Créer des sites web : html5 et css3

Chapitre 7 : graphisme et animation en CSS3

Du design avec CSS3

Le web design évolue, il se diversifie énormément grâce aux possibilités qu'offre le langage CSS. Cette évolution du langage permet la création et l'animation d'objets graphique de manière bien plus simple et plus intuitive qu'il y a quelques années.

Regardons un exemple de site internet réalisé en CSS3.
http://www.goetter.fr/

Sur ce site, en plus de la mise en page en 3 colonnes de type responsive, on observe la création de boutons arrondis, une rotation 3D de l'image du logo, une rotation de quelques degrés de certaines images de la colonne de gauche, une animation des icônes du pied de page.

Toutes ces animations font l'objet d'une syntaxe en CSS3.

Comment créer ou modifier des formes en CSS3 ?
Comment animer des objets graphiques en CSS3 ?

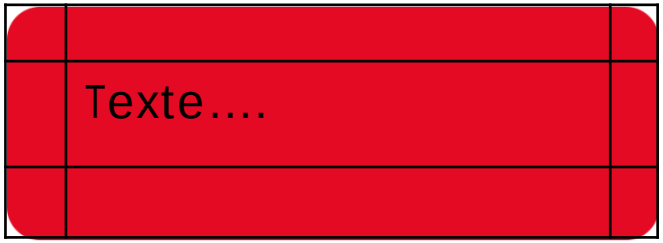
Bloc aux coins arrondis

Vouloir **créer facilement des blocs aux coins arrondis** est un rêve de webdesigner aussi vieux que le Web lui-même.

Il y a encore 10 ans, on pratiquait de la manière suivante :

- Création d'un tableau 3 colonnes et 3 lignes.
- Dans chacune des cellules du tableau, on plaçait en fond (background) une image. La cellules en haut à gauche recevait l'image du coin arrondi haut gauche, la cellule en haut au milieu recevait l'image du haut de l'image etc…

Ce qui donnait
Rq .les bordures du tableau ont été laissées visibles.



Aujourd'hui en CSS3, on utilise la propriété : **border radius.**

```
#cadre
{ Border-radius : 10px; }
```

ici un bloc arrondi

Dans ce cas on obtient un arrondi d'un rayon de 10px à chaque coin du bloc.

Il est possible de **définir l'arrondi de chacun des angles**, à l'aide d'une écriture raccourcie qui se lit dans le sens des aiguilles d'une montre en débutant par le haut (top, right, bottom, left).
Ainsi la règle suivante va créer un bloc arrondi de 5px en haut à gauche, 10px en haut à droite, 0px en bas à droite et 5px en bas à gauche

```
#cadre
{ Border-radius : 5px 10px 0 5px; }
```

ici un bloc arrondi

Ouvrez une page utilisée dans les exercices précédents et créer cet effet sur un élément de votre choix (balises div, article, section, li etc…)

2ème partie : Les ombrages

Avec le CSS3, il est possible de créer facilement des ombrages sur les différents éléments de votre page sans nécessiter d'image décorative particulières.

Box-Shadow

Cette propriété permet de générer une **ombre portée sur n'importe quel élément HTML**. Il est possible d'indiquer le **décalage vertical et horizontal** ainsi que la **force du dégradé et sa couleur**.
La propriété s'applique sur la boite de l'élément, et non sur sa bordure. L'ombrage n'affecte pas la taille de la boite de l'élément.

Exemples :

```
img
{
box-shadow : 8px 8px 0px #aaa;
}
```

```
img
{
box-shadow : 8px 8px 12px #aaa;
}
```

```
img
{
box-shadow : 1px 1px 12px #555;
}
```

Syntaxe dans le cas du 1er exemple

- La première valeur indique le décalage horizontal vers la droite (ici 8px)
- le deuxième correspond au décalage vertical vers le bas (ici 8px)
- le chiffre suivant indique la force du dégradé (ici 0px)
- et enfin, la couleur (ici #aaa)

Ouvrez une page utilisée dans les exercices précédents et créer cet effet sur un élément de votre choix (balises div, article, section, li etc…)

Text-Shadow

C'est une propriété permettant de produire une **ombre portée sur le texte** de contenu sur lequel elle est appliquée.
Il est possible de spécifier les décalages de l'ombrage, la couleur et sa zone de floue. Ces effets s'appliquent dans l'ordre spécifié et peuvent ainsi se recouvrir, mais ceux-ci ne recouvriront jamais le texte lui-même. **L'ombrage n'affecte pas la taille de la boîte de texte.**

```
h1
{
text-shadow : 0px 0px 9px #777;
color : #fff;
}
```

