**CSS3 - animation**

* Les animations avec CSS
* Support par les navigateurs
* Propriétés CSS associées
* Contrôler une animation ● Utilisation de JavaScript
* Outils de test
* Applications (slideshow, sprite,...)

# Les animations avec CSS

* Les **transitions** avec CSS ne permettent que des animations simples :
  + limitées à un état de début et de fin pour la définir
  + limitées dans le contrôle (déclenchée par des interactions souris) si on n’utilise pas JavaScript
* Pour avoir un contrôle plus abouti, les animations CSS ont été proposées. On peut définir **autant d’états différents** que nécessaires et on peut préciser la **répétition** de l’animation ainsi que son **état final** et son **délai de déclenchement**.

# Support par les navigateurs

Les propriétés CSS3 liées aux **animations** sont supportées depuis de nombreuses versions https://caniuse.com/#search=css3 animation

# @keyframes

* Pour **définir les différents états**, on utilise des **keyframes** (traduit par images-clés ou états-clés)
* Les **étapes** de l'animation vont être identifiées par un **pourcentage** ou l'un des mots-clés **from** qui équivaut à 0 % et **to** qui équivaut à 100 %
* **Pour chaque étape**, on définit les **règles** qui subissent une **modification**
* On donne un **nom** à cette **@keyframes** pour pouvoir **l'associer avec une animation**

# Exemple 1 (HTML)



**Exemple 1 (CSS)**

# animation-name

* Pour **appliquer une animation** à un élément, il suffit de lui définir la propriété **animation-name** en précisant le **nom de la @keyframes** à utiliser
* Puis on indique la **durée** avec la propriété **animation-duration**
* La propriété animation-name peut prendre **comme valeur** un ou plusieurs noms d'animation ou bien encore la valeur none pour par exemple annuler un héritage

# animation-duration

* La propriété **animation-duration** permet de définir la durée de l’animation
* Elle n'accepte que des **durées positives**, exprimées en secondes ou millisecondes
* Contrairement à une transition, **une animation** **débute lors du chargement de la page** et non pas uniquement lors d'un changement d'état

# animation-delay

* Il est possible de définir un **départ retardé** de l'animation avec la propriété **animation-delay** qui prend comme valeur une durée.
* Si la **valeur** est **positive**, alors l'animation débutera après n secondes
* Si la **valeur** est **négative**, l'animation débutera instantanément mais à partir de la position qu'elle aurait dû avoir après n secondes

# Exemple 2



# animation-iteration-count

* La propriété **animation-iteration-count** définit le nombre de répétitions de l'animation.
* C'est une chose impossible à réaliser avec les transitions.
* Il existe même le mot-clé **infinite** qui permet de répéter l'animation sans fin
* Si un **délai** a été défini par animation-delay, il ne sera pris en compte que lors de la première itération

# Exemple 3



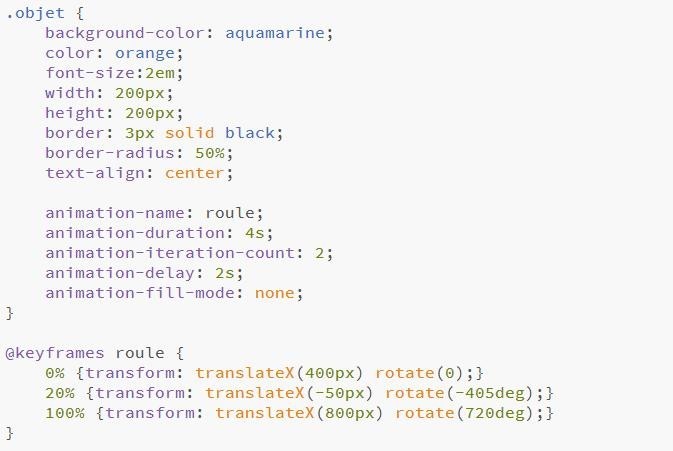
# animation-fill-mode

* **Par défaut**, une animation ne modifie pas les propriétés sur lesquelles elle s'applique avant le début (durant la animation-delay) et après la fin de celle-ci.
* La propriété **animation-fill-mode** permet de modifier ce comportement. Les valeurs possibles sont :
  + **none** : comportement par défaut
  + **backwards** : applique les propriétés définies dans l'état

