

# Introduction

Historique	6
Organismes de normalisation	7
Définition du HTML5	8
Principe du HTML5	8
Quand choisir HTML5/CSS3	8

# Le balisage

Structure d'une page	10
Contenus & nouvelle balise	11
Exemple: utilisation nouveaux éléments	14

# Récapitulatif balise

Nouveaux éléments	26
Nouveaux attributs	26
Changements balises & attributs	29
Obsolescence d'attributs	30

# Les formulaires

Composants d'un formulaire	34
Place holder, required, pattern & validation	35
Les nouveaux types <input>	39

## CSS 3

Préfixe navigateur	50
Les bordures	50
Les ombrages	52
Transparence & opacité	54
Les arrières plans multiples	55
Les dégradés	57
Les transformations	60
Les transitions	62
Les animations	65

## Le multimédia

Audio	72
Vidéo	74



# Introduction

Historique

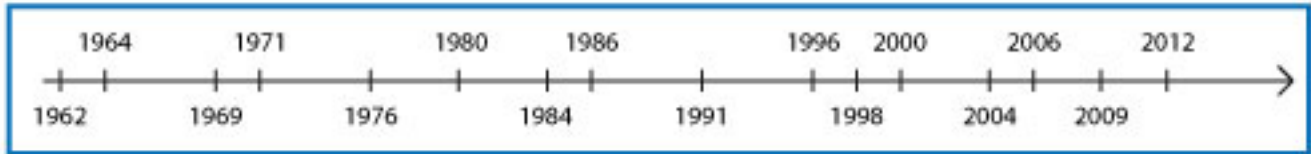
Organismes de normalisation

Définition du HTML5

Principe du HTML5

Quand choisir HTML5/CSS3

## Historique



- **1962** : US Air Force commande la création d'un réseau de communication militaire pouvant résister à une attaque nucléaire.
- **1964** : [Paul Baran](#) crée un réseau en forme de toile, évitant ainsi un système central vulnérable.
- **1969** : ARPANET est créé pour relier 4 universités, indépendamment d'un objectif militaire. (Présentation public en 1972)
- **1971** : 1er Mail électronique. Le caractère @ est déjà présent. (Amélioration en 1972 avec boîte mail possédant un système d'archivage, de tri et « faire suivre »)
- **1976** : Déploiement du protocole **TCP** (débuté en 1972) permettant la transmission de paquet et la gestion des erreurs. Il sera scindé en 2 protocoles en 1978 : **TCP/IP**
- **1980** : [Tim Berners-lee](#) et [Robert Caillau](#) mettent au point un système de navigation Hypertext. (ancêtre des navigateurs)
- **1984** : Mise en place du système de nommage **DNS**. (Domain Name System-système de noms de domaine) C'est un service permettant de traduire un nom de domaine en informations de plusieurs types qui y sont associées, notamment en adresses IP de la machine portant ce nom.
- **1986** : Création du langage **SGML** (Standard General Markup Language) pour structurer des contenus divers, considéré comme le 1er langage à balise.
- **1991** : [Tim Berners-lee](#) met au point le protocole **HTTP** (Hyper Text Transfer Protocol), ainsi que le langage **HTML** (Hyper Text Markup Language). Naissance du **World Wide Web** (WWW).
- **1996** :
  - 1ère spécification HTML 2.0 par le **W3C** (World Wide Web Consortium). le HTML 1.0 n'a jamais vraiment existé.
  - Apparition du **CSS** (Cascade Style Sheet – feuille de style) créé par [Håkon Wium Lie](#).
- **De 1997 à 1999** :
  - Spécification HTML 3.2, 4.0 et 4.01
  - Spécification CSS 2.0
  - Création du **XML** (eXtensible Markup Language)
- **2000** : Le W3C lance le XHTML 1.0, celui-ci possède exactement le même contenu que le html 4.01 la différence tient sur la syntaxe, le xhtml suivait les règles du xml. Exemple tous les attributs d'une balise ne s'exprime plus qu'en minuscule. (cette sortie coïncida avec l'essor des navigateurs compatible avec CSS).
- **2001** : Le W3C recommande le XHTML 1.1, celui-ci est entièrement en XML, cependant gros problème le navigateur le plus populaire du moment ne peut pas l'interpréter (ie explorer). Le W3C perd pied avec la réalité des publications web. Ceci ne l'empêche pas de se lancer dans le XHTML 2.0 cependant celui-ci possède plusieurs problèmes techniques et ne répond toujours pas aux besoins développeurs du moment.
- **2004** : Scission au sein du W3C. Les représentants d'Opéra, Apple et Mozilla ne sont pas en

accord avec les priorités du W3C. Proposition de reprise du développement HTML pour création d'application web mais celle-ci est rejetée. Les représentants d'Opéra (dont *Ian Hickson*) et autres forment leur propre groupe WHATWG (Web Hypertext Application Technology Working Group)

- **2006** : *Tim Berners-Lee* admet que la migration du HTML vers XML était une erreur. Quelques mois plus tard le W3C planche sur la version HTML 5 (avec espace) et continue cependant sur le XHTML 2.0. De son côté WHATWG travaille sur le HTML5 (sans espace).
- **2007** : Le W3C accepte les propositions de recommandations du WHATWG sur le HTML5 (sans espace).
- **2009** : Le format XHTML 2.0 est définitivement mort
- **2012** : La spécification HTML5 doit devenir « recommandation candidate », c'est-à-dire fin prête dans le discours normatif.

## Organismes de normalisation

### IETF

L'Internet Engineering Task Force, abrégée IETF, littéralement traduit de l'anglais en « Détachement d'ingénierie d'Internet » est un groupe informel, international, ouvert à tout individu, qui participe à l'élaboration de standards Internet. L'IETF produit la plupart des nouveaux standards d'Internet. L'IETF est un groupe informel, sans statut, sans membre, sans adhésion. Le travail technique est accompli dans une centaine de groupes de travail. En fait, un groupe est généralement constitué d'une liste de courrier électronique. L'IETF tient trois réunions par année.

### W3C

Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de normalisation à but non-lucratif, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML, RDF, SPARQL, CSS, PNG, SVG et SOAP. Fonctionnant comme un consortium international, il regroupe en 2012, 378 entreprises partenaires.

Le W3C a été fondé par Tim Berners-Lee après qu'il eut quitté le CERN en octobre 1994. Le W3C a été fondé au MIT/LCS (Massachusetts Institute of Technology / Laboratory for Computer Science) avec le soutien de l'organisme de défense américain DARPA et de la Commission européenne.

### WHATWG

Le Web Hypertext Application Technology Working Group (ou WHATWG) est une collaboration non officielle des différents développeurs de navigateurs web ayant pour but le développement de nouvelles technologies destinées à faciliter l'écriture et le déploiement d'applications à travers le Web. La liste de diffusion du groupe de travail est publique et ouverte à tous. La Mozilla Foundation, Opera Software et Apple, Inc. en sont les premiers contributeurs.

Ce groupe de travail se limite aux technologies qu'il estime imprésentables dans les navigateurs Web sur la base des implémentations actuelles, et particulièrement de celles d'Internet Explorer. Il se présente notamment comme une réponse à la lenteur supposée du développement des standards par le W3C et au caractère supposé trop fermé de son processus interne d'élaboration de spécification. Cependant, de nombreux participants à ce projet sont également des membres actifs du W3C, et le nouveau groupe de travail HTML du W3C a adopté en 2007 les propositions du WHATWG comme base de travail d'un futur HTML5.

## Définition du HTML5

HTML5 est une combinaison de nouvelles balises HTML, de propriété CSS3, de JavaScript et de plusieurs technologies associées mais structurellement séparées de la spécification HTML5.

## Principe du HTML5

- HTML5 devra supporter le contenu déjà existant (pas d'an zéro du HTML5)
- S'il existe une façon répandue d'accomplir une tâche chez les webdesigners (même mauvaise), celle-ci doit être codifiée en HTML5 (« si ce n'est pas cassé, on répare pas »).
- En cas de conflit on privilégie d'abord l'utilisateur, les webdesigners, les implémenteurs (dans les navigateurs), les spécificateurs et enfin la beauté du code.

## Quand choisir HTML5/CSS3

Le W3C recommande dès à présent aux développeurs Web à utiliser HTML 5. De plus la majorité des navigateurs semblent avoir intégré les spécifications CSS3 et celui est le mieux à même à répondre aux nouveaux supports de diffusion du web (tablettes, téléphone...).

La question devrait plutôt être quand ne pas l'utiliser. Il existe encore une majorité d'administration travaillant encore avec des versions d'internet explorer antérieure à la version 9.0 dans ce cas mieux vaut rester dans les recommandations de L'HTML 4.0.1. Pour le reste foncez vers le HTML5.

# Le balisage

Structure d'une page

Contenus & nouvelle balise

Exemple: utilisation des nouveaux éléments



## Structure d'une page

### Les éléments de base

Le langage HTML5 est une amélioration du langage HTML4, avec des simplifications par rapport à la version XHTML. Tout document peut donc débuter par la déclaration du doctype puis est suivi de l'élément racine html qui inclut les éléments head et body.

#### Doctype

La déclaration du doctype est traditionnellement utilisée pour spécifier le type de balisage du document.

Celui de XHTML 1.0 était le suivant :

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

Celui de HTML 4.01 était :

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

Un des principes d'HTML5 étant la rétrocompatibilité avec les anciennes versions, sa définition se résume à un simple :

```
<!DOCTYPE html>
```

#### Html

L'élément **<html>** est l'élément racine du document. Il est le parent de tous les autres, soit **<head>** et **<body>**. Celui-ci possède des attributs dont le plus utiles est **lang**. Celui-ci indique la langue utilisée par défaut dans la page. Cette valeur sera reconnue par les moteurs de recherche pour leur permettre d'indexer les pages du site en effectuant un tri par langue.

```
<html lang= « fr » >
```

#### Head

Cette section donne quelques informations générales sur la page, comme son titre et l'encodage de la page (pour la gestion des caractères spéciaux). Ces informations sont les 2 seules obligatoires dans cette section.

Si la balise **<title>** n'a pas changé, la déclaration de l'encodage a elle été simplifiée. Si avant nous avions :

```
<meta http-equiv=»Content-Type» content=»text/html; charset=utf-8» />
```

En HTML5 :

```
<meta charset= « utf-8 »>
```

## Minimum Page HTML5

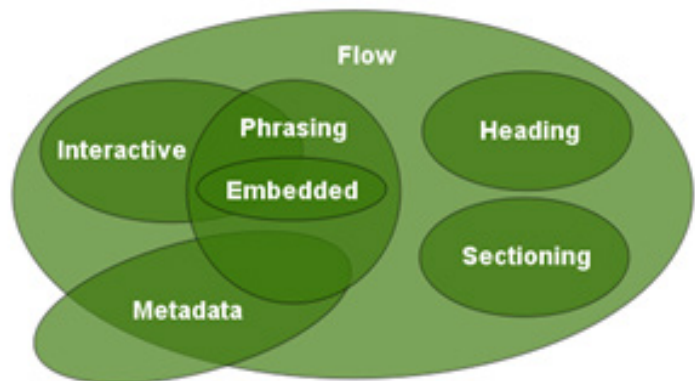
```
<!DOCTYPE html>
<html lang= « fr » >
  <head>
    <meta charset= « utf-8 »>
    <title>page simple HTML5</title>
  </head>
  <body>
    <!--ici contenu de ma page html5 -->
  </body>
</html>
```

## Contenus & nouvelles balises

### Les contenus

Dans les versions précédentes de HTML les contenus étaient divisés en 2 grandes catégories, « en lignes » ou en « bloc », celui-ci est remplacé par un nouveau schéma ; les éléments HTML sont regroupés selon les catégories suivantes :

- Contenu Metadonnées (*Metadata*)
- Contenu de flux (*Flow*)
- Contenu de section (*Sectioning*)
- Contenu d'en-tête (*Heading*)
- Contenu de phrasé (*Phrasing*)
- Contenu embarqué (*Embedded*)
- Contenu interactif (*Interactive*)



Il est à noter que certains éléments peuvent apparaître dans plusieurs catégories.

### Contenu de métadonnées

Les métadonnées déterminent la présentation ou le comportement du contenu de la page. Elles servent aussi à configurer les relations entre ce document et d'autres (exemple : balise **<meta>** contenant des mots clé ou bien un descriptif de la page).

Les modifications apportées par HTML5 concernent les balises **<link>** et **<script>**. Concernant la balise script il n'est plus nécessaire d'indiquer le **type**. Celui-ci sera par principe considéré comme étant du javascript.

```
<script src= « file/js » ></script>
```

Il en va de même pour l'appel des feuilles de styles, l'attribut réellement nécessaire étant le média sur lequel il s'applique :

```
<link rel= « stylesheet » href= « styles.css » media= « screen » />
```

Les différents média : all, print, screen, tv, aural, braille, embossed, handled, projection, tty

## Contenu de flux

Le contenu de flux représente les éléments qui sont considérés comme formant l'ensemble du contenu d'une page web. Un flux est en règle générale un texte ou un fichier embarqué comme une image ou bien une vidéo.

Cette catégorie possède plusieurs nouveaux éléments :

- **header** : spécifie une introduction, ou un groupe d'éléments de navigation pour le document.
- **footer** : définit le pied de page d'un article ou un document. Contient généralement le nom de l'auteur, la date à laquelle le document a été écrit et / ou ses coordonnées.

*Exemple* : voir exemple complet page suivante

- **figure** : définit des images, des diagrammes, des photos, du code, etc.
- **figcaption** : légende pour la balise <figure>.

*Exemple* : Création du document *00-figure.html*

Entre les balises **<body>** insérer le code suivant:

```
<figure >
  
  <figcaption>Photo promotion du film <strong>Les oiseaux</strong></figcaption>
</figure>
```



- **canvas** : utilisé pour afficher des éléments graphiques, il faut utiliser un script pour l'animer.
- Exemple : ces balises feront l'objet d'un chapitre complet

## Contenu de section

Ceci est une nouvelle catégorie. Le W3C décrit cette catégorie comme « définissant la portée des en-têtes et des pieds de pages ». Un contenu de section est un sous-ensemble d'un contenu de flux. Elle contient 4 nouveaux éléments HTML5 :

- **section** : définit les sections dans un document. Tels que les chapitres, en-têtes, pieds de page, ou toutes autres sections du document.
- **article** : partie indépendante du site, comme un commentaire.
- **aside** : associé à la balise qui le précède. Il pourrait être utilisé par exemple pour faire référence à des sujets complémentaires ou pour définir un terme spécifique.
- **nav** : définit une section dans la navigation. Il contient les liens utiles à la navigation

Exemple : voir exemple global

## Contenu d'en-tête

Contient tous les éléments d'en-tête standard comme **<h1>**, **<h2>** et ainsi de suite. En plus HTML5 comprend l'élément :

- **hgroup** : permet de définir des groupes de titres. Celui-ci ne peut contenir que les éléments de **<h1>** à **<h6>**. Il sert essentiellement à définir un plan, une table des matières.

