



# Formation HTML5, CSS3 JavaScript



# HTML5 - CSS3

# Introduction

Dans cette partie :

- HTML, XHTML et CSS
- Les navigateurs Internet
- Les éditeurs XHTML/CSS

# HTML, XHTML et CSS

HTML est le langage de base permettant de définir des pages Web. Il est composé de balises (aussi appelées marqueurs ou tags) qui décrivent et mettent en forme des contenus.

XHTML est un langage assez proche d'HTML, à quelques nuances près : toutes les balises doivent avoir un parent, et le parent de tous les parents est <html>. Voici la structure minimale d'une page XHTML :

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

## Quelques restrictions :

- pas de majuscules dans les noms et attributs des balises ;
- balises de fin obligatoires (les balises HTML non terminées prennent un slash de fin, comme par exemple <br> ou <img> ;
- les balises doivent être correctement imbriquées. A titre d'exemple, seule la seconde ligne de code est correcte en XHTML :

```
<b><i>Ce texte est gras et italique</b></i>
```

```
<b><i>Ce texte est gras et italique</i></b>
```

Le langage HTML ou XHTML est utilisé pour définir le contenu des pages Web.

Le langage CSS est utilisé pour définir la mise en forme et la mise en page des contenus HTML. Il peut être utilisé directement dans les balises HTML/XHTML, dans l'en-tête des pages ou dans un fichier externe appelé "feuille de styles".



# Editeurs HTML/XHTML/CSS

La saisie du code HTML peut se faire dans un simple éditeur de texte, comme le Bloc-Notes de Windows. Mais vous lui préférerez un éditeur de code à coloration syntaxique.

Plusieurs éditeurs de ce type sont diffusés sous la forme de freewares. Vous utiliserez par exemple NotePad ++, Sublime Text 2 ou VS code.

# Pourquoi utiliser HTML5 et CSS3 ?

Le World Wide Web Consortium (W3C) est un organisme de standardisation chargé de promouvoir la compatibilité des technologies du Web, telles que HTML, XHTML, CSS, PNG, SVG, SOAP, etc. La dernière spécification du langage HTML 4.01 date de 1999.

Il était temps qu'une nouvelle version du langage voie le jour, car le Web a bien évolué depuis presque 15 ans !

Vous en saurez plus sur les spécifications de ce tout nouveau langage en accédant aux pages [www.w3.org/TR/html5/](http://www.w3.org/TR/html5/) et [www.w3.org/TR/html-markup/](http://www.w3.org/TR/html-markup/).

# Pourquoi HTML5 et CSS3 ?

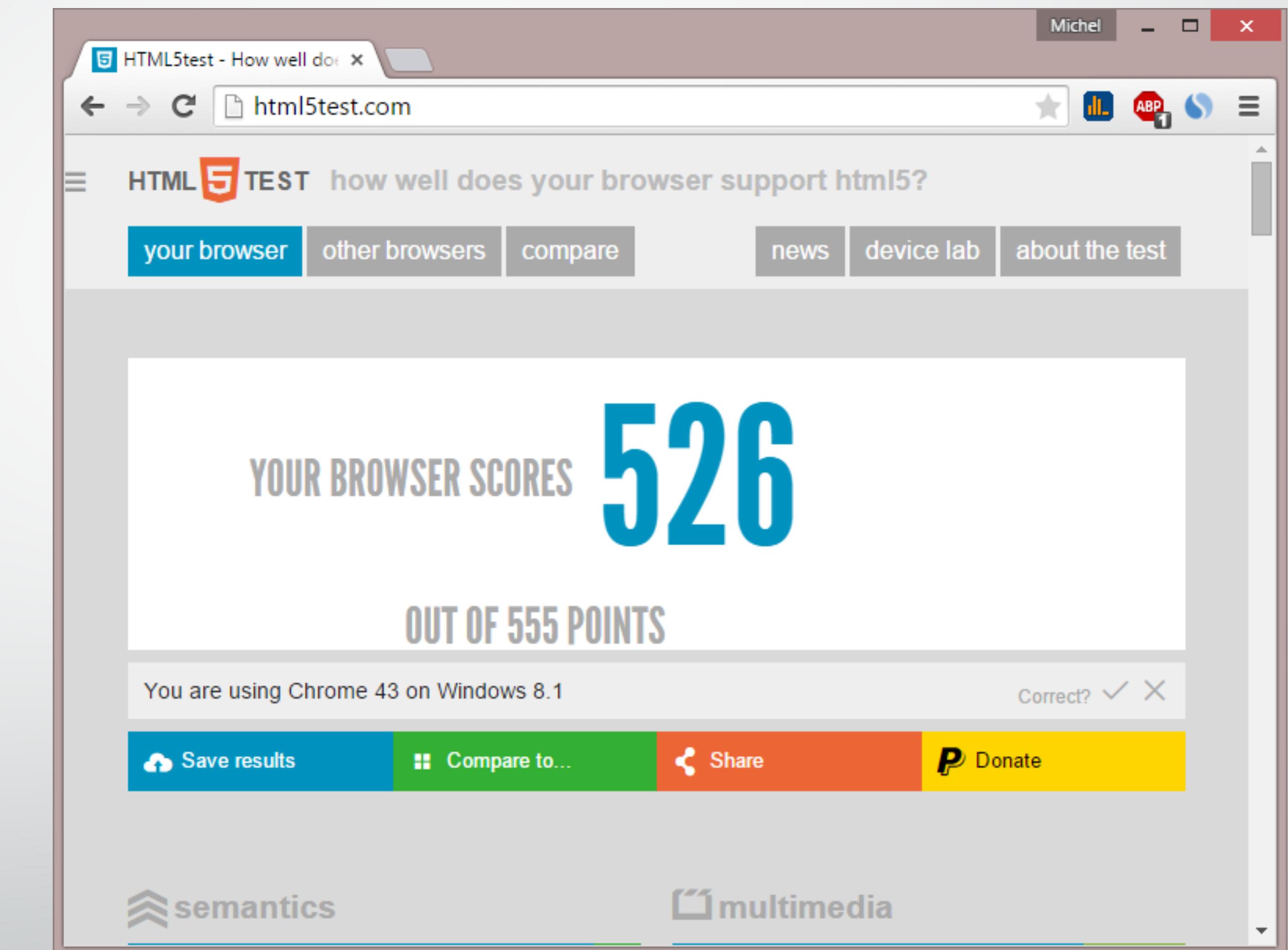
HTML est en mesure d'afficher des éléments textuels, graphiques et vidéos dans un navigateur web et d'assurer leur mise en page. Cette seconde fonction est cependant assez limitée. C'est la raison d'être du langage CSS, qui se charge de la mise en forme et de la mise en page des éléments HTML. La version 3 du langage apporte son lot de nouveautés, toutes plus impressionnantes les unes que les autres.

La standardisation de ce langage est assurée par la section CSS3 Working Group du W3C. Pour suivre l'état d'avancement des différents modules qui la composent, rendez-vous à [www.css3.info/modules/](http://www.css3.info/modules/).

# Compatibilité avec les navigateurs

Les nouveaux éléments propres au langage HTML5 ne sont pas encore entièrement implémentés sur les principaux navigateurs.

Pour connaître le taux de compatibilité de votre navigateur, rendez-vous à la page <http://html5test.com/>.



Testez la compatibilité de vos navigateurs

# Compatibilité avec les navigateurs

Les principaux navigateurs utilisent un préfixe pour (pré)implémenter les nouvelles propriétés CSS3 :

- -moz pour les navigateurs Mozilla (Firefox) ;
- -webkit pour les navigateurs Webkit (Safari, OmniWeb, Midori, etc.) ;
- -khtml pour les navigateurs Konqueror ;
- -o pour les navigateurs Opera ;
- -ms pour le navigateur Internet Explorer 9 et supérieur.

Tant que la spécification du langage n'a pas atteint au moins le statut de *recommandation candidate*, vous devrez utiliser plusieurs préfixes dans les propriétés CSS3 pour assurer la plus grande compatibilité possible. À terme, ces préfixes ne devraient plus avoir cours et une seule instruction devrait être interprétée à l'identique dans tous les navigateurs.

# Différences HTML5/prédécesseurs

HTML5 est rétrocompatible

La déclaration de type de document (DTD) est simplifiée à son extrême : <!DOCTYPE html>

Le jeu de balises propre à HTML5 introduit une nouvelle logique de formulation, plus sémantique et plus intuitive : la balisage ne décrit plus le contenu du document. Il détaille sa structure et désigne le rôle de chaque section.

Enfin, les développeurs web peuvent désormais utiliser des API JavaScript gérées nativement par les navigateurs. Le but est d'obtenir des niveaux de performances comparables à ceux des applications Desktop, mais aussi de tirer parti des possibilités offertes par les périphériques mobiles, qui vont prendre une part de plus en plus importante dans le paysage Internet.

# Les bases du HTML5/CSS3

Dans cette partie :

- Déclaration de type de document (DTD)
- Syntaxes HTML et XHTML
- Jeux de caractères
- Définir la langue dans un document HTML5
- Structure d'un document HTML5
- Valider du code HTML5
- Styles CSS dans les balises
- Feuille de styles interne
- Feuille de styles externe
- Éléments, sélecteurs, propriétés et valeurs CSS
- Sélecteurs CSS 2.1
- Sélecteurs CSS3
- id ou class ?
- Pseudo-classes et pseudo-éléments
- Unités CSS

# Déclaration de type de document

La balise <!DOCTYPE> est utile pour que les navigateurs adaptent le rendu d'un document HTML.

La balise <!DOCTYPE> des documents écrits en HTML5 est réduite à sa plus simple expression :

```
<!DOCTYPE html>
```

En HTML4.01, les trois variantes généralement utilisées étaient les suivantes :

- HTML 4.01 strict (mode rigoureux) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

- HTML 4.01 transitionnel (langage XHTML utilisé):

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

- HTML 4.01 frameset (la page contient des frames) :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

# DTD

Voici le squelette d'un document écrit en HTML5 :

```
<!DOCTYPE html>

<html>

  <head>

    </head>

  <body>

    </body>

  </html>
```

# Jeu de caractères

Les *charsets* (jeux de caractères) sont associés à la notion de claviers nationaux. Pour indiquer aux navigateurs dans quel jeu de caractères vous travaillez, vous pouvez insérer une balise dans l'en-tête de votre document :

<meta charset="jeu-à-utiliser">

Vous utiliserez le jeu de caractères :

- **ISO-8859-1** pour accéder directement à la majorité des caractères des langues occidentales (français, anglais, allemand, espagnol, etc.).
- **utf-8** pour afficher sur une même page des caractères issus de plusieurs langues (français et japonais par exemple). Consultez la page [www.columbia.edu/kermit/utf8.html](http://www.columbia.edu/kermit/utf8.html) pour vous rendre compte des immenses possibilités du jeu de caractères utf-8.

# Squelette standard HTML5

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta ...>
    ...
    <meta ...>
    <title>...</title>
    <link ...>
    <style>...</style>
  </head>
  <body>
    </body>
</html>
```

Les éléments meta, s'ils sont présents, permettent, entre autres, d'associer un jeu de caractères, une description et un nom d'auteur au document. Par exemple :

```
<meta charset="utf-8">
<meta name="description" content="Description de la page en quelques phrases.">
<meta name="author" content="Samuel Michaux">
```

link lie le document courant à un autre document (HTML, CSS, image, XML). Par exemple, pour définir une feuille de styles externe :

```
<link rel="stylesheet" href="style.css">
```

# Valider le code HTML/XHTML

Avant de tester un code HTML/XHTML dans votre navigateur,  
je vous conseille de vous assurer de sa conformité. Pour cela,  
rendez-vous à la page <http://validator.w3.org/>.

Pour faire apparaître du texte dans une page Web, il suffit de le taper tel qu'il doit apparaître, entre les balises <body> et </body> :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    ...
  </head>
  <body>
    Ce texte apparaît dans le navigateur.
  </body>
</html>
```

Le texte est affiché tel qu'il est saisi dans le navigateur. Si nécessaire, vous pouvez effectuer un passage à la ligne avec la balise `<br>` :

```
Texte sur la ligne 1 <br>
Texte sur la ligne 2
```

Vous pouvez aussi définir des paragraphes avec la balise `<p> </p>` :

```
<p>Ceci est un paragraphe</p>
<p>Ceci est un autre paragraphe</p>
```

Quelle est la différence entre `<br>` et `<p>` ?

# ***Des commentaires dans le texte***

Si vous concevez des pages longues et/ou complexes, vous voudrez peut-être y insérer des commentaires pour faciliter leur maintenance. Pour cela, vous utiliserez le marqueur <!-- :

```
<!-- Commentaires -->
```

Le texte placé dans le marqueur n'est pas affiché dans le navigateur : il est uniquement visible dans le code source.

# ***Titres et sous-titres***

Vous pouvez affecter jusqu'à six niveaux de titre à vos documents HTML. Pour cela, vous utiliserez les marqueurs <hx> et </hx>, où x est un chiffre compris entre 1 et 6.

## **Exemple :**

Ce petit exemple montre comment utiliser les marqueurs <hx>. Remarquez qu'un marqueur de fin de titre (</hx>) provoque un passage à la ligne. Il est donc inutile de faire appel aux marqueurs <br> ou <p>.

```
<html>
<head>
<title>Les balises de titre</title>
</head>
<body>
<h1>titre de niveau 1</h1>
<h2>titre de niveau 2</h2>
<h3>titre de niveau 3</h3>
<h4>titre de niveau 4</h4>
<h5>titre de niveau 5</h5>
<h6>titre de niveau 6</h6>
texte normal
</body>
</html>
```

# ***Liens hypertexte***

Une grande part de la magie inhérente au Web est liée à la présence de liens hypertexte dans les pages HTML : en cliquant sur certains mots ou sur certaines images, on peut avoir accès à d'autres parties du document en cours de consultation, à d'autres pages du même site ou d'un autre site, ou encore à d'autres services Internet (mail, ftp, gopher, telnet).

Avant de passer à la pratique, vous devez savoir que l'intérêt d'un site tient à la qualité et à la quantité de ses liens.

Au niveau de la qualité, il est nécessaire de placer les liens à des endroits logiques et de bien spécifier vers quoi ils pointent. N'oubliez pas que les informations qui se trouvent dans vos pages sont consultées on-line. La rapidité d'accès aux données recherchées est donc essentielle.

Au niveau quantité, veillez à ne pas submerger l'utilisateur sous une profusion de liens. Un document aéré est toujours plus facile à lire. Sauf cas particuliers (tableau, sommaire, etc.), limitez-vous à quelques liens par page et révisez-les fréquemment pour que votre site soit toujours proche de l'actualité (c'est un des intérêts majeurs d'Internet).

# **Lien dans la même page**

Si une de vos pages HTML vous semble un peu longue, vous pouvez définir quelques liens pour en faciliter la lecture.

Pour chaque lien, vous procéderez en deux étapes :

1. Définition d'un signet pour marquer la partie à accéder.
2. Définition d'un lien hypertexte vers ce signet.

Pour placer un signet dans le document courant, vous insérerez un id dans une balise quelconque. Par exemple, une balise <p></p> :

```
<p id="Signet">Texte</p>
```

où Signet est le nom du signet et Texte le texte vers lequel vous désirez pointer.

Pour créer un lien hypertexte vers le signet que vous venez de définir, vous utiliserez une autre variante de la balise <a> :

```
<a href="#Signet">  
    Texte  
</a>
```

où Signet est le nom de la balise à pointer et Texte le texte qui servira de lien.

## Exercice

Définissez un document qui contient plusieurs paragraphes de texte. Ce texte pourra être pris sur le site <http://fr.lipsum.com/>.

Placez ce texte dans des paragraphes. Définissez des signets pour chacun d'entre eux et des liens hypertextes pour accéder aux différents paragraphes.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Liens et signets</title>
  </head>
  <body>
    <a href="#par1">Paragraphe 1</a> <a href="#par2">Paragraphe 2</a> <a href="#par3">Paragraphe 3</a> <a href="#par4">Paragraphe 4</a> <a href="#par5">Paragraphe 5</a>
    <p id="par1">Paragraphe 1 : Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    <p id="par2">Paragraphe 2 : Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    <p id="par3">Paragraphe 3 : Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    <p id="par4">Paragraphe 4 : Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    <p id="par5">Paragraphe 5 : Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>

  </body>
</html>
```

## ***Lien vers un document local***

Comme nous venons de le voir, les signets facilitent l'accès à un passage déterminé dans un même document. Lorsqu'un lien doit donner accès à des informations qui ne se trouvent pas dans le document courant mais dans un autre document du site, les paramètres de la balise <a> doivent être légèrement différents :

```
<a href="fichier">  
    Texte  
</a>
```

où Fichier est le nom du fichier HTML auquel accéder et Texte le texte qui servira de lien.

## ***Lien vers une partie d'un document local***

Vous savez maintenant définir un lien hypertexte vers un signet du même document et vers un autre document HTML. En combinant les deux syntaxes, il est possible d'accéder à un signet d'un autre document :

```
<a href="fichier#signet">  
    Texte  
</a>
```

où Fichier est le nom du fichier HTML auquel accéder, Signet le nom du signet dans le fichier HTML, et Texte le texte qui servira de lien.

# **Lien vers un document distant**

Jusqu'à présent, vous avez appris à utiliser un lien hypertexte pour pointer vers un passage du même document, un autre document du site ou un passage d'un autre document du site. En utilisant une syntaxe légèrement différente, il est possible d'accéder à un document d'un autre site :

```
<a href="https://adresse_url">  
    Texte  
</a>
```

où [http://Adresse\\_URL](http://Adresse_URL) est l'adresse URL de la page HTML et Texte le texte qui servira de lien.

Un URL (*Uniform Ressource Locator*) est une adresse permettant d'identifier de manière unique un objet Internet. Sa syntaxe "façon UNIX" détermine le nom du serveur, le chemin d'accès dans le serveur et le nom du fichier HTML à accéder : **http://serveur/chemin/document#signet**

## ***Lien vers un fichier***

Pour donner accès à un fichier, il suffit de préciser son nom dans le chemin URL. Par exemple, pour accéder au document ebook.pdf dans le dossier documents du serveur [www.monsite.com](http://www.monsite.com), vous utiliserez ce lien :

```
<a href="http://www.monsite.com/documents/ebook.pdf">Cliquez ici</a>
```

# **Accéder à un service Internet**

Dans les lignes précédentes, vous avez appris à accéder à une page Web située sur un site distant. Comme le montre ce tableau, il suffit de modifier la première partie de l'adresse pour accéder tout aussi simplement à un autre service Internet.

<b>Service</b>	<b>Marqueur</b>
Site ftp	<a href = "ftp://adresse ">Texte</a>
Serveur e-mail	<a href = "mailto:adresse ">Texte</a>
Gopher	<a href = "gopher://adresse ">Texte</a>
Telnet	<a href = "telnet://adresse ">Texte</a>

Dans chacun des marqueurs précédents, adresse désigne l'adresse URL du service auquel accéder et texte le texte utilisé pour réaliser le lien.

Pour renseigner le visiteur de votre site, vous pouvez ajouter des informations concernant une image ou un lien via une infobulle.

Voici la syntaxe à utiliser pour un lien :

```
<a href="adresse" title="Texte de l'infobulle">Texte du lien</a>
```

Essayez ce code

# Tableaux

Certaines pages Web contiennent des tableaux. Ceux-ci ont été réalisés à l'aide de balises `<table>` `</table>`.

La syntaxe du marqueur `<table>` est assez complexe :

```
<table  
    [border = "bordure"]  
    [cellpadding = "espace"]  
    [cellspacing = "epaisseur"]  
    [width = "largeur"]  
    [height = "hauteur"]>  
</table>
```

Bordure est un entier qui définit l'épaisseur de la bordure autour du tableau. Si BORDER n'apparaît pas dans le marqueur ou si Bordure vaut 0, le tableau n'a pas de bordure.

Espace définit l'espace entre le contenu des cellules et la bordure gauche de la cellule, en pixels.

Epaisseur définit l'épaisseur du trait entre les cellules, en pixels.

Largeur force la largeur du tableau, en pixels. Si ce paramètre n'est pas spécifié, la largeur du tableau est calculée automatiquement.

Hauteur force la hauteur du tableau, en pixels. Si ce paramètre n'est pas spécifié, la hauteur du tableau est calculée automatiquement.

Vous utiliserez le marqueur <tr> </tr> pour créer chacune des lignes du tableau :

```
<tr  
    [align = alignement]>  
</tr>
```

où alignement définit l'alignement des données sur la ligne. Il peut prendre la valeur left, center ou right selon l'effet recherché. Si ce paramètre n'est pas spécifié, les données sont alignées à gauche.

Enfin, pour définir les caractéristiques d'une cellule, vous utiliserez le marqueur **<td> </td>** dont voici la syntaxe :

```
<td  
    [colspan = "nbcol"]  
    [rowspan = "nblig"]  
    [align = "alignhoriz"]  
    [valign = "alignvert"]  
    [width = "largeur"]>  
    élément  
</td>
```

**NbCol** est le nombre de colonnes occupées par la cellule.

**NbLig** est le nombre de lignes occupées par la cellule.

**AlignHoriz** indique l'alignement horizontal du texte dans la cellule. Ce paramètre peut prendre la valeur **LEFT**, **CENTER** ou **RIGHT**. S'il n'est pas spécifié, les données sont alignées à gauche.

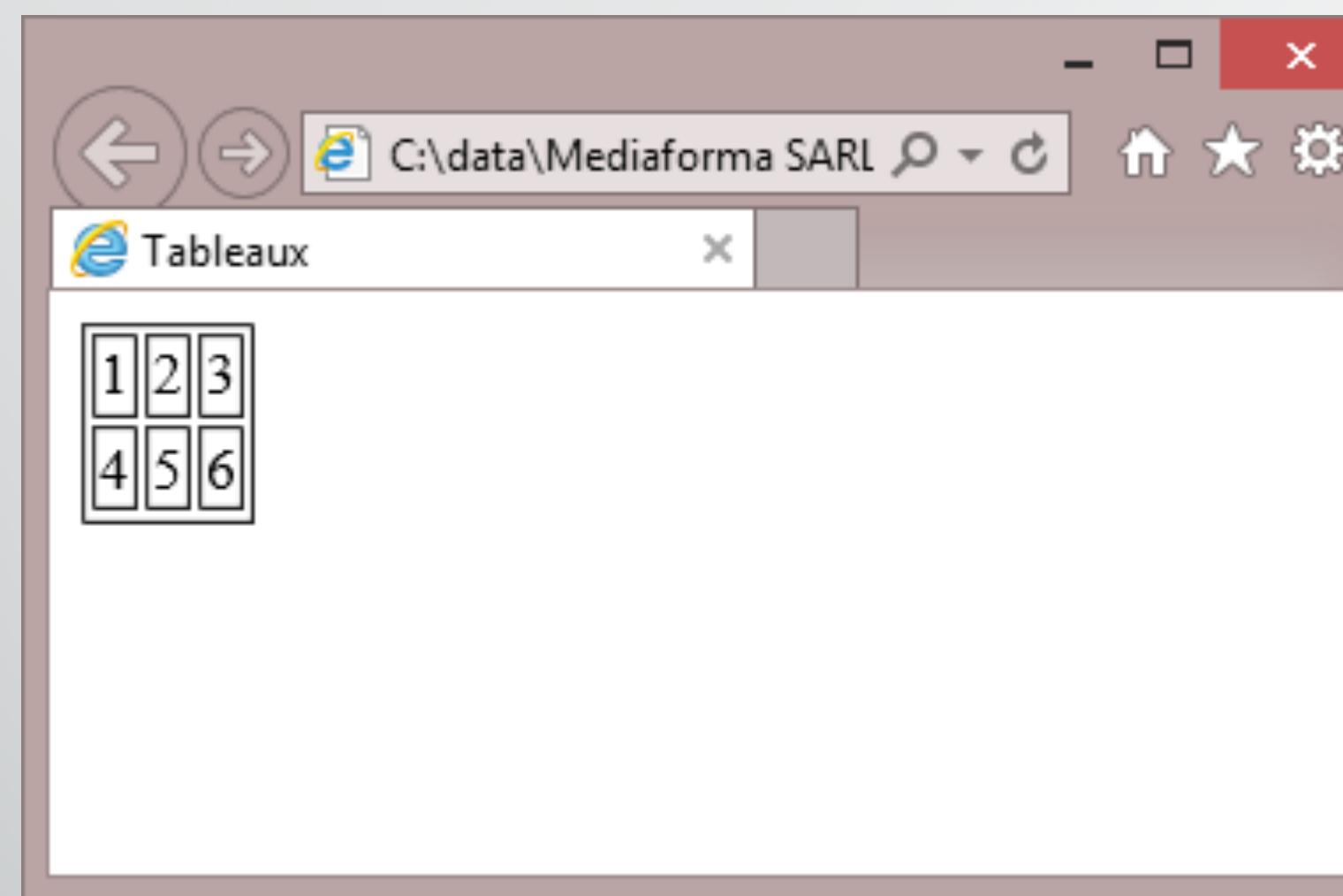
**AlignVert** indique l'alignement vertical du texte dans la cellule. Ce paramètre peut prendre la valeur **TOP**, **MIDDLE** ou **BOTTOM**. S'il n'est pas spécifié, les données sont alignées sur la partie supérieure de la cellule.

**Largeur** indique la largeur de la cellule, en pixels.

**Elément** décrit le contenu de la cellule. Il peut s'agir d'un texte ou d'une image (marqueur **IMG**).

## Exercice 1

Définissez le tableau suivant :



A screenshot of a web browser window titled "Tableaux". The address bar shows "C:\data\Mediaforma SARL". The main content area displays a 2x3 grid of numbers:

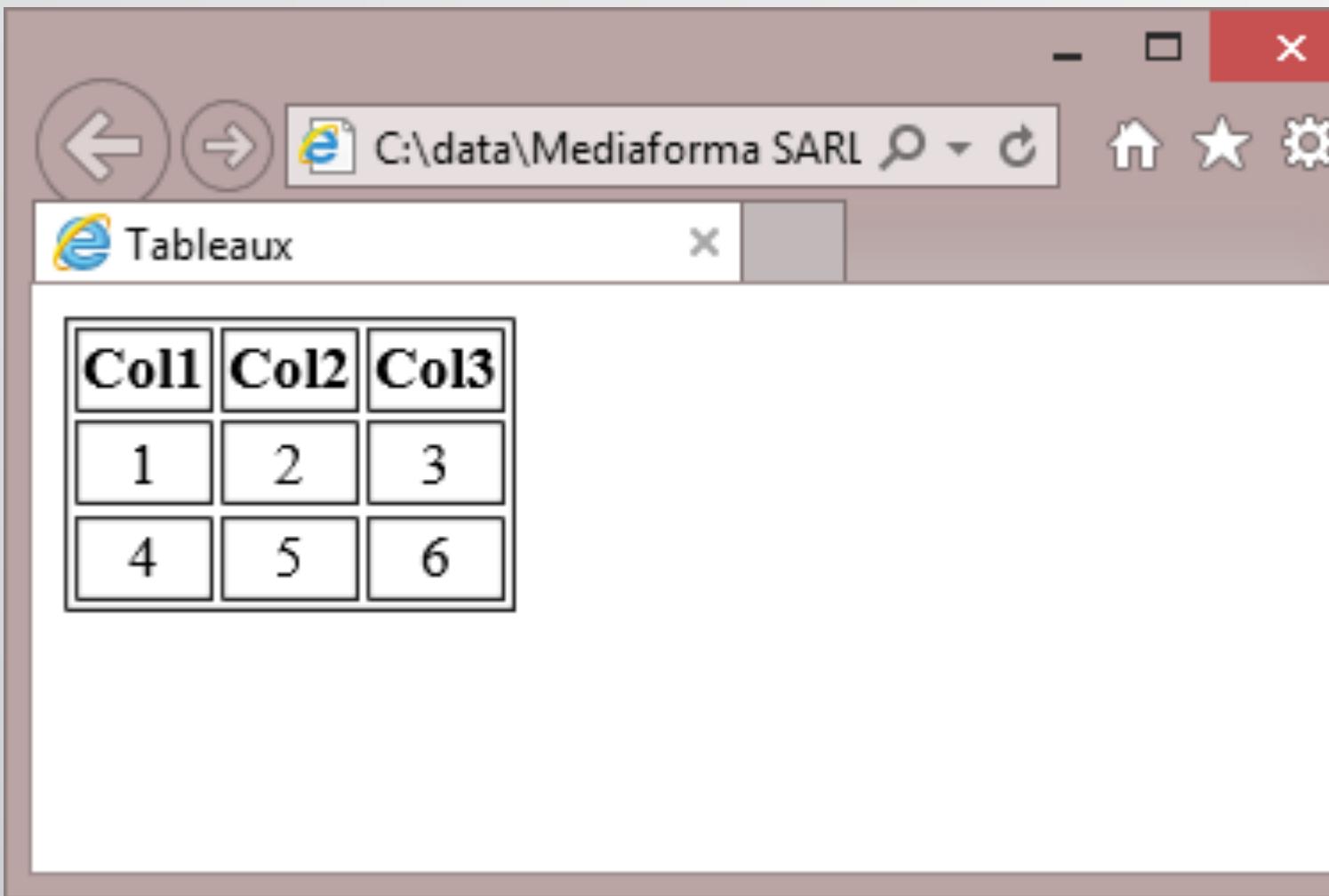
1	2	3
4	5	6

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Tableaux</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
      </tr>
      <tr>
        <td>4</td>
        <td>5</td>
        <td>6</td>
      </tr>
    </table>
  </body>
</html>
```

## Exercice 2

Définissez le tableau suivant :



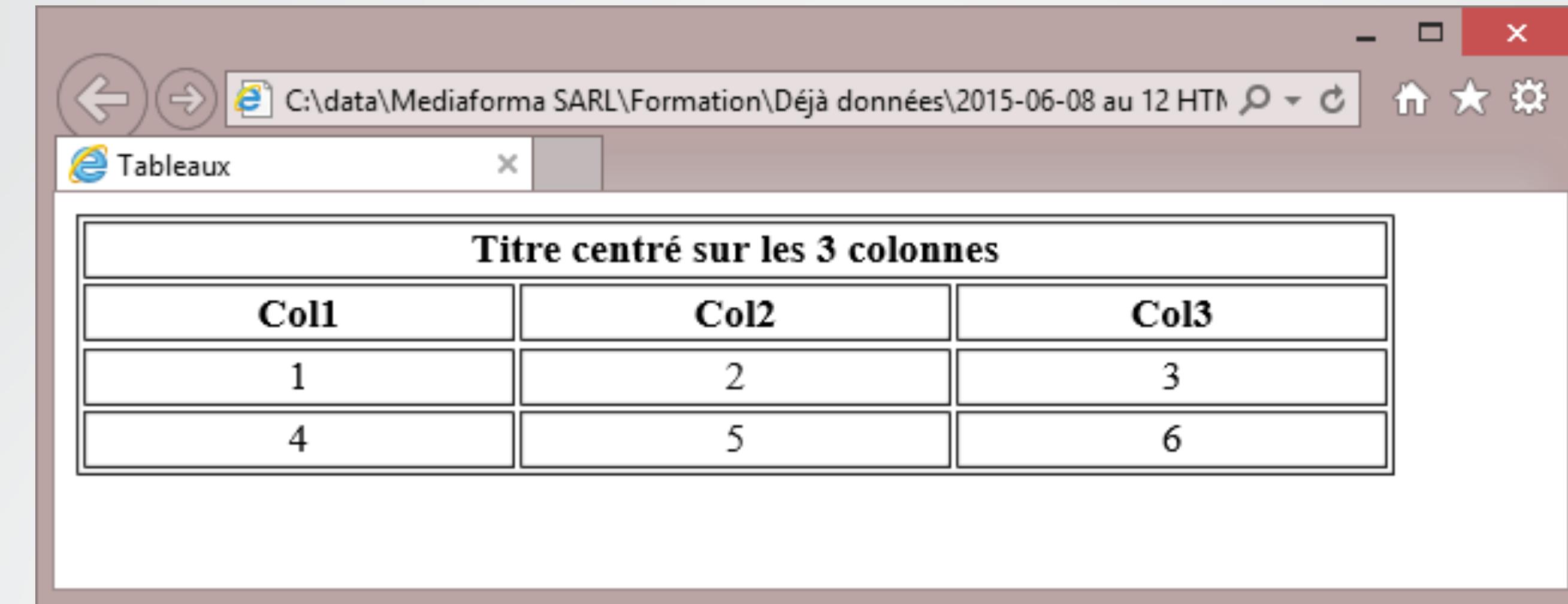
The screenshot shows a web browser window with the title "Tableaux". The address bar displays "C:\data\Mediaforma SARL". The main content area contains a 3x3 grid of cells. The first row is labeled "Col1", "Col2", and "Col3". The second row contains the numbers 1, 2, and 3. The third row contains the numbers 4, 5, and 6.

Col1	Col2	Col3
1	2	3
4	5	6

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Tableaux</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <th>Col1</th>
        <th>Col2</th>
        <th>Col3</th>
      </tr>
      <tr>
        <td align="center">1</td>
        <td align="center">2</td>
        <td align="center">3</td>
      </tr>
      <tr>
        <td align="center">4</td>
        <td align="center">5</td>
        <td align="center">6</td>
      </tr>
    </table>
  </body>
</html>
```

## Exercice 3

Définissez le tableau suivant :



The screenshot shows a web browser window with a title bar reading "C:\data\Mediaforma SARL\Formation\Déjà données\2015-06-08 au 12 HTM". The main content area displays a table with a header row containing three columns labeled "Col1", "Col2", and "Col3". Below this is a data row with three cells containing the numbers 1, 2, and 3 respectively. A second data row follows with cells containing 4, 5, and 6. The title "Titre centré sur les 3 colonnes" is centered above the first row.

Col1	Col2	Col3
1	2	3
4	5	6

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Tableaux</title>
  </head>
  <body>
    <table border="1">
      <tr><th width="500px" colspan="3" align="center">Titre centré sur les 3 colonnes</th></tr>
      <tr><th>Col1</th><th>Col2</th><th>Col3</th></tr>
      <tr><td align="center">1</td><td align="center">2</td><td align="center">3</td></tr>
      <tr><td align="center">4</td><td align="center">5</td><td align="center">6</td></tr>
    </table>
  </body>
</html>
```

Exercice : Définissez le tableau suivant

 Titres dans un tableau X

**Recettes budgétaires par région**

	<b>En millions d'Euros</b>		<b>2015 en €/hab.</b>
	<b>2014</b>	<b>2015</b>	
Ile-de-france	399258	415920	178,8
Midi-pyrénées	73961	63826	219,4
Centre	43638	55802	185,9

```
<table border>
  <caption>Recettes bugétaires par Région</caption>
  <thead>
    <tr>
      <th></th>
      <th colspan="2">En millions d'Euros</th>
      <th rowspan="2">2015 en €/hab.</th>
    </tr>
    <tr>
      <th></th>
      <th>2014</th>
      <th>2015</th>
    </tr>
  </thead>
  <tbody>
    <tr align="center">
      <td>Ile-de-france</td>
      <td>399258</td>
      <td>415920</td>
      <td>178,8</td>
    </tr>
    <tr align="center">
      <td>Midi-pyrénées</td>
      <td>73961</td>
      <td>63826</td>
      <td>219,4</td>
    </tr>
    <tr align="center">
      <td>Centre</td>
      <td>43638</td>
      <td>55802</td>
      <td>185,9</td>
    </tr>
  </tbody>
</table>
```

# ***Insertion d'une image dans un document HTML***

L'insertion d'une image dans un document HTML s'effectue en trois étapes :

1. Définition ou choix de l'image.
2. Sauvegarde au format GIF (.GIF), JPEG (.JPG) ou PNG (.PNG) et stockage dans le même dossier que le fichier HTML.
3. Insertion d'une balise `<img>` dans le document HTML.

## Avantages et inconvénients de ces trois types d'images

GIF : limité à 256 couleurs, peut posséder une couleur de transparence, non compressé

JPG : 16 millions de couleurs, pas de couleur de transparence, compressé

PNG : 16 millions de couleurs, peut posséder une couleur de transparence, non compressé

Téléchargez l'application PhotoFiltre 7 sur  
<http://www.photofiltre-studio.com/news.htm>

Entraînez-vous à créer des images au format GIF, JPG (avec plusieurs formats de compression) et PNG et comparez leurs tailles en octets

Lorsque l'image a été sauvegardée au bon format, insérez un marqueur <IMG> dans la page HTML, à l'endroit où vous désirez l'afficher. La syntaxe du marqueur est la suivante :

```
<img  
    src = "nom de l'image"  
    [align = {top|bottom|middle|left|right} ]  
    [border = epaisseur]  
    [alt = "texte"]>
```

1. Nom de l'image est le nom complet de l'image (nom du fichier et extension).
2. ALIGN définit l'alignement de l'image par rapport au texte : TOP, BOTTOM et MIDDLE alignent le texte respectivement par rapport à la partie supérieure, à la partie inférieure et au centre de l'image ; LEFT et RIGHT alignent l'image respectivement à gauche et à droite du texte.
3. Epaisseur définit l'épaisseur du cadre autour de l'image, en pixels.
4. Texte est un court texte qui apparaît à la place de l'image lorsque l'affichage des images a été désactivé.



Entraînez-vous à insérer les images GIF, JPG et PNG définies précédemment et comparez leur rendu.

## Rendre une image cliquable

Pour insérer un lien sur une image, il suffit de l'entourer d'une balise `<a></a>` :

```
<a href="http://www.site.com"></a>
```

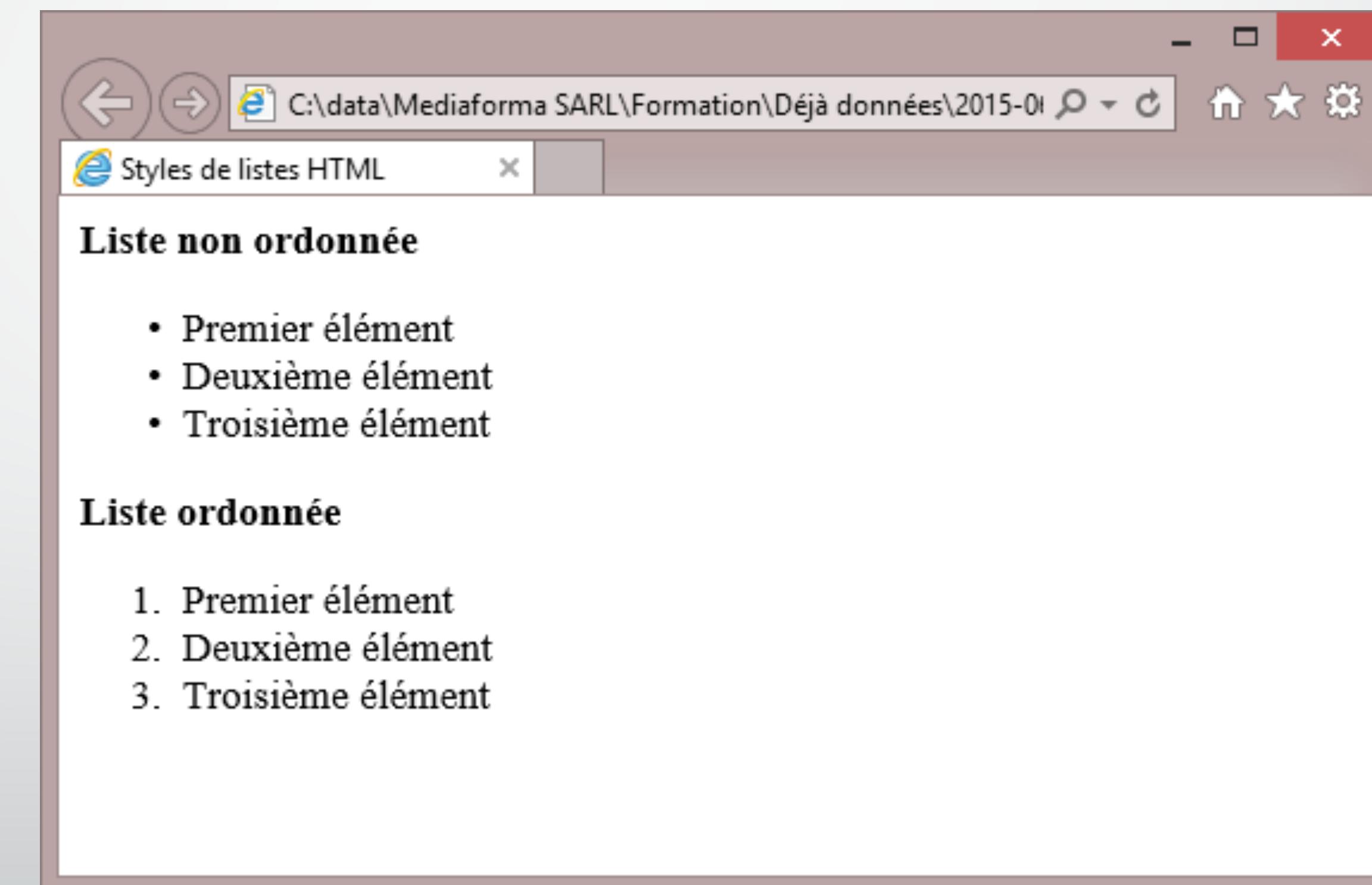
# Listes

Le langage HTML compte trois types de listes :

- non ordonnées ou listes à puces : éléments `ul` et `li` ;
- ordonnées : éléments `ol` et `li` ;

Cet exemple illustre ces deux types de listes :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Styles de listes HTML</title>
  </head>
  <body>
    <b>Liste non ordonnée</b>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ul>
    <b>Liste ordonnée</b>
    <ol>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ol>
  </body>
</html>
```



# Formulaires

Si la plupart des pages Web se contentent de diffuser des informations, elles sont également en mesure d'en recueillir par l'intermédiaire de formulaires. Les données sont stockées sur le serveur. Elles peuvent être traitées à l'aide d'un programme spécifique écrit par vos soins, fourni par le prestataire ou tout simplement expédiées sans traitement dans votre boîte aux lettres.

Pour définir un formulaire, vous utiliserez le marqueur <form> </form> dont voici la syntaxe :

```
<form  
    action = "traitement"  
    method = "{get|post}">  
</form>
```

1. traitement est le nom d'un programme de traitement, par exemple <http://www.site.com/traitement.php>, ou encore l'adresse de votre boîte aux lettres, précédée de mailto. Par exemple. <mailto:samuel.michaux@gmail.com>.
2. method définit la méthode à utiliser pour prendre en charge les données du formulaire. La méthode GET étant limitée quant à la quantité de données transmissibles, la méthode POST est presque toujours utilisée.

Entre les balises <form> et </form>, vous utiliserez des balises <input> pour définir les données à saisir :

```
<input  
  [type = {"text|password|checkbox|radio|submit|reset|hidden|image"}]  
  [name = "nom"]  
  [value = "valeur"]  
  [size = "taille1[,taille2]"]  
  [maxlength = "longueur"]  
  [checked]>
```

Le paramètre optionnel type définit le type de l'objet :

Valeur	Effet
<b>text</b>	Champ de saisie
<b>password</b>	Mot de passe
<b>checkbox</b>	Case à cocher
<b>radio</b>	Bouton radio
<b>submit</b>	Bouton d'envoi, pour envoyer les données du formulaire au serveur
<b>reset</b>	Bouton de réinitialisation du formulaire
<b>hidden</b>	Champ caché qui n'apparaît pas sur le formulaire mais qui est envoyé lors de l'appui sur le bouton SUBMIT ou sur une image cliquable

Exercice :

Définissez un formulaire pour obtenir le résultat suivant :

The screenshot shows a Windows application window titled "Un premier formulaire". The window has a standard title bar with icons for back, forward, search, and other system functions. The main content area contains the following elements:

- A label "Entrez votre nom" followed by a text input field.
- A label "Entrez votre adresse e-mail" followed by another text input field.
- A question "Etes-vous client de la fnac ?" with two radio button options: "OUI" (selected) and "NON".
- Two buttons at the bottom: "Envoyer" and "Annuler".

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <title>Un premier formulaire</title>
  </head>
  <body>
    <form action = "mailto:bertrand.oscar@oracle.fr" method = "post">
      Entrez votre nom <input type = "text" name = nom><br>
      Entrez votre adresse e-mail <input type = "text" name = nom><br>
      Etes-vous client de la fnac ?
      OUI <input type = "radio" name = "client" value = "oui" checked>
      NON <input type = "radio" name = "client" value = "non"><br>
      <input type = "submit" value = "Envoyer">
      <input type = "reset" value = "Annuler">
    </form>
  </body>
</html>
```

## Exercice

Utilisez un tableau pour améliorer le formulaire comme ceci :

The screenshot shows a Windows application window titled "Un premier formulaire". The window has a standard title bar with icons for back, forward, search, and settings. The main content area contains the following form elements:

- A label "Entrez votre nom" followed by an empty text input field.
- A label "Entrez votre adresse e-mail" followed by an empty text input field.
- A question "Etes-vous client de la fnac ?" with two radio button options: "OUI" (selected) and "NON".
- Two buttons at the bottom: "Envoyer" on the left and "Annuler" on the right.

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <title>Un premier formulaire</title>
  </head>
  <body>
    <form action = "mailto:bertrand.oscar@oracle.fr" method = "post">
      <table>
        <tr><td>Entrez votre nom</td><td><input type = "text" name="nom"></td></tr>
        <tr><td>Entrez votre adresse e-mail</td><td><input type = "text" name="email"></td></tr>
        <tr><td>Etes-vous client de la fnac ?</td>
          <td>OUI <input type = "radio" name = "client" value = "oui" checked>
          NON <input type = "radio" name = "client" value = "non"></td></tr>
        <tr><td><input type = "submit" value = "Envoyer"></td>
        <td><input type = "reset" value = "Annuler"></td></tr>
      </table>
    </form>
  </body>
</html>
```

## Une liste déroulante dans un formulaire

Pour définir une liste déroulante, vous utiliserez la balise <select>. Chacun des éléments de la liste sera placé dans une balise enfant <option>

```
<form>
  <select name="jour">
    <option>Lundi</option>
    <option>Mardi</option>
    <option>Mercredi</option>
    <option>Jeudi</option>
    <option>Vendredi</option>
  </select>
</form>
```



Dans la balise **<select>**, vous pouvez définir l'attribut **size** pour indiquer combien d'éléments sont visibles dans la liste (par défaut, **size** vaut **1**).

Dans l'une des balises **<option>** vous pouvez définir l'attribut **selected**, sans lui affecter aucune valeur. L'élément correspondant est alors sélectionné par défaut dans la liste.

Outre les boutons de formulaires **submit** et **reset** :

```
<input type="submit">  
<input type="reset">
```

Vous pouvez utiliser de simples boutons à l'intérieur ou à l'extérieur d'un formulaire avec la balise **<button>** :

```
<button>Texte</button>
```

# Découpage d'une page

Pour découper une page en blocs, vous utiliserez les balises `<div>` et `</div>`.

Ces balises sont de type **block**. Elles s'affichent automatiquement l'une sous l'autre, sans qu'il soit nécessaire d'utiliser une balise `<br>` pour passer à la ligne ou de les insérer dans une balise `<p> </p>`.

Les balises `<div>` peuvent être personnalisées en utilisant des instructions CSS. Nous étudierons ces instructions dans la suite de la formation.

# Mise en forme en CSS

Dans cette partie :

- Styles dans les balises, dans une feuille de styles interne ou externe
- Sélecteurs, propriétés et valeurs
- Sélecteurs CSS 2.1
- id ou class ?
- Sélecteurs CSS 3
- Pseudo-classes et pseudo-éléments
- Unités CSS



Vous utiliserez le langage CSS pour mettre en forme les balises d'un ou de plusieurs documents, à travers des attributs ou des feuilles de styles.

Les prochaines diapositives vont vous montrer comment utiliser ce langage.

# Styles CSS dans les balises

style=""

où :

- balise est un nom de balise : <p> ou <h1> par exemple.
- propriété1 sont des propriétés de style de la balise.
- valeur1 sont les valeurs affectées aux propriétés.

Cette technique n'est pas optimale, car limitée au seul élément qui l'utilise. Dans la mesure du possible, privilégiez l'utilisation d'une feuille de styles externe, d'extension .css et reliée au document à l'aide d'un élément link. Cette même feuille de styles pourra être utilisée dans tous les documents apparentés pour uniformiser leur apparence.

# Styles CSS dans les balises

À titre d'exemple, pour affecter la couleur jaune à l'arrière-plan d'un élément `p`, vous utiliserez le code suivant :

```
<p style="background: yellow;">  
Ce texte a un arrière-plan jaune  
</p>
```

Ou encore, pour utiliser la police Verdana corps 18 dans un titre `h2`, vous utiliserez le code suivant :

```
<h2 style="font-family: Verdana; font-size: 18px;">  
Ce titre est en Verdana corps 18  
</h2>
```

Essayez ces deux codes

# Feuille de styles interne

Pour étendre la portée des définitions CSS, vous pouvez les regrouper dans l'en-tête du document HTML.

Voici la syntaxe permettant d'affecter des propriétés à une balise :

élément {propriété1:valeur1; ... propriétéN:valeurN}

où :

- élément est un nom d'élément : p ou h1 par exemple.
- propriété1 sont des propriétés de style de l'élément.
- valeur1 sont les valeurs affectées aux propriétés.

Ces lignes de code doivent être insérées entre les balises <style> et </style>, à l'intérieur de l'élément head.

# Exercice

Ecrivez une feuille de styles interne pour :

- Définir dans tout le document un arrière-plan de couleur jaune pour les éléments `p`
- Utiliser la police Verdana corps 18 dans tous les titres `h2`

# Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une feuille de styles interne</title>
    <style>
      p {background-color:yellow; }
      h2 {font-family:Verdana; font-size: 18px; }
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

# Feuille de styles externe

Pour augmenter encore le champ d'action des styles définis dans l'en-tête d'un document, vous pouvez les stocker dans un fichier CSS. Dans ce cas, plusieurs documents HTML peuvent référencer cette "feuille de styles" (le fichier CSS) pour utiliser les styles qui y sont définis.

L'opération se fait avec un élément link (ici, la feuille de styles a pour nom *moncss.css*) :

```
<link rel="stylesheet" href="moncss.css">
```

# Exercice

Définissez une feuille de styles externe qui reprend les règles définies dans l'exercice précédent et faites référence à cette feuille de styles dans un document HTML.

# Solution

```
p {background-color:yellow;}  
h2 {font-family:Verdana; font-size: 18px;}
```

La feuille de styles  
moncss.css

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>Une feuille de styles externe</title>  
<link rel="stylesheet" href="moncss.css">  
</head>  
<body>  
<!-- Les instructions HTML5 peuvent utiliser les styles -->  
<!-- définis dans la feuille de styles moncss.css -->  
</body>  
</html>
```

Le document HTML

# Sélecteurs, propriétés et valeurs CSS

La syntaxe à utiliser est la suivante :

```
sélecteur {propriété1: valeur1; ... propriétéN: valeurN;}  
élément {propriété1: valeur1; ... propriétéN: valeurN;}
```

Pour définir le style d'un élément, il suffit d'entrer le nom de l'élément, d'ouvrir des accolades, d'énumérer des propriétés et de leur affecter les valeurs souhaitées.

# Exercice

Définissez un document HTML5 qui contient une liste à puces.

Définissez une feuille de styles interne qui donne les caractéristiques de la liste à puces :

- Couleur : blue
- Arrière-plan : #FF4040
- Marges : 14px
- Style des puces : square

# Solution

code001.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Style des li</title>
    <style>
      li
      {
        color: blue;
        background: #FF4040;
        margin: 14px 14px 14px 14px;
        list-style: square;
      }
    </style>
  </head>
  <body>
    Une liste à puces :
    <ul>
      <li>Premier</li>
      <li>Deuxième</li>
      <li>Troisième</li>
      <li>Quatrième</li>
    </ul>
  </body>
</html>
```

# Sélecteurs CSS 2.1

Le sélecteur universel étoile (`*`) s'adresse à tous les éléments. Vous l'utiliserez essentiellement pour modifier le style de tous les éléments de la page. Par exemple, la police.

```
* { font-family: Verdana; }
```

Le sélecteur de type permet de changer le style d'un élément. Ici, par exemple, le style de tous les éléments `h1` :

```
h1 { font-size: 120%; }
```

Le sélecteur de classe est représenté par un point (`.`). Il cible les éléments en fonction de la valeur de leur attribut `class`. Par exemple, le style ci-après s'applique à tous les éléments `p` dont l'attribut `class` vaut `info`.

```
p.info { background: yellow; }
```

Si aucun élément ne restreint le sélecteur de classe, tous les éléments dont l'attribut `class` a la valeur spécifiée sont concernés. Ici, par exemple, il s'agira de tous les éléments dont l'attribut `class` vaut `info` :

```
.info { background: yellow; }
```

# Sélecteurs CSS 2.1

Il est possible d'affecter plusieurs classes à un élément (l'attribut class peut recevoir plusieurs valeurs séparées par des espaces). On parle alors de "sélecteur de classes multiples". Ici par exemple, seuls les éléments p dont l'attribut class vaut info et premier sont affectés :

```
p.info.premier {background: yellow;}
```

Voici un exemple d'élément p concerné par cette règle CSS :

```
<p class="info premier">texte</p>
```

# Sélecteurs CSS 2.1

Le sélecteur d'identificateur est représenté par le caractère dièse (#). Il cible l'élément dont l'attribut id a la valeur spécifiée. Ici, seul l'élément dont la propriété id vaut corne est concerné :

```
#corne {font-family: Serif;}
```

Vous pouvez restreindre le ciblage en précisant un élément devant le sélecteur d'id. Ici, par exemple, seul l'élément p dont le marqueur id vaut corne est concerné :

```
p#corne {font-family: Serif;}
```

Un sélecteur descendant est construit avec deux sélecteurs traditionnels (ou plus) séparés par un espace. Il cible les enfants du premier sélecteur. Par exemple, pour affecter un arrière-plan jaune à tous les éléments li enfants d'un élément div, vous utiliserez le sélecteur descendant ci-après :

```
div li {color: yellow;}
```

# Sélecteurs CSS 2.1

Un sélecteur d'attribut cible les éléments en fonction de la présence d'attributs et/ou de leurs valeurs. Par exemple, pour affecter une taille de 120% à tous les éléments qui possèdent un attribut perso, vous utiliserez le sélecteur suivant :

```
[perso] {font-size: 120%;}
```

Pour cibler les seuls éléments dont l'attribut perso a pour valeur vert, vous utiliserez le sélecteur suivant :

```
[perso=vert] {font-size: 120%;}
```

# Sélecteurs CSS 2.1

Enfin, pour sélectionner les éléments **el2** immédiatement précédés par une élément **el1**, vous utiliserez le sélecteur suivant :

`el1+el2`

# Exercice

- 1) Affectez la couleur rouge aux balises p de classe rouge
- 2) Affectez la police Courier aux balises div de classe code et exemple
- 3) Affectez la couleur d'arrière-plan jaune aux balises <li> contenues dans une balise <ol>

Voici le code HTML sur lequel appliquer ces règles :

```
<body>
  <p class="rouge">Ceci est un paragraphe de classe rouge</p>
  <p class="vert">Ceci est un paragraphe de classe vert</p>
  <p class="noir">Ceci est un paragraphe de classe noir</p>
  <div class="code">Ce texte est affiché dans une balise div de classe code</div>
  <div>Ce texte est affiché dans une balise div sans classe</div>
  <div class="exemple">Ce texte est affiché dans une balise div de classe exemple</div>
  Une liste UL
  <ul>
    <li>Premier</li>
    <li>Deuxième</li>
    <li>Troisième</li>
  </ul>
  Une liste OL
  <ol>
    <li>Premier</li>
    <li>Deuxième</li>
    <li>Troisième</li>
  </ol>
</body>
```

# Exercice

Et voici le résultat à obtenir :

Ceci est un paragraphe de classe rouge

Ceci est un paragraphe de classe vert

Ceci est un paragraphe de classe noir

Ce texte est affiché dans une balise div de classe code

Ce texte est affiché dans une balise div sans classe

Ce texte est affiché dans une balise div de classe exemple

Une liste UL

- Premier
- Deuxième
- Troisième

Une liste OL

1. Premier
2. Deuxième
3. Troisième

# Solution

code002.htm

```
<style>
    p.rouge {color: red; }
    div.code {font-family:Courier New; }
    div.exemple {font-family:Courier New; }
    ol li {background: yellow; }
</style>
```

# id ou class ?

L'attribut id est généralement utilisé :

- Sur les éléments qui structurent le document et qui sont désignés comme uniques : par exemple, les éléments header, container, content, nav, footer, etc. Une fois ces éléments identifiés avec l'attribut id, ils peuvent être stylés avec des propriétés CSS.
- Sur d'autres éléments qui doivent être accessibles en JavaScript.
- Pour accéder à une ancre.

```
<header id="hperso">  
  
<nav id="nperso">  
  
<p id="position1">, puis <a href="#position1">Aller au repère 1</a>
```

L'attribut class est utilisé dans tous les autres cas.

```
<div class="perso">  
  
<p class="perso">
```

# Sélecteurs CSS3

Tout ce qui a été dit sur les sélecteurs à la section précédente reste valable en CSS3. Cependant, de nouveaux sélecteurs sont maintenant disponibles, qui permettent de cibler encore plus précisément les éléments auxquels vous vous adressez.

# Sélecteurs CSS3

Syntaxe	Signification
<u>nom_elément[attr^="valeur"]</u>	Tout élément <u>nom_elément</u> dont la valeur de l'attribut <u>attr</u> commence exactement par la chaîne <u>valeur</u>
<u>nom_elément[attr\$="valeur"]</u>	Tout élément <u>nom_elément</u> dont la valeur de l'attribut <u>attr</u> finit exactement par la chaîne <u>valeur</u>
<u>nom_elément[attr*="valeur"]</u>	Tout élément <u>nom_elément</u> dont la valeur de l'attribut <u>attr</u> contient la sous-chaîne <u>valeur</u>
<u>:root</u>	Racine du document
<u>nom_elément:nth-child(n)</u>	Un élément <u>nom_elément</u> qui est le n-ième enfant de son parent
<u>nom_elément:nth-last-child(n)</u>	Un élément <u>nom_elément</u> qui est le n-ième enfant de son parent en comptant depuis le dernier enfant
<u>nom_elément:nth-of-type(n)</u>	Un élément <u>nom_elément</u> qui est le n-ième enfant de son parent et de ce type
<u>nom_elément:nth-last-of-type(n)</u>	Un élément <u>nom_elément</u> qui est le n-ième enfant de son parent et de ce type en comptant depuis le dernier enfant
<u>nom_elément:last-child</u>	Un élément <u>nom_elément</u> , dernier enfant de son parent
<u>nom_elément:first-of-type</u>	Un élément <u>nom_elément</u> , premier enfant de son type

# Sélecteurs CSS3

Syntaxe	Signification
<code>nom élément:last-of-type</code>	Un élément <u>nom élément</u> , dernier enfant de son type
<code>nom élément:only-child</code>	Un élément <u>nom élément</u> , seul enfant de son parent
<code>nom élément:only-of-type</code>	Un élément <u>nom élément</u> , seul enfant de son type
<code>nom élément:empty</code>	Un élément <u>nom élément</u> qui n'a aucun enfant
<code>nom élément:target</code>	Un élément <u>nom élément</u> qui est la cible de l'URL d'origine
<code>nom élément:enabled</code>	Un élément d'interface utilisateur <u>nom élément</u> qui est actif ou inactif
<code>nom élément:checked</code>	Un élément d'interface utilisateur <u>nom élément</u> qui est coché ou dont l'état est indéterminé (bouton radio ou case à cocher par exemple)
<code>nom élément:contains("attr")</code>	Un élément <u>nom élément</u> dont le contenu textuel concaténé contient la sous-chaîne <u>attr</u>
<code>nom élément:selection</code>	La partie d'un élément <u>nom élément</u> qui est actuellement sélectionnée par l'utilisateur
<code>nom élément:not(sel)</code>	Un élément <u>nom élément</u> qui n'est pas représenté par le sélecteur simple <u>sel</u>
<code>nom élément ~ E</code>	Un élément <u>E</u> précédé par un élément <u>nom élément</u>

# Pseudo-classes, pseudo-éléments

Le sélecteur de pseudo-classe est représenté par le caractère deux-points (`:`).

Vous ferez appel aux pseudo-classes pour cibler des éléments en fonction de caractéristiques inaccessibles aux sélecteurs traditionnels : premier enfant ou focus par exemple.

La pseudo-classe `:first-child` permet de cibler le premier enfant d'un élément. Par exemple, pour mettre en gras le premier élément `p` enfant de l'élément `div`, vous utiliserez le sélecteur ci-après :

```
div p:first-child {font-weight: bold;}
```

Les pseudo-classes `:link` et `:visited` ciblent les éléments `a` dont (respectivement) le lien n'a pas été visité/a été visité. Les deux lignes suivantes définissent la couleur des liens :

```
:link {color: fuchsia;}
```

```
:visited {color: navy;}
```

# Pseudo-classes, pseudo-éléments

La pseudo-classe `:focus` cible les éléments qui ont le focus (par défaut, les liens et les éléments de formulaires). Elle permet très simplement de modifier la couleur d'arrière-plan (ou un autre style) d'un élément. Ici, par exemple, nous affectons un arrière-plan rouge à l'élément `input` de type `text` qui a le focus :

```
input[type=text]:focus {background: lightgrey;}
```

La pseudo-classe `:hover` cible les éléments dont le contenu est survolé. Cela permet par exemple de changer la bordure d'une image lorsqu'elle est pointée par la souris :

```
img:hover {border: black 5px solid;}
```

# Pseudo-classes, pseudo-éléments

CSS3 s'enrichit de plusieurs pseudo-classes :

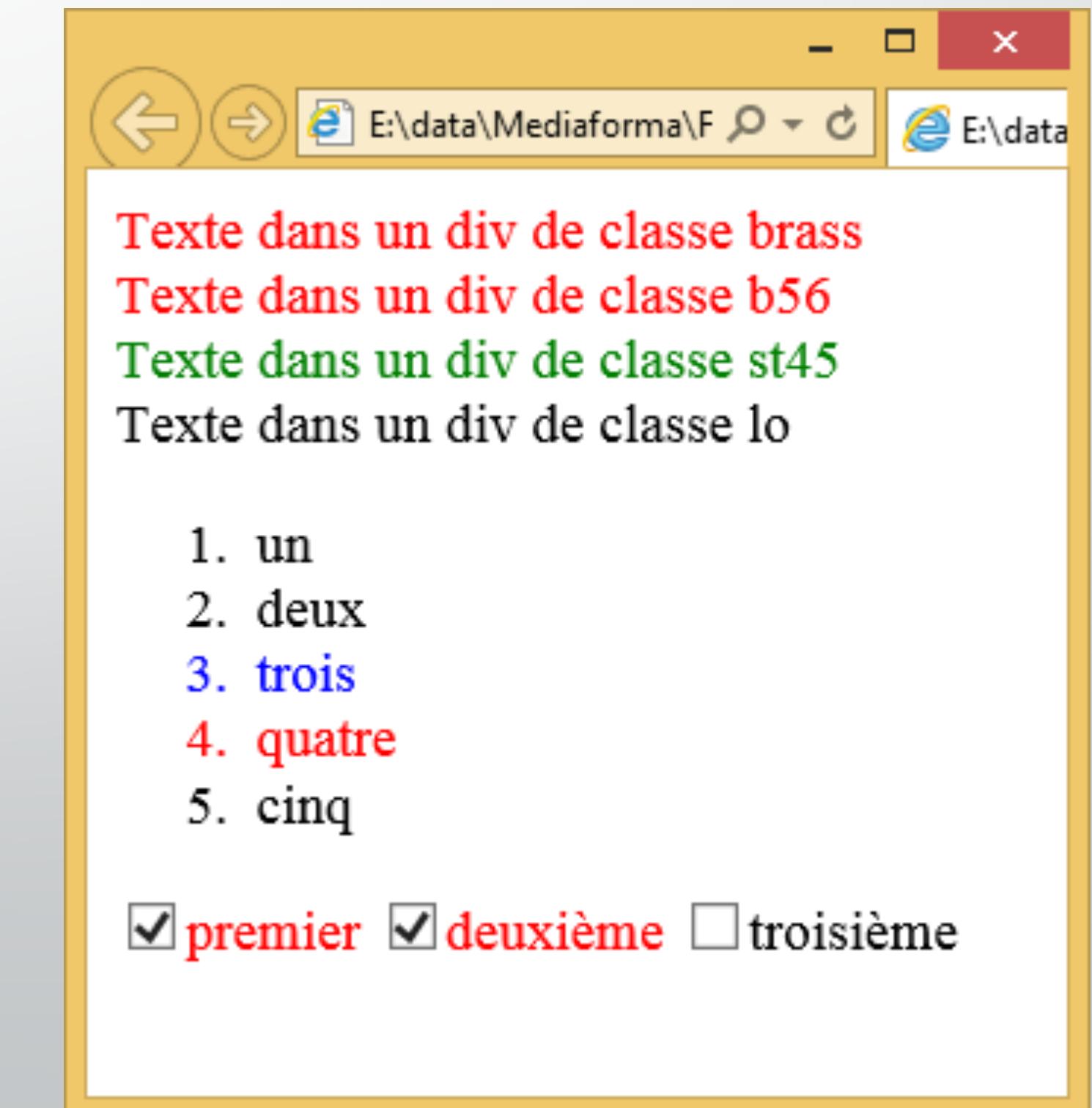
Pseudo-classe	Signification
<code>:nth-child(position)</code>	Éléments repérés par leur position dans le parent
<code>:nth-last-child(position)</code>	Dernier élément du parent
<code>:type:nth-of-type(position)</code>	Éléments du type spécifié repérés par leur position dans le parent
<code>:type:nth-last-of-type(N)</code>	Dernier élément du type spécifié du parent
<code>:last-child</code>	Dernier élément du parent
<code>:type:first-of-type</code>	Premier élément du type spécifié
<code>:last-of-type</code>	Dernier élément du type spécifié
<code>:only-child</code>	Élément qui est l'enfant unique du parent
<code>:type:only-of-type</code>	Élément qui est l'enfant unique du type spécifié du parent
<code>:root</code>	Éléments à la racine du document
<code>:empty</code>	Éléments sans enfant
<code>:target</code>	Élément dont l'identifiant est l'ancre spécifiée dans l'URL de la page
<code>:enabled</code>	Éléments validés (enabled)
<code>:disabled</code>	Éléments dévalidés (disabled)
<code>:checked</code>	Éléments (cases à cocher ou boutons radio) cochés
<code>:not(Sélecteur)</code>	Éléments qui sont différents du sélecteur

# Exercice

Quelle syntaxe utiliser pour s'adresser :

- 1) Aux éléments div dont l'attribut class commence par "br"
- 2) Aux éléments div dont l'attribut class contient la lettre "t"
- 3) L'avant-dernier élément li enfant de ol
- 4) Le troisième enfant li de l'élément ol de classe "t"
- 5) Les labels associés aux éléments checkbox qui sont cochés

Définissez le code HTML5 et CSS3 nécessaire pour obtenir ce résultat :



# Solution

- 1) div[class^="br"]
- 2) div[class\*="t"]
- 3) ol li:nth-last-child(2)
- 4) ol li:nth-child(3)
- 5) input[type=checkbox]:checked+label

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <div class="brass">Texte dans un div de classe brass</div>
    <div class="br56">Texte dans un div de classe b56</div>
    <div class="st45">Texte dans un div de classe st45</div>
    <div class="lo">Texte dans un div de classe lo</div>
    <ol class="t">
      <li>un</li>
      <li>deux</li>
      <li>trois</li>
      <li>quatre</li>
      <li>cinq</li>
    </ol>
    <form>
      <input type="checkbox" checked><label>premier</label>
      <input type="checkbox" checked><label>deuxième</label>
      <input type="checkbox"><label>troisième</label>
    </form>
  </body>
</html>
```

# Solution

```
div[class^="br"] {color: red;}  
div[class*="t"] {color: green;}  
ol li:nth-last-child(2) {color: red;}  
ol li:nth-child(3) {color: blue;}  
input[type=checkbox]:checked+label {color:red; }
```

# Unités CSS

Voici la liste des unités exprimées de façon absolue :

- cm : centimètre ;
- in : pouce (2,54 cm) ;
- mm : millimètre ;
- pt : point (1/62 inch) ;
- pc : pica (12 points).

À titre d'information, 1 in = 2,54 cm = 25,4 mm = 72 pt = 6 pc.

# Unités CSS

Il existe également des unités exprimées de façon relative, qui dépendent de leur élément parent :

- em : cadratin. Un em correspond grossièrement à la hauteur de la lettre "C" de l'élément parent (taille du texte définie dans la propriété font-size).
- ex : fraction de la hauteur des caractères. Dans les polices occidentales, ex correspond à la hauteur de la lettre minuscule "x".
- px : pixel. Cette unité dépend de la résolution du périphérique d'affichage.
- % : pourcentage de la taille de l'élément ou de son parent.

L'unité em n'est pas à exclure. Elle permet en effet aux utilisateurs finaux de redimensionner le texte proportionnellement à la taille de police, tout en conservant la mise en page intacte. C'est la raison pour laquelle je ne saurais trop vous recommander de l'utiliser !

# Ossature d'un document HTML 5

Dans cette partie :

- Nouvelle organisation des documents HTML
- Sections et niveaux de titre
- hgroup
- address

# Organisation des documents HTML5

Bien souvent, les pages web sont constituées d'un en-tête et d'un pied de page, d'un menu, d'une zone réservée au contenu (article, images, vidéos, etc.) et, éventuellement, d'une zone annexe, n'ayant aucun rapport direct avec le contenu de la page.

Le langage HTML5 a de nouvelles balises pour répondre à ces besoins.

# HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Un document HTML5 avec en-tête, bas de page, menu, zone annexe et Zone de contenu</title>
    <link rel="stylesheet" href="styles004.css">
    <script src="Creation-elements-HTML5.js"></script>
  </head>
  <body>
    <header>
      <h1>En-tête du document</h1>
      Texte de l'en-tête
    </header>
    <nav>
      <h2>Menu</h2>
      <ul>
        <li><a href="page1.htm">Page 1</a></li>
        <li><a href="page2.htm">Page 2</a></li>
        <li><a href="page3.htm">Page 3</a></li>
      </ul>
      <br><br><br>
    </nav>
    <aside>
      Texte affiché dans la partie droite de la page avec la balise &lt;aside&gt;<br><br>
    </aside>
    <article>
      <h2>Premier article</h2>
      <p>Texte du premier article</p>
    </article>
    <article>
      <h2>Deuxième article</h2>
      <p>Texte du deuxième article</p>
    </article>
    <footer>
      <p>Copyright et e-mail du Webmaster</p>
    </footer>
  </body>
</html>
```

Les balises <div> n'ont pas disparu du langage HTML5. Elles sont simplement moins souvent utilisées, car plusieurs nouvelles balises plus spécialisées viennent lui prêter main-forte.

Saisissez ce code et sauvegardez-le sous le nom code004.html

## CSS3

Saisissez ce code et sauvegardez-le sous le nom styles004.htm

```
header, nav, article, section, footer, aside  
{  
    display: block;  
}
```

Le rendu block (`display: block;`) est affecté aux nouveaux éléments HTML5 pour assurer le support de ces nouveaux éléments dans les navigateurs peu ou pas compatibles.

```
header  
{  
    background-color: red;  
}
```

header défini l'en-tête d'une section ou d'une page web.

```
nav  
{  
    float:left;  
    width:20%;  
    background-color:yellow;  
}
```

nav est destiné à contenir des liens de navigation dans le site.

```
article  
{  
    background-color: #66FFFF;  
    width:80%;  
}
```

article correspond à une entité autonome du document : par exemple, un article (ou un ensemble d'articles) extrait d'un blog ou d'un forum.

```
aside  
{  
    float:right;  
    width:20%;  
    background-color:yellow;  
}
```

aside définit un contenu qui n'a pas de rapport direct avec la page. Il peut être utilisé pour définir un menu ou pour donner accès aux archives du site.

```
footer  
{  
    clear:both;  
    background-color: #99FF66;  
}
```

footer définit le pied de page d'une section ou d'une page web.

Les navigateurs Internet Explorer 8 et inférieur ne sont pas en mesure d'affecter un style aux nouveaux éléments HTML5. En effet, puisqu'ils ne les connaissent pas, ils doivent les intégrer au DOM *via* la méthode `createElement`. Pour ce faire, quelques lignes de JavaScript sont nécessaires :

```
document.createElement("footer");  
document.createElement("header");  
document.createElement("article");  
document.createElement("nav");  
document.createElement("aside");
```

Saisissez ce code et  
sauvegardez-le sous le nom  
`Creation-elements-HTML5.js`

Exécutez le code HTML. Voici ce que vous devriez obtenir :



# Sections et niveaux de titre

Les éléments header, nav, aside, article et footer ne sont pas les seuls à faire leur apparition dans le langage HTML5. Vous pouvez également utiliser l'élément section pour définir des sections dans un document.

Il sert généralement pour segmenter une page ou un article en plusieurs sujets.

Il peut être employé conjointement aux éléments de titre h1 à h6 pour définir la structure du document.

Il correspond plus ou moins à l'ancien élément div du HTML 4.0x/XHTML 1.x, à ceci près qu'il y ajoute une signification sémantique. En effet, il regroupe des éléments qui traitent d'un même sujet.

# Sections et niveaux de titre

Ici, par exemple, nous utilisons un élément section pour englober plusieurs articles au sein du document :

```
<section>
    <article>
        Contenu du premier article
    </article>
    ...
    <article>
        Contenu du dernier article
    </article>
</section>
```

# Sections et niveaux de titre

Vous pouvez également inclure plusieurs contenus dans un article en utilisant plusieurs sections.

```
<article>

    <section>
        Contenu de la première section
    </section>

    ...
    <section>
        Contenu de la dernière section
    </section>

</article>
```

# Sections et niveaux de titre

Prenons l'exemple d'un blog, constitué d'un empilement de plusieurs articles (ou *posts*), non nécessairement liés les uns aux autres. Ces composants pourraient être insérés dans des éléments article, eux-mêmes insérés dans un élément section :

```
<section>
  <article>
    <h1>Titre de la première section</h1>
    Texte ou autre contenu qui présente la section
  </article>
  ..
  <article>
    <h1>Titre de la dernière section</h1>
    Texte ou autre contenu qui présente la section
  </article>
</section>
```

# Sections et niveaux de titre

Si vous avez du mal à faire la différence entre <section> et <article>, dites-vous qu'une section peut être composée d'un ou de plusieurs articles qui traitent d'un même sujet.

Par exemple, si une page web contient plusieurs sujets sur les baleines blanches et plusieurs autres sur les arbres fruitiers, les sujets sur les baleines blanches seront regroupés dans une même section et décomposés en autant d'articles que nécessaire.

De même en ce qui concerne les sujets sur les arbres fruitiers.



```
<section>
  <h1>Articles sur les baleines blanches</h1>
  <article>
    <h2>Les baleines blanches, habitat</h2>
    <p>texte de l'article</p>
  </article>
  <article>
    <h2>Les baleines blanches, population</h2>
    <p>texte de l'article</p>
  </article>
  <article>
    <h2>Les baleines blanches, une espèce menacée</h2>
    <p>texte de l'article</p>
  </article>
</section>
<section>
  <h1>Articles sur les arbres fruitiers</h1>
  <article>
    <h2>Les arbres fruitiers, catégories</h2>
    <p>texte de l'article</p>
  </article>
  <article>
    <h2>Les arbres fruitiers, taille</h2>
    <p>texte de l'article</p>
  </article>
  <article>
    <h2>Les arbres fruitiers, récolte</h2>
    <p>texte de l'article</p>
  </article>
</section>
```

# Sections et niveaux de titre

Un ou plusieurs niveaux de titre `<h1>` à `<h6>` peuvent être utilisés à l'intérieur d'un élément section, mais aussi article, aside ou nav. Dans le code suivant, des niveaux de titre `<h2>` sont utilisés dans les sections et un niveau de titre `<h1>` dans l'article qui englobe les sections :

# Mise en forme d'un document en HTML5 et CSS3

Dans cette partie :

- Polices exotiques
- Couleur et image d'arrière-plan
- Gradients linéaires
- Gradient radiaux
- Masquer une image avec un gradient
- RGBA() et HSLA()
- Regrouper des éléments
- etc..

# Polices exotiques

CSS3 permet d'utiliser des polices qui ne sont pas installées sur les "postes clients". C'est la propriété CSS3 @font-face qui autorise cette prouesse.

