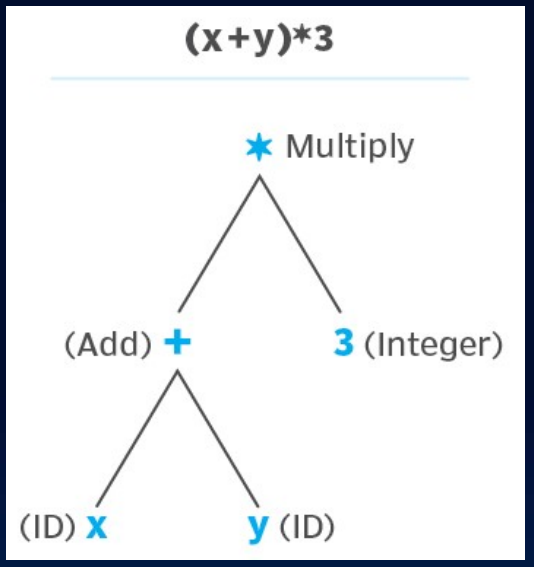
- Arbres de décision

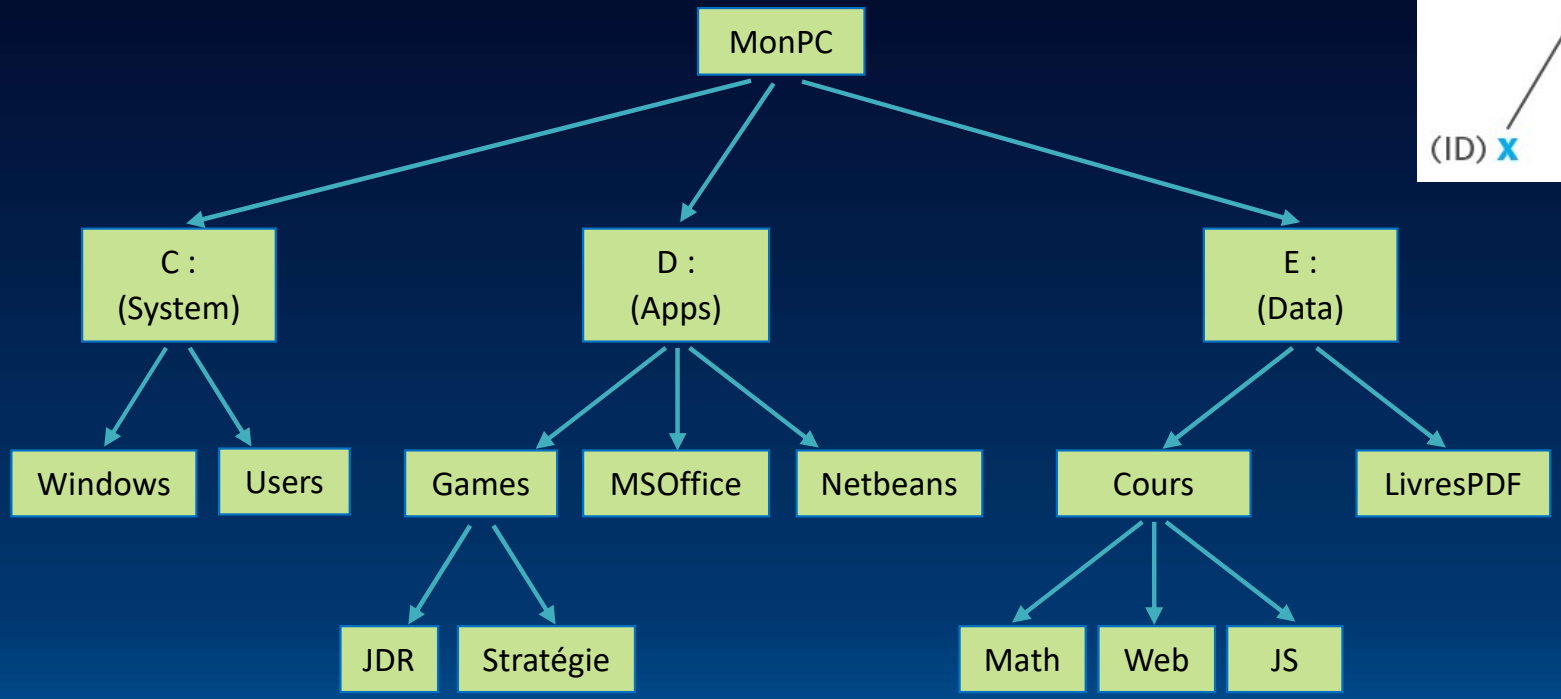


Structure d'arbre/arborescence

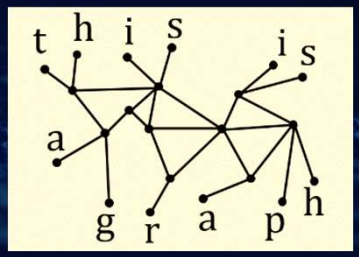
Utilisée dans de nombreux domaines :