*Exemple* : Création du document *01-hgroup.html*

Entre les balises **<body>** insérer le code suivant:

```
<hgroup>
  <h1>La vie des castors</h1>
  <h2>...ou la passionnante aventure d'une espèce en danger</h2>
</hgroup>
```

## La vie des castors

...ou la passionnante aventure d'une espèce en danger

### Contenu de phrasé

Ce contenu est le texte du document, y compris les mises en évidence de passage à l'intérieur d'un paragraphe. Nouvelles balise HTML5 :

- **mark** : définit un texte marqué.

*Exemple* : Création du document [02-mark.html](#).

Entre les balises **<body>** insérer le code suivant:

```
<p>L'avantage d'avoir de <strong>bonnes</strong> chaussures de marche de randonnée devient
<em>extrêmement</em> clair <mark>après 3 jours de marche</mark>.</p>
```

L'avantage d'avoir de **bonnes** chaussures de marche de randonnée devient *extrêmement* clair après 3 jours de marche

### Contenu embarqué

Un contenu embarqué importe une autre source dans la page, comme une image, une musique ou bien une vidéo. Ce contenu contient de nouvelles balises HTML5 permettant une alternative à FLASH :

- **audio** : pour définir un son, comme la musique ou les autres flux audio (streaming).
- **video** : Insérer un contenu vidéo en streaming.
- **track** : Insérer un sous-titre (au format WebVTT) à une vidéo affichée avec la balise video .
- **embed** : définit un contenu incorporé, comme un plug in.

*Exemple* : Ces contenus font l'objet du chapitre 7

### Contenu interactif

L'une des balises les plus basiques **<a>**, est considérée comme un élément interactif ainsi que les éléments de formulaire tel **<textarea>** ou encore **<button>**.

La balise **<a>** subit un profond changement. En effet si dans les versions précédentes cette balise est considérée inline :

Bienvenue !

```
<h2><a href= http://www.afci.fr> AFCl NEWSOFT </a></h2>
```

```
<p><a href= http://www.afci.fr> formation en micro-informatique diplomate 3d web pao bureau-
tique infographie video centre de formation informatique diplomante internet web </a></p>
```

En HTML5, il est possible d'envelopper plusieurs éléments dans un seul élément **<a>**

*Exemple* : Création du document [03-a\\_custom.html](#).

Entre les balises **<body>** insérer le code suivant:

```
<a href= http://www.afci.fr>
  <h2> APCI NEWSOFT </h2>
  <p>formation en micro-informatique diplomate 3d web pao bureautique infographie
video centre de formation informatique diplomante internet web.</p>
</a>
```

## Exemple: utilisation des nouveaux éléments

### Création page modèle

Nous allons dans cette partie créer une page modèle dont nous nous resservirons dans les prochains exercices. Celle-ci contiendra un simple en tête incluant une image ainsi qu'un titre et un sous-titre.



Création du fichier html de base : *modele\_page.html* (encodage UTF-8 (sans BOM))

```
<!DOCTYPE html>
<html lang= « fr » >
  <head>
    <meta charset= « utf-8 »>
    <title>Modèle page HTML5</title>
  </head>
  <body>
  </body>
</html>
```

Création de la feuille de style associé : *assets/css/base.css*

```
@charset «utf-8»;
/* CSS Document */
body {
  font: normal 90% «Trebuchet MS»,Verdana,»Lucida Grande»,Tahoma,Arial,Helvetica,Sans-Serif;
  color: #444;
```

```
background:#f4f4f4;
padding:0 0 2em 0;
margin:0;
}
```

Nous devons à présent attacher cette feuille de style à notre page html. Dans l'entête de notre page inclure le lien suivant:

```
<link rel="stylesheet" href="assets/css/base.css" media="screen" />
```

Enregistrer et visualiser le résultat.

La balise **<header>** :

La balise **< header>** (ainsi que **<footer>**) ont été créées afin de donner au créateur et au développeur des options de balisage sémantique plus précis inhérent à la structure d'un document. Ces balises donnent une indication plus précise du contenu d'une section (auparavant celle-ci était contenu dans un **<div>** dont l'id permettait de connaître son contenu ex : **<div id="header">**).

Dans le corps du document HTML :

```
<body>
  <header>
    
    <h1>HTML5</h1>
    <h2>Page modèle exercice</h2>
  </header>
</body>
```



Dans la feuille de style :

```
header{
  color: #ee662a;
  background:white;
  border-bottom:2px dotted #ccc;
  margin:0;
  padding:0.8em;
}

header h1{
  font-size: 4em;
  font-weight: normal;
  margin:0;
}

header h2{
  font-size: 1.5em;
  font-weight:bold;
  margin: 0 0 2.5em 10em;
  color: #e74b25;
}
```

```
header img{
  float:left;
  margin-right:3em;
}
```

Si l'on enregistre le document et qu'on le visualise dans les navigateurs les plus récents aucun problème (opéra, safari, chrome et ie9). Cependant si l'on essaie de le visualiser dans un navigateur antérieur celui-ci ne reconnaît pas la balise header (voir application **ietester**).

#### Correctif IE

Pour pallier cette erreur nous allons procéder en 2 temps :

*Création de la feuille de style associé : [assets/css/correctif\\_balise.css](#)*

```
@charset «utf-8»;
/* CSS Document */
header, section, aside, nav, footer, figure, figcaption{
  display:block;
}
```

*Puis création d'un document javascript : [assets/js/correctif.js](#)*

```
// JavaScript Document
document.createElement('header');
document.createElement('footer');
document.createElement('nav');
document.createElement('aside');
document.createElement('section');
document.createElement('figure');
document.createElement('figcaption');
```

Enfin il ne nous reste plus qu'à inclure ces 2 nouveaux fichiers au sein de l'en-tête de notre document :

```
<link rel="stylesheet" href="assets/css/correctif_balise.css" media="screen" />
<script src="assets/js/correctif.js"></script>
```

*Enregistrer et visualiser le résultat.*

A présent nous avons notre document de référence pour l'ensemble de nos exercices.

#### Exemple architecture de site:

Nous allons à présent dans l'exemple qui suit créer une page HTML contenant l'ensemble des nouvelles balises.

*Création du fichier html: [04-architectureSite.html](#) à partir de la page [modele\\_page.html](#)*


*Création de la feuille de style associé : [assets/css/styles\\_architecture.css](#)*

Puis modifier le titre de la page HTML («architecture site») et inclure la nouvelle feuille css:

```
<link rel="stylesheet" href="assets/css/styles_architecture.css" media="screen" />
```



Le contenu de la page sera défini de la manière suivante :



# HTML5

Page modèle exercice

header

ACCUEIL

A PROPOS

DIAPORAMA

CONTACT

nav

Recommendations

aside

Donec libero eros, elementum vitae lacinia bibendum, malesuada et est. Pellentesque elit odio, lobortis non commodo quis, varius sed justo. Morbi aliquam purus eu leo bibendum et aliquet eros rhoncus. Ut feugiat porttitor magna, in mollis turpis ultricies non. Maecenas at tortor eros, et feugiat dolor. Pellentesque tincidunt dignissim commodo. Proin mi tortor, tincidunt sit amet mollis non, rhoncus ut est. Aliquam augue odio, lacinia sit amet hendrerit vel, bibendum eu arcu. Etiam dui leo, feugiat a molestie vel, tristique a ante.

Quisque fermentum scelerisque lacus eget sollicitudin. Nunc feugiat molestie odio vel accumsan. Etiam non est tortor. Morbi arcu odio, lacinia vitae aliquet in, vulputate vel ante. Curabitur mollis sapien non nunc semper sit amet ultricies nibh placerat. Cras facilis varius consectetur. Phasellus nibh odio, luctus auctor mattis vitae, aliquam et arcu.

## Mission du HTML5

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

What's new?

header

article

Par David, le 05 septembre 2012

time

Donec libero eros, elementum vitae lacinia bibendum, malesuada et est. Pellentesque elit odio, lobortis non commodo quis, varius sed justo. Morbi aliquam purus eu leo bibendum et aliquet eros rhoncus. Ut feugiat porttitor magna, in mollis turpis ultricies non. Maecenas at tortor eros, et feugiat dolor. Pellentesque tincidunt dignissim commodo. Proin mi tortor, tincidunt sit amet mollis non, rhoncus ut est. Aliquam augue odio, lacinia sit amet hendrerit vel, bibendum eu arcu. Etiam dui leo, feugiat a molestie vel, tristique a ante.

Plus d'info:

aside

Les dernières recommandations du W3C.

lien

Fusce fermentum congue velit ac lobortis. Donec id erat a mauris sagittis dignissim id ut lorem. Suspendisse dignissim, massa sit amet vehicula mattis, nisi mauris rhoncus felis, ut sodales odio turpis a metus. Maecenas elementum semper enim. Quisque volutpat lectus at turpis porta eget volutpat arcu auctor. Sed egestas vehicula ipsum, quis tristique nibh et orci libero, a scelerisque augue. Suspendisse vestibulum cursus malesuada. Vivamus varius mauris vitae lorem auctor placerat. Vivamus dignissim suscipit pharetra. Proin et nulla elit.

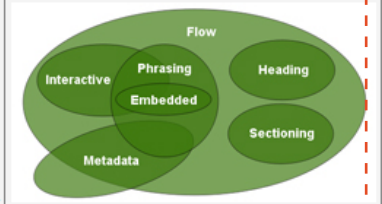


figure + figcaption

Copyright 2012 APCI NEWSOFT

footer

Copyright APCI NEWSOFT 2012

En vous inscrivant ou en utilisant ce site, vous acceptez notre contrat utilisateur et notre charte sur le Respect de la vie privée

footer

La balise `<nav>` :

ACCUEIL

A PROPOS

DIAPORAMA

CONTACT

Nous allons à présent créer notre menu de navigation. Sous la balise `</header>` inclure le code suivant:



```
<nav>
  <ul>
    <li><a href="#">ACCUEIL</a></li>
    <li><a href="#">A PROPOS</a></li>
    <li><a href="#">DIAPORAMA</a></li>
    <li><a href="#">CONTACT</a></li>
  </ul>
</nav>
```

Nous définissons à présent dans la feuille de style associée : *styles\_architecture.css*

```
/* navigation */
nav {
  background-color:#f2662a;
  height:35px;
}

nav li{
  float:left;
  width:140px;
  height:35px;
  background-color:black;
  text-align: center;
  border-left: 1px white solid;
  border-right: 1px white solid;
  line-height:35px;
  list-style:none;
}

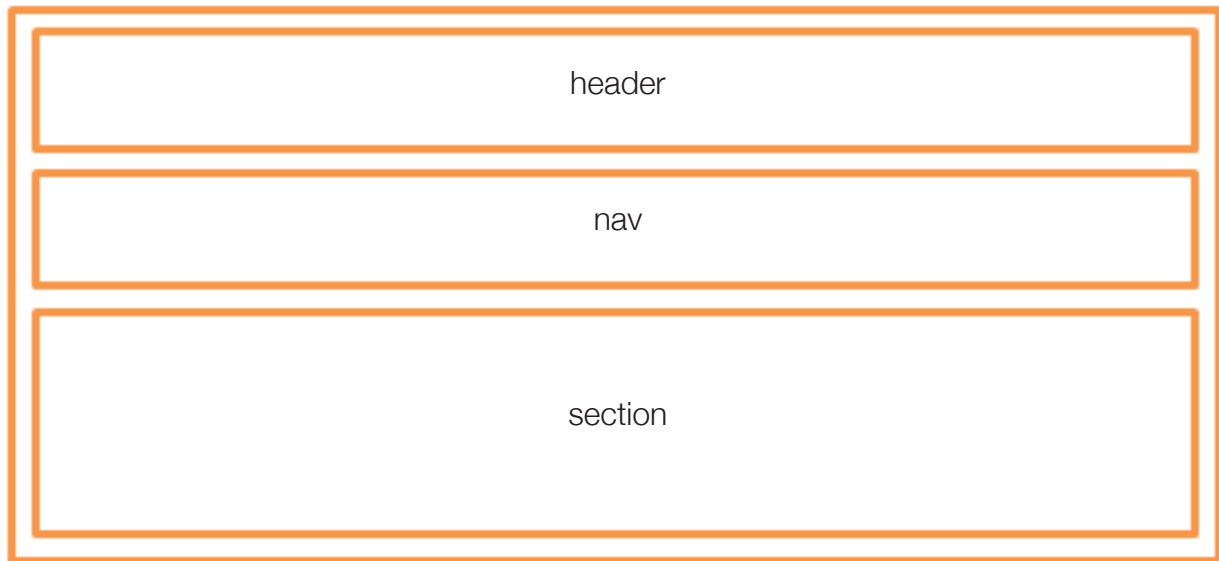
nav li a{
  color:white;
  text-decoration:none;
  display:block;
}

nav li a:hover{
  background-color:#f0462b;
  color:#ebebea;
}
```

*Enregistrer et visualiser le résultat.*

Il existe une alternative à la balise **<nav>** qui est la balise **<menu>**. Cependant celle-ci avait été déclarée obsolète pour le XHTML. Celle-ci est censée permettre la création rapide de menu déroulant, cependant les navigateurs gèrent très peu cette balise, surtout qu'il est possible de réaliser le même effet avec liste à puces et feuilles de style.

La balise <section> :



Une section est définie comme étant un regroupement thématique de contenu. Dans notre exemple celui-ci sera notre conteneur principal.

Ajoutons le contenu de notre section dans la page architecture (aidez-vous des textes lorem ipsum pour remplir celle-ci). A la suite de la balise **</nav>**

```
<section>
  <h2>Mission du HTML5</h2>
  <p>Lorem ipsum ... bibendum.</p>
  <p>Quisque ... arcu.</p>
  <h2>What's new?</h2>
  <p>Par David, posté le 05 Septembre 2012</p>
  
  <p>Les différents conenus HTML5</p>
  <p>Donec ... ante.</p>
  <p>Fusce ... elit.</p>
  <p>Quisque ... purus.</p>
</section>
```

Puis définition dans CSS associé

```
/* section */
section {
  width: 600px;
  float: left;
}

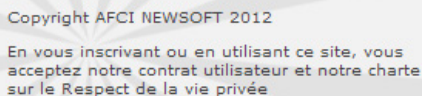
section p, h1, h2 {
  margin-left: 20px;
  margin-right: 20px;
}
```

```
section h2 {  
    margin-top:15px;  
}
```

*Enregistrer et visualiser le résultat.*

On peut remarquer simplement que section se comporte de la même manière qu'un **<div>**.

La balise **<footer>** :

A screenshot of a website footer. It has a light gray background with a subtle sunburst pattern. The text is in a small, sans-serif font. It reads: "Copyright AFCI NEWSOFT 2012" followed by "En vous inscrivant ou en utilisant ce site, vous acceptez notre contrat utilisateur et notre charte sur le Respect de la vie privée".