```
@font-face
{
    font-family: 'nom-police';
    [font-style: style-police;]
    [font-variant: petites-capitales;]
    [font-weight: graisse-police;]
    [font-stretch: condensé-étendu;]
    [font-size: taille-police;]
    src: [local('nom-local'),] url('url-police')
    [format(format-police)];
}
```

- nom-police est le nom de la police, tel qu'il sera utilisé dans la feuille de styles.
- style-police définit le style de la police : all, normal, italic ou oblique.
- petites-capitales indique la casse des caractères : normal ou small-caps.
- graisse-police indique la graisse de la police : all, normal, bold, 100, 200, 300, 400, 500, 600, 700, 800 ou 900.
- condensé-étendu définit l'état de compression/extension horizontale de la police : all, normal, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded ou ultra-expanded.
- taille-police définit les tailles possibles pour la police : all pour toutes les polices proportionnelles, une ou deux tailles pour les polices bitmap ou devant être rendues dans une taille ou une fourchette de tailles spécifiques.
- nom-local définit le nom local de la police, au cas où cette dernière serait installée sur l'ordinateur.
- url-police définit l'adresse URL de la police accessible sur le serveur.
- format-police définit le format de la police : truedoc-pfr (TrueDoc™ Portable Font Resource, .pfr), embedded-opentype (Embedded OpenType, .eot), type-1 (PostScript™ Type 1, .pfb ou .pfa), truetype (TrueType, .ttf) ou opentype (OpenType, .ttf).

# Polices exotiques

Cette formation va vous montrer comment :

1. Installer un kit de polices en utilisant le site Font Squirrel
2. Utiliser des Google Fonts
3. Insérer des caractères Font Awesome dans vos documents HTML

# Polices exotiques - Font Squirrel

Rendez-vous sur le site <https://www.fontsquirrel.com/> et trouvez la police que vous voulez utiliser en vous aidant des catégories sous **FIND FONTS**.

Attention, les polices qui portent la mention **OFF SITE** sont payantes.

The screenshot displays four distinct font preview sections on the Fontsquirrel website:

- Intro Rust:** Shows the text "INTRO RUST AABBC C DDEEFFG". Below it is a navigation bar with icons for desktop, mobile, and tablet, followed by the text "Intro Rust", "Fontfabric", "3 Styles", and a blue "DOWNLOAD OTF (OFFSITE)" button.
- Source Sans Pro:** Shows the text "Source Sans Pro AaBbCcDdEeFfGgH". Below it is a navigation bar with icons for desktop, mobile, and tablet, followed by the text "Source Sans Pro", "Adobe", "12 Styles", and a blue "DOWNLOAD OTF" button.
- Nexa Rust:** Shows the text "NEXA RUST ABCDEFGHIJ". Below it is a navigation bar with icons for desktop, mobile, and tablet, followed by the text "Nexa Rust", "Fontfabric", "5 Styles", and a blue "DOWNLOAD OTF (OFFSITE)" button.
- Open Sans:** Shows the text "Open Sans AaBbCcDdEeFfGgHhIij". Below it is a URL: <https://www.fontsquirrel.com/fonts/nexa-rust>.

# Polices exotiques - Font Squirrel

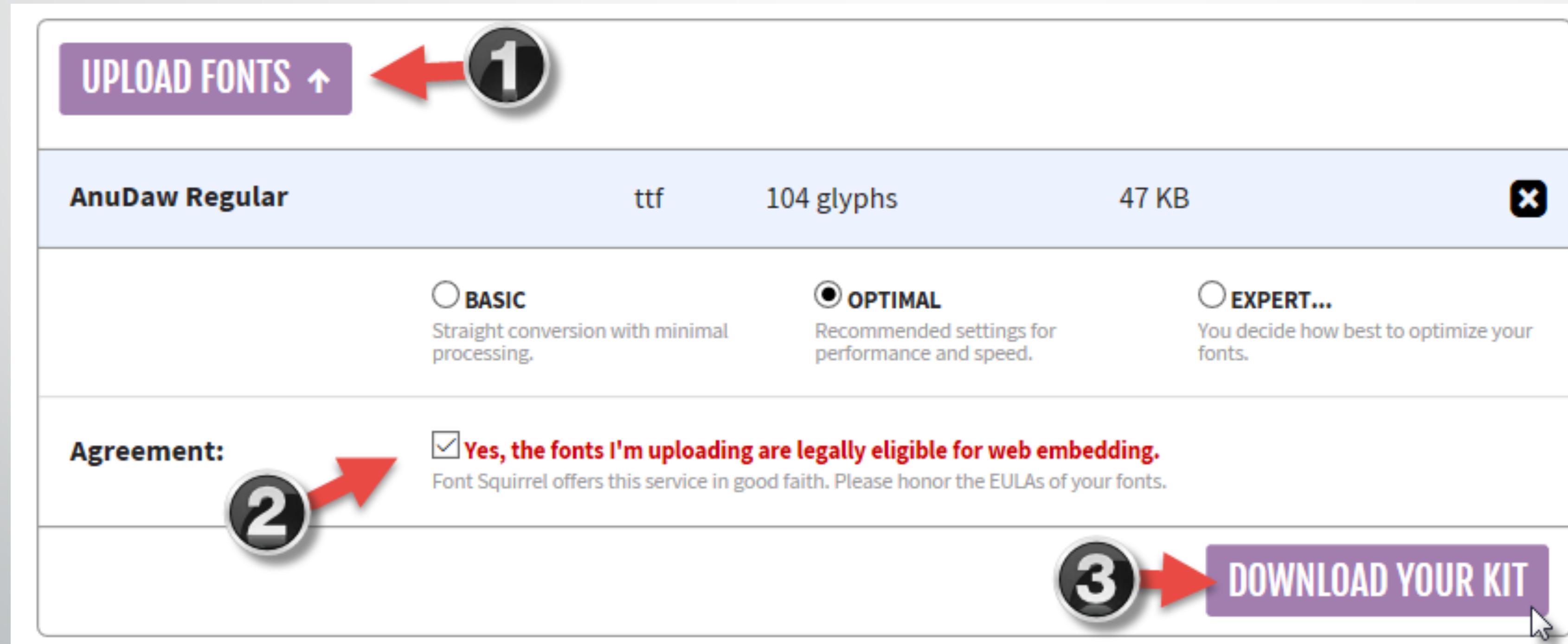
Cliquez sur le bouton **DOWNLOAD** de la police que vous voulez utiliser. Vous obtenez un fichier ZIP. Dézippez ce fichier. Vous obtiendrez un ou plusieurs fichiers de police et un fichier de licence. Lisez bien ce fichier pour savoir dans quel cadre vous pouvez utiliser la police.

Malheureusement, il existe plusieurs formats de polices et tous les navigateurs ne supportent pas tous les formats. Vous allez donc devoir décliner la police que vous avez téléchargée en plusieurs variantes pour assurer une compatibilité aussi grande que possible avec les divers navigateurs disponibles sur le marché.

Cette opération se fera sur le site Font Squirrel, en utilisant l'onglet **Generator**.

# Polices exotiques - Font Squirrel

Uploadez la police que vous avez téléchargée en cliquant sur **UPLOAD FONTS**, sélectionnez l'option **OPTIMAL** et cliquez sur **DOWNLOAD YOUR KIT** :



# Polices exotiques - Font Squirrel

Le kit de Font Squirrel est un fichier ZIP. Dézipez-le. Vous obtiendrez plusieurs fichiers de polices et un fichier CSS qui contient une instruction @font-face prête à l'emploi. Par exemple :

```
@font-face {  
    font-family: 'anudawregular';  
    src: url('anudrg__webfont.woff2') format('woff2'),  
         url('anudrg__webfont.woff') format('woff');  
    font-weight: normal;  
    font-style: normal;  
}
```

Utilisez cette instruction dans vos pages et faites référence à la police (ici anudawregular) avec une instruction du type suivant :

```
h1 {  
    font-family: 'anudawregular';  
}
```

# Polices exotiques - Font Squirrel

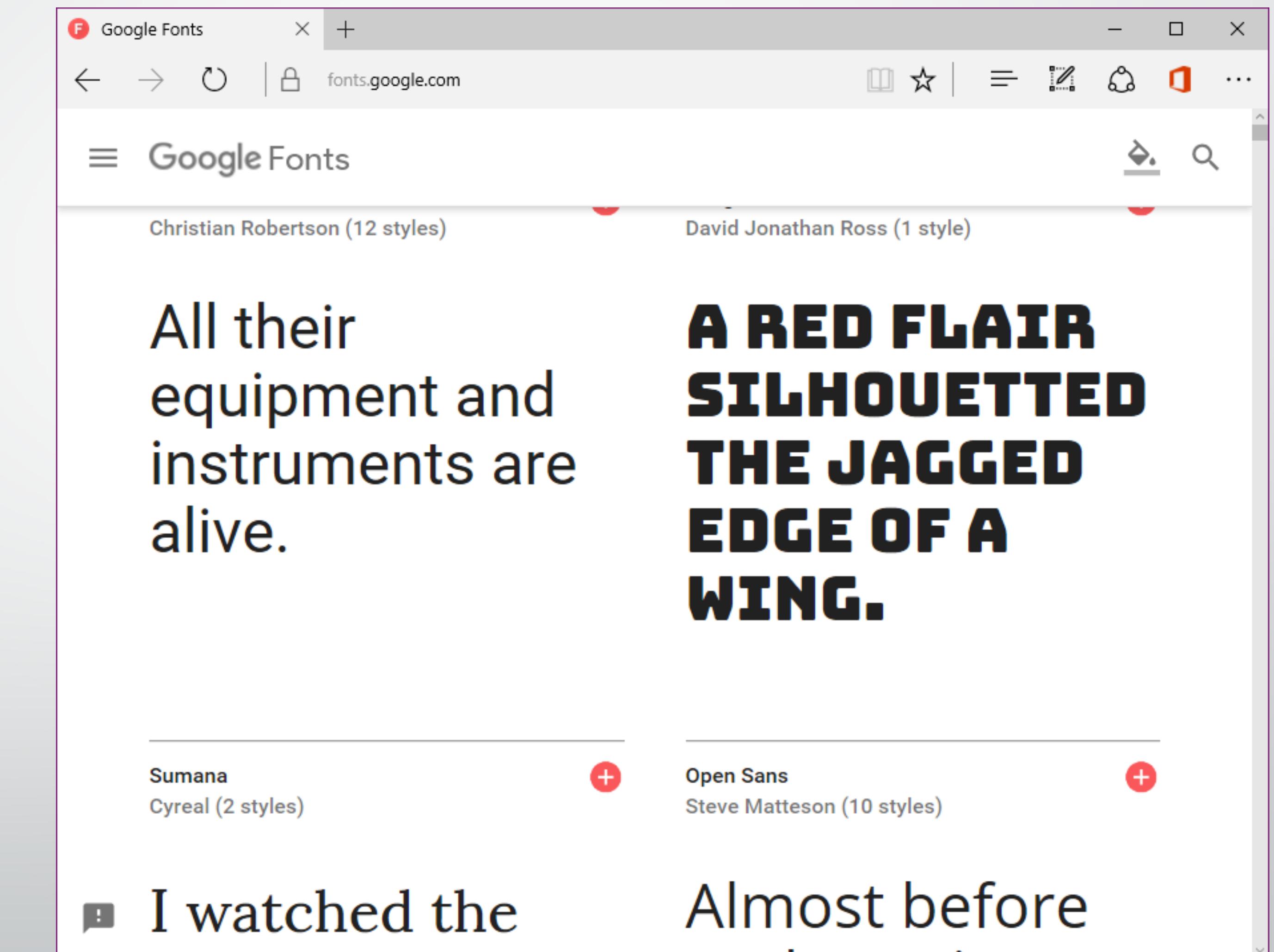
## Exercice

Téléchargez une police gratuite sur Font Squirrel, créez un kit de polices à partir de cette police et utilisez-le dans un document HTML.

# Polices exotiques - Google Fonts

Google propose un grand nombre de polices libres de droits. Il est très simple de les utiliser dans vos sites Web.

Commencez par vous rendre sur le site <https://fonts.google.com/> et trouvez la police que vous voulez utiliser.



# Polices exotiques - Google Fonts

Ici par exemple, nous allons utiliser la police **Bungee Inline**.

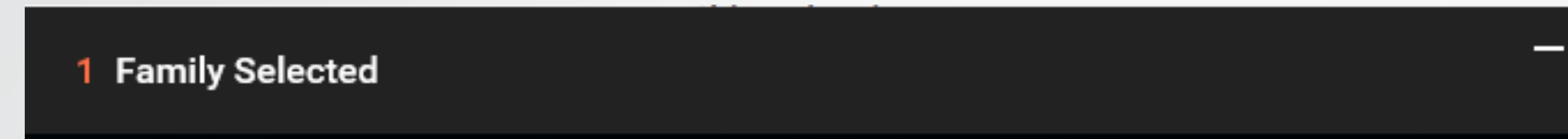
Cliquez sur le signe "+" correspondant :



# Polices exotiques - Google Fonts

La barre de titre d'une fenêtre popup s'affiche dans la partie inférieure de la page :

Cliquez dessus pour la développer la fenêtre. Vous trouverez les deux instructions à utiliser pour accéder à la police :

A screenshot of the Google Fonts interface showing expanded details for the selected font family. The "EMBED" tab is still active. The "Embed Font" section contains instructions for embedding the font into an HTML document, with a code snippet provided:

```
<link href="https://fonts.googleapis.com/css?family=Bungee+Inline" rel="stylesheet">
```

The "STANDARD" and "@IMPORT" options are also shown below this. The "Specify in CSS" section contains a CSS rule:

```
font-family: 'Bungee Inline', cursive;
```

At the bottom of the interface, a note reads: "For examples of how fonts can be added to webpages, see the [getting started guide](#)".

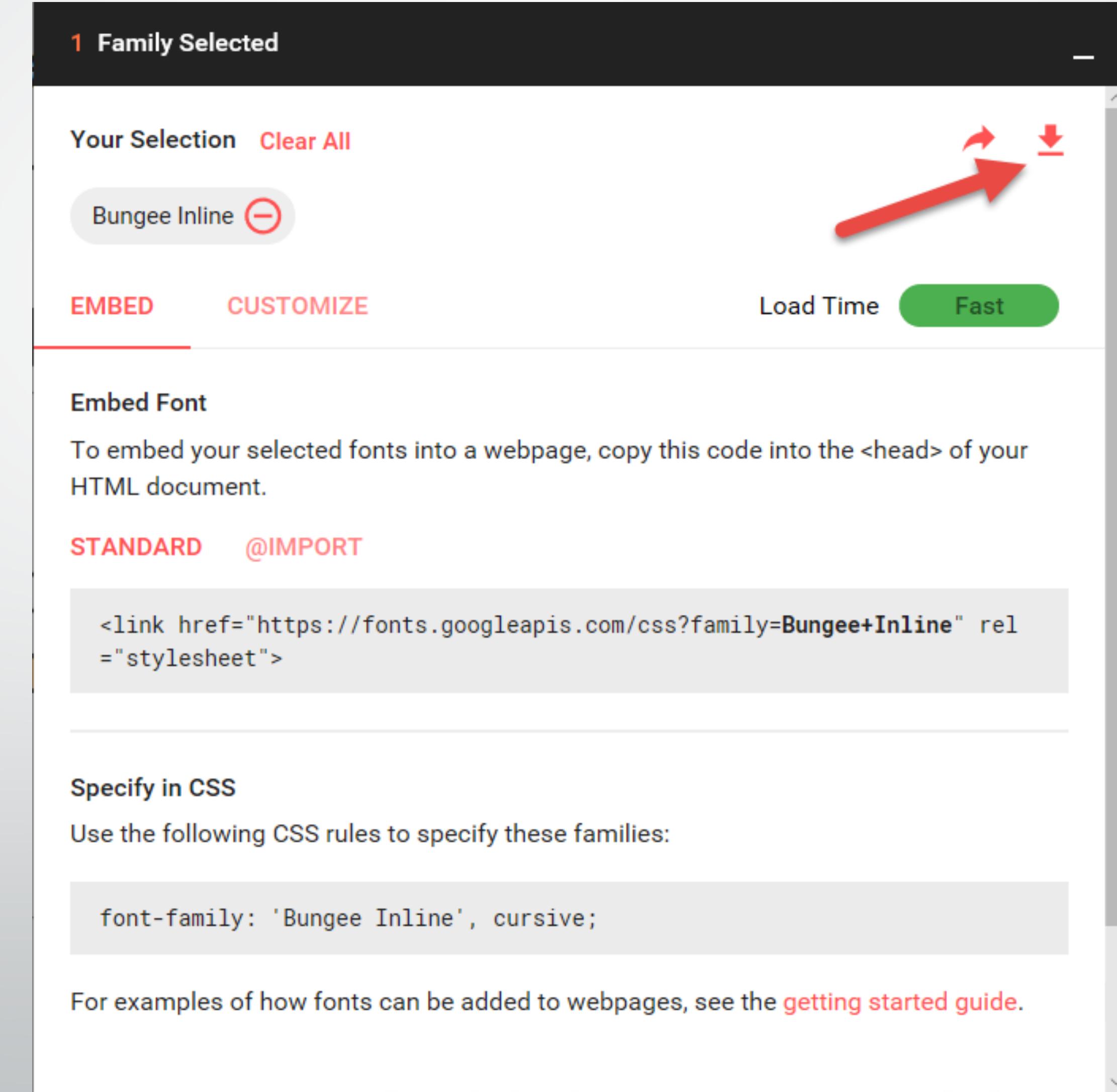
# Polices exotiques - Google Fonts

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Google Fonts</title>
    <link href="https://fonts.googleapis.com/css?family=Bungee+Inline" rel="stylesheet">
    <style>
      h1 {
        font-family: 'Bungee Inline', cursive;
      }
    </style>
  </head>
  <body>
    <h1>Ce titre utilise la Google Font Bungee Inline</h1>
  </body>
</html>
```



# Polices exotiques - Google Fonts

Si vous le souhaitez, vous pouvez également télécharger la police et l'inclure dans votre site en cliquant sur l'icône de téléchargement :



1 Family Selected

Your Selection Clear All

Bungee Inline -

EMBED CUSTOMIZE Load Time Fast

Embed Font

To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.

STANDARD @IMPORT

```
<link href="https://fonts.googleapis.com/css?family=Bungee+Inline" rel="stylesheet">
```

Specify in CSS

Use the following CSS rules to specify these families:

```
font-family: 'Bungee Inline', cursive;
```

For examples of how fonts can be added to webpages, see the [getting started guide](#).

# Polices exotiques - Google Fonts

Exercice

Entraînez-vous à utiliser une ou plusieurs polices Google Fonts.

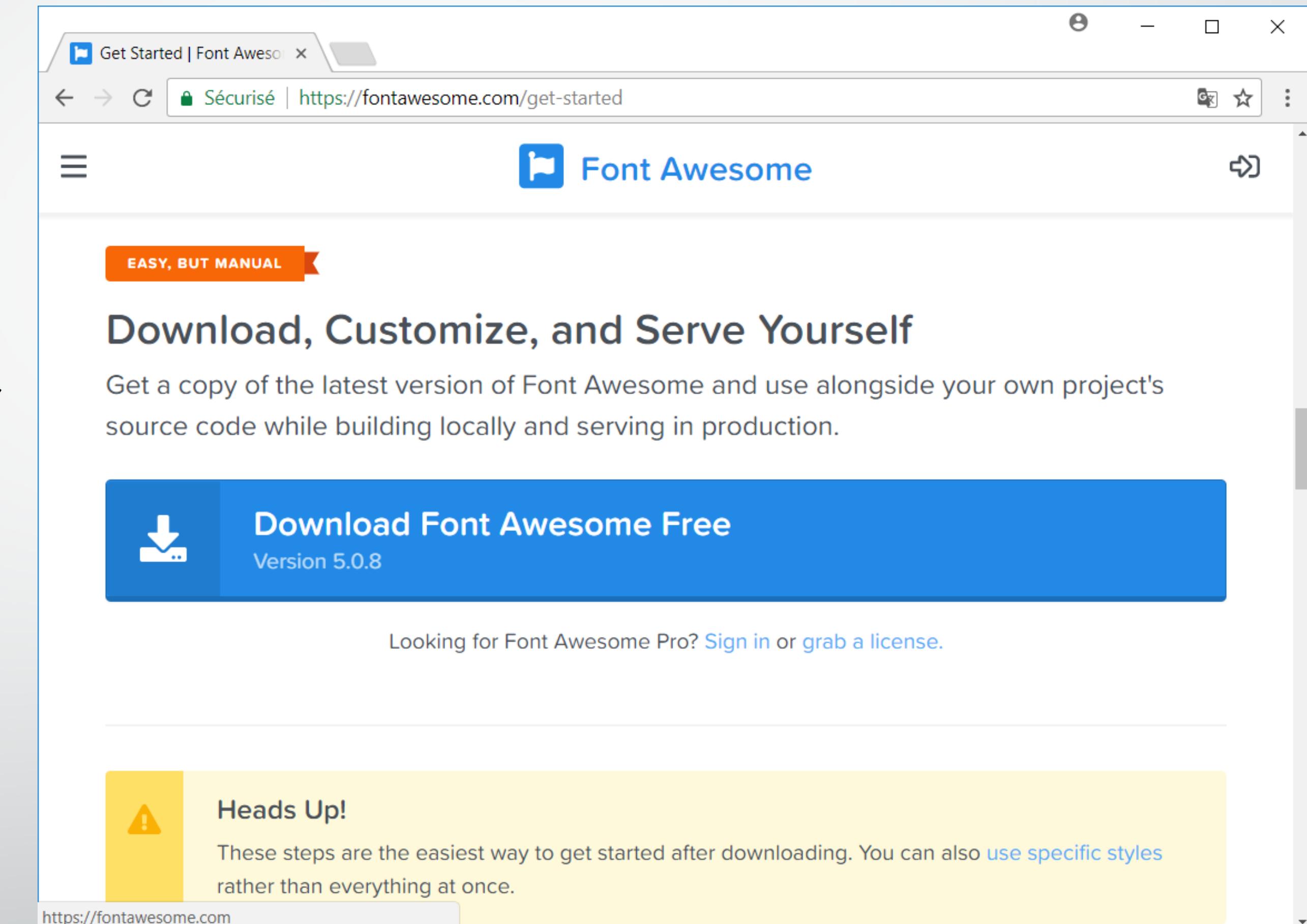
# Font Awesome

Font Awesome donne accès à de nombreuses icônes dignes d'intérêt.

Pour y accéder, commencez par vous rendre sur le site

<https://fontawesome.com/get-started>

Et téléchargez Font Awesome en cliquant sur le bouton **Download Font Awesome Free** :



Extrayez le fichier **fontawesome-all.js** de cette archive et placez-le dans un dossier de votre ordinateur.

Vous y ferez référence avec une balise <script> dans le <head> :

```
<script defer src="fontawesome-all.js"></script>
```

Ca y est, vous pouvez utiliser les icônes de Font Awesome dans votre code HTML !

# Font Awesome

Pour accéder aux icônes de FontAwesome, allez sur la page

<https://fontawesome.com/icons?d=gallery>

Cliquez sur une des icônes proposées pour savoir comment l'intégrer dans votre code HTML. Vers le bas de la page, vous obtiendrez le code de l'icône. Par exemple :

```
<i class="fab fa-angellist"></i>
```

# Font Awesome

Exercice :

Insérez le code nécessaire pour afficher une bicyclette.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Font Awesome</title>
    <script defer src="fontawesome-all.js"></script>
    <style>

      </style>
  </head>
  <body>
    <p>ceci est un texte</p>
    <i class="fas fa-cubes"></i>
    <i class="fas fa-camera-retro"></i>
  </body>
</html>
```

Solution

# Font Awesome

## Exercice

En consultant les pages du site <http://fontawesome.io/> ; trouvez comment changer la couleur des icônes Font Awesome, comment choisir leur taille ou encore comment avoir des icônes animées.



# Font Awesome

## Solution

```
<i class="fa fa-bicycle fa-5x" aria-hidden="true"></i>
<i class="fa fa-spinner fa-spin fa-5x"></i>
<i class="fa fa-refresh fa-spin fa-5x fa-fw"></i>
```

Un exemple de remplacement des puces d'une liste à puces par des caractères FontAwesome.

Ici, on utilise le sélecteur li::before pour insérer un contenu devant les <li>. Le contenu est inséré sous la forme d'un caractère Unicode de police FontAwesome.

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Arriere plan fluide</title>
    <link rel="stylesheet"
          href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <style>
      ul {
        list-style-type: none;
        padding-left: 20px;
      }
      li{color:#ff00ff;font-size: 30px;list-style:none; }
      li {
        position: relative;
        padding-left: 20px;
        margin-bottom: 10px
      }
      li::before{
        font-family: FontAwesome;
        content: "\f1b0";
      }
    </style>
  </head>
  <body>
    <ul>
      <li>
        un
      </li>
      <li>
        deux
      </li>
      <li>
        trois
      </li>
    </ul>
  </body>
</html>
```

# Couleur et image d'arrière-plan

La couleur d'arrière-plan d'un élément est définie par la propriété background-color. Voici sa syntaxe :

```
background-color: couleur;
```

La valeur couleur peut être spécifiée :

- "En dur", c'est-à-dire sous la forme d'un nom normalisé : yellow ou skyblue par exemple.
- À l'aide d'un code hexadécimal sur 6 digits : #RRVVBB (où RR, VV et BB sont les composantes rouge, vert et bleu de la couleur, codées en hexadécimal entre 00 et FF).
- À l'aide de la fonction RGB(R,V,B), où R, V et B sont des valeurs comprises entre 0 et 255.

L'image d'arrière-plan d'un élément est définie par la propriété background-image. Voici sa syntaxe :

```
background-image: url (image);
```

Où image est l'adresse URL de l'image. Par exemple img/fond.jpg, ou encore http://monsite.com/image/fond.png.

Il n'est pas recommandé d'utiliser des guillemets pour délimiter l'adresse URL de l'image d'arrière-plan car ils peuvent provoquer des dysfonctionnements dans les navigateurs anciens.

# Couleur et image d'arrière-plan

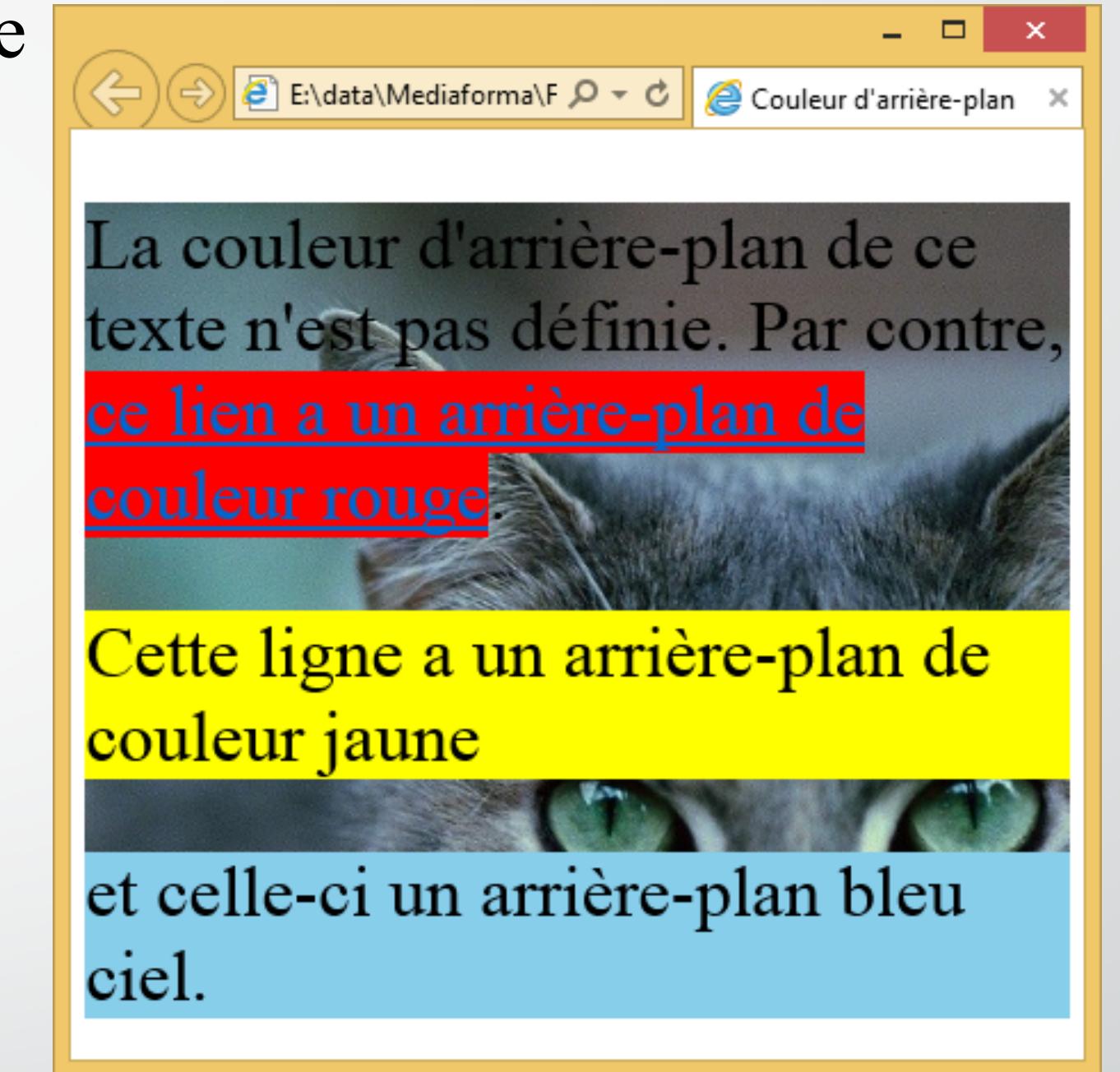
Une image d'arrière-plan peut être répétée horizontalement et/ou verticalement. Pour cela, on donne la valeur repeat (répétition horizontale et verticale), repeat-x (répétition horizontale seulement) ou repeat-y (répétition verticale seulement), à la propriété CSS background-repeat.

La position d'une image d'arrière-plan peut être définie horizontalement et/verticalement : affectez la position horizontale (left, center ou right) et éventuellement la position verticale (top, center ou bottom) à la propriété CSS background-position.

# Couleur et image d'arrière-plan

Dans cet exemple (code005.htm), du texte est affiché au-dessus d'une image en arrière-plan. Nous modifions tour à tour la couleur d'arrière-plan d'un lien hypertexte, d'un paragraphe, puis d'un élément div. Les styles correspondants sont directement définis dans les éléments concernés :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Couleur d'arrière-plan</title>
  </head>
  <body>
    <div style="background-image: url(chat.jpg); font-size: 40px; color: black;">
      <p>La couleur d'arrière-plan de ce texte n'est pas définie. Par contre, <a href="#" style="background-color: red;">ce lien a un arrière-plan de couleur rouge</a>.</p>
      <p style="background-color: yellow;">Cette ligne a un arrière-plan de couleur jaune</p>
      <div style="background-color: skyblue;">et celle-ci un arrière-plan bleu ciel.</div>
    </div>
  </body>
</html>
```



# Gradient vertical

Pour définir un gradient vertical, vous utiliserez ces propriétés CSS :

```
-webkit-gradient(linear, p1, p2, from(c1), to(c2));  
-ms-linear-gradient(p1, c1, c2) no-repeat;  
-moz-linear-gradient(p1, c1, c2) no-repeat;
```

Où :

- p1 représente les coordonnées du point de départ du gradient : top, left, right et/ou down sur les navigateurs IE et Firefox ; pourcentage horizontal et pourcentage vertical sur les navigateurs Webkit.
- p2 représente les coordonnées du point d'arrivée du gradient sur les navigateurs Webkit. Il est défini par un pourcentage horizontal et un pourcentage vertical de la zone cible.
- c1 est la couleur de départ du gradient.
- c2 est la couleur de fin du gradient.

Cette solution n'est pas encore implémentée sur tous les navigateurs et il est nécessaire d'utiliser des préfixes pour s'adresser à chaque navigateur : -webkit pour Chrome, -ms pour IE, -moz pour Firefox.

# Exercice

Définissez un gradient vertical de la couleur rouge à la couleur bleue.

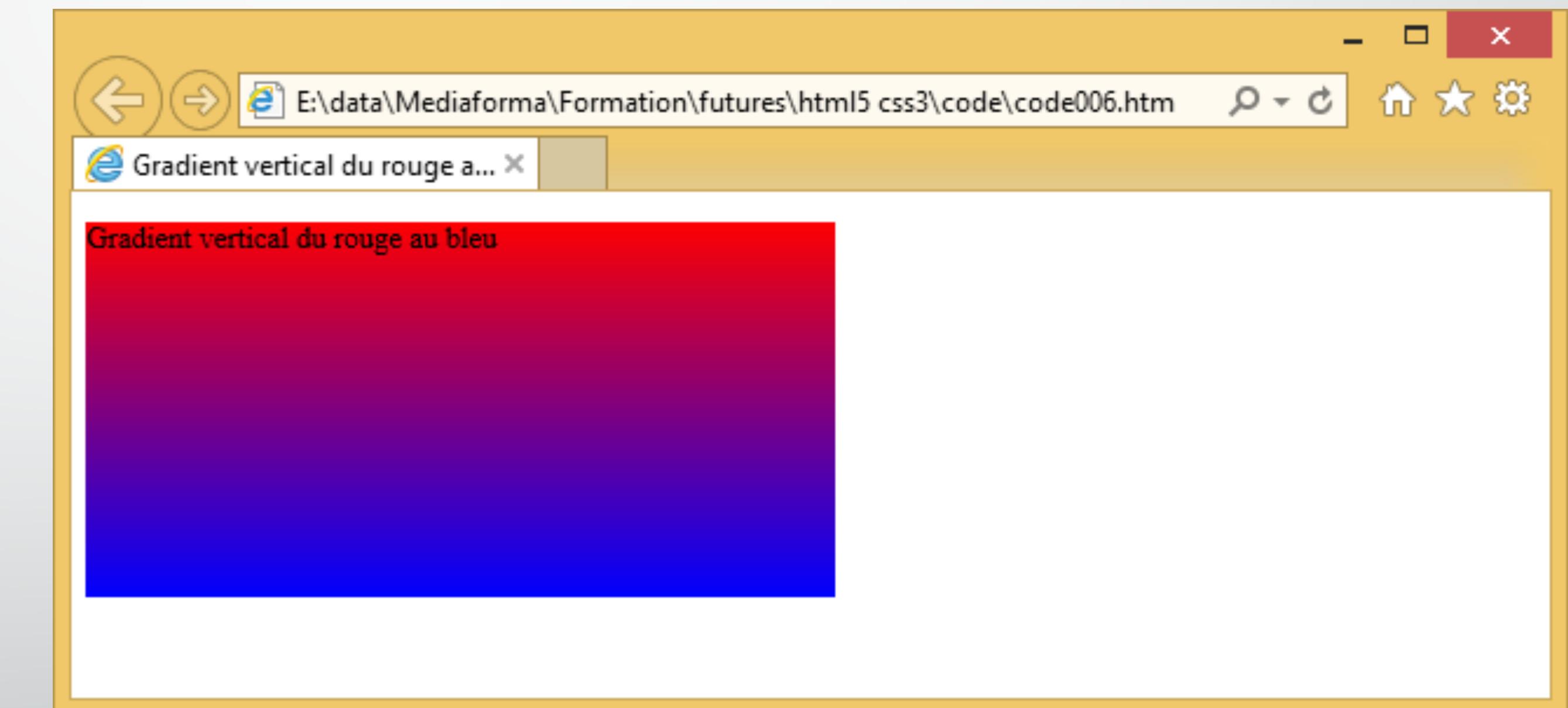
# Solution (code006.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient vertical du rouge au bleu</title>
    <link rel="stylesheet" href="gradient-vertical.css">
  </head>
  <body>
    <p class="gradient-vertical">Gradient vertical du rouge au bleu</p>
  </body>
</html>
```

Code CSS3

```
.gradient-vertical
{
  width:400px;
  height: 200px;
  background: -ms-linear-gradient(top, red, blue);
  background: -moz-linear-gradient(top, red, blue);
  background: -webkit-linear-gradient(top, red, blue);
}
```

Code HTML5



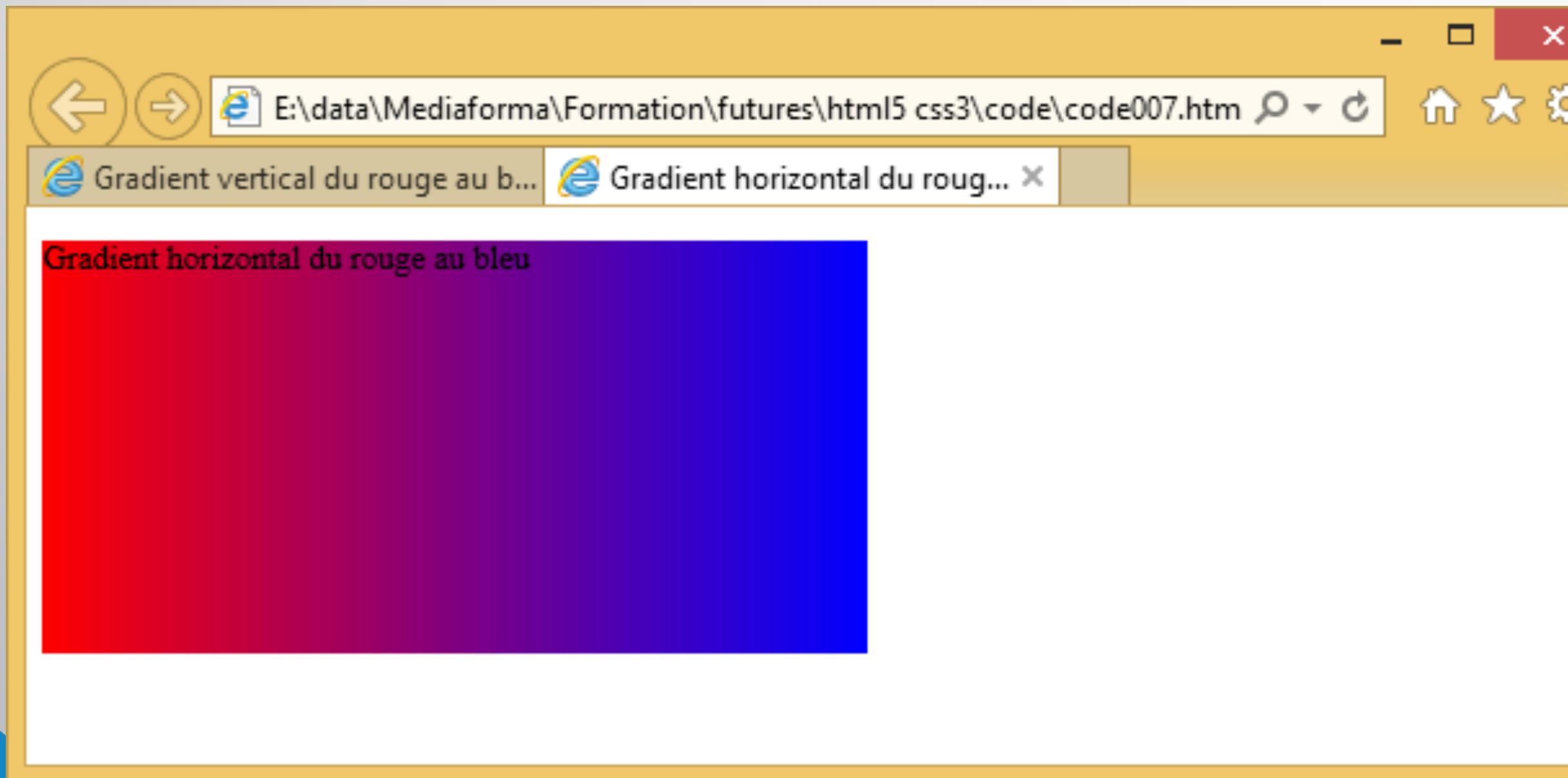
# Exercice

Définissez un gradient horizontal de la couleur rouge à la couleur bleue.

# Solution (code007.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient horizontal du rouge au bleu</title>
    <link rel="stylesheet" href="gradient-horizontal.css">
  </head>
  <body>
    <p class="gradient-horizontal">Gradient horizontal du rouge
au bleu</p>
  </body>
</html>
```

Code HTML5



Code CSS3

```
.gradient-horizontal
{
  width:400px;
  height: 200px;
  background: -ms-linear-gradient(left, red, blue);
  background: -moz-linear-gradient(left, red,
blue);
  background: -webkit-gradient(linear, 0% 0%, 100%
0%, from(red), to(blue));
}
```

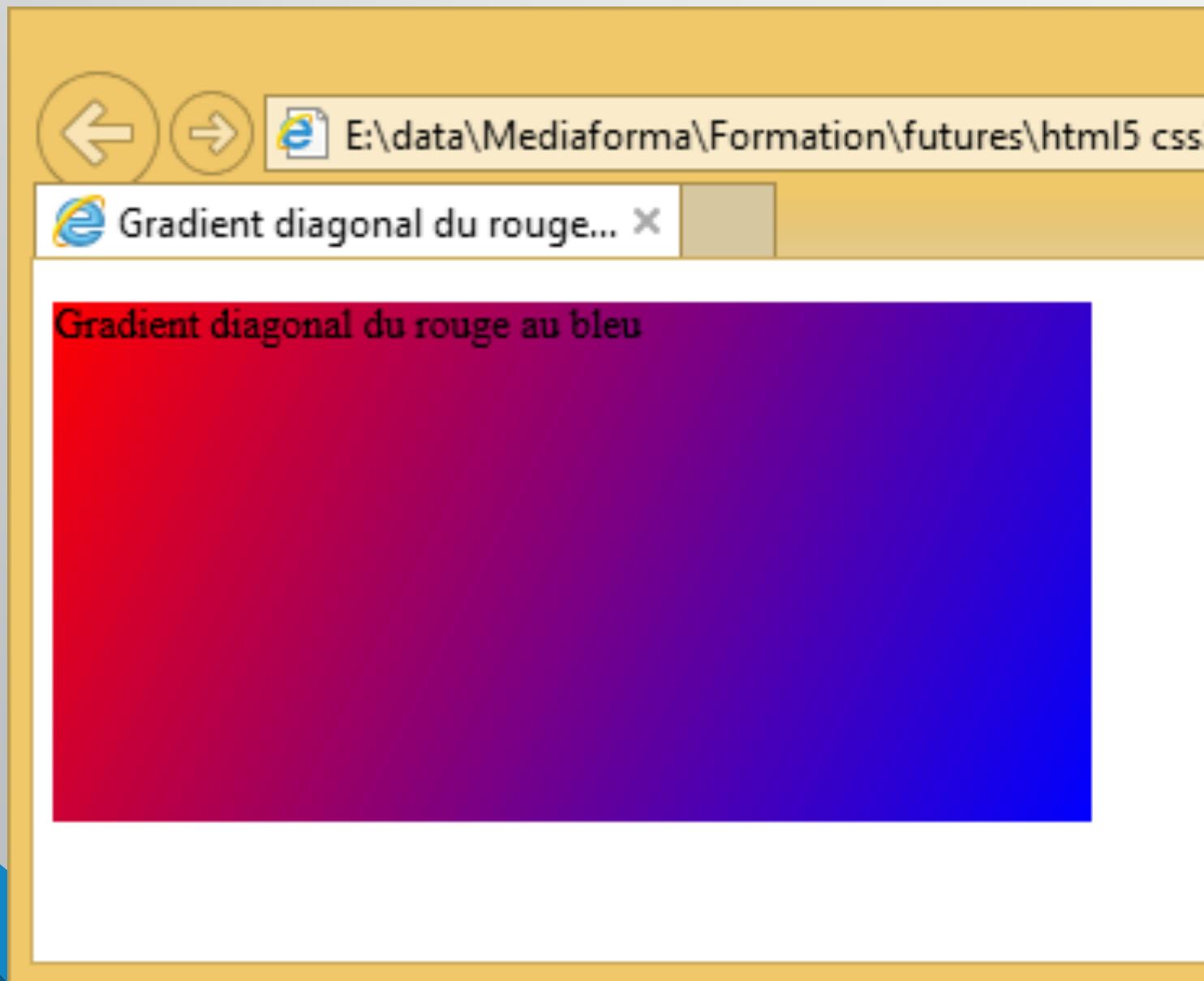
# Exercice

Définissez un gradient diagonal de la couleur rouge à la couleur bleue.

# Solution (code008.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient diagonal du rouge au bleu</title>
    <link rel="stylesheet" href="gradient-diagonal.css">
  </head>
  <body>
    <p class="gradient-diagonal">Gradient diagonal du rouge au bleu</p>
  </body>
</html>
```

Code HTML5



Code CSS3

```
.gradient-diagonal
{
  width:400px;
  height: 200px;
  background: -moz-linear-gradient(top left, red, blue);
  background: -ms-linear-gradient(top left, red, blue);
  background: -webkit-gradient(linear, 0% 0%, 100% 100%,
from(red), to(blue));
}
```

# Plusieurs couleurs dans un gradient linéaire

Vous savez définir un gradient linéaire à deux couleurs. Il est possible de lui adjoindre des couleurs intermédiaires, grâce à l'ajout de quelques paramètres dans les propriétés -webkit-gradient, -ms-linear-gradient et -moz-linear-gradient.

# Plusieurs couleurs dans un gradient linéaire

## Navigateurs Gecko et IE

```
-moz-linear-gradient(p1, c1 p1, c2 p2, ..., cN);  
-ms-linear-gradient(p1, c1 p1, c2 p2, ..., cN);
```

Où :

- p1 à pN-1 sont les coordonnées des différentes transitions de couleurs, exprimées en pourcentages.
- c1 à cN sont les différentes couleurs du gradient.

# Plusieurs couleurs dans un gradient linéaire

## Navigateurs Webkit

```
-webkit-gradient(linear, p1, p2, from(c1), color-stop(v2, c2), color-stop(v3, c3), ..., to(cN));
```

Où :

- p1 et p2 sont les coordonnées des points de départ et d'arrivée du gradient, exprimées en pourcentages.
- c1 à cN sont les différentes couleurs du gradient.
- vI sont les positions des différentes transitions de couleurs, exprimées dans un nombre décimal compris entre 0 et 1.

# Exercice

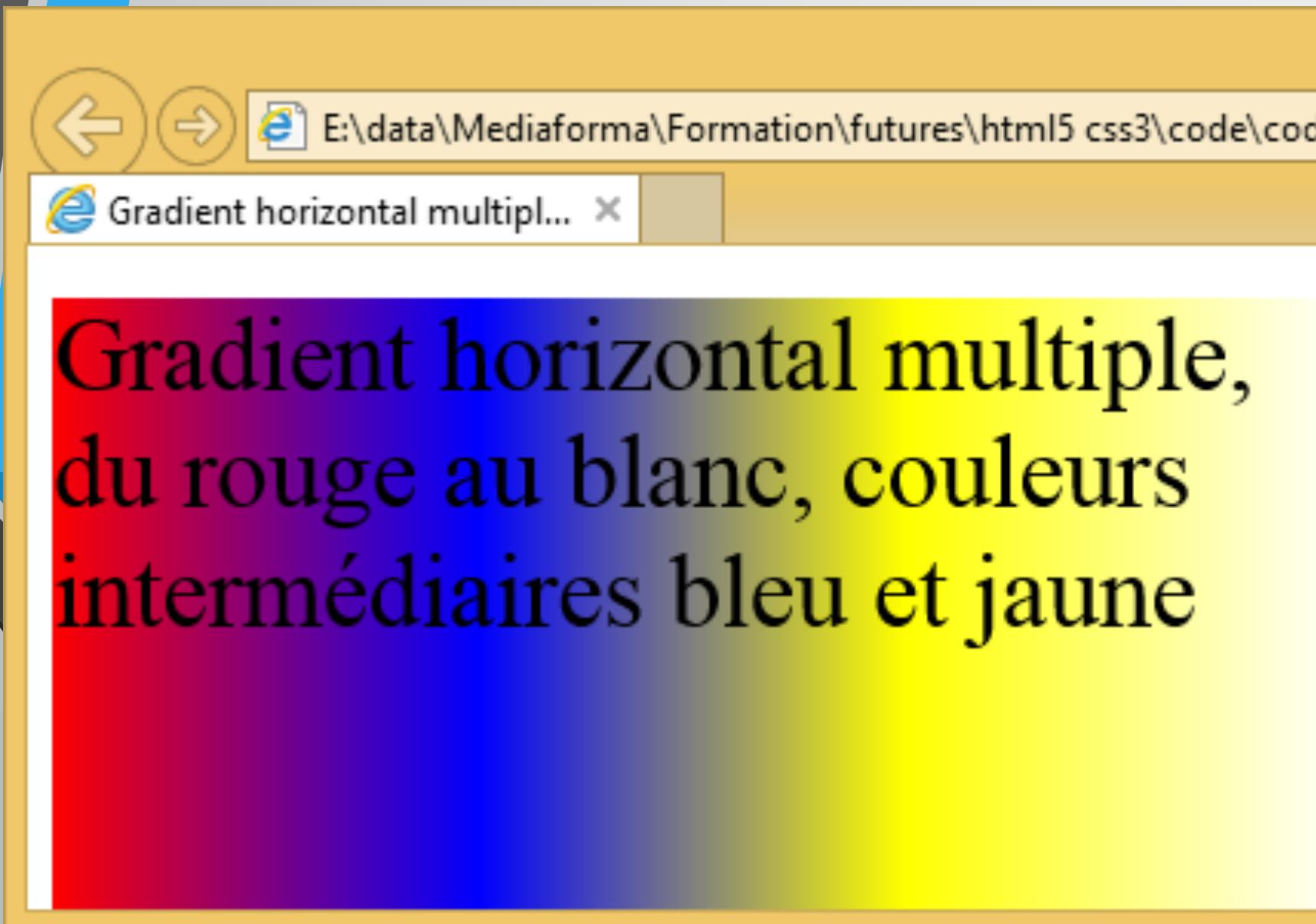
Définissez un gradient linéaire horizontal – qui passe du rouge au bleu au jaune et au blanc – en répartissant de façon égale les différentes transitions. Ce gradient doit être compatible Gecko, IE et Mozilla.

# Solution (code009.htm)

Deux points intermédiaires doivent être définis : un à 33 % et un à 66 % de la zone cible.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient horizontal multiple, du rouge au blanc, couleurs
intermédiaires bleu et jaune</title>
    <link rel="stylesheet" href="gradient-horizontal-multiple.css">
  </head>
  <body>
    <p class="gradient-horizontal-multiple"><font size=6>Gradient horizontal
multiple, du rouge au blanc, couleurs intermédiaires bleu et jaune</font></
p>
  </body>
</html>
```

Code HTML5



Code CSS3

```
.gradient-horizontal-multiple
{
  width:400px;
  height: 200px;
  background: -moz-linear-gradient(left, red 0%, blue 33%, yellow 66%, white);
  background: -ms-linear-gradient(left, red 0%, blue 33%, yellow 66%, white);
  background: -webkit-gradient(linear, 0% 0%, 100% 0%, from(red), color-stop
(0.33, blue), color-stop(.66, yellow), to(white));
}
```

# Plusieurs couleurs dans un gradient linéaire

Pour créer vos gradients plus aisément, vous pouvez vous rendre à la page <http://www.colorzilla.com/gradient-editor/>, où quelques réglages élémentaires permettent d'obtenir le code CSS3 correspondant.

# Gradient radial

Les gradients linéaires ne sont pas les seuls utilisables : on peut également définir des gradients radiaux, qui fixent un dégradé de couleurs d'une zone circulaire vers le reste de l'élément ciblé. Les propriétés à utiliser sont les suivantes :

- `-webkit-gradient(radial, ...)` sur les navigateurs Webkit (Safari, Chrome) ;
- `-ms-radial-gradient` sur les navigateurs Internet Explorer ;
- `-moz-radial-gradient` sur les navigateurs Gecko (Mozilla Firefox).

# Gradient radial

## Navigateurs Webkit

```
background: -webkit-gradient(radial, p1, r1, p2, r2, from(c1), to(c2));
```

Où :

- p1 représente les coordonnées du point intérieur du gradient.
- r1 est le rayon intérieur du gradient.
- p2 représente les coordonnées du point extérieur du gradient.
- r2 est le rayon extérieur du gradient.
- c1 est la couleur intérieure du gradient.
- c2 est la couleur extérieure du gradient.

# Gradient radial

## Navigateurs Gecko et Internet Explorer

```
background: -moz-radial-gradient(p1, forme taille, c1, c2);
```

```
background: -ms-radial-gradient(p1, forme taille, c1, c2);
```

Où :

- p1 représente les coordonnées du point intérieur du gradient.
- forme représente la forme du gradient : circle ( cercle) ou ellipse ( ellipse).
- taille est la taille du gradient. Ce paramètre peut prendre d'une des valeurs suivantes :
  - closest-side (ou contain) pour que le gradient occupe toute la taille du conteneur.
  - closest-corner pour que le gradient occupe tout l'espace disponible jusqu'au coin le plus proche du conteneur.
  - farthest-side (ou cover) pour que le gradient occupe tout l'espace disponible jusqu'au coin le plus éloigné du conteneur.
- c1 est la couleur intérieure du gradient.
- c2 est la couleur extérieure du gradient.

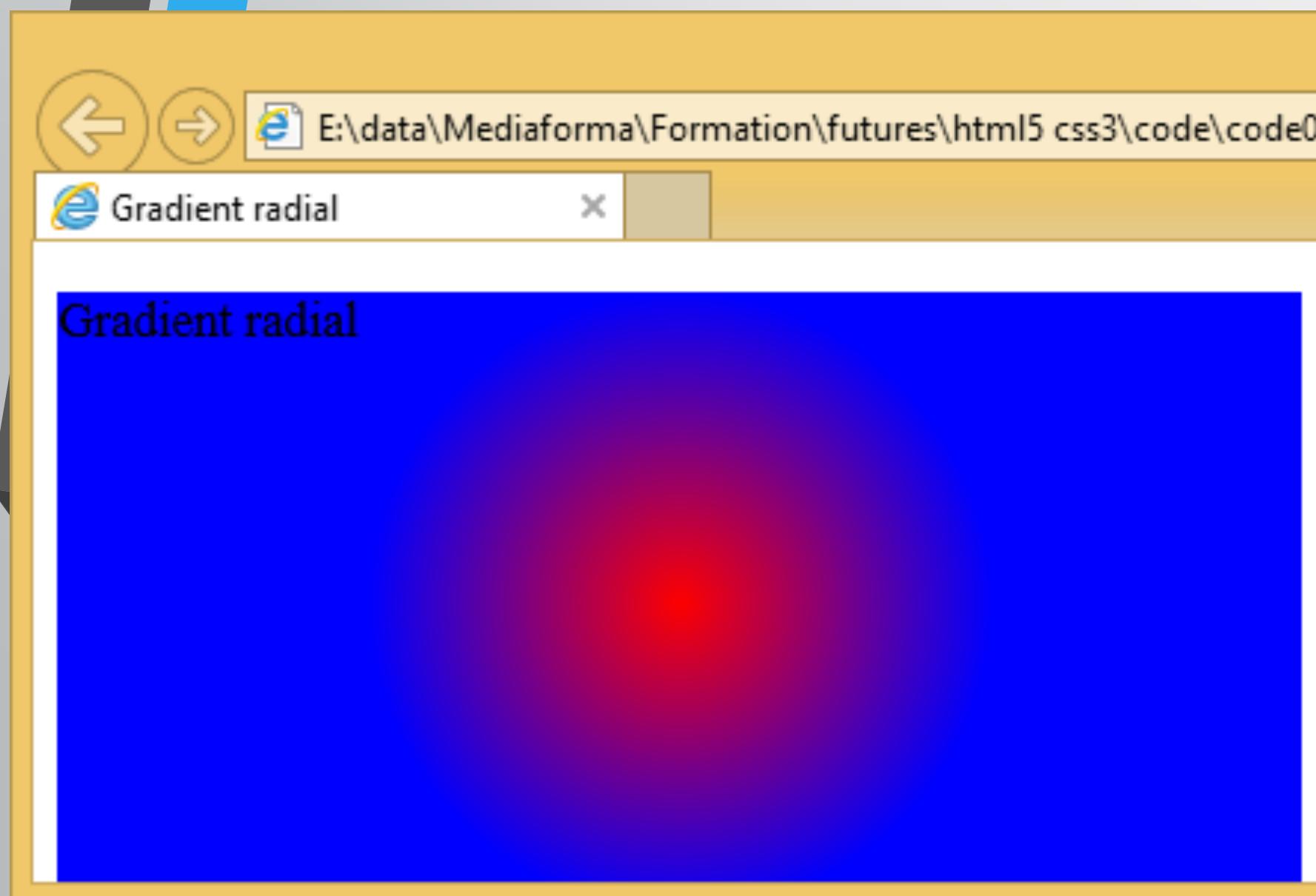
# Exercice

Définissez un gradient vertical du rouge au bleu, compatible Mozilla, Gecko et IE.

# Solution (code010.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient radial</title>
    <link rel="stylesheet" href="gradient-radial.css">
  </head>
  <body>
    <p class="gradient-radial">Gradient radial</p>
  </body>
</html>
```

Code HTML5



Code CSS3

```
.gradient-radial
{
  width: 400px;
  height: 200px;
  background: -webkit-gradient(radial, 50% 0%, 20, 50% 100%, 150, from(red),
  to(blue));
  background: -ms-radial-gradient(50% 50%, circle closest-side, red, blue);
  background: -moz-radial-gradient(50% 50%, circle closest-side, red, blue);
}
```

# Gradient radial

Pour que le gradient n'occupe pas tout le conteneur, la méthode est totalement différente dans les navigateurs Webkit, Internet Explorer et Gecko.

## Navigateurs Webkit

Définissez un rayon extérieur de plus petite taille dans le code CSS3. Ici, 50 pixels au lieu de 150 :

```
background: -webkit-gradient(radial, 50% 50%, 0, 50% 50%, 50,  
from(red), to(blue));
```

# Gradient radial

## Navigateurs Internet Explorer et Gecko

Il est nécessaire d'ajouter deux pourcentages lors de la définition des couleurs. Ici, le rouge constitue le point de départ du gradient (0 %) et le bleu s'arrête à 30 % de la taille maximale du gradient (30 % de farthest-side) :

```
background: -moz-radial-gradient(50% 50%, circle farthest-side, red 0%, blue 30%);
```

```
background: -ms-radial-gradient(50% 50%, circle farthest-side, red 0%, blue 30%);
```

# Exercice

Définissez un gradient radial de 50 pixels du jaune au rouge. Ce gradient sera compatible avec les navigateurs Gecko, IE et Mozilla.

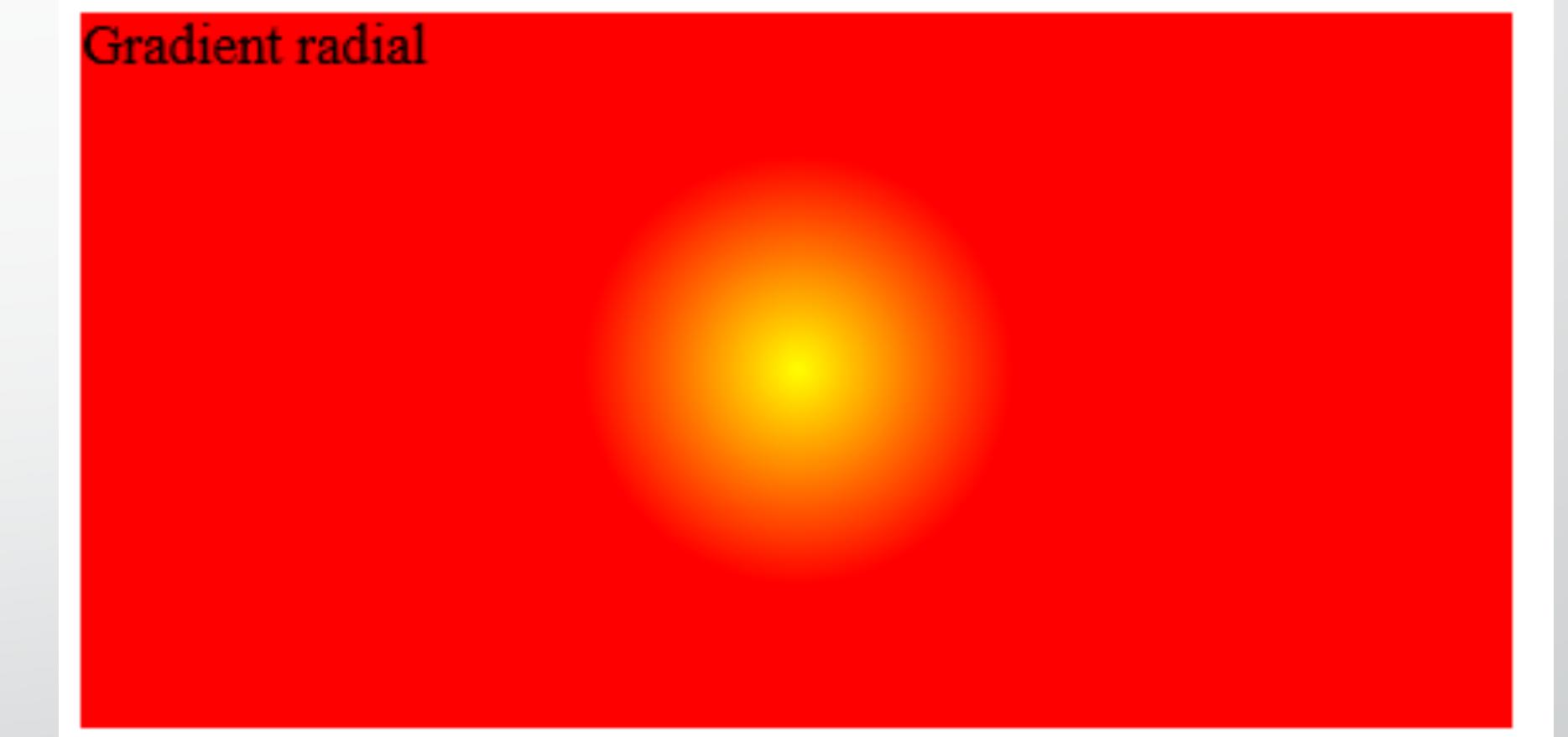
# Solution (code011.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient radial</title>
    <link rel="stylesheet" href="gradient-radial2.css">
  </head>
  <body>
    <p class="gradient-radial">Gradient radial</p>
  </body>
</html>
```

Code CSS3

```
.gradient-radial
{
  width: 400px;
  height: 200px;
  background: -webkit-gradient(radial, 50% 50%, 0, 50% 50%, 50, from(yellow), to(red));
  background: -moz-radial-gradient(50% 50%, circle farthest-side, yellow 0%, red 30%);
  background: -ms-radial-gradient(50% 50%, circle farthest-side, yellow 0%, red 30%);
}
```

Code HTML5



# Plusieurs couleurs dans un gradient radial

Tout comme pour les gradients linéaires, il est possible de définir plusieurs couleurs dans un gradient radial. La syntaxe à utiliser est très différente selon qu'on s'adresse à un navigateur Webkit, IE ou Gecko.

## Navigateurs Webkit

Un ou plusieurs color-stop doivent être définis entre la couleur de départ (from) et la couleur d'arrivée (to). Ici, par exemple, le gradient va du rouge au noir en passant par le jaune et le bleu :

```
background: -webkit-gradient(radial, 50% 50%, 0, 50% 50%, 150,  
from(red), color-stop(.25,yellow), color-stop(.50, blue), to (black));
```

# Plusieurs couleurs dans un gradient radial

## **Navigateurs IE et Gecko**

La syntaxe est beaucoup plus simple dans les navigateurs Gecko. Il suffit de préciser les différentes couleurs comme paramètres de la propriété -moz-radial-gradient. Ici, le gradient va du rouge au noir en passant par le jaune et le bleu :

```
background: -moz-radial-gradient(circle, red, yellow, blue, black);
```

# Exercice

Utilisez les instructions précédentes pour définir un gradient radial multicolore compatible avec IE, Gecko et Mozilla.

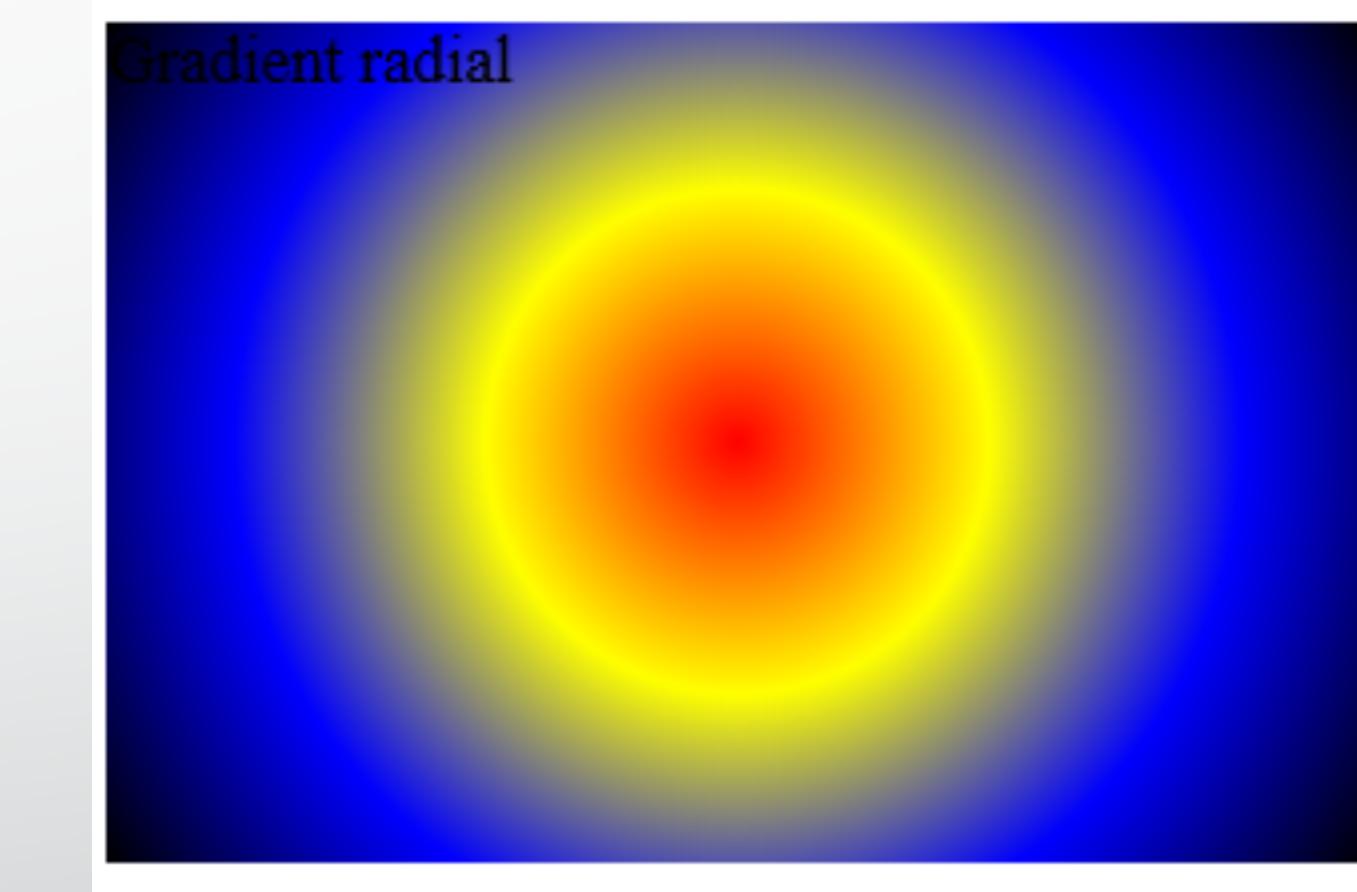
# Solution (code012.htm)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gradient radial</title>
    <link rel="stylesheet" href="gradient-radial3.css">
  </head>
  <body>
    <p class="gradient-radial">Gradient radial</p>
  </body>
</html>
```

Code CSS3

```
.gradient-radial
{
  width:300px;
  height: 200px;
  background: -webkit-gradient(radial, 50% 50%, 0, 50% 50%, 150, from(red), color-stop(.25,yellow), color-stop(.50, blue), to (black));
  background: -moz-radial-gradient(circle, red, yellow, blue, black);
  background: -ms-radial-gradient(circle, red, yellow, blue, black);
}
```

Code HTML5



# Fonctions `RGBA()` et `HSLA()`

En utilisant les fonctions `RGBA()` ou `HSLA()`, on peut appliquer un effet de transparence aux couleurs manipulées dans le code CSS3. Ces fonctions s'appliquent à n'importe quelle propriété dont la valeur est une couleur : `color`, `background-color`, `border-color`, etc.

La fonction `RGBA()` admet quatre paramètres :

`RGBA(red, green, blue, alpha)`

Où :

- `red`, `green` et `blue` sont les composantes rouge, vert et bleu de la couleur. Ces trois informations peuvent prendre 256 valeurs, codées entre 0 et 255.
- `alpha` est le degré d'opacité de la couleur. Cette information est un nombre décimal codé entre `0` (transparent) et `1` (opaque).

# Fonctions RGBA() et HSLA()

La fonction HSLA() admet également quatre paramètres :

HSLA(hue, sat, light, alpha)

Où :

- hue représente la teinte. Les valeurs autorisées pour ce paramètre sont comprises entre 0 et 360. La valeur 0(ou 360) correspond au rouge, la valeur 120 correspond au vert et la valeur 250 au bleu.
- sat représente la teinte de la couleur, entre 0% et 100%.
- light représente la luminosité de la couleur, entre 0% (noir) et 100% (blanc).
- alpha est le degré d'opacité de la couleur. Cette information est un nombre décimal codé entre 0 (transparent) et 1 (opaque).

# Fonctions RGBA() et HSLA()

À titre d'exemple, nous allons définir plusieurs éléments div contenant du texte et dont la couleur d'arrière-plan est plus ou moins transparente. Voici le code HTML5 utilisé :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Facteur Alpha avec RGBA</title>
    <link rel="stylesheet" href="alpha.css">
  </head>
  <body>
    <div id="alpha00">Texte affiché sur un fond vert avec alpha à 0%</div>
    <div id="alpha20">Texte affiché sur un fond vert avec alpha à 20%</div>
    <div id="alpha50">Texte affiché sur un fond vert avec alpha à 50%</div>
    <div id="alpha80">Texte affiché sur un fond vert avec alpha à 80%</div>
  </body>
</html>
```

Ce code se contente d'afficher quatre blocs <div> de classe alpha00, alpha20, alpha50 et alpha80.

Saisissez ce code dans le fichier code013.htm

# Fonctions RGBA() et HSLA()

Voici le code CSS associé :

```
div
{
    width: 200px;
    height:100px;
}

.alpha00
{
    background-color: rgba(0, 100, 100, 0);
}

.alpha20
{
    background-color: rgba(0, 100, 100, .2);
}

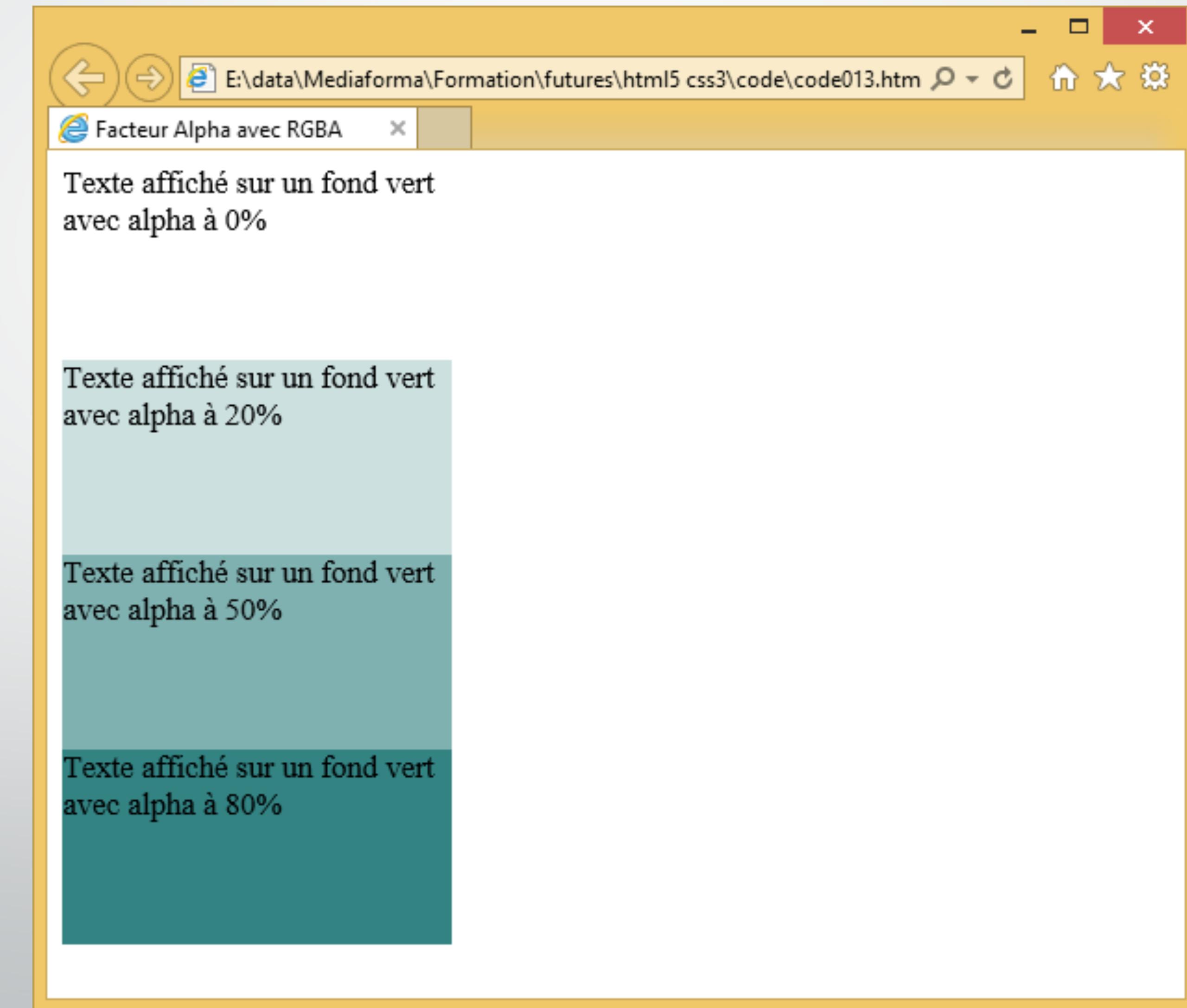
.alpha50
{
    background-color: rgba(0, 100, 100, .5);
}

.alpha80
{
    background-color: rgba(0, 100, 100, .8);
}
```

Saisissez ce code dans le fichier alpha.css

# Fonctions RGBA() et HSLA()

Voici ce que vous devriez obtenir :



# Fonctions `RGBA()` et `HSLA()`

La fonction `HSLA()` est une alternative à la fonction `RGBA()`. Le fichier *alpha.css* précédent pourrait être réécrit comme suit avec la fonction `HSLA()` :

```
div
{
    width: 200px;
    height:100px;
}

.alpha00
{
    background-color: hsla(120, 94%, 18%, 0);
}

.alpha20
{
    background-color: hsla(120, 94%, 18%, .2);
}

.alpha50
{
    background-color: hsla(120, 94%, 18%, 0.4);
}

.alpha80
{
    background-color: hsla(120, 94%, 18%, .8);
}
```

Pour choisir une couleur plus facilement, vous pouvez vous rendre sur les pages [www.colorschemer.com/online.html](http://www.colorschemer.com/online.html) (RBG) ou [www.workwithcolor.com/hsl-color-schemer-01.htm](http://www.workwithcolor.com/hsl-color-schemer-01.htm) (HSL).

# Positionner en flux

Lorsque la position d'un élément n'est pas spécifiée, il se place d'après l'ordre dans lequel il apparaît dans le code, en ligne ou en bloc selon son rendu.

Par exemple, l'élément a va tout naturellement se placer à la suite du texte qui le précède.

Cliquez [pour accéder au site de formation Mediaforma](http://www.mediaforma.com)

Cliquez pour accéder au site de formation Mediaforma

Par contre, l'élément de rendu block <h2> va s'afficher sur la ligne suivante :

Ce texte est suivi<h2> par un titre de style H2</h2>

Ce texte est suivi

**par un titre de style H2**

# Positionner en flux

Il en va de même des objets placés à l'intérieur d'un conteneur. Ici, par exemple, deux images, un texte, puis deux autres images sont affichés à l'intérieur d'un élément div. Ces éléments se succèdent en ligne. Le passage à la ligne après le mot "templum" a été provoqué par un manque de place dans le navigateur

```

```

```

```

Iam summus Pater architectus Deus hanc quam videmus mundanam  
domum, divinitatis templum augustissimum, archanae legibus  
sapientiae fabrefecerat.

```

```

```

```



Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum  
augustissimum, archanae legibus sapientiae fabrefecerat.



# Mise en page avec la propriété display

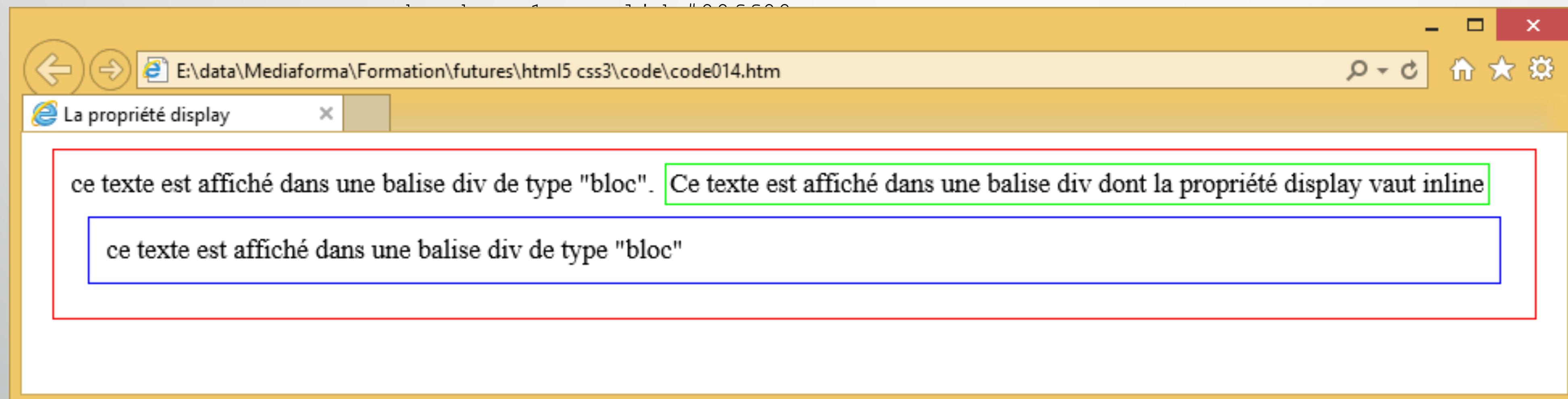
La propriété display permet de spécifier la façon dont un élément est rendu. Elle peut prendre de nombreuses valeurs. Voici les plus courantes :

- none : l'élément n'est pas rendu.
- inline : l'élément est rendu sur la même ligne que son parent.
- block : l'élément est rendu en mode bloc.
- inline-block : l'élément se comporte comme une "boîte en ligne" : il peut être dimensionné (width, height) et avoir des marges (margin, padding), mais il se place également sur la ligne de texte de son parent. Vous utiliserez -moz-inline-box pour assurer la compatibilité avec Firefox 2 et supérieur.
- list-item : l'élément est affiché sous la forme d'une liste.
- table, inline-table, table-header-group, table-footer-group, table-row, table-row-group, table-column, table-column-group, table-cell, table-caption : l'élément se comporte comme les composants d'un tableau (attention, ces valeurs sont incompatibles avec Internet Explorer).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>La propriété display</title>
    <style>
      .un
      {
        border: 1px solid #ff0000;
        margin: 10px;
        padding: 10px;
      }
      .deux
      {
```

## Premier exemple (code014.htm)

Saisissez ce code.  
Qu'obtenez-vous ?



```
ce texte est affiché dans une balise div de type "bloc".
<div class="deux" style="display: inline">Ce texte est affiché dans une balise div dont la
propriété display vaut inline</div>
  <div class="trois">ce texte est affiché dans une balise div de type "bloc"</div>
</div>
</body>
</html>
```

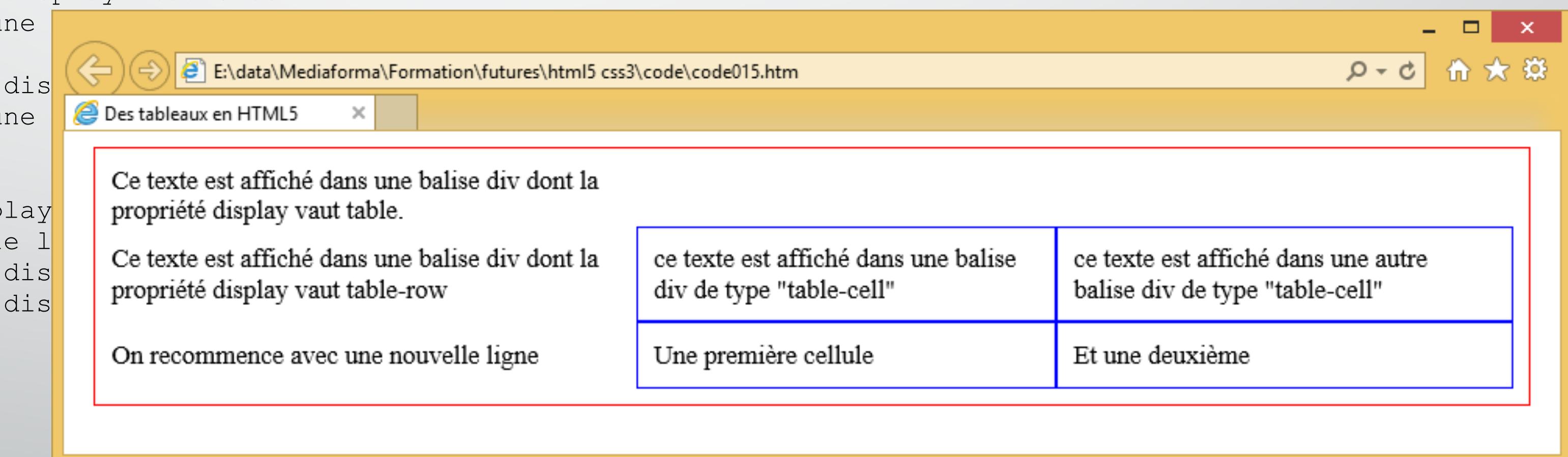
```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Des tableaux en HTML5</title>
    <style>
      .un
      {
        border: 1px solid #ff0000; margin: 10px; padding: 10px;
      }
      .deux
      {
        border: 1px solid #00ff00; margin: 2px; padding: 2px;
      }
      .trois
      {
        border: 1px solid #0000ff; margin: 10px; padding: 10px;
      }
    </style>
  </head>
  <body>
    <div class="un" style="display: table;">
      Ce texte est affiché dans une balise div dont la propriété display vaut table.
      <div class="deux" style="display: table-row">
        Ce texte est affiché dans une balise div dont la propriété display vaut table-row
        <div class="trois" style="display: table-cell">
          ce texte est affiché dans une
        </div>
        <div class="trois" style="display: table-cell">
          ce texte est affiché dans une
        </div>
      </div>
      <div class="deux" style="display: table-row">
        On recommence avec une nouvelle ligne
        <div class="trois" style="display: table-cell">
          <div class="trois" style="display: table-cell">
            ce texte est affiché dans une autre
            balise div de type "table-cell"
          </div>
        </div>
      </div>
    </body>
  </html>

```

## Deuxième exemple (code015.htm)

Saisissez ce code  
Qu'obtenez-vous ?

