CSS- 09- e- Boucles

Plutôt que d'écrire à la main plusieurs classes CSS similaires avec une légère variation, SCSS permet de créer des boucles générant pour nous ces classes.

@for

Les boucles `@for` fonctionnent de façon très similaire aux boucles for en JavaScript (que nous verrons dans un prochain chapitre). Il faut premièrement définir une variable, lui donner une valeur de départ et une valeur à atteindre.

Par exemple, afin de configurer la taille de plusieurs titres *(headings)* à l'aide d'une boucle `@for`, il est possible de faire :

```
@for $index from 1 to 5 {
  .h#{$index} {
    font-size: 40px - ($index * 5);
  }
}
```

Ce qui produit le code suivant :

```
h1 { font-size: 35px; }
h2 { font-size: 30px; }
h3 { font-size: 25px; }
h4 { font-size: 20px; }
```

Remarquez comment pour concaténer une variable SCSS avec du texte on l'enveloppe avec `#{}`. Un peu comme les `${}` avec les littéraux de gabarit.
Remarquez que le dernier chiffre n'est jamais atteint. La boucle indique `1 to 5`. Cependant, le dernier heading est `h4` et non `h5`!

through

Lorsque through est utilisé à la place de to, le chiffre à atteindre dans la boucle est inclus.

Par exemple, le même code que précédemment, mais avec through:

```
@for $index from 1 through 5 {
  .h#{$index} {
    font-size: 40px - ($index * 5);
  }
}
```

Produit le code suivant:

```
h1 { font-size: 35px; }
h2 { font-size: 30px; }
h3 { font-size: 25px; }
h4 { font-size: 20px; }
h5 { font-size: 15px; }
```

Où le nombre 5 est inclus.

@for

@each

Les boucles @each ressemblent aux boucles @for à la différence qu'**elles servent à itérer sur une liste d'items**. À tour de rôle, une variable prend la valeur de chaque item dans la liste et devient accessible.

Par exemple, une de boucle configurant la couleur de plusieurs messages peut être écrite ainsi avec une boucle @each:

```
$colorsArr: red, yellow, blue, gray;

@each $color in $colorsArr {
  .msg-#{$color} {
    background-color: $color;
  }
}
```

Ce qui produit le code suivant:

```
.msg-red { background-color: red; }
.msg-yellow { background-color: yellow; }
.msg-blue { background-color: blue; }
.msg-gray { background-color: gray; }
```

Map

Dans certains cas, identifié une valeur à l'aide d'une clé dans une boucle peut s'avérer très pratique. Heureusement, la boucle @each peut aussi itérer sur un tableau de clés et de valeurs.

Par exemple, pour créer une rapidement des classes ayant des noms textuels et des valeurs numériques il est possible de faire:

```
$sizesArr: (small: 12px, medium: 16px, big: 30px);
@each $key, $value in $sizesArr {
  .text-#{$key} {
    font-size: $value;
  }
}
```

Ce qui produit le code suivant:

```
.text-small { font-size: 12px; }
.text-medium { font-size: 16px; }
.text-big { font-size: 30px; }
```

@each

Exercice

Pour cet exercice vous devez recréer un menu permettant de choisir parmi certains personnages du jeu Overwatch. Afin d'éviter de vous répéter, vous devrez tirer profit de la boucle @each de Scss.

Aperçu du résultat : fichier each-menu-overwatch-resultat.mp4

Médias

Arrière-plan : Copier dans le presse-papier
https://ex.smnarnold.com/scss/overwatch/bg.jpg

- Ajoutez l'image d'arrière-plan en fond de page. Celle-ci doit couvrir entièrement la page, être centrée et ne doit afficher qu'une seule fois.
- Faites-en sorte que l'élément avec la classe .overwatch soit centré verticalement ↕ dans la page.
- Les personnages *(.characters)* doivent être centrés horizontalement ↔, doivent pouvoir s'afficher sur plusieurs lignes, mais tenter de s'afficher autant que possible sur une même ligne si l'espace le permet.
- Utilisez une boucle @each afin de générer des noms de classes ciblant chacun des personnages suivants: *echo, genji, hanzo, junkrat, lucio, mccree, mercy, moira, orisa, pharah, reaper, reinhardt*
- Pour chacun de ces personnages, ajouter une image d'arrière-plan dont le chemin de fichier correspond à: https://ex.smnarnold.com/scs/overwatch/[nom du personnage].jpg et assurez que cette image soit entièrement visible *(les noms des personnages ne devraient pas être tronqués)*.
- Profitez du nesting de Scss afin de créer un effet de survole sur les personnages. Ceux-ci doivent grandir de 10% en l'espace d'un tiers de seconde, lorsque survolé par la souris.

Notes de cours

- Boucles (CSS 9)
- Flexbox
- Transformation
- Transition