Copyright AFCI NEWSOFT 2012  
En vous inscrivant ou en utilisant ce site, vous acceptez notre contrat utilisateur et notre charte sur le Respect de la vie privée

Après la balise section nous ajoutons un pied de page sur l'ensemble du document.

```
<footer>  
    <p>Copyright AFCI NEWSOFT 2012</p>  
    <p>En vous enregistrant ou en utilisant ce site, vous acceptez notre contrat utilisateur et notre  
charte de Respect de la vie privée.</p>  
</footer>
```

Le CSS associé :

```
footer{  
    clear: both;  
    width: 400px;  
    margin-left: 20px;  
    font-size: 10px;  
    background: #f2f2f2 url(file:../images/footer_background.jpg) no-repeat left;  
    height: 118px;  
    padding-top: 10px;  
    border-top:#e2e2e2 solid 2px;  
    border-bottom:#e2e2e2 solid 2px;  
}
```

*Enregistrer et visualiser le résultat.*

La balise **<article>** :

La définition du W3C indique qu'il s'agit d'une composition autonome, indépendante du reste, dans un document, une application ou d'un site, et qu'elle est par principe distribuable ou réutilisable en soi. Il peut par exemple s'agir d'un post dans un forum, d'un article de presse, d'une entrée, d'un blog, d'un commentaire soumis par un utilisateur ou tout autre contenu autonome.

Le concept clé est qu'un article est un contenu *en soi*. Celui-ci peut être facilement republié dans différents contextes. Il pourra être multi-diffusé dans divers formats, tel qu'un flux RSS, dans un email, via des applications mobiles.

Repérer au milieu de votre **<section>** le 2ème **<h2>** et entourer ce contenu par la balise article.

```
<article>
  <h2>What's new?</h2>
  ...
  <p><small>© copyright 2012 AFCI NEWSOFT.</small></p>
</article>
```

En HTML5 un article peut posséder son propre en-tête et pied de page

```
<article>
  <header>
    <h2>What's new?</h2>
    <p>Par David, posté le 05 Septembre 2012</p>
  </header>
  ...
  <footer>
    <p>© copyright 2012 AFCI NEWSOFT</p>
  </footer>
</article>
```

Et dans le fichier CSS

```
/* article */
article header{
  background:none;
  height:auto;
  border-bottom:none;
}
article header h2{
  color:black;
  margin:0;
}

article header p{
  color: gray;
  margin: 0;
}
article footer{
  background:none;
  height:auto;
  border:none;
}
article footer p{
  font-size:10px;
}
```

Enregistrer et visualiser le résultat.

### La balise **<aside>** :

Cette balise peut être utilisée de 2 manières. L'une correspond au placement classique d'une barre latérale dans une page web. La seconde concerne une zone de contenu complémentaire, non indépendant, à l'intérieur d'une **<section>**.

1er cas :



Dans votre document html après la balise **</nav>**

```
<aside id=»main_aside»>
  <h3>Dernières recommandations</h3>
  <p>Nullam ... egestas.</p>
  <p>Quisque ..libero.</p>
</aside>
```

Dans le CSS :

```
/* aside */
#main_aside{
  margin-top:15px;
  float: left;
  width: 300px;
  background-color: #378059;
  padding:7px;
  color:white;
}
```

*Enregistrer et visualiser le résultat.*

2ème cas :

Dans la section **<article>** juste avant le 2ème gros paragraphe ajouter le code suivant

```
<aside id=»article_aside»>
  <h3>Plus d'infos ?</h3>
  <p>Lisez les recommandations du W3C. <a href=»#»>Lien.</a></p>
</aside>
```

Dans le CSS :

```
#article_aside{
  float: left;
  width: 12em; /* 1 em est la hauteur de caractère de la police utilisée, unité relative */
  background: #F9A890;
  padding: 10px;
  margin: 0 10px;
}
```

```
#article_aside h3{
    color:#378059;
    margin:5px 0 5px 0;
}
#article_aside p{
    margin: 0;
}
```

Enregistrer et visualiser le résultat.

#### La balise **<figure>**:

Parmi les nouveaux éléments certains donnent davantage de sens à nos pages web. Par exemple **<figure>** et **<figcaption>** permettent d'identifier des images et des légendes associées à l'intérieur de notre contenu.

Dans la partie **<article>** localiser la balise image et modifier le contenu pour obtenir.

```
<figure>
    
    <figcaption>Les différents contenus HTML5</figcaption>
</figure>
```

Dans le CSS :

```
/*figure */
figure {
    float: right;
    border: 1px solid gray;
    padding: 0.25em;
    margin: 0 1.5em 1.5em 0;
}

figcaption{
    text-align:center;
    font: italic 0.9em Georgia, «Times New Roman», Times, serif;
}
```

Enregistrer et visualiser le résultat.

#### La balise **<time>** :

Au début de **<article>** nous pouvons trouver une date de publication. Le problème avec ce format est que l'on ne l'indique pas dans un format lisible par un système informatique. HTML5 ajoute l'élément **<time>**. L'attribut `datetime` formaté selon l'année, le mois et le jour. S'il n'est pas obligatoire celui-ci sera reconnu par un système informatique. De plus si l'on publie le contenu, il est également possible d'ajouter l'attribut `pubdate`.

```
<p>Par David, posté le <time datetime="2012-09-05" pubdate>05 Septembre 2012</time></p>
```



# Récapitulatif balise

Nouveaux éléments

Nouveaux attributs

Changement balises et attributs

Obsolescence d'attributs



## Nouveaux éléments

- **section** : définit les sections dans un document. Tels que les chapitres, en-têtes, pieds de page, ou toutes autres sections du document.
- **article** : partie indépendante du site, comme un commentaire.
- **aside** : associé à la balise qui le précède.
- **header** : spécifie une introduction, ou un groupe d'éléments de navigation pour le document.
- **footer** : définit le pied de page d'un article ou un document. Contient généralement le nom de l'auteur, la date à laquelle le document a été écrit et / ou ses coordonnées.
- **nav** : définit une section dans la navigation.
- **figure** : définit des images, des diagrammes, des photos, du code, etc.
- **figcaption** : légende pour la balise <figure>.
- **audio** : pour définir un son, comme la musique ou les autres flux audio (streaming).
- **video** : Insérer un contenu video en streaming.
- **track** : Insérer un sous-titre (au format WebVTT) à une vidéo affichée avec la balise video .
- **embed** : définit un contenu incorporé, comme un plug in.
- **mark** : définit un texte marqué.
- **meter** : Permet d'utiliser les mesures avec un minimum et maximum connus, pour afficher une jauge.
- **progress** : définit une barre de progression sur le travail en cours d'exécution.
- **time** : définit une date ou une heure, ou les deux. Cette balise a été abandonnée en octobre 2011 en faveur de la balise data 2 avant d'être réintroduite<sup>3</sup>.
- **canvas** : utilisé pour afficher des éléments graphiques, il faut utiliser un script pour l'animer.
- **command** : définit un bouton.
- **details** : précise les détails supplémentaires qui peuvent être masqués ou affichés sur demande.
- **keygen** : permet de générer une clé (sécurisé).
- **output** : représente le résultat d'un calcul.
- **ruby, rt et rp** : annotations Ruby.

## Nouveaux attributs

Pour la balise **<a>**:

- **media** : permet de spécifier pour quel media ou device il est optimisé.
- **type** : définit le MIME de la cible URL.

Pour la balise **<area>** :

- **hreflang** : spécifie le langage de l'url.

- **media** : permet de spécifier pour quel media ou appareil il est optimisé.
- **rel** : Indique la relation entre le document courant et l'URL cible.
- **type** : définit le MIME de la cible URL.

Pour la balise `<button>` :

- **autofocus** : Indique que le bouton doit avoir le focus pendant le chargement de la page.
- **form** : spécifie à quel formulaire le bouton appartient.
- **formaction** : Spécifie où envoyer le form-data quand un formulaire est soumis. Remplace l'attribut action du formulaire.
- **formenctype** : Indique comment le form-data doit être encodé avant d'être envoyé à un serveur. Remplace l'attribut enctype du formulaire.
- **formmethod** : définit comment il faut envoyer le form-data.
- **formnovalidate** : si présent, indique que le formulaire ne doit pas être validé quand il est envoyé.
- **formtarget** : spécifie où ouvrir/exécuter l'action.

Pour la balise `<fieldset>` :

- **name** : définit le nom du fieldset.
- **disabled** : désactive le fieldset.
- **form** : définit le formulaire du fieldset.

Pour la balise `<form>` :

- **autocomplete** : auto complétion.
- **novalidate** : si présent le formulaire n'est pas validé lorsqu'il est soumis.

Pour la balise `<html>` :

- **manifest** : URL de déclaration (manifest) des fichiers pour un usage hors ligne.

Pour la balise `<iframe>` :

- **sandbox** : Spécifie des restrictions sur le contenu de l'iframe
- **seamless** : Indique que l'iframe doit être parfaitement intégrée dans le document.
- **srcdoc** : le code HTML du document affiché dans l'iframe.

Pour la balise `<input>` :

- **autocomplete** : auto completion.
- **autofocus** : définit le focus lors du chargement de la page.
- **form** : spécifie à quel formulaire le champ appartient.
- **formaction** : Remplace l'attribut "action" du formulaire. Indique l'URL à laquelle envoyer les données du formulaire.
- **formenctype** : Remplace l'attribut "enctype" du formulaire. Indique comment la forme-données doit être encodé avant d'être envoyé au serveur.

- **formmethod** : Remplace l'attribut "method" du formulaire. Définit la méthode HTTP d'envoi des données à l'URL.
- **formnovalidate** : Remplace l'attribut "novalidate" du formulaire. S'il est présent le champ de saisie ne devrait pas être validé lors de son envoi.
- **formtarget** : Remplace l'attribut "target" du formulaire. Indique la fenêtre cible utilisée lorsque le formulaire est soumis.
- **height** : Définit la hauteur.
- **list** : Désigne un "datalist" contenant des options prédéfinies pour le champ de saisie.
- **max** : Indique la valeur maximale du champ d'entrée.
- **min** : Indique la valeur minimale du champ d'entrée.
- **multiple** : Si présent, l'utilisateur peut entrer plusieurs valeurs.
- **pattern** : Définit un motif.
- **placeholder** : Un conseil pour aider les utilisateurs à remplir le champ de saisie.
- **required** : Indique que la valeur du champ de saisie est nécessaire pour soumettre le formulaire.
- **step** : Indique l'intervalle entre les valeurs.

nouveaux types :

- **datetime**
- **datetime-local**
- **date**
- **month**
- **week**
- **time**
- **number**
- **range**
- **email**
- **url**
- **search**
- **color**

Pour la balise **<link>** :

- **sizes** : Définit la taille, hauteur et largeur.

Pour la balise **<menu>** :

- **label** : Label visible du menu.
- **type** : Définit le type de menu à afficher. La valeur par défaut est « list ».

Pour la balise **<meta>** :

- **charset** : Définit la table de caractères pour l'encodage de la page.

Pour la balise **<ol>** :

- **reversed** : si présent, change l'ordre d'affichage.

Pour la balise **<script>** :

- **async** : définit si le script doit être exécuté de manière asynchrone ou pas.

Pour la balise **<select>** :

- **autofocus** : active le focus sur cet élément.
- **form** : définit un ou plusieurs formulaires pour le "select".

Pour la balise **<style>** :

- **scoped** : Si présent, le style est appliqué uniquement sur le parent et les fils.

Pour la balise **<textarea>** :

- **autofocus** : focus l'élément textarea.
- **dirname** : Indique le nom du textarea.
- **form** : définit une ou plusieurs formulaires pour le textarea.
- **maxlength** : nombre maximum de caractères.
- **placeholder** : définit une astuce pour aider l'utilisateur.
- **required** : Indique que la valeur du champ de saisie est nécessaire.
- **wrap** : définit comment le texte est affiché dans le textarea.

Ainsi que les attributs globaux qui s'appliquent à toutes les balises :

- **contenteditable**
- **contextmenu**
- **data-\***
- **draggable**
- **hidden**
- **on\*** (gestionnaires d'événements)
- **spellcheck**

## Changements dans les balises et attributs

Cette section est vide, insuffisamment détaillée ou incomplète. Votre aide est la bienvenue !

Les balises suivantes sont supprimées car leurs effets étaient purement représentatifs, ce qui est le rôle de CSS.

- *basefont*,
- *big*,
- *center*,

- *font*,
- *strike*,
- *tt*,
- *u*

Les balises *frame*, *frameset* et *noframes* ont été supprimées elles aussi ; elles étaient déjà désuètes car elles créaient des problèmes d'accessibilité et d'utilisation pour l'utilisateur final.

Les balises suivantes sont elles aussi supprimées :

- *acronym* n'est plus incluse car elle créait beaucoup de confusions;
- *applet* est remplacé par *object*;
- *isindex*, car elle peut être remplacée par l'utilisation des contrôleurs de formes;
- *dir* est obsolète en faveur de *ul*

Enfin, *noscript* n'est fourni que dans la version HTML, il n'est pas inclus dans la version XML.

## Obsolescence d'attributs

- *charset*
- *coords*
- *rev*
- *shape*

*Sur la balise area*

- *nohref*

*Sur la balise head*

- *nohref*

*Sur la balise html*

- *version*

*Sur la balise iframe*

- *longdesc*

*Sur la balise img*

- *longdesc*

- *name* (préférer l'attribut *id*)

*Sur la balise link*

- *charset*
- *rev*
- *target*

*Sur la balise meta*

- scheme

*Sur la balise object*

- archive
- classid
- codebase
- declare
- standby

*Sur la balise param*

- type
- valuetype

*Sur la balise td*

- axis
- scope

*Sur la balise th*



# Les formulaires

Composant d'un formulaire

Place holder, required, pattern & validation

Les nouveaux types `<input>`



Les formulaires comptent parmi les plus anciens et les plus familiers exemples d'interactivité sur le web. L'élément FORM a été introduit dans le langage HTML en 1993 et les contrôles associés font partie de l'environnement quotidien de l'utilisateur.

Les nouveaux composants HTML5 ajoutent des fonctionnalités que les concepteurs web incorporaient traditionnellement par d'autres moyens, comme javascript ou flash. Il est courant dans un formulaire de rendre tel ou tel champ obligatoire, d'en vérifier le contenu avant de soumettre un formulaire. Ces fonctionnalités sont maintenant intégrées à HTML5.

