Les images pour le web

Jean Martial-Guilhem

Les formats d'images

- Les images sont essentielles dans un site web, soit à titre purement graphique et décoratif soit à titre informatif comme les icônes par exemple.
- Le choix du format des images est essentiel au bon fonctionnement d'un site web.
- Il joue sur les performances du site. Certains formats plus lourd ou volumineux afficheront des images de plus grande qualités mais auront tendance à ralentir le chargement du site. Les formats les plus légers permettront au site de se charger plus rapidement mais donneront des images de moindre qualité.
- Il faut donc trouver un équilibre entre performance et qualité en choisissant les bons formats selon que l'image doit être décorative ou informative.

Les formats d'images

- On va donc distinguer plusieurs types et formats de fichiers d'images numériques.
- Chaque format se distingue par l'extension qui définit sa nature (.jpg .png .svg)
- Parmi tous ces formats on distingue deux types d'images :
  - Les images matricielles ou tramées (.jpg .png par exemple).
  - Les images vectorielles (.svg)

Les images tramées ou matricielles

- Les fichiers d'images matricielles numériques, sont constituées de pixels (px) qui, à la manière des atomes, sont comme les éléments de bases de la définition de l'image.
- A chaque pixel est associé une couleur issue du mélange des trois couleurs primaires : rouge, vert, bleu. On parlera de couleurs RGB.
- Les fichiers d'images numériques affichent des images *statiques* avec des couleurs, des positions et des proportions pour chaque pixels, ces informations varient en fonction de la résolution de l'image.
- Plus on aura de couleurs dans les pixels mieux l'image sera définie mais plus le fichier image sera lourd.
- A l'inverse, on aura un fichier image plus léger mais qui pixélisera lorsqu'on voudra agrandir l'image.
- Une image selon sa définition affiche de 16 à 16 000 000 de couleurs par pixels.

Les images tramées ou matricielles

Les formats d'images matricielles les plus utilisées dans le web.

- Le format JPEG (.jpg), abréviation de Joint Photographie Experts Group créé en 1986. Ce sont des formats de fichiers compressés avec perte par rapport à l'original, c'est-à-dire que la taille du fichier est réduite ainsi que sa qualité mais elles peuvent tout de même comporter des millions de couleurs. C'est le format idéal pour les images photographiques.
- Le format PNG (.png) est un fichier d'image compressé sans perte. On préférera ce format au JPEG pour les images comprenant du texte et qui ont donc besoin d'une meilleure résolution.
- Le WEBP. Il s'agit d'un format relativement récent d'images plus compressées que les autres formats donc plus léger avec qui comporte peu de perte de définition. On aura donc des images de bonnes qualités moins volumineuses qui ne ralentiront donc pas le chargement des pages web. Mais tous les navigateurs ne prennent pas tous ce format en compte.

Les images vectorielles

- A la différence des images matricielles statiques définies par des pixels, les images vectorielles utilisent un système de lignes et de courbes positionnées sur un plan cartésien de coordonnées.
- L'image n'est plus définie par son élément ultime matériel qu'est le pixel mais à une simple fonction mathématique.
- De fait on peut augmenter indéfiniment la résolution de l'image originelle sans perdre en qualité.
- Dans le web le format d'image vectorielle le plus usité est le SVG (.svg) qui signifie *Scalable Vector Graphics*.
- Le SVG ne se base donc pas sur des pixels informatifs mais sur du code XML à la manière d'une équation mathématique.
- On peut alors redimensionner à l'infini un élément SVG sans perdre en qualité puisque la fonction qui définie l'image demeure inchangée.
- C'est un format idéal pour des figures ou des dessins assez simples, comme des logos ou des illustrations. Mais il ne sera pas approprié pour des images photographiques.

Insérer une image matricielle sur une page web

- Il existe deux possibilités pour insérer une image dans une page web soit par le fichier html soit par le fichier css.
- **Insérer une image dans le fichier html.**
- On utilise la balise orpheline <img/>
- La balise comporte deux attributs:
  - **src** : indique la source ou le lien de l'image.
  - **alt**: qui permet de donner une description alternative à l'image qui s'affichera si l'image ne peut être affichée au moment du chargement de la page et qui permet d'obtenir une audio description sur les navigateurs des personnes présentant un handicap de cécité. Et c'est utile pour le référencement.
  - On peut ajouter un attribut **title** c'est-à-dire un titre à l'image et qui affichera une info bulle au passage de la sourie sur l'image.
- Remarque : on veillera à toujours nommer nos fichiers images sans accents et sans espace. On peut remplacer les espaces par des underscore (_) : *mon_image.png*

Insérer une image en html

- **L'attribut src:**
- On peut appeler une image de deux manières:
  - Soit par un **chemin absolu** si l'image possède une url, c'est-à-dire si elle est en ligne sur un site. On fait un clic droit sur l'image et on choisit « copier l'adresse de l'image ».

