



PDF Creator Trial

Langages Web

HTML et CSS



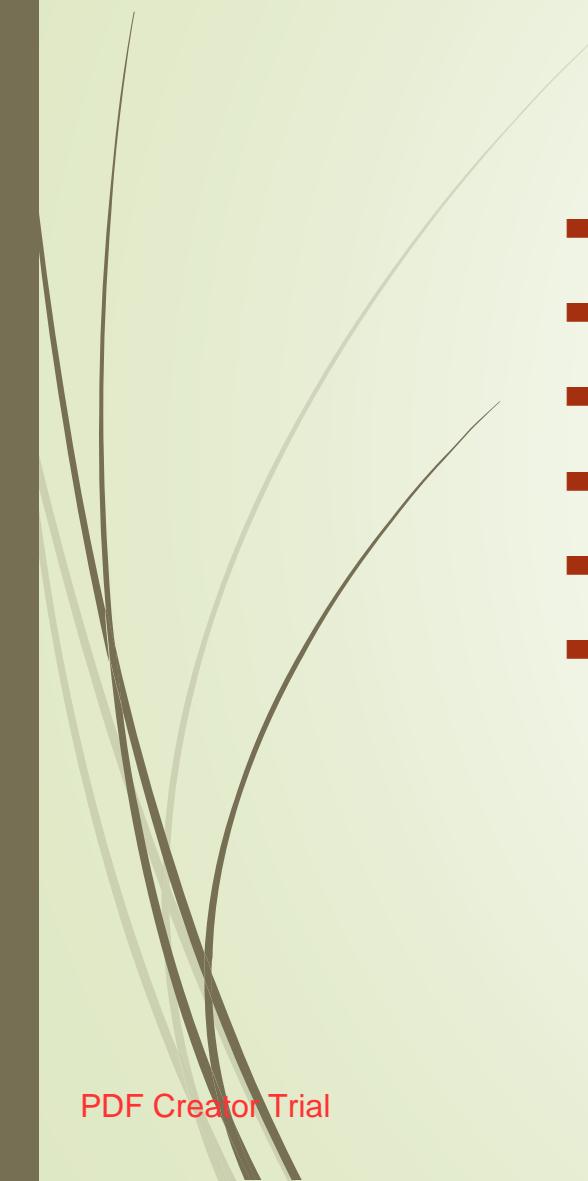
Gestion de versions

Version	Date	Rédacteur	Description
1.0	13/11/2018	Soupramanien	Description détaillée des langages HTML et CSS
1.1	29/09/2020	Lesly	Corrections, mise en page et ajout d'animations
1.2	03/10/2021	Lesly	Application de la charte Baobab et de la traçabilité

Sommaire

- ▶ [Comprendre le rôle d'HTML](#)
- ▶ [Apprendre la syntaxe d'HTML5](#)
- ▶ [Ecrire du HTML sémantique](#)
- ▶ [Comprendre la relation entre HTML et CSS](#)
- ▶ [Apprendre la syntaxe et les sélecteurs de CSS3](#)
- ▶ [Méthodes pour appliquer des styles en HTML](#)
- ▶ [Utiliser CSS3 pour formater et mettre en forme une page web](#)
- ▶ [Tableau HTML](#)
- ▶ [Formulaire HTML](#)
- ▶ [Boîtes flexibles CSS](#)

Comprendre le rôle d'HTML



Qu'est-ce qu'HTML ?

- ▶ HyperText Markup Language
- ▶ Inventé par Tim Berners-Lee
- ▶ N'est pas un langage de programmation comme Java ou C
- ▶ Est un langage à balise
- ▶ Spécifie la structure d'une page Web
- ▶ A des éléments (balises) pré-définis qu'on peut utiliser pour marquer le contenu
 - ▶ <p>, <h1>, <a>, , etc...

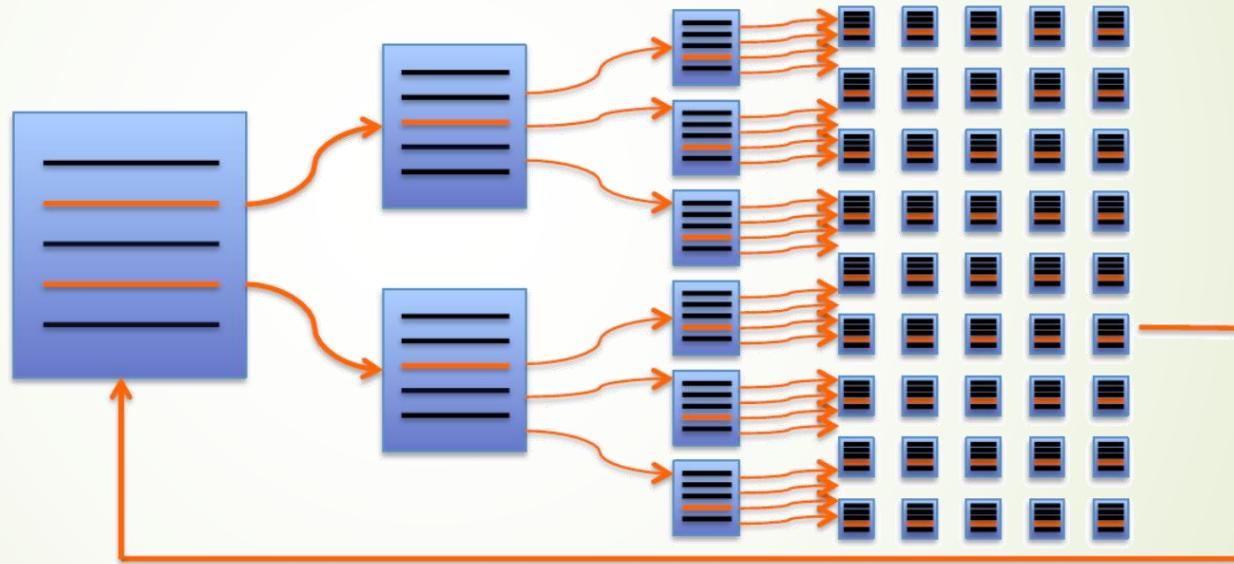
Qu'est-ce qu'HTML ?

H T M L
HyperText Markup Language

Langage à balises de description de documents hypertextes

Qu'est-ce qu'HTML ?

Hypertext Markup Language



Hyper : Non linéaire, liens entre noeuds

Text : Composé de texte

Qu'est-ce qu'HTML ?

Hypertext **Markup** Language

```
<!doctype html>
<html>
<head>
    <title>Why I Love This Course</title>
</head>
<body> [ . . . ]
</body>
</html>
```

content

Markup : Marqué, balisé

Qu'est-ce qu'HTML ?

Hypertext Markup **Language**

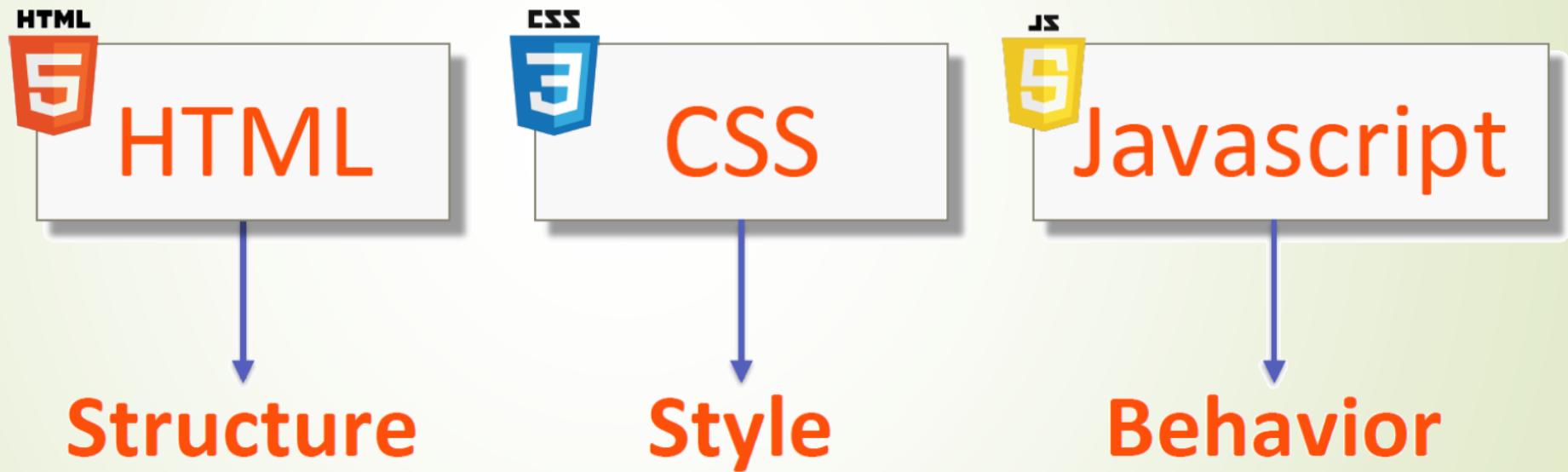
```
<h1>  
  <div>Hello World!</h1>  
</div>
```

```
<h1>  
  <div>Hello World!</div>  
</h1>
```

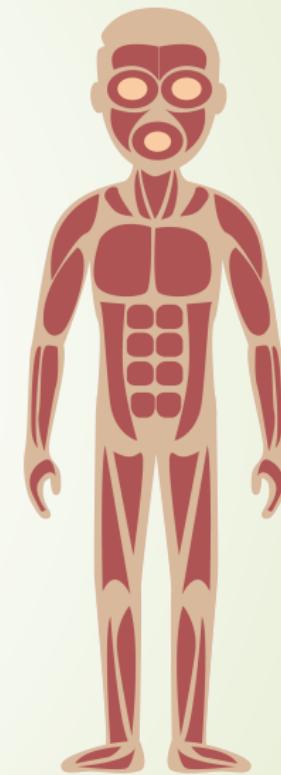
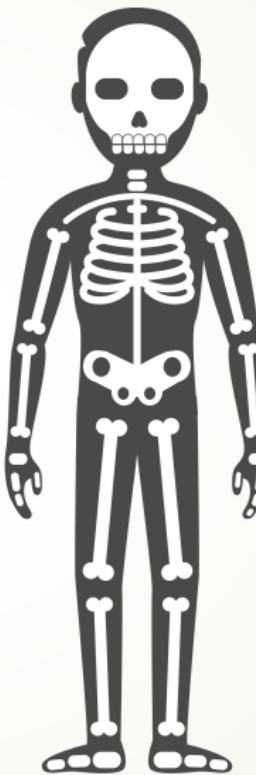
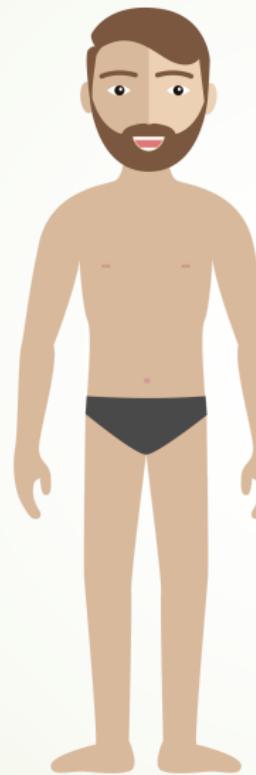


Language : Langage

Technologies qui pilotent le web

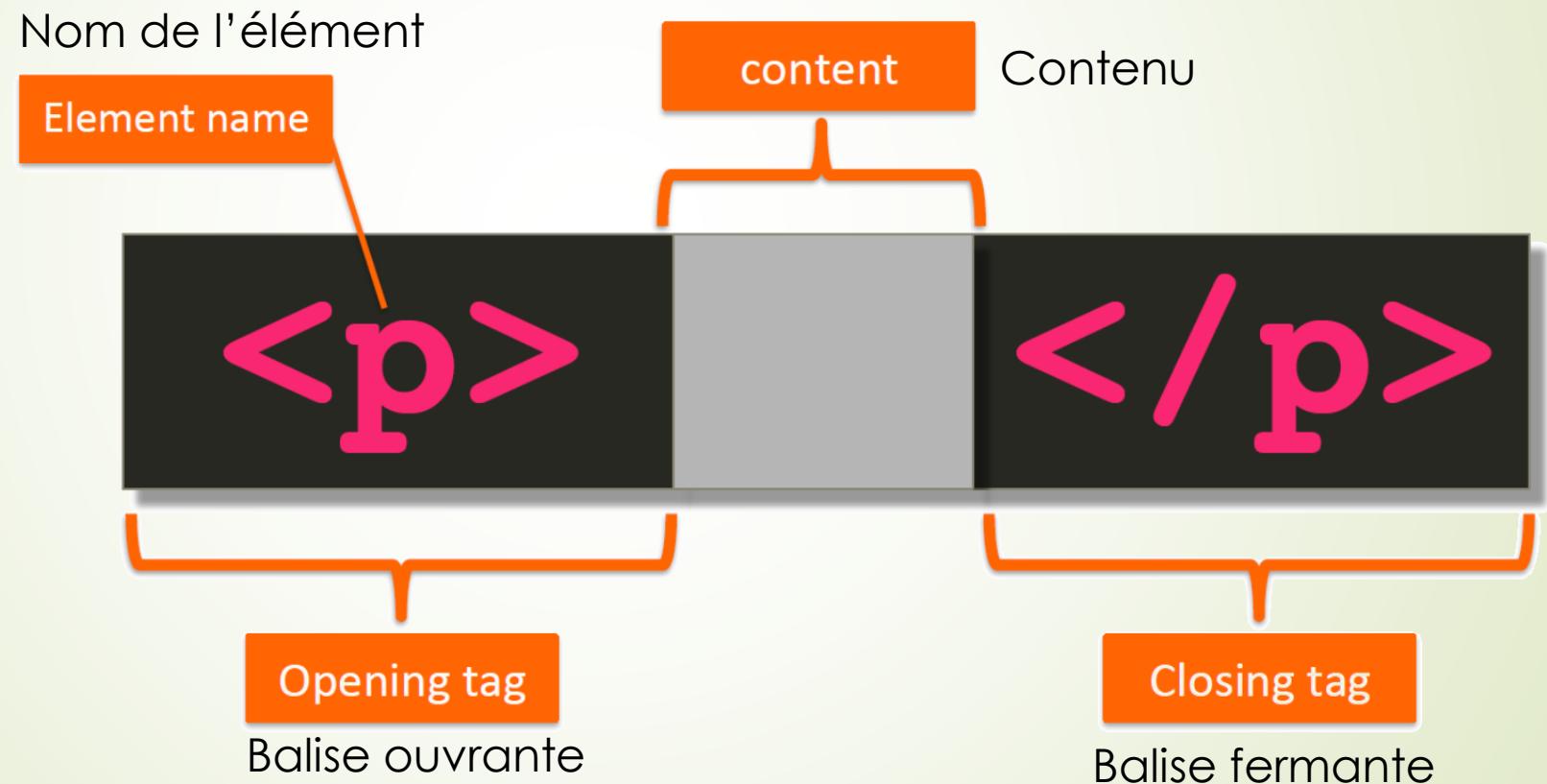


Technologies qui pilotent le web



Apprendre la syntaxe d'HTML5

Anatomie d'une balise HTML



Anatomie d'une balise HTML

Saut de ligne

Line Break

The diagram shows a black rectangular box containing the pink text "
". A white bracket is positioned below the box, spanning its width. An orange callout box labeled "Line Break" is positioned above the top edge of the black box. Another orange callout box labeled "Only opening tag" is positioned below the bottom edge of the black box.

Only opening tag

Uniquement la balise ouvrante

Règle horizontale

Horizontal Rule

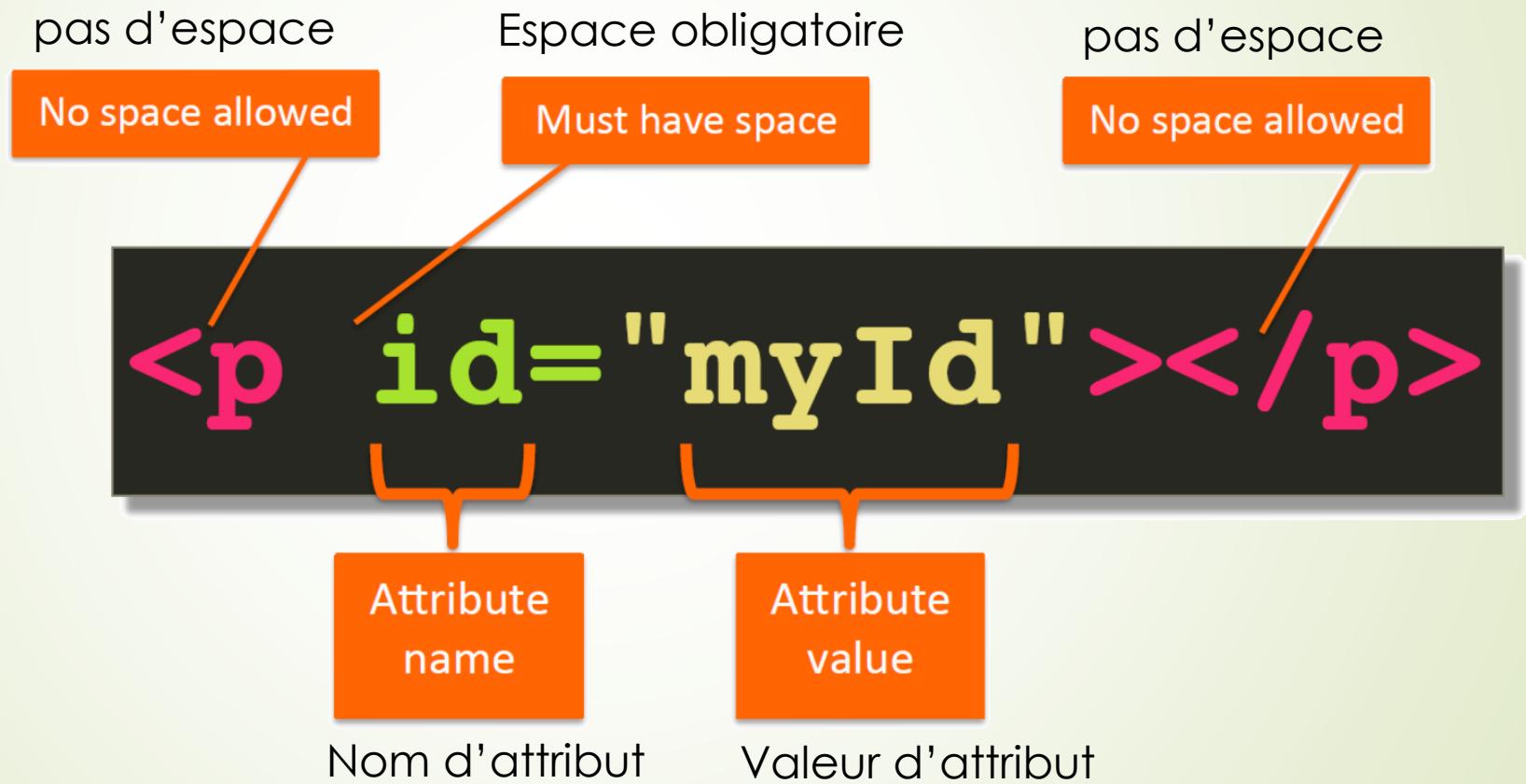
The diagram shows a black rectangular box containing the pink text "<hr>". A white bracket is positioned below the box, spanning its width. An orange callout box labeled "Horizontal Rule" is positioned above the top edge of the black box. Another orange callout box labeled "Only opening tag" is positioned below the bottom edge of the black box.

<hr>

Only opening tag

Uniquement la balise ouvrante

Anatomie d'une balise HTML



Anatomie d'une balise HTML

```
<p onclick="alert('hi')"></p>
```

Guillemets double externe

Outer double quotes

Guillemets simple interne

Inner single
quotes

Anatomie d'une balise HTML

```
<p/>
```



```
<p></p>
```



Anatomie d'un document HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ma première page</title>
  </head>
  <body>
    <p>C'est ma première page web</p>
    <span>Hello</span>
    <span>World !</span>
  </body>
</html>
```

Balise <Head>

```
<head>
  <meta charset="utf-8">
  <title>My test page</title>
</head>
```

- ▶ Contenu n'est pas affiché sur la page
- ▶ Contient metadata (métadonnée) du document HTML
 - ▶ Une **métadonnée** est une donnée qui décrit une donnée. Ici, la description du document HTML
- ▶ Élément <title> est utilisé pour ajouter un titre au document
 - ▶ affiché dans la barre de titre du navigateur ou dans l'onglet de la page
 - ▶ Est utilisé pour décrire la page quand on l'ajoute dans favori.

Metadata : balise <meta>

- ▶ Une **métadonnée** est une donnée qui décrit une donnée
 - ▶ <meta charset="utf-8">
 - ▶ déclare l'encodage utilisé par la page
 - ▶ Plusieurs éléments <meta> incluent les attributs "name" et "content"
 - ▶ **Name** définit le nom d'un métadonnée au niveau du document
 - ▶ **Content** fournit la valeur associée avec l'attribut http-equiv ou l'attribut name suivant le contexte utilisé

```
<meta name="author" content="Soupramanien BOUVANESVARY">
<meta name="description" content="The MDN Web Docs Learning Area aims to
provide complete beginners to the Web with all they need to know to get
started with developing web sites and applications.">
```

- ▶ Pour plus d'info : [consulter cette page](#)

Autres éléments dans <head>

- ▶ Ajout d'icône personnalisé à votre site

```
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```

- ▶ Ici, on ajoute favicon (favorites icon)

- ▶ Liaison de JavaScript et CSS à HTML

```
<link rel="stylesheet" href="my-css-file.css">  
<script src="my-js-file.js"></script>
```

Entité HTML : inclure les caractères spéciaux dans HTML

- ▶ <, > et & sont des caractères réservés
- ▶ On doit toujours échapper ces caractères dans notre documents HTML

Character	Description	Entity Name	Entity Number
©	Copyright symbol	©	©
®	Registered trademark	®	®
€	Euro	€	€
&	Ampersand	&	&
<	Less than	<	<
>	Greater than	>	>
	Nonbreaking space	 	



Ecrire du HTML sémantique

Élément HTML Sémantique

- ▶ Élément qui implique du sens au contenu
- ▶ L'un des principaux buts de HTML est de **structurer du texte** et lui **donner du sens** (ce que l'on appelle la sémantique) afin que le navigateur puisse l'afficher correctement
 - ▶ Humain et/ou machine peut mieux comprendre le contenu entouré par un élément sémantique.

Racine de sectionnement <body>

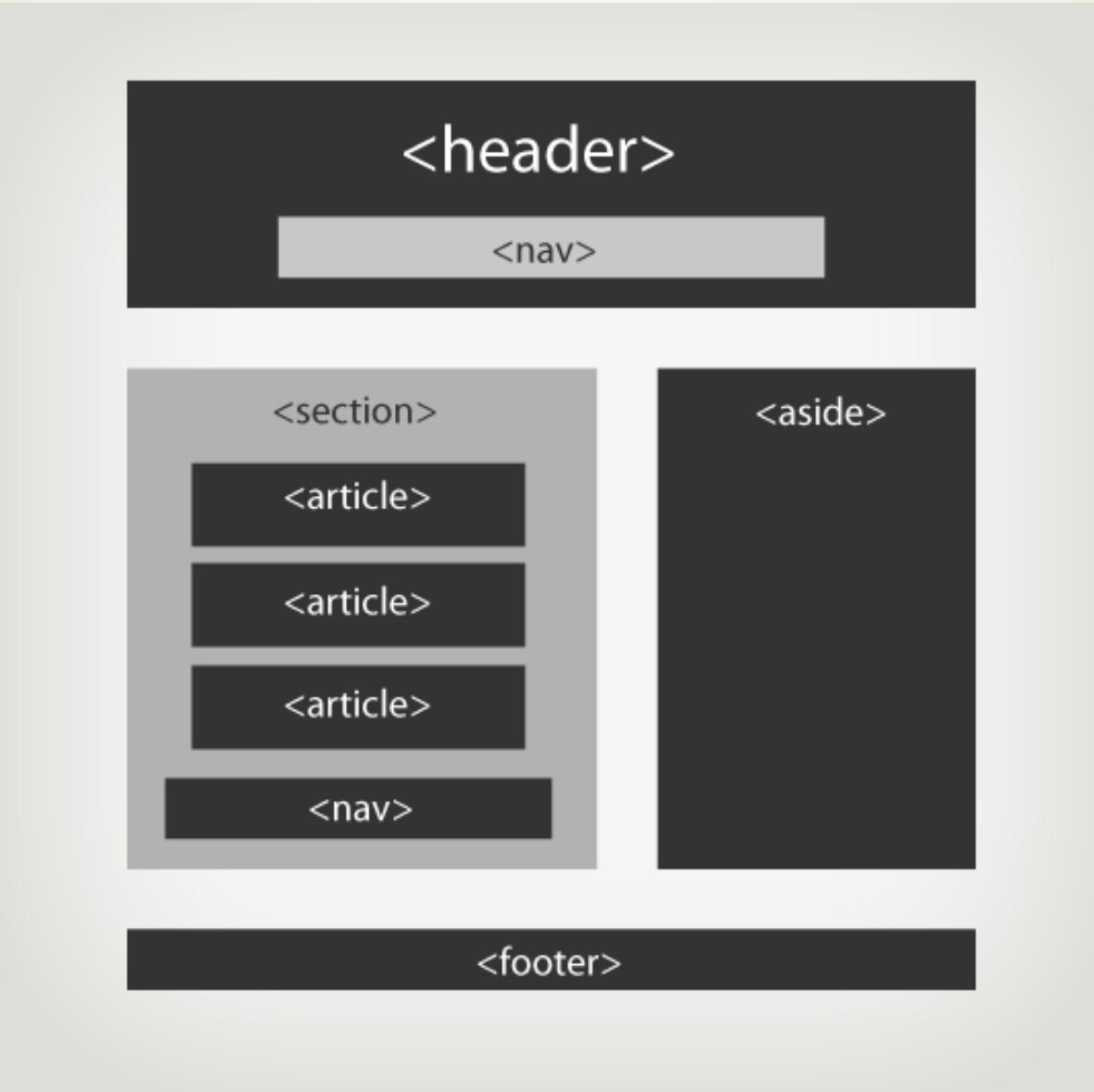
- ▶ L'élément <body> représente le **contenu principal** du document HTML.
- ▶ Il ne peut y avoir qu'un élément <body> par document.

Sectionnement du contenu

- ▶ Organiser le contenu d'une page en différentes **sections** permet d'avoir une structure logique au sein d'un document.
- ▶ Grâce à ces éléments
 - ▶ on peut créer un plan pour la page
 - ▶ ajouter des titres pour identifier les sections et
 - ▶ également gérer un en-tête et un bas de page.

Sectionnement du contenu

Nom	Détails
<code><article></code>	Portion de contenu indépendante, se suffisant en termes de compréhension (pouvant être syndiquée dans un flux RSS). Exemples : article de presse, fiche cinéma, réponse de forum, commentaire d'article
<code><aside></code>	Contenu indirectement lié au contenu principal, non nécessaire à sa compréhension : définitions, digressions, sidebar, compléments...
<code><section></code>	Élément générique pour une section de contenu ou d'application web, utilisé à défaut d'un autre élément de section plus sémantique tel que <code><article></code> , <code><nav></code> , <code><header></code> , <code><footer></code> , <code><aside></code> . À ne pas confondre avec <code><div></code> qui n'a aucune valeur sémantique. En général, <code><section></code> contient un titre qui la définit.
<code><nav></code>	Regroupe la sélection des principaux liens pour naviguer dans le site, l'application ou le document.
<code><header></code>	S'applique au document entier, mais aussi à toute section au sens large, <code><article></code> y compris.
<code><footer></code>	S'applique au document entier, mais aussi à toute section au sens large, <code><article></code> y compris.