## Composant d'un formulaire

Nous allons créer un simple formulaire permettant de recevoir un nom et un prénom :

A partir du modèle de page créé précédemment, insérer le formulaire suivant

*Création du fichier html: [chap\\_4/05-formulaire\\_simple.php](#) (attention à l'extension) à partir de la page [modele\\_page.html](#)*

Insérer après la balise `</header>`

```
<div id="wrap">
  <form id="mon_formulaire" action="" method="post">
    <label for="nom">Nom : </label><input type="text" name="nom" id="nom" /> <br/>
    <label for="prenom">Prénom : </label><input type="text" name="prenom"
id="prenom" />
    <br/>
    <input type="submit" value="Valider" />
  </form>
</div>
```

Dans le fichier de base CSS : [assets/css/base.css](#) ajoutez

```
/* formulaire */
#wrap {
  padding: 0 3em 3em 3em;
  margin:auto;
}
```

La balise formulaire contient 3 attributs :

- **id** : permet d'associer un style CSS ou bien servira d'identifiant pour un traitement (js ou autre)
- **action** : spécifie l'agent traitement le formulaire. Cette valeur est l'URL qui pointe vers un script serveur, celui-ci récupère les valeurs et effectue un traitement
- **method** : (post/get) précise le mode d'envoi des données, ici POST.

Nous avons ensuite deux éléments `<input>` de type **text**, ceux-ci représentent les champs dans lesquels seront saisies les valeurs de internautes. Ils possèdent 2 attributs supplémentaires :

- **id**
- **name**

Ces 2 attributs devront toujours être présents afin de faciliter le traitement des informations que cela soit en javascript ou en PHP. (On peut remarquer qu'il possède la même valeur)

Et enfin il reste un dernier élément **<input>** celui-ci étant de type **submit**. C'est lui qui donnera l'ordre d'envoi du formulaire au serveur lorsque l'on cliquera dessus.

*Nous avons à présent le minimum requis pour avoir un formulaire.*

Nous allons tout de même un peu customiser notre formulaire en ajoutant l'élément **<fieldset>** permettant de regrouper des éléments de formulaire et y ajouter une légende.

```
<fieldset>
  <legend> Informations personnelles </legend>
  <label for=»nom«>Nom : </label><input type=»text« name=»nom« id=»nom« />
  <br/>
  <label for=»prenom«>Prénom : </label><input type=»text« name=»prenom« id=»prenom« />
</fieldset>
```

Puis dans le css:

```
#mon_formulaire {
  margin-top: 5px;
  padding: 10px;
  width: 400px;
}

#mon_formulaire label{
  float:left;
  font-size:13px;
  width:110px;
}
```

Il est à noter que la balise **<form>** s'enrichit de 2 nouveaux attribut :

- **autocomplete** : (on/off) permet de lancer la saisie semi-automatique
- **novalidate** : (on/off) Si présent le formulaire n'est pas validé lorsqu'il est soumis

## Placeholder, required, pattern et validation

HTML5 introduit de nombreuses nouveautés pour les formulaires pour améliorer l'aide à la saisie et les contrôles disponibles pour l'utilisateur. Plusieurs attributs simples à mettre en place améliorent la prise en charge des formulaires, tout en se passant de JavaScript.

### Placeholder

**placeholder** est un attribut qui permet de renseigner un texte indicatif par défaut dans un champ de formulaire.

*Modifier la page : 05-formulaire\_simple.php*

Ajouter l'attribut placeholder sur chaque champ de saisie

```
<label for=»nom«>Nom : </label><input type=»text« name=»nom« id=»nom« placeholder=»votre nom« /><br/>
```

```
<label for="prenom">Prénom : </label><input type="text" name="prenom" id="prenom"
placeholder="votre prénom" /><br/>
```

### Enregistrer et visualiser le résultat

L'attribut placeholder peut être placé sur les éléments :

- **<input>** : de type text, search, password, url, tel, email
- **<textarea>**

Compatibilité :

Firefox 4.0+, Opera 11.01+, Chrome 3.0+, Safari 3.0+, ie 10+

Il est bien sûr possible de styler cet élément néanmoins le flou actuel nous oblige à passer par des pseudo class :

```
-webkit-input-placeholder { //safari
-moz-placeholder { //firefox
```

A l'heure actuelle, les styles applicables au placeholder sont très limités, ils se concentrent majoritairement sur des styles de texte :

- Text-decoration
- Font-style
- Color

Dans le fichier CSS :

```
/* firefox */
#mon_formulaire input:-moz-placeholder {
    color: red;
}
/*safari */
#mon_formulaire input::-webkit-input-placeholder {
    color: red;
}
```

### Alternatives

Tous les navigateurs ne se comportent pas de la même manière : compréhension de l'attribut, possibilité de modification des styles via CSS, etc.

- **jQuery Placeholder** : Permet l'attribution du comportement du placeholder ainsi que d'une classe CSS permettant d'uniformiser la méthode de sélection.
- <https://github.com/NV/placeholder.js> : propose une alternative compatible jQuery ou en JavaScript pur.

### Les champs requis

L'attribut required permet de rendre obligatoire le remplissage d'un champ et bloquer la validation du formulaire si l'un des champs (concernés par cet attribut) n'a pas été renseigné.

Modifier la page : [chap\\_4/05-formulaire\\_simple.php](#)

Ajouter l'attribut **required** sur chaque champ de saisie

```
<label for=»nom»>Nom : </label><input type=»text» name=»nom» id=»nom» placeholder=»votre nom» required=»required « /><br/>
<label for=»prenom»>Prénom : </label><input type=»text» name=»prenom» id=»prenom» placeholder=»votre prénom» required=»required « /><br/>
```

L'attribut placeholder peut être placé sur les éléments :

- **<input>** : de type text, search, password, url, tel, email, date, datetime, datetime-local, months, week, time, number, checkbox, radio, file

- **<textarea>**

#### Compatibilité :

Firefox 4.0+, Opera 9.5+, Chrome 3.0+, ie 10+

Nous allons pouvoir appliquer un CSS sur le champ de saisie afin d'indiquer que celui-ci est obligatoire

```
[required] {
    border: 1px solid orange;
}
```

#### *Enregistrer et visualiser le résultat*

Il est possible aussi de modifier l'affichage du message obligatoire, l'exemple ci-dessous modifiera celui du navigateur Chrome :

```
[type=»text»]::-webkit-validation-bubble-arrow {
    background-color: #f4f4f4;
    border: 1px solid #D8000C;
    border-width: 1px 0 0 1px;
    box-shadow: none;
}
[type=»text»]::-webkit-validation-bubble-icon {
    display:none;
}
[type=»text»]::-webkit-validation-bubble-message {
    background: #f4f4f4;
    border: 1px solid #D8000C;
    color: #D8000C;
    font-size: 100%;
    box-shadow: 0 1px 5px #bbb;
    text-shadow: 0 1px 0 #fff;
}
[type=»text»]::-webkit-validation-bubble-message::before {
    content: «Téléphone : »;
```

## Les patterns et la validation à la volée

Les champs requis non complétés ou ne respectant pas une certaine règle de syntaxe définie par le type du formulaire, ou par l'attribut **pattern** bloquent le processus d'envoi des données d'un formulaire.

Cette validation à la volée permet à l'utilisateur d'être informé très rapidement de ses erreurs et de les corriger étape par étape grâce aux indications fournies par les infobulles situées sous chacun des champs invalides.

Certains patterns par défaut existent pour certains types de champ, comme les champs de type **email** ou **url** par exemple. Nous aborderons ces nouveaux types dans le prochain chapitre.

Il est cependant possible de créer ces propres patterns grâce aux **expressions régulières**.

Exemple : `<input type="text" pattern="[A-F][0-9]{5}" />`

Ce champ attend une valeur numérique d'au moins 5 chiffres précédés d'une lettre majuscule comprise entre A et F. Si le format n'est pas respecté, le navigateur en informe l'utilisateur.

Dans notre fichier nous allons ajouter un nouveau champ permettant la saisie d'un code. Celui-ci devra répondre au critère du précédent exemple :

```
<label for="code">Code : </label><input type="text" name="code" id="code" pattern="[A-F][0-9]{5}" />
```

L'attribut pattern peut être placé sur les éléments :

- **<input>** : de type text, search, password, url, tel, email

CSS3 prévoit deux pseudo-classes que sont :valid et :invalid afin de marquer clairement les champs... valides et invalides. Il est alors possible d'écraser les styles par défaut appliqués à ces éléments par les navigateurs, qui diffèrent :

- Firefox attribue la propriété box-shadow (rouge) au champ.
- Internet Explorer utilise la propriété border-color (rouge).
- Chrome et Opera n'ajoutent aucun style, mais placent le focus sur le premier champ erroné

*Modifier le CSS et ajouter*

```
:valid {  
    box-shadow: 0 0 2px 1px green;  
}
```

*Enregistrer et tester sur Firefox*

### Compatibilité :

Firefox 4.0+, Opera 9.5+, Chrome 3.0+, ie 10+

### Les bibliothèques JavaScript comme complément

Pour combler le retard des anciens navigateurs quant à la compatibilité avec les nouveautés introduites par HTML5 pour les formulaires, quelques solutions construites sur JavaScript existent. Il s'agit de charger une bibliothèque qui va détecter si le navigateur ne reconnaît pas les attributs évoqués, ou les nouveaux types de champs, et tenter de les simuler dynamiquement avec JavaScript le cas échéant.

La prise en charge par les navigateurs web est très inégale et variée. Étant donné qu'il s'agit de fonctionnalités disparates, il se peut très bien qu'une version relativement passée prenne en charge un attribut mais pas l'autre et inversement selon le moteur de rendu, voire qu'un type spécifique pour `<input>` soit passé sous silence. Il n'est donc pas possible de déclarer - dans l'état actuel du parc de navigateurs - que l'ensemble des apports des formulaires HTML5 est supporté par une version précise, bien que de nombreux progrès eussent été accomplis.

### Alternatives pour l'émulation des fonctionnalités de formulaires HTML5 avec JavaScript

Ces bibliothèques sont couramment nommées polyfills en version originale, et prennent parfois en compte de bien anciens navigateurs (jusqu'à Internet Explorer 6). Dans les cas d'utilisation les plus simples, il suffira d'inclure le fichier JavaScript dans la page courante grâce à la balise `<script>`.

- H5F : propose un support des nouveautés apportées par HTML5 Forms pour les vieux navigateurs. (<http://www.thecssninja.com/javascript/H5F>)

- H5Validate : propose un support des validations. (<http://ericleads.com/h5validate>)

- Webforms 2 : comprend une émulation de certains attributs (pattern, required, autofocus), des nouveaux types pour `<input>`, et des méthodes JavaScript pour connaître l'état de la validation.

<https://github.com/westonruter/webforms2>

- html5Widgets : embarque une panoplie de contrôles supplémentaires (datepicker/calendrier pour les champs de dates), une gestion des attributs required, pattern, autofocus, placeholder, etc et des éléments range, output. <https://github.com/zoltan-dulac/html5Widgets>

### Les nouveaux types `<input>`

HTML5 apporte plus d'une douzaine de nouveaux types, dont nous allons découvrir ou redécouvrir la liste et le descriptif pour chacun d'eux.

Les tests suivant seront principalement effectués sur Opéra, safari, chrome

#### Le type tel

Il s'agit d'une déclinaison d'un champ de type text. Aussi, aucun format spécifique n'est attendu par le navigateur. Il n'y a donc pas de pattern précis qui vérifierait si au moins un chiffre est renseigné.

Cependant, sur certains navigateurs de SmartPhone, comme sur Safari pour iOS par exemple, l'entrée du numéro de téléphone est facilitée par le basculement à un clavier de type numérique.

Créer la page : [chap\\_4/06-formulaire\\_typeTel.php](#)

Le navigateur ne proposant pas cette vérification, un contrôle plus fin grâce à l'attribut pattern ainsi qu'un contrôle côté serveur sera nécessaire.

Pour effectuer ce type de contrôle, vous aurez besoin d'une expression régulière pour détecter un

numéro de téléphone local ou international.

```
pattern="^(\\+\\d{1,3}(-|)?\\(\\d{1,5}\\)|\\(\\d{2,6}\\)?)(-|)?\\d{3,4}(-|)?\\d{4}(( x| ext)\\d{1,5}){0,1}$"
```

```
<form id="mon_formulaire" action="" method="post">
  <label for="champ1">Tel.</label>
  <input type="tel" id="champ1" required> <em>(sans pattern)</em><br />
  <label for="champ2">Tel.</label>
  <input type="tel" id="champ2" pattern="^(\\+\\d{1,3}(-|)?\\(\\d{1,5}\\)|\\(\\d{2,6}\\)?)(-|)?\\d{3,4}(-|)?\\d{4}(( x| ext)\\d{1,5}){0,1}$" required> <em>(avec pattern)</em><br />
  <input type="submit" value="Valider" />
</form>
```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 9.5+, Chrome 3.0+, ie 10+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

### Le type url

En apparence ce champ ressemble à celui de type **text**. Cependant, le navigateur attend cette fois un format bien spécifique devant respecter un pattern de type url.

Tous les types d'URL sont admis (ftp://, mailto:, http://, etc.). Par défaut, sur Opera, lorsque vous ne spécifiez pas le type de protocole, le navigateur ajoute automatiquement le type http://, ce qui valide forcément le format attendu pour ce champ.

*Créer la page : [chap\\_4/07-formulaire\\_typeURL.php](#)*

```
<form id="mon_formulaire" action="" method="post">
  <label for="champ1">Votre site web</label>
  <input type="url" id="champ1" required value="http://">
  <br/>
  <input type="submit" value="Valider" />
</form>
```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 9.5+, Chrome 3.0+, ie 10+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

### Le type email

Toujours très proche du type text, ce champ est équivalent au type url, seul le format attendu change.

Ce champ attend au minimum un caractère (caractère non accentué comprenant les séparateurs tirets ou underscore) suivi d'un @ suivi à son tour d'un caractère.

*Exemple d'entrée invalide* : é@c

*Exemples d'entrées valides* : -@\_ , f@r

Une nouvelle fois, le clavier du SmartPhone compatible avec ce type de champ de formulaire vous présentera un clavier adapté incluant le symbole arobase.

Créer la page : [chap\\_4/08-formulaire\\_typeEmail.php](#)

```
<form id="mon_formulaire" action=" method="post">
  <label for="champ1">Votre e-mail</label>
  <input type="email" id="champ1" required>
  <br />
  <input type="submit" value="Valider" />
</form>
```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 9.0+, Chrome 6.0+, ie 10+, Safari 5+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

### Le type date,time et datetime

#### Champ de type date

Ce champ visuellement proche de celui de type text vous permet d'activer une aide au remplissage (type *datepicker*) présente uniquement sur quelques navigateurs et différente de l'un à l'autre.

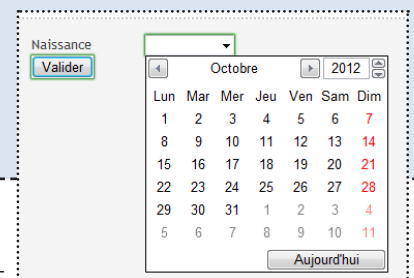
Le contenu attendu est une date du calendrier Grégorien au **format RFC3339** sans précision de la *timezone*, mais le champ accepte un contenu vide ou mal formaté sans retourner d'erreur.