    Ce qui donne :

    ****
  - Soit en utilisant un **chemin relatif** si l'image est dans un dossier du site. On utilise le nom et l'arborescence du fichier de l'image pour indiquer sa source.

    Si l'image est dans un sous-dossier « images » on écrira :

    ****

Faire d'une image un lien

- On insère un lien dans une page html grâce à la balise ouvrante <a>
- A l'intérieur de cette balise on renseigne l'adresse du lien grâce à l'attribut href.
- Ce qui donne :

```
<a href="#"><img src="images/mon_image.png" alt="Logo" title="Cliquez ici"/></a>
```

Faire une miniature cliquable

- On formate une image en deux version de tailles différentes, une version « minifiée » et la version originale.
- On place les deux versions dans le même dossier « images ».
- On nomme le petit format mon_image_mini.jpg et la version originale mon_image.jpg.
- Le code :
- On peut améliorer le résultat avec du JavaScript en utilisant cet exemple :
  - https://lokeshdhakar.com/projects/lightbox2/

Insérer une image de fond (header, héro…).

- On a vu comment insérer une image à partir du fichier html. Les images de fond s'insèrent à partir du fichier CSS grâce à la propriété background-image attribuée à une .

  html :

```
<div class=''image_fond''>
    <h1>Mon site</h1>
</div>
```

  CSS :

```
.image_fond {
    padding:100px;
    background-image:url('img.jpg ');
}
```

Comportement d'une image de fond à partir des propriétés CSS.

- **background-attachment : fixed**;
- **background-size : cover; (100% 100%) …** (l'image s'adapte à son conteneur)…
  - https://developer.mozilla.org/en-US/docs/Web/CSS/background-size
- **background-position : top bottom left right center;**
- **background-repeat : no-repeat;** pour annuler la répétition de l'image de fond.
- On peut combiner ces propriétés avec la super propriété **background** qui combine toutes les propriétés précédemment énumérées, dont l'URL de l'image.

  On aura : **background : url("images/mon_image_mini.jpg") cover center;**

  L'ordre des valeurs est sans importance, et on n'a pas obligé de spécifier toutes les valeurs.

Jouer sur l'opacité

- La propriété **opacity** à laquelle on attribut une valeur comprise entre 1 (valeur par défaut) et 0 (transparence totale) permet d'obtenir un effet d'opacité.

  .image { opacity : 0.4; }

- La propriété opacity affecte tous les éléments de la , texte compris.
- On peut alors utiliser la propiété linear-gradient avec une valeur rgba qui permet d'ajouter une couleur à l'image de fond.

  background: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,1)), url(images/image_code.jpg);

Bords arrondis

- Faire des bords arrondis avec la propriété **border-radius** appliquée au css de l'image.
- Afin d'avoir des bordures qui correspondent à ce que l'on souhaite :

  https://9elements.github.io/fancy-border-radius/

Rajouter des ombres

- On peut rajouter des ombres aux images avec la propriété **box-shadow** qui peut prendre quatre valeurs.
- box-shadow: 10px 5px 5px red;
  - Décalage horizontal de l'ombre
  - Décalage vertical
  - Adoucissement du dégradé par rapport à la valeur du décalage
  - La couleur de l'ombre

Organiser une galerie avec flexbox

- On va définir un conteneur dans lequel on lace nos éléments images.

```
<div class="container">
    <div class="boxune">Élément 1</div>
    <div class="boxdeux">Élément 2</div>
    <div class="boxtrois">Élément 3</div>
<div>
```

- Ici en principe tous les éléments se mettent les uns et dessous des autres. Avec la propriété flex appliquée au conteneur, les éléments se mettent les uns à coté des autres.

