Créez des animations CSS modernes

Table des matières

Partie 1 : Réalisez vos premières animations CSS

Chapitre 1 : Découvrez les animations Web

En résumé

Vous pouvez créer vos animations CSS sur vos propres éditeurs de code, ou bien sur des outils de code en ligne tels que CodePen ou CodeSandbox ;

L'animation a été démocratisée auprès du grand public par Disney ;

Les techniques d'animation web n'ont cessé d'évoluer depuis les débuts du web : GIF, Flash, etc. ;

Aujourd'hui, il existe de nombreuses techniques d'animation web : JavaScript, SVG, Canvas, WebGL mais surtout CSS ;

L'animation permet d'ajouter de la vie à une page web, et ainsi de vraiment impacter l'expérience d'un utilisateur sur cette page.

Chapitre 2 : Créez des animations simples avec les transitions

En résumé

Les 5 éléments nécessaires pour créer une transition sont :

- Une propriété CSS à modifier,
- Une valeur initiale pour votre propriété CSS,
- Une valeur finale pour cette même propriété,
- Une durée,
- Une pseudoclasse pour déclencher l'animation ;

On applique la valeur initiale à l'élément qu'on veut modifier, et la valeur finale dans la pseudoclasse qui déclenche la transition ;

La durée peut s'exprimer en secondes : 0.4s, ou en millisecondes : 400ms ;

Les propriétés d'une transition peuvent être déclarées individuellement : transition-duration: 400ms ;

Ou bien combinées en une seule propriété comme : transition: transform 400ms .

Chapitre 3 : Déclenchez vos transitions avec les pseudoclasses

En résumé

Les **pseudoclasses** sont essentielles pour déclencher des transitions en CSS ;

Les pseudoclasses les plus adaptées pour déclencher des transitions sont celles qui impliquent une **interaction** avec l'utilisateur ;

Les pseudoclasses les plus courantes pour déclencher une transition sont
:hover, **:active**, **:focus**, **:valid**, **:invalid**, **:not()**, **:checked**, **:enabled**, et **:disabled**

On peut combiner des pseudoclasses entre elles pour créer des sélecteurs plus précis ;

Les pseudoclasses peuvent aussi être utilisés pour changer le style d'un élément voisin.

Chapitre 4 : Appliquez les 12 principes de l'animation au web

En résumé

Les 12 principes de l'animation sont :

- Squash and Stretch, pour **compresser** et **étirer** un élément ;
- Anticipation, pour **préparer l'audience** à un mouvement à venir ;
- Staging (mise en scène), pour **guider** les yeux de l'utilisateur vers les éléments importants d'une page ;
- Straight Ahead and Pose to Pose (toute l'action d'un coup et partie par partie), qui correspond davantage à l'animation traditionnelle mais fait penser à la différence **transitions/keyframes** ;
- Follow Through and Overlapping Action (continuité du mouvement initial et chevauchement de deux mouvements consécutifs), pour faire **accélérer** et **décélérer** un élément en fonction de sa masse ;
- Slow in and slow out (ralentissement en début et en fin de mouvement), basé sur le fait que les objets ne démarrent pas et ne s'arrêtent pas instantanément ;
- Arc, pour créer des **mouvements naturels** ;
- Secondary Action (action secondaire), pour **séparer** différentes parties d'une scène dans une animation ;
- Timing, pour faire se **déplacer** les objets à une vitesse crédible ;
- Exaggeration (exagération), pour donner du **caractère** et de la **personnalité** à une animation ;
- Solid Drawing (notion de volume), pour que les animations correspondent au résultat souhaité ;
- Appeal (charisme), pour rendre les animations plus dynamiques.

Chapitre 5 : Créez des transitions CSS à propriétés multiples

En résumé

Il est possible **d'animer** autant de propriétés que l'on veut avec les transitions ;

Le mot clé all permet d'appliquer des transitions **simultanément** à toutes les propriétés ;

Ou bien on peut séparer les animations par **des** virgules, ce qui permet de leur donner des valeurs différentes. Exemple :
transition: transform 450ms, background-color 300ms;

On peut **décaler** le départ des transitions avec : **transition-delay** ;

La valeur de : **transition-delay** peut également être définie directement dans la propriété : **transition**.