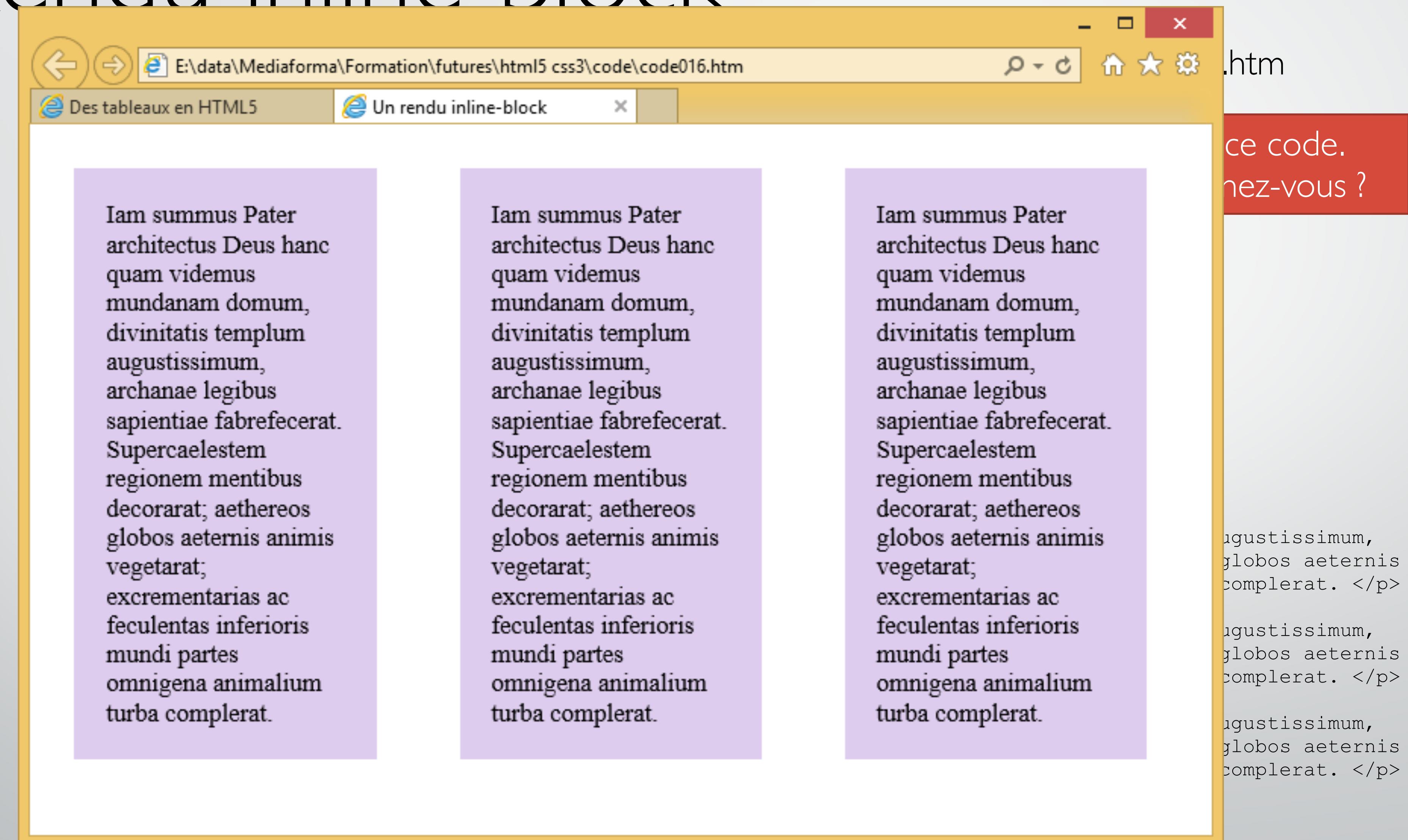


# Rendu inline-block

Le rendu inline-block profite des propriétés propres aux rendus inline et block. En effet, il permet aux éléments concernés de s'afficher côte à côté (rendu inline) et d'être dimensionnés et de recevoir des marges externes (rendu block).

Pour définir des blocs de texte de largeur fixe à l'aide de simples éléments p, rien de tel qu'un affichage inline-block.

# Rendu inline-block



# Rendu inline-block

Cette technique ne se limite pas aux éléments p.

Elle peut être étendue à tous les éléments dont le rendu par défaut est inline. Essentiellement a, b, br, code, em, font, i, img, input, s, select, small, span, strike, strong, sub, sup, textarea et u.

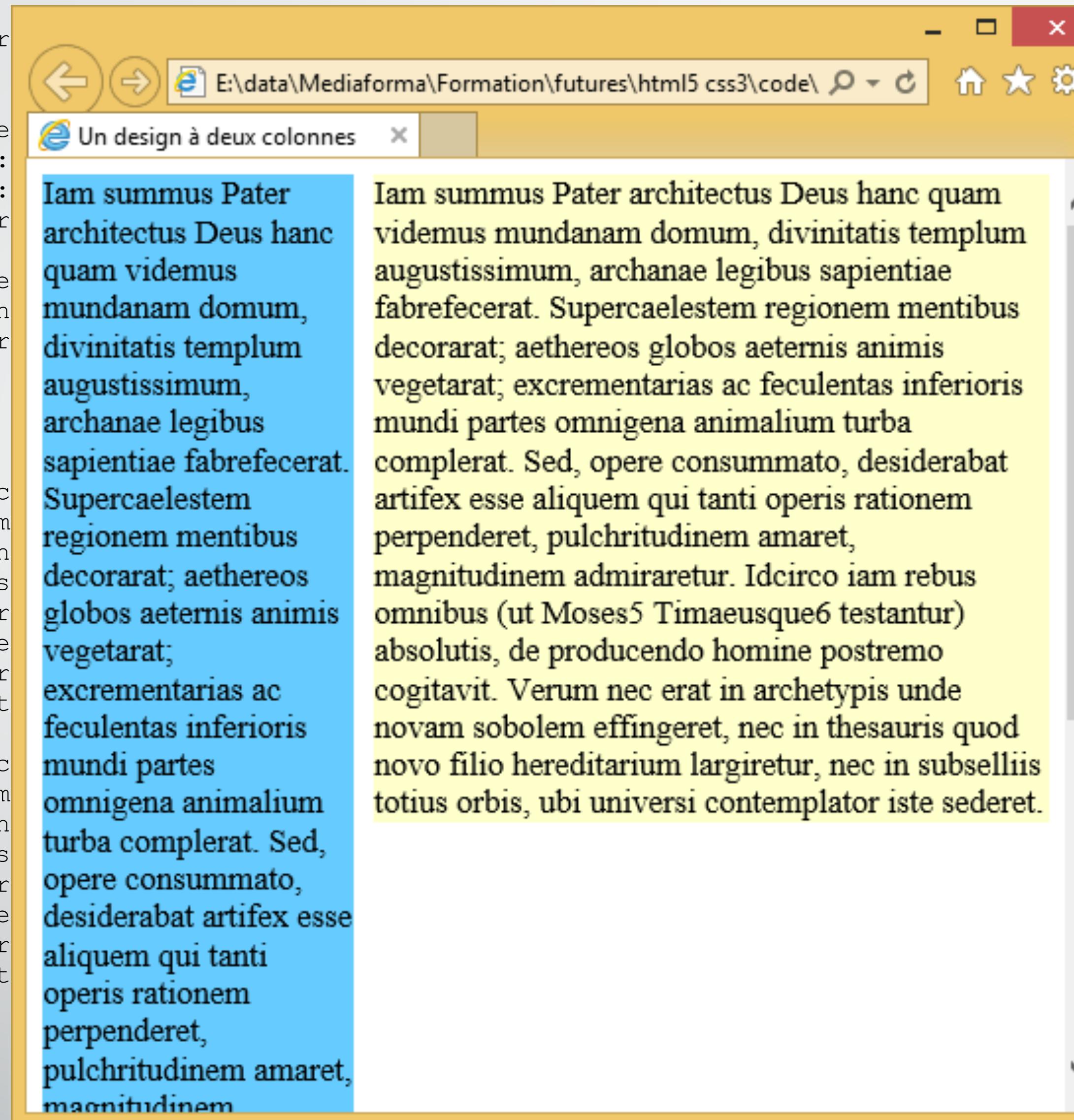
# Design à deux colonnes

Le principe consiste à définir une boîte <div> flottante (à gauche ou à droite selon les besoins) et une zone non flottante, dont la marge est légèrement supérieure à la largeur de l'élément flottant.

Dans cet exemple, nous définissons une "colonne" flottante à gauche de largeur 150 pixels, et une colonne non flottante qui occupe le reste de la page. Cette colonne a une marge gauche de 160 pixels, de façon à se détacher légèrement vers la droite de la colonne gauche.

# Design à deux colonnes

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Un design à deux colonnes</title>
    <style>
      #colonne_gauche {
        float: left;
        width: 300px;
        background-color: #f0f0ff;
      }
      #colonne_droite {
        margin-left: 300px;
        background-color: #fffff0;
      }
    </style>
  </head>
  <body>
    <div id="colonne_gauche">
      Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum augustissimum, archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat; aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis rationem perpendere, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut Moses5 Timaeus6que testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec in subselliis totius orbis, ubi universi contemplator iste sederet.
    </div>
    <div id="colonne_droite">
      Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum augustissimum, archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat; aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis rationem perpendere, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut Moses5 Timaeus6que testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec in subselliis totius orbis, ubi universi contemplator iste sederet.
    </div>
  </body>
</html>
```



code016.htm

Saisissez ce code.  
Qu'obtenez-vous ?

s templum augustissimum, archanae os globos aeternis animis vegetarat; rat. Sed, opere consummato, em amaret, magnitudinem admiraretur.

hominem postremo cogitavit. Verum io hereditarium largiretur, nec in

s templum augustissimum, archanae os globos aeternis animis vegetarat; rat. Sed, opere consummato, em amaret, magnitudinem admiraretur.

hominem postremo cogitavit. Verum io hereditarium largiretur, nec in

# Design à 3 colonnes dont 2 flottantes

Pour réaliser un design à trois colonnes dont deux sont flottantes, ancrées sur les bords gauche et droit de la page, vous utiliserez trois éléments div :

- un premier de largeur fixe (width), ancré sur le bord gauche de la page (float: left) ;
- un deuxième de largeur fixe (width), ancré sur le bord droit de la page (float: right) ;
- un troisième qui occupera librement l'espace situé entre les deux autres.

## Exercice

Ecrivez le code pour réaliser un design à 3 colonnes :

- Colonne 1 : 150 px flottante à gauche
- Colonne 2 : 300 px flottante à droite
- Colonne 3 : La place restante

# Solution

```
<!DOCTYPE html>
<html>
    <head>
```

E:\data\Mediaforma\Formation\futures\html5 css3\code\code018.htm

Un design fluide à 3 colonnes

```
Iam summus Pater architectus Deus hanc mundanam domum, divinitatis templum augustissimum, quam videmus mundanam domum, divinitatis archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat; aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis rationem perpendere, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut Moses5 Timaeusque6 testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec in subselliis totius orbis, ubi universi contemplator iste sederet.
```

code018.htm

chanae legibus sapientiae  
arias ac feculentas  
quem qui tanti operis  
aeusque6 testantur)  
nec in thesauris quod

anae legibus sapientiae  
arias ac feculentas  
quem qui tanti operis  
aeusque6 testantur)  
nec in thesauris quod

chanae legibus sapientiae  
arias ac feculentas  
quem qui tanti operis  
aeusque6 testantur)  
nec in thesauris quod

# Design à 3 colonnes de même hauteur

Pour obtenir des colonnes de même hauteur, et ce, indépendamment de leur contenu, vous utiliserez la technique dite des "colonnes factices", qui consiste à :

1. Insérer les boîtes des colonnes dans un conteneur commun.
2. Définir un arrière-plan graphique qui se répète en hauteur dans le conteneur, de façon à tracer un séparateur entre les colonnes.
3. Modifier les marges de la colonne centrale pour obtenir un léger décalage de part et d'autre des colonnes séparatrices.
4. Conférer au conteneur un nouveau contexte de formatage *via* la déclaration overflow: hidden, ce qui permet à ce conteneur d'englober ses enfants flottants.

A titre d'exemple, nous allons définir une colonne gauche de largeur 160 pixels, une colonne droite de largeur 300 pixels et une colonne centrale qui occupe le reste de l'espace disponible, soit  $800 - 160 - 300 = 340$  pixels.

L'arrière-plan est une image de 800 pixels sur 3 pixels :



E:\data\Mediaforma\Formation\futures\html5 css3\code\code019.htm

Un design à 3 colonnes de ...

Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum augustissimum, archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat; aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis rationem perpendere, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut Moses<sup>5</sup> Timaeusque<sup>6</sup> testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec in subselliis totius orbis, ubi universi contemplator iste sederet.

Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum augustissimum, archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat; aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis rationem perpendere, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut Moses<sup>5</sup> Timaeusque<sup>6</sup> testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec in subselliis totius orbis, ubi universi contemplator iste sederet.

# Multicolonnage CSS3

Pour définir un multicolonnage CSS3, il suffit de fixer la largeur des colonnes et l'espace entre deux colonnes. Voici le code HTML5/CSS3 utilisé :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Multicolonnage</title>
    <style>
      .multi-col
      {
        column-width: 150px;
        -moz-column-width: 150px;
        -webkit-column-width: 150px;
        -moz-column-gap: 20px;
        -webkit-column-gap: 20px;
      }
    </style>
  </head>
  <body>
    <div class="multi-col">
      Iam summus Pater architectus Deus hanc quam videmus mundanam domum, divinitatis templum
      augustissimum, archanae legibus sapientiae fabrefecerat. Supercaelestem regionem mentibus decorarat;
      aethereos globos aeternis animis vegetarat; excrementarias ac feculentas inferioris mundi partes omnigena
      animalium turba complerat. Sed, opere consummato, desiderabat artifex esse aliquem qui tanti operis
      rationem perpenderet, pulchritudinem amaret, magnitudinem admiraretur. Idcirco iam rebus omnibus (ut
      Moses5 Timaeus6 testantur) absolutis, de producendo homine postremo cogitavit. Verum nec erat in
      archetypis unde novam sobolem effingeret, nec in thesauris quod novo filio hereditarium largiretur, nec
      in subselliis totius orbis, ubi universi contemplator iste sederet.</p>
    </div>
  </body>
</html>
```

Iam summus Pater  
architectus Deus hanc  
quam videmus mundanam  
domum, divinitatis  
templum augustissimum,  
archanae legibus sapientiae  
fabrefecerat.  
  
Supercaelestem regionem  
mentibus decorarat;  
aethereos globos aeternis  
animis vegetarat;

excrementarias ac  
feculentas inferioris mundi  
partes omnigena animalium  
turba complerat. Sed, opere  
consummato, desiderabat  
artifex esse aliquem qui  
tanti operis rationem  
perpendret,  
pulchritudinem amaret,  
magnitudinem admiraretur.  
Idcirco iam rebus omnibus

(ut Moses<sup>5</sup> Timaeus<sup>6</sup>  
testantur) absolutis, de  
producendo homine  
postremo cogitavit. Verum  
nec erat in archetypis unde  
novam sobolem effingeret,  
nec in thesauris quod novo  
filio hereditarium  
largiretur, nec in subselliis  
totius orbis, ubi universi  
contemplator iste sederet.

Iam summus Pater  
architectus Deus hanc  
quam videmus mundanam  
domum, divinitatis  
templum augustissimum,  
archanae legibus  
sapientiae fabrefecerat.

Supercaelestem regionem  
mentibus decorarat;  
aethereos globos aeternis  
animis vegetarat;  
excrementarias ac  
feculentas inferioris  
mundi partes omnigena

animalium turba  
complerat. Sed, opere  
consummato, desiderabat  
artifex esse aliquem qui  
tanti operis rationem  
perpendret,  
pulchritudinem amaret,

magnitudinem  
admiraretur. Idcirco iam  
rebus omnibus (ut Moses<sup>5</sup>  
Timaeus<sup>6</sup> testantur)  
absolutis, de producendo  
homine postremo  
cogitavit. Verum nec erat  
in archetypis unde novam  
sobolem effingeret, nec in  
thesauris quod novo filio  
hereditarium largiretur,  
nec in subselliis totius  
orbis, ubi universi  
contemplator iste sederet.

# Multicolonnage CSS3

Si le nombre de colonnes doit rester fixe, utilisez les propriétés CSS3 column-count, -moz-column-count et -webkit-column-count. Par exemple, pour définir trois colonnes, quelle que soit la largeur de la fenêtre, ajoutez les deux lignes suivantes dans la classe multi-col :

```
column-count: 3;  
-moz-column-count: 3;  
-webkit-column-count: 3;
```

## Exercice

Modifiez le code précédent pour que le nombre de colonne soit fixe et égal à 4.

# Solution

Modifiez le style .multi-col comme ceci :

```
<style>
  .multi-col
  {
    column-width: 150px;
    -moz-column-width: 150px;
    -webkit-column-width: 150px;
    -moz-column-gap: 20px;
    -webkit-column-gap: 20px;
    column-count: 4;
    -moz-column-count: 4;
    -webkit-column-count: 4;
  }
</style>
```

# Multicolonnage CSS3

Si nécessaire, vous pouvez également insérer un trait séparateur entre les colonnes en utilisant les propriétés CSS3 column-rule, -moz-column-rule et -webkit-column-rule dont voici la syntaxe :

```
column-rule: largeur style couleur;  
-moz-column-rule: largeur style couleur;  
-webkit-column-rule: largeur style couleur;
```

Où :

- largeur est la largeur du trait en pixels.
- style est le style du trait. Il peut prendre l'une des valeurs suivantes : dotted, dashed, solid, insert, double, groove, ridge ou outset.
- couleur est la couleur du trait.

Par exemple, pour définir un trait séparateur continu noir et d'épaisseur 1 pixel, vous utiliserez les deux propriétés suivantes :

```
column-rule: 1px solid black;  
-moz-column-rule: 1px solid black;  
-webkit-column-rule: 1px solid black;
```

Entraînez-vous à utiliser ces instructions dans le code précédent.

# Multicolonnage CSS3

Pour terminer, sachez qu'il est possible de définir la couleur d'arrière-plan du texte avec la propriété background et l'alignement du texte dans les colonnes avec la propriété text-align. Par exemple, pour affecter un arrière-plan gris et pour justifier le texte dans les colonnes, servez-vous des propriétés suivantes :

```
background:#ccc;  
text-align:justify;
```

# Exercice

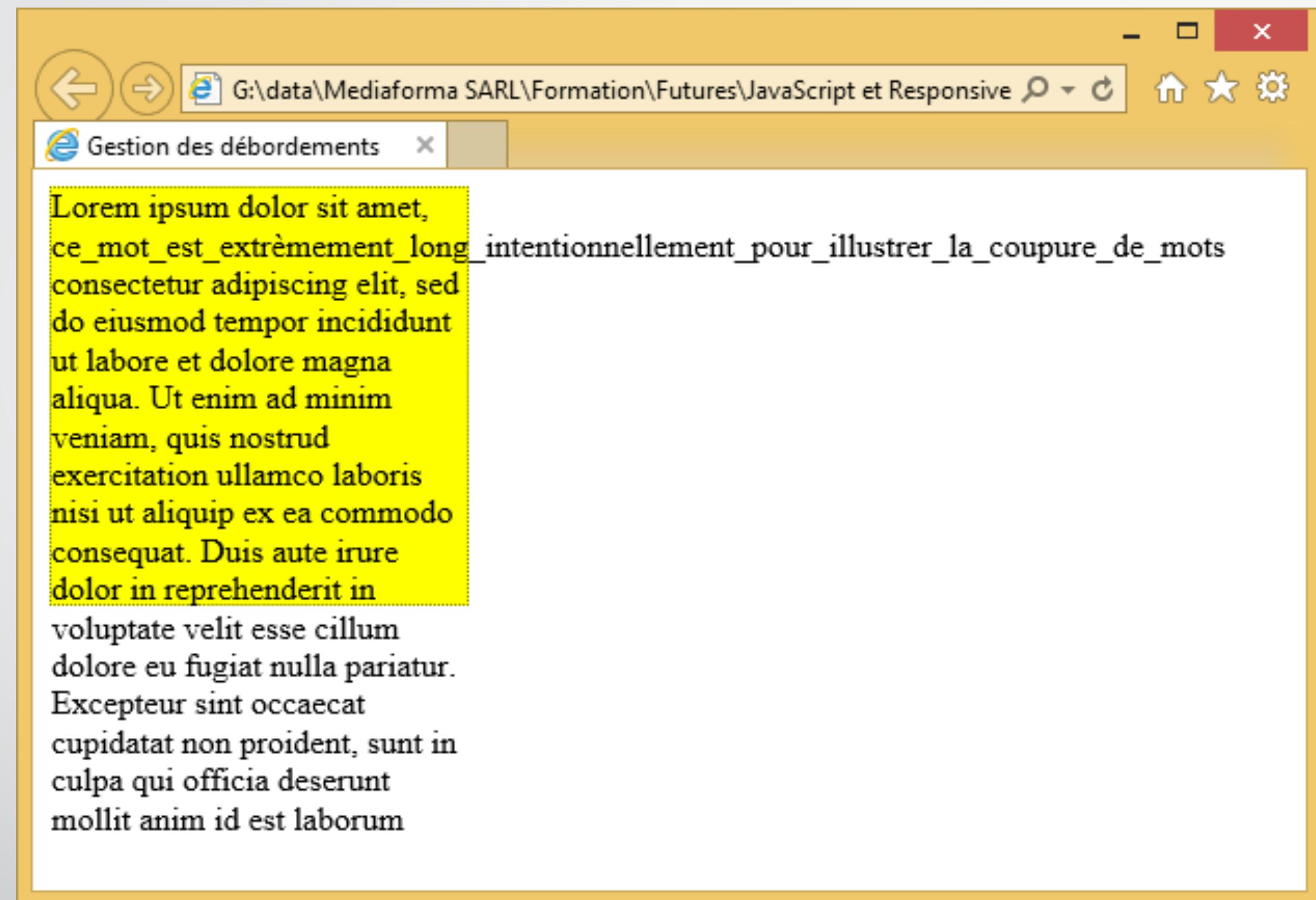
En utilisant le code HTML précédent, définissez un multicolonnage fixe sur trois colonnes séparées entre elles par un trait de 1 pixel, justifiées et d'arrière-plan #ccc.

## Gestion des débordements de contenus

Qu'il s'agisse de texte, d'images ou de tout autre type de contenu, les éléments placés dans un conteneur peuvent déborder de ce conteneur. Vous devez alors gérer les débordements, en particulier si le contenu peut provenir de sources que vous ne contrôlez pas.

## Examinez le code suivant :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Gestion des débordements</title>
        <style>
            #mabox {
                width: 200px;
                height: 200px;
                border: 1px dotted black;
                background-color: yellow;
            }
        </style>
    </head>
    <body>
        <div id="mabox">
            Lorem ipsum dolor sit amet,
            ce_mot_est_extrêmement_long_intentionnellement_pour_illustrer_la_coupure_de_mots consectetur
            adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
            ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
            consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu
            fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
            officia deserunt mollit anim id est laborum
        </div>
    </body>
</html>
```



Manifestement, la balise <div> conteneur n'est pas assez grande pour héberger tout le texte. Etant donné que ses dimensions ont été fixées "en dur" (width et height), le texte déborde.

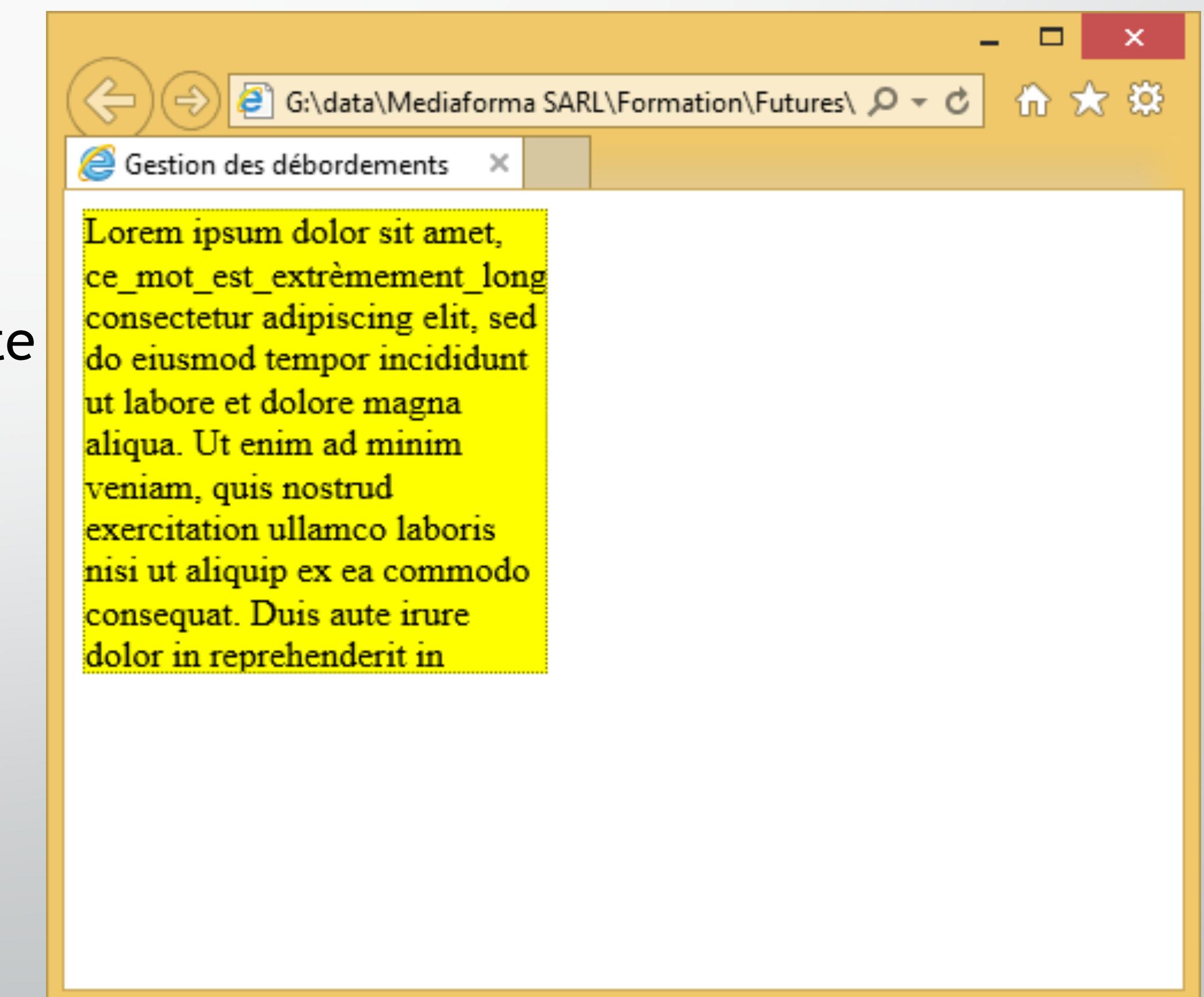
Plusieurs technique vont permettre de gérer les débordements.

## Première technique : overflow: hidden

Tout ce qui dépasse du conteneur est affacé avec cette instruction CSS :

```
#mabox {  
    overflow: hidden;  
}
```

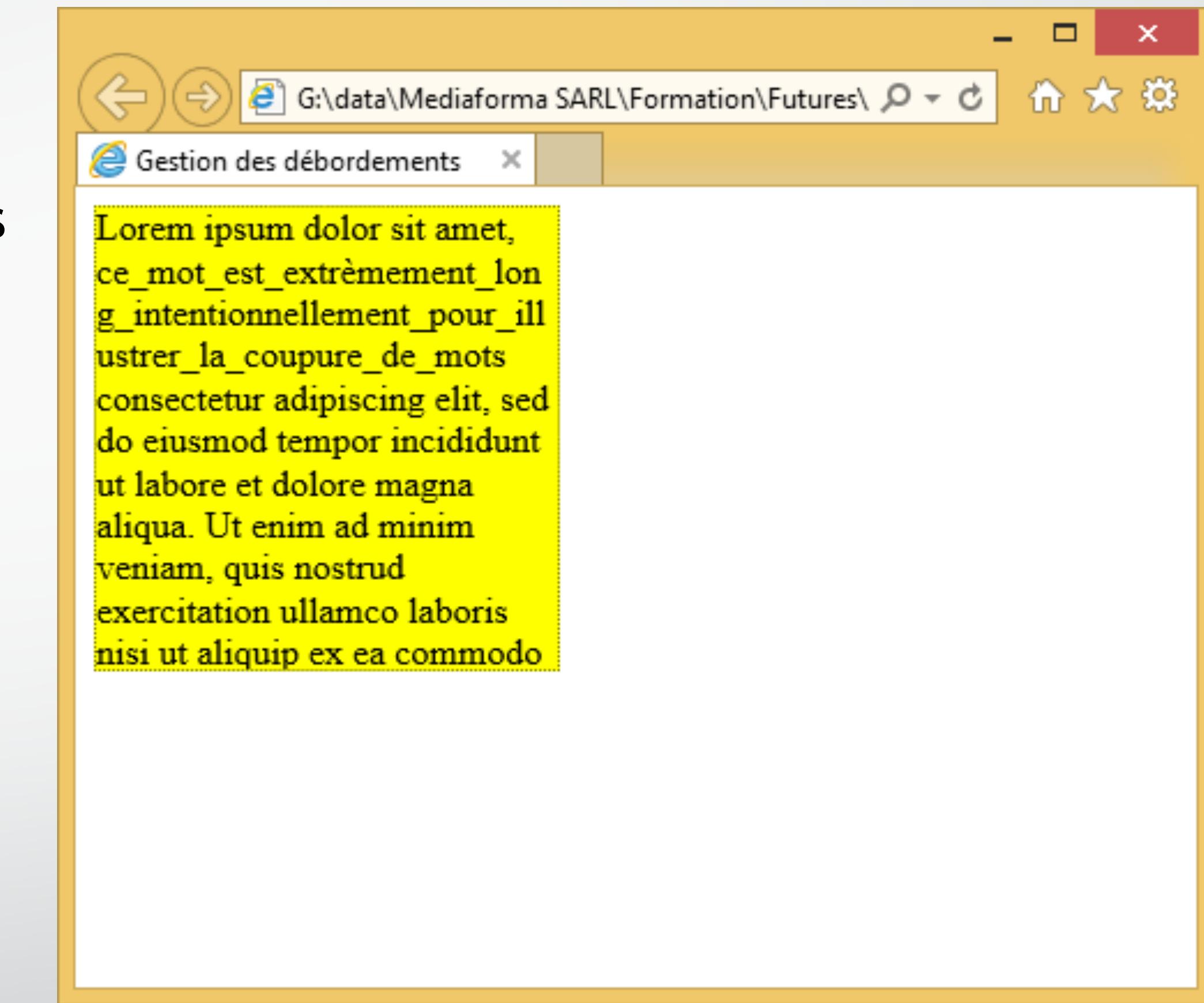
Voici le résultat :



## Deuxième technique : passage à la ligne des mots longs

Lorsqu'elle est initialisée à break-word, la propriété CSS overflow-wrap permet de découper proprement les mots trop longs pour le conteneur :

```
#mabox {  
    overflow: hidden;  
    word-wrap: break-word;
```



Comme vous pouvez le constater, le mot long n'est pas partiellement dissimulé par la propriété overflow: hidden;. Au contraire, le mot est renvoyé à la ligne autant de fois que nécessaire pour s'afficher dans son intégralité.

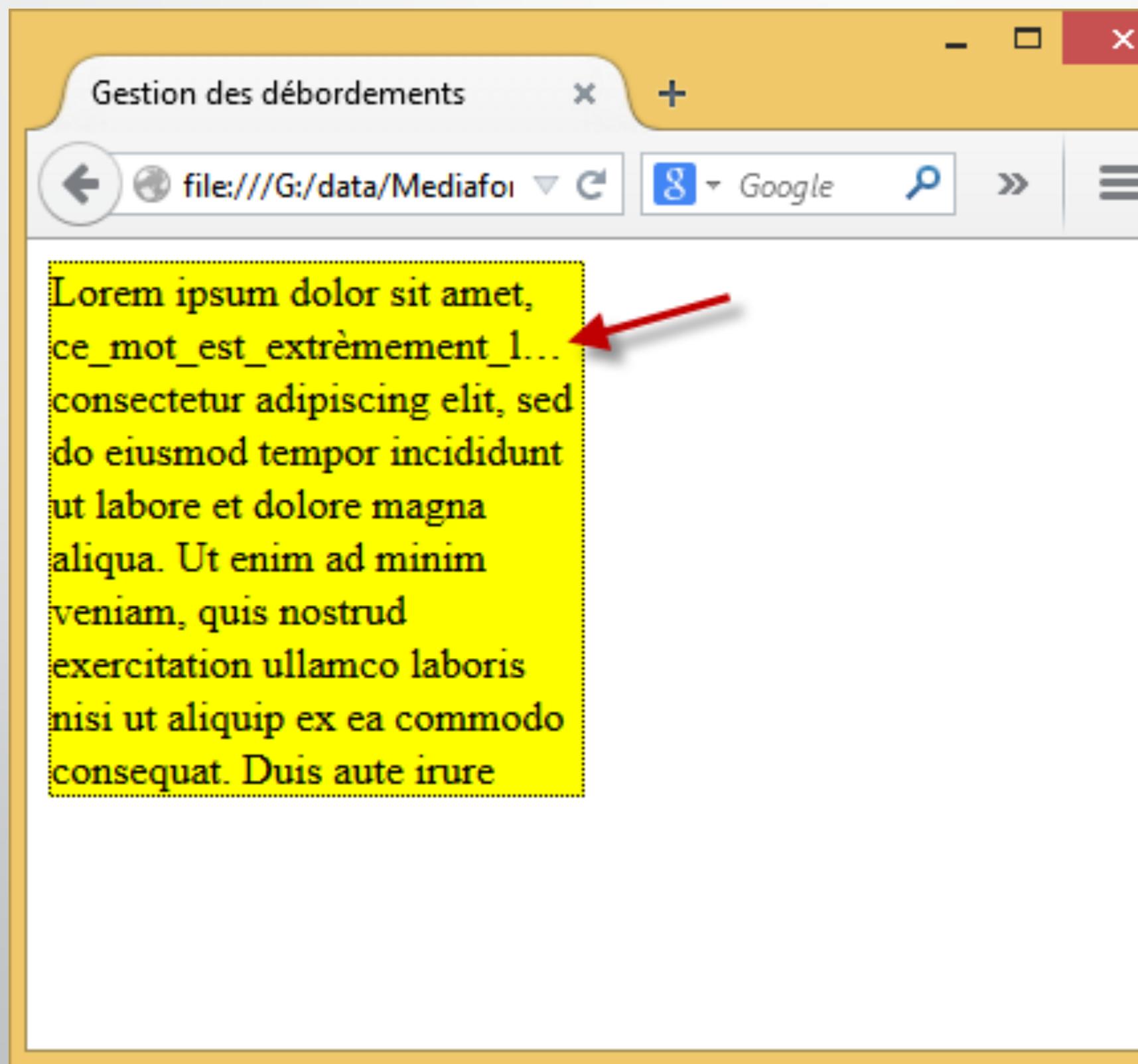
## Troisième technique : points de suspension

En affectant la valeur **ellipsis** à la propriété **text-overflow**, des points de suspension seront affichés chaque fois qu'un texte est rogné :

```
#mabox {  
    overflow: hidden;  
    text-overflow: ellipsis;
```

### Attention

Cette propriété n'est pas supportée par tous les navigateurs.



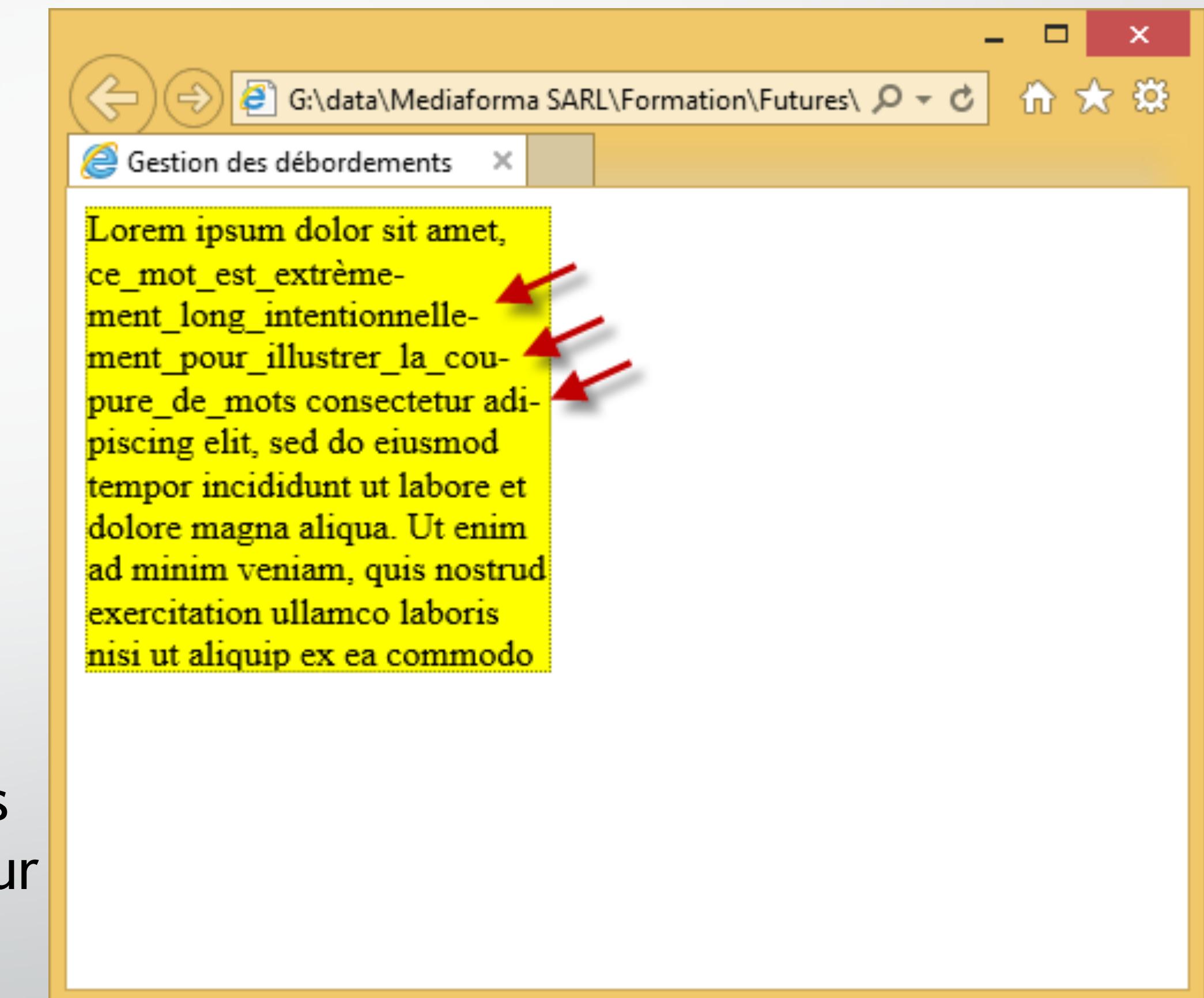
## Quatrième technique : césure de mots

Affectez la valeur **auto** à la propriété **hyphens** pour activer la césure de mots :

```
#mabox {  
    overflow: hidden;  
    hyphens: auto;  
    -ms-hyphens: auto;  
    -moz-hyphens: auto;  
    -webkit-hyphens: auto;
```

### Attention

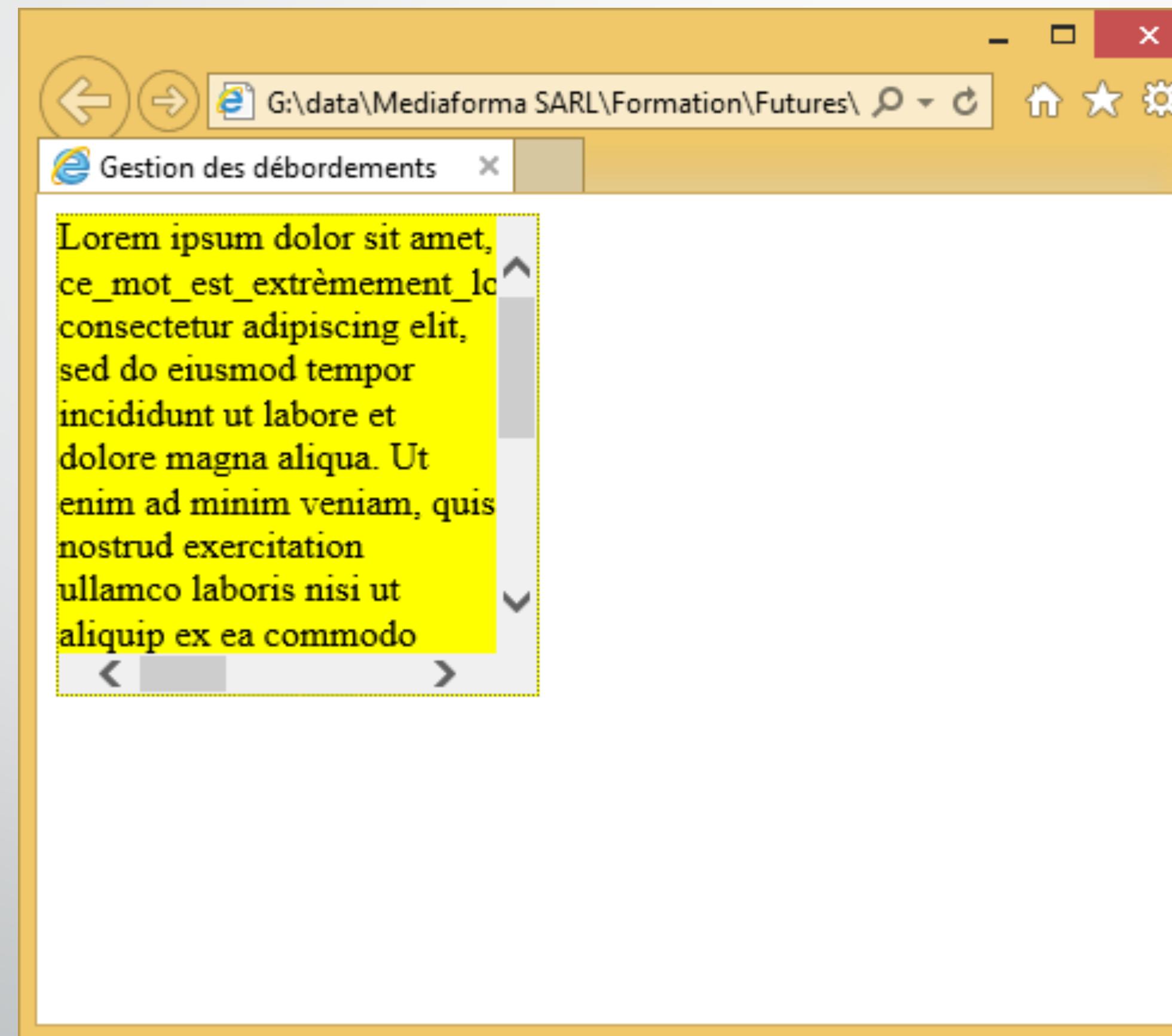
Cette propriété n'est pas encore supportée par tous les navigateurs. Pensez à utiliser les préfixes habituels pour améliorer sa compatibilité.



## Cinquième technique : barres de défilement

En affectant la propriété **auto** à la propriété **overflow**, des barres de défilement permettent d'accéder au contenu qui déborder du conteneur :

```
#mabox {  
    overflow: auto;
```



# Composants graphiques

## Images fluides

Pour que la taille d'une image soit automatiquement redimensionnée en fonction de l'espace dont elle dispose, il suffit de définir sa largeur maximale à 100% de son conteneur. Voici un exemple de code :

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8">
        <title>Gestion des débordements</title>
    </head>
    <body>
        <br>
        Ipsam vero urbem Byzantiorum fuisse refertissimam atque ornatissimam signis quis ignorat? Quae illi, exhausti sumptibus bellisque maximis, cum omnis Mithridaticos impetus totumque Pontum armatum affervescentem in Asiam atque erumpentem, ore repulsum et cervicibus interclusum suis sustinerent, tum, inquam, Byzantii et postea signa illa et reliqua urbis ornanemta sanctissime custodita tenuerunt.
    </body>
</html>
```

Redimensionnez la fenêtre. Comme vous pouvez le voir, l'image se redimensionne automatiquement en fonction des dimensions de la fenêtre :



Ipsam vero urbem Byzantiorum fuisse refertissimam atque ornatissimam signis quis ignorat? Quae illi, exhausti sumptibus bellisque maximis, cum omnis Mithridaticos impetus totumque Pontum armatum affervescentem in Asiam atque erumpentem, ore repulsum et cervicibus interclusum suis sustinerent, tum, inquam, Byzantii et postea signa illa et reliqua urbis ornamenti sanctissime custodita tenuerunt.

# Arrière-plan de page fluide

Pour définir un arrière-plan de page fluide, la technique est un peu différente :

```
<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title>Arrière-plan de la page fluide</title>

        <style>

            body{

                background-image:url(foret.jpg);
                background-attachment:fixed;
                background-repeat:no-repeat;
                background-position:center center;
                background-size:cover;
            }

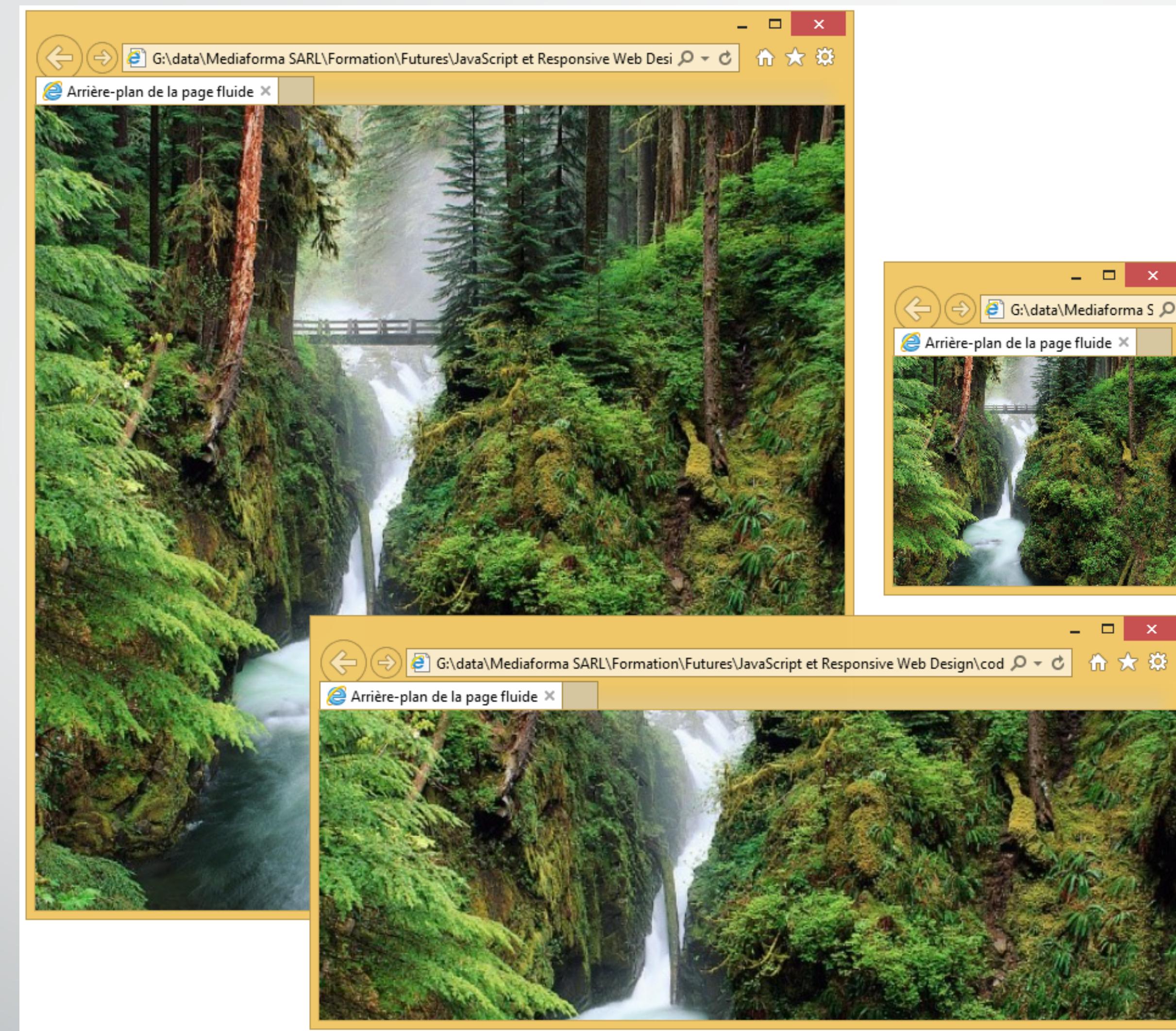
        </style>

    </head>

    <body>

    </body>

</html>
```



# Affichage fluide, Grid Layout et FlexBox

## Principe d'un affichage fluide

On parle d'**affichage fluide** lorsque le contenu et la disposition sont élastiques. Pour arriver à ce résultat, on utilise des blocs dont la largeur est exprimée en pourcentages de la largeur totale de la fenêtre (ordinateur) ou de l'écran (mobile).

## Exercice

Définissez quelques lignes de HTML5 et de CSS3 pour obtenir la mise en page suivante :



Voici la largeur des différents blocs :

- Couleur jaune : 20% ;
- Couleur bleue : 80% ;
- Couleurs rouget et verte : 100%.

# Solution

Le code HTML5 est très simple : il met en place quatre zones distinctes : **header**, **article**, **aside** et **footer** :

```
● ● ●

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8 <main>
9     <header>
10         <h1>En-tête du document</h1>
11         Texte de l'en-tête
12     </header>
13     <section>
14         <article>
15             <h2>Premier article</h2>
16             <p>Texte du premier article</p>
17         </article>
18         <aside>
19             Texte affiché dans la partie droite de la page avec la balise &lt;aside&gt;
20         </aside>
21     </section>
22     <footer>
23         <p>Copyright et e-mail du Webmaster</p>
24     </footer>
25 </main>
26 </body>
27 </html>
```

```
● ● ●
1 :root {
2   --hauteur-section: 350px;
3 }
4 header {
5   background-color: red;
6 }
7
8 p {
9   margin: 0;
10}
11
12 section {
13   position: relative;
14 }
15
16 section article {
17   background-color: aqua;
18   display: inline-block;
19   width: 80%;
20   height: var(--hauteur-section);
21   position: absolute;
22   top: 0;
23   left: 0;
24 }
25
26 section aside {
27   background-color: yellow;
28   display: inline-block;
29   width: 20%;
30   height: var(--hauteur-section);
31   position: absolute;
32   top: 0;
33   right: 0;
34 }
35
36 footer {
37   background-color: greenyellow;
38   position: relative;
39   top: var(--hauteur-section);
40 }
```

Le code CSS3 est également très simple. Il indique que les quatre zones seront affichées en mode block, puis il définit les caractéristiques de chaque zone :

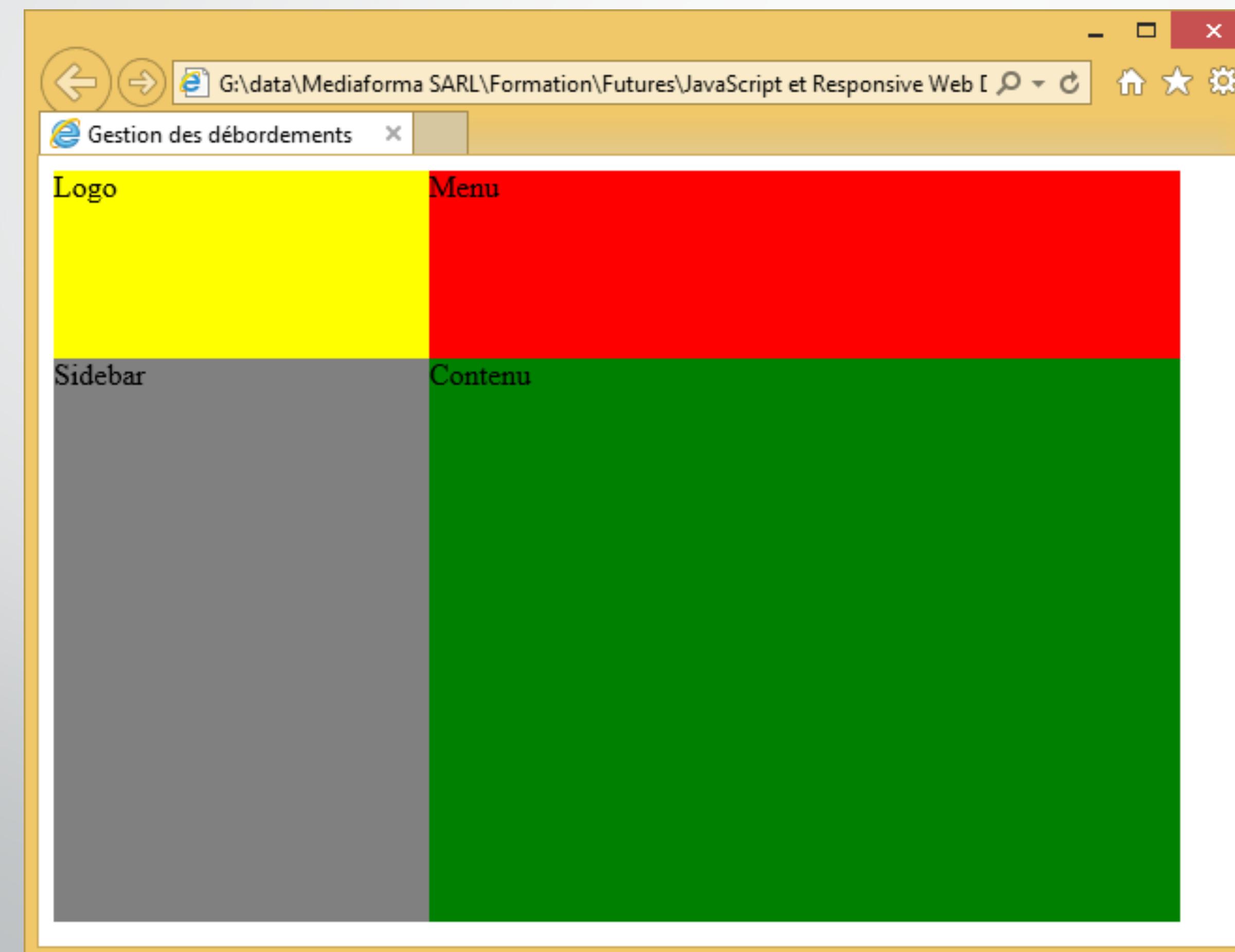
Vérifiez que l'affichage s'adapte bien à l'espace horizontal disponible en redimensionnant la fenêtre du navigateur.

## Remarque

Il est possible de définir les marges extérieures (**margin**) et intérieures (**padding**) d'un élément HTML5 quelconque (**img**, **div**, **nav**, etc.) sous la forme d'un pourcentage de leur conteneur pour leur conférer un caractère responsive.

## Principe d'un affichage Grid Layout

Le principe de fonctionnement du Grid layout consiste à diviser l'écran en plusieurs zones, comme dans un tableau, à ceci près que le découpage se fait en CSS et non en HTML. Voici un exemple de découpage Grid layout (propre à Internet Explorer et Microsoft Edge) :



Par exemple, pour découper la page en quatre zones (logo, menu, sidebar et contenu) comme dans la figure précédente, vous utiliserez ce code :

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8">
        <title>Gestion des débordements</title>
        <style>
            body {display: grid; grid-template-columns: 200px 400px; grid-template-rows: 100px 300px;}

            header, menu, aside, article {display: block; margin: 0; padding: 0; }

            header {grid-column: 1; grid-row: 1; background-color: yellow;}

            menu {grid-column: 2; grid-row: 1; background-color: red;}

            aside {grid-column: 1; grid-row: 2; background-color: grey;}

            article {grid-column: 2; grid-row: 2; background-color: green;}
        </style>
    </head>

    <body>
        <header>Logo</header>
        <menu>Menu</menu>
        <aside>
            Sidebar
        </aside>
        <article>
            Contenu
        </article>
    </body>

</html>
```

## Principe de la grille flexible (flexbox)

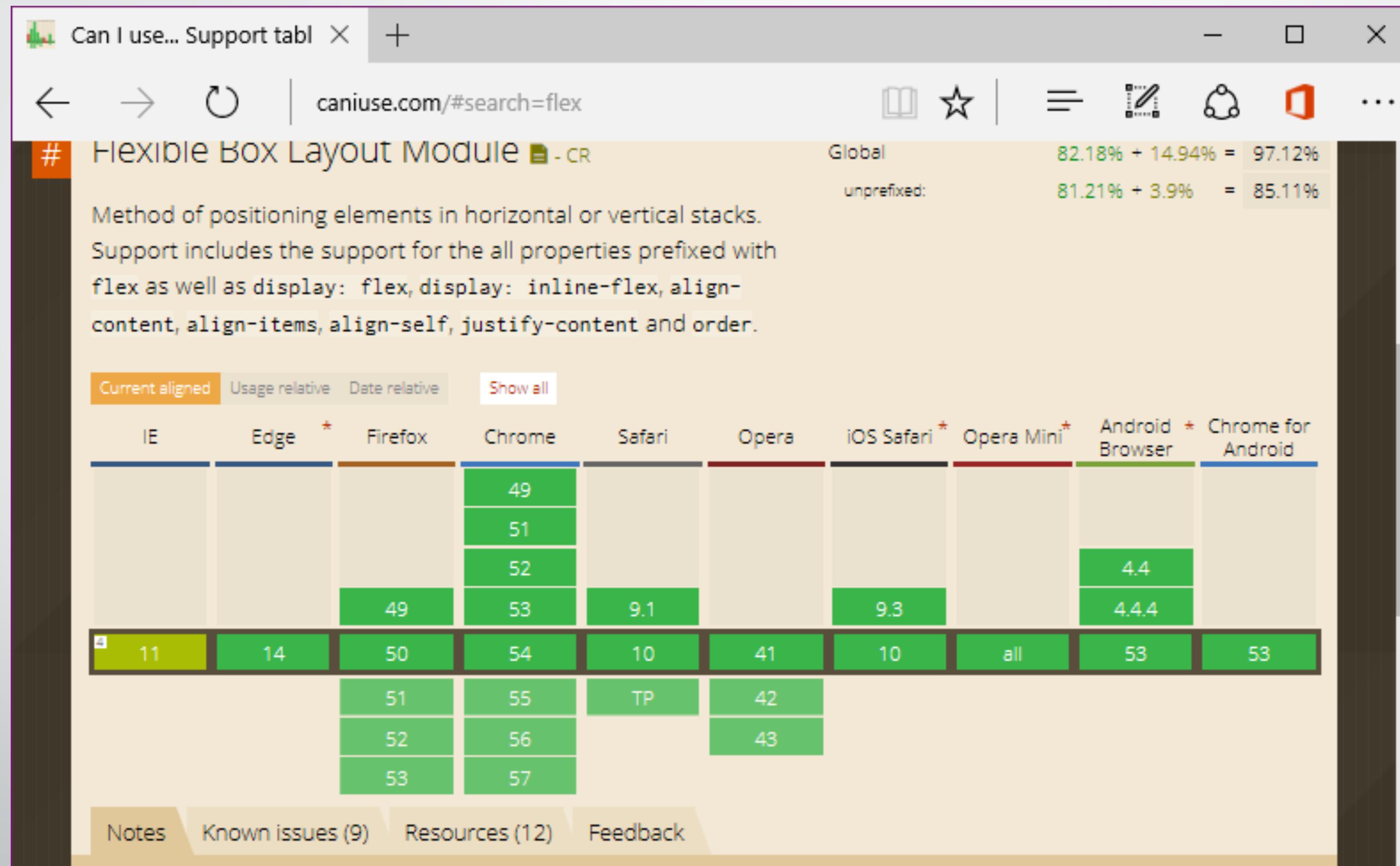
L'affichage **Flexbox** est un nouveau type de rendu défini dans la propriété CSS **display** :

```
display: flex;
```

Il permet de gérer la mise en page de plusieurs éléments enfants dans un élément parent. Avec un affichage flexbox, vous pouvez entre autres :

- mettre en place plusieurs éléments sur une ligne sans vous soucier de leur taille et/ou de leurs marges ;
- modifier le sens de lecture vertical-horizontal ;
- aligner les éléments à gauche, à droite ou au centre du parent ;
- modifier l'ordre d'affichage des éléments enfants.

Le mode d'affichage flex est utilisable sur tous les navigateurs récents. Pour vous assurer de sa compatibilité avec votre navigateur, allez sur la page <http://caniuse.com/#search=flex> :



## Affichage flexbox – display: flex

Pour mettre en place un affichage flexbox, commencez par définir le mode d'affichage **flex** pour le conteneur :

```
.conteneur { display: flex; }
```

## En ligne ou en colonne – flex-flow

Si vous voulez que les enfants s'affichent en ligne, affectez la valeur **row** à la propriété **flex-flow** du conteneur :

```
.conteneur {  
    display: flex;  
    flex-flow: row;  
}
```

Si vous voulez que les enfants s'affichent en colonne, affectez la valeur **column** à la propriété **flex-flow** du conteneur :

```
.conteneur {  
    display: flex;  
    flex-flow: column;  
}
```

## Direction de l'affichage – flex-direction

La propriété **flex-direction** définit l'ordre d'affichage des enfants dans le parent. Elle peut prendre les valeurs suivantes :

Valeur de flex-direction	Signification
row	de gauche à droite
row-reverse	de droite à gauche
column	de haut en bas
column-reverse	de bas en haut

## Wrapping des enfants – flex-wrap

Par défaut, les enfants essayeront de s'afficher sur une même ligne (ou une même colonne). Vous pouvez modifier ce comportement en agissant sur la propriété **flex-wrap** :

Valeur de flex-wrap	Signification
<code>nowrap</code>	Sur une seule ligne ou une seule colonne
<code>wrap</code>	Sur plusieurs lignes (ou plusieurs colonnes) si nécessaire
<code>wrap-reverse</code>	Sur plusieurs lignes (ou plusieurs colonnes) si nécessaire, dans l'ordre inverse d'affichage

## Alignement des enfants sur l'axe principal – justify-content

La propriété `justify-content` définit l'alignement des enfants sur l'axe principal (horizontal ou vertical selon la valeur affectée à la propriété `flex-flow`). Elle peut prendre les valeurs suivantes :

Valeur de <code>justify-content</code>	Signification
<code>flex-start</code>	Alignment au début du conteneur
<code>flex-end</code>	Alignment à la fin du conteneur
<code>center</code>	Centré dans le conteneur
<code>space-between</code>	Espacement entre les enfants
<code>space-around</code>	Espacement avant, entre et après les enfants

## Alignement des enfants sur l'axe secondaire– align-items

La propriété align-items définit l'alignement des enfants sur l'axe secondaire (l'axe perpendiculaire à l'axe principal). Elle peut prendre les valeurs suivantes :

Valeur de align-items	Signification
<b>flex-start</b>	Alignement au début du conteneur
<b>flex-end</b>	Alignement à la fin du conteneur
<b>center</b>	Centré dans le conteneur
<b>stretch</b>	Etiré pour occuper toute la dimension de l'axe secondaire
<b>baseline</b>	Aligné sur la baseline

## Alignment des enfants sur l'axe secondaire– align-content

La propriété align-content définit l'alignment des enfants sur l'axe secondaire (l'axe perpendiculaire à l'axe principal). Elle n'a aucun effet lorsque les enfants occupent une seule ligne. Cette propriété peut prendre les valeurs suivantes :

Valeur de align-items	Signification
<b>flex-start</b>	Alignment au début du conteneur
<b>flex-end</b>	Alignment à la fin du conteneur
<b>center</b>	Centré dans le conteneur
<b>stretch</b>	Etiré pour occuper toute la dimension de l'axe secondaire
<b>space-between</b>	Espacement entre les enfants
<b>space-around</b>	Espacement avant, entre et après les enfants

## Ordre d'affichage d'un enfant - order

Par défaut, les enfants s'affichent dans l'ordre où ils apparaissent dans le code.

Cependant, vous pouvez modifier cet ordre en définissant la propriété **order** dans un des enfants :

```
#enfant3 { order: 5; }
```

## Taille des enfants – flex-grow

La propriété **flex-grow** définit la taille des éléments affichés sur une même ligne (ou une même colonne). Si tous les éléments d'une même ligne ont une propriété **flex-grow** initialisée à 1, ils auront la même largeur. Si l'un d'entre eux a une propriété **flex-grow** initialisée à 2 et les autres à 1, il aura une largeur deux fois plus grande que les autres éléments.

## Rétrécissement des enfants – flex-shrink

La propriété **flex-shrink** indique si un élément peut être rétréci.

## Taille d'un élément – flex-basis

La propriété **flex-basis** définit la taille par défaut d'un élément avant que l'espace restant ne soit distribué :

```
#enfant2 { flex-basis: 200px; }
```

ou :

```
#enfant2 { flex-basis: auto; // Valeur par défaut}
```

## Alignment d'un enfant – align-self

La propriété **align-self** permet de définir l'alignment d'un enfant. Elle peut prendre les valeurs suivantes : **auto**, **flex-start**, **flex-end**, **center**, **baseline** ou **stretch**.

Et maintenant , vous allez vous entraîner pour comprendre le fonctionnement des **flexbox**.

Commencez par saisir ce code HTML :

```
<!DOCTYPE html>

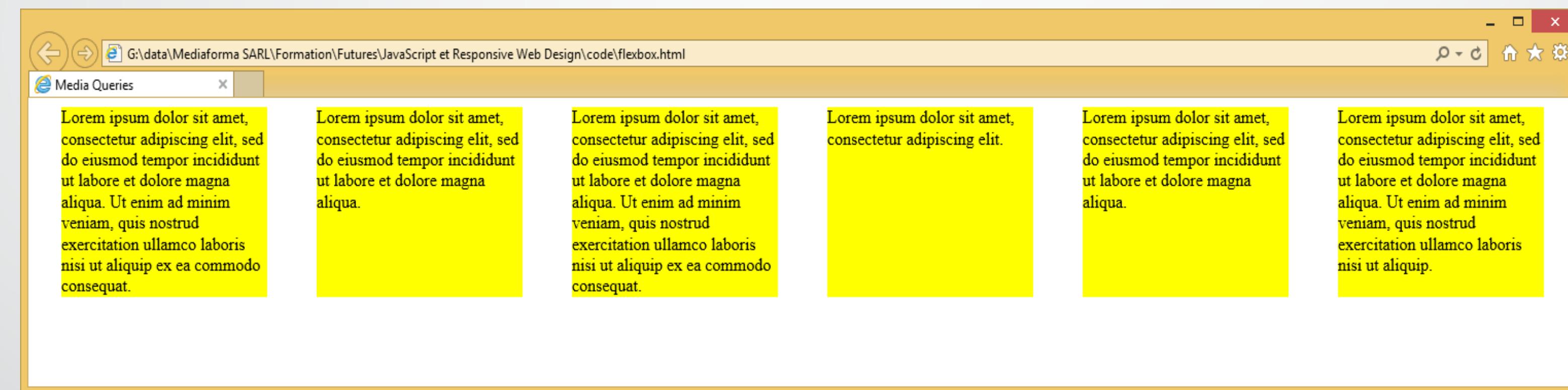
<html>

  <head>
    <meta charset="utf-8">
    <title>Media Queries</title>
    <style>
    </style>
  </head>
```

```
<body>
  <main class="conteneur">
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </div>
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    </div>
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </div>
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    </div>
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    </div>
    <div class="enfant">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip.
    </div>
  </main>
</body>
</html>
```

## Exercice

Définissez les classes conteneur et enfant pour mettre en place une mise en page flexbox dans laquelle tous les enfants ont une largeur de 200 pixels, une couleur d'arrière-plan yellow et se partagent équitablement l'espace horizontal restant.



## Indices

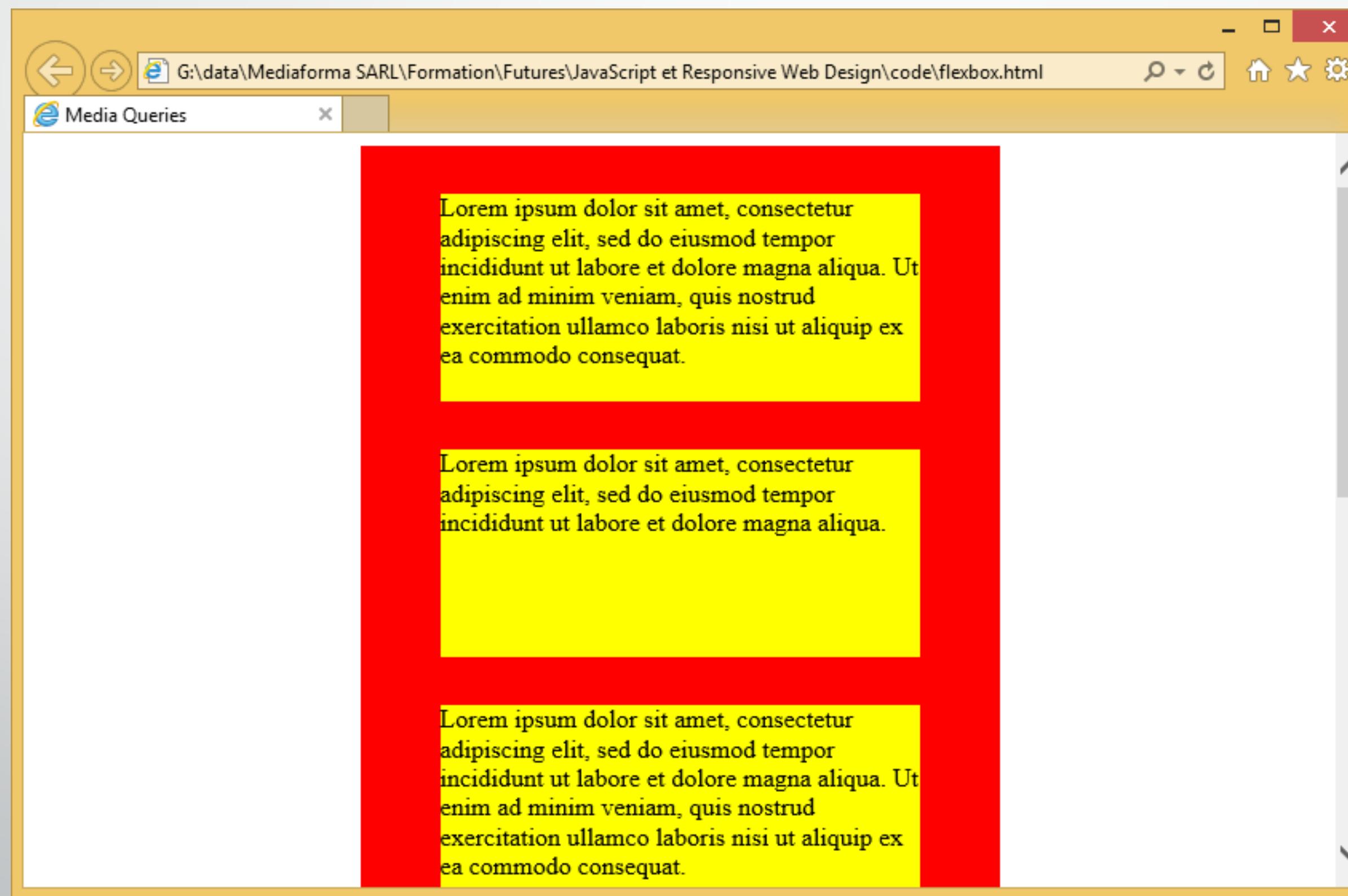
```
.conteneur {  
    display: flex;  
    flex-flow: row|column; // enfants en ligne ou en colonne  
    flex-direction: row|row-reverse|column|column-reverse; // Ordre d'affichage des enfants dans le parent  
    flex-wrap: nowrap|wrap|wrap-reverse; // Sur 1 ligne/col|plusieurs lignes/col|plusieurs lignes/col inversé  
    justify-content: flex-start|flex-end|center|space-between|space-around; // align enfants sur axe princ  
    align-items: flex-start|flex-end|center|stretch|baseline; // alignment des enfants sur l'axe secondaire  
    align-content: flex-start|flex-end|center|stretch|space-between|space-around; // esp. enfants sur axe sec.  
    flex-grow: 1[2|3|...]; //Largeur des éléments affichés sur 1 même ligne/col (n X + large)  
    flex-shrink: 1[2|3|...]; //Largeur des éléments affichés sur 1 même ligne/col (n X + étroit)  
}  
  
#enfant3 {order: 5;} //ordre d'affichage d'un enfant  
#enfant2 { flex-basis: 200px; } // Largeur par défaut d'un élément  
#enfant2 {align-self: auto|flex-start|flex-end|center|baseline|stretch;} //Alignement d'un enfant
```

# Solution

```
<style>  
    .conteneur {  
        display: flex;  
    }  
    .enfant {  
        flex-basis: 200px;  
        background-color: yellow;  
        margin: auto;  
    }  
</style>
```

## Exercice

Définissez les classes conteneur et enfant pour mettre en place une mise en page flexbox sur une colonne centrée de largeur 400 px et d'arrière-plan red. Les enfants auront une hauteur de 130px, une largeur de 300px et un arrière-plan yellow. Chaque enfant sera séparé du suivant par une marge verticale de 30 px et sera centré dans son parent.



# Solution

```
<style>
    .conteneur {
        display: flex;
        flex-flow: column;
        width: 400px;
        background-color: red;
        margin-left: auto;
        margin-right: auto;
    }
    .enfant {
        height: 130px;
        width: 300px;
        background-color: yellow;
        margin-top: 30px;
        margin-left: auto;
        margin-right: auto;
    }
</style>
```

# Textes et polices

Dans cette partie :

- Eléments redéfinis en HTML5
- Nouveaux éléments en HTML5
- Bien afficher les caractères accentués

# Eléments redéfinis en HTML5

## <a>

Dans les versions précédentes du langage, l'élément a avait un modèle de contenu inline. En d'autres termes, il ne pouvait envelopper que des éléments de type inline.

En HTML5, il peut envelopper des éléments de tous niveaux : p, hr, pre, ul, ol, dl, div, h1, h2, h3, h4, h5, h6, hgroup, address, blockquote, ins, del, object, map, noscript, section, nav, article, aside, header, footer, video, audio, figure, table, form, fieldset, menu, canvas ou details.

# <a> Exemples

```
<!-- Lien traditionnel -->
<a href="http://www.siteweb.com">texte</a>
```

```
<!-- Lien sur un tableau -->
<a href="http://www.siteweb.com">
  <table>
    <tr><td></td></tr>
  </table>
</a>
```

```
<!-- Lien avec identifiant sur une image. L'image s'affiche dans une fenêtre complémentaire -->
<a id="monlien" href="http://www.siteweb.com" target="_blank"></a>
```

```
<!-- Lien pour tous les types de média sauf print -->
<a href= "http://www.siteweb.com" media="not print">next</a>
```

# Eléments redéfinis en HTML5

## <b>, <strong>, <i> et <small>

L'élément b met en avant de façon modérée une portion de texte.  
L'élément strong met en avant un texte plus important.

L'élément i met en italique une portion de texte sans toutefois le mettre en avant. Il sera utilisé pour afficher des termes techniques, des pensées, des désignations taxonomiques, des expressions idiomatiques, etc.

L'élément small est essentiellement utilisé pour afficher des commentaires secondaires et des mentions légales. Le texte encadré par cet élément apparaît en caractères de petite taille.

# Nouveaux éléments HTML5

## ***time***

L'élément time définit une information horaire ou une date. Voici sa syntaxe :

```
<time>date ou heure</time>
```

ou

```
<time datetime="date ou heure">texte</date>
```

Quelques exemples :

Le magasin ouvre à <time>09:00</time> du lundi au vendredi.

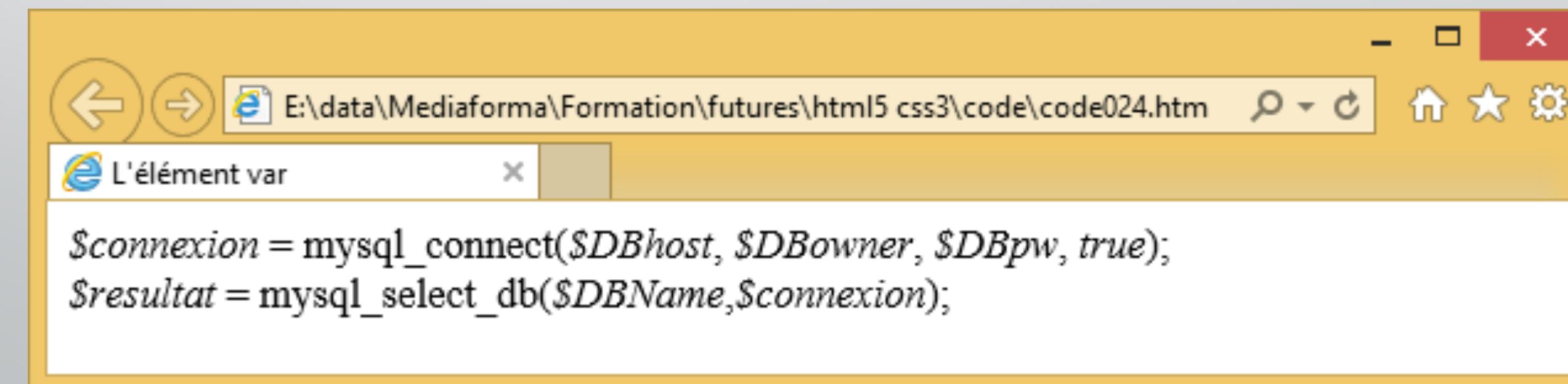
Que faites-vous à <time datetime="25-12-2010">Noël</time> ?

# Nouveaux éléments HTML5

## **var**

L'élément var permet de différencier du reste du texte les variables définies dans du code informatique ou des équations mathématiques. Par défaut, le texte ciblé apparaît en italique, mais il est possible de modifier cet attribut avec des déclarations de styles CSS.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément var</title>
  </head>
  <body>
    <var>$connexion</var> = mysql_connect(<var>$DBhost</var>, <var>$DBowner</var>, <var>$DBpw</var>, <var>true</var>);
    <br><var>$resultat</var> = mysql_select_db(<var>$DBName</var>,<var>$connexion</var>);
  </body>
</html>
```

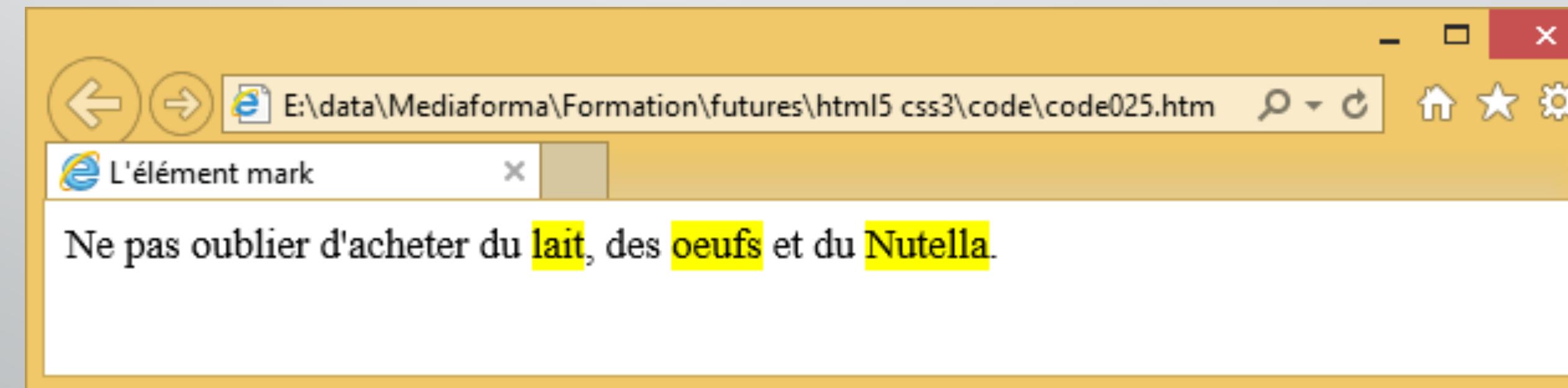


# Nouveaux éléments HTML5

## ***mark***

L'élément mark surligne une portion de texte. Au moment de l'écriture de cet ouvrage, il n'est supporté que par les navigateurs Webkit (Chrome et Safari). Voici sa syntaxe : <mark>texte</mark>

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément mark</title>
  </head>
  </body>
    Ne pas oublier d'acheter du <mark>lait</mark>, des <mark>oeufs</mark> et du <mark>Nutella</mark>.
  </body>
</html>
```



# Bien afficher les caractères accentués

Pour que les caractères accentués apparaissent correctement dans tous les navigateurs du monde, vous pouvez utiliser :

Code	Caractère
&Aacute;	Á
&aacute;	á
&Agrave;	À
&eacute;	é
&agrave;	à
&Acirc;	Â
&acirc;	â

etc.

Vous pouvez aussi utiliser le code ASCII des caractères :

&#210;

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Les codes des caract&egrave;res</title>
  </head>
  <body>
    <pre>
      32 : &#32;    33 : &#33;    34 : &#34;    35 : &#35;    36 : &#36;    37 : &#37;    38 : &#38;    39 : &#39;
40 : &#40;    41 : &#41;    42 : &#42;    43 : &#43;    44 : &#44;    45 : &#45;    46 : &#46;    47 : &#47;    48 : &#48;    49 : &#49;
50 : &#50;    51 : &#51;    52 : &#52;    53 : &#53;    54 : &#54;    55 : &#55;    56 : &#56;    57 : &#57;    58 : &#58;    59 : &#59;
60 : &#60;    61 : &#61;    62 : &#62;    63 : &#63;    64 : &#64;    65 : &#65;    66 : &#66;    67 : &#67;    68 : &#68;    69 : &#69;
70 : &#70;    71 : &#71;    72 : &#72;    73 : &#73;    74 : &#74;    75 : &#75;    76 : &#76;    77 : &#77;    78 : &#78;    79 : &#79;
80 : &#80;    81 : &#81;    82 : &#82;    83 : &#83;    84 : &#84;    85 : &#85;    86 : &#86;    87 : &#87;    88 : &#88;    89 : &#89;
90 : &#90;    91 : &#91;    92 : &#92;    93 : &#93;    94 : &#94;    95 : &#95;    96 : &#96;    97 : &#97;    98 : &#98;    99 : &#99;
100 : &#100;   101 : &#101;   102 : &#102;   103 : &#103;   104 : &#104;   105 : &#105;   106 : &#106;   107 : &#107;   108 : &#108;   109 : &#109;
110 : &#110;   111 : &#111;   112 : &#112;   113 : &#113;   114 : &#114;   115 : &#115;   116 : &#116;   117 : &#117;   118 : &#118;   119 : &#119;
120 : &#120;   121 : &#121;   122 : &#122;   123 : &#123;   124 : &#124;   125 : &#125;   126 : &#126;   127 : &#127;   128 : &#128;   129 : &#129;
130 : &#130;   131 : &#131;   132 : &#132;   133 : &#133;   134 : &#134;   135 : &#135;   136 : &#136;   137 : &#137;   138 : &#138;   139 : &#139;
140 : &#140;   141 : &#141;   142 : &#142;   143 : &#143;   144 : &#144;   145 : &#145;   146 : &#146;   147 : &#147;   148 : &#148;   149 : &#149;
150 : &#150;   151 : &#151;   152 : &#152;   153 : &#153;   154 : &#154;   155 : &#155;   156 : &#156;   157 : &#157;   158 : &#158;   159 : &#159;
160 : &#160;   161 : &#161;   162 : &#162;   163 : &#163;   164 : &#164;   165 : &#165;   166 : &#166;   167 : &#167;   168 : &#168;   169 : &#169;
170 : &#170;   171 : &#171;   172 : &#172;   173 : &#173;   174 : &#174;   175 : &#175;   176 : &#176;   177 : &#177;   178 : &#178;   179 : &#179;
180 : &#180;   181 : &#181;   182 : &#182;   183 : &#183;   184 : &#184;   185 : &#185;   186 : &#186;   187 : &#187;   188 : &#188;   189 : &#189;
190 : &#190;   191 : &#191;   192 : &#192;   193 : &#193;   194 : &#194;   195 : &#195;   196 : &#196;   197 : &#197;   198 : &#198;   199 : &#199;
200 : &#200;   201 : &#201;   202 : &#202;   203 : &#203;   204 : &#204;   205 : &#205;   206 : &#206;   207 : &#207;   208 : &#208;   209 : &#209;
210 : &#210;   211 : &#211;   212 : &#212;   213 : &#213;   214 : &#214;   215 : &#215;   216 : &#216;   217 : &#217;   218 : &#218;   219 : &#219;
220 : &#220;   221 : &#221;   222 : &#222;   223 : &#223;   224 : &#224;   225 : &#225;   226 : &#226;   227 : &#227;   228 : &#228;   229 : &#229;
150 : &#150;   151 : &#151;   152 : &#152;   153 : &#153;   154 : &#154;   155 : &#155;   156 : &#156;   157 : &#157;   158 : &#158;   159 : &#159;
240 : &#240;   241 : &#241;   242 : &#242;   243 : &#243;   244 : &#244;   245 : &#245;   246 : &#246;   247 : &#247;   248 : &#248;   249 : &#249;
250 : &#250;   251 : &#251;   252 : &#252;   253 : &#253;   254 : &#254;   255 : &#255;

    </pre>
  </body>
</html>
```

The screenshot shows a web browser window with a yellow header bar. The address bar displays the path: E:\data\Mediaforma\Formation\futures\html5 css3\code\code026.htm. The main content area contains a table titled "Les codes des caractères". The table lists various characters and their corresponding ASCII codes. The columns are numbered from 32 to 255. The table includes characters such as parentheses, punctuation marks, letters, and special characters like €, ©, and ñ.

