Semaine 06 - Effet de défilement et animation CSS (début)

Effet de défilement

Les effets de défilement sont tout type de transition ou d'animation qui se déroule pendant que le visiteur fait défiler un site Web vers le haut ou vers le bas. Habituellement, l'animation de défilement est déclenchée lorsque l'élément apparaît à une certaine hauteur et elle peut être appliquée à pratiquement n'importe quel élément tel que du texte, des images et des vidéos.

Nos yeux sont naturellement attirés par le mouvement, donc cette fonctionnalité permet généralement d'attirer et de garder le visiteur engagé.

Création d'une fonction JavaScript pour cibler et contrôler les éléments à animer

Nous aurons besoin d'une fonction JavaScript pour cibler les éléments à animer et pour modifier la valeur de certaines propriétés lorsqu'ils entrent dans la fenêtre d'affichage lors du défilement de la page. Dans notre exemple, les éléments qui seront modifiés sont les sections du document ayant la classe CSS « *.apparition* » dont le code est le suivant:

```
.apparition {
    /*
        Au démarrage, les éléments ne sont pas visibles
        On va les faire apparaître avec le défilement de la page
        quand ils vont atteindre une certaine hauteur
    */
    transform: translateX(110%);
    opacity: 0;
    transition: opacity 1s, transform 1s;
}
```

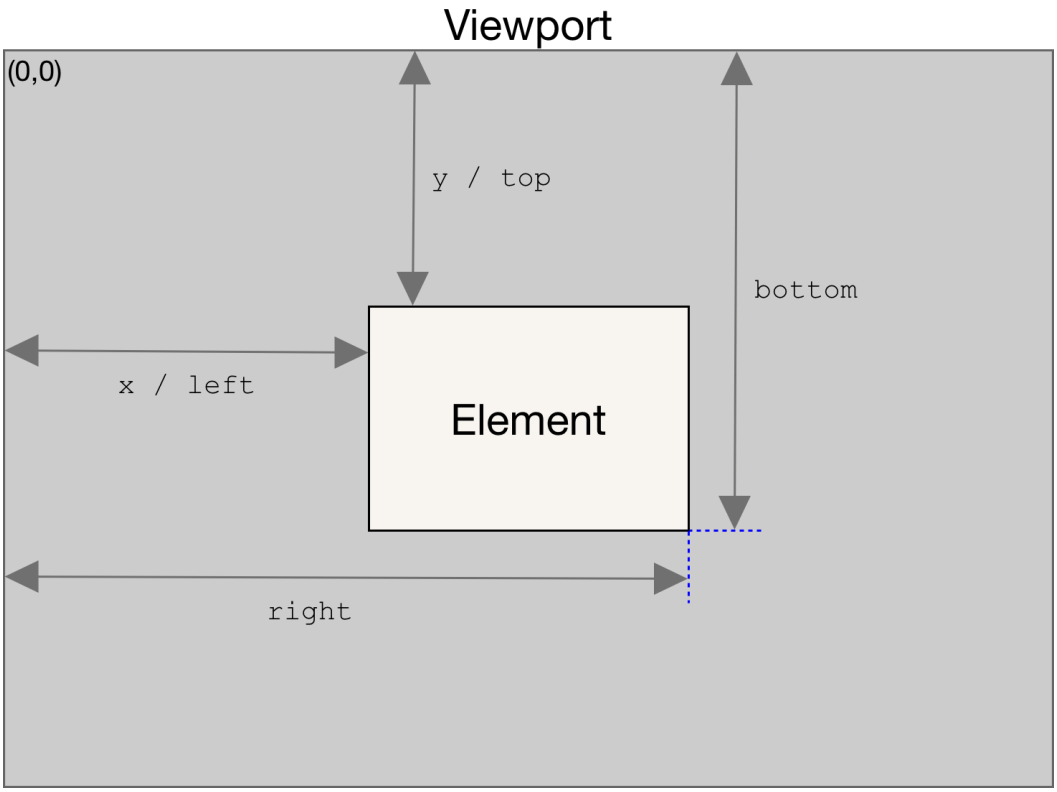
Le code de la fonction « ***faireApparaître*** » commence par cibler tous les éléments à l'aide de la méthode **document.querySelectorAll()** que nous avons vue dernièrement:

```
function faireApparaitre() {
    let lesSectionsAfaireApparaitre = document.querySelectorAll(".apparition");
}
```