Ombrage sur le texte avec CSS3

Syntaxe

- La première valeur indique le décalage horizontal vers la droite (ici 0px)
- le deuxième correspond au décalage vertical vers le bas (ici 0px)
- le chiffre suivant indique la taille de la zone de flou
- et enfin, la couleur (ici #777 soit gris). Le texte étant en blanc

Analysez l'exemple suivant pour expliquer le rendu obtenu

```
h1
{
text-shadow : 0px 0px 4px #fff,
0 -5px 4px #FFFF33 ,          /*jaune)*/
2px -10px 6px #FFDD33  ,      /*jaune orangé*/
 -2px -15px 11px #FF8800,     /*orange*/,
2px -25px 18px #FF2200,       /*rouge*/
color : #000;                 /*noir*/
}
```

Et encore plus fort !

3ème partie : Les arrières plans

Dès la création des feuilles de styles, la gestion des couleurs et images d'arrière plan est prévue. Ainsi, les styles suivants s'appliquent à la gestion de l'arrière plan :

- **background-color** pour définir une couleur (ex : background-color:#b0c4de;)
- **background-image** pour définir une image(ex : background-image : url(image.gif');)
- **background-repeat** répéter ou non l'image d'arrière plan

(ex : background-image: url('image.png');
background-repeat: repeat-x;) /* ici repeat-x définit une répétition sur l'axe horizontal */

- **background-position** pour positionner l'image d'arrière plan (par défaut en haut à gauche)

(ex : background-image : url(image.jpg');
background-repeat: no-repeat;
background-position: right top;

Tester ces différentes propriétés sur un élément de mise en page : article, section ....

Il existe de nouvelles propriétés en CSS3 :

background-size

La propriété CSS background-size spécifie la taille de l'image dans l'arrière plan.

Syntaxe

background-size: *x y*;

- x détermine la dimension horizontale (pixels, em, auto, pourcentage %, etc.),
- y détermine la dimension verticale (pixels, em, auto, pourcentage %, etc.)

Background-size peut également prendre la valeur **cover**. Dans ce cas, l'image occupe au mieux tout le fond disponible sans déformation.

Un exemple d'utilisation de cette propriété avec le fichier fond-extensible.html

Background multiple

CSS3 permet l'affichage de plusieurs images en arrière plan dans un même élément.
Le résultat est similaire à des calques (ou *strates*) d'un logiciel graphique tel que Photoshop : l'image la plus proche de la propriété (la première énumérée) sera l'image de premier plan.
Si une couleur de fond est déclarée, elle sera toujours reléguée au dernier plan.

Syntaxe

- background-image: ***url("image1"), url("image2")***;
- background-position: *x y*, *x y*;
- background-repeat: *no-repeat*;

L'ordre de déclaration est important : dans l'exemple ci-après, la position left top s'applique uniquement à la deuxième image et "right bottom" s'applique uniquement à la première image.
Si une seule propriété est spécifiée, elle sera appliquée à l'ensemble des images.

```
#deux-fonds {
        background-image: url(image_01.jpg) , url(les_saintes.jpg) ;
        background-position: bottom  right, top  left;
        background-size: auto, cover;
        background-clip: content-box;
        background-repeat: no-repeat;
}
```

Un exemple d'utilisation de cette propriété dans le fichier arriere-plan.html

Pour aller plus loin ....

Utilisation d'une image pour créer une bordure

Une autre propriété intéressante est de pouvoir créer une bordure sur une image ou un bloc en utilisant une image. Voyons l'exemple suivant :

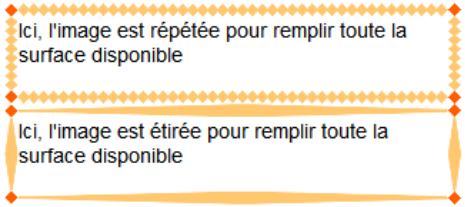
Code HTML

```
< div id="entoure">Ici, l'image est répétée pour remplir toute la surface disponible</div>
< div id="etire">Ici, l'image est étirée pour remplir toute la surface disponible</div>
```

```
/* styles CSS */
#entoure { border-image : url(image.png) 30 30 round; }
#etire {border-image : url(image.png) 30 30 stretch; }
```

Image d'origine

Pour fonctionner sous firefox, il faut spécifier la taille de la bordure (border-width : Xpx;) et son style (border-style : solid;)



Background-clip

La propriété background-clip permet de définir les limites de l'arrière-plan à l'intérieur de la boîte représentée par l'élément.