		32 :	33 :	!	34 :	"	35 :	#	36 :	\$	37 :	%	38 :	&	39 :	'	
40 :	(	41 :	)	42 :	*	43 :	+	44 :	,	45 :	-	46 :	.	47 :	/	48 :	0
50 :	2	51 :	3	52 :	4	53 :	5	54 :	6	55 :	7	56 :	8	57 :	9	58 :	:
60 :	<	61 :	=	62 :	>	63 :	?	64 :	@	65 :	A	66 :	B	67 :	C	68 :	D
70 :	F	71 :	G	72 :	H	73 :	I	74 :	J	75 :	K	76 :	L	77 :	M	78 :	N
80 :	P	81 :	Q	82 :	R	83 :	S	84 :	T	85 :	U	86 :	V	87 :	W	88 :	X
90 :	Z	91 :	[	92 :	\	93 :	]	94 :	^	95 :	_	96 :	`	97 :	a	98 :	b
100 :	d	101 :	e	102 :	f	103 :	g	104 :	h	105 :	i	106 :	j	107 :	k	108 :	l
110 :	n	111 :	o	112 :	p	113 :	q	114 :	r	115 :	s	116 :	t	117 :	u	118 :	v
120 :	x	121 :	y	122 :	z	115 :	s	124 :		125 :	}	126 :	~	127 :		128 :	€
130 :,	,	131 :	f	132 :	"	133 :	...	134 :	†	135 :	‡	136 :	^	137 :	‰	138 :	š
140 :	€	141 :		142 :	ž	143 :		144 :	‘	145 :	’	146 :	“	147 :	”	148 :	”
150 :-	-	151 :-	-	152 :-	~	153 :-	™	154 :-	š	155 :-	>	156 :-	œ	157 :-		158 :-	ž
160 :;	;	161 :;	;	162 :;	¢	163 :;	ƒ	164 :;	¤	165 :;	⌘	166 :;	¡	167 :;	§	168 :;	“
170 :`	`	171 :«	«	172 :¬	¬	173 :`		174 :®	®	175 :—	—	176 :°	°	177 :±	±	178 :²	²
180 :'	'	181 :µ	µ	182 :¶	¶	183 :·	·	184 :,	,	185 :¹	¹	186 :º	º	187 :»	»	188 :¼	¼
190 :¾	¾	191 :¿	¿	192 :À	À	193 :Á	Á	194 :Â	Â	195 :Ã	Ã	196 :Ä	Ä	197 :Å	Å	198 :Æ	Æ
200 :È	È	201 :É	É	202 :Ê	Ê	203 :Ë	Ë	204 :Ì	Ì	205 :Í	Í	206 :Î	Î	207 :Ï	Ï	208 :Ð	Ð
210 :Ò	Ò	211 :Ó	Ó	212 :Ô	Ô	213 :Õ	Õ	214 :Ö	Ö	215 :×	×	216 :Ø	Ø	217 :Ù	Ù	218 :Ú	Ú
220 :Ü	Ü	221 :Ý	Ý	222 :Þ	Þ	215 :×	×	224 :à	à	225 :á	á	226 :â	â	227 :ã	ã	228 :ä	ä
150 :-	-	151 :-	-	152 :-	~	153 :-	™	154 :-	š	155 :-	>	156 :-	œ	157 :-		158 :-	ž
240 :ð	ð	241 :ñ	ñ	242 :ò	ò	243 :ó	ó	244 :ô	ô	245 :õ	õ	246 :ö	ö	247 :÷	÷	248 :ø	ø
250 :ú	ú	251 :û	û	252 :ü	ü	253 :ý	ý	254 :þ	þ	255 :ÿ	ÿ						249 :ù

# Regroupement du contenu

Dans cette partie :

- Concept de boîte
- Styles de liste HTML
- Choisir le style d'une liste
- Des images à la place des puces
- Légendage
- Texte non proportionnel
- Citations
- Ruptures thématiques

# Concept de boîte

Les "boîtes de bloc" sont des zones rectangulaires qui se comportent comme autant de zones d'affichage indépendantes. Chaque boîte peut être dotée de barres de défilement, d'une image d'arrière-plan, de styles indépendants et être positionnée précisément sur la page pour obtenir un affichage digne d'un professionnel.

Les boîtes sont mises en place avec l'élément `div`. Leurs principales propriétés sont les suivantes :

- `width` : largeur ;
- `height` : hauteur ;
- `margin` : marges extérieures ;
- `padding` : marges intérieures ;
- `background-color` : couleur d'arrière-plan.

# Exercice

Définissez une boîte dont les caractéristiques sont les suivantes :

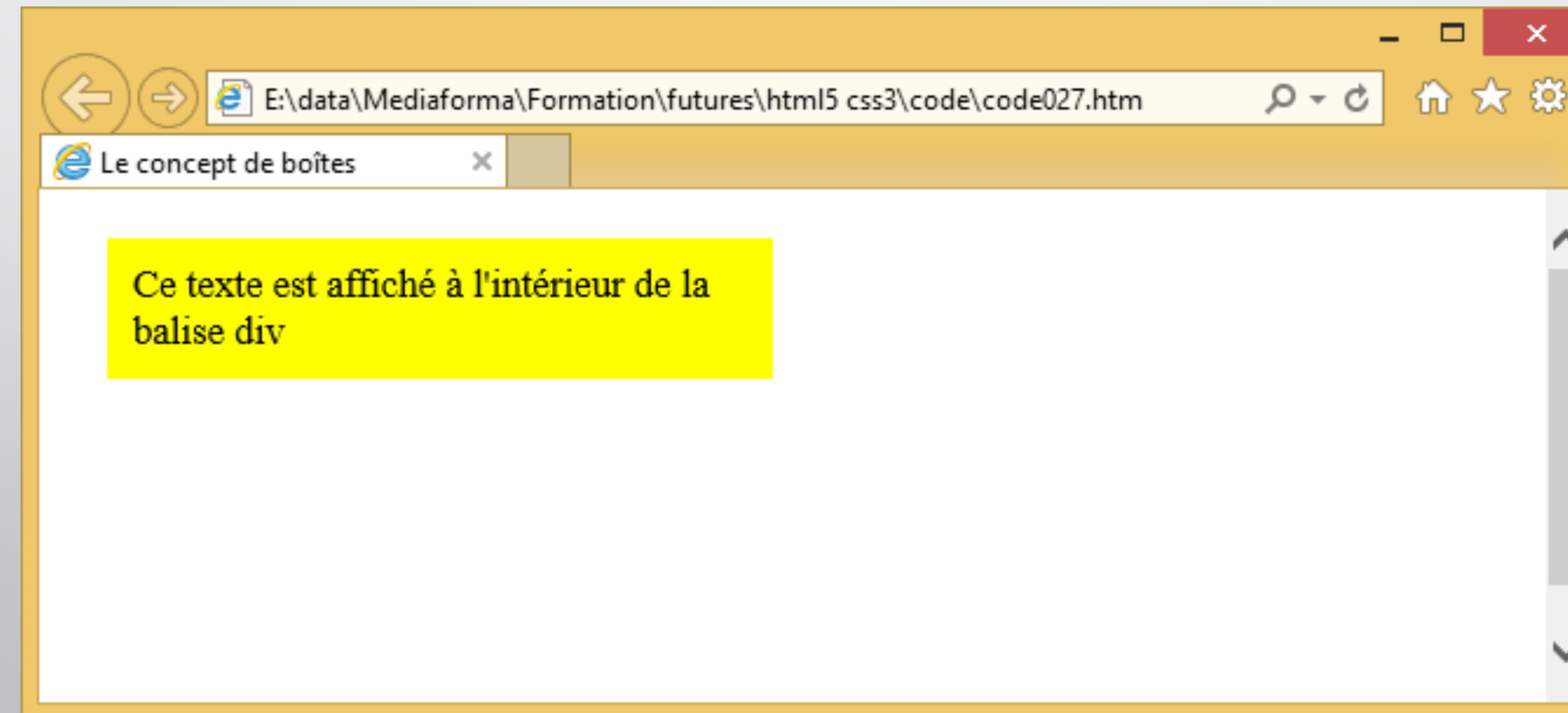
- largeur : 250 pixels ;
- hauteur : 60 pixels ;
- marges extérieures : 20 pixels ;
- marges intérieures : 10 pixels ;
- couleur d'arrière-plan : jaune.

Affichez du texte dans cette boîte.

# Solution

code027.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Le concept de boîtes</title>
  </head>
  <body>
    <div style="width:250px; height=60px; margin: 20px; padding: 10px; background-color: yellow;">
      Ce texte est affiché à l'intérieur de la balise div
    </div>
  </body>
</html>
```



# Listes HTML5

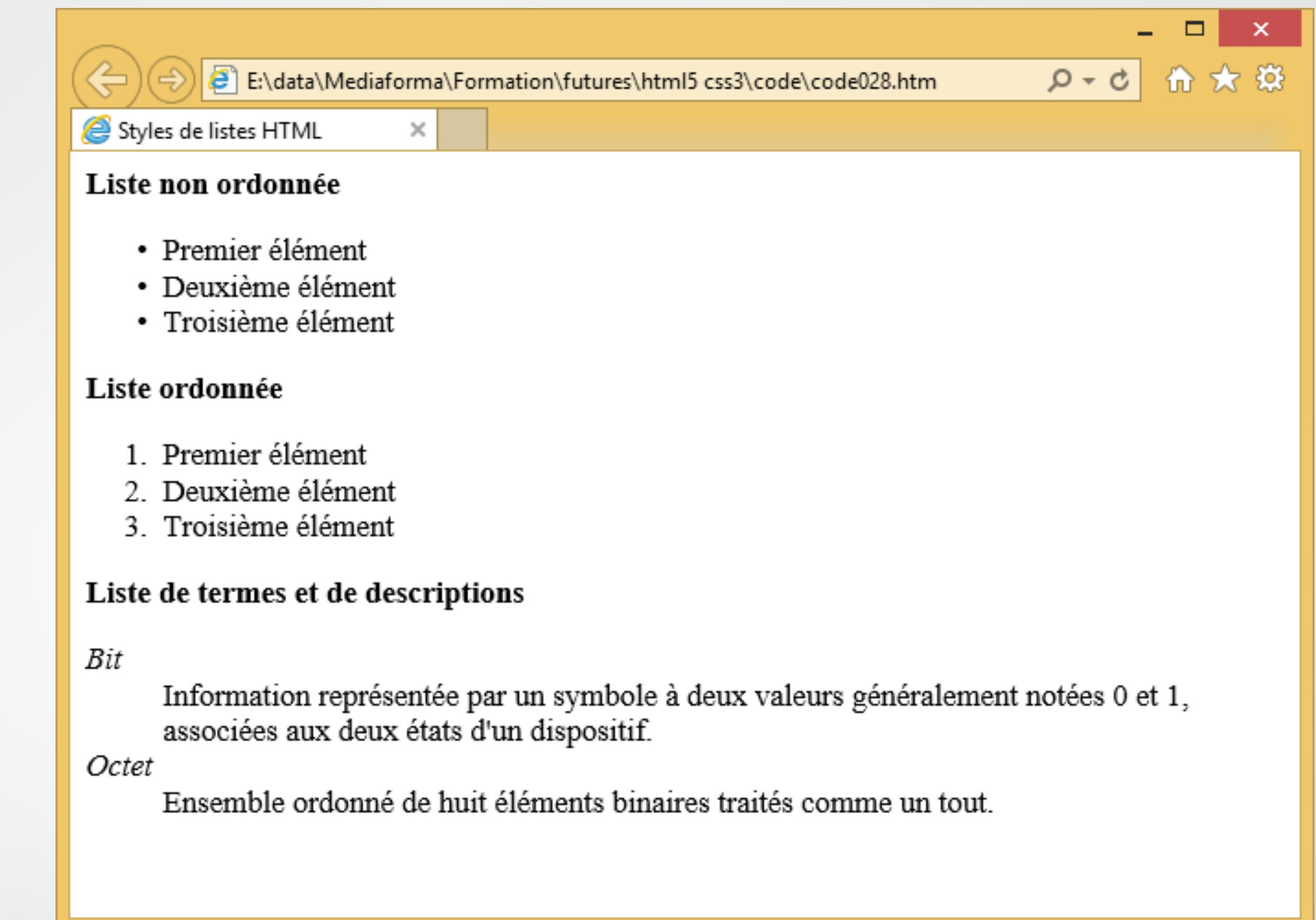
Le langage HTML5 compte trois types de listes :

- non ordonnées ou listes à puces : éléments ul et li ;
- ordonnées : éléments ol et li ;
- de description : éléments dl, dt, dd et dfn.

Les listes de description étaient déjà présentes dans HTML 4.x. Souvent boudées et utilisées de manière non sémantique, elles ont subi un lifting pour être remises au goût du jour dans HTML5. L'élément dfn est essentiellement utilisé pour encapsuler les mots d'un glossaire, à l'intérieur de l'élément dt. Notez qu'on peut également définir plusieurs descriptions (dd) pour un seul nom (dt), ou encore regrouper plusieurs noms pour une seule description.

Cet exemple illustre ces trois types de listes :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Styles de listes HTML</title>
  </head>
  <body>
    <b>Liste non ordonnée</b>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ul>
    <b>Liste ordonnée</b>
    <ol>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ol>
    <b>Liste de termes et de descriptions</b>
    <dl>
      <dt><dfn>Bit</dfn></dt>
      <dd>Information représentée par un symbole à deux valeurs généralement notées 0 et 1, associées aux deux états d'un dispositif.</dd>
      <dt><dfn>Octet</dfn></dt>
      <dd>Ensemble ordonné de huit éléments binaires traités comme un tout.</dd>
    </dl>
  </body>
</html>
```



# Choisir le style d'une liste

CSS3 est bien plus riche que HTML5 en matière de listes et la propriété list-style-type permet de choisir le style d'une liste. Elle admet de nombreuses valeurs.

Pour une liste à puces (élément ul) :

- disc : puce ronde remplie ;
- circle : puce ronde vide ;
- square : carré rempli ;
- none : aucun signe.

Pour une liste ordonnée alphabétique (élément ol) :

- lower-roman : i, ii, iii, iv ;
- upper-roman : I, II, III, IV ;
- lower-greek : α, β, χ, δ, ε ;
- lower-alpha : a, aa, aaa ;
- upper-alpha : A, AA, AAA ;
- lower-latin : a, b, c ;
- upper-latin : A, B, C ;
- none : aucune numérotation.

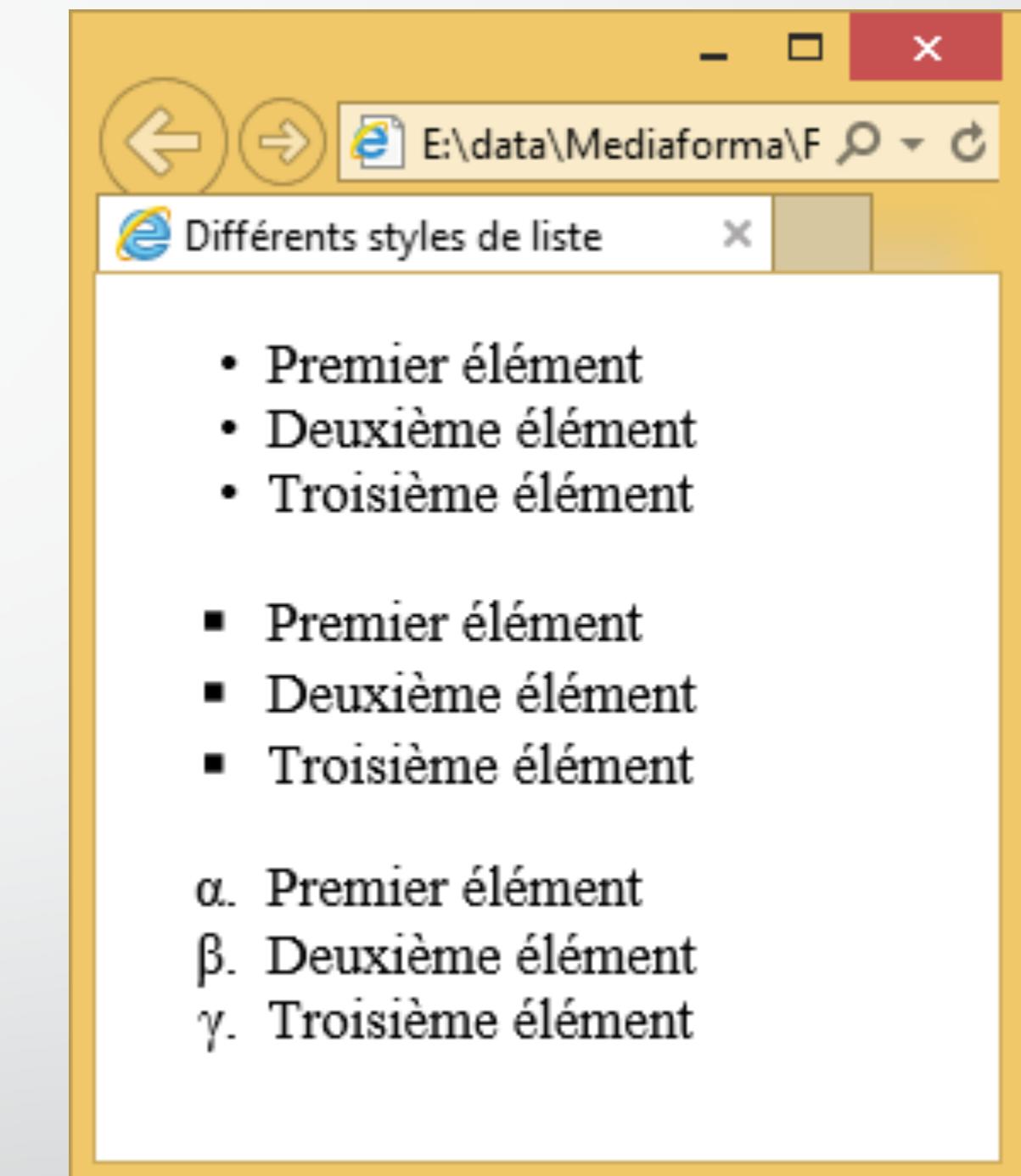
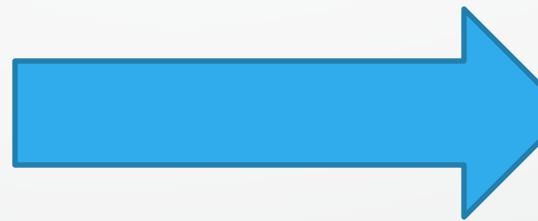
Pour une liste ordonnée décimale (élément ol) :

- decimal : 1, 2, 3 ;
- decimal-leading-zero : 01, 02, 03 ;
- hebrew : numérotation hébraïque traditionnelle ;
- armenian : numérotation arménienne traditionnelle ;
- georgian : numérotation géorgienne traditionnelle ;
- hiragana : a, i, u, e, o, ka, ki, ku ;
- katakana : A, I, U, E, O, KA, KI, KU ;
- hiragana-iroha : i, ro, ha, ni, ho, he ;
- katakana-iroha : I, RO, HA, NI, HO, TE ;
- none : aucune numérotation.

Exercice : Entraînez-vous à utiliser la propriété list-style-type dans des listes ul et ol.

## code029.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Différents styles de liste</title>
    <style>
      .rond {list-style-type: disc;}
      .carre {list-style-type: square;}
      .lgreek {list-style-type: lower-greek;}
    </style>
  </head>
  <body>
    <ul class="rond">
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ul>
    <ul class="carre">
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ul>
    <ol class="lgreek">
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ol>
  </body>
</html>
```



# Des images à la place des puces

Si vous vous sentez limité par les styles de puces accessibles par défaut en CSS3, sachez que la propriété list-style-image permet d'utiliser n'importe quelle image comme symbole de puce. Voici sa syntaxe :

```
list-style-image: url (adresse);
```

Où adresse est l'adresse URI de la puce, au format GIF ou JPEG.

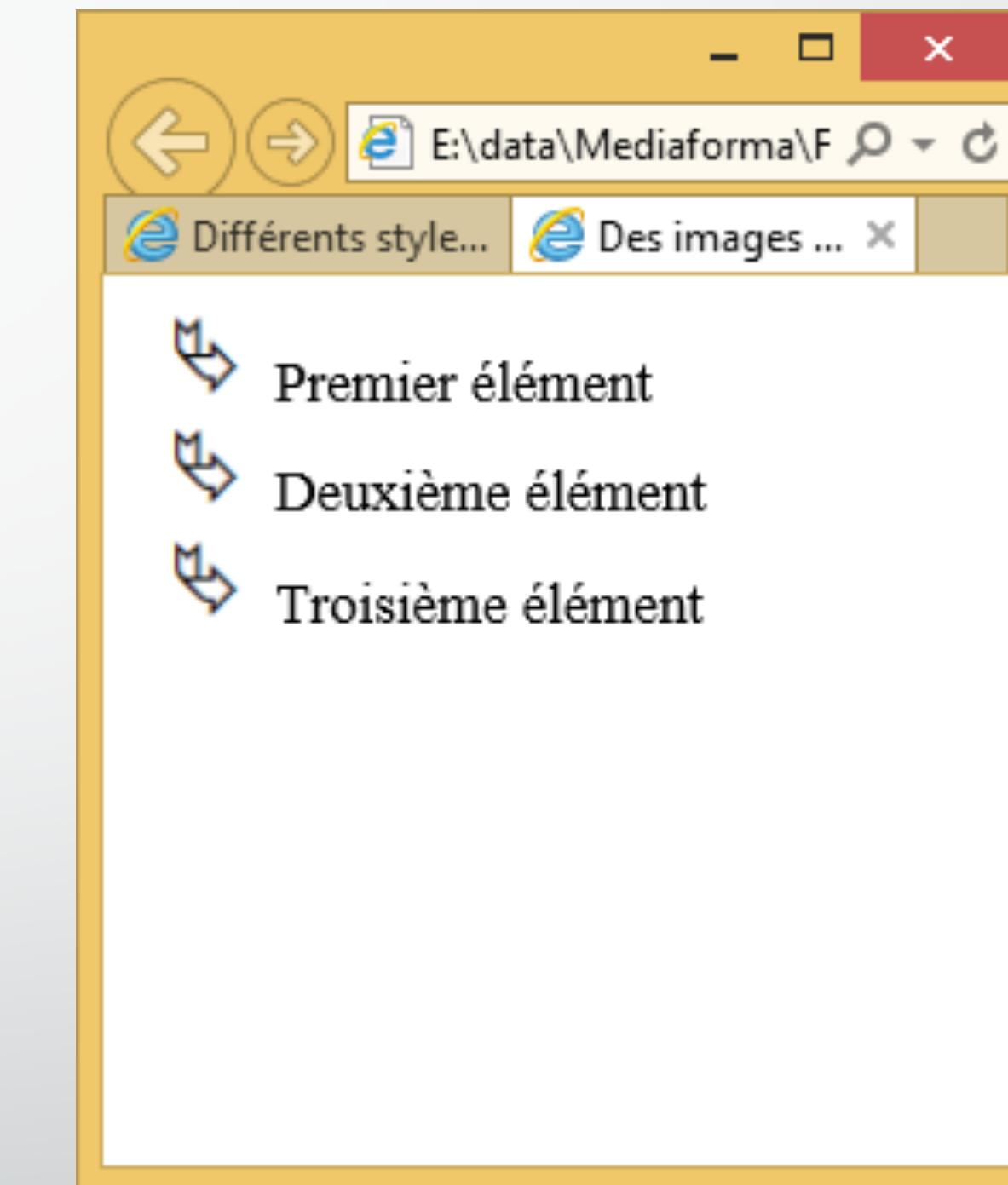
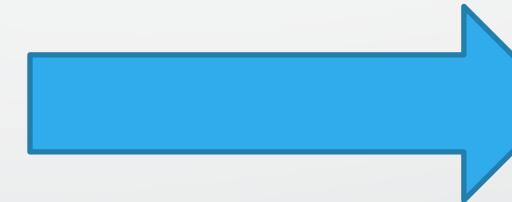
# Exercice

Trouvez sur le Web un clipart de puce et utilisez-le dans une liste à puces.

# Solution

Code030.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Des images à la place des puces</title>
    <style>
      .puce {list-style-image: url(puce.gif); }
    </style>
  </head>
  <body>
    <ul class="puce">
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
    </ul>
  </body>
</html>
```



La puce utilisée est une image GIF :



# Des images à la place des puces

Si l'image spécifiée dans la propriété list-style-image n'est pas trouvée, la puce par défaut la remplacera. Si vous souhaitez une puce circle, square ou aucune puce, vous devez initialiser la propriété list-style-type en conséquence. Par exemple :

```
.puce
{
    list-style-image: url(puces.gif);
    list-style-type: square;
}
```

# Légendage

Dans les livres et les magazines, les illustrations, graphiques, tableaux, exemples de code, etc., sont souvent repérés par une légende. Jusqu'ici, il n'existe aucun moyen de marquer sémantiquement ce genre de contenu dans un document HTML. Pour combler cette lacune, deux nouveaux éléments font leur apparition dans HTML5 : figure et figcaption.

Vous les utiliserez comme ceci :

```
<figure>  
    <img>, <table>, <video>, <code>, etc.  
    <figcaption>La légende de l'élément.</figcaption>  
</figure>
```

## Attention

figcaption doit être défini comme premier ou dernier enfant de l'élément figure.

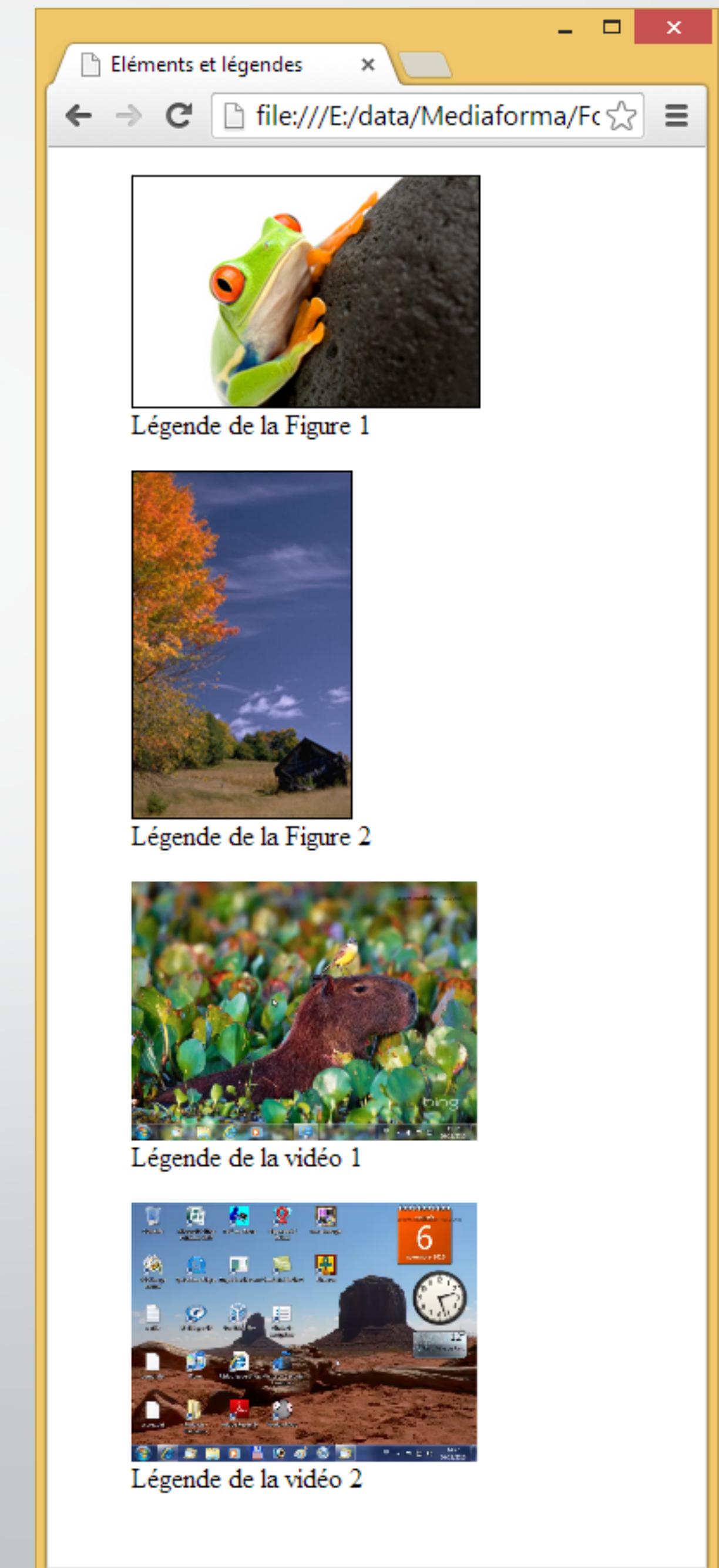
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Eléments et légendes</title>
  </head>
  <body>
    <figure>
      
      <figcaption>Légende de la Figure 1</figcaption>
    </figure>

    <figure>
      
      <figcaption>Légende de la Figure 2</figcaption>
    </figure>

    <figure>
      <video src="video1.mp4" width="200" border="1"></video>
      <figcaption>Légende de la vidéo 1</figcaption>
    </figure>

    <figure>
      <video src="video2.mp4" width="200" border="1"></video>
      <figcaption>Légende de la vidéo 2</figcaption>
    </figure>

  </body>
</html>
```



Images et vidéos ne sont pas les seuls éléments HTML5 qui peuvent être accompagnés d'une légende. Ici, par exemple, nous définissons une légende pour un code HTML5 et une autre pour un extrait de poème.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Eléments et légendes</title>
  </head>
  <body>
    <p>Le <a href="#1">Listing 1</a> représente le code à utiliser pour afficher une vidéo en HTML5.</p>
    <figure id="1">
      <figcaption>Listing 1. Affichage d'une vidéo en HTML5.</figcaption>
      <pre><code>
        &lt;video src="nom de la vidéo" width="largeur" height="hauteur"&gt;
        &lt;/video&gt;
      </code></pre>
    </figure>

    <figure>
      <p>Si on venait à nous séparer,<br>
        Jamais je ne m'en remettrais,<br>
        Il fait si bon être à tes côtés,<br>
        Que plus rien ne semble mauvais.<br>
        Tu es ma joie de vivre,<br>
        Mon seul et unique bonheur,<br>
        Ton amour m'enivre,<br>
        Et fait chavirer mon coeur.</p>
      <figcaption><cite>Pour toi, mon amour éternel</cite> (extrait). Hubert</figcaption>
    </figure>
  </body>
</html>
```

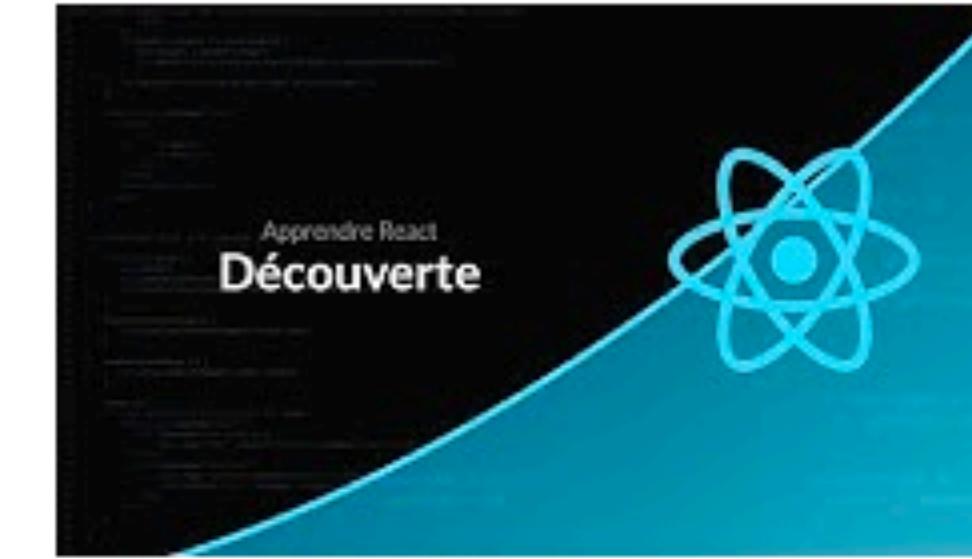
# Texte non proportionnel

Lorsqu'un texte est encadré avec les éléments `pre` et `/pre`, ses caractères sont affichés avec une police non proportionnelle, et les passages à la ligne respectent ceux rencontrés dans le code. Les éléments `br` ne sont donc plus nécessaires. Cet élément est généralement utilisé pour différencier le code informatique sur une page web.

Le code ci-après affiche du texte et du code HTML. Il est utilisé deux fois pour montrer l'importance des caractères `&lt;` et `&gt;` et pour mettre en évidence l'interprétation des caractères d'espacement dans l'élément `pre`.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Balise Pre</title>
  </head>
  <body>
    Ce code affiche une imagette sur
    <pre>
      <a href="https://www.youtube.co
        
  </body>
</html>
```

Ce code affiche une imagette sur laquelle le visiteur peut cliquer pour afficher la vidéo "Enregistrer le son du PC" :



Ce code affiche une imagette sur laquelle le visiteur peut cliquer pour afficher la vidéo "Enregistrer le son du PC" :

```
<a href="https://www.youtube.com/watch?v=SMgQlTSof0&t=0s">
  
</a>
```

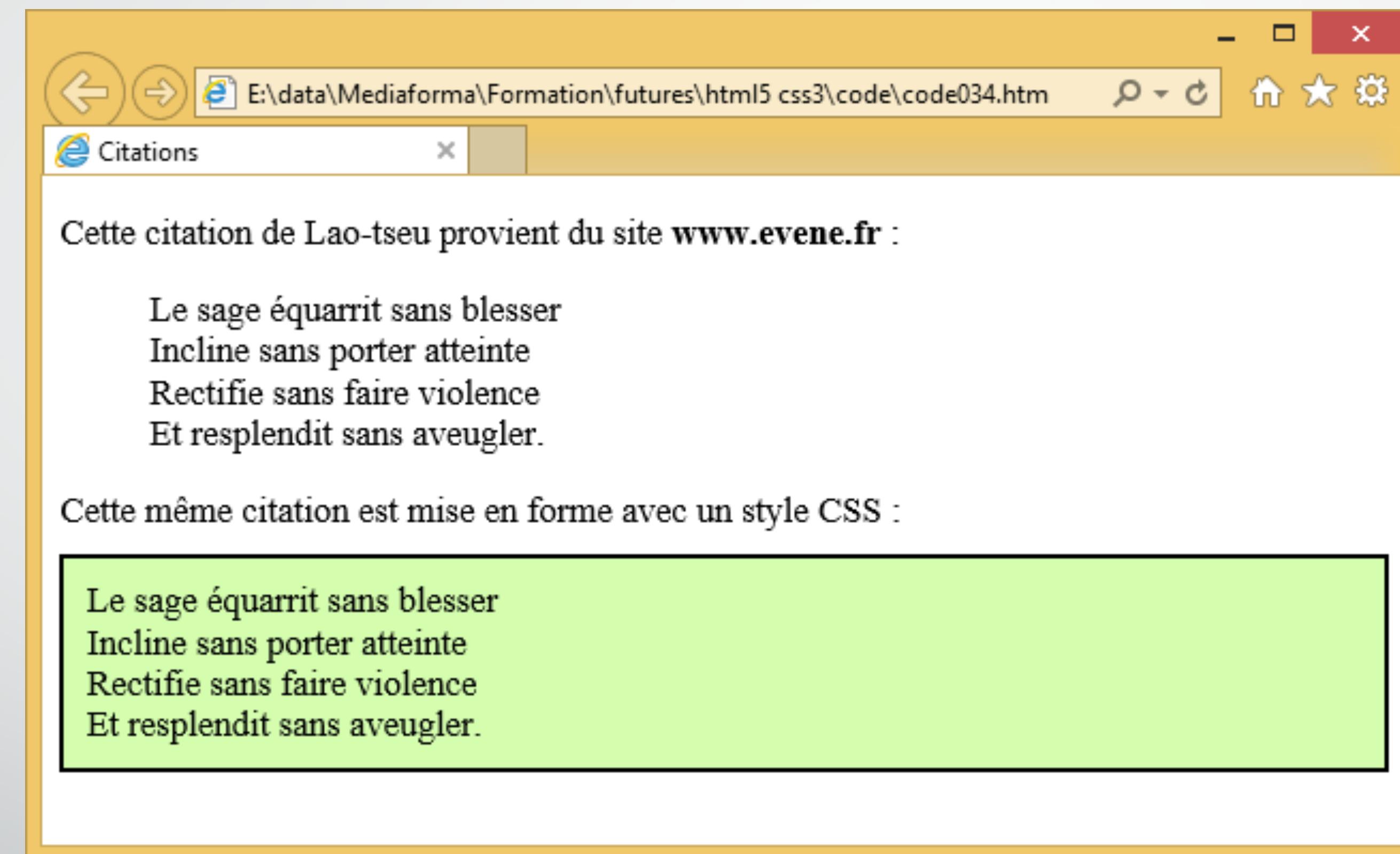
# Citations

L'élément blockquote permet d'insérer une citation dans une page web, ce qui est intéressant pour faire référence à une source d'informations extérieure ou pour mettre en valeur des blocs de texte dans vos pages.

Le code ci-après affiche une citation provenant d'un site externe,  
sans mise en forme CSS puis avec mise en forme CSS :

Code034.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Citations</title>
    <style>
      .avecstyle
      {
        font-family: "Courier New", serif;
        border : 2px solid #000;
        margin: 10px 0 0 0;
        padding: 10px;
        background-color: #D4FEAE;
      }
    </style>
  </head>
  <body>
    <p>Cette citation de Lao-tseu provient du site <b>www.evane.fr</b> :
    <blockquote cite="www.evane.fr">
      Le sage équarrit sans blesser<br>
      Incline sans porter atteinte<br>
      Rectifie sans faire violence<br>
      Et resplendit sans aveugler.<br>
    </blockquote></p>
    Cette même citation est mise en forme avec un style CSS :
    <blockquote cite="www.evane.fr" class="avecstyle">
      Le sage équarrit sans blesser<br>
      Incline sans porter atteinte<br>
      Rectifie sans faire violence<br>
      Et resplendit sans aveugler.<br>
    </blockquote>
  </body>
</html>
```



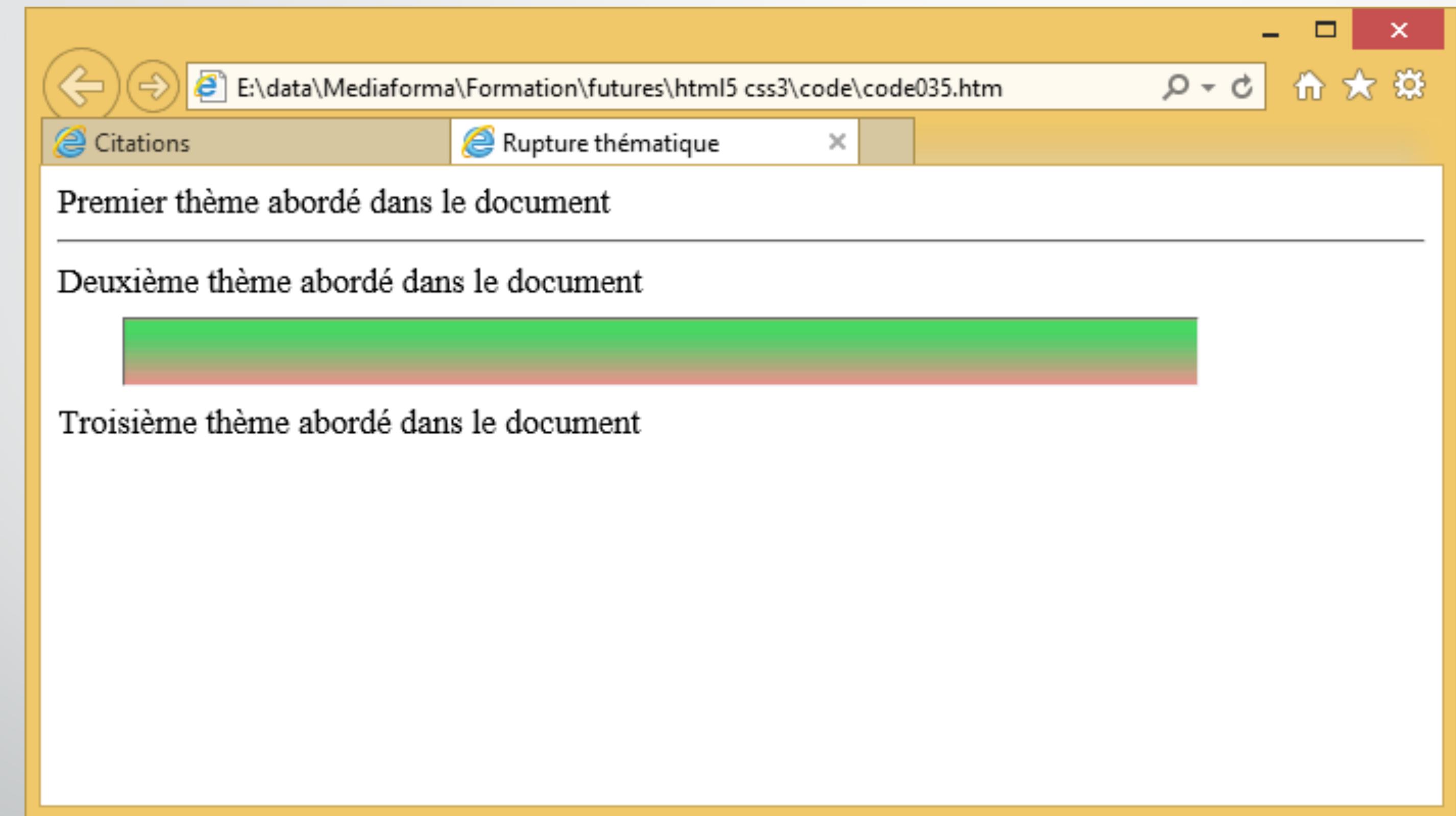
# Rupture thématique

L'élément hr insère une ligne horizontale qui représente une rupture thématique dans le contenu de la page. Les attributs HTML 4.01 align, noshade, size et width ne sont plus supportés dans la version HTML5 de cet élément. Pour modifier les caractéristiques d'un élément hr, vous devrez désormais passer par le code CSS.

Ce code affiche un élément `hr` standard et un autre dont les caractéristiques ont été modifiées par quelques lignes de CSS

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Rupture thématique</title>
    <style>
      .hrstyle1
      {
        text-align:left;
        margin-left: 30px;
        width:500px;
        height:30px;
        background: url(_fond.gif);
      }
    </style>
  </head>
  <body>
    Premier thème abordé dans le document
    <hr>
    Deuxième thème abordé dans le document
    <hr class="hrstyle1">
    Troisième thème abordé dans le document
  </body>
</html>
```

Voici une  
version  
agrandie de  
l'image  
`_fond.gif`:



# Les éléments interactifs

Certains éléments HTML5 sont interactifs de façon native, c'est-à-dire qu'ils ne nécessitent pas de code CSS, JavaScript ou jQuery supplémentaire. Ils ont été regroupés dans cette partie :

- Version résumée et étendue d'un texte
- Eléments éditables
- Menus HTML5

# <details> et <summary>

Les éléments <details> et <summary> permettent d'afficher des versions résumée et étendue d'un texte.  
Voici la syntaxe à utiliser :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Balises details et summary</title>
  </head>
  <body>
    <details>
      <summary>En savoir plus</summary>
      <p>Vous en saurez plus sur le référencement en vous rendant sur le site http://www.monsite.com.</p>
    </details>
  </body>
</html>
```

Essayez ce code dans Google Chrome, Firefox et Internet Explorer.

# Eléments éditables

Si vous affectez la valeur `true` à l'attribut `contenteditable` d'éléments comme `span`, `div` et `p`, vous pourrez les modifier.

Si le cœur vous en dit, vous pouvez même ajouter quelques lignes de JavaScript basées sur la fonction `getElementById()` pour récupérer le texte modifié afin de lui appliquer un traitement.

# Exercice

Définissez un document HTML contenant une balise <div> que vous rendrez éditable.

Insérez un bouton et une deuxième balise <div> dans le document.

Lorsque l'utilisateur clique sur le bouton, copiez le contenu de la première <div> dans la deuxième. Pour cela, vous utiliserez la fonction JavaScript getElementById().

# Solution

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Eléments éditables</title>
    <style>
      div
      {
        border: dotted 1px #666;
        padding: 5px 10px;
        margin-top: 40px;
        margin-bottom: 40px;
      }
    </style>
    <script>
      function traitement()
      {
        var contenu = document.getElementById('texte-editable').innerHTML;
        document.getElementById('cible').innerHTML = 'Texte lu dans la div : ' + contenu;
      }
    </script>
  </head>
  <body>
    <div id="texte-editable" contenteditable="true">Le texte de cette div peut être modifié</div>
    <button onclick="traitement()">Cliquez ici pour accéder au texte de la div</button>
    <div id="cible"></div>
  </body>
</html>
```

# Des menus en HTML5

Les éléments menu et command font leur apparition dans HTML5. Grâce à eux, il est possible de mettre en place un menu en quelques lignes de code.

L'élément menu définit le menu et les éléments command associent commandes de menus et actions.

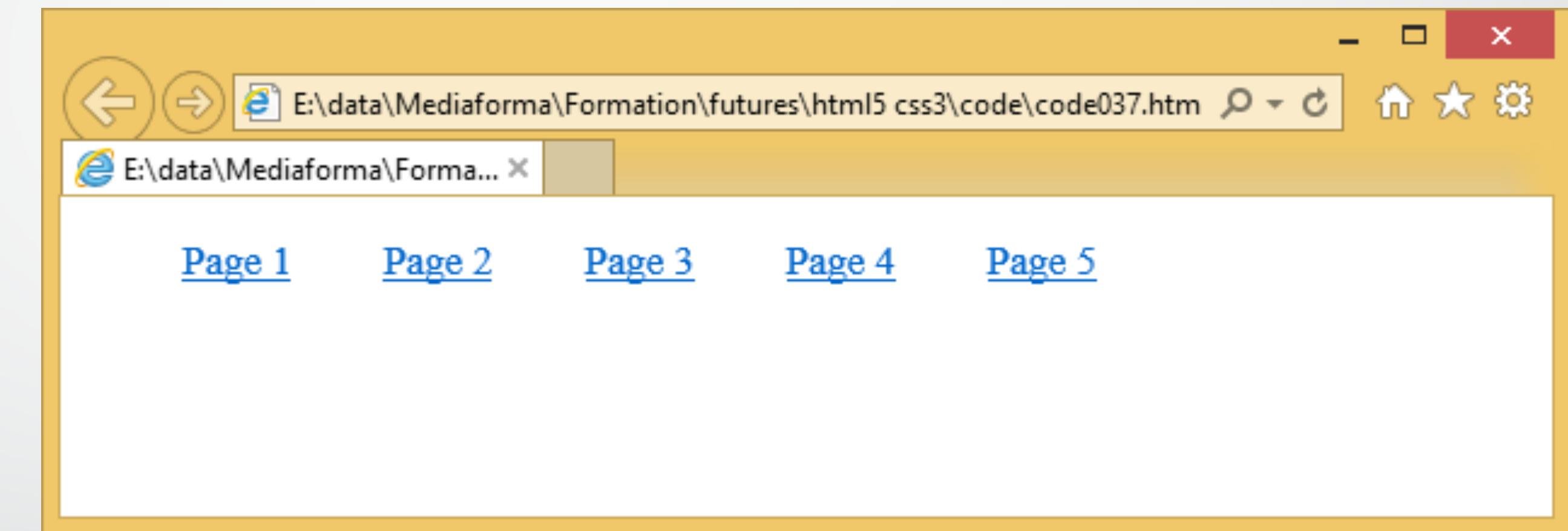
Malheureusement, ces balises sont très mal supportées dans les navigateurs actuels. C'est pourquoi vous devrez utiliser des techniques plus conventionnelles pour mettre en place un menu.

# Des menus en HTML5

Code037.htm

Pour créer un menu horizontal, le plus simple consiste à utiliser une liste `<ul>` dont la propriété `display` vaut `inline`. Ainsi, les éléments de la liste ne seront pas affichés verticalement, en tant que blocs, mais horizontalement, sur une même ligne.

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
    ul#horizontal
    {
        list-style-type: none;
    }
    ul#horizontal li
    {
        display: inline;
        padding-right: 2em;
    }
</style>
</head>
<body>
    <nav>
        <ul id="horizontal">
            <li><a href="page1.htm">Page 1</a></li>
            <li><a href="page2.htm">Page 2</a></li>
            <li><a href="page3.htm">Page 3</a></li>
            <li><a href="page4.htm">Page 4</a></li>
            <li><a href="page5.htm">Page 5</a></li>
        </ul>
    </nav>
</body>
</html>
```



# Un menu déroullant

Le principe de base consiste à imbriquer deux listes `<ul>`, la première contenant les entrées principales du menu et la deuxième les entrées secondaires.

```
<style type="text/css">
#principal
{
    background: #E0FCC8;
    width: 800px;
    height: 30px;
    position: absolute;
    top: 0;
    left: 0;
}
#principal li
{
    list-style-type: none;
    float: left;
    width: 150px;
    border-style: dashed;
    border-width: 1px;
}
#principal li a:link, #principal li a:visited
{
    display: block;
    background: #E0FCC8;
    margin: 0;
    padding: 5px 6px;
    text-decoration: none;
    color: black;
}
#principal li a:hover
{
    background-color: #A5F95B;
}
#principal .secondaire
{
    display: none;
    list-style-type: none;
    margin: 0;
    padding: 0;
    border: 0;
}
```

```
#principal .secondaire li
{
    float: none;
    border: 0;
}
#principal .secondaire li a:link, #principal .secondaire li a:visited
{
    display: block;
    text-decoration: none;
}
#principal .secondaire li a:hover
{
    background-color: #A5F95B;
}
#principal li:hover> .secondaire
{
    display: block;
}
</style>
```

---

```
<body>
<ul id="principal">
<li><a href="#">Page 1</a>
<ul class="secondaire">
<li><a href="page1-1.htm">Page 1 - Sous-page 1</a></li>
<li><a href="page1-2.htm">Page 1 - Sous-page 2</a></li>
<li><a href="page1-3.htm">Page 1 - Sous-page 3</a></li>
</ul>
</li>
<li><a href="#">Page 2</a>
<ul class="secondaire">
<li><a href="page2-1.htm">Page 2 - Sous-page 1</a></li>
<li><a href="page2-2.htm">Page 2 - Sous-page 2</a></li>
<li><a href="page2-3.htm">Page 2 - Sous-page 3</a></li>
</ul>
</li>
<li><a href="#">Page 3</a>
<ul class="secondaire">
<li><a href="page3-1.htm">Page 3 - Sous-page 1</a></li>
<li><a href="page3-2.htm">Page 3 - Sous-page 2</a></li>
<li><a href="page3-3.htm">Page 3 - Sous-page 3</a></li>
</ul>
</li>
</ul>
</body>
```

# Les liens hypertextes

Avec les liens hypertextes, vos pages web peuvent faire référence à des pages situées sur le même site web ou sur des sites annexes. Ce chapitre montre comment personnaliser vos éléments XHTML a en définissant un ou plusieurs styles CSS associés.

# Liens hypertextes

Plusieurs types de liens peuvent être utilisés :

Lien vers	Exemple
un document local	<a href="nom-du-document.htm">texte du lien</a>
un signet dans un document local	<a href="nom-du-document.htm#nom-du-signet">texte du lien</a> (le signet est défini à l'aide de l'attribut id ou name d'un élément quelconque)
un document sur un autre site	<a href="http://nom-du-site/nom-du-dossier/nom-du-document.htm">texte du lien</a>
un fichier déposé sur un serveur FTP	<a href= "ftp://adresse-du-fichier">Texte du lien</a>
un correspondant e-mail	<a href= "mailto:adresse-du-correspondant">Texte du lien</a>

# Liens hypertextes

En HTML 4.01, l'élément a pouvait être utilisé pour définir un lien ou une ancre. La nuance se faisait au niveau de l'attribut href.

Cette ligne définit un lien vers le site Google :

```
<a href="http://www.google.fr">Cliquez ici pour aller sur Google</a>
```

Par contre, cette ligne définit l'ancre "plusloin" :

```
<a name="plusloin">Texte quelconque</a>
```

Pour faire afficher la portion de page qui commence à l'ancre plusloin, le code à utiliser est le suivant :

```
<a href="#plusloin">Cliquez ici pour aller plus loin</a>
```

Supposons que l'ancre ait été définie sur une autre page HTML nommée page2.htm. Pour afficher la portion de page qui commence à l'ancre plusloin dans page2.htm, le code à utiliser est le suivant :

```
<a href="page2.htm#plusloin">Cliquez ici pour aller plus loin</a>
```

# Liens hypertexte en HTML5

En HTML5, l'attribut name utilisé sur des éléments autres que input devient obsolète. L'élément a ne peut donc plus être utilisé pour définir des ancrages. Assurez-vous : définir des ancrages est toujours possible si on affecte un identifiant à un élément et si on y fait référence dans un élément a :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Balise a</title>
  </head>
  <body>
    <h1 id="test">Titre 1</h1>
    un peu de texte<br>
    <a href="#test">Aller au titre 1</a>
  </body>
</html>
```

# Liens hypertextes en HTML5

En HTML5, l'élément `a` peut contenir un ou plusieurs éléments quelconques, mis à part les éléments interactifs.  
Le code ci-après est ainsi totalement licite :

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <aside>
      Atteindre
      <a href="http://www.mediaforma.com/windows8">
        Les formations
        <h2>Une minute par jour</h2>
        Une minute tous les jours pour apprendre à utiliser Windows 8
      </a><br>
      Cette page donne des trucs et astuces en vidéo pour bien utiliser Windows 8.
    </aside>
  </body>
</html>
```

Code039.htm

# Mettre en forme des liens

Par défaut, les liens hypertextes apparaissent en caractères bleus soulignés lorsqu'ils n'ont pas été visités. Ils sont affichés en caractères bordeaux soulignés lorsqu'ils ont été visités.

Vous utiliserez la propriété CSS color pour choisir d'autres couleurs :

- a {color} pour les liens non visités ;
- a:visited {color} pour les liens visités.

Le type de soulignement peut également être choisi avec la propriété CSS text-decoration :

⚠ text-decoration ne fonctionne plus sur le a:visited. Vous pouvez l'utiliser sur d'autres balises sans problèmes.

La propriété text-decoration peut prendre les valeurs suivantes :

- none : aucun soulignement ;
- underline : soulignement par défaut, en dessous du lien ;
- overline : soulignement au-dessus du lien ;
- overline underline : soulignement double, en dessous et au-dessus du lien ;
- line-through : barre transversale sur toute la longueur du lien.

# Mettre en forme des liens

Les propriétés relatives au texte sont également utilisables dans un style relatif aux liens hypertextes. Vous utiliserez en particulier les propriétés suivantes :

- font-family : police(s) à utiliser ;
- font-size : taille des caractères ;
- font-style : orientation des caractères (normal, italic ou oblique) ;
- font-weight : graisse de la police ;
- font-variant : caractères par défaut (normal) ou petites majuscules (small-caps) ;
- text-transform : casse des caractères (none, capitalize, uppercase ou lowercase).

Enfin, notez qu'il est possible d'utiliser la pseudo-classe a:hover, associée à la propriété color, pour changer la couleur des liens hypertextes lorsque le pointeur les survole.

# Coins arrondis et ombres

Avec CSS3, réaliser des boîtes à coins arrondis et ajouter des ombrages sur des éléments quelconques devient un jeu d'enfant. Ce chapitre montre comment procéder en l'état actuel de l'avancement des spécifications et de leur adoption par les différents navigateurs du marché.

Dans cette section :

- Boîte à coins arrondis
- Mise en page avec arrondis
- Ombrage
- Ombrage de texte

# Boîte à coins arrondis

La méthode consiste à utiliser la propriété border-radius. En l'état actuel du développement et de l'adoption de la norme, vous devrez spécifier des préfixes sur cette propriété, en fonction du navigateur :

- -moz pour les navigateurs Gecko (Mozilla Firefox) ;
- -webkit pour les navigateurs Webkit (Safari, Chrome) ;
- -khtml pour les navigateur Konqueror ;
- -ms pour Internet Explorer 9.

La syntaxe de la propriété border-radius est la suivante :

```
border-radius: r1 r2 r3 r4;
```

Où r1, r2, r3 et r4 représentent respectivement le rayon des coins supérieur gauche, supérieur droit, inférieur droit et inférieur gauche de l'élément.

Si un seul rayon est spécifié, il s'applique aux quatre coins de l'élément.

# Exercice

Définissez une bordure noire d'épaisseur 1 px autour d'un élément div en utilisant la propriété border-radius.

# Solution

Code041.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une boîte à coins arrondis en CSS3</title>
    <style>
      .arrondi
      {
        border-radius: 15px;
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <div class="arrondi">Boîte à bords arrondis en CSS3</div>
  </body>
</html>
```

# Exercice

L'arrondi s'étend sur toute la largeur de la page. Il est possible de limiter sa largeur avec la propriété width et même de définir sa hauteur avec la propriété height. D'autre part, le texte à l'intérieur de l'élément div est un peu trop collé sur le bord gauche. Pour résoudre ce problème, il suffit de définir un padding. Pour finaliser la mise en forme, vous pouvez également modifier l'alignement du texte avec la propriété text-align et colorer l'arrière-plan avec la propriété background.

Modifiez le code précédent en tirant profit de toutes ces propriétés.

# Solution

Code042.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une boîte à coins arrondis en CSS3</title>
    <style>
      .arrondi
      {
        border-radius: 15px;
        border: 1px solid black;
        padding: 10px;
        width: 250px;
        background:cyan;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <div class="arrondi">Boîte à bords arrondis en CSS3</div>
  </body>
</html>
```

# Exercice

L'arrière-plan peut également être une image, répétée si nécessaire pour couvrir toute la surface de l'élément. Pour cela, vous utiliserez la propriété background :

```
background: url(url de l'image) repeat;
```

Remplacez l'arrière-plan de la boîte par une image répétée.

# Solution

Code043.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une boîte à coins arrondis en CSS3</title>
    <style>
      .arrondi {
        border-radius: 15px;
        border: 1px solid black;
        padding: 10px;
        width: 250px;
        background: url(_fond.gif) repeat;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <div class="arrondi">Boîte à bords arrondis en CSS3</div>
  </body>
</html>
```

# Exercice

Définissez une boîte et appliquez-lui quatre coins arrondis de rayons différents.

# Solution

Code044.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une boîte à coins arrondis en CSS3</title>
    <style>
      .arrondi {
        border-radius: 50px 50px 200px 50px;
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <p class="arrondi">
      <br>
      Boîte à bords arrondis en CSS3
      <br><br>
    </p>
  </body>
</html>
```

# Mise en page avec arrondis

Tous les éléments HTML5 peuvent être arrondis, y compris les nouveaux : header, footer, nav, aside et article. Dans cet exemple, des arrondis sont appliqués à tous les éléments header, nav, article, section, footer, aside et img du document.

# Exercice

Code045avant.htm

Saisissez le code HTML et le code CSS suivants. Complétez la feuille de styles mep.css pour définir des angles arrondis sur les éléments header, nav, article, section, footer, aside et img.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mise en page en arrondis</title>
    <link rel="stylesheet" href="mep.css">
  </head>
  <body>
    <nav>
      <ul>
        <li><a href="#">Accueil</a></li>
        <li><a href="#">Page 1</a></li>
        <li><a href="#">Page 2</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>

    <aside>
      <p>Contenu texte ajouté sur le côté avec la balise &lt;aside&gt;</p>
    </aside>

    <article>
      <header>
        <h2>Titre de l'article</h2>
      </header>
      <p>Contenu principal de l'article, suivi d'une image.</p>
      <br>
      <footer>
        <a href="#">Afficher les commentaires</a>
      </footer>
    </article>
  </body>
</html>
```

```
header
{
  background-color: red;
}

nav
{
  float:left;
  width:20%;
  background-color:yellow;
}

article
{
  background-color: #66FFFF;
  width:80%;
}

aside
{
  float:right;
  width:20%;
  background-color:yellow;
}

footer
{
  clear:both;
  background-color: #99FF66;
}
```

# Solution

Code045apres.htm

Voici le code CSS à insérer dans la feuille de styles mep.css :

```
header, nav, article, section, footer, aside, img
{
    display: block;
    border-radius: 15px;
    border: 1px solid black;
    padding: 10px;
}
```

# Ombrage sur un élément

L'ombrage des éléments HTML5 non textuels se fait *via* la propriété CSS3 box-shadow :

box-shadow : offset-horiz offset-vert rayon-ombrage couleur

Où :

- offset-horiz est le décalage horizontal de l'ombrage en pixels.
- offset-vert est le décalage vertical de l'ombrage en pixels.
- rayon-ombrage est le rayon de l'ombrage en pixels.
- couleur est la couleur de l'ombrage.

Cette propriété présente deux avantages : elle est compatible avec la plupart des navigateurs actuels (y compris Internet Explorer) et elle s'applique à tous les éléments HTML5. Notez toutefois que vous aurez besoin des habituels préfixes -moz-, -webkit- et -khtml- pour (respectivement) les navigateurs Gecko, Webkit et Konqueror.

# Exercice

Appliquez un ombrage box-shadow à un élément img.

# Solution

Code046.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ombrage CSS3</title>
    <style>
      .ombrageCSS3
      {
        width: 230px;
        box-shadow: 10px 10px 5px #bbb;
      }
    </style>
  </head>
  <body>
    
  </body>
</html>
```

Essayez de modifier les quatre paramètres de la propriété box-shadow.

# Ombrage de texte

L'ombrage CSS3 d'éléments textuels se fait *via* la propriété text-shadow :

text-shadow : offset-horiz offset-vert épaisseur couleur

Où :

- offset-horiz est le décalage horizontal de l'ombrage en pixels.
- offset-vert est le décalage vertical de l'ombrage en pixels.
- épaisseur est l'épaisseur de l'ombrage en pixels.
- couleur est la couleur de l'ombrage.

# Exercice

Appliquez la propriété text-shadow à un élément p qui contient du texte.

# Solution

Code047.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ombrage de texte en CSS3</title>
    <style>
      .otexte
      {
        font-size: 60px;
        text-shadow: 4px 4px 4px grey;
      }
    </style>
  </head>
  <body>
    <p class="otexte">Ce texte est ombré</p>
  </body>
</html>
```

# Mise en page avec arrondis et ombrages

Il est tout à fait possible d'appliquer conjointement les techniques CSS3 permettant de définir des arrondis sur des éléments, celles permettant d'ombrer des éléments non textuels et celles permettant d'ombrer des éléments textuels. Il suffit pour cela de définir les styles CSS nécessaires dans la feuille de styles.

Code048.htm

Mep-arr-omb.css

# Transformations 2D

Avec CSS3, il est désormais très simple d'appliquer des transformations (translations, rotations, changements d'échelle). Et, bonne nouvelle, les transformations sont supportées par la plupart des navigateurs de dernière génération : Webkit (Safari, Chrome), Gecko (Firefox), Opera et Internet Explorer 9 et ultérieur.

Les transformations n'affectent pas le flux. En d'autres termes, les divers éléments qui environnent un élément ne seront pas déplacés suite à sa transformation. Cependant, vous pourrez utiliser la propriété overflow de l'élément transformé pour déterminer le rendu.

Dans cette section :

- Translations
- Rotations
- Changements d'échelle
- Inclinaisons
- Matrices

# Translations

Tous les éléments HTML5 peuvent subir une translation par l'intermédiaire de la fonction translate() appliquée à la propriété transform. Voici la syntaxe à utiliser :

```
transform: translate(h, v);
```

Où h définit le déplacement horizontal (positif pour un déplacement vers la droite, négatif pour un déplacement vers la gauche) et v le déplacement vertical (positif pour un déplacement vers le bas, négatif pour un déplacement vers le haut).

Voici un exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Translations</title>
    <style>
      div
      {
        width: 200px;
        height: 60px;
        background-color: SkyBlue;
        margin: 20px;
        border-color: black;
        border-width: 5px;
        border-style: dashed;
      }
      .droite
      {
        transform : translate(20px, 0px);
      }
      .gauche
      {
        transform : translate(-20px, 0px);
      }
    </style>
  </head>
  <body>
    <div>
      Boîte sans déplacement
    </div>
    <div class="droite">
      Boîte déplacée à droite de 20px
    </div>
    <div class="gauche">
      Boîte déplacée à gauche de 20px
    </div>
  </body>
</html>
```

Code049.htm

Entraînez-vous à effectuer d'autres translations en modifiant les paramètres de la fonction translate.

# Rotations

Tous les éléments HTML5 peuvent subir une rotation *via* la fonction rotate() appliquée à la propriété transform. Voici la syntaxe à utiliser :

```
transform: rotate (rdeg);
```

Où r est l'angle de la rotation en degrés. Une valeur positive provoque une rotation dans le sens horaire, et une valeur négative provoque une rotation dans le sens trigonométrique.

Voici un exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Rotations</title>
    <style>
      div
      {
        width: 200px;
        height: 60px;
        background-color: SkyBlue;
        margin-top: 120px;
        border-color: black;
        border-width: 5px;
        border-style: dashed;
        display: inline-block;
      }
      .rotdroite45
      {
        -webkit-transform : rotate(45deg);
        -moz-transform : rotate(45deg);
        -o-transform : rotate(45deg);
        -ms-transform : rotate(45deg);
      }
      .rotgauche90
      {
        -webkit-transform : rotate(-90deg);
        -moz-transform : rotate(-90deg);
        -o-transform : rotate(-90deg);
        -ms-transform : rotate(-90deg);
      }
    </style>
  </head>
  <body>
    <div style="border-width: 5px;">
      La &lt;div&gt; originale
    </div>
    <div class="rotdroite45">
      La &lt;div&gt; après une rotation à droite de 45°
    </div>
    <div class="rotgauche90">
      La &lt;div&gt; après une rotation à gauche de 90°
    </div>
  </body>
</html>
```

Code050.htm

Entraînez-vous à effectuer d'autres rotations en modifiant les paramètres de la fonction rotate.

# Changements d'échelle

Les éléments HTML5 peuvent subir un changement d'échelle quand la fonction scale() est appliquée à la propriété CSS3 transform. Voici la syntaxe à utiliser :

```
transform: scale(ex, ey);
```

Où ex et ey représentent les facteurs d'échelle selon les axes des abscisses et des ordonnées. Par défaut, si ey n'est pas spécifié, il est supposé égal à ex.

Voici un exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Changements d'échelle</title>
    <style>
      .e200
      {
        -webkit-transform : scale(2);
        -moz-transform : scale(2);
        -o-transform : scale(2);
        -ms-transform : scale(2);
      }
      .e50
      {
        -webkit-transform : scale(.5);
        -moz-transform : scale(.5);
        -o-transform : scale(.5);
        -ms-transform : scale(.5);
      }
    </style>
  </head>
  <body>
    
    
    
  </body>
</html>
```

Code051.htm

# Inclinaison

Les éléments HTML5 peuvent être inclinés avec la fonction skew() appliquée à la propriété CSS3 transform. Voici la syntaxe à utiliser :

```
transform: skew(ax, ay);
```

Où ax et ay représentent les angles d'inclinaison selon les axes des abscisses et des ordonnées. Les valeurs positives de ax inclinent l'élément dans le sens trigonométrique. Quant aux valeurs positives de ay, elles l'inclinent vers l'arrière. Par défaut, si ay n'est pas spécifié, il est supposé égal à 0. Dans ce cas, il n'y a donc pas d'inclinaison selon l'axe des ordonnées.