L'animation de défilement doit être déclenchée lorsque les éléments apparaissent à une certaine hauteur, nous devons donc déterminer la **position de l'élément sur la page**, c'est-à-dire la **distance** entre **l'élément** et le **haut de la fenêtre**. Pour cela nous allons utiliser la méthode **elementHTML.getBoundingClientRect()** et la propriété **window.innerHeight**.

La propriété **window.innerHeight** nous donne la hauteur de la fenêtre d'affichage. La méthode **elementHTML.getBoundingClientRect()** retourne un objet fournissant des informations sur la taille d'un élément

et sa position relative par rapport à la zone d'affichage. Les propriétés left, top, right, bottom, x, y, width, et height, de l'objet récupéré décrivent la position et la taille du rectangle en pixels:



Source: https://developer.mozilla.org/fr/docs/Web/API/Element/getBoundingClientRect

Avec cela, nous pouvons définir les conditions d'apparition de chaque élément en utilisant une boucle for…of:

```
for (let uneSection of lesSectionsAfaireApparaitre) {
    let hauteurVisible = window.innerHeight * 0.75;
    let hauteurDeLaSection = uneSection.getBoundingClientRect().top;
}
```

La variable **hauteurDeLaSection** est la distance entre la position **y** ou **top** de la section à faire apparaître et le haut de la fenêtre, et la variable **hauteurVisible** est la hauteur choisie à laquelle l'élément HTML (la section) doit être révélée à l'utilisateur. Dans cet exemple on a défini cette hauteur au ¾ de la hauteur du viewport. Il reste uniquement à définir la structure conditionnelle qui permettre de modifier les propriétés CSS selon la hauteur choisie:

```
if (hauteurDeLaSection < hauteurVisible) {
    uneSection.style.opacity = "1";
    uneSection.style.transform = "translateX(0%)";
} else {
    uneSection.style.opacity = "0";
    uneSection.style.transform = "translateX(100%)";
}
```

La fonction complète ressemblera à ceci:

```
function faireApparaitre() {
    let lesSectionsAfaireApparaitre = document.querySelectorAll(".apparition");

    for (let uneSection of lesSectionsAfaireApparaitre) {
        let hauteurVisible = window.innerHeight * 0.75;
        let hauteurDeLaSection = uneSection.getBoundingClientRect().top;

        if (hauteurDeLaSection < hauteurVisible) {
            uneSection.style.opacity = "1";
            uneSection.style.transform = "translateX(0%)";
        } else {
            uneSection.style.opacity = "0";
            uneSection.style.transform = "translateX(100%)";
        }
    }
}
```

Pour terminer, nous devons déclarer un gestionnaire d'événement pour exécuter la fonction **faireApparaitre** chaque fois que le visiteur fait défiler la page dans n'importe quelle direction, c.-à-d.chaque fois que l'événement « **scroll** » sera déclenché. La position de défilement sera également vérifiée au chargement de la page. Le code est le suivant:

```
/*Gestionnaire d'événement scroll sur la fenêtre*/
window.addEventListener("scroll", faireApparaitre);

// Pour vérifier la position de défilement au chargement de la page
faireApparaitre();
```

Animation CSS (début)

En CSS, deux propriétés : **animation** et **transition** permettent de créer facilement des **effets animés**, sans l'usage de scripts JavaScript.

