

Formation HTML5-CSS3-JavaScript

Créeée et animée par Vincent CAILLIEREZ

Introduction

Présentation de la formation

- **Contenu**
 - **HTML/HTML5** : Code qui **structure** le contenu des pages web (“ceci est un titre”, “ceci est un paragraphe”...).
 - **CSS/CSS3 (feuilles de styles)** : Instructions de **mise en forme** des pages web (“ce texte doit apparaître en rouge”, “cette image est alignée à droite”...).
 - **JavaScript** : **Langage de programmation** permettant d’ajouter de l’interactivité aux pages web (“déplier/replier une boîte”, “faire défiler une galerie d’images”...)
- **Ordre** : HTML → CSS → JavaScript
- **HTML5 et CSS3 ?** Sont les dernières versions des langages HTML et CSS.

Esprit de la formation

- **Public**
 - Ceux qui souhaitent apprendre à concevoir et à construire des sites web **à partir de zéro**.
 - Ceux qui possèdent déjà un site web (créé via CMS, moteur de blog, etc) et qui souhaitent **maîtriser l'apparence** de leurs pages.
- **Focus sur :**
 - Les techniques les plus utilisées. On passe sous silence les autres (vous saurez trouver vous-même).
 - La mise en pratique.
- **Objectif** : Vous rendre autonomes.

Formateur & Stagiaires

- **Formateur (Vincent CAILLIEREZ)**
 - Développeur web depuis 15 ans. Formateur depuis 10 ans.
 - Spécialité : Angular (framework JavaScript).
- **Stagiaires**
 - Profil ?
 - Attentes ?
 - Projet particulier ?

Logiciels

- **Navigateur web moderne**

- Google Chrome



- Firefox



- **Éditeur de texte**

- Recommandé : Visual Studio Code - <https://code.visualstudio.com/>

- Sinon : Sublime Text, WebStorm...

RAPPEL : Fonctionnement du web

- Les pages d'un site web sont hébergées sur un **serveur web**, qui peut se trouver n'importe où dans le monde.
- Pour accéder à un site web, on utilise un **navigateur web** dans lequel on saisit une **URL**.
- L'URL permet d'identifier le **site** et la **page** précise à **télécharger et afficher** dans le navigateur.
Exemple : <https://www.amazon.fr/gp/css/homepage.html>
- NB. Il faut normalement installer un **serveur local** pour créer un site web **dynamique** sur sa machine, mais pour un site statique on peut s'en passer.

EXERCICE: Installer les logiciels

- Téléchargez et installez **Google Chrome** :
<https://www.google.fr/chrome/browser/desktop/>
- Téléchargez et installez **Visual Studio Code** :
<https://code.visualstudio.com/>
- Récupérez les **fichiers** nécessaires à la formation (exercices) en cliquant le lien communiqué par le formateur.
ATTENTION ! Lien temporaire. Merci de télécharger sur votre ordinateur tous les fichiers que vous souhaitez conserver.
- **IMPORTANT.** Vous ferez chaque exercice dans le répertoire dédié de exos_starters.

Partie 1 - HTML

1. Structure

Notion de structure
Introduction au balisage
Balises et éléments

Introduction

- Le web contient **différent types de pages** : journaux, catalogues, formulaires d'inscription.
- La **structure** de ces pages est essentielle. Elle indique au lecteur **comment interpréter ces documents**.
- Dans cette leçon :
 - Description de la structure d'une page web en HTML ;
 - Ajout de balises, ou d'éléments, à un document ;
 - Écriture d'une première page web.

Une redevance sur les smartphones exclue, mais pas pour les box

Le Monde.fr avec AFP | 02.09.2015 à 10h34 • Mis à jour le 02.09.2015 à 11h55

Abonnez-vous
à partir de 1 €

Réagir ★ Classer

Partager     

 Recommander

Partager

150 personnes recommandent ça. Soyez le premier parmi vos amis !



 LECTURE
ZEN

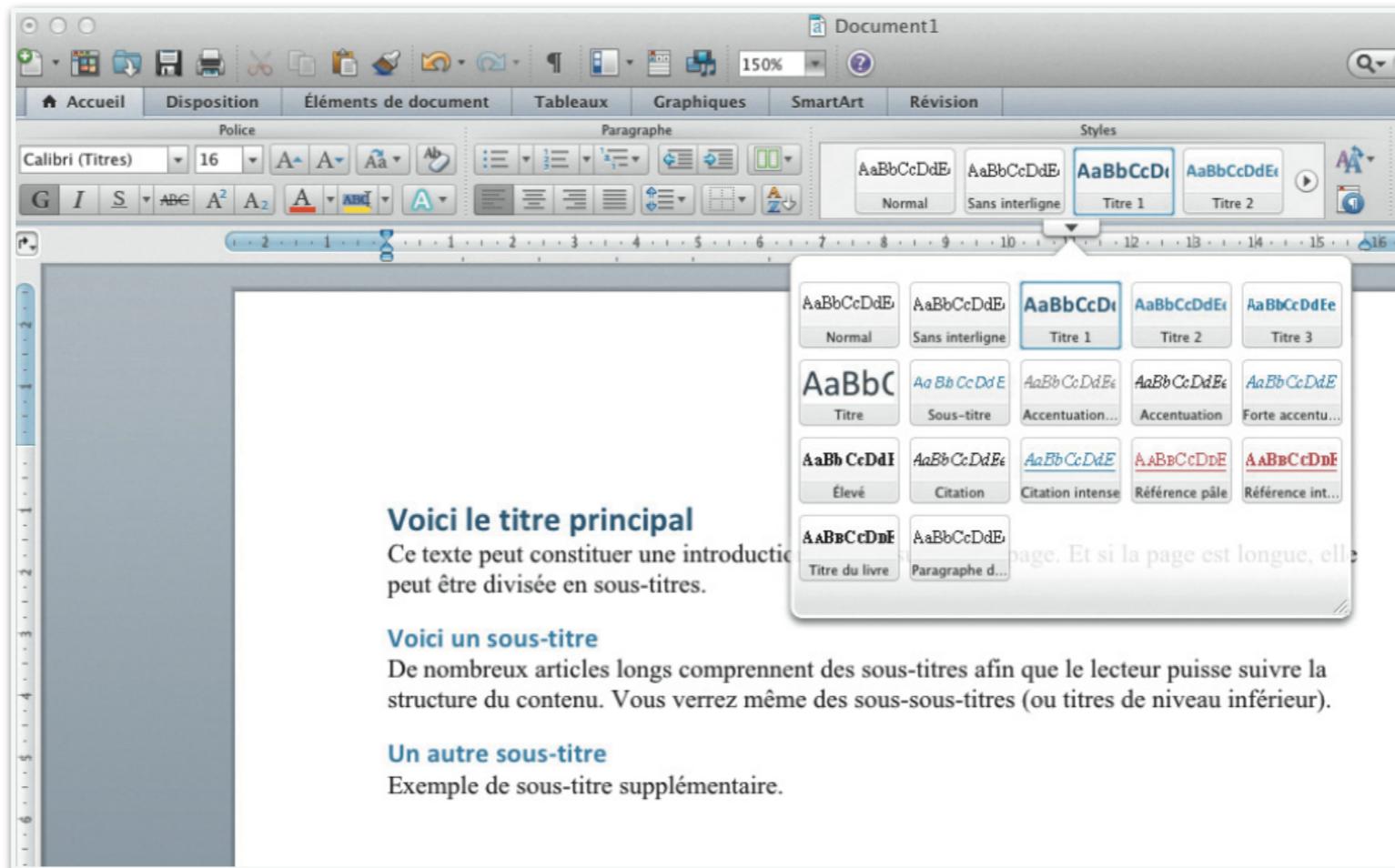
« Il n'est pas question de taxer les smartphones ou les tablettes comme je l'ai lu parfois. » La réponse de la ministre de la culture Fleur Pellerin à la nouvelle présidente de France Télévisions mercredi 2 septembre sur [France Info](#) est claire. La ministre a rejeté la piste de réforme de la redevance formulée par [Delphine Ernotte-Cunci](#) lundi, lui préférant une extension aux box de connexion à Internet.

« A la demande du président (...) j'ai étudié les moyens de moderniser l'assiette de la redevance en l'étendant non pas aux smartphones et aux tablettes, mais en regardant quels

Exemple : Structure d'un article de journal

- Titre
- Méta-données (auteur, date...)
- Photo
- Corps de texte
- Sous-titres
- Citations
- Etc.

Exemple : Structure d'un document WORD



- Les titres reflètent la **hiérarchie** des informations.
- Dans Word, les différents niveaux de titres sont exposés dans une galerie. Chaque titre a **son propre style**.

Structuration des pages en HTML

```
<html>
  <body>
    <h1>Voici le titre principal</h1>
    <p>Ce texte peut constituer une introduction pour la suite de la page. Et si la page est longue, elle peut être divisée en sous-titres.<p>
    <h2>Voici un sous-titre</h2>
    <p>De nombreux articles longs comprennent des sous-titres afin que le lecteur puisse suivre la structure du contenu. Vous verrez même des sous-sous-titres (ou titres de niveau inférieur).<p>
    <h2>Un autre sous-titre</h2>
    <p>Exemple de sous-titre supplémentaire. </p>
  </body>
</html>
```



- Le langage HTML utilise des **balises** (en bleu, dans l'exemple) pour structurer le texte. Une page HTML mélange donc **texte et balises**.
- Les balises viennent souvent **par deux** : balise ouvrante + balise fermante. Exemple : <**h1**>Titre principal</**h1**>

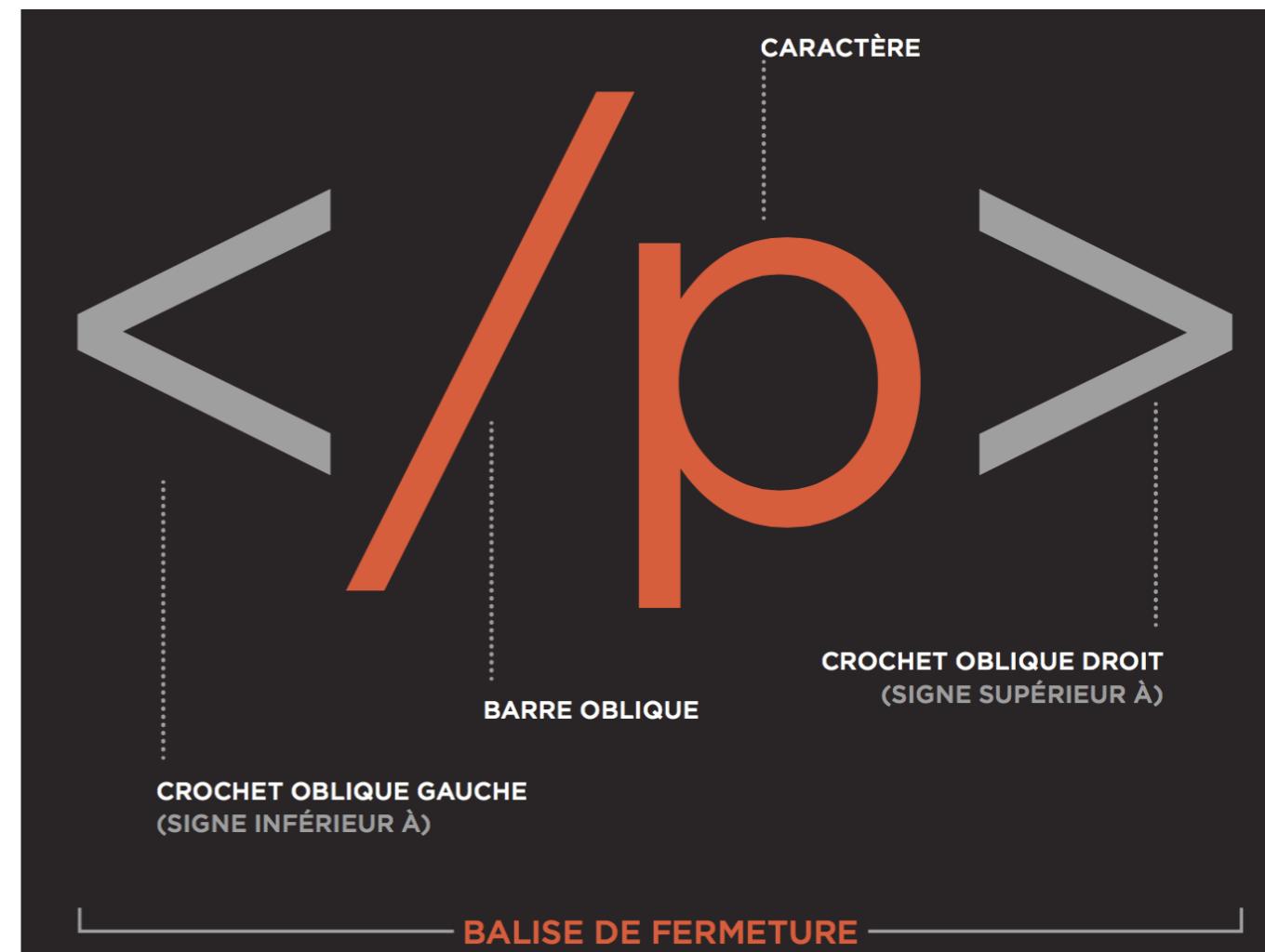
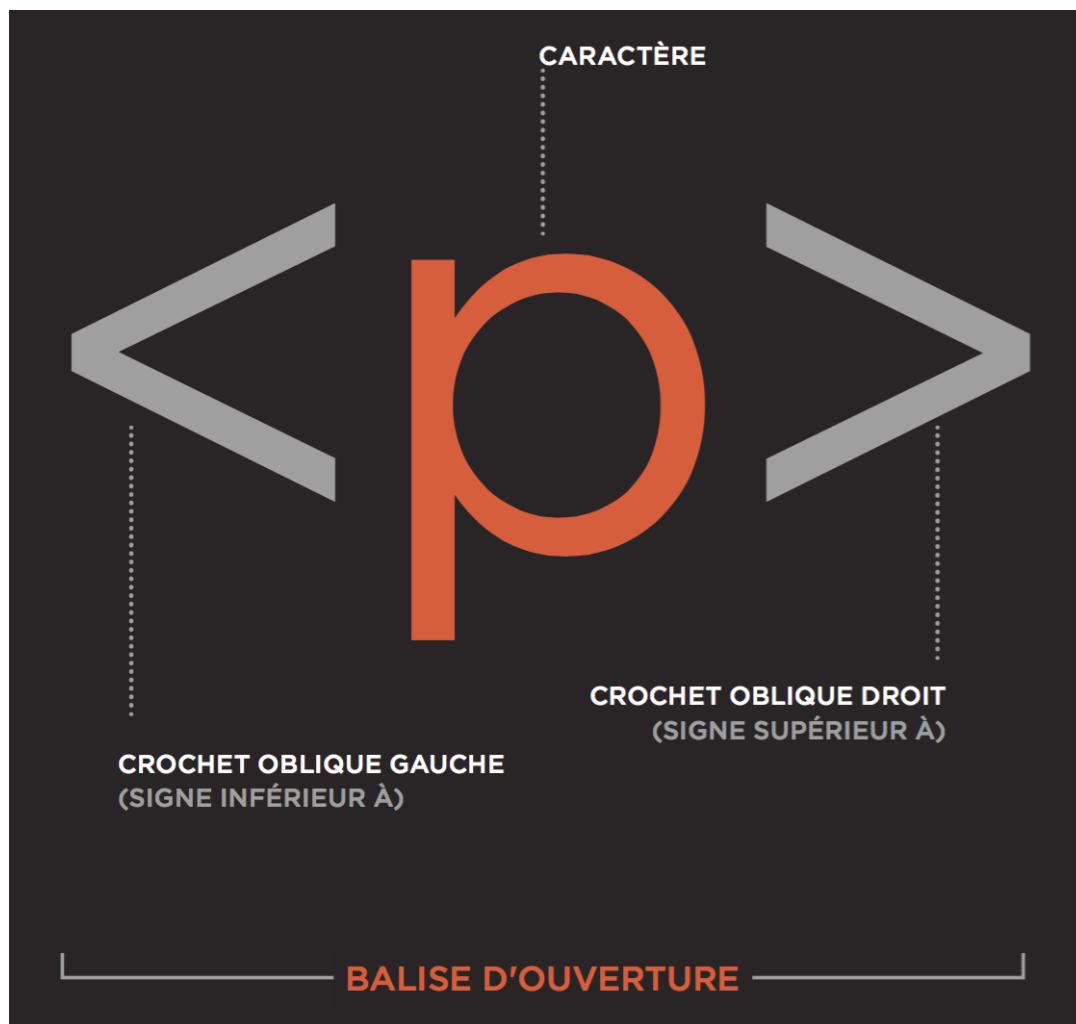
Détaillons le code précédent

The diagram illustrates the structure of an HTML document. It features a dark gray background with white rectangular boxes containing text. A green horizontal bar at the top represents the <html> tag, which encloses an orange <body> tag. The <body> tag contains several white boxes: one for the main title <h1>, one for introductory text <p>, one for a subtitle <h2>, one for detailed text <p>, another for a secondary subtitle <h2>, and one for additional text <p>. A red vertical bar on the right side represents the closing </body> tag, and a blue horizontal bar at the bottom represents the closing </html> tag.

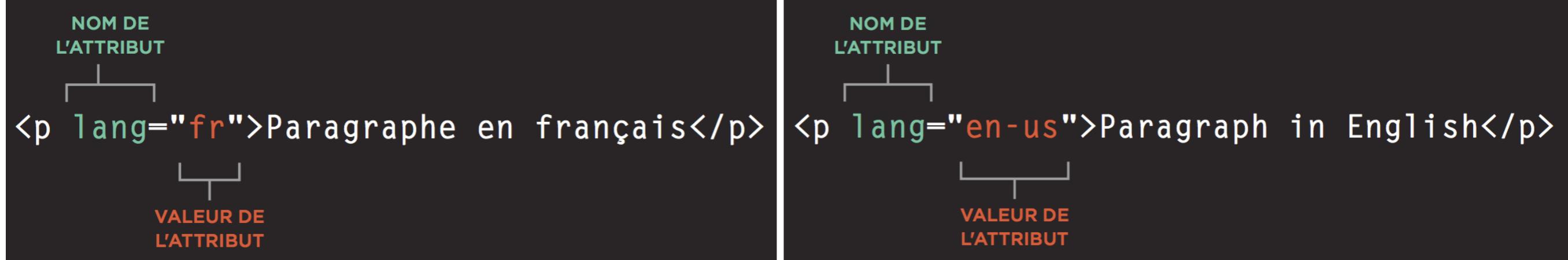
```
<html>
  <body>
    <h1>Voici le titre principal</h1>
    <p>Ce texte peut constituer une introduction pour la suite de la page. Et si la page est longue, elle peut être divisée en sous-titres.</p>
    <h2>Voici un sous-titre</h2>
    <p>De nombreux articles longs comprennent des sous-titres afin que le lecteur puisse suivre la structure du contenu. Vous verrez même des sous-sous-titres (ou titres de niveaux inférieurs).</p>
    <h2>Un autre sous-titre</h2>
    <p>Exemple de sous-titre supplémentaire.</p>
  </body>
</html>
```

- <html>...</html> : Page HTML
- <body>...</body> : Corps de la page
- <h1>...</h1> : Titre principal
- <h2>...</h2> : Sous-titre (titre de niveau 2)
- <p>...</p> : Paragraphe

Détail des balises



Attributs



- Apportent des **informations complémentaires** sur le contenu d'un élément.
- Placés dans la balise d'ouverture ; comprennent 2 parties, un **nom** et une **valeur**, séparées par un **signe égal =**.
- La valeur doit être placée entre **guillemets**.

Corps, en-tête et titre

```
<html>
  <head>
    <title>Voici le titre de la page</title>
  </head>
  <body>
    <h1>Voici le corps de la page</h1>
    <p>Le contenu du corps de la page est affiché
       dans la fenêtre principale.</p>
  </body>
</html>
```

Voici le corps de la page

Le contenu du corps de la page est affiché dans la fenêtre principale.

- **<body>** : Son contenu est affiché dans la fenêtre principale du navigateur.
- **<head>** : Avant la balise **<body>**, cet élément contient des informations **à propos de** la page, comme le titre, qui ne sont PAS affichées dans la fenêtre principale du navigateur.
- **<title>** : Son contenu est affiché dans la **barre de titre** (ou l'onglet) du navigateur.

EXERCICE: Première page HTML

- Grâce à votre éditeur de texte, créez un nouveau fichier appelé `index.html` dans le répertoire `premiere_page`. Si votre éditeur offre cette fonctionnalité, choisissez un fichier de type HTML.
- Entrez le code suivant dans le fichier :

```
<html>
  <head>
    <title>Ma première page web</title>
  </head>
  <body>
    Bienvenue sur ma première page web.
  </body>
</html>
```

- Consultez le fichier dans votre navigateur. Tout semble-t-il correct ?

Examiner le HTML des autres sites web

- Une technique pour apprendre le HTML est d'**examiner le HTML des sites existants.**
- **HTML STATIQUE.** Lorsque vous consultez un site web, cliquez sur le menu **Affichage > Source** du navigateur (ou équivalent) pour voir le code HTML statique (c. à d. téléchargé depuis le serveur).
- **HTML DYNAMIQUE.** Attention, pour les sites qui utilisent JavaScript et/ou Ajax, le HMTL est généré **dynamiquement** dans le navigateur. Utilisez la fonction “Inspecter” de la **Console Développeur** pour visualiser le HTML “live”.

Résumé

- Les pages HTML sont des **documents textuels**.
- HTML se fonde sur des **balises** (une suite de caractères placés entre des crochets obliques, < et >) pour donner un sens particulier aux informations qu'elles englobent. Les balises sont souvent appelées **éléments**.
- Les balises s'utilisent généralement par **paire**. La balise d'**ouverture** indique le début d'un contenu, la balise de **fermeture** en signale la fin.
- Les balises d'ouverture peuvent comprendre des **attributs**, qui apportent des **informations complémentaires** sur le contenu de l'élément.
- Les attributs sont constitués d'un **nom** et d'une **valeur**.
- Pour apprendre à écrire des pages web en HTML, vous devez connaître les **balises disponibles**, leur **rôle** et leurs **emplacements valides**.

2. Texte

Titres et paragraphes
Gras, italique et accentuation
Balisages structurel et sémantique

Introduction

- Lorsque vous créez une page web, vous ajoutez des **balises** à son **contenu**. Elles lui donnent une signification supplémentaire et permettent aux navigateurs de présenter aux internautes une structure de page appropriée.
- Dans cette leçon, nous nous focalisons sur **l'ajout du balisage** au texte affiché par les pages :
 - **Balisage structurel** : Éléments qui permettent de décrire les titres et les paragraphes.
 - **Balisage sémantique** : Fournit des infos supplémentaires, comme le point important dans une phrase, l'indication d'une citation.

Titres

```
<h1>Voici un titre principal</h1>
<h2>Voici un titre de niveau 2</h2>
<h3>Voici un titre de niveau 3</h3>
<h4>Voici un titre de niveau 4</h4>
<h5>Voici un titre de niveau 5</h5>
<h6>Voici un titre de niveau 6</h6>
```

Voici un titre principal

Voici un titre de niveau 2

Voici un titre de niveau 3

Voici un titre de niveau 4

Voici un titre de niveau 5

Voici un titre de niveau 6

- HTML propose **six “niveaux” de titres** : `<h1>` est employé pour les titres principaux, `<h2>`, pour les sous-titres, et ainsi de suite.
- Les navigateurs affichent le contenu des titres dans des **tailles différentes**. Celle d'un élément `<h1>` est la plus grande, celle d'un élément `<h6>`, la plus petite.

Paragraphes

< p > Un paragraphe comprend une ou plusieurs phrases qui forment un contenu autonome. Le début du paragraphe est signalé par une nouvelle ligne.
< /p >

< p > Le texte est plus facile à comprendre lorsqu'il est décomposé en unités. Par exemple, un livre peut avoir des chapitres. Les chapitres peuvent avoir des sous-sections. Chaque section pourra contenir un ou plusieurs paragraphes. < /p >

Un paragraphe comprend une ou plusieurs phrases qui forment un contenu autonome. Le début du paragraphe est signalé par une nouvelle ligne.

Le texte est plus facile à comprendre lorsqu'il est décomposé en unités. Par exemple, un livre peut avoir des chapitres. Les chapitres peuvent avoir des sous-sections. Chaque section pourra contenir un ou plusieurs paragraphes.

- Pour créer un **paragraphe**, placez le texte correspondant entre une balise **< p >** d'ouverture et une balise **< /p >** de fermeture.
- Par défaut, un navigateur affichera chaque paragraphe sur une **nouvelle ligne**, en ajoutant un **espace** entre ce paragraphe et le suivant.

Gras et Italique

```
<p>Voici comment mettre un mot en <b>gras</b>.  
  </p>
```

```
<p>Dans la description d'un produit, certaines  
  <b>caractéristiques clés</b> peuvent être en  
  gras.</p>
```

Voici comment mettre un mot en **gras**.

Dans la description d'un produit, certaines
caractéristiques clés peuvent être en gras.

```
<p>Voici comment mettre un mot en <i>italique</i>.  
  </p>
```

```
<p>Cette pomme de terre est une <i>Solanum  
  teberosum</i>.</p>  
<p>Le capitaine Cook est arrivé en Australie à bord  
  du navire <i>Endeavour</i>.</p>
```

Voici comment mettre un mot en *italique*.

Cette pomme de terre est une *Solanum teberosum*.

Le capitaine Cook est arrivé en Australie à bord du
navire *Endeavour*.

- Les mots placés entre les balises **** et **** sont affichés en **gras**.
- Les mots placés entre les balises **<i>** et **</i>** sont affichés en **italique**.
- Aujourd'hui, **** est préféré à ****, et **** est préférable à **<i>**.

Espace blanc

```
<p>La Lune s'éloigne de la Terre.</p>
<p>La Lune           s'éloigne de la Terre.</p>
<p>La Lune s'éloigne de
la Terre.</p>
```

La Lune s'éloigne de la Terre.
La Lune s'éloigne de la Terre.
La Lune s'éloigne de la Terre.

- **Multiples espaces contigus** dans le code source ==> un seul espace à l'affichage.
- **Saut de ligne** dans le code source ==> un seul espace à l'affichage.
- Ce phénomène est appelé la **fusion des espaces blancs**.
- Pour **forcer l'espace blanc**, utiliser l'espace insécable :

Saut de ligne et lignes horizontales

```
<p>Le poids de la Terre<br />augmente de cent tonnes  
par jour<br />en raison des poussières venues de  
l'espace.</p>
```

Le poids de la Terre
augmente de cent tonnes par jour
en raison des poussières venues de l'espace.

- Utilisez la balise **
** pour créer un **saut de ligne** au milieu d'un paragraphe. (Les paragraphes créés avec **<p></p>** sont déjà affichés automatiquement à la ligne.)
- Remarque : Pour créer une séparation entre différents contenus, par exemple un changement de sujet dans un livre ou une nouvelle scène dans une pièce de théâtre, vous pouvez ajouter une **ligne horizontale** à l'aide de la balise **<hr />**.

Balisage sémantique

N'affecte pas la structure des pages
mais ajoute des informations complémentaires

Renforcement et accentuation

```
<p><strong>Attention :</strong> des pickpockets  
sont à l'oeuvre dans cette zone.</p>  
<p>Ce jouet comprend de nombreuses petites pièces  
et n'est <strong>pas adapté aux enfants de  
moins de cinq ans</strong>.</p>
```

RÉ

Attention : des pickpockets sont à l'oeuvre dans cette zone.

Ce jouet comprend de nombreuses petites pièces et n'est **pas adapté aux enfants de moins de cinq ans.**

```
<p>Je <em>pense</em> que Jean est le premier.</p>  
<p>Je pense que <em>Jean</em> est le premier.</p>  
<p>Je pense que Jean est le <em>premier</em>.</p>
```

RÉS

Je *pense* que Jean est le premier.

Je pense que *Jean* est le premier.

Je pense que Jean est le *premier*.

- L'élément **** indique que son contenu est de **haute importance**. Par exemple, les mots de cet élément peuvent être énoncés en insistant fortement. (Affiché en **gras** par défaut.)
- L'élément **** indique une **accentuation** qui change subtilement le sens d'une phrase. (Affiché en **italique** par défaut.)

Citations

```
<blockquote cite="http://fr.wikipedia.org/wiki/  
Winnie_1%27ourson">  
<p>Ce que je préfère avec la pluie, c'est qu'elle  
cesse toujours. Enfin, parfois.</p>  
<p>Comme l'a dit A. A. Milne, <q>Il n'y a pas  
d'urgence, nous y arriverons un jour.</q></p>
```

Ce que je préfère avec la pluie, c'est qu'elle cesse toujours. Enfin, parfois.

Comme l'a dit A. A. Milne, "Il n'y a pas d'urgence, nous y arriverons un jour."

- L'élément **<blockquote>** s'utilise avec les **citations longues** qui occupent l'intégralité d'un paragraphe. Notez l'élément **<p>** qui se trouve dans **<blockquote>**. (Affiché indenté.)
- L'élément **<q>** s'emploie avec les **citations courtes** placées dans un paragraphe. (Affiché avec des guillemets)

Balises sémantiques diverses

- **Abréviation** — `<abbr>` :

<p>Le <abbr title="Professeur">Pr</abbr> Stephen Hawking est un physicien théoricien et cosmologiste britannique.</p>

- **Référence à un autre document** — `<cite>` :

<p><cite>Une brève histoire du temps</cite> par Stephen Hawking s'est vendu à plus de dix millions d'exemplaires dans le monde.</p>

- **Définition** (première fois qu'on explique une nouvelle terminologie) — `<dfn>` :

<p>Un <dfn>trou noir</dfn> est une région de l'espace d'où rien, pas même la lumière, ne peut s'échapper.</p>

- Et aussi : **détails sur l'auteur** (`<address>`), **modifications du contenu** (`<ins>`, `` et `<s>`).

En résumé

- Les éléments HTML servent à décrire la **structure de la page**, comme les titres, les sous-titres et les paragraphes.
- Ils fournissent également des **informations sémantiques**, par exemple le point important dans un texte, la définition d'un acronyme ou l'emploi d'une citation.