- Analyse lexicale d'un code
- Arborescences de dossiers

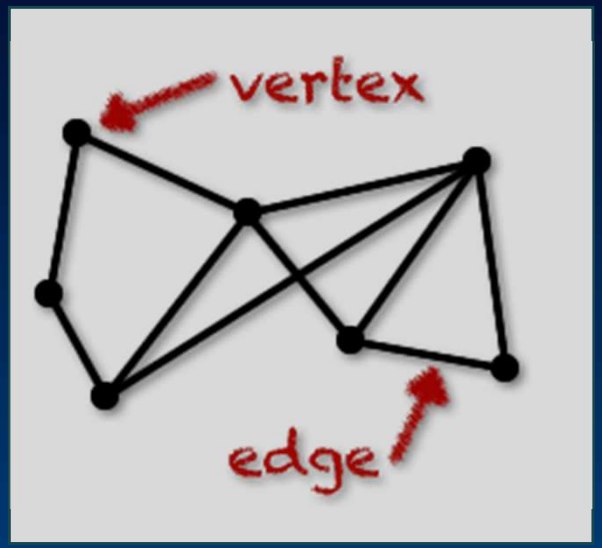


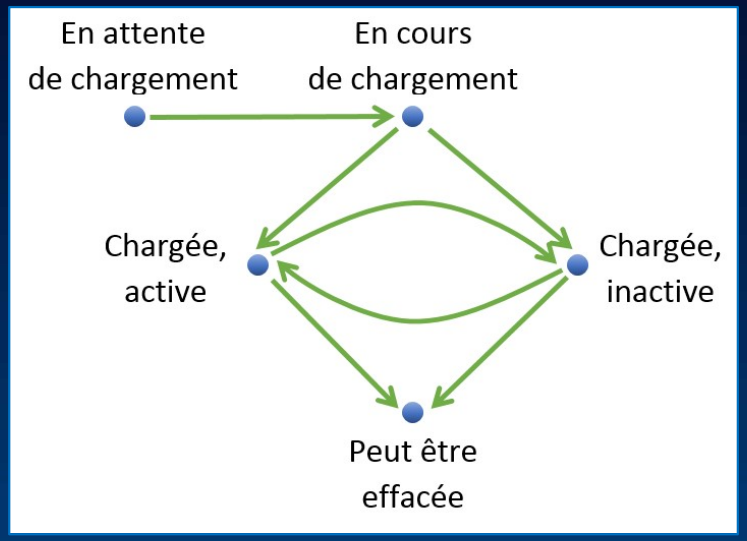


Théorie des graphes



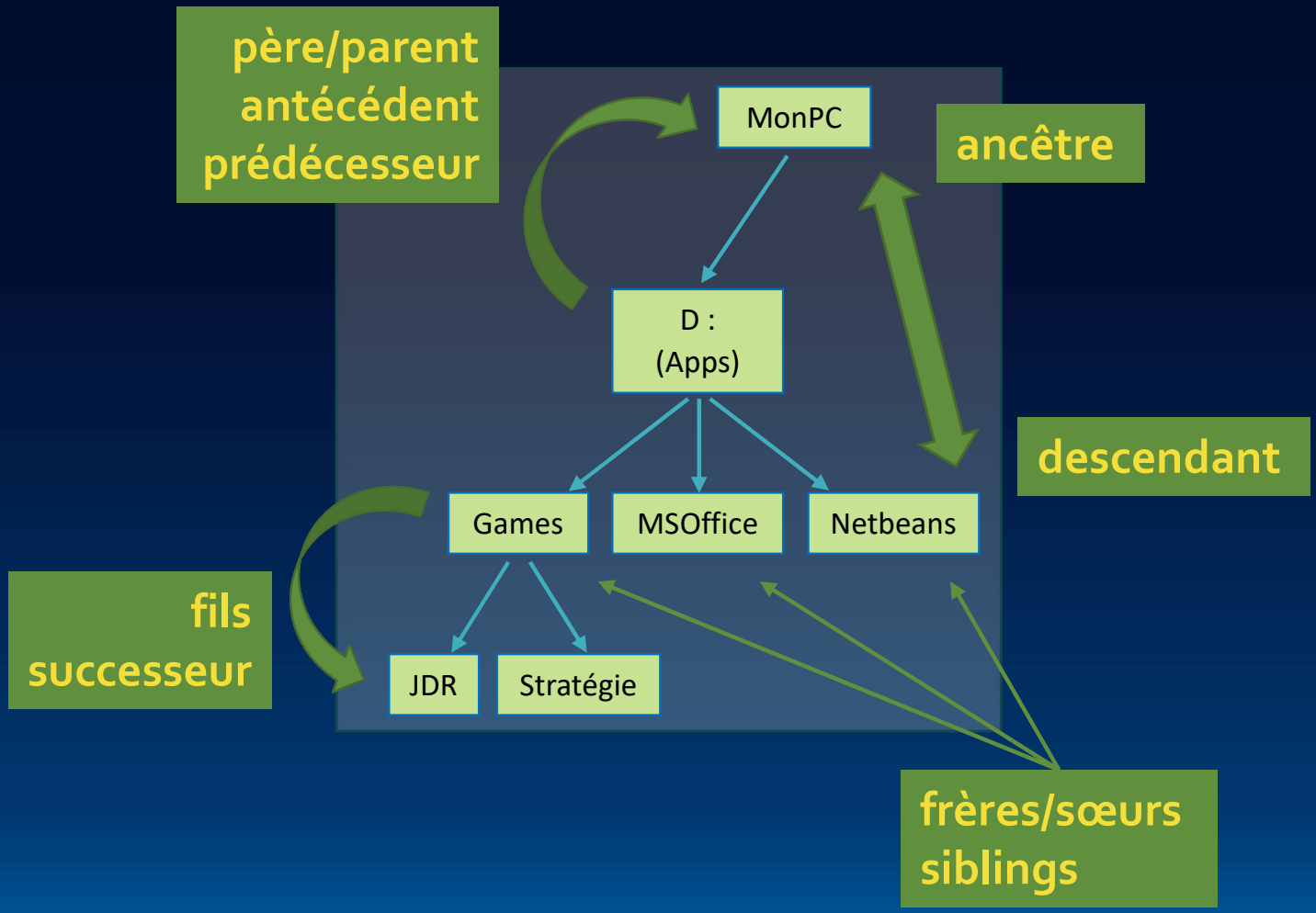
- La notion d'arbre (techniquement, d'*arborescence*) vient de la théorie des graphes.
- **Graphe** : structure composée de
  - **sommets** (vertex/vertices) qui peuvent être reliés par des
  - **arêtes** ou **arcs** (edges ou oriented edges)