Syntaxe

- **content-box** : L'arrière plan se limite au contenu (par défaut)
- **border-box** : L'arrière plan s'étend jusqu'a l'extrême limite de la bordure
- **padding-box** : Aucun arrière-plan sera présent en-dessous de la bordure : l'extrême limite sera celle du padding

rq. cette dernière propriété n'est pas encore complétement implémentée

Attention ces effets ne fonctionnent pas encore avec tous les navigateurs dont, comme c'est curieux, internet explorer.

Vous retrouverez un exemple d'utilisation de ces propriétés dans les fichiers : bordure_ombrage.html et arriere-plan.html

4ème partie : Les dégradés

De nouvelles propriétés permettent de réaliser et gérer des arrière-plans de teintes dégradées. Il s'agit de **linear-gradient** pour les dégradés linéaires et **radial-gradient** pour les dégradés radiaux.

linear-gradient

Les dégradés linéaires empruntent la valeur-fonction linear-gradient() de la propriété background-image (ou background en raccourci).
Il convient de préciser au **minimum** : **le point d'arrivée** du dégradé ainsi que **deux couleurs** (ou plus).

background : linear-gradient(*x y*, *couleur1*, couleur2, *couleurN*)

Valeurs possibles

- x et/ou y définissent la direction du dégradé. Exemple : "to right" ou "to left bottom", ou même un angle en degré : 45deg.
- couleur : le nom, la valeur hexadécimal ou la valeur RGBa

rq. on peut ajouter un % pour indiquer l'arrêt de la couleur.

Exemples :

- **Dégradé du rouge au violet**

background: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);

- **Dégradé du vert au transparent**

background: linear-gradient(to right, green, rgba(0,255,0,0));

Tester cette propriété sur un élément de type block.

radial-gradient

Les dégradés radiaux sont réalisables à l'aide de la valeur-fonction radial-gradient().
Elle nécessite d'indiquer la **forme** du dégradé : **circulaire** (circle) ou **elliptique** (ellipse) ainsi que son **point de départ** (x et y)

```
background : radial-gradient(forme at x y, couleur1, ..., couleurN)
```

Valeurs possibles

x et/ou y définissent le point d'origine du dégradé radial,
couleur : le nom, la valeur hexadécimal ou la valeur RGBa

Exemple

```
background: radial-gradient(ellipse at center top, green, yellow);
```

Vous retrouverez tous ces exemples précédents dans le fichier : degrade.html

Nous avons donc vu des propriétés CSS3 pour transformer l'aspect graphique d'un objet (image ou boite : article, section, div...)

Regardons maintenant comment animer ces différents objets...

5ème partie : Les transformations 2D

CSS3 apporte les transformations en 2 dimensions à travers la propriété **transform** et une liste de fonctions prédéfinies.
La propriété CSS **transform** permet de manipuler un élément HTML sur les axes X et Y (horizontal et vertical) grâce à des fonctions diverses de transformation graphique. Il est donc possible de modifier l'apparence d'un élément grâce à un ensemble fonctions 2D :

- **Translation (translate),**
- **Mise à l'échelle (scale),**
- **Rotation (rotate)**
- **Inclinaison (skew)**

La propriété transform-origin

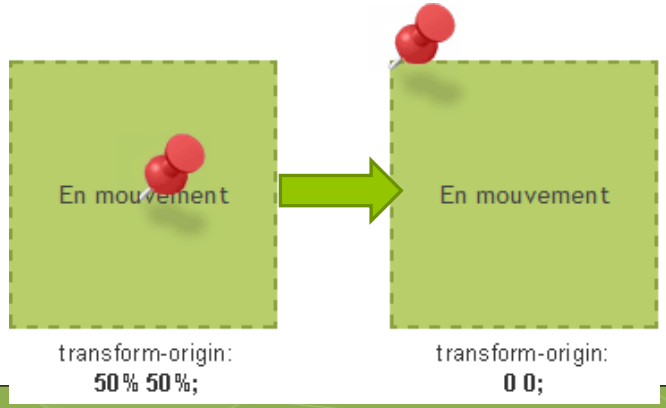
Pour pouvoir appliquer des transformations, nous avons besoin de savoir quel est le point d'origine (d'ancrage) de la transformation.
La propriété **transform-origin** définit ce point d'origine.
La valeur initiale de cette propriété est le centre de l'élément, ce qui équivaut à la notation :

- **transform-origin: 50% 50%;**

Il est possible de changer cette valeur en utilisant un mot-clef de position **(top,right, bottom, left)** suivi d'une valeur chiffrée dont l'unité peut varier (px, %, etc.)