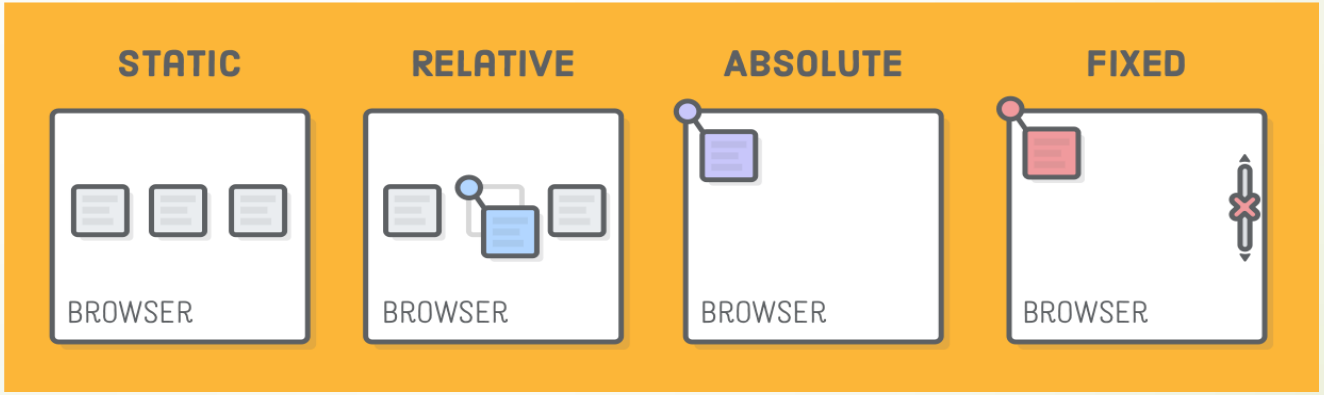
```
.container { display:flex;}
```

Organiser une galerie avec flexbox

- On peut jouer sur le comportement avec les propriétés suivantes :
  - flex-direction : row, colimn, row-reverse, column-reverse;
  - flex-wrap : nowrap, wrap, wrap-reverse;
- Ranger les éléments sur un axer horizontal
  - justify-content : flex-start, flex-end, center, space-between, space aound.
- Ranger les éléments sur l'axe vertical.
  - align-items : stretch, flex-start, flex-end, center.
- Remarque: pour aligner tous les éléments au centre, il est aussi possible de mettre un margin : auto à nos éléments.
- Répartir les blocs sur plusieurs lignes.
  - align-content : stretch, flex-start, flex-end, center, space-between, space-around.

Les positions

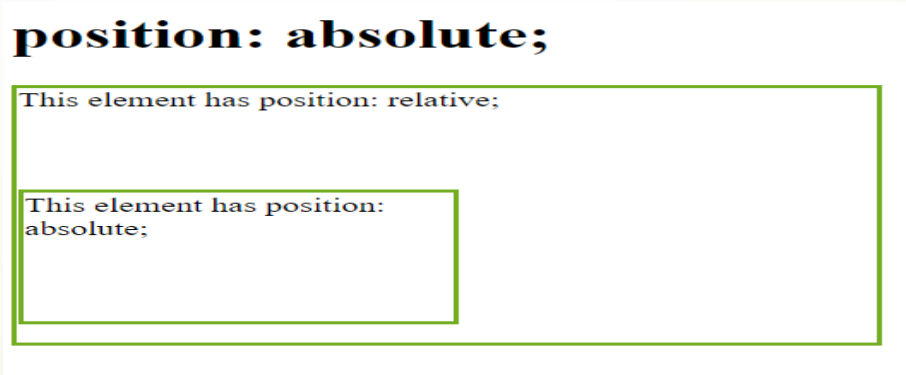
- La propriété position permet de positionner où l'on veut un élément sur la page et de superposer plusieurs éléments.
- On a plusieurs modes de positionnements possibles :
  - Positionnement relatif : permet de positionner l'élément par rapport à sa propre position normale : .element{position: relative; top: 6px; left: 10px;}
  - Positionnement absolu : permet de positionner l'élément n'importe où sur la page : .element{position: absolute; top: 6px; left: 10px;}
  - Positionnement fixe : permet à l'élément de rester fixe lorsqu'on scrolle sur la page : .element{position: fixed; top: 6px; left: 10px;}



Les positions

- Si on a un élément « absolute » à l'intérieur d'un élément « relative », l'élément « absolute » se positionne par rapport à l'élément relative auquel il appartient. Ce qui permet de placer un élément par-dessus l'autre.
- Lorsque les éléments se supperposent comme un texte sur une image par exemple. Ils peuvent se cacher les uns les autres.