# Exercice

Définissez le code HTML5 et CSS3 nécessaire pour incliner une image comme ceci :

- 15 degrés sur les axes X et Y
- -25 degrés sur les axes X et Y

Voici le résultat à obtenir



# Solution

Code052.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Inclinaison</title>
    <style>
      img
      {
        margin-top: 60px;
        margin-left: 40px;
      }
      .xy15
      {
        transform : skew(15deg, 15deg) ;
      }
      .xy-25
      {
        transform : skew(-25deg, -25deg) ;
      }

    </style>
  </head>
  <body>
    
    
    
  </body>
</html>
```

# Matrices

D'un point de vue mathématique, toutes les transformations CSS3 peuvent être représentées par une matrice 3x3 de la forme suivante :

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice de transformation permet de relier les anciennes et les nouvelles coordonnées des éléments transformés avec la relation suivante :

$$\begin{bmatrix} \text{ancien X} \\ \text{ancien Y} \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \text{nouveau X} \\ \text{nouveau Y} \\ 1 \end{bmatrix}$$

# Matrices

Les transformations étudiées jusqu'ici sont équivalentes aux matrices représentées sur cette figure.

Translation

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

Echelle

$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation par rapport à l'origine

$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inclinaison sur l'axe des X

$$\begin{bmatrix} 1 & \tan(a) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inclinaison sur l'axe des Y

$$\begin{bmatrix} 1 & 0 & 0 \\ \tan(a) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Matrices

Six valeurs seulement sont utilisées dans ces matrices, c'est pourquoi il est possible de les exprimer en tant que simples vecteurs [ a b c d e f ].

Si vous êtes à l'aise avec la manipulation des matrices, vous pouvez employer la fonction matrix() appliquée à la propriété CSS3 transform. En voici la syntaxe :

```
transform: matrix(a, b, c, d, e, f);
```

Où a, b, c, d, e et f sont les coefficients de la matrice  $3 \times 3$ .

# Matrices

Pour effectuer une rotation de 90 degrés dans le sens des aiguilles d'une montre, vous utiliserez la matrice suivante :

$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Les coefficients sont donc les suivants : 0, 1, -1, 0, 0, 0.

## Exercice

Ecrivez le code nécessaire pour appliquer une rotation de 90° sur une image en utilisant une matrice.

# Solution

Code053.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Matrices de transformation</title>
    <style>
      img
      {
        margin-top: 60px;
        margin-left: 40px;
      }
      .mat90
      {
        transform : matrix(0, 1, -1, 0, 0, 0);
      }

    </style>
  </head>
  <body>
    
    
  </body>
</html>
```

# Transitions

Dans cette section :

- Modification progressive d'une propriété
- Animer une transformation
- Déclencher une animation en JavaScript

# Transition

Vous pouvez animer les transformations CSS3 avec des pseudo-classes ou du code JavaScript comme facteurs déclencheurs. Ainsi par exemple, une rotation ou une translation peut être affichée progressivement lorsque l'utilisateur déplace le pointeur ou clique sur un élément quelconque. Cette fabuleuse nouveauté réside dans la propriété CSS3 transition dont voici la syntaxe :

```
transition: propriété durée type délai;
```

Où :

propriété est une propriété CSS : background, opacity ou width par exemple.

durée est la durée de l'animation en secondes.

type est le type de l'animation. Ce paramètre peut prendre l'une des valeurs suivantes :

- linear : aucun effet ;
- cubic-bezier(x1,y1,x2,y2) : courbe de Bézier dont les points P<sub>0</sub> et P<sub>3</sub> ont pour valeur (0,0) et (1,1) ;
- ease : équivalent à cubic-bezier(0.25,0.1,0.25,1.0) ;
- ease-in : (0.42,0,1.0,1.0) ;
- ease-out : (0,0,0.58,1.0) ;
- ease-in-out : (0.42,0,0.58,1.0).

délai est le délai avant le déclenchement de l'animation.

# Modification progressive d'une couleur

Pour modifier progressivement une couleur, vous devez :

- définir une classe contenant la propriété transition: color et l'affecter à l'élément concerné ;
- définir la pseudo-classe associée et fixer la couleur à atteindre.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Animation de la couleur</title>
    <style>
      .anim-couleur
      {
        font-size: 40px;
        transition: color 2s ease-in;
      }
      .anim-couleur:hover
      {
        color: red;
      }
    </style>
  </head>
  <body>
    <p class="anim-couleur">Placez le pointeur sur ce texte</p>
  </body>
</html>
```

Code054.htm

# Modification progressive de la transparence

La propriété opacity permet de régler la transparence d'un élément. Pour faire disparaître un élément au survol, il suffit de définir :

- Une classe dans laquelle la propriété transition: opacity est initialisée et l'affecter à l'élément concerné.
- La pseudo-classe correspondante dans laquelle l'opacité cible est spécifiée.

# Exercice

Insérez une image dans un document HTML5 et faites-la disparaître progressivement lorsqu'elle est survolée par le pointeur de la souris.

Pour cela, vous utiliserez la propriété CSS3 transition.

# Solution

Code055.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Animation de la transparence</title>
    <style>
      .anim-opacite {
        transition: opacity 2s linear;
      }
      .anim-opacite:hover {
        opacity: 0;
      }
    </style>
  </head>
  <body>
    
  </body>
</html>
```

# Animer une transformation

En utilisant conjointement les propriétés CSS3 transform et transition, vous n'aurez besoin que de quelques lignes de code pour animer une transformation.

## Exercice

Insérez une image dans un document HTML5 et déplacez-la progressivement de 50 pixels vers la droite lorsqu'elle est survolée par la souris.

# Solution

Code056.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Animation d'une translation</title>
    <style>
      .animation {
        transition: all 2s linear;
      }
      .animation:hover {
        transform: translate(50px, 0);
      }
    </style>
  </head>
  <body>
    
  </body>
</html>
```

La valeur all étant appliquée au premier paramètre, toutes les propriétés modifiées pour cet élément seront automatiquement animées. Ici, seule une translation est appliquée à l'image, mais on aurait pu également ajouter une disparition progressive, une rotation ou un quelconque autre effet.

On aurait également pu effectuer une transition sur la propriété transform (et non sur all) pour parvenir au même résultat.

# Exercice

Modifiez le code de l'exercice précédent pour effectuer simultanément trois transformations lorsque l'image est survolée :

- Une translation de 50 pixels ;
- Une rotation de 360 degrés ;
- Une modification de l'opacité pour que l'image devienne entièrement transparente.

# Solution

Code057.htm

Seul le code de la pseudo-classe .animation:hover doit être modifié.  
Remarquez l'utilisation conjointe des fonctions translate() et rotate()

```
.animation:hover
{
    transform: translate(400px, 0) rotate(360deg);
    opacity: 0;
}
```

# Déclencher une animation en JavaScript

Il peut être intéressant de déclencher une animation par une fonction JavaScript. Cette fonction étant associée à la propriété onclick d'un élément HTML button par exemple, l'animation sera déclenchée lorsque l'utilisateur cliquera sur un bouton.

Le code de la diapositive suivante montre comment déclencher une animation sur l'attribut top d'une <div> à la suite d'un clic sur un <button>, puis comment restituer la position initiale de la <div> à la suite d'un clic sur un autre bouton.

## Code058.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Animation déclenchée en JavaScript</title>
    <style>
      div
      {
        position: absolute;
        left: 100px;
        top: 100px;
        height: 100px;
        width: 100px;
        background: Yellow;
        transition: top 1s ease-in;
      }
    </style>

    <script>
      function animer()
      {
        var el = document.getElementById('madiv');
        el.setAttribute("style","top: 150px;");
      }
      function reinit()
      {
        var el = document.getElementById('madiv');
        el.setAttribute("style","top:100px;");
      }
    </script>
  </head>

  <body>
    <button onclick="animer()">Cliquez ici pour animer la &lt;div&gt;</button>
    <button onclick="reinit()">Cliquez ici pour réinitialiser la &lt;div&gt;</button>
    <div id="madiv">Une simple &lt;div&gt;</div>
  </body>
</html>
```

# Formulaires

Dans cette section :

- Eléments dédiés aux formulaires
- Placeholder
- Autofocus
- Validation de données
- Labels implicites
- Aligner les éléments d'un formulaire
- Arrondir les angles des éléments d'un formulaire
- Modifier l'allure d'un bouton

# Formulaires

Si les pages web servent à diffuser des informations, elles sont également en mesure d'en recueillir par l'intermédiaire de formulaires. Les données saisies sont alors communiquées à un programme ou expédiées sans traitement dans une boîte e-mail.

Les diapos suivantes résument les différentes balises utilisables en HTML5.

Balise	Signification
<code>&lt;fieldset&gt;contenu&lt;/fieldset&gt;</code>	Groupe d'options permettant de rassembler plusieurs champs.
<code>&lt;legend&gt;texte&lt;/legend&gt;</code>	Légende affichée dans un élément <u>fieldset</u> .
<code>&lt;label&gt;texte&lt;/label&gt;</code>	Simple légende textuelle.
<code>&lt;input type="text" name="Nom" id="identifiant" value="valeur"&gt;</code>	Zone de texte.
<code>&lt;input type="button" name="Nom" id="identifiant" value="Légende du bouton"&gt;</code>	Bouton.
<code>&lt;input type="image" src="image.jpg" id="identifiant"&gt;</code>	Un bouton contenant une image.
<code>&lt;input type="password" name="Nom" id="identifiant" value="valeur par défaut"&gt;</code>	Zone de saisie d'un mot de passe.
<code>&lt;input type="checkbox" name="Nom" id="identifiant" value="Valeur"&gt;</code>	Case à cocher.
<code>&lt;input type="radio" name="Nom" id="identifiant" value="Valeur"&gt;</code>	Bouton radio.

Balise	Signification
<code>&lt;input type="submit" name="Nom" id="identifiant" value="Texte affiché sur le bouton"&gt;</code>	Bouton d'envoi, pour envoyer les données du formulaire au serveur.
<code>&lt;input type="reset" name="Nom" id="identifiant" value="Texte affiché sur le bouton"&gt;</code>	Bouton de réinitialisation du formulaire.
<code>&lt;input type="date" name="Nom" id="identifiant" title="Date(jj-mm-aaaa)" value="Une date"&gt;</code>	Champ spécialisé dans la saisie de dates (seulement pour les navigateurs Opera et Chrome alors que j'écris ces lignes [voir Figure 12.1]).
<code>&lt;input type="time" name="Nom" id="identifiant" title="Heure(hh-mm)" value="10:00"&gt;</code>	Champ spécialisé dans la saisie d'heures (seulement pour les navigateurs Opera et Chrome alors que j'écris ces lignes).
<code>&lt;input type="datetime-local" name="Nom" id="identifiant"&gt;</code>	Champ spécialisé dans la saisie de date et heures (seulement pour les navigateurs Opera et Chrome alors que j'écris ces lignes).
<code>&lt;input type="number" name="Nom" id="identifiant" value="55"&gt;</code>	Champ spécialisé dans la saisie des nombres.
<code>&lt;input type="color" name="Nom" id="identifiant" value="red"&gt;</code>	Palette de couleurs (aucun navigateur n'est encore compatible).
<code>&lt;input type="search" name="Nom" id="identifiant" list="datalist" value="Valeur par défaut"&gt;</code>	Zone de texte permettant d'effectuer des recherches.

Balise	Signification
<code>&lt;input type="email" name="Nom" id="identifiant"&gt;</code>	Champ de saisie spécialisé pour recevoir des adresses e-mail.
<code>&lt;input type="url" name="Nom" id="identifiant"&gt;</code>	Champ de saisie spécialisé pour recevoir des adresses URL.
<code>&lt;input type="file" name="Nom" id="identifiant"&gt;</code>	Désignation d'un fichier local.
<code>&lt;datalist id="identifiant"&gt;&lt;option value="valeur1"&gt;...&lt;option value="valeurN"&gt;&lt;/datalist&gt;</code>	Facilite la saisie en implémentant l'autocomplete sur plusieurs valeurs prédéfinies dans des éléments <u>option</u> (aucun navigateur n'est encore compatible).
<code>&lt;textarea cols="Nombre colonnes" rows="Nombre lignes" id="identifiant"&gt;Texte par défaut&lt;/textarea&gt;</code>	Zone de saisie multiligne.
<code>&lt;select name="Nom" id="identifiant"&gt;&lt;option value="valeur1"&gt;...&lt;option value="valeurN"&gt;&lt;/select&gt;</code>	Liste déroulante.
<code>&lt;select name="Nom" size="4" id="identifiant"&gt;&lt;option value="valeur1"&gt;...&lt;option value="valeurN"&gt;&lt;/select&gt;</code>	Zone de liste (ici, quatre éléments sont affichés).
<code>&lt;output name="Nom"&gt;</code>	Affichage d'un résultat.

# Formulaires

Les éléments qui composent un formulaire doivent obligatoirement être inclus entre des balises <form> et </form> dont la syntaxe est la suivante :

```
<form name="nom" action="programme ou action" method="get|post">  
</form>
```

Où :

- name est le nom du formulaire.
- action est le nom du programme auquel les données doivent être transmises ou l'action à accomplir (mailto:adresse e-mail par exemple).
- method est la méthode de transmission des données du formulaire : get poste les données dans l'adresse URL et post dans le corps de la requête. La seconde méthode est préférable à la première car elle n'affiche pas en clair les données saisies dans le formulaire (ce qui pourrait être gênant si des données confidentielles sont transmises). De plus, la taille des données n'est pas limitée, alors qu'elle ne peut dépasser 256 octets dans la méthode get.

## Voici un exemple de formulaire

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Un formulaire</title>
  </head>
  <body>
    <form name="MonFormulaire" method="post" action="traitement.php">
      <fieldset>
        <legend>Légende du &lt;fieldset&gt;</legend>
        Contenu du du &lt;fieldset&gt;;
      </fieldset>

      <br><label>Un champ texte</label>
      <input type="text" name="Nom" value="valeur">

      <br><label>Un bouton</label>
      <input type="button" name="Nom" value="Légende du bouton">

      <br><label>Un bouton image</label>
      <input type="image" src="commande.jpg">

      <br><label>Un champ mot de passe</label>
      <input type="password" name="Nom" value="valeur par défaut">

      <br><label>Une case à cocher</label>
      <input type="checkbox" name="Nom" value="Valeur">
      <label>Un bouton radio</label>
      <input type="radio" name="Nom" value="Valeur">

      <br><label>Des boutons &lt;submit&gt; et &lt;reset&gt;</label>
      <input type="submit" name="Nom" value="Envoyer">
      <input type="reset" name="Nom" value="Annuler">

      <br><label>Champs spécialisés date et heure</label>
      <input type="date" name="Nom" title="Date (aaaa-mm-jj)" value="2010-11-22">
      <input type="time" name="Nom" title="Heure (hh-mm)" value="10:00">
      <input type="datetime-local" name="Nom">

      <br><label>Saisie de nombres entiers autour de la valeur 55</label>
      <input type="number" name="Nom" value="55">
```

```
<br><label>Une zone de texte spécialisée recherche</label>
    <input type="search" name="Nom" list="datalist" value="Valeur par
défaut">

    <br><label>Un curseur pour sélectionner des valeurs numériques</label>
    <input type="range" min="-500" max="500" value="0" step="10" name="Nom">

    <br><label>Un bouton pour désigner un fichier local</label>
    <input type="file" name="Nom">

    <br><label>Une zone de saisie multiligne</label>
    <textarea cols="40" rows="4">Texte par défaut</textarea>

    <br><label>Liste déroulante et zone de liste</label>
    <select name="Nom">
        <option>Valeur 1</option>
        <option>Valeur 2</option>
        <option>Valeur 3</option>
        <option>Valeur 4</option>
        <option>Valeur 5</option>
    </select>

    <select size="2" name="Nom">
        <option>Valeur 1</option>
        <option>Valeur 2</option>
        <option>Valeur 3</option>
        <option>Valeur 4</option>
        <option>Valeur 5</option>
    </select>

    <br><label>Résultat d'un calcul : </label>
    <output name="resultat">123</output>
</form>
</body>
</html>
```

# Placeholder

Code060.htm

L'attribut placeholder des balises `<input type="text">` permet d'ajouter un texte grisé dans la zone de saisie. Ce texte est visible à l'affichage de la page, puis il disparaît lorsque la zone de saisie a le focus, laissant ainsi l'utilisateur taper son texte.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Une zone de texte améliorée</title>
  </head>
  <body>
    <form name="MonFormulaire" method="post" action="traitement.php">
      <label>Quel est le meilleur système d'exploitation selon vous ?</label>
      <input name="texte" placeholder="Entrez votre réponse ici">
      <input type="submit" value="Envoyer">
    </form>
  </body>
</html>
```

Essayez ce code

# Autofocus

Code061.htm

Pour donner le focus à un champ d'un formulaire, il suffit d'utiliser l'attribut autofocus sur le champ concerné.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Utilisation de l'attribut Autofocus</title>
  </head>
  <body>
    <form action="action.php" method="post">
      <fieldset>
        <legend>Positionnement du point d'insertion dans le premier champ input</legend>
        <label>Nom</label>
        <input type="text" id="champ1" autofocus>
        <label>Prénom</label>
        <input type="text" id="champ2">
        <label>Adresse</label>
        <input type="text" id="champ3">
        <label>Ville</label>
        <input type="text" id="champ4">
      </fieldset>
    </form>
  </body>
</html>
</html>
```

Essayez ce code

# Validation de données

Lorsqu'un champ de saisie doit obligatoirement être rempli, il suffit de lui affecter l'attribut required. Dans ce formulaire, le champ de saisie Nom est obligatoire. Si l'utilisateur clique sur le bouton Submit sans l'avoir renseigné, un message d'erreur est généré et le formulaire n'est pas envoyé.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Champ de saisie obligatoire</title>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label>
      <input name="Nom" required>
      <br><br><br><br>
      <input type=submit value="Valider">
    </form>
  </body>
</html>
```

Code062.htm

Essayez ce code

# Validation de données

Le champ d'action des attributs required est bien plus étendu, puisqu'il permet également de valider des données complexes (telles que des adresses e-mail ou des URL). À titre d'exemple, le formulaire ci-après teste la validité des champs email et url. Lorsqu'on clique sur le bouton Submit, un message d'erreur s'affiche si un de ces deux champs n'est pas conforme.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Validation e-mail et url</title>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label> <input name="Nom"><br>
      <label>e-mail</label> <input type="email" name="email" required><br>
      <label>Adresse URL</label> <input type="url" name="url" required><br>
      <label>Commentaire</label> <textarea name="comment"></textarea><br>
      <input type=submit value="Valider">
    </form>
  </body>
</html>
```

Code063.htm

Essayez ce code

# Validation de données

Pour faciliter la reconnaissance des champs obligatoires, il est possible d'utiliser quelques lignes de CSS. Par exemple, cette ligne affecte un arrière-plan de couleur jaune aux champs `<input>` dont l'attribut `required` est spécifié :

```
input:required { background:yellow }
```

# Labels implicites et explicites

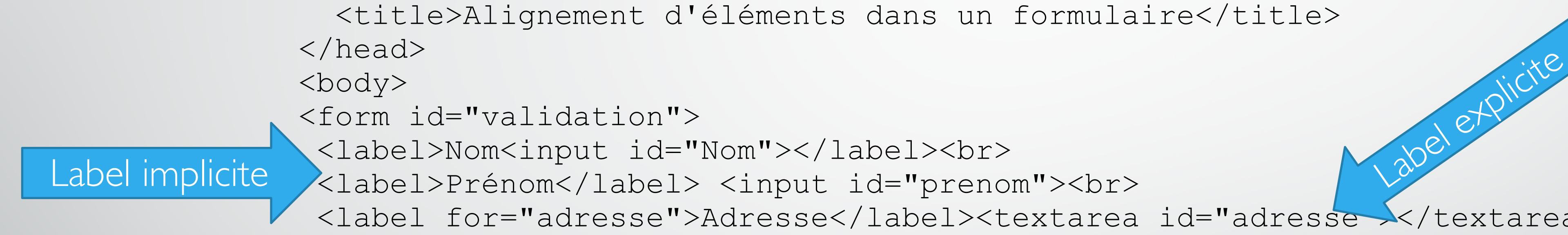
Les labels spécifiés dans un formulaire peuvent être reliés de façon implicite ou explicite à un champ de saisie, qu'il s'agisse d'un `<input type="text">`, d'un `<input type="checkbox">`, d'un `<textarea>` ou d'un quelconque autre champ de saisie.

On réalise la liaison implicite en définissant le champ de saisie comme enfant de l'élément `label`, c'est-à-dire en l'insérant entre les balises `<label>` et `</label>`.

La liaison explicite se fait à travers l'attribut `for` de l'élément `label`, qui fait référence à l'`id` du champ cible.

# Labels implicites et explicites

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Alignement d'éléments dans un formulaire</title>
  </head>
  <body>
    <form id="validation">
      <label>Nom<input id="Nom"></label><br>
      <label>Prénom</label> <input id="prenom"><br>
      <label for="adresse">Adresse</label><textarea id="adresse"></textarea><br>
    </form>
  </body>
</html>
```



Les labels implicites provoquent des problèmes d'accessibilité. Vous utiliserez des labels explicites légèrement modifiés.

# Labels implicites et explicites

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Alignement d'éléments dans un formulaire</title>
  </head>
  <body>
    <form id="validation">
      <label for="nom">Nom<input id="nom"></label><br>
      <label for="prenom">Prénom</label> <input id="prenom"><br>
      <label for="adresse">Adresse</label><textarea id="adresse"></textarea><br>
    </form>
  </body>
</html>
```

La valeur de l'attribut id de l'élément input et celle de l'attribut for de l'élément label doivent être identiques.

# Aligner les éléments d'un formulaire

D'une façon générale, les divers éléments utilisés dans un formulaire (labels, zones de texte, cases à cocher, listes déroulantes, etc.) sont de type inline. Pour pouvoir modifier leur taille en utilisant des propriétés CSS, vous devez les redéfinir en inline-block.

Dans la diapo suivante, nous allons aligner à gauche deux zones de texte et un textarea en fixant la largeur des labels qui précèdent chacun d'eux.



## Code064.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Labels implicites et explicites</title>
    <style>
      label
      {
        display: inline-block;
        width: 80px;
      }
    </style>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label><input id="Nom"><br>
      <label>Prénom</label><input id="Prenom"><br>
      <label>Adresse</label><textarea id="Adresse" rows="8" cols="30"></textarea><br><br>
      <input type="button" name="Nom" value="Valider">
    </form>
  </body>
</html>
```

Essayez ce code

# Aligner les éléments d'un formulaire

Une autre technique consiste à utiliser un positionnement absolu.

Sans la diapositive suivante, les éléments input et textarea sont alignés à 85px du bord gauche.

La propriété top de l'élément button doit également être initialisée pour éviter un chevauchement du textarea et du button.

## Code065.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Alignement d'éléments dans un formulaire</title>
    <style>
      input, textarea
      {
        position: absolute;
        left: 85px;
      }
      .button
      {
        position: absolute;
        left: 85px;
        top: 200px;
      }
    </style>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label><input id="Nom"><br>
      <label>Prénom</label><input type=email id="email"><br>
      <label>Adresse</label><textarea id="adresse" rows="8" cols="30"></textarea><br><br>
      <input type="button" name="Nom" value="Valider" class="button">
    </form>
  </body>
</html>
```

Essayez ce code

# Arrondir les angles des éléments d'un formulaire

La propriété CSS3 border-radius permet d'arrondir les angles de tous les éléments HTML5, y compris des éléments utilisés dans un formulaire. Ici par exemple, les angles de deux zones de texte, d'un textarea et d'un bouton sont arrondis.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Arrondir les angles d'un formulaire</title>
    <style>
      .arrondi
      {
        -moz-border-radius: 15px;
        -webkit-border-radius: 15px;
        -khtml-border-radius: 15px;
        -ms-border-radius: 15px;
        border-radius: 15px;
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label><input id="Nom" class="arrondi"><br>
      <label>Prénom</label><input id="Prenom" class="arrondi"><br>
      <label>Adresse</label><textarea id="adresse" class="arrondi" rows="8" cols="30"></textarea><br><br>
      <input type="button" name="Nom" value="Valider" class="arrondi">
    </form>
  </body>
</html>
```

Code066.htm

Essayez ce code

# Modifier l'allure d'un bouton avec un style

Plusieurs techniques CSS3 peuvent être appliquées pour définir le style d'un bouton :

- incorporation de polices sur le serveur ;
- gradient de couleur pour définir l'arrière-plan ;
- différents arrondis sur les angles ;
- ombre portée.

## Exercice

Ecrivez du code HTML pour donner l'allure suivante à un bouton :



## Code067.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Un bouton de formulaire entièrement redéfini</title>
    <style>
      @font-face {
        font-family: algerian;
        src: url('ALGER.TTF');
      }
      .arrondi {
        width: 220px;
        height: 50px;
        -moz-border-radius: 50px 50px 200px 50px; -webkit-border-radius: 50px 50px 200px 50px;
        -khtml-border-radius: 50px 50px 200px 50px; -ms-border-radius: 50px 50px 200px 50px;
        border-radius: 50px 50px 200px 50px; border: 1px solid black;
        background: -moz-linear-gradient(top, #40FFFF, #C0FFFF);
        background: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#40FFFF), to(#C0FFFF));
        box-shadow: 5px 5px 5px grey;
        -moz-box-shadow: 5px 5px 5px grey; -webkit-box-shadow: 5px 5px 5px grey;
        -khtml-box-shadow: 5px 5px 5px grey; box-shadow: 5px 5px 5px grey;
        font-family: 'algerian'; font-size: 40px;
      }
      .arrondi:hover {
        background: -moz-linear-gradient(top, #C0FFFF, #40FFFF);
        background: -ms-linear-gradient(top, #C0FFFF, #40FFFF);
        background: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#C0FFFF), to(#40FFFF));
      }
    </style>
  </head>
  <body>
    <form id="validation">
      <input type="button" name="bouton1" value="Valider" class="arrondi">
    </form>
  </body>
</html>
```

# I5 – Eléments graphiques et multimédias

Dans cette section :

- audio
- vidéo
- canvas
- Méthodes de tracé 2D
- Images dans un canvas
- Texte dans un canvas
- Graphiques vectoriels SVG

# L'élément audio

Avec l'élément audio, l'insertion d'un objet audio dans une page web devient un vrai jeu d'enfant. D'autre part, il n'est plus nécessaire de se soucier de savoir si l'internaute a installé les bons codecs : ces derniers sont inclus de façon native dans le langage !

Voici la syntaxe de l'élément audio :

```
<audio src="nom" controls preload="none|metadata|auto" loop autoplay>
```

Où :

- src="nom" définit le chemin et le nom du fichier audio.
- controls, s'il est présent, demande l'affichage d'une barre de contrôle pour agir sur le son.
- preload indique comment le son doit être téléchargé avant qu'il ne soit joué : none (aucun téléchargement), metadata (téléchargement des métadonnées associées uniquement) ou auto (laisse le navigateur décider).
- loop, s'il est présent, provoque la lecture sans fin du son.
- autoplay, s'il est présent, déclenche la lecture du son dès que possible, en accord avec l'attribut preload.

# L'élément audio

L'attribut src peut être spécifié en dehors de l'élément audio sous la forme de un ou de plusieurs éléments source. Ainsi, en fonction du navigateur utilisé, un format audio ou un autre sera utilisé :

```
<audio controls preload="auto">
  <source src="nom" type="type du son">
  <source src="nom" type="type du son">
  <source src="nom" type="type du son">
  ...
</audio>
```

Où **type du son** vaut audio/ogg, audio/aac, audio/mp3, audio/wav, audio/aiff ou audio/au.

Tous les navigateurs ne sont pas compatibles avec tous les formats audio. C'est la raison pour laquelle vous devez, dans la mesure du possible, fournir un fichier audio dans les formats OGG, AAC, MP3, WAV, AIFF et AU.

# Exercice

Ecrivez le code HTML5 nécessaire pour lire un fichier audio au format OGG, AAC, MP3, WAV, AIFF ou AU.

# Solution

Code084.htm

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément audio</title>
  </head>

  <body>
    <audio controls preload="auto">
      <source src="son.ogg" type="audio/ogg">
      <source src="son.aac" type="audio/aac">
      <source src="son.mp3" type="audio/mp3">
      <source src="son.wav" type="audio/wav">
      <source src="son.aiff" type="audio/aiff">
      <source src="son.au" type="audio/au">
      <a href="son.mp3">Télécharger <cite>le commentaire audio au format mp3</cite></a>
    </audio>
  </body>
</html>
```

# L'élément video

Avec l'élément video, l'insertion d'un objet vidéo dans une page web n'est plus une corvée. Voici la syntaxe à utiliser :

```
<video src="nom" controls preload="none|metadata|auto" loop autoplay>
```

Où :

- src="nom" définit le chemin et le nom du fichier vidéo, au format OGG ou MP4.
- controls, s'il est présent, demande l'affichage d'une barre de contrôle pour agir sur la vidéo.
- preload indique comment la vidéo doit être téléchargée avant qu'elle ne soit jouée : none (aucun téléchargement), metadata (téléchargement des métadonnées associées uniquement) ou auto (laisse le navigateur décider).
- loop, s'il est présent, provoque la lecture sans fin de la vidéo.
- autoplay, s'il est présent, déclenche la lecture de la vidéo dès que possible, en accord avec l'attribut preload.

# L'élément video

L'attribut src peut être spécifié en dehors de l'élément video sous la forme d'un ou de plusieurs éléments source. Ainsi, en fonction du navigateur, un format vidéo ou un autre sera utilisé.

```
<video controls>
  <source src="une video" type="un format de video">
  <source src="une video" type="un format de video">
  <source src="une video" type="un format de video">
  ...
</video>
```

Où type peut valoir :

- video/mp4 : vidéo au format MP4 codé en H264 (IE, Safari, Chrome) ;
- video/ogg : vidéo au format OGG (Firefox, Safari, Chrome) ;
- video/webm : vidéo au format WEBM (Opera, Chrome, Firefox, IE 9+).

# L'élément canvas

L'élément canvas fait partie intégrante de HTML5. Il permet d'effectuer des rendus dynamiques 2D et 3D en JavaScript. Sa syntaxe est élémentaire :

```
<canvas id="identifiant" width="largeur" height="hauteur">  
    Texte affiché si le navigateur ne supporte pas la balise canvas  
</canvas>
```

Une fois l'élément canvas mis en place dans le corps du document, quelques lignes de JavaScript vont permettre de récupérer l'objet canvas et le contexte de travail 2D ou 3D.

```
var elem = document.getElementById('identifiant');  
var context = elem.getContext('2d');
```

Ces deux éléments récupérés, vous devez ensuite utiliser une ou plusieurs fonctions pour effectuer vos tracés.

# Tracé d'un rectangle

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément canvas</title>
    <script>
      function init()
      {
        var elem = document.getElementById('cannevas');
        if (!elem)
        {
          return;
        }
        var context = elem.getContext('2d');
        if (!context)
        {
          return;
        }
        context.fillStyle = '#ff0';
        context.fillRect(0, 0, 150, 100);
      }
    </script>
  </head>

  <body onload="init();">
    <canvas id="cannevas" width="400" height="250">
      Votre navigateur ne supporte pas la balise HTML5 &lt;canvas&gt;;
    </canvas>
  </body>
</html>
```

Code086.htm

Test de compatibilité du navigateur

Obtention du contexte 2D

Tracé dans le canvas

# Méthodes de tracé 2D

Le contexte 2D d'un élément `<canvas>` met à disposition plusieurs fonctions JavaScript qui permettent de créer des graphiques en deux dimensions.

Fonction	Syntaxe	Effet
<code>fillRect</code>	<code>fillRect(x1, y1, largeur, hauteur);</code>	Rectangle plein ou point élémentaire si largeur=1 et hauteur=1.
<code>strokeRect</code>	<code>strokeRect(x1, y1, x2, y2);</code>	Contour de rectangle.
<code>clearRect</code>	<code>clearRect(x1, y1, x2, y2);</code>	Efface une zone rectangulaire.
<code>beginPath</code>	<code>beginPath()</code>	Définit le début d'un chemin (sorte de tracé à main levée).
<code>closePath</code>	<code>closePath()</code>	Ferme le chemin en cours.
<code>lineTo</code>	<code>lineTo(x, y)</code>	Trace une ligne droite entre le dernier point du chemin et les coordonnées spécifiées en argument.
<code>moveTo</code>	<code>moveTo(x, y)</code>	Définit le début d'un nouveau chemin avec les coordonnées spécifiées en argument.
<code>stroke</code>	<code>stroke()</code>	Trace les chemins définis en utilisant les propriétés de tracé définies dans <code>strokeStyle</code> (style du tracé), <code>lineWidth</code> (épaisseur des traits), <code>lineCap</code> (terminaisons), <code>lineJoin</code> (jointures) et <code>miterLimit</code> (distance limite pour joindre les points éloignés).
<code>fill</code>	<code>fill()</code>	Remplit le chemin courant en utilisant une couleur, un gradient ou une image.
<code>arc</code>	<code>arc(x, y, rayon, angleDébut, angleFin, sens)</code>	Cercle ou arc de cercle. Les angles doivent être spécifiés en radians et le paramètre <code>sens</code> prend la valeur <code>false</code> pour un tracé dans le sens des aiguilles d'une montre et <code>true</code> pour un tracé dans le sens trigonométrique.
<code>bezierCurveTo</code>	<code>bezierCurveTo(ctrl1x, ctrl1y, ctrl2x, ctrl2y, x, y)</code>	Courbe de Bézier entre le chemin courant et le point de coordonnées <code>x, y</code> , qui utilise les points de contrôle de coordonnées <code>ctrl1x, ctrl1y</code> et <code>ctrl2x, ctrl2y</code> .
<code>quadraticCurveTo</code>	<code>quadraticCurveTo(ctrlx, ctrly, x, y)</code>	Courbe quadratique entre le chemin courant et le point de coordonnées <code>x, y</code> , qui utilise les points de contrôle de coordonnées <code>ctrlx, ctrly</code> .
<code>rect</code>	<code>rect(x1, y1, x2, y2)</code>	Ajoute un contour rectangulaire au chemin courant.

# Images dans un canvas

L'affichage d'une image dans un canevas se fait avec la fonction drawImage() dont voici les trois variantes :

```
drawImage(image, x, y);
```

```
drawImage(image, x, y, largeur, hauteur);
```

```
drawImage(image, sx, sy, slargeur, shauteur, dx, dy, dlargeur, dhauteur);
```

La première se contente de placer l'image sur le canevas aux coordonnées spécifiées.

La deuxième la place sur le canevas puis la redimensionne.

La troisième la place sur le canevas, la redimensionne et définit la position et la taille du canevas destination.

# Images dans un canvas

Cet exemple utilise la première variante de la fonction drawImage() pour afficher une image PNG sur le canevas.

Code088.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Des images dans un canevas</title>
    <script>
      function init()
      {
        var elem = document.getElementById('canevas');
        var context = elem.getContext('2d');
        var img = new Image();
        img.onload = function()
        {
          context.drawImage(img,0,0);
        }
        img.src = 'fbas.png';
      }
    </script>
  </head>

  <body onload="init();">
    <canvas id="canevas" width="250" height="200" style="border: dotted;">
      Votre navigateur ne supporte pas la balise HTML5 &lt;canvas&gt;;
    </canvas>
  </body>
</html>
```

## Code089.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Des images dans un canevas</title>
    <script>
      function init()
      {
        var elem = document.getElementById('canevas');
        var context = elem.getContext('2d');
        context.drawImage(document.getElementById('cadre'), 0, 0)
        context.drawImage(document.getElementById('image'), 100, 300, 400, 400, 50, 40, 300, 370);
      }
    </script>
  </head>

  <body onload="init();">
    Résultat :<br>
    <canvas id="canevas" width="400" height="440">
      Votre navigateur ne supporte pas la balise HTML5 &lt;canvas&gt;;
    </canvas>
    <br>A partir de ces images :<br>
    
    
  </body>
</html>
```

Cet autre exemple utilise la troisième variante de la fonction drawImage(). La photo de l'éléphant fait  $683 \times 1\ 024$  pixels. Une portion de  $40 \times 400$  pixels décalée de 100 pixels horizontalement et de 300 pixels verticalement (les quatre premiers paramètres) est placée dans un espace de  $300 \times 370$  pixels, décalée de 50 pixels horizontalement et 40 pixels verticalement (les quatre derniers paramètres) dans le canevas.

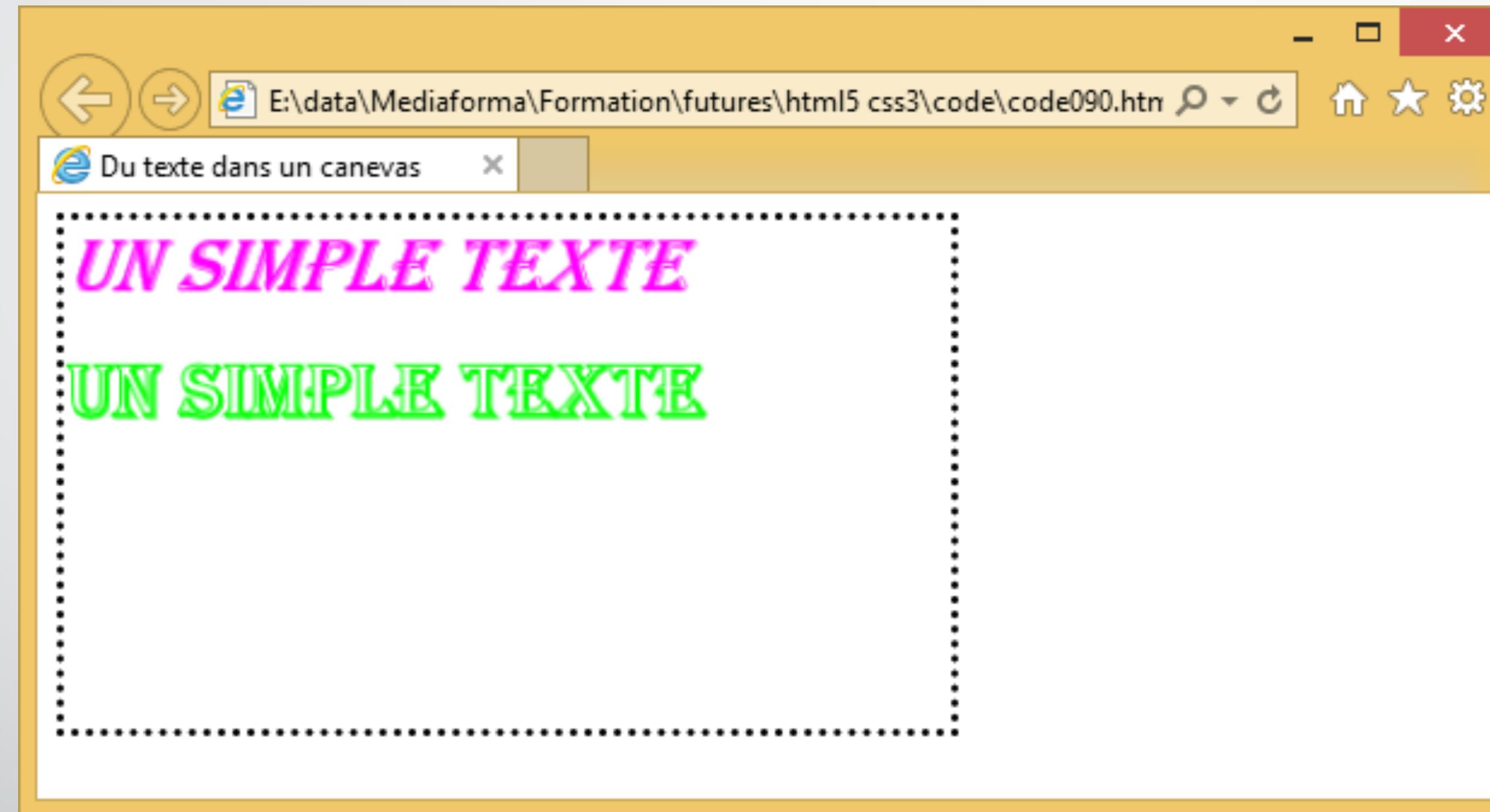
# Texte dans un canvas

Plusieurs fonctions JavaScript sont destinées à l'affichage de texte. Nous les avons résumées dans ce tableau.

Propriété de l'objet context	Signification
<code>font</code>	Police
<code>textAlign</code>	Alignment horizontal : start, end, left, right ou center (start par défaut)
<code>textBaseline</code>	Alignment vertical : top, hanging, middle, alphabetic, ideographic ou bottom (alphabetic par défaut)
<code>fillText</code>	Dessine le texte avec le remplissage défini dans la propriété <code>fillStyle</code>
<code>strokeText</code>	Dessine le contour du texte avec le remplissage défini dans la propriété <code>strokeStyle</code>

# Exercice

Définissez le code nécessaire pour obtenir le résultat suivant (le police utilisée est Algerian) :



# Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Du texte dans un canevas</title>
    <script>
      function init()
      {
        var elem = document.getElementById('canevas');
        var context = elem.getContext('2d');
        context.fillStyle      = '#f0f';
        context.font          = 'italic 30px Algerian';
        context.textBaseline = 'top';
        context.fillText ('Un simple texte', 0, 0);
        context.strokeStyle  = '#0f0';
        context.font          = 'bold 30px Algerian';
        context.strokeText ('Un simple texte', 0, 50);
      }
    </script>
  </head>

  <body onload="init();">
    <canvas id="canevas" width="350" height="200" style="border: dotted;">
      Votre navigateur ne supporte pas la balise HTML5 &lt;canvas&gt;;
    </canvas>
  </body>
</html>
```

# Graphiques vectoriels SVG

L'acronyme SVG vient de l'anglais *Scalable Vector Graphic*. Il s'agit d'un format de données basé sur la syntaxe XML et conçu pour décrire des graphiques vectoriels 2D.

L'étude du langage SVG n'est pas prévue dans cette formation. Par contre, nous allons voir comment utiliser un fichier SVG externe dans du code HTML5. Pour cela, nous utiliserons un élément object.

## Code091.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Utilisation d'un fichier SVG</title>
  </head>
  <body>
    <object type="image/svg+xml" data="departements.svg" width="800" height="600">
      <p>Votre navigateur ne supporte pas le SVG.</p>
    </object>
  </body>
</html>
```

Le fichier departements.svg a été téléchargé sur le site  
<http://www.orvinfait.fr/svg/carte/france.html>

# Graphiques vectoriels SVG

Voici un extrait du fichier departements.svg :

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<svg>
  ...
    <path id="dp11" fill="#ffffffff" d="M 274,438 L 273,442 L 270,440
L 266,440 L 266,439 L 264,439 L 260,441 L 259,438 L 256,441 L
257,443 L 254,444 L 253,447 L 251,448 L 253,451 L 252,452 L
262,457 L 263,464 L 263,467 L 264,472 L 259,472 L 257,474 L
264,479 L 267,477 L 272,482 L 271,483 L 272,483 L 280,479 L
278,476 L 278,473 L 296,473 L 296,470 L 300,468 L 305,472 L
308,473 L 307,468 L 308,461 L 305,461 L 303,458 L 305,456 L
308,459 L 311,457 L 313,455 L 313,453 L 311,453 L 310,450 L
307,450 L 305,446 L 303,446 L 301,445 L 301,442 L 300,443 L
300,445 L 298,445 L 298,448 L 294,450 L 292,446 L 290,447 L
288,446 L 287,443 L 288,441 L 288,439 L 287,439 L 281,439 L
275,438 L 274,438 z ">
  ...
    <text x="549" y="523" id="tp0">2a</text>
    <text x="557" y="483" id="tp20">2b</text>
  ...
</svg>
```

# Graphiques vectoriels SVG

Pour créer des images au format SVG, vous pouvez utiliser des logiciels :

- gratuits (**Inkskape** ou **LibreOffice Draw** par exemple) ;
- payants (**Illustrator** ou **PhotoShop** par exemple).

# HTML5 avancé

Cette dernière section regroupe plusieurs techniques avancées utilisables en HTML5. Depuis l'implémentation de WAI-ARIA et des microformats à l'utilisation de Media Queries pour adapter le code aux mobiles, vous découvrirez des techniques qui vous offriront de nouvelles perspectives quant aux multiples utilisations du langage HTML5.

# Barres de progression

Deux éléments HTML5 permettent de définir des barres de progression : progress et meter.

Voici la syntaxe de l'élément progress :

```
<progress value="valeur" max="valeur-maximale">
```

Où valeur-maximale est une donnée numérique qui correspond au maximum affichable dans la barre de progression et valeur est la valeur à afficher.

Dans la diapositive suivante, quatre barres de progression sont affichées. Dans la première, l'attribut value n'est pas défini. Dans la deuxième, il est égal à 3 sur 10. Dans la troisième, il est égal à 2345 sur 2345. Enfin, dans le quatrième il est initialisé *via* le DOM par la fonction JavaScript init() à la valeur 65 sur 100.



## Code092.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément progress</title>
    <style>
      label
      {
        display: inline-block;
        width: 200px;
      }
    </style>
    <script>
      function init()
      {
        var el = document.getElementById('progress-js');
        el.setAttribute("value", "65");
      }
    </script>
  </head>
  <body onload="init();">
    <ul>
      <li>
        <label>Indéterminé </label>
        <progress max="100"></progress>
      </li>
      <li>
        <label>Progress = 30%</label>
        <progress max="10" value="3"></progress>
      </li>
      <li>
        <label>Progress = 100%</label>
        <progress max="2345" value="2345"></progress>
      </li>
      <li>
        <label>Progress contrôlé par JavaScript</label>
        <progress max="100" id="progress-js"></progress>
      </li>
    </ul>
  </body>
</html>
```

# Barres de progression

L'élément meter a un comportement légèrement différent. Voici sa syntaxe :

```
<meter value="v1" min="v2" low="v3" high="v4" max="v5" optimum="v6">
```

Où v1 est la valeur à afficher, v2 la valeur minimale, v3 une valeur considérée comme faible, v4 une valeur considérée élevée, v5 la valeur maximale et v6 une valeur considérée comme optimale.

## Code093.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>L'élément meter</title>
    <style>
      label { display: inline-block; width: 200px; }
      meter { width: 200px; }
    </style>
    <script>
      function init() {
        var el = document.getElementById('meter-js');
        el.setAttribute("value","65");
      }
    </script>
  </head>
  <body onload="init();">
    <ul class="compact">
      <li>
        <label>Indéterminé</label>
        <meter></meter>
      </li>
      <li>
        <label>Meter = 0.5</label>
        <meter value="0.5"></meter>
      </li>
      <li>
        <label>Meter = 0.2 avec low = 0.2</label>
        <meter value="0.2" low="0.2"></meter>
      </li>
      <li>
        <label>Meter = maximum</label>
        <meter value="1"></meter>
      </li>
      <li>
        <label>Meter contrôlé par JavaScript</label>
        <meter min="0" max="100" id="meter-js"></meter>
      </li>
    </ul>
  </body>
</html>
```

# MathML et HTML5

Sachez tout d'abord que MathML est un langage de type XML qui contient de nombreux éléments fortement spécialisés et hiérarchisés. Tous les éléments utilisés dans MathML font partie d'une des trois catégories suivantes : présentation, contenu ou interface.

Pour insérer des éléments MathML dans un document HTML5, vous utiliserez l'élément math en déclarant l'espace de noms associé dans l'élément xmlns :

```
<math xmlns="http://www.w3.org/1998/Math/MathML">  
...  
</math>
```

# MathML et HTML5

Code094.htm

À titre d'exemple, voici le code MathML permettant de générer l'équation  $x^2 + 1 = 0$  :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>MathML dans HTML5</title>
  </head>
  <body>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <mi>a</mi>
        <msup>
          <mi>x</mi>
          <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mi>b</mi>
        <mi>x</mi>
        <mo>+</mo>
        <mi>c</mi>
        <mo>=</mo>
        <mn>0</mn>
      </mrow>
    </math>
  </body>
</html>
```

## Quelques explications :

- Les identifiants sont encadrés par les balises <mi> et </mi>.
- Les opérateurs sont encadrés par les balises <mo> et </mo>.
- Les nombres sont encadrés par les balises <mn> et </mn>.
- Les expressions disposées sur une même ligne sont encadrées par les balises <mrow> et </mrow>.

Pour l'instant, ce code n'est opérationnel que dans Firefox.

# WAI-ARIA

Le but de cette spécification est de faciliter l'accès aux pages et aux applications web pour les personnes malvoyantes. Pour rendre une page web compatible WAI-ARIA, le développeur insérera des attributs role (aussi appelés *landmarks* ou "points de repère") dans les éléments HTML traditionnels, comme indiqué dans le tableau suivant :

Éléments	Rôle	Signification
<u>body</u>	document ou application	Document HTML ou application RIA
<u>header</u>	banner	En-tête et/ou titre de la page
<u>nav</u>	navigation	Liens de navigation
<u>div</u>	main	Contenu principal de la page
<u>div</u> ou <u>form</u>	search	Moteur de recherche dans le document
<u>aside</u>	complementary	Contenu complémentaire à la page
<u>footer</u>	contentinfo	Notes de bas de page, pied de page, copyright

## Un exemple de code WAI-ARIA

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Un document HTML5 avec en-tête, pied de page, menu, zone annexe et zone de contenu</title>
        <link rel="stylesheet" href="style.css">
        <script src="Creation-elements-HTML5.js"></script>
    </head>
    <body role="document">
        <header role="banner">
            <h1>En-tête du document</h1>
            Texte de l'en-tête
        </header>
        <nav role="navigation">
            <h2>Menu</h2>
            <ul>
                <li><a href="page1.htm">Page 1</a></li>
                <li><a href="page2.htm">Page 2</a></li>
                <li><a href="page3.htm">Page 3</a></li>
            </ul>
            <br><br><br>
        </nav>
        <aside role="complementary">
            Texte affiché dans la partie droite de la page avec la balise &lt;aside&gt;<br><br>
        </aside>
        <article>
            <h2>Premier article</h2>
            <p>Texte du premier article</p>
        </article>
        <article>
            <h2>Second article</h2>
            <p>Texte du second article</p>
        </article>
        <footer role="contentinfo">
            <p>Copyright et e-mail du webmaster </p>
        </footer>
    </body>
</html>
```

Code095.htm

# WAI-ARIA

Si vous voulez aller plus loin dans WAI-ARIA, consultez la documentation W3C officielle, aux pages [www.w3.org/WAI/intro/aria](http://www.w3.org/WAI/intro/aria) et [www.w3.org/TR/wai-aria/#roles](http://www.w3.org/TR/wai-aria/#roles), ou encore l'excellent article de Gez Lemon sur <http://dev.opera.com/articles/view/introduction-to-wai-aria/>.

# Microformats

Les microformats sont utilisés pour formater des données afin d'automatiser leur traitement dans les applications qui les consomment (carnet d'adresses, coordonnées géographiques, etc.) et leur reconnaissance par les moteurs de recherche.

# Le format hCard

Vous utiliserez le format hCard pour représenter des personnes, sociétés ou organisations. À titre d'exemple, l'élément div suivant regroupe des informations relatives à un contact :

```
<div>

    <strong>Samuel Michaux</strong>

    Formateur chez Mediaforma

    12 bis, rue des Lauriers

    75010 Paris

</div>
```

# Le format hCard

Pour rendre cet élément plus "lisible" par les moteurs de recherche, nous allons lui associer plusieurs microformats :

```
<div class="vcard">
    
    <strong class="fn">Samuel Michaux</strong>
    <span class="title">Formateur</span> chez <span class="org">Mediaforma</span>
    <span class="adr">
        <span class="street-address">12 bis, rue des Lauriers</span>
        <span class="postal-code">75010</span>
        <span class="locality">Paris</span>
    </span>
</div>
```

Le nom de la classe associée à la div indique que les données qu'elle renferme décrivent une personne. Les classes utilisées dans les diverses composantes de la div sont caractéristiques : photo correspond à la photographie de la personne, fn à son nom, title à son métier, org au nom de la société pour laquelle il travaille. Il en va de même pour ce qui concerne les classes de l'adresse : street-address correspond à l'adresse, postal-code au code postal et locality à la ville.

Ces noms de classes n'ont pas été choisis au hasard : ils correspondent au format hCard et peuvent être complétés par de nombreuses autres informations, comme indiqué à la page <http://microformats.org/wiki/hcard-fr>.

# Le format hCalendar

Vous utiliserez le format hCalendar pour définir une représentation sémantique HTML d'informations concernant un événement. Le code ci-après définit l'événement "Office 365" qui a lieu dans la salle de conférence C12, sur le campus de Microsoft, le 10 décembre de 9 heures à 23 heures.

```
<div id="hcalendar-Conférence-Salle-C12" class="vevent">
  <a class="url" href="http://www.microsoft.com/fr/fr/default.aspx">
    <abbr class="dtstart" title="2010-12-10T09:00+01:0000">December 10, 2010 9h00</abbr> -
    <abbr class="dtend" title="2010-12-10T23:00+01:00">23:00</abbr> :
    <span class="summary">Conférence Salle C12</span> au
    <span class="location">Campus Microsoft</span></a>
    <div class="description">Office 365</div>
</div>
```

Ce code a été créé avec hCalendar Creator, disponible sur <http://microformats.org/code/hcalendar/creator>. Vous en saurez plus sur les microformats relatifs à hCalendar en consultant la page <http://microformats.org/wiki/hcalendar-fr>.

# Le format hReview

Lorsque les commentaires postés sur une page sont balisés à l'aide de microformats, Google peut les identifier et en rendre une partie visible dans les résultats des recherches. Par exemple, voici comment pourrait apparaître un résultat de recherche pour une page qui a défini des microformats relatifs aux commentaires des visiteurs :

## [Pizza Suprema - New York, NY, 10001 - Citysearch](#)

 ★★★★☆ 8 avis

Jun 15, 2010 ... What People Are Saying About **Pizza Suprema**. The Owner. **Pizza Suprema**. Owner. Located in Midtown, across from Penn Station and Madison Square ...  
[newyork.citysearch.com](http://newyork.citysearch.com) › Manhattan › Restaurants - Cached - Similar

# Le format hReview

Voici un exemple de commentaire laissé par un visiteur :

```
<div>
  <span>5 étoiles sur 5</span>
  <h4>Le Guide de survie HTML5/CSS3 est très efficace</h4>
  <span>Critique : <span>Zorbox</span> - 10 décembre 2012</span>
  <blockquote>
    <p>Le Guide de survie HTML5/CSS3, des éditions Pearson, est un excellent guide pour
       tous ceux qui veulent découvrir les possibilités de ces langages.
       Il couvre des sujets proches des utilisateurs et montre comment les mettre en œuvre
       en proposant du code directement utilisable. A conseiller à tous ceux qui veulent bien
       commencer avec HTML5/CSS3. </p>
  </blockquote>
  <p>Date de visite : <span>Décembre 2012</span></p>
</div>
```

# Le format hReview

Pour insérer des microformats dans ce commentaire, nous allons utiliser plusieurs classes dédiées :

```
<div class="hreview">
  <span><span class="rating">5</span> étoiles sur 5</span>
  <h4 class="summary">Le Guide de survie HTML5/CSS3 est très efficace</h4>
  <span class="reviewer vcard">Critique : <span class="fn">Zorbox</span> -
  <abbr class="dtreviewed" title="20101210T2300-0700">10 décembre 2010</abbr></span>
  <div class="description item vcard">
    <p>Le Guide de survie HTML5/CSS3, des <span class="fn org">éditions Pearson</span>
    est un excellent guide pour tous ceux qui veulent découvrir les possibilités de
    ces langages.
    Il couvre des sujets proches des utilisateurs et montre comment les mettre en
    oeuvre en proposant du code directement utilisable. A conseiller à tous ceux
    qui veulent bien commencer avec HTML5/CSS3.</p>
  </div>
  <p>Date de visite : <span>Décembre 2010</span></p>
</div>
```

# Le format hReview

Le nom de la classe associée à la div indique que les données qu'elle renferme décrivent un commentaire d'utilisateur. Les classes utilisées dans les diverses composantes de la div sont caractéristiques :

Classe	Définition
<u>rating</u>	Notation
<u>reviewer vcard</u>	Informations relatives à la personne qui a laissé le commentaire
<u>dtreviewed</u>	Date du commentaire
<u>description item vcard</u>	Commentaire

Ces noms de classes n'ont pas été choisis au hasard : ils correspondent au format hReview et peuvent être complétés par de nombreuses autres informations, comme indiqué sur la page <http://microformats.org/wiki/hreview-fr>.

# Le format hResume

Le format hResume est utilisé pour publier des CV. Pour créer le code HTML contenant les bons microformats, le plus simple consiste à utiliser la page hResume Creator, à l'adresse <http://hresume.weblogswork.com/hresumecreator/>. Remplissez les champs nécessaires et cliquez sur Create hResume pour obtenir le code HTML.

Vous en saurez plus sur ce format en consultant la page <http://microformats.org/wiki/hresume-fr>.

D'une façon générale, pour être au courant des avancées dans les microformats, n'hésitez pas à consulter la page [http://microformats.org/wiki/Main\\_Page-fr](http://microformats.org/wiki/Main_Page-fr).

# Media Queries

Le Web est de plus en plus consulté sur des périphériques de petite taille (téléphones, tablettes, netbooks). C'est pourquoi il est très important d'adapter l'affichage de vos pages web à cela. Les Media Queries du langage CSS3 proposent une solution élégante à ce problème. Ils permettent en effet de définir le style d'une page web en fonction de plusieurs facteurs liés à la surface d'affichage : la largeur, la hauteur, l'orientation, la résolution, etc.

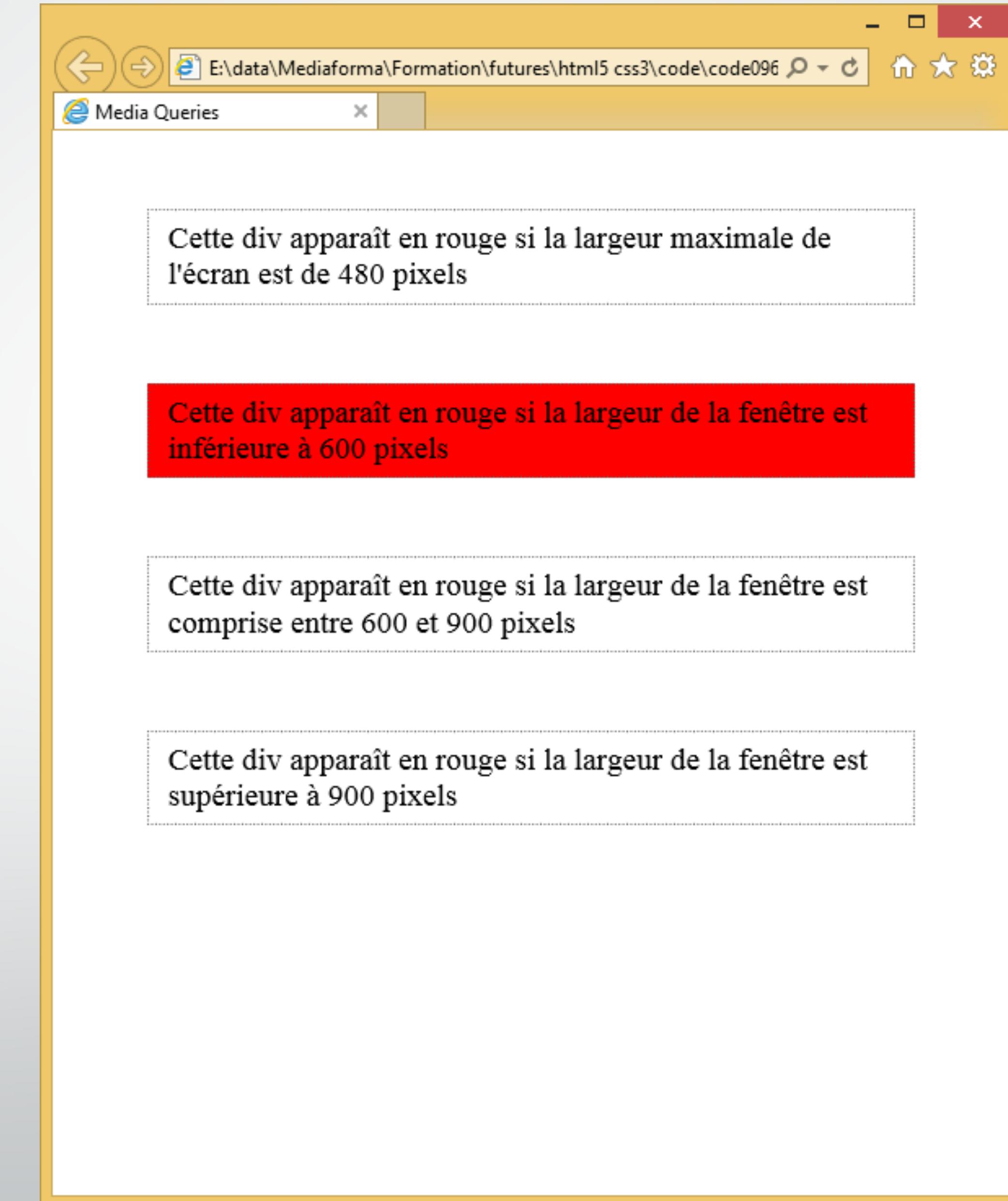
# Media Queries

Les Media Queries reposent sur la propriété CSS3 @media. Voici quelques-unes des syntaxes utilisables :

Syntaxe	Signification
<code>@media (max-width: largeur) { ... }</code>	Largeur de la fenêtre inférieure à la largeur spécifiée
<code>@media (max-device-width: largeur) { ... }</code>	Largeur du périphérique inférieure à la largeur spécifiée
<code>@media (min-width: largeur1) and (max-width: largeur2) { ... }</code>	Largeur de la fenêtre comprise entre les deux largeurs spécifiées
<code>@media (max-device-width: largeur) and (orientation: landscape) { ... }</code>	Largeur du périphérique inférieure à la largeur spécifiée, et écran tenu horizontalement

# Exercice

Ecrivez le code HTML5/CSS3 nécessaire pour modifier la couleur d'arrière-plan de quatre div en fonction de la largeur de la fenêtre. Pour cela, vous utiliserez des media-queries.



## Code096.htm

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Queries</title>
    <style>
      div
      {
        border: dotted 1px #666;
        padding: 5px 10px;
        margin: 40px;
      }
      @media (max-device-width: 480px)
      {
        .iphone
        {
          background: red;
        }
      }
      @media (max-width: 600px)
      {
        .inf600
        {
          background: red;
        }
      }
      @media (min-width: 900px)
      {
        .sup900
        {
          background: red;
        }
      }
      @media (min-width: 600px) and (max-width: 900px)
      {
        .de600a900
        {
          background: red;
        }
      }
    </style>
  </head>
  <body>
    <div class="iphone">Cette div apparaît en rouge si la largeur maximale de l'écran est de 480 pixels</div>
    <div class="inf600">Cette div apparaît en rouge si la largeur de la fenêtre est inférieure à 600 pixels</div>
    <div class="de600a900">Cette div apparaît en rouge si la largeur de la fenêtre est comprise entre 600 et 900 pixels</div>
    <div class="sup900">Cette div apparaît en rouge si la largeur de la fenêtre est supérieure à 900 pixels</div>
  </body>
</html>
```

# Solution

Il est possible d'utiliser une feuille de style externe ou une autre en fonction de la résolution de l'écran/du device en utilisant un media query.

Ici par exemple, la feuille de style **feuille1.css** sera utilisée si la largeur de l'écran est inférieure ou égale à 800 pixels. La feuille de style **feuille2.css** sera utilisée si la largeur de l'écran est supérieure à 800 pixels :

```
<link rel="stylesheet" media="screen and (max-width:800px)" href="css/feuille1.css">
<link rel="stylesheet" media="screen and (min-width:801px)" href="css/feuille2.css">
```

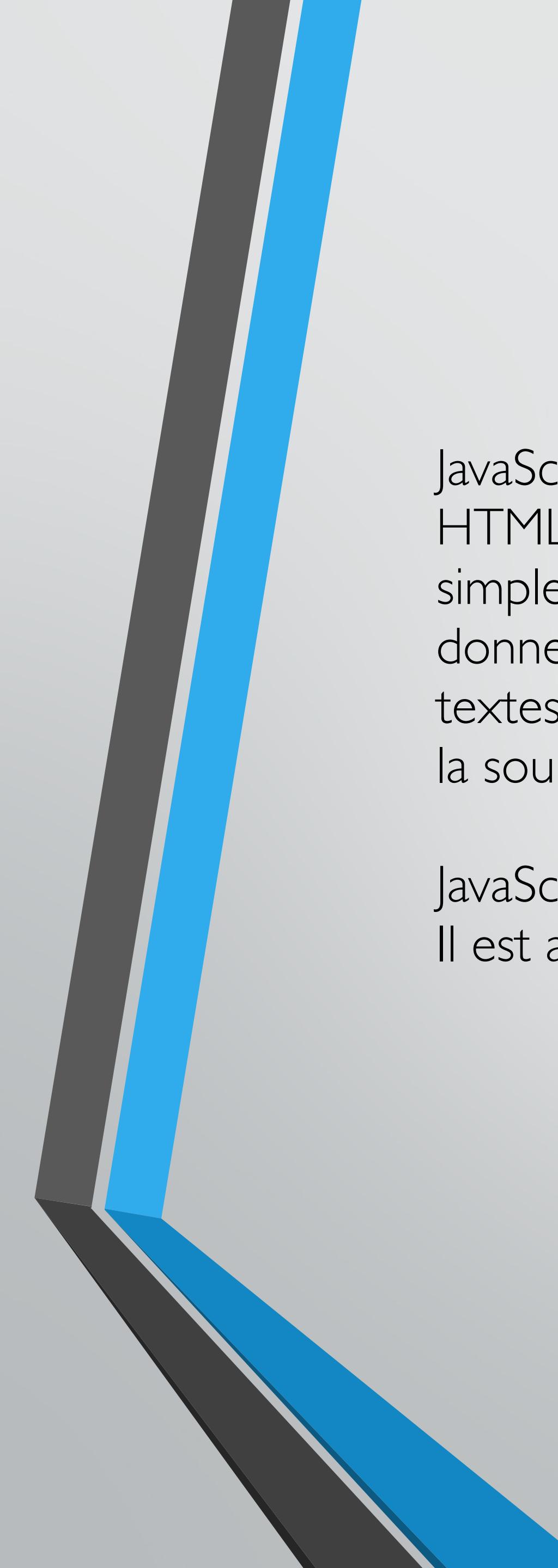
Pour en savoir plus sur l'utilisation de l'attribut media dans une balise <link>, consultez cette page : [https://www.w3schools.com/tags/att\\_link\\_media.asp](https://www.w3schools.com/tags/att_link_media.asp)

# JavaScript

# JavaScript

Cette section va vous montrer comment utiliser le langage :

- JavaScript et HTML
- Objets, propriétés, fonctions et méthodes
- Définition d'un nouveau type d'objet
- Utilisation d'un objet par défaut
- Tour d'horizon du langage JavaScript



JavaScript est un langage objet interprété dont les instructions sont stockées (en clair) dans des pages HTML. Petit frère de Java, JavaScript en reprend les grandes lignes. Son accès est cependant plus simple pour les non-programmeurs. Après cette formation, les concepteurs de pages Web pourront donner vie à leurs œuvres en y intégrant du code JavaScript. Ainsi, ils pourront par exemple animer textes et graphiques et réagir aux actions de l'utilisateur telles que clic sur une zone, déplacement de la souris, etc..

JavaScript a été créé en 1995 par Brendan Eich pour Netscape sous le nom ECMAScript. Il est apparu en 1996 dans le navigateur Netscape.

Pour saisir du code JavaScript dans vos pages HTML, vous pouvez utiliser un simple éditeur de texte, comme le bloc-notes de Windows, ou mieux, un éditeur dédié à la saisie de code, comme Notepad++ par exemple.

Ce programme est librement téléchargeable sur <http://notepad-plus-plus.org/fr/>

L'intégration de code JavaScript dans une page HTML peut se faire de deux façons :

1. A l'aide d'une balise <script> </script>.
2. En mettant en place un gestionnaire d'événements.

Dans le premier cas, le code est exécuté pendant le chargement de la page. Dans le deuxième cas, il est en mesure de réagir à des événements utilisateur : clics, focus ou perte de focus sur un contrôle, appui sur le bouton **Submit** ou sur le bouton **Quit** dans un formulaire, etc.

## **Définition de code JavaScript avec une balise <script>**

La syntaxe du balise <script></script> est la suivante :

```
<script language = "JavaScript">  
    Lignes de code  
</script>
```

la balise <script> doit être intégré à la suite de la balise <head> :

```
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="utf-8">  
    <script>  
        //Lignes de code JavaScript  
    </script>  
</head>  
<body>  
    <!-- Balises HTML -->  
<script></script>  
</body>  
</html>
```

Cet exemple simpliste vous montre où et comment implémenter vos premières instructions JavaScript.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <p>Alors que celui-là provient de la section BODY du document HTML.</p>

    <script>
        document.write('Ce texte est écrit par une instruction JavaScript.');
    </script>
</body>
</html>
```

Dans la diapositive précédente, nous avons utilisé la fonction JavaScript **document.write()** pour afficher du texte dans le DOM.

Vous auriez également pu utiliser :

- La fonction JavaScript alert() pour afficher une boîte de message :  
`alert('ceci est un message JavaScript');`
- Ou encore afficher du texte dans la console avec l'instruction suivante :  
`console.log('Texte affiché dans la console');`

Pour visualiser le texte dans la console, appuyez sur la touche **F12** du clavier et basculez sur l'onglet ou dans le panneau (selon le navigateur) **Console**.

Le code JavaScript peut être placé entre les balises <head> et </head> ou entre les balises <body> et </body>.

Si vous le souhaitez, vous pouvez stocker tout votre code JavaScript dans un fichier externe d'extension .js. Vous ferez référence à ce fichier avec les instructions HTML suivantes entre les balises <head> et </head> ou <body> et </body> :

```
<script src="monJavascript.js"></script>
```

# ***Une fonction dans une balise <script>***

Comme la plupart des langages, JavaScript permet de définir des fonctions. Ces fonctions doivent impérativement être placées entre les balises <head> et </head>. En effet, le code compris entre ces deux balises étant chargé en premier, cela garantit que les fonctions JavaScript auront été mémorisées avant que l'utilisateur ne tente une quelconque action qui provoquerait leur appel.

La syntaxe permettant de déclarer une fonction est proche de celle utilisée dans le langage C :

```
function nom([param1, ...paramN]) {  
    // Une ou plusieurs instructions  
}
```

Où :

- nom est le nom de la fonction ;
- param1 à paramN sont les éventuels paramètres passés à la fonction ;
- Instructions représente une ou plusieurs instructions exécutées par la fonction.

Lorsqu'une fonction a été définie, vous utiliserez un marqueur <script> </script> pour l'appeler :

```
<script>  
    nom( [param1, ...paramN] ) ;  
</script>
```

- où nom est le nom de la fonction à appeler ;
- et param1 à paramN sont les éventuels paramètres passés à la fonction.

Exercice :

Définissez la fonction **carre()** qui calcule et affiche le carré du nombre qui lui est passé en argument.

Solution :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  Test de la fonction <b>carre</b>:
  <script>
    function carre(x) {
      document.write('Le carré de ' + x + ' est ' + x*x + '.<br>');
    }
    carre(7);
    carre(12);
  </script>
</body>
</html>
```

# Portée des variables

Une variable déclarée au début du script, avant toutes fonctions, sera globale. Elle peut être utilisée n'importe où dans le script.

Une variable déclarée dans une fonction aura une portée limitée à cette seule fonction : elle n'est pas utilisable en dehors de la fonction. On parle alors de « **variable locale** ».

# ***Mise en place d'un gestionnaire d'événements***

Dans le navigateur, certaines actions effectuées par l'utilisateur donnent lieu à des événements. Si vous mettez en place un gestionnaire d'événements, il sera automatiquement exécuté lorsque l'événement correspondant se présentera.

Pour définir un gestionnaire d'événements, vous utiliserez la syntaxe suivante :

```
<balise NomEvent = "code JavaScript">
```

- où balise est le nom d'une balise HTML ;
- NomEvent est le nom de l'événement déclenchant ;
- code JavaScript est une instruction JavaScript exécutée lorsque l'événement NomEvent se produit.

A titre d'exemple, l'instruction ci-après définit un bouton de commande, lui donne le nom Résultat et met en place un gestionnaire d'événements qui est sollicité lorsque l'utilisateur clique sur ce bouton.

```
<input  
    type = "button"  
    value = "Résultat"  
    onclick = "suivant(this.form);">
```

Le code qui suit le déclencheur onclick correspond au nom d'une fonction. Vous pouvez spécifier au choix une instruction JavaScript ou le nom d'une fonction. Cette deuxième possibilité est bien plus intéressante, car elle donne le contrôle à une fonction qui contient autant d'instructions que vous le souhaitez.

Exercice :

Définissez un formulaire contenant une zone de texte et un bouton. Lorsque l'utilisateur clique sur le bouton, exécuter la fonction **carre()** qui affiche le carré du nombre tapé dans la zone de texte.

Indice :

Si une balise **<input>** de type **text** a pour attribut **name xyz**, JavaScript peut accéder en lecture et en écriture au contenu de cette balise avec l'instruction suivante :

```
form.xyz.value
```

Solution :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script language = "JavaScript">
function carre(form) {
    form.resultat.value = form.resultat.value * form.resultat.value;
}
</script>
</head>
<body>
<form>
    Test de la fonction <B>carre</B>:</P>
    <input type="text" name="resultat" size="25">
    <input type="button" value="Carré" onclick = "carre(this.form);">
</form>
</body>
</html>
```

# ***Objets, propriétés, fonctions et méthodes***

Lorsque vous chargez un document dans le navigateur, quatre objets prédéfinis lui sont instantanément associés :

1. L'objet window représente la fenêtre affichée dans le navigateur.
2. L'objet location contient l'URL du document courant.
3. L'objet document donne accès aux propriétés du document : titre, couleur ou image d'arrière-plan, couleur par défaut du texte, etc.
4. L'objet history donne accès aux URL des pages précédemment visualisées.

Tous ces objets sont accessibles par une syntaxe hiérarchique "à point. Par exemple :

objet1.objet2.objet3

Ici, objet3 fait partie d'objet2, qui fait lui-même partie d'objet1.

Cette syntaxe à point se termine souvent par un élément qui n'est pas un objet, mais une propriété.

## Exercice :

Définissez un formulaire composé de deux zones de texte, deux boutons radio et trois boutons de commande : Envoyer, Annuler et Majuscules. Un appui sur le bouton Majuscules transforme les caractères de l'adresse e-mail en majuscules.

The screenshot shows a web browser window titled "Un premier formulaire". Inside the window, there is a form with the following elements:

- A text input field labeled "Entrez votre nom" containing the value "Michel".
- A text input field labeled "Entrez votre adresse e-mail" containing the value "MM54@FREE.FR". A red arrow points from this field towards the "Majuscules" button.
- A question "Connaissez-vous Mediaforma ?" followed by two radio buttons: "OUI" (selected) and "NON".
- Three buttons at the bottom: "Envoyer", "Annuler", and "Majuscules". The "Majuscules" button has a mouse cursor pointing at it, and a dashed red arrow points from the "Entrez votre adresse e-mail" field towards this button.

### Remarque :

Pour accéder à la valeur contenue dans une zone de texte en JavaScript, vous utiliserez la syntaxe suivante :

```
document.f.n.value
```

Où f est le nom (name) du formulaire et n le nom (name) de la zone de texte

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <script language = "JavaScript">
        function Majusc(form) {
            document.Quest.Adresse.value = document.Quest.Adresse.value.toUpperCase();

            // Une autre syntaxe possible
            // form.Adresse.value = form.Adresse.value.toUpperCase();
        }
    </script>
    <title>Un premier formulaire</title>
</head>
<body>
    <form name="Quest" action="mailto:bertrand.oscar@Oracle.fr" method="post">
        Entrez votre nom <input type="text" name="Nom"><br>
        Entrez votre adresse e-mail <input type="text" name="Adresse"><br>
        Connaissez-vous Mediaforma ? OUI <input type="radio" name="Client" value="OUI" checked>
        NON <input type="radio" name="Client" value="non"><br>
        <input type="submit" value="Envoyer">
        <input type="reset" value="Annuler">
        <input type="button" value = "Majuscules" onclick="Majusc(this.form)">
    <script>
    </form>
</body>
</html>
```

# ***Utilisation d'un objet par défaut***

JavaScript implémente plusieurs types d'objets. Lorsqu'un programme utilise fréquemment des propriétés et/ou des méthodes relatives à un même type d'objet, il est possible de simplifier la syntaxe en utilisant l'instruction with. Cette dernière définit un objet par défaut à l'intérieur de ses délimiteurs. Il n'est alors plus nécessaire de le spécifier comme préfixe des propriétés et méthodes.

Exemple :

Les instruction ci-après utilisent plusieurs méthodes relatives à l'objet Math :

```
document.write("Le sinus de " + x + " est " + Math.sin(x) + ".<br>");  
document.write("Le cosinus de " + x + " est " + Math.cos(x) + ".<br>");  
document.write("La tangente de " + x + " est " + Math.tan(x) + ".<br>");  
document.write("L'arc sinus de " + x + " est " + Math.asin(x) + ".<br>");  
document.write("L'arc cosinus de " + x + " est " + Math.acos(x) + ".<br>");  
document.write("L'arc tangente de " + x + " est " + Math.atan(x) + ".<br>");
```

Essayez ce code

Pour soulager l'écriture, il est possible d'utiliser une instruction `with` :

```
with (Math) {  
    document.write("Le sinus de " + x + " est " + sin(x) + ".<br>");  
    document.write("Le cosinus de " + x + " est " + cos(x) + ".<br>");  
    document.write("La tangente de " + x + " est " + tan(x) + ".<br>");  
    document.write("L'arc sinus de " + x + " est " + asin(x) + ".<br>");  
    document.write("L'arc cosinus de " + x + " est " + acos(x) + ".<br>");  
    document.write("L'arc tangente de " + x + " est " + atan(x) + ".<br>");  
}
```

Essayez ce code

Comme vous le voyez, il n'est plus nécessaire de préciser l'objet `Math` devant les méthodes, puisqu'il est utilisé par défaut sur toute l'étendue de l'instruction `with`.

Les fonctions mathématiques sin(), cos(), tan(), etc. demandent des arguments en radians.  
Si vous voulez travailler avec des degrés, vous devez effectuer une conversion degrés -> radians.

Exercice :

Trouvez comment effectuer cette conversion et affichez le sinus de 90°.

Solution

```
<script>
    function radians(degres) {
        return degres * Math.PI / 180;
    };

    function degres(radians) {
        return radians * 180 / Math.PI;
    };

    var x=radians(90);
    document.write('Le sinus de 90° est ' + Math.sin(x) + '.<br>');
</script>
```

# ***Tour d'horizon du langage JavaScript***

Dans les diapositives qui suivent, vous allez faire connaissance avec les instructions du langage JavaScript et avec les divers objets qui peuvent être accédés par son intermédiaire.

# **Variables et types de données**

JavaScript est en mesure de travailler avec des nombres réels ou entiers.

Les entiers peuvent être exprimés en décimal (base 10), en octal (base 8) ou en hexadécimal (base 16).

Un nombre octal commence toujours par un zéro, et un nombre hexadécimal par 0x ou 0X. Les éventuelles lettres d'un nombre hexadécimal peuvent indifféremment être exprimées en minuscules ou en majuscules.

Les nombres réels sont composés d'une partie entière suivie d'un point, d'une partie décimale, de la lettre e (ou E) et d'un exposant, éventuellement précédé d'un signe + ou -. Voici quelques exemples de nombres réels :

1 . 4142135

-32E-7

.5e12

JavaScript peut également travailler avec des chaînes de caractères délimitées par des guillemets ou des apostrophes.

Les affectations de chaînes ci-après sont correctes :

```
var x = "essai";  
x = 'essai';  
x = "essai\concluant";
```

Si nécessaire, vous pouvez inclure un ou plusieurs des caractères de contrôle suivants dans une valeur chaîne :

Code de contrôle	Effet
\b	Backspace
\f	Form feed
\n	Line feed
\r	Carriage return
\t	Tabulation

Il est également possible d'utiliser toutes les balises HTML qui affectent le style des caractères. Par exemple, l'affectation suivante est correcte :

```
var texte = "essai <font size='5' color='#FF0000'>concluant</font>";  
document.write(texte);
```

JavaScript peut également manipuler :

- des booléens qui prennent la valeur **true** ou **false**.
- des tableaux monodimensionnels, accessibles par leur indice.

Par exemple, **memo[0]** désigne le premier élément du tableau et **memo[12]** le treizième élément de ce même tableau.

Pour définir une variable, il n'est pas nécessaire de préciser son type. Utilisez simplement le mot réservé `var`, indiquez le nom de la variable et affectez-lui une valeur.

Exemples :

```
var petit = 10;  
var grand = 100;  
var titre = 'Types de données';  
var choix = true;
```

#### Attention

Tout comme le C++, le langage JavaScript tient compte des majuscules et des minuscules. Ainsi, par exemple, les variables `choix` et `Choix` ne seront pas, *a priori*, égales. De même, n'essayez pas d'affecter la valeur `True` à une variable booléenne. Une erreur serait générée à l'exécution. En effet, JavaScript connaît la valeur littérale `true` mais pas la valeur `True`.

Lorsqu'une valeur typée a été affectée à une variable, il n'est pas interdit de lui affecter par la suite un autre type de valeur. Par exemple, le code ci-après est tout à fait correct :

```
var ma_variable = 'valeur texte';
...
ma_variable = 123.456e12;
...
ma_variable = true;
```

Le signe + peut être utilisé pour concaténer deux valeurs de type différent. Par exemple, l'instruction suivante est tout à fait licite :

```
ma_variable = 500 + 'kilomètres';
```

# ***Conversions de types***

JavaScript est doté de trois fonctions de conversion chaîne/numérique :

- eval() : évaluation et conversion numérique d'une chaîne.
- parseInt() : conversion d'une chaîne en utilisant une base de numération donnée.
- parseFloat() : conversion d'une chaîne en un nombre réel.

## Fonction eval()

```
Résultat = eval(Expression);
```

- Expression est une chaîne de caractères contenant un nombre ou une formule dont le résultat est un nombre ;
- et Résultat est la valeur numérique retornée par la fonction.

Exemples :

```
var a = 5;  
eval("a*2"); // retourne la valeur numérique 10  
eval("123.45"); // retourne la valeur numérique 123.45
```

## Fonction parseInt()

Résultat = parseInt(Chaîne [,Base] )

- Chaîne est la chaîne à convertir ;
- Base est la base de numération de la chaîne ;
- Résultat est le résultat numérique de la conversion.

Exemples :

parseInt('FF',16); retourne la valeur numérique 255

parseInt('12',8); retourne la valeur numérique 10

parseInt('essai',10); retourne la valeur NaN

## Fonction `parseFloat()`

Résultat = parseFloat(Chaîne)

- Chaîne est la chaîne à convertir ;
- Résultat est le résultat réel de la conversion.

Exemples :

`parseFloat('427.35E-1');` retourne la valeur numérique 42.735

`parseFloat('X12');` retourne la valeur NaN

Exercice :

Créez un formulaire contenant deux zones de texte et un bouton.

L'utilisateur pourra entrer un calcul quelconque dans la première zone de texte. Au clic sur le bouton, le résultat du calcul sera affiché dans la deuxième zone de texte.

<input type="text" value="5+7*10"/>	<input type="button" value="Calculer"/>	<input type="text" value="75"/>
-------------------------------------	---	---------------------------------

## Solution

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Calculs dans un formulaire</title>
</head>
<body>
    <form name="f">
        <input type="text" name="saisie">
        <input type="button" value="Calculer" onclick="calcul();">
        <input type="text" name="resultat">
    </form>
    <script>
        function calcul() {
            document.f.resultat.value = eval(document.f.saisie.value);
        }
    </script>
</body>
</html>
```

# ***Opérateurs et expressions***

JavaScript utilise la plupart des opérateurs des autres langages de programmation. Pour faciliter leur énumération, nous les répartirons en six grands groupes. Ainsi, nous distinguerons les opérateurs :

- D'affectation.
- Arithmétiques.
- Relationnels et logiques.
- Dédiés aux chaînes de caractères.
- Dédiés aux opérations binaires (qui agissent au niveau des bits).