Exemple de transformation de l'origine d'une boite contenant le texte : En mouvement

```
div
{
 transform-origin: top 0 left 0;
}
```



Une fois l'origine choisie, nous pouvons affecter des transformations à nos éléments.
La syntaxe est simple d'emploi.

transform: *function*(*value*);

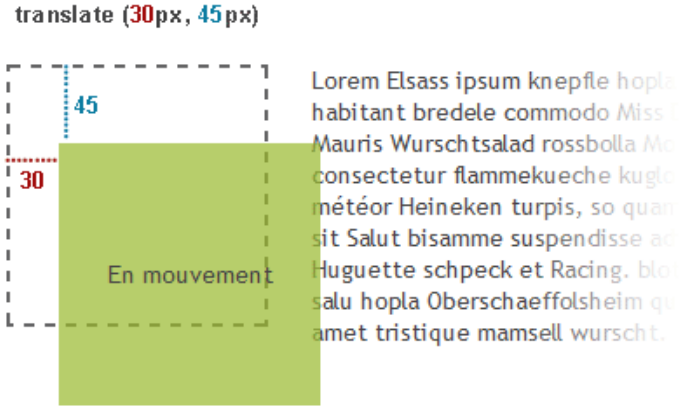
Les fonctions de la propriété transform

La fonction translate

Elle permet d'effectuer une translation (un déplacement) de l'élément sur les axes X et Y.

transform: **translate**(*x*,*y*);

y est une valeur optionnelle équivalente à 0 si elle n'est pas renseignée.
Les deux valeurs peuvent être négative.
Il n'y a dans cette transformation aucune notion de flux.



L'application de cette transformation nécessite une interaction avec l'utilisateur de la page. Ainsi, il est courant de déclencher cette transformation lors du survol du curseur de la souris sur l'élément à animer.

Pour cela, on utilise la pseudo-classe **hover** qu'on applique à l'élément et qui définira donc tous les effets à appliquer au survol de l'élément.

Exemple pour une boite se nommant toto.

```
#toto:hover {
transform:translate(10px, 65px);
}
```

Tester cet effet sur un bloc de votre choix. Modifier le point d'origine de la transformation et observer.

Les fonctions de la propriété transform

La fonction scale

Cette fonction permet d'agir sur l'échelle (les dimensions) de l'élément. La valeur initiale est **1**, tandis que les valeurs supérieures à 1 créent un effet d'agrandissement, et les valeurs inférieures créent un effet de réduction.

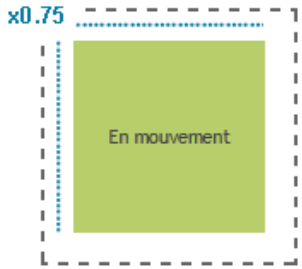
transform: **scale**(*x*,*y*);

La valeur y est optionnelle et sera égale à la valeur de x si elle est non renseignée. Ainsi, si x est uniquement renseignée, on obtient alors une mise à l'échelle proportionnelle sur x et y.

Exemple de transformation pour un agrandissement d'un facteur 1.25 et une réduction d'un facteur 0.75

transform: **scale**(*1.25*);

transform: **scale**(*0.75*);



Les fonctions scaleX et scaleY

Sur le même principe que pour les fonctions dérivées de translate, ces deux fonctions permettent de définir indépendamment les valeurs x et y.