3. Listes

Listes numérotées

Listes à puces

Listes de définitions

3 types de liste en HTML

- **Listes ordonnées.** Chaque élément d'une liste ordonnée possède un **numéro**. La liste peut, par exemple, représenter les étapes d'une recette qui doivent être exécutées dans l'ordre ou un contrat dont chaque point doit être identifié par un numéro de section.
- **Listes non ordonnées.** Les éléments d'une liste non ordonnée sont marqués par une **puce** (à la place des chiffres, qui établissent un ordre).
- **Listes de définitions.** Ces listes sont constituées d'un ensemble de termes accompagnés de leurs définitions.

Listes ordonnées

```
<ol>
<li>Couper les pommes de terre en quatre.</li>
<li>Les faire cuire dans l'eau salée pendant 15
    à 20 minutes.</li>
<li>Faire chauffer le lait, le beurre et la
    muscade.</li>
<li>Égoutter les pommes de terre et les écraser
    en purée.</li>
<li>Incorporer le mélange à base de lait.</li>
</ol>
```

1. Couper les pommes de terre en quatre.
2. Les faire cuire dans l'eau salée pendant 15 à 20 minutes.
3. Faire chauffer le lait, le beurre et la muscade.
4. Égoutter les pommes de terre et les écraser en purée.
5. Incoporer le mélange à base de lait.

- Pour créer une **liste ordonnée**, utilisez l'élément ``.
- Chaque élément de la liste est placé entre une balise `` d'ouverture et une balise `` de fermeture (`li` signifie *list item*, c'est-à-dire élément de liste).

Listes non ordonnées

```
<ul>
  <li>1 kg de pommes de terre Oeil de perdrix.</li>
  <li>100 ml de lait.</li>
  <li>50 g de beurre salé.</li>
  <li>Muscade fraîchement râpée.</li>
  <li>Sel et poivre.</li>
</ul>
```

- 1 kg de pommes de terre Oeil de perdrix.
- 100 ml de lait.
- 50 g de beurre salé.
- Muscade fraîchement râpée.
- Sel et poivre.

- Pour créer une **liste non ordonnée**, utilisez l'élément ``.
- Chaque élément de la liste est placé entre une balise `` d'ouverture et une balise `` de fermeture, comme pour les listes ordonnées.

Listes de définitions

```
<dl>
  <dt>Sashimi</dt>
  <dd>Plat japonais de poisson cru découpé en fines lamelles, servi accompagné d'une sauce piquante.</dd>
  <dt>Balance</dt>
  <dd>Appareil qui sert à comparer des grandeurs, particulièrement des masses.</dd>
  <dd>Petit filet pour la pêche des crevettes et des écrevisses.</dd>
  <dt>Grana</dt>
  <dt>Grana padano</dt>
  <dd>Fromage italien à base de lait de vache, à pâte pressée cuite.</dd>
</dl>
```

Sashimi

Plat japonais de poisson cru découpé en fines lamelles, servi accompagné d'une sauce piquante.

Balance

Appareil qui sert à comparer des grandeurs, particulièrement des masses.

Petit filet pour la pêche des crevettes et des écrevisses.

Grana

Grana padano

Fromage italien à base de lait de vache, à pâte pressée cuite.

- Pour créer une **liste de définitions**, utilisez l'élément **<dl>**. Une telle liste est généralement constituée d'une suite de termes et de leurs définitions.
- L'élément **<dt>** contient le **terme en passe d'être défini**.
- L'élément **<dd>** contient la **définition du terme**.

Listes imbriquées

```
<ul>
  <li>Mousses</li>
  <li>Pâtisseries
    <ul>
      <li>Croissant</li>
      <li>Mille-feuille</li>
      <li>Palmier</li>
      <li>Profiterole</li>
    </ul>
  </li>
  <li>Tartes</li>
</ul>
```

- Mousses
- Pâtisseries
 - Croissant
 - Mille-feuille
 - Palmier
 - Profiterole
- Tartes

- Il est tout à fait possible de placer une seconde liste dans un élément **** de manière à créer une **sous-liste** ou **liste imbriquée**.
- Les navigateurs affichent les listes imbriquées en **augmentant leur indentation** par rapport à la liste englobante.

En résumé

- Il existe **3 types de listes HTML** : ordonnées, non ordonnées et de définitions.
- Les éléments des listes ordonnées sont **numérotés**.
- Les éléments des listes non ordonnées sont marqués par des **puces**.
- Les listes de définitions servent à donner les **définitions de différents termes**.
- Des listes peuvent être **imbriquées** à l'intérieur d'autres listes.

EXERCICE: Reproduire la mise en page

- Utilisez les balises abordées jusqu'à maintenant pour reproduire la mise en page suivante :

Salade Cajun

Proposée par Capu24

Temps de préparation : 15 minutes

Temps de cuisson : 10 minutes

Ingédients (pour 4 personnes)

- 200 g de dés de lardons fumés
- 50 g de riz
- 50 g de haricots rouges cuits
- 1 poivron rouge *de très petite taille*
- 2 cuillères à soupe :
 - De Miel
 - D'huile

Préparation de la recette

Faire cuire le riz et l'égoutter.

Couper le poivron en dés et l'oignon en rondelles.

Faire dorer les lardons dans une poêle puis y ajouter le miel, le vinaigre et l'huile. Saler et poivrer. Egoutter les lardons et réserver la sauce.

"Je me suis régalee avec cette recette."
— Capu24

- STARTER :
`balises_texte/index.htm`

4. Liens

Création de liens entre des pages

Liens vers d'autres sites

Liens de messagerie électronique

Introduction

- Les liens sont au cœur du Web, car ils permettent à l'internaute de **se déplacer entre les pages**, apportant là le concept de navigation ou de surf.
- Vous rencontrerez généralement les **différents types de liens** suivants :
 - Liens d'un site web vers un autre ;
 - Liens d'une page vers une autre au sein du même site web ;
 - Liens d'une partie d'une page vers une autre partie de la même page ;
 - Liens qui ouvrent une nouvelle fenêtre du navigateur ;
 - Liens qui déclenchent le lancement d'un programme de messagerie et envoient un nouveau message à quelqu'un.

Syntaxe d'un lien



IMDB

- La création d'un lien se fonde sur l'élément **<a>**. Les internautes pourront cliquer sur le contenu placé entre la balise d'ouverture **<a>** et la balise de fermeture ****. L'attribut **href** précise la page liée.

Liens vers d'autres sites

```
<p>Critiques de films :  
<ul>  
  <li><a href="http://www.premiere.fr">  
    Première</a></li>  
  <li><a href="http://www.telerama.fr">  
    Télérama</a></li>  
  <li><a href="http://www.critikat.com">  
    Critikat</a></li>  
  <li><a href="http://www.critique-film.fr">  
    Critique film</a></li>  
</ul>  
</p>
```

Critiques de films :

- [Première](#)
- [Télérama](#)
- [Critikat](#)
- [Critique film](#)

- Lorsqu'un lien doit cibler un autre site web, la valeur de l'attribut **href** sera l'**adresse web complète du site**, autrement dit une **URL absolue**.

Liens vers d'autres pages du même site

```
<ul>
<li><a href="index.html">Accueil</a></li>
<li><a href="apropos.html">A propos</a></li>
<li><a href="films.html">Films</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
```

- [Accueil](#)
- [A propos](#)
- [Films](#)
- [Contact](#)

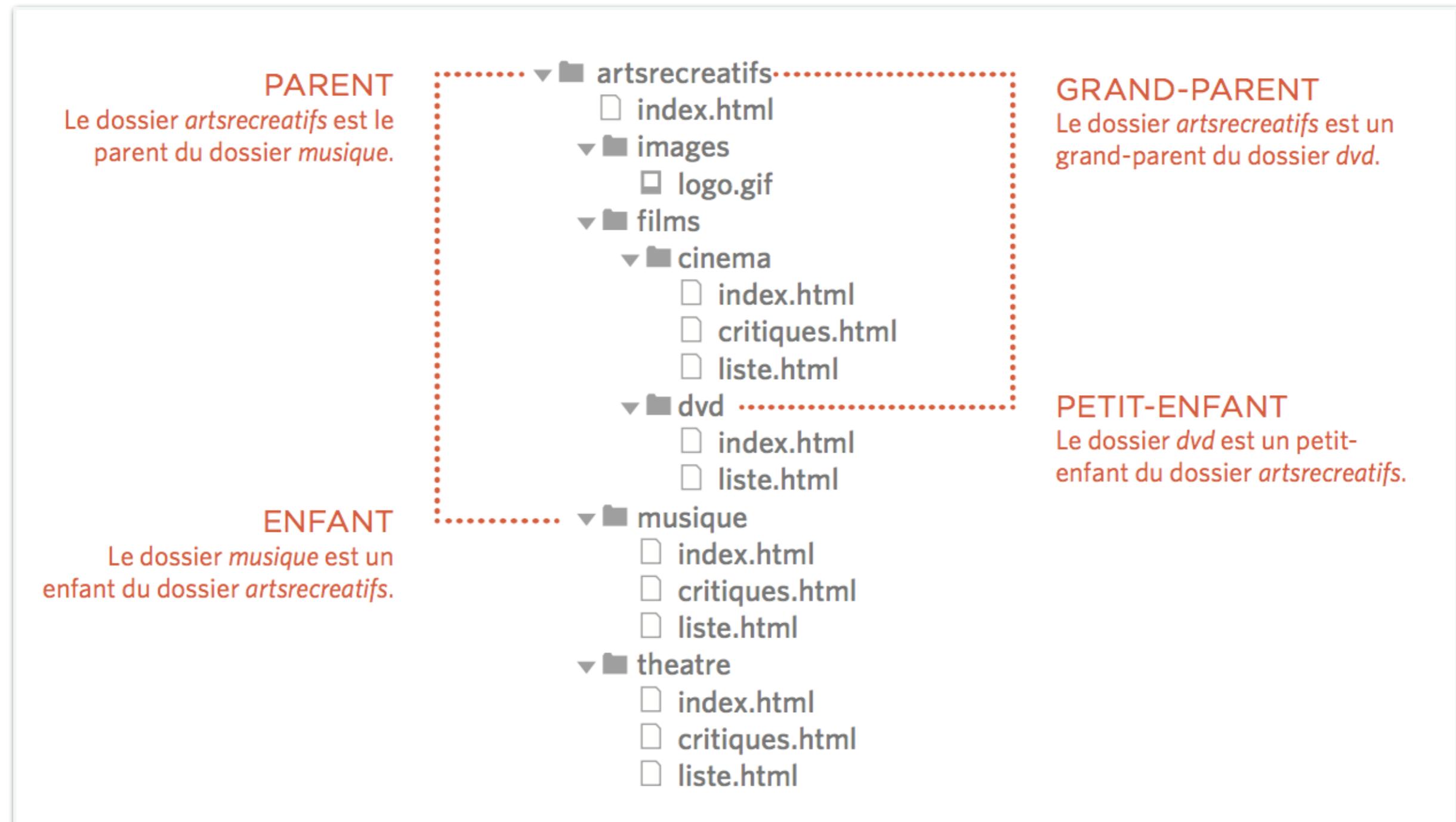
- Lorsque des liens vers des pages du même site web sont créés, le **nom de domaine n'est pas précisé** dans les URL. Il s'agit d'**URL relatives**.
- Si toutes les pages du site se trouvent **dans le même dossier**, les différents attributs **href** contiennent uniquement les **noms des fichiers**.

Structure de répertoires

- Chaque page et chaque image de votre site possède une URL :
URL = nom de domaine + chemin page ou fichier
Exemple : www.monsite.com/images/logo.gif
- L'organisation des fichiers sur le serveur web impacte donc la structure des URLs.
- **Conseils d'organisation :**
 - Placer chaque section du site dans un dossier séparé.
 - Regrouper les fichiers de même type (images, CSS...)

Structure de répertoires

Exemple



URLs relatives

- Peuvent être utilisées dans les liens vers des **pages du même site web**.
- Permettent d'indiquer au navigateur l'emplacement des fichiers à l'aide d'une **écriture raccourcie** (car le nom de domaine n'est pas précisé).

Type de lien	Exemple
Même dossier	Critiques
Dossier enfant	Liste
Dossier petit-enfant	Critiques
Dossier parent	Accueil
Dossier grand-parent	Accueil

Liens de messagerie électronique

```
<a href="mailto:jean@example.fr">Message à Jean</a>
```

Message à Jean

- Pour créer un lien qui **démarre le programme de messagerie électronique** de l'internaute et **envoie un message** à une adresse précisée, utilisez l'élément `<a>`, mais...
- ... la valeur de l'attribut `href` doit cette fois-ci commencer par `mailto:`, suivi de l'adresse électronique du destinataire du message.

Liens ouverts dans une nouvelle fenêtre

```
<a href="http://www.imdb.com" target="_blank">  
Internet Movie Database</a> (s'ouvre dans une  
nouvelle fenêtre)
```

[Internet Movie Database](#) (s'ouvre dans une
nouvelle fenêtre)

- Si vous souhaitez que la page ciblée par un lien s'ouvre **dans une nouvelle fenêtre**, définissez l'attribut **target** dans la balise [d'ouverture](#). Sa valeur doit être **_blank**.
- **ATTENTION.** Réfléchir si ce comportement est souhaitable. (Pas très “friendly” pour l'utilisateur.)

Liens vers des sections de la même page

```
<h1 id="top">Glossaire cinématographique</h1>
<a href="#arc_shot">Dorsale</a><br />
<a href="#interlude">Internégatif</a><br />
<a href="#prologue">Prégénérique</a><br /><br />
<h2 id="arc_shot">Dorsale</h2>
<p>Face postérieure de la pellicule, par opposition à la face antérieure, qui reçoit l'émulsion photosensible.</p>
<h2 id="interlude">Internégatif</h2>
<p>Tirage intermédiaire d'un interpositif. Utilisé
```

1. **Identifiez les cibles** en ajoutant l'attribut **id** aux éléments cibles, par exemple à des titres H1, H2... (l'attribut **id** est utilisable avec toutes les balises HTML). L'ID doit être unique et commencer par une lettre ou un underscore.
2. **Créez un lien** avec **<a>**, la valeur de l'attribut **href** doit commencer par **#**, suivi de la valeur de l'attribut **id** de l'élément cible.

En résumé

- Les liens sont créés à l'aide de l'élément `<a>`.
- L'élément `<a>` précise la **page liée** grâce à l'attribut `href`.
- Si le lien se fait vers une page du même site, il est préférable d'employer une **URL relative** à la place d'une **URL absolue**.
- Il est possible de créer un lien de façon à ouvrir un programme de **messagerie électronique** avec un nouveau message dont l'adresse indiquée est ajoutée dans le champ “À”.
- L'attribut `id` permet de créer un lien qui cible un élément d'une page.

EXERCICE : Liens

- **Ajoutez les liens aux endroits appropriés** dans tous les fichiers HTML du répertoire liens en commençant par index.html.
- Faites en sorte que le texte “Revenir en tête de page” qui apparaît en bas de index.html soit un lien qui **pointe vers le début de la page**.
- Faites en sorte que le lien vers Bing **s’ouvre dans une nouvelle fenêtre**.

5. Images

Ajout d'images dans les pages

Choix du format approprié

Optimisation des images pour le Web

Introduction

- Les raisons d'insérer une image dans une page web sont nombreuses : logo, photographie, illustration, diagramme ou graphique.
- Dans cette leçon :
 - **Inclusion d'images** dans les pages web en utilisant HTML ;
 - Choix d'un **format** d'image adapté ;
 - Affichage d'une image dans la **taille** appropriée ;
 - **Optimisation** d'une image afin d'accélérer le chargement des pages web.

Choix et Stockage

- **Choix des images**
 - Vous pouvez trouver des images dans des banques d'image comme www.istockphoto.com
 - Dans tous les cas, attention aux **copyrights**.
- **Stockage des images**
 - Sur le serveur, regroupez toutes les images du site dans un même dossier, par exemple **images**.
 - S'il y a beaucoup d'images, plusieurs sous-dossiers peuvent être créés.

Ajouter une image

```

```

- Pour ajouter une image dans une page, utilisez un élément ``. Il s'agit d'un élément vide, qui n'a donc **aucune balise de fermeture**. Il doit comprendre les deux attributs suivants :
- **src** : Cet attribut précise au navigateur l'**emplacement du fichier d'image**. Il s'agit habituellement d'une **URL** relative qui désigne une image sur votre site.
- **alt** : Cet attribut contient une **description textuelle de l'image**, qui permet d'expliquer l'image si elle ne peut pas être vue.
- **title** : Attribut contenant des informations complémentaires sur l'image.

Hauteur et largeur des images

```

```

- **height** : Hauteur de l'image en pixels.
- **width** : Largeur de l'image en pixels.
- Indiquer les dimensions de l'image au navigateur lui permet de l'afficher plus rapidement.

Emplacement des images dans le code



Il existe environ 10 000 espèces d'oiseaux qui vivent dans différents écosystèmes, de l'Arctique à l'Antarctique. De nombreuses espèces parcourent de longues distances au cours de leur migration annuelle et bien d'autres encore effectuent des déplacements irréguliers plus courts.



Il existe environ 10 000 espèces d'oiseaux qui vivent dans différents écosystèmes, de l'Arctique à l'Antarctique. De nombreuses espèces parcourent de longues distances au cours de leur migration annuelle et bien d'autres encore effectuent des déplacements irréguliers plus courts.

Il existe environ 10 000 espèces d'oiseaux qui vivent dans différents



écosystèmes, de l'Arctique à l'Antarctique. De nombreuses espèces parcourent de longues distances au cours de leur migration annuelle et bien d'autres encore effectuent des déplacements irréguliers plus courts.

- **Avant** un paragraphe (paragraphe débute sur une nouvelle ligne après l'image)
- **Au début** d'un paragraphe (1ère ligne s'aligne sur le bord inférieur de l'image)
- **Au milieu** d'un paragraphe (image placée entre les mots du paragraphe)

Alignement horizontal

```
<p>Il
```



Il existe environ 10 000 espèces d'oiseaux qui vivent dans différents écosystèmes, de l'Arctique à l'Antarctique. De nombreuses espèces parcourent de longues distances au cours de leur migration annuelle et bien d'autres encore effectuent des déplacements irréguliers plus courts.

Il existe environ 10 000 espèces d'oiseaux qui vivent dans différents écosystèmes, de l'Arctique à l'Antarctique. De nombreuses espèces parcourent de longues distances au cours de leur migration annuelle et bien d'autres encore effectuent des déplacements irréguliers plus courts.



- Autrefois, on alignait les images avec l'attribut **align="left"** ou **align="right"**.
- Aujourd'hui, l'alignement des images est géré en CSS (nous le verrons dans la suite de la formation).
- Même remarque pour l'alignement vertical (**align="top"**).

3 règles pour la création des images

1. Enregistrer les images dans le **format** approprié (JPEG, GIF ou PNG).
2. Enregistrer les images dans la **taille** appropriée (c. à d. avec la largeur et la hauteur qu'elles auront sur le site web).
3. Choisir une **RÉSOLUTION** appropriée (la plupart des écrans d'ordinateur ont une résolution de **72 pixels par pouce**, inutile de faire plus).

Formats d'image



- **JPEG** : Photos avec de nombreuses couleurs différentes.
- **GIF** : Couleurs uniformes, nombreux aplats (ex : logos, illustrations).
- **GIF animé** : plusieurs trames affichés en séquence ; permet de créer des animations simples.
- **Transparence** : GIF transparent ou PNG.

En résumé

- L'élément `` permet d'ajouter des images dans une page web.
- Vous devez **toujours définir l'attribut `src`** pour préciser la source de l'image et l'attribut `alt` pour décrire son contenu.
- Vous devez enregistrer les images dans les **dimensions** qu'elles auront sur la page web et choisir le **format** approprié.
- Il est préférable d'enregistrer les **photos** au format JPEG. Pour les **illustrations** ou les **logos** qui utilisent des couleurs uniformes, le format GIF est préconisé.

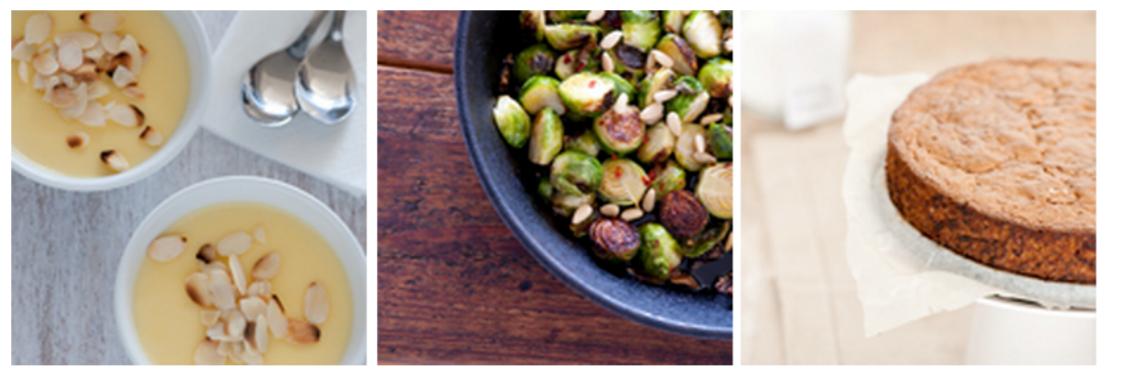
EXERCICE : Images

Notre recette du moment :



- Dans le fichier images/index.html, reproduire la mise en page ci-contre **en insérant des images** (fournies dans le répertoire “images”).

Plus de recettes :



6. Tableaux

Création de tableaux

Informations adaptées aux tableaux

Représentation de données complexes en tableaux

Introduction

- Différents types d'information ont intérêt à être présentés sous forme d'une grille ou d'un tableau : résultats sportifs, rapports d'inventaire ou horaires de train.
- Dans cette leçon :
 - Utilisation des **quatre éléments clés** de la création des tableaux ;
 - Représentation tabulaire de données complexes ;
 - Ajout de légendes à des tableaux.

Exemple de tableau

Cours A à Z	Palmarès	Actualités	Conseils	Consensus	Calendriers	Actionnaires	Introductions
France	International	Secteur	Capitalisation	Dividendes	PER	Indicateurs	Boursomap
Périmètre de sélection							
Indice SBF 120		Palmarès des		Depuis			
Hausses		veille					
<input type="checkbox"/> Éligibilité PEA		<input type="checkbox"/> Éligibilité PEA-PME					
Afficher							
Libellé		Dernier	Var.	Ouv.	+ haut	+ bas	Vol.
➡ ADOCIA	⌚	85.64 (c)	4.07%	82.57	86.00	80.95	37 119
➡ FONCIERE DES REGIONS	⌚	75.34 (c)	3.47%	74.89	76.78	74.63	164 366
➡ RUBIS	⌚	66.59 (c)	2.97%	65.00	67.20	65.00	193 911
➡ KERING (Ex: PPR)	⌚	147.95 (c)	2.56%	145.35	149.15	144.80	461 103
➡ ALSTOM	⌚	27.995 (c)	2.26%	28.070	28.485	27.915	2 222 223
➡ LVMH MOET VUITTON	⌚	147.15 (c)	2.05%	145.25	148.50	143.95	1 084 102
➡ ACCOR	⌚	42.035 (c)	1.73%	41.625	42.505	41.595	1 312 557
➡ SANOFI	⌚	87.24 (c)	1.44%	86.40	87.97	86.19	3 093 789
➡ LAGARDERE SCA N	⌚	24.250 (c)	1.40%	24.095	24.400	23.900	534 735

boursorama.com

- Un tableau présente des informations sous forme d'une **grille**. Exemples : rapports financiers, horaires des programmes TV et résultats sportifs.

Structure de base d'un tableau

```
<table>
  <tr>
    <td>15</td>
    <td>15</td>
    <td>30</td>
  </tr>
  <tr>
    <td>45</td>
    <td>60</td>
    <td>45</td>
  </tr>
  <tr>
    <td>60</td>
    <td>90</td>
    <td>90</td>
  </tr>
</table>
```

15	15	30
45	60	45
60	90	90

- **<table>** — La création d'un tableau se fonde sur l'élément **<table>**, avec une construction ligne par ligne.
- **<tr>** — Le **début de chaque ligne** est indiqué par la balise **<tr>** d'ouverture (**tr** signifie *table row*, ou ligne de tableau). La fin de la ligne est indiquée par une balise **</tr>** de fermeture.
- **<td>** — Chaque **cellule** d'un tableau est représentée par un élément **<td>** (**td** signifie *table data*, ou donnée de tableau). La balise **</td>** de fermeture indique la fin d'une cellule.

En-têtes de tableau

```
<table>
  <tr>
    <th></th>
    <th scope="col">Samedi</th>
    <th scope="col">Dimanche</th>
  </tr>
  <tr>
    <th scope="row">Billets vendus :</th>
    <td>120</td>
    <td>135</td>
  </tr>
  <tr>
    <th scope="row">Total des ventes :</th>
    <td>600 €</td>
    <td>675 €</td>
  </tr>
</table>
```

Samedi	Dimanche
Billets vendus :	120 135
Total des ventes :	600 € 675 €

- **<th>** — L'élément **<th>** s'utilise comme l'élément **<td>**, mais il représente l'**en-tête d'une colonne ou d'une ligne** (**th** signifie *table heading*, ou en-tête de tableau).
- Même si une cellule est **vide**, vous devez la représenter avec un élément **<td>** ou **<th>**, sinon la structure du tableau ne sera pas correcte.
- L'attribut **scope** de l'élément **<th>** permet d'indiquer si l'en-tête concerne une **colonne** ou une **ligne** (**col** ou **row**).

Chevauchement de colonnes

```
<table>
  <tr>
    <th></th>
    <th>9h00</th>
    <th>10h00</th>
    <th>11h00</th>
    <th>12h00</th>
  </tr>
  <tr>
    <th>Lundi</th>
    <td colspan="2">Géographie</td>
    <td>Maths</td>
    <td>Dessin</td>
  </tr>
  <tr>
    <th>Mardi</th>
    <td colspan="3">EPS</td>
    <td>Économie</td>
  </tr>
</table>
```

	9h00	10h00	11h00	12h00
Lundi	Géographie		Maths	Dessin
Mardi	EPS			Économie

- Les entrées d'un tableau doivent parfois **occuper (chevaucher) plusieurs colonnes.**
- L'attribut **colspan** est reconnu par les éléments **<th>** et **<td>** et indique le **nombre de colonnes** occupées par une cellule.

Chevauchement de lignes

```
<table>
  <tr>
    <th></th>
    <th>ABC</th>
    <th>BBC</th>
    <th>CNN</th>
  </tr>
  <tr>
    <th>18h00 - 19h00</th>
    <td rowspan="2">Film</td>
    <td>Comédie</td>
    <td>Journal</td>
  </tr>
  <tr>
    <th>19h00 - 2000</th>
    <td>Sport</td>
    <td>Actualités</td>
  </tr>
</table>
```

	ABC	BBC	CNN
18h00 - 19h00	Film	Comédie	Journal
19h00 - 2000		Sport	Actualités

- Les entrées d'un tableau doivent parfois **s'étendre vers le bas sur plusieurs lignes.**
- L'attribut **rowspan** est reconnu par les éléments **<th>** et **<td>** et indique le **nombre de lignes** occupées par une cellule.

```
<table>
  <thead>
    <tr>
      <th>Date</th>
      <th>Recettes</th>
      <th>Dépenses</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>1 janvier</th>
      <td>250</td>
      <td>36</td>
    </tr>
    <tr>
      <th>2 janvier</th>
      <td>285</td>
      <td>48</td>
    </tr>
    <!-- lignes supplémentaires -->
    <tr>
      <th>31 janvier</th>
      <td>129</td>
      <td>64</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td></td>
      <td>7824</td>
      <td>1241</td>
    </tr>
  </tfoot>
</table>
```

Tableaux longs

- Trois éléments facilitent la distinction entre le **contenu principal** du tableau et les **première** et **dernière** lignes (dont le contenu peut être différent).
- **<thead>** — L'**en-tête** du tableau se place dans un élément **<thead>**.
- **<tbody>** — Le **corps** du tableau occupe l'élément **<tbody>**.
- **<tfoot>** — Le **pied** du tableau se définit dans un élément **<tfoot>**.
- L'élément **<caption>... </caption>** permet d'ajouter une légende au tableau. Cet élément DOIT apparaître **juste après la balise <table> ouvrante**.

Ancien code (maintenant = CSS)

- **Largeur et espacement**
 - Attribut **width** pour définir (en pixels) la largeur du tableau et/ou de cellules individuelles. Balises compatibles : **<table>**, **<th>** et **<td>**.
 - Attribut **cellpadding** pour ajouter un espace (en pixels) à l'intérieur de chaque cellule du tableau, ainsi que l'attribut **cellspacing** pour créer un espace entre chaque cellule. Balise compatible : **<table>**.
- **Bordures et arrière-plans**
 - Attribut **border** sur les éléments **<table>** et **<td>** pour définir en pixels l'épaisseur de la bordure.
 - Attribut **bgcolor** pour définir les couleurs d'arrière-plan du tableau et de ses cellules. Sa valeur est généralement un code hexadécimal.

En résumé

- L'élément `<table>` est utilisé pour ajouter des **tableaux** dans une page web.
- Un tableau se construit **ligne par ligne**. Une ligne est créée par un élément `<tr>...</tr>`.
- Chaque ligne comprend une ou plusieurs **cellules** représentées par des éléments `<td>...</td>` (ou `<th>...</th>` pour les cellules d'en-tête).
- Les cellules d'un tableau peuvent **chevaucher plusieurs lignes ou colonnes** en définissant les attributs `rowspan` et `colspan`.
- Pour les **longs tableaux**, il est préférable de les décomposer en éléments `<thead>`, `<tbody>` et `<tfoot>`.

EXERCICE : Tableau

- Mettre en forme le texte contenu dans tableau/index.html de sorte qu'il ressemble à ça :

	Home starter hosting	Premium business hosting
Disk space	250mb	1gb
Bandwidth	5gb per month	50gb per month
Email accounts	3	10
Server	Shared	VPS
Support	Email	Telephone and email
Setup	Free	Free
FTP accounts	1	5
	Sign up now and save 10%!	

