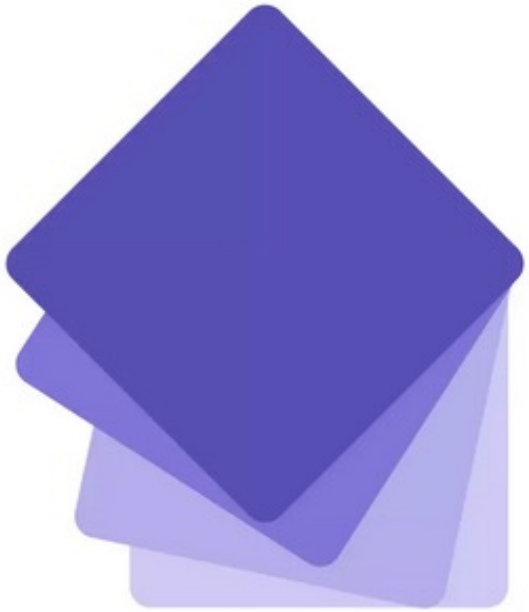


ANIMATIONS EN CSS

Jean-Christophe **DUBOIS**
jean-christophe.dubois@univ-rennes.fr

Pseudo-classe

- Les pseudo-classes permettent
  - de préciser un sélecteur en fonction d'une situation spécifique.
  - de définir une mise en forme propre à la situation décrite :
    - :focus : focus d'un champ texte dans un formulaire
    - :checked : sélection d'un bouton radio ou d'une case à cocher dans un formulaire
    - :hover : survol d'un élément
    - :active : activation d'un élément lors du clic (ou tab)
    - :link, :visited : lien affiché, lien visité
    - :target : destination d'une ancre

```
selecteur:pseudo-classe
{ propriété: valeur; }
```

Pseudo-classe

- :target est utilisé en lien avec la définition d'une ancre.
  Si un lien vers une ancre permet d'accéder à une zone précise d'une page, :target permet d'activer les changements au niveau de la zone atteinte.
- HTML :

```
<a href="#ancre">Accès à la formation</a>
  …
<h1 id="ancre"> Présentation formation </h1>
```

- CSS :

```
#ancre { color: red;}                    // ancre
#ancre:target { color: white;    // ancre atteinte
                background-color: red}
```

Exercice

- Mettre en majuscule et en jaune pâle sur un fond rouge, les titres atteints dans la page du Louvre

- Mettre le texte en blanc lors du survol d’un article dans la barre latérale

Transition

- Une transition permet le changement d'une valeur d'une propriété de façon progressive.
- Elle est définie par 4 valeurs:
  - la(les) propriété(s) CSS affectée(s) par l'effet : *couleur, dimension, position, inclinaison…*
  - la durée de la transition
  - le délai avant déclenchement (opt.)
  - l'accélération (opt.)
- Le déclenchement d'une transition peut se faire :
  - via une pseudo-classe : `:hover`, `:focus`, `:active`, `:target`, `:checked` ...
  - via une modification programmée en JavaScript

Transition

- transition-property : propriété(s) concernée(s) par la transition

```
transition-property: width | height
                     | color | font
                     | background-color
                     | margin| padding …
                     | all;
```

- transition-duration : durée de la transition en seconde (s)
  ou en milliseconde (ms)

```
transition-duration:3s
                    | .5s
                    | 500ms;
```