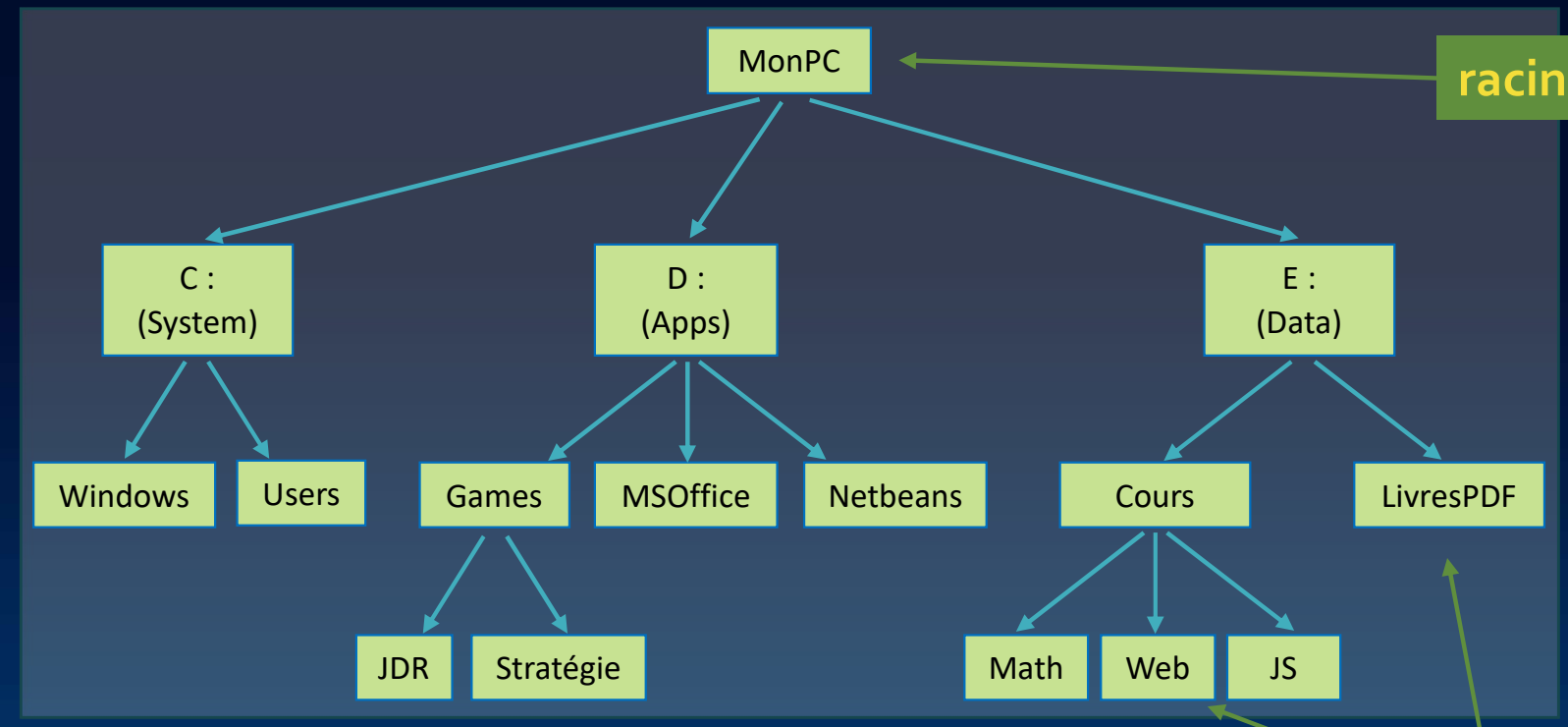
Vocabulaire des arborescences

- Vocabulaire (inspiré des arbres généalogiques)



Vocabulaire des arborescences

- Vocabulaire spécifique aux arborescences



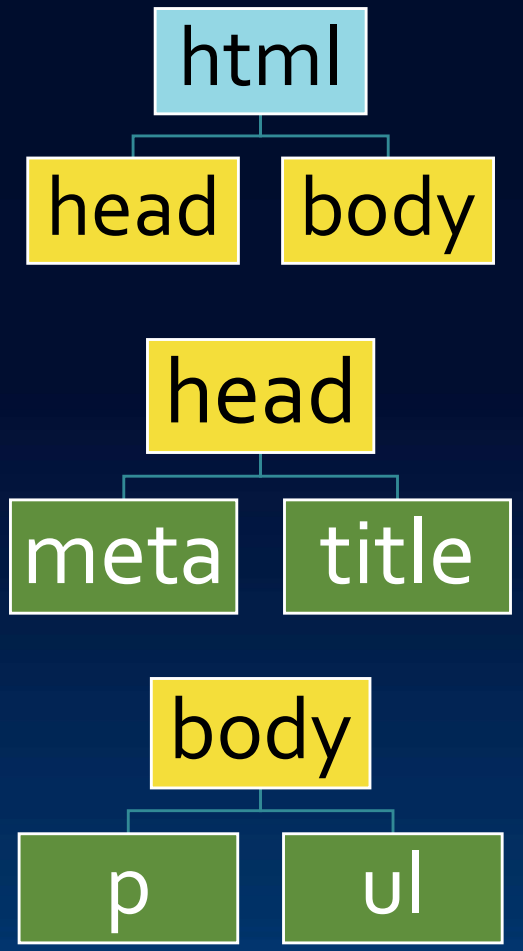
L'arborescence HTML


```
<html>

  <head>
    <meta charset="ISO-8859-1"/>
    <title>Exemple en HTML</title>
  </head>

  <body>
    <p>Un <strong>arbre</strong> possède :</p>
    <ul>
      <li>Une <em>racine</em> et</li>
      <li>des <em>feuilles</em></li>
    </ul>
  </body>

</html>
```