7. Formulaires

Collecte d'informations auprès des internautes
Différentes sortes de contrôles de formulaire

Introduction

- Traditionnellement, le terme “formulaire” fait référence à un document imprimé qui comprend des zones vides dans lesquelles vous écrivez des informations.
- HTML emprunte ce concept de formulaire pour faire référence à différents éléments qui permettent de **collecter des informations** auprès des internautes qui visitent un site.
- Dans cette leçon :
 - Création d'un formulaire sur un site web ;
 - Outils de collecte des données ;

Exemple de formulaire

- Le formulaire web le plus connu est probablement le **champ de recherche** affiché au milieu de la page d'accueil de Google :



- Outre la possibilité de recherche offerte aux internautes, les formulaires leur permettent également d'effectuer d'autres actions en ligne : **inscription** en tant que membre d'un site web, **achats en ligne** et **abonnement** à une lettre d'actualité ou une liste de diffusion, etc.

Contrôles de formulaire

AJOUT DE TEXTE

Champ de texte

Saisie d'une ligne de texte, tel un nom ou une adresse e-mail.

Marc

Mot de passe

Idem au champ de texte, mais avec masquage des caractères.

Zone de texte

Pour les textes longs, comme les messages et les commentaires.

Saisissez votre commentaire...

SÉLECTION

Boutons radio

Sélection d'une option parmi plusieurs.

Rock Pop Jazz

Cases à cocher

Sélection et désélection d'une ou de plusieurs options.

iTunes Last.fm Spotify

Listes déroulantes

Sélection d'une option dans une liste prédefinie.

iPod

ENVOI DE FORMULAIRE

Bouton d'envoi

Soumission des données du formulaire à une autre page web.

M'abonner

Bouton image

Idem au bouton d'envoi, mais avec affichage d'une image.

M'abonner

ENVOI DE FICHIER

Envoi d'un fichier

Envoi d'un fichier, par exemple une image, à un site web.

Parcourir...

Fonctionnement des formulaires

VOTEZ POUR VOTRE MUSICIEN DE JAZZ PRÉFÉRÉ

Votre nom :

Je vote pour :

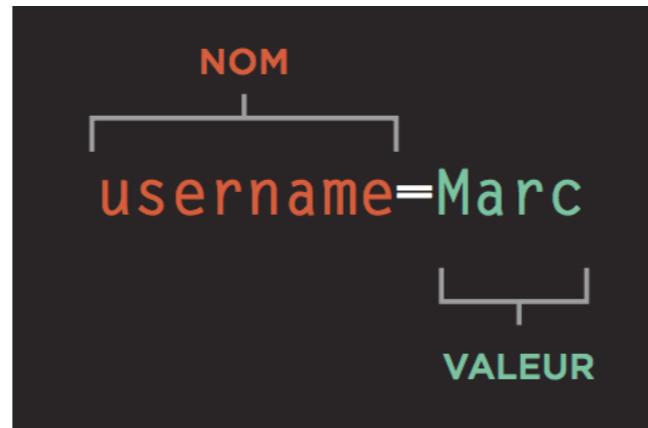
- Ella Fitzgerald
- Herbie Hancock
- John Coltrane
- Miles Davis
- Thelonious Monk

ENVOYER

1. L'internaute **remplit le formulaire**, puis appuie sur un bouton pour **soumettre** les informations au serveur.
2. Les noms des contrôles de formulaire sont **envoyés au serveur**, accompagnés des valeurs saisies ou sélectionnées par l'internaute.
3. Le **serveur traite les informations reçues** à l'aide d'un programme écrit, par exemple, en PHP, C#, VB.net ou Java. Il peut également les placer dans une base de données.
4. Le serveur **crée une nouvelle** page fondée sur les informations reçues et la renvoie au navigateur.

Fonctionnement des formulaires (2)

- Un formulaire peut comprendre **plusieurs contrôles**, chacun récoltant des informations différentes. Le serveur doit pouvoir **mettre en correspondance** les données saisies et les éléments du formulaire :



- Pour différencier les données saisies, les informations sont envoyées du navigateur au serveur sous forme de **couples nom-valeur**. Dans notre exemple, voici les couples nom-valeur envoyés au serveur :

username=Marc vote=Herbie

Structure d'un formulaire

```
<form action="http://www.exemple.fr/subscribe.php"
      method="get">
  <p>Emplacement des contrôles du formulaire.
  </p>
</form>
```

Emplacement des contrôles du formulaire.

- **<form>** — Les contrôles de formulaire sont placés dans un élément **<form>**. Celui-ci doit toujours définir l'attribut **action**. En général, les attributs **method** et **id** le seront également.
- **action** — Tout élément **<form>** a besoin d'un attribut **action**. Sa valeur précise l'**URL de la page qui recevra les informations du formulaire** lorsqu'il sera soumis.
- **method** — Il existe **deux méthodes d'envoi** des formulaires : **get** et **post**. Avec la méthode **get**, les valeurs saisies dans le formulaire sont ajoutées à la fin de l'URL indiquée par l'attribut **action**. Avec la méthode **post**, les valeurs sont envoyées dans ce que l'on appelle **en-têtes HTTP**.

Champ de texte

```
<form action="http://www.exemple.fr/login.php">
    <p>Nom d'utilisateur :
        <input type="text" name="username" size="15"
               maxlength="30" />
    </p>
</form>
```

Nom d'utilisateur :

- **<input>** — L'élément **<input>** permet de créer **différents contrôles de formulaire**. La valeur de son attribut **type** détermine le type du champ qui sera ajouté.
- **type="text"** — Lorsque l'attribut **type** vaut **text**, il crée un champ de saisie d'une seule ligne de texte.
- **name** — Lorsque l'internaute saisit des informations dans un formulaire, le serveur doit être capable de **différencier chaque contrôle de formulaire** qui fournit les données. Par exemple, dans un formulaire d'ouverture de session, le serveur doit pouvoir déterminer le nom d'utilisateur et le mot de passe saisis. C'est pourquoi chaque contrôle de formulaire exige un attribut **name**. Sa valeur identifie le contrôle et est **envoyée au serveur conjointement aux données**.

Mot de passe

```
<p>Mot de passe :  
  <input type="password" name="password" size="15"  
        maxLength="30" />  
</p>
```

Mot de passe :

- **<input type="password">** — Lorsque l'attribut **type** vaut **password**, il crée un contrôle qui présente les caractéristiques d'un champ de saisie, excepté que **les caractères sont masqués**.
- De cette manière, quiconque regarde par-dessus votre épaule ne pourra pas voir les informations sensibles que vous saisissez.

Zone de texte

```
<form action="http://www.exemple.fr/comments.php">  
  <p>Que pensez-vous de ce concert ?</p>  
  <textarea name="comments" cols="20" rows="4">  
    Saisissez votre commentaire...</textarea>  
</form>
```

Que pensez-vous de ce concert ?

Saisissez votre commentaire...

- L'élément `<textarea>` permet de créer une **zone de saisie de texte multiligne**. Contrairement aux autres éléments de saisie, celui-ci n'est pas vide. Vous devez donc inclure des **balises d'ouverture et de fermeture**.
- Le texte qui apparaît entre les balises `<textarea>...</textarea>` sera affiché dans la zone de texte lors du chargement de la page. Si l'internaute ne supprime pas le texte placé entre ces balises, le message envoyé au serveur contiendra ce texte et celui saisi ensuite.
- L'attribut `cols` indique la largeur de la zone de texte, en nombre de caractères. L'attribut `rows` précise le nombre de lignes verticales que la zone doit occuper.

Boutons radio/checkbox

```
<form action="http://www.exemple.fr/profile.php">  
  <p>Choisissez votre genre de musique préféré :<br />  
    <input type="radio" name="genre" value="rock" checked="checked" /> Rock  
    <input type="radio" name="genre" value="pop" />  
      Pop  
    <input type="radio" name="genre" value="jazz" />  
      Jazz  
  </p>  
</form>
```

Choisissez votre genre de musique préféré :

Rock Pop Jazz

- **<input type="radio">** — Les boutons radio permettent à l'internaute de **sélectionner une option parmi plusieurs**.
- **name** — L'attribut **name** est envoyé au serveur accompagné de la valeur de l'option sélectionnée. Lorsque le formulaire propose de répondre à une question via des boutons radio, **la valeur de name doit être identique pour l'ensemble des boutons radio qui servent à répondre à la question**.
- **value** — L'attribut **value** précise la **valeur envoyée au serveur** lorsque l'option correspondante est sélectionnée. Pour que le serveur sache quelle option l'internaute a sélectionné, chaque bouton d'un groupe doit avoir une valeur différente.
- **checked** — L'attribut **checked** permet d'indiquer l'**option sélectionnée** au moment du chargement de la page. Sa valeur est **checked**. Un seul bouton radio du groupe doit définir cet attribut.

Liste déroulante

```
<form action="http://www.example.fr/profile.php">
<p>Sur quel appareil écoutez-vous de la musique ?</p>
<select name="devices">
  <option value="ipod">iPod</option>
  <option value="radio">Radio</option>
  <option value="computer">Ordinateur</option>
</select>
</form>
```

Sur quel appareil écoutez-vous de la musique ?

A screenshot of a dropdown menu. The visible options are "iPod", "Radio", and "Ordinateur". The option "Radio" is highlighted with a blue background, indicating it is selected.

- Une liste déroulante permet à l'internaute de **sélectionner une option parmi celles prédefinies**.
- **<select>** — L'élément **<select>** sert à créer une liste déroulante. Il comprend deux éléments **<option>** ou plus.
- **<option>** — L'élément **<option>** est utilisé pour définir les **options parmi lesquelles l'internaute pourra choisir**.
- **value** — Pour préciser la valeur qui sera envoyée au serveur avec le nom du contrôle lors de la sélection de l'option, l'élément **<option>** passe par l'attribut **value**.
- **selected** — L'attribut **selected** sert à indiquer que l'option sera sélectionnée au chargement de la page. Sa valeur doit être **selected**.

Bouton d'envoi

```
<form action="http://www.exemple.fr/subscribe.php">
  <p>Abonnez-vous à notre liste de diffusion :</p>
  <input type="text" name="email" />
  <input type="submit" name="subscribe"
    value="M'abonner" />
</form>
```

Abonnez-vous à notre liste de diffusion :

 M'abonner

- **<input type="submit">** — Le bouton d'envoi est utilisé pour **soumettre le formulaire au serveur**.
- **name** — L'attribut **name** est reconnu, mais il n'est pas obligatoire.
- **value** — L'attribut **value** détermine le **texte qui s'affiche dans le bouton**. Il est préférable de préciser ce texte car, sur certains navigateurs, la valeur par défaut est “Soumettre la requête”, ce qui n'est pas nécessairement adapté à tous les formulaires.

Formulaire - Autre

- **Étiquette des contrôles :**
`<label for="nomDuChamp">Mot de passe</label>`
- **Boutons** - L'élément `<button>` a été ajouté afin d'offrir une plus grande maîtrise sur l'aspect des boutons.
- **Champs caché** - Contrôle pas affiché sur la page, mais dont la valeur est envoyée au serveur :
`<input type="hidden" name="referer" value="home" />`
- **Regroupement d'éléments** (utile avec les longs formulaires) :
 - Pour regrouper des contrôles de formulaire connexes, placez-les dans un élément `<fieldset>`.
 - L'élément `<legend>` peut être placé directement après la balise `<fieldset>` d'ouverture. Il contient une légende qui permet de décrire l'objectif du groupe de contrôles de formulaire.

Résumé

- Si vous souhaitez **collecter des informations** auprès des internautes qui se rendent sur votre site, vous avez besoin d'un **formulaire**, construit à l'aide d'un élément `<form>`.
- Les informations saisies dans un formulaire sont envoyées sous forme de **couples nom-valeur**.
- Un **nom** est attribué à chaque contrôle de formulaire et le **texte** saisi par l'internaute ou les **valeurs** qu'il sélectionne sont **envoyés au serveur**.
- **HTML5** apporte de **nouveaux contrôles** qui permettent aux internautes de remplir plus facilement les formulaires, par ex champ “date”, “email”, etc. (non abordés dans cette formation).

EXERCICE : Formulaire

- Créez un formulaire valide à partir du texte contenu dans formulaire/index.html, de sorte qu'il ressemble à ça :

Vos informations :

Nom :

E-mail :

Votre avis :

Comment avez-vous entendu parler de nous ? ---

Recommanderiez-vous notre site à un ami ?

Oui Non Peut-être

Commentaire :

Recevoir la newsletter

Vous testerez votre formulaire en envoyant les données soumises vers l'URL <https://postman-echo.com/post> en méthode POST.

8. Balisage autre

Différentes versions de HTML
Identification et regroupement d'éléments
Commentaires et méta-information

Introduction

- À ce stade, nous avons présenté les **principales balises** qu'il est facile de classer en différents groupes et sections.
- Dans cette leçon, nous nous intéressons à des aspects importants qui n'entrent pas aisément dans des catégories :
 - Les différentes **versions de HTML** et comment indiquer celle utilisée ;
 - L'ajout de **commentaires** dans le code ;
 - **Attributs globaux** (attributs qui peuvent être employés avec n'importe quel élément), notamment **class** et **id**;
 - Éléments employés pour **regrouper** des parties de la page lorsque aucun autre élément n'est adapté ;
 - Ajout d'**informations relatives à la page web** à l'aide de l'élément **<meta>** ;
 - Ajout de **caractères spéciaux**, comme les crochets obliques ou le symbole de copyright.

Évolution de HTML (1)

- Depuis la naissance du Web, HTML a grandi et **plusieurs versions ont existé**.
- Chaque nouvelle version de HTML a pour objectif d'améliorer la précédente, en **apportant de nouveaux éléments et attributs**, et en **retirant** tout ce qui n'est plus adapté.
- Les **navigateurs** qui affichent les pages web existent également en plusieurs versions, chacune apportant de nouvelles possibilités. (Toutefois, tous les internautes ne sont pas équipés des dernières versions...).

Évolution de HTML (2)

- **HTML 4 (1997)** : HTML historique.
- **XHTML 1.0 (2000)** : Reformulation de HTML4 en suivant les règles XML (balise de fermeture obligatoire, noms des attributs en minuscule...)
- **HTML 5 (2000)** : Fermeture de toutes les balises pas obligatoire, nouveaux éléments et attributs ajoutés. ATTENTION : Spécifications HTML5 pas terminées.

Type de document

```
HTML5
<!DOCTYPE html>

HTML 4
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

XHTML 1.0 Transitional
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/
  xhtml1-transitional.dtd">

XHTML 1.0 Strict
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/
  xhtml1-strict.dtd">

Déclaration XML
<?xml version="1.0" ?>
```

- Puisque nous disposons de plusieurs versions de HTML, une page web doit débuter par une **déclaration DOCTYPE** qui précise au navigateur celle qui est employée.
- Cette déclaration doit être la **première ligne** du fichier HTML.

Commentaires HTML

```
<!-- Début de l'introduction. -->
<h1>Expositions en cours</h1>
<h2>Olafur Eliasson</h2>
<!-- Fin de l'introduction. -->
<!-- Début du texte principal. -->
<p>Olafur Eliasson est né à Copenhague, Danemark,
en 1967, de parents islandais.</p>
```

Expositions en cours

Olafur Eliasson

Olafur Eliasson est né à Copenhague, Danemark, en 1967, de parents islandais.

- `<!-- -->` — Pour ajouter un commentaire dans le code, sans qu'il soit visible par l'internaute, placez-le entre les caractères indiqués :
`<!-- Un commentaire. -->`
- Ajouter des commentaires dans le code est utile pour **se rappeler de l'organisation** d'une page, ou pour les **autres personnes** qui examineront votre code.
- Les commentaires ne sont **pas visibles dans la fenêtre du navigateur**, mais ils sont consultables dans le code source de la page.

Attribut `id`

```
<p>L'eau et l'air. Ces substances sont si communes  
qu'elles attirent difficilement l'attention,  
pourtant elles garantissent notre existence.</p>
```

```
<p id="pullquote">Chaque fois que je regarde la mer,  
je me sens rassuré, comme si je retournais dans  
ma demeure ancestrale ; j'embarque pour un voyage  
enchanteur.  

```

L'eau et l'air. Ces substances sont si communes qu'elles attirent difficilement l'attention, pourtant elles garantissent notre existence.

CHAQUE FOIS QUE JE REGARDE LA MER, JE ME SENS RASSURÉ, COMME SI JE RETOURNAIS DANS MA DEMEURE ANCESTRALE ; J'EMBARQUE POUR UN VOYAGE ENCHANTEUR.

- Tout élément HTML peut définir un attribut `id` afin d'**identifier de façon unique** cet élément dans la page.
- Sa valeur doit commencer par une **lettre** ou un **caractère de soulignement _** (non un chiffre ou un autre caractère). Il est important qu'une même valeur ne soit PAS donnée à l'attribut `id` de plusieurs éléments de la même page (la valeur DOIT être unique).
- L'`id` est très utilisé en CSS pour **cibler les éléments** à mettre en forme dans la page.

Attribut `class`

<code><p class="important">À partir de novembre 2010 et pendant un an, le musée d'art contemporain de Marugame Genichiro-Inokuma (MIMOCA) proposera quatre expositions de Hiroshi Sugimoto.</p>

<code><p>Chacune accueillera le travail de l'artiste sur les thèmes "Science", "Architecture", "Histoire" et "Religion", cela afin d'offrir un panorama de ses œuvres.</p>

<code><p class="important admittance">Horaires : 10h00 - 18h00 (fermeture des entrées à 17h30)</p>

À PARTIR DE NOVEMBRE 2010 ET PENDANT UN AN, LE MUSÉE D'ART CONTEMPORAIN DE MARUGAME GENICHIRO-INOKUMA (MIMOCA) PROPOSERA QUATRE EXPOSITIONS DE HIROSHI SUGIMOTO.

Chacune accueillera le travail de l'artiste sur les thèmes "Science", "Architecture", "Histoire" et "Religion", cela afin d'offrir un panorama de ses œuvres.

HORAIRES : 10H00 – 18H00 (FERMETURE DES ENTRÉES À 17H30)

- Chaque élément HTML peut également définir un attribut `class`.
- Parfois, au lieu de vouloir identifier de façon unique un élément dans un document, nous souhaitons **identifier un groupe d'éléments** pour indiquer qu'ils sont différents. Par exemple : une série de paragraphes qui contiennent des informations plus importantes que d'autres.
- Comme l'`id`, l'attribut `class` est très utilisé en CSS pour **cibler les éléments** à formater.

Niveau bloc & Niveau texte

- **Éléments de niveau bloc.** Éléments toujours affichés **sur une nouvelle ligne** dans la fenêtre du navigateur. Exemples : `<h1>`, `<p>`, `` et ``.
- **Éléments de niveau texte (ou en-ligne).** Éléments toujours affichés **sur la même ligne** que les éléments qui les entourent. Exemples : `<a>`, ``, `` et ``.

- **Groupe d'éléments de niveau bloc.**

L'élément `<div>` permet de regrouper un jeu d'éléments dans une boîte de niveau bloc. (Dans un navigateur, le contenu de l'élément `<div>` est affiché sur une nouvelle ligne, mais c'est le seul impact visuel.)

```
<div id="header">
  
  <ul>
    <li><a href="index.html">Accueil</a></li>
    <li><a href="biographie.html">Biographie</a></li>
    <li><a href="oeuvres.html">Oeuvres</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</div><!-- Fin de header -->
```

- **Groupe d'éléments de niveau texte.** L'élément `` est l'équivalent en-ligne de l'élément `<div>`.

```
<p>Anish Kapoor a reçu le prix Turner en 1991 et
a exposé au musée <span class="gallery">Tate
Modern</span> de Londres en 2003.</p>
```

Informations relatives aux pages

```
<meta name="description"
      content="Essai sur l'installation (genre d'art
              contemporain)" />
<meta name="keywords"
      content="installation, art, opinion" />
<meta name="robots"
      content="nofollow" />
<meta http-equiv="author"
      content="Jon Duckett" />
<meta http-equiv="pragma"
      content="no-cache" />
<meta http-equiv="expires"
      content="Fri, 04 Apr 2014 23:59:59 GMT" />
```

- L'élément `<meta>` réside dans l'élément `<head>` et comprend des **informations relatives à la page web**.
- Il n'est **pas visible** des internautes mais remplit de nombreux rôles, comme fournir aux moteurs de recherche des informations sur la page et indiquer si elle évolue avec le temps (elle peut expirer).
- L'élément `<meta>` est un élément vide et n'a donc **pas besoin d'une balise de fermeture**. Ses attributs `name` et `content` fournissent les informations.

Caractères spéciaux

<	Inférieur à ‐< ‐<	¢	Centime ‐¢ ‐¢
>	Supérieur à ‐> ‐>	£	Livre ‐£ ‐£
&	Esperluette ‐& ‐&	¥	Yen ‐¥ ‐¥
”	Guillemet ‐" ‐"	€	Euro ‐€ ‐€

- Certains caractères sont utilisés dans le code HTML et sont donc **réservés**. C'est notamment le cas des crochets obliques gauche et droit.
- Si vous souhaitez que des caractères spéciaux soient affichés sur une page, vous devez employer les "**caractères d'échappement**", également appelés références d'entités ou séquences d'échappement.
- Par exemple, pour insérer un **crochet oblique** gauche, vous devez écrire **<** ou **<**. Pour le symbole **copyright** © : **©** ou **©**.
- Liste des entités HTML : https://alexandre.alapetite.fr/doc-alex/alx_special.html

En résumé

- Les déclarations **DOCTYPE** précisent au navigateur la **version de HTML utilisée** par la page.
- Pour ajouter des **commentaires** dans le code, placez le texte entre les marqueurs `<!--` et `-->`.
- Les attributs **id** et **class** permettent d'identifier des éléments précis.
- Les éléments `<div>` et `` permettent de **regrouper**, respectivement, des **éléments de niveau bloc** et des **éléments de niveau texte**.
- La balise `<meta>` permet de fournir toutes sortes d'informations relatives à la page web.
- Les **caractères d'échappement** sont utilisés pour inclure des caractères spéciaux dans les pages, comme les crochets obliques (`<` et `>`) et le symbole de copyright (`©`).

9. Flash, vidéo et audio

Ajout d'animations Flash à un site
Ajout de contenus vidéo et audio à un site
Éléments <video> et <audio> de HTML5

Flash

- Depuis la fin des années 1990, Flash a été très utilisé pour la création d'**animations**, puis, plus tard, pour la lecture de contenu **audio** et **vidéo** sur les sites web.
- Les contenus Flash publiés sur le web sont appelés des “**animations Flash**”, générées avec le logiciel **Adobe Flash**.
- Pour lire une animation Flash, les navigateurs ont besoin d'un plug-in (une extension logicielle qui s'exécute dans le navigateur) appelé **Flash Player**.
- Depuis 2005, plusieurs facteurs ont conduit au **déclin de Flash** : popularisation des effets animés créés en JavaScript, Flash non supporté par l'iPhone (2007) et l'iPad (2010), apparition des balises HTML5 **<video>** et **<audio>**, accessibilité.

Vidéo

- **FORMATS ET LECTEURS :**

- **FORMAT** : AVI, Flash, H264, MPEG... Il vous faudra **encoder** vos vidéos dans un format largement compatible. Si vous prévoyez d'utiliser la balise HTML5 `<video>` : H264 pour IE et Safari, WebM pour Android, Chrome, Firefox, et Opera.
- **LECTEUR** : Flash Player, balise HTML5 `<video>`, JW Player (www.jwplayer.com) qui combine Flash et HTML5.

- **HÉBERGEMENT** - Utiliser un service d'hébergement vidéo (YouTube, Vimeo...)

- PLUS : large compatibilité, gère l'encodage, parfois gratuit.
- MOINS : vidéos parfois publiques, ajout de publicités au vidéo, impossibilité de monétiser les vidéos
- ALTERNATIVE : héberger la vidéo vous-même, mais attention aux performances et au coût.

Ajout d'une vidéo avec Flash

```
<!DOCTYPE html>
<html>
<head>
  <title>Ajout d'une vidéo Flash</title>
  <script type="text/javascript"
    src="http://ajax.googleapis.com/ajax/libs/
    swfobject/2.2/swfobject.js"></script>
  <script type="text/javascript">
    var flashvars = {};
    var params = {movie:"../video/puppy.flv"};
    swfobject.embedSWF("flash/OSplayer.swf",
      "snow", "400", "320", "8.0.0",
      flashvars, params);</script>
</head>
<body>
  <div id="snow"><p>Vidéo d'un petit chien qui
    joue dans la neige.</p></div>
</body>
</html>
```



- Cet exemple se fonde sur le **lecteur OS FLV** pour jouer une vidéo nommée **puppy.flv** (déjà convertie au format FLV).
- Les vidéos Flash ont parfois besoin d'**informations** pour être lues (chemin du fichier vidéo, params divers...). Ici, SWFObject utilise des variables JavaScript pour les lui fournir.

Ajout d'une vidéo avec HTML5

```
<video src="video/puppy.mp4"
       poster="images/puppy.jpg"
       width="400" height="300"
       preload
       controls
       loop>
<p>Vidéo d'un petit chien qui joue dans
la neige.</p>
</video>
```

- **<video>** — L'élément **<video>** reconnaît plusieurs attributs pour le contrôle de la lecture de la vidéo :
- **src** — Cet attribut précise le **chemin de la vidéo**. Le fichier d'exemple est au format H264 et n'est donc compatible qu'avec IE et Safari.
- **poster** — Cet attribut permet d'indiquer **l'image qui sera affichée pendant le téléchargement** de la vidéo ou jusqu'au démarrage de la lecture.
- **width, height** — Ces attributs définissent la **taille** du lecteur, en pixels.

Audio

- Format le plus utilisé : **MP3**.
- Pour ajouter un contenu audio dans une page web :
 1. Service hébergé (ex : SoundCloud.com)
 2. Flash (player simple, jukebox...)
 3. Balise HTML5 **<audio>**
Format **Ogg Vorbis** pour Firefox, Chrome et Opera.
Format **MP3** pour Safari5 et IE 9.

```
<audio src="audio/test-audio.ogg"
       controls autoplay>
<p>Votre navigateur ne prend pas en charge
       notre format audio.</p>
</audio>
```

En résumé

- **Flash** permet de proposer des animations ainsi que du contenu audio et vidéo sur le Web.
- Flash n'est **pas pris en charge sur l'iPhone et l'iPad**.
- **HTML5** apporte les éléments `<video>` et `<audio>` pour ajouter du contenu vidéo et audio aux pages web, mais ils ne sont reconnus que par les navigateurs récents.
- Les navigateurs qui prennent en charge les éléments HTML5 ne sont pas tous capables de lire les mêmes formats vidéo et audio. Vous devez donc fournir vos fichiers **dans différents formats** pour que tous les internautes puissent les visionner ou les écouter.

HTML5

Définition, Support, Syntaxe

Qu'est-ce que HTML5 ?

- Dernière itération en date du langage HTML.
- **Objectif du HTML5** : Entre sa naissance et aujourd'hui, le web a beaucoup évolué. Initialement, HTML servait à afficher des documents. Aujourd'hui, il sert à créer des applications bien qu'il n'ait pas été conçu pour ça. HTML5 vise à **combler les lacunes du HTML**, en le rendant **plus adapté à l'usage qu'on en fait aujourd'hui**.
- Que contient HTML5 ?
 - **Balises** faisant partie du standard HTML5, comme `<audio>` ou `<video>`.
 - Fonctionnalités **JavaScript** : stockage local de données, mode hors-ligne...
 - **Fonctionnalités** apparues après le standard : CSS3, géolocalisation...

Philosophie de HTML5

1. **Ne pas casser le web.** Les pages web qui n'ont pas été créées selon les spécifications HTML5 doivent continuer à s'afficher/fonctionner normalement.
Exemple : anciennes balises comme `` encore supportées.
2. **Chemin le plus emprunté.** HTML5 standardise des techniques non-officielles mais utilisées par tous.
Exemple : `drag&drop` initialement créé par Microsoft pour IE5.
3. **Approche pratique.** Dans l'idéal, un langage plus strict et sans bidouille est préférable. Mais il faut tenir compte des milliards de page web qui existent déjà.
Exemple : YouTube utilise `Flash` pour lire les vidéos. Flash n'est pas un standard, c'est un plugin, mais il est installé sur quasiment tous les navigateurs.