Transition

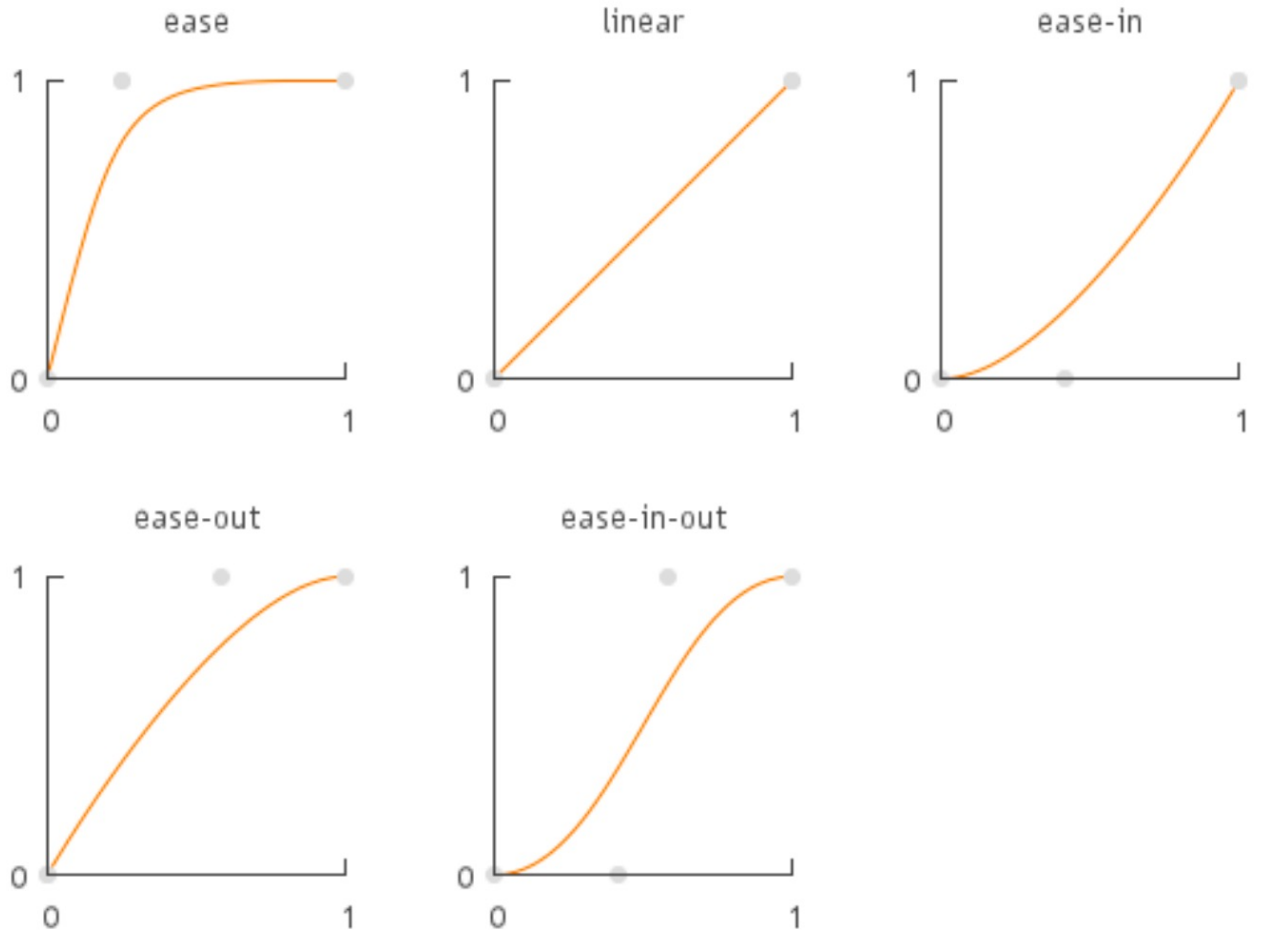
- transition-timing-function : accélération de la transition

```
transition-timing-function:
                    ease            // rapide-lent
                   |linear          // linéaire
                   |ease-in         // lent-normal
                   |ease-out        // normal-lent
                   |ease-in-out     // lent-normal-lent
                   |cubic-bezier(n,n,n,n);
                                    // courbe d'accélération personnalisée
                                    // valeurs de n comprises entre 0 et 1
```

Transition

- transition-timing-function : accélération de la transition

`transition-timing-function:`



https://www.alsacreations.com

Transition

- transition-delay : délai de déclenchement en seconde (s) ou en milliseconde (ms)

```
transition-delay: 3s
                  |.5s
                  |500ms;
```