Éléments <h1> à <h6>

- ▶ Les éléments <h1> à <h6> représentent six **niveaux de titres** dans un document
- ▶ <h1> est le plus important et <h6> est le moins important
- ▶ Un élément de titre décrit brièvement le sujet de la section qu'il introduit
 - ▶ L'information d'un titre peut être utilisée par les agents utilisateurs, par exemple, pour construire automatiquement une table des matières d'un document.
 - ▶ Les titres ne doivent pas être utilisé afin de réduire ou d'augmenter la taille de la police d'un texte : il faut pour cela utiliser la propriété CSS font-size à la place.
 - ▶ On évitera de sauter des niveaux de titre : on commence toujours par <h1> puis <h2> et ainsi de suite.
 - ▶ Il est préférable d'éviter d'utiliser plus d'un élément <h1> sur une même page.
- ▶ L'élément <hgroup> permet de regrouper le titre principal d'une section avec son (ou ses) sous-titre(s) afin d'obtenir un titre sur plusieurs niveaux sémantiques.

Contenu textuel

- ▶ Le contenu HTML textuel permet d'organiser des blocs ou des sections de contenu entre la balise ouvrante `<body>` et la balise fermante `</body>`.
- ▶ Ces éléments sont cruciaux pour **l'accessibilité** et le **référencement** car ils permettent d'identifier le sens du contenu.

Éléments de groupement de contenu

► Paragraphe

- ▶ L'élément HTML `<p>` représente un **paragraphe de texte**.
- ▶ Les paragraphes sont généralement représentés comme des blocs et séparés par un espace vertical, leur première ligne est également parfois indentée.
- ▶ Les paragraphes sont des éléments blocs.

```
<p>Premier paragraphe du texte. J'aime les licornes beaucoup beaucoup beaucoup.</p>
```

```
<p>Deuxième paragraphe du texte. Et si j'en avais une apprivoisée je serais très contente.</p>
```

Éléments de groupement de contenu

► Règle horizontale

- ▶ L'élément HTML `<hr>` représente un **changement thématique** entre des éléments de paragraphe
- ▶ Exemples
 - ▶ un changement de décor dans un récit
 - ▶ un changement de sujet au sein d'une section

```
<p>paragraph 1</p>
<hr />
<p>paragraph 2</p>
```

Éléments de groupement de contenu

► Texte préformaté

- ▶ L'élément HTML `<pre>` représente du texte préformaté, généralement écrit avec une **police à chasse fixe**.
- ▶ Le texte est affiché tel quel, les espaces utilisés dans le document HTML seront retranscrits
- ▶ Il est nécessaire d'échapper les caractères '`<`' en '`<`' afin de s'assurer que le code écrit entre les éléments ne soit pas interprété de façon involontaire.

```
<pre>
body {
    color:red;
}
</pre>
```



```
body {
    color:red;
}
```

Éléments de groupement de contenu

► main

- ▶ L'élément HTML <main> représente le **contenu majoritaire** du <body> du document.
- ▶ Le contenu principal de la zone est constitué de contenu directement en relation, ou qui étend le sujet principal du document ou de la fonctionnalité principale d'une application.
- ▶ Un document ne peut pas avoir plus d'un seul élément <main> sans attribut hidden.

► division

- ▶ L'élément HTML <div> (qui signifie **division du document**) est un conteneur générique qui permet d'organiser le contenu sans représenter rien de particulier.
- ▶ Il peut être utilisé afin de grouper d'autres éléments pour leur appliquer un style (en utilisant les attributs class ou id) ou parce qu'ils partagent des attributs aux valeurs communes, tel que lang.
- ▶ Il doit uniquement être utilisé lorsqu'aucun autre élément sémantique (par exemple <article> ou <nav>) n'est approprié.

Éléments de groupement de contenu

► **blockquote / cite**

- ▶ L'élément `<blockquote>` (qui signifie **bloc de citation**) indique que le texte contenu dans l'élément est une citation longue.
- ▶ Le texte est généralement affiché avec une indentation
- ▶ Une URL indiquant la source de la citation peut être donnée grâce à l'attribut « `cite` » tandis qu'un texte représentant la source peut être donné via l'élément `<cite>`.



```
<blockquote  
cite="https://www.huxley.net/bnw/four.html">  
<p>Words can be like X-rays, if you use them  
properly—they'll go through anything. You read  
and you're pierced.</p>  
<footer>—Aldous Huxley, <cite>Brave New  
World</cite></footer>  
</blockquote>
```



“Words can be like
X-rays, if you use
them properly—
they’ll go through
anything. You read
and you’re pierced.”

—Aldous Huxley, *Brave
New World*

```
blockquote {  
    margin: 0;  
}  
  
blockquote p {  
    padding: 15px;  
    background: #eee;  
    border-radius:  
    5px;  
}  
  
blockquote p::before {  
    content: '\201C';  
}  
  
blockquote p::after {  
    content: '\201D';  
}
```

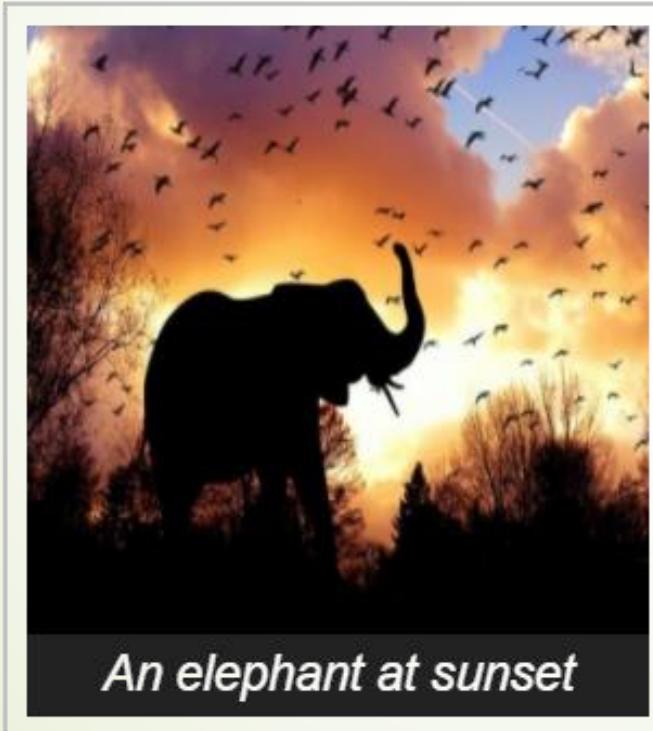


Éléments de groupement de contenu

► **figure**

- ▶ L'élément HTML <figure> représente une figure (un **schéma**), qui peut être accompagné d'une **légende** grâce à l'élément <figcaption>.
- ▶ Il est normalement référencé de manière unique.
- ▶ C'est souvent une image, une illustration, un diagramme, un fragment de code ou un schéma auquel il est fait référence dans le texte principal mais qui peut être utilisé sur une autre page ou dans une annexe sans que cela affecte le contenu principal.

```
<figure>
  
  <figcaption>An elephant at sunset</figcaption>
</figure>
```



```
figure {
  border: thin #c0c0c0 solid;
  display: flex;
  flex-flow: column;
  padding: 5px;
  max-width: 220px;
  margin: auto;
}

img {
  max-width: 220px;
  max-height: 150px;
}

figcaption {
  background-color: #222;
  color: #fff;
  font: italic smaller sans-serif;
  padding: 3px;
  text-align: center;
}
```



Éléments de groupement de contenu

► details / summary

- ▶ L'élément HTML `<details>` est utilisé comme un outil permettant de révéler une information. Un résumé ou un intitulé peuvent être fournis grâce à un élément `<summary>`.
- ▶ La plupart du temps, le contrôle utilisé pour cet élément est un triangle qui est tourné ou tordu afin d'indiquer si l'élément est révélé ou non. Si le premier élément fils de l'élément `<details>` est un élément `<summary>`, c'est le contenu de ce dernier qui est utilisé comme intitulé pour le contenu à révéler (l'intitulé est donc toujours visible).

```
<details>
  <summary>Details</summary>
  Something small enough to escape
  casual notice.
</details>
```



▼ Details

Something small enough to escape
casual notice.

Éléments de groupement de contenu

► Listes de définitions

- ▶ **<dl>** Liste de définitions : Associe un « terme à définir » (**<dt>**) et des « définitions » (**<dd>**) s'y rapportant. Contenu autorisé : succession de **<dt>** et **<dd>**.
- ▶ **<dt>** Terme/titre de définition : Enfant d'une liste **<dl>**, balisant le terme à définir. Contenu autorisé : contenus textuels y compris des balises de niveau intra-paragraphe chargées de contenu.
- ▶ **<dd>** Contenu de la définition : Enfant d'une liste **<dl>**, recueillant la définition du terme **<dt>** qui le précède immédiatement.

```
<dl>
  <dt>Beast of Bodmin</dt>
  <dd>A large feline inhabiting Bodmin Moor.</dd>
  <dt>Morgawr</dt>
  <dd>A sea serpent.</dd>
  <dt>Owlman</dt>
  <dd>A giant owl-like creature.</dd>
</dl>
```

Beast of Bodmin
A large feline inhabiting Bodmin Moor.

Morgawr
A sea serpent.

Owlman
A giant owl-like creature.

Éléments de groupement de contenu

► Liste ordonnée

- ▶ L'élément HTML `` représente une liste ordonnée.
- ▶ Les éléments d'une telle liste sont généralement affichés avec un **indicateur ordinal** pouvant prendre la forme de nombres, de lettres, de chiffres romains ou de points.
- ▶ La mise en forme de la numérotation n'est pas utilisée dans la description HTML mais dans la feuille de style CSS associée grâce à la propriété `list-style-type`.
- ▶ Il doit être utilisé pour regrouper plusieurs éléments qui ont une relation d'ordre.

```
<ol>
  <li>Premier élément</li>
  <li>Deuxième élément</li>
  <li>Troisième élément</li>
</ol>
```



1. Premier élément
2. Deuxième élément
3. Troisième élément

Éléments de groupement de contenu

► Liste sans ordre

- ▶ L'élément HTML `` représente une liste d'éléments **sans ordre particulier**.
- ▶ Il est souvent représenté par une liste à puces.
- ▶ Il doit être utilisé pour regrouper plusieurs éléments qui n'ont pas de relation d'ordre.

```
<ul>
  <li>1 artichaut</li>
  <li>De l'essuie-tout</li>
  <li>200g de chocolat</li>
</ul>
```



- 1 artichaut
- De l'essuie-tout
- 200g de chocolat

Éléments de groupement de contenu

```
<h2>Book Topics</h2>
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
</ul>
<h2>HTML Chapters</h2>
<ol start="4">
    <li>Structural Elements</li>
    <li>Text Elements</li>
    <li>Table Elements</li>
    <li>Embedded Elements</li>
    <li>Form Elements</li>
</ol>
```

Book Topics

- HTML
- CSS
- JavaScript

HTML Chapters

4. Structural Elements
5. Text Elements
6. Table Elements
7. Embedded Elements
8. Form Elements

L'élément **** dispose les attributs suivants :

- **reversed** : Cet attribut booléen précise que les objets listés par l'élément sont affichés avec un ordre inversé
- **start** : La valeur de cet attribut, un nombre entier, définit l'indice à partir duquel les objets de la liste sont numérotés.

Sémantique du texte en ligne

- ▶ Les éléments pour le texte en ligne peuvent être utilisés pour définir la signification, la structure ou la mise en forme d'un terme, d'une ligne ou d'un fragment de texte.

Mettre en évidence du texte

► Importance -

- L'élément HTML indique que le texte a une **importance** particulière ou un certain sérieux voire un caractère urgent.
- Cela se traduit généralement par un affichage en gras.

► Emphase -

- L'élément HTML (pour emphase) est utilisé afin de marquer un texte sur lequel on veut **insister**.
- Les éléments peuvent être imbriqués, chaque degré d'imbrication indiquant un degré d'insistance plus élevé

Mettre en évidence du texte

► Marqué - <mark>

- ▶ L'élément HTML <mark> représente un texte marqué ou **surligné** à cause de sa pertinence dans le contexte.
- ▶ Il peut par exemple être utilisé afin d'indiquer les correspondances d'un mot-clé recherché au sein d'un document.

► Voix alternative - <i>

- ▶ L'élément HTML <i> représente un morceau de texte qui se **différencie** du texte principal.
- ▶ Cela peut par exemple être le cas pour des termes techniques, des phrases dans une langue étrangère ou encore l'expression des pensées d'un personnage.
- ▶ Le contenu de cet élément est généralement affiché en italique.

Mettre en évidence du texte

► Petit - <small>

- L'élément HTML <small> permet de représenter des commentaires ou des textes à écrire en **petits caractères** (des termes d'un contrat, des mentions relatives au droit d'auteur, etc.) quelle que soit la présentation.

► Barré - <s>

- L'élément HTML <s> permet d'afficher du texte qui est **barré** car il n'est plus pertinent ou car il est obsolète.
- <s> ne doit pas être employé pour indiquer des éditions dans un document (on utilisera alors et <ins>).

►

- L'élément permet d'attirer l'attention du lecteur sur le contenu d'un élément sans que ce contenu revêt une importance particulière

Mettre en évidence du texte

► <u>

- ▶ L'élément HTML `<u>` permet d'afficher un fragment de texte qui est annoté avec des éléments non textuels.
- ▶ Par défaut, le contenu de l'élément est **souligné**.
- ▶ Cela pourra par exemple être le cas pour marquer un texte comme étant un nom propre chinois, ou pour marquer un texte qui a été mal orthographié.

Exemples

```
<p>  
"Read my lips: <mark>no new taxes</mark>", declared presidential candidate George H.  
W. Bush in 1988. However, the 1990 budget agreement increased taxes in several areas.  
</p>
```

"Read my lips: **no new taxes**". declared presidential candidate George H. W. Bush in 1988. However, the
1990 budget agreement increased taxes in several areas.

```
<p>  
"Class, pay attention!" <i>I wonder if they're even listening to me.</i> "Who's ready  
for tomorrow's exam?"  
</p>
```

"Class, pay attention!" *I wonder if they're even listening to me.* "Who's ready for tomorrow's exam?"

```
<p>I am <em>glad</em> you weren't <em>late</em>. </p>
```

I am *glad* you weren't *late*.

```
<p>This liquid is <strong>highly toxic</strong>. </p>
```

This liquid is **highly toxic**.

Date et heure

- ▶ Balise attribuant une signification standardisée à une date (et heure si nécessaire), compréhensible par un robot ou un programme, via l'attribut **datetime**.
- ▶ La présence de l'attribut **pubdate** signifie que la valeur de temps vaut pour la date de publication du plus proche ancêtre `<article>`.

```
<article>
  <p>Publié le <time datetime="2012-04-01T13:37:00Z">1er avril 2012</time></p>
</article>
```

Abréviations

► <abbr>

- L'élément <abbr> représente une **abréviation** ou un **acronyme** et permet, de façon optionnelle, d'en fournir une description complète.
- S'il est présent, l'attribut title doit contenir cette même description complète et rien d'autre.

```
<p>  
The use of <abbr title="Hypertext Markup Language">HTML</abbr> has  
contributed greatly to the popularity of web-based applications. <br />  
</p>
```

The use of HTML has contributed greatly to the popularity of web-based applications.

Hypertext Markup Language

Subscripts et Superscripts

► <sub>

- L'élément HTML <sub> est utilisé, pour des raisons typographiques, afin d'afficher du texte souscrit (ou en **indice**) (plus bas et généralement plus petit) par rapport au bloc de texte environnant.

► <sup>

- L'élément HTML <sup> est utilisé, pour des raisons typographiques, afin d'afficher du texte en **exposant** (plus haut et généralement plus petit) par rapport au bloc de texte environnant.

```
<p>
H<sub>2</sub>O is the chemical formula for water.<br />
e=mc<sup>2</sup> is the formula for mass-energy equivalence.
</p>
```

H₂O is the chemical formula for water.

e=mc² is the formula for mass-energy equivalence.

Span et saut de ligne

► Span

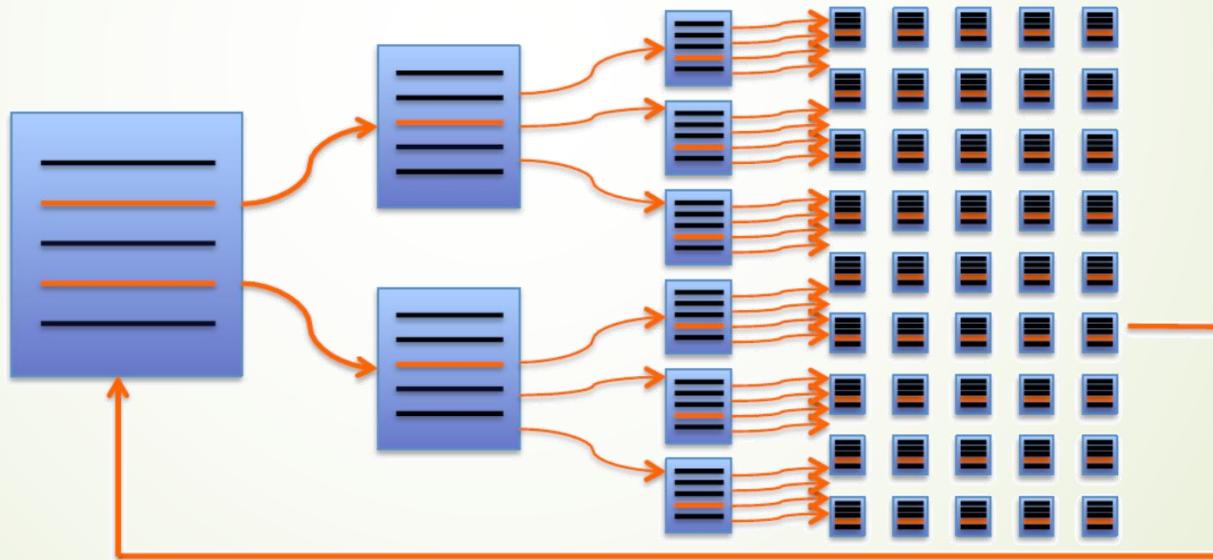
- ▶ L'élément HTML `` est un conteneur générique en ligne (inline) pour les contenus phrasés.
- ▶ Il ne représente rien de particulier.
- ▶ Il peut être utilisé pour **grouper des éléments** afin de les mettre en forme (grâce aux attributs `class` ou `id` et aux règles CSS) ou parce qu'ils partagent certaines valeurs d'attribut comme `lang`.
- ▶ Il doit uniquement être utilisé lorsqu'aucun autre élément sémantique n'est approprié.
- ▶ `` est très proche de l'élément `<div>`, mais l'élément `<div>` est un élément de bloc, alors que `` est un élément en ligne.

► Saut de ligne

- ▶ L'élément HTML `
` crée un saut de ligne (un **retour chariot**) dans le texte.
- ▶ Il s'avère utile lorsque les sauts de ligne ont une importance (par exemple lorsqu'on écrit une adresse ou un poème).

Hyperliens

- ▶ **Hyperliens** sont très important — ce sont eux qui forment le web (la toile)



Ancre (a)

- ▶ L'élément <a> (pour ancre ou anchor en anglais) définit un **hyperlien vers un autre endroit** de la même page ou vers une autre page sur le Web.
- ▶ Le contenu dans l'élément ancre peut être de type bloc ou inline.
- ▶ Il supporte l'attribut **href**,
 - ▶ qui indique la cible du lien sous la forme d'une URL ou d'un fragment d'URL
 - ▶ Un fragment d'URL est un nom précédé par un dièse (#), qui indique une cible interne au document (un ID)
 - ▶ Les URL ne se limitent pas au documents web sur HTTP.
 - ▶ Les URL peuvent utiliser n'importe quel protocole supporté par le navigateur.
 - ▶ Par exemple, file, ftp et mailto fonctionnent avec la plupart des agents-utilisateur
 - ▶ http: - ressource web
 - ▶ ftp: - transfère de fichier
 - ▶ mailto: - envoie un email
 - ▶ tel: - compose un numéro de téléphone (pour les téléphones mobiles)
 - ▶ file: - ouvre un fichier

Ancre (a)

- ▶ L'attribut **target** indique où afficher la ressource liée.
- ▶ Il s'agit du nom, ou du mot-clé, d'un contexte de navigation (par exemple, un onglet, une fenêtre, ou une <iframe>).
- ▶ Les mots-clés suivants ont un sens particulier :
 - ▶ **_self** :
 - ▶ charge la réponse dans le contexte de navigation courant.
 - ▶ Il s'agit de la valeur par défaut quand l'attribut n'est pas renseigné.
 - ▶ **_blank** :
 - ▶ charge la réponse dans un nouveau contexte de navigation.
 - ▶ Ajouter un lien vers une autre page en utilisant target="_blank" exécutera la page dans le même processus que la page courante.
 - ▶ Si la nouvelle page consomme de nombreuses ressources, les performances de la page courante pourront en pâtir.
 - ▶ Pour éviter ce problème, on pourra utiliser rel="noopener".

Ancre (a)

- ▶ L'attribut **download** indique au navigateur de télécharger l'objet vers lequel pointe l'URL plutôt que d'y diriger l'utilisateur.
- ▶ Cela ouvre une invite pour enregistrer la cible du lien comme un fichier local.
- ▶ Si l'attribut a une valeur, le navigateur doit l'interpréter comme le nom de fichier par défaut pour l'invite qui s'ouvre. Ex: download="MyPicture.png"
- ▶ Les autres attributs fournissent uniquement les informations sémantiques :
 - ▶ **hreflang** - Cet attribut indique la langue de la ressource liée
 - ▶ **rel** - Cet attribut indique la relation entre la cible du lien et l'objet faisant le lien
 - ▶ **type** - Cet attribut indique le type de média de la cible du lien, sous la forme d'un type MIME



Exemples

```
<p>I'm creating a link to  
<a href="https://www.mozilla.org/en-US/"  
    title="The best place to find more information about Mozilla's  
    mission and how to contribute">the Mozilla homepage</a>.  
</p>
```

```
<p>I'm creating a link to  
<a href="https://developer.mozilla.org/en-US/docs/Web#Scripting"  
    title="The best place to learn about scripting"> Scripting</a>.  
</p>
```

Images

- ▶ L'élément HTML permet de représenter une **image** dans un document.
- ▶ C'est un élément vide (ce qui signifie qu'il ne contient ni texte ni balise de fermeture)
- ▶ Exemple :
 - ▶ ``

Images

- ▶ L'attribut **src** est obligatoire et contient le chemin de l'image qu'on souhaite afficher.
- ▶ L'attribut **alt** n'est pas obligatoire mais recommandé et contient une description textuelle de l'image ; il est recommandé pour des raisons d'accessibilité et sera utilisé par les lecteurs d'écran ou sera affiché si l'image ne peut pas être chargée.
 - ▶ Cet attribut définit le texte alternatif utilisé lorsqu'il est impossible d'afficher l'image (par exemple si l'URL est incorrecte ou si l'image n'est pas encore téléchargée)
- ▶ Le standard HTML ne fournit pas de liste exhaustive des formats que doit prendre en charge un agent utilisateur et chaque agent utilisateur couvre différents formats.
- ▶ Un [guide à propos des formats d'image pris en charge par les navigateurs web](#) est disponible
- ▶ JPEG, GIF et PNG sont supportés par tous les navigateurs.



Images

- ▶ L'attribut **height** indique la hauteur intrinsèque de l'image exprimée en pixels
- ▶ L'attribut **width** indique la largeur intrinsèque de l'image, exprimée en pixels
 - ▶ seules les valeurs exprimées en pixels sont acceptées
- ▶ Ces attributs sont utilisés pour s'assurer que l'image occupe un espace stable, y compris lors du chargement

Vidéo

- Avant, une **vidéo** c'était ça :

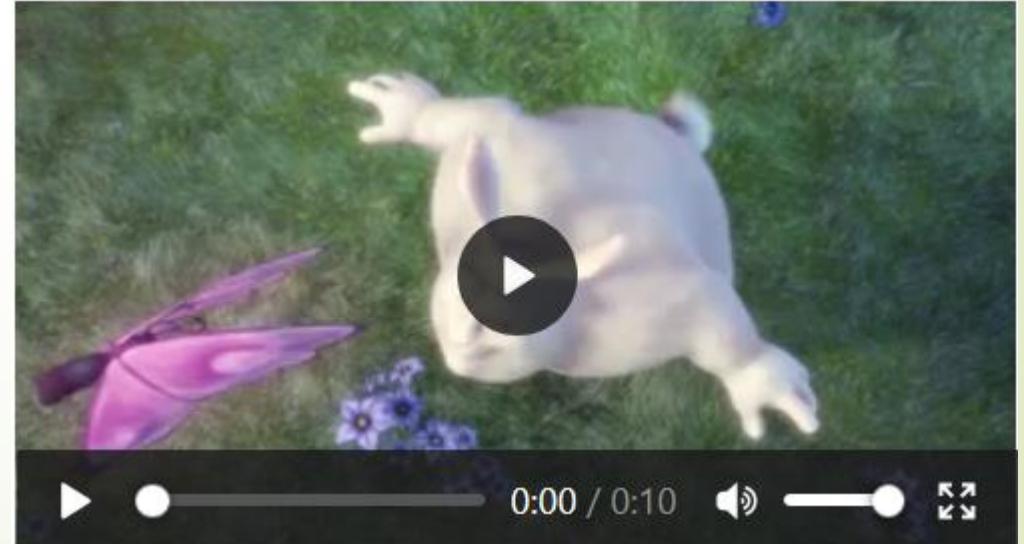
```
<object width="425" height="344">
  <param name="movie"
    value="http://www.youtube.com/v/9sEI1AUFKw&hl=en_GB&fs=1">
  </param>
  <param name="allowFullScreen" value="true"></param>
  <param name="allowscriptaccess" value="always"></param>

  <embed src="http://www.youtube.com/v/9sEI1AUFKw&hl=en_GB&fs=1"
    type="application/x-shockwave-flash"
    allowscriptaccess="always" allowfullscreen="true"
    width="425" height="344">
  </embed>
</object>
```

Vidéo : principe de base

- ▶

```
<video width="320" height="240" controls>
  <source src="bunny.mp4" type="video/mp4" />
  <source src="bunny.ogg" type="video/ogg" />
  Votre navigateur ne supporte pas le tag video.
</video>
```
- ▶ L'attribut **controls** ajoute les boutons de contrôle (play, stop, etc.)
- ▶ Le premier format supporté sera lu
- ▶ **<track>** via l'attribut **kind** lie un fichier de piste texte : sous-titres, légendes, audiodescription, chapitrage ou métainformations



<https://www.bigbuckbunny.org/>

Vidéo : les formats

- ▶ Il existe une grande bataille d'influence autour de cette balise :
 - ▶ Flash vs HTML5
 - ▶ WebM (Google) vs H.264 (Apple, Microsoft) vs Ogg (Firefox), etc.
 - ▶ Codecs, brevets, contrôles, etc.
- ▶ Une des solutions consiste à transcoder la vidéo dans un des formats supportés par le navigateur client

Navigateur	H.264/MP4	WebM	OGG Theora
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

Vidéo : les attributs

- ▶ **src** : sources de la vidéo
- ▶ **width** et **height** : dimension de la vidéo
 - ▶ Si non spécifiés, valent la largeur et la hauteur du fichier vidéo
 - ▶ Si une dimension est spécifiée mais pas l'autre, le navigateur va ajuster la taille de la dimension non spécifiée afin de préserver les proportions de la vidéo (homothétie)
- ▶ **poster** : permet de spécifier une image que le navigateur utilisera alors que la vidéo est en cours de téléchargement, ou jusqu'à ce que l'utilisateur commence la lecture de la vidéo
 - ▶ Si non spécifié, la première image de la vidéo sera utilisée à la place
- ▶ **controls** : si cet attribut booléen est présent, le navigateur affiche ses propres contrôles vidéo pour la lecture et le volume
 - ▶ Si non présent la première image est affichée (ou l'image poster spécifiée)
 - ▶ Lecture via événement JavaScript seulement.