Squelette de page HTML5

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Un mini-doc HTML</title>
  <link href="styles.css"
        rel="stylesheet">
  <script src="scripts.js">
  </script>
</head>
<body>
  <p>Tu déchires, HTML5.</p>
</body>
</html>
```

- **Doctype** HTML5 (obligatoire)
- **Encodage** de caractères (facultatif; permet d'éviter les erreurs d'affichage)
- **Langue** du texte de la page (facultatif)
- **Feuilles de styles** : disparition de l'attribut **type="text/css"**.
- **JavaScript** : pas d'attribut **language="JavaScript"**.

Règles HTML5 moins strictes

- En HTML5, les éléments `<html>`, `<body>` et `<head>` sont **facultatifs**. MAIS : il est conseillé de les conserver.
- En HTML5, les balises peuvent être écrites en **majuscules ou minuscules**. MAIS : minuscules conseillées.
`<P>Ceci est un paragraphe.</P>`
- En HTML5, on peut **omettre le slash de fermeture** pour les éléments vides : `
` au lieu de `
`.
- En HTML5, les **attributs** ne doivent pas être écrits **entre guillemets** tant qu'ils ne contiennent pas de caractères spéciaux : ``
MAIS : c'est mieux de garder les guillemets en préventif.
- Pour **valider** que votre page est conforme au standard HTML5: <http://validator.w3.org/>

Éléments HTML5 (1)

CATEGORY	ELEMENTS
Semantic elements for structuring a page	<article>, <aside>, <figcaption>, <figure>, <footer>, <header>, <nav>, <section>, <details>, <summary>
Semantic elements for text	<mark>, <time>, <wbr> (previously supported, but now an official part of the language)
Web forms and interactivity	<input> (not new, but has many new subtypes) <datalist>, <keygen>, <meter>, <progress>, <command>, <menu>, <output>
Audio, video, and plug-ins	<audio>, <video>, <source>, <embed> (previously supported, but now an official part of the language)
Canvas	<canvas>
Non-English language support	<bdo>, <rp>, <rt>, <ruby>

- De nombreux éléments ont été ajoutés aux éléments existants.
- Liste officielle :
<http://w3c.github.io/html-reference/>

Éléments HTML5 (2)

- **Éléments retirés.** Surtout des éléments présentationnels comme ``, `<big>`, `<center>`...
- **Éléments adaptés.** Ils ont une nouvelle signification dans la norme HTML5.
Exemple : `<hr>` (ligne horizontale) représente maintenant un changement de sujet dans un document.
- **Éléments standardisés.** Étaient supportés, mais sans faire partie du standard.
Exemple : `<embed>` pour insérer un plugin dans une page.

Utiliser HTML5 aujourd’hui

- **Tout dépend du navigateur utilisé par VOS utilisateurs !**
- Comment savoir **si une fonctionnalité précise de HTML5 est supportée** par tel ou tel navigateur ? <https://caniuse.com/>
- Comment connaître les **navigateurs les plus utilisés** sur le web ? <https://gs.statcounter.com/>
- **Détection de fonctionnalité** avec **Modernizr** (1) (modernizr.com). Permet d'avertir l'utilisateur, d'afficher une version dégradée ou d'implémenter une solution alternative.
- **Recréer les fonctionnalités manquantes** avec les **Polyfills** (2) : tinyurl.com/polyfill

Structurer vos pages avec les éléments sémantiques

- En HTML5, plusieurs nouveaux éléments n'apportent vraiment pas de nouvelles fonctionnalités. Leur raison d'être est de rendre vos pages web **plus sémantiques**, de leur donner **plus de sens**, et notamment :
- **Maintenance des pages facilitée.** Car leur structure est plus facile à comprendre.
- **Accessibilité.** Par ex, un lecteur d'écran peut “comprendre” la structure d'une page.
- **Optimisation moteurs de recherche.** Google peut comprendre la structure d'une page, son auteur, sa langue, la date à laquelle elle a été écrite.

Structurer vos pages avec les éléments sémantiques

- En **HTML classique**, les “blocs” d’une page sont typiquement matérialisés par une **série d’éléments** `<div>` avec des attributs `id` ou `class`. **Pas en HTML5 !**
- **Structure générale** de page : `<header>` (en-tête), `<main>` (contenu principal), `<footer>` (pied de page).
- Bloc de **liens de navigation** : `<nav>`
- Bloc **Article** : `<article>` (pouvant lui-même contenir `<header>`, `<footer>`, etc.)
- Bloc **générique** (si aucune balise plus précise ne convient) : `<section>`
- Bloc **encadré / mise en exergue** (liée au texte) : `<aside>`
- Bloc **image d’illustration** (liée au texte) : `<figure>` et `<figcaption>`

EXERCICE: Convertir une page en HTML5

- Vous partirez de la page `html5_structure/index_original.html`.
- Assurez-vous que le DOCTYPE et les éléments du `<head>` sont conformes à la norme HTML5.
- Référez la nouvelle feuille de styles `styles_revision.css`.
- Ajoutez tous les **éléments sémantiques** présentés sur le slide précédent.
- Enfin, affichez :
 - L'**image** `human_skull.jpg`, que vous alignerez à gauche du texte “But don't get too smug.” en lui appliquant la classe CSS `FloatFigure`.
 - Une **mise en exergue** de la phrase “We don't know how the universe started...” que vous alignerez à droite grâce à la class CSS `PullQuote`.

Formulaires

Nouveautés HTML5

- **Validation.** Les éléments de formulaire supportent de nouveaux attributs de validation :
 - **required** : Signale qu'un champ est obligatoire.
 - **pattern** : Validation par expression régulière.
Exemple : `<input type="text" pattern="[A-Z]{3}-[0-9]{3}>` doit contenir un code de 3 lettres majuscules, un tiret, et 3 chiffres.
 - Pour désactiver la validation sur un formulaire : `<form novalidate>`
- Des **pseudo-classes** sont ajoutées aux champs pour changer leur apparence en fonction de leur état : **required/optional, valid/invalid...**
Dans la CSS : `input:required { background-color: lightyellow; }`
- De **nouveaux champs** `<input type="">` sont supportés, offrant un widget de saisie adapté et/ou une validation adéquate : **email, url, tel, number, date...**
Exemple : `<input type="email" name="your_email">`
- Les **placeholders** permettent de clarifier les valeurs à saisir pour l'utilisateur :
`<input id="telephone" placeholder="(xxx) xxx-xxxx">`

EXERCICE: Convertir un formulaire en HTML5

- Vous partirez de la page `html5_formulaire/index_original.html`.
- Coloriez en jaune les champs qui sont obligatoires ET invalides.
- Ajoutez l'attribut `required` à tous les champs marqués comme obligatoires.
- Utilisez les nouveaux types de champ HTML5 quand c'est possible.

Partie 2 - CSS

10. Introduction à CSS

Objectif de CSS
Fonctionnement de CSS
Règles, propriétés et valeurs

Introduction

- Dans cette partie de la formation, nous allons voir comment rendre les pages web plus attrayantes, en **contrôlant leur présentation avec CSS**.
- CSS permet de créer des **règles** qui régissent l'affichage du contenu d'un élément. Par exemple, nous pouvons indiquer que l'arrière-plan de la page est de couleur crème, que le texte de tous les paragraphes est en gris et en Arial ou que les titres de niveau un sont en bleu, en italique et en Times.
- Dès lors que vous savez comment écrire une règle CSS, il ne vous reste plus qu'à connaître les différentes **propriétés** disponibles.
- Dans cette leçon : introduction au fonctionnement de CSS ; apprentissage de l'écriture des règles CSS ; application des règles CSS aux pages HTML.

CSS : Réfléchir en termes de boîtes

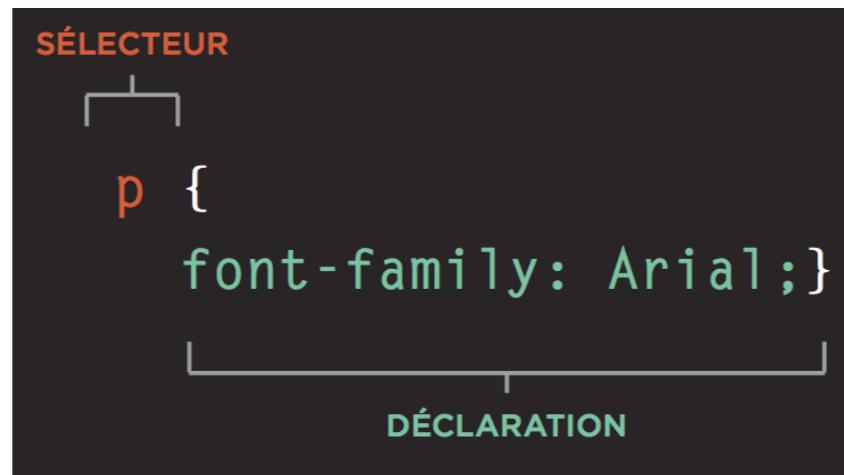
- Pour comprendre le fonctionnement de CSS, il faut imaginer qu'une **boîte invisible** entoure chaque élément HTML :



- CSS permet de **créer des règles qui contrôlent la présentation de chaque boîte** individuelle et de son contenu.

Association des règles de style aux éléments HTML

- CSS se fonde sur l'**association de règles aux éléments HTML**. Ces règles régissent l'affichage du contenu des éléments indiqués et comprennent deux parties : un **sélecteur** et une **déclaration**.



- Le **sélecteur** cible l'**élément auquel la règle s'applique**. La même règle peut concerner plusieurs éléments si les noms de ceux-ci sont **séparés par des virgules**.
- Les **déclarations** précisent le **style des éléments** désignés par le sélecteur. Chaque déclaration est constituée d'une **propriété** et de sa **valeur**. Les différentes déclarations sont **séparées par des points-virgules**.

Déclaration CSS (Liste de propriétés/valeurs)

- Les **déclarations CSS** sont placées entre **accolades** et chacune est constituée d'une **propriété** et d'une **valeur**, séparées par des **deux-points**. Les différentes déclarations sont séparées par des **points-virgules**.

```
h1, h2, h3 {  
    font-family: Arial;  
    color: yellow;}
```



The diagram shows a dark rectangular box containing a snippet of CSS code. The code defines styles for h1, h2, and h3 elements, specifically setting the font family to Arial and the color to yellow. Below the box, two horizontal arrows point from the text 'PROPRIÉTÉ' and 'VALEUR' to the colon ':' and the space before the value respectively. The word 'PROPRIÉTÉ' is positioned under the first colon, and 'VALEUR' is positioned under the second colon.

- Les **propriétés** indiquent les **aspects** des éléments que vous souhaitez modifier. Il s'agit par exemple de la couleur, de la police, de la largeur, de la hauteur et de la bordure.
- Les **valeurs** correspondent aux **paramètres** à appliquer aux propriétés indiquées. Par exemple, pour donner une couleur précise au texte des éléments sélectionnés, la valeur de la propriété doit être cette couleur.

Référence CSS

- Maîtriser CSS consiste donc essentiellement à connaître toutes les **propriétés et valeurs existantes**, et savoir quand les utiliser.
- Pour cela :

A CONNAÎTRE : Référence CSS MDN (liste de toutes les propriétés et valeurs CSS) - <https://developer.mozilla.org/fr/docs/Web/CSS/Reference>

Styles CSS externes

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Styles CSS externes</title>
    <link href="css/styles.css" type="text/css"
          rel="stylesheet" />
  </head>
  <body>
    <h1>Pommes de terre</h1>
    <p>Il existe des dizaines de variétés de pommes de terre. On distingue les variétés de primeur et de conservation.</p>
  </body>
</html>
```

HTML

styles.css

Chapitre 10/Styles.css

CSS

```
body {
  font-family: arial;
  background-color: rgb(185,179,175);}
h1 {
  color: rgb(255,255,255);}
```

<link> — L'élément **<link>** est utilisé dans un document HTML pour indiquer au navigateur l'**emplACEMENT DU FICHIER CSS** qui définit les styles de la page. Cet élément vide (sans balise de fermeture) est placé dans l'élément **<head>** et définit **TROIS ATTRIBUTS** :

- **href** — Chemin du fichier CSS (souvent enregistré dans un dossier nommé *css* ou *styles*).
- **type** — Type du document lié. Sa valeur doit être **text/css**.
- **rel** — Relation entre la page HTML et le fichier lié. Sa valeur doit être **stylesheet** dans le cas d'un lien vers un fichier CSS.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Styles CSS internes</title>
    <style type="text/css">
      body {
        font-family: arial;
        background-color: rgb(185,179,175);
      }
      h1 {
        color: rgb(255,255,255);
      }
    </style>
  </head>
  <body>
    <h1>Pommes de terre</h1>
    <p>Il existe des dizaines de variétés de pommes de terre. On distingue les variétés de primeur et de conservation.</p>
  </body>
</html>
```

RÉSULTAT

Pommes de terre

Il existe des dizaines de variétés de pommes de terre. On distingue les variétés de primeur et de conservation.

Styles CSS internes

- **<style>** — Les règles CSS peuvent également être déclarées **dans la page HTML**, en les insérant dans un élément **<style>**, généralement placé dans l'élément **<head>**.
- L'élément **<style>** doit utiliser l'attribut **type** pour indiquer que les styles sont définis en CSS. Sa valeur doit être **text/css**.
- NB. La feuille de style CSS **externe** est **PRÉFÉRABLE** :
 - Toutes les pages référencent les **mêmes règles de style** sans avoir à les répéter dans chaque page.
 - Le contenu de la page reste **séparé** de sa présentation.
 - Les styles employés par les pages peuvent être modifiés à partir d'**un seul fichier** (au lieu de retoucher chaque page individuelle).

Sélecteurs CSS (1)

SÉLECTEUR	DESCRIPTION	EXEMPLE
SÉLECTEUR UNIVERSEL	S'applique à tous les éléments du document	* {} Cible tous les éléments de la page
SÉLECTEUR DE TYPE	Correspond aux noms des éléments	h1, h2, h3 {} Cible les éléments <h1>, <h2> et <h3>
SÉLECTEUR DE CLASSE	Correspond aux éléments dont la valeur de l'attribut class est celle indiquée après le point	.note {} Cible tout élément dont l'attribut class a la valeur note p.note {} Cible uniquement les éléments <p> dont l'attribut class a la valeur note
SÉLECTEUR D'IDENTIFIANT	Correspond à l'élément dont la valeur de l'attribut id est celle indiquée après le symbole dièse	#introduction {} Cible l'élément dont l'attribut id a la valeur introduction

Sélecteurs CSS (2)

SÉLECTEUR D'ENFANT

Correspond aux éléments qui sont les enfants directs d'un autre

`li>a {}`

Cible tous les éléments `<a>` qui sont des enfants d'un élément `` (mais sans les autres éléments `<a>` de la page)

SÉLECTEUR DE DESCENDANT

Correspond aux éléments qui sont des descendants de celui indiqué (pas uniquement ses enfants directs)

`p a {}`

Cible tous les éléments `<a>` qui se trouvent dans un élément `<p>`, même ceux à plusieurs niveaux d'imbrication

SÉLECTEUR DE FRÈRE ADJACENT

Correspond à l'élément enfant qui suit immédiatement un autre enfant de même parent

`h1+p {}`

Cible le premier élément `<p>` qui vient après un élément `<h1>` (sans les éléments `<p>` suivants)

SÉLECTEUR DE FRÈRE GÉNÉRAL

Correspond aux éléments qui sont les frères d'un autre, même s'ils ne le suivent pas directement

`h1~p {}`

Si deux éléments `<p>` sont les frères d'un élément `<h1>`, la règle s'applique aux deux

```
<h1>Pommes de terre</h1>
<p id="intro">Il existe des <i>dizaines</i> de
variétés de <b>pommes de terre</b>.</p>
<p>On distingue les variétés de primeur et de
conservation.</p>
```

```
* {
  font-family: Arial, Verdana, sans-serif;}
h1 {
  font-family: "Courier New", monospace;}
i {
  color: green;}
i {
  color: red;}
b {
  color: pink;}
p b {
  color: blue !important;}
p b {
  color: violet;}
p#intro {
  font-size: 100%;}
p {
  font-size: 75%;}
```

Pommes de terre

Il existe des *dizaines* de variétés de **pommes de terre**.

On distingue les variétés de primeur et de conservation.

Cascade des règles CSS

- Si deux règles ou plus s'appliquent aux mêmes éléments, il est important de comprendre **comment est donnée la priorité** :
- **DERNIÈRE RÈGLE.** Si deux sélecteurs sont identiques, le dernier est prioritaire. Dans l'exemple, le second sélecteur **i** prend le pas sur le premier.
- **SPÉCIFICITÉ.** Si un sélecteur est plus spécifique que les autres, les règles les plus précises sont prioritaires sur les plus générales. Dans notre exemple : **h1** est plus précis que *****, **p b** est plus précis que **p**, et **p#intro** est plus précis que **p**.
- **IMPORTANT.** En ajoutant **!important** après la valeur d'une propriété, vous indiquez qu'elle doit être considérée plus importante que les autres règles qui s'appliquent au même élément.

HTML

```
<div class="page">
  <h1>Pommes de terre</h1>
  <p>Il existe des dizaines de variétés de pommes de terre.</p>
  <p>On distingue les variétés de primeur et de conservation.</p>
</div>
```

CSS

```
body {
  font-family: Arial, Verdana, sans-serif;
  color: #665544;
  padding: 10px;}
.page {
  border: 1px solid #665544;
  background-color: #efefef;
  padding: inherit;}
```

RÉSULTAT

Pommes de terre

Il existe des dizaines de variétés de pommes de terre.

On distingue les variétés de primeur et de conservation.

Héritage CSS

- Si vous définissez les propriétés **font-family** ou **color** sur l'élément **<body>**, elles s'appliquent à la plupart des éléments enfants. En effet, la valeur de la propriété **font-family** est **héritée** par les éléments enfants. Cela vous évite d'appliquer ces propriétés à un grand nombre d'éléments et conduit à des feuilles de style **plus simples**.
- Ce principe ne concerne pas certaines autres propriétés, comme **background-color** ou **border**, qui ne sont **PAS héritées** par les éléments enfants.
- Vous pouvez **obliger** un grand nombre de propriétés à hériter de la valeur donnée par leurs parents en leur affectant la valeur **inherit**.

Version de CSS et navigateurs

- Comme il a existé plusieurs versions de HTML, CSS a fait l'objet de **différentes évolutions** :
 - **CSS1** a été publiée en 1996, suivie deux ans plus tard de **CSS2**. Les travaux sur **CSS3** ne sont pas encore terminés, mais les principaux navigateurs ont déjà commencé à la mettre en œuvre.
 - **Inconsistances.** Les navigateurs ne mettent pas en œuvre toutes les caractéristiques de CSS et certains ne gèrent pas certaines propriétés.
 - Avant de mettre un nouveau site en production, **testez-le avec plusieurs navigateurs**, car ils peuvent afficher les pages avec de légères différences. Services de testing : browsershots.org, browserstack.com

En résumé

- CSS traite chaque élément HTML comme s'il apparaissait **à l'intérieur de sa propre boîte** et se fonde sur des **règles** pour déterminer l'apparence de cet élément.
- Les règles sont constituées de **sélecteurs**, qui précisent les éléments auxquels elles s'appliquent, et de **déclarations**, qui définissent l'apparence de ces éléments.
- Grâce aux **types de sélecteurs** disponibles, vous pouvez appliquer les règles aux différents éléments.
- Les déclarations sont constituées de deux parties : les **propriétés** de l'élément que vous souhaitez modifier et les **valeurs** de ces propriétés. Par exemple, la propriété **font-family** permet de changer de police de caractères et la valeur **arial** indique d'utiliser la police Arial.
- Les règles CSS sont généralement données dans un **document séparé**, mais elles peuvent également être insérées directement dans la page HTML.

EXERCICE : Sélecteurs CSS

- Dans `css_intro/index.html`, référez une **feuille de styles externe** `styles.css` (**À CRÉER**) dans laquelle vous appliquerez les styles suivants :
 - A toute la page :
`font-family: Arial, Verdana, sans-serif;`
 - Aux titres de niveau 1 et 2 : `color: #EE3E80;`
 - A tous les paragraphes : `color: #665544;`
- La page stylée devra ressembler à ça :
- **Surchargez** la couleur des titres H2 directement dans la page HTML.

From Garden to Plate

A *potager* is a French term for an ornamental vegetable or fruit garden. It is used to enhance the garden's beauty. The goal is to make the function

What to Plant

Plants are chosen as much for their functionality as for their beauty. They are chosen for the home and herbs for the garden with very little maintenance. Potager plants range from the cottage garden to the formality of a knot garden.

11. Couleurs

Syntaxe des couleurs

Terminologie de la couleur et contraste

Couleur d'arrière-plan

Dans cette leçon

- **Syntaxe d'écriture des couleurs**, car il existe trois manières de préciser la couleur souhaitée (sans mentionner celle apportée par CSS3) ;
- **Terminologie** associée au monde de la couleur, car certains termes permettront de mieux comprendre comment choisir des couleurs ;
- **Contraste**, car le texte doit être parfaitement lisible ;
- **Couleur d'arrière-plan** de l'intégralité de la page ou de certaines parties.
- NB. Toutes les connaissances couleur de cette leçon seront applicables à la couleur du texte et des boîtes (qu'on verra après).

Couleur d'avant-plan

```
/* Nom de couleur. */  
h1 {  
    color: DarkCyan;}  
/* Code hexadécimal. */  
h2 {  
    color: #ee3e80;}  
/* Valeurs RVB. */  
p {  
    color: rgb(100,100,90);}
```

color

Biologie marine

Composition de l'eau de mer

L'eau de mer est composée d'eau et de sels, ainsi que de diverses substances en faible quantité. Si plus des deux tiers des 94 éléments chimiques naturels sont présents dans l'eau de mer, la plupart le sont en faible quantité et difficilement décelables.

- La propriété **color** permet de préciser la couleur du texte d'un élément.
- En CSS, il existe **trois façons d'écrire une couleur** :
 - **VALEURS RVB** — Ces valeurs expriment la couleur sous forme de ses composantes rouge, vert et bleu. Par exemple : **rgb(100,100,90)**.
 - **CODE HEXADÉCIMAL** — Ce code à six chiffres, précédés d'un signe dièse (#), représente la quantité de rouge, de vert et de bleu dans la couleur. Par exemple : **#ee3e80**.
 - **NOM DE COULEUR** — Il existe 147 noms de couleurs prédéfinis et reconnus par les navigateurs. Par exemple : **Dark- Cyan**.

Couleur d'arrière-plan **background-color**

```
body {  
    background-color: rgb(200,200,200);}  
h1 {  
    background-color: DarkCyan;}  
h2 {  
    background-color: #ee3e80;}  
p {  
    background-color: white;}
```

Biologie marine

Composition de l'eau de mer

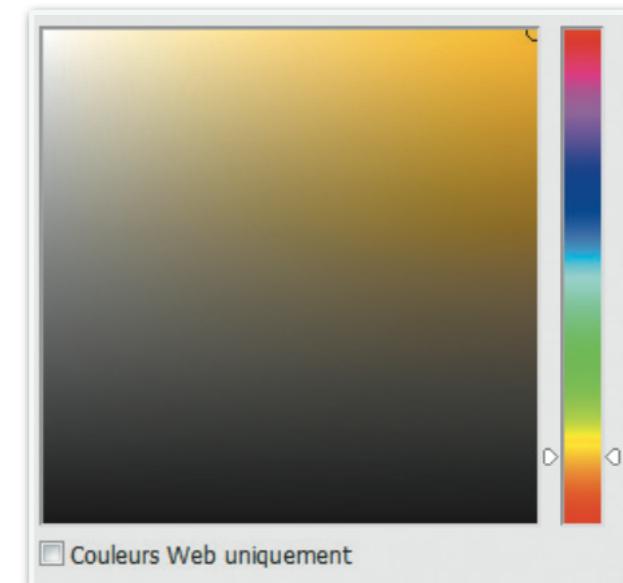
L'eau de mer est composée d'eau et de sels, ainsi que de diverses substances en faible quantité. Si plus des deux tiers des 94 éléments chimiques naturels sont présents dans l'eau de mer, la plupart le sont en faible quantité et difficilement décelables.

- CSS traite chaque élément HTML comme s'il apparaissait **à l'intérieur d'une boîte**. La couleur d'**arrière-plan** de cette boîte peut être modifiée à l'aide de la propriété **background-color**.
- L'écriture de la couleur d'arrière-plan se fait de la même manière que pour la couleur d'avant-plan : **valeurs RVB, code hexadécimal ou nom de couleur**.
- Si aucune couleur d'arrière-plan n'est précisée, le fond de la boîte est **transparent**.

3 façons d'écrire une couleur

- En CSS, il y a 3 façons d'écrire une couleur :

VALEURS RVB	CODE HEXADÉCIMAL	NOM DE COULEUR
Les valeurs des composantes rouge, vert et bleu s'expriment sous forme de nombres entre 0 et 255.  rgb(102,205,170)	Des valeurs données en hexadécimal représentent les quantités de rouge, de vert et de bleu.  #66cdAA	Les couleurs sont représentées par des noms prédéfinis, mais leur nombre est très limité.  MediumAquaMarine



- Sur un écran, une couleur s'obtient en mélangeant une quantité de **rouge**, de **vert** et de **bleu**. Un **sélecteur de couleur** aide à faire votre choix.

- Les **logiciels de création d'images**, comme Photoshop et GIMP, proposent des outils de sélection d'une couleur. Ils indiquent souvent les valeurs RVB et la valeur hexadécimale. Il existe également des outils en ligne, comme paletton.com.

Contraste

- Lors du choix des couleurs d'avant-plan et d'arrière-plan, il est important de vérifier que le **contraste** est suffisant pour que le texte reste lisible:

CONTRASTE
FAIBLE

CONTRASTE
ÉLEVÉ

CONTRASTE
MOYEN

Le texte est plus difficile à lire lorsque le contraste entre les couleurs d'avant et d'arrière-plan est faible.

Le texte est plus facile à lire lorsque le contraste entre les couleurs d'avant et d'arrière-plan est élevé.

Pour les longs paragraphes de texte, une diminution du contraste améliore légèrement la lisibilité.

CSS3 : TRANSPARENCE

CSS

```
p.one {  
background-color: rgb(0,0,0);  
opacity: 0.5;}  
  
p.two {  
background-color: rgb(0,0,0);  
background-color: rgba(0,0,0,0.5);}
```

RÉSULTAT



RÉSULTAT DANS LES ANCIENS NAVIGATEURS



opacity, rgba

- CSS3 amène une nouvelle propriété, **opacity**, qui permet de préciser la **transparence d'un élément et de ses enfants**. Sa valeur est un nombre entre 0.0 (totalement transparent) et 1.0 (totalement opaque).
- La propriété CSS3 **rgba** permet d'affecter une couleur, à la manière d'une valeur RVB, mais avec une **4ème composante pour la transparence**. Cette valeur **alpha** est un nombre entre 0.0 et 1.0 (0.5 correspond à une transparence de 50 %, 0.15, à une opacité de 15%). La valeur de **rgba** affecte uniquement l'élément auquel elle est appliquée.
- Puisque certains navigateurs ne reconnaissent pas les couleurs RVBA, il est bon de proposer une **couleur de repli unie**.

CSS3 : COULEURS TSL

- CSS3 propose une nouvelle manière intuitive de définir des couleurs : à partir de valeurs de **teinte**, de **saturation** et de **luminosité** :

TEINTE

Dans le monde de la couleur, la notion de teinte est familière. Elle est souvent représentée par un cercle coloré dans lequel l'angle représente la couleur, mais on utilise parfois un curseur avec des valeurs allant de 0 à 360.

SATURATION

La saturation représente la quantité de gris dans une couleur. Elle est indiquée sous forme de pourcentage. 100 % donne une saturation totale, tandis que 0% correspond à une nuance de gris.

LUMINOSITÉ

La luminosité correspond à la quantité de blanc (clarté) ou de noir (obscurité) dans une couleur. Elle est indiquée sous forme de pourcentage. Une luminosité de 0% équivaut à du noir, de 100 %, à du blanc, et de 50 %, à une couleur normale.

CSS3 : TSL ET TSLA

css

```
body {  
    background-color: #C8C8C8;  
    background-color: hsl(0,0%,78%);}  
  
p {  
    background-color: #ffffff;  
    background-color: hsla(0,100%,100%,0.5);}
```

RÉSULTAT

Biologie marine

Composition de l'eau de mer

L'eau de mer est composée d'eau et de sels, ainsi que de diverses substances en faible quantité. Si plus des deux tiers des 94 éléments chimiques naturels sont présents dans l'eau de mer, la plupart le sont en faible quantité et difficilement décelables.

hsl, hsla

- La fonction **hsl** a été ajoutée à CSS3 de manière à proposer une autre expression des couleurs. La valeur commence par les lettres **hsl**, suivies, entre parenthèses, de la valeur de chaque composante :
 - **TEINTE** La teinte (hue) s'exprime sous forme d'un angle entre 0 et 360 degrés.
 - **SATURATION** Elle est donnée sous forme de pourcentage.
 - **LUMINOSITÉ** La luminosité est indiquée sous forme de pourcentage, 0 % correspondant au blanc, 50 %, à la couleur normale, et 100 %, au noir.
- La fonction **hsla** permet de définir une couleur comme précédemment, à partir d'une teinte, d'une saturation et d'une luminosité, mais ajoute une **quatrième valeur pour la transparence** (à l'instar de `rgba`).
- **ALPHA** Cette valeur est un nombre entre 0 et 1.0. Par exemple, 0.5 correspond à une transparence de 50 %, et 0.75, à une opacité de 75 %.

En résumé

- Non seulement la couleur donne vie à votre site, mais elle aide également à transmettre une **humeur** et à susciter des **réactions**.
- En CSS, il existe **3 manières de préciser une couleur** : valeurs RVB, code hexadécimal et nom de couleur.
- Les **sélecteurs de couleur** peuvent vous aider à trouver celle qui convient.
- Il est important de vérifier que le **contraste** entre les couleurs de texte et d'arrière-plan est suffisant (sinon le contenu risque d'être difficile à lire).
- CSS3 propose une valeur supplémentaire pour les couleurs RVB afin d'indiquer leur **transparence** : **RVBA**.
- CSS3 permet également de préciser les couleurs sous forme de **valeurs TSL**, avec une valeur de transparence facultative (TSLA).

12. Texte

Taille et police du texte
Gras, italique, majuscule et souligné
Espace entre les lignes, les mots et les lettres

Introduction

- Les propriétés qui permettent de contrôler l'apparence d'un texte se répartissent en deux groupes :
 - **Celles qui affectent directement la police et son aspect**, comme le type du texte, sa taille et s'il est en gras, en italique ou en normal.
 - **Celles qui auraient le même effet sur le texte quelle que soit la police employée**, comme sa couleur et l'espace entre les mots et les lettres.
- NB. La mise en forme du texte peut avoir un impact important sur la **lisibilité** des pages.

Terminologie (1)

AVEC EMPATTEMENT

Dans les polices avec empattement, les terminaisons des lettres comprennent des détails supplémentaires : les empattements.

im

SANS EMPATTEMENT

Dans les polices sans empattement, les terminaisons des lettres sont droites. Elles ont donc un aspect plus net.

im

CHASSE FIXE

Dans une police à chasse fixe, ou à espacement constant, chaque lettre possède la même largeur (dans les autres polices, la largeur de chaque caractère varie).

im

Terminologie (2)

GRAISSE

Light

Medium

Bold

Black

La graisse, ou poids, de la police non seulement permet de mettre l'accent sur le texte, mais peut également agir sur l'espace vide et le contraste de la page.

STYLE

Normal

Italic

Oblique

Les caractères des polices italiques ont un aspect cursif. Le style oblique se fonde sur le style normal et applique un angle aux caractères.

ÉTIREMENT

Condensed

Regular

Extended

En version condensée, ou étroitiée, de la police, les lettres sont plus fines et plus resserrées. En version étendue, elles sont plus épaisses et plus éloignées.

Choix d'une police

- Avant de choisir une police de caractères, il est important de comprendre que, en général, le navigateur pourra l'afficher **uniquement si elle est installée sur l'ordinateur de l'internaute.**
- Par conséquent, les sites emploient souvent un **jeu de polices réduit** que l'on retrouve sur la plupart des ordinateurs. (Il existe des techniques pour contourner cela.)

AVEC EMPATTEMENT

Les polices avec empattement ajoutent des détails sur les terminaisons des lettres.

EXEMPLES :

Georgia

Times

Times New Roman

SANS EMPATTEMENT

Dans les polices sans empattement, les lettres ont des terminaisons plus droites et sont ainsi plus nettes.

EXEMPLES :

Arial

Verdana

Helvetica

CHASSE FIXE

Toutes les lettres d'une police à chasse fixe ont la même largeur; elle varie dans les autres polices.

EXEMPLES :

Courier

Courier New

CURSIVE

Dans les polices cursives, les lettres sont arrondies ou se rejoignent, comme dans une écriture manuelle.

EXEMPLES :

Comic Sans MS

Monotype Corsiva

FANTAISIE

Les polices fantaisistes sont généralement décoratives, sont employées pour les titres et ne sont pas adaptées aux longs paragraphes de texte.

EXEMPLES :

Impact

Haettenschweiler

Techniques pour un large choix de polices

- **FONT-FAMILY** — La police doit être installée sur l'ordinateur. CSS permet d'indiquer la police employée.
- **FONT-FACE** — CSS précise où la police peut être téléchargée si elle n'est pas présente sur l'ordinateur.
- **FONT-FACE par un service** — Un service commercial donne accès à un plus grand nombre de polices via [@font-face](#).
- **IMAGES** — Vous pouvez créer une image à partir du texte affiché dans la police souhaitée.
- **sIFR** — La police est incorporée dans une animation Flash et un script JavaScript remplace le texte HTML indiqué par sa version Flash.
- **CUFON** — Cufon apporte une fonctionnalité comparable à sIFR. Il utilise JavaScript pour créer une version SVG ou VML du texte.

Choix de la police **font-family**