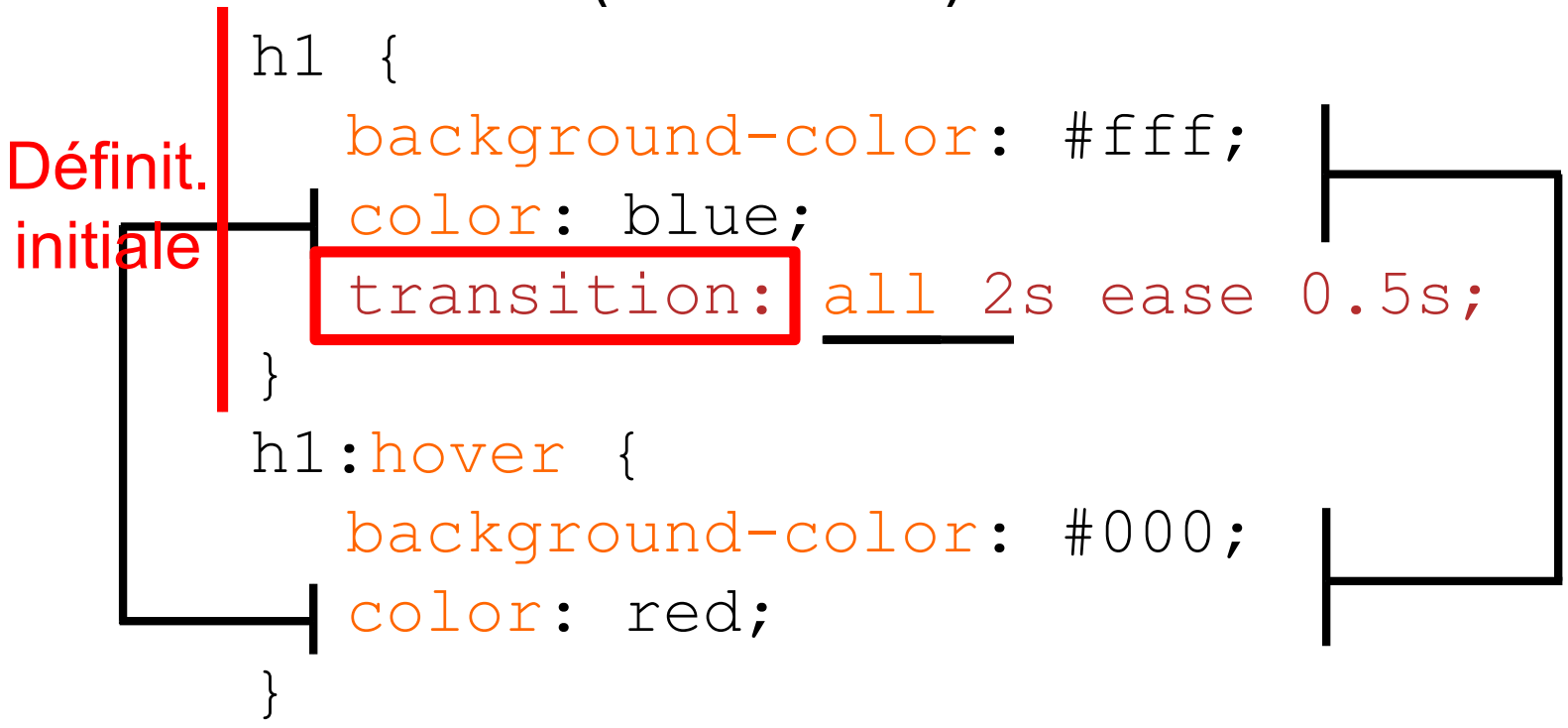
✓ Syntaxe raccourcie :

- transition : property duration timing-function delay;

```
transition: color 4s ease 1s;
transition: all 200ms;
```

Transition

- Une transition nécessite 2 mises en forme:
  - la 1ère est décrite lors de la définition initiale de l'élément
  - la 2nde est appliquée lorsqu'un événement survient sur l'élément (ici le survol)



Exercice

- Modifier en ½ seconde l'approche des lettres (5px) lors du survol d'un titre de niveau 3 après 1 dixième de seconde d'attente. La transition se fera doucement au début et à la fin.

Transformation

- transform: propriété exécutant une ou plusieurs fonctions de transformation (intéressantes si appliquées via une transition)

```
transform: none | fonction1() fonction2() … ;
```

Attention, l'ordre dans lequel elles sont indiquées change le résultat final

- Les fonctions de transformation se regroupent en 3 catégories :
  - les transformations 2D : `skew`
  - les transformations 2D/3D : `translate`, `scale`, `rotate` et `matrix`
  - les transformations 3D : `perspective-origin` et `perspective`

Transformation

- transform-origin : point d'origine de la transformation
  par défaut, le point central de l'élément

```
transform-origin: 50% 50% | initial;     // par défaut
transform-origin: hpos vpos;
```

avec `hpos:` `left |right |center`
`vpos:` `top |bottom |center`

- transform-style : détermine si les éléments enfants d'un conteneur, qui subissent une transformation, conservent ou non leur positionnement 3D.

```
transform-style: flat;                   // aplatissement
transform-style: preserve-3D;            // animation 3D
```

- backface-visibility : définition de la visibilité de la face arrière

```
backface-visibility: hidden | visible;
```

Fonctions de transformation 2D

- translateX(*valX*) | translateY(*valY*) | translateZ(*valZ*) | translate(*valX*) | translate(*valX*, *valY*) : translation

```
transform: translate(8px);     // vers la droite de 8px
transform: translateY(-5px);   // vers le haut de 5px
transform: translate(10px, 20px);
```