Les fonctions de la propriété transform

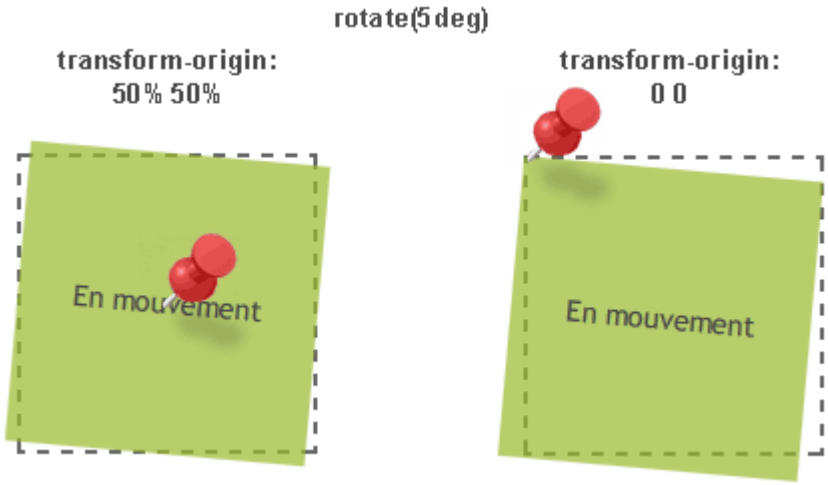
La fonction rotate

Il s'agit d'une des plus simples fonctions à comprendre. Comme son nom l'indique, elle permet d'effectuer une **rotation** de l'élément ciblé

transform: **rotate(valeur)**;

La valeur exprime le degré de rotation (exprimée en degrés). Elle peut être négative et supérieure à 360°.
Ce dernier point n'a de réel intérêt que lors d'une animation d'un état à un autre afin de présenter, par exemple, une rotation de plusieurs tours d'un élément. Autrement, sans animation de la rotation, la valeur 380° équivaut visuellement à une rotation de 20°.

transform: **rotate(5deg)**;



Dans cet exemple, on comprend mieux l'intérêt du point d'origine de la transformation.

Les fonctions skewX et skewY

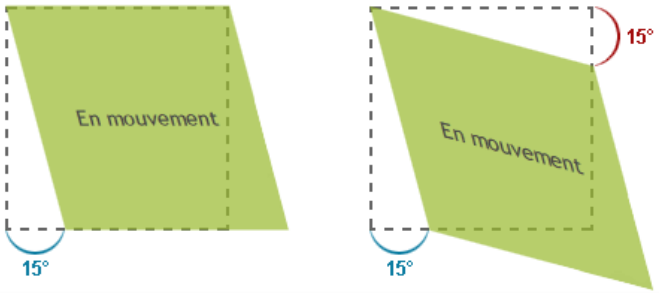
Ces fonctions permettent d'obliquer la forme d'un élément

transform: **skewX (valeur)**;

La valeur exprime le degré de rotation (exprimée en degrés).

transform: **skewX (15deg)**;

transform: **skewX (15deg) skewY(15deg)**;



Là aussi, cet exemple illustre bien l'intérêt du point d'origine de la transformation.

Exercice

Créer dans une page HTML, 4 blocs à l'aide de balises
Donner un identifiant distinct à chacun de ces blocs.
Créer un style pour donner aux blocs les caractéristiques suivantes :

- hauteur et largeur : 150px
- fond de la couleur que vous souhaitez mais transparent à 50%
- une origine de transformation située en haut à gauche.

Créer un style qui au survol du bloc (#nom du bloc : hover {.....})permet :

- une translation pour le bloc 1
- une modification d'échelle pour le bloc 2
- une rotation pour le bloc 3
- une modification d'obliquité pour le bloc 4

Correction dans la page : transformations.html

6ème partie : Les transitions

Ce module de CSS3 permet d'animer les pages web uniquement à l'aide de CSS sans apport de JavaScript.

Principe de base

Le principe de base d'une **transition CSS3** est de permettre une transition douce entre l'ancienne valeur et la nouvelle valeur d'une propriété CSS lorsqu'un événement est déclenché :

- soit via une pseudo-classe telles que **:hover, :focus ou :active**
- soit via JavaScript

Précédemment, ce genre de comportement n'était possible qu'avec l'usage de JavaScript. Ce nouveau module CSS3 permet dorénavant de s'en affranchir au profit exclusif des feuilles de style.

Pour définir une nouvelle transition animée, il est nécessaire de préciser au minimum :

- La ou les propriété(s) à animer
- La durée de l'animation

Les propriétés de transition CSS

Les **deux propriétés minimales nécessaires** pour rendre fonctionnelle une transition en CSS 3 sont : **transition-property et transition-duration.**

Il existe d'autres propriétés CSS spécifiques aux transitions : **transition-timing-function, transition-delay**.