```
body {  
    font-family: Georgia, Times, serif;}  
h1, h2 {  
    font-family: Arial, Verdana, sans-serif;}  
.credits {  
    font-family: "Courier New", Courier,  
    monospace;}
```

- La propriété **font-family** permet d'indiquer la **police dans laquelle doit être affiché le texte** des éléments sélectionnés par la règle CSS.
- La valeur de cette propriété est le **nom de la police choisie**. Pour que le texte soit affiché dans cette police, elle doit être **installée sur l'ordinateur de l'internaute** qui se rend sur votre site.
- Vous pouvez fournir une **liste de polices séparées par des virgules** afin que, si l'internaute ne dispose pas de votre premier choix, le navigateur puisse essayer l'une des autres polices de la liste.
- Il est également fréquent de **terminer la liste par le nom générique** du type de police souhaité.

Taille du texte

font-size

css

```
body {  
    font-family: Arial, Verdana, sans-serif;  
    font-size: 12px;}  
  
h1 {  
    font-size: 200%;}  
  
h2 {  
    font-size: 1.3em;}
```

RÉSULTAT

Berger de Brie

source Wikipédia

Le [berger de Brie](#), ou briard, est un chien de berger français qui à l'origine servait à la garde et la conduite des troupeaux.

Histoire de la race

On ignore quels sont les ancêtres du berger de Brie, cependant, on estime généralement qu'il serait le résultat d'un croisement entre le barbet et le beauceron. Race longtemps ignorée par son faible nombre d'individus, la mention de "Chien de Brie" n'est apparue qu'en 1863, lors d'une exposition canine à Paris.

- La propriété **font-size** permet de préciser la **taille de la police**. Il existe plusieurs manières de procéder, mais voici les plus répandues :
- **PIXELS** — La taille est souvent indiquée en pixels car les designers web ont ainsi un contrôle très précis sur l'espace occupé par le texte. Le nombre de pixels est suivi des lettres **px**.
- **POURCENTAGE** — Dans les navigateurs, le texte a une taille par défaut fixée à 16px. Par conséquent, une taille de 75 % équivaut à 12px, et une taille de 200 %, à 32px.
- **EM** — Un **em** est équivalent à la largeur de la lettre **m**.

Grand choix de police @font-face

The screenshot shows a web page with two main sections: 'CSS' and 'RÉSULTAT'.

CSS:

```
@font-face {  
    font-family: 'ChunkFiveRegular';  
    src: url('fonts/chunkfive.eot');}  
  
h1, h2 {  
    font-family: ChunkFiveRegular, Georgia, serif;}
```

RÉSULTAT:

Berger de Brie

source Wikipédia

Le [berger de Brie](#), ou briard, est un chien de berger français qui à l'origine servait à la garde et la conduite des troupeaux.

Histoire de la race

On ignore quels sont les ancêtres du berger de Brie, cependant, on estime généralement qu'il serait le résultat d'un croisement entre le barbet et le beauceron.

- **@font-face** permet d'utiliser une police même si elle n'est pas installée sur l'ordinateur de l'internaute. Pour cela, vous indiquez le **chemin vers un exemplaire de cette police**, qui sera téléchargé si nécessaire.
- La police souhaitée est ajoutée à la feuille de style via une règle **@font-face** illustrée ci-contre.
- **font-family** Cette propriété précise le **nom de la police**. Il peut ensuite être employé comme valeur de la propriété font-family dans les autres règles, à l'instar des éléments **<h1>** et **<h2>** de l'exemple.
- **src** Cette propriété indique le **chemin de la police**. Pour que la technique fonctionne dans tous les navigateurs, vous devrez probablement proposer des chemins vers différentes versions de la police.

Formats de polices

NAVIGATEUR	eot	woff	ttf / otf	svg
Chrome (all)				●
Chrome 6+		●	●	●
Firefox 3.5			●	
Firefox 3.6+		●	●	
IE 5 - 8	●			
IE 9+	●	●	●	
Opera 10+			●	●
Safari 3.1+			●	●
iOS <4.2	●			●
iOS 4.2+			●	●

- Comme ils reconnaissent différents formats audio et vidéo, les navigateurs prennent en charge différents formats de polices. Pour cibler tous les navigateurs, vous devrez donc **fournir plusieurs variantes de la police.**
- Si vous ne disposez pas de la police dans tous ces formats, envoyez-la sur le site **Font Squirrel**, qui la convertira pour vous : <http://www.fontsquirrel.com/tools/webfont-generator>

⚠ —> <https://caniuse.com/#feat=woff>

Gras et italique

- **Gras** : **font-weight**

```
.credits { font-weight: bold; }
```

- **Italique** : **font-style**

```
.credits { font-style: italic; }
```

Divers (1)

- **Majuscules/Minuscules** : **text-transform**

MAJUSCULES : p { text-transform: uppercase; }

MINUSCULES : p { text-transform: lowercase; }

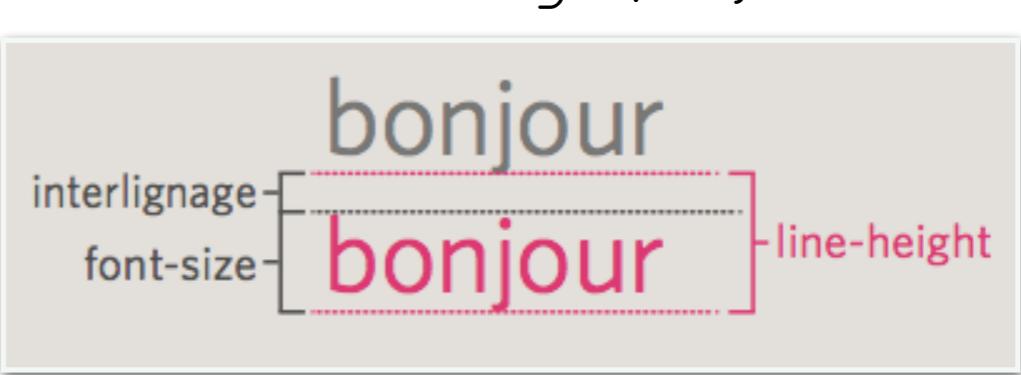
- **Texte souligné et barré** : **text-decoration**

SOULIGNÉ : p { text-decoration: underline; }

BARRRÉ : p { text-decoration: line-through; }

- **Interlignage** : **line-height**

p { line-height: 1.4em; }



- **Espace entre les mots**: **letter-spacing**, **word-spacing**

h1 { letter-spacing: 0.2em; }

Divers (2)

- **Alignement** : `text-align`

Valeurs possibles : left, right, center, justify

- **Alignement vertical** : `vertical-align`

Valeurs possibles : baseline, top, bottom

- **Indentation du texte** : `text-indent`

```
.credits { text-indent: 20px; }
```

- **Ombre portée (CSS3)** : `text-shadow`

```
p { text-shadow: 1px 1px 3px #666666; }
```

Style des liens

```
a:link {  
    color: deeppink;  
    text-decoration: none;}  
  
a:visited {  
    color: black;}  
  
a:hover {  
    color: green;  
    text-decoration: underline;}  
  
a:active {  
    color: darkcyan;}
```

- Par défaut, les navigateurs ont tendance à afficher les **liens en bleu et à les souligner**. Ils changent la couleur des liens visités afin d'indiquer aux internautes les pages qu'ils ont consultées.

CSS définit **deux pseudo-classes** pour appliquer des styles différents aux liens qui ont été **visités** ou **non visités**.

- **:link** — Cette pseudo-classe permet de créer des styles pour les **liens qui n'ont pas encore été visités**.
- **:visited** — Cette pseudo-classe permet de définir des styles pour les **liens sur lesquels l'internaute a cliqué**.

Réponse à l'internaute

```
input {  
    padding: 6px 12px 6px 12px;  
    border: 1px solid #665544;  
    color: #ffffff;}  
  
input.submit:hover {  
    background-color: #665544;}  
  
input.submit:active {  
    background-color: chocolate;}  
  
input.text {  
    color: #cccccc;}  
  
input.text:focus {  
    color: #665544;}
```

- Trois **pseudo-classes** permettent de modifier l'apparence d'un élément **en réponse à une action de l'internaute** :
 - **:hover** — Cette pseudo-classe est appliquée lorsque l'internaute **survole** un élément avec un dispositif de pointage tel qu'une souris, par ex un bouton ou un lien.
 - **:active** — Cette pseudo-classe est appliquée lorsqu'un élément est **activé** par l'internaute, par exemple lorsqu'il appuie sur un bouton ou clique sur un lien.
 - **:focus** — Cette pseudo-classe est appliquée lorsqu'un élément possède le **focus**. Tout élément avec lequel l'internaute peut interagir, comme un lien ou un contrôle de formulaire, peut posséder le focus.

Sélecteur d'attribut

- Nous avons déjà décrit les sélecteurs les plus connus. Il existe toutefois d'autres sélecteurs d'attribut grâce auxquels vous pouvez créer des règles qui s'appliquent aux **éléments dont un attribut a une valeur précise** :

SÉLECTEUR	DESCRIPTION	EXEMPLE
EXISTENCE	[Correspond à un attribut précis, quelle que soit sa valeur	p[class] Cible tout élément <p> qui possède un attribut class
ÉGALITÉ	[=] Correspond à un attribut précis qui a une valeur spécifique	p[class="chien"] Cible tout élément <p> qui possède un attribut class dont la valeur est chien
ESPACE	[~=] Correspond à un attribut précis dont la valeur est donnée dans une liste de mots séparés par des espaces	p[class~="chien"] Cible tout élément <p> avec un attribut class dont la valeur est une liste de mots séparés par des espaces, l'un d'eux étant chien
PRÉFIXE	[^=] Correspond à un attribut précis dont la valeur commence par la chaîne indiquée	p[attr^"c"] Cible tout élément <p> qui possède un attribut dont la valeur commence par la lettre "c"

En résumé

- Il existe des propriétés pour contrôler la **police**, la **taille**, la **graisse**, le **style** et l'**espacement** du texte.
- Le choix parmi les polices installées sur les ordinateurs de tous les internautes est **limité**.
- Pour utiliser d'autres polices, il existe **plusieurs solutions**, mais vous devez disposer d'une licence qui autorise ces pratiques.
- Vous pouvez ajuster l'espace qui sépare les **lignes** de texte, les **lettres** et les **mots**. Le texte peut être aligné à gauche, à droite, au centre ou justifié. Il peut également être indenté.
- Des **pseudo-classes** permettent de modifier le style d'un élément lorsque l'internaute survole ou clique sur du texte, ou lorsqu'il a visité un lien.

EXERCICE : Texte CSS

- Dans `css_texte/index.html`, apportez les modifications suivantes :
 - Ajouter 20px d'espace tout autour de la page.
 - Donner la couleur `#0088dd` à tous les titres.
 - Donner une ombre au titre `<h1>`.
 - Passer en majuscules le titre `<h2>`.
 - Afficher la première ligne du premier paragraphe en gras.
 - Afficher le nom de l'auteur "by Ivy Duckett" aligné à droite et en italique.
 - Faites en sorte que les liens se soulignent automatiquement au survol de la souris.
- Une capture d'écran du résultat final est disponible : `css_texte/screenshot.png`

13. Boîtes

Contrôle de la taille des boîtes
Modèle pour les bordures, les marges et l'espacement
Affichage et masquage des boîtes

Introduction

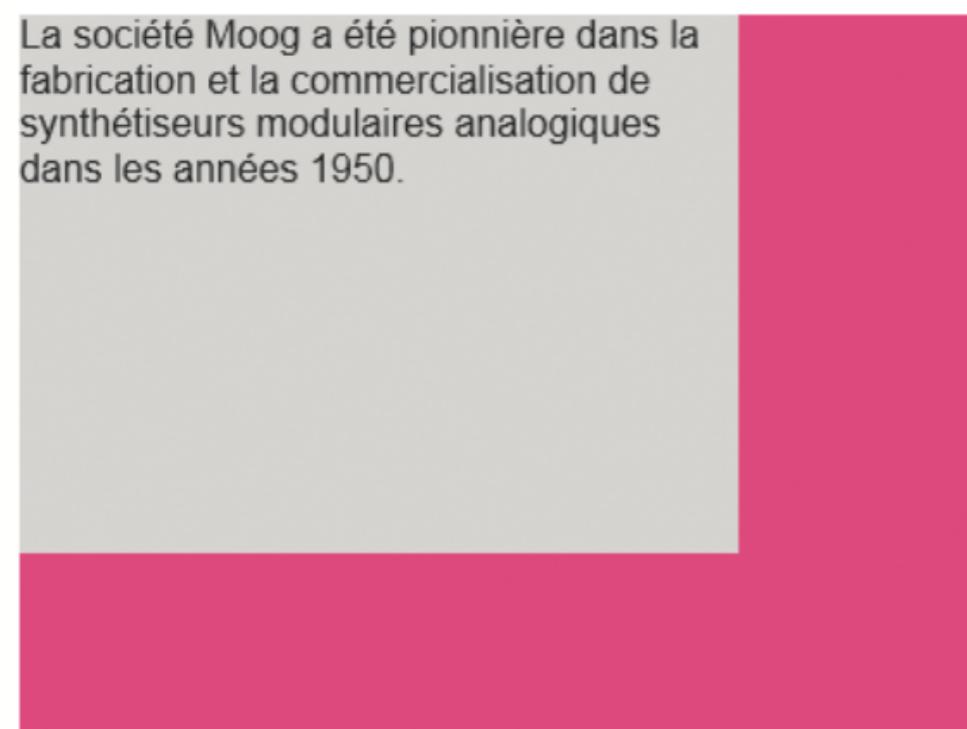
- Au début de cette partie, nous avons expliqué que CSS traite chaque élément HTML comme s'il se trouvait **à l'intérieur de sa propre boîte**.
- **Plusieurs propriétés** affectent l'apparence de ces boîtes.
- Dans cette leçon :
 - Contrôle des **dimensions** des boîtes ;
 - Création de **bordures** autour des boîtes ;
 - Ajustement des **marges** et de l'**espacement** des boîtes ;
 - **Affichage** et **masquage** des boîtes.

Dimensions d'une boîte

width, height

```
div.box {  
    height: 300px;  
    width: 300px;  
    background-color: #bbbbbaa;}  
  
p {  
    height: 75%;  
    width: 75%;  
    background-color: #0088dd;}
```

La société Moog a été pionnière dans la fabrication et la commercialisation de synthétiseurs modulaires analogiques dans les années 1950.



- Par défaut, les dimensions d'une boîte sont **adaptées à son contenu**. Pour les changer, utilisez les propriétés **height** et **width**.
- En général, la taille d'une boîte est indiquée en **pixels**, en **pourcentages** ou en **em**. Les pixels sont les plus employés car ils permettent aux designers de donner une taille précise.
- Avec les pourcentages, la taille de la boîte est **relative à celle de la fenêtre du navigateur** ou, si la boîte se trouve à l'intérieur d'une autre, à la taille de celle-ci.
- Lorsque les mesures sont données en **em**, la taille de la boîte se fonde sur celle du texte qu'elle contient. Les designers se servent de plus en plus souvent des pourcentages et des **em** car ils tentent de **créer des mises en page qui s'adaptent aux appareils équipés d'écrans de taille variée**.

Largeur et hauteur limitées

- **Largeur limitée** `min-width`, `max-width` — Certaines mises en page sont capables de s'adapter à la taille de l'écran de l'internaute. La propriété `min-width` précise alors la **largeur minimale** d'une boîte lorsque la fenêtre du navigateur est réduite, tandis que la propriété `max-width` indique la largeur maximale lorsque la fenêtre est agrandie.
- Ces propriétés se révèlent très utiles lorsqu'on veut **garder le contenu des pages lisible**, notamment sur les petits écrans des appareils portables. Par exemple, vous pouvez utiliser la propriété `max-width` pour que les lignes de texte ne soient **pas trop longues** dans une fenêtre de navigateur de grande largeur, et `min-width` pour qu'elles ne soient **pas trop courtes**.
- **Hauteur limitée** `min-height`, `max-height` — Idem que `min-width` et `max-width`, mais pour la hauteur.

HTML

```
<h2>Fender Stratocaster</h2>
<p class="one">La Stratocaster est le second
modèle de guitare électrique produit par la
marque américaine Fender... </p>
<h2>Gibson Les Paul</h2>
<p class="two">C'est en 1952 que le premier modèle
de guitare Les Paul est sorti de l'usine Gibson
de Kalamazoo... </p>
```

CSS

```
p.one {
  overflow: hidden;}
p.two {
  overflow: scroll;}
```

RÉSULTAT

Fender Stratocaster

La Stratocaster est le second modèle de guitare électrique produit par la marque américaine Fender. Elle succède à la Telecaster, sans

Gibson Les Paul

C'est en 1952 que le premier modèle de guitare Les Paul est sorti de l'usine Gibson de Kalamazoo.

Contenu en excès: **overflow**

- La propriété **overflow** indique au navigateur ce qu'il doit faire lorsque **la taille d'une boîte n'est pas suffisante pour son contenu**.
- Voici ses deux valeurs valides :
 - **hidden** — Dans ce cas, le contenu qui n'entre pas dans la boîte est simplement **masqué**.
 - **scroll** — Avec cette valeur, une **barre de défilement** est ajoutée à la boîte afin que l'internaute puisse la faire défiler et voir le contenu en excès.

Bordure, marge et espacement

- Chaque boîte dispose de **3 propriétés** que vous pouvez ajuster pour modifier son apparence :

BORDURE	MARGE	ESPACEMENT
<p>Chaque boîte dispose d'une bordure, même si elle n'est pas visible ou que son épaisseur est fixée à zéro pixel. Elle sépare le bord d'une boîte du bord d'une autre boîte.</p> <p>Les dimensions de la bordure, de la marge et de l'espacement sont ajoutées à la largeur et à la hauteur de la boîte.</p>	<p>La marge se trouve à l'extérieur de la bordure. En modifiant sa largeur, vous créez un espace entre les bordures de boîtes adjacentes.</p>	<p>L'espacement correspond à l'espace qui se trouve entre la bordure d'une boîte et son contenu. L'ajout d'un espacement peut améliorer la lisibilité du contenu.</p>
		

Bordure

- **Largeur de bordure** **border-width** — Définit la largeur d'une bordure. Elle peut être indiquée en pixels ou à l'aide des valeurs prédéfinies : **thin medium thick**.
- **Style de bordure** **border-style** — Modifie le style d'une bordure ; voici les valeurs : **solid** pour une ligne pleine, **dotted** pour des pointillés carrés, **dashed** pour des tirets, etc.
- **Couleur de bordure** **border-color** — La couleur d'une bordure est indiquée par une valeur RVB, un code hexadécimal ou son nom CSS.
- **Raccourci d'écriture** **border** — Permet de préciser la largeur, le style et la couleur d'une bordure en une opération (les valeurs doivent être données dans cet ordre).

```
p {  
width: 250px;  
border: 3px dotted #0088dd;}
```

Espacement & marge

- **Espacement padding** — Précise l'espace qui apparaît **entre le contenu d'un élément et sa bordure**.
p.exemple { padding: 10px; }
- **Marge margin** — Définit l'**espace entre les boîtes**. Sa valeur est souvent donnée en pixels, mais les pourcentages et les em sont également employés.
p.exemple { margin: 20px; }
- **ATTENTION ! Contenu centré** — Si vous souhaitez centrer une boîte dans la page ou à l'intérieur d'un autre élément, fixez les propriétés **margin-left** et **margin-right** à **auto**.
p.exemple {
 margin: 10px auto 10px auto;
 text-align: left; }

Niveau de texte / Bloc

chapitre-13/display.html

HTML

```
<ul>
  <li>Accueil</li>
  <li>Produits</li>
  <li class="coming-soon">Services</li>
  <li>À propos</li>
  <li>Contact</li>
</ul>
```

CSS

```
li {
  display: inline;
  margin-right: 10px;}
li.coming-soon {
  display: none;}
```

- **Niveau de texte / Bloc `display`** — Permet de transformer un élément de niveau texte en élément de niveau bloc, et vice versa. Elle sert également à masquer un élément. Voici ses valeurs reconnues :
 - **inline** : L'élément de niveau bloc se comporte comme un élément de niveau texte.
 - **block** : L'élément de niveau texte se comporte comme un élément de niveau bloc.

- Boîtes masquées `visibility`** — Permet de masquer le contenu d'une boîte mais l'espace qu'aurait dû occuper l'élément est conservé. Voici ses valeurs reconnues :
- **hidden** L'élément est masqué.
 - **visible** L'élément est affiché.

Boîtes et CSS3

- **Image de bordure border-image** — Applique une image à la bordure d'une boîte. Sa valeur est une image d'arrière-plan qu'elle découpe en neuf parties.
- **Ombre de boîte box-shadow** — Ajoute une ombre portée à une boîte. Elle opère à la manière de la propriété text-shadow. Elle exige au moins la première des deux valeurs suivantes, ainsi que la couleur : décalage horizontal, décalage vertical, distance de flou, diffusion de l'ombre.
- **Angles arrondis border-radius** — Ajoute la possibilité d'obtenir des angles arrondis sur n'importe quelle boîte. La valeur de la propriété précise la taille du rayon en pixels.

En résumé

- CSS traite chaque élément HTML comme s'il possédait **sa propre boîte**.
- À l'aide de CSS, vous pouvez contrôler les **dimensions** d'une boîte.
- Vous pouvez également ajuster la **bordure**, la **marge** et l'**espacement** de chaque boîte.
- Il est possible de **masquer** des éléments à l'aide des propriétés **display** et **visibility**.
- Les éléments de **niveau bloc** peuvent être transformés en éléments de **niveau texte**, et inversement. La lisibilité sera améliorée en ajustant la largeur des boîtes qui contiennent du texte et l'interlignage.
- CSS3 offre la possibilité d'employer des **images** pour les **bordures** et de créer des **angles arrondis**.

EXERCICE : CSS - Boîtes



- Dans `css_boites/index.html`, apportez les modifications suivantes :
 - Tout le contenu de la page apparaît dans une boîte fond blanc bordure deux traits (style : double).
 - Le **logo** apparaît centré avec de l'espace autour.
 - La **liste** `` devient un **menu horizontal** centré dans la page.
 - Les **liens** apparaissent en **majuscules**.
- Une capture d'écran du résultat final est disponible : `css_texte/screenshot.png`

14. Listes, tableaux et formulaires

Changement de style des marqueurs de listes

Ajout de bordures et d'arrière-plans aux tableaux

Modification de l'apparence des éléments de formulaires

Introduction

- Plusieurs propriétés CSS sont prévues pour opérer sur des éléments HTML spécifiques, comme les listes, les tableaux et les formulaires.
- Dans cette leçon :
 - Indication du type de puces ou de la numérotation des **listes** ;
 - Ajout de bordures et d'arrière-plans aux cellules des **tableaux** ;
 - Contrôle de l'apparence des éléments de **formulaires**.

Listes

- **Style des marqueurs** `list-style-type` — Permet de changer la forme ou le style des marqueurs des éléments de listes. Valeurs possibles : `none`, `disc`, `circle`, `square`.
- **Image comme marqueur** `list-style-image` — Préciser une image qui servira de marqueur pour les éléments d'une liste.