- scaleX(*valX*) | scaleY(*valY*) | scaleZ(*valZ*) | scale(*val*) | scale(*valX*,*valY*) : agrandissement

```
transform: scaleX(0.5);  // Réduction de la largeur par 2
transform: scale(2);     // Agrand. par 2 en larg. et haut.
transform: scale(2, 4);  // Agrandissement par 2 en largeur
                         //  et 4 en hauteur
```

valX: longueur
valY: hauteur
valZ: profondeur

Fonctions de transformation 2D

- rotate(*Xdeg*) : rotation

```
transform: rotate(45deg);      // 1/8ème de tour dans le sens
                               // des aiguilles d'une montre
transform: rotate(-90deg);     // ¼ de tour dans le sens inverse
                               // des aiguilles d'une montre
```

- skew(*Xdeg*,*Ydeg*) | skewX(*Xdeg*) | skewY(*Ydeg*) : inclinaison

```
transform: skewX(30deg);       // inclinaison de 30 degrés en X
transform: skew(10deg, 25deg); // inclinaisons de 10 degrés
                               // en X et 25 en Y
```

Fonctions de transformation 2D

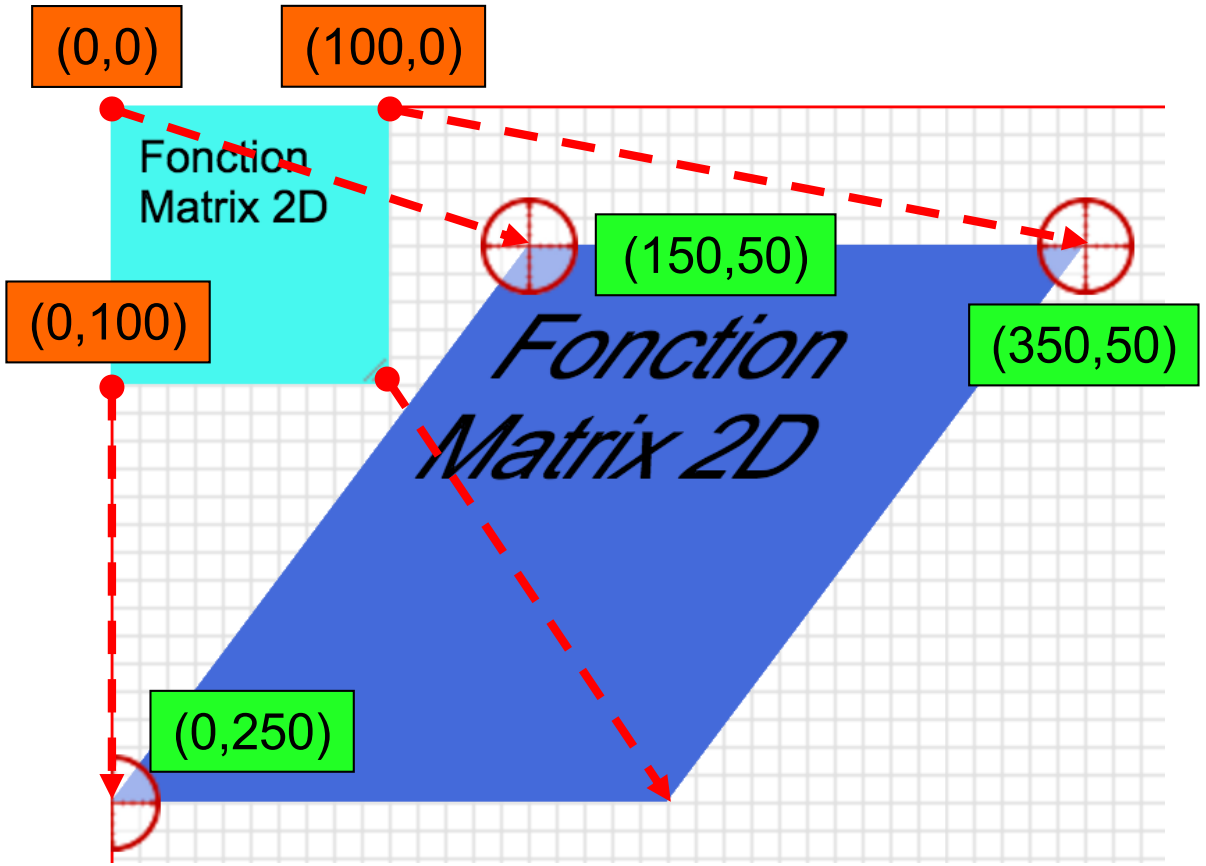
- matrix(*a, b, c, d, e, f*) : combinaison de rotation, d'échelle, de déformation et de translation qui se traduit par une matrice

```
transform: matrix(2, 0, -1.5, 2, 150, 50);
```