Chapitre 6 : Créez des animations plus naturelles avec les fonctions de timing

En résumé

L'accélération et la décélération des transitions sont contrôlées par la propriété :
transition-timing-function ;

Si aucune fonction de timing n'est indiquée, la transition utilisera la fonction **ease**. Elle suit un profil d'accélération plus net, et une rampe de décélération plus prononcée ;

Parmi les autres mots clés pour les fonctions de **temporisation**, on peut trouver :
ease-in, **ease-out**, **ease-in-out**, et **linear** ;

Quand aucune fonction de timing par défaut ne correspond à l'animation, il est possible de créer ses propres courbes d'animation personnalisée à l'aide de la fonction cubic-bezier() ;

Il existe des outils pour ajuster les effets de timing avec la fonction cubic-bezier() .

Partie 2 : Maitrisez les translations, les rotations et l'opacité

Chapitre 1 : Optimisez les performances de votre navigateur pour vos animations CSS

En résumé

À l'écran, il n'y a pas de véritable mouvement, mais une succession d'images s'enchaînant suffisamment rapidement pour être interprétées par notre cerveau comme du mouvement ;

Cette succession d'images s'appelle les FPS (Frame Per Second, ce qui signifie images par seconde) ;
Plus le FPS est élevé, plus l'animation est fluide ;

Le taux d'images par seconde idéal est 60 FPS ;

Les quatre étapes de la création d'une page web sont :

- **Style** : le navigateur comprend la structure HTML du code qu'il reçoit et prépare le style qui sera appliqué,
- **Layout** (mise en page) : le navigateur détermine la mise en page et la taille des éléments en fonction du style qu'il a reçu,
- **Paint** : il transforme les éléments en pixels,
- **Composition** : il combine tous les éléments pour composer la page qui s'affiche ;

Pour assurer la fluidité des animations, il faut se contenter d'animer des propriétés de l'étape composition. Les plus utiles sont transform et opacity.

Chapitre 2 : Créez des animations fluides avec la propriété CSS transform

En résumé

La propriété *transform* s'avère un véritable couteau suisse en termes d'animation. En effet, elle dispose de plus de *20 fonctions différentes* pour *déplacer*, faire *pivoter*, *déformer* et *changer la taille* des éléments.

La *propriété transform* nous permet de manipuler et animer nos sites de presque toutes les manières, et comme tout se passe pendant l'étape composition, les animations sont bien fluides sur tous les supports ;

On peut déplacer des éléments avec les fonctions translate : *translate()*, *translateX()*, *translateY()* et *translate3d()* ;

On peut agrandir avec les fonctions *scale* : *scale()*, *scaleX()*, *scaleY()* et *scale3d()* ;

Et on peut les faire pivoter grâce aux fonctions rotate : rotate(), rotateX(), rotateY() et rotateZ() ;

Si on ajoute une deuxième propriété *transform*, elle annule la première. On ne peut donc définir qu'une seule *transform* dans un même sélecteur

Pour effectuer plusieurs transformations, on peut les lister dans une même propriété transform comme

```
1 transform: translateX(200%) scale(2) ;
```

Une propriété avec plusieurs fonctions exécute les fonctions dans l'ordre, de droite à gauche ;

Les fonctions de transformations en 3D comme *translate3d()*, *rotateZ()* et *scale3d()* ont également besoin de la fonction perspective pour indiquer au navigateur la distance à laquelle l'utilisateur se trouve : plus la distance est grande, moins l'animation sera marquée.

Chapitre 3 : Modifiez le point d'ancrage d'un élément grâce à transform-origin

En résumé

Transform-origin permet de repositionner le point d'ancrage, qui se trouve par défaut au centre de l'élément ;

On peut régler ce point d'origine en utilisant des unités comme px, rem, vh, etc. ;

Il est aussi possible d'utiliser des pourcentages pour X et Y ;

Ou encore, on peut utiliser des mots clés : **left** et **right** pour l'axe X, **top** et **bottom** pour l'axe Y, et **center** pour les deux ;

Il est possible de ne pas indiquer la valeur de l'axe Y ou, quand on utilise des mots clés, de mettre uniquement une valeur : le navigateur comprend de lui-même à quel axe la valeur s'applique ;

Quand on change le point d'origine en 3D, la valeur de Z doit être exprimée en unités (et non en pourcentages) !

Chapitre 4 : Analysez la performance de vos animations avec Chrome DevTools

En résumé

Chrome DevTools est l'outil de prédilection des développeurs. Il permet d'**inspecter** le code source d'une page, d'**analyser** les performances de notre page, de **brider** la performance de notre machine pour simuler un appareil plus lent. Pour cette dernière option, activez l'option "CPU throttling" ;

L'outil **Performance** permet d'analyser les performances d'une page, notamment le taux d'images par seconde d'une animation ;

On peut utiliser l'onglet Performance pour analyser nos animations, ce qui permet de repérer les problèmes dans notre code qui pourraient causer des problèmes de fluidité sur certains supports ;

Zoomer sur une image précise d'une animation permet de détailler les calculs effectués par le navigateur, que nous avions vus dans le chapitre sur le fonctionnement du navigateur.