```
ul { list-style-image: url ("images/star.png"); }  
li { margin: 10px 0px 0px 0px; }
```
- **Position du marqueur** `list-style-position` — Précise si le marqueur doit être affiché à l'intérieur ou à l'extérieur de la boîte qui comprend les éléments de la liste. Valeurs possibles : `outside`, `inside`.
- **Ecriture raccourcie** `list-style` — Permet de préciser le style, l'image et la position du marqueur, cela dans n'importe quel ordre.

Propriétés de tableau (1)

- Nous avons déjà décrit plusieurs propriétés couramment employées avec les tableaux :
- **width**, pour fixer la largeur du tableau.
- **padding**, pour ajuster l'espace entre la bordure de chaque cellule du tableau et son contenu.
- **text-transform**, pour convertir en majuscule les en-têtes du tableau.
- **letter-spacing** et **font-size**, pour donner un style supplémentaire aux en-têtes du tableau.
- **border-top** et **border-bottom**, pour ajouter des bordures au-dessus et en dessous des en-têtes du tableau.
- **text-align**, pour aligner à gauche le contenu de certaines cellules et à droite celui des autres.
- **background-color**, pour modifier en alternance la couleur d'arrière-plan des lignes du tableau.
- **:hover**, pour mettre en avant une ligne du tableau lorsque le pointeur de la souris la survole.

Propriétés de tableau (2)

- **Bordure des cellules vides** `empty-cells` — Si votre tableau comprend des cellules vides, servez-vous de la propriété `empty-cells` pour indiquer si leur bordure doit être affichée. Valeurs possibles : `show`, `hide`.
- **Espaces entre cellules** `border-spacing`, `border-collapse` — Contrôle la distance qui sépare des cellules adjacentes. Collapse permet de fusionner les bordures adjacentes en une seule.

Formulaires

- Là encore, on peut réutiliser des propriétés connues :
- Style des champs de saisie : **font-size**, **color**, **border**, **:focus**, **background-image**.
- Style des boutons d'envoi : **color**, **text-shadow**, **border-bottom**, **background-color**, **:hover**.
- **Alignment des contrôles** : PROBLÈME = Les étiquettes des éléments de formulaire ont souvent des longueurs variées et les contrôles ne sont donc pas alignés. SOLUTION = Styler les **<label>** et les **<div>** contenant les champs pour corriger l'alignement.

Style du pointeur

cursor

- La propriété **cursor** permet de contrôler la **forme du pointeur de souris** affiché aux internautes. Par exemple, vous pourriez souhaiter que le pointeur ait la forme d'une main lorsqu'il survole un formulaire.
- Voici les valeurs les plus employées avec cette propriété : **auto crosshair default pointer move text wait help url("cursor.gif")**.
- Vous devez utiliser ces valeurs uniquement pour apporter une **information utile** à l'internaute, là où il s'attend à voir le curseur approprié.

En résumé

- Outre les propriétés CSS présentées dans les autres leçons et applicables au contenu de tous les éléments, **plusieurs sont réservées à l'apparence des listes, des tableaux et des formulaires.**
- L'aspect des marqueurs associés aux éléments de liste est modifiable au travers des propriétés `list-style-type` et `list-style-image`.
- Les **cellules des tableaux** ont éventuellement des bordures et un espacement qui varient selon les navigateurs, mais quelques propriétés permettent de contrôler ces aspects et d'obtenir une présentation plus cohérente.
- Les **formulaires** sont d'un emploi plus simple lorsque les contrôles sont alignés verticalement en utilisant CSS.
- Les **formulaires** profiteront des styles qui les rendent **plus interactifs**.

15. Mise en page

Positionnement des éléments

Création de la mise en page d'un site

Adaptation aux écrans de tailles différentes

Introduction

- Dans cette leçon, nous allons voir **comment positionner les éléments sur la page** et comment créer des mises en page attrayantes.
- Pour atteindre ces objectifs, vous devez comprendre la différence entre concevoir une page pour un écran d'ordinateur et la concevoir pour un autre support, par exemple une impression.
- Dans cette leçon :
 - Les **différentes façons de positionner des éléments**, que ce soit dans le flux normal, par un positionnement relatif ou absolu, ou de manière flottante ;
 - Les différences entre une **mise en page de largeur fixe** et une **mise en page flexible**, ainsi que leur création ;
 - L'utilisation des **grilles** pour des mises en page plus professionnelles.

En bref

- Des éléments `<div>` servent souvent de **conteneurs** pour regrouper des parties de la page.
- Les navigateurs affichent les pages en **flux normal**, sauf si un positionnement **relatif**, **absolu** ou **fixe** est précisé. (Voir notes ci-dessous.)
- La propriété `float` déplace le contenu vers la gauche ou la droite de la page et permet de créer une présentation sur plusieurs colonnes (la largeur des éléments flottants doit être précisée). (Voir notes ci-dessous.)
- La mise en page peut être de **largeur fixe** ou **flexible**.
- Les designers conservent la largeur des pages entre 960 et 1000 pixels. Ils indiquent l'objet du site dans les 600 pixels supérieurs (lisible sans défilement).
- Les **grilles** aident à créer des présentations professionnelles et flexibles. Les **frameworks CSS** fournissent des règles pour les tâches récurrentes.
Exemple : **Bootstrap**.

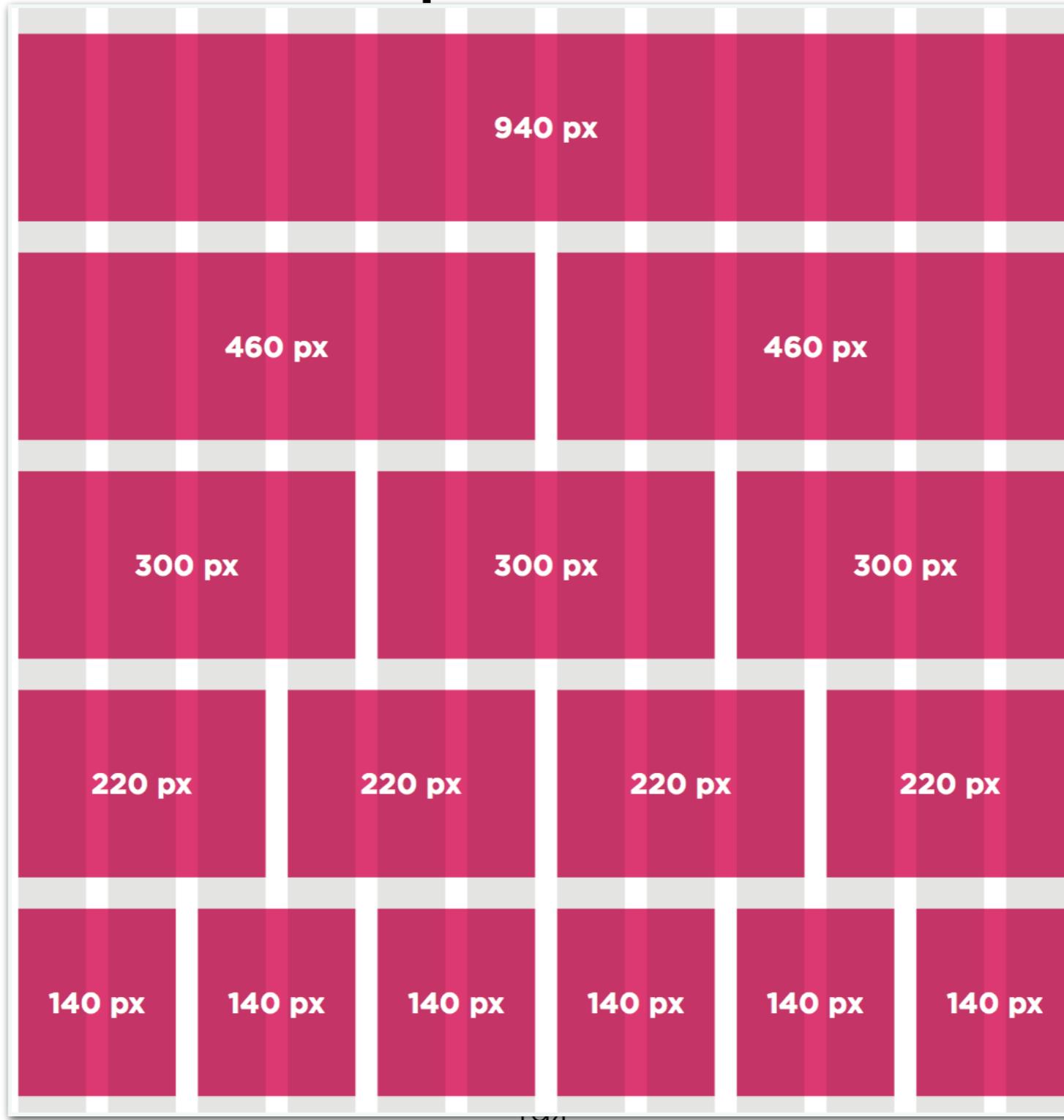
Frameworks CSS

- Les problématiques et les techniques de mise en page sont **extrêmement complexes** (incompatibilités entre navigateurs, différentes tailles de terminaux, etc.)
- Il y a très peu de chances que vous deviez gérer ces problématiques manuellement dans vos projets. A la place, on utilise un système de grille comme celui qu'on trouve dans le framework CSS **Bootstrap** (<https://getbootstrap.com/>).
- Pour la suite de cette partie, nous allons utiliser le framework CSS **Bootstrap**.

Exemple de grille

The screenshot shows the homepage of guardian.co.uk under the 'Culture' category. At the top, there's a navigation bar with links for 'Mobile site', 'Sign in', 'Register', 'Text larger', 'smaller', 'About us', 'Today's paper', and 'Zeitgeist'. Below the navigation is a large banner image of a red flower. The main header 'guardian.co.uk' is followed by a search bar with placeholder 'Your search terms...' and a dropdown menu set to 'Culture'. A horizontal menu below the header includes 'News', 'Sport', 'Comment', 'Culture', 'Business', 'Money', 'Life & style', 'Travel', 'Environment', 'TV', 'Blogs', 'Data', 'Mobile', 'Offers', and 'Jobs'. The 'Culture' link is highlighted with a purple background. Below the menu, the word 'culture' is written in a large, bold, purple font. To the right of the word is a small orange RSS icon labeled 'Webfeed'. A purple horizontal bar spans across the page. Below it, the 'Editors' picks' section features a thumbnail for 'Cannes 2011' showing three men in tuxedos. To the right, a 'Music' section shows a collage of various musicians performing. Further down, a 'Film Weekly from Cannes' section discusses the festival with links to 'Post your comment', 'Video', and 'Full coverage'. Other sections include 'Books', 'Photography', 'Stage', 'TV and radio', and 'Children's books'. On the right side, there's a 'Live webchat' with a photo of a woman, a sidebar titled 'On this site' listing categories like 'Art and design', 'Art', 'Architecture', etc., and a 'Comments' section for the 'Children's books' article.

Mises en page possible (largeur 960px, 12 colonnes)



Bootstrap : Largeur fixe vs flexible

- Bootstrap propose **deux types de conteneurs**.
- Pour un conteneur de **largeur fixe** (et responsive), placez votre code HTML dans **.container** :

```
<div class="container">  
  ...  
</div>
```

- Pour un conteneur de **largeur flexible**, qui prendra toute la largeur de votre *viewport*, placez votre code HTML dans **.container-fluid** :

```
<div class="container-fluid">  
  ...  
</div>
```

Bootstrap : Système de grille

- Bootstrap inclut un **système de grille** fluide, responsive, et mobile first, basé sur une largeur de **12 colonnes**.
- La grille doit être placée dans un **.container** (largeur fixe) ou **.container-fluid** (largeur flexible) pour obtenir un alignement et un espacement corrects.
- On crée une nouvelle **ligne** avec **.row**. Chaque ligne peut contenir jusqu'à 12 colonnes.
- Les **colonnes** sont créées avec les classes **.col-xx-NUMCOL**, en précisant le nombre de colonnes à couvrir sur les 12 disponibles. Par exemple, pour obtenir 3 colonnes égales on utilisera trois **.col-xs-4** ($3 \times 4 = 12$).
- **Responsiveness**. Les préfixes de classe permettent de définir différentes largeurs de colonne pour les différents terminaux : le préfixe **.col-xs-** s'applique aux téléphones, le préfixe **.col-md-** s'applique aux PC de bureau, etc.

Bootstrap : Ex. de grille

.col-xs-12 .col-md-8

.col-xs-6 .col-md-4

.col-xs-6 .col-md-4

.col-xs-6 .col-md-4

.col-xs-6 .col-md-4

.col-xs-6

.col-xs-6

```
<!-- Stack the columns on mobile by making one full-width and the other half-width -->
```

```
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

```
<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
```

```
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

```
<!-- Columns are always 50% wide, on mobile and desktop -->
```

```
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

Feuilles de style multiples

```
@import url("tables.css");
@import url("typography.css");
body {
  color: #666666;
  background-color: #f8f8f8;
  text-align: center;
}
```

```
<html>
  <head>
    <title>Feuilles de style multiples - link
    </title>
    <link rel="stylesheet" type="text/css"
      href="css/site.css" />
    <link rel="stylesheet" type="text/css"
      href="css/tables.css" />
    <link rel="stylesheet" type="text/css"
      href="css/typography.css" />
  </head>
```

- Certains créateurs de pages web répartissent leurs règles CSS dans des **feuilles de style séparées**. Ils utilisent par exemple une feuille de style pour la mise en page et une autre pour les polices, les couleurs, etc.
- Deux façons d'**ajouter plusieurs feuilles de style** :
 1. La page HTML inclut un lien vers une feuille de style, qui emploie la règle `@import` pour importer d'autres feuilles de style.
ATTENTION. Les différentes règles `@import` doivent se trouver **avant toute autre règle**.
 2. La page HTML comprend un élément `<link>` séparé pour chaque feuille de style.

EXERCICE: Faire une mise en page avec Bootstrap

- Vous partirez de `css_mise_en_page/index.html` pour reproduire la mise en page suivante avec le framework CSS Bootstrap :

Attention ! Plus que deux jours pour vous inscrire à la formation !

Comment s'inscrire ?

Saisissez vos informations dans le formulaire ci-contre et nous vous contacterons très prochainement.

E-mail

Mot de passe

Liste des derniers inscrits :

- Pierre
- Jean
- Paul

16. Images

Contrôle de la taille des images en CSS

Alignment des images en CSS

Ajout d'images d'arrière-plan

Introduction

- En contrôlant la taille et l'alignement des images à l'aide de règles CSS, vous pouvez **garder séparé le code qui affecte la présentation des pages et le balisage HTML.**
- Grâce aux **images d'arrière-plan**, vous pouvez également obtenir plusieurs effets intéressants.
- Dans cette leçon :
 - Indication de la **taille** et de **l'alignement** d'une image en CSS;
 - Ajout d'**images d'arrière-plan** à des boîtes.

Contrôle de la taille des images avec CSS

```
  
  

```

HTML

```
img.large {  
    width: 500px;  
    height: 500px;}  
img.medium {  
    width: 250px;  
    height: 250px;}  
img.small {  
    width: 100px;  
    height: 100px;}
```

CSS

- Pour **définir la taille d'une image**, comme pour définir celle d'une boîte, utilisez les propriétés CSS **width** et **height**.
- Lorsque les tailles des images sont définies, le **chargement de la page est plus régulier**. En effet, le contenu HTML et CSS arrive généralement avant les images, et indiquer au navigateur l'espace qu'il doit réserver à une image lui permet d'afficher le contenu environnant sans attendre que cette image soit téléchargée.

Alignement des images avec CSS

HTML

```
<p>
Le genre <b><i>Magnolia</i></b> comprend des
arbres mais aussi des arbustes...</p>
<p>
<i>Magnolia grandiflora</i> est une espèce aux
grandes fleurs et au feuillage persistant et
luisant qui pousse lentement...</p>
```

CSS

```
img.align-left {
  float: left;
  margin-right: 10px;}
img.align-right {
  float: right;
  margin-left: 10px;}
img.medium {
  width: 250px;
  height: 250px;}
```

RÉSULTAT



Le genre **Magnolia** comprend des arbres mais aussi des arbustes. Il est recherché pour ses fleurs remarquables en forme de tulipe ou d'étoile. Pour son entretien, ajoutez régulièrement du terreau de feuilles. C'est un arbre très résistant à la pollution grâce à la texture de ses feuilles.



Magnolia grandiflora est une espèce aux grandes fleurs et au feuillage persistant et luisant qui pousse lentement, ainsi on peut envisager de l'installer en bac. Au bout de quelques années, de grosses fleurs blanches en forme de tulipe apparaissent.

- À la place de l'attribut **align** de l'élément ****, les créateurs de pages web se tournent de plus en plus souvent vers la propriété **float** pour aligner des images. Voici les deux approches les plus employées :

- La propriété **float** est ajoutée à la classe qui représente la taille de l'image (comme la classe **small** dans notre exemple).
- De nouvelles classes sont définies, avec des noms comme **align-left** et **align-right**, pour aligner les images sur la gauche ou la droite de la page. Ces nouvelles classes sont ajoutées à celles qui fixent la taille des images.

- Centrage des images :**

```
img.align-center {
  display: block;
  margin: 0px auto;}
```

Images d'arrière-plan background-image

- La propriété **background-image** place une image derrière n'importe quel élément HTML. Il peut s'agir de la page complète ou seulement d'une partie. Par défaut, l'image d'arrière-plan est répétée de façon à remplir l'intégralité de la boîte.

- Le chemin de l'image est indiqué entre **guillemets** et **parenthèses** après les lettres **url** :

```
p {  
    background-image: url("images/pattern.gif");}
```

- La propriété **background-repeat** permet de contrôler la façon dont l'image se répète : **repeat** (répétée horizontalement et verticalement), **repeat-x** (répétée horizontalement), **repeat-y** (répétée verticalement), **none** (pas répétée) :

```
body {  
    background-image: url("images/header.gif");  
    background-repeat: repeat-x;}
```

- La propriété **background** est un raccourci d'écriture (cf. notes).

En résumé

- Les **dimensions des images** peuvent être **définies en CSS**. Cette solution est très pratique lorsque des images de même taille sont employées sur plusieurs pages.
- CSS permet d'**aligner les images** à la fois horizontalement et verticalement.
- Vous pouvez appliquer une **image d'arrière-plan** à la boîte créée pour tout élément de la page.
- Les images d'arrière-plan peuvent apparaître **une fois** ou **être répétées** sur l'intégralité de l'arrière-plan de la boîte.
- Pour obtenir un **effet de survol** de l'image, déplacez sa position sur l'arrière-plan.
- Pour réduire le nombre d'images que le navigateur doit télécharger, utilisez les **sprites**.

CSS3

Définition, Support, Principales fonctionnalités

Qu'est-ce que CSS3 ?

- Dernière itération en date du langage CSS.
- Le standard CSS3 regroupe en réalité **plusieurs “modules”** (environ 50), que chaque éditeur de navigateur est libre d'implémenter ou non.
- Certains modules contiennent des fonctionnalités purement **visuelles** (polices de caractères riches, animations...) alors que d'autres sont plus **spécialisés** (ex : lecture de texte à voix haute).
- Dans cette formation : **SEULEMENT UNE INTRODUCTION** à CSS3, impossible de voir toutes les fonctionnalités.

Préfixes éditeur

- Les éditeurs de navigateur ajoutent un **préfixe** aux propriétés CSS3 qui sont **en cours de développement**.
- Exemple : Alors que la fonction **radial-gradient()** n'était pas encore disponible dans Firefox, il était possible d'utiliser la syntaxe alternative **moz-radial-gradient()** pour la version en cours de développement.
- Chaque éditeur possède **son propre préfixe** (1) :

PREFIX	FOR BROWSERS
-moz-	Firefox
-webkit-	Chrome, Safari, and the latest versions of Opera (the same rendering engine powers all three browsers)
-ms-	Internet Explorer
-o-	Old versions of Opera (before version 15)

Propriétés CSS3 (1)

- **Transparence** — **rgba** et **opacity** :
(s'applique aux couleurs et aux images)

```
.semitransparentBox {  
    background: rgba(170, 240, 0, 0.5);  
}
```

```
.semitransparentBox {  
    background: rgb(170, 240, 0);  
    opacity: 0.5;  
}
```

- **Coins arrondis** - **border-radius** (1) :

```
.roundedBox {  
    background: yellow;  
    border-radius: 25px 50px 25px 85px;  
}
```

Propriétés CSS3 (2)

- **Images de fond multiples** — **background-image** :
(valeurs séparées par des virgules)

```
.decoratedBox {  
    margin: 50px;  
    padding: 20px;  
    background-image: url('top-left.png'), url('bottom-right.png');  
    background-position: left top, right bottom;  
    background-repeat: no-repeat, no-repeat;  
}
```

- **Ombres portées** — **box-shadow** et **text-shadow** (1) :

```
.shadowedBox {  
    border: thin #336699 solid;  
    border-radius: 25px;  
    box-shadow: 5px 5px 10px gray;  
    text-shadow: gray 10px 10px 7px;  
}
```

Propriétés CSS3 (3)

- **Dégradés :**

- Linéaire — **linear-gradient** (1) :

```
.colorBlendBox {  
    background: linear-gradient(from top, white, blue);  
}
```

- Radial — **radial-gradient** (2) :

```
.colorBlendBox {  
    background: radial-gradient(circle, white, lightblue);  
}
```

Transitions

- Grâce aux **pseudo-classes**, on peut styler les **differents états** d'un élément tel qu'un bouton. Mais le passage d'un état à l'autre est immédiat.
- Les **transitions** permettent de définir comment passer d'un état à l'autre :

```
.slickButton {  
    color: white;  
    font-weight: bold;  
    padding: 10px;  
    border: solid 1px black;  
    background: lightgreen;  
    cursor: pointer;  
    -webkit-transition: background 0.5s;  
    transition: background 0.5s;  
}
```

```
.slickButton:hover {  
    color: black;  
    background: yellow;  
}
```

- La transition est définie dans l'état de départ. Au minimum, il faut définir **la propriété CSS à animer** et la **durée** de l'animation.

Design responsive

- PRINCIPE : La page web s'adapte à la taille du terminal sur lequel elle s'affiche.
- Cela repose sur plusieurs techniques :
 - **Mise en page fluide** (cf. grille Bootstrap fluide).
 - **Images fluides** (occupent toute la taille de leur conteneur) :
`img { max-width: 100%; }`
 - **Typographie fluide** :
`body { font-size: 100% } /* taille de base */
p { font-size: 0.9em } /* em relatif à taille de base */
h1 { font-size: 2em }`
 - **Media Queries** (1) .Permettent de définir des blocs de styles qui s'activent quand les paramètres d'affichage remplissent une certaine condition :
`@media (media-feature-name: value) {
 /* Mettre les styles ici. */
}`

Les frameworks CSS peuvent aider !

Autres fonctionnalités CSS3

- **Canvas**

- Surface vierge sur laquelle on peut dessiner (lignes, formes, dessin libre, images...).
- Implémentation : élément `<canvas>` pour insérer le canvas dans la page + API JavaScript pour dessiner dans le canvas.

- **Animations**

- Consiste à redessiner le canvas un peu différemment plusieurs fois par seconde pour générer un effet d'animation.

- **Interactivité**

- Sur un site web complexe, les effets CSS3 sont fréquemment manipulés en JavaScript de façon à réagir à un événement, à une action de l'utilisateur...

Partie 3 - JavaScript

Introduction

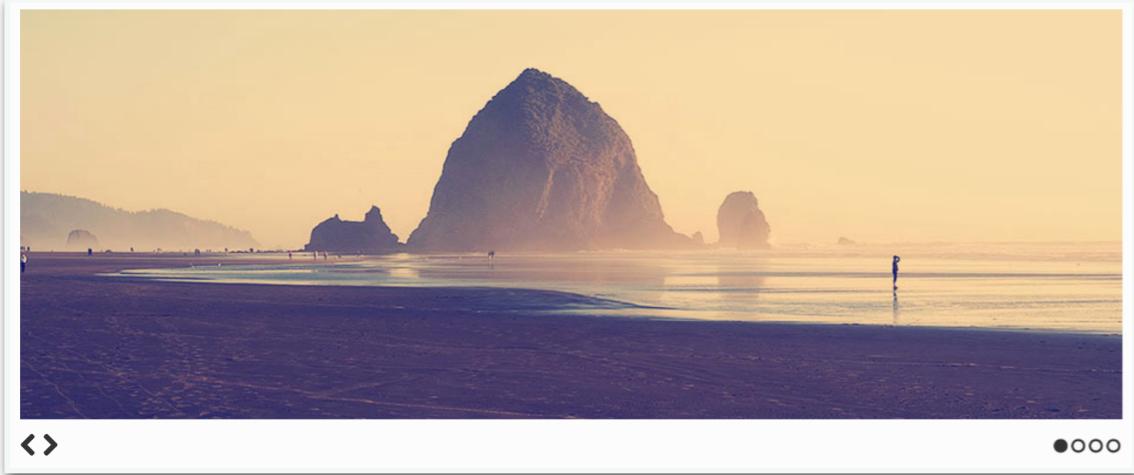
Introduction

- Cette partie de la formation explique comment utiliser JavaScript pour rendre les sites web **plus interactifs** et **faciles à utiliser**.
- Apprendre à programmer en JavaScript implique :
 1. De comprendre quelques **concepts de base** de la programmation et les **mots utilisés** par les programmeurs JavaScript pour désigner ces concepts.
 2. D'apprendre le **langage** lui-même : son vocabulaire, sa structure.
 3. D'apprendre comment JavaScript est utilisé dans des **scénarios concrets**.

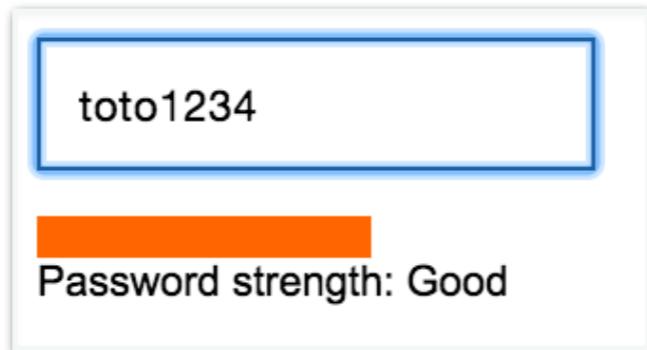
Comment JavaScript rend les pages web plus interactives

- 1. En **accédant** au contenu.
 - Utilisez JavaScript pour sélectionner n'importe quel élément, attribut ou texte d'une page HTML.
- 2. En **modifiant** le contenu.
 - Utilisez JavaScript pour ajouter des éléments, des attributs et du texte à la page, ou pour les retirer.
- 3. En exécutant des **règles de programmation**.
 - Spécifiez une série d'étapes que le navigateur doit suivre (comme une recette), et qui vont lui permettre d'accéder ou de modifier le contenu d'une page.
- 4. En **réagissant aux événements**.
 - Spécifiez quel script doit être exécuté quand un événement particulier se produit.

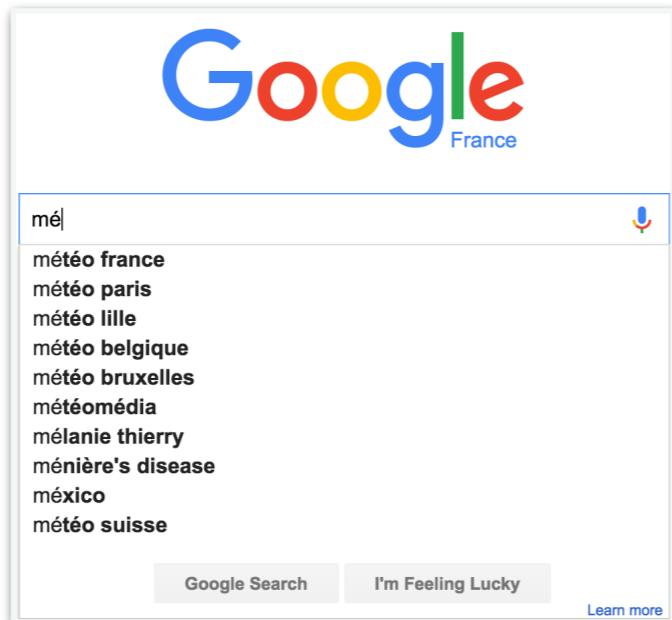
Exemples de JavaScript dans le navigateur



- **Diaporama** (défilement des images, contrôles interactifs...)



- **Formulaires** (validation, mise en avant des erreurs, calcul de valeurs...)



- **Recharger** une partie de la page :



- **Filtrer** les données

17. Les bases de la programmation

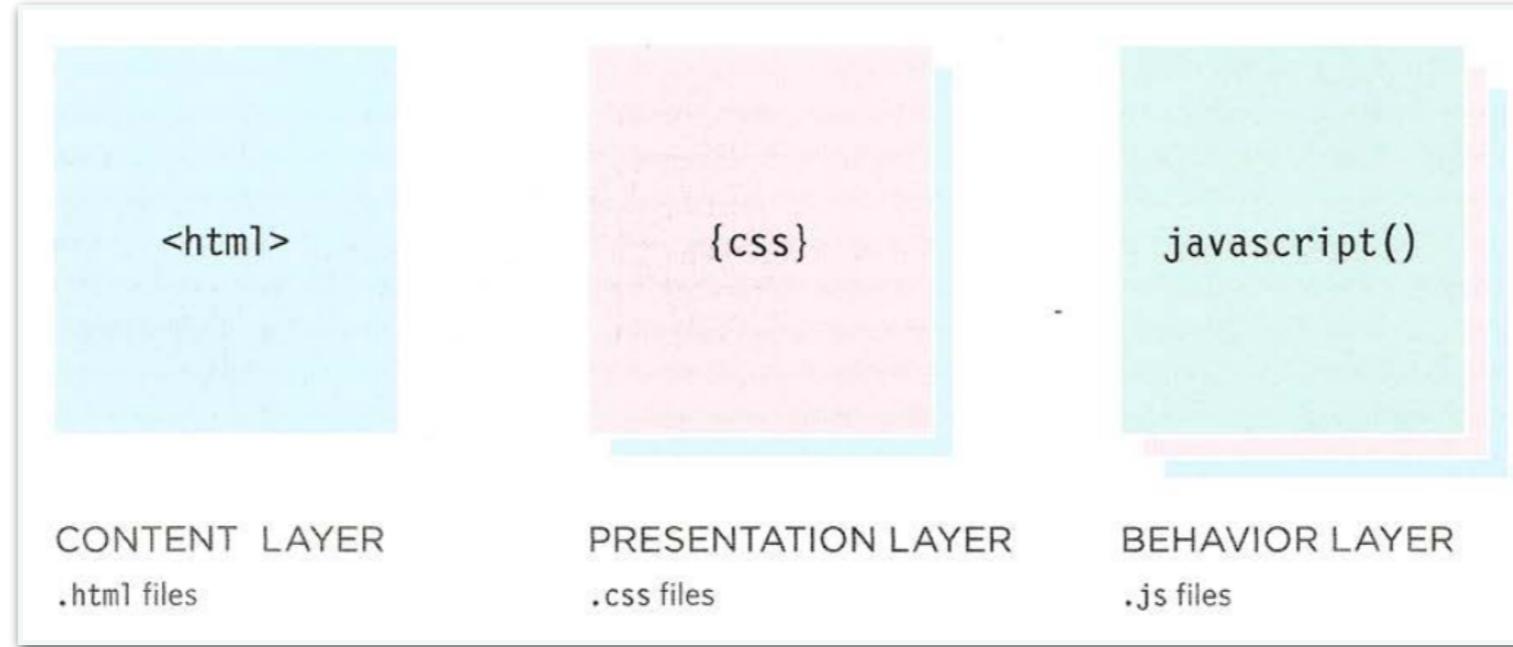
Les scripts

- Un script est une **série d'instructions** qu'un ordinateur peut suivre pour atteindre un objectif. On peut comparer un script à une **recette** ou un **manuel**.
- Chaque fois qu'un script est exécuté, il se peut qu'il n'utilise qu'un **sous-ensemble** de toutes les instructions.
- Les ordinateurs approchent les tâches différemment des humains. Vos instructions doivent donc permettre à l'ordinateur de **résoudre les tâches programmatiquement**.
- Pour écrire un script, il faut d'abord définir l'**objectif**, puis le décomposer en **tâches à exécuter** pour atteindre cet objectif. Chaque tâche est ensuite convertie en **code**.

Les ordinateurs et le monde qui les entoure

- Les ordinateurs créent des **modèles** du monde qui les entourent.
- Ces modèles se basent sur des **objets** pour représenter les choses physiques. Les objets peuvent avoir : des **propriétés** qui nous renseignent sur les caractéristiques de l'objet; des **méthodes** qui exécutent des tâches utilisant les propriétés de l'objet; des **événements** qui sont déclenchés lorsqu'un utilisateur interagit avec l'ordinateur.
- Les programmeurs peuvent écrire du code pour dire "**Quand cet événement se produit, exécute ce code**".
- Les navigateurs web utilisent le HTML pour créer un **modèle de la page web**. Chaque élément HTML crée son propre **noeud** (*node*), qui est une sorte d'objet.
- Pour rendre les pages web interactives, on écrit du code qui **manipule le modèle de la page web** généré par le navigateur, **pas le code source original**.

Comment HTML, CSS et JavaScript sont liés

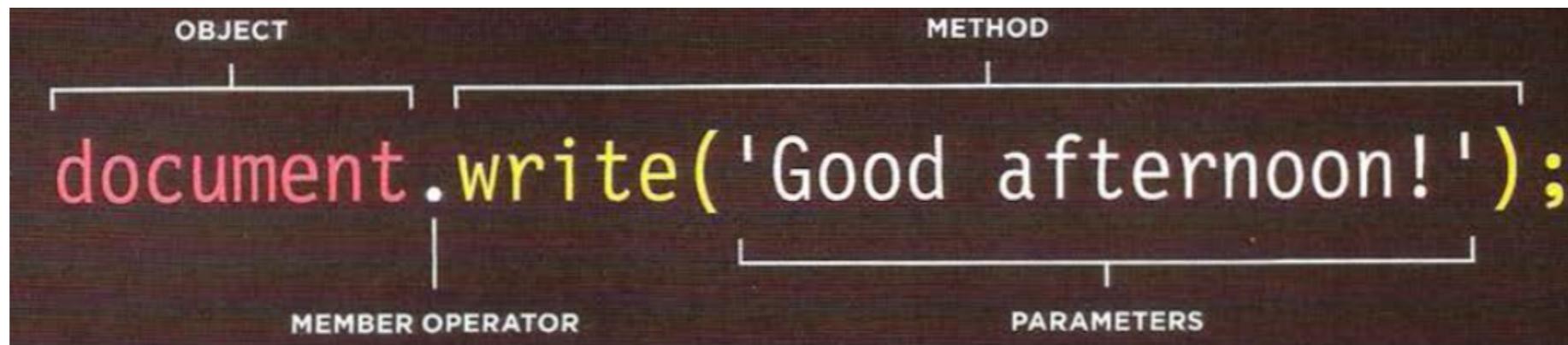


- **HTML**. C'est le contenu de la page. Le HTML ajoute de la **structure** et des **informations sémantiques** à la page.
- **CSS**. Le CSS enrichit le HTML avec à des règles indiquant **comment le contenu doit être présenté** (fonds, bordures, couleurs, polices, positions...)
- **JavaScript**. Permet de **modifier le comportement** d'une page et d'ajouter de l'**interactivité**.
- **Amélioration progressive**. Idée qu'une page web doit quand même être utilisable, même si le JavaScript est désactivé. Vous devez concevoir votre code JavaScript de sorte qu'il améliore les fonctionnalités de la page, sans que la page dépende de lui pour être fonctionnelle.