Vidéo : les attributs

- ▶ **autoplay** : spécifie au navigateur de lancer la lecture de la vidéo automatiquement
- ▶ **autobuffer** : indique au navigateur de télécharger la vidéo tout de suite, en anticipant le fait que l'utilisateur lira la vidéo
 - ▶ Cette partie de la spécification est actuellement en pleine mutation et sujette à changement)
- ▶ **loop** : autre attribut booléen indiquant de lire la vidéo en boucle
 - ▶ Tous les attributs ne sont pas encore supportés par tous les navigateurs, mais les plus courants le sont

Audio : principe de base

- ▶ Similaire à la balise <video>
- ▶ <audio controls="controls">
 <source src="song.ogg" type="audio/ogg" />
 <source src="song.mp3" type="audio/mpeg" />
 Your browser does not support the audio element.
 </audio>
- ▶ Comme pour <video> l'attribut **controls** ajoute des boutons de contrôle



Audio : les formats

- ▶ Ici également, il existe une grosse bataille sur les formats car le format **mp3** n'est pas supporté partout
- ▶ Votre serveur doit donc pouvoir fournir plusieurs formats
- ▶ Une des solutions consiste à transcoder l'audio dans un des formats supportés par le navigateur client

Navigateur	MP3	WAV	OGG
Edge/IE	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

Attributs Universels

- ▶ Les **attributs universels** sont des attributs communs à l'ensemble des éléments HTML.
- ▶ Ces attributs peuvent donc être ajoutés sur tous les éléments (dans certains cas, les attributs n'auront aucun effet).
- ▶ **id**
 - ▶ Cet attribut définit un identifiant, unique au sein de tout le document, pour un élément.
 - ▶ Il doit permettre d'identifier un élément lorsqu'on crée un lien vers lui et/ou lorsque le manipule avec des scripts ou avec CSS.
- ▶ **class**
 - ▶ Une liste de classes, séparées par des espaces, qui permettent de catégoriser l'élément.
 - ▶ Les classes permettent au CSS et à JavaScript de manipuler des éléments spécifiques grâce à des sélecteurs de classe (pour CSS) ou grâce à des fonctions telles que **document.getElementsByClassName()** (pour JavaScript).

Attributs Universels

- ▶ **contenteditable**
 - ▶ Attribut à valeur contrainte qui indique si l'élément peut être édité par l'utilisateur (**true**) ou non (**false**)
- ▶ **data-***
 - ▶ Attributs de données personnalisés permettant d'échanger des informations dans un format propriétaire entre le HTML et le DOM afin de pouvoir les manipuler via des langage de scripts. La propriété **HTMLElement.dataset** permet d'accéder à l'ensemble des attribut définis de cette façon.
- ▶ **lang**
 - ▶ Aide à définir la langue utilisée pour l'élément.
- ▶ **title**
 - ▶ Attribut contenant une représentation textuelle de l'information auquel il est lié. Celle-ci est généralement affichée sous la forme d'une **bulle** d'informations.
- ▶ [Consulter la liste complètes des attributs universels ici](#)

Éléments en ligne vs Éléments de bloc

Éléments en ligne

- ▶ les éléments en ligne (*inline elements* en anglais) sont ceux qui occupent l'espace délimités par leurs balises plutôt que d'interrompre le flux du contenu.
- ▶ Un élément en ligne ne commence pas sur une nouvelle ligne et prend uniquement la largeur qui lui est nécessaire

Éléments de bloc

- ▶ ils sont séparés par un saut de ligne avant et après l'élément
- ▶ Un élément de bloc commence toujours sur une nouvelle ligne et prend toute la largeur disponible (autrement dit, il s'étend le plus possible vers la droite et vers la gauche).

<https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html#kinds-of-content>

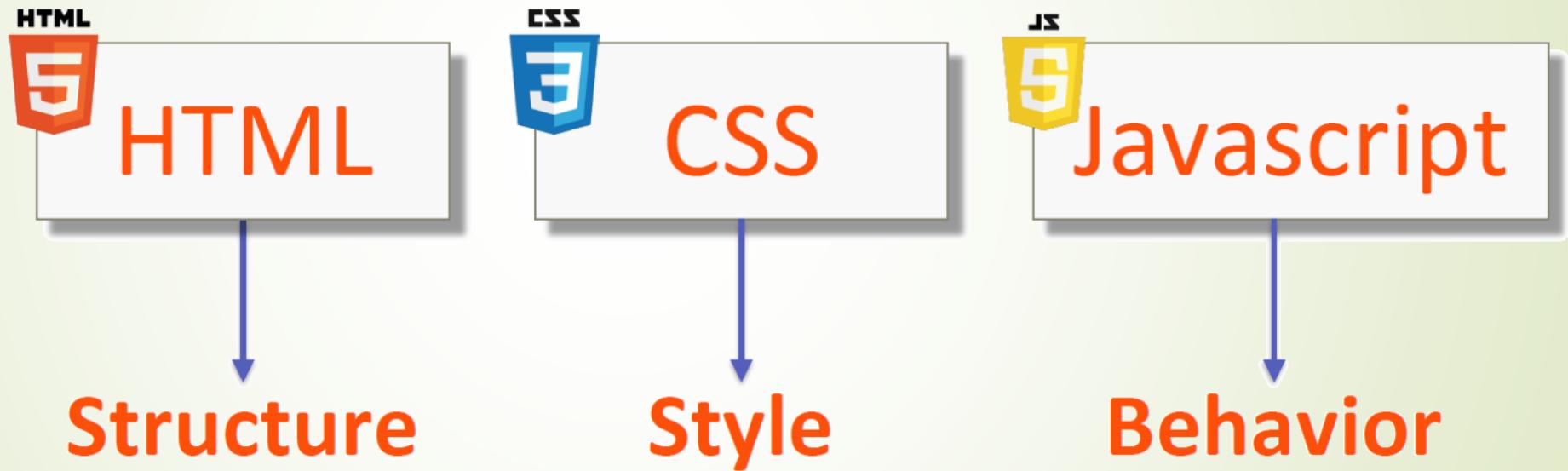
Validation des Documents HTML

- ▶ La page est toujours affichée même s'il y a des erreurs de syntaxe.
- ▶ Chaque Navigateur a des règles spécifiques pour interpréter les balises erronées
- ▶ Mais le résultat est imprévisible
- ▶ C'est la bonne pratique de vous assurer que vos pages sont en HTML légal
- ▶ <https://validator.w3.org/>

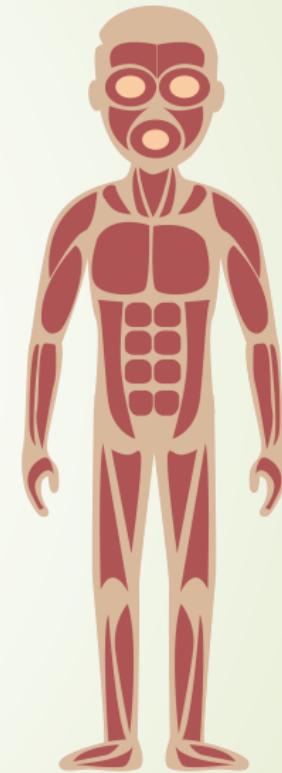
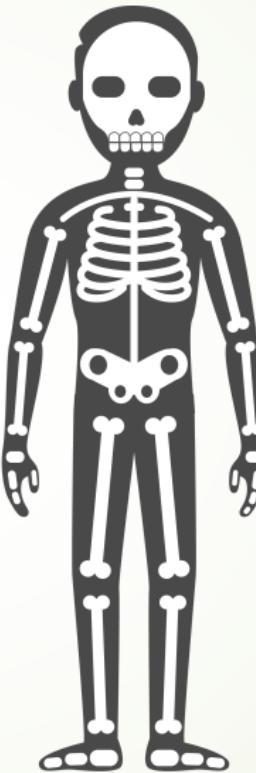
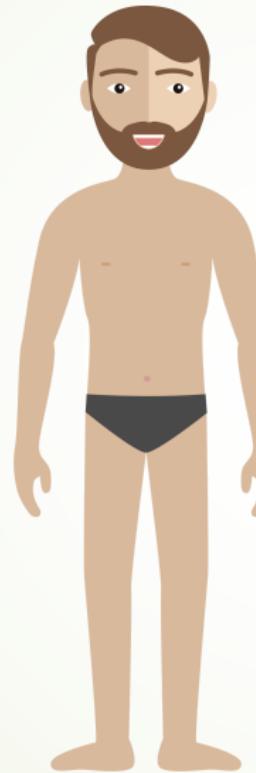


Comprendre la relation entre HTML et CSS

Technologies qui pilotent le web



Technologies qui pilotent le web



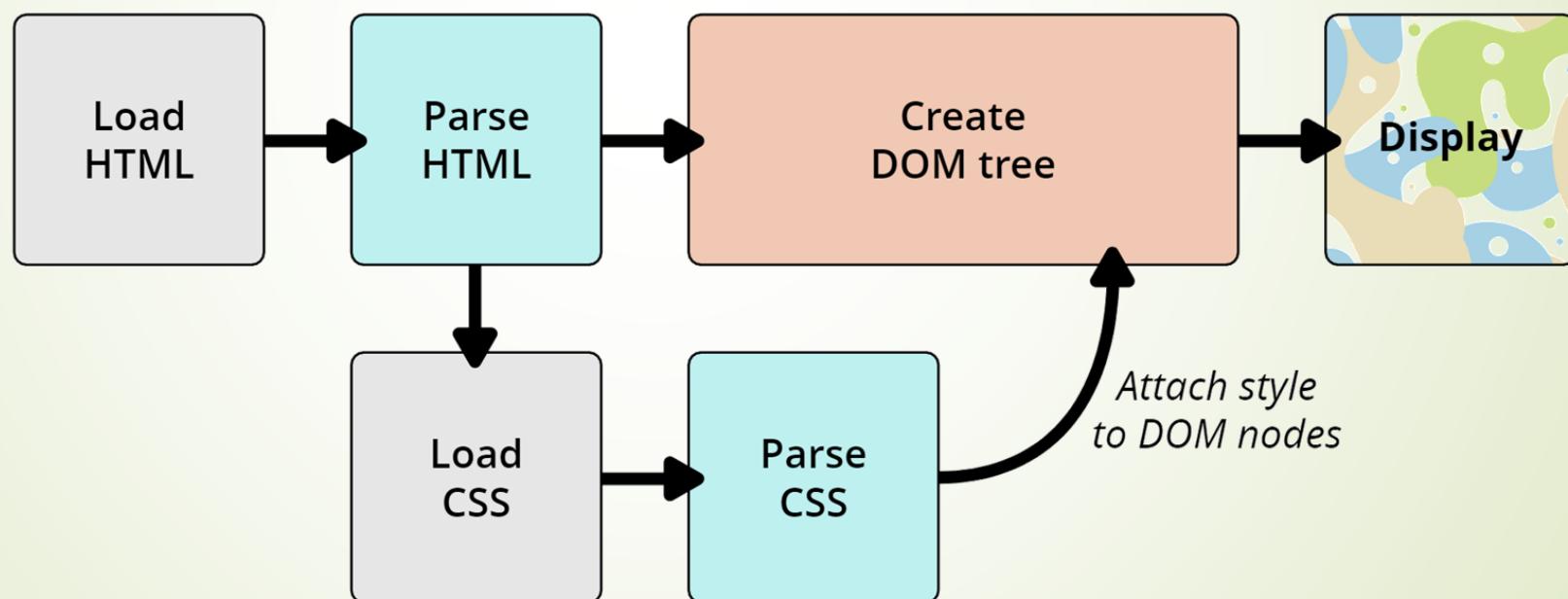


CSS

- ▶ Cascading Style Sheets en anglais, ou « feuilles de style en cascade »
- ▶ HTML définit la structure et sémantique du contenu d'une page web
- ▶ CSS est utilisé pour mettre en forme une page web
- ▶ CSS est utilisé pour modifier les polices, la couleur, la taille et l'espacement de votre contenu, pour le répartir sur plusieurs colonnes ou bien pour ajouter des animations et autres fonctionnalités décoratives

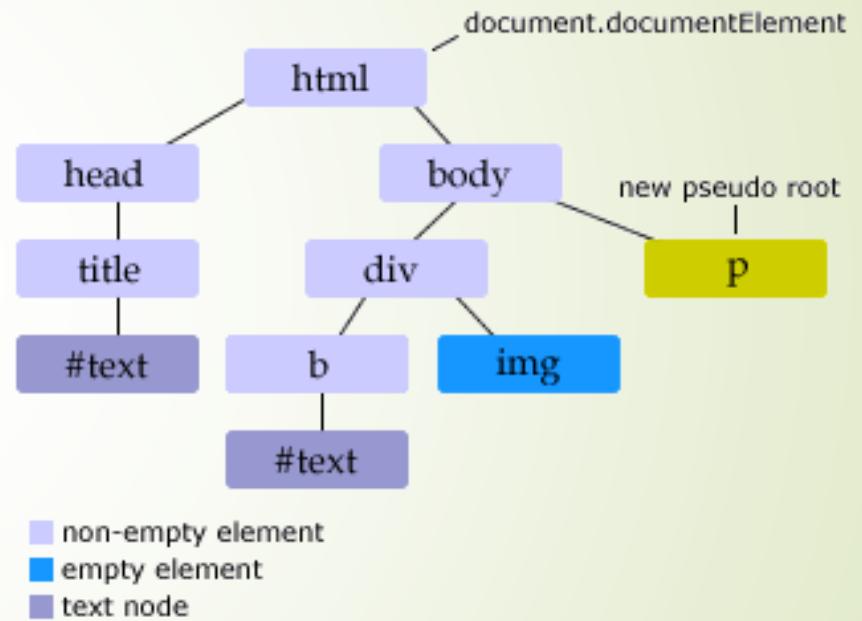
CSS comment ça marche ?

- ▶ Pour afficher un document, un navigateur doit combiner le contenu du document et les informations de mise en forme



Fonctionnement du navigateur

- ▶ Le navigateur lit le document
- ▶ Il en fait l'analyse syntaxique et produit un arbre syntaxique
- ▶ Il lit la feuille de style associée
- ▶ Il interprète les règles de formatage
- ▶ Il parcourt l'arbre et applique à chaque élément sa règle de formatage.
- ▶ En cas de conflit de style, il applique des règles de priorité



Représentation DOM

```
<p>
  Let's use:
  <span>Cascading</span>
  <span>Style</span>
  <span>Sheets</span>
</p>
```

```
P
└ "Let's use:"
  └ SPAN
    └ "Cascading"
  └ SPAN
    └ "Style"
  └ SPAN
    └ "Sheets"
```

Exemple

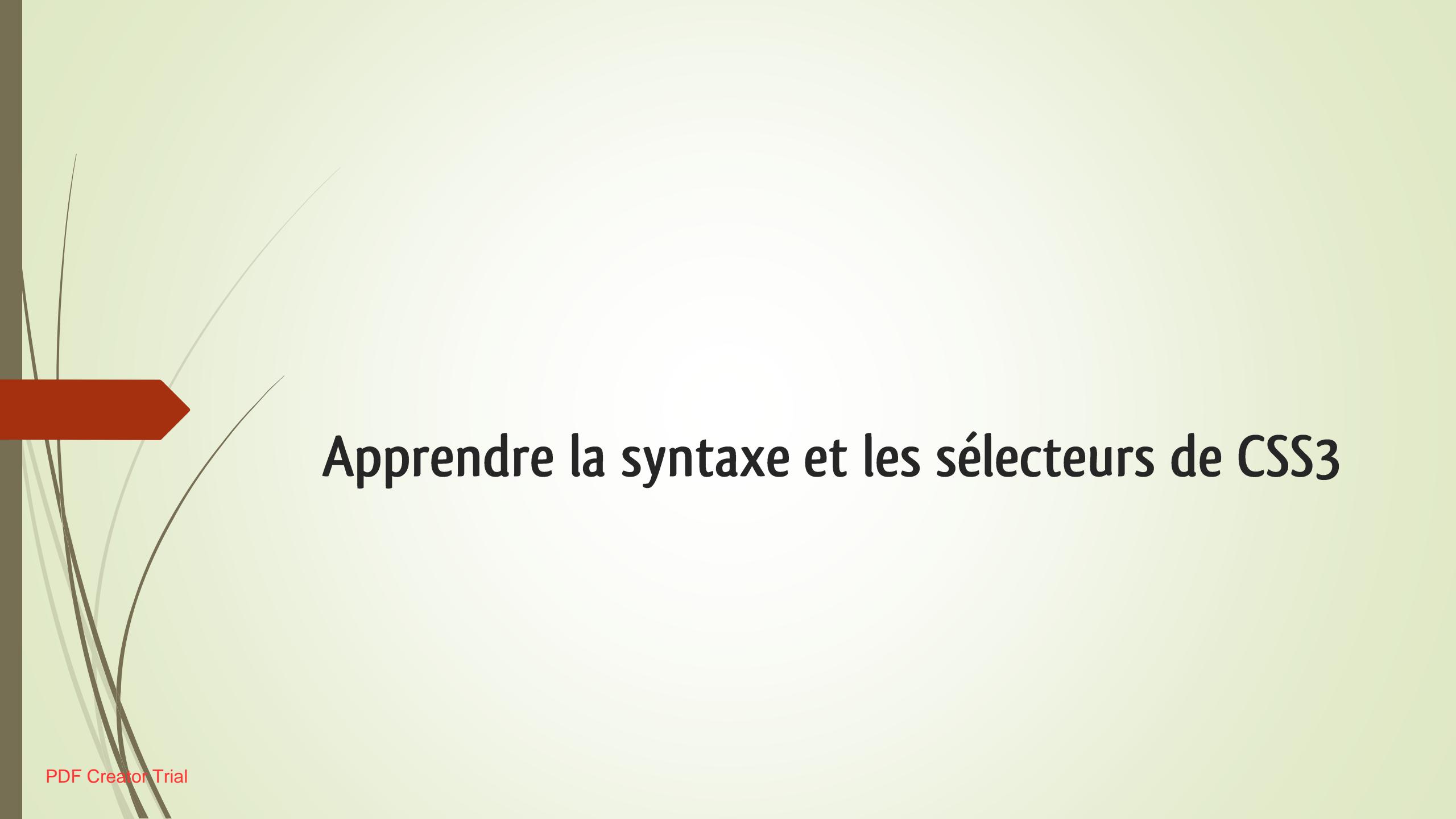
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS
experiment</title>
    <link rel="stylesheet"
href="style.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS
example</p>
  </body>
</html>
```



```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

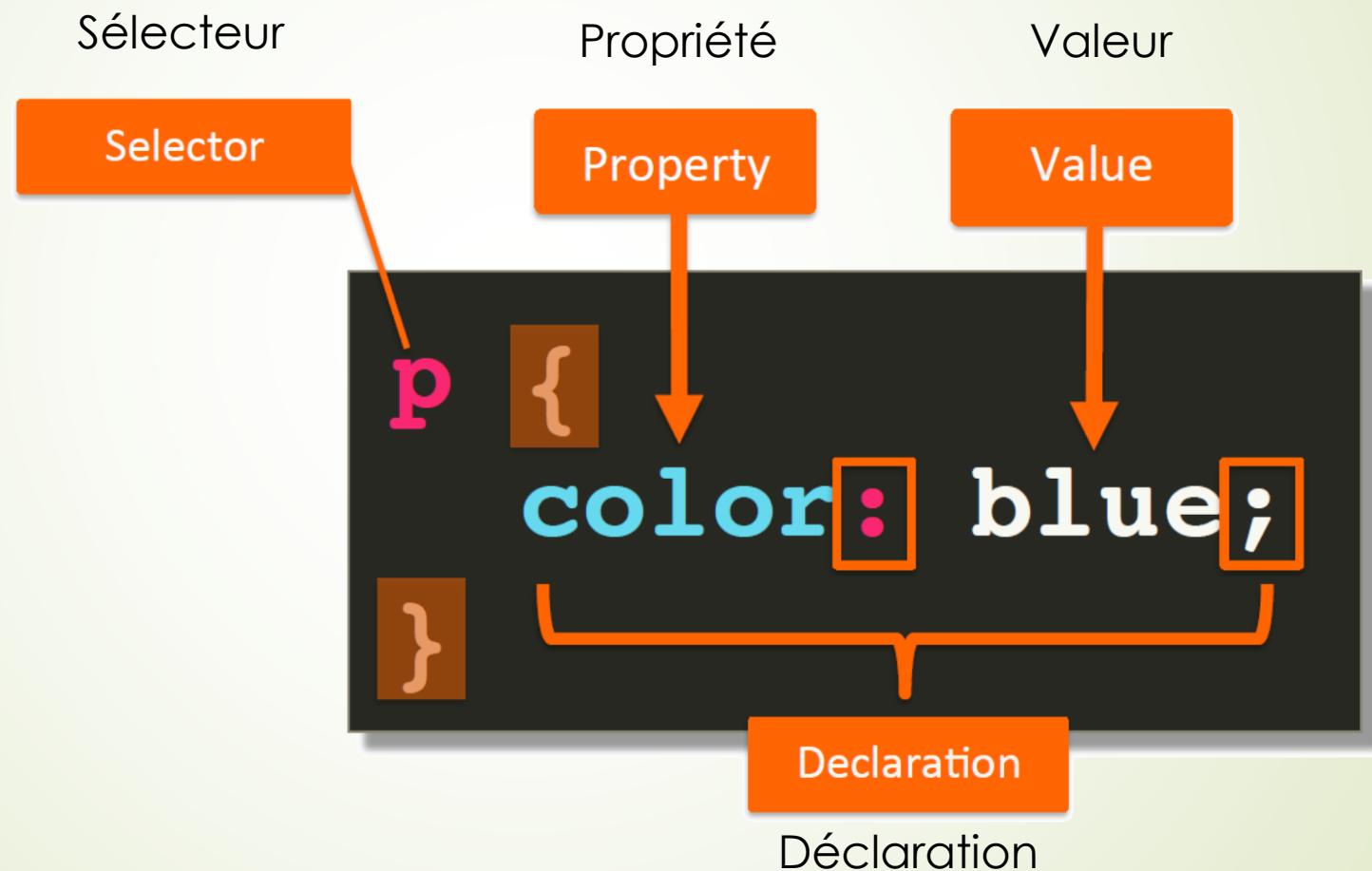
p {
  color: red;
}
```





Apprendre la syntaxe et les sélecteurs de CSS3

Anatomie d'une règle CSS



Anatomie d'une règle CSS

► Sélecteur

- ▶ C'est le nom de l'élément HTML situé au début de l'ensemble de règles.
- ▶ Il permet de sélectionner les éléments sur lesquels appliquer le style souhaité (en l'occurrence tous les **éléments p**).
- ▶ Pour mettre en forme un élément différent, il suffit de changer le sélecteur.

► Déclaration

- ▶ C'est une règle simple comme **color: blue;** qui détermine les propriétés de l'élément que l'on veut mettre en forme.

► Propriétés

- ▶ Les différentes façons dont on peut mettre en forme un élément HTML (dans ce cas, **color** est une propriété des éléments **p**).
- ▶ En CSS, vous choisissez les différentes propriétés que vous voulez utiliser dans une règle CSS.

Anatomie d'une règle CSS

► Valeur de la propriété

- ▶ À droite de la propriété, après les deux points, on a la valeur de la propriété.
- ▶ Celle-ci permet de choisir une mise en forme parmi d'autres pour une propriété donnée (par exemple, il y a d'autres couleurs que red pour la propriété color).

Zéro ou plusieurs déclarations autorisées

Zero or More Declarations are allowed

```
p {  
    color: blue;  
    font-size: 20px;  
    width: 200px;  
}
```

Anatomie d'une règle CSS

```
p {  
    color: blue;  
    font-size: 20px;  
    width: 200px;  
}  
  
h1 {  
    color: green;  
    font-size: 36px;  
    text-align: center;  
}  
...
```

Feuille de style

Stylesheet

Sélecteur d'élément

Nom d'élément

Element name

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

Sélecteur d'élément

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

Texte bleu

Blue text

```
...  
<p>...</p>  
...  
<p>...</p>
```

Blue text

Texte bleu

La classe "class" du HTML

- ▶ Class est un attribut "générique" qui s'applique à toutes sortes d'éléments.
- ▶ Une classe permet de définir un sous-ensemble.
- ▶ Elle peut s'appliquer à plusieurs éléments.
- ▶ Elle s'ajoute dans une balise du document html :

```
<h1 class="intro">titre 1</p>  
<p class="intro">texte 2</p>  
<p class="intro">texte 3</p>  
<img class="logo" />
```

Sélecteur de classe

Nom de classe

Class name

```
.blue {  
    color: blue;  
}
```

Sélecteur de classe

```
.blue {  
    color: blue;  
}
```

```
...  
<p class="blue">...</p>  
<p>...</p>  
<div class="blue">...</div>  
...
```

Unaffected

Texte bleu

Blue text

Blue text

Texte bleu

Cet élément n'est pas affecté

Sélecteur de classe

Définit avec .

Defined with .

```
.blue {  
    color: blue;  
}
```

Utilisé sans .

Used without .