# Opérateurs d'affectation

Outre l'opérateur d'affectation "`=`", JavaScript dispose de six autres opérateurs qui allègent l'écriture :

Opérateur	Fonction	Exemple	Équivalent à
<code>+ =</code>	Addition de l'opérateur de droite à l'opérateur de gauche et affectation à l'opérateur de gauche	<code>a += b</code>	<code>a = a + b</code>
<code>- =</code>	Soustraction de l'opérateur de droite à l'opérateur de gauche et affectation à l'opérateur de gauche	<code>a -= b</code>	<code>a = a - b</code>
<code>* =</code>	Multiplication de l'opérateur de droite par l'opérateur de gauche et affectation à l'opérateur de gauche	<code>a *= b</code>	<code>a = a * b</code>
<code>/ =</code>	Division de l'opérateur de gauche par l'opérateur de droite et affectation à l'opérateur de gauche	<code>a /= b</code>	<code>a = a / b</code>
<code>% =</code>	Reste de la division entière de l'opérateur de gauche par l'opérateur de droite et affectation à l'opérateur de gauche	<code>a %= b</code>	<code>a = a % b</code>
<code>? :</code>	Affectation conditionnelle (condition ternaire)	<code>a = (c &lt; d) ? e : f</code>	<code>if (c &lt; d) a = e else a = f</code>

Exercice :

Définissez un formulaire contenant deux zones de saisie, un bouton et un textarea. Lorsque l'utilisateur clique sur le bouton, appliquez les opérateurs `+=`, `*=`, `/=` et `%=` aux valeurs numériques et affichez les résultats dans le textarea.

The screenshot shows a web browser window with the following content:

**Test des opérateurs élémentaires**

**Entrez vos données**

Premier nombre réel R1 =

Deuxième nombre réel R2 =

**Résultats**

Après l'opération `r1 += r2`, r1 a pour valeur 36.620000000000004  
Après l'opération `r1 *= r2`, r1 a pour valeur 301.5  
Après l'opération `r1 /= r2`, r1 a pour valeur 0.5182421227197347  
Après l'opération `r1 %= r2`, r1 a pour valeur 12.5

A red dashed arrow points from the "Calcul" button to the results area.

Voici le code HTML à utiliser :

```
<form name="Demo">
  <table>
    <tr align="center">
      <td colspan="2">
        <h1>Test des opérateurs d'affectation</h1>
        <hr>
      </td>
    </tr>
    <tr align = "center">
      <td>
        <b>Entrez vos données </b>
        <p>Premier nombre réel R1 = <input type ="text" name="Reel1" size="10"></p>
        <p>Deuxième nombre réel R2 = <input type="text" name="Reel2" size="10"></p>
      </td>
      <td>
        <input type="button" value="Calcul" onclick="Calcul(this.form)">
      </td>
    </tr>
    <tr align = "center">
      <td colspan="2">
        <b>Résultats</b>
        <textarea name = Result rows = 4 cols = 65></textarea>
      </td>
    </tr>
  </table>
</form>
```

Ecrivez le code JavaScript nécessaire pour arriver au résultat demandé.

## Solution :

```
<script language="JavaScript">
    function Calcul(form) {
        // Récupération des données entrées
        var r1 = parseFloat(form.Reell1.value);
        var r2 = parseFloat(form.Reel2.value);

        // Calculs
        var r = "Après l'opération r1 + = r2, r1 a pour valeur "+eval(r1+r2);
        var ChRes = r+"\r\n";
        r = "Après l'opération r1 - = r2, r1 a pour valeur "+eval(r1-r2);
        r = r1+"-"+r2+" = "+ eval("r1-r2");
        var r = "Après l'opération r1 * = r2, r1 a pour valeur "+eval(r1*r2);
        ChRes = ChRes+r+"\r\n";
        var r = "Après l'opération r1 / = r2, r1 a pour valeur "+eval(r1/r2);
        ChRes = ChRes+r+"\r\n";
        var r = "Après l'opération r1 % = r2, r1 a pour valeur "+eval(r1%r2);
        ChRes = ChRes+r+"\r\n";

        // Affichage des résultats
        form.Result.value = ChRes;
    }
</script>
```

# Opérateurs arithmétiques

Ces opérateurs effectuent des opérations arithmétiques sur leurs opérandes.

Opérateur	Fonction	Exemple
=	affectation	$a = 65$
+	addition	$c = a + b$
-	soustraction	$d = -a$
*	multiplication	$c = c * a$
/	division	$c = a / b$
%	modulo (reste de la division entière)	$c = a \% b$

Les opérateurs sont exécutés selon une hiérarchie bien définie ; du plus prioritaire vers le moins prioritaire :

- - de changement de signe
- /
- \*
- 
- +
- =



Il est parfois nécessaire d'utiliser des parenthèses pour forcer l'ordre d'exécution ou pour clarifier une expression de calcul.

A titre d'exemple, l'expression :

$$z = a + b / c + 4;$$

N'est pas du tout équivalente à l'expression :

$$z = (a + b) / c + 4;$$

## Exercice :

Définissez le code JavaScript nécessaire pour obtenir le résultat suivant lorsque l'utilisateur appuie sur le bouton Calcul :

The screenshot shows a web browser window with the title "Test des opérateurs arithmétiques". The browser's address bar displays the URL "E:\data\Mediaforma\Formation\Futures\JavaScript\cd\JavaScript\Arithm.htm".

The page contains two sections:

- Entrez vos données**: A form with four input fields:
  - Entier 1 :
  - Entier 2 :
  - Réel 1 :
  - Réel 2 :
- Résultats**: A scrollable list box displaying the results of various arithmetic operations:
  - 13+42=55
  - 13-42=-29
  - 13\*42=546
  - 13/42=0.30952380952380953
  - 13%42=13
  - 1.56+4.7=6.26
  - 1.56-4.7=-3.14
  - 1.56\*4.7=7.332000000000001
  - 1.56/4.7=0.331914893617021
  - 25

Voici le code HTML à utiliser :

```
<form name="demo">
  <table>
    <tr align="center">
      <td colspan="3">
        <h1>Test des opérateurs arithmétiques</h1>
        <hr>
      </td>
    </tr>
    <tr align ="center">
      <td>
        <b>entrez vos données</b><br>
        <p>entier 1 : <input type="text" name="Entier1" size="10"></p>
        <p>entier 2 : <input type = " text " name="Entier2" size="10"></p>
        <p>réel 1 : <input type="text" name="Reell1" size="10"></p>
        <p>réel 2 : <input type="text" name="Reel2" size="10"></p>
      </td>
      <td>
        <input type="button" value="calcul" onclick="calcul(this.form);">
      </td>
      <td>
        <p><b>résultats</b></p>
        <textarea name="Result" rows="10" cols="26"></textarea>
      </td>
    </tr>
  </table>
</form>
```

Solution :

```
<script language = "JavaScript">
    function calcul(form) {
        var e1 = parseInt(form.Entier1.value);
        var e2 = parseInt(form.Entier2.value);
        var r1 = parseFloat(form.Reel1.value);
        var r2 = parseFloat(form.Reel2.value);

        // Calculs sur les nombres entiers
        var r = e1+"+"+e2+" = "+ eval("e1+e2");
        var ChRes = r+"\r\n";
        r = e1+"-"+e2+" = "+ eval("e1-e2");
        ChRes = ChRes+r+"\r\n";
        r = e1+"*"+e2+" = "+ eval("e1*e2");
        ChRes = ChRes+r+"\r\n";
        r = e1+"/"+e2+" = "+ eval("e1/e2");
        ChRes = ChRes+r+"\r\n";
        r = e1+"%" +e2+" = "+ eval("e1%e2");
        ChRes = ChRes+r+"\r\n";

        // Calculs sur les nombres réels
        r = r1+"+"+r2+" = "+ eval("r1+r2");
        var ChRes = ChRes+r+"\r\n";
        r = r1+"-"+r2+" = "+ eval("r1-r2");
        ChRes = ChRes+r+"\r\n";
        r = r1+"*"+r2+" = "+ eval("r1*r2");
        ChRes = ChRes+r+"\r\n";
        r = r1+"/"+r2+" = "+ eval("r1/r2");
        ChRes = ChRes+r+"\r\n";
        r = r1+"%" +r2+" = "+ eval("r1%r2");
        ChRes = ChRes+r+"\r\n";

        // Affichage des résultats
        form.Result.value = ChRes;
    }
</script>
```

## Opérateurs relationnels et logiques

Ces opérateurs effectuent des comparaisons sur leurs opérandes qui peuvent être numériques, chaînes ou logiques. Le résultat renvoyé est 0 si la condition n'est pas vérifiée. Il est différent de 0 dans le cas contraire.

Opérateur	Fonction	Exemple
!	NON logique	$!(a == b)$
<	Inférieur à	$a < b$
>	Supérieur à	$a > b$
$\leq$	Inférieur ou égal à	$a \leq b$
$\geq$	Supérieur ou égal à	$a \geq b$
$\neq$	Different de	$a \neq b$
$==$	Egal à	$a == b$
$\&\&$	ET logique	$(a == b) \&\& (c > d)$
$\ $	OU logique	$(a == b) \  (c > d)$

## Attention

Une des erreurs les plus fréquentes en JavaScript consiste à utiliser l'opérateur `≡` à la place de l'opérateur `==` dans une comparaison logique.

Par exemple, l'instruction :

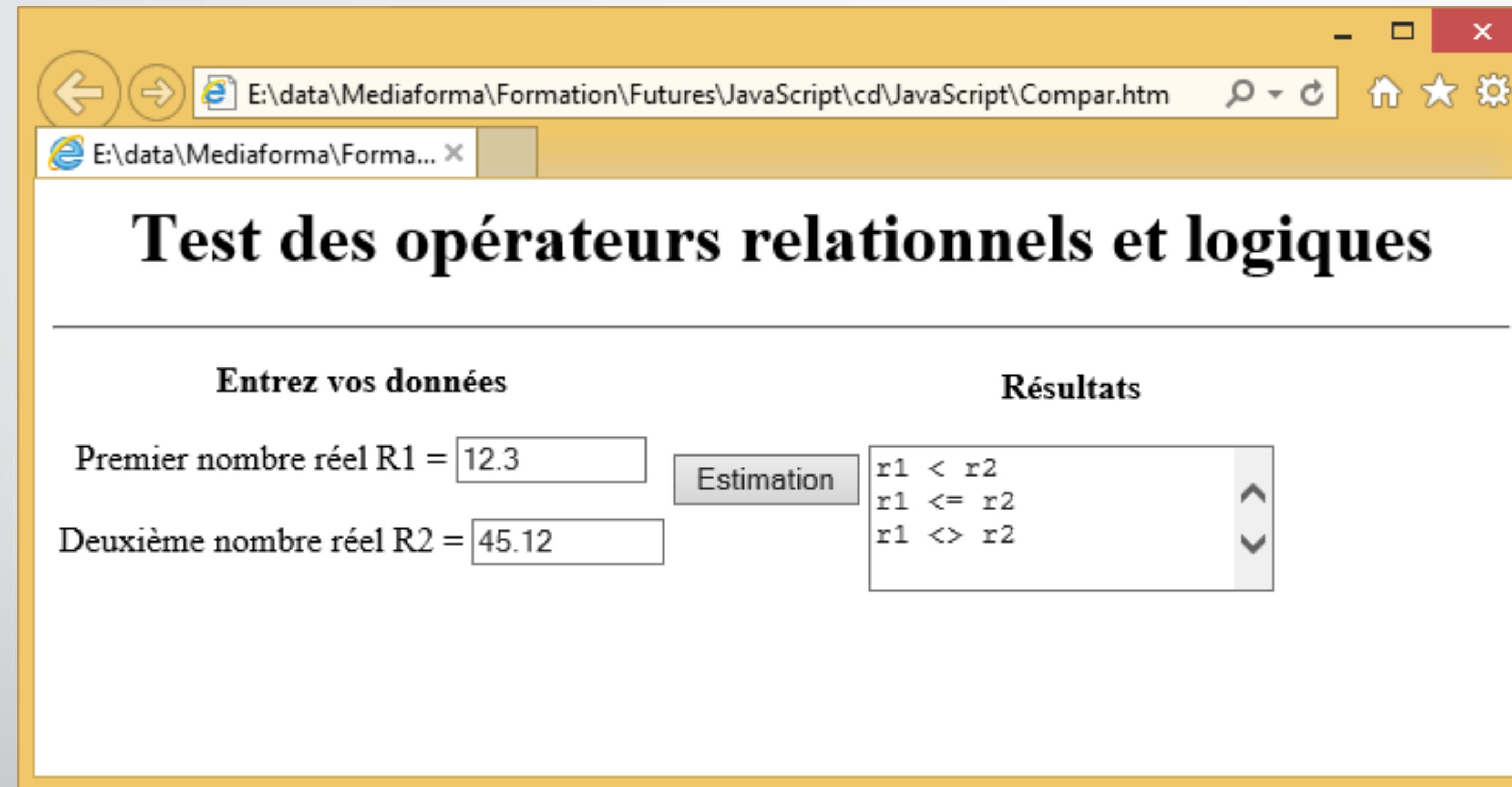
```
if (couleur = 3)
```

Ne compare pas le contenu de la variable couleur avec la valeur 3. Elle affecte au contraire la valeur 3 à la variable couleur ! L'instruction permettant d'effectuer la comparaison est :

```
if (couleur == 3)
```

Exercice :

Définissez le code JavaScript nécessaire pour obtenir le résultat suivant lorsque l'utilisateur clique sur le bouton **Estimation** :



Voici le code HTML :

```
<form name="demo">
  <table>
    <tr align="center">
      <td colspan="3">
        <h1>Test des opérateurs relationnels et logiques</h1>
        <hr>
      </td>
    </tr>
    <tr align="center">
      <td>
        <b>Entrez vos données </b></p>
        Premier nombre réel r1 = <input type="text" name="Reel1" size="10"><br>
        Deuxième nombre réel r2 = <input type="text" name="Reel2" size="10"><br>
      </td>
      <td>
        <input type="button" value="Estimation" onclick="Estimation(this.form);">
      </td>
      <td>
        <b>Résultats</b><br>
        <textarea name="Result" rows="4" cols="20"></textarea>
      </td>
    </tr>
  </table>
</form>
```

## Solution :

```
<script language="JavaScript">
    function Estimation(form) {
        // Récupération des données entrées
        var r1 = parseFloat(form.Reel1.value);
        var r2 = parseFloat(form.Reel2.value);
        // Estimation des données
        var r = "";
        var ChRes = "";
        if (r1 < r2) {
            r = "r1 < r2";
            ChRes = ChRes+r+"\r\n";
        }
        if (r1> r2) {
            r = "r1> r2";
            ChRes = ChRes+r+"\r\n";
        }
        if (r1 <= r2) {
            r = "r1 <= r2";
            ChRes = ChRes+r+"\r\n";
        }
        if (r1>= r2) {
            r = "r1>= r2";
            ChRes = ChRes+r+"\r\n";
        }
        if (r1 == r2) {
            r = "r1 = r2";
            ChRes = ChRes+r+"\r\n";
        }
        if (r1 != r2) {
            r = "r1 <> r2";
            ChRes = ChRes+r+"\r\n";
        }

        // Affichage du résultat
        form.Result.value = ChRes;
    }
</script>
```

## Opérateurs dédiés aux chaînes de caractères

En plus des opérateurs de comparaison dont nous venons de parler, vous pouvez utiliser les opérateurs + et += pour (respectivement) concaténer deux chaînes et stocker une chaîne à la suite d'une autre.

Exemples :

L'instruction ci-après :

```
a = "gauche " + "droite";
```

stocke la valeur "gauche droite" dans la variable a.

Supposons maintenant que la variable a contienne la valeur "gauche ". En utilisant l'instruction suivante, la variable a contiendra la valeur "gauche droite" :

```
a += "droite";
```

## Opérateurs d'incrémentation

Les opérateurs d'incrémentation (++) et de décrémentation (--) sont utilisés, respectivement pour augmenter et pour diminuer de 1 la valeur stockée dans une variable.

Selon la position de l'opérateur, on parle de post incrémentation, de pré incrémentation, de post décrémentation et de pré décrémentation :

Opérateur	Fonction	Exemple
++	post incrémentation	a++
++	pré incrémentation	++a
--	post décrémentation	a--
--	pré décrémentation	--a

Dans une expression comportant un opérateur de pré incrémentation ou de pré décrémentation, l'incrémentation (la décrémentation) se déroule avant l'évaluation de l'expression.

Inversement, dans une expression comportant un opérateur de post incrémentation ou de post décrémentation, l'incrémentation (la décrémentation) se déroule après l'évaluation de l'expression.

La position (post ou pré) des opérateurs d'incrémentation et de décrémentation n'a aucune importance dans une expression ne comportant qu'un facteur. Par exemple, les expressions suivantes sont la plupart du temps équivalentes :

`a++;`

est équivalent à :

`++a;`

Cependant, testez le code suivant :

```
var i = 5;  
var j = 5;  
var k = i++;  
var l = ++j;  
document.write(i + ' ' + j + ' ' + k + ' ' + l);
```

Qu'allez-vous obtenir ?

i et j valent 6.

Par contre, k vaut 5 et l vaut 6 !

Exercice :

Ecrivez du code HTML/JavaScript qui met en évidence l'importance de la post/pré incrémentation en appliquant les opérations  $E1 * ++E2$  et  $E1 * E2++$  à deux données saisies par l'utilisateur :

The screenshot shows a Microsoft Internet Explorer window with the following details:

- Title Bar:** Displays the URL `E:\data\Mediaforma\Formation\Futures\JavaScript\cd\JavaScript\IncrDecr.htm`.
- Content Area:**
  - Section Header:** **Test des opérateurs d'incrémentation / décrémentation**
  - Input Fields:** Two input fields labeled "Entrez vos données":
    - Premier nombre entier E1 =
    - Deuxième nombre entier E2 =
  - Button:** A button labeled "Inc/Déc" positioned next to the E2 input field.
  - Results Section:** A box titled "Résultats" containing two text blocks:
    - Soit  $E3 = E1 * ++E2$  avec  $E1=12$  et  $E2=16$   
Après le calcul, on obtient  $E2 = 16$  et  $E3 = 192$
    - Soit  $E3 = E1 * E2++$  avec  $E1=12$  et  $E2=16$   
Après le calcul, on obtient  $E2 = 16$  et  $E3 = 180$

Voici une variante du code :

```
<script language = "JavaScript">
    function IncDec(form) {
        // Récupération des données entrées
        var e1 = parseInt(form.Entier1.value)
        var e2 = parseInt(form.Entier2.value)

        // Sauvegarde de ces données
        var s1 = e1
        var s2 = e2

        // Premier calcul
        var e3 = e1*++e2
        var r = "Soit E3 = E1 * ++E2 avec E1 = "+e1+" et E2 = "+e2+"\r\n"
        r = r+" Après le calcul, on obtient E2 = "+e2+" et E3 = "+e3+"\r\n\r\n"
        var ChRes = r

        //Restitution des valeurs et deuxième calcul
        e1 = s1
        e2 = s2
        e3 = e1*e2++
        r = "Soit E3 = E1 * E2++ avec E1 = "+e1+" et E2 = "+e2+"\r\n"
        r = r+" Après le calcul, on obtient E2 = "+e2+" et E3 = "+e3
        ChRes = ChRes+r

        // Affichage du résultat
        form.Result.value = ChRes
    }
</script>
```

## Suite :

```
<form name="demo">
  <table>
    <tr align="center">
      <td colspan="2">
        <h1>Test des opérateurs d'incrémentation/décrémentation</h1>
        <hr>
      </td>
    </tr>
    <tr align="center">
      <td>
        <b>Entrez vos données </b>
        <p>Premier nombre entier e1 = <input type="text" name="Entier1" size="10"></p>
        <p>Deuxième nombre entier e2 = <input type="text" name="Entier2" size="10"></p>
      </td>
      <td>
        <input type="button" value="Inc/Déc" onclick="IncDec(this.form);">
      </td>
    </tr>
    <tr align="center">
      <td colspan="2">
        <b>Résultats</b><br>
        <textarea name="Result" rows="5" cols="60"></textarea>
      </td>
    </tr>
  </table>
</form>
```

## Opérations intervenant au niveau des bits

JavaScript possède plusieurs opérateurs qui effectuent des calculs au niveau binaire, tels que OU logique ou décalage d'un ou de plusieurs bits. Le tableau ci-après résume ces opérateurs.

Opérateur	Fonction	Exemple
&	ET logique	a = 0xfa & 0x35
	OU logique	a = 0xfa   0x12
^	OU EXCLUSIF logique	a ^ = 0x32
~	NON logique	a = ~b
>>	Décalage à droite avec signe	a>>= c
>>>	Décalage à droite sans signe	a>>>= c
<<	Décalage à gauche	a = b << c

### Attention

Les opérateurs `&`, `|` et `^` ne doivent pas être confondus avec leurs homologues `&&`, `||` et `^`. Les premiers effectuent des opérations binaires sur leurs opérandes, alors que les seconds effectuent des comparaisons sur leurs opérandes.

Exercice :

Définissez le code HTML et JavaScript nécessaire pour tester les opérateurs binaires. Vous pourriez réaliser le formulaire suivant :

The screenshot shows a web browser window with a yellow title bar. The address bar displays the URL `E:\data\Mediaforma\Formation\Futures\JavaScript\cd\JavaScript\Decal.htm`. The main content area has a title **Opérateurs intervenant au niveau binaire**. Below it, there is a section titled **Entrez vos données** containing three input fields:

- Premier opérande E1 =
- Deuxième opérande E2 =  Inc/Déc
- Décalage D =

Below these fields, there is a section titled **Appelons e1 et e2 les deux opérateurs et d le décalage**. A scrollable text box contains the following code:

```
e1 & e2 = 32
e1 | e2 = 231
e1 ^ e2 = 199
~e1 = -38
e1 << d = 296
e1 >> d = 4
e1 >>> d = 4
```

Voici le code HTML

```
<form name="demo">
  <table>
    <tr align="center">
      <td colspan="2">
        <h1>Test des opérateurs d'affectation</h1>
        <hr>
      </td>
    </tr>
    <tr align="center">
      <td>
        <b>Entrez vos données</b><br>
        Premier opérande e1 = <input type="text" name="Entier1" size="10"><br>
        Deuxième opérande e2=<input type="text" name="Entier2" size="10"><br>
        Décalage d = <input type="text" name="Decal" size="3"><p>
      </td>
      <td>
        <input type="button" value="Décalage" onclick="Decalage(this.form);">
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <b>Appelons e1 et e2 les deux opérateurs et d le décalage</b><p>
        <textarea name="Result" rows="7" cols="30"></textarea>
      </td>
    </tr>
  </table>
</form>
```

Et le code JavaScript

```
<script language = "JavaScript">
    function Decalage(form)
    {
        // Récupération des données entrées
        var e1 = parseInt(form.Entier1.value);
        var e2 = parseInt(form.Entier2.value);
        var d = parseInt(form.Decal.value);

        // Application des opérandes
        r = "e1 & e2 = "+eval(e1&e2)+"\r\n";
        r = r+ "e1 | e2 = "+eval(e1 | e2) +" \r\n";
        r = r+ "e1 ^ e2 = "+eval(e1 ^ e2) +" \r\n";
        r = r+"~e1 = "+eval(~e1) +" \r\n";
        e3 = e1<<d;
        r = r+"e1 << d = "+e3+"\r\n";
        e3 = e1>> d ;
        r = r+"e1>> d = "+e3+"\r\n";
        e3 = e1>>> d ;
        r = r+"e1>>> d = "+e3+"\r\n";

        // Affichage du résultat
        form.Result.value = r;
    }
</script>
```

# **L'instruction de contrôle if... else [.. else]**

Dans une fonction JavaScript ou entre les marqueurs `<script>` et `</script>`, les instructions s'exécutent séquentiellement. Il peut s'agir d'instructions provenant du langage de programmation lui-même ou d'appels de fonctions. Dans le second cas, les instructions appelées peuvent comporter une ou plusieurs instructions simples et/ou appels de fonctions.

Il est souvent nécessaire d'exécuter une portion de code, à la condition qu'une expression logique soit vérifiée. Pour cela, il faut utiliser l'instruction `if else` dont voici la syntaxe :

```
if (condition) {  
    instruction 1;  
    ...  
    instruction N;  
}  
else{  
    Instruction 1;  
    ...  
    Instruction P;  
}
```

- condition est une condition logique. Elle sera vérifiée si son évaluation est différente de 0. Elle ne sera pas vérifiée si son évaluation est égale à 0. Par exemple, la condition `if(3)` est toujours vérifiée, et la condition `if(0)` n'est jamais vérifiée.
- instruction 1 à instruction N sont les instructions exécutées dans le cas où la condition logique est vérifiée.
- instruction 1 à instruction P sont les instructions exécutées dans le cas contraire.

Exercice :

Ecrivez du code HTML/JavaScript pour :

- Demander à l'utilisateur d'entrer une lettre au clavier.
- Afficher un des trois messages suivants en fonction du caractère entré :

Lettre majuscule → Vous avez entré une lettre majuscule.

Lettre minuscule → Vous avez entré une lettre minuscule.

Autre caractère → Le caractère entré n'est pas une lettre.

## Solution :

```
<script language="JavaScript">
function test_casse(form) {
    // Récupération de la lettre entrée
    var L = form.Lettre.value;

    // Estimation des données
    if (L>= "A" && L <= "Z") {
        Res = "Vous avez entré une lettre majuscule";
    }
    else{
        if (L>= "a" && L <= "z") {
            Res = "Vous avez entré une lettre minuscule";
        }
        else {
            Res = "Le caractère entré n'est pas une lettre";
        }
    }

    // Affichage des résultats
    form.Result.value = Res;
}
</script>
```

```
<form name = "demo">


|                                                                                                                                                   |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <h1>Test de la casse d'une lettre</h1>                                                                                                            |  |
| <p>Entrez une lettre : <input name="Lettre" size="1" type="text"/><br/> <input onclick="test_casse(this.form);" type="button" value="test"/> </p> |  |
| <p><input name="Result" size="40" type="text"/></p>                                                                                               |  |


</form>
```

# **Instructions répétitives**

Dans un programme, il est souvent nécessaire de faire exécuter plusieurs fois un même bloc d'instructions. Plutôt que de réécrire plusieurs fois ce bloc, il est plus judicieux d'utiliser une instruction de répétition. JavaScript dispose de deux instructions de ce type : for et while. Nous allons examiner leur fonctionnement dans les diapositives suivantes.

## L'instruction for

L'instruction for exécute de façon répétitive un bloc d'instructions tant qu'une condition est vérifiée. Cette condition est basée sur la valeur d'une variable incrémentée par l'instruction. Voici la syntaxe de l'instruction for :

```
for (expression 1; expression 2; expression 3) {  
    instruction 1;  
    ...  
    instruction N;  
}
```

- expression 1 initialise la variable compteur utilisée dans la boucle ;
- expression 2 définit la condition qui doit être vérifiée pour que la boucle se poursuive ;
- expression 3 indique comment faire varier la variable compteur utilisée dans expression 2 ;
- instruction 1 à instruction N sont les instructions ou appels de fonctions exécutés tant que l'expression 2 est vérifiée.

## L'instruction for

## Les caractères de code 32 à 255

## Exercice :

Utilisez une boucle for pour afficher les caractères de code ASCII 32 à 255, comme dans la copie d'écran ci-contre :

## Solution :

```
<script language="JavaScript">
    document.write("<h1>L'instruction for </h1><hr>");
    document.write("Les caractères de code 32 à 255 \n\n");
    document.write("<pre>");
    for (var i=32; i<256; i++) {
        if ((i % 16) == 0){
            document.write("<br>");
        }
        document.write("&#" + i + " ");
    }
    document.write("</pre>");
</script>
```

Remarque :

La balise `<pre></pre>` permet d'afficher des caractères de largeur fixe, et donc d'aligner verticalement les caractères affichés sur l'écran.

## Un cas particulier de l'instruction **for**

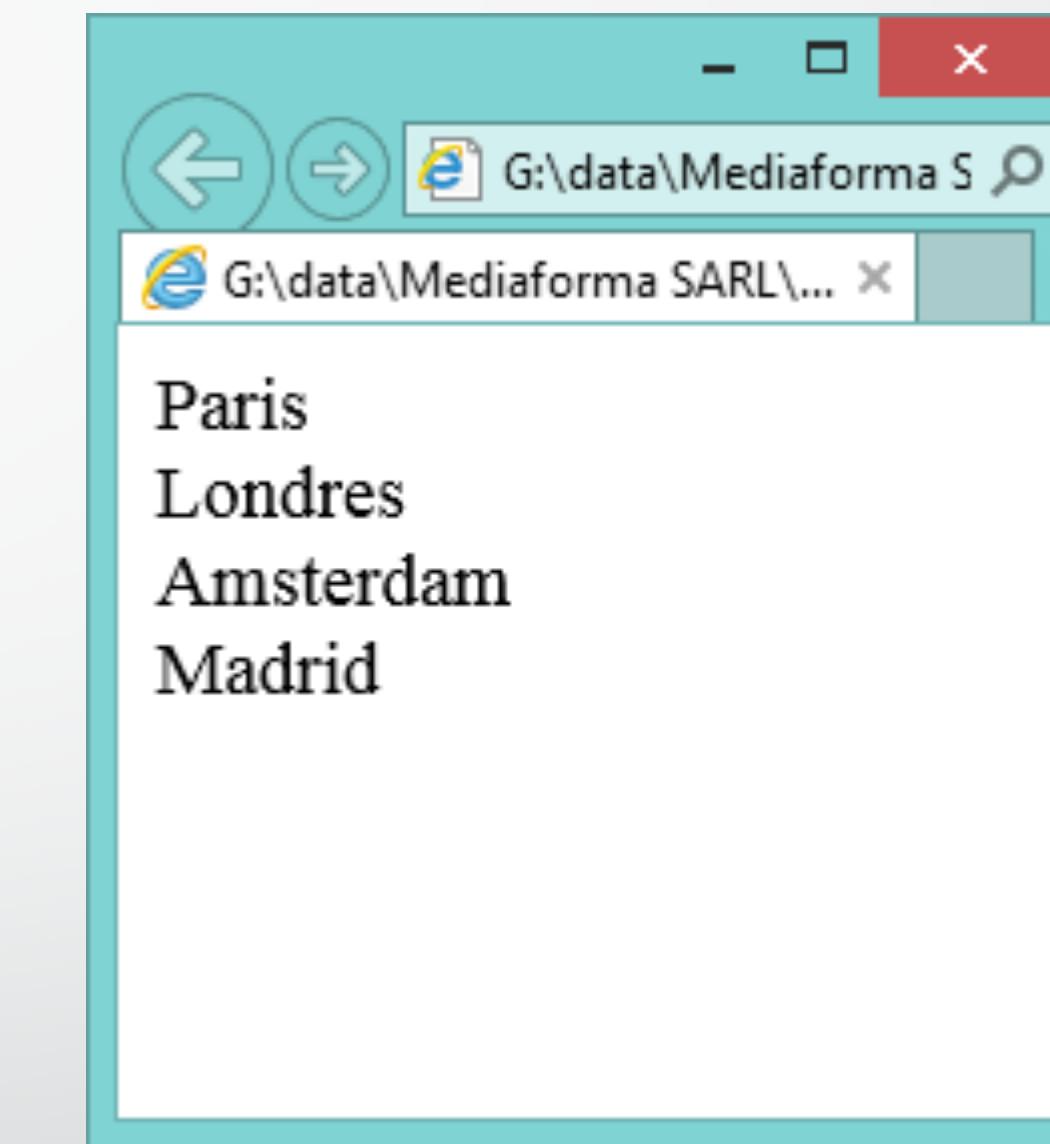
L'instruction for ... in accède séquentiellement aux propriétés d'un objet. Sa syntaxe est la suivante :

```
for (var in objet) {  
    instruction 1;  
    ...  
    instruction N;  
}
```

- var est la variable compteur utilisée pour décrire l'objet ;
- objet est un objet quelconque dont on désire connaître les propriétés ;
- instruction 1 à instruction N sont les instructions ou appels de fonctions exécutés pour chaque propriété de l'objet.

## Exemple : Passage en revue des cellules d'un tableau

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <script language="JavaScript">
        var t = ["Paris", "Londres", "Amsterdam",
"Madrid"];
        for (var i in t){
            document.write(t[i] + '<br>');
        }
    </script>
</body>
</html>
```



## L'instruction while

L'instruction while exécute de façon répétitive un bloc d'instructions tant qu'une condition est vérifiée. Le test de la condition est fait avant l'exécution de la boucle. Voici la syntaxe de l'instruction while :

```
while (expression) {  
    instruction 1;  
    ...  
    instruction N;  
}
```

où expression est une condition qui doit être vérifiée pour que les instructions de la boucle s'exécutent et instruction 1 à instruction N les instructions ou appels de fonctions exécutés tant que l'expression est vérifiée.

## Réaliser une boucle infinie

Pour définir une boucle infinie (qui boucle sans fin), vous utiliserez l'une des deux instructions suivantes :

```
for ( ; ; ) {  
    instructions;  
}
```

Ou :

```
while (1) {  
    instructions;  
}
```

## L'instruction continue

Il est parfois nécessaire d'interdire l'exécution d'une boucle lorsqu'une condition logique est satisfaite. Pour cela, vous utiliserez l'instruction **continue**, en tête du bloc :

Dans une instruction for :

```
for (expression 1; expression 2; expression 3) {  
    if (condition) {  
        continue;  
    }  
    instruction 1;  
    ...  
    instruction N;  
}
```

Dans une instruction while :

```
while (expression) {  
    if (condition) {  
        continue;  
    }  
    instruction 1;  
    ...  
    instruction N;  
}
```

Exemple :

Le code de la diapositive suivante calcule le résultat de la fonction  $y = 1/(x - 3)$  pour des valeurs comprises entre 2 et 4, par pas de 0.1. Il est nécessaire d'exclure la valeur 3 du calcul qui donne un résultat indéterminé. Une instruction **continue** s'en charge.

Saisissez et testez le code suivant :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script language = "JavaScript">
function calcul(){
    document.write("<PRE>");
    var x;
    document.write("<B>Résultats :</B>\n\n");
    for (x = 2; x<= 4; x = x+0.1)
    {
        if (((x-3)<0.01) && ((x-3)>-0.01))
            continue; // Interdit l'exécution de la boucle pour x = 3
        document.write("x = " + Math.round(x*10)/10,"t y = " + Math.round(1/(x-3)*100)/100,"\\n");
    }
    document.write("</PRE>");
}
</script>
</head>
<body>
<h1>Test de l'instruction continue</h1>
<hr>
<form>
Appuyez sur le bouton Calcul pour calculer les valeurs de la fonction  $y = 1/(x - 3)$  pour x compris entre 2 et 4
<input type="button" name="Calcul" value="Calcul" onclick="calcul();">
</form>
</body>
</html>
```

## L'instruction `break`

Il est parfois nécessaire d'arrêter l'exécution de la boucle lorsqu'une condition logique est satisfaite. Pour cela, utilisez l'instruction `break` en tête du bloc :

Dans une instruction `for` :

```
for (expression 1; expression 2; expression 3) {  
    if (condition) {  
        break;  
    }  
    instruction 1;  
    ...  
    instruction N;  
}
```

Dans une instruction `while` :

```
while (expression) {  
    if (condition) {  
        break;  
    }  
    instruction 1;  
    ...  
    instruction N;  
}
```

# ***Des commentaires dans le texte***

Quel que soit le langage utilisé, il est toujours intéressant d'insérer des commentaires dans un programme. Cela facilite sa maintenance et précise les points épineux.

Comme nous l'avons déjà vu, la balise suivante permet d'insérer des commentaires dans un document HTML.

```
<!-- Ceci est un commentaire -->
```

Si les commentaires doivent être placés entre les balises <script> et </script>, vous procérez différemment :

- Pour insérer une ligne de commentaire, faites-la commencer par un double slash.
- Pour insérer plusieurs lignes de commentaire consécutives, encadrez-les par les caractères /\* et \*/.

# **L'instruction return**

Une fonction JavaScript est en mesure de renvoyer une valeur calculée ou issue d'une variable. Pour cela, il suffit d'utiliser l'instruction return à l'intérieur de la fonction.

Exercice :

Définissez un formulaire contenant deux zones de texte et un bouton.

L'utilisateur entre une valeur hexadécimale dans la première zone de texte et appuie sur le bouton. La valeur décimale correspondante est alors affichée dans la deuxième zone de texte.



Pour effectuer une conversion hexa -> decimal, vous utiliserez la function parseInt :

```
parseInt(St, 16);
```

## Solution :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
    <h1>Test de l'instruction return</h1>
    <hr>
    <form>
        Entrez la valeur hexa à convertir, puis appuyez sur le bouton
        <input type="text" name="Hex" size="4">
        <input type="button" value="->" onclick="Conversion(this.form);">
        <input type="text" name="Dec" size="4">
    </form>
    <script language = "JavaScript">
        function HexDec(St) {
            return parseInt(St,16);
        }
        function Conversion(form) {
            var valeur = HexDec(form.Hex.value);
            form.Dec.value = valeur;
        }
    </script>
</body>
</html>
```

# Définition d'un tableau en JavaScript Objet

La définition d'un tableau peut se faire de façon traditionnelle ou en instanciant la classe **Array**.

De façon traditionnelle :

```
var t = [] ;
```

En objet :

```
var t = new Array;
```

Pour définir les éléments d'un tableau lors de sa création.

De façon traditionnelle :

```
var t = ['chien', 'chat', 'cochon'] ;
```

En objet :

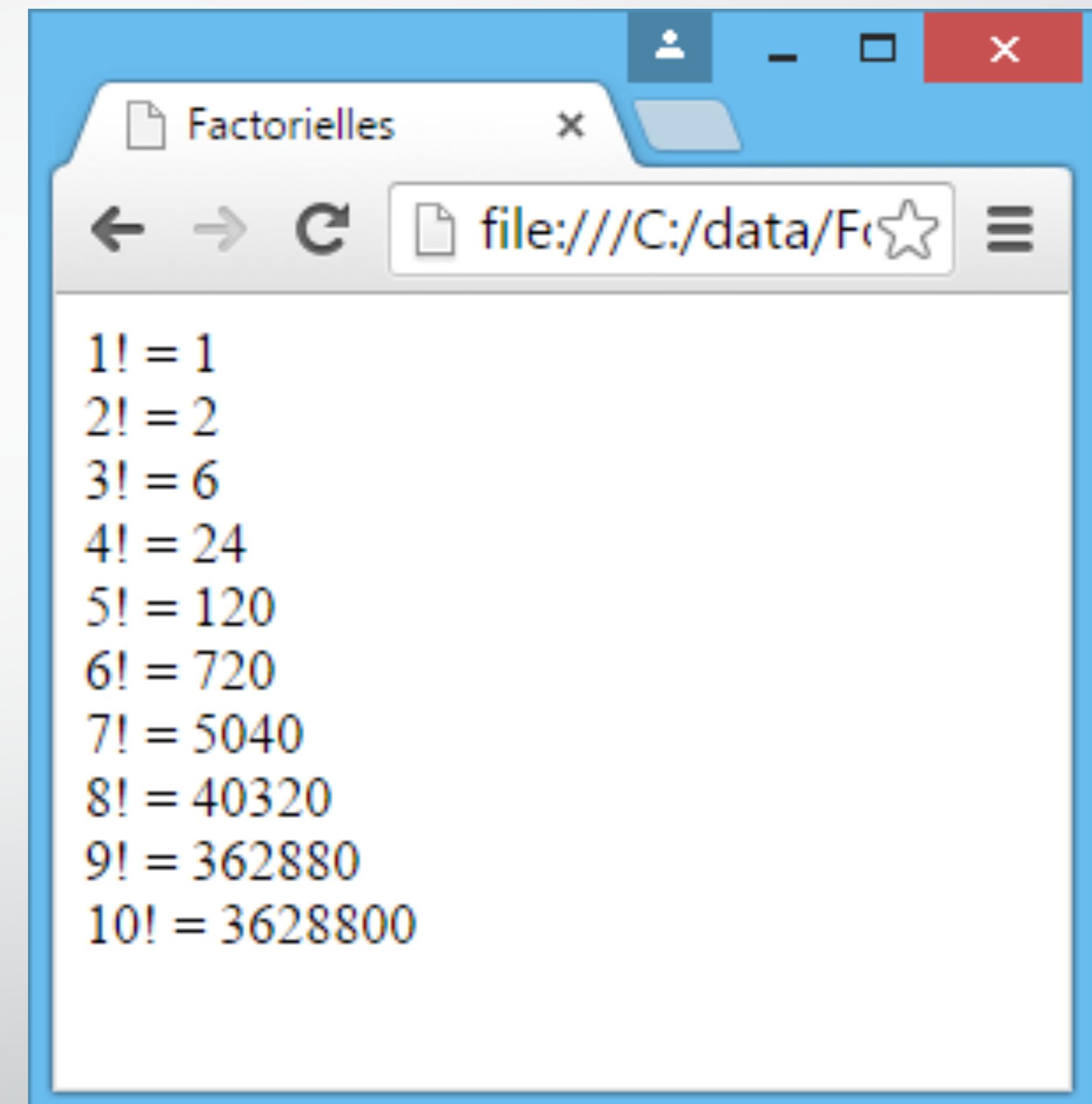
```
var t = new Array('chien', 'chat', 'cochon') ;
```

## Exercice

Définissez le code HTML et JavaScript pour calculer les factorielles de  $1!$  à  $10!$ . Stockez-les dans un tableau et affichez le contenu de ce tableau.

# Solution

```
<html>
<head>
  <meta charset="utf-8">
  <title>Factorielles</title>
</head>
<body>
  <script>
    var t = new Array;
    t[1]=1;
    for (var i=2; i<=10; i++)
      t[i] = t[i-1]*i;
    for (i=1; i<=10; i++)
      document.write(i + ' ! = ' + t[i] + '<br>');
  </script>
</body>
</html>
```



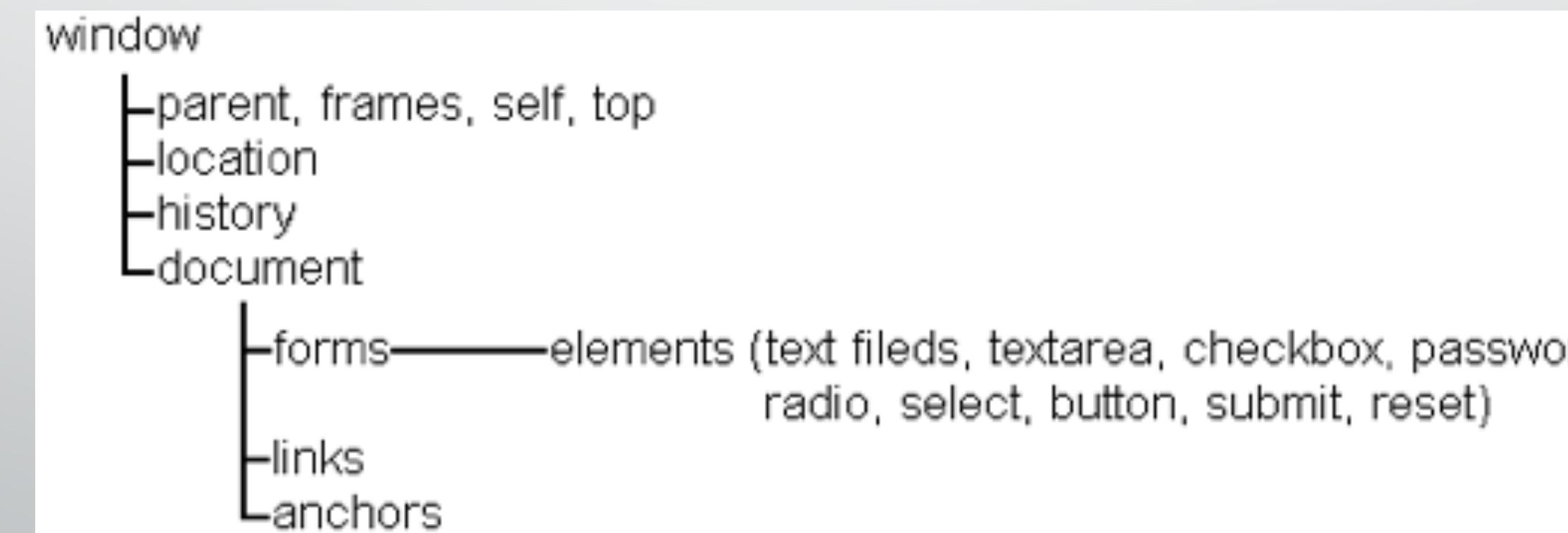
# **Les objets standards de JavaScript**

JavaScript est en mesure de manipuler divers types d'objets. Nous allons les passer en revue dans les pages qui suivent.

## **Les objets liés au navigateur**

Lorsqu'une page est chargée dans le navigateur, plusieurs objets sont automatiquement créés :

- **window** : la fenêtre affichée dans le navigateur.
- **location** : l'URL courante et ses propriétés.
- **history** : les URL déjà visitées.
- **document** : donne accès aux propriétés du document courant (titre, couleur d'arrière-plan, etc.).



Supposons qu'une page HTML nommée hexdec.htm soit disponible à cette URL :  
<http://www.mediaforma.com/hexdec.htm>. Voici son code :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Conversion Hexa->Décimal</TITLE>
    <script language = "JavaScript">
        function HexDec(St) {
            return parseInt(St,16);
        }
        function Conversion(form) {
            valeur = HexDec(form.Hex.value);
            form.Dec.value = valeur;
        }
    </script>
</head>
<body>
    <h1>Conversion hexa -> decimal</h1>
    <hr>
    <form name = "Donnees">
        Entrez la valeur hexa à convertir, puis appuyez sur le bouton
        <input type = "text" name = "Hex" size = 4>
        <input type = "button" name = "Action" value = "->" onclick = "Conversion(this.form)">
        <input type = "text" name = "Dec" size = 4>
    </form>
</body>
</html>
```

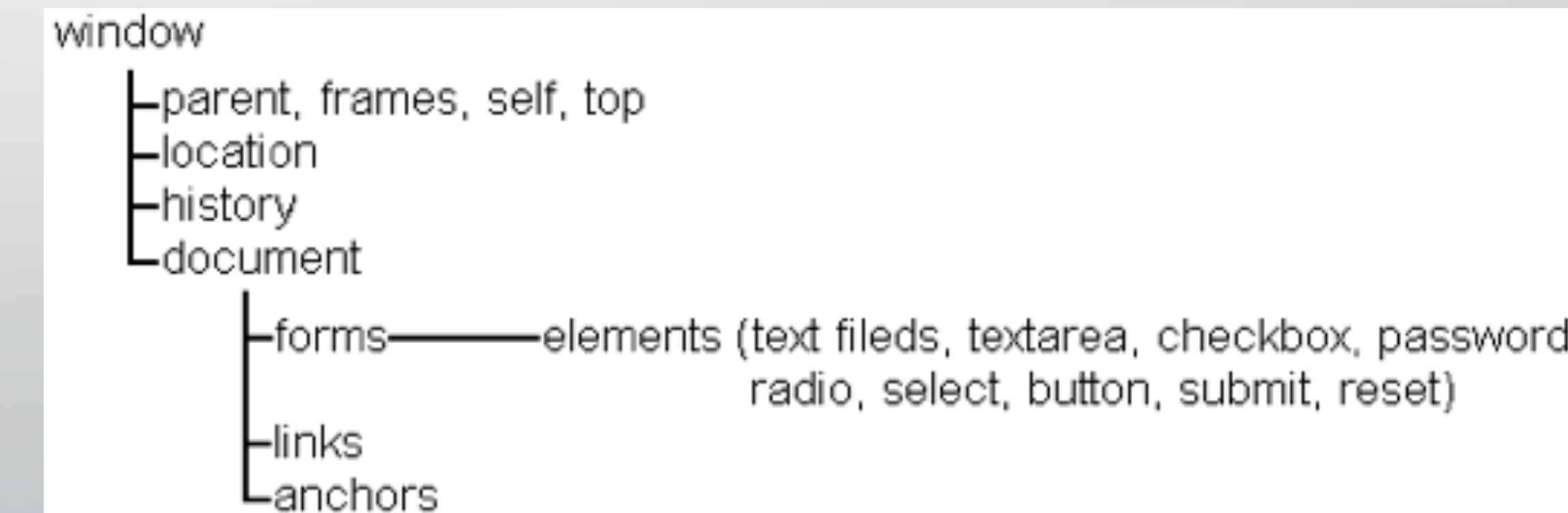
Les objets standard location, history et document sont automatiquement créés lors du chargement de ce document. Voici quelques-unes de leurs propriétés :

```
location.href = "http://www.google.fr"  
document.title = "Conversion Hexa->Décimal"  
document.body.style.color = #000000  
document.body.style.backgroundColor = #ffffff  
history.length = 5 (par exemple)  
document.Donnees.Hex.name = "Hex"  
document.Donnees.Hex.value = "3F" (par exemple)  
document.Donnees.Action.value = "->"  
document.Donnees.Dec.value = 63 (par exemple)
```

Comme vous le voyez, l'accès aux objets se fait à l'aide d'instructions "à point". Ces instructions respectent la hiérarchie de la figure, en omettant toutefois l'objet window. Ainsi, par exemple, l'entité document.Donnees.Action.value représente la propriété value du contrôle Action. Ce contrôle fait partie du formulaire Données, qui est lui-même un sous-ensemble de l'objet document.

Cet exemple contient un formulaire qui a été nommé à l'aide du marqueur suivant :

```
<form name = "Donnees">  
</form>
```



Il est ainsi possible d'accéder aux éléments du formulaire à l'aide de l'entité document.Donnees. Lorsque le formulaire n'a aucun nom, il est cependant représenté par le tableau forms[]. Le premier formulaire rencontré dans la page est appelé forms[0], le deuxième forms[1], et ainsi de suite.

Pour accéder à la propriété value du contrôle Action, vous pouvez donc utiliser indifféremment les deux entités suivantes (l'indice 0 laisse supposer que le formulaire est le premier de la page) :

```
document.Donnees.Action.value
```

```
document.forms [0] .Action.value
```

# Les autres objets

Nous allons maintenant nous intéresser aux autres objets accessibles au langage JavaScript. Dans la mesure du possible, nous définirons un ou plusieurs exemples pour faciliter leur compréhension.

## *L'objet anchor*

Les anchors permettent de définir des signets qui pointent sur les destinations à accéder ou des liens hypertexte vers des signets. Reportez-vous au paragraphe intitulé "Liens hypertexte" dans la première partie de cet ouvrage pour plus d'informations sur ces marqueurs.

La syntaxe HTML permettant de définir un anchor est la suivante :

```
<a [href = "#Signet"] name = "Nom" [target = "Nom de la fenêtre"]>  
    Texte  
</a>
```

- Signet est le nom du signet à atteindre.
- Nom est le nom donné au signet.
- Nom de la fenêtre est le nom de la fenêtre dans laquelle est défini le lien. Ce paramètre n'a un sens que dans le cas où le marqueur est utilisé pour définir un signet.
- Texte est le texte utilisé pour définir le signet ou le lien hypertexte.

Lorsqu'un document contient des marqueurs `<a>`, il est possible d'utiliser la propriété `document.anchors.length` pour en connaître le nombre.

Lorsqu'un marqueur est utilisé pour définir un lien hypertexte, il est référencé dans les tableaux `anchors[]` et `links[]`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <a name="Aligne"><H1>Alignement</H1></a>
    <p align="LEFT">Ce texte est aligné à gauche</p>
    <p align="CENTER">Ce texte est centré</p>
    <p align="RIGHT">Ce texte est aligné à droite</p>

    <a name="Attrib"><H1>Attributs de caractères</H1></a>
    Le mot <B>gras</B> est en gras, le mot <I>italique</I> en italique.
  </body>
</html>
```

Définissez le code nécessaire pour afficher le nombre d'ancres et les ancre

```
<script>
    var nombre=document.anchors.length;
    document.write('Nombre de balises &lt;a&gt; dans le document : ' +
nombre + '<br>');
    for (var i=0; i<=nombre; i++)
        document.write(document.anchors[i].innerHTML + '<br>');
</script>
```

## ***L'objet button***

Les objets button correspondent aux boutons de commande placés dans les formulaires HTML.

La syntaxe permettant de définir un bouton est la suivante :

```
<input type="button" name="Nom" value="Valeur" [ONCLICK="Traitement"]>
```

- Nom est le nom du bouton ;
- Valeur est le texte affiché sur le bouton ;
- Traitement est une éventuelle instruction ou fonction de traitement JavaScript activée lorsque le bouton est cliqué.

Deux propriétés sont accessibles en lecture et en écriture au code JavaScript : name et value. En d'autres termes, le nom et la légende du bouton. Pour ce faire, vous utiliserez les syntaxes ci-après :

```
document.NomFormulaire.NomBouton.name  
document.NomFormulaire.elements[N].name  
document.NomFormulaire.NomBouton.value  
document.NomFormulaire.elements[N].value
```

- NomFormulaire est le nom du formulaire donné dans le marqueur <form> ;
- NomBouton est le nom du bouton donné dans le marqueur <input> ;
- N est le numéro d'ordre du bouton dans le formulaire.

## *L'objet checkbox*

Les objets checkbox correspondent aux cases à cocher placées dans les formulaires HTML.

La syntaxe permettant de définir un objet checkbox est la suivante :

```
<input type="checkbox" name="NOM" value="Valeur"  
      [checked] [onclick="Traitement"]>
```

Légende de la case à cocher

- Nom est le nom de la case à cocher ;
- Valeur est la valeur renvoyée au serveur lorsque la case est cochée ;
- [checked] (s'il est précisé) coche la case par défaut ;
- Traitement est une éventuelle instruction ou fonction de traitement JavaScript activée lorsque l'utilisateur clique sur la case.

Quatre propriétés sont accessibles au code JavaScript : checked, defaultChecked, name et value. Elles correspondent à l'état du bouton radio, à l'état par défaut, au nom et à la valeur du bouton radio.

Pour accéder à ces propriétés, vous utiliserez les syntaxes ci-après :

```
document.NomFormulaire.NomBouton.checked  
document.NomFormulaire.NomBouton.defaultChecked  
document.NomFormulaire.NomBouton.name  
document.NomFormulaire.NomBouton.value
```

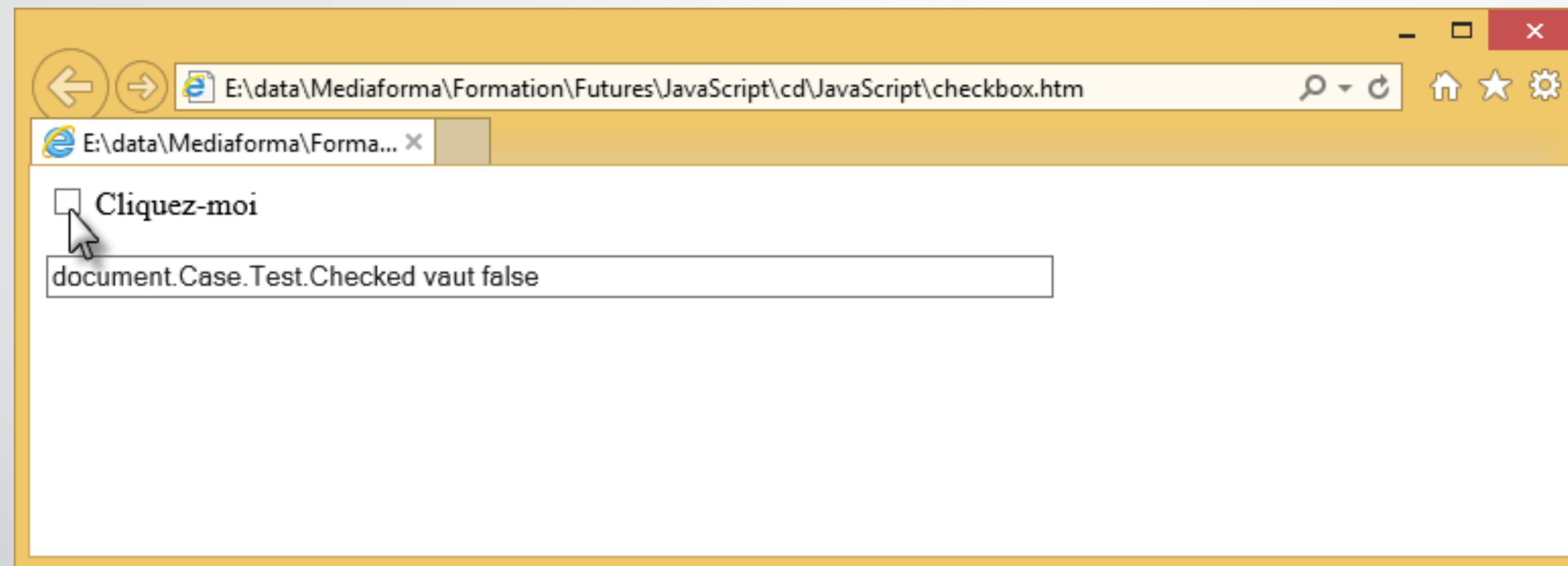
ou encore :

```
document.NomFormulaire.elements[N].checked  
document.NomFormulaire.elements[N].defaultChecked  
document.NomFormulaire.elements[N].name  
document.NomFormulaire.elements[N].value
```

- NomFormulaire est le nom du formulaire tel qu'il apparaît dans le marqueur <form> ;
- NomBouton est le nom du bouton tel qu'il apparaît dans le marqueur <input> ;
- N est le numéro d'ordre du bouton dans le formulaire. La méthode click() est associée aux objets checkbox. Reportez-vous à l'Annexe 2 pour savoir comment l'utiliser.

## Exercice

Ecrivez le code HTML et JavaScript nécessaire pour afficher la valeur de la propriété Checked d'une case à cocher dans une zone de texte :



## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <form name="Case">
      <input type="checkbox" value="Case" name="Test" onclick="cli(this.form)">
      Cliquez-moi
      <input type="text" name="resultat" size="80">
    </form>
    <script>
      function cli(form) {
        form.resultat.value = "document.Case.Test.checked vaut "+document.Case.Test.checked;
      }
    </script>
  </body>
</html>
```

## *L'objet document*

L'objet document donne accès aux propriétés du document courant. Le code HTML permettant de définir un document est placé entre les marqueurs <body> et </body> :

```
<body>  
    Corps du document  
</body>
```

Le marqueur <body> peut comporter de nombreux paramètres :

```
<body  
  [background] = Image ou URL,  
  [bgcolor] = "#Couleur arrière-plan"  
  [text] = "#Couleur du texte"  
  [link] = "#Couleur des liens"  
  [alink] = "#Couleur du lien actif"  
  [vlink] = "#Couleur des liens visités"  
  [onLoad = "Instruction"]  
  [onUnload = "Instruction"]>
```

Tous les paramètres sont facultatifs.

- background définit le nom de l'image à afficher sur l'arrière-plan ou son URL ;
- bgcolor définit la couleur uniforme de l'arrière-plan ;
- text, link, alink et vlink définissent respectivement la couleur du texte, des liens, du lien actif et des liens déjà visités ;
- OnLoad et onUnload font référence à une instruction ou à une fonction JavaScript exécutée (respectivement) au chargement de la page et au passage à la page suivante.

Pour accéder à un objet document, vous utiliserez l'une des deux syntaxes suivantes :

document.propriété

ou

document.méthode(paramètres)

- propriété est une propriété du document (alinkColor, anchors, bgColor, cookie, fgColor, forms, lastModified, linkColor, links, location, referrer, title ou vlink) ;
- méthode est une méthode applicable à l'objet document (clear, close, open, write ou writeln) ;
- paramètres correspond aux éventuels paramètres passés à la méthode.

## Exercice

Affichez un texte H1 de couleur cyan. Définissez un formulaire comportant deux zones de texte et deux boutons. Définissez le code HTML et JavaScript nécessaire pour :

- 1) afficher la couleur hexadécimale du texte dans la première zone de texte lorsque l'utilisateur clique sur le premier bouton
- 2) Modifier la couleur d'arrière-plan de la page en utilisant le code couleur de la deuxième zone de texte lorsque l'utilisateur clique sur le deuxième bouton.



## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
    <script language = "JavaScript">
      function Lit_Couleur_Texte(form) {
        form.Texte.value = document.bgColor;
      }
      function Change_Couleur_Fond(form) {
        document.bgColor = form.Fond.value;
      }
    </script>
  </head>
  <body text = "cyan">
    <h1>Les propriétés fgColor et bgColor</h1>
    <hr>
    <form name="leform">
      <input type="text" name="Texte" SIZE="25">
      <input type="button" value="Lit couleur texte" onclick="Lit_Couleur_Texte(this.form)">
      <input type="text" name="Fond" SIZE="25">
      <input type="button" value="Modifie couleur fond" onclick="Change_Couleur_Fond(this.form)">
    </form>
  </body>
</html>
```

## *Le tableau elements[]*

Les contrôles qui composent un formulaire peuvent être référencés à l'aide du tableau elements[].

Supposons qu'un formulaire nommé Test comporte :

- Une zone de texte de nom Saisie.
- Une case à cocher de nom Option.
- Un bouton de commande de nom Action.

Ces contrôles sont accessibles à l'aide des instructions suivantes :

```
Test.Saisie.value  
Test.Option.value  
Test.Action.value
```

Mais également par l'intermédiaire du tableau elements[] :

```
Test.elements[0].value  
Test.elements[1].value  
Test.elements[2].value
```

Ce deuxième jeu d'instructions suppose que le code HTML a défini, dans l'ordre, les contrôles Saisie, Option puis Action.

En utilisant la propriété length, vous pouvez connaître le nombre d'entrées dans le tableau elements[], c'est-à-dire le nombre de contrôles définis dans le formulaire.

Pour reprendre l'exemple précédent, l'expression :

```
Test.elements.length
```

renvoie la valeur 3. Cela signifie que trois contrôles ont été définis dans le formulaire.

### Exercice

En partant de l'exercice précédent, ajoutez un bouton de commande dans le formulaire et affichez la valeur des quatre contrôles contenus dans le formulaire.

# Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
    <script language = "JavaScript">
      function Lit_Couleur_Texte(form) {
        form.Texte.value = document.bgColor
      }
      function Change_Couleur_Fond(form) {
        document.bgColor = form.Fond.value
      }
      function Affiche_Controles(form) {
        var nbControls = form.elements.length;
        document.write("<pre>Il y a ", nbControls, " contrôles dans ce formulaire.<p>");
        document.write("Voici leurs valeurs :<br>");
        for (i=0; i<nbControls; i++) {
          document.write("Contrôle n° " + i + " : " + form.elements[i].value + "<br>");
        }
      }
    </script>
  </head>

  <body text = "cyan">
    <h1>Les propriétés fgColor et bgColor</h1>
    <hr>
    <form name="leform">
      <input type="text" name="Texte" size="25">
      <input type="button" value="Lit couleur texte" onclick="Lit_Couleur_Texte(this.form);">
      <input type="text" name="Fond" size="25">
      <input type="button" value="Modifie couleur fond" onclick="Change_Couleur_Fond(this.form);">
      <input type="button" value="Valeur des contrôles" onclick="Affiche_Controles(this.form);">
    </form>
  </body>
</html>
```

## *L'objet form*

Les formulaires sont utilisés pour afficher des contrôles évolués sur une page HTML (zones de texte, boutons radio, cases à cocher, etc.) afin que l'utilisateur puisse entrer des données. Ces dernières seront communiquées au serveur par un appui sur le bouton de commande submit.

La syntaxe permettant de définir un objet form est la suivante :

```
<form  
    name = "Nom"  
    target = "Fenêtre"  
    action = Traitement  
    method = {get|post}  
    enctype = "Type d'encodage"  
    onSubmit = "Fonction">  
</form>
```

- name est le nom du formulaire ;
- target est la fenêtre dans laquelle doit être affiché le résultat renvoyé par le serveur (il peut s'agir d'une fenêtre ou d'un cadre existant) ;
- action est le nom d'un programme de traitement, par exemple /cgi-bin/Trait.pl ou encore l'adresse de votre boîte aux lettres, précédée de mailto. Par exemple <mailto:pierre.durand@Bertold.fr> ;
- method définit la méthode à utiliser pour prendre en charge les données du formulaire. La méthode get() étant limitée quant à la quantité de données transmissibles, la méthode post() est presque toujours utilisée ;
- enctype est le type d'encodage utilisé pour transmettre les données ;
- onSubmit spécifie le nom de la fonction JavaScript activée lors d'un appui sur le bouton Submit.

Pour accéder à un formulaire en JavaScript, vous utiliserez l'une des deux syntaxes suivantes :

Nom\_formulaire.propriété  
forms [index].propriété

où Nom\_formulaire est la valeur affectée au paramètre NAME dans le marqueur <FORM>, et propriété l'une des propriétés suivantes :

- action : correspond au paramètre ACTION.
- elements[] : tableau qui fait référence aux contrôles du formulaire.
- encoding : correspond au paramètre ENCTYPE.
- length : nombre de contrôles dans le formulaire.
- method : correspond au paramètre METHOD.
- target : correspond au paramètre TARGET.
- index est le numéro du formulaire à atteindre.

Dans la deuxième syntaxe, le terme "propriété" ne peut prendre que la valeur length.

## ***L'objet hidden***

Bien qu'ils fassent partie d'un formulaire, les objets hidden n'apparaissent pas sur l'écran. Ils sont utilisés pour envoyer le nom et la valeur d'une variable au serveur sans en informer l'utilisateur final.

La syntaxe permettant de définir un objet hidden est la suivante :

```
<input  
    type = "hidden"  
    name = "Nom"  
    value = "Texte">
```

où Nom est le nom de l'objet, et Texte sa valeur initiale.

Pour accéder à un objet hidden en JavaScript, vous utiliserez l'une des deux syntaxes suivantes :

Nom.Propriété  
NomFormulaire.elements[index].Propriété

- Nom est le nom de l'objet ;
- Propriété est la propriété à atteindre (name ou value) ;
- NomFormulaire est le nom du formulaire où se trouve l'objet hidden (variable name dans le marqueur <form>) ;
- index est le numéro d'ordre de l'objet hidden dans le formulaire.

## ***L'objet history***

L'objet history contient les URL des divers sites visités.

Pour connaître le nombre d'entrées dans l'objet history, vous utiliserez l'expression history.length.

Pour vous déplacer dans l'objet history, vous utiliserez la méthode back() (vers l'arrière), forward() (vers l'avant) ou go() (saut chiffré) dans une expression du type :

```
history.méthode ( [paramètre] ) ;
```

La syntaxe la plus simple est la suivante :

```
history.back () ;
```

Elle affiche la page précédemment visitée dans la fenêtre active.

## Exercice :

Entraînez-vous à manipuler l'objet history.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
  </head>

  <body>
    <h1>L'objet history</h1>
    <hr>
    <a href="test.htm">test</a>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
  </head>

  <body>
    <h1>L'objet history</h1>
    <hr>
    <a href="javascript:history.back()">Page Précédente</a>
    <form>
      <input type="button" value="Précédent"
             onclick="javascript:history.back()">
    </form>
  </body>
</html>
```

## *L'objet link*

Les objets link définissent un lien hypertexte ou hypermédia. Lorsque l'utilisateur clique sur un tel objet, le document HTML référence est chargé.

La définition d'un objet link se fait à l'aide du marqueur   :

```
<a  
    href = "URL"  
    name = "Nom du lien"  
    target = "Nom de la fenêtre"  
    onClick = "fonction"  
    onMouseOver = "fonction">  
    Texte  
</a>
```

- url est l'adresse du lien à atteindre ;
- nom définit un signet sur le texte spécifié entre les marqueurs  et  ;
- target indique le nom de la fenêtre dans laquelle doit s'afficher le document lié ;
- onClick déclenche une instruction ou une fonction JavaScript lorsque l'utilisateur clique sur le lien ;
- onMouseOver déclenche une instruction ou une fonction JavaScript lorsque le pointeur de la souris se trouve au-dessus du lien.

Pour accéder à un objet link, vous utiliserez le tableau `links[]` comme suit :

```
document.links[index]
```

où `index` est le numéro d'ordre de l'objet `link` sur la page.

Vous pouvez connaître le nombre de liens contenus dans la page en utilisant la propriété `length` de l'objet `link` :

```
document.links.length
```

Les objets link possèdent de nombreuses propriétés accessibles à travers le tableau links[] :

- hash : fait référence au nom du lien (NAME).
- host : nom de l'URL visée (une partie de REF).
- hostname : nom et domaine de l'URL visée (une partie de REF).
- href : URL visée (REF).
- pathname : path de l'URL (une partie de REF).
- port : port de communication utilisé par le serveur.
- protocol : début de l'URL (une partie de REF).
- search : l'éventuelle question posée dans l'URL (une partie de REF).
- target : nom de la fenêtre (TARGET).

### Exercice

Utilisez les propriétés de l'objet link pour décortiquer et afficher les différentes parties de l'URL suivante :

<https://www.google.fr/search?site=&source=hp&q=java+javascript&oq=java+javascript>

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre du document</title>
  </head>
  <body>
    <h1>Les propriétés d'un objet link</h1>
    <hr>
    <a href = "https://www.google.fr/search?site=&source=hp&q=java+javascript&oq=java+javascript">
      Google.fr sur les termes java et javascript
    </a>

    <script language = "JavaScript">
      document.write("<p>Le lien utilisé est <b>https://www.google.fr/search?site=&source=hp&q=java+javascript&oq=java+javascript</b>");

      document.write("<P>document.links[0].hash = ",document.links[0].hash);
      document.write("<BR>document.links[0].host = ",document.links[0].host);
      document.write("<BR>document.links[0].hostname = ",document.links[0].hostname);
      document.write("<BR>document.links[0].href = ",document.links[0].href);
      document.write("<BR>document.links[0].pathname = ",document.links[0].pathname);
      document.write("<BR>document.links[0].port = ",document.links[0].port);
      document.write("<BR>document.links[0].protocol = ",document.links[0].protocol);
      document.write("<BR>document.links[0].search = ",document.links[0].search);
    </script>
  </body>
</html>
```

## ***L'objet location***

L'objet location fait référence à l'URL courante. Il est défini implicitement par le navigateur. Pour y accéder, vous utiliserez la syntaxe suivante :

location.Propriété

où Propriété est l'une des propriétés suivantes :

- protocol : début de l'URL ;
- hostname : nom et domaine de l'URL visé ;
- port : port de communication utilisé par le serveur ;
- href : URL visé ;
- pathname : path de l'URL ;
- search : l'éventuelle question posée dans l'URL ;
- hash : fait référence au nom du lien.

# *L'objet Math*

L'objet Math donne accès à huit constantes accessibles sous la forme d'une propriété de l'objet Math :

Intitulé	Constante
E	Constante d'Euler (environ 2,718)
LN2	Logarithme népérien de 2 (environ 0,693)
LN10	Logarithme népérien de 10 (environ 2,3026)
LOG2E	Logarithme à base 2 de E (environ 1,442)
LOG10E	Logarithme à base 10 de E (environ 0,434)
PI	Constante PI (environ 3,14159265)
SQRT1_2	Racine carrée de _ (environ 0,7071)
SQRT2	Racine carrée de 2 (environ 1,4142)

Pour accéder à la constante LOG10E, vous utiliserez l'instruction

`Math.LOG10E`

ou encore le bloc suivant :

```
with (Math) {  
    LOG10E...  
}
```

Cette deuxième version facilite l'écriture si vous devez manipuler de nombreuses constantes et méthodes

L'objet **Math** donne également accès à dix-sept méthodes :

Méthode	Signification
<b>abs()</b>	Valeur absolue
<b>acos()</b>	Arc cosinus
<b>asin()</b>	Arc sinus
<b>atan()</b>	Arc tangente
<b>ceil()</b>	Entier supérieur ou égal à la valeur spécifiée
<b>cos()</b>	Cosinus
<b>exp()</b>	Exponentielle
<b>floor()</b>	Entier inférieur ou égal à la valeur spécifiée
<b>log()</b>	Logarithme décimal
<b>max()</b>	Valeur maximale
<b>min()</b>	Valeur minimale
<b>pow()</b>	Mise à la puissance
<b>random()</b>	Nombre aléatoire
<b>round()</b>	Arrondi
<b>sin()</b>	Sinus
<b>sqrt()</b>	Racine carrée
<b>tan()</b>	Tangente

## *L'objet navigator*

Cet objet renvoie des informations sur le type et la version du navigateur utilisé. Il est défini implicitement par le navigateur. Pour y accéder, vous utiliserez la syntaxe suivante :

navigator.Propriété

où Propriété est l'une des propriétés suivantes :

- appCodeName : Nom de code du navigateur ;
- appName : Nom du navigateur ;
- appVersion : Version du navigateur ;
- userAgent : Information "userAgent".

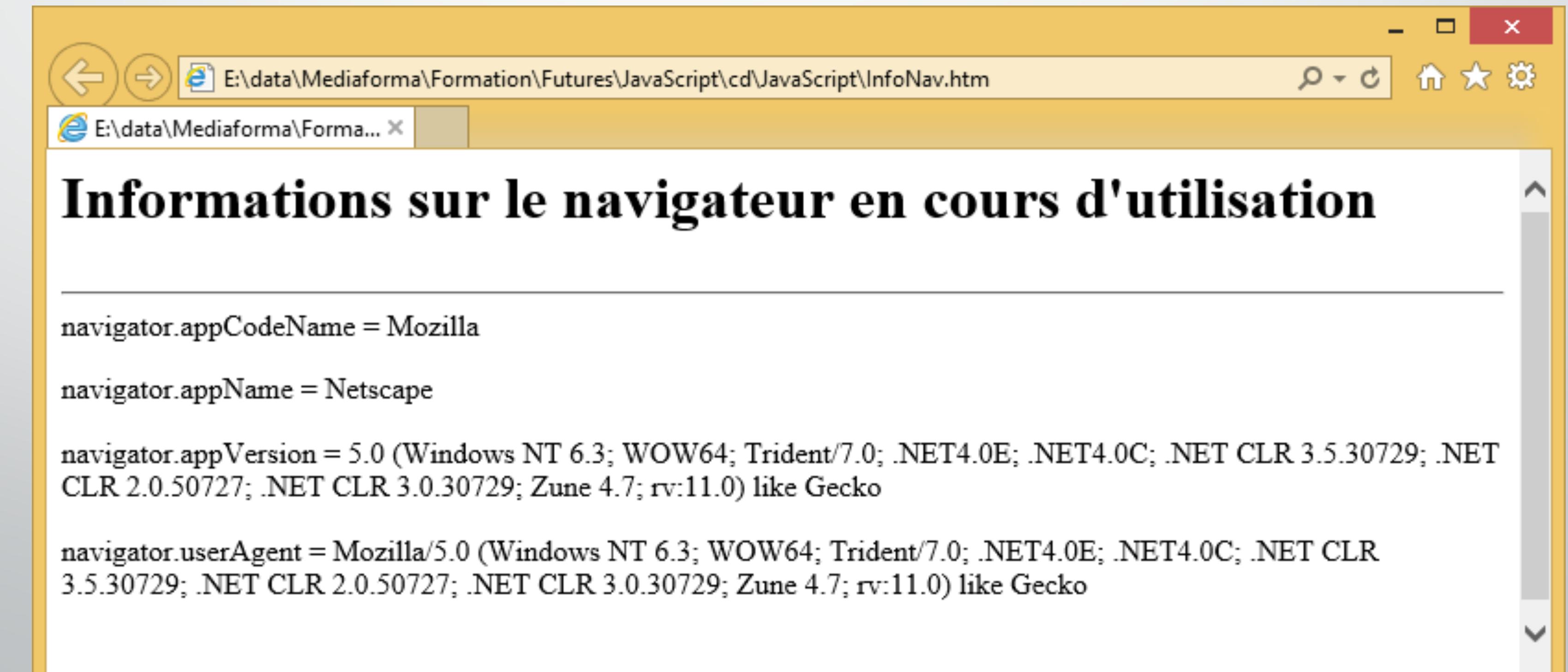
### Exercice

Ecrivez le code nécessaire pour afficher les informations sur votre navigateur Web.