La liste des attributs compatible est la suivante :

name, disabled, form, type, autocomplete, autofocus, list, min, max, step (chiffre entier), readonly, required, value, pattern ainsi que les attributs classiques, d'événements et xml.

Créer la page : [chap\\_4/09-formulaire\\_typeDate.php](#)

```
<form id="mon_formulaire" action="09-formulaire_typeDate.php" method="post">
  <label for="champ1">Votre date</label>
  <input type="date" id="champ1" required>
  <br />
  <input type="submit" value="Valider" />
</form>
```



### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 9.0+, Chrome 6.0+, ie 10+, Safari 5+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

#### Champ de type time

Ce type de champ permet de renseigner une heure, avec plus ou moins de précision. Le format attendu est le même que pour le champ de type date, sans localisation (*timezone*).

La liste des attributs compatibles est très proche, seul l'attribut step autorise ici un nombre à virgule (*float*).

À l'instar du champ de type date, ce type de champ déclenche l'affichage d'une sorte de sélecteur



permettant d'incrémenter ou décrémenter la valeur du champ minute par minute.

On peut rencontrer ce comportement sur Safari et Opera sous la forme suivante.

Créer la page : [chap\\_4/10-formulaire\\_typeTime.php](#)

```
<form id="mon_formulaire" action="10-formulaire_typeTime.php" method="post">
  <label for="champ1">Heure de r&eacute;veil</label>
  <input type="time" id="champ1" name="time">
  <br />
  <input type="submit" value="Valider" />
</form>
```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 10.6+, Chrome 16, ie 10+,Safari 5+

**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

#### champ de type datetime

Ce type de champ est une combinaison des deux types **date** et **time**. Il permet donc de renseigner une date et une heure précise dans un même champ (visuel), dans le cas de Safari, et dans deux champs séparés dans le cas d'Opera.

datetime propose un timezone qui est précisé en dehors des champs sous Opera, et directement intégré au champ dans Safari ("Z" équivaut ici à "UTC").

Créer la page : [chap\\_4/11-formulaire\\_typeDateTime.php](#)

```
<form id="mon_formulaire" action="11-formulaire_typeDateTime.php" method="post">
  <label for="champ1">Prochain RDV</label>
  <input type="datetime" name="datetime" id="champ1"><br />
  <input type="submit" value="Valider" />
</form>
```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 10.6+, Chrome 16, ie 10+,Safari 5+

**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

#### Champ de type datetime-local

Ce type de champ est très proche du type **datetime**, la seule différence étant l'absence de précision du fuseau horaire.

Les informations visuelles fournies par les champs de formulaire sont semblables à ceux de datetime, "UTC" et "Z" en moins pour Opera et Safari, respectivement.

Créer la page : [chap\\_4/12-formulaire\\_typeDateTime-Local.php](#)

```
<form id="mon_formulaire" action="12-formulaire_typeDateTime-Local.php" method="post">
  <label for="champ1">Prochain RDV</label>
  <input type="datetime" name="datetime" id="champ1"><br />
  <input type="submit" value="Valider" />
```

Valeur renvoyée par le champ :

Prochain RDV

&lt;/form&gt;

**Compatibilité :****Nav.** : Firefox 4.0+, Opera 10.6+, Chrome 16, ie 10+,Safari 5+**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1**Champ de type week**

Le type de champ **week** permet de renseigner une semaine dans une année. Il attend donc un format de type 2012W05 qui correspond à l'année et au numéro de semaine dans celle-ci.

La liste des attributs compatibles reste la même que pour le type date.

Créer la page : [chap\\_4/13-formulaire\\_typeWeek.php](#)

```
<form id="mon_formulaire" action="13-formulaire_typeWeek.php" method="post">
  <label for="champ1">Semaine </label>
  <input type="week" id="champ1" name="week">
  <br />
  <input type="submit" value="Valider" />
</form>
```

Valeur renvoyée par le champ :

Semaine

Valider

Semaine	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
40	1	2	3	4	5	6	7
41	8	9	10	11	12	13	14
42	15	16	17	18	19	20	21
43	22	23	24	25	26	27	28
44	29	30	31	1	2	3	4
45	5	6	7	8	9	10	11

Aujourd'hui

**Compatibilité :****Nav.** : Firefox 4.0+, Opera 10.6+, Chrome 16, ie 10+,Safari 5+**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1**Champ de type number**

Le type de champ **number** permet de renseigner une valeur numérique. Le champ de formulaire est alors transformé en une sorte de boîte permettant l'incrémentation et la décrémentation d'une valeur numérique initiale (0 par défaut), lorsque la prise en charge par le navigateur est complète.

Créer la page : [chap\\_4/14-formulaire\\_typeNumber.php](#)

```
<form id="mon_formulaire" action="14-formulaire_typeNumber.php" method="post">
  <label for="champ1">Code postal </label>
  <input type="Number" id="champ1" name="cp">
  <br />
  <input type="submit" value="Valider" />
</form>
```

Valeur renvoyée par le champ :

nombre

Valider

Il est possible d'ajouter un attribut step pour spécifier un pas d'incrémentation. Mais aussi de fixer un minimum et un maximum.

Créer la page : [chap\\_4/14-formulaire\\_typeNumber\\_Bis.php](#)

```
<form id="mon_formulaire" action="14-formulaire_typeNumber_Bis.php" method="post">
  <label for="champ1">Nb bits </label>
  <input type="number" id="champ1" name="number" step="8" min="0" max="64">
  <br />
  <input type="submit" value="Valider" />
</form>
```

**Compatibilité :**

**Nav.** : Firefox 4.0+, Opera 10.6+, Chrome 16, ie 10+,Safari 5+

**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

**Champ de type range**

Ce champ propose un contenu évalué approximatif.

Bien que l'on puisse convertir la position du curseur numériquement (par défaut la valeur la plus basse est 0, la plus haute 100), l'utilisateur n'a pas de repère numérique, seule la position du curseur est un indice. La valeur la plus haute se trouve à gauche (il fallait le deviner ?) pour un sens de lecture *ltr* (left to right) bien entendu !

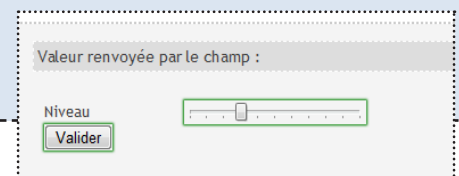
Si vous avez l'habitude de passer d'une lecture *rtl* à *ltr* de par votre polyglottisme, cet affichage approximatif pourrait vous poser des soucis (cela reste une supposition), d'autant plus que le curseur se place par défaut au milieu.

Créer la page : [chap\\_4/15-formulaire\\_typeRange.php](#)

```
<form id="mon_formulaire" action="15-formulaire_typeRange.php" method="post">
  <label for="champ1">Niveau </label>
  <input type="range" id="champ1" name="range"><br />
  <input type="submit" value="Valider" />
</form>
```

Possibilité d'une valeur de départ, un max un min et un pas d'incrément

```
<form id="mon_formulaire" action="15-formulaire_typeRange.php" method="post">
  <label for="champ1">Niveau </label>
  <input type="range" id="champ1" name="range" value="15" max="50" min="0"
step="5"><br />
  <input type="submit" value="Valider" />
</form>
```

**Compatibilité :**

**Nav.** : Firefox 4.0+, Opera 11+, Chrome 10, ie 10+,Safari 5+

**Tel.** : Firefox,Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

**Champ de type color**

Ce type permet de transformer le champ en une palette de couleurs.

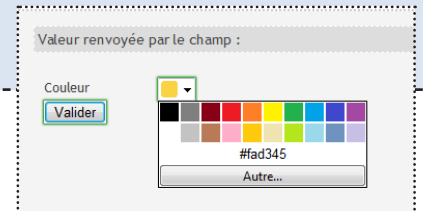
La valeur attendue est une couleur au format hexadécimal (un dièse suivi de 6 caractères alpha-numériques compris entre A et F, et 0 et 9).

Actuellement seul Opera 11+ permet l'affichage de cette palette, avec un nombre de couleurs limité, proposant, par l'intermédiaire d'un bouton "Autre...", d'étendre la palette et un colorpicker vous permettant de composer une palette de couleurs personnalisées temporaire.

Les autres navigateurs affichent un simple champ de type texte, il n'est pas possible de prévoir un *pattern* contrôlant la valeur renseignée par l'utilisateur, puisque cet attribut n'est pas compatible avec ce type.

Créer la page : [chap\\_4/16-formulaire\\_typeColor.php](#)

```
<form id="mon_formulaire" action="16-formulaire_typeColor.php" method="post">
  <label for="champ1">Couleur </label>
  <input type="color" id="champ1" name="color" value="#fad345"> <input type="submit"
value="Valider" />
</form>
```



### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 11+, Chrome 10, ie 10+, Safari 5+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

### L'élément output

Ce nouvel élément représente la somme d'un calcul.

Il accepte comme attribut : for, name, et form.

La valeur de l'attribut for de l'élément <output> reprend les valeurs des id de chaque champ <input> nécessaires au calcul.

L'attribut form de l'<output> reprend la valeur de l'id du formulaire qui comprend les différents champs servant au calcul.

Ce dernier point est important car l'élément output n'est pas forcément interne au formulaire.

Créer la page : [chap\\_4/17-formulaire\\_typeOutput.php](#)

```

<form action="17-formulaire_typeOutput.php" method="post" id="tva_form" onsubmit="ttc.value
= ht.value * (1 + tva.value/100); return false;">
  <p>
    <label for="t_ht">Tarif HT</label><input type="number" name="ht" id="t_ht"> &euro;
  </p>
  <p>
    <label for="t_tva">TVA</label><input type="number" name="tva" id="t_tva" va-
lue="19.6"> %
  </p>
  <p>
    Prix TTC :
    <output for="t_ht t_tva" name="ttc" form="tva_form"></output> &euro;
  </p>
  <button>valider</button>
</form>

```

### Compatibilité :

**Nav.** : Firefox 4.0+, Opera 11+, Chrome 10, ie 10+, Safari 5+

**Tel.** : Firefox, Chrome, Opera (Android 4), Safari (iOS 3.1), Android Browser 3.1

### L'élément Keygen

Cet élément nécessite un certain bagage technique pour en comprendre et maîtriser les usages.

Essayons d'en résumer les grandes lignes.

**<keygen>** permet de générer un jeu de clefs pour le cryptage et le décryptage d'informations (enregistrées en base de données ou transmises d'un serveur à l'autre par exemple).

Le jeu équivaut à une paire de clefs, l'une dite publique, l'autre dite privée. La clef privée est stockée localement, tandis que la clef publique est envoyée sur le serveur. Différents niveaux de cryptage existent et correspondent à un niveau de sécurité plus ou moins élevé.

### L'attribut keytype

Lié à l'élément keygen cet attribut permet de renseigner le type de clef attendue. Pour le moment la seule existante est celle de type RSA (Rivest, Shamir et Adleman, les inventeurs de la méthode de cryptage).

**<keygen keytype="rsa" name="key">**

### L'attribut challenge

Cet attribut ajoute une chaîne de caractère envoyée avec la clef publique.

**<keygen keytype="rsa" challenge="sel\_de\_mer" name="key">**

Créer la page : [chap\\_4/18-formulaire\\_typeKeygen.php](chap_4/18-formulaire_typeKeygen.php)

```

<p class=»hint»>Génération d'une clef. Le niveau peut-être augmenté grâce à l'attribut
<code>challenge</code>.</p>
<form action=»element-keygen.php» method=»post»>
  <p>
    <label for=»e-mail»>E-mail personnel</label>
    <input type=»email» name=»email» id=»e-mail»>
  </p>
  <p>
    <label for=»secu»>Sécurisation</label>
    <keygen type=»secu» name=»secu» id=»secu»>
  </p>
  <br /><br />
  <button>Tayst</button>
</form>

<script src=»https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js»></script>

<script type=»text/javascript»>
  $(function()
  {
    $(«form p+p»).after('<p><label for=»plusplus»>Ajouter un «challenge» ?</label><input
type=»checkbox» name=»plusplus» id=»plusplus» /></p>');
    $(«#plusplus»).live('change', function()
    {
      if($(this).filter(':checked').length==1) {
        var date = new Date();
        $(«#secu»).attr('challenge', date.getTime());
        $(«#secu»).after('<span class=»moreplus»> ++</span>');
      }else {
        $(«#secu»).removeAttr('challenge');
        $(«.moreplus»).remove();
      }
    });
  });
</script>

```



# CSS 3

Préfixe Navigateur

Les bordures

Les ombrages

Transparance & opacité

Les arrières-plans multiples

Les dégradés

Les transformations

Les transitions

Les animations



## Préfixe de navigateur

Nous sommes aujourd'hui dans un entre-deux; toutes les nouveautés CSS3 ont été définies par le W3C cependant celles-ci n'ont pas encore été implémentées directement dans les navigateurs. Cependant certains navigateurs avaient développé leurs propres règles CSS3; celles-ci ont d'ailleurs servi aux recommandations du W3C.

Pour cette raison il sera nécessaire de donner une définition spécifique CSS pour chaque navigateur et que la page HTML devra toujours être testée sur le plus grand nombre de navigateur.



Safari

-webkit-



Microsoft

-ms-



Mozilla

-moz-



Chrome

-chrome-



## Les bordures

Dans la version CSS 2.1 il est possible de définir des styles de bordures mais ceci ne répondait pas à toutes les attentes des designers. Très souvent ceux-ci avaient à faire des bords arrondie ou bien des blocs avec des bordures très spécifiques et la seule solution était de placer une image en fond. Dans ce cas le contenu devait s'adapter au contenant (problème si celui-ci est changeant).

Avec CSS3 il possible de définir ces types de bordure.

Créer le document : [chap\\_5/01-bordure.html](#) à partir de la la page [modele\\_page.html](#)

Inclure le code suivant après la balise header:

```
<aside>
  <div id="introduction-content">
    <h2>Le CSS3 du jour</h2>
    <p>Lorem .. bibendum.</p>
  </div>
</aside>
```

Création de la feuille de style associé :  
[assets/css/styles\\_bordure.css](#)

```
#introduction-content{
  width:400px;
  background:#F9A890;
  margin:20px 0 0 20px;
  border: 2px #f2662a solid;
  padding:10px;
}
```

### Le CSS3 du jour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

## Border-radius

Ajoutons des bord arrondies dans le CSS :

```
#introduction-content{
    ...
    border-radius:24px;
    ...
}
```

Enregistrer et visualiser le résultat

### Le CSS3 du jour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

Cette propriété est utilisée sur la plupart des nouveaux navigateurs. Cependant sous d'anciennes versions cette propriété n'est pas appliquée. Sur les anciens navigateurs nous devons ajouter 2 propriétés supplémentaires adressant ce que l'on appelle généralement des préfixes vendeurs.

```
#introduction-content{
    ...
    -webkit-border-radius:24px;
    -moz-border-radius:24px;
    -o-border-radius:24px;
    -ms-border-radius:24px;
    ...
}
```

Dans le cas précédant le bord arrondi a été ajouté à tous les coins mais il est possible de s'adresser à des bords spécifiques :

Dans le fichier CSS modifié le border-radius :

```
#introduction-content{
    ...
    border-radius: 24px 48px 24px 48px
    ...
}
```

Enregistrer et visualiser le résultat

### Le CSS3 du jour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

Nous avons jusqu'ici créé des bords bien arrondis , mais nous pouvons modifier le rayon x et y

Dans le CSS

```
#introduction-content{
    ...
    border-radius: 48px/24px; /* x / y */
    -webkit-border-radius:48px/24px;
    -moz-border-radius:48px/24px;
    -o-border-radius:48px/24px;
    -ms-border-radius:48px/24px;
    ...
}
```

### Le CSS3 du jour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

## Border-image

Afin de pouvoir utiliser la propriété Border-image nous mettrons en commentaire les border-radius et nous allons grossir la taille de la bordure.