```
...  
<p class="blue">...</p>  
<p>...</p>  
<div class="blue">...</div>  
...
```

Texte bleu

Blue text

Unaffected

Blue text

Texte bleu

Cet élément n'est pas affecté

L'identifiant id du HTML

- ▶ id est un attribut "générique" qui s'applique à toutes sortes d'éléments.
- ▶ Il sert à identifier une balise précise.
- ▶ Il doit être unique dans un document.
- ▶ Il s'ajoute dans une balise du document html :

```
<p id="intro">texte d'introduction</p>
```

Sélecteur d'id

Valeur d'id

id Value

```
#name {  
    color: blue;  
}
```

Sélecteur d'id

```
#name {  
    color: blue;  
}
```

Cet élément n'est pas affecté

Unaffected

```
...  
<p>...</p>  
<div id="name">...</div>  
...
```

Blue text

Texte bleu

Sélecteur d'id

Défini avec #

Defined with #

```
#name {  
    color: blue;  
}
```

Cet élément n'est pas affecté

Unaffected

```
...  
<p>...</p>  
<div id="name">...</div>  
...
```

Used without

Utilisé sans #

Blue text

Texte bleu

Sélecteur de groupe

Séparez sélecteurs avec les virgules

Separate selectors
with commas

```
div, .blue {  
    color: blue;  
}
```

Texte bleu

Blue text

```
...  
<p class="blue">...</p>  
<p>...</p>  
<div>...</div>  
...
```

Blue text

Texte bleu

Élément avec Sélecteur de classe

Chaque **p** qui a la **class="big"**

Every **p** that has **class="big"**

```
p.big {  
    font-size: 20px;  
}
```

NOTE lack of space between element and class definition

NOTE : Il n'y a pas d'espace entre l'élément et la définition de la classe

Élément avec Sélecteur de classe

```
p.big {  
    font-size: 20px;  
}
```

```
...  
<p class="big">...</p>  
<div class="big">...</div>  
...
```

Taille du texte 20 px

Text size 20px

Unaffected text

Cet élément n'est pas affecté

Sélecteur d'enfant

Chaque **p** qui est un enfant direct d'**article**

Every **p** that is a direct child of **article**

```
article > p {  
    color: blue;  
}
```

Sélecteur d'enfant

```
article > p {  
    color: blue;  
}
```

```
<article>...  
  <p>...</p>  
</article>  
...  
<p>...</p>  
<article>...  
  <div><p>...</p></div>  
</article>
```

Texte bleu

Blue text

Unaffected text

Cet élément
n'est pas
affecté

Unaffected text

Cet élément
n'est pas
affecté

Sélecteur de descendant

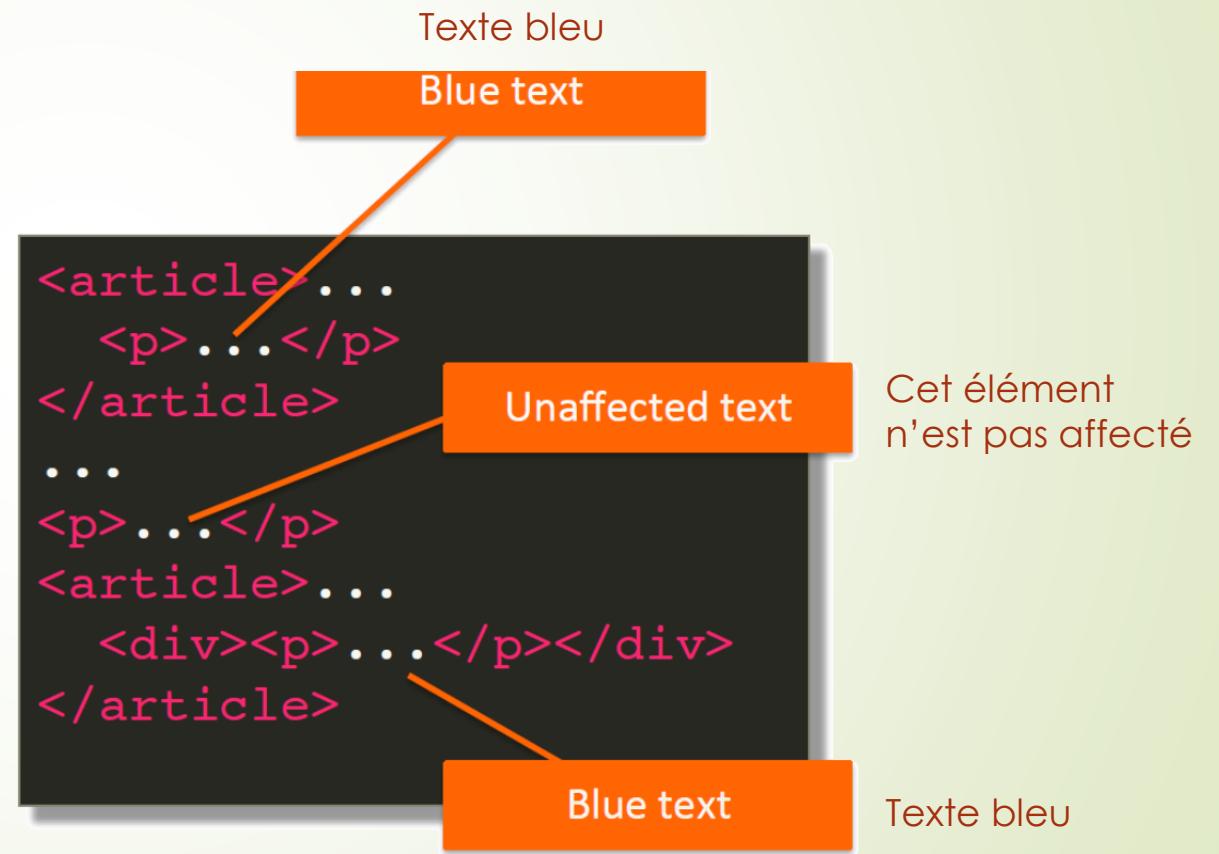
Chaque **p** qui est dans **article** (quelque soit le niveau)

Every **p** that is inside (at any level) of **article**

```
article p {  
    color: blue;  
}
```

Sélecteur de descendant

```
article p {  
    color: blue;  
}
```



Autres combinateurs

- ▶ Les combinateurs sont des sélecteurs qui établissent une relation entre deux sélecteurs ou plus, tel que "A est un enfant de B" ou "A est adjacent à B".
- ▶ **Combinateur de voisin direct A + B**
 - ▶ Indique que les éléments sélectionnés par A et par B ont le même parent et que celui sélectionné par B suit immédiatement celui sélectionné par A.
- ▶ **Combinateur de voisin général A ~ B**
 - ▶ Indique que les éléments sélectionnés par A et par B ont le même parent et que celui sélectionné par B suit celui sélectionné par A, mais pas nécessairement immédiatement après lui.

N'est pas limité au sélecteur d'élément

```
.colored p {  
    color: blue;  
}
```

Every **p** that is inside (at any level) an element with **class="colored"**

Chaque **p** qui est dans l'élément avec **class="colored"** (quelque soit le niveau)

```
article > .colored {  
    color: blue;  
}
```

Every element with **class="colored"** that is a direct child of **article** element

Chaque élément avec **class="colored"** qui est un enfant direct d'**article**



Sélecteur de Pseudo-Classe

```
selector:pseudo-class {  
    ...  
}
```

Sélecteur de Pseudo-Classe

- ▶ Les éléments donnés mais uniquement dans un certain état - par exemple quand on passe la souris dessus :
 - **:link**
 - **:visited**
 - **:hover**
 - **:active**
 - **:nth-child(...)**

Sélecteur de Pseudo-Classe d'ancre

- ▶ **:link**
 - ▶ permet de sélectionner les liens à l'intérieur d'éléments.
 - ▶ Elle sélectionnera tout lien n'ayant pas été visité, même ceux qui seraient déjà mis en forme via des sélecteurs utilisant d'autres pseudo-classes comme :hover, :active ou :visited.
- ▶ **:visited**
 - ▶ permet de modifier l'aspect d'un lien après que l'utilisateur l'a visité.
- ▶ **:hover**
 - ▶ permet de spécifier l'apparence d'un élément au moment où l'utilisateur le survole avec le pointeur, sans nécessairement l'activer.
- ▶ **:active**
 - ▶ permet de cibler un élément lorsque celui-ci est activé par l'utilisateur.

Sélecteur de Pseudo-Classe

- ▶ **:nth-child(an+b)**
 - ▶ Permet de cibler les éléments fils d'un élément dont les positions correspondent au motif $an+b$
- ▶ **:first-child**
 - ▶ permet de cibler un élément qui est le premier élément fils par rapport à son élément parent.
- ▶ **:last-child**
 - ▶ permet de cibler un élément qui est le dernier enfant de son parent.
- ▶ [Consulter la référence des sélecteurs CSS ici](#)

Sélecteur de Pseudo-éléments

- ici appliqués à l'élément p

`p::first-line {font-variant: small-caps;}` → 1re ligne des paragraphes

`p::first-letter {font-size:2em;color:red;}` → 1re lettre des paragraphes

`::before` permet d'insérer un texte avant

`::after` permet d'insérer un texte après

► Exemples

```
td::before { content:"Prix: "; }
```

12,50



```
td::after { content:" euros"; }
```

Prix: 12,50 euros

```
th::before { content:url(carre.gif)" ";
```

Produits vendus

Sélecteur universel *

- ▶ * remplace tout élément du document
- ▶ Se met dans la partie style CSS, par exemple :
`* {margin: 0}`
- ▶ Tous les éléments du document (h1, h2, p, div, ul...) auront une marge de 0.

Sélecteurs d'attribut

Exemples

- ▶ `a[title]`
 - ▶ Les éléments `<a>` avec un attribut `title`
- ▶ `a[href="https://example.org"]`
 - ▶ Les éléments `<a>` avec un `href` qui correspond à <https://example.org>
- ▶ `a[href*="example"]`
 - ▶ Les éléments `<a>` dont `href` contient "example"
- ▶ `a[href$=".org"]`
 - ▶ Les éléments `<a>` dont `href` finit par ".org"

/* Utilisation des commentaires en CSS */

```
/* Auteur : Soupramanien BOUVANESVARY */  
a { color:#333; /* gris sombre */  
}
```

Commentaire conditionnel

- ▶ Si le navigateur est inférieur (**lt** pour less than) à la version d'IE, alors on ajoute un style...

```
<!--[if lt IE 7]>  
  <style type="text/css">  
    div {  
      width:expression(document.body.clientWidth >= 1000? "1000px": "auto" );  
    }  
  </style>  
<![endif]-->
```

Codage des couleurs en CSS

Propriétés CSS pour les couleurs :

- ▶ `background-color` : couleur de l'arrière-plan
- ▶ `border-color` : couleur de la bordure
- ▶ `color` : couleur de l'avant plan (texte, par exemple)

Il existe plusieurs manières de spécifier une couleur en CSS :

- ▶ La plus classique est le code de couleur **Hexadécimal** - `color : #FFB6C1`
- ▶ La moins classique est le code couleur **nommé** (16 sont reconnues par le W3C mais tous les navigateurs actuels reconnaissent les couleurs définies par la norme X11) - `color : CornflowerBlue`
- ▶ Le code couleur **RGB** (Red Green Blue ou Rouge Vert Bleu) - `color : rgb(255,182,193)`
- ▶ Le code couleur **HSL** (Hue Saturation Lightness ou Teinte Saturation Luminosité) - `color : hsl(351,100%,86%)`
- ▶ Il est maintenant aussi possible depuis CSS3 de spécifier une couleur incluant la transparence avec les fonctions **RGBA**, **HSLA** et bientôt `#RRGGBBAA`

Codage des couleurs en CSS

Les notations de type RGB :

- ▶ Pour créer une couleur RGB on doit préciser trois niveaux d'intensité de Rouge, de Vert, et de Bleu qui vont ensuite être mélangés pour créer la couleur finale
- ▶ Chaque niveau d'intensité qu'on va renseigner va être compris entre 0 (intensité nulle ou absence de la couleur en question) et 255 (intensité maximale ou couleur pure).
- ▶ Exemples :
 - ▶ Codage décimal `rgb(x, y, z)` avec nombre de 0 à 255 : `rgb(255, 0, 122)`
 - ▶ Codage décimal `rgb(x%, y%, z%)` en pourcentage : `rgb(100%, 0%, 50%)`
 - ▶ Codage décimal `rgba(x, y, z)` avec nombre de 0 à 255 et transparence : `rgb(255, 0, 122, 0.5)`

Codage des couleurs en CSS

Les valeurs de type hexadécimales : # **XX XX XX**

- ▶ Elles vont reposer sur le même principe que les valeurs de type RGB : préciser trois niveaux d'intensité de rouge, de vert et de bleu pour créer une couleur personnalisée.
- ▶ La seule différence va être la façon dont on va compter. Le système hexadécimal utilise 16 symboles pour compter et représenter les chiffres.

Décimal	0 (ou 00)	1 (ou 01)	2 (ou 02)	9 (ou 09)	10	11	15	16	17	...	254	255
Hexadécimal	0 (ou 00)	1 (ou 01)	2 (ou 02)	9 (ou 09)	A (ou 0A)	B (ou 0B)	F (ou 0F)	10	11	...	FE	FF

- ▶ Exemples :
 - ▶ Codage hexadécimal classique de type #rrggb : #cc0088
 - ▶ Codage hexadécimal abrégé de type #rgb : #c08

#000000

#454545

#999999

#FF0000

#00FF00

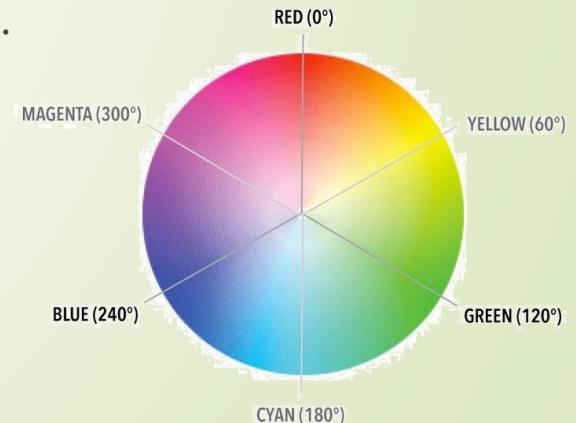
#0000FF

#FFFFFF

Codage des couleurs en CSS

Les notations de type HSL :

- ▶ Pour créer une couleur en utilisant les notations HSL, il faut renseigner trois valeurs :
 - ▶ **La teinte** : nombre entre 0 et 360 qui représente un angle du cercle chromatique. Le rouge correspond à un angle de 0 (degré) ou de 360 (degrés), tandis que le vert va se situer à 120 degrés et le bleu à 240 degrés.
 - ▶ **La saturation** : sous forme de pourcentage. 100% correspond à une saturation maximale tandis que 0% correspond à une saturation minimale.
 - ▶ **La luminosité** : également représentée par un pourcentage. 100% de luminosité correspond à du blanc tandis que 0% correspond à du noir.
- ▶ Exemples :
 - ▶ Teinte = 45° (à mi-chemin entre le orange et le jaune) / Saturation à 100% et luminosité à 60% : `hsl(45, 100%, 60%)`
 - ▶ Teinte à 150° (dans les vert), saturation à 50% et luminosité à 50% : `hsl(150, 50%, 50%)`



Codage des couleurs en CSS

16 couleurs de base

- A l'époque, la plupart des écrans d'ordinateurs avait encore des capacités limitées et ne pouvaient reproduire qu'un jeu réduit de couleurs

#00FFFF	#000000	#0000FF	#FF00FF
Aqua	Black	Blue	Fuchsia
#000080	#808000	#800080	#FF0000
Navy	Olive	Purple	Red
#808080	#008000	#00FF00	#800000
Gray	Green	Lime	Maroon
#COCOCO	#008080	#FFFFFF	#FFFF00
Silver	Teal	White	Yellow

Les couleurs X11

- La norme X11 fut définie pour gérer les terminaux à l'époque des mini-ordinateurs. Elle comporte un nombre relativement étendu de couleurs qui sont reconnues par les principaux navigateurs actuels.

Nom de couleur	Exemple	HTML	Composantes (rouge, vert, bleu)			Décimal
			Hexadécimal			
AliceBlue		#F0F8FF	F0	F8	FF	240 248 255
AntiqueWhite		#FAEBD7	FA	EB	D7	250 235 215
Aqua		#00FFFF	00	FF	FF	0 255 255
Aquamarine		#7FFFDD	7F	FF	D4	127 255 212
Azure		#FOFFFF	F0	FF	FF	240 255 255
Beige		#F5F5DC	F5	F5	DC	245 245 220
Bisque		#FFE4C4	FF	E4	C4	255 228 196

Unités absolues

Unité	Nom	Équivalence
cm	Centimètre	$1\text{cm} = 96\text{px}/2.54$
mm	Millimètre	$1\text{mm} = 1/10\text{e}$ de 1cm
Q	Quart de millimètre	$1\text{Q} = 1/40\text{e}$ de 1cm
in	Pouces (inches)	$1\text{in} = 2.54\text{cm} = 96\text{px}$
pc	Picas	$1\text{pc} = 1/16\text{e}$ de 1in
pt	Points	$1\text{pt} = 1/72\text{e}$ de 1in
px	Pixels	$1\text{px} = 1/96\text{e}$ de 1in

Unités relatives

Unité	Relative à
em	La taille (corps) de police de l'élément
ex	La hauteur d'un x avec la police utilisée par l'élément
rem	La taille (corps) de police de l'élément racine
%	parent
vw	1% de la largeur de la zone d'affichage (viewport)
vh	1% de la hauteur de la zone d'affichage (viewport)
vi	1% de la taille de la zone d'affichage sur l'axe en ligne (inline axis)
vb	1% de la taille de la zone d'affichage sur l'axe de bloc (block axis)
vmin	1% de la zone d'affichage selon sa plus petite dimension
vmax	1% de la zone d'affichage selon sa plus grande dimension

Remarques

- ▶ L'abréviation de l'unité doit être collée à la valeur, sans espace
- ▶ La taille de police est exprimée le plus souvent en pt, en em ou en %
- ▶ Largeur et hauteur d'une image sont exprimées en général en px
- ▶ Les tailles pour l'impression (marges...) sont exprimées en cm ou en in

Propriétés de police de caractères

- ▶ color
- ▶ background-color
- ▶ font-style: normal | italic | oblique
- ▶ font-variant: normal | small-caps
- ▶ font-weight : normal | bold | bolder | lighter | 100 | 700 | 900
- ▶ font-size : n | p% | xx-small | small | smaller | large | larger | xx-large
- ▶ line-height: n p% (interligne entre 2 lignes)
- ▶ font-family: police1, police2, police3
- ▶ **Exemple de propriété raccourcie (ordre important) :**
font: italic small-caps bold 10pt/14pt Times

Propriété font-family

- ▶ **Liste de polices** (pour différents OS + police générique en dernier)
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
font-family: Georgia, "Bookman Old Style", serif;
- ▶ **5 familles de police génériques**
 - ▶ serif (ex. Times, Times new roman, Palatino, Georgia)
 - ▶ sans-serif (ex. Geneva, Arial, Helvetica, Tahoma, Verdana)
 - ▶ cursive (ex. Zapf-Chancery, Comic Sans MS)
 - ▶ fantasy (ex. Western, Impact)
 - ▶ monospace ou à taille fixe (ex. Courier)
- ▶ **Nom de police avec espace**

font-family: Times, "Times New Roman", serif
font-family: Times, 'Times New Roman', serif

Règle font-face

- ▶ Permet de définir les polices d'écriture à utiliser pour afficher le texte de pages web.
- ▶ Elle peut être chargée depuis un serveur distant **url()** ou depuis l'ordinateur de l'utilisateur **local()**.
- ▶ Plusieurs types de polices :
 - ▶ Embedded OpenType (EOT)
 - ▶ True Type (TTF)
 - ▶ Open Type (OTF)
 - ▶ Web Open Font (WOFF)
 - ▶ SVG sur iPad et iPhone
- ▶ Accès via CDN :
 - ▶ <https://fonts.googleapis.com/>



Propriétés de texte

- ▶ word-spacing: n | -n
- ▶ letter-spacing: n | -n
- ▶ text-decoration: none | underline | overline | blink | line-through
- ▶ text-transform: uppercase | lowercase | capitalize
- ▶ text-align: left | right | center | justify
- ▶ text-indent (retrait de première ligne) : n p%
- ▶ white-space : normal | pre | nowrap

Propriétés de blocs : marges

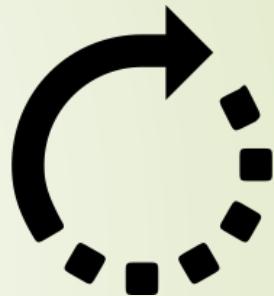
- ▶ margin-top
- ▶ margin-right
- ▶ margin-bottom
- ▶ margin-left

valeurs possibles :

n p% -n -n% auto

exemples :

5pt -12px 2em 10%



▶ Exemples de propriété raccourcie :

On part d'en haut, on tourne dans le sens des aiguilles d'une montre

margin: 5em 10em 5em 10em

margin: 5em --> 5em pour toutes les marges

margin: 5em 10em --> 5em haut/bas 10em droit/gauche

margin: 5em 3em 2em (=droit) --> 5em haut 3em droit 2em bas 3em gauche

Propriétés de blocs : remplissage

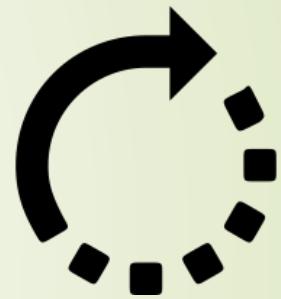
- ▶ padding-top: n p%
- ▶ padding-right
- ▶ padding-bottom
- ▶ padding-left
- ▶ padding

valeurs possibles :

n p% -n -n% auto

exemples :

5pt -12px 2em 10%



▶ Exemples de propriété raccourcie :

on part d'en haut, on tourne dans le sens des aiguilles d'une montre

padding: 5em 10em 5em 10em

padding: 5em --> 5em pour toutes

padding: 5em 10em --> 5em haut/bas 10em droit/gauche

padding: 5em 3em 2em --> 5em haut 3em droit 2em bas 3em gauche

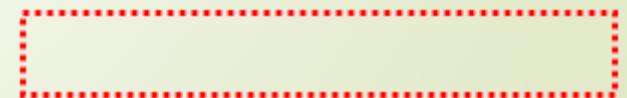
Propriétés de blocs : bordures

- ▶ border-top-width: n | thin | medium | thick (idem avec bottom)
- ▶ border-left-width: n | thin | medium | thick (idem avec right)
- ▶ border-width: 2pt 9pt 9pt 2pt

- ▶ border-color: red blue green black
- ▶ pas de couleur précisée : color de l'élément

- ▶ border-style: none | solid | dotted | dashed | ...
- ▶ border-style: double solid

- ▶ border-top: 2px dashed red
- ▶ border: 2px dotted red --> 4 bordures identiques



Propriétés de blocs : coins arrondis

- ▶ La propriété **border-radius** permet de définir des coins arrondis pour la bordure d'un élément.
- ▶ La courbure de chaque coin est définie avec un ou deux rayons de courbures qui permettent de définir un arc de cercle ou un arc d'ellipse.
- ▶ Exemples :
 - ▶ border-radius: 30px;
 - ▶ **border-radius: 25% 10%;**
 - ▶ border-radius: 10% 30% 50% 70%;
 - ▶ border-radius: 10% / 50%;
 - ▶ border-radius: 10px 100px / 120px;
 - ▶ border-radius: 50% 20% / 10% 40%;

This is a box with rounded corners.

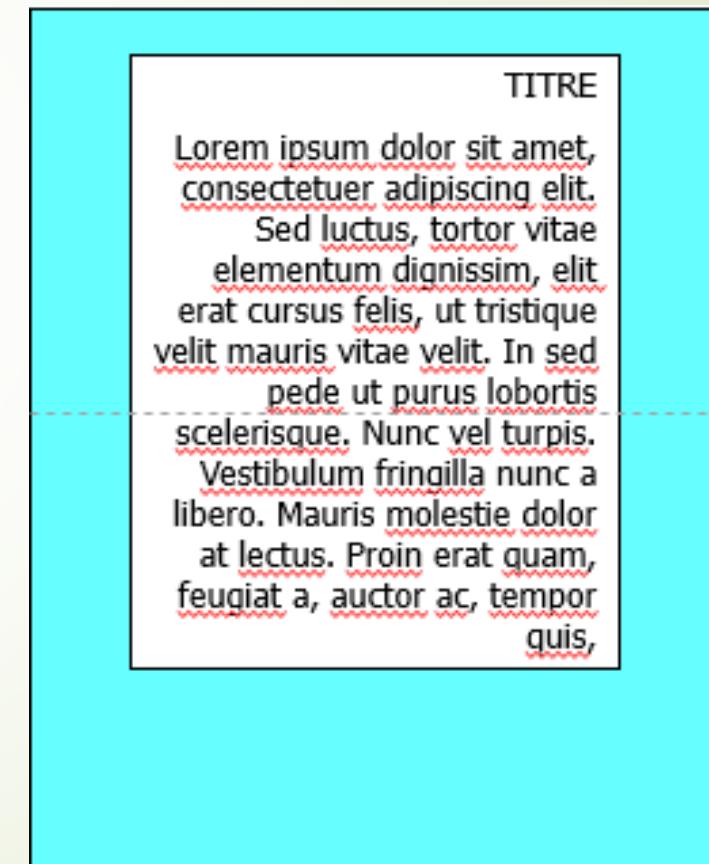
Utilisation de marges automatiques pour centrer un bloc

- ▶ Dans la partie **CSS**

```
#conteneur {  
    width: 600px;  
    background-color:cyan;  
    margin-left: auto;  
    margin-right: auto;  
    padding: 1em;  
    text-align:right;  
}
```

- ▶ en **HTML**

```
<body>  
<div id="conteneur">...  
</div>  
</body>
```



Propriétés d'arrière-plan

- ▶ Background-attachment : scroll | fixed
- ▶ Background-position : 50% 50% (ou 20px 10px)
d'abord verticale puis horizontale
- ▶ Background-color : #ccc
- ▶ Background-image : url(logo.gif) | gradient
- ▶ Background-repeat : repeat | repeat-x | repeat-y | no-repeat

Propriété raccourcie :

- ▶ Background: scroll 50% 50 % #ccc url(logo.gif) no-repeat

Background-attachment

haut de l'écran

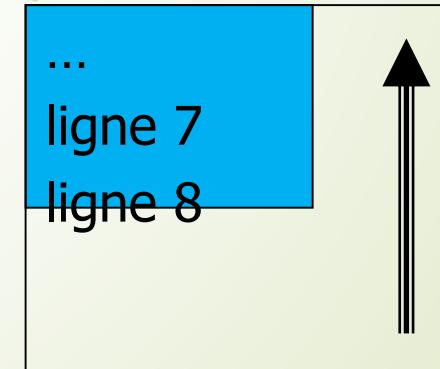


fixed



*bas de l'écran
après défilement*

par défaut : scroll



Background-repeat

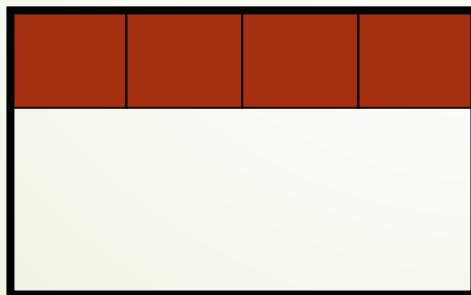
no-repeat



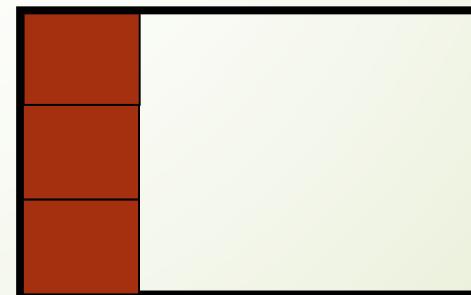
par défaut : repeat



repeat-x



repeat-y



Background-position

background-position: x y (valeur négative possible)

par
défaut

0% 0% top left left top	50% 0% top center center top top	100% 0% top right right top
0% 50% left center center left left	50% 50% center center center	100% 50% right center center right right
0% 100% bottom left left bottom	50% 100% bottom center center bottom bottom	100% 100% bottom right right bottom

Background-image

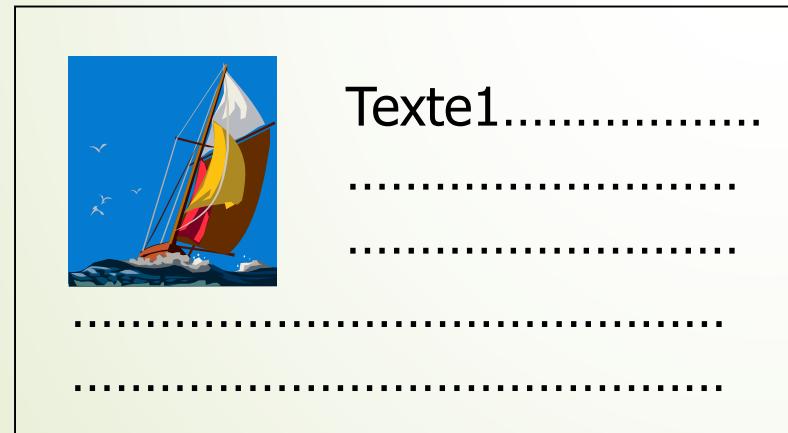
- ▶ Les fonctions **linear-gradient()** et **radial-gradient()** permettent de créer une image représentant un dégradé de couleur linéaire.
- ▶ Exemples :
 - ▶ **linear-gradient(#e66465, #9198e5);**
 - ▶ linear-gradient(red, orange, yellow, green, blue);
 - ▶ linear-gradient(red 0%, orange 25%, yellow 50%, green 75%, blue 100%);
 - ▶ linear-gradient(to left top, blue, red);
 - ▶ **radial-gradient(#e66465, #9198e5);**
 - ▶ radial-gradient(circle at center, red 0, blue, green 100%)



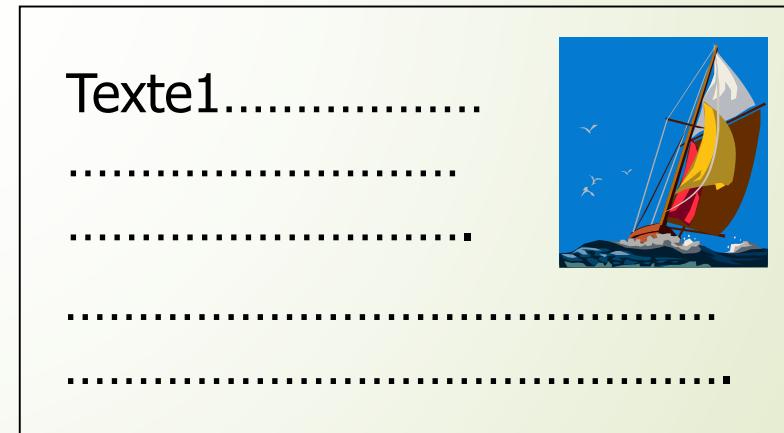
Propriété Float (valeur : right | left)

- float: left | right
- Utilisation : image, menu, bloc de textes, lettrine...
- **Exemple :** <p>Texte1... </p>

► `img {float: left;}`



► `img {float: right;}`



Mise en page d'images et de blocs avec "Float"

► Galerie d'images