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Informations sur le navigateur en cours d'utilisation</h1>
    <hr>
    <script language = JavaScript>
      document.write("navigator.appCodeName = ",navigator.appCodeName,"<br>");
      document.write("navigator.appName = ",navigator.appName,"<br>");
      document.write("navigator.appVersion = ",navigator.appVersion,"<br>");
      document.write("navigator.userAgent = ",navigator.userAgent,"<br>");
    </script>
  </body>
</html>
```

Un exemple  
d'exécution dans  
IE11 :



## *L'objet password*

Les objets password sont des champs texte. Ils sont utilisés dans un formulaire lorsque l'information tapée ne doit pas apparaître sur l'écran. Chaque caractère est remplacé par un astérisque.

La syntaxe permettant de définir un objet password est la suivante :

```
<input  
    type = "password"  
    name = "nom"  
    value = "Valeur"  
    size = Taille>
```

- Nom est le nom du champ ;
- Valeur est la valeur par défaut de l'objet password ;
- Taille représente le nombre de caractères qui peuvent être entrés sans provoquer de scrolling.

Les propriétés defaultValue, name et value permettent de connaître respectivement la valeur par défaut (paramètres value), le nom (paramètres name) et la valeur courante de l'objet.

Pour accéder à ces propriétés, vous utiliserez l'une des deux syntaxes suivantes :

Nom.Propriété

ou

Nom\_du\_formulaire.elements [index].Propriété

- Nom est le nom de l'objet password ;
- Nom du formulaire est le nom du formulaire dans lequel se trouve l'objet password ;
- index est le numéro d'ordre de l'objet password dans le formulaire ;
- Propriété est la propriété à accéder.

Vous pouvez également utiliser les méthodes suivantes :

- blur(), pour enlever le focus de l'objet ;
- focus(), pour donner le focus à l'objet ;
- select(), pour sélectionner le contenu de l'objet.

Ces trois méthodes sont accessibles à l'aide des syntaxes suivantes :

Nom.Méthode ()

ou

Nom\_du\_formulaire.elements[index].Méthode ()

- Nom est le nom de l'objet password ;
- Méthode est le nom de la méthode à utiliser (blur, focus ou select) ;
- Nom\_du\_formulaire est le nom du formulaire dans lequel se trouve l'objet password ;
- index est le numéro d'ordre de l'objet password dans le formulaire.

## Exercice :

Définissez un formulaire contenant deux zones de texte et donnez le focus à la deuxième

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Focus</h1>
    <hr>
    <form name="leForm">
      <input type="text" name="premier">
      <input type="text" name="deuxieme">
    </form>
    <script language = JavaScript>
      document.forms['leForm'].deuxieme.focus();
    </script>
  </body>
</html>
```

## *L'objet reset*

Dans un formulaire, le bouton reset permet d'affecter les valeurs par défaut à chacun des contrôles.

La syntaxe permettant de définir un bouton reset est la suivante :

```
<INPUT  
    TYPE = "reset"  
    NAME = "Nom"  
    VALUE = "Libellé"  
    [onClick = "Traitement"]>
```

- Nom est le nom du bouton ;
- Libellé est le texte qui apparaît sur le bouton ;
- Traitement est la procédure de traitement JavaScript exécutée lorsque l'utilisateur clique sur le bouton.

Les propriétés name et value donnent accès aux paramètres NAME et VALUE du marqueur INPUT. Pour y accéder, vous pouvez utiliser l'une des deux syntaxes suivantes :

Nom.Propriété

ou

Nom\_du\_formulaire.elements[index].Propriété

- Nom est le nom de l'objet reset ;
- Propriété est la propriété à accéder (name ou value) ;
- Nom du formulaire est le nom du formulaire qui contient l'objet reset ;
- index est le numéro d'ordre de l'objet reset dans le formulaire.

## *L'objet select*

Les objets select permettent de définir des listes déroulantes et des zones de liste.  
Pour définir un objet select, vous utiliserez la syntaxe suivante :

```
<select  
    name = Nom  
    [size = Hauteur]  
    [multiple]  
    [onBlur = "Traitement"]  
    [onChange = "Traitement"]  
    [onFocus = "Traitement"]>  
    <option value = "valeur1" [selected]>  
    <option value = "valeur2" [selected]>  
    ...  
    <option value = "valeurN" [selected]>  
</select>
```

Le paramètre name définit le nom de l'objet.

Le paramètre optionnel size définit la hauteur du champ. Une hauteur égale à 1 définit une liste déroulante. Une valeur supérieure définit une zone de liste.

Lorsque le paramètre multiple est spécifié, il autorise la sélection multiple dans l'objet.

Les paramètres onBlur, onChange et onFocus permettent (respectivement) de lancer une fonction lorsque l'objet select perd le focus, change de valeur et prend le focus.

Enfin, les marqueurs <option> définissent les entrées dans la liste déroulante ou la zone de liste. Le paramètre value indique la valeur renvoyée au serveur lorsqu'une option particulière est sélectionnée.

Pour accéder aux propriétés d'un objet select, vous utiliserez l'une des syntaxes suivantes :

Nom.Propriété

Nom\_du\_formulaire.elements[Index].Propriété

Nom.options[I].Propriété

Nom\_du\_formulaire.elements[Index].options[i].Propriété

- Nom est le nom de l'objet select ;
- Propriété est la propriété à accéder ;
- Nom du formulaire est le nom du formulaire dans lequel se trouve l'objet select (paramètre NAME dans le marqueur <FORM>) ;
- Index est le numéro d'ordre de l'objet dans le formulaire ;
- i est le numéro d'ordre de l'entrée à tester dans la liste déroulante/zone de liste.

Nom . Propriété

Nom \_du \_formulaire .elements [Index] .Propriété

Si vous utilisez l'une des deux premières syntaxes, vous aurez accès aux propriétés suivantes :

- length : Nombre d'options dans l'objet select ;
- name : Paramètre NAME de l'objet select ;
- options : Tableau contenant les options de l'objet select ;
- selectedIndex : numéro d'ordre de l'option sélectionnée dans l'objet select. Dans le cas d'une sélection multiple, selectedIndex contient le numéro de la première sélection.

Nom.options[I].Propriété

Nom\_du\_formulaire.elements[Index].options[I].Propriété

Si vous utilisez l'une des deux dernières syntaxes, vous aurez accès aux propriétés suivantes :

- defaultSelected : Option sélectionnée par défaut (paramètre selected) ;
- index : Numéro d'ordre de l'option sélectionnée ;
- length : Nombre d'options dans l'objet select ;
- name : Paramètre name de l'objet select ;
- selected : Permet de sélectionner une option.
- selectedIndex : Numéro d'ordre de l'option sélectionnée dans l'objet select. Dans le cas d'une sélection multiple, selectedIndex contient le numéro de la première sélection.
- text : Valeur texte affichée dans un marqueur <option> ;
- value : Valeur du paramètre value.

Supposons qu'un élément select de nom **Decision** soit défini dans un formulaire de nom **Utilisateur** :

```
<form name = "Utilisateur">
  <select name = "Decision">
    <option selected>Oui</option>
    <option>Non</option>
    <option>Peut-être</option>
  </select>
</form>
```

Pour connaître le nombre d'options dans l'objet **Decision**, vous utiliserez l'une des deux instructions suivantes :

`document.Utilisateur.Decision.length`

ou

`document.Utilisateur.elements[0].length`

La deuxième syntaxe suppose que **Decision** est le premier objet défini dans le formulaire.

Pour connaître la valeur sélectionnée par défaut dans l'objet **Decision**, vous utiliserez l'instruction suivante :

`Utilisateur.elements[0].defaultSelected`

Enfin, pour connaître la deuxième option possible dans l'objet **Decision**, vous utiliserez l'instruction suivante :

`Decision.options[1].text`

Vous pouvez également utiliser les méthodes `blur()` et `focus()` :  
`blur()` permet d'enlever le focus et `focus()` donne le focus à l'objet `select`.

Ces méthodes sont accessibles à l'aide des syntaxes suivantes :

`Nom.Méthode ()`

ou

`Nom_du_formulaire.elements[index].Méthode ()`

- `Nom` est le nom de l'objet `select` ;
- `Méthode` est le nom de la méthode à utiliser (`blur` ou `focus`) ;
- `Nom_du_formulaire` est le nom du formulaire dans lequel se trouve l'objet `select` ;
- `index` est le numéro d'ordre de l'objet `select` dans le formulaire.

## *L'objet submit*

Le bouton submit permet d'envoyer le contenu des objets d'un formulaire à l'ordinateur distant.

La syntaxe permettant de définir un bouton submit est la suivante :

```
<input  
    type = "submit"  
    name = "Nom"  
    value = "Libellé"  
    [onClick = "Traitement"]>
```

- Nom est le nom du bouton ;
- Libellé est le texte qui apparaît sur le bouton ;
- Traitement est la procédure de traitement JavaScript exécutée lorsque l'utilisateur clique sur le bouton.

Les propriétés name et value donnent accès aux paramètres name et value du marqueur input. Pour y accéder, vous pouvez utiliser l'une des deux syntaxes suivantes :

Nom.Propriété

ou

Nom\_du\_formulaire.elements[index].Propriété

- Nom est le nom de l'objet submit ;
- Propriété est la propriété à accéder (name ou value) ;
- Nom\_du\_formulaire est le nom du formulaire qui contient l'objet submit ;
- index est le numéro d'ordre de l'objet submit dans le formulaire.

## *L'objet text*

Les objets text sont utilisés pour définir des zones de texte dans un formulaire. Pour définir un objet text, vous utiliserez un marqueur <input> du type suivant :

```
<input  
    type = "text"  
    name = "Nom"  
    value = "Valeur"  
    size = "Taille1"  
    [onBlur = "Traitement1"]  
    [onChange = "Traitement2"]  
    [onFocus = "Traitement3"]  
    [onSelect = "Traitement4"]>
```

- name est le nom de l'objet text ;
- value est la valeur par défaut initialement affichée dans l'objet texte ;
- size est la taille en caractère de l'objet ;
- les TraitementN sont les fonctions JavaScript associées aux événements suivants :
  - onBlur() : Perte de focus.
  - onChange() : Changement de valeur.
  - onFocus() : Prise de focus.
  - onSelect() : Sélection du texte.

Pour accéder aux propriétés d'un objet text, vous utiliserez l'une des deux syntaxes suivantes :

Nom.Propriété

ou

Nom\_du\_formulaire.elements[Index].Propriété

- Nom est le nom de l'objet text ;
- Propriété est la propriété à accéder,
- Nom du formulaire est le nom du formulaire dans lequel se trouve l'objet text (paramètre name dans le marqueur <form>) ;
- index est le numéro d'ordre de l'objet text dans le formulaire.

Les propriétés d'un objet text sont :

- defaultValue : valeur par défaut (paramètre value) ;
- name : nom de l'objet (paramètre name) ;
- value : valeur courante de l'objet.

## *L'objet textarea*

Les objets textarea sont utilisés pour définir des zones de texte multiligne dans un formulaire. Pour définir un objet textarea, vous utiliserez un marqueur <input> du type suivant :

```
<textarea  
    type = "textarea"  
    name = "Nom"  
    rows = "NbLignes"  
    cols = "NbColonnes"  
    [onBlur = "Traitement1"]  
    [onChange = "Traitement2"]  
    [onFocus = "Traitement3"]  
    [onSelect = "Traitement4"]>  
    Texte  
</textarea>
```

- name est le nom de l'objet ;
- rows et cols définissent le nombre de lignes et de colonnes de l'objet ;
- Texte est le texte à afficher par défaut dans la zone de texte ;
- les TraitementN sont les fonctions JavaScript associées aux événements suivants :
  - onBlur : Perte de focus.
  - onChange : Changement de valeur.
  - onFocus : Prise de focus.
  - onSelect : Sélection du texte.

Pour accéder aux propriétés d'un objet textarea, vous utiliserez l'une des deux syntaxes suivantes :

Nom . Propriété

ou

Nom\_du\_formulaire.elements[Index].Propriété

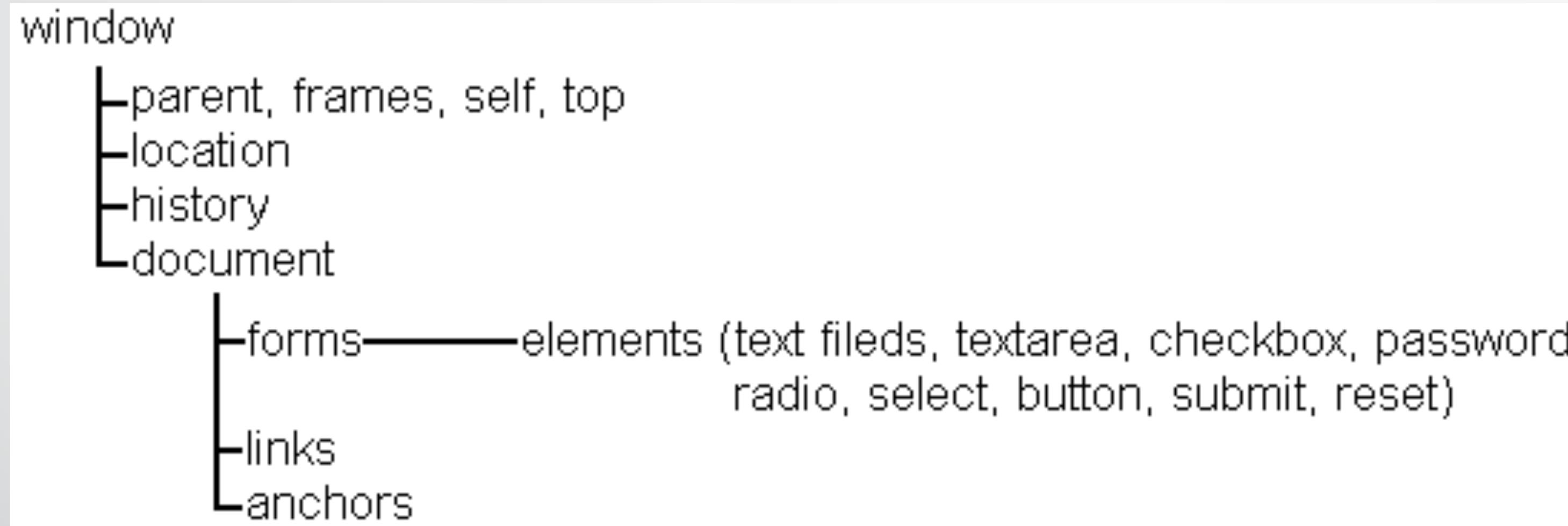
- Nom est le nom de l'objet textarea ;
- Propriété est la propriété à accéder,
- Nom\_du\_formulaire est le nom du formulaire dans lequel se trouve l'objet textarea (paramètre name dans le marqueur <form>) ;
- index est le numéro d'ordre de l'objet textarea dans le formulaire.

Les propriétés d'un objet textarea sont :

- name : nom de l'objet (paramètre name).
- value : valeur courante de l'objet.

# *L'objet window*

L'objet window se trouve au sommet de la hiérarchie :



Vous vous en doutez certainement, les propriétés relatives aux objets window sont très nombreuses. Pour y accéder, vous utiliserez l'une des syntaxes suivantes :

window.Propriété  
self.Propriété  
top.Propriété  
parent.Propriété  
variable.Propriété

- où window et self font référence à la fenêtre courante ;
- top est la fenêtre hiérarchiquement supérieure (dans le cas où plusieurs fenêtres sont affichées) ;
- parent fait référence à un document dans lequel ont été définies plusieurs cadres ;
- variable représente un nom de variable utilisé pour définir une nouvelle fenêtre.

Voici la liste des diverses propriétés accessibles :

Propriété	Fonction
<code>defaultStatus</code>	Message affiché par défaut dans la barre d'état
<code>length</code>	Nombre de cadres dans la fenêtre parent
<code>name</code>	Nom de la fenêtre, tel qu'il est défini dans le deuxième paramètre de la méthode <code>open()</code>
<code>parent</code>	Nom du document dans lequel ont été définies plusieurs cadres
<code>self</code>	Nom de la fenêtre courante
<code>status</code>	Message affiché dans la barre d'état
<code>top</code>	Nom de la fenêtre hiérarchiquement supérieure (dans le cas où plusieurs fenêtres sont affichées)
<code>window</code>	Nom de la fenêtre courante

Pour manipuler les objets `window`, vous utiliserez les méthodes suivantes :

Méthode	Fonction
<code>alert()</code>	Affichage d'une boîte de message contenant un message texte et un bouton OK
<code>close()</code>	Fermeture de la fenêtre spécifiée
<code>confirm()</code>	Affiche une boîte de dialogue de confirmation contenant un message texte, un bouton Annuler et un bouton OK
<code>open()</code>	Ouverture d'une fenêtre
<code>prompt()</code>	Saisie d'une donnée dans une boîte de dialogue contenant un bouton Annuler et un bouton OK
<code>setTimeout()</code>	Exécution d'une expression après un délai paramétrable
<code>clearTimeout()</code>	Suppression d'un délai initialisé avec <code>setTimeout()</code>

## Exercice

Entraînez-vous à :

1. Afficher une boîte de message avec la méthode alert()
2. Afficher une boîte de confirmation avec la méthode confirm()

```
<script language="JavaScript">
    alert('Ceci est une boîte de message affichée par JavaScript');
</script>
```

```
<script language="JavaScript">
function Confirmation() {
    if (confirm("Etes-vous sûr de vouloir fermer le navigateur ?"))
        window.close();
}
</script>
...
<input type="button" value="Fermer" onClick="Confirmation();">
```

# **Ouverture d'une fenêtre - La méthode open()**

Pour ouvrir une nouvelle fenêtre, vous utiliserez la méthode open() :

```
Nom = window.open("URL", "NomFenetre", "Caractéristiques")
```

- Nom est le nom de la nouvelle fenêtre ;
- URL est l'URL de la nouvelle fenêtre ;
- NomFenetre est le nom qui sera utilisé dans le paramètre target d'un marqueur <form> ou <a> ;
- Caractéristiques fixe une ou plusieurs caractéristiques de la fenêtre :
- **⚠️ Elles ne sont pas toutes compatibles avec tous les navigateurs (voir sur MDN)**

Paramètre	Fonction
<code>toolbar=[yes no]</code>	Affiche/cache la barre d'outils dans la fenêtre
<code>location=[yes no]</code>	Affiche/cache la barre location dans la fenêtre
<code>directories=[yes no]</code>	Affiche/cache la barre contenant les boutons What's new, What's cool et Handbook
<code>status=[yes no]</code>	Affiche/cache la barre d'état
<code>menubar=[yes no]</code>	Affiche/cache la barre de menus
<code>scrollbars=[yes no]</code>	Affiche/cache les barres de défilement
<code>resizable=[yes no]</code>	Permet/interdit à l'utilisateur de redimensionner la fenêtre
<code>width=Largeur</code>	Définit la largeur de la fenêtre en pixels
<code>height=Hauteur</code>	Définit la hauteur de la fenêtre en pixels

## Exercice

Ouvrez une fenêtre de 400x400 pixels sur le site [www.google.fr](http://www.google.fr) suite au clic sur un bouton

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ouverture d'une autre fenêtre</title>
  </head>
  <body>
    Cliquez sur ce bouton pour ouvrir une nouvelle fenêtre sur Google :
    <button onclick="window.open('http://www.google.fr','google','scrollbars=yes,
      status=yes,width=400,height=400')>Google</button>
  </body>
</html>
```

## Exercice

Définissez un document contenant deux boutons.

Le premier bouton ouvre une fenêtre secondaire de 200x100 pixels et y écrit le texte suivant : "Ceci est la fenêtre secondaire".

Le deuxième bouton ferme la fenêtre secondaire.

## Solution

```
<!DOCTYPE html>
<html>
<head>
</head>
<script>
    var maFenetre;
    function Ouvrir() {
        maFenetre = window.open(' ', 'maFenetre', 'width=200, height=100');
        maFenetre.document.write('<p>Ceci est la fenêtre secondaire</p>');
    }
    function Fermer() {
        maFenetre.close();
    }
</script>

<body>
    <button onclick="Ouvrir()">Ouvrir</button>
    <button onclick="Fermer()">Fermer</button>
</body>
</html>
```

## *Saisie d'une donnée - La méthode prompt()*

La méthode prompt() affiche une boîte de dialogue dans laquelle l'utilisateur peut saisir une donnée numérique ou texte. Une valeur par défaut peut être affichée. Un appui sur le bouton Cancel renvoie la valeur null et un appui sur le bouton OK renvoie la valeur entrée.

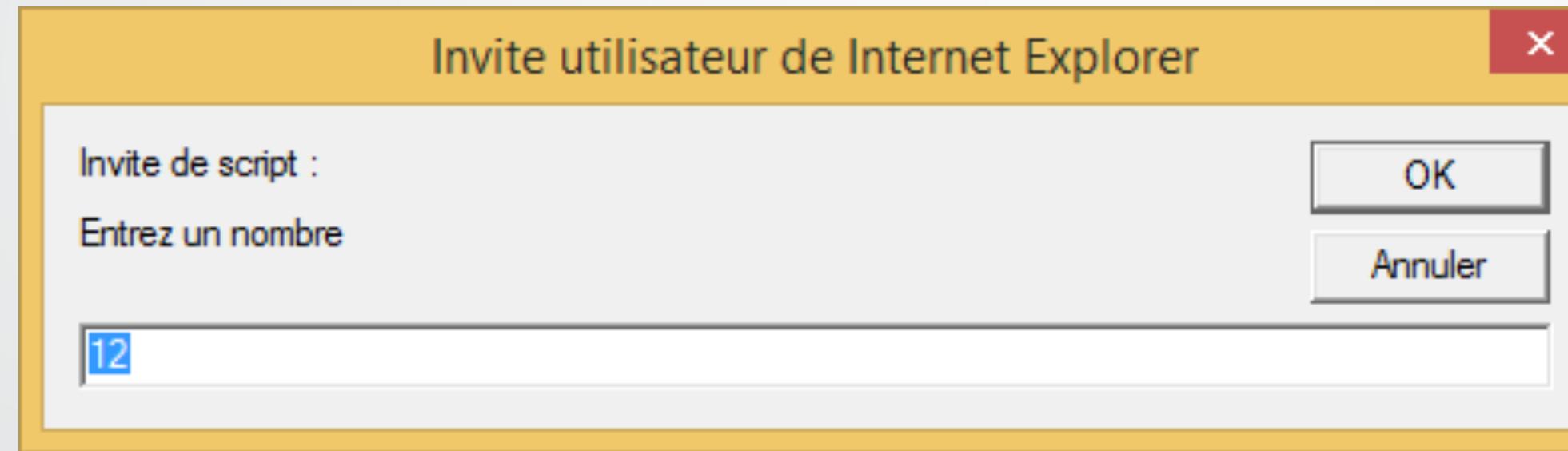
La syntaxe de la méthode prompt() est la suivante :

```
Retour=prompt ("Texte" [, Défaut]);
```

- Texte est le texte affiché dans la boîte de dialogue ;
- Défaut est la valeur par défaut ;
- Retour est la valeur renvoyée après appui sur le bouton OK ou Cancel.

## Exercice

Ecrivez le code nécessaire pour afficher la boîte de saisie suivante à l'ouverture de la page :



## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Boîte de saisie</title>
  </head>
  <body>
    <script>
      var valeur = prompt("Entrez un nombre",12);
      document.write("Vous avez entré la valeur ", valeur);
    </script>
  </body>
</html>
```

## *La classe Date*

Les objets issus de la classe Date permettent de travailler avec les dates (jours, mois, années) et les heures (heures, minutes, secondes).

Pour définir un objet Date, vous pouvez utiliser l'une des quatre syntaxes suivantes :

```
var d = new Date();  
d = new Date("mois, jour, année heures:minutes:secondes");  
d = new Date(année, jour, mois);  
d = new Date(année, jour, mois, heures, minutes, secondes);
```

où d représente le nom d'un nouvel objet ou d'une propriété d'un objet existant, et mois, jour, année, heures, minutes et secondes les composantes de l'objet Date. Dans la deuxième syntaxe, ils sont exprimés sous une forme chaîne. Dans la troisième et la quatrième syntaxe, ils sont exprimés sous une forme entière.

De nombreuses méthodes sont attachées aux objets Date :

getDate(), getDay(), getMinutes(), getMonth(), getSeconds(), getTime(),  
getTimeZoneoffset(), getYear(), getFullYear(), parse(), setDate(), setHours(),  
setMinutes(), setMonth(), setSeconds(), setTime(), setYear(), toGMTString(),  
toLocaleString(), UTC().

Exercice

Affichez la date et l'heure système.

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Date et heure système</title>
    <script language="JavaScript">
      function WhatDayIsIt() {
        Aujourd'hui = new Date();
        Jour = Aujourd'hui.getDate();
        Mois = Aujourd'hui.getMonth()+1;
        Année = Aujourd'hui.getFullYear();
        document.write("Aujourd'hui, nous sommes le " + Jour + " / " + Mois + " / " + Année + ".<br>");
      }

      function WhatTimeIsIt() {
        Maintenant = new Date ();
        Heures = Maintenant.getHours();
        Minutes = Maintenant.getMinutes();
        Secondes = Maintenant.getSeconds();
        document.write ("Il est "+ Heures + " heures " + Minutes + " min " + Secondes + " sec.");
      }
    </script>
  </head>
  <body>
    <script>
      WhatDayIsIt();
      WhatTimeIsIt();
    </script>
  </body>
</html>
```

Une horloge JavaScript qui s'affiche dans un `<input>` de type text.

Ici, on utilise la fonction `setInterval()` pour exécuter la fonction `affiche()` à intervalles réguliers de 1000 ms.

La fonction `commence()` autorise l'affichage de l'horloge et la fonction `arrete()` l'interdit.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Date</title>
<script>
    var startStop = false;
    function affiche() {
        if (startStop) {
            var d = new Date();
            f.hms.value=d.toLocaleString();
        }
    }
    function commence() {
        startStop = true;
    }
    function arrete() {
        startStop = false;
    }
</script>
</head>
<body>
<form name="f">
    <input type="text" name="hms">
    <input type="button" value="Start" onclick="commence();">
    <input type="button" value="Stop" onclick="arrete();">
</form>
<script>
    setInterval(affiche, 1000);
</script>
</body>
</html>
```

## ***La classe String***

Les chaînes de caractères sont des objets de classe String. Lorsque vous utilisez une instruction du type ci-après, vous définissez un objet String de nom "chaine" :

```
chaine = "Créer ses pages Web - Comment faire"
```

Pour connaître la longueur d'un objet String, vous utiliserez la propriété length.

En reprenant l'exemple précédent, l'instruction :

```
chaine.length
```

renvoie la valeur 35, car la chaîne comporte 35 caractères.

De nombreuses méthodes sont associées aux chaînes de caractères. Le tableau ci-après en dresse la liste :

Méthode	Utilisation
<code>anchor()</code>	Définition d'un signet
<code>big()</code>	Caractères de grande taille (identique au marqueur <code>&lt;BIG&gt;</code> )
<code>blink()</code>	Affecte l'attribut clignotant au texte (identique au marqueur <code>&lt;BLINK&gt;</code> )
<code>bold()</code>	Affecte l'attribut gras au texte (identique au marqueur <code>&lt;B&gt;</code> )
<code>charAt()</code>	Renvoie le caractère qui se trouve à la position indiquée
<code>fixed()</code>	Police non proportionnelle (identique au marqueur <code>&lt;TT&gt;</code> )
<code>fontcolor()</code>	Définit la couleur des caractères (identique au paramètre COLOR du marqueur <code>&lt;FONT&gt;</code> )
<code>fontsize()</code>	Définit la taille des caractères (identique au marqueur <code>&lt;FONTSIZE&gt;</code> )
<code>indexOf()</code>	Première position d'une sous-chaîne dans une chaîne
<code>italics()</code>	Affecte l'attribut italique au texte
<code>lastIndexOf()</code>	Dernière position d'une sous-chaîne dans une chaîne
<code>link()</code>	Définit un lien hypertexte
<code>small()</code>	Caractères de petite taille (identique au marqueur <code>&lt;SMALL&gt;</code> )
<code>strike()</code>	Affecte l'attribut barré au texte (identique au marqueur <code>&lt;STRIKE&gt;</code> )
<code>sub()</code>	Caractères en indice (identique au marqueur <code>&lt;SUB&gt;</code> )
<code>substring()</code>	Extraction d'une sous-chaîne
<code>sup()</code>	Caractères en exposant (identique au marqueur <code>&lt;SUP&gt;</code> )
<code>toLowerCase()</code>	Conversion en minuscules
<code>toUpperCase()</code>	Conversion en majuscules

## **Exemple 1 - Caractères de grande taille et de petite taille : méthodes big() et small().**

Ces instructions affichent le texte de la variable chaîne en caractères de grande taille, puis de petite taille :

```
chaine = "Créer ses pages Web - Comment faire";
document.write(chaine.big());
document.write(chaine.small());
```

## **Exemple 2 - Attributs gras, italique et barré : méthodes blink(), bold(), italics() et strike().**

Ces instructions affectent les attributs gras et italique aux mots "Créer", et "ses pages".

```
ch1 = "Créer ";
ch2 = "ses pages";
document.write(ch1.bold());
document.write(ch2.italics());
```

## **Exemple 3 - Extraction de caractères : méthode charAt().**

Ces instructions extraient les caractères en position 3 et 11 dans la chaîne ch :

```
ch = "Créer ses pages Web - Comment faire<br>";
document.write(ch, "<br>");
document.write("Le caractère en position 3 est un ", ch.charAt(3), "<br>");
document.write("et le caractère en position 11 est un ", ch.charAt(11));
```

## **Exemple 4 - Caractères non proportionnels : méthode fixed().**

Ces instructions affichent le texte de la chaîne ch en utilisant une police conventionnelle, puis une police non proportionnelle :

```
ch = "Créer ses pages Web - Comment faire<BR>";
document.write(ch);
document.write(ch.fixed());
```

## **Exemple 5 - Modification de la taille des caractères : méthode fontsize().**

Ces instructions JavaScript affichent un texte en taille 4 et en taille 7 :

```
ch = "Créer ses pages Web - Comment faire<BR>";
document.write(ch.fontsize(4));
document.write(ch.fontsize(7));
```

## **Exemple 6 - Position d'une sous-chaîne dans une chaîne : méthodes indexOf() et lastIndexOf().**

Ces instructions affichent la position du premier et du dernier "e" dans la chaîne "Créer ses pages Web - Comment faire" :

```
ch = "Créer ses pages Web - Comment faire<BR>";
document.write(ch);
document.write("Le premier e occupe la position ",ch.indexOf("e"),"<BR>");
document.write("Le dernier e occupe la position ",ch.lastIndexOf("e"));
```

## **Exemple 7 - Définition d'un lien hypertexte : méthode link().**

Les instructions JavaScript ci-après définissent deux liens hypertextes :

```
ch1 = "Recherche Google";
ch2 = "Recherche Bing";
URL1 = "http://www.google.fr";
URL2 = "http://www.bing.com"
document.write("Sélectionnez l'un de ces sites :<BR>");
document.write(ch1.link(URL1), "<BR>");
document.write(ch2.link(URL2), "<BR>");
```

## **Exemple 8 - Caractères en indice et en exposant : méthodes sub() et sup().**

Ces instructions JavaScript affichent une formule mathématique qui utilise un indice et un exposant :

```
ch1 = "(x)"
ch2 = "2"
document.write("f", ch1.sub(), " = a x", ch2.sup(), " + b x + c");
```

## **Exemple 9 - Extraction d'une sous-chaîne : méthode substring().**

Les instructions suivantes affichent les caractères "ivr" qui occupent les positions 4 à 7 dans la chaîne ch :

```
ch = "Le livre d'or Java<BR>"
document.write("ch.substring(4,7) = ", ch.substring(4,7));
```

Remarquez que le premier caractère a pour index 0 (et non 1).

## **Exemple 10 - Caractères majuscules et minuscules : méthodes toLowerCase() et toUpperCase().**

Ces instructions affichent la chaîne originale, sa conversion en minuscules, puis sa conversion en majuscules :

```
ch = "Créer ses pages Web – Comment faire<BR>"  
document.write(ch);  
document.write(ch.toLowerCase());  
document.write(ch.toUpperCase());
```

Voici le résultat obtenu :

```
Créer ses pages Web – Comment faire  
créer ses pages Web – comment faire  
CREER SES PAGES WEB – COMMENT FAIRE
```



# Validation de données en HTML et en JavaScript

# Validation de données

Lorsqu'un champ de saisie doit obligatoirement être rempli, il suffit de lui affecter l'attribut required. Dans ce formulaire, le champ de saisie Nom est obligatoire. Si l'utilisateur clique sur le bouton Submit sans l'avoir renseigné, un message d'erreur est généré et le formulaire n'est pas envoyé.

```
<!DOCTYPE html>                                         Code062.htm
<html>
  <head>
    <meta charset="utf-8">
    <title>Champ de saisie obligatoire</title>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label><input type="text" name="Nom" required><br><br><br><br>
      <input type="submit" value="Valider"></p>
    </form>
  </body>
</html>
```

Essayez ce code

# Validation de données

Le champ d'action des attributs required est bien plus étendu, puisqu'il permet également de valider des données complexes (telles que des adresses e-mail ou des URL). À titre d'exemple, le formulaire ci-après teste la validité des champs email et url. Lorsqu'on clique sur le bouton Submit, un message d'erreur s'affiche si un de ces deux champs n'est pas conforme.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Validation e-mail et url</title>
  </head>
  <body>
    <form id="validation">
      <label>Nom</label> <input type="text" name="Nom"><br>
      <label>e-mail</label> <input type="email" name="email" required><br>
      <label>Adresse URL</label> <input type="url" name="url" required><br>
      <label>Commentaire</label> <textarea name="comment"></textarea><br>
      <input type=submit value="Valider"></p>
    </form>
  </body>
</html>
```

Code063.htm

Essayez ce code

# Validation de données

Pour faciliter la reconnaissance des champs obligatoires, il est possible d'utiliser quelques lignes de CSS. Par exemple, cette ligne affecte un arrière-plan de couleur jaune aux champs `<input>` dont l'attribut `required` est spécifié :

```
input:required { background:yellow }
```

# Validation de données

JavaScript permet de contrôler la validité des champs d'un formulaire lors du clic sur le bouton Submit. Le formulaire n'est pas communiqué au serveur si les données ne sont pas valides.

Examinez et testez ce code :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Texte</title>
    <script>
      function validation() {
        var leNom = document.form1.nom.value;
        if (leNom == "") {
          alert("Vous devez remplir le champ Nom avant d'envoyer le formulaire");
          return false; // Le formulaire n'est pas validé
        }else{
          return true; // Le formulaire est validé
        }
      }
    </script>
  </head>
  <body>
    <form name="form1" action="traitement.php" method="post" onsubmit="return validation();">
      Nom <input type="text" name="nom">
      <input type="submit" value="Envoyer">
    </form>
  </body>
</html>
```

## Accès aux éléments du DOM avec la fonction document.getElementById()

Les éléments insérés dans le DOM peuvent avoir un attribut id (identifiant) qui les identifie de façon unique. Ces éléments peuvent alors être accédés en lecture et en écriture en JavaScript avec la fonction document.getElementById().

Exemple :

```
<div id="laDiv"></div>
<script>
    document.getElementById('laDiv').innerHTML = 'test';
</script>
```

Supposons qu'un élément d'id unElement soit défini :

```
<span id="unElement">Un simple texte</span>
```

Si vous définissez cette variable :

```
var el = document.getElementById('unElement');
```

Vous pouvez :

- connaître le contenu de cet élément avec la propriété innerHTML :

```
document.write(el.innerHTML);
```

- modifier le contenu de cet élément avec la propriété innerHTML :

```
el.innerHTML = 'Un autre contenu';
```

- connaître ou modifier la couleur de cet élément avec la propriété style.color :

```
el.style.color='red';
```

- connaître ou modifier la couleur de cet élément avec la propriété style.color :

```
el.style.backgroundColor='yellow';
```

- cacher ou afficher l'élément avec la propriété display :

```
el.style.display='none';
```

```
el.style.display='inline';
```

Et beaucoup d'autres choses encore...

## Affichage ou dissimulation d'un <div> en fonction de boutons radio Mr et Mme :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Mr Mme</title>
    <script>
        function mSelectionne() {
            var leDiv = document.getElementById('d');
            leDiv.style.display = 'none';
        }
        function mmeSelectionnee() {
            var leDiv = document.getElementById('d');
            leDiv.style.display = 'block';
        }
    </script>
</head>
<body>
    <form name="f">
        <input type="radio" value="M" name="MMme" onclick="mSelectionne();"> Mr
        <input type="radio" value="Mme" name="MMme" checked onclick="mmeSelectionnee();"> Mme<br>
        <div id="d">
            <label>Nom de jeune fille</label>
            <input type="text" name="njkf">
        </div>
    </form>
</body>
</html>
```

## Exercice

Dans un formulaire contenant plusieurs zones de texte, mettez en place les instructions nécessaires pour colorer l'arrière-plan de l'élément qui a le focus (c'est-à-dire celui dans lequel se fait la saisie).

# Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Coloration sur focus en JS</title>
    <script>
      function vert(el) {
        el.style.backgroundColor = 'green';
      }
      function blanc(el) {
        el.style.backgroundColor = 'white';
      }
    </script>
  </head>
  <body>
    <form>
      <input type="text" onfocus="vert(this);" onblur="blanc(this);">
      <input type="text" onfocus="vert(this);" onblur="blanc(this);">
      <input type="text" onfocus="vert(this);" onblur="blanc(this);">
      <input type="text" onfocus="vert(this);" onblur="blanc(this);">
      <input type="text" onfocus="vert(this);" onblur="blanc(this);">
    </form>
  </body>
</html>
```

## Exercice :

Définissez une page composée de deux liens et d'une balise <div> qui contient du texte.  
Lorsque l'utilisateur clique sur le premier lien, déplacez le <div> à 100 pixels du bord gauche.  
Lorsqu'il clique sur le deuxième lien, déplacez le <div> à 500 pixels du bord gauche.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>getElementById() dans un formulaire</title>
    <script>
      function gauche() {document.getElementById('laDiv').style.left = '100px';}
      function droite() {document.getElementById('laDiv').style.left = '500px';}
    </script>
    <style>
      div {
        background-color: yellow; border: dotted 1px #666;
        width: 150px; height: 150px; left: 200px; display: block; position: absolute;
      }
    </style>
  </head>
  <body>
    <a href="javascript:gauche();">Vers la gauche</a><br>
    <a href="javascript:droite();">Vers la droite</a><br>
    <div id="laDiv">Un peu de texte</div>
  </body>
</html>
```

Vous pouvez également modifier un attribut d'une balise HTML avec la fonction **setAttribute()**.

Exercice :

En partant de ce code, ajoutez un bouton de commande qui modifie l'image affichée.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test de setAttribute()</title>
  </head>
  <body>
    
  </body>
</html>
```

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Test de frames</title>
    <script>
      function changeImage() {
        var im = document.getElementById('image');
        im.setAttribute('src', 'chat2.jpg');
      }
    </script>
  </head>
  <body>
    
    <button onclick="changeImage()">Changer l'image</button>
  </body>
</html>
```

Vous pouvez également accéder à plusieurs éléments avec la méthode **getElementsByName()**. Voici un exemple :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>getElementsByName</title>
</head>
<body>
    <p>un texte</p>
    <p>un autre texte</p>
    <p>un dernier texte</p>
    <script>
        var el = document.getElementsByTagName('p');
        for (var i=0; i<el.length; i++) {
            el[i].innerHTML='Un nouveau texte';
        }
    </script>
</body>
</html>
```

Enfin, vous pouvez accéder à plusieurs éléments via leur classes avec la fonction **getElementsByClassName()**. Voici un exemple :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>getElementsByClassName</title>
</head>
<body>
    <p class="rouge">un texte</p>
    <p>un autre texte</p>
    <p class="rouge">un dernier texte</p>
    <script>
        var el = document.getElementsByClassName('rouge');
        for (var i=0; i<el.length; i++) {
            el[i].style.color='red';
        }
    </script>
</body>
</html>
```

# Version moderne du gestionnaire d'événements

Si vous utilisez un navigateur récent, vous pouvez mettre en place des gestionnaires d'événements en utilisant une syntaxe JavaScript plus moderne. Pour cela, vous passerez par la fonction `addEventListener()` :

```
élément.addEventListener(type, listener);
```

Où :

- **élément** est l'élément sur lequel s'applique le gestionnaire ;
- **type** représente le type de l'événement à capturer ;
- **listener** est la fonction JavaScript à exécuter quand l'événement se produit.

Essayez ce code :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestionnaire d'événement moderne</title>
  </head>
  <body>
    <button id="bouton">Cliquez ici</button>
    <script>
      var element = document.getElementById('bouton');
      var maFonction = function() {
        alert('Vous avez cliqué');
      };
      element.addEventListener('click', maFonction);
    </script>
  </body>
</html>
```

Pour supprimer un gestionnaire d'événements, vous utiliserez la fonction `removeEventListener()` :

```
élément.removeEventListener(type, listener);
```

Où :

- **élément** est l'élément sur lequel s'applique le gestionnaire ;
- **type** représente le type de l'événement à capturer ;
- **listener** est le gestionnaire d'événements rattaché à cet événement.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestionnaire d'événement moderne</title>
  </head>
  <body>
    <button id="bouton">Cliquez ici</button>
    <button onclick="ajoute();">Ajouter le gestionnaire</button>
    <button onclick="supprime();">Supprimer le gestionnaire</button>
    <script>
      var element = document.getElementById('bouton');
      var maFonction = function() {
        alert('Vous avez cliqué');
      };
      function ajoute() {
        element.addEventListener('click', maFonction);
      }
      function supprime() {
        element.removeEventListener('click', maFonction);
      }
    </script>
  </body>
</html>
```

Essayez ce code :

Comment auriez-vous fait pour ajouter ou supprimer un gestionnaire d'événements sur le premier bouton lorsque l'utilisateur clique sur les deux autres boutons ? Et cela, sans utiliser `addEventListener()` ni `removeEventListener()`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestionnaire d'événement moderne</title>
  </head>
  <body>
    <button id="bouton">Cliquez ici</button>
    <button onclick="ajoute();">Ajouter le gestionnaire</button>
    <button onclick="supprime();">Supprimer le gestionnaire</button>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Gestionnaire d'événement moderne</title>
    </head>
    <body>
        <button id="bouton">Cliquez ici</button>
        <button onclick="ajoute();">Ajouter le gestionnaire</button>
        <button onclick="supprime();">Supprimer le gestionnaire</button>
        <script>
            var element = document.getElementById('bouton');

            function ajoute() {
                element.onclick = function() {
                    alert('Vous avez cliqué');
                };
            }
            function supprime() {
                element.onclick = function() {};
            }
        </script>
    </body>
</html>
```

Solution :

# Gestion de clic unique

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Clic unique</title>
    <script>
      function traitement () {
        var leBouton = document.getElementById('b');
        leBouton.removeEventListener('click', traitement);
        leBouton.setAttribute('disabled',true);
        alert('Merci d\'avoir cliqué !');
      }
    </script>
  </head>
  <body>
    <button id="b">Cliquez ici</button>
    <script>
      var leBouton = document.getElementById('b');
      leBouton.addEventListener('click', traitement);
    </script>
  </body>
</html>
```

# Cookies

Un cookie est un petit fichier texte stocké côté client par le navigateur.

Il contient plusieurs données :

1. Une paire nom / valeur qui contient les données du cookie.
2. Une date d'expiration au-delà de laquelle il n'est plus valide.
3. Un domaine et une arborescence qui indiquent quel répertoire de quel serveur y aura accès.

Les cookies peuvent être créés, lus et supprimés par JavaScript.  
Ils sont accessibles à travers la propriété **document.cookie**.

Voici un exemple de création de cookie :

```
document.cookie = 'cookie1=testcookie; expires=Sat, 27 Jun 2015  
00:00:00 UTC; path=/';
```

Cette chaîne comporte :

1. Une paire nom / valeur : 'cookie1=testcookie'
2. Un point-virgule et un espace
3. La date d'expiration : expires=Sat, 27 Jun 2015 00:00:00 UTC
4. Un point-virgule et un espace
5. Le domaine : path=/

Si vous définissez un deuxième cookie :

```
document.cookie = 'cookie2=testcookie2; expires=Sat, 20 Jun 2015  
08:00:00 UTC; path=/';
```

Le premier cookie n'est pas effacé.

Pour lire les cookies, vous utiliserez cette syntaxe :

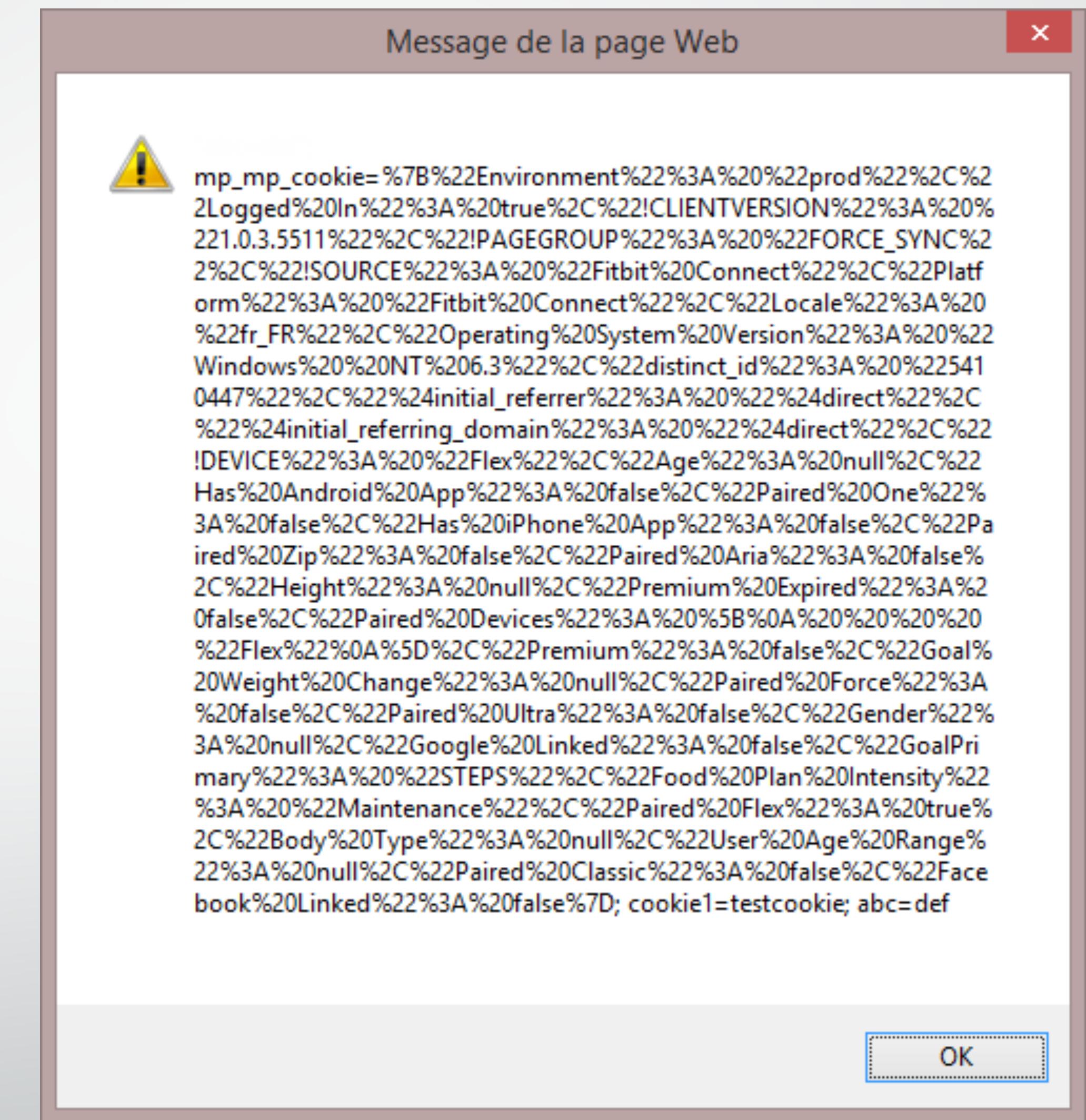
```
var x = document.cookie;
```

Après l'exécution de cette instruction, x contiendra la liste des paires/valeurs de tous les cookies stockés par le navigateur, séparés entre eux par des ";"

**Exercice :**

Définissez deux cookies et affichez les cookies du navigateur dans un <textarea>

Voici ce que vous devez obtenir :



## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion des cookies</title>
    <script>
      function ajouter(form) {
        var nom = form.nom.value;
        var valeur = form.valeur.value;
        var leCookie = nom + '=' + valeur + '; expires=Sat, 27 Jun 2015 00:00:00 UTC; path=/';
        document.cookie = leCookie;
      }
      function lire() {
        alert(document.cookie);
      }
    </script>
  </head>
  <body>
    <form>
      <p><label>Nom du cookie</label> <input type="text" name="nom"></p>
      <p><label>Valeur du cookie</label> <input type="text" name="valeur"></p>
      <button onclick="ajouter(this.form);">Ajouter le cookie</button>
      <button onclick="lire();">Lire les cookies</button><br>
    </form>
  </body>
</html>
```

Pour modifier un cookie, vous utiliserez la même technique que pour le créer.

Par exemple, supposons que vous ayez défini ce cookie :

```
document.cookie = 'cookie1=testcookie; expires=Sat, 27  
Jun 2015 00:00:00 UTC; path=/';
```

Et que vous vouliez changer :

- La valeur **testcookie** en **nouveauTestCookie**
- La date d'expiration au **Dimanche 28 Juin**

Vous écrirez quelque chose comme ceci :

```
document.cookie = 'cookie1=nouveauTestCookie;  
expires=Sun, 28 Jun 2015 00:00:00 UTC; path=/';
```

Enfin, pour supprimer un cookie, affectez une date antérieure à aujourd'hui au paramètre expires. Vous pouvez ou non affecter une valeur au cookie. Ici par exemple, nous ne lui affectons aucune valeur :

```
document.cookie = 'cookie1=; expires=Sat, 16 May 2015  
00:00:00 UTC; path=/';
```

## Exercice

Entraînez-vous à modifier et à supprimer des cookies en complétant le programme précédent.

# Stockage local – Local Storage

Avec HTML5, le stockage de données en local est désormais un jeu d'enfant. D'ici à prédire la disparition des cookies, il n'y a qu'un pas...

Voyons ce que HTML5 propose. En fait, le code à utiliser est du JavaScript. Il repose sur l'utilisation des fonctions associées à l'objet localStorage :

Fonction	Signification
<u>localStorage.getItem('nom')</u>	Retourne la valeur correspondant au nom spécifié dans l'argument
<u>localStorage.setItem('nom', 'valeur')</u>	Stocke la valeur spécifiée sous le nom spécifié dans l'argument
<u>localStorage.key(position)</u>	Retourne le nom de l'élément stocké à la position spécifiée (l'indice 0 correspond à la première position de sauvegarde)
<u>localStorage.length</u>	Retourne le nombre de données mémorisées
<u>localStorage.clear()</u>	Efface toutes les données sauvegardées
<u>localStorage.removeItem('nom')</u>	Efface la donnée sauvegardée sous le nom spécifié dans l'argument

# Exercice

Définissez un formulaire comparable à la copie d'écran ci-contre et donnez vie aux quatre boutons en utilisant les fonctions liées au stockage local.

The screenshot shows a web browser window titled "Stockage local HTML5". The address bar displays the URL "file:///C:/Data/Pearson/HTML%205/code/storage.htm?name=a&data=". The main content area has two sections: "Entrez la donnée à stocker" and "Données stockées".

**Entrez la donnée à stocker**

Nom:

Donnée:

```
Une autre donnée texte  
stockée sur  
plusieurs lignes
```

Buttons: Lit, Enregistre, Efface, Efface tout

**Données stockées**

Nom	Valeur
deuxième	Une autre donnée texte stockée sur plusieurs lignes
premier	Cette donnée est stockée sous le nom "premier"

```

<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8" />
<title>Stockage local HTML5</title>

<style>
div
{
    border-width: 1px;
    border-style: dotted;
}
</style>

<script>
    function Affiche()
    {
        var key = "";
        var NomValeur = "<tr><th>Nom</th><th>Valeur</th></tr>\n";
        var i=0;
        for (i=0; i<localStorage.length-1; i++)
        {
            key = localStorage.key(i);
            NomValeur += "<tr><td>" +key+"</td>\n<td>" +localStorage.getItem(key) +"</td></tr>\n";
        }
        document.getElementById('NomValeur').innerHTML = NomValeur;
    }

    function Enregistre()
    {
        var nom = document.forms.editor.name.value;
        var valeur = document.forms.editor.data.value;
        localStorage.setItem(nom, valeur);
        Affiche();
    }

    function Lit()
    {
        var nom = document.forms.editor.name.value;
        document.forms.editor.data.value = localStorage.getItem(nom);
        Affiche();
    }

```

Testez ce code  
dans Google Chrome

```

function Efface()
{
    var nom = document.forms.editor.name.value;
    document.forms.editor.data.value = localStorage.removeItem(nom);
    Affiche();
}

function EffaceTout()
{
    localStorage.clear();
    Affiche();
}

</script>
</head>

<body onload="Affiche()">
<h1>Stockage local HTML5</h1>
<form name="editor">
    <div id="donnees">
        <h2>Entrez la donnée à stocker</h2>
        <p><label>Nom <input name="name"></label></p>
        <p><label>Donnée <textarea name="data" cols="41" rows="10"></textarea></label></p>
        <p>
            <input type="button" value="Lit" onclick="Lit();">
            <input type="button" value="Enregistre" onclick="Enregistre();">
            <input type="button" value="Efface" onclick="Efface();">
            <input type="button" value="Efface tout" onclick="EffaceTout();">
        </p>
    </div>
    <br><br>
    <div id="valeurs">
        <h2>Données stockées</h2>
        <table id="NomValeur" border="1"></table>
    </div>
</form>
</body>
</html>

```

# Stockage local – Session Storage

Vous savez maintenant utiliser l'objet **localStorage**. Une alternative s'offre à vous avec l'objet **sessionStorage**.

Les propriétés et méthodes de l'objet **sessionStorage** sont les mêmes que celles de l'objet **localStorage**. Ce qui change, c'est la durée de vie des données stockées :

- avec `localStorage`, le stockage est permanent, jusqu'à ce que les données soient effacées avec la méthode `clear()` ou `removeItem()`.
- avec `sessionStorage`, le stockage dure le temps de la session de navigation, c'est-à-dire tant que la page est ouverte. Les données stockées sont automatiquement supprimées à la fermeture de la page.

# Gestion événementielle

Avec HTML5, de nombreux nouveaux types d'événements font leur apparition. Ils concernent la fenêtre, le clavier, la souris, les formulaires et les médias. Nous allons les découvrir au fil de ce chapitre.

Dans cette section :

- Bases de la gestion événementielle
- Evénements liés à la fenêtre
- Evénements liés au clavier
- Evénements liés à la souris
- Evénements liés aux formulaires
- Evénements liés aux médias

# Bases de la gestion événementielle

Pour capturer un événement, il suffit d'insérer l'attribut correspondant dans l'élément cible et de préciser le nom de la procédure JavaScript à exécuter, en lui passant zéro, un ou plusieurs arguments. Lorsque l'événement se produit, le code JavaScript correspondant est exécuté.

Dans le code de la diapositive suivante, trois événements sont capturés :

- chargement du document ;
- clic sur le bouton 1 ;
- clic sur le bouton 2.

Ils exécutent respectivement les fonctions JavaScript load(), bouton() avec un argument égal à 1, et bouton() avec un argument égal à 2.

## Code077.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion événementielle</title>
    <script>
      function load()
      {
        document.getElementById("status").innerHTML = "L'événement 'load' a été généré.";
      }
      function bouton(b)
      {
        document.getElementById("status").innerHTML = "Le bouton " +b + " a été cliqué.";
      }
    </script>
  </head>
  <body onload="load()" onunload="unload();">
    <p>Status: <span id="status">En attente.'</span></p>
    <button onClick="bouton(1)">Bouton 1</button>
    <button onClick="bouton(2)">Bouton 2</button>
  </body>
</html>
```

# Événements liés à la fenêtre

Mis à part onblur, onfocus et onload, qui étaient déjà présents dans HTML 4.x, tous les événements de ce tableau sont nouveaux.

Attribut	Exécution du script
<u>onafterprint</u>	Après l'impression
<u>onbeforeprint</u>	Avant l'impression
<u>onbeforeonload</u>	Avant le chargement du document
<u>onblur</u>	Lorsque la fenêtre perd le focus
<u>onerror</u>	Lorsqu'une erreur est détectée
<u>onfocus</u>	Lorsque le focus est donné à la fenêtre
<u>onhaschange</u>	Lorsque le contenu du document est modifié
<u>onload</u>	Lorsque le document est chargé
<u>onmessage</u>	Quand le message est généré
<u>onoffline</u>	Au passage de l'état en ligne à l'état déconnecté
<u>ononline</u>	Au passage de l'état déconnecté à l'état en ligne
<u>onpagehide</u>	Lorsque la fenêtre devient invisible
<u>onpageshow</u>	Lorsque la fenêtre devient visible
<u>onpopstate</u>	Lorsque l'historique de la page change
<u>onredo</u>	Après l'exécution d'un "redo"
<u>onresize</u>	Au redimensionnement de la fenêtre
<u>onstorage</u>	Au chargement d'un document
<u>onundo</u>	À l'exécution d'un "undo"
<u>onunload</u>	Lorsque l'utilisateur change de document

# Événements liés à la fenêtre

Code078.htm

```
<!DOCTYPE html>
<html>
  <head>
    <title>Événements liés à la fenêtre</title>
    <script>
      function unload()
      {
        document.getElementById("status").innerHTML = "L'événement 'unload' a été généré";
      }
      function load()
      {
        document.getElementById("status").innerHTML = "L'événement 'load' a été généré";
      }
    </script>
  </head>
  <body onload="load()" onunload="unload();">
    <p>Status: <span id="status">En attente de l'événement 'unload'</span></p>
  </body>
</html>
```

# Événements liés au clavier

Tous les événements clavier étaient déjà disponibles dans HTML 4.x.

Attribut	Exécution du script
<u>onkeydown</u>	Lorsqu'une touche est pressée
<u>onkeypress</u>	Lorsqu'une touche est pressée puis relâchée
<u>onkeyup</u>	Lorsqu'une touche est relâchée

Le code de la diapositive suivante affiche un élément span et une zone de texte <input type="text">. Un caractère tapé dans la zone de texte est récupéré *via* la fonction événementielle faitecho, déclenchée sur l'événement onkeypress. Le code de la touche utilisée est récupéré (ev.keyCode), converti en une chaîne (String.fromCharCode()) affichée dans l'élément span (document.getElementById("echo").innerHTML).

## Code079.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion événementielle</title>
    <script>
      function faitecho(ev)
      {
        document.getElementById("echo").innerHTML =
document.getElementById("echo").innerHTML + String.fromCharCode(ev.keyCode);
      }
    </script>
  </head>
  <body>
    <p>echo: <span id="echo"></span></p>
    <input type="text" id="saisie" onkeypress="faitecho(event);">
  </body>
</html>
```

# Événements liés à la souris

Attribut	Exécution du script
<u>onclick</u>	Au clic du bouton gauche
<u>ondblclick</u>	Au double-clic du bouton gauche
<u>ondrag</u>	Lorsqu'un élément est déplacé par la technique du glisser-déposer
<u>ondragend</u>	Lorsqu'un élément a fini d'être déplacé par la technique du glisser-déposer
<u>ondragenter</u>	Lorsqu'un élément a été déplacé sur une destination valide
<u>ondragleave</u>	Lorsqu'un élément est déplacé depuis un emplacement valide
<u>ondragover</u>	Lorsqu'un élément est en cours de déplacement vers une destination valide
<u>ondragstart</u>	Au début du glisser-déposer
<u>ondrop</u>	Au dépôt de l'élément sur la destination
<u>onmousedown</u>	Lorsque le bouton de la souris est enfoncé
<u>onmousemove</u>	Lorsque le pointeur se déplace
<u>onmouseout</u>	Lorsque le pointeur se déplace en dehors d'un élément
<u>onmouseover</u>	Lorsque le pointeur est déplacé sur un élément
<u>onmouseup</u>	Au relâchement du bouton de la souris
<u>onmousewheel</u>	Au déplacement de la roulette de la souris
<u>onscroll</u>	Lorsque la barre de défilement de l'élément est utilisée

Ce code capture les événements souris liés à un élément img et les affiche dans un élément span.

Code080.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion événementielle</title>
    <script>
      function traitement(param)
      {
        document.getElementById("activite").innerHTML = param;
      }
    </script>
  </head>
  <body>
    
    <p>Activité : <span id="activite"></span></p>
  </body>
</html>
```

## Exercice

Définissez un document HTML qui contient une image et un <span>. Lorsque l'utilisateur déplace la souris sur l'image, affichez ses coordonnées dans le <span>.

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Coordonnées de la souris</title>
    <script>
      function coord(e) {
        var y = e.clientY - 8;
        var x = e.clientX - 8;
        document.getElementById('xy').innerHTML = "x = " + x + ", y = " + y;
      }
    </script>
  </head>
  <body>
    <br/>
    <span id="xy"></span>
  </body>
</html>
```

## Exercice

Définissez un document HTML qui contient deux <div> dont le premier est éditable (attribut **contenteditable="true"**).

Capturez les caractères tapés dans le premier <div> et affichez-les en majuscules dans le deuxième.

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Transformation des touches frappées</title>
    <script>
      function capture(e) {
        var car = e.charCodeAt();
        document.getElementById('span2').innerHTML = String.fromCharCode(car).toUpperCase();
      }
    </script>
  </head>
  <body>
    <div id="span1" style="width: 200px; height: 20px; border: 1px solid black; background-color: yellow;" onkeypress="capture(event); contenteditable=true"></div><br>
    <div id="span2" style="width: 200px; height: 20px; border: 1px solid black; background-color: green;"> </div>
  </body>
</html>
```

# Événements liés aux formulaires

Ces événements sont déclenchés par les actions effectuées par l'utilisateur à l'intérieur d'un formulaire. Cinq d'entre eux sont une nouveauté du langage HTML5 : oncontextmenu, onformchange, onforminput, oninput et oninvalid.

Attribut	Exécution du script
<u>onblur</u>	Lorsqu'un élément perd le focus
<u>onchange</u>	Lorsque la valeur/le contenu d'un élément change
<u>oncontextmenu</u>	Lorsqu'un menu contextuel est déroulé
<u>onfocus</u>	Lorsqu'un élément reçoit le focus
<u>onformchange</u>	Lorsque le contenu du formulaire change
<u>onforminput</u>	Lorsque l'utilisateur entre des données dans le formulaire
<u>oninput</u>	Lorsqu'un élément reçoit des données entrées par l'utilisateur
<u>oninvalid</u>	Lorsqu'un élément n'est pas valide
<u>onselect</u>	Lorsqu'un élément est sélectionné
<u>onsubmit</u>	Lorsque le formulaire est soumis (généralement au clic sur le bouton Submit)

Ce code capture les actions effectuées sur une zone de texte et un bouton Submit et les affiche dans un élément span.

Code081.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion événementielle</title>
    <script>
      function traitement(param)
      {
        document.getElementById("activite").innerHTML = param;
      }
    </script>
  </head>
  <body>
    <form name="MonFormulaire" method="post">
      <label>Quel est le meilleur système d'exploitation selon vous ?</label>
      <input name="texte"
            placeholder="Entrez votre réponse ici"
            onforminput="traitement('Saisie de données');"
            onformchange="traitement('Le contenu du formulaire change');"
            onfocus="traitement('La zone de texte a le focus');"
            onblur="traitement('La zone de texte a perdu le focus');"
            >
      <input type="submit" value="Envoyer" onsubmit="traitement('Le bouton Submit a été pressé');">
    </form>
    <p>Activité : <span id="activite"></span></p>
  </body>
</html>
```

# Événements liés aux médias

Ces événements s'appliquent aux éléments video, image et audio. Mis à part l'événement onabort, ils sont tous apparus dans HTML5.

Attribut	Exécution du script
<u>onabort</u>	Sur l'événement "abort"
<u>oncanplay</u>	Lorsque le média peut commencer à être lu (il peut être amené à s'arrêter si le buffer de lecture devient vide)
<u>oncanplaythrough</u>	Lorsque le média peut être lu sans interruption jusqu'à la fin
<u>ondurationchange</u>	Lorsque la longueur du média change
<u>onemptied</u>	Lorsque le média n'est plus accessible à la suite d'un problème de réseau ou d'une erreur de chargement par exemple
<u>onended</u>	Lorsque le média a été entièrement joué
<u>onerror</u>	Lorsqu'une erreur survient pendant le chargement du média
<u>onloadeddata</u>	Lorsque les données du média ont été chargées
<u>onloadedmetadata</u>	Lorsque la durée et les autres caractéristiques du média ont été lues
<u>onloadstart</u>	Lorsque le navigateur commence à charger les données du média
<u>onpause</u>	Lorsque le média est mis en pause
<u>onplay</u>	Lorsque le média est mis en lecture
<u>onplaying</u>	Lorsque le média a commencé à être joué
<u>onprogress</u>	Lorsque l'élément est en cours de récupération des données pour le média
<u>onratechange</u>	Lorsque la vitesse de lecture change
<u>onreadystatechange</u>	Lorsque l'état (prêt/pas prêt) du média change
<u>onseeked</u>	Lorsque la recherche a pris fin
<u>onseeking</u>	Pendant la recherche (attribut <u>seeking=true</u> )
<u>onstalled</u>	Lorsqu'une erreur est rencontrée lors de la récupération des données
<u>onsuspend</u>	Lorsque la récupération des données est arrêtée avant la fin
<u>ontimeupdate</u>	Lorsque la position de lecture change
<u>onvolumechange</u>	Lorsque le volume du média est modifié
<u>onwaiting</u>	Lorsque le média n'est plus en mode de lecture mais que l'utilisateur peut demander une relecture

## Code082.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Gestion événementielle</title>
    <script>
      function etat(param)
      {
        document.getElementById("activite").innerHTML = param;
      }
    </script>
  </head>
  <body>
    <video id="video" src="https://www.youtube.com/watch?v=zas5BE2FvBc"
      controls="controls"
      onabort="etat('onabort');"
      oncanplay="etat('oncanplay');"
      oncanplaythrough="etat('oncanplaythrough');"
      ondurationchange="etat('ondurationchange');"
      onemptied="etat('onemptied');"
      onended="etat('onended');"
      onerror="etat('onerror');"
      onloadeddata="etat('onloadeddata');"
      onloadedmetadata="etat('onloadedmetadata');"
      onloadstart="etat('onloadstart');"
      onpause="etat('onpause');"
      onplay="etat('onplay');"
      onplaying="etat('onplaying');"
      onprogress="etat('onprogress');"
      onratechange="etat('onratechange');"
      onreadystatechange="etat('onreadystatechange');"
      onseeked="etat('onseeked');"
      onseeking="etat('onseeking');"
      onstalled="etat('onstalled');"
      onsuspend="etat('onsuspend');"
      ontimeupdate="etat('ontimeupdate');"
      onvolumechange="etat('onvolumechange');"
      onwaiting="etat('onwaiting');"
    >
    </video>
    <p>Activité : <span id="activite"></span></p>
  </body>
</html>
```

Cet exemple affiche une vidéo issue du site Mediaforma ([www.mediaforma.com](http://www.mediaforma.com)) dans un élément video. Tous les événements liés à cet élément sont capturés et affichés dans un élément span, en dessous de l'élément video

# Événements liés à l'orientation

Les ordinateurs récents, les téléphones mobiles et les tablettes sont équipés de capteurs qui fournissent des informations sur l'orientation, le mouvement et l'accélération de ces appareils. Certains navigateurs web donnent accès à ces informations.

Pour savoir si le navigateur peut détecter le changement d'orientation, vous utiliserez la fonction **window.DeviceOrientationEvent**.

Si cette fonction retourne true, le navigateur peut détecter le changement d'orientation. Si elle retourne false, le navigateur n'est pas en mesure de détecter l'orientation de l'appareil :

```
if (window.DeviceOrientationEvent) alert('DeviceOrientation supporté');  
else alert('DeviceOrientation non supporté');
```

# Evénements liés à l'orientation

Si l'orientation de l'appareil peut être détectée, définissez un gestionnaire d'événements pour l'événement **orientationchange** :

```
window.addEventListener('orientationchange', function(event) {}, false);
```

# Evénements liés à l'orientation

Ces quelques lignes de code affichent l'orientation du matériel dans un élément span :

```
<!DOCTYPE html>
<html>
  <head>
    <title>device Orientation</title>
    <script>
      if (window.DeviceOrientationEvent) {
        alert('DeviceOrientation supporté');
        window.addEventListener('orientationchange',
          function(event) {
            document.getElementById('status').innerHTML='orientation : ' +
              window.screen.orientation.angle + ' degrés';
          }, false);
      }
    </script>
  </head>
  <body>
    <span id="status">Modifiez l'orientation de votre device</span>
  </body>
</html>
```

# Evénements liés à l'orientation

Supposons maintenant que vous vouliez afficher l'inclinaison d'un appareil. Commencez par tester la valeur renvoyée par `window.DeviceMotionEvent` : `true` indique que le navigateur est compatible avec cette fonctionnalité, `false` indique qu'il ne l'est pas :

```
if (window.DeviceMotionEvent)
    alert('DeviceMotion supporté');
Else
    alert('DeviceMotion non supporté');
```

Si l'inclinaison de l'appareil peut être détectée, définissez un gestionnaire d'événements pour l'événement `devicemotion` :

```
window.addEventListener('devicemotion', function(event) {}, false);
```

# Événements liés à l'orientation

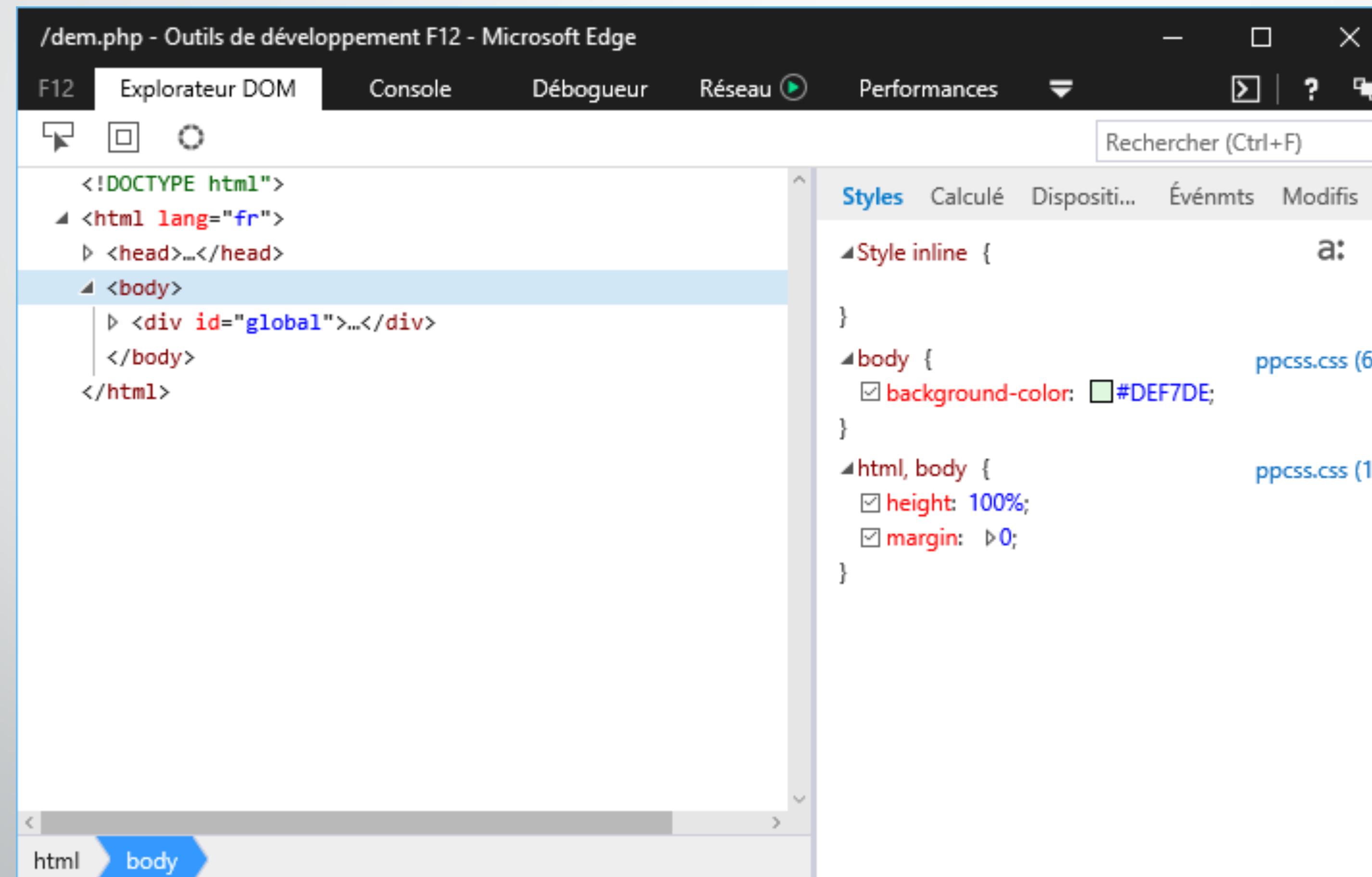
Ces quelques lignes de code suivantes affichent l'inclinaison du matériel selon les axes X,Y,Z dans un élément span :

```
<!DOCTYPE html>
<html>
  <head>
    <title>device Motion</title>
    <script>
      if (window.DeviceMotionEvent) {
        window.addEventListener('devicemotion', function(event) {
          var x = event.accelerationIncludingGravity.x;
          var y = event.accelerationIncludingGravity.y;
          var z = event.accelerationIncludingGravity.z;
          document.getElementById('status').innerHTML =
            '<ul><li>X : ' + x + '</li><li>Y : ' + y +
            '</li><li>Z : ' + z + '</li></ul>';
        }, false);
      }
    </script>
  </head>
  <body>
    <span id="status">Modifiez l'inclinaison de votre device</span>
  </body>
</html>
```

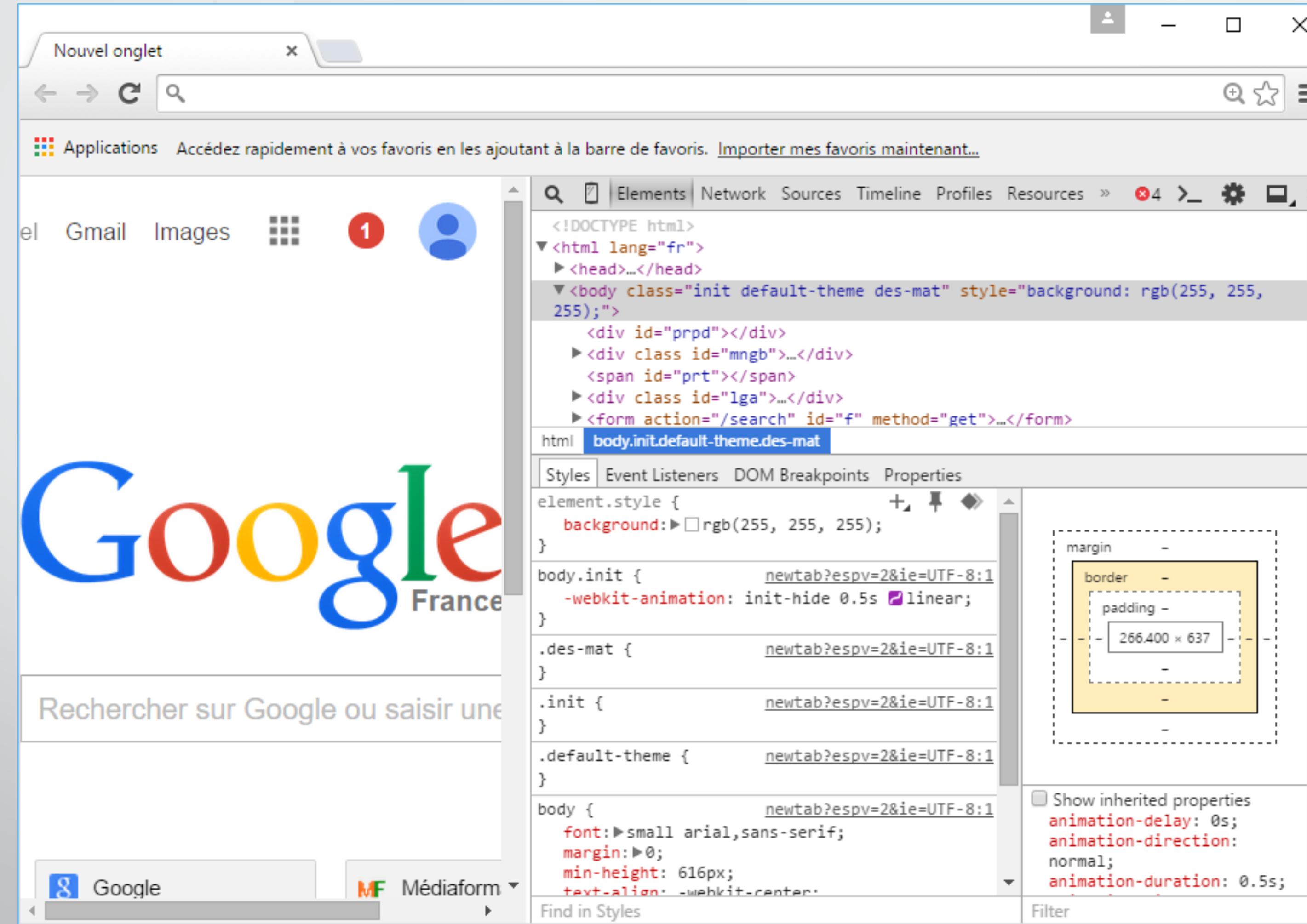
# Débogage dans la console du navigateur

# Débogage dans le navigateur

Tous les navigateurs Web actuels intègrent un débogueur accessible avec la touche de fonction **F12**. Ici, par exemple dans Microsoft Edge :



Ou encore dans Google Chrome :





Quel que soit le navigateur utilisé, les possibilités sont identiques. La différence se fait au niveau de l'interface. Les possibilités sont les suivantes :

- Console pour voir les erreurs de code
- Surlignage du DOM en pointant une instruction HTML et inversement
- Modification live du code CSS des éléments

Vous avez eu l'occasion de tester toutes ces possibilités tout au long de cette formation.

# Introduction à la POO

# Introduction à la POO

La programmation orientée objet repose sur plusieurs notions fondamentales :

- Les objets
- Les classes
- Les instances
- Les membres de classe et d'instance
- L'encapsulation
- L'héritage
- Le polymorphisme

# Objets

Pour faire un parallèle avec le monde réel, les objets sont omniprésents : humains, livres, plantes, etc.

En POO, un objet est caractérisé par une **identité**, des **états** et des **comportements**.

- **L'identité** permet d'identifier l'objet sans ambiguïté
- **Les états** sont stockés dans des propriétés (équivalentes à des variables)
- **Les comportements** sont implémentés à l'aide de méthodes (procédures et fonctions)

A titre d'exemple, voici comment pourrait être modélisé un objet "être humain" :

- **Identité** : nom ou numéro
- **Etats** : taille, poids, couleur des yeux, etc.
- **Comportements** : respirer, marcher, parler, etc.

# Classes

Le monde réel regroupe des objets du même type. Il est pratique de concevoir une maquette (un moule) d'un objet et de produire les objets à partir de cette maquette.

En POO, une maquette se nomme une **classe**. Une classe est donc un modèle de la structure statique (variables d'instance) et du comportement dynamique (les méthodes) des objets associés à cette classe



# Instances

Les objets associés à une classe se nomment des **instances**. Une instance est un objet, occurrence d'une classe, qui possède la structure définie par la classe et sur lequel les opérations définies dans la classe peuvent être appliquées.

C'est un gâteau issu du moule.



# Héritage

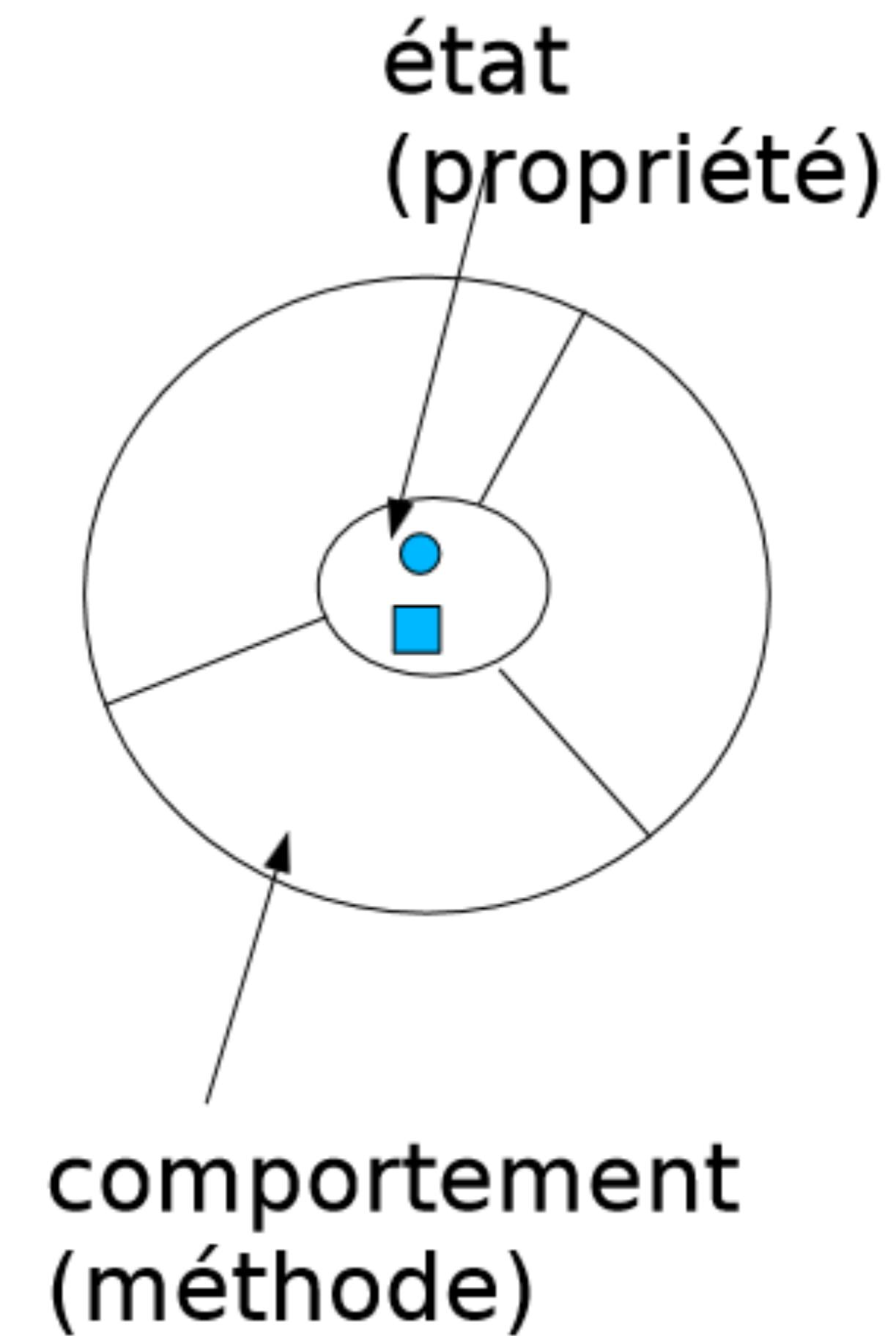
Il est possible de créer un nouveau type d'objet en prenant comme modèle un type objet déjà existant. Le nouvel objet possèdera (héritera) les mêmes champs et les mêmes méthodes que son modèle, avec toutefois la possibilité d'ajouter des nouveaux champs, d'ajouter des nouvelles méthodes ou de redéfinir les méthodes existantes. On dira que le nouvel objet est un **objet dérivé** ou un **descendant** de l'ancien.

Cette notion d'héritage permet de ne pas avoir à réécrire des méthodes déjà écrites. D'autre part les corrections ou modifications du code pourront être réalisées plus rapidement et de façon plus sûre.

# Encapsulation

En programmation objet, le concept d'encapsulation consiste à protéger l'information contenue dans un objet (ses propriétés et ses données) et de ne proposer que des méthodes spécifiques à la manipulation de cet objet.

Ainsi, les propriétés de l'objet ne seront accessibles et modifiables que par les méthodes de l'objet. Le programmeur ne pourra donc pas modifier directement les propriétés d'un objet et risquer de le mettre en péril.



# Polymorphisme

La notion de polymorphisme permet à une méthode commune à plusieurs objets descendants d'un même 'ancêtre' de fonctionner différemment selon l'objet qui l'appelle. De telles méthodes sont dites **virtuelles**.

Un exemple : Supposons que les classes **Rectangle** et **Cercle** héritent de la classe **ObjetGraphique**. On peut définir une instance **Rect1** d'un **Rectangle** comme un **ObjetGraphique**. Mais aussi, une instance **Cercle1** d'un **Cercle** comme un **ObjetGraphique**. Et par la suite, utiliser des méthodes **ObjetGraphique** sur les instances **Rect1** et **Cercle1**.