  La propriété **z-index** permet alors de déterminer les priorités d'affichages. Plus la valeur du z-index est élevée plus l'élément auquel on l'attribut remonte sur la pile des éléments supperposés.
- La propriété z-index n'a d'effets que pour les éléments positionnés.



Les animations CSS

- Il existe deux moyens de faire des animations en CSS sur les images comme sur n'importe quel type de « block ».
- Les transitions et les keyframes.
- **Les transitions :**
- Pour créer des transitions on aura besoin de plusieurs infos :
  - Une propriété CSS à modifier.
  - Une valeur initiale pour la propriété CSS.
  - Une valeur finale.
  - Une durée.
  - Un événement qui déclenchera cette transition.

Les animations CSS

- On va faire un qui grossit de 115% quand on passe la souris dessus.
- On utilise la propriété **transform** et la fonction **scale()**.
- *Voir le fichier animation css.html*
- *Remarque: La propriété transition permet de combiner toutes les propriétés en une seul,* ***transition : transfom 400ms;***
  - ***transition-property : transform;***

    transition-duration : 400ms;
  - ***transition : transform 400ms;***

Les animations CSS

- **Les pseudo classes :**
  - **:hover**, qui est déclenché au **survol** de la souris ;
  - **:active**, activé au **clic** de l'utilisateur (le plus souvent pour les liens et boutons) ;
  - **:focus**, qui se déclenche lorsque son élément reçoit le **focus** (soit il est sélectionné à l'aide du clavier, soit il est activé avec la souris) ;
  - **:valid**, dont la validation du contenu s'effectue correctement par rapport au type de donnée attendu ;
  - **:invalid**, qui inversement, correspond à un élément dont la validation du contenu ne s'effectue pas correctement par rapport au type de donnée attendu ;
  - **:not(),** qui correspond à la **négation**. Elle prend un sélecteur en argument et permet de cibler les éléments qui ne sont pas représentés par cet argument ;
  - **:checked**, qui correspond aux **input** de type checkbox, option ou radio qui sont cochés ;
  - **:enabled**, un élément avec lequel on peut **interagir ;**
  - **:disabled**, qui correspond à un élément dont l'interaction a été **bloquée**.

Les animations CSS

- **Les pseudo classes :**
- Pour ce qui concerne les transformations d'images seules les pseudo classes **:hover** et **:active** vont nous intéresser.

Les animations CSS

- **Combiner les transitions :**
- On va rajouter un effet de changement de couleur.
- On change la valeur de la propriété **background-color** en rgba en passant la dernière variable de 0 à 1, de la transparence totale à l'opacité totale.
- Lorsqu'on crée une transition à propriété multiple, plutôt que de lister le nom de chaque propriété à laquelle nous souhaitons ajouter une transition, on utilise le clef **all** → transition: all 400ms;
- On peut aussi assigner des durées différentes chaque éléments :
  - transition : transform 450ms, background-color 300ms;
- On peut différer le début d'une transition avec transition-delay :
  - transition : transform 450ms, background-color 300ms;
  - transition-delay : 0, 150ms;
  - *Le changement de couleur commence e, différé de 150ms et se termine en même temps que l'effet appliqué à tansform.*
  - *On peut aussi écrire :* transition : transform 450ms, background-color 300ms 150ms;

Les animations CSS

- **Créer des accélérations et décélérations dans les animations avec ease-in et ease-out:**
  - transition : transform 1000ms ease-in;
  - transition : transform 1000ms ease-out;

Les animations CSS

- **Les différentes fonctions de la propriété transform :**
- Remarque : Les transformations ne fonctionnent pas sur les balises inline. Il faut les transformer en block ou inline-block → display:block; On peut mettre overflow:hidden pour que les éléments comme les textes n'apparaissent que lorsque l'élément est suffisamment gros.
  - La fonction **scale()** qui permet d'agrandir un élément → transform : scale(1.05);
    - On peut agrandir sur l'axe X et Y en rajoutant un paramètre Y à la fonction → transform : scale (1.15, 0.5);
    - On peut aussi utiliser les focntions scaleX et scaleY pour agrandir seulement en largeur ou seulement en longueur → transform : scaleX(2);
  - La fonction **translate()** qui permet de déplacer un élément. On lui ajoute deux valeurs en paramètres : déplacement sur l'axe des abscisses et ou déplacement sur l'axe des ordonnées (X et Y) → transform : translate(150px, -150px);
    - On peut séparer les déplacements verticaux et horizontaux grâce aux fonctions translateX() et translateY().