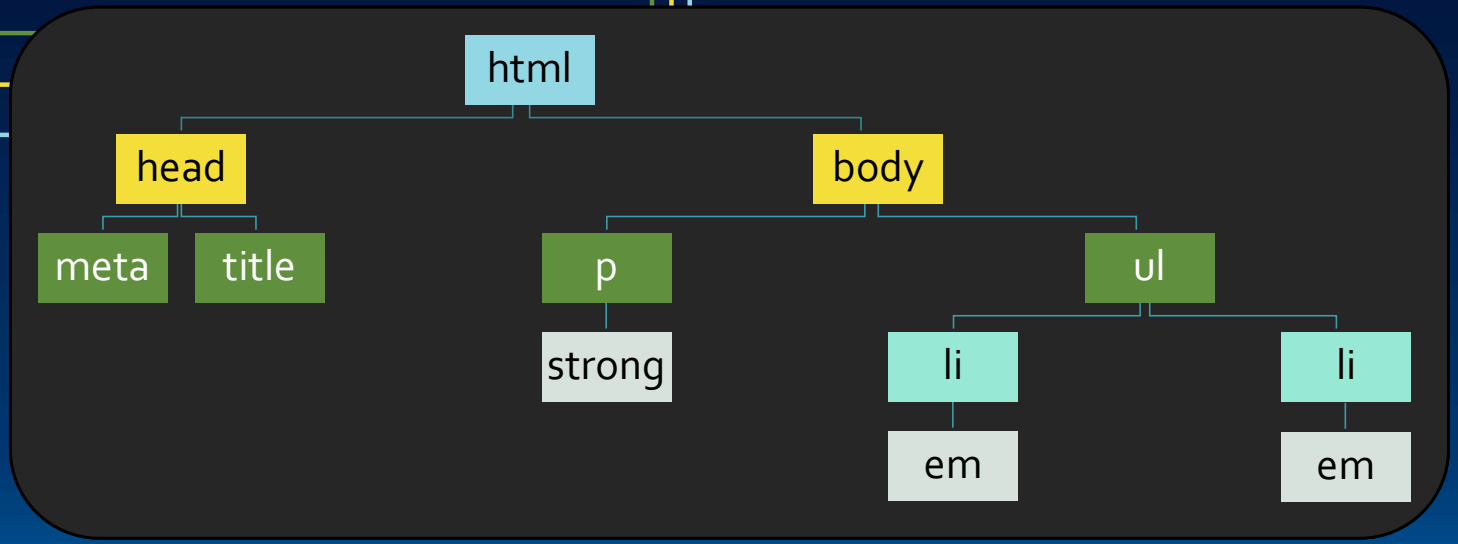


Note : il s'agit d'une version simplifiée, car on ignore les blancs et les textes.

L'arborescence HTML


```
<html>
  <head>
    <meta charset="ISO-8859-1"/>
    <title>Exemple en HTML</title>
  </head>

  <body>
    <p>Un <strong>arbre</strong> possède :</p>
    <ul>
      <li>Une <em>racine</em> et</li>
      <li>des <em>feuilles</em></li>
    </ul>
  </body>
</html>
```



L'arborescence HTML

Vrai/faux ?

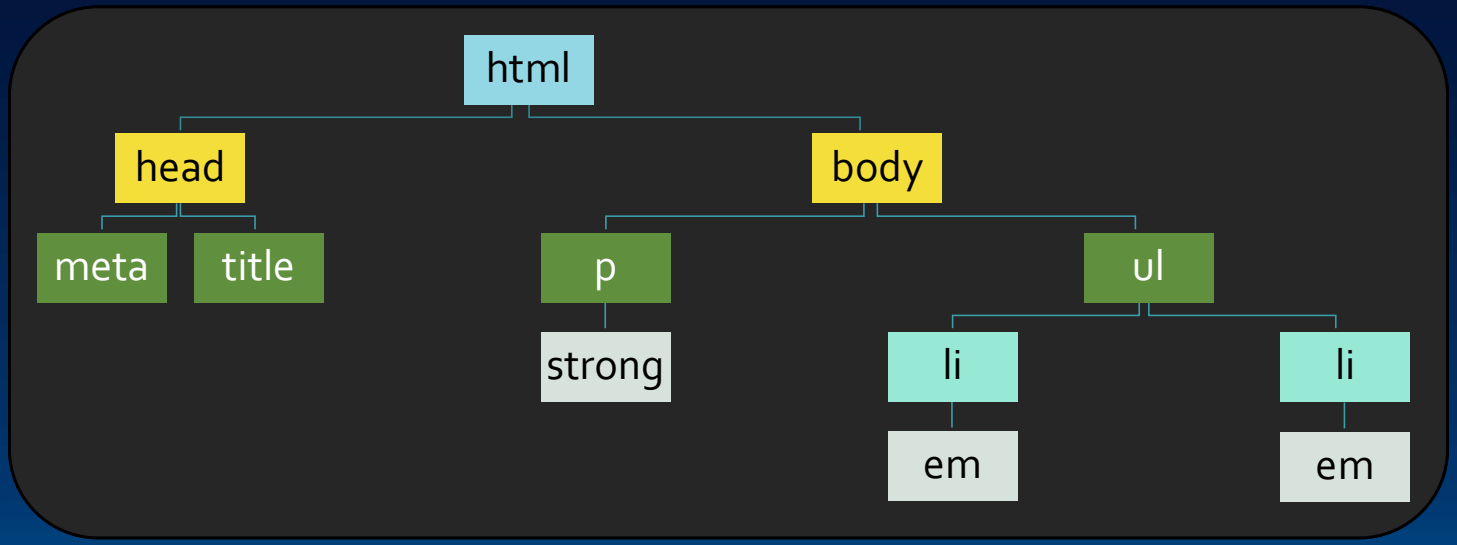
a) Tous les nœuds `<li>` ont un père `<ul>`.

b) Les deux nœuds `<em>` sont des frères/siblings.

c) Le nœud `<p>` est un ancêtre des nœuds `<em>`.

d) Le nœud `<ul>` a quatre fils : deux `<li>` et deux `<em>`.

e) Le nœud `<body>` est le seul ancêtre commun à `<strong>` et aux `<em>`.