Chapitre 5 : Animez les couleurs de manière performante avec opacity

En résumé

Animer la couleur d'une propriété déclenche des **calculs** de paint ;

La propriété **opacity** nous permet de faire des transitions entre des couleurs en évitant ces calculs

La propriété **opacity** reçoit une valeur entre 0 et 1, 0 étant complètement transparent et 1 complètement opaque

Pour éviter d'avoir à ajouter des ** ** supplémentaires, que l'on aurait dû ajouter à chaque fois dans le HTML, on peut utiliser le pseudo élément **::before** ou **::after**

Pour créer un pseudo-élément, ajoutez le nom du pseudoélément à un sélecteur, en utilisant le préfixe double deux-points : .selector::after{...}

Les pseudo-éléments **::before** et **::afte** créent un élément qui est respectivement le premier ou le dernier enfant de l'élément sélectionné

Il est possible de créer des dégradés de couleur. Il suffit d'attribuer un dégradé au background-color de l'élément d'arrière-plan. On fera ensuite disparaître l'élément superposé avec opacity : 0.

Partie 3 : Réalisez des animations dynamiques avec les @keyframes

Chapitre 1 : Créez des animations plus complexes avec la règle CSS @keyframes

Cours :

Quelles que soient les propriétés CSS que nous plaçons à l'intérieur de notre animation @keyframes, elles remplaceront toutes les propriétés CSS existantes sur le sélecteur où elles sont assignées.
Il est important de se rappeler que nous ne pouvons avoir qu'une seule propriété de transformation appliquée à un sélecteur.
Par exemple, si nous avons un sélecteur ayant une propriété **transform** utilisant la fonction **translate()**, nos keyframes ayant également **scale** l'écraseront et supprimeront toutes les translations que nous avions appliquées.

Sans keyframe de départ, le navigateur démarrera l'animation avec la valeur dans le sélecteur, tout comme il le fait à la fin d'une animation si nous ne fournissons pas de keyframe de fin.

Si un pourcentage de keyframe est défini plusieurs fois au sein d'un ensemble et que ces keyframes contiennent des valeurs contradictoires, la valeur du keyframe qui a été définie en dernier remplacera la ou les valeur(s) précédente(s) du ou des keyframe(s).

En résumé
Les animations **@keyframes** nous permettent de construire des animations plus complexes en créant plusieurs étapes pour les propriétés tout au long de l'animation ;

Les keyframes CSS sont instanciées à l'aide de la règle @keyframes suivie d'un nom pour l'ensemble des keyframes :

- **@keyframes example-frames {...} ;**

Chaque keyframe peut être établi en utilisant comme valeur le pourcentage d'animation réalisé :

- **33% {...} ;**

Si vous n'avez besoin que d'un keyframe de début et de fin, les mots clés « **from** » et « **to** » peuvent être utilisés à la place de **0 %** et **100 %** respectivement ;

Si aucun keyframe de début ou de fin n'est fourni, l'animation commence et/ou se termine avec les valeurs de propriété assignées au sélecteur. Si aucune valeur n'est explicitement assignée dans le sélecteur, c'est la valeur par défaut qui est choisie ;

Une animation définie par la règle @keyframes peut contenir différents keyframes avec des propriétés distinctes ;

Plusieurs pourcentages peuvent être assignés à un seul keyframe. Les valeurs définies dans ce keyframe seront appliquées à ces pourcentages dans l'animation ;

Les propriétés et valeurs d'un ensemble de keyframes remplaceront les valeurs de propriétés attribuées à un sélecteur au cours de l'animation.

Chapitre 2 : Utilisez les propriétés de l'animation CSS

En résumé

Les animations CSS @keyframes peuvent être déclenchées en utilisant des **pseudoclasses** telles que **:hover**, tout comme les transitions ;

Les @keyframes CSS peuvent également être déclenchés par le chargement des éléments auxquels ils sont assignés, comme un sélecteur. Par exemple, dès le chargement d'une page ;

Nous pouvons retarder le démarrage des animations avec keyframes en utilisant la propriété **animation-delay**, avec un délai exprimé en secondes ou en millisecondes, tout comme les transitions ;
Cette propriété définit le délai entre le moment où l'élément est chargé et le moment où l'animation commence.