Propriété	Explication
transition-property	Précise les propriétés CSS à transformer
transition-duration	Précise la durée de la transition
transition-timing-function	Précise la fonction de transition à utiliser, le modèle d'interpolation (accélération, décélération...)
transition-delay	Précise le retard (ou l'avance) du départ de la transition

Exemple : transformation progressive d'un carré en rectangle

Créer la page web et la feuille de style CSS suivantes. Expliquer l'effet obtenu.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles/transition.css">
</head>
<body>

<p>
<strong>Note:</strong> Cet exemple ne fonctionne pas avec internet explorer 9 ainsi qu'avec les versions antérieures.</p>

<div></div>

</body>
</html>
```

```
/* la feuille de style transition.css */

div
{
width:100px;
height:100px;
background: red;
transition-property: width;
transition-duration:1s;
transition-delay:1s;
}

div:hover
{
width:200px;
}
```

Ajouter sur la page un lien quelconque : balise <a>
Appliquer sur ce lien une transition pour qu'au survol du lien, la couleur change et la taille augmente progressivement.

Correction et d'autres exemples avec la page : transitions.html

La propriété transition-property

La propriété **transition-property** permet de définir quelles propriétés seront affectées par les transitions.
Ces propriétés peuvent être précisées en les listant séparées par une virgule.
Dans l'exemple précédent on peut donc écrire :

```
a
{
color: green;
font-size: 16px;
transition-property:   color,   font-size;
transition-duration:1s;
}

a:hover
{
color: red;
font-size:  24px;
}
```

Par défaut, la valeur de transition-property est **all**, ce qui signifie que toutes les propriétés animables le sont.

Vous pouvez consulter la liste des propriétés animables à l'adresse suivante : http://www.w3.org/TR/css3-transitions/#properties-from-css-

La propriété transition-duration

Cette propriété permet de préciser la durée de la transition. Si plusieurs propriétés ont été précisées à l'aide de la propriété précédente, il est possible de préciser plusieurs valeurs pour cette propriété en les séparant également d'une virgule.
Les valeurs acceptées sont des valeurs de temps. Une valeur de temps est donnée par un nombre suivi d'une unité de temps. Les deux unités de temps définies en CSS sont :

- **s** : la seconde
- **ms** : la milliseconde

La propriété transition-timing-function

Pour calculer les valeurs intermédiaires, le navigateur procède à une interpolation. La fonction calculant la valeur de cette interpolation influence directement la fluidité de l'animation.

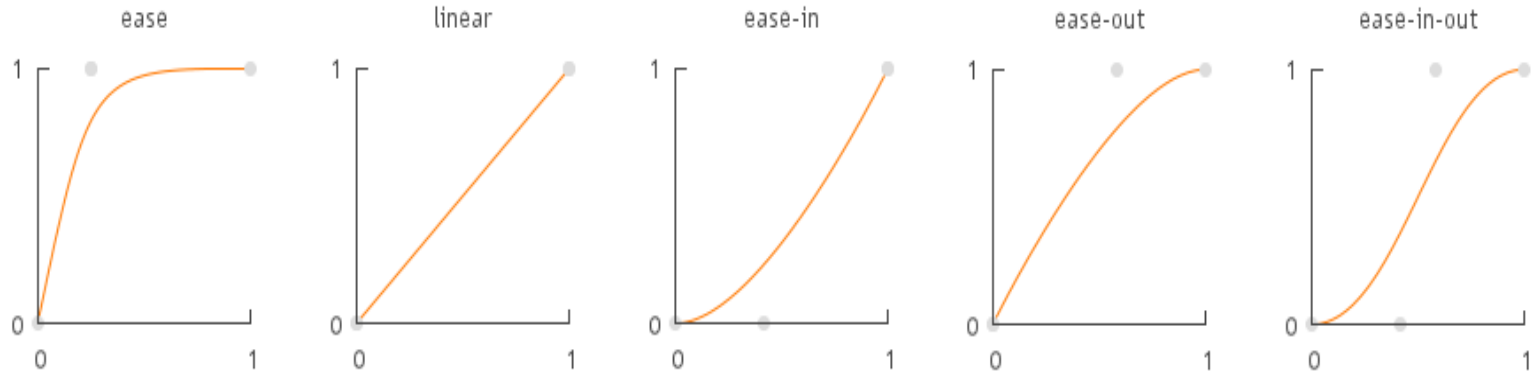
Les fonctions prédéfinies sont les suivantes :

- **ease** : Rapide sur le début et ralenti sur la fin.
- **linear** : La vitesse est constante sur toute la durée de l'animation.
- **ease-in** : Lent sur le début et accélère de plus en plus vers la fin.
- **ease-out** : Rapide sur le début et décélère sur la fin.
- **ease-in-out** : Le départ et la fin sont lents.

Ces mots-clés sont des valeurs admissibles pour la propriété transition-timing-function.

Les fonctions sont illustrées dans la figure suivante.