```
img {float:left}
```

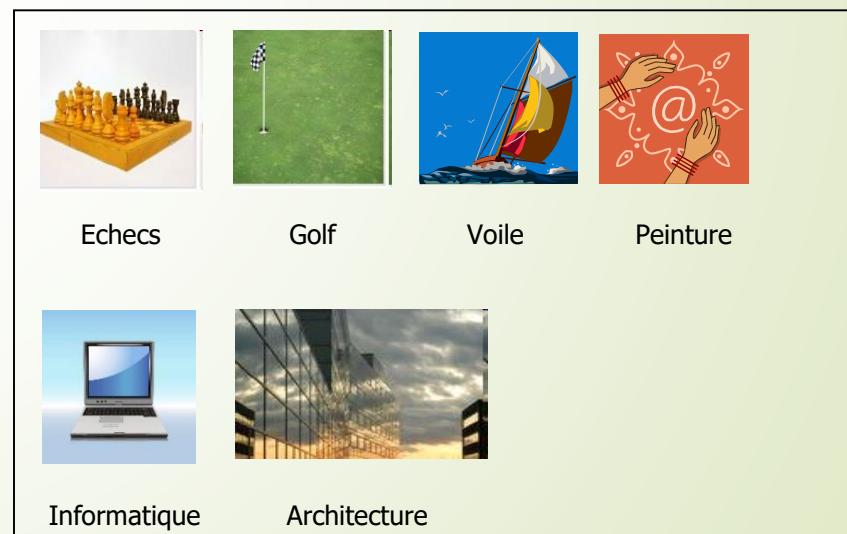
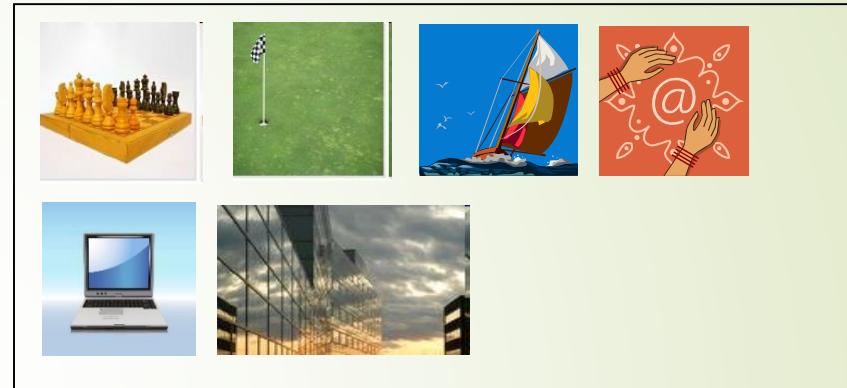
```
  
...
```

► Blocs avec image et légende

```
figure {float:left; width:100px;}
```

```
<figure>  
    
  <figcaption>legende 1</figcaption>  
</figure>  
<figure>  
    
  <figcaption>legende 2</figcaption>  
</figure>...
```

► Plus facile à gérer qu'avec un tableau



Propriété Clear (valeur left | right | both)

- ▶ **code HTML :**

```
<p><img .... /></p>
<p>Texte A...</p>
<p id="textB">Texte B</p>
```

- ▶ **code CSS exemple 1 :**

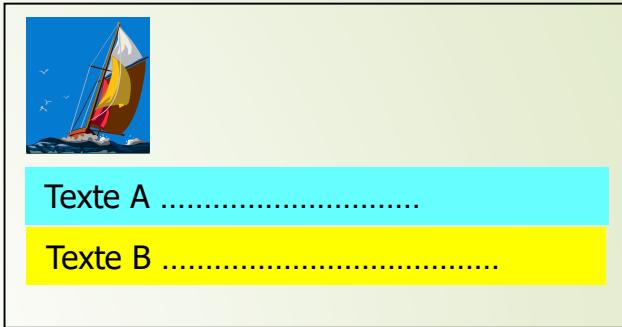
```
img {float : right;}
```

- ▶ **code CSS exemple 2 :**

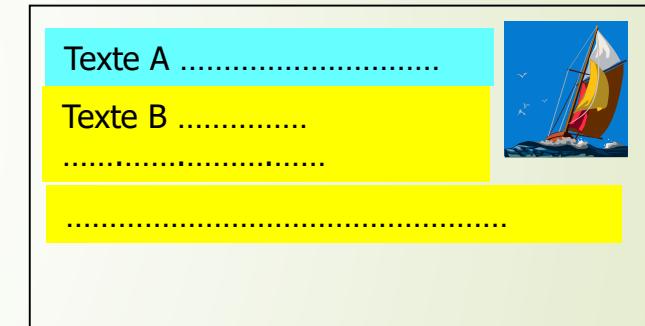
```
img {float : right;}
#textB {clear: right;}
```

- ▶ **Clear** permet qu'un élément se place tel qu'il n'ait rien à sa droite, à sa gauche ou ni l'un, ni l'autre.

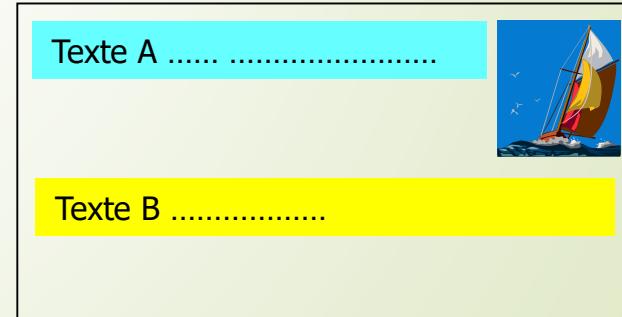
initial



ex 1



ex 2



Propriétés de listes



- ▶ list-style-type : none | disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha
- ▶ list-style-image : url("chemin/fichier.gif")
- ▶ list-style-position : outside | inside
- ▶ **Propriété raccourcie :**
`list-style : list-style-type list-style-image list-style-position`
- ▶ **Exemples :**
`list-style-image : url(carrebleu1.gif)`
`list-style-type : none`
`list-style : square inside`

• liste avec
outside

• liste avec
inside

Propriété Display

Display : none | inline | block | list-item

► Exemple 1 :

```
<ul>  
<li>item1</li>  
<li>item2</li>  
</ul>
```

- item1
- item2
- item3

► en CSS :

```
li {display:inline;}
```

item1 item2

► Exemple 2 :

```
<a href="#">item1</a>  
<a href="#">item2</a>
```

item1 item2

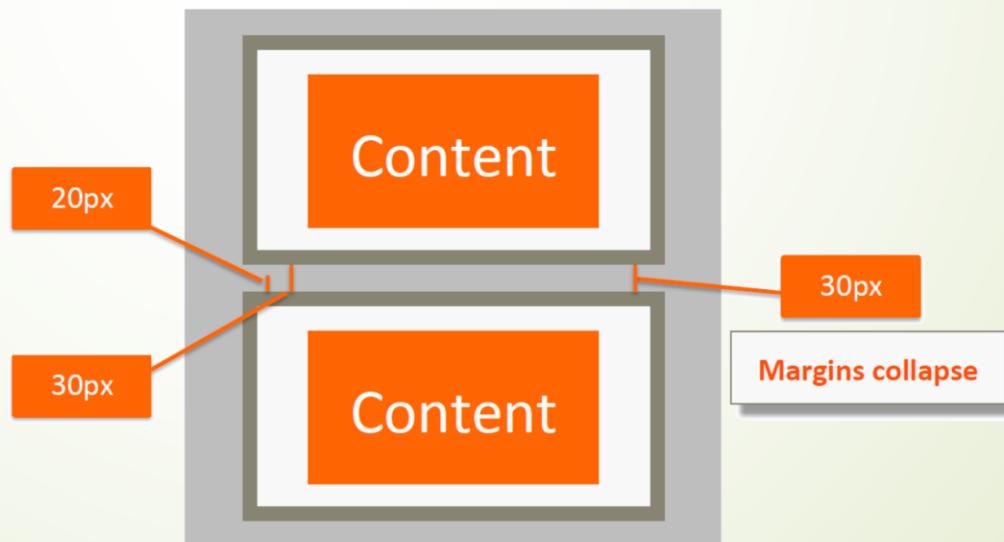
► en CSS :

```
a {display:block;}
```

item1
item2

Fusion des marges

- ▶ Si deux éléments de votre page ont des margins qui se touchent, alors ces margins **fusionnent pour ne faire plus qu'un seul margin** qui aura pour taille la plus grande des deux tailles des margins initiaux.

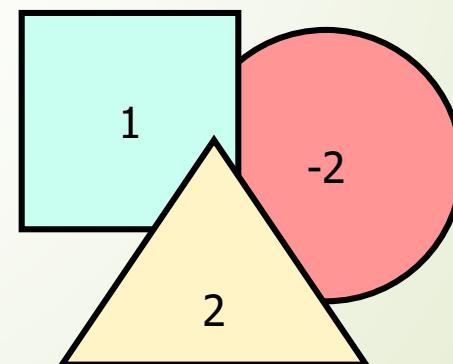


Le positionnement

- ▶ Type de **positionnement** : static | relative | absolute | fixed
 - static (statique) : flux normal du code HTML
 - relative : décalage par rapport à la position statique
 - absolute (absolue) : position par rapport au parent
 - fixed (fixe) : position par rapport à la zone de visualisation
- ▶ **Emplacement**

top : n | p % | -n | -p %
bottom : n | p % | -n | -p %
right : n | p % | -n | -p %
left : n | p % | -n | -p %
- ▶ **Superposition**

z-index : n | -n
plus n est grand, plus l'élément est au-dessus des autres



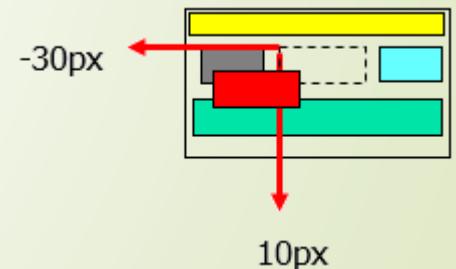
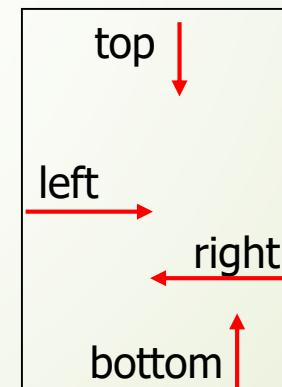
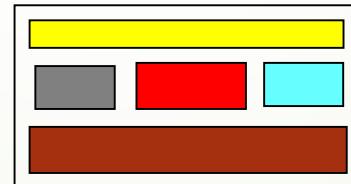
Positionnement static

- ▶ Comportement normal (par défaut).
- ▶ L'élément est alors positionné dans le flux avec sa position.
- ▶ Les propriétés top, right, bottom, left et z-index ne s'appliquent pas.

Positionnement relative

- ▶ L'élément est positionné dans le flux normal du document puis décalé, par rapport à lui-même, selon les valeurs fournies par top, right, bottom et left.
- ▶ Le décalage n'a pas d'impact sur la position des éléments.
- ▶ Aussi, l'espace fourni à l'élément sur la page est le même que celui fourni avec static.

```
div#bloc1 {  
    position: relative;  
    top: 10px;  
    left: -30px;  
}
```

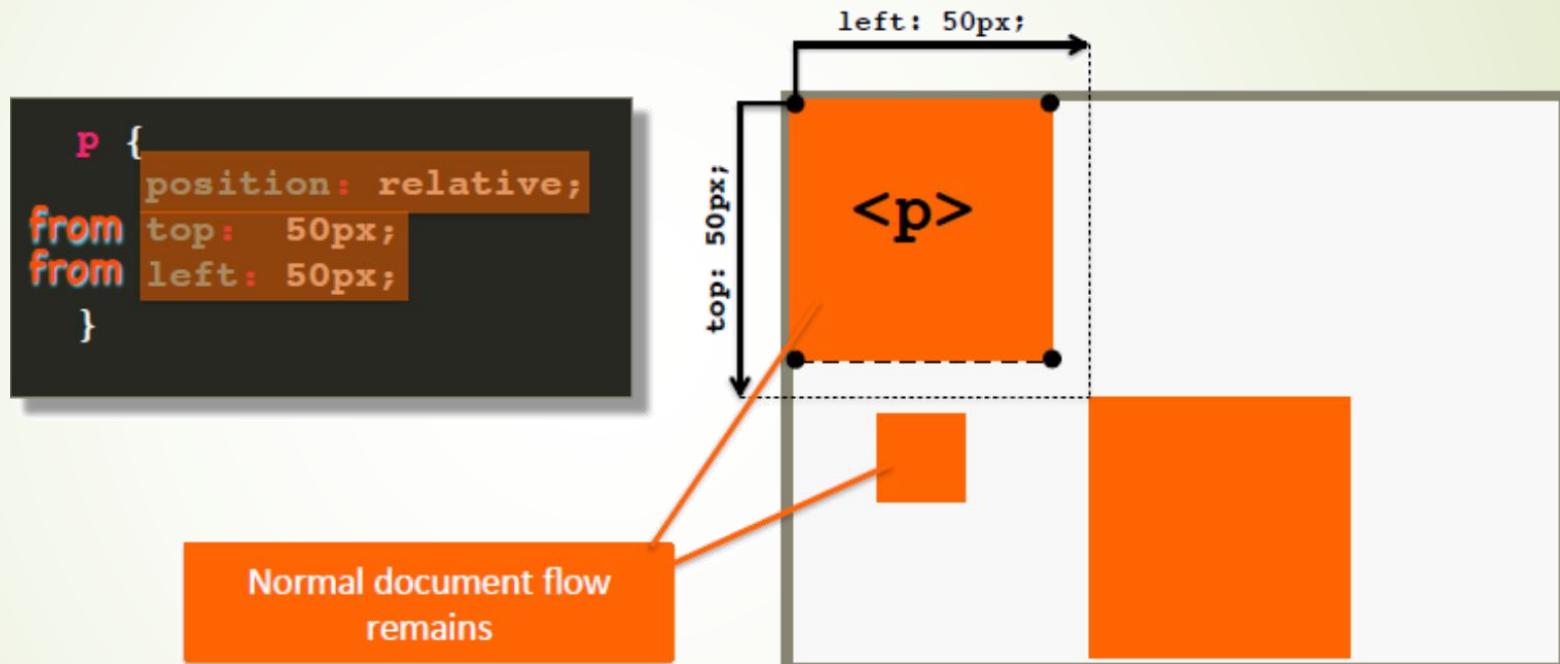


Positionnement relative

```
p {  
}  
}
```

<p>

Positionnement relative



Positionnement absolu

- ▶ L'élément est retiré du flux normal et aucun espace n'est créé pour l'élément sur la page.
- ▶ Il est ensuite positionné par rapport à son ancêtre le plus proche qui est positionné autre que static s'il y en a un ou par rapport au bloc englobant initial sinon.
- ▶ Par défaut, HTML est le seul élément qui a le positionnement non-static.
- ▶ La position finale de l'élément est déterminée par les valeurs de top, right, bottom et left.

Positionnement absolu

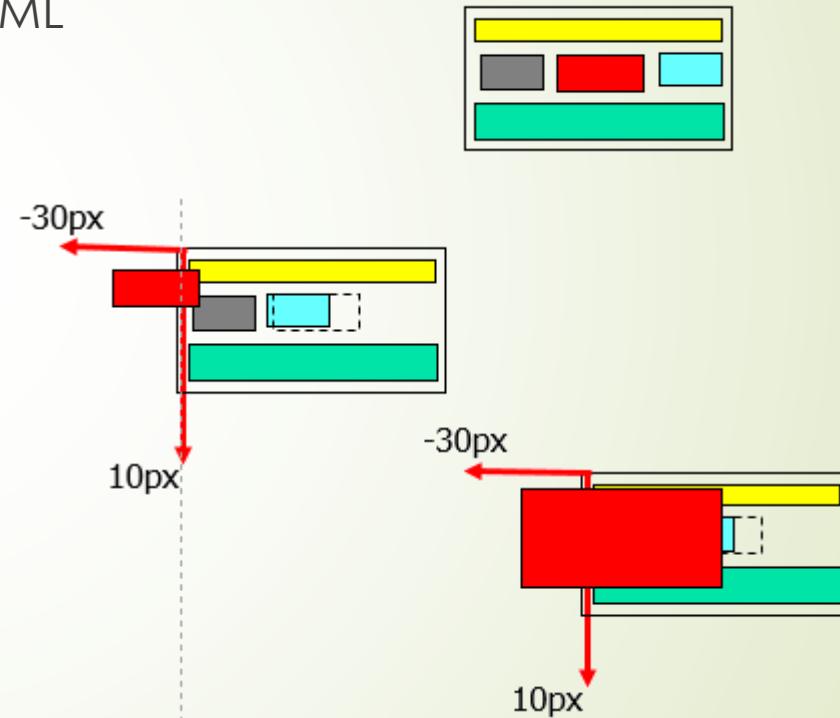
- ▶ **Statique** : flux normal du code HTML

- ▶ **Exemple 1**

```
div#bloc1 {  
    position: absolute;  
    top: 10px;  
    left: -30px; }
```

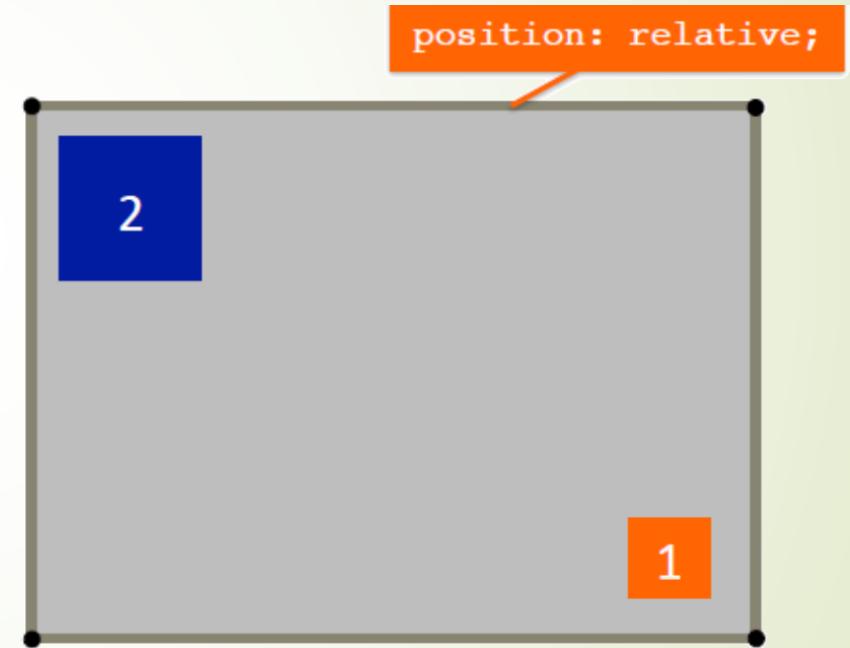
- ▶ **Exemple 2**

```
div#bloc2 {  
    position: absolute;  
    top: 10px;  
    left: -30px;  
    width: 150px;  
    height: 75px; }
```



Positionnement absolu

```
p { /* #1 */  
  position: absolute;  
  bottom: 10px;  
  right: 10px;  
}
```



Le positionnement : superposition d'élément

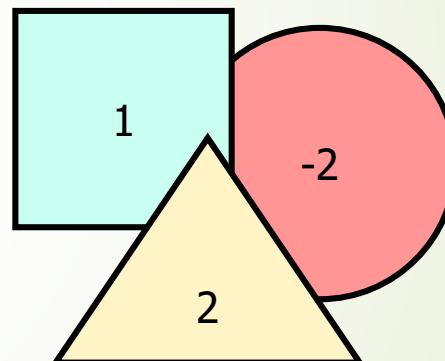
► Superposition

L'élément de **z-index** supérieur est au-dessus des autres.

```
div#cercle {z-index: -2; ...}
```

```
div#carre{z-index: 1; ...}
```

```
div#triangle {z-index: 2; ...}
```



Propriétés de page pour l'impression

- ▶ Pages imprimées

```
@page {  
    size: landscape;  
    margin: 2cm;  
}
```
- ▶ Pseudo-classes de pages imprimées
 - `@page:first` → 1ère page du document
 - `@page:left` → pages de gauche du document
 - `@page:right` → pages de droite du document
- ▶ propriétés de taille (size) : portrait | landscape (paysage)

Apparition d'une image au survol dans un menu

▶ Partie HTML

```
<ul id="menu">  
    <li><a href="#">menu 1</a></li>  
    <li><a href="#">menu 2</a></li>  
    <li><a href="#">menu 3</a></li>  
</ul>
```

menu 1
menu 2
menu 3

▶ Partie CSS

```
#menu a:hover {  
    background-image: url(images/guillemet.png);  
    background-repeat: no-repeat;  
    background-position: 1% 50%;  
    padding-left: 15px;  
}
```

menu 1
menu 2
» menu 3

Application css : bouton avec relief

Partie HTML

```
<input class="bouton" type="submit" value="Envoyer" />
```

Partie CSS

```
input.bouton {  
    border:2px outset red;  
    font-weight:bold;  
    cursor:pointer;  
}
```

```
input.bouton:hover {  
    border:2px outset white;  
    background-color:white; color:red;  
}
```

```
input.bouton:active {  
    border:2px inset red;  
    background-color:red;  
    color:white;  
}
```



Envoyer



Envoyer



Envoyer

Propriété content

- ▶

```
h1::before {  
    content:"Rubrique : ";  
}
```

-> ajoute l'expression avant l'élément h1
- ▶

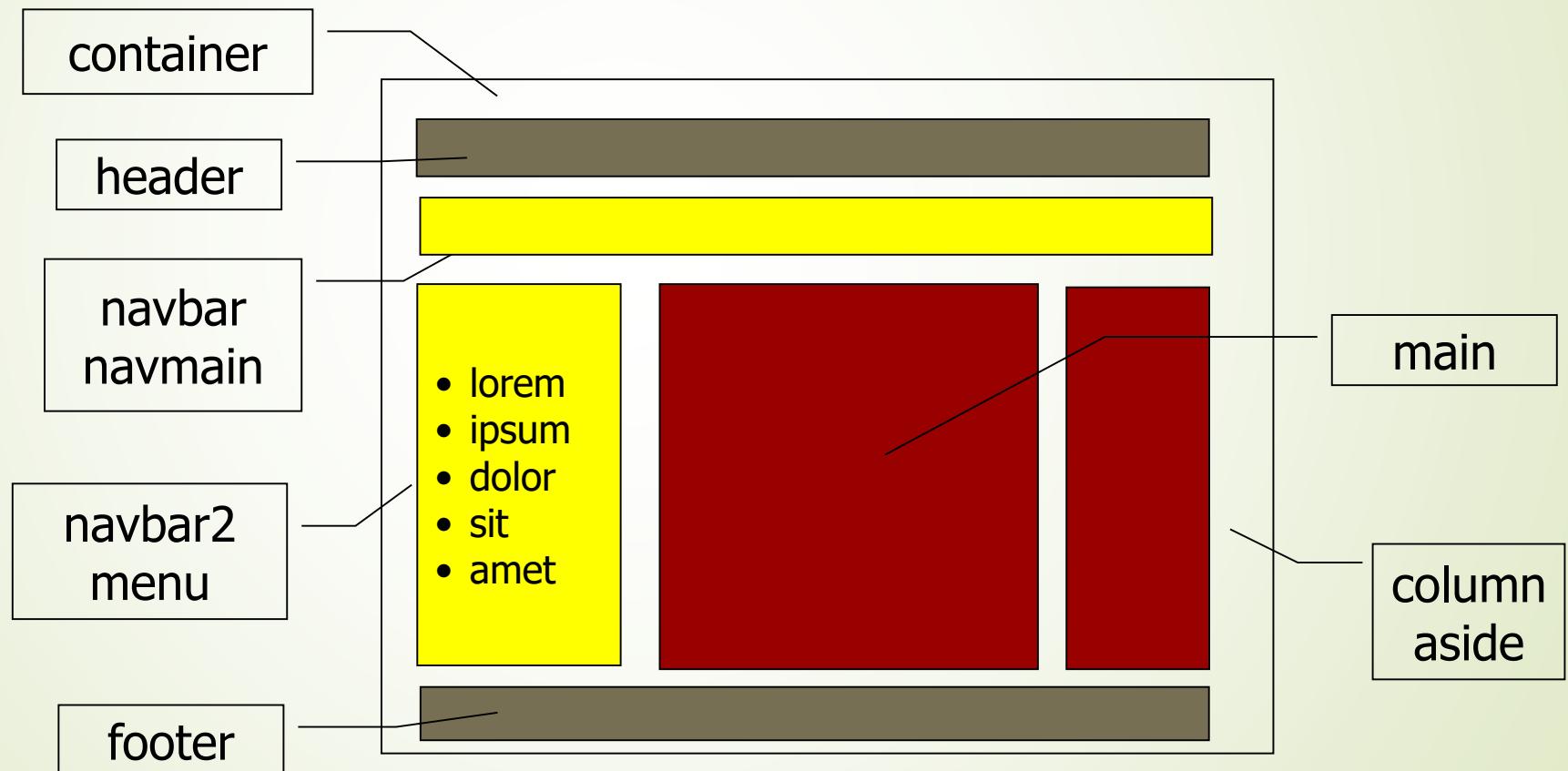
```
h2::after {  
    content:url(haut.gif) ;  
}
```

-> ajoute l'image après l'élément
- ▶ Ne fonctionne pas avec IE6

Feuille de style CSS externe liée : link

- ▶ Pour **tous périphériques** de sortie
`<link rel="stylesheet" type="text/css" href="nomdufichier1.css" />`
- ▶ Pour **différents périphériques** de sortie
`<link rel="stylesheet" type="text/css" href="nomdufichier1.css" media="screen" />`
`<link rel="stylesheet" type="text/css" href="nomdufichier2.css" media="print" />`
`<link rel="stylesheet" type="text/css" href="nomdufichier2.css" media="braille" />`
`<link rel="stylesheet" type="text/css" href="nomdufichier2.css" media="handheld" />`

Structure de page : noms usuels des zones



Structuration de la feuille de style : exemple d'ordre

- ▶ Reset global (ex: marge à 0...)
- ▶ Base typographique (h1, h2...)
- ▶ Formulaire
- ▶ Structure de page (container, header, main, footer)
- ▶ Structure des templates (colonne 1, colonne 2...)
- ▶ Éléments de contenu
- ▶ Éléments de navigation
- ▶ Eléments-outils ou toolbox (.clearer, .floatleft)
- ▶ Variantes par gabarit
- ▶ Surcharges pour autres medias (print, handled...)

Structuration en n feuilles de style

- ▶ Disposition de la page
`layout.css`
- ▶ Typographie de la page
`style.css`
- ▶ Typographie de la partie principale de la page
`main.css`



Quelques Concepts



Cascade

- ▶ la cascade des styles signifie que l'ordre d'apparition des règles dans le CSS a une importance ; quand deux règles applicables ont la même spécificité, c'est la dernière déclarée qui sera utilisée pour la mise en forme.
- ▶ Pour rappel, HTML est interprété séquentiellement de haut en bas