Nous avons déjà vu que les **transitions CSS** permettent de modifier progressivement l'apparence d'un élément HTML **à chaque fois qu'un changement d'état se produit**, comme par exemple lorsque le pointeur de la souris survole cet élément. Les **animations CSS** permettent quant à elles de modifier l'apparence d'un élément HTML selon **plusieurs images clés** (ou plusieurs états CSS).

C'est quoi une animation en CSS ?

Les animations en CSS permettent de modifier les valeurs de propriétés CSS plusieurs fois dans le temps, tel un scénario Flash/Animate. C'est en fait plusieurs transitions qui s'enchaînent! On peut donc comparer cela avec une plage d'interpolation d'Adobe/AfterEffects ou Adobe/Animate détenant plusieurs images-clés, à l'exception près qu'avec CSS les modifications de propriétés doivent être déclarées littéralement, tandis qu'avec Flash elles sont effectuées manuellement…

Comment déclarer une animation CSS

Règle @keyframes

Pour déclarer une animation CSS, il **faut d'abord** utiliser la règle CSS **@keyframes**, lui donner un **nom**, puis gérer les étapes ou images-clés (« *keyframes* ») en pourcentage (« % ») à l'intérieur de cette déclaration.

```
@keyframes nom-animation {
    /*étapes et valeurs des propriétés CSS à animer*/
}
```

Les « *keyframes* » représentent le déroulement de l'animation. Les valeurs en pourcentage représentent un moment temporel précis au cours de la durée de l'animation où les valeurs des propriétés CSS « animables » sont modifiées. Ainsi, 10% représentera la 2ème seconde d'une animation durant 20 secondes.

0% indique le **premier instant** de la séquence animée, tandis que **100%** indique l'**instant final de l'animation**. **Pour un meilleur support de l'ensemble des navigateurs, ces deux instants doivent préférablement être spécifiés pour que les navigateurs sachent où l'animation doit commencer et se terminer**[1] ; puisqu'ils sont **importants**, ces deux instants ont aussi des **alias** particuliers : **from** et **to** correspondent respectivement à **0%** et **100%**.

L'exemple qui suit déclare une séquence d'animation dénommée « *changerCouleur* », où la couleur d'un élément HTML varierait du rouge au jaune :

[1] *Si les images-clés aux instants 0% et 100% ne sont pas spécifiées, le navigateur doit construire ces instants à partir des valeurs initiales des propriétés CSS de l'élément HTML animé.*

```css
@keyframes changer-couleur {
    0% {
        background-color: red;
    }

    100% {
        background-color: yellow;
    }
}
```

L'écriture de la règle précédente, est équivalente à celle-ci :

```css
@keyframes changer-couleur {
    from {
        background-color: red;
    }

    to {
        background-color: yellow;
    }
}
```

ATTENTION!!! Les valeurs des propriétés CSS déclarées aux étapes 0% et 100% ne correspondent pas nécessairement aux valeurs initiales et finales, avant et après l'animation de l'élément HTML. Pour gérer les valeurs appliquées aux animations en dehors du temps d'exécution de l'animation, il faut gérer la sous-propriété « ***animation-fill-mode*** » que nous examinons plus loin dans cette note de cours.

Pour réaliser une animation plus élaborée, il suffit de **déclarer plusieurs étapes** ou **images-clés** dans la règle **@keyframes** et d'y modifier plusieurs propriétés CSS. L'exemple suivant va changer à la fois la couleur et l'échelle horizontale d'un élément HTML, du début de l'animation (from ou 0%), à 25%, 50% et 75% du temps d'exécution de l'animation, et à nouveau à la fin de l'animation (to ou 100%) :

```css
@keyframes changer-couleur-et-taille {
    from {
        background-color: red;
        transform:scaleX(1);
    }

    25%{
        background-color: green;
        transform:scaleX(2);
    }

    50%{
        background-color: blue;
        transform:scaleX(4);
    }
```

```
    75%{
        background-color: green;
        transform:scaleX(2);
    }

    to {
        background-color: yellow;
        transform:scaleX(1);
    }
}
```