Plus la courbe monte fortement et plus l'animation sera rapide sur cette portion.



La propriété transition-delay

La transition commence, par défaut, dès que la propriété est changée suite à l'événement. La propriété **transition-delay** permet d'adapter ce comportement en retardant ou en avançant le début de la transition.

Exercices

Exercice 1

Créer un bloc (balise) contenant un texte quelconque. Ce bloc à une taille de 200px de hauteur et de largeur.

Créer la feuille de style qui au survol de ce bloc permet les transformations suivantes :

- **le bloc s'agrandit de 50 pixels en largeur et hauteur**
- **les couleurs du fond et du texte changent**
- **la bordure devient en pointillé**
- **le bloc tourne de 90 degrés**

Aide : border-style : dashed; pour une bordure en pointillé.

Exercice 2

Créer un bloc contenant deux images de même taille.

Les deux images doivent être positionnées l'une sous l'autre, ainsi l'image du dessus cache complétement celle du dessous.

Créer la feuille de style qui permet lors du survol du bloc :

- **de faire disparaitre l'image du dessus et de faire apparaitre l'image du dessous de manière progressive.**
- **Inversement quand le bloc n'est plus survolé.**

Aide : utiliser le positionnement absolu ainsi que la propriété opacity (1 : visible; 0 : complétement transparent)

Exercice 3

Créer deux blocs portant une identification précise (volet1 et volet2), **le deuxième bloc étant contenu dans le premier.**

Affecter à chacun des blocs les images de fond utilisées dans l'exercice précédent.

En jouant avec les propriétés CSS de positionnement, placer le deuxième bloc (volet2) exactement sous le premier (rq. Les images utilisées ont 200px de largeur et 203px de hauteur)

A ce niveau de l'exercice, vous devriez , dans votre navigateur, observer les deux images l'une sous l'autre.

Modifier la feuille de style pour :

- **qu' à l'écran, on ne voit plus l'image contenu dans le volet 2.**
- **un survol du volet1 modifie le positionnement du volet2 afin que l'image du volet2 se positionne exactement sur celle du volet1.**
- **ce changement de position devra s'effectuer en douceur.**

Aide : Le volet 2 étant inclus dans le volet1 dans le fichier html, il est possible de le cacher en écrivant ***overflow:hidden;*** dans le fichier css pour les propriétés du volet1.

Utiliser la pseudo-classe :hover appliquée au volet 1 ou au volet 2 pour déclencher le changement de position

Correction dans la page : exercices.html

Pour aller plus loin...beaucoup plus loin....

La possibilité de créer des animations d'objets sans utiliser d'autres logiciels tel que flash ou d'autres langages comme javascript est assez récent.
Encore une fois, internet explorer 9 et les versions précédentes ne gèrent pas les propriétés d'animation.

@keyframes (ou images clés)

Avant de pouvoir animer un élément que vous aurez créé dans votre page, il faut maitriser les **@keyframes**.
Celles-ci définissent l'état initial, les états intermédiaires et l'état final de votre animation.
Ainsi, elles définissent le changement d'état progressif depuis le style d'origine vers le style final.

La syntaxe est la suivante :