Dans le fichier CSS retiré le border-radius & ajout du border-image:

```
#introduction-content{
  border: 20px #f2662a solid;
  border-image: url(..images/border-bg.png) 33%;
  /* cas particulier navigateur */
  -webkit-border-image: url(..images/border-bg.png) 33%;
  -moz-border-image: url(..images/border-bg.png) 33%;
  -o-border-image: url(..images/border-bg.png) 33%;
  -ms-border-image: url(..images/border-bg.png) 33%;
}
```

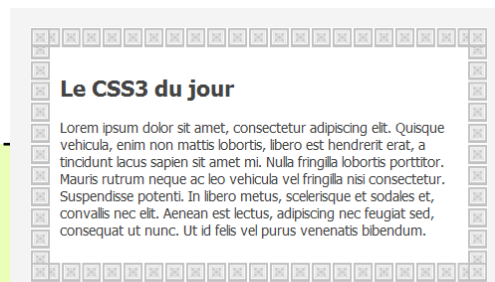
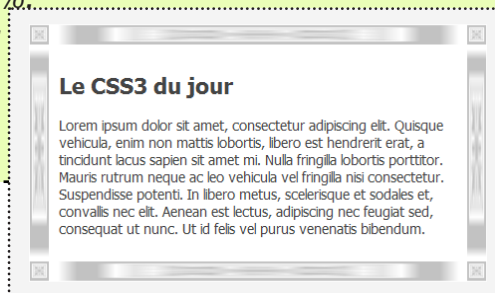


La valeur 33% indique la position où l'image devra être coupée relativement à sa hauteur et à sa largeur.

Enregistrer et visualiser.

On peut remarquer que les bords centraux sont étirés. Afin que ce motif se répète ajouter dans le css :

```
#introduction-content{
  ....
  border-image: url(..images/border-bg.png) 33% repeat;
  /* cas particulier navigateur */
  -webkit-border-image: url(..images/border-bg.png) 33% repeat;
  -moz-border-image: url(..images/border-bg.png) 33% repeat;
  -o-border-image: url(..images/border-bg.png) 33% repeat;
  -ms-border-image: url(..images/border-bg.png) 33% repeat;
  ...
}
```



## Les ombrages

Nouvelle propriété CSS3: l'ombrage sur une boîte et sur du texte. Pour cela rien de plus simple.

Créer le document : [chap\\_5/02-ombrage.html](#) à partir de la page [01-ombrage.html](#)

Puis modifier le ccs pour obtenir [assets/css/styles\\_ombrage.css](#) :

```
#introduction-content{
  width:400px;
  background:#F9A890;
  margin:20px 0 0 20px;
  border:2px #f2662A solid;
```

```
padding:10px;
```

```
box-shadow:15px 10px;
-webkit-box-shadow:15px 10px;
-moz-box-shadow:15px 10px;
-o-box-shadow:15px 10px;
-ms-box-shadow:15px 10px;
}
```

### Le CSS3 du jour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vehicula, enim non mattis lobortis, libero est hendrerit erat, a tincidunt lacus sapien sit amet mi. Nulla fringilla lobortis porttitor. Mauris rutrum neque ac leo vehicula vel fringilla nisi consectetur. Suspendisse potenti. In libero metus, scelerisque et sodales et, convallis nec elit. Aenean est lectus, adipiscing nec feugiat sed, consequat ut nunc. Ut id felis vel purus venenatis bibendum.

Les 2 paramètres sont obligatoires ils indiquent le décalage vers la droite puis le décalage vers le bas. Il est possible d'ajouter un troisième paramètre afin de changer le flou (x et y) du dégradé de l'ombrage.

```
box-shadow : 15px 10px 20px;
```

Il existe enfin un 4ème paramètre pour définir la couleur de l'ombre.

```
box-shadow : 15px 10px 20px #ccc;
```

Afin étendre cette propriété au plus grand nombre de navigateur nous devons nous appuyer sur les moteurs Webkit et Moz.

Pour s'adapter à internet explorer ie8

```
<!--[if lte IE 8]>
  <style type="text/css">
    #introduction-content
    {
      zoom: 1;
      filter: progid:DXImageTransform.Microsoft.Shadow(color='#aaaaaa', Direction=135, Strength=6);
    }
  </style>
<![endif]-->
```

Pour ajouter une ombre sur un texte utiliser la propriété text-shadow.

Modification de la feuille de style associé : [assets/css/base.css](#) pour ajouter une ombre au titre.

```
header h1{
  ...
  text-shadow: 1px 1px 2px #999;
  -webkit-text-shadow: 1px 1px 2px #999;
  -moz-text-shadow: 1px 1px 2px #999;
  -o-text-shadow: 1px 1px 2px #999;
  -ms-text-shadow: 1px 1px 2px #999;
}
```

# HTML5

## Transparence et opacité

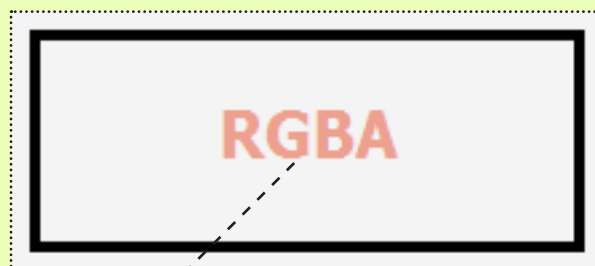
Il existe plusieurs possibilités de créer une opacité. La première est la définition d'une couleur via **rgba**.

Dans un nouveau document [chap\\_5/03-transparence.html](#) ( page [modele\\_page.html](#) )

```
<div class="rgba">
  <h1>RGBA</h1>
</div>
```

```
.rgba {
  width: 200px;
  height: 75px;
  border: #000 4px solid;
  margin: 20px;
}

.rgba h1 {
  text-align: center;
  font-weight: bold;
  font-size: 24px;
}
```



Nous ajoutons au titre **h1**

```
color: rgba(228,76,39,0.5);
```

*Enregistrer et visualiser le résultat.*

Nous allons à présent utiliser la propriété **opacity**. Dans la page HTML ajouter le code suivant :

```
<div class="opacity">
  <h1>OPACITY</h1>
</div>
```

*Puis dans la feuille de style*

```
.opacity {
  width: 200px;
  height: 75px;
  border: #000 4px solid;
  margin: 20px;
  background-color: #FFF;
  box-shadow: 8px 8px 8px;
  -webkit-box-shadow: 8px 8px 8px;
  -moz-box-shadow: 8px 8px 8px;
  -o-box-shadow: 8px 8px 8px;
  -ms-box-shadow: 8px 8px 8px;
}
```

```
.opacity h1 {
  text-align:center;
  font-weight:bold;
  font-size:24px;
  color:rgb(228,76,39);
}
```



Ajoutons au **div opacity** la propriété suivante :

```
opacity: 0.5; - - - - -
```

*Enregistrer et visualiser le résultat.*

Nous allons à présent utiliser les propriétés **rgba** et **opacité** conjointement avec la propriété drop-shadow.

*Dans le corps de la page HTML:*

```
<div class=»dropshadow»>
  <h1> DROPSHADOW </h1>
</div>
```

*Puis dans la feuille de style*

```
dropshadow {
  width: 200px;
  height: 75px;
  border: #000 4px solid;
  margin:20px;
  background-color:#FFF;
  box-shadow: 8px 8px 8px rgba(0,0,0,0.5); - - - - -
  -webkit-box-shadow: 8px 8px 8px rgba(0,0,0,0.5);
  -moz-box-shadow: 8px 8px 8px rgba(0,0,0,0.5);
  -o-box-shadow: 8px 8px 8px rgba(0,0,0,0.5);
  -ms-box-shadow: 8px 8px 8px rgba(0,0,0,0.5);
}
```

```
.dropshadow h1 {
  text-align:center;
  font-weight:bold;
  font-size:24px;
  color:rgb(228,76,39);
}
```



*Enregistrer et visualiser le résultat.*

## Les arrières plans multiples

Background-image est un outil essentiel en CSS2.1, mais celui-ci était limité à une seule image. CSS3 étend cette capacité à plusieurs images.

Nouveau document : [chap\\_5/04-bg\\_multiple.html](#) & nouvelle feuille de style : [assets/css/styles\\_bg\\_multiples.css](#).

On ajoute un fond dégradé via une image :

```
body{
  background-image: url(../images/bg1.png); - - - - -
```

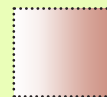


Donnant le résultat suivant :



Sans options celle-ci se répète de manière verticale et horizontale. Maintenant ajoutons une 2ème image au background de la manière suivante :

```
body{
  background-image: url(../images/bg1.png),url(../images/bg2.png) ; - - - -
```



On peut remarquer que les 2 fonds se chevauchent indépendamment et qu'ils ont leur propre vie. Ajoutons une 3ème image :

```
body{
  background-image: url(../images/bg1.png),url(../images/bg2.png), url(../images/bg3.png) ;
```



Et ainsi de suite...

Il est important de noter que si les images se trouvent toutes dans le background celles-ci ne sont pas toutes sur le même plan. La 1ère image mentionnée est celle qui est la plus éloignée et celle indiqué en dernier est celle qui se trouve la plus proche de nous.

Il est possible aussi de définir la position d'une des images du background. Dans l'exemple qui suit nous placerons un dégradé de chaque côté:

```
body{
  background-image: url(../images/bg4.png),url(../images/bg5.png);
  background-repeat: repeat-y;
  background-position: top left, top right;
```





## Les dégradés

Les propriétés précédentes n'offrent que la possibilité de créer une couleur de fond unie. Si l'on voulait intégrer un dégradé au sein de notre site web il fallait recourir au background-image auquel on associait une image élaboré dans un logiciel tel que photoshop. Avec CSS3 et la propriété background nous allons pouvoir créer nos dégradés directement dans nos pages en évitant le chargement d'une image par le réseau.

Nous pouvons obtenir deux types de dégradés: linéaire, horizontal ou vertical, centré ou non, avec deux ou plusieurs couleurs.

*Nouveau document : [chap\\_5/05-degrade.html](#) & nouvelle feuille de style : [assets/css/styles\\_degrade.css](#).*

Création des conteneurs de dégradé dans la page HTML:

```
<div id="simple">
  Dégradé simple
</div>
<div id="angulaire">
  Dégradé angulaire
</div>
<div id="circulaire">
  Dégradé circulaire
</div>
<div id="repetitif">
  Dégradé répétitif
</div>
<div id="rgba">
  <h1>Dégradé RGBA</h1>
</div>
```

```
div{
  height:75px;
  width:400px;
  margin:20px;
  text-align:center;
  border: 2px #f2662a solid;
  font-size:20px;
  font-weight:bold;
}
```

**Dégradé simple**

**Dégradé angulaire**

**Dégradé circulaire**

**Dégradé répétitif**

**Dégradé RGBA**

### Dégradé simple

Pour obtenir des dégradés linéaires, la syntaxe à utiliser est la suivante:

**background : linear-gradient(top|left, couleur 1 X%, couleur 2 Y%, ... couleur N z%);**

ou bien sur la propriété *background-image*. Dans le fichier CSS



```
#simple{
  background-image:linear-gradient(top, white,black);
  background-image:-moz-linear-gradient(top,white,black);
  background-image:-o-linear-gradient(top,white,black);
  background-image:-ms-linear-gradient(top,white,black);
  background-image:-webkit-linear-gradient(top,white,black);
}
```

Dégradé simple

Nous pouvons remarquer que le seul navigateur sur lequel ne s'applique pas le dégradé est IE. Cependant n'oubliez pas d'indiquer quand même sa valeur dans le cas où cette propriété finirait par être implémentée.

Si nous avons voulu que le dégradé linéaire soit appliqué de façon horizontale, il suffirait de remplacer la valeur **top** par **left**.

### Dégradé angulaire

Le dégradé ne se limite pas à une application vertical ou horizontal d'une couleur vers une autre.

Dégradé angulaire

```
#angulaire{
  background-image:linear-gradient(45deg, white, green, black);
  background-image: -moz-linear-gradient(45deg, white, green, black);
  background-image: -o-linear-gradient(45deg, white, green, black);
  background-image: -ms-linear-gradient(45deg, white, green, black);
  background-image: -webkit-linear-gradient(45deg, white, green, black);
}
```

### Dégradé radial

Pour créer un fond en dégradé radial nous disposons encore de propriétés *background* ou *background-image* :

**background : radial-gradient(départ, forme, coul1 X%, coul2 Y%);**

Valeur pour départ:

*top, middle, bottom left, center, right* - exemple: *top right* (départ du cercle en haut à droite)

Il est aussi possible d'indiquer directement une position précise en indiquant un pourcentage 60% 40% (à 60% de hauteur et 40% de largeur).

Valeur pour la forme:

Il est d'abord composé des mots-clés *circle* ou *ellipse* pour la forme elle-même et des mots clés suivant pour son extension:

*cover* : occupe toute la surface du conteneur.

*closest-side* : associé à *circle*, le dégradé est circulaire et s'arrête sur le côté le plus proche de son centre. Avec *ellipse* il s'étend sur toute la surface.

*closest-corner* : le dégradé s'étend jusqu'au coin le plus proche de son conteneur.

*farthest-side* : le dégradé s'étend jusqu'au côté le plus éloigné de son conteneur.

*farthest-corner* : le dégradé s'étend jusqu'au coin le plus éloigné de son conteneur.

*contain* : la forme circulaire ou elliptique est entièrement contenue dans le conteneur.