CSS (approfondissement)

Au programme de ce chapitre…

- **Cascading Style Sheets**
  - *En quoi est-ce « cascading » ?*
  - *L'héritage selon l'arborescence HTML*
  - *Les layers CSS*

Ensuite : *Sélecteurs avancés*

Cascading Style Sheets

- Deux phénomènes de « **cascade** » :
  - (I) entre les sources/origines (layers) des styles (voir plus loin)
  - (II) au sein de l'arborescence HTML

- (II) Cascade / **héritage** au sein de l'arborescence HTML
  - Certaines propriétés sont héritées de père en fils.
  - Exemple :

Ce *mot* est mis en emphase.

```
p { color:blue }
em { font-decoration:underline }
…
<p>Ce <em>mot</em> est mis
en emphase.</p>
```

Le mot est…	Raison
souligné	Règle CSS sur em
italique	Style par défaut du navigateur
bleu	Hérité de p

Cascading Style Sheets

- (I) Cascade entre les **trois layers** (sources) CSS :
  - Styles du **navigateur** par défaut (le moins important)
    ex : par défaut, les `<em>` *sont en italique…*
  - Styles définis par l'**utilisateur** (via userContent.css)
    ex : … sauf si l'utilisateur du navigateur a défini autre chose …
  - Styles de l'**auteur** (dans la page HTML ou la feuille CSS)
    ex : … et c'est l'auteur de la page web qui a le dernier mot…

Source des illustrations : https://www.stefanjudis.com/

Cascading Style Sheets

- L'annotation `!important` placée après une valeur CSS permet de modifier les priorités déterminées par les layers.

```
p { color: green !important };
```

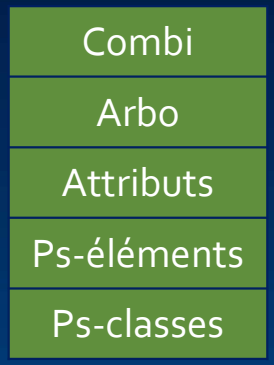
- Elle renverse l'ordre de priorités des layers.
  - Par exemple : un élément important dans les règles définies par l'utilisateur prend le pas sur tout le reste (accessibilité).
  - À utiliser avec grande parcimonie !

Sélecteurs CSS avancés

Au programme de ce chapitre…

- **Le point sur le CSS jusqu'ici**
  - *Syntaxe et pragma*
  - *Sélecteurs déjà abordés*
- **Combinaison de sélecteurs**
- **Sélecteurs basés sur l'arborescence HTML**
- **Sélecteurs basés sur les attributs**
- **Pseudo-éléments**
- **Pseudo-classes**

Nouvelles options pour écrire des sélecteurs
… pas forcément coton !



CSS jusqu'ici...

- Syntaxe
  - Règle : `sélecteur { déclarations }`
  - Déclaration : `propriété : valeur;`

- Sélecteurs simples déjà abordés

Format	Exemple	Signification
Balise	p	Tous les paragraphes <p>
Identificateur	#intro	L'élément d'identificateur intro
Classe	.motsclefs	Tous les éléments possédant la classe motsclefs

CSS jusqu'ici...

- Pragma / bonnes pratiques CSS
  - Tout écrit en minuscules
  - Attention à l'indentation

```
article {
    border: 2px dashed #1B512D;
    padding: 4px;
    margin-top: 12px;
    margin-bottom: 12px;
}
```

  - Attention à la différence entre classe et identificateur
    - Identificateur = cible un élément unique
    - Classe = peut être appliquer à plusieurs éléments
  - Attention au choix des noms (pour les classes / identificateurs) :
    - Utiliser un nom sémantique (ce que ça représente)
    - Pas un nom lié au format d'affichage (qui pourrait changer)
    - Exemples à ne pas suivre : ~~rouge~~, ~~textegras~~

Combinaisons de sélecteurs

- Combinaisons de sélecteurs :

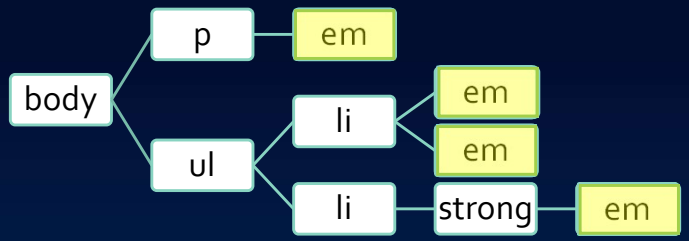
Format	Exemple	Signification
Accolé = et	`p.intro`	Les paragraphes <p> de classe intro
	`.info.new`	Les éléments possédant les classes info et new
	`li.old.nice`	Les possédant les classes old et nice
Virgule = ou	`td, th`	Les <td> et les <th>
Universel	*	Tous les éléments HTML

- Attention : « , » utilisable seulement pour des sélecteurs entiers
  - `td, th {...}`
  - `p.intro,.outro` ≡ `p.intro` ou `.outro` ≠ `p.intro` ou `p.outro` !
  - `#table1 td, th` ≡ `#table1 td` ou `th` ≠ `#table1 td` ou `#table1 th` !