Créer un script de base

- Le code JavaScript est écrit dans un simple **fichier texte**, tout comme HTML et CSS. Aucun outil particulier n'est nécessaire pour créer du JavaScript.
- Les fichiers contenant du code JavaScript portent l'**extension .js**.
- Pour exécuter un fichier JavaScript dans une page web précise, on utilise la balise **<script>** :
 - Soit en référençant un **fichier extérieur** qui contient le code JavaScript :
`<script src="chemin/vers/fichier.js"></script>`
Le navigateur va charger et exécuter le fichier au moment où il rencontrera la balise.
 - Soit en plaçant le code JavaScript **en-ligne**, directement entre les balises **<script>...</script>**. **Pas recommandé**, car pas de séparation HTML/JS, difficile à tester, impossible d'appeler le même script depuis plusieurs pages.

Comment utiliser les objets et leurs méthodes



- L'objet **document** représente toute la page web. Tous les navigateurs implémentent cet objet, et vous pouvez l'utiliser via son nom **document**.
- La méthode **write()** de l'objet **document** permet d'insérer du contenu dans la page, à l'endroit où se trouve la balise `<script>`.

En résumé

- C'est mieux de mettre le code JavaScript dans son **propre fichier**. Les fichiers JavaScript sont des **fichiers texte** (comme les pages HTML et les feuilles de styles CSS), mais ils portent l'**extension .js**.
- Dans une page HTML, on utilise l'élément `<script>` pour indiquer au navigateur de charger un fichier JavaScript (de la même façon qu'on utilise l'élément `<link>` pour charger un fichier CSS).
- Si l'on affiche le code source de la page dans le navigateur, il ne **réflétera pas les modifications effectuées** en JavaScript. En effet, le script travaille sur une représentation de la page générée par le navigateur, pas sur le code source original.

EXERCICE: Votre premier script

- Partir de js_premier/index.html.
- Ajoutez une balise <script> à l'endroit approprié pour charger le fichier script.js situé dans le même répertoire.
- Notons que si l'on regarde le code source de la page, le HTML n'a pas été modifié. En effet, JavaScript modifie le modèle de page web créé par le navigateur, pas le HTML original.
- **Comment voir le code modifié ?**

18. Instructions JavaScript de base

Dans cette leçon, vous allez apprendre
à lire et écrire du JavaScript.

Statements (déclarations)

- Un script est une **série d'instructions** qu'un ordinateur peut exécuter une par une.
- Chaque instruction est un **statement**.
- Les statements doivent se terminer par un **point-virgule** ;

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening';
} else if (hourNow > 12) {
    greeting = 'Good afternoon';
} else if (hourNow > 0) {
    greeting = 'Good morning';
} else {
    greeting = 'Welcome';
}
document.write(greeting);
```

- **Vert** = statements
- **Rose** (accolades) = blocs de code
- **Violet** = détermine le code à exécuter

**ATTENTION. JavaScript
est sensible à la casse !**

Commentaires

- **BONNE PRATIQUE.** Écrivez des commentaires pour **décrire ce que fait votre code**. Ils rendent votre code plus facile à lire et à comprendre, y compris pour les autres !

```
/* This script displays a greeting to the user based upon the current time.  
It is an example from JavaScript & jQuery book */  
  
var today = new Date();          // Create a new date object  
var hourNow = today.getHours();  // Find the current hour  
var greeting;  
  
// Display the appropriate greeting based on the current time  
if (hourNow > 18) {  
    greeting = 'Good evening';  
} else if (hourNow > 12) {  
    greeting = 'Good afternoon';
```

- Commentaire **multi-lignes** : `/* Commentaire */`
- Commentaire **mono-ligne** : `// Commentaire`
- Les commentaires ne sont pas traités par l'interpréteur JavaScript.

Variable

- Pour effectuer son travail, le script doit **stocker temporairement des informations**. Pour cela, il utilise des **variables**. (Ex : calculer la surface d'une pièce, stocker longueur et largeur.)
- **Déclarer** une variable (on crée la variable en lui donnant un nom)

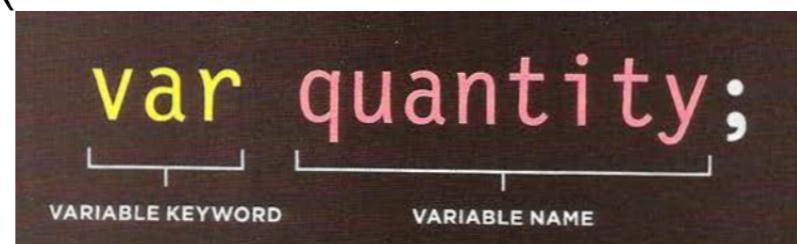


Diagram illustrating the declaration of a variable:

```
var quantity;
```

The code is annotated with labels:

- VARIABLE KEYWORD**: Points to the word "var".
- VARIABLE NAME**: Points to the word "quantity".

- **Assigner** une valeur (dire quelle information la variable contient) :

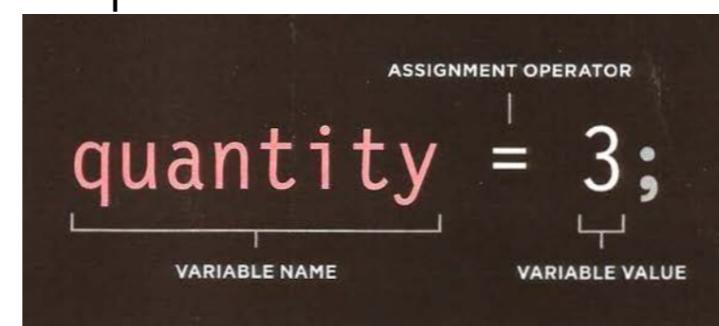


Diagram illustrating the assignment of a value to a variable:

```
quantity = 3;
```

The code is annotated with labels:

- VARIABLE NAME**: Points to the word "quantity".
- ASSIGNMENT OPERATOR**: Points to the equals sign (=).
- VARIABLE VALUE**: Points to the number "3".

- Comme le nom l'indique, la valeur d'une variable peut **changer en cours de script**. Il suffit de lui assigner une nouvelle valeur :

```
inStock = true;  
// Ici, le programme exécute d'autres actions  
// Ici, le programme exécute d'autres actions  
inStock = false;
```

Types de données

- JavaScript distingue **plusieurs types de données**.
- **Numérique** (pour les nombres)
 - Exemple : `price = 0.75;`
- **Chaîne** (pour les lettres ou chaînes de caractères)
 - Exemple : `message = 'Salut, Vince !';`
 - Attention #1 : `'Salut'` ou `"Salut"`, mais pas `"Salut'`
 - Attention #2 : Guillemets dans une chaîne : `"J'espère"` ou `'J\'espère'` (Slash = caractère d'échappement)
- **Booléen** (vrai ou faux)
 - Exemple : `isCustomer = true; // OU BIEN : false`

Raccourcis pour créer les variables

1. **Déclaration et assignation** dans le même statement

```
var price = 5;
```

2. **Plusieurs** variables déclarées **sur la même ligne** (puis valeurs assignées)

```
var price, quantity, total;  
price = 5;  
quantity = 14;  
total = price * quantity;
```

3. **Deux variables déclarées et assignées sur la même ligne**

```
var price = 5, quantity = 14;
```

4. Variable stocke une **référence à un élément HTML** (permet de raccourcir la syntaxe des lignes suivantes)

```
var el = document.getElementById('cost');  
el.textContent = '$' + total;
```

Règles de nommage des variables

1. Le nom doit commencer par une **lettre**, un **\$**, ou un **_**. Pas par un numéro !
2. Le nom peut contenir des **lettres**, des **chiffres**, un **\$** ou un **_**. Pas de tiret (**-**) ou de point (**.**).
3. Ne pas utiliser de **mot-clé** ou de **mot réservé** (comme **var**).
4. Les noms de variables sont **sensibles à la casse** : **score** est différent de **Score**.
5. Utilisez un nom **explicite** : **firstName**, **age**...
6. Utilisez la notation "**camel case**" : **firstName**

Tableaux (arrays)

- Un tableau est un **type particulier de variable**. Au lieu de stocker une seule valeur, il en stocke **plusieurs**. Ces valeurs ont généralement un lien entre elles. Exemple : les articles à acheter dans une liste de courses.
- **CRÉER un tableau :**
 - **Notation littérale** : `var colors = ['white', 'black', 'red'];`
 - **Constructeur Array** : `var colors = new Array('white', 'black', 'red');`
- **ACCÉDER aux valeurs** d'un tableau se fait via l'**indice** de la valeur :
 - **Écrire** : `colors[2] = 'beige';`
 - **Lire** : `var selectedColor = colors[2];`
- **NB. L'index commence à zéro** : `colors[0] == 'white'`
- **Nombre d'éléments** dans un tableau : `colors.length`

Expressions

- Une expression est un "bout de code" qui **résulte en une seule valeur**.
- En gros, il y a **deux types d'expression** :
 1. Les expressions qui assignent une valeur à une variable :

```
var color = 'beige';
// La valeur de `color` est maintenant beige
```
 2. Les expressions qui utilisent deux valeurs ou plus pour renvoyer une seule valeur :

```
var area = 3 * 2;
// La valeur de `area` est maintenant 6
```

Opérateurs

- Les expressions utilisent des **opérateurs**. Ils permettent de créer **une seule valeur à partir de plusieurs** :

ASSIGNMENT OPERATORS

Assign a value to a variable

```
color = 'beige';
```

The value of color is now beige.

COMPARISON OPERATORS

Compare two values and return true or false

```
buy = 3 > 5;
```

The value of buy is false.

ARITHMETIC OPERATORS

Perform basic math

```
area = 3 * 2;
```

The value of area is now 6.

LOGICAL OPERATORS

Combine expressions and return true or false

```
buy = (5 > 3) && (2 < 4);
```

The value of buy is now true.

STRING OPERATORS

Combine two strings

```
greeting = 'Hi ' + 'Molly';
```

The value of greeting is now Hi Molly.

Opérateurs arithmétiques

- **Opérateurs arithmétiques** = Opérateurs mathématiques, qui peuvent être utilisés sur des nombres.

NAME	OPERATOR	EXAMPLE	RESULT
ADDITION	+	10 + 5	15
SUBTRACTION	-	10 - 5	5
DIVISION	/	10 / 5	2
MULTIPLICATION	*	10 * 5	50
INCREMENT	++	i = 10; i++;	11
DECREMENT	--	i = 10; i--;	9
MODULUS	%	10 % 3	1

- **Opérateur de chaîne** : Il n'y en a qu'un, le symbole **+**

```
var firstName = 'Vincent ';  
var lastName = 'Caillierez';  
var fullName = firstName + lastName;
```

En résumé

- Un script est constitué d'une **série de statements**. Chaque statement est comme une **étape** dans une recette.
- Les **variables** sont utilisées pour **stocker temporairement** des informations utilisées dans le script.
- Les **tableaux** sont des types spéciaux de variables qui stockent **plusieurs informations qui ont un lien entre elles**.
- JavaScript fait la distinction entre les **nombres** (0-9), les **chaînes** (texte), et les **valeurs booléennes** (vrai ou faux).
- Les **expressions** résultent en **une seule valeur** après évaluation.
- Les expressions utilisent les **opérateurs** pour calculer une valeur.

EXERCICE: Utiliser les opérateurs

- Partir de js_operateurs/index.html.
- Ajoutez une balise <script> qui référence le fichier script.js situé dans le même répertoire.
- Complétez le fichier script.js de sorte que :
 - Les frais d'envoi (shipping) soient égaux à 15% du sous-total.
 - Le total soit calculé (sous-total + envoi).
 - Les 3 variables (sous-total, envoi, total) soient affichées dans la page.

19. Fonctions, méthodes et objets

Introduction

- Les scripts contiennent des centaines ou des milliers de lignes. Les programmeurs utilisent les **fonctions**, les **méthodes** et les **objets** pour **organiser leur code**.
- Dans cette leçon :
 - **Fonctions & Méthodes.** Les fonctions consistent en une série de *statements* qui ont été *regroupés* pour exécuter une tâche spécifique. Une méthode est identique à une fonction, sauf qu'elle est créée à l'intérieur de (et fait partie de) un objet.
 - **Objets.** Les objets permettent de modéliser les concepts et les choses physiques du monde. Ils possèdent des propriétés et des méthodes.
 - **Objets natifs.** Les navigateurs contiennent un certain nombre d'objets natifs, que vous utiliserez pour rendre vos pages interactives.

Fonction

- Une fonction permet de **regrouper une série de statements** pour exécuter une tâche spécifique. Si différentes parties du script répètent la même tâche, vous pouvez **réutiliser la fonction** (plutôt que de répéter la même série de statements).
- Exemple :

```
var msg = 'Sign up to receive our newsletter for 10% off!';  
function updateMessage() {  
    var el = document.getElementById('message');  
    el.textContent = msg;  
}  
updateMessage();
```

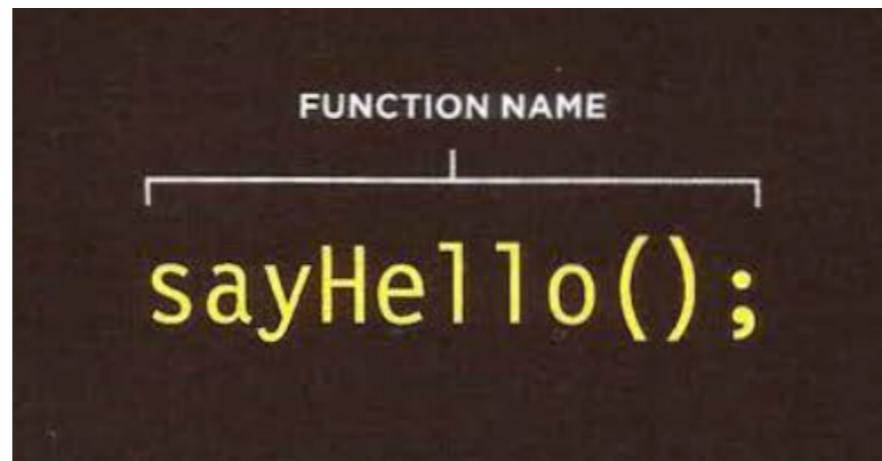
Déclarer une fonction

The diagram shows a snippet of JavaScript code: `function sayHello() { document.write('Hello!'); }`. Annotations with arrows explain the structure:

- A bracket labeled "FUNCTION KEYWORD" points to the word `function`.
- A bracket labeled "FUNCTION NAME" points to the identifier `sayHello()`.
- A bracket labeled "CODE BLOCK (IN CURLY BRACES)" points to the entire block of code enclosed in curly braces {}.

- On déclare une fonction avec le mot-clé **function**.
- On donne un **nom** à la fonction, suivi de **parenthèses**.
- Le statement qui exécute la tâche se trouve dans un **bloc de code** (à l'intérieur d'**accolades**).

Appeler une fonction



- Une fois la fonction déclarée, on peut exécuter tous les statements qu'elle contient avec juste une ligne de code. On dit qu'on **appelle la fonction**.
- Pour exécuter le code de la fonction, il suffit d'écrire son **nom suivi des parenthèses**.
- La même fonction peut être appelée **autant de fois que souhaité**.

Fonction qui nécessite des informations

- Parfois, une fonction a besoin d'informations pour exécuter sa tâche.
- **DÉCLARATION.** On spécifie alors des **paramètres** à la déclaration de la fonction. Ces paramètres agissent comme des **variables** à l'intérieur de la fonction :

```
PARAMETERS  
function getArea(width, height) {  
    return width * height;  
}  
THE PARAMETERS ARE USED LIKE  
VARIABLES WITHIN THE FUNCTION
```

- **APPEL.** Pour appeler une fonction avec des paramètres, on passe les valeurs des paramètres **entre parenthèses** :

- Arguments comme **valeurs** : `getArea(3, 5);`

- Arguments comme **variables** :

```
wallWidth = 3;  
wallHeight = 5;  
getArea(wallWidth, wallHeight);
```

Renvoyer une valeur

- Certaines fonctions **renvoient des informations au code qui les a appelées**. Par exemple, une fonction qui effectue un calcul renvoie le résultat.
- Pour retourner une valeur, on utilise le mot-clé **return** :

```
function calculateArea(width, height) {  
    var area = width * height;  
    return area;  
}  
var wallOne = calculateArea(3, 5);  
var wallTwo = calculateArea(8, 5);
```

- **NB. L'interpréteur quitte la fonction après le mot-clé return.**
- Remarque : Il est possible de **retourner plusieurs valeurs** via un tableau : **return [area, volume]** ; Le code appelant peut ensuite accéder aux valeurs du tableau via leur indice.

Fonctions anonymes & expressions de fonction

```
function area(width, height) {  
    return width * height;  
};  
  
var size = area(3, 4);
```

```
var area = function(width, height) {  
    return width * height;  
};  
  
var size = area(3, 4);
```

- **Déclaration de fonction (*function statement*)**
- Une déclaration de fonction crée une fonction qui peut être appelée plus tard dans le code.
- L'interpréteur recherche toutes les variables et déclarations de fonctions AVANT de parcourir le script ligne par ligne. Une fonction déclarée ainsi peut donc **être appelée avant d'être déclarée**.
- **Expression de fonction (*function expression*)**
- Quand on met une fonction à un endroit où l'interpréteur attend une expression, on crée une expression de fonction. Dans ce cas, le nom de la fonction est souvent omis et on parle de **fonction anonyme**.
- Les fonctions déclarées ainsi ne sont découvertes par l'interpréteur qu'au moment où il atteint la ligne contenant l'expression. Impossible de les appeler avant leur déclaration.

IIFE (*Immediately Invoked Function Expressions*)

- Permet d'éviter que les noms de variable **entre en conflit** (surtout lorsque plusieurs scripts sont utilisés).

```
var area = (function() {  
    var width = 3;  
    var height = 2;  
    return width * height;  
}());
```

- Dans cet exemple, la variable **area** contient la valeur renvoyée par la fonction (plutôt que de contenir la fonction elle-même, qui pourrait être appelée plus tard)
- Quand utiliser** les fonctions anonymes et IIFE ?
 - En gros** : Pour du code qui doit être exécuté une fois dans le cadre d'une tâche, plutôt qu'être appelé plusieurs fois par différentes parties du script.
 - Typiquement** : Dans les gestionnaires d'événements. Aux endroits qui attendent une fonction de callback et quand on veut passer des paramètres à la fonction de callback.

Portée des variables (Scope)

- **L'endroit où une variable est déclarée** affecte où elle peut être utilisée :

```
function getArea(width, height) {  
    var area = width * height;  
    return area;  
}  
  
var wallSize = getArea(3, 2);  
document.write(wallSize);
```

LOCAL (OR FUNCTION-LEVEL) SCOPE
GLOBAL SCOPE

- **Local.** Quand une variable est déclarée dans une fonction avec le mot-clé **var**, elle n'est visible (et utilisable) que dans cette fonction. C'est une **variable locale**.
- **Global.** Quand une variable est déclarée en dehors d'une fonction, elle peut être utilisée n'importe où dans le script. C'est une **variable globale**.

Objet

- Les objets regroupent un ensemble de **variables** et de **fonctions** pour modéliser quelque chose du monde réel.
- Dans un objet, les variables s'appellent des **propriétés**, et les fonctions s'appellent des **méthodes**.

Créer un objet (notation littérale)

```
var hotel = {  
  
    name: 'Quay',  
    rooms: 40,  
    booked: 25,  
    gym: true,  
    roomTypes: ['twin', 'double', 'suite'],  
  
    checkAvailability: function() {  
        return this.rooms - this.booked;  
    }  
};
```

KEY
VALUE

PROPERTIES
These are variables

METHOD
This is a function

Accéder à un objet

- On accède aux propriétés et aux méthodes d'un objet via la **notation point** :

The diagram shows two lines of JavaScript code on a dark background. The first line is `var hotelName = hotel.name;` and the second is `var roomsFree = hotel.checkAvailability();`. Above the code, a bracket labeled "OBJECT" spans the word "hotel" in both lines. Another bracket labeled "PROPERTY/METHOD NAME" spans ".name" and ".checkAvailability()". A vertical line labeled "MEMBER OPERATOR" points down to the dot character in each line.

```
var hotelName = hotel.name;
var roomsFree = hotel.checkAvailability();
```

- Ou via les **crochets** (pour accéder aux propriétés uniquement) :

```
var hotelName = hotel['name'];
```

Créer un objet (notation constructeur)

```
var hotel = new Object();

hotel.name = 'Quay';
hotel.rooms = 40;
hotel.booked = 25;

hotel.checkAvailability = function() {
    return this.rooms - this.booked;
};
```

The diagram illustrates the creation of an object named 'hotel' using the `new Object()` constructor. It shows three properties (`name`, `rooms`, and `booked`) grouped together with a bracket labeled 'PROPERTIES'. Below them is a method (`checkAvailability`) enclosed in curly braces, with a bracket labeled 'METHOD' to its right.

- Le mot-clé `new` et le **constructeur objet** permettent de créer un objet vierge.
- On peut ensuite ajouter des **propriétés** et des **méthodes** à l'objet.

Modifier un objet

- Pour mettre à jour les valeurs des propriétés, on utilise la **notation point** :

The diagram shows the expression `hotel.name = 'Park';` with annotations. A bracket labeled "OBJECT" spans from the start of `hotel` to the dot. Another bracket labeled "PROPERTY NAME" spans from the dot to the `.name` part. A third bracket labeled "PROPERTY VALUE" spans from the equals sign to the end of the string. Below the code, the word "MEMBER OPERATOR" is aligned under the dot, and "ASSIGNMENT OPERATOR" is aligned under the equals sign.

```
hotel.name = 'Park';
```

OBJECT PROPERTY NAME PROPERTY VALUE
| | |
MEMBER OPERATOR ASSIGNMENT OPERATOR

- Ou les **crochets** :

```
hotel['name'] = 'Park';
```

- Pour **effacer** une propriété, on utilise le mot-clé **delete** :

```
delete hotel.name;
```

Créer plusieurs objets similaires (notation constructeur)

- Lorsqu'on veut **plusieurs objets qui représentent des choses similaires**, on utilise un **constructeur** qui représente un template de création des objets.

- **1) Déclaration du constructeur :**

```
function Hotel(name, rooms, booked) {  
    this.name = name;  
    this.rooms = rooms;  
    this.booked = booked;  
  
    this.checkAvailability = function() {  
        return this.rooms - this.booked;  
    };  
}
```

Le **mot-clé this** représente l'instance de l'objet qui sera instancié.

- **2) Crédit des instances**
avec **new Objet()** :

```
OBJECT  
CONSTRUCTOR FUNCTION  
var quayHotel = new Hotel('Quay', 40, 25);  
var parkHotel = new Hotel('Park', 120, 77);  
ASSIGNMENT OPERATOR NEW KEYWORD VALUES USED IN PROPERTIES OF THIS OBJECT
```

Objets natifs

- Les navigateurs fournissent des **objets natifs** qui représentent des choses telles que la **fenêtre** du navigateur ou la **page web en cours** d'affichage dans cette fenêtre.
- Ce sont ces objets natifs qui permettent de créer des **pages web interactives**.
- On peut diviser ces objets natifs en **3 catégories** :
 - L'objet **window** — Représente le navigateur. Permet d'accéder à l'URL, l'historique de navigation...
 - L'objet **document** — Représente la page en cours d'affichage sous forme d'objets. Permet de manipuler la page programmatiquement.
 - Les **objets globaux JavaScript** — Utiles à la programmation JavaScript, par ex l'objet **Date**.

L'objet `window`

PROPERTY	DESCRIPTION
<code>window.innerHeight</code>	Height of window (excluding browser chrome/user interface) (in pixels)
<code>window.innerWidth</code>	Width of window (excluding browser chrome/user interface) (in pixels)
<code>window.pageXOffset</code>	Distance document has been scrolled horizontally (in pixels)
<code>window.pageYOffset</code>	Distance document has been scrolled vertically (in pixels)
<code>window.screenX</code>	X-coordinate of pointer, relative to top left corner of screen (in pixels)
<code>window.screenY</code>	Y-coordinate of pointer, relative to top left corner of screen (in pixels)
<code>window.location</code>	Current URL of <code>window</code> object (or local file path)
<code>window.document</code>	Reference to <code>document</code> object, which is used to represent the current page contained in <code>window</code>
<code>window.history</code>	Reference to <code>history</code> object for browser window or tab, which contains details of the pages that have been viewed in that window or tab
<code>window.history.length</code>	Number of items in <code>history</code> object for browser window or tab
<code>window.screen</code>	Reference to <code>screen</code> object
<code>window.screen.width</code>	Accesses <code>screen</code> object and finds value of its <code>width</code> property (in pixels)
<code>window.screen.height</code>	Accesses <code>screen</code> object and finds value of its <code>height</code> property (in pixels)
METHOD	DESCRIPTION
<code>window.alert()</code>	Creates dialog box with message (user must click OK button to close it)
<code>window.open()</code>	Opens new browser window with URL specified as parameter (if browser has pop-up blocking software installed, this method may not work)
<code>window.print()</code>	Tells browser that user wants to print contents of current page (acts like user has clicked a print option in the browser's user interface)

- **IMPORTANT.** Comme cet objet est celui de plus haut niveau, c'est aussi l'objet par défaut. Il est donc fréquent de l'omettre dans les instructions.

L'objet document

PROPERTY	DESCRIPTION
<code>document.title</code>	Title of current document
<code>document.lastModified</code>	Date on which document was last modified
<code>document.URL</code>	Returns string containing URL of current document
<code>document.domain</code>	Returns domain of current document

METHOD	DESCRIPTION
<code>document.write()</code>	Writes text to document (see restrictions on p226)
<code>document.getElementById()</code>	Returns element, if there is an element with the value of the <code>id</code> attribute that matches (full description see p195)
<code>document.querySelectorAll()</code>	Returns list of elements that match a CSS selector, which is specified as a parameter (see p202)
<code>document.createElement()</code>	Creates new element (see p222)
<code>document.createTextNode()</code>	Creates new text node (see p222)

Les objets globaux

- L'objet **String**

```
var saying = "L'argent ne fait pas le bonheur.";  
saying.length  
saying.toUpperCase()
```

- Toute valeur qui est une chaîne peut utiliser les propriétés et les méthodes de l'objet String. Exemples : **length**, **toUpperCase()**.

- L'objet **Number**

- Toute valeur qui est un nombre peut utiliser les propriétés et les méthodes de l'objet Number. Exemples : **isNaN()**, **toFixed()**.

- L'objet **Math** : propriétés et méthodes pour tâches mathématiques.

- L'objet **Date**. ATTENTION. Contrairement aux autres objets, il faut créer une instance de l'objet Date pour p



En résumé

- Les fonctions permettent de **regrouper une série de *statements* liés** qui représentent une **tâche unique**.
- Les fonctions peuvent **prendre des paramètres** (des informations requises pour effectuer leur travail) et peuvent **renvoyer une valeur**.
- Un objet est une série de **variables** et de **fonctions** qui représente quelque chose du monde environnant.
- Dans un objet, les variables sont appelées **propriétés** de l'objet, et les fonctions sont appelées **méthodes** de l'objet.
- Les navigateurs implémentent des objets qui représentent aussi bien la **fenêtre** du navigateur que le document **chargé** dans cette fenêtre.
- Le langage JavaScript fournit également plusieurs **objets natifs** tels que String, Number, Math et Date. Leurs propriétés et méthodes offrent des fonctionnalités utiles lors de la création de scripts.

EXERCICE: Utiliser les fonctions et les objets

- Partir de `js_func_obj/index.html`. (**NB. Cette fois-ci, le fichier est déjà chargé :-)**).
- Créez un objet `hotel` en utilisant la syntaxe **objet littéral**, avec ces caractéristiques :
 - **Propriétés** : `name`, `roomRate` (prix de la chambre), `discount` (% réduction).
 - **Méthode** : `offerPrice()` (renvoie le prix au tarif réduit)
- **Afficher les détails de l'hôtel** aux endroits appropriés dans la page.
- Créez une fonction `offerExpires()` qui renvoie le **jour où la promo expire**. Pour cet exercice, vous renverrez aléatoirement un jour compris entre “lundi” et “dimanche”. Afficher l’expiration dans la page.

20. Décision et boucles

Introduction

- Le **flot d'exécution du code** peut emprunter **plusieurs chemins**, ce qui signifie que le navigateur exécute différentes parties du code dans différentes situations.
- Dans cette leçon, nous allons apprendre à créer et contrôler le flot d'exécution du code en fonction des situations à gérer. Pour cela, nous utiliserons :
 - **Des évaluations.** Analyser des valeurs, les comparer à des valeurs attendues.
 - **Des décisions.** Choisir tel ou tel chemin d'exécution en fonction du résultat d'une évaluation.
 - **Des boucles.** Pour exécuter les mêmes étapes plusieurs fois successivement.

Décisions

- Il y a **2 composants** dans une décision :
 - 1. Une **expression** est évaluée et retourne une valeur.
 - 2. Un **statement conditionnel** dit le code à exécuter dans telle ou telle situation.