Dans le fichier CSS définir le dégradé radial suivant:

Dégradé circulaire

```
#circulaire{
  background-image: radial-gradient(center top, circle cover, white, #111);
  background-image: -moz-radial-gradient(center top, circle cover, white, #111);
  background-image: -o-radial-gradient(center top, circle cover, white, #111);
  background-image: -ms-radial-gradient(center top, circle cover, white, #111);
  background-image: -webkit-radial-gradient(center top, circle cover, white, #111);
}
```

### Dégradé répétitifs

Pour répéter un effet de dégradé il suffit de définir la valeur suivante à la propriété background ( ou background-image) : X% définissant la zone restante

**background : repeating-linear-gradient(top|left, couleur 1 X%, couleur 2 Y%, ... );**

**background : repeating-radial-gradient(départ, forme, coul1 X%, coul2 Y%);**

```
#repetitif{
  background-image: repeating-linear-gradient(left, white 80%, black, white);
  background-image: -moz-repeating-linear-gradient(left, white 80%, black, white);
  background-image: -o-repeating-linear-gradient(left, white 80%, black, white);
  background-image: -ms-repeating-linear-gradient(left, white 80%, black, white);
  background-image: -webkit-repeating-linear-gradient(left, white 80%, black, white);
}
```

Dégradé répétitif

### Dégradé RGBA

Nous pouvons ajouter des couleurs RGBA afin d'obtenir un effet de transparence dans les dégradés, ce qui leurs donnent une certaine profondeur.

Dans l'exemple qui suit le dégradé transparent est appliqué au style <h1>, ce qui permet à l'image d'arrière plan du conteneur d'être visible:

```
#rgba h1{
  height:75px;
  margin:0;

  background-image:linear-gradient(top, rgba(147,185,196,0.9),rgba(110,124,140,0.9));
  background-image:-webkit-linear-gradient(top,rgba(147,185,196,0.9),rgba(110,124,140,0.9));
  background-image:-moz-linear-gradient(top,rgba(147,185,196,0.9),rgba(110,124,140,0.9));
  background-image:-ms-linear-gradient(top,rgba(147,185,196,0.9),rgba(110,124,140,0.9));
  background-image:-o-linear-gradient(top,rgba(147,185,196,0.9),rgba(110,124,140,0.9));
}
```

Dégradé RGBA

## Les transformations

Les transformations CSS vont nous permettre de faire tourner, de changer l'échelle ou encore de tordre un élément de notre page. De même il est possible de définir une perspective afin de simuler une position ou une animation dans l'espace 3D.

### Transformation 2D

Nouveau document : [chap\\_5/06-transformation.html](#) & nouvelle feuille de style : [assets/css/transformation.css](#).

```
<div id="wrap">
  <div class="transformation">
    
  </div>
</div>
```

```
#wrap{
  width:80px;
  margin-right:10px;
}
#transformation{
  float:right;
  margin:10px;
  border: 20px solid #fff;
  border-radius: 10px;
  transform: rotate(5deg);
  -webkit-transform: rotate(5deg);
  -moz-transform: rotate(5deg);
  -o-transform: rotate(5deg);
  -ms-transform: rotate(5deg);
}
```



Si l'on avait voulu modifier les proportions du div on aurait appliqué :

transform: scale(1.5) -> 1.5 représente la valeur 150%, à taille normal un objet vaut 1 (100%)

### Transformation 3D

#### Les rotations

rotate3d(x,y,z,angle) : spécifie une rotation 3D autour de l'axe défini par le vecteur (x,y,z).

rotateX(angle) : spécifie une rotation autour de l'axe X.

rotateY(angle) : spécifie une rotation autour de l'axe Y.

rotateZ(angle) : spécifie une rotation autour de l'axe Z. Équivalent à rotate(angle) en 2D.

#### Les translations

translate3d(x,y,z) : spécifie une translation en 3D.

translateZ(z) : spécifie une translation sur l'axe Z.

#### Les changements d'échelle

scale3d(x,y,z) : spécifie un changement d'échelle en 3D.

scaleZ(z) : spécifie un changement d'échelle sur l'axe Z.

#### Matrice

matrix3d(une matrice 4x4) : spécifie une matrice de transformation. Par défaut, toutes les fonctions de transformations sont converties en utilisant une matrice, mais cela est transparent pour nous.

#### Perspective

perspective(nombre) : spécifie la profondeur de perspective.

Il est possible d'appliquer plusieurs transformations différentes sur un même élément HTML. Pour cela, séparez les différentes fonctions par un espace. Attention toutefois, les fonctions sont appliquées dans l'ordre d'écriture, et donc l'effet final peut varier (par exemple, une rotationX de 45deg + une rotationY de 45deg n'est pas équivalente à une rotationY de 45deg + une rotationX de 45deg).

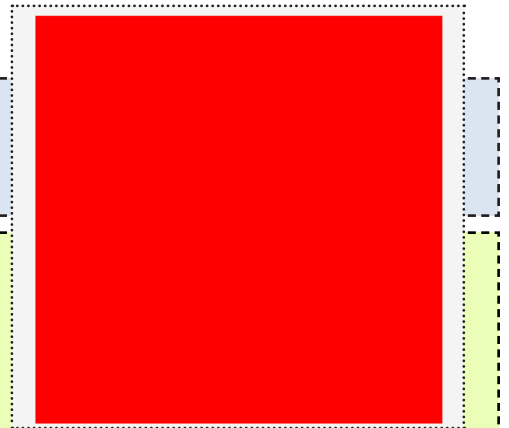
*Nouveau document : [chap\\_5/06-transformation-bis.html](#) & nouvelle feuille de style : [assets/css/transformation.css](#).*

Nous créons une boîte rouge contenue dans un autre div

```
<div id="scene3D">
  <div id="red"></div>
</div>
```

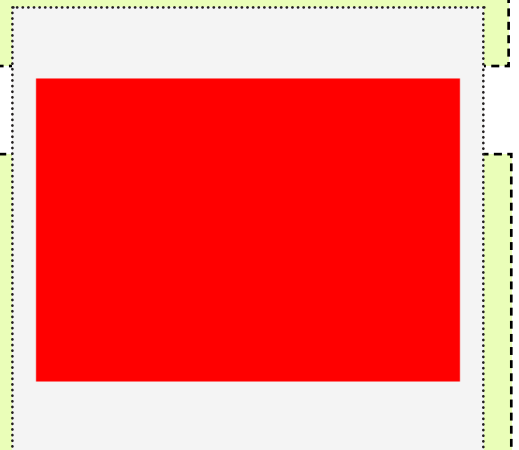
```
#red{
  width:300px;
  height:300px;
  background:red;
}
```

```
#scene3D{
  margin:auto;
  width:300px;
  height:300px;
}
```



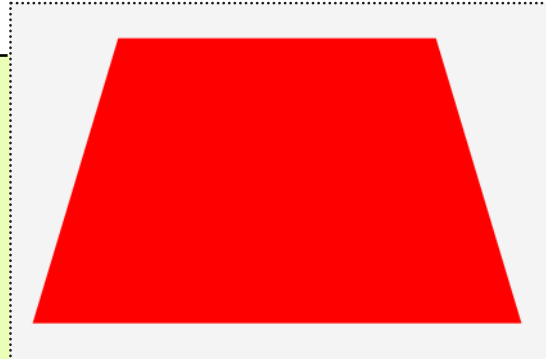
J'applique une rotation de 45° sur l'axe des X (axe horizontal)

```
#red{
  ...
  transform: rotateX(45deg);
  -webkit-transform: rotateX(45deg);
  -moz-transform: rotateX(45deg);
  -o-transform: rotateX(45deg);
  -ms-transform: rotateX(45deg);
}
```



La transformation provoque un aplatissement de la forme, ce qui est tout à fait normal. Pour obtenir une impression de 3D (de profondeur), il faut spécifier la perspective. Mais celle-ci devra être spécifiée sur le conteneur de notre boîte rouge.

```
#scene3D{
  ...
  perspective:500px;
  -webkit-perspective:500px;
  -moz-perspective:500px;
  -o-perspective:500px;
  -ms-perspective:500px;
}
```



Enregistrer et visualiser le résultat. Essayer d'ajouter du texte.

## Les transitions

Les transformations que nous venons de réaliser sont opérées de manière immédiate dans les navigateurs. Les nouveautés CSS3 permettent d'effectuer une transition dans les transformations, ce qui donne un effet visuel intéressant. Elle se définit de la manière suivante:

**transition-property: prop1, prop2 ... propN** : Liste des propriétés sur lesquels on applique la transition.

**transition-duration : Ns** : durée de la transition en seconde.

**transition-timing-function : value** : type de fonction de transition. Les valeurs peuvent être:

- *linear* : vitesse constante du début à la fin
- *ease-in* : la vitesse de transition augmente
- *ease-out* : la vitesse de transition diminue
- *ease-in-out* : la vitesse de transition est lente au début et à la fin

**transition-delay : Ns** : temps d'attente avant déclenchement de la transition

Il est possible de définir toutes ces informations sur une même ligne définie de la manière suivante:

**transition : prop || durée || fonction || délai**

Pour notre 1er exemple nous allons appliquer notre transition sur un élément possédant une transformation précise à savoir un lien : a vers a:hover. Et pour ce faire nous l'appliquerons sur le menu navigation de notre exemple:

Ouvrir le CSS [assets/css/styles\\_structure.css](#) et ajouter la transition suivante :

```
nav li a{
  ...
  transition-property:background;
  transition-duration:1s;
  transition-timing-function:ease-out;
  transition-delay:0s;
```

```

-webkit-transition-property:background;
-webkit-transition-duration:1s;
-webkit-transition-timing-function:ease-out;
-webkit-transition-delay:0s;

-o-transition-property:background;
-o-transition-duration:1s;
-o-transition-timing-function:ease-out;
-o-transition-delay:0s;

-moz-transition-property:background;
-moz-transition-duration:1s;
-moz-transition-timing-function:ease-out;
-moz-transition-delay:0s;
}

```

#### Enregistrer et visualiser le résultat.

Nous pouvons remarquer qu'au survol des menus la couleur passe du noir au orange de manière progressive. Attention la definition de la transition dans ce cas se place dans la propriété lien et pas lien survolé.

Dans l'exemple qui suit nous allons définir un diaporama et au survol de chaque miniatures nous effectuerons une transition vers une rotation et une nouvelle echelle.

Créer un nouveau document [chap5/07-transition.html](#) ainsi que la feuille de style [assets/css/styles\\_diaporama.css](#).

Dans la page html insertion du diaporama :

```

<div id="diaporama">
  <ul class="galerie">
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
    <li><a href="#"></a></li>
  </ul>
</div>

```

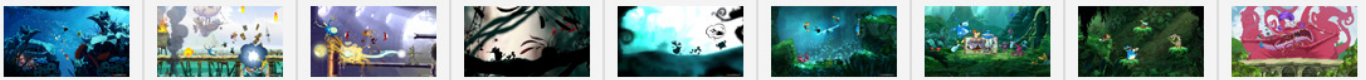
```

#diaporama{
  float:left;
  padding:20px;
}

```

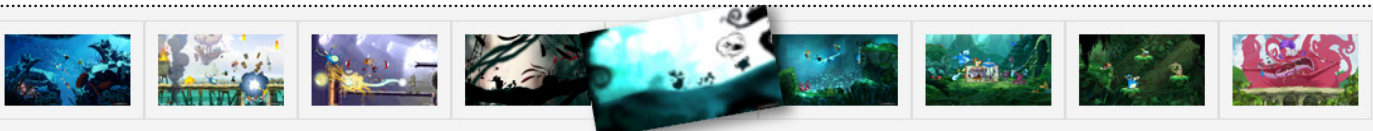
```
ul.galerie li{
  float:left;
  margin:0, 10px;
  padding: 10px;
  border: 1px solid #ddd;
  list-style:none;
}
ul.galerie li a img{
  float:left;
  width:100px
}
```

Enregistrer et visualiser le résultat.



Définition d'une image survolé

```
ul.galerie li a: hover img{
  /* grossissement */
  -webkit-transform: scale(1.5) rotate(-10deg);
  -moz-transform: scale(1.5) rotate(-10deg);
  -o-transform: scale(1.5) rotate(-10deg);
  -ms-transform: scale(1.5) rotate(-10deg);
  transform: scale(1.5) rotate(-10deg);
  /* ombre */
  -webkit-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
  -moz-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
  -o-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
  -ms-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
  box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
}
```



Application de la transition

```
ul.galerie li a img{
  /* transition */
  -webkit-transition: -webkit-transform 0.2s ease-in-out;
  -moz-transition: -moz-transform 0.2s ease-in-out;
  -o-transition: -o-transform 0.2s ease-in-out;
  transition: transform 0.2s ease-in-out;
}
```



## Les animations

La différence entre animations et transitions CSS n'est pas immédiatement apparente, car les deux propriétés créent une illusion de mouvement. Dans le chapitre précédent, la transition est provoquée par le survol d'un menu ou bien d'une image, donc réalisé par un évènement utilisateur. En aucun cas on ne peut spécifier un nombre de déclenchement; exemple : faire clignoter un lien trois fois lors du survol, ici ce n'est pas une transition que nous utiliserons mais une animation.

Pour réaliser une animation nous aurons besoin de placer dans les propriétés de l'objet animé:

**animation-name** : nom de l'animation

**animation-duration** : durée de l'animation

**animation-iteration-count** : nombre de fois ou l'animation est réalisée

**animation-timing-function** : type de fonction d'animation

- *linear* : vitesse constante du début à la fin
- *ease-in* : la vitesse de transition augmente
- *ease-out* : la vitesse de transition diminue
- *ease-in-out* : la vitesse de transition est lente au début et à la fin

**animation-delay** : délai d'attente avant exécution de l'animation.

Une fois ces paramètres réglés nous devrons définir l'animation en elle même

**@-nom\_animation{**

**from {**

définition de la position de départ

**}**

**X% {**

définition de la position à X% de l'animation. La position X% n'est pas obligatoire mais est très pratique lorsque l'on veut faire une animation en boucle ou bien la rotation d'un objet pour lui donner un sens.

**}**

**to {**

définition de la position d'arrivée

**}**

**}**

### L'effet BOUNCE

Dans l'exemple qui suit nous allons animer une icône en boucle avec un effet «Bounce» (effet de rebond).