```
h1 {  
    color: red;  
}  
h1 {  
    color: blue;  
}
```

```
<h1>This is my heading.</h1>
```

This is my heading.

Feuilles de style en cascades

styl1.css

```
h1.cs {color: blue; text-align:center}
```

styl2.css

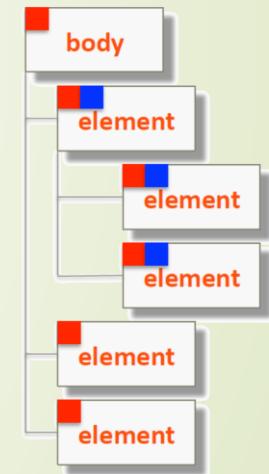
```
@import "styl1.css";      → à mettre en premier  
p.cs {color: green}
```

toto.htm

```
<link href="styl2.css" rel="stylesheet">  
<h1 class="cs">texte h1.cs bleu</h1>  
<p class="cs">texte p.cs vert</p>  
<p>texte</>texte p normal</p>
```

Héritage

- ▶ Pour chaque propriété CSS, la spécification indique si, par défaut, cette propriété est héritée ou non.
- ▶ Si on applique une couleur (color) au body, tous les éléments du body héritent de cette couleur.
- ▶ Pas d'héritage de margin et padding.
- ▶ On peut **forcer l'héritage** d'une propriété par défaut non héritable avec la valeur inherit : `{margin: inherit }`



Héritage - Exemple

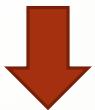


```
body {  
    color: blue;  
}  
span {  
    color: black;  
}
```



<p>As the body has been set to have a color of blue this is inherited through the descendants.</p>

<p>We can change the color by targetting the element with a selector, such as this span.</p>



As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targetting the element with a selector, such as this span.



Spécificité

- ▶ Quand des règles avec des sélecteurs différents s'appliquent sur un même élément, le navigateur choisit la règle qui a **la plus grande spécificité**.
- ▶ La spécificité mesure essentiellement combien la sélection est précise :
 - ▶ Un sélecteur d'élément est peu spécifique — il cible tous les éléments d'un type donné dans la page — son score est donc faible ;
 - ▶ Un sélecteur de classe est plus spécifique — dans la page, il ne cible que les éléments dont l'attribut class a la valeur choisie — son score est plus important.

Spécificité - Exemple



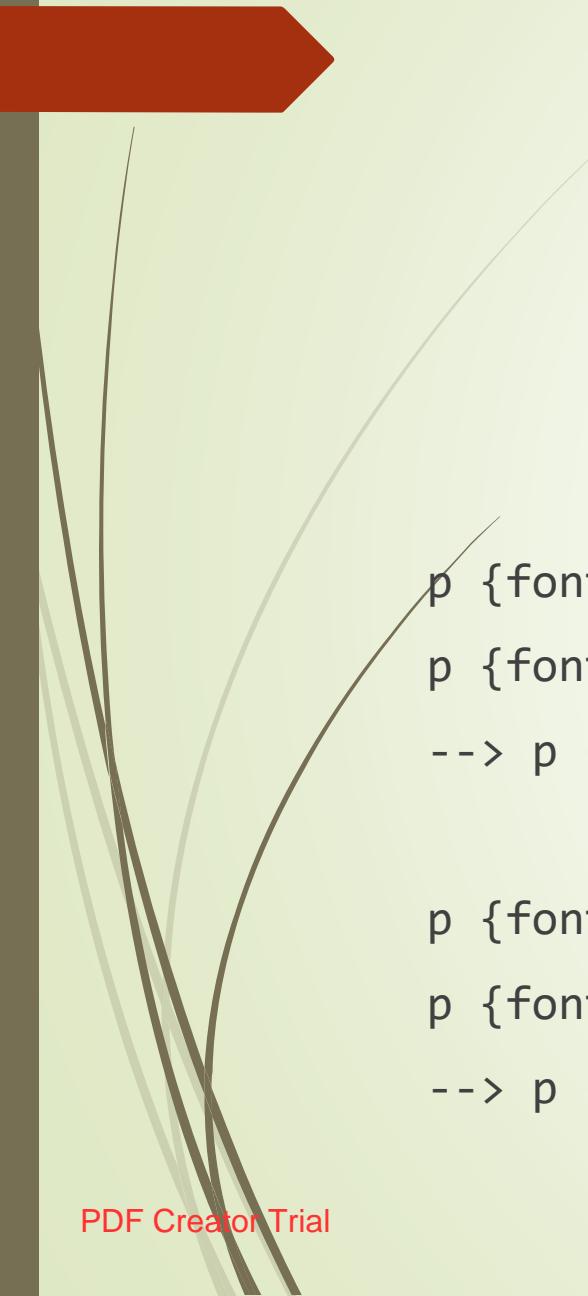
```
.main-heading {  
    color: red;  
}  
  
h1 {  
    color: blue;  
}
```



```
<h1 class="main-heading">This is my heading.</h1>
```



This is my heading.



Conflits de règles : priorité à la dernière

Règle postérieure > Règle antérieure

Dans un même emplacement (fichier .css, <style>),
la dernière règle annule la règle antérieure.

```
p {font-family: arial}  
p {font-size: 10 pt}  
--> p {font-family:arial ; font-size: 10 pt ;}
```

```
p {font-family: arial ; font-size: 10 pt}  
p {font-size: 12 pt}  
--> p {font-family:arial; font-size: 12 pt ;}
```

Conflits de style : priorité à la forme la plus proche

Style intra-ligne > Style interne > Style externe (lié ou importé)

```
<style>
  p {text-align: center; font-size:10pt;}
</style>
...
<body>
  <p align="right">Texte 1</p>
  <p style="text-align:left">Texte 2</p>
  <p>Texte 3</p>
</body>
```

Conflits de style : sélecteur contextuel

```
h1, p { color: blue }  
em { color: red }  
h1 em { color: red }
```

```
<h1>Chapitre I</h1>  
<p>Texte avec <em>point  
important</em></p>  
<h1>Chapitre II avec <em>appui sur un  
thème</em></h1>
```

Exemples d'utilisation

```
ul li { list-style-type: disc }  
ul ul li {list-style-type: square ; font-size: 10pt }  
#menu ol a:hover {color: red; }
```

Rendre prioritaire avec !important

- ▶ L'ajout de **!important** rend prioritaire la déclaration quel que soit son emplacement.

```
h2 {color: maroon !important}  
h2 {color: yellow}  
h1 {  
    color: black !important;  
    background: white !important;  
}  
p {  
    font-size: 12pt !important;  
    font-style: italic;  
}
```

Priorité des feuilles de style selon l'origine

Style auteur > Style utilisateur > Style agent-utilisateur

- ▶ Les styles définis par l'utilisateur écrasent les styles par défaut du navigateur (agent utilisateur ou user agent = UA)
- ▶ Les styles définis par l'auteur écrasent les styles de l'utilisateur
- ▶ **Exception** : une règle "importante" de l'utilisateur est prioritaire sur une règle normale de l'auteur.

Priorité en fonction des types de sélecteurs

Sélecteur avec identifiant

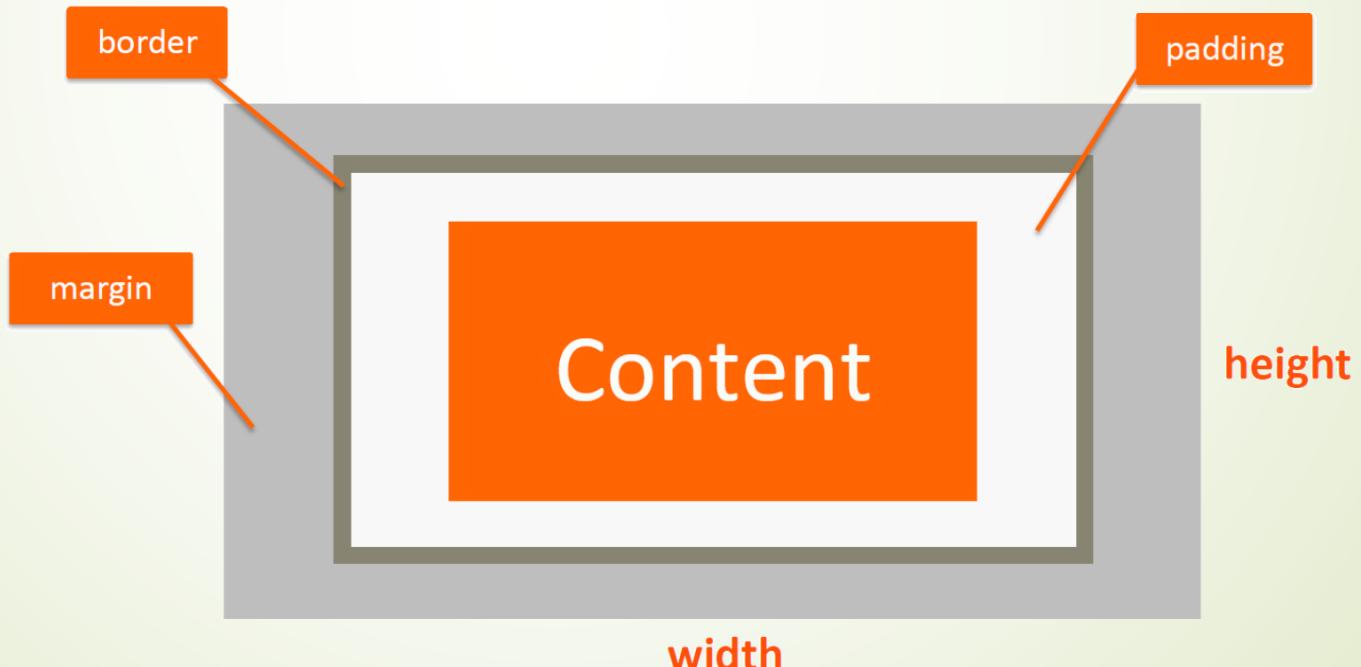
> Sélecteur avec classe ou pseudo-classe

> Sélecteur contextuel > Sélecteur simple

sélecteur	déclaration	ID	classe	élément	total
		a	b	c	
li	{...}	a=0	b=0	c=1	= 001
ul li	{...}	a=0	b=0	c=2	= 002
ul ol li	{...}	a=0	b=0	c=3	= 003
.cl	{...}	a=0	b=1	c=0	= 010
li.cl	{...}	a=0	b=1	c=1	= 011
ul ol li.cl	{...}	a=0	b=1	c=3	= 013
#toto	{...}	a=1	b=0	c=0	= 100

Le modèle de boite (The Box Model)

- ▶ En CSS, tout élément est inclus dans une boîte ("box" en anglais)
- ▶ Il existe deux type de boîtes : les boîtes **en bloc** ("block boxes" en anglais) et les boîtes **en ligne** ("inline boxes").





Boite en bloc

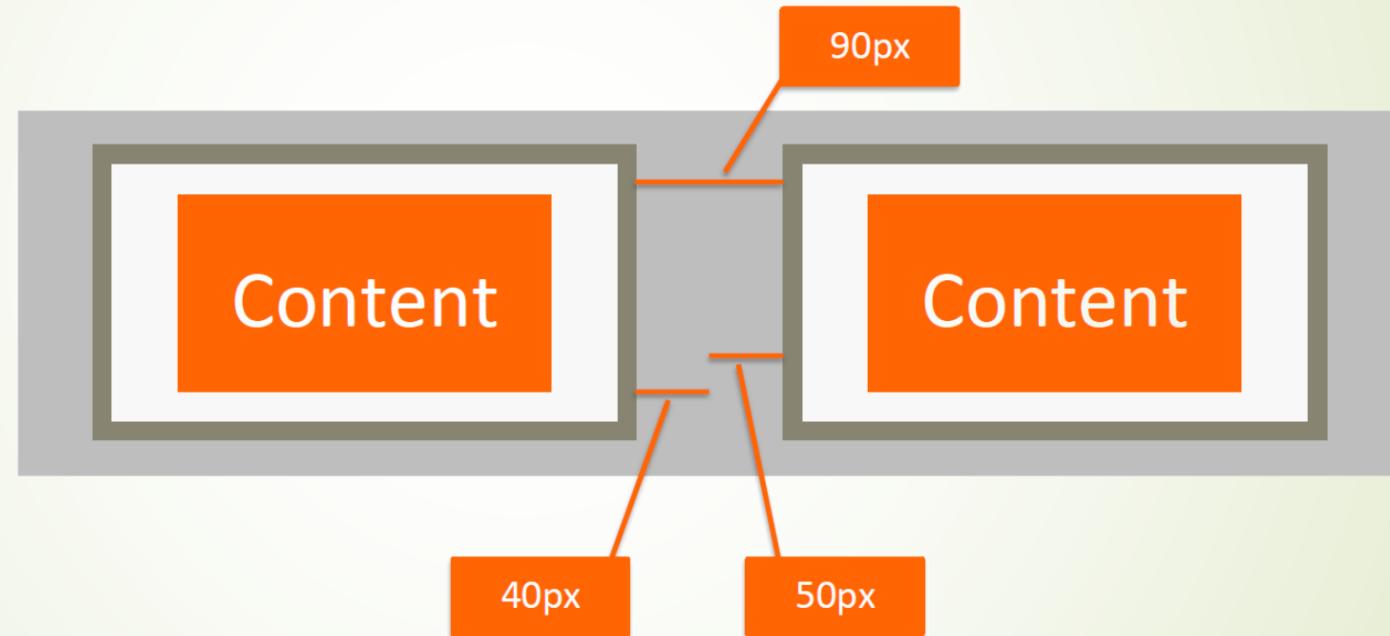
- ▶ La boîte s'étend en largeur pour remplir totalement l'espace offert par son conteneur. Dans la plupart des cas, la boîte devient alors **aussi large que son conteneur**, occupant 100% de l'espace disponible.
- ▶ La boîte occupe sa propre nouvelle ligne et créé un retour à la ligne, faisant ainsi passer les éléments suivants à la ligne d'après.
- ▶ Les propriétés de largeur (width) et de hauteur (height) sont toujours respectées.
- ▶ Les propriétés padding, margin et border — correspondantes respectivement aux écarts de remplissage interne, externe et à la bordure de la boîte — auront pour effet de repousser les autres éléments.
- ▶ Les titres (`<h1>`, `<h2>`, etc.) et les paragraphes (`<p>`) utilisent le mode "bloc" comme propriété de positionnement extérieur par défaut.



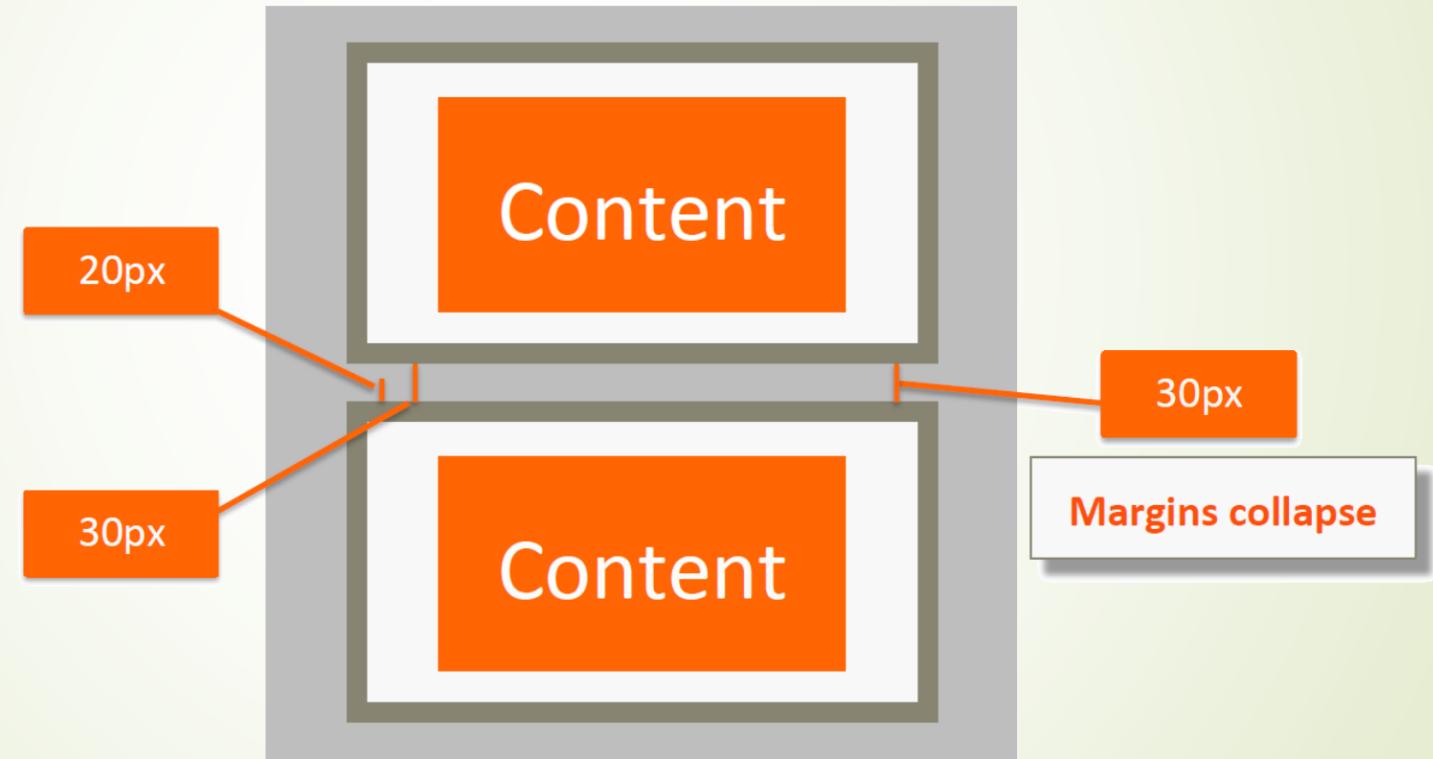
Boite en ligne

- ▶ La boîte **ne crée pas de retour à la ligne**, les autres éléments se suivent donc en ligne.
- ▶ Les propriétés de largeur (width) et de hauteur (height) ne s'appliquent pas.
- ▶ Les propriétés padding, margin et border seront appliquées mais ne provoqueront pas de déplacement des éléments alentours.
- ▶ Les éléments <a>, utilisés pour les liens, ou encore , et sont tous des éléments qui s'affichent "en ligne" par défaut.

Marge et boîte en ligne



Marge et boite en bloc





Méthodes pour appliquer des styles en HTML

Méthodes pour appliquer des styles en HTML

► Feuille de style interne au document

- liste des règles CSS dans balise `<style...></style>`
- placée dans la partie `<head>`

► Feuille de style externe liée

- liste des règles CSS dans un fichier externe .css
- appel à ce fichier externe avec une balise `<link ... />`

► Feuille de style externe importée

- liste des règles CSS dans un fichier externe .css
- appel à un fichier externe avec `@import`

► Style en ligne (intra-ligne ou incorporé)

- attribut `style="..."` dans la balise d'un élément

Feuille de style interne au document

- ▶ Balise `<style type="text/css"></style>` placée dans l'en-tête du document
- ▶ Liste de règles CSS placée dans cette balise ouvrante et fermante
- ▶ Règle appliquée à tous les sélecteurs correspondant du document
- ▶ Utilisation pour un document isolé, éventuellement pour une page particulière d'un site
- ▶ Utilisation ne convenant pas pour mettre en forme un site entier

Exemple

```
<head>
  <style type="text/css">
    body {background-color: silver; color: maroon;}
    h1 {text-align: center; font-size: 1.5em; color: black;}
  </style>
</head>
<body>
  <h1>Titre A</h1>
  <p>Texte 1</p>
  <p>Texte 2</p>
</body>
```

Feuille de style CSS externe liée

- ▶ Liste de règles CSS placée dans un fichier CSS externe (extension css)
- ▶ Balise **link** à ajouter dans l'en-tête du document html permettant de faire un "lien" vers ce fichier css
- ▶ Utilisation bien adaptée à un site web dont toutes les pages HTML comporteront la balise <link>
- ▶ Règle appliquée à tous les sélecteurs correspondant dans les documents liés

Exemple

1. Code CSS : contenu d'un fichier CSS nommé monstyle.css

```
body { background-color: white ;  
      color: blue;  
}  
h1 { text-align: center;  
     font-size: 1.5em ;  
     color: black;  
}
```

2. Code HTML (extrait du fichier)

```
...  
<head><title>...</title>  
<link rel="stylesheet" type="text/css" href="chemin/monstyle.css">  
</head>  
...
```

Feuille de style CSS importée

- ▶ Liste de règles CSS dans un fichier CSS externe
- ▶ Règle **@import** à ajouter dans la feuille de style (interne ou externe)
- ▶ Se place avant tout autre déclaration de style
- ▶ Règle appliquée à tous les sélecteurs correspondant dans les documents liés
- ▶ Utilisation adaptée à un site web, notamment pour "cascader les styles"
- ▶ Syntaxe :
 - ▶ `@import "fichier.css"`
 - ▶ `@import url("fichier.css")`

Exemple

```
<style type="text/css">
    @import url("fichier.css")
    p {
        color:red;
    }
</style>
```

Style en ligne, dans une balise HTML

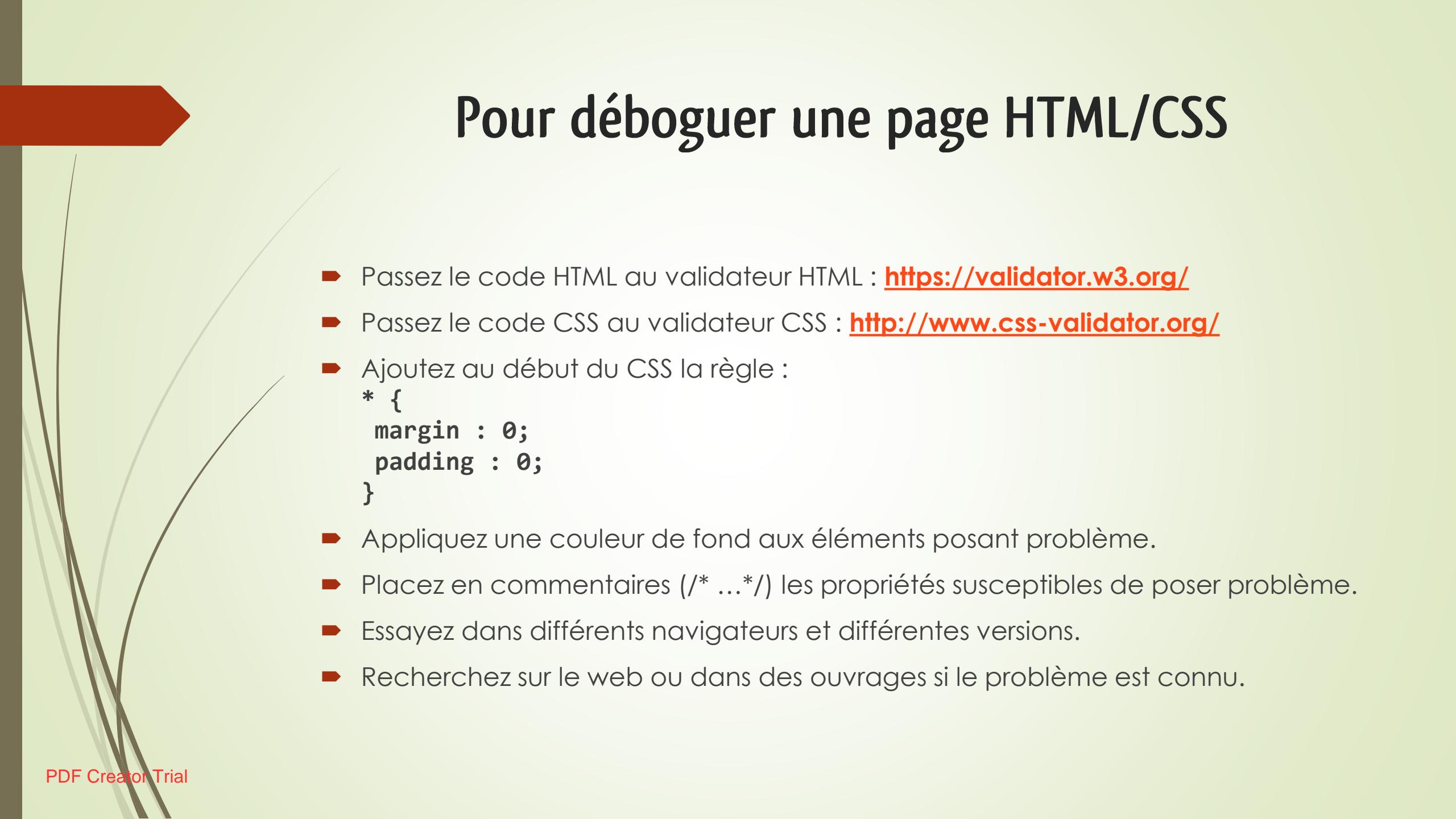
- ▶ Bloc de déclarations CSS
 - ▶ incorporé dans la balise
 - ▶ introduit par l'attribut **style**
- ▶ Exemple :
`<p style="text-align: center; background-color:#ccc;">Texte</p>`
- ▶ S'applique à une balise précise du document
- ▶ Intéressant pour un essai lors du développement
- ▶ Déconseillé actuellement : à remplacer plutôt par l'utilisation d'un identifiant

HTML : `<element id="truc">`

CSS : `#truc {text-align: center; background-color:#ccc; }`

4 manières d'appliquer un style : exemple

```
<html>
<head>
<title>...</title>
<link rel="stylesheet" type="text/css" href="fichier1.css" />
<style type="text/css">
    @import url("fichier2.css");
    h1 { color: blue }
</style>
</head>
<body>
    <h1>Ce texte est en bleu à cause du style interne</h1>
    <p style="color: green" >Ce paragraphe est en vert.</p>
</body>
</html>
```



Pour déboguer une page HTML/CSS

- ▶ Passez le code HTML au validateur HTML : <https://validator.w3.org/>
- ▶ Passez le code CSS au validateur CSS : <http://www.css-validator.org/>
- ▶ Ajoutez au début du CSS la règle :