```
CONDITION
if [score > 50] {
    document.write('You passed!');
} else {
    document.write('Try again...');

}
```

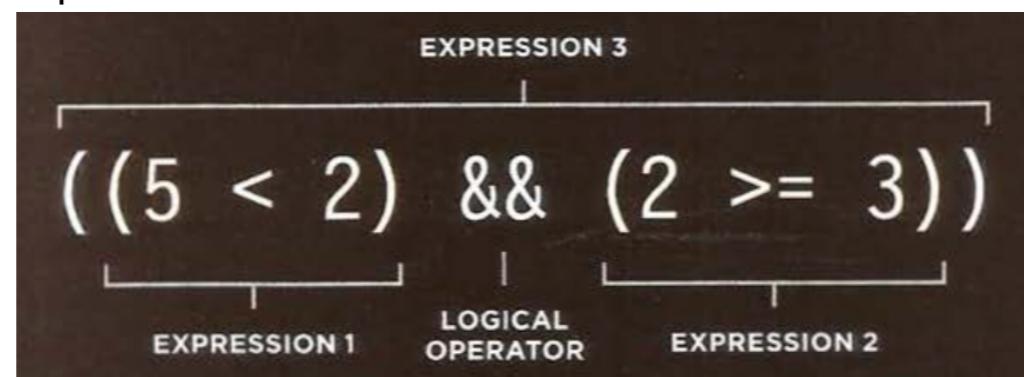
- **if...else** — Si la condition évaluée par le statement if évalue à **true**, les statements du 1er bloc de code sont exécutés. Sinon, c'est le 2nd bloc de code qui est exécuté. Notons que la partie **else** est optionnelle.

Opérateurs de comparaison

- On peut évaluer une situation en comparant une valeur du script à la valeur attendue grâce aux **opérateurs de comparaison**. Le résultat de l'évaluation sera un **booléen** : vrai ou faux.
 - Est égal à (**attention**) : `==`
 - N'est pas égal à : `!=`
 - Égalité stricte (valeur + type) : `====`
 - Inégalité stricte (valeur + type) : `!==`
 - Supérieur : `>`
 - Inférieur : `<`
 - Supérieur ou égal : `>=`
 - Inférieur ou égal : `<=`

Opérateurs logiques

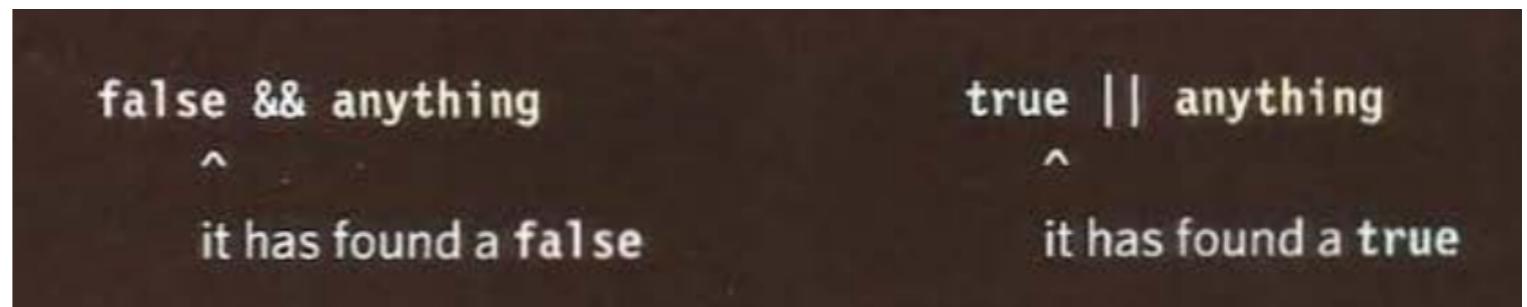
- Les opérateurs de comparaison renvoient généralement des valeurs uniques, **true** ou **false**.
- Les opérateurs logiques permettent de **combiner les résultats de plusieurs comparaisons**. Exemple :



- Opérateurs :



- Évaluations court-circuit (1) :



Statements **switch**

```
switch (level) {  
    case 'One':  
        title = 'Level 1';  
        break;  
  
    case 'Two':  
        title = 'Level 2';  
        break;  
  
    case 'Three':  
        title = 'Level 3';  
        break;  
  
    default:  
        title = 'Test';  
        break;  
}
```

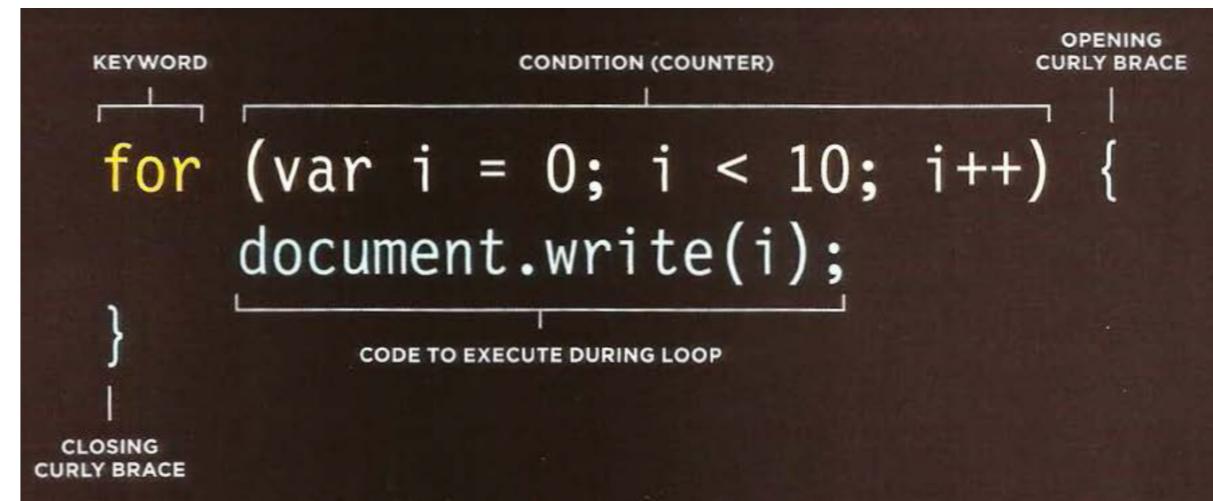
- Un statement **switch** part d'une variable dont la **valeur est évaluée**.
- Chaque **cas** représente une **valeur possible** de cette variable, et contient le code à exécuter si la variable correspond à cette valeur.
- Il existe aussi un **cas par défaut**, au cas où aucune valeur n'est matchée.

Boucles

- **Les boucles vérifient une condition.** Si la condition est **true**, un bloc de code est exécuté. La condition est à nouveau vérifiée, si elle renvoie encore **true**, le bloc de code est à nouveau exécuté. Le processus se répète jusqu'à ce que la condition soit **false**.

- **Boucle for**

(utile pour exécuter un bloc un nombre précis de fois) :



- **Boucle while**

(utile quand on ne sait pas à l'avance combien de fois exécuter le bloc) :

```
var i = 1;          // Set counter to 1
var msg = '';       // Message

// Store 5 times table in a variable
while (i < 10) {
    msg += i + ' x 5 = ' + (i * 5) + '<br />';
    i++;
}
```

En résumé

- Les **statements conditionnels** permettent à votre code de décider **quelle est la prochaine étape**.
- Les **opérateurs de comparaison** sont utilisés pour comparer deux opérandes (`==`, `!=`, `==`, `!=`, `<`, `>`, `<=`, `>=`).
- Les **opérateurs logiques** permettent de combiner plusieurs comparaisons.
- Les **statements `if... else`** permettent d'exécuter un ensemble de code si une condition est vraie, et un autre ensemble si elle est fausse.
- Les **statements `switch`** permettent de comparer une valeur avec les différents scénarios possibles (et aussi de fournir un scénario par défaut si aucun ne correspond).
- Il y a **3 types de boucles** : `for`, `while`, et `do...while`. Chacune répète un ensemble de statements.

21. Document Object Model (DOM)

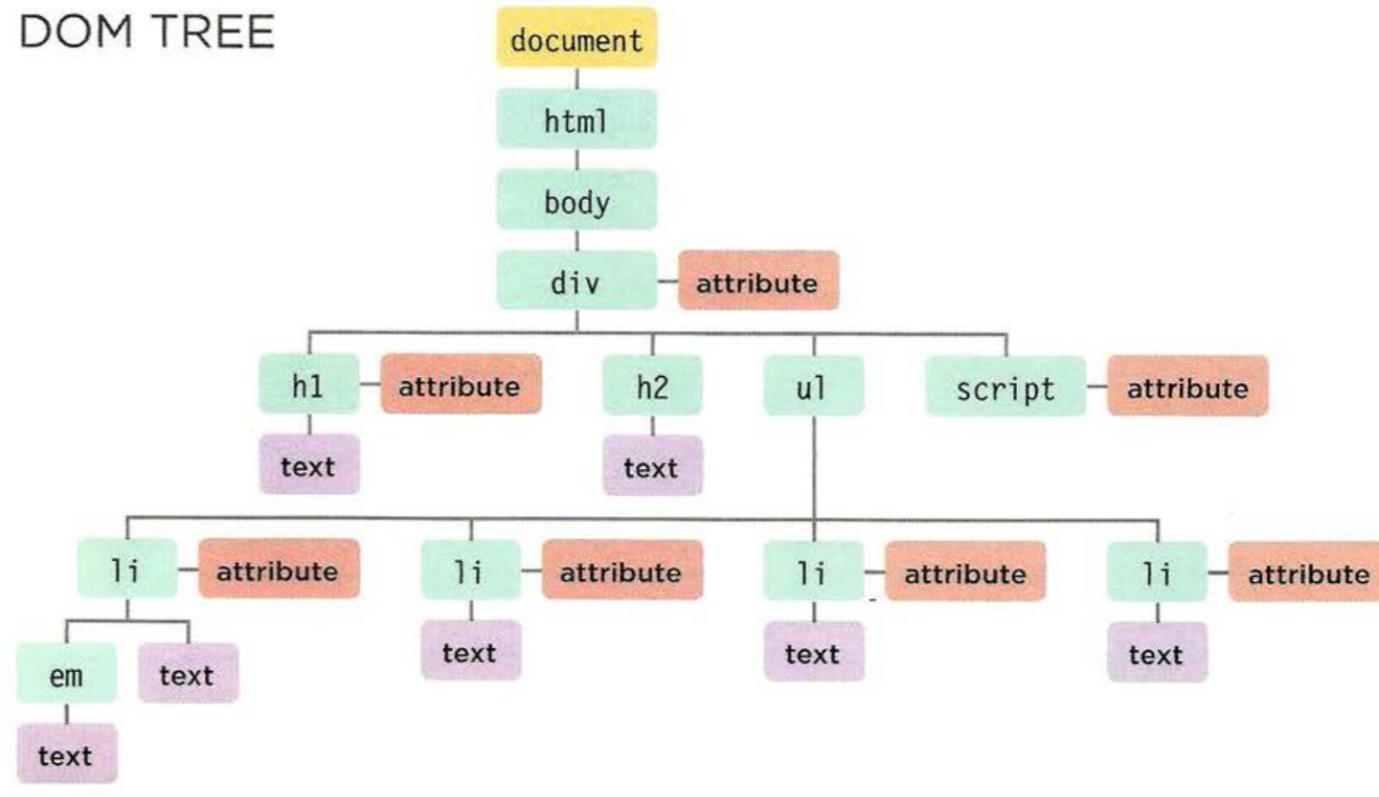
Introduction

- Le **Document Object Model** (DOM) est la représentation "programmatique" que le navigateur se fait de toute page web qu'il affiche.
- Le DOM permet à JavaScript d'**accéder** et de **modifier** le contenu d'une page quand elle est chargée dans le navigateur.
- Le DOM est une des **fondations** de la programmation web avec JavaScript.

L'arbre du DOM

- L'arbre du DOM représente la **page web en cours** sous forme sous la forme d'une hiérarchie de **noeuds**.

```
<html>
  <body>
    <div id="page">
      <h1 id="header">List</h1>
      <h2>Buy groceries</h2>
      <ul>
        <li id="one" class="hot"><em>fresh</em> figs</li>
        <li id="two" class="hot">pine nuts</li>
        <li id="three" class="hot">honey</li>
        <li id="four">balsamic vinegar</li>
      </ul>
      <script src="js/list.js"></script>
    </div>
  </body>
</html>
```



- Légende : **jaune** (noeud document), **vert** (noeuds élément), **rouge** (noeuds attribut), **violet** (noeuds texte).

Travailler avec l'arbre du DOM

- **Accéder et modifier l'arbre du DOM** se fait en 2 étapes :
 1. **Localiser le noeud** qui représente l'élément avec lequel vous voulez travailler.
 2. **Utiliser** son contenu texte, ses éléments enfant, ou ses attributs.
- **IMPORTANT.** L'arbre du DOM est visible dans la **console développeur** de votre navigateur. Cette console montre l'arbre DOM "live", avec toutes les modifications que JavaScript y a apporté (par opposition au code source de la page, qui ne change pas).

Étape 1 :

Accéder aux éléments

- Liste des propriétés et méthodes permettent d'accéder aux éléments du DOM.
- A. Sélectionner un **noeud individuel** (**méthodes**)

```
getElementById('id')
querySelector('css selector') // Retourne uniquement
                                // le 1er élément qui matche
```

- B. Sélectionner **plusieurs éléments** ou *nodelists* (**méthodes**)

```
getElementsByClassName('class')
getElementsByTagName('tagName')
querySelectorAll('css selector')
```

- C. Traverser le DOM d'un node à l'autre (**propriétés**)

```
parentNode
previousSibling / nextSibling
firstChild / lastChild
```

Étape 2 : Travailler avec les éléments

- Liste des propriétés et méthodes permettent d'accéder à et de modifier les éléments du DOM.
- **A.** Accéder à ou modifier les **noeuds texte**
 - Propriété `nodeValue` — **ATTENTION. Passer par le noeud enfant !!**
- **B.** Travailler avec le **contenu HTML**
 - Accéder/Modifier les éléments enfant + le contenu texte : `innerHTML` (propriété)
 - Accéder/Modifier le texte uniquement : `textContent` (propriété)
 - Plusieurs méthodes permettent de **créer ou supprimer des noeuds** : `createElement()`, `createTextNode()`, `appendChild()`, `removeChild()`
- **C.** Accéder à ou modifier les **attributs**
 - Propriétés pour lire/écrire : `className`, `id`
 - Méthodes : `hasAttribute()`, `getAttribute()`, `setAttribute()`, `removeAttribute()`

Exemples de syntaxe



```
var elements = document.getElementsByClassName('hot')
if (elements.length >= 1) {
    var firstItem = elements.item(0);
}
```

```
var elements = document.getElementsByClassName('hot');
if (elements.length >= 1) {
    var firstItem = elements[0];
}
```

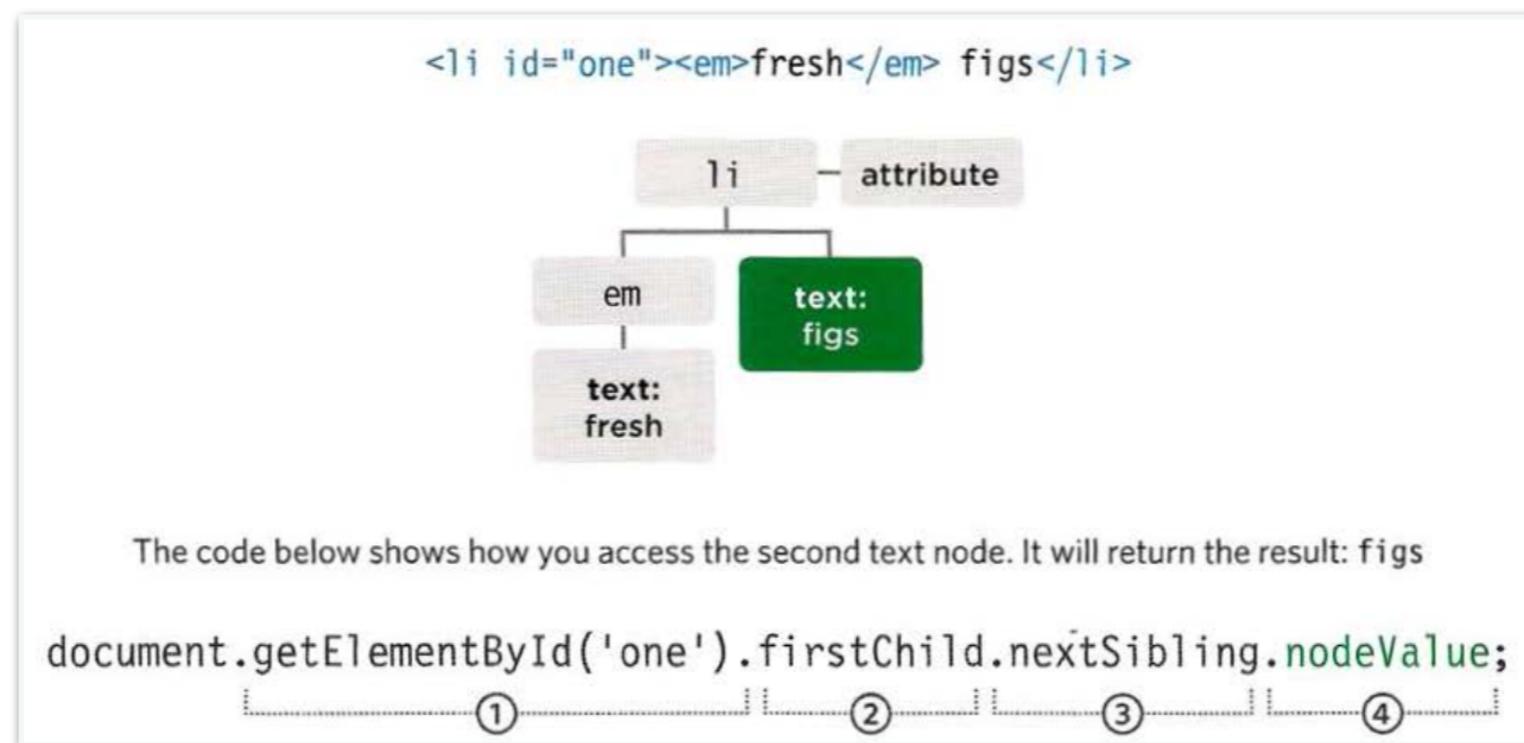
```
var hotItems = document.querySelectorAll('li.hot');
for (var i = 0; i < hotItems.length; i++) {
    hotItems[i].className = 'cool';
}
```

- **getElementById()**
Ce code renvoie le noeud élément dont l'attribut **id** a une valeur de "one".
- **getElementsByClassName()**
Pour sélectionner un élément dans une NodeList, on utilise soit la méthode **item()** soit la **syntaxe tableau**. Les deux méthodes nécessitent de connaître l'**indice** de l'élément souhaité.
- **Répéter une action pour toute une NodeList.**
On peut utiliser une boucle **for** avec la propriété **length** de la NodeList

Accéder et modifier un noeud texte avec **nodeValue**

- Quand on sélectionne un noeud texte, on peut lire ou modifier son contenu grâce à la propriété **nodeValue**.

- Syntaxe :



- Exemple :

```
var itemTwo = document.getElementById('two');  
  
var elText = itemTwo.firstChild.nodeValue;  
  
elText = elText.replace('pine nuts', 'kale');  
  
itemTwo.firstChild.nodeValue = elText;
```

Accéder et modifier le texte + markup avec `innerHTML`

- HTML :

```
<li id="one"><em>fresh</em> figs</li>
```

- JavaScript :

```
// Store the first list item in a variable
var firstItem = document.getElementById('one');

// Get the content of the first list item
var itemContent = firstItem.innerHTML;

// Update the content of the first list item so it is a link
firstItem.innerHTML = '<a href=\"http://example.org\">' + itemContent + '</a>';
```

Ajouter des éléments via les manipulations DOM

```
// Create a new element and store it in a variable.  
var newEl = document.createElement('li');  
  
// Create a text node and store it in a variable.  
var newText = document.createTextNode('quinoa');  
  
// Attach the new text node to the new element.  
newEl.appendChild(newText);  
  
// Find the position where the new element should be added.  
var position = document.getElementsByTagName('ul')[0];  
  
// Insert the new element into its position.  
position.appendChild(newEl);
```

- Les manipulations DOM offrent une autre technique pour ajouter du contenu à une page (à la place de `innerHTML`). Cela se fait en 3 étapes :

1. **Créer un nouvel élément - `createElement()`**
L'élément créé est stocké dans une variable.
2. **Donner du contenu à cet élément - `createTextNode()`**
Cette étape est facultative si l'élément n'a pas de contenu texte.
3. **Ajouter le nouvel élément au DOM - `appendChild()`**
Permet de préciser à quel élément du DOM le nouvel élément doit être rattaché (comme enfant).

Retirer des éléments via les manipulations DOM

1. Stocker l'**élément à supprimer** dans une variable.
2. Stocker le **parent** de cet élément dans une variable.
3. **Supprimer l'élément** de son conteneur parent.

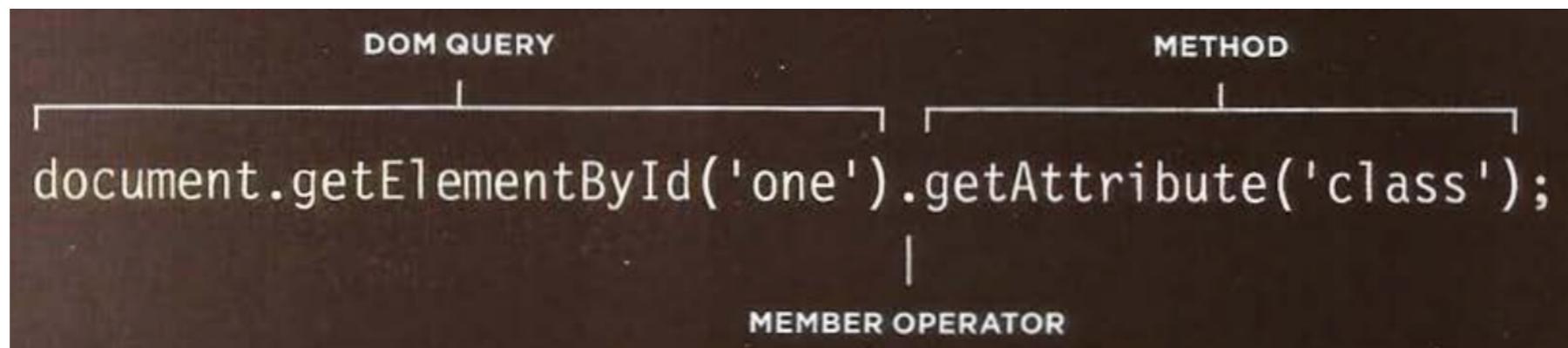
```
var removeEl = document.getElementsByTagName('li')[3]; // The element to remove  
  
var containerEl = removeEl.parentNode; // Its containing element  
  
containerEl.removeChild(removeEl); // Removing the element
```

Comparaison des techniques pour modifier le HTML

- `document.write()`
 - Simple et facile, mais fonctionne uniquement au chargement initial de la page.
- `element.innerHTML()`
 - Peut ajouter/modifier beaucoup de HTML d'un coup, le HTML est facile à générer. MAIS : risque de sécurité (si le HTML n'est inséré n'est pas *safe*) et les gestionnaires d'événement risquent de ne pas bien fonctionner.
- **Manipulation DOM**
 - Plus rigoureux (le HTML est toujours bien formé), les événements fonctionnent, incrémental. MAIS : Plus lent et plus verbeux que `innerHTML`.

Noeuds attribut

- Une fois qu'on a un noeud élément sous la main, on peut utiliser des propriétés et des méthodes pour **lire/écrire ses attributs** :



- Exemple de code pour modifier l'attribut **class** :

```
var firstItem = document.getElementById('one'); // Get the first item
firstItem.className = 'complete';           // Change its class attribute

var fourthItem = document.getElementsByTagName('li').item(3); // Get fourth item
el2.setAttribute('class', 'cool');           // Add an attribute to it
```

En résumé

- Le navigateur représente la page sous forme d'un **arbre DOM**.
- Les arbres DOM contiennent **4 types de noeuds** : noeuds document, noeuds élément, noeuds attribut et noeuds texte.
- On peut **sélectionner les noeuds élément** par leur attribut `id` ou `class`, par leur nom de **balise HTML**, ou grâce à la syntaxe **sélecteur CSS**.
- Lorsqu'une requête DOM retourne plus d'un noeud, elle retourne une **NodeList**.
- Depuis un noeud élément, on peut accéder et modifier son contenu grâce à des propriétés comme `textContent` et `innerHTML` ou grâce à des techniques de **manipulation du DOM**.
- Un noeud élément peut contenir plusieurs **noeuds texte** et des **éléments enfant** qui sont frères et soeurs.

22. Événements

Introduction

- Quand on surfe sur un site web, le navigateur déclenche **différents types d'événements**. Votre script peut **réagir** à ces événements.
- Dans cette leçon, nous verrons :
 - Les **actions de l'utilisateur** déclenchent des événements.
 - Les événements déclenchent du **code**.
 - Le code change l'**interface** (UI) ou crée une **modification** visible par l'utilisateur.

Types d'événements

- Il y a **différents types d'événements**.
- Ces événements sont associés aux différents composants d'un site web et portent des **noms prédéfinis** :
 - Au **navigateur** : `load`, `scroll`, `resize`...
 - Au **clavier** : `keydown`, `keyup`...
 - A la **souris** : `click`, `mouseover`...
 - Aux **formulaires** : `input`, `change`, `submit`...
- Tous peuvent **déclencher du code JavaScript**.

Déclencher du code JavaScript sur un événement

- Pour "gérer un événement", il faut :
 1. **Sélectionner le noeud élément** à surveiller.
 2. **Indiquer l'événement** sur le noeud sélectionné auquel il faut réagir.
 3. **Écrire le code à exécuter** lorsque l'événement se produit.
- On parle souvent de ***binding*** pour décrire le fait d'associer du code JavaScript à un événement précis.

Gestionnaires d'événement

DOM - Syntaxe

- Tous les navigateurs modernes comprennent cette syntaxe, mais on ne peut attacher qu'**une seule fonction** à chaque gestionnaire d'événement :

The diagram shows the syntax `element.onevent = functionName;` with three horizontal brackets below it. The first bracket covers "element", the second covers ".onevent", and the third covers "functionName". Below these labels are their definitions: "ELEMENT" is a "DOM element node to target", "EVENT" is an "Event bound to node(s) preceded by word 'on'", and "CODE" is "Name of function to call (with no parentheses following it)".

element.*onevent* = *functionName*;

ELEMENT EVENT CODE

DOM element node to target Event bound to node(s) preceded by word "on" Name of function to call (with no parentheses following it)

- Exemple :

```
function checkUsername() {  
    // code to check the length of username  
}  
var el = document.getElementById('username');  
el.onblur = checkUsername;
```

Écouteurs d'événements (*event listeners*)

- Les *event listeners* sont une approche plus récente. Ils acceptent **plus d'une fonction par événement**, mais ils ne sont pas supportés par les vieux navigateurs :

Adds an event listener to the DOM element node(s)

METHOD

`element.addEventListener('event', functionName [, Boolean]);`

ELEMENT EVENT CODE EVENT FLOW

DOM element node to target Event to bind node(s) to in quote marks Name of function to call Indicates something called capture, and is usually set to false (see p260)

- Exemple :

```
function checkUsername() {  
    // code to check the length of username  
}  
var el = document.getElementById('username');  
el.addEventListener('blur', checkUsername, false);
```

En résumé

- Les événements sont une manière pour le navigateur d'indiquer que **quelque chose s'est produit** (par exemple : une page a fini de charger ou un bouton a été cliqué).
- Le ***binding*** est la technique consistant à déclarer quel événement on attend, et pour quel élément on attend que cet événement se produise.
- Lorsqu'un événement se produit sur un élément, il peut **déclencher une fonction JavaScript**. Quand cette fonction modifie la page web d'une façon ou d'une autre, cela rend la page interactive car elle réagit aux actions de l'utilisateur.

EXERCICE : Événements et manipulations DOM

- Partir de `exos_starters/js_formulaire` (c'est une copie de l'exercice formulaire réalisé précédemment).
- Apportez les modifications suivantes en JavaScript :
 - Quand l'utilisateur **quitte le champ e-mail**, validez si la valeur saisie est un e-mail correct. Si incorrect, coloriez le champ E-mail en rouge (et vice versa).
 - Quand l'utilisateur coche la case “Recevoir la newsletter”, **faites apparaître un champ supplémentaire** “A quelle fréquence ?”. Ce champ doit disparaître dès que la case n'est plus cochée.

23. JavaScript pour HTML5

APi JS supplémentaires

- Les navigateurs récents qui implémentent la norme HTML5 exposent des APIs JavaScript supplémentaires :
 - **Stockage local de données** (`localStorage` et `sessionStorage`). Idéal pour enregistrer l'état de l'application ou les préférences de l'utilisateur.
 - **Applications hors-ligne**. Consiste à mettre en cache une copie des fichiers de l'application dans le navigateur de l'utilisateur grâce à un `manifeste`.
 - **Géolocalisation** (`Geolocation`). Permet d'obtenir programmatiquement la position géographique de vos utilisateurs.
 - Et bien d'autres.

Stockage local de données

- **Deux stockages possibles** (1) : `localStorage` pour stocker les données pérennemment et pour tout un site web ; `sessionStorage` pour stocker les données temporairement pour une fenêtre ou un onglet précis.
- **Stocker** des données :
`localStorage.setItem(keyName, data);`
- **Récupérer** les données stockées :
`localStorage.getItem(keyName);`
- Exemple :
`localStorage.setItem("user_name", "Vince");
alert("Prénom : " + localStorage.getItem("user_name"));`

EXERCICE: Utiliser localStorage

- Partir de `exos_starters/js_localstorage`.
- Faire en sorte qu'il soit possible d'enregistrer son prénom après l'avoir saisi en cliquant sur le bouton “Enregistrer”.
- Si l'utilisateur revient sur le site ou rafraîchit la page après avoir enregistré son prénom, celui-ci doit être affiché au-dessus du formulaire.
- **ASTUCE.** Comment examiner le contenu de `localStorage`? Console Développeur > Ressources > Local Storage.

Conclusion

Pour terminer...

- Dans cette formation, vous avez appris les **fondations du développement web**.
- Aller plus loin :
 - Maîtriser un **framework CSS** comme Bootstrap.
 - Maîtriser un **framework JavaScript** comme Angular.
- Qu'avez-vous pensé de la formation ? (contenu, rythme, niveau...)

Merci !