Sélecteurs d'arborescence

- Connecteurs descendant (espace) et fils (>)
  - **sans restriction** : `em` ≡ les balises
  - **descendant** : `ul em` ≡ les balises descendant d'un
  - **fils** : `li > em` ≡ les balises fils (directs) d'un

- Exemple `li > em`

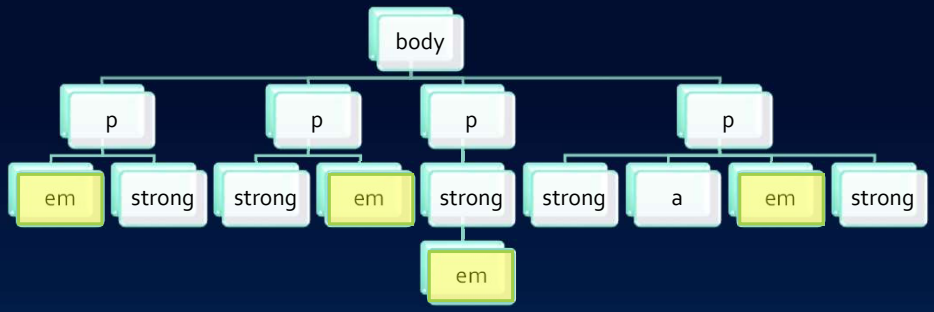



```
<p>Dans ce paragraphe, un <em>mot</em> en emphase.</p>
<ul>
  <li>Premier <em>élément</em> de la <em>liste</em></li>
  <li>Dans <strong>du gras <em>l'emphase</em>.</strong></li>
</ul>
```

Sélecteurs d'arborescence

- Connecteurs relatifs aux siblings
  - **sans restriction** : `em` ≡ toutes les balises
  - **frère** : `strong ~ em` ≡ les venant après un
  - **adjacent** : `strong + em` ≡ les juste après un

- Exemple `strong + em`




```
<p>Voici un <em>mot</em> en <strong>emphase</strong>.</p>
<p><strong>Gras</strong> <em>italique</em>.</p>
<p>Du <strong><em>gras italique</em></strong>.</p>
<p><strong>Avant</strong> <a href="www.google.be">lien</a>
  et <em>après</em> le <strong>lien</strong>.</p>
```

Sélecteurs d'arborescence

Symbole	Format	Exemple	Signification *(lire de droite à gauche)*
Espace	Descendant	`ul em`	Les se trouvant à l'intérieur d'un
>	Fils	`li > em`	Les fils (descendants directs) d'un
+	Frère suivant	`strong + em`	Les qui ont un comme précédent sibling (et viennent juste après lui)
~	Frère	`strong ~ em`	Les qui ont un parmi leurs siblings qui les précèdent

Quelques exemples plus complexes :

- `#grandcadre strong` ≡ les qui se trouvent dans #grandcadre
- `table.totaux td.num` ≡ <td> de classe "num" dans une <table> de classe "totaux"
- `p.intro + p` ≡ les <p> suivant directement un <p> de classe "intro"
- `header table em` ≡ les dans une <table> dans la section <header>
- `#cadre h2 + p` ≡ les <p> juste après un titre <h2> situé dans l'élément #cadre
- `h1 + p + p + p` ≡ tous les 3e <p>aragraphes suivant un titre <h1>

Sélecteurs via les attributs

Symbole	Format	Exemple	Signification
`[]`	Existence	`video[loop]`	<video> possédant un attribut loop
`[=]`	Valeur	`a[target="_blank"]`	Liens s'ouvrant dans un nouvel onglet
`[^=]`	Préfixe	`a[href^="http"]`	Liens dont l'url commence par http
`[$=]`	Suffixe	`a[href$=".pdf"]`	Liens dont l'url finit par .pdf
`[*=]`	Contient	`a[href*="be"]`	Liens don't l'url contient be

- On peut aussi utiliser ces conditions sans préciser de balise :
  `[id^="prem"]` ≡ les éléments (de tout type) dont l'id commence par "prem"
- On peut combiner plusieurs conditions en les "collant".
  `img[href^="http"][target="_blank"]`
- Attention à ne pas laisser de blanc !
  `a[target]` ≡ les liens possédant un attribut target
  `a [target]` ≡ les éléments descendant d'un <a> et possédant un attribut target

Pseudo-éléments

- Ciblage d'un morceau de contenu
  - qui pourrait correspondre à un élément HTML
  - Notation : préfixe `::`

Exemple	Signification
`::first-letter`	Première lettre
`::first-line`	Première ligne
`::before`	Contenu à ajouter en tant que 1er fils
`::after`	Contenu à ajouter en tant que dernier fils

```
p::first-letter {
  color: blue;
  font-weight: bold;
  font-size: 1.5em
}
a::before {
  content: "\21DB ";
  background-color: yellow;
}
```

Ce court texte comporte pas moins de ⇛deux liens qui, malheureusement, ne mènent absolument ⇛nulle part.

Pseudo-classes

- = filtres fonctionnant comme des classes prédéfinies
  - Syntaxe : préfixe :

	Format	Exemple	Signification
	:hover	`tr:hover`	Ligne de table survolée par le curseur
	:lang	`span:lang(fr)`	avec un attribut lang correspondant au français
		`:lang(en)`	Les éléments avec un attribut lang pour l'anglais
Liens	:link	`a:link`	Liens pas encore visités
Liens	:local-link	`a:local-link`	Liens internes à la page
Liens	:visited	`a:visited`	Liens déjà visités
Médias	:muted	`audio:muted`	Bandes sons actuellement en sourdine
Médias	:paused	`video:paused`	Vidéos mises en pause
Médias	:playing	`video:playing`	Vidéos en cours de lecture

Pseudo-classes liées à l'arborescence

- Pseudo-classes liées à l'ordre parmi les siblings

Format	Exemple	Signification
Premier	`li:first-child`	Premier item
Dernier	`li:last-child`	Dernier item
Ne	`li:nth-child(4)`	4e item
Ne de la fin	`li:nth-last-child(4)`	4e item depuis la fin
Seul	`li:only-child`	Item des listes à 1 item