0% avant le début de l'animation, y compris le délai

* + **forwards** : applique les propriétés définies dans l'état final de l'animation une fois l'animation terminée (NB : selon la direction de l’animation : 100% ou 0%)
  + **both** : combinaison de forwards et backwards

# Exemple 4



# animation-direction

* La propriété **animation-direction** permet de définir si la répétition de l'animation se déroule normalement ou à l’envers ou si elle alterne
* Les valeurs possibles pour cette propriété sont :
  + **normal** : par défaut, on joue normalement l’animation de 0 % à 100 %
  + **reverse** : à l’envers, de 100 % à 0 %
  + **alternate** : d’abord normal, ensuite reverse
  + **alternate-reverse** : d’abord reverse, ensuite normal

# Exemple 5



# animation-timing-fonction

* Comme dans les transitions, il est possible de **contrôler la vitesse et l’accélération** de l’animation.
* On utilise la propriété **animation-timing-function**
* Les **valeurs** sont identiques : ease (par défaut), linear, ease-in, ease-out, ease-in-out ou encore cubic-bezier(n,n,n,n) (courbe de Bézier spéciale)
* Mais dans les animations, on peut la définir de manière **globale** ou modifiée **individuellement** pour chaque keyframe

# animation

* Il existe aussi la propriété **animation** qui est la **notation raccourcie** reprenant toutes les autres propriétés
* **L’ordre des propriétés** n’a pas d’importance mais il faut juste veiller à placer **d’abord la durée** (animation-duration) **et ensuite le délai** (animation-delay) puisqu’elles ont la même unité
* Ne pas oublier le **nom** (animation-name) qui est nécessaire pour jouer l’animation

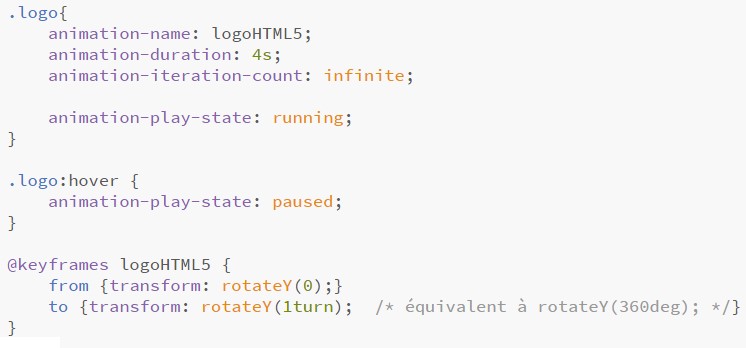
# Contrôler une animation

* Par défaut, une animation sur un élément HTML démarre dès qu’on a défini dans la propriété **animation-name** la valeur du **@keyframes**

(éventuellement après un certain délai)

* On peut déjà contrôler avec la propriété **animation-play-state** qui peut prendre deux valeurs : **paused** ou **running** (par défaut)
* Pour un meilleur contrôle, il faudra utiliser **JavaScript** et gérer les événements

# Exemple 6



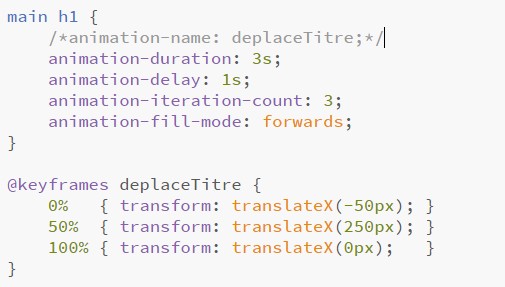
# JavaScript et animations

● On dispose de 3 événements JavaScript pour contrôler le déroulement d’une animation :