Quelques remarques sur la règle «@keyframes »

- Une transition est effectuée entre chaque « keyframe » rendant l'animation plus fluide;
- Bien qu'il soit préférable de les déclarer dans leur ordre d'exécution pour avoir une bonne compréhension des effets d'animations souhaités, les « keyframes » **n'ont pas besoin d'être en ordre**, le « *keyframe* » 100% peut très bien être déclaré avant le « *keyframe* » 0%;
- Si plusieurs « keyframes » ont la même valeur, il est possible de les **insérer sur la même ligne en les séparant d'une virgule**. Dans le code CSS ci-haut, les « keyframes » 25% et 75% ont la même valeur, ainsi le code suivant produirait le même effet :

```
@keyframes changer-couleur-et-taille {
    from {
        background-color: red;
        transform:scaleX(1);
    }

    25%, 75%{
        background-color: green;
        transform:scaleX(2);
    }

    50%{
        background-color: blue;
        transform:scaleX(4);
    }


    to {
        background-color: yellow;
        transform:scaleX(1);
    }
}
```

Le code suivant, qui simule le rebond d'un élément, est aussi tout à fait valide :

```
@keyframes rebond {
    40%, 70%, 90%, 100% {
        bottom: 0;
    }

    0% {
        bottom: 200px;
    }

    55% {
        bottom: 50px;
    }

    80% {
        bottom: 25px;
    }

    95% {
        bottom: 10px;
    }
}
```

Comment lancer une animation CSS?

Une fois les « ***keyframes*** » déclarés et référencés par le **nom** qui leur a été donné, il ne reste plus qu'à appeler cette animation à partir d'un **sélecteur CSS**. Il existe plusieurs sous-propriétés d'animation, dont **deux** doivent être **obligatoirement déclarées** pour l'exécution de l'animation:

Propriétés obligatoires :

- **animation-name:** indique le **nom** de l'animation à utiliser qui correspond au nom de la règle @keyframes décrivant les keyframes de l'animation;
- **animation-duration:** indique la durée en milliseconde (ms) ou en seconde (s) de l'animation.

Ainsi, pour affecter l'animation « *changerCouleurEtTaille* » décrite précédemment avec une **durée de 3 secondes** sur un élément HTML de type div, il suffirait d'écrire le code suivant :

```
div {
    ...

    /*Effet d'animation*/
    animation-name: changer-couleur-et-taille;
    animation-duration: 3s;
}
```

Propriétés facultatives :

- **animation-delay** : Cette valeur, exprimée en milliseconde (ms) ou en seconde (s), définira un **délai d'attente entre l'instant auquel l'élément est chargé et le début de l'animation**, ou, dans le cas d'une valeur négative, l'avance que doit prendre l'animation avant de débuter. Par exemple, si on souhaite que l'animation « *changerCouleurEtTaille* » s'exécute uniquement 2 secondes après le chargement de la page, on ajouterait le code suivant au sélecteur CSS :

```
div {
    ...

    /*Effet d'animation*/
    animation-name: changer-couleur-et-taille;
    animation-duration: 3s;
    animation-delay: 2s;
}
```

- **animation-iteration-count :** Précise le **nombre de fois** que l'animation doit être effectuée. La valeur devra être un **nombre entier** ou le mot clé « **infinite** » pour que l'animation se répète à l'infinie. La valeur par défaut est de 1, c.-à-d. que l'animation joue par défaut une seule fois. Par exemple, dans le code qui suit, l'animation « *changerCouleurEtTaille* » se répétera 3 fois :

```
div {
    ...

    /*Effet d'animation*/
    animation-name: changer-couleur-et-taille;
    animation-duration: 3s;
    animation-delay: 2s;
    animation-iteration-count: 3;
}
```

Tandis qu'avec le code suivant, l'animation jouera en boucle (à l'infini) :

```
div {
    ...

    /*Effet d'animation*/
    animation-name: changer-couleur-et-taille;
    animation-duration: 3s;
    animation-delay: 2s;
    animation-iteration-count: infinite;
}
```

Il est à noter que dans les deux exemples précédents, le délai d'exécution ne sera appliqué qu'une seule fois.