```
class ObjetGraphique {  
    methode1 () {  
    }  
    methode2 () {  
    }  
    ...  
}  
class Rectangle extends ObjetGraphique {  
    methode3 () {  
    }  
}  
class Cercle extends ObjetGraphique {  
    methode4 () {  
    }  
}
```

# *JavaScript Objet en ES5*

# Extension des classes prédéfinies

Pour étendre une classe, vous passerez par son prototype.

Par exemple, pour ajouter la fonction **even()** à la classe **Number**, vous utiliserez la syntaxe suivante :

```
Number.prototype.even = function() {  
    return this.valueOf() % 2 === 0;  
}
```

Et pour utiliser la fonction **even()**, vous ferez ceci :

```
var a = 10;  
alert(a.even()); // true car 10 est pair  
a = 11;  
alert(a.even()); //false car 11 est impair
```

# Les objets par défaut de JavaScript

JavaScript peut manipuler :

- des variables non typées;
- des objets.

Plusieurs objets prédéfinis sont disponibles :

Number, Boolean, Date, Math, String **et** Array

Chacun possède des méthodes dédiées. Pour en savoir plus sur les méthodes disponibles, connectez-vous sur le site <http://www.w3schools.com/js/default.asp> et cliquez sur :

- JS Number Methods
- JS Booleans
- JS Date Methods
- JS Math
- JS String Methods
- JS Array Methods

dans le volet gauche.

Voici un extrait des méthodes de l'objet **Number** :

## Global Methods

JavaScript global functions can be used on all JavaScript data types.

These are the most relevant methods, when working with numbers:

Method	Description
Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

## Number Methods

JavaScript number methods are methods that can be used on numbers:

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length
valueOf()	Returns a number as a number

Définition de la méthode toCapitalize() dans la classe String qui retourne la chaîne passée en mettant en majuscules la première lettre de chaque mot.

```
String.prototype.toCapitalize = function() {  
    var s = this.valueOf();  
    var mots = s.split(' ');  
    var resultat = '';  
    for (var i in mots) {  
        resultat += mots[i].slice(0,1).toUpperCase() +  
                    mots[i].slice(1) + ' ';  
    }  
    return resultat.slice(-1);  
}
```

```
<script>  
    var s = 'Il fait beau aujourd'hui';  
    alert(s.toCapitalize());  
</script>
```

# **Définition d'un objet**

JavaScript implémente de nombreux types d'objets à caractère général. En utilisant une syntaxe particulière, il est possible de définir de nouveaux objets parfaitement adaptés à chaque situation.

Pour créer un nouvel objet, vous pouvez :

- 1) instancier l'objet **Object** :

```
var objet1 = new Object();  
//Description de l'objet
```

- 2) utiliser la notation JSON :

```
var objet2 = { // Description de l'objet};
```

## Création d'un nouvel objet en instanciant la classe Object()

Supposons qu'un commerçant souhaite créer un objet **reference** pour mémoriser le nom, le prix, la quantité en stock et la remise possible sur un des produits de son catalogue.

Pour cela, il définira l'objet **reference** qui possède les propriétés **nom**, **pu**, **quantite** et **remise**.

Supposons maintenant qu'il réapprovisionne de temps en temps le stock de ce produit à raison de 10 produits à chaque fois. Il pourra définir la méthode `ajoute10()` pour réapprovisionner son stock.

Voici le code à utiliser :

```
var reference = new Object();

reference.nom = 'HD';
reference.pu = 2.2;
reference.quantite = 10;
reference.remise = 5;

reference.ajoute10 = function() {
    reference.quantite = reference.quantite + 10;
}

reference.ajoute10();
console.log(reference.quantite);
```

La première instruction crée l'objet **reference**.

Les quatre instructions suivantes définissent les propriétés de l'objet **reference**.

Le bloc d'instructions suivant définit la méthode **ajoute10()** de l'objet **reference** et ajoute 10 à la propriété **quantite** de cet objet.

Le bloc d'instructions suivant appelle la fonction **ajoute10()** et affiche la valeur de la propriété **quantité** de l'objet **reference**.

Supposons maintenant que le commerçant veuille référencer plusieurs produits.

Le plus simple consiste à créer une fonction qui définit les propriétés et la méthode **ajoute10()** du produit, et de créer autant d'objets que nécessaire en utilisant l'opérateur **new**.

Voici le code à utiliser :

```
function Reference(nom, pu, quantite, remise) {  
    this.nom = nom;  
    this.pu = pu;  
    this.quantite = quantite;  
    this.remise = remise;  
    this.ajoute10 = function() {  
        this.quantite = this.quantite + 10;  
    }  
}  
  
var E54V67 = new Reference("HD", 2.2, 10, 5);  
E54V67.ajoute10();  
console.log(E54V67.quantite);
```

# Définition d'un objet en JSON

Nous allons créer un objet **chat** de couleur noire et de nom Tosca :

```
var chat = {couleur: 'noir', nom: 'tosca'};
```

Si nécessaire, vous pouvez ajouter des propriétés à l'objet **chat** :

```
chat.age = 12;
```

Vous pouvez modifier une propriété existante :

```
chat.couleur = 'beige';
```

Voire même supprimer une propriété :

```
delete chat.age;
```

L'opérateur **in** permet de tester si un objet possède ou non une certaine propriété. Par exemple pour tester si la propriété **age** fait partie de l'objet **chat**, on utilisera cette syntaxe :

```
if ('age' in chat)
```

# Définition d'un objet en JSON

Il est possible d'ajouter des méthodes dans l'objet **chat** :

```
var chat = { couleur: 'noir',
             nom: 'tosca',
             miaule: function() {
               console.log('miaou');
             }
           };
```

Pour faire miauler le chat, il suffit d'écrire :

```
chat.miaule();
```



jours 5 et 6  
jQuery

# jQuery et AJAX

Dans cette section :

- Pourquoi utiliser jQuery
- Installation et accès à jQuery (local et CDN)
- Modifier un élément du DOM
- Sélectionner des éléments dans le DOM
- Gestion événementielle
- AJAX - Echanges avec un serveur
- Le concept d'échanges asynchrones avec http
- Les fonctions load(), \$.get(), \$.post(), \$.getJSON(),  
\$.ajax()

# Introduction à jQuery

jQuery est une bibliothèque JavaScript libre qui simplifie la manipulation du code JavaScript côté client. Dans cette formation, vous allez apprendre à utiliser jQuery pour établir des échanges avec un serveur Web via AJAX.

# Introduction à jQuery

## Pourquoi utiliser jQuery ?

Les intérêts de jQuery sont multiples. Voici les principaux :

- Dynamiser les pages Web afin d'améliorer l'IHM (*Interface Home Machine*) et rendre la navigation aussi agréable et naturelle que possible.
- Uniformiser le rendu dans les différents navigateurs disponibles sur le marché.
- Diminuer les temps de développement en utilisant les fonctionnalités avancées de jQuery
- Faciliter la maintenance du code (le nombre d'instructions est bien plus limité qu'en JavaScript)
- Manipuler des ensembles de données, rendant souvent inutile l'utilisation de boucles
- Chaîner des méthodes pour effectuer plusieurs manipulations en une seule instruction

# Introduction à jQuery

## Installation et accès à jQuery

Pour "installer" jQuery, il suffit de faire référence à ce fichier. Vous pouvez pour cela :

1. Placer le fichier **jquery.js** à l'endroit où sont hébergés les autres fichiers qui constituent votre site Web.
2. Faire appel à un **CDN** (*Content Delivery Network*)

Dans un Intranet d'entreprise, le mieux est de placer jquery.js sur le serveur de l'entreprise.

Sur un site Web, le mieux est de faire appel à un CDN

En développement, le mieux est de placer le fichier jquery.js sur l'ordinateur du développeur

# Introduction à jQuery

Dans cette formation, nous utiliserons jQuery à partir d'un CDN

Vous n'avez rien à installer sur l'ordinateur local : il vous suffit de faire référence à la bibliothèque jQuery sur le CDN. Par exemple, en utilisant l'une des deux URL suivantes :

<http://code.jquery.com/jquery.min.js>

<http://ajax.googleapis.com/ajax/libs/jquery/1/jquerymin.js>

# Introduction à jQuery

**La documentation sur jQuery**

Toute la documentation sur jQuery se trouve sur <http://api.jquery.com/>

# Introduction à jQuery

## Où placer le code jQuery ?

Le code jQuery doit être placé entre les balises <script> et </script>, quelque part dans le corps du document, c'est-à-dire entre les balises <body> et </body>.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <script src="jquery.min.js"></script>
    <script>
        <!-- Le code jQuery sera écrit ici -->
    </script>
</body>
</html>
```

# Vocabulaire jQuery

## Fonction jQuery

C'est le point d'entrée de la bibliothèque jQuery. Vous pouvez utiliser au choix l'instruction `jQuery()` ou son alias `$()`. Dans ce cours, nous utiliserons systématiquement l'alias pour limiter l'écriture.

# Méthodes jQuery

La bibliothèque jQuery est constituée d'un ensemble de blocs de code autonomes appelés **méthodes**. Ce qui fait la puissance de cette bibliothèque, c'est avant tout la grande diversité des méthodes proposées. Pour exécuter une méthode jQuery, il suffit de préciser son nom à la suite d'un sélecteur en le séparant de ce dernier par un point :

**`$(sélecteur).méthode(paramètres);`**

# Objet jQuery

On appelle « objet jQuery » l'entité retornée par la fonction `jQuery`, c'est-à-dire par `$()`. Cet objet représente un ensemble de zéro, un ou plusieurs éléments issus du DOM.

Pour faire référence à la bibliothèque jQuery, vous utiliserez une balise `<script>` dans le `<head>` :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Le titre du document</title>
    <script src="http://code.jquery.com/jquery.min.js"></script>
  </head>
  <body>
    <!-- Une ou plusieurs balises HTML pour définir le contenu du document -->
    <script>
      // Code jQuery
    </script>
  </body>
</html>
```

## Attendre la disponibilité du DOM

Le langage jQuery est utilisé pour manipuler (en lecture et en écriture) le DOM, c'est-à-dire l'arborescence du document. Il est important d'attendre la fin de la définition de l'arbre DOM avant de commencer à le manipuler. Sans quoi, des erreurs imprévisibles pourraient se produire...

Voici le code à utiliser :

```
$(function() {  
    // Le DOM a été entièrement défini  
    // Des instructions jQuery peuvent donc être exécutées  
});
```

# Syntaxe générale d'une instruction jQuery

Toutes les instructions jQuery sont construites sur le modèle suivant :

```
$ ('requête') .méthode (paramètre1, ... paramètreN);
```

La première partie requête le DOM. Il en résulte un ensemble de 0, 1 ou plusieurs éléments. la méthode jQuery est alors appliquée à cet ensemble en utilisant les paramètres entre parenthèses.

# Callbacks

Une fonction de rappel (ou **callback**) est exécutée lorsqu'une autre fonction a fini de s'exécuter. En jQuery, les fonctions de rappel sont essentiellement utilisées dans les animations et dans les appels AJAX.

Par exemple :

```
$('#element').slideUp(1000, function() {  
    // Une ou plusieurs instructions exécutées lorsque  
    // la méthode slideUp() a fini de s'exécuter  
});
```

# L'objet jQuery

## Chaînage des méthodes

En jQuery, il est possible (et souvent très efficace) de chaîner plusieurs méthodes. Il suffit pour cela de séparer les méthodes par un ".", comme par exemple dans :

```
$('#element').css('background', 'yellow').css('color', 'blue');
```



Vous en savez maintenant assez sur jQuery pour envoyer des requêtes AJAX.  
Mais avant de passer à la pratique, vous allez devoir installer un serveur local.

## WAMP Server

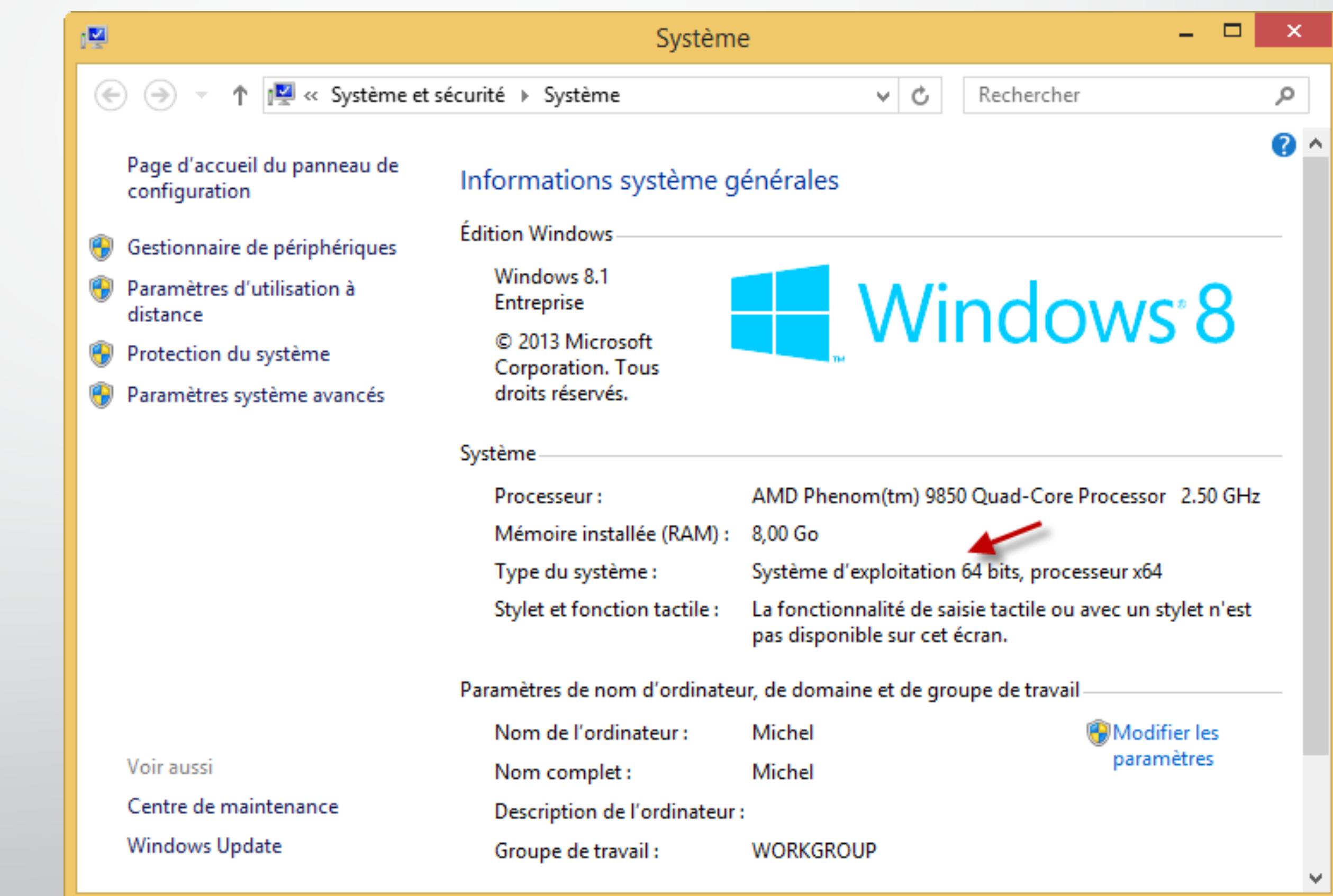
Pour exécuter du code AJAX localement, vous devez au préalable installer :

- 1) Un serveur Web Apache.** C'est lui qui sera chargé de délivrer les pages Web aux visiteurs. Par défaut, Apache ne peut gérer que des sites statiques, constitués de pages HTML. Pour programmer en PHP, vous devez installer le support des langages PHP et MySQL dans ce serveur.
- 2) Le plugin PHP pour Apache.** Ainsi, Apache sera en mesure de traiter des pages écrites en PHP.
- 3) Le logiciel de gestion de bases de données MySQL.** Vous pourrez ainsi créer des bases de données et les interroger pour créer vos pages Web dynamiquement.

Plusieurs paquetages incluant Apache, PHP et MySQL sont disponibles. Par exemple :

- WAMP Server sous Windows ;
- MAMP sous Mac OS X ;
- XAMPP sous Linux.

Avant d'installer un paquetage, vérifiez le type de votre système d'exploitation : **32 ou 64 bits**. Pour cela, appuyez simultanément sur les touches *Windows* et *Pause*. La fenêtre **Système** s'affiche. L'information apparaît en face du libellé **Taille du système** :

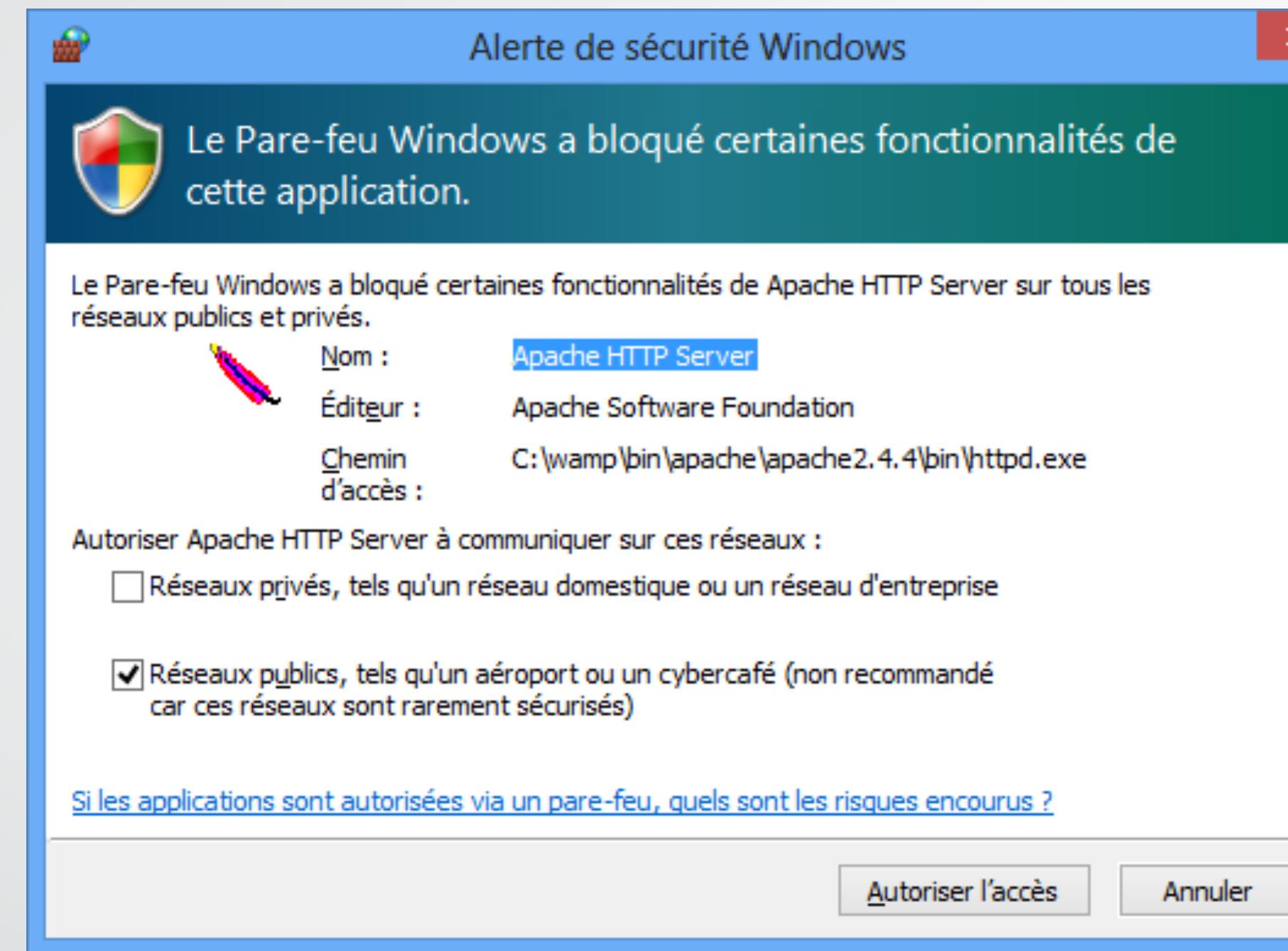


Rendez-vous sur la page <http://sourceforge.net/projects/wampserver/> et installez le logiciel WAMP Server, en version 32 ou 64 bits selon le type de votre système d'exploitation. Pour cela, sélectionnez l'onglet **Files**, cliquez sur **Wamp Server 2**, sur la dernière version du logiciel (2.5 alors que j'écris ces lignes), puis sur la version **32** ou **64** bits en fonction de votre système.

A titre d'information, la version 32 bits est moins "lourde" que la version 64 bits. Il est donc facile de la reconnaître même si son nom n'est pas totalement affiché sur l'écran. Vous pouvez aussi pointer les liens proposés pour afficher le nom complet du fichier correspondant. Si vous voyez "32b" dans le nom du fichier, il s'agit d'une version 32 bits. Si vous voyez "64b", il s'agit d'une version 64 bits :

Looking for the latest version? <a href="#">Download wampserver2.5-Apache-2.4.9-Mysql-5.6.17-php5.5.12-32b.exe (39.9 MB)</a>				
<a href="#">Home</a> / <a href="#">WampServer 2</a> / <a href="#">Wampserver 2.5</a>				
Name	Modified	Size	Downloads / Week	
<a href="#">Parent folder</a>				
<a href="#">wampserver2.5-Apache-2.4.9-Mysql...</a>	2014-05-01	39.9 MB	34 035	 
<a href="#">wampserver2.5-Apache-2.4.9-Mysql...</a>	2014-05-01	43.5 MB	38 737	 
Totals: 2 Items		83.4 MB	72 772	

Installez Wamp Server en conservant les options par défaut. A la fin de l'installation, le pare-feu de Windows se manifeste :



Cliquez sur **Autoriser l'accès** pour autoriser Apache à communiquer sur votre réseau.

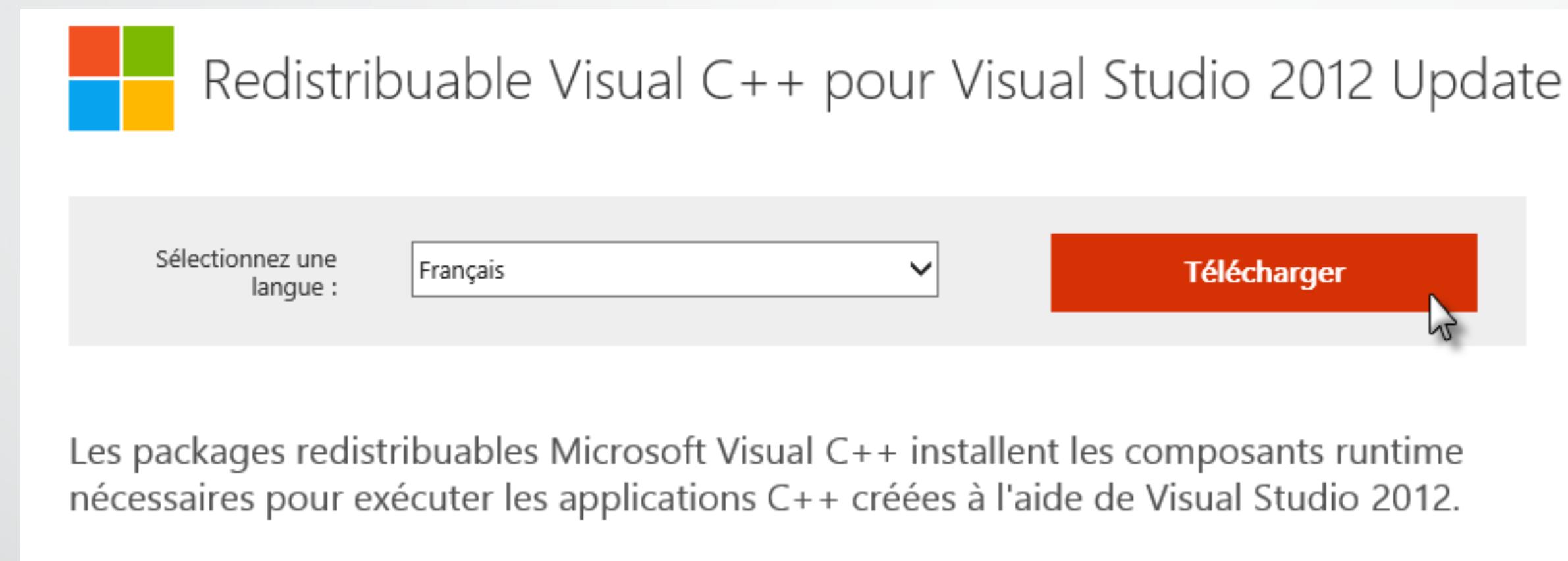
Acceptez toutes les options par défaut jusqu'à la fin de l'installation.

## Erreur d'installation

En cas d'erreur 0xc000007b sur la DLL msvcr100.dll lors de l'installation de WAMP Server, téléchargez et installez le logiciel Visual C++ Redistributable for Visual Studio 2012. Pour cela, rendez-vous sur la page suivante :

<http://www.microsoft.com/en-us/download/details.aspx?id=30679>

Choisissez Français dans la liste déroulante et cliquez sur Télécharger :



Choisissez la version **32 bits** ou **64 bits** en fonction de votre système d'exploitation et installez-la sur votre ordinateur :

Choisissez le téléchargement souhaité	
<input type="checkbox"/> Nom du fichier	Taille
<input type="checkbox"/> VSU4\vcredist_arm.exe	1.4 MB
<input type="checkbox"/> VSU4\vcredist_x64.exe	6.9 MB
<input type="checkbox"/> VSU4\vcredist_x86.exe	6.3 MB

Désinstallez WAMP Server, redémarrez l'ordinateur et réinstallez-le.

## *WAMP Server s'est-il bien installé ?*

Une icône représentant WAMP Server devrait se trouver dans la Zone de notification. Si ce n'est pas le cas, tapez *wamp* dans le menu Démarrer (Windows XP/7) ou dans la page d'accueil (Windows 8) et cliquez sur **Start WampServer**. Quelques secondes plus tard, une icône représentant Wamp Server est disponible dans la Zone de notification :



Si l'icône de WAMP Server reste orange dans la zone de notification, il se peut que le service Apache ou MySQL ne fonctionne plus. Dans ce cas :

- cliquez sur l'icône de **WAMP Server**, puis sur **Apache, Service et Installer le service.**
- cliquez sur l'icône de **WAMP Server**, puis sur **MySQL, Service et Installer le service.**

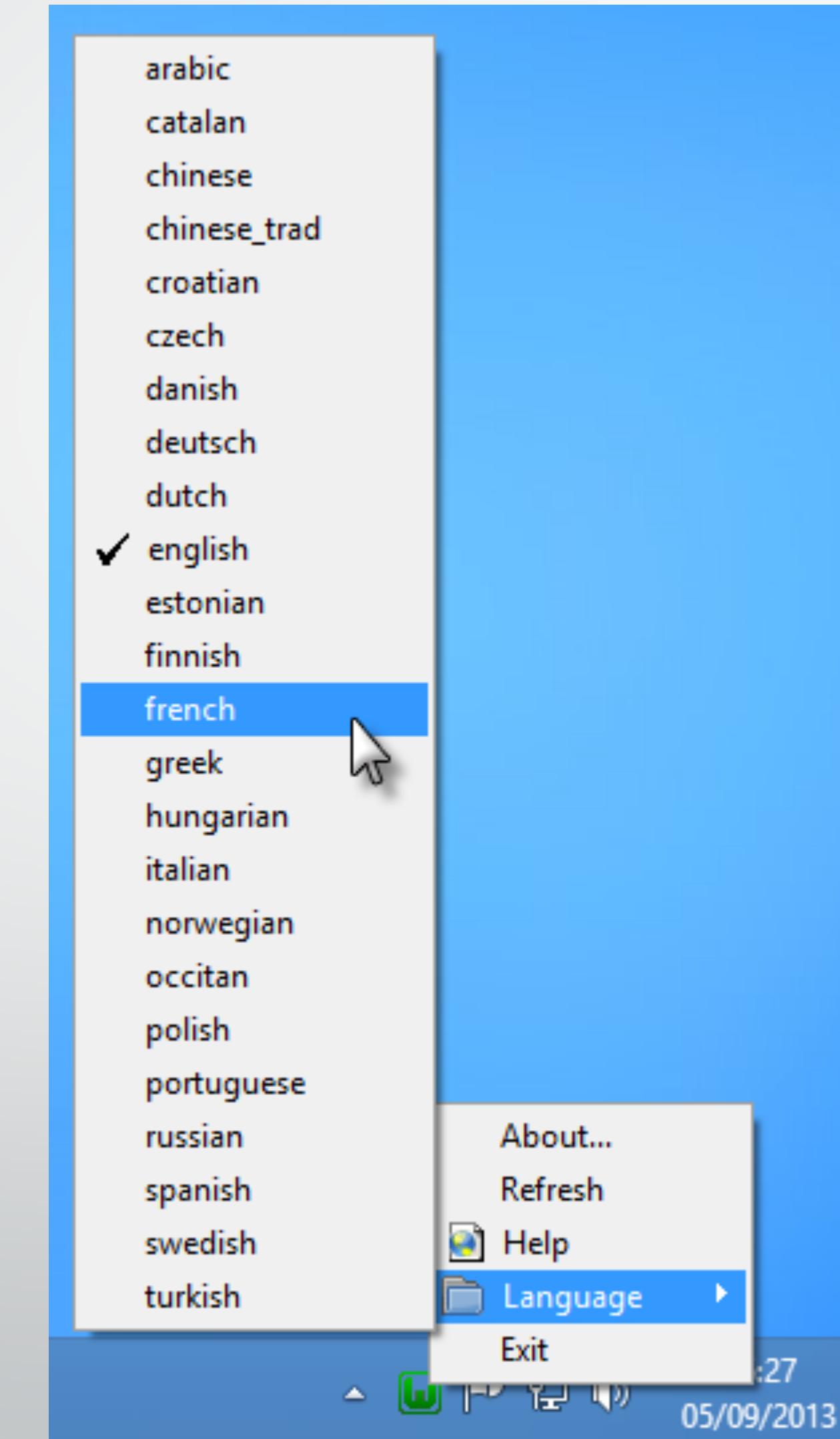
Il se peut aussi que le port **80** soit utilisé par un autre service que Apache.

Cliquez sur l'icône de **WAMP Server**, puis sur **Apache**, **Service** et **Tester le port 80**.

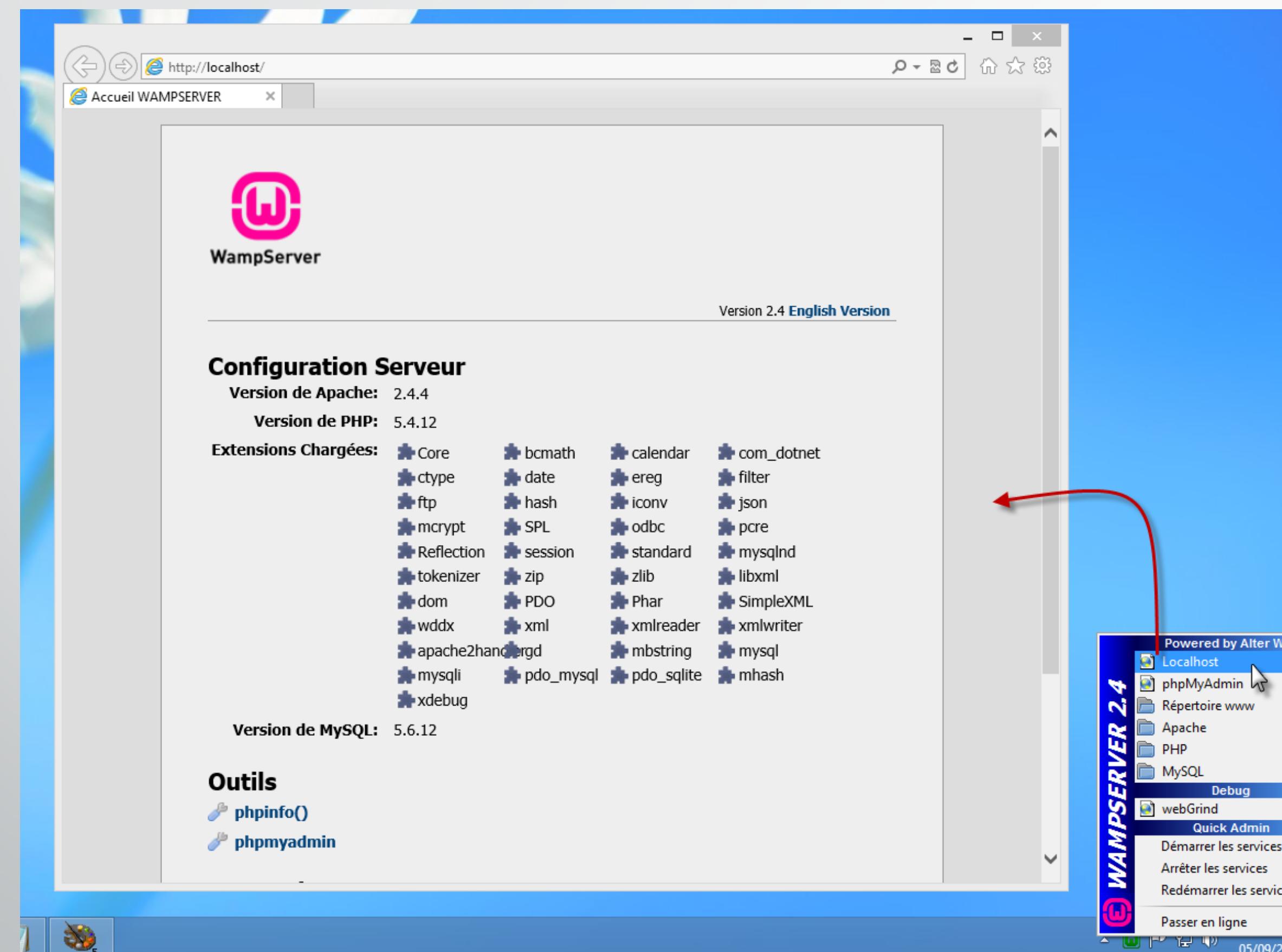
Si le port 80 n'est pas utilisé par Apach, :

- cliquez sur l'icône de **WAMP Server**, sur **Apache** puis sur **httpd.conf**.
- recherchez le port **80** dans ce fichier et remplacez-le par **81**.
- fermez puis redémarrez **Wamp Server** pour prendre en compte la nouvelle configuration.

Par défaut, WAMP s'installe en anglais. Pour lui faire parler la langue de Molière, cliquez du bouton droit sur son icône, pointez **Language** et cliquez sur **French**.



Pour savoir si WAMP s'est bien installé, cliquez sur son icône et choisissez **localhost** dans le menu. Au bout de quelques instants, la page d'accueil de WAMP Server s'affiche dans votre navigateur par défaut. Apache est donc opérationnel :



Si cette page refuse de s'afficher, relancez Apache. Pour cela, cliquez sur l'icône de **WAMP Server**, puis sur **Redémarrer les services**.

Si cela n'a toujours aucun effet, désinstallez puis réinstallez WAMP Server.

Ca y est, vous allez (enfin) passer à la pratique et lancer vos premières requêtes AJAX en jQuery !

Attention : tous les codes développés dans la suite de la formation ne fonctionnent que sur un serveur

# AJAX

## Charger un fichier

Pour mettre à jour un élément sur une page Web en utilisant des données stockées sur le serveur, le plus simple consiste à utiliser la méthode `jQuery load()` :

```
load('URL de l'élément', function() {  
    //une ou plusieurs instructions exécutées après le chargement des données  
});
```

La fonction `callback` est facultative. Si elle est présente, les instructions qui la composent seront exécutées lorsque le fichier aura été entièrement rapatrié par la méthode `load()`.

A titre d'exemple, le document de la diapo suivante ([load.htm](#)) contient deux boutons et un élément `div`. Le premier bouton va être utilisé pour afficher un texte dans l'élément `div` et le deuxième pour afficher une image dans ce même élément.

# AJAX

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Chargement AJAX avec load()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
  <body>
    <script src="jquery.js"></script>
    <button id="charge-texte">Charger le texte</button>
    <button id="charge-image">Charger l'image</button><br><br>
    <div id="zone"></div>
    <script>
      $(function() {
        $('#charge-texte').on('click', function() {
          $('#zone').load('texte.htm');
        });
        $('#charge-image').on('click', function() {
          $('#zone').load('image.htm');
        });
      });
    </script>
  </body>
</html>
```

Dans cet exemple, aucune fonction callback n'étant spécifiée dans les paramètres de la méthode **load()**, cette dernière se contente de charger les fichiers HTML correspondants et de les afficher dans l'élément **div**.

81.htm

# AJAX

Voici le code du fichier **texte.htm** :

```
<p>
  <font size="3"><i>Lorem <b>ipsum</b> dolor sit amet, consectetur adipiscing
  elit. Sed non isus. Lectus tortor, dignissim sit amet, adipiscing nec,
  ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa,
  varius a, semper congue, euismod non, mi. Proin porttitor, orci nec
  nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet
  erat. Duis semper.</i></font>
</p>
```

Et voici le code du fichier **image.htm** :

```
<style type="text/css">
  p { text-align: center; }
</style>
<p></p>
```

# AJAX

## Charger une partie d'un fichier

En ajoutant un deuxième paramètre à la méthode `load()`, il est possible de limiter le chargement de données en utilisant un sélecteur jQuery :

```
load('URL sélecteur', function() {  
    //une ou plusieurs instructions exécutées après le chargement des données  
});
```

Où **URL** est l'URL de l'élément à charger et **sélecteur** est un sélecteur jQuery sans le signe **\$** et sans les parenthèses.

❖ En partant du code de l'exemple précédent ([load.htm](#)), insérer trois boutons dans le corps du document et associez-leur un gestionnaire d'événement qui charge les parties #p1, #p2 et #p3 de ce document ([texte2.htm](#)) :

```
<p id="p1">  
<font size="3"><i>Lorem <b>ipsum</b> dolor sit amet, consectetur adipiscing elit. Sed non  
risus. Lectus tortor, dignissim sit amet, adipiscing nec,  
ultricies sed, dolor. Cras elementum ultrices diam. Maecenas  
ligula massa, varius a, semper congue, euismod non, mi. Proin  
porttitor, orci nec nonummy molestie, enim est eleifend mi, non  
fermentum diam nisl sit amet erat. Duis semper.</i></font>  
</p>  
  
<p id="p2">  
Sed ut perspiciatis unde omnis iste natus error sit voluptatem  
accusantium doloremque laudantium, totam rem aperiam, eaque ipsa  
quae ab illo inventore veritatis et quasi architecto beatae vitae  
dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas  
sit aspernatur aut odit aut fugit, sed quia consequuntur magni  
dolores eos qui ratione voluptatem sequi nesciunt.  
</p>  
  
<p id="p3">  
Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet,  
consectetur, adipisci velit, sed quia non numquam eius modi tempora  
incident ut labore et dolore magnam aliquam quaerat voluptatem.  
Ut enim ad minima veniam, quis nostrum exercitationem ullam  
corporis suscipit laboriosam, nisi ut aliquid ex ea commodi  
consequatur? Quis autem vel eum iure reprehenderit qui in ea  
voluptate velit esse quam nihil molestiae consequatur, vel  
illum qui dolorem eum fugiat quo voluptas nulla pariatur?  
</p>
```

# AJAX

## Solution

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Chargement AJAX avec load()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
  <body>
    <script src="jquery.js"></script>
    <button id="charge-texte1">Charger le texte #p1</button>
    <button id="charge-texte2">Charger le texte #p2</button>
    <button id="charge-texte3">Charger le texte #p3</button>
    <div id="zone"></div>
    <script>
      $(function() {
        $('#charge-texte1').on('click', function(){
          $('#zone').load('texte2.htm #p1');
        });
        $('#charge-texte2').on('click', function(){
          $('#zone').load('texte2.htm #p2');
        });
        $('#charge-texte3').on('click', function(){
          $('#zone').load('texte2.htm #p3');
        });
      });
    </script>
  </body>
</html>
```

82.htm

# AJAX

## Requête GET

La méthode `load()` n'est pas la seule à pouvoir récupérer des données via AJAX. Vous pouvez également utiliser :

- la fonction jQuery `$.get()` pour obtenir des données envoyées par le serveur en utilisant une requête HTTP GET.
- la fonction jQuery `$.post()` pour obtenir des données envoyées par le serveur en utilisant une requête HTTP POST.

Vous utiliserez la fonction `$.get()` si les données envoyées au serveur sont de petite taille. Vous utiliserez la fonction `$.post()` si les données envoyées au serveur sont de grande taille ou contiennent des informations confidentielles (des mots de passe par exemple).

# AJAX

Voici la syntaxe de la fonction **\$.get()** :

```
$.get(URL, function() {  
    // Une ou plusieurs instructions exécutées lorsque les données ont été  
    // rapatriées  
});
```

A titre d'exemple, nous allons exécuter un fichier PHP afin d'extraire les données qui y sont stockées. Ces données correspondent aux trois lois de la robotique d'Isaac Asimov. L'URL passée sera du type suivant :

```
donnees.php?l=1 // pour obtenir la première loi  
donnees.php?l=2 // pour obtenir la deuxième loi  
donnees.php?l=3 // pour obtenir la troisième loi
```

# AJAX

Voici le code du fichier PHP :

```
<?php
    $loi = array("Un robot ne peut porter atteinte à un être humain, ni, restant passif, permettre qu'un être humain soit exposé au danger.",
                "Un robot doit obéir aux ordres que lui donne un être humain, sauf si de tels ordres entrent en conflit avec la Première loi.",
                "Un robot doit protéger son existence tant que cette protection n'entre pas en conflit avec la Première ou la Deuxième loi.");
    $l=$_GET["l"];
    echo "<u>Loi de la robotique N° ".$l."</u><br><br>";
    echo "<b>".$loi[$l-1]."</b>";
?>
```

donnees.php

# AJAX

Voici le résultat obtenu lorsque l'utilisateur clique sur le premier bouton :



## Et voici le code HTML5/jQuery utilisé :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Requête AJAX $.get()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
  <body>
    <script src="jquery.js"></script>
    <h2>Les lois de la robotique, selon Isaac Asimov</h2>
    <button id="loi1">Loi N° 1</button>
    <button id="loi2">Loi N° 2</button>
    <button id="loi3">Loi N° 3</button><br>
    <div id="zone"></div>
```

83.htm

```
<script>
  $(function() {
    $('#loi1').on('click', function() {
      $.get(' http://lem.korp.free.fr/jquery/donnees.php?l=1', function(data) {
        $('#zone').html(data);
      });
    });
    $('#loi2').on('click', function() {
      $.get('http://lem.korp.free.fr/jquery/donnees.php?l=2', function(data) {
        $('#zone').html(data);
      });
    });
    $('#loi3').on('click', function() {
      $.get('http://lem.korp.free.fr/jquery/donnees.php?l=3', function(data) {
        $('#zone').html(data);
      });
    });
  });
</script>
</body>
</html>
```

# AJAX

## Requête POST

Dans l'exemple précédent, les paramètres passés apparaissaient dans l'URL. Pour garder les paramètres secrets ou pour passer des données de taille importante, vous utiliserez une requête POST.

Voici la syntaxe de la fonction `$.post()` :

```
$.post(URL, {donnée1: 'valeur1', donnée2: 'valeur2',...}, function() {  
    // Une ou plusieurs instructions exécutées lorsque les données ont été  
    // rapatriées  
});
```

Ici, les données sont passées dans le deuxième paramètre de la fonction `$.post()`.

# AJAX

Nous allons modifier le code du programme précédent pour accéder aux données *via* une requête POST.

L'URL interrogée aura pour nom **donneesPost.php**. Une seule donnée nommée "l" sera communiquée. Elle aura pour valeur 1, 2 ou 3 selon la loi à afficher.

Le programme PHP est légèrement différent :

```
<?php
    $loi = array("Un robot ne peut porter atteinte à un être humain, ni, restant
passif, permettre qu'un être humain soit exposé au danger.",
                "Un robot doit obéir aux ordres que lui donne un être humain, sauf
si de tels ordres entrent en conflit avec la Première loi.",
                "Un robot doit protéger son existence tant que cette protection
n'entre pas en conflit avec la Première ou la Deuxième loi.");
    $l=$_POST["l"];
    echo "<u>Loi de la robotique N° ".$l."</u><br><br>";
    echo "<b>".$loi[$l-1]."</b>";
?>
```

donneesPost.php

# Le code HTML5/jQuery est également légèrement différent :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Requête AJAX $.post()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
  <body>
    <script src="jquery.js"></script>
    <h2>Les lois de la robotique, selon Isaac Asimov</h2>
    <button id="loi1">Loi N° 1</button>
    <button id="loi2">Loi N° 2</button>
    <button id="loi3">Loi N° 3</button><br><br>
    <div id="zone"></div>
```

Suite ...

```
<script>
$(function() {
    $('#loi1').on('click', function() {
        $.post(' http://lem.korp.free.fr/jquery/donneesPost.php', {l:'1'}, function(data) {
            $('#zone').html(data);
        });
    });
    $('#loi2').on('click', function() {
        $.post(' http://lem.korp.free.fr/jquery/donneesPost.php', {l:'2'}, function(data) {
            $('#zone').html(data);
        });
    });
    $('#loi3').on('click', function() {
        $.post(' http://lem.korp.free.fr/jquery/donneesPost.php', {l:'3'}, function(data) {
            $('#zone').html(data);
        });
    });
});
</script>
</body>
</html>
```

# AJAX

## Charger des données JSON

Les fichiers au format **JSON** (*JavaScript Object Notation*) permettent de représenter des informations structurées.

Par exemple :

```
{  
    "Employés": [  
        { "Prénom": "John" , "Nom": "Doe" } ,  
        { "Prénom": "Anna" , "Nom": "Smith" } ,  
        { "Prénom": "Peter" , "Nom": "Jones" }  
    ]  
}
```

Ce qui équivaut à :

```
<Employés Prénom="John" Nom="Doe">  
<Employés Prénom="Anna" Nom="Smith">  
<Employés Prénom="Peter" Nom="Jones">
```

# AJAX

Pour accéder à des données JSON, vous utiliserez la fonction **\$.getJSON()** :

```
$.getJSON(URL, function(données) {  
    // Une ou plusieurs instructions pour traiter les données lues  
});
```

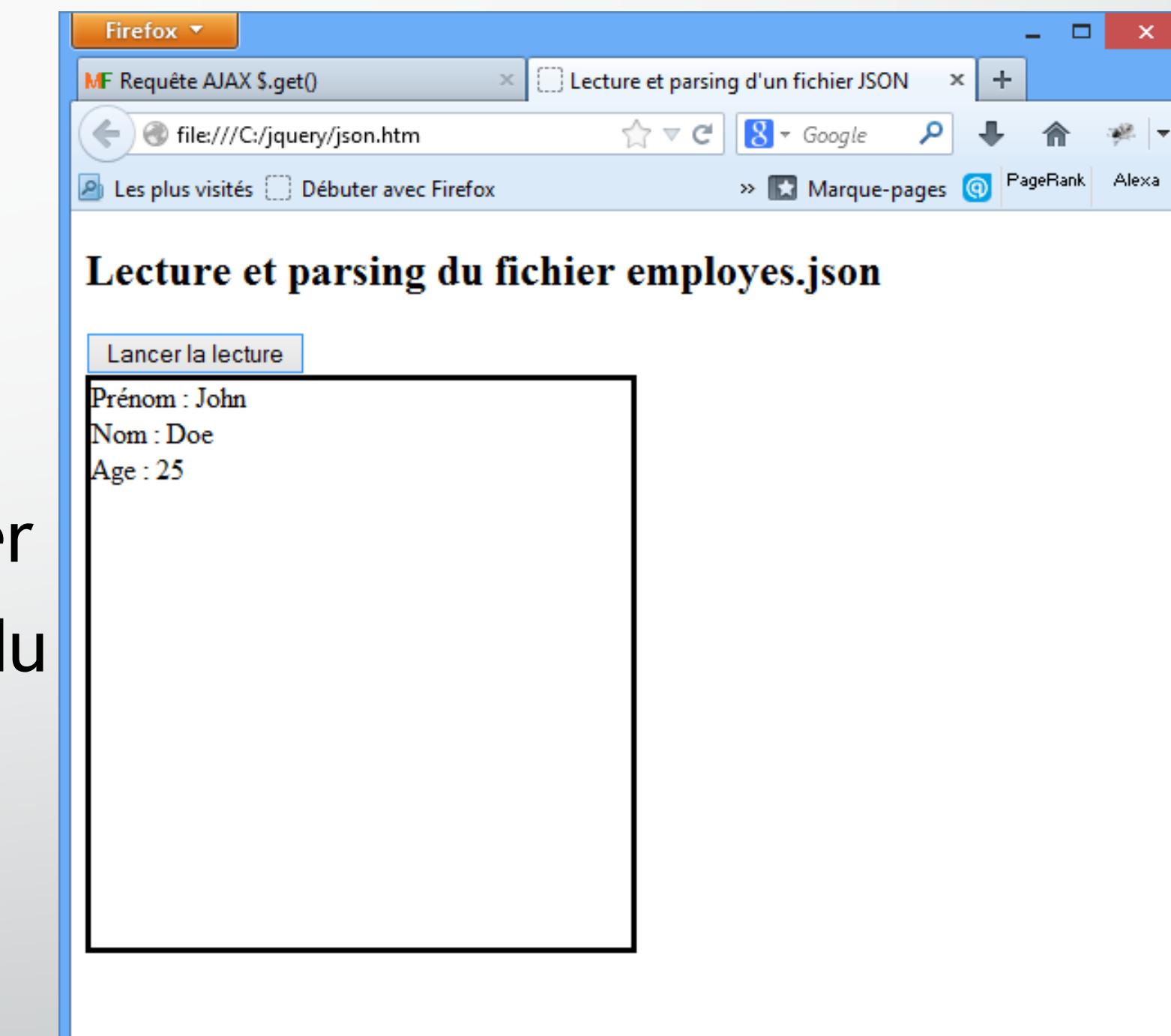
# AJAX

Saisissez ces données dans NotePad++ et sauvegardez-les dans le fichier **employes.json** :

```
{  
    "Prenom": "John",  
    "Nom": "Doe",  
    "Age": "25"  
}
```

employes.json

Nous allons accéder à ce fichier en jQuery et exploiter les données qui le composent. Voici le résultat attendu



# AJAX

Et voici le fichier HTML5/jQuery utilisé :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Lecture et parsing d'un fichier JSON</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
```

85.htm

Suite ...

```
<body>
    <script src="jquery.js"></script>
    <h2>Lecture et parsing du fichier employes.json</h2>
    <button id="lecture">Lancer la lecture</button>
    <div id="zone"></div>
    <script>
        $(function() {
            $('#lecture').on('click', function() {
                $.getJSON('employes.json', function(data) {
                    $('#zone').append('Prénom : ' + data.Prenom + '<br>');
                    $('#zone').append('Nom : ' + data.Nom + '<br>');
                    $('#zone').append('Age : ' + data.Age + '<br>');
                });
            });
        });
    </script>
</body>
</html>
```

# AJAX

L'extraction (aussi appelée **parsing**) des données se fait dans la fonction callback :

```
$.getJSON('employes.json', function(data) {
    $('#zone').append('Prénom : ' + data.Prenom + '<br>');
    $('#zone').append('Nom : ' + data.Nom + '<br>');
    $('#zone').append('Âge : ' + data.Age + '<br>');
});
```

La structure du fichier JSON pris en exemple est ultra simple. Pour accéder à ses différentes composantes, on utilise le paramètre de la fonction callback, suffixé par le nom des différents champs à accéder :

```
data.Prenom  
data.Nom  
data.Age
```

## Exercice :

### Créez le fichier **employees2.json** :

```
[  
  { "Prenom":"John" , "Nom":"Doe", "Age":"25" } ,  
  { "Prenom":"Anna" , "Nom":"Smith", "Age":"34" } ,  
  { "Prenom":"Peter" , "Nom":"Jones", "Age":"17" }  
]
```

employees2.json

Modifiez le code jQuery pour afficher toutes les données de ce fichier JSON. Pour cela, vous parcourrez les données retournées par la fonction **\$.getJSON()** en utilisant la fonction **\$.each()** dont voici la syntaxe :

```
$.each(données à examiner, function(index, données extraites) {  
  // Instructions pour examiner les données extraites  
});
```

## Solution :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Lecture et parsing d'un fichier JSON</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
```

## Suite ...

```
<body>
    <script src="jquery.js"></script>
    <h2>Lecture et parsing du fichier employes.json</h2>
    <button id="lecture">Lancer la lecture</button>
    <div id="zone"></div>
    <script>
        $(function() {
            $('#lecture').on('click', function() {
                $.getJSON('employes2.json', function(data) {
                    $.each(data, function(index, d) {
                        $('#zone').append('Prénom : ' + d.Prenom + '<br>');
                        $('#zone').append('Nom : ' + d.Nom + '<br>');
                        $('#zone').append('Age : ' + d.Age + '<br><br>');
                    });
                });
            });
        });
    </script>
</body>
</html>
```

# AJAX

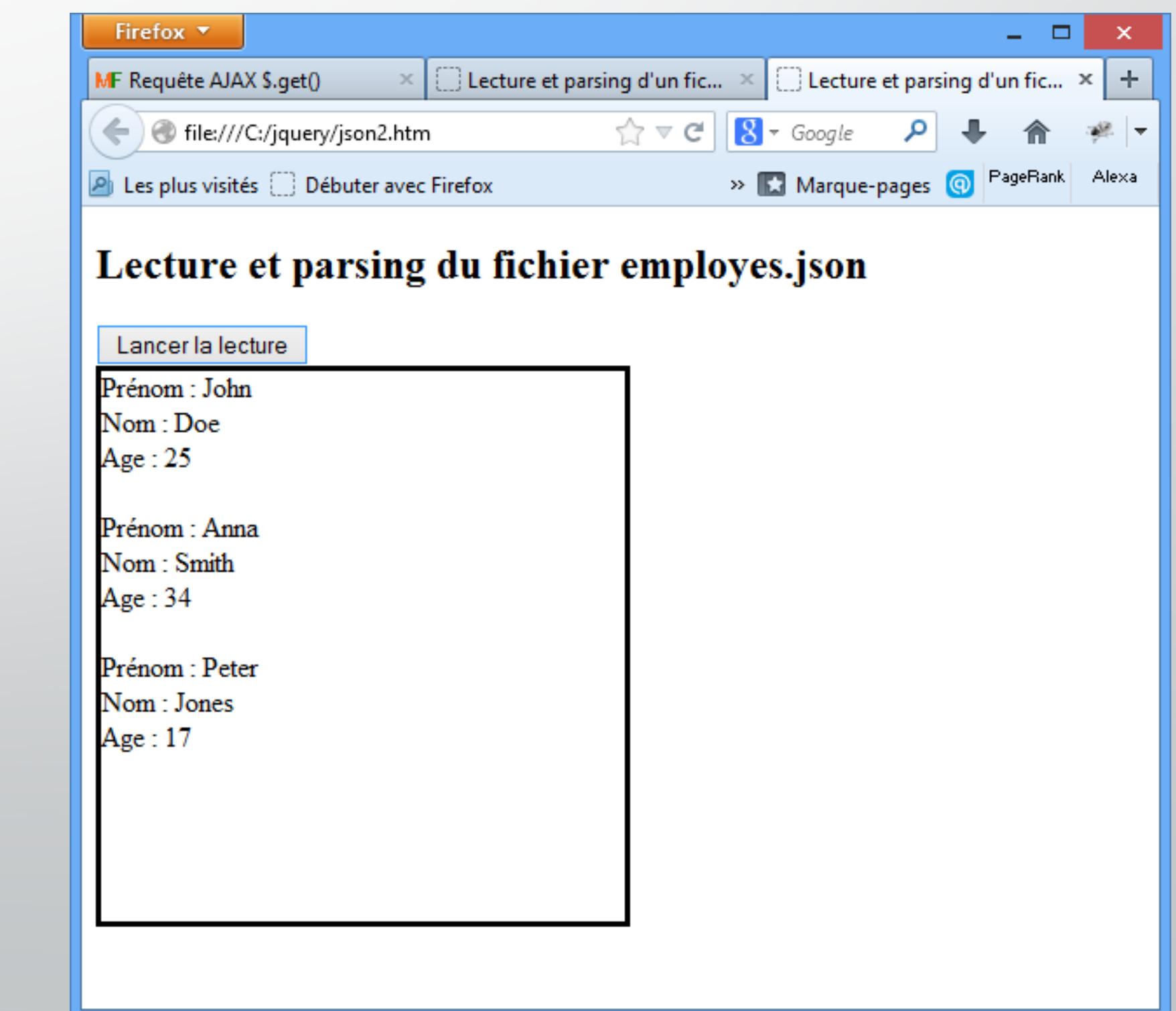
La fonction callback de `$.getJSON()` obtient la totalité des données du fichier JSON. La boucle `$.each()` porte sur les données extraites par `$.getJSON()`. Sa fonction callback va s'intéresser individuellement à chaque enregistrements, ici représentés par la lettre "d" :

```
$.getJSON('employees2.json', function(data) {  
    $.each(data, function(index, d) {
```

Pour extraire les données d'un enregistrement, il suffit de passer par "d" et de le suffixer par le nom des champs concernés : **Prenom**, **Nom** et **Age** :

```
$( '#zone' ).append('Prénom : ' + d.Prenom + '<br>');  
$( '#zone' ).append('Nom : ' + d.Nom + '<br>');  
$( '#zone' ).append('Age : ' + d.Age + '<br><br>');
```

Voici le résultat :



# AJAX - Lecture de données XML

Voici un fichier XML nommé **data.xml** :

```
<sites>
  <site id="1">
    <title>Google</title>
    <url>http://www.google.fr</url>
  </site>
  <site id="2">
    <title>Microsoft</title>
    <url>https://www.microsoft.com/fr-fr/default.aspx</url>
  </site>
  <site id="3">
    <title>Mediaforma</title>
    <url>http://www.mediaforma.com</url>
  </site>
</sites>
```

Nous allons extraire les données de ce fichier et les afficher dans une balise **<div>**.  
Pour cela, nous utiliserons :

- La fonction **find()** pour trouver un nœud dans la structure ;
- La fonction **each()** pour boucler sur les nœuds **<site>**.

Voici le code utilisé :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Lecture et parsing XML</title>
    <style>
      #zone {
        width: 500px;
        height: 300px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
```

```
<body>
  <script src="jquery.js"></script>
  <h2>Lecture et parsing du fichier data.xml</h2>
  <button id="lecture">Lancer la lecture</button>
  <div id="zone"></div>
  <script>
    $(function() {
      $('#lecture').on('click', function() {
        $.get('data.xml', function(donnees) {
          $(donnees).find('site').each(function() {
            var id = $(this).attr('id');
            var title = $(this).find('title').text();
            var url = $(this).find('url').text();
            $('#zone').append(id + ' : <a href="' +
              url + '">' + title + '</a><br>');
          }));
        });
      });
    });
  </script>
</body>
</html>
```

Testez ce code

87.htm

# AJAX

## La fonction `$.ajax()`

La fonction `$.ajax()` permet d'émettre des requêtes AJAX. Elle admet de très nombreux paramètres. Pour avoir une description exhaustive de cette fonction, reportez-vous à la documentation en ligne : <http://api.jquery.com/jQuery.ajax/>.

# AJAX

Dans cette formation, nous utiliserons l'une des syntaxes de la fonction `$.ajax()` :

```
$.ajax(options);
```

Où `options` peut contenir les éléments suivants :

- **type** : type de la requête, **GET** ou **POST** (**GET** par défaut)
- **url** : adresse à laquelle la requête doit être envoyée
- **data** : données à envoyer au serveur
- **dataType** : type des données qui doivent être retournées par le serveur : **xml**, **html**, **script**, **json**, **text**
- **success** : fonction à appeler si la requête aboutit
- **error** : fonction à appeler si la requête n'aboutit pas
- **timeout** : délai maximum (en millisecondes) pour que la requête soit exécutée. Si ce délai est dépassé, la fonction spécifiée dans le paramètre **error** sera exécutée

A titre d'exemple, nous allons réécrire les programmes `get.htm` et `post.htm` pour utiliser la fonction `$.ajax()` à la place des fonctions `$.get()` et `$.post()`.

# Requête GET via la fonction \$.ajax()

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Requête get $.ajax()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
  <body>
    <script src="jquery.js"></script>
    <h2>Les lois de la robotique, selon Isaac Asimov</h2>
    <button id="loi1">Loi N° 1</button>
    <button id="loi2">Loi N° 2</button>
    <button id="loi3">Loi N° 3</button><br><br>
    <div id="zone"></div>
```

88.htm

Suite ...

```
<script>
$(function() {
    $('#loi1').on('click', function() {
        $.ajax({
            type: 'GET',
            url: 'donnees.php?l=1',
            timeout: 3000,
            success: function(data) {
                $('#zone').html(data);
            },
            error: function() {
                $('#zone').html('Cette requête AJAX n\'a pas abouti');
            }
        });
    });
    $('#loi2').on('click', function() {
        $.ajax({
            type: 'GET',
            url: 'donnees.php?l=2',
            timeout: 3000,
            success: function(data) {
                $('#zone').html(data);
            },
            error: function() {
                $('#zone').html('Cette requête AJAX n\'a pas abouti');
            }
        });
    });
}) ;
```

88.htm

Suite ...

88.htm

```
$('#loi3').on('click', function() {
    $.ajax({
        type: 'GET',
        url: 'donnees.php?l=3',
        timeout: 3000,
        success: function(data) {
            $('#zone').html(data);
        },
        error: function() {
            $('#zone').html('Cette requête AJAX n\'a pas abouti');
        }
    });
});
</script>
</body>
</html>
```

# AJAX

Le code est bien plus "verbeux" qu'avec la fonction `$.get()`, mais il est aussi très simple à comprendre, facile à maintenir et plus complet. Ici, nous avons introduit un message d'erreur si la requête n'aboutit pas :

```
error: function() {
    $('#zone').html('Cette requête AJAX n\'a pas abouti');
}
```

Ce code est accessible ici : <http://www.mediaforma.com/encours/getAjax.htm>

# Requête POST via la fonction \$.ajax()

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Requête post $.ajax()</title>
    <style>
      #zone {
        width: 300px;
        height: 315px;
        border-style: solid;
        border-width: 3px;
        border-color: black;
      }
    </style>
  </head>
```

```
<body>
    <script src="jquery.js"></script>
    <h2>Les lois de la robotique, selon Isaac Asimov</h2>
    <button id="loi1">Loi N° 1</button>
    <button id="loi2">Loi N° 2</button>
    <button id="loi3">Loi N° 3</button><br><br>
    <div id="zone"></div>
    <script>
        $(function() {
            $('#loi1').on('click', function() {
                $.ajax({
                    type: 'POST',
                    url: 'donneesPost.php',
                    data: {l:'1'},
                    timeout: 3000,
                    success: function(data) {
                        $('#zone').html(data);
                    },
                    error: function() {
                        $('#zone').html('Cette requête AJAX n\'a pas abouti');
                    }
                });
            });
        });
    </script>
```

```
$('#loi2').on('click', function() {
    $.ajax({
        type: 'POST',
        url: 'donneesPost.php',
        data: {l:'2'},
        timeout: 3000,
        success: function(data) {
            $('#zone').html(data);
        },
        error: function() {
            $('#zone').html('Cette requête AJAX n\'a pas abouti');
        }
    });
});
```

```
$('#loi3').on('click', function() {
    $.ajax({
        type: 'POST',
        url: 'donneesPost.php',
        data: {l:'3'},
        timeout: 3000,
        success: function(data) {
            $('#zone').html(data);
        },
        error: function() {
            $('#zone').html('Cette requête AJAX n\'a pas abouti');
        }
    });
});
</script>
</body>
</html>
```

Ce code est accessible ici :

<http://www.mediaforma.com/encours/postAjax.htm>

89.htm

# Lecture de données dans une base de données

Pour être en mesure de lire des données dans une base de données serveur en utilisant une requête AJAX, vous devez utiliser du code côté serveur. Ici par exemple, nous utiliserons du code PHP.

Voici les étapes que nous allons suivre :

- 1) PHP : Création d'une base de données et d'une table puis injection des données dans la table.
- 2) PHP : Création d'un programme d'interrogation de la base de données.
- 3) jQuery : Interrogation du programme créé dans l'étape 2 pour afficher les données de la table en AJAX avec (par exemple) une requête GET.

# Etape I – Création de la base, de la table et des données

```
<?php
    // Création de la base de données
    try {
        $base = new PDO('mysql:host=localhost', 'root', '');
    }
    catch(exception $e) {
        die('Erreur '.$e->getMessage());
    }
    $base->exec("CREATE DATABASE basephp DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci");
    $base = null;

    // Création de la table
    try {
        $base = new PDO('mysql:host=localhost; dbname=basephp', 'root', '');
    }
    catch(exception $e) {
        die('Erreur '.$e->getMessage());
    }
```

```
$base->exec("CREATE TABLE tablephp(id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id), prenom  
varchar(50), nom varchar(50), comteurvisite smallint, dernierevisite timestamp");  
  
    // Ajout de données dans la table  
    $base->exec("INSERT INTO tablephp(prenom, nom, comteurvisite, dernierevisite)  
VALUE('Pierre', 'Dubur', 34, NOW())");  
    $base->exec("INSERT INTO tablephp(prenom, nom, comteurvisite, dernierevisite)  
VALUE('Chantal', 'Garnier', 128, NOW())");  
    $base->exec("INSERT INTO tablephp(prenom, nom, comteurvisite, dernierevisite)  
VALUE('Jean', 'Dupont', 2, NOW())");  
    $base->exec("INSERT INTO tablephp(prenom, nom, comteurvisite, dernierevisite)  
VALUE('Belle', 'Vercor', 45, NOW())");  
  
    $retour = $base->query('SELECT * FROM tablephp');  
echo "<table border>";  
while ($data = $retour->fetch()){  
    echo "<tr><td>".$data['prenom']."</td>";  
    echo "<td>".$data['nom']."</td>";  
    echo "<td>".$data['comteurvisite']."</td>";  
    echo "<td>".$data['dernierevisite']."</td></tr>";  
}  
echo "</table>";  
$base = null;  
?>
```

Ce code est accessible sur <http://www.mediaforma.com/orsys/creation-donnees.php>  
Placez-le dans le dossier www de Wamp Server

## Etape 2 – Lecture des données dans la table

```
<?php
    try {
        $base = new PDO('mysql:host=localhost; dbname=basephp', 'root', '');
    }
    catch(exception $e) {
        die('Erreur '.$e->getMessage());
    }
    $retour = $base->query('SELECT * FROM tablephp');
    echo "<table border>";
    while ($data = $retour->fetch()) {
        echo "<tr><td>".$data['prenom']."</td>";
        echo "<td>".$data['nom']."</td>";
        echo "<td>".$data['compteurvisite']."</td>";
        echo "<td>".$data['dernierevisite']."</td></tr>";
    }
    echo "</table>";
    $base = null;
?>
```

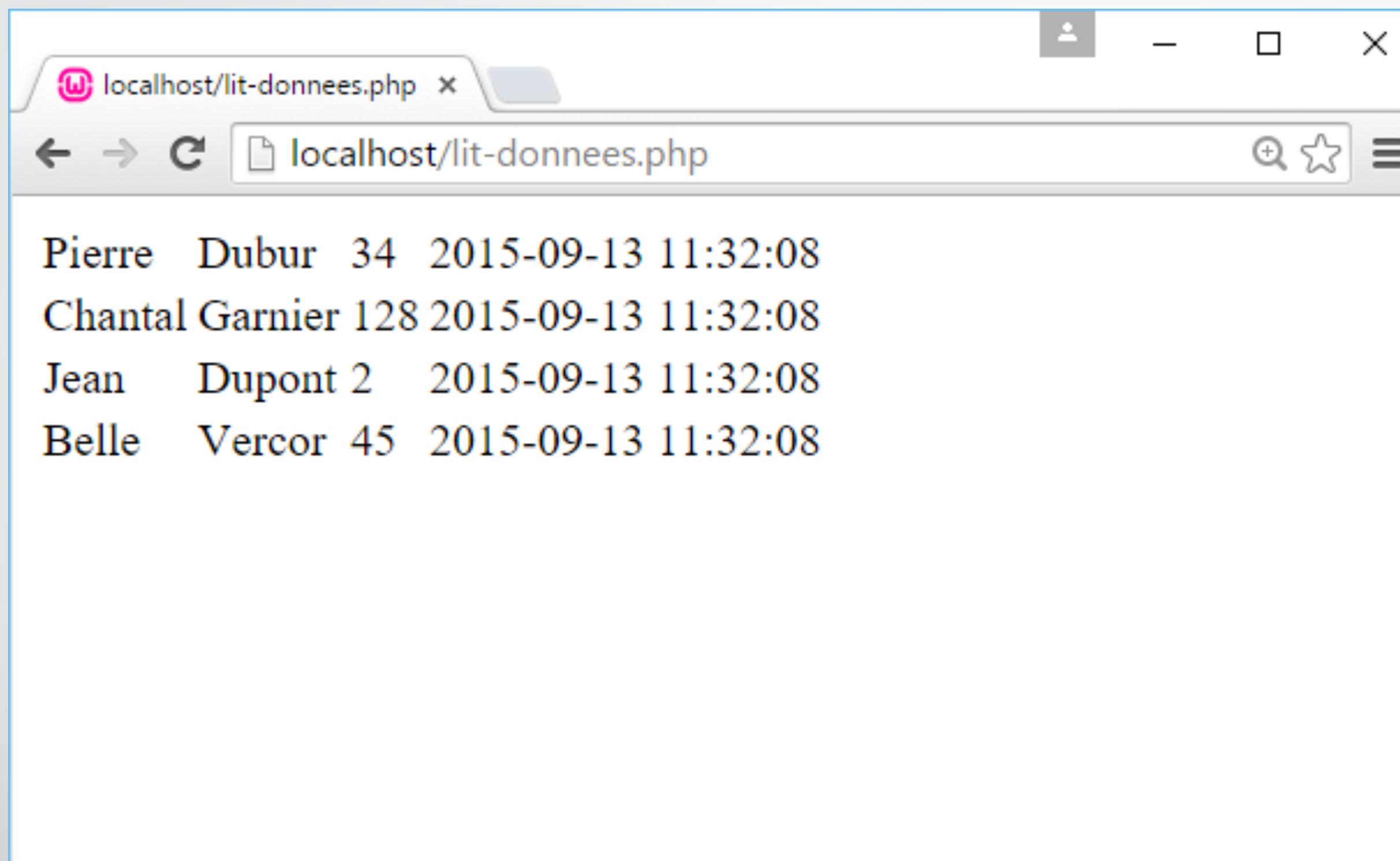
Ce code est accessible sur <http://www.mediaforma.com/orsys/lit-donnees.php>

Placez-le dans le dossier www de Wamp Server

## **Etape 3 – Obtention des données de la table en jQuery/AJAX**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Requête AJAX $.get()</title>
        <style>
            #zone {
                width: 300px;
                height: 315px;
                border-style: solid;
                border-width: 3px;
                border-color: black;
            }
        </style>
    </head>
    <body>
        <script src="jquery.js"></script>
        <h2>Récupération de données dans une base de données</h2>
        <button id="lecture">Lecture des données</button>
        <div id="zone"></div>
        <script>
            $(function() {
                $('#lecture').on('click', function() {
                    $.get('lit-donnees.php', function(data) {
                        $('#zone').html(data);
                    });
                });
            });
        </script>
    </body>
</html>
```

Voici le résultat :





La formation est terminée