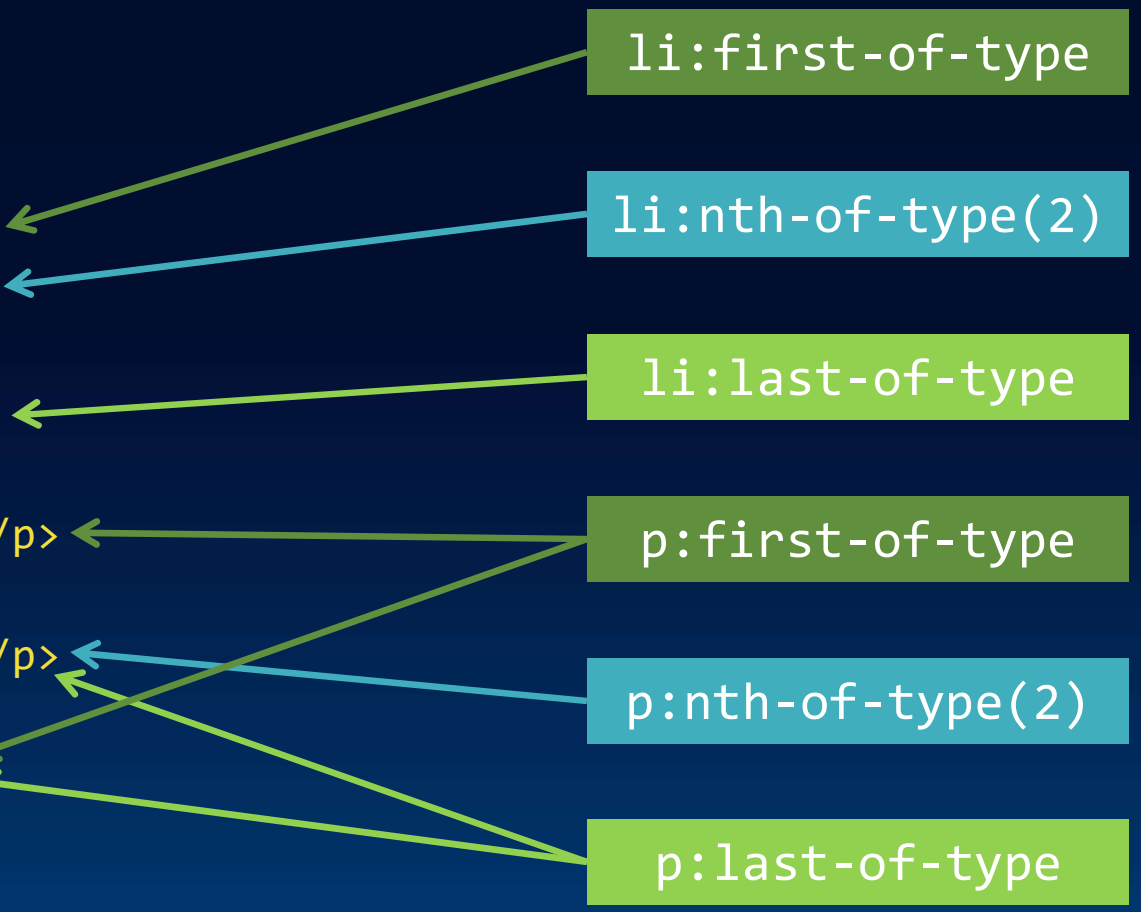
- En ignorant les siblings de type/balise différent(e) :

`em:first-of-type`	`em:nth-of-type(3)`
`em:last-of-type`	`em:nth-last-of-type(3)`
	`em:only-of-type`

Pseudo-classes liées à l'arborescence

Un exemple…

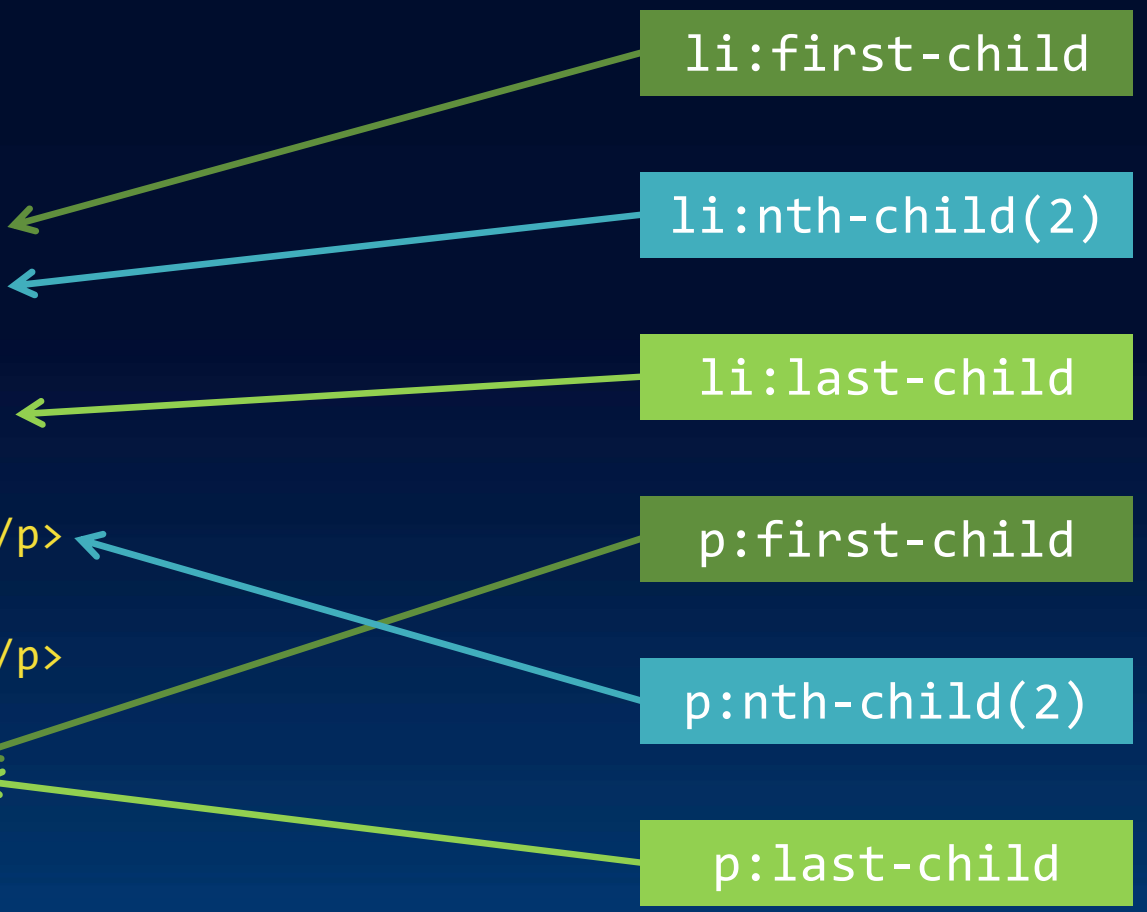
```
<body>
  <ul>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
    <li>Élément 4</li>
  </ul>
  <p>Petit paragraphe</p>
  <table>…</table>
  <p>Autre paragraphe</p>
  <aside>
    <p>Encore un.</p>
  </aside>
</body>
```