- **animation-direction**: Cette propriété définit si l'animation doit se jouer à l'endroit (du « keyframe » 0% au « keyframe » 100%) ou à l'envers (du « keyframe » 100% au « keyframe » 0%). Les différentes valeurs sont les suivantes :
    - **normal** : L'animation doit se jouer vers l'avant à chaque cycle (du « keyframe » 0% au « keyframe » 100%), c'est-à-dire, à chaque itération. En d'autres termes, à chaque fois que l'animation fait un cycle, elle est réinitialisée à son début et recommence de nouveau. **Ceci est la valeur par défaut**;
    - **reverse** : L'animation est jouée à l'envers à chaque cycle (du « keyframe » 100% au « keyframe 0%);
    - **alternate** : L'animation doit **changer de sens** à chaque cycle, du « keyframe » 0% au « keyframe » 100%, puis du « keyframe » 100% au « keyframe » 0%, et *ainsi de suite*;
    - **alternate-reverse** : Comme pour la valeur **alternate**, l'animation change de sens à chaque cycle. L'animation sera jouée à l'envers lors du premier cycle, puis à l'endroit, ensuite à l'envers, et ainsi de suite.
- **animation-fill-mode** : Cette propriété configure les valeurs des styles CSS qui doivent être appliquées à l'élément HTML animé **avant** et **après** que l'animation soit exécutée Les différentes valeurs possibles sont les suivantes:
    - **none** : Le navigateur n'appliquera pas de changements de style à l'élément HTML cible avant ou après l'exécution de l'animation; c'est-à-dire qu'il conservera les valeurs définies initialement. *Ceci est le comportement par défaut*;
    - **forwards** : Cette propriété indique au navigateur de laisser le style de l'élément HTML animé dans son état final défini par la dernière image-clé rencontrée (« *keyframe* ») lors de l'exécution de l'animation. *Toutefois, la dernière image-clé rencontrée (« keyframe ») ne correspond pas nécessairement au « keyframe » 100% (ou to), mais dépend aussi de la valeur des sous-propriétés « animation-direction » et « animation-iteration-count »*;
    - **backwards**: Cette *propriété* indique au navigateur de placer l'élément dans son état défini à la première image-clé « keyframe » de l'animation au chargement de la page, même si un délai est indiqué. *La première image-clé (« keyframe ») n'est pas nécessairement le « keyframe » 0%, mais dépend de la valeur de la sous-propriété « animation-direction »*;
    - **both**: Le navigateur appliquera les deux valeurs précédentes. Ainsi, l'animation suivra les règles pour les valeurs « *forwards* » et « *backwards* » ce qui étendra les propriétés de l'animation aux deux directions de lecture.
- **animation-timing-function** : Cette dernière propriété configure la **temporisation** de l'animation; c'est-à-dire, comment l'animation se déroule entre chaque « *keyframe* », en définissant des **courbes de vitesse ou d'accélération**. De base, les navigateurs fournissent quelques fonctions décrivant des courbes d'accélération par défaut (linear, ease, ease-in, ease-out, ease-in-out). **Il est à noter que les fonctions d'accélération seront abordées en détail à la semaine 7.**

La notation raccourcie « animation »

Tout comme pour d'autres propriétés CSS, il existe une **notation raccourcie** pour déclarer les animations CSS. Cette notation permet de décrire de manière concise la majorité des sous-propriétés en jeu à l'aide d'une seule propriété.