```
* {  
    margin : 0;  
    padding : 0;  
}
```
- ▶ Appliquez une couleur de fond aux éléments posant problème.
- ▶ Placez en commentaires /* ... */ les propriétés susceptibles de poser problème.
- ▶ Essayez dans différents navigateurs et différentes versions.
- ▶ Recherchez sur le web ou dans des ouvrages si le problème est connu.

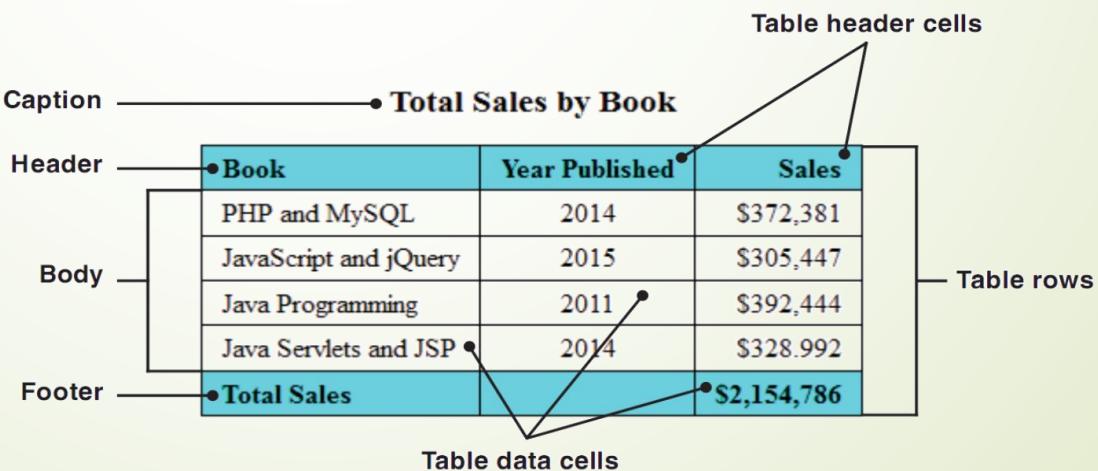
Tableau HTML

Création et application du style CSS

Tableau (Table)

► L'élément HTML <table>

- ▶ permet de représenter un tableau de données, c'est-à-dire des informations exprimées sur un tableau en deux dimensions.
- ▶ est un ensemble structuré de données (**table de données**) présentées en lignes et colonnes.
- ▶ **ligne** est une ligne d'information horizontale
- ▶ **colonne** est une ligne d'information verticale
- ▶ **cellule** est l'intersection d'une ligne et une colonne et contient une donnée



Éléments de tableau

Élément	Indique le début et la fin de	Contient
<table> ... </table>	un tableau de données dans une page web	Tous les éléments liés à un tableau
<tr> ... </tr>	une ligne de cellules dans un tableau	Cellules de données et d'en-tête
<th> ... </th>	une cellule d'un tableau comme une cellule d'en-tête	Contenu d'en-tête
<td> ... </td>	une cellule d'un tableau qui contient des données	Contenu de cellule de donnée

Exemple

Éléments de tableau

Élément	Indique le début et la fin de	Contient
<caption> ... </caption>	la légende (ou le titre) d'un tableau	la légende (ou le titre) d'un tableau
<thead> ... </thead >	L'en-tête d'un tableau	un ou plusieurs éléments d'en-tête
<tbody> ... </tbody >	le corps d'un tableau	un ou plusieurs éléments de corps
<tfoot> ... </tfoot >	le pied de page d'un tableau	un ou plusieurs éléments de pied de page

Exemple

Propriétés CSS

- ▶ border-collapse : collapse | separate
 - ▶ détermine si les bordures d'un tableau sont séparées ou fusionnées.
 - ▶ Quand elles sont séparées, chaque cellule du tableau a ses propres bordures, distinctes.
 - ▶ Quand elles sont fusionnées, les bordures des cellules sont partagées.
- ▶ Border-spacing : n
 - ▶ définit la distance qu'il y a entre les bordures de cellules adjacentes d'un tableau (uniquement lorsque border-collapse vaut separate).
- ▶ Padding : n
 - ▶ permet de définir les différents écarts de remplissage sur les quatre côtés d'un élément

Propriétés CSS

- ▶ `Text-align : left | right | center | justify`
 - ▶ définit l'alignement horizontal d'un élément de bloc ou de la boîte d'une cellule de tableau
- ▶ `Vertical-align : baseline | sub | super | text-top | text-bottom | middle | top | bottom`
 - ▶ définit l'alignement vertical d'une boîte en ligne (inline) ou d'une cellule de tableau

Exemple

Pseudo-Classes CSS3

► Sélecteurs

syntaxe	description
:nth-child(n)	Nième enfant de l 'élément parent
:nth-last-child(n)	Nième enfant de l 'élément parent (le comptage s'effectue depuis la fin)
:nth-of-type(n)	Nième enfant de l 'élément parent d'un type donné
:nth-last-of-type(n)	Nième enfant de l 'élément parent d'un type donné (le comptage s'effectue depuis la fin)

► Valeur de n

valeur	description
odd	Tous les enfants impair
even	Tous les enfants pair
n	Nème enfant
2n	Similaire à even
3n	Tous les 3 ^{ème} enfants (3,6,9,...)
2n+1	Similaire à odd
3n+1	Tous les 3 ^{ème} enfants commençant à 1 (1,4,7, ...)

Exemple

Fusion des cellules

- ▶ Attributs de <th>, <td>
- ▶ Colspan
 - ▶ Cet attribut contient un entier positif indiquant le nombre de colonnes sur lesquelles s'étend la cellule
 - ▶ La valeur par défaut est 1
- ▶ Rowspan
 - ▶ Cet attribut contient un entier positif indiquant sur combien de lignes s'étend la cellule.
 - ▶ La valeur par défaut est 1

Exemple

Accessibilité

- ▶ Tableaux sont difficiles à déchiffrer pour les personnes malvoyantes
- ▶ Caption
 - ▶ représente la légende (ou le titre) d'un tableau.
 - ▶ Il doit être le première élément parmi les descendants de l'élément `<table>`
 - ▶ une description claire et concise permet aux utilisateurs de décider s'ils doivent lire le contenu du tableau ou le passer
- ▶ Attributs pour améliorer l'accessibilité
 - ▶ Headers : list<String>
 - ▶ Cet attribut est une liste de chaînes de caractères séparées par des espace.
 - ▶ Chacune correspond à l'attribut id de l'élément `<th>` qui s'applique à cet élément
 - ▶ Scope : row | col | rowgroup | colgroup
 - ▶ Cet attribut référence les cellules auxquelles l'élément `<th>` est lié

[Exemple](#)

Tableaux imbriqués

- ▶ Il est possible d'inclure un tableau dans un autre, à condition d'en spécifier la structure complète, y compris l'élément <table>
- ▶ [Exemple](#)

Contrôler le passage à la ligne

- ▶ Par défaut, le contenu d'une cellule passe à la ligne automatiquement. Les passages à la ligne sont faits naturellement pour remplir les boîtes.
- ▶ La propriété qui stoppe le passage à la ligne est la suivante:
 - ▶ Table { white-space: nowrap }
- ▶ [Exemple](#)



Utilisation des tableaux

- ▶ Les tableaux HTML ne doivent être utilisés que pour des données tabulaires
- ▶ Utiliser les tableaux pour la mise en page au lieu des techniques des CSS est une mauvaise idée
- ▶ L'avantage du tableau tient dans sa rigueur
- ▶ L'information est facilement interprétée par des associations visuelles entre les en-têtes de lignes et colonnes

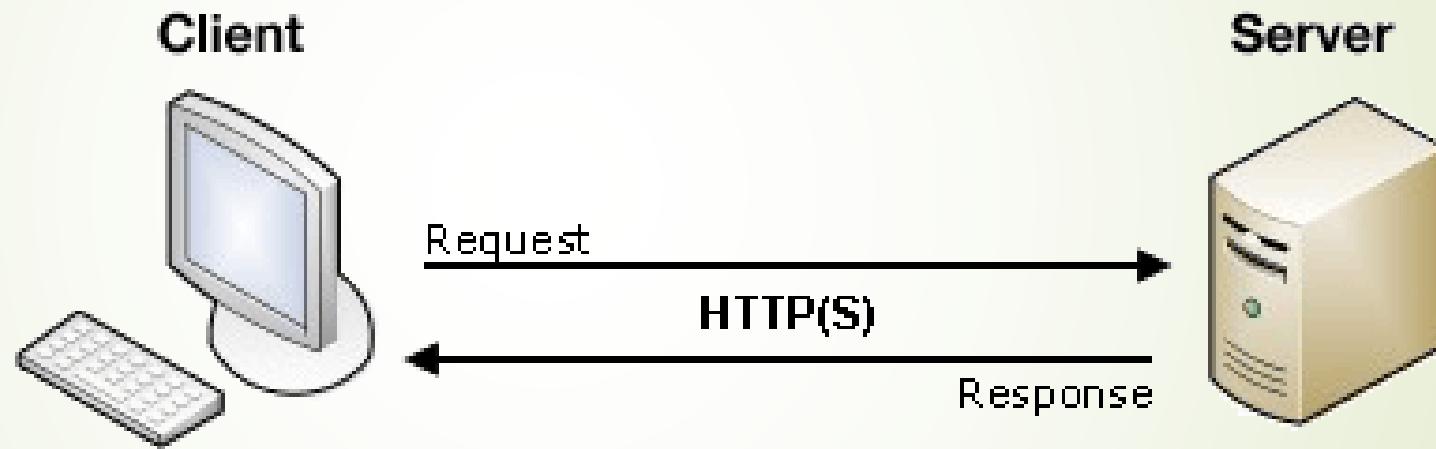
Formulaire HTML

Création et application du style CSS

Formulaire HTML

- ▶ On utilise les formulaires HTML pour créer des pages Web dynamiques
- ▶ Ils sont un des principaux vecteurs d'interaction entre un utilisateur et un site Web ou une application
- ▶ Ils permettent à l'utilisateur d'envoyer des données au site Web
- ▶ La plupart du temps, ces données sont envoyées à des serveurs Web mais la page peut aussi les intercepter et les utiliser elle-même

Architecture client/server



Création d'un formulaire

- ▶ Tous les formulaires HTML débutent par un élément **<form>**

Attributs d'un formulaire :

- ▶ Action : L'attribut **action** définit l'emplacement (une URL) où doivent être envoyées les données collectées par le formulaire.
- ▶ Method : L'attribut **method** définit la méthode HTTP utilisée pour envoyer les données (cela peut être « get » ou « post »).
- ▶ Name : L'attribut **name** définit le nom du formulaire.
- ▶ Enctype : Lorsque la valeur de l'attribut method est post, cet attribut définit le type MIME qui sera utilisé pour encoder les données envoyées au serveur. C'est un attribut énuméré qui peut prendre les valeurs suivantes :
 - ▶ application/x-www-form-urlencoded : la valeur par défaut si l'attribut n'est pas défini
 - ▶ multipart/form-data : la valeur utilisée par un élément **<input>** dont l'attribut type vaut "file"

Élément <Input>

- ▶ L'élément HTML <input> est utilisé pour créer **un contrôle interactif dans un formulaire** Web qui permet à l'utilisateur de saisir des données.
- ▶ Les saisies possibles et le comportement de l'élément <input> dépend fortement de la valeur indiquée dans son attribut type.

Les différents types de champs <input>

Les types de champs disponibles sont :

- ▶ **button** : un bouton sans comportement défini.
- ▶ **checkbox** : une case à cocher qui permet de sélectionner/désélectionner une valeur
- ▶ **color** : un contrôle HTML5 qui permet de choisir une couleur.
- ▶ **date** : un contrôle HTML5 qui permet de saisir une date (composé d'un jour, d'un mois et d'une année).
- ▶ **datetime-local** : un contrôle HTML5 qui permet de saisir une date et une heure (sans fuseau horaire).
- ▶ **email** : un champ HTML5 qui permet de saisir une adresse électronique.
- ▶ **file** : un contrôle qui permet de sélectionner un fichier. L'attribut accept définit les types de fichiers qui peuvent être sélectionnés.

Les différents types de champs <input>

Les types de champs disponibles sont :

- ▶ **hidden** : un contrôle qui n'est pas affiché mais dont la valeur est envoyée au serveur.
- ▶ **image** : un bouton graphique d'envoi du formulaire. L'attribut src doit être défini avec la source de l'image et l'attribut alt doit être défini avec le texte alternatif. Les attributs height et width permettent de définir la taille de l'image en pixels.
- ▶ **month** : un contrôle HTML5 qui permet de saisir un mois et une année (sans fuseau horaire).
- ▶ **number** : un contrôle HTML5 qui permet de saisir un nombre.
- ▶ **password** : un champ texte sur une seule ligne dont la valeur est masquée. Les attributs maxlength et minlength définissent la taille maximale et minimale de la valeur à saisir dans le champ.

Les différents types de champs <input>

Les types de champs disponibles sont :

- ▶ **radio** : un bouton radio qui permet de sélectionner une seule valeur parmi un groupe de différentes valeurs.
- ▶ **range** : un contrôle HTML5 qui permet de saisir un nombre dont la valeur exacte n'est pas importante.
- ▶ **reset** : un bouton qui réinitialise le contenu du formulaire avec les valeurs par défaut.
- ▶ **search** : un champ texte HTML5 sur une ligne pour des termes de recherche. Les sauts de ligne sont automatiquement retirés.
- ▶ **submit** : un bouton qui envoie le formulaire.
- ▶ **tel** : un contrôle HTML5 pour saisir un numéro de téléphone.

Les différents types de champs <input>

Les types de champs disponibles sont :

- ▶ **text** : un champ texte sur une seule ligne. Les sauts de ligne sont automatiquement retirés. **C'est la valeur par défaut !**
- ▶ **time** : un contrôle HTML5 pour saisir l'heure sans la zone UTC
- ▶ **url** : un champ HTML5 permettant de saisir une URL.
- ▶ **week** : un contrôle HTML5 permettant de saisir une date représentée par un numéro de semaine et une année (sans indication de fuseau horaire).

Attributs communs à l'ensemble des types Input

Attribut	Description
autocomplete	Une chaîne de caractères qui indique le type d'autocomplétion à utiliser.
autofocus	Un attribut booléen qui passe le focus sur le champ lorsque le formulaire est affiché.
disabled	Un attribut booléen qui indique si le champ doit être désactivé.
form	L'identifiant du formulaire (la valeur de l'attribut id de l'élément <form>) auquel le champ est rattaché. Si cet attribut est absent, le champ sera rattaché au formulaire le plus proche qui le contient s'il existe.
list	L'identifiant (valeur de l'attribut id) d'un élément <datalist> qui fournit une liste de suggestions.
name	Le nom du champ qui sera rattaché à la donnée envoyée via le formulaire.

► Example : [01_form.html](#)

Attributs communs à l'ensemble des types Input

Attribut	Description
readonly	Un attribut booléen qui indique si le champ peut être édité ou non.
required	Un attribut booléen qui indique que le champ doit être renseigné avant de pouvoir envoyer le formulaire.
tabindex	Une valeur numérique qui indique à l'agent utilisateur l'ordre selon lequel naviguer au clavier grâce à la touche Tab.
type	Une chaîne de caractère qui indique le type de champ représenté par l'élément <input>.
value	La valeur du champ.

► Exemple : [exemples\form\01_form.html](#)

Boutons dans le formulaire

► Attributs des éléments boutons

Attribut	Description
alt	Pour les boutons image, le texte alternatif si l'image n'est pas chargée
height	Pour les boutons image, la longueur de l'image en pixels
src	Pour les boutons image, l'URL de l'image
width	Pour les boutons image, la largeur de l'image
formaction	L'URL à laquelle envoyer les données du formulaire. Cette valeur prend le pas sur l'attribut action du formulaire s'il est défini.
formenctype	Une chaîne de caractères qui indique le type d'encodage à utiliser pour les données du formulaire.
formmethod	La méthode HTTP à utiliser pour envoyer le formulaire.
formnovalidate	Un booléen qui, lorsqu'il est présent, indique que les champs du formulaire ne sont pas soumis aux contraintes de validation avant l'envoi des données au serveur.

► Exemple : [exemples\form\02_buttons.html](#)

Champ de saisie avec du texte

- ▶ Les éléments <input> dont l'attribut type vaut "text" permettent de créer des champs de saisie avec du texte sur une seule ligne.

Attribut	Description
maxlength	Le nombre de caractères maximal qui peut être écrit dans ce champ.
minlength	Le nombre de caractères minimal qui peut être écrit dans ce champ pour qu'il soit considéré comme valide.
pattern	Une expression rationnelle à laquelle doit correspondre le texte saisi pour être valide.
placeholder	Une valeur d'exemple qui sera affichée lorsqu'aucune valeur n'est saisie.
readonly	Un attribut booléen qui indique si le contenu du champ est en lecture seule.
size	Un nombre qui indique le nombre de caractères affichés par le champ.
spellcheck	Cet attribut contrôle l'activation de la vérification orthographique sur le champ ou si la vérification orthographique par défaut doit être appliquée.

- ▶ Exemple : [exemples\form\03_textfields.html](#)

Boutons radio et Boîtes à cocher

► Boutons radio

- Les éléments `<input>` dont l'attribut `type` vaut **radio** sont généralement utilisés pour construire des groupes d'options parmi lesquelles on ne peut choisir qu'une valeur.
- Les « boutons radio » sont représentés par des cercles remplis lorsqu'ils sont sélectionnés.

► Boîtes à cocher

- Les éléments `<input>` de type **checkbox** sont affichés sous la forme de boîtes à cocher qui sont cochées lorsqu'elles sont activées.
- Elles permettent de sélectionner une ou plusieurs valeurs dans un formulaire.

Attribut	Description
<code>checked</code>	Un attribut booléen. Si celui-ci est présent, la case à cocher sera cochée.

- Exemple : [exemples\form\04_radio_and_checkbox.html](#)

Liste déroulante

- ▶ L'élément HTML <**select**>
 - ▶ représente un contrôle qui fournit une liste d'options parmi lesquelles l'utilisateur pourra choisir
 - ▶ Attribut **multiple**
 - ▶ permet de choisir plusieurs options simultanément
 - ▶ Attribut **size**
 - ▶ indique le nombre d'options affichées en même temps

Liste déroulante

- ▶ L'élément HTML **<option>**
 - ▶ Chaque option est définie grâce à un élément **<option>** qui se situe à l'intérieur de l'élément **<select>**
 - ▶ Chaque élément **<option>** doit avoir un attribut **value** qui contient la valeur qui sera envoyée au serveur lorsque l'option est sélectionnée.
 - ▶ Si aucune valeur n'est fournie, la valeur par défaut sera le texte contenu dans l'élément.
 - ▶ Il est possible d'inclure un attribut **selected** sur un élément **<option>** afin que cette option soit sélectionnée par défaut au chargement de la page.
- ▶ L'élément HTML **<optgroup>**
 - ▶ Il est possible de regrouper plusieurs éléments **<option>** à l'intérieur d'éléments **<optgroup>** afin de créer des groupes d'options distincts.
- ▶ Exemples : [exemples\form\05_lists.html](#), [exemples\form\06_list_box.html](#)

Textarea

- ▶ L'élément HTML **<textarea>** représente un contrôle qui permet d'éditer du texte sur plusieurs lignes.

Attributs :

▶ **rows** et **cols**

- ▶ permettent de définir la taille exacte qui doit être occupée par l'élément **<textarea>**.
- ▶ Les navigateurs pouvant être différents, c'est une bonne idée que d'utiliser ces attributs pour garantir une certaine homogénéité.

▶ **wrap**

- ▶ indique la gestion des retours à la ligne et la façon d'afficher le texte saisi lorsque celui-ci atteint le bord de la zone du **<textarea>**

- ▶ Exemple : [exemples\form\07_textarea.html](#)

Étiquettes

- ▶ L'élément HTML **<label>** représente une légende pour un objet d'une interface utilisateur.
- ▶ Il peut être associé à un contrôle en utilisant l'attribut **for** ou en plaçant l'élément du contrôle à l'intérieur de l'élément **<label>**.
- ▶ Attribut **for**
 - ▶ L'identifiant (la valeur de l'attribut id) de l'élément de formulaire associé, appartenant au même document que l'élément label.
 - ▶ Le premier élément du document dont l'identifiant correspond est alors le contrôle étiqueté par l'élément.
- ▶ Exemple : [exemples\form\08_labels.html](#)

Groupe de contrôles et Légende

- ▶ L'élément HTML **<fieldset>** est utilisé afin de regrouper plusieurs contrôles interactifs ainsi que des étiquettes (**<label>**) dans un formulaire web.
- ▶ L'élément HTML **<legend>** représente une légende pour le contenu de son élément parent **<fieldset>**.
- ▶ Exemple : [exemples\form\09_fieldset.html](#)

Téléversement de fichier

- ▶ Les éléments <input> dont l'attribut type vaut "**file**" permettent à un utilisateur de sélectionner un ou plusieurs fichiers depuis leur appareil et de les téléverser (ou uploader) vers un serveur via un formulaire

Attribut	Description
accept	Un ou plusieurs identifiants de type de fichier décrivant les types de fichier autorisés (ex : .pdf, image/*, audio/*, video/*, ...).
multiple	Un attribut booléen qui, lorsqu'il est présent, indique que plusieurs fichiers peuvent être sélectionnés.

- ▶ Exemple : [exemples\form\10_fileupload.html](#)

Aligner des contrôles

- ▶ On utilise CSS pour aligner correctement les contrôles.
- ▶ Exemple : [exemples\form\11_align_controls.html](#)

Formater les contrôles

- ▶ On utilise CSS pour formater les contrôles
- ▶ Exemple : [exemples\form\12_format_controls.html](#)

Navigation au clavier et Raccourci clavier

► **tabindex**

- Une valeur numérique qui définit si le champ peut recevoir le focus via la navigation au clavier (en navigant avec la touche Tab) et la façon dont l'élément s'inscrit dans l'ordre de la navigation au clavier.

► **accesskey**

- Cet attribut fournit une indication permettant de générer un raccourci clavier pour l'élément courant. Cet attribut se compose d'une liste de caractères séparés par des espaces. Le navigateur doit utiliser le premier caractère qui existe selon la disposition du clavier utilisée.
- Exemple : [exemples\form\13_access_keys.html](#)

Validation de données

- ▶ Une des caractéristiques de HTML5 est la possibilité de valider la plupart des données utilisateur sans avoir recours à des scripts.
- ▶ Ceci se fait en utilisant des attributs de validation sur les éléments de formulaire
- ▶ Ils permettent de spécifier des règles pour une entrée de formulaire comme :
 - ▶ une valeur doit-elle être remplie ?
 - ▶ y a-t-il une longueur minimale et/ou maximale des données ?
 - ▶ doit-elle être un nombre, une adresse e-mail ou autre chose ?
 - ▶ doit-elle correspondre à un modèle ?
- ▶ Si les données saisies suivent toutes ces règles, elles sont considérées comme valides ;
- ▶ si ce n'est pas le cas, elles sont considérées comme non valides.

Validation de données

- ▶ L'attributs de validation
- ▶ **required**
 - ▶ La fonctionnalité de validation HTML5 la plus simple à utiliser est l'attribut required
 - ▶ si vous voulez rendre une entrée obligatoire, vous pouvez marquer l'élément en utilisant cet attribut.
 - ▶ Lorsque cet attribut est mis, le formulaire ne sera pas soumis (et affichera un message d'erreur) si l'entrée est vide (l'entrée sera également considérée comme invalide).
- ▶ **novalidate**
 - ▶ Cet attribut booléen indique si le formulaire doit être validé au moment de sa soumission.
 - ▶ S'il n'est pas défini, le formulaire sera validé lors de sa soumission.
 - ▶ Il peut être surchargé par l'attribut formnovalidate des éléments <button> ou <input> appartenant au formulaire.
- ▶ Exemple : [exemples\form\14_validation.html](#)

Expressions régulières (regex) pour la validation de données

- ▶ L'attribut **pattern** permet d'indiquer une expression rationnelle que devra respecter la valeur saisie afin d'être valide
- ▶ L'attribut **title** pourra être utilisé afin d'afficher une bulle d'informations qui explique les conditions à respecter.
- ▶ Quelques motifs:

Used for	Pattern
Password (6+ alphanumeric)	[a-zA-Z0-9]{6,}
Zip code (99999 or 99999-9999)	\d{5}([-\d{4})?
Phone number (999-999-9999)	\d{3}[-]\d{3}[-]\d{4}
Date (MM/DD/YYYY)	[01]?[0-3]\d/\d{4}
URL (starting with http:// or https://)	https?://.+
Credit card (9999-9999-9999-9999)	^\d{4}-\d{4}-\d{4}-\d{4}\$

Expressions régulières (regex)

► Quelques exemples de base :

- ▶ a — correspond à un caractère qui doit être un a (ni b, ni aa, etc.)
- ▶ abc — correspond à a, suivi de b, suivi de c.
- ▶ a^* — correspond au caractère a, absent ou présent plusieurs fois (+ correspond à un caractère une ou plusieurs fois).
- ▶ $[^a]$ — correspond à un caractère qui n'est pas un a.
- ▶ $a | b$ — correspond à un caractère qui est a ou b.
- ▶ $[abc]$ — correspond à un caractère qui est a, b ou c.
- ▶ $[^abc]$ — correspond à un caractère qui n'est pas a, b ou c.
- ▶ $[a-z]$ — correspond à tout caractère de la plage a-z, en minuscules seulement (utilisez $[A-Za-z]$ pour minuscules et majuscules et $[A-Z]$ pour les majuscules uniquement).

Expressions régulières (regex)

- ▶ **Quelques exemples de base :**
 - ▶ a.c — correspond à a, suivi par n'importe quel caractère, suivi par c.
 - ▶ a{5} — correspond à a, 5 fois.
 - ▶ a{5,7} — correspond à a, 5 à 7 fois, mais ni plus, ni moins.
- ▶ Vous pouvez utiliser des nombres ou d'autres caractères dans ces expressions, comme :
 - ▶ [-] — correspond à une espace ou un tiret.
 - ▶ [0-9] — correspond à tout nombre compris entre 0 et 9.
- ▶ Exemple : [exemples\form\15_regex.html](#)

Contenus auto-complétés (Datalist)

- ▶ L'élément HTML <**datalist**> contient un ensemble d'éléments <option> qui représentent les valeurs possibles pour d'autres contrôles.
- ▶ Une fois la liste de données affiliée au widget de formulaire, ses options s'utilisent comme complémentation du texte saisi par l'utilisateur
- ▶ Exemple : [exemples\form\16_datalist.html](#)

Email, URL et Tel

► Email

- ▶ Les éléments `<input>` dont l'attribut type vaut "**email**" permettent à un utilisateur de saisir et d'éditer une adresse mail ou,
- ▶ si l'attribut multiple est indiqué, une liste d'adresses mail.
- ▶ La valeur saisie est automatiquement validée afin de vérifier que le champ est vide ou que l'adresse (ou la liste d'adresses) est correcte.

► URL

- ▶ Les éléments `<input>` dont l'attribut type vaut "**url**" sont employées afin de permettre à un utilisateur de saisir ou d'éditer une URL.

► Tel

- ▶ Les éléments `<input>` dont l'attribut type vaut "**tel**" permettent à un utilisateur de saisir un numéro de téléphone.
- ▶ Contrairement aux contrôles utilisés pour `<input type="email">` et `<input type="url">`, la valeur saisie n'est pas automatiquement validée selon un format donné car les formats des numéros de téléphone varient à travers le monde.

▶ Exemple : [exemples\form\17_email_url_tel_controls.html](#)

Number et Range

▶ Number

- ▶ Les éléments `<input>` dont l'attribut type vaut **number** permettent à un utilisateur de saisir des nombres dans un formulaires.
- ▶ De tels contrôles incluent des mécanismes de validation natifs afin de rejeter les valeurs non-numériques.
- ▶ Le navigateur peut agrémenter le contrôle avec des flèches afin d'incrémenter/décrémenter la valeur grâce à la souris ou avec le doigt.

▶ Range

- ▶ Les éléments `<input>` dont l'attribut type vaut **range** permettent à l'utilisateur d'indiquer une valeur numérique comprise entre deux bornes.

Attribut	Description
max	La valeur maximale autorisée.
min	La valeur minimale autorisée.
step	Le pas utilisé pour incrémenter la valeur du champ. Cette valeur est utilisée pour l'interface utilisateur du contrôle et pour la validation de la valeur.

- ▶ Exemple : [exemples\form\18_number_range_controls.html](#)

Date et Time

► Date

- ▶ Les éléments `<input>` dont l'attribut type vaut **date** permettent de créer des champs permettant de saisir des dates (composées d'une année, d'un mois et d'un jour)

► Month

- ▶ Les éléments `<input>` dont l'attribut type vaut "**month**" permettent de créer des contrôles où l'utilisateur peut saisir un mois et année.
- ▶ La valeur associée à un tel élément suit le format "YYYY-MM", où YYYY représente l'année sur quatre chiffre et MM le mois sur deux chiffres.

► Week

- ▶ Les éléments `<input>` dont l'attribut type vaut **week** permettent de créer des champs de saisie où l'on peut saisir une année et le numéro de la semaine pendant cette année (allant de 1 à 52 ou 53, suivant la norme ISO 8601).

Date et Time

► Time

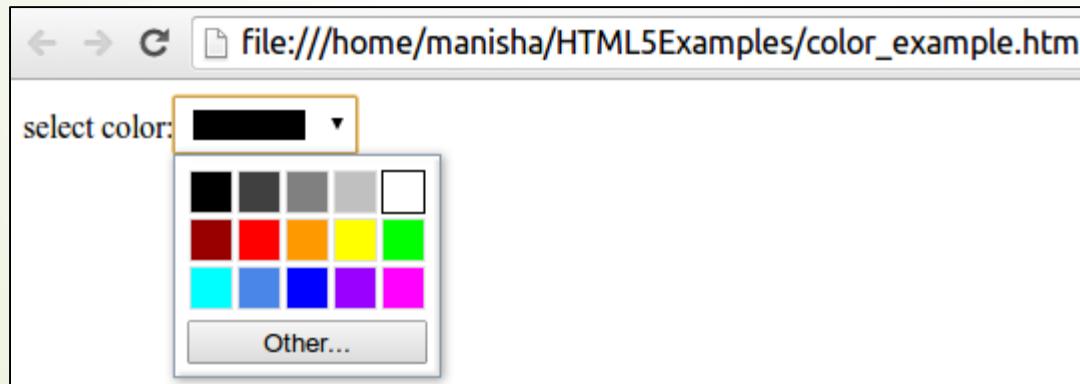
- Les éléments `<input>` dont l'attribut type vaut **time** permettent de créer des contrôles où l'utilisateur peut saisir une heure (avec des minutes et éventuellement des secondes).
- Exemple : [exemples\form\19_dates.html](#)

Champ de recherche

- ▶ Les éléments <**input**> dont l'attribut type vaut **search** permettent à un utilisateur de saisir des termes de recherche.
- ▶ Example : [exemples\form\20_search.html](#)

Couleur

- ▶ Les éléments <**input**> de type "color" permettent de sélectionner une couleur via
 - ▶ une interface (un sélecteur de couleur) ou
 - ▶ en saisissant le code hexadécimal de la couleur au format "#rrggbb".



- ▶ Example : [exemples\form\21_color.html](#)

Boîtes flexibles CSS

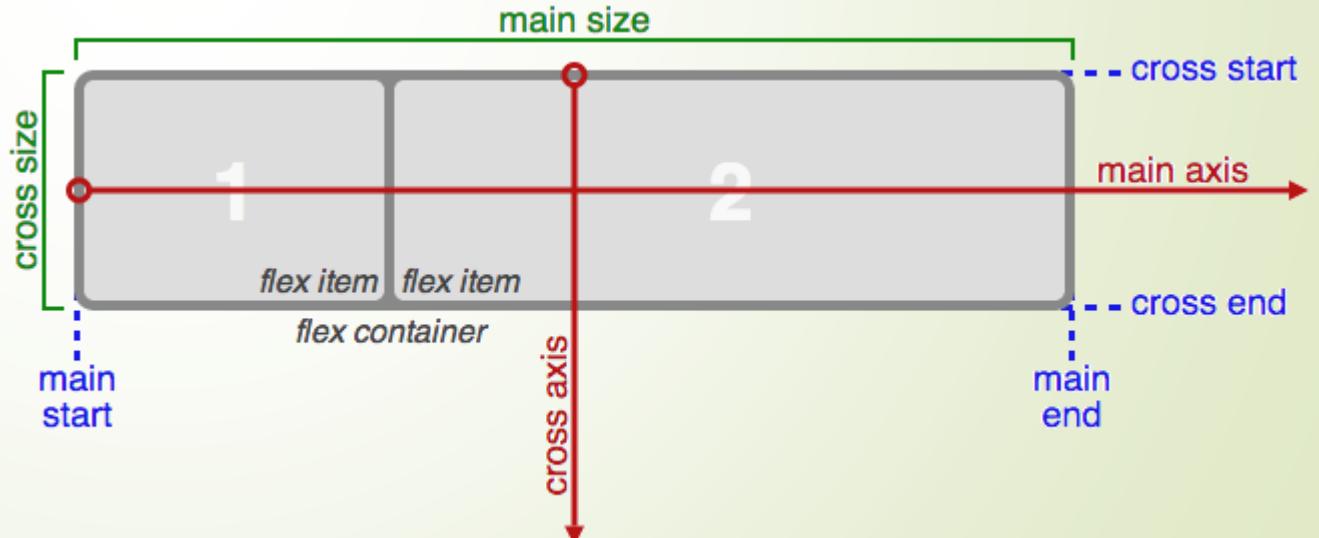
CSS Flexible Box Layout

Boîtes flexibles CSS

- ▶ Le module de disposition des boîtes flexibles CSS (CSS Flexible Box Layout)
 - ▶ Est un module de CSS qui définit un modèle de boîtes optimisé pour la conception des interfaces utilisateurs.
 - ▶ Groupes « flexibles »
 - ▶ Dans tous les navigateurs récents : <http://caniuse.com>
 - ▶ Permet de faire facilement, par exemple, des éléments de même hauteur
 - ▶ Doc et démos sympathiques : css-tricks.com

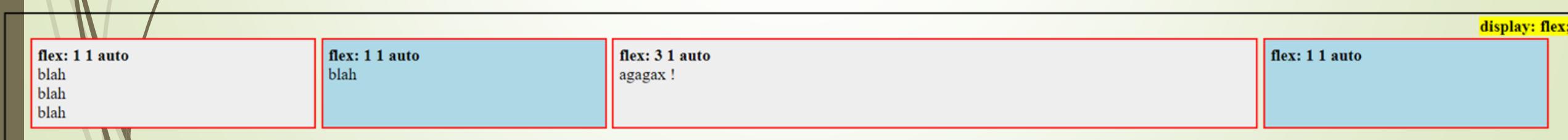
Principe

- ▶ On marque la boîte parente en **display: flex**
- ▶ Les boîtes filles sont organisée le long d'un axe principal (horizontal ou vertical)
- ▶ Dans l'autre dimension, elles auront par défaut la même taille : pour un axe horizontal, on aura des boîtes disposées côte à côte et de même hauteur
- ▶ on peut demander à répartir sur plusieurs lignes (ou colonnes)
- ▶ nombreux réglages possibles
(par exemple : alignement)



Exemple

- ▶ Cases de même hauteur sur une seule ligne
- ▶ L'élément parent a la propriété **display: flex**
- ▶ par défaut, il est organisé en lignes
- ▶ il fait par défaut une seule ligne
- ▶ Ses enfants se partageront l'espace selon la valeur de leur propriété **flex**
- ▶ dans l'autre direction (ici la verticale), ils auront par défaut la même hauteur



Autre exemple

- ▶ Ici, la valeur de **flex** n'est pas précisée pour les **enfants** ;
 - ▶ elle équivaut à **flex: 0 1 auto;**
 - ▶ On voit ici que les éléments se sont placés les uns à côté des autres, en prenant chacun la place dont il a besoin.
 - ▶ Ils ont tous la même hauteur.

Premier élément, Premier élément.
Second paragraphe

Second élément

Troisième élément

display: flex;

La propriété flex

- ▶ Elle est définie sur l'élément enfant
- ▶ 3 paramètres : flex-grow, flex-shrink et flex-basis
- ▶ **grow**
 - ▶ capacité de l'élément à utiliser de l'espace libre.
 - ▶ 0: l'élément ne grossit pas.
 - ▶ Un élément avec grow=2 grossit deux fois plus qu'un élément à 1.
- ▶ **shrink**
 - ▶ capacité de l'élément à rétrécir
 - ▶ fonctionne comme grow.
- ▶ **basis**
 - ▶ fixe la taille de départ (à partir de laquelle on essaie de grossir ou de rétrécir).
 - ▶ Peut être une dimension (comme 3cm ou 2em),
 - ▶ content, qui utilise le contenu de l'élément lui-même pour calculer sa taille, ou
 - ▶ auto, qui réparti l'espace total disponible dans cette alignment (c'est à dire celui laissé par tous les éléments).

Exemple 1

- ▶ Ici, on a trois éléments en **flex : 1 1 auto**
- ▶ On remarque que:
 - ▶ à cause de **auto**, les éléments ont une taille de base qui dépend de la largeur « naturelle » de leur contenu

flex: 1 1 auto;

Premier élément, Premier élément, Premier élément.

flex: 1 1 auto;

Second élément

flex: 1 1 auto;

Troisième élément

display: flex;

Exemple 2

flex: 1 1 auto;

Premier élément, Premier élément, Premier élément, Premier élément, Premier élément, Premier élément,

aaaaaaaaaaaa

flex: 1 1 auto;

flex: 1 1 auto;

bla bla bla

flex: 1 1 auto;

a a a a a a a a a a

flex: 1 1
auto;
du texte