Pseudo-classes liées à l'arborescence

Un exemple…

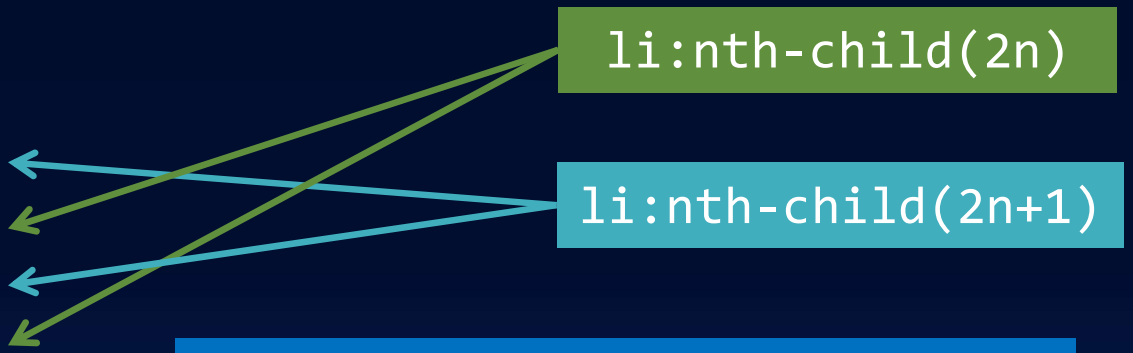
```
<body>
  <ul>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
    <li>Élément 4</li>
  </ul>
  <p>Petit paragraphe</p>
  <table>…</table>
  <p>Autre paragraphe</p>
  <aside>
    <p>Encore un.</p>
  </aside>
</body>
```



Pseudo-classes liées à l'arborescence

Listes alternées (utiles aussi pour les lignes de tableau) :

```
<body>
  <ul>
    <li>Élément 1</li>
    <li>Élément 2</li>
    <li>Élément 3</li>
    <li>Élément 4</li>
  </ul>
  <p>Petit paragraphe</p>
  <table>…</table>
  <p>Autre paragraphe</p>
  <aside>
    <p>Encore un.</p>
  </aside>
</body>
```



Paramètre entre parentheses :

- un nombre
- even (pair) = 2n
- odd (impair) = 2n+1
- une formule au format An+B où
  - A et B sont des entiers
  - n prend les valeurs de ℕ (0, 1, 2…)
  - ex : 2n+3 — 3, 5, 7, 9…
  - ex : -n+4 — 4, 3, 2, 1, ~~0, -1, -2, …~~ (les 4 premiers)
- peut être suivi de of + sélecteur

Pseudo-classes fonctionnelles

- Pseudo-classes fonctionnelles
  - Suivies d'un argument entre parenthèses
  - Qui peut être un sélecteur ou une liste de sélecteurs (virgule)

Format	Exemple	Signification
:not	`p:not(.intro)`	Paragraphes qui n'ont pas la classe intro
:is	`:is(ul, ol)`	Listes ordonnées et listes non ordonnées
:has	`p:has(em)`	Paragraphes qui contiennent un em
:where		Identique à :is sauf pour la spécificité (voir plus tard)

- Listes de sélecteurs = disjonctions (ou). Attention à la logique !
  - `p:has(em, strong)` ≡ paragraphes avec un em ou un strong
    ≠ `p:has(em):has(strong)`
  - `td:is(.int, .float)` ≡ cellules de classe int ou float
    ≠ `td:is(.int):is(.float)` ≡ `td.int.float`
  - `td:not(.int, .float)` ≡ cellules avec ni la classe int ni la classe float

Pseudo-classes fonctionnelles

- Les différents types de sélecteurs sont combinables.
  - `table a:not([target="_blank"])`
    ≡ liens dans une table, sauf ceux qui s'ouvrent dans un nouvel onglet.
  - `ul li:not(:first-of-type, :last-of-type) em`

- Utilisation de `:is` pour rassembler des règles
  - `#matable td a, #matable th a` → `#matable :is(td,th) a`
  - `:is(ul,ol) :is(ul,ol) li`

- Les paramètres de `:has` sont des sélecteurs relatifs.
  - = accolés à l'élément considéré
  - Si aucun connecteur spécifié, par défaut connecteur de descendance (espace)
  - `p:has(em)` ≡ p pour lesquels le sélecteur `p em` est satisfaisable
  - `p:has(+ p)` ≡ p pour lesquels le sélecteur `p + p` est satisfaisable
  - `a:has(> img)`
  - `:is(h1,h2,h3):has(+ :is(h2,h3,h4))`
  - `li:has(+ li em, + li.important)`