Les animations CSS

- **Les différentes fonctions de la propriété transform :**
  - La fonction **rotate()** qui permet de faire pivoter un élément. On exprime la valeur du paramètre en degrés ou en tours → transform : rotate(360deg); ou transfom : rotate(1turn);
  - Plusieurs transformation sur une même propriété. On liste toutes les fonctions que l'on souhaite utiliser dans la propriété transform → transfrom : scale(1.5) rotate(90deg);
  - La fonction **perspective** permet d'obtenir un effet 3D → transform : perspective(200px);

    Plus la valeur sera petite plus l'effet sera grand.
  - Modifier le point d'ancrage. Par défaut les transformations partent du centre de l'élément. La propriété transform-origin(X Y) avec pour valeur soit des px soit des % soit les valeurs left, right, top, bottom permet de modifier le point d'ancrage de la transformation.

Les images SVG

- Insérer une image SVG sur une page web
  - Insérer comme une image matricielle avec la balise <img>. Ne pas recourir à cette méthode, les balise <img/> ne sont pas faites pour accueillir des comportements scriptés.
  - Insérer pas le fichier css grâce à la propriété background-image (en rajoutant toujours une dimension).

    ```
    img { background: url(''image.svg''); height:500px;}
    ```

  - Insérer avec la balise <svg>. On insère directement le code qui définit l'image dans la page html :

    ```
    <svg width="605pt" height="450pt" viewBox="0 0 605 450"
    preserveAspectRatio="xMidYMid meet"><!- -CODE - -></svg>
    ```

Les images SVG

- Il est possible de créer des images au format SVG ou de convertir des images matricielles en format SVG avec la plus part des logiciels de graphisme. Incskape est un logiciel gratuit d'image SVG.

- Générateur de formes SVG (vagues etc.) :

  https://app.haikei.app/

  https://getwaves.io/

- Convertir une image en SVG en ligne :

  https://image.online-convert.com/fr

Les notions de viewport et viewBox

- Le **viewport** correspond à la partir visible de l'image SVG, il est défini par les valeurs attribées à **width** et **height**, c'est-à-dire la largeur et la hauteur de l'image. Mais ces valeurs sont dépendantes des valeurs des la viewBox.
- La **viewBox** détermine le cadre ou canevas dans lequel se positionne l'image à l'intérieur de la page web.
- <svg width="115px" height="190px" viewBox="0 0 115 190">
  - width et height déterminent le **viewport**, c'est-à-dire les dimensions de l'image.
  - Les deux premières valeurs (0 0) de la **viewBox** fixent l'angle supérieur gauche du cadre (X, Y).
  - Les deux dernières valeurs (115 190) de la **viewBox** fixent l'angle inférieur droit du cadre : 115 sur la droite, 190 vers le bas.

Les notions de viewport et viewBox

- Si on dessin un cercle sans préciser la position de son centre, le centre se positionnera dans l'angle supérieur gauche du cadre.
- Les éléments se trouvant à l'extérieur de la **viewBox** seront coupés.
- Si on donne la même valeur de largeur et de hauteur à la viewBox et au viewport, l'image entrera parfaitement dans les limites de son conteneur, de son cadre.
- Si on réduit de 50px la hauteur et la largeur de la viewBox, on réduit la fenêtre d'affichage et la figure dépasse de son cadre.
- Si on agrandit de 200px la hauteur et la largeur du viewport, l'image se redimensionne pour correspondre à ces dimensions. La viewBox couvre l'image entière. Le viewport contient toute l'image et définit les limites que doit remplir la poire.

Faire des formes SVG simples directement en XML

- **Le rectangle** : la balise **<rect>** à laquelle on ajoute les attribut longueur/largeur. Les éléments x et y déterminent la position de la figure au sein de l'élément svg.

```
<svg version="1.1" fill="pink" viewBox="0 0 350 350"
xmlns="http://www.w3.org/2000/svg">
        <rect x="25px" y="25px" width="50" height="40"/>
</svg>
```

- **Le cercle** : la balise **<circle**>, l'attribut r → rayon, cx etcy → les coordonnées du centre. Si cx est différent de cy, on a une ellipse.

```
<svg version="1.1" fill="pink" viewBox="0 0 350 350"
xmlns="http://www.w3.org/2000/svg">
        <circle cx="110" cy="110" r="50" />
</svg>
```

Formes SVG

- **Les lignes** : La balise <line> à laquelle on spécifie 4 paramètres obligatoires.
  - **X1** et **Y1** → Coordonnées du point de départ.
  - **X2** et **Y2** → Coordonnées du point d'arrivée.
- Modifier l'apparence:
  - Stroke → couleur du trait.
  - Stroke-width → largeur du trait.
  - Stroke-dasharray → pointillés (:5, 3, 2;)
  - Stroke-linecap → apparence des extrémités (butt→coupé, square→carré, round→arrondi).