```
@keyframes nom_de_l_animation
{
définition de l'état initial
définitions des états intermédiaires
définition de l'état final
}
```

Exemple :

```
@keyframes mon animation
{
from { background : red; }
to {background: yellow; }
}
```

Dans cet exemple, les états intermédiaires ne sont pas définis, ce n'est pas forcément obligatoire.

Rq. dans le cas des navigateurs Chrome et Safari, il faut utiliser un préfixe sur les @keyframes pour qu'elles soient reconnues.
Ainsi : **@keyframes** devient **@-webkit-keyframes**

Une fois que les images clés de votre animation sont crées, vous devez indiquer sur quel élément elles s'appliqueront.
Ainsi, vous devez lier votre animation avec un élément de mise en forme (en général une balise mais ce n'est pas une obligation) en précisant deux propriétés d'animation:

- **le nom de l'animation que vous avez défini dans lors de la création des @keyframes**
- **la durée de l'animation.**

Exemple

```
@keyframes monanimation
{
from { background : red; }
to {background: yellow; }
}

div
{
animation : monanimation 5s;
-webkit-animation : monanimation 5s;
}
```

Remarque :
L'utilisation de version ancienne des différents navigateurs disponibles peut entraîner quelques surprises (désagréables ?). Ainsi, les propriétés de transition, de transformation et d'animation nécessitent quelques aménagements.
Il est nécessaire d'utiliser des préfixes avant les propriétés :
-moz- pour mozilla
-ms- pour internet explorer
-o- pour opéra
-webkit- pour chrome et safari

Créer une page web contenant une balise .
Modifier la largeur et la hauteur de cette balise pour lui donner l'aspect d'un carré puis appliquez-lui un fond rouge.
Liez à cette balise l'animation définie dans l'exemple ci-dessus.

Il existe une autre syntaxe permettant de définir les changements d'états de votre animation.
La syntaxe est la suivante :

```
@keyframes monanimation
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

 /* Safari and Chrome */
@-webkit-keyframes monanimation
{
0%   {background: red;}
25%  {background: yellow;}
50%  {background: blue;}
100% {background: green;}
}

div
{
animation : monanimation 5s;
-webkit-animation : monanimation 5s;
}
```

Dans cet exemple, on obtient un changement de la couleur de fond de la balise div, qui passera progressivement du rouge au vert en passant par le jaune et le bleu.
On peut spécifier d'autres modifications en plaçant les propriétés les unes à la suite des autres et en les séparant d'un ;)
Ex : 25% {background : yellow ; opacity:1; etc…}

Reprenez l'exemple précédent en modifiant les @keyframes pour que le carré change de position en plus du changement de couleur.

Correction dans la page : animations.html

Propriétés d'une animation CSS3

Il existe plusieurs propriétés permettant de personnaliser davantage les animations. Celles-ci sont très similaires à celles utilisées dans les transitions (cf. partie précédente).

Propriété	Description
animation	Raccourci d'écriture pour définir les unes à la suite des autres les différentes propriétés de l'animation
animation-name	Le nom de l'animation définit dans les @keyframes
animation-duration	La durée de l'animation. (s ou ms)
animation-timing-function	Précise la progression de l'animation : le modèle d'interpolation : linear, ease-out etc…
animation-delay	Précise le retard ou l'avance du départ de l'animation
animation-iteration-count	Précise le nombre de fois que l'animation est "jouée" (1 par défaut, infinite pour une animation en boucle)
animation-direction	Précise si l'animation se joue à l'endroit ou à l'envers et ce de manière alternative.
animation-play-state	Précise si l'animation démarre en pause ou si elle est active.(running)
animation-fill-mode	Définit l'état de départ et de fin de votre animation. Ex : forwards: indique au navigateur de laisser l'élément dans sont état final lors de la dernière itération…….

Concernant le raccourci d'écriture, un exemple simple permet de comprendre

Cette écriture est identique à celle-ci.

```
@keyframes monanimation
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

div
{
width:100px;
height:100px;
background: red;
position: relative;
animation-name: monanimation;
animation-duration:5s;
animation-timing-function: linear;
animation-delay:2s;
animation-iteration-count: infinite;
animation-direction: alternate;
animation-play-state: running;
}
```

```
@keyframes monanimation
{
0%   {background: red; left:0px; top:0px;}
25%  {background: yellow; left:200px; top:0px;}
50%  {background: blue; left:200px; top:200px;}
75%  {background: green; left:0px; top:200px;}
100% {background: red; left:0px; top:0px;}
}

div
{
width:100px;
height:100px;
background: red;
position: relative;
animation: monanimation 5s linear 2s infinite
alternate;}
```

Un exemple à étudier : carrousel.html

Sitographie :
L'ensemble des chapitres présentés ont pour source les sites suivants :

La référence en matière d'apprentissage des règles correctes de CSS (en anglais):
http://www.w3schools.com/default.asp

Site très intéressant, avec de nombreux exemples et en français :
http://www.alsacreations.com/apprendre/
http://www.alsacreations.com/astuce/lire/1216-arriere-plan-background-extensible.html

Un tutoriel à suivre pour créer des sphères. Ce dernier reprend beaucoup d'éléments étudiés dans ce chapitre et est donc un excellent exercice.
http://hop-ie.developpez.com/tutoriels/css3/spheres-css3/

Excellent site pour revoir les bases et un peu plus, simple et clair.
http://fr.openclassrooms.com/informatique/cours/apprenez-a-creer-votre-site-web-avec-html5-et-css3/

D'autres sites à visiter :
http://www.the-art-of-web.com/css/
http://caniuse.com/