```
flex: 1 1  
auto;  
bla bla bla
```

flex: 1 1 auto;

a a a a a
a a a a

flex: 1 1
auto;
du texte

```
flex: 1 1  
auto;  
bla bla  
bla
```

flex: 1 1 auto;

a a a
a a a
a a a
a a

```
flex:  
1 1  
auto;  
du  
texte
```

```
flex:  
1 1  
auto;  
bla  
bla  
bla
```



Auto et élément large

- ▶ Ici, nous sommes toujours sur des **flex à 1 1 auto**, mais le premier élément est trop long pour que tout tienne sur une ligne.
- ▶ Flex doit réduire les éléments (en utilisant la propriété shrink qui est à 1). L'espace assigné par défaut dépend de la largeur « naturelle » de l'élément, ainsi que de sa capacité à être réduit.
- ▶ Observez le premier et le second élément en particulier.

Basis à auto

- ▶ avec une **basis** à **auto**, les éléments se partagent l'espace restant disponible : on retire la taille naturelle des éléments de l'espace disponible, et on partage équitablement (en fonction de **grow**) l'espace qui reste.

display: flex;

flex: 1 1 auto; Premier élément. Premier.

flex: 1 1 auto; 2

flex: 1 1 auto; 3

Basis à 0em

- ▶ avec une **basis** de **0em**, les éléments se partagent équitablement l'espace **total** (les colonnes ont alors toutes la même largeur)

flex: 1 1 0em; Premier élément, Premier élément, Premier élément,
Premier élément

flex: 1 1 0em; Second élément

flex: 1 1 0em; Troisième élément

display: flex;

- ▶ Comparez avec :

flex: 1 1 auto; Premier élément, Premier élément, Premier élément, Premier élément

flex: 1 1 auto; Second élément

flex: 1 1 auto; Troisième élément

display: flex;

Basis fixée à un pourcentage

- ▶ On peut aussi décrire basis avec un pourcentage
- ▶ Ici, le troisième élément a une basis de 80%
- ▶ Du coup, il ne reste plus beaucoup d'espace pour les autres

flex: 1 1
0em;Premier
élément,
Premier
élément,
Premier
élément,
Premier
élément

flex: 1 1
0em;Second
élément

flex: 1 1 80%;Troisième élément

display: flex;

Basis fixée à un pourcentage

- En ajoutant la possibilité de placer les éléments en plusieurs couches (avec wrap), on pourra utiliser une largeur de 100% pour occuper tout une ligne :

```
flex: 1 1 0em;Premier élément, Premier élément, Premier élément, Premier élément
```

```
flex: 1 1 0em;Second élément
```

```
display: flex; flex-flow: row wrap;
```

```
flex: 1 1 100%;Troisième élément
```

```
...  
...  
...  
...  
...  
...  
...
```

Grow

- ▶ **grow** permet de fixer comment l'espace disponible sera réparti
- ▶ Un élément avec un grow de 4 récupérera 4 fois plus d'espace disponible qu'un élément avec un grow de 1
- ▶ Un élément avec un grow de 0 ne sera pas étendu
- ▶ Ici, avec une **basis** de 0em, si l'élément central a un coefficient grow de 4, il prend 4 fois plus de place qu'un élément avec un grow à 1

display: flex;

flex: 1 1 0em; Premier élément,
Premier élément, Premier élément,
Premier élément

flex: 4 1 0em; Second élément

flex: 1 1 0em; Troisième élément

Grow

- ▶ Même chose, mais avec **basis=auto**
 - ▶ s'il n'y a pas d'espace à répartir, c'est l'élément avec le plus de contenu qui sera le plus large.
 - ▶ S'il y a de l'espace, il sera partagé selon la valeur de grow

`flex: 1 1 auto; Premier élément, Premier élément, Premier élément, Premier élément`

`flex: 4 1 auto; Second élément`

`flex: 1 1 auto; Troisième élément`

`display: flex;`

Shrink à 0

- ▶ Si on place le **shrink** du premier élément à 0, il n'est plus capable de diminuer, et essaie d'être sur une seule ligne

display: flex;

display: flex;

flex: 1 1 auto; Second element

`flex: 1 1 auto;` Troisième élément

Utilisation de flex-flow

- ▶ Cette propriété se place sur le parent et permet :
 - ▶ des mises en page en ligne ou en colonne (row ou column)
 - ▶ d'activer ou de désactiver la possibilité de mettre en page sur plusieurs couches (wrap ou nowrap)
 - ▶ **par défaut : row nowrap**
 - ▶ est un raccourci pour les deux propriétés flex-direction et flex-wrap.

Utilisation de flex-flow - Exemple

- ici, les éléments sont tous en **flex 1 1 auto**, et **flex-flow** est **row wrap**
 - Le premier élément prend sa taille naturelle, qui est de la largeur de la page, et repousse les deux autres sur la ligne suivante
 - Dans le cas présent, les éléments ont les mêmes caractéristiques : c'est juste la taille du contenu de la première case qui déclenche la mise en page sur plusieurs rangées

display: flex; flex-flow: row wrap;

flex: 1 1 auto; Second élément

flex: 1 1 auto; Troisième élément

Utilisation de flex-flow - Exemple

- ▶ En enlevant du contenu au premier élément, on obtient :

flex: 1 1 auto;Premier élément, Premier élément

flex: 1 1 auto;Second élément

flex: 1 1 auto;Troisième élément

display: flex; flex-flow: row wrap;

Quelques essais en jouant sur les paramètres

- En fixant une basis différente d'auto ou zéro, on se donne une taille de départ en dessous de laquelle on aura recours à un saut de ligne

flex: 1 1 10em; Second élément

display: flex; flex-flow: row wrap;

flex: 1 1 10em; Troisième élément Troisième élément
Troisième élément Troisième élément Troisième élément
Troisième élément

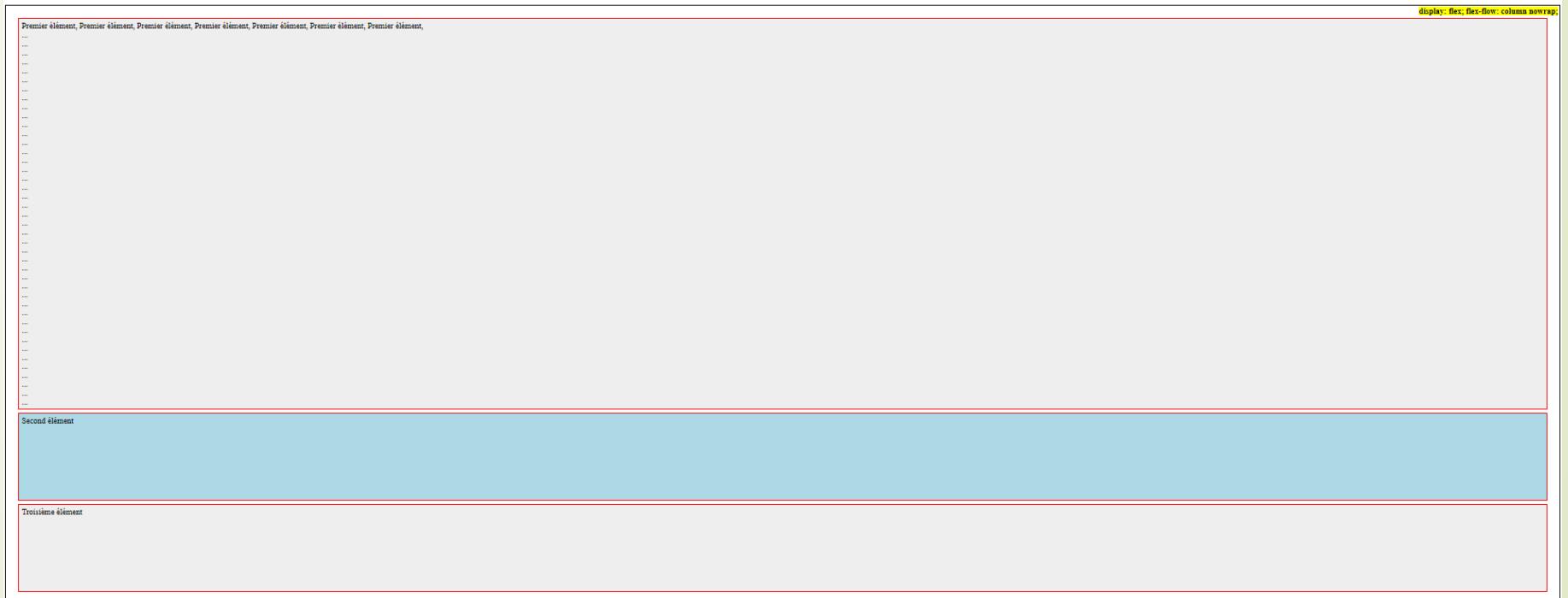
flex: 2 1 0em; Second élément

```
display: flex; flex-flow: row wrap;
```

flex: 2 1 0em; Troisième élément Troisième élément
Troisième élément Troisième élément Troisième
élément Troisième élément Troisième élément
Troisième élément Troisième élément Troisième
élément Troisième élément Troisième élément
Troisième élément Troisième élément Troisième
élément

Usage de flex-direction: column

- ▶ Dans le cas des colonnes, les éléments de la même colonne ont normalement une même largeur, et remplissent la colonne
- ▶ Contrairement à la largeur d'une ligne, la hauteur d'une colonne n'est pas bornées par les limites de l'écran



- ▶ L'utilisation de column wrap est beaucoup plus compliquée, et ne fonctionne bien que si les éléments ne sont pas trop larges

Utilisation pour une mise en page complexe

- ▶ Flow permet de créer très facilement des mises en pages complexes et adaptatives (« responsive »)
- ▶ Dans l'exemple qui suit, en-tête et pied de page prennent toute la largeur de l'écran ; le reste de la page est sur trois colonnes, sauf s'il y a moins de 30em d'espace disponible, auquel cas, les éléments se placent les uns en dessous des autres

```
display: flex; flex-flow: row wrap;
```

flex 1 1 100%; En-tête			
flex 0 0 10em; Côté gauche.	flex 1 1 10em; Contenu de la page.	flex 0 0 10em; Troisième élément	flex 0 0 10em; Pied de page

Autres propriétés

- ▶ **order**
 - ▶ Permet de modifier l'ordre des enfants d'un élément flex.
 - ▶ intéressant en combinaison avec les media-queries pour présenter les éléments dans des ordres différents selon la taille des supports
- ▶ **justify-content, align-items, align-content, align-self**
 - ▶ Permettent de régler la disposition des éléments les uns par rapport aux autres et leur espacement dans le conteneur.
- ▶ **inline-flex**
 - ▶ Un élément en display flex fonctionne extérieurement comme un bloc. Si on veut le placer à côté d'un autre élément, il faut utiliser inline-flex.

Optimisation pour le web mobile

Bonnes pratiques

Source : www.opquast.com



Interactions

- ▶ Les champs de saisie de type mail, URL, téléphone, nombre, recherche, mots de passe, heure et date sont dotés du type approprié.
- ▶ Les numéros de téléphone sont activables via le protocole approprié.
- ▶ Chaque champ de formulaire peut être activé via une action sur son étiquette.
- ▶ Chaque zone d'interaction tactile est de taille suffisante.
- ▶ Les espaces entre les zones d'interaction tactiles sont suffisants.
- ▶ Les espaces entre les zones d'interaction tactile et les bords de la zone d'affichage du navigateur sont suffisants.



Multimédia

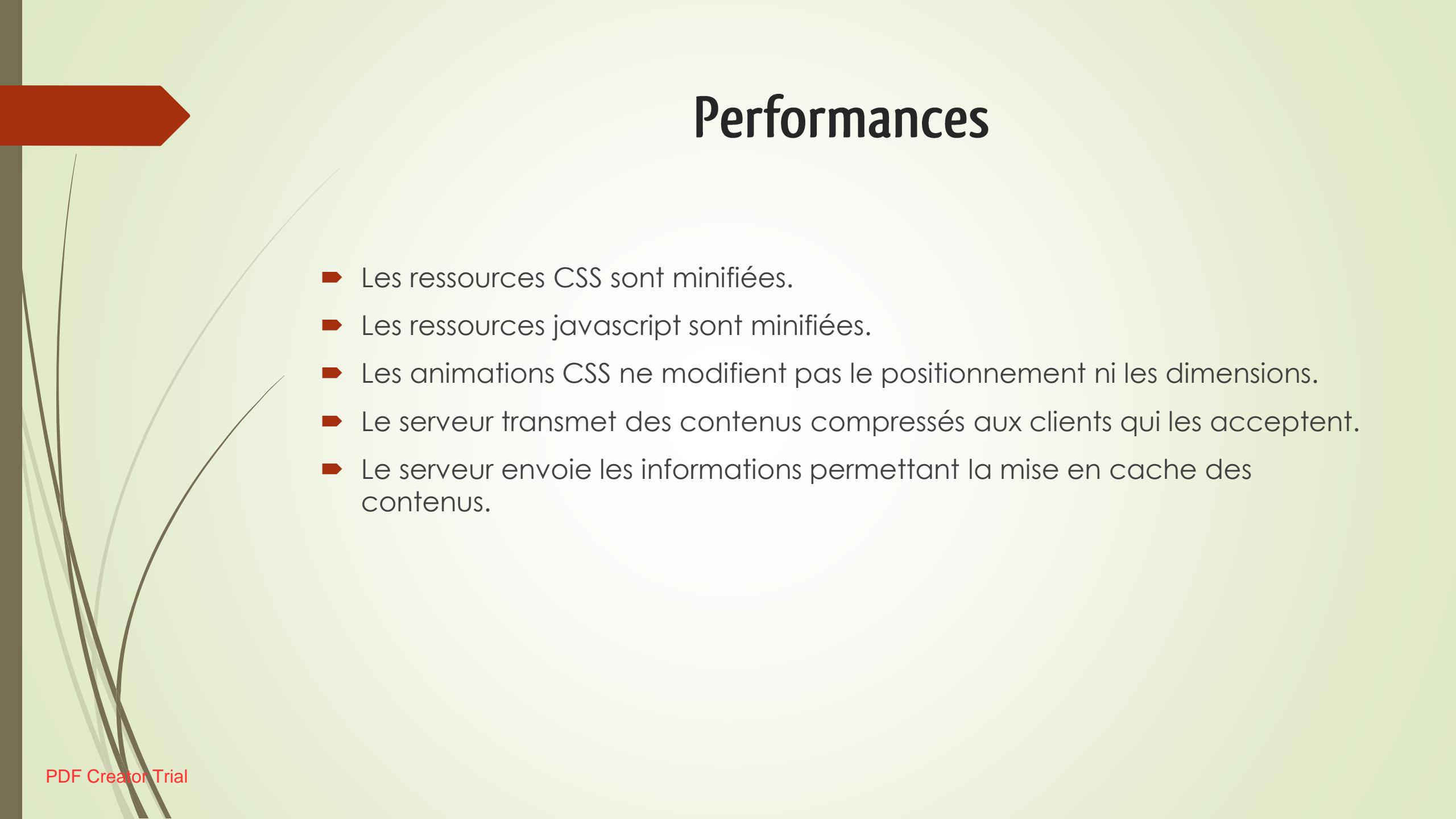
- ▶ Les contenus audiovisuels sont accompagnés d'une transcription textuelle équivalente.
- ▶ Les animations et contenus multimédia utilisent les éléments HTML5 appropriés.
- ▶ Les contenus multimédia ne sont téléchargés qu'à la demande explicite de l'utilisateur.

Navigation

- ▶ Le serveur ne force pas la redirection de la version desktop vers la version mobile.
- ▶ Il est possible de basculer depuis chaque page entre les versions dédiées (mobile, desktop).
- ▶ L'utilisateur est redirigé vers la page équivalente de la version desktop si la page n'existe pas dans la version mobile, et non vers l'accueil de celle-ci.
- ▶ Le site propose une version mobile générique si aucune version n'est prévue pour le client détecté.
- ▶ La cible et le rôle d'un lien, d'un bouton ou d'un élément de formulaire sont compréhensibles hors survol
- ▶ Les pictogrammes utilisés pour la navigation sont accompagnés d'une légende explicite.

Présentation

- ▶ Le site ne bloque pas les fonctionnalités de zoom du navigateur.
- ▶ Le site n'impose pas de rafraîchissement automatique de la page entière.
- ▶ L'utilisation du zoom du navigateur ne provoque pas de superpositions de contenus.
- ▶ Les alertes javascript invitant à l'installation d'une application mobile ne se produisent qu'une seule fois par session.
- ▶ La promotion de l'application mobile ne recourt ni aux alertes javascript ni aux splashscreens.
- ▶ Le contenu des pages s'adapte via les styles aux caractéristiques de l'appareil de l'utilisateur.



Performances

- ▶ Les ressources CSS sont minifiées.
- ▶ Les ressources javascript sont minifiées.
- ▶ Les animations CSS ne modifient pas le positionnement ni les dimensions.
- ▶ Le serveur transmet des contenus compressés aux clients qui les acceptent.
- ▶ Le serveur envoie les informations permettant la mise en cache des contenus.