La syntaxe est la suivante :

sélecteurCSS {

animation: animation-name] [animation-duration] [animation-timing-function] [animation-delay] [animation-iteration-count] [animation-direction] [animation-fill-mode] [animation-play-state];

}

L'exemple ci-dessous utilise cinq des sous-propriétés d'une animation CSS:

```
selecteurCSS {
    animation-name: monAnimation;
    animation-duration: 5s;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

Le même effet d'animation peut être déclaré en utilisant la propriété d'animation raccourcie, comme suit:

```
selecteurCSS {
    animation: monAnimation 5s 2s infinite alternate;
}
```

La notation raccourcie, permet donc de réduire l'ampleur du code CSS, d'autant plus si on désire appliquer plusieurs animations sur un même élément HTML:

```
selecteurCSS {
    animation: animation-1 5s 2s infinite alternate, animation-2 3s 2 reverse both;

}
```

La propriété « animation-play-state »

La propriété animation-play-state définit si une animation est en cours d'exécution ou si elle est en pause. Les deux valeurs possibles sont les suivantes:È

- **running**. L'animation est en cours;
- **paused**. L'animation est en pause.

Il est à noter que lorsqu'on reprend une animation en pause, celle-ci reprendra où elle avait été interrompue (elle ne recommencera pas nécessairement depuis le début de la séquence).

Les propriétés CSS animables et optimisation

Comme pour les transitions, de nombreuses propriétés CSS sont « animables ». Citons par exemple, les propriétés suivantes :

- background-color, background-position, border, bottom, color, font-size, font-weight, height, left, letter-spacing, line-height, margin-bottom, margin-left, margin-right, margin-top, max-height, max-width,

min-height, min-width, opacity, outline-color, outline-width, padding-bottom, padding-left, padding-right, padding-top, right, text-indent, text-shadow, top, vertical-align, visibility, width, word-spacing, z-index.

Précisons toutefois que pour des questions d'**optimisation**, il est **fortement recommandé** de ne **PAS animer** des propriétés qui affectent la **disposition** des éléments (*layout*) de la page Web, comme par exemple, les propriétés *height*, *width*, *top*, *left*, *margin*, etc.

En effet, **certaines propriétés CSS sont beaucoup plus chères à animer que d'autres.** Par exemple, lorsque la hauteur d'un élément augmente ou diminue cela provoque une réaction en chaîne; tous ses frères et sœurs devront également monter ou descendre, pour remplir l'espace! Tous ces déplacements exigent beaucoup de calculs de la part du navigateur, ce qui risque de produire une animation saccadée.

Les deux propriétés les moins coûteuses à animer sont : opacity et transform (fonctions: translate, scale, rotate, etc.).

Quelques références utiles sur les animation CSS

Site officiel du W3C:

- Spécifications officielles du W3C, CSS Animation Level 1 (Octobre 2018). Repéré à: http://www.w3.org/TR/css3-animations/

Bonnes références en français

- OpenClassrooms. (Septembre 2022). Créer des animations CSS modernes. Repéré à : https://openclassrooms.com/fr/courses/5919246-creez-des-animations-css-modernes
- Mozilla Corporation. (Septembre 2022). Utiliser les animations CSS | MDN. Repéré à https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Animations/Using_CSS_animations
- Mozilla Corporation. (Septembre 2022). Liste des propriétés CSS animables | MDN. Repéré à https://developer.mozilla.org/fr/docs/Web/CSS/CSS_animated_properties

Bonnes références en anglais

- CSS Tricks (Septembre 2022). Animation. Repéré à : https://css-tricks.com/almanac/properties/a/animation/
- Google Developers. (Octobre 2020). How to create high-performance CSS animations. Repéré à : https://web.dev/animations-guide/
- Joshua Comeau. (Septembre 2021).An Interactive Guide to Keyframe Animations. repéré à: https://www.joshwcomeau.com/animation/keyframe-animations
- W3Schools. (s.d.). CSS Animations. Repéré à : http://www.w3schools.com/css/css3_animations.asp