Nous pouvons étendre ces valeurs du début à la fin de ces animations en utilisant la propriété a**nimation-fill-mode** avec :

- Le mot clé « **backwards** » prolonge les valeurs de départ d'une animation avant son lancement, couvrant la durée du délai assigné avant que l'animation elle-même ne commence,

- Le mot clé « **forwards** » prolonge les valeurs finales d'une animation jusqu'à ce que la page soit rechargée ou que le navigateur soit fermé,

- Le mot clé « **both** » prolonge l'animation dans les deux sens ;

Cette propriété indique les valeurs qui doivent être appliquées aux propriétés avant et après l'exécution de l'animation.

Nous pouvons définir une fonction de timing des @keyframes en utilisant la fonction **animation-timing-function** sur le sélecteur où l'animation a été assignée ;

Nous pouvons également définir un timing spécifique keyframe par keyframe, en assignant la propriété **animation-timing-function** aux keyframes en question.

Cette propriété configure la fonction de minutage d'une animation, autrement dit comment celle-ci accélère entre l'état initial et l'état final notamment grâce à des fonctions décrivant des courbes d'accélération.

Chapitre 3 : Manipulez et réutilisez les animations CSS

En résumé

Nous pouvons répéter un ensemble de keyframes autant de fois que nous le souhaitons en utilisant la propriété **animation-iteration-count**, avec le **nombre de cycles** comme valeur ;

Nous pouvons régler nos keyframes pour qu'ils se répètent à l'infini en utilisant la propriété **animation-iteration-count** avec le mot clé « **infinite** » ;

Cette propriété détermine le nombre de fois que l'animation est répétée. On peut utiliser le mot-clé **infinite** afin de répéter une animation infiniment.

La propriété **animation-direction** nous permet de lire un ensemble de keyframes normalement, avec le mot clé « **normal** » ;

La propriété **animation-direction** nous permet de lire un ensemble de keyframes vers **l'arrière** avec le mot clé « **reverse** » ;

La propriété **animation-direction** nous permet de lire un ensemble de keyframes avec des **allers-retours** avec le mot clé « **alternate** » ;

Et enfin, la propriété **animation-direction** nous permet de lire un ensemble de keyframes avec des **allers-retours**, mais en commençant par la fin avec le mot clé « **alternate-reverse** » ;

Cette propriété indique si l'animation doit alterner entre deux directions de progressions (faire des allers-retours) ou recommencer au début à chaque cycle de répétition.

Nous pouvons mettre en **pause** une animation avec keyframe en assignant à la propriété **animation-play-state** la valeur réglée sur « **paused** » ;

Nous pouvons reprendre la lecture d'une animation avec keyframe en assignant la propriété **animation-play-state** avec la valeur réglée sur « **running** ».

Cette propriété permet d'interrompre (« pause ») ou de reprendre l'exécution d'une animation.

Chapitre 4 : Affiner vos animations CSS avec DevTools

En résumé

Itérer, c’est le secret d'une bonne animation. Le simple fait d'ajouter quelques chiffres et d'appuyer sur Enregistrer est rarement suffisant ;

Notre expérience nous donne une bonne intuition quant aux valeurs de départ qui paraissent bien pour nos propriétés d'animation ;

Le panneau Animations de DevTools nous permet d'affiner rapidement les animations, d'improviser et d'expérimenter jusqu'à ce que nous trouvions la bonne durée ou le bon délai pour un élément ;

Le panneau Changes (Modifications) nous permet de voir les propriétés que nous avons modifiées ainsi que leurs nouvelles valeurs, de façon à mettre à jour notre code source en conséquence.

Chapitre 5 : Résumé du cours

Ensemble, nous avons appris comment :

- Appliquer les principes de l’animation au CSS ;
- Créer des transitions CSS simples ;
- Mais aussi des transitions complexes ;
- Créer des animations ayant de nombreuses étapes avec les keyframes ;
- Créer une animation qui boucle à l’infini avec les propriétés d’itération et de direction.

Personnellement, quand j'ai besoin de créer une nouvelle animation, je recherche toujours ce qui existe déjà. Voilà les sites que je consulte en priorité :

- **Awwwards** : ce site recense les plus beaux sites Internet. La catégorie "animation" a de quoi faire rêver n'importe quel animateur 😍 ;
- **Dribbble** : vous pouvez chercher le mot clé qui correspond le plus à ce que vous voulez créer, et rassembler vos résultats dans un panier (bucket en anglais). Très pratique !
- **Codepen** : ce site est parfait lorsque vous cherchez une animation sur un élément bien spécifique, comme un burger menu, par exemple ;
- **UI Movement** : votre dose d'inspiration quotidienne en UI (User Interface), qui recense de nombreuses animations ;

Et encore bien plus en vous perdant dans le web. Alors bonnes découvertes à vous 👩‍🚀 !