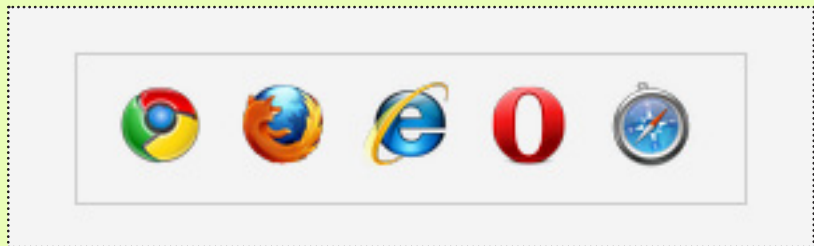
Créer un nouveau document [chap5/08-animation.html](#) ainsi que la feuille de style [assets/css/styles\\_animation.css](#).



```
<div id="icoList">
  <div id="icoChrome"></div>
  <div id="icoMoz"></div>
  <div id="icoIE"></div>
  <div id="icoOpera"></div>
  <div id="icoSafari"></div>
</div>
```

Puis appliquer le css suivant :

```
#icoList{
  margin: 20px auto 0 auto;
  padding: 10px;
  border: 1px solid #CCC;
  width: 230px;
  height: 40px;
}
#icoList div{
  position: relative;
  width: 32px;
  height: 32px;
  border: 0;
  float: left;
  margin: 7px;
}
#icoChrome{
  background: url('../images/chrome1.png');
}
#icoMoz{
  background: url('../images/firefox1.png');
}
#icoIE{
  background: url('../images/ie1.png');
}
#icoOpera{
  background: url('../images/opera1.png');
}
#icoSafari{
  background: url('../images/safari1.png');
}
```



Nous allons à présent définir l'effet de rebond sur l'icone Opéra. D'abord appel de l'animation

```
#icoOpera{
  ...
  /* appel animation */
  animation: bounce 0.7s ease infinite;
```

```

-webkit-animation: bounce 0.7s ease infinite;
-o-animation: bounce 0.7s ease infinite;
-moz-animation: bounce 0.7s ease infinite;
-ms-animation: bounce 0.7s ease infinite;
}
@keyframes bounce{
  from {top: 0px;}
  50% {top: -10px;}
  to {top: 0px;}
}
@-webkit-keyframes bounce{
  from {top: 0px;}
  50% {top: -10px;}
  to {top: 0px;}
}
@-o-keyframes bounce{
  from {top: 0px;}
  50% {top: -10px;}
  to {top: 0px;}
}
@-moz-keyframes bounce{
  from {top: 0px;}
  50% {top: -10px;}
  to {top: 0px;}
}
@-ms-keyframes bounce{
  from {top: 0px;}
  50% {top: -10px;}
  to {top: 0px;}
}

```

Enregistrer et visualiser le résultat.

Notre animation est ajoutée: elle dure 0.7secondes, avec une méthode d'accélération **ease** et de manière **infinite**. Il est possible d'ajouter cette animation sur chaque icone avec un délai de lancement : **exemple pour ico-moz : animation: bounce 0.7s ease-in-out infinite 0.1s;**

### Slider «photos en boucles»

Pour la réalisation du slider nous allons utiliser une **<section>** de type conteneur qui ne laissera entrevoir que certaines images. Les images seront placées au coeur d'une liste que nous déplacerons à tempo régulier.

Afin de donner l'effet «photos en boucles» nous allons lister 6 images puis recopier les 3 premières images à la fin du slideshow à déplacer (soit 9 images au total).

Créer un nouveau document <chap5/08-animation-slider.html> ainsi que la feuille de style [assets/css/styles\\_animation\\_slider.css](assets/css/styles_animation_slider.css).

Dans la page HTML ajouter le code suivant

```

<section id="galerie">
  <ul class="slider">
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</section>

```

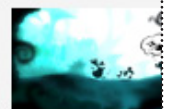
Puis appliquer le css suivant :

```

#galerie{
  width:200px;
  height:100px;
  border:1px solid #CCC;
  margin:20px auto 0px auto;
}
#galerie .slider{
  display: table;
  width:1260px;
  padding-left:25%;
}
#galerie .slider li{
  display: table-cell;
  width:140px;
  height:70px;
  list-style:none;
}
#galerie .slider li img{
  margin-top:7px;
  display: inline-block;
}

```

Enregistrer et visualiser le résultat.



Pour cacher le slider en dehors de la section: ajouter au CSS:

```
#galerie{
  ...
  overflow:hidden;
}
```

Puis nous allons définir l'animation de la manière suivante

```
@keyframes slideAnim{
  from,11%,to {margin-left:-140px;}
  15%,26%{margin-left:-280px;}
  30%,41%{margin-left:-420px;}
  45%,56%{margin-left:-560px;}
  65%,76%{margin-left:-700px;}
  80%,91%{margin-left:-840px;}
  99.99%{margin-left:-980px;}
}
```

Au démarrage de l'animation (**from**) nous décalons le slide de 140px (ce qui correspond à la largeur d'un **<ul>** contenant l'image, ce qui permet d'afficher en premier l'image n°2. A 11% de l'animation la position du slider est toujours de -140px, ce qui donne l'impression d'une image fixe pendant un peu plus de 2 secondes, puis entre 11% et 15% le slider effectue un glissement de -140px à -280px puis garde cette position jusqu'à 26% de l'animation et ainsi de suite.

A 99.99% la position -980px correspond visuellement à la même position que -140px. Juste avant la fin de l'animation, on déplace tout le bloc en position de départ très rapidement (en 0,01% dans ce cas). Cela provoque une superposition des images invisible à l'oeil nu. L'animation reprend donc depuis le début, mais on a l'impression qu'elle continue...

Il ne nous reste plus qu'à placer l'animation dans le slider:

```
#galerie .slider{
  ...
  animation: slideAnim 20s ease 0s infinite;
  -webkit-animation: slideAnim 20s ease-out 0s infinite;
  -moz-animation: slideAnim 20s ease-out 0s infinite;
  -ms-animation: slideAnim 20s ease-out 0s infinite;
  -o-animation: slideAnim 20s ease-out 0s infinite;
}
```

Il ne reste plus qu'à définir nos keyframes pour l'ensemble des navigateurs :

```
@-webkit-keyframes slideAnim{
  from,11%,to {margin-left:-140px;}
  15%,26%{margin-left:-280px;}
  30%,41%{margin-left:-420px;}
  45%,56%{margin-left:-560px;}
  65%,76%{margin-left:-700px;}
```

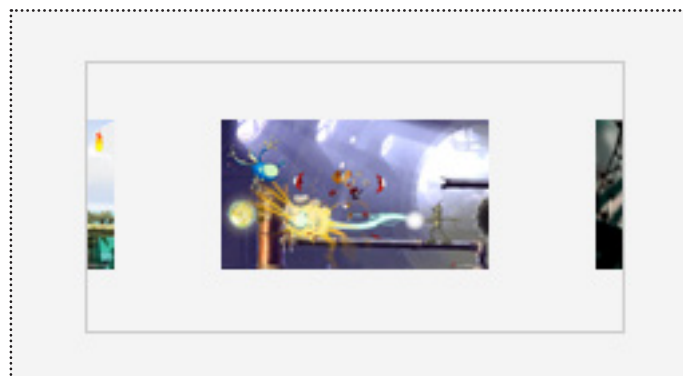
```
80%,91%{margin-left:-840px;}
99.99%{margin-left:-980px;}
}

@-moz-keyframes slideAnim{
  from,11%,to {margin-left:-140px;}
  15%,26%{margin-left:-280px;}
  30%,41%{margin-left:-420px;}
  45%,56%{margin-left:-560px;}
  65%,76%{margin-left:-700px;}
  80%,91%{margin-left:-840px;}
  99.99%{margin-left:-980px;}
}

@-ms-keyframes slideAnim{
  from,11%,to {margin-left:-140px;}
  15%,26%{margin-left:-280px;}
  30%,41%{margin-left:-420px;}
  45%,56%{margin-left:-560px;}
  65%,76%{margin-left:-700px;}
  80%,91%{margin-left:-840px;}
  99.99%{margin-left:-980px;}
}

@-o-keyframes slideAnim{
  from,11%,to {margin-left:-140px;}
  15%,26%{margin-left:-280px;}
  30%,41%{margin-left:-420px;}
  45%,56%{margin-left:-560px;}
  65%,76%{margin-left:-700px;}
  80%,91%{margin-left:-840px;}
  99.99%{margin-left:-980px;}
}
```

Enregistrer et visualiser le résultat.



# Multimédia

Audio

Vidéo

Depuis l'arrivée de Youtube et Dailymotion, il est devenu courant aujourd'hui de regarder des vidéos sur des sites web. Il faut dire que l'arrivée du haut débit a aidé à démocratiser les vidéos sur le Web.

Cependant, aucune balise HTML ne permettait jusqu'ici de gérer la vidéo. Il fallait à la place utiliser un plugin, comme Flash. Encore aujourd'hui, Flash reste de loin le moyen le plus utilisé pour regarder des vidéos sur Youtube, Dailymotion, Vimeo et ailleurs. Mais utiliser un plugin a de nombreux défauts : on dépend de ceux qui gèrent le plugin (en l'occurrence, l'entreprise Adobe, qui possède Flash), on ne peut pas toujours contrôler son fonctionnement, il y a parfois des failles de sécurité. . . Au final, c'est assez lourd.

C'est pour cela que deux nouvelles balises standard ont été créées en HTML5 : `<video>` et `<audio>`

## La lecture audio

La balise **<audio>** que nous allons découvrir est reconnue par tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

En théorie, il suffit d'une simple balise pour jouer un son sur notre page.

*Créer un nouveau document [chap6/01-fichier-audio.html](#).*

```
<audio src="test.mp3"></audio>
```

*Enregistrer et visualiser le résultat.*

En théorie il ne se passe rien. En effet, le navigateur va seulement télécharger les informations générales sur le fichier (on parle de **métadonnées**) mais il ne se passera rien de particulier.

Pour que l'on puisse entendre le fichier audio nous allons ajouter un attribut supplémentaire à la balise:

**autoplay** : la musique sera jouée dès le chargement de la page. Évitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !

```
<audio src="test.mp3" autoplay="autoplay"></audio>
```

*Enregistrer et visualiser le résultat.*

La musique se lance mais l'internaute n'a aucun pouvoir sur celle-ci, et la musique peut devenir très vite perturbante pour lui. Il risque de changer tout simplement de site.

Nous allons indiquer au navigateur que l'on veut afficher les controls du player audio:

**controls** : pour ajouter les boutons «Lecture», «Pause» et la barre de défilement. Cela peut sembler indispensable, et vous vous demandez peut-être pourquoi cela n'y figure pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript.

```
<audio src="test.mp3" autoplay="autoplay" controls="controls"></audio>
```

*Enregistrer et visualiser le résultat.*

Les boutons par défaut sont définis par les navigateurs ce qui donne un aspect différent selon les navigateurs. Pour une harmonie entre les navigateurs vous pourrez avoir envie de définir vos propres

boutons.

```
<audio id="player" src="test.mp3"></audio>
<div>
  <button onclick="document.getElementById('player').play();">Play</button>
  <button onclick="document.getElementById('player').pause();">Pause</button>
  <button onclick="document.getElementById('player').volume+=0.1;">Volume up</button>
  <button onclick="document.getElementById('player').volume-=0.1;">Volume Down</button>
</div>
```

*Enregistrer et visualiser le résultat.*

Il est aussi possible que pour une musique d'ambiance de notre site on veuille écouter notre fichier audio en bloc :

**loop** : la musique sera jouée en boucle.

```
<audio src="test.mp3" autoplay="autoplay" controls="controls" loop="loop"></audio>
```

*Enregistrer et visualiser le résultat.*

Au chargement de la page il se peut que nous souhaitons ne pas lancer la musique mais que celle-ci soit chargée en tâche de fond sans que l'internaute puisse s'en rendre compte. Ceci est le rôle de l'attribut **preload** :

**preload** : indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :

- **auto (par défaut)** : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
- **metadata** : charge uniquement les métadonnées (durée, etc.).
- **none** : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

*On ne peut pas forcer le préchargement de la musique, c'est toujours le navigateur qui décide. Les navigateurs mobiles, par exemple, ne préchargent jamais la musique pour économiser la bande passante (le temps de chargement étant long sur un portable).*

Il reste maintenant à corriger les différentes erreurs engendrées par les nombreux navigateurs. Dans un premier temps nous remarquons que le format **.mp3** n'est pas reconnu par Firefox, mais qui reconnaît le format **.ogg**.

```
<audio controls="controls" >
  <source src="BigBuck.ogg" type="audio/ogg" />
  <source src="BigBuck.mp3" type="audio/mpeg" />
</audio>
```

*Enregistrer et visualiser le résultat.*

Bien sûr il reste toujours les anciennes versions d'IE qui ne reconnaissent pas le player audio. La parade sera d'insérer un Flash et en dernier recours un lien sur le téléchargement du morceau.



```
<audio controls=»controls» >
  <source src=»BigBuck.ogg» type=»audio/ogg» />
  <source src=»BigBuck.mp3» type=»audio/mpeg» />
  <object type=»application/x-shockwave-flash» data=»player.swf?soundFile=BigBuck.mp3»
    <param name=»movie» value=»player.swf?soundFile=BigBuck.mp3» >
    <a href=»BigBuck.mp3»>Télécharger la chanson</a>
  </object>
</audio>
```

*Enregistrer et visualiser le résultat.*

## La lecture vidéo

La balise `<video>` que nous allons découvrir est reconnue par tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

Il suffit d'une simple balise `<video>` pour insérer une vidéo dans la page :

*Créer un nouveau document [chap6/02-fichier-vidéo.html](#).*

```
<video src=»BigBuck.mp4» controls width=»360» height=»240»></video>
```

*Enregistrer et visualiser le résultat.*

De manière général la balise **<video>** se rapproche de la balise `<audio>` et possède les mêmes attributs et les mêmes problématiques liés au format.

**controls** : pour ajouter les boutons Lecture , Pause et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !

**width** : pour modifier la largeur de la vidéo.

**height** : pour modifier la hauteur de la vidéo.

**loop** : la vidéo sera jouée en boucle.

**autoplay** : la vidéo sera jouée dès le chargement de la page. Là encore, évitez d'en abuser, c'est en général irritant d'arriver sur un site qui lance quelque chose tout seul !

**preload** : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :

**auto** (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.

**metadata** : charge uniquement les métadonnées (durée, dimensions, etc.).

**none** : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

Nous pouvons ajouter un dernier attribut:

**poster** : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, je vous conseille d'en créer une !

```
<video controls width=»360» height=»240» poster=»poster320.jpg»>
  <source src=»BigBuck.ogg» type=»video/ogg» />
  <source src=»BigBuck.mp4» type=»video/mp4» />
  <h1><a href=»BigBuck.mp4»>Charger la vidéo</h1>
</video>
```

### Enregistrer et visualiser le résultat.

Une des principales force de pouvoir manier directement notre vidéo au sein de notre page HTML c'est la possibilité de lui appliquer un CSS.

Dans notre fichier HTML placer une balise <div> autour de la balise <video>.

```
<div id=»wrap» >
  <video controls width=»360» height=»240» poster=»poster320.jpg»>
    ...
  </video>
</div>
```

### Créer un nouveau document CSS `assets/css/ styles_video.css`

```
#wrap{
  float:right;
  width:360px;
  height:240px;
  padding:20px;
  background:#fff;
  margin:-20px 30px;

  box-shadow:8px 8px 8px rgba(0,0,0,0.33);
  border-radius:10px;

  -webkit-transform: rotate(5deg);
  -moz-transform: rotate(5deg);
  -o-transform: rotate(5deg);
  -ms-transform: rotate(5deg);
  transform: rotate(5deg);
}
```



**HTML5**  
Page Video