Formes SVG

- **Les chemins** : La balise <path> et l'attribut « d » pour data. On aura pour paramètres :
  - **M** → Coordonnées du point de départ.
  - **L** → Trace une ligne à partir de la précédente coordonnée renseignée.
  - **Z** → Pour fermer le tracer.
  - **V** → Pour tracer des lignes verticales.
  - **H** → Pour tracer des lignes horizontales.

Formes SVG

- **Les balises de couleurs** :
  - **Stroke** → Couleur de contour.
  - **Fill** → Couleur de remplissage.
  - **Stroke-opacity** → Opacité du contour (de 0 à 1).
  - **Stroke-fill** → Opacité du remplissage (de 0 à 1).

Formes SVG

- **Utiliser une image que l'on a défini comme modèle** :
  - La balise <defs></defs> permet de définir un groupe en tant que modèle réutilisable.
  - A l'intérieur de la balise <defs></defs> on crée un groupe à l'aide des balises <g></g> à laquelle on attribue une classe ou un identifiant et qui permettra de regrouper plusieurs formes sous une même figure avec un seul identifiant qui sera rappeler comme modèle.
  - On rappelle la figure définie précédemment grâce à la balise orpheline <use/>, l'identifiant du modèle et une syntaxe un peu particulière :

    ```
    <use xlink:href="#identifiant" x="20" y="30" />
    ```

Appliquer des règles CSS au SVG

- On intègre le code directement dans la balise SVG. Comme en général on intègre peut d'images SVG, ça reste le plus pratique.
- On intègre le CSS en interne dans le fichier html entre les balises <style></style>
- On intègre le CSS dans un fichier externe qu'on appelle dans le fichier html.

Les animations SVG

- On peut utiliser des animations CSS que l'on a vu pour les images matricielles.
- On peut utiliser des animations spécifiques aux images vectorielles à partir de librairies JavaScript appliquées à des classes ou des identifiants comme Vivus par exemple : http://maxwellito.github.io/vivus/
- Il existe aussi des balises natives, les balises *animate*, qui permettent de créer des animations assez simplement.

Les animations SVG

- Paramètres qui déterminent l'animation à l'intérieur de la balise <animate></animate>
  - **attribut-name** → spécifie l'attribut ou la propriété CSS à animer.
  - **fill** → spécifie le comportement de l'animation (geler l'animation lorsqu'elle arrive à son terme).
  - **from** → spécifie la valeur de départ de l'animation.
  - **to** → spécifie la valeur finale.
  - **begin** → spécifie l'instant de départ.
  - **dur** → spécifie la durée de l'animation.

Animation SVG

- Il est possible de créer une succession ordonnée d'animations en faisant commencer une animation lorsqu'une autre s'est terminée.
- On spécifie alors « id.end » à **begin** où id correspond à l'identifiant attribué à la figure animate précédente.

Animations SVG

- **Animations des couleurs**
- On remplace les valeurs de **from** et **to** par des couleurs.
- On peut rajouter deux paramètres :
  - « repeatCount » qui spécifie le nombre de répétitions de l'animation : 1,2,3… ou indefinite.
  - « repeateDur » qui spécifie la durée durant laquelle se répéte l'action : 1,2,3… ou indefinite).

Animations SVG

- **Visibilité : Faire apparaitre et disparaitre une image.**
- L'élément « **set** » permet de fixer une valeur pendant une certaine durée.
- On va ici faire apparaitre une figure pendant 3 secondes.
- On modifie l'attribut **visibility** qui récupère sa valeur initiale (hidden) au bout de 3 secondes.

Animations SVG

- **Transformation : Avec animateTransform.**
- On spécifie le type de transformation : translate, rotate, scale (mise à l'échelle)
- On spécifie deux paramètres : skewX, skewY si on veut jouer sur l'inclinaison.

Animations SVG

- **Mouvements : Avec animateMotion.**
- **animateMotion** permet de faire défiler un objet sur un chemin (path).
- On définit notre chemin et on l'appelle à l'intérieur de la balise animateMotion grâce à la balise **mpath** à laquelle on donne la valeur de l'id du path.