Nouvelles coordonnées du point (0,0) :

X = a*x + c*y + e
= 2*0 + -1,5*0 + 150
= 150

Y = b*x + d*y + f
= 0*0 + 2*0 + 50
= 50



Source : http://www.useragentman.com/matrix

Exercice

- Appliquer sur chacun des 4 tableaux surréalistes des transformations (durée d'1 seconde) déclenchées au survol
  - rotation : rotation verticale d'un tour
  - translation : déplacement de 2em vers le haut et vers la droite
  - agrandissement : doubler la largeur et diminuer de 1/2 la hauteur
  - et inclinaison verticale de 30°

Fonctions de transformation 2D

- translate: translation en x, en x et y, en x, y et z

```
translate: 10px;                  // en x
translate: 10px -20px;            // en x et y
translate: 10px -20px 30px;       // en x, y et z
```

- rotate: rotation en deg/rad en x, y et z

```
rotate: 45deg;                    // en z, par défaut
rotate: x 120deg;                 // en x
rotate: y -280deg;                // en y
```

- scale: agrandissement en x et y

```
scale: 2;          // fois 2 en largeur (x) et hauteur (y)
scale: 2 0.5;      // fois 2 en x et divisé par 2 en y
scale: 200% 50%;   // fois 2 en x et divisé par 2 en y
```

Animation

- Définie par la propriété animation, une animation nécessite :
  - des images clés définies à l'aide de la commande `@keyframes`
  - une référence à ces images clés dans la propriété animation
- Définition d'images clés

```
@keyframes monanimation {
    0%  {                       // début de l'animation (from)
        propriete1: valA1; }
    50% {                       // étape(s) intermédiaire(s) optionnelles
        propriete2: valB1; }
    100%{                       // fin de l'animation (to)
        propriete1: valA2;
        propriete2: valB2;   }          }
```

- Référence à l'animation

```
selecteur:pseudo-classe {
  animation: monanimation duree acceleration delai
             répétition direction conservation}
```

Animation

- La propriété animation est un raccourci des propriétés suivantes :
  - animation-name : nom de l'animation
  - animation-duration : temps total de l'animation exprimé en milliseconde *ms* ou en seconde s
  - animation-timing-function : accélération/décélération de l'animation. Outre les valeurs ease (par défaut), linear, ease-in, ease-out, ease-in-out, cubic-bezier(*n, n, n, n*) il est possible de définir un mode image par image en indiquant le pas voulu : steps(nombre, start | end)
  - animation-delay : délai avant que l'animation ne démarre

Paramétrage d'une animation

- animation-iteration-count : nombre de répétition de l'animation.

```
animation-iteration-count: 1          // par défaut
                          |infinite   // en boucle
                          | 2 | 3 | 4;
```

- animation-direction : sens de lecture de l'animation

- `normal` : animation jouée de l'étape 1 à n.
- `reverse` : animation jouée de l'étape n à 1, en sens inverse
- `alternate` : sens normal d'abord (cycles impairs 1,3,5...) et sens inverse ensuite (cycles pairs 2,4,6...)
- `alternate-reverse` : sens inverse d'abord (cycles impairs 1,3,5...) et sens normal ensuite (cycles pairs 2,4,6...)

Paramétrage d'une animation

- animation-play-state : définition de l'état de l'animation

```
animation-play-state: running  // en marche
                               // valeur par défaut
                    | paused;  // en pause
```

- animation-fill-mode : conservation d'un état après l'animation

```
animation-fill-mode: forwards     // état final
                   | backwards;   // état initial
```

Exercice

- A l'aide d'une animation, faire rebondir à 3 reprises une image, comme le ferait une balle, sur une durée de ½ seconde.
  Le mode de vitesse choisi doit être constant mais l'accélération du rebond est gérée via l'animation elle-même.

Exercice

- Programmer une « image survolée » présentant 2 images contenues dans 2 fichiers image

Exercice

- Programmer un diaporama présentant 2 images contenues dans 1 seul fichier (2 images côte à côte)

Exercice

- Programmer, au survol, un affichage d'une légende de photo
  - Code HTML
  - Code CSS