* **animationstart** : se produit quand une animation démarre
* **animationend** : se produit quand une animation se termine
* **animationiteration** : se produit quand une animation est répétée

(si animation-iteration-count > 1)

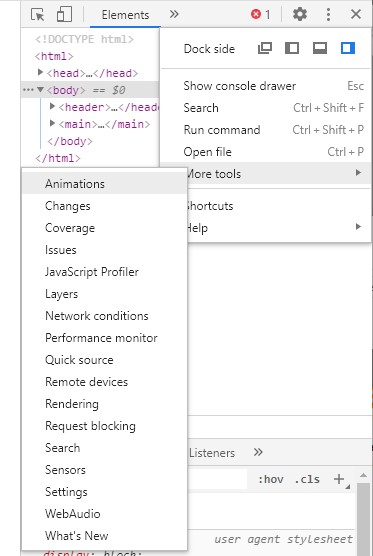
# Exemple 7 - CSS



# Exemple 7 – JavaScript & HTML



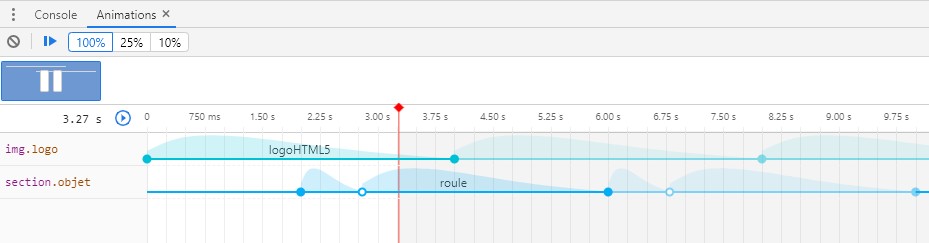
# Outils dans les navigateurs

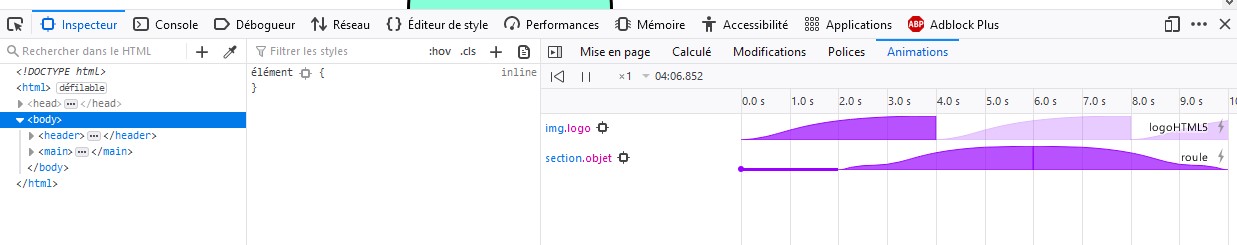
* Chrome, Firefox,… ont des **outils de test** d’animation

(Webdev Tools – F12)

* Dans **Chrome**, il faut activer le panneau Animations dans More Tools
* Dans **Firefox**, il est installé dans l’Inspecteur dans un onglet Animations

# Outils Chrome - Firefox





# Applications

* **Slideshow** : avec une animation, on fait défiler des images automatiquement
* **Sprite** : avec une animation image par image, on peut simuler le mouvement d’un objet ou

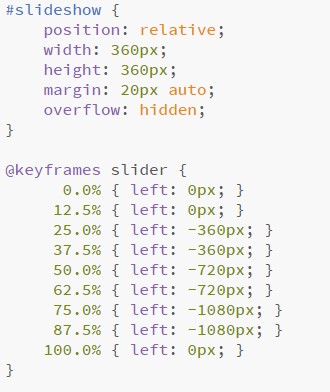
d’un personnage à la manière d’un dessin animé

NB : pour réaliser le passage d’une image à l’autre, il faut progresser **par paliers** au lieu d’avancer de manière continue, on emploie alors la fonction **steps(nombre)** comme animation-timing-function

# Slideshow – HTML



# Slideshow – CSS



# Sprite – HTML



# Sprite – CSS

