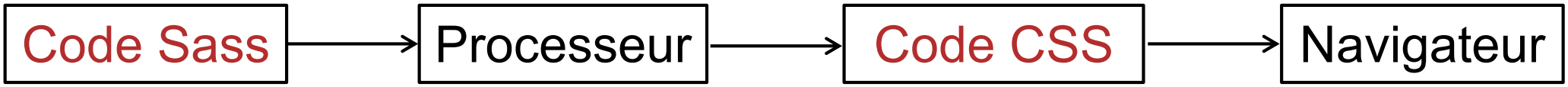
Langage SASS

Syntactically Awesome StyleSheets

Jean-Christophe DUBOIS
jean-christophe.dubois@univ-rennes.fr

Principes de base

- Préprocesseur : Générateur dynamique de CSS



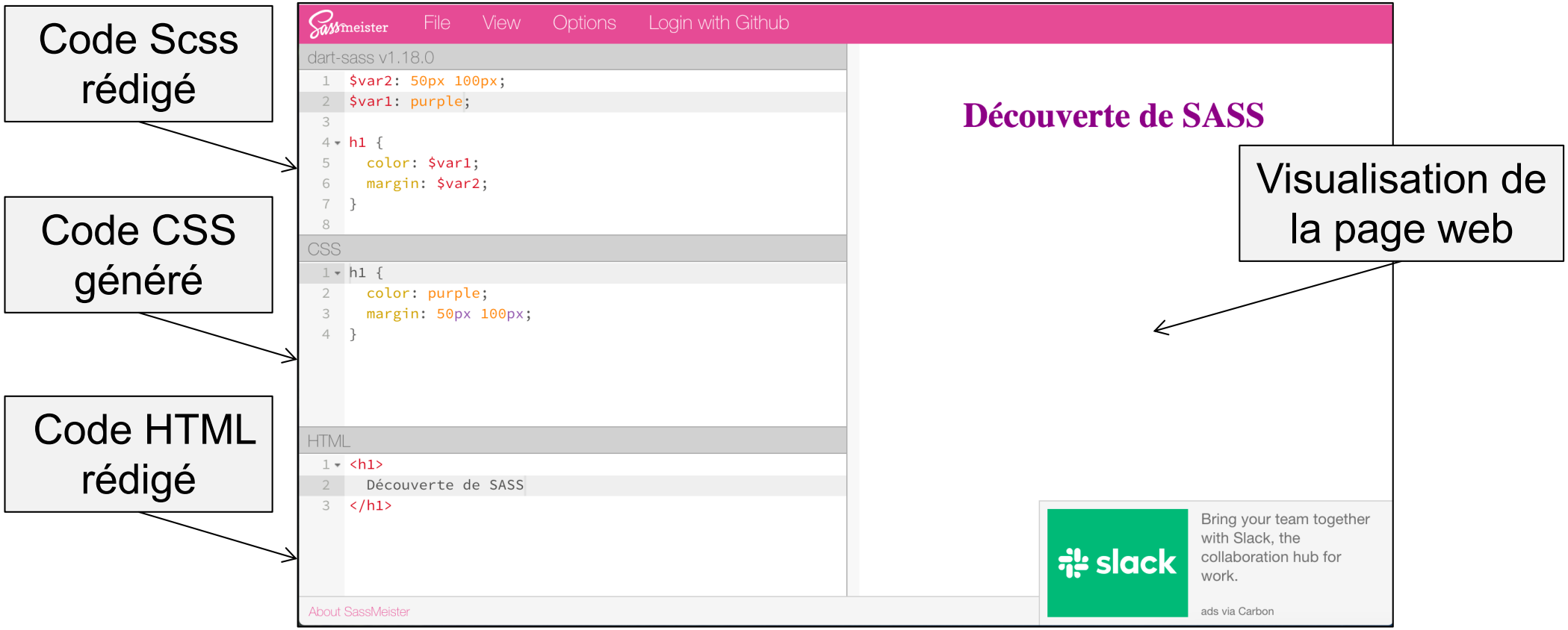
- Avantages
  - Enrichissement du CSS
    - variable, liste, map : nombres, chaînes, couleurs…
    - mixin, extension, condition, boucle…
  - Structuration du code
    - gain de temps, réutilisation du code
    - amélioration de la lisibilité
- SASS mais aussi Less ou Stylus

Principes de base

- Extension .scss, basé d'après le langage Ruby
- Deux syntaxes très proches
  - SASS: sans ponctuation ; ou {} et des indentations
  - SCSS : avec ponctuation mais plus usité
- Pré-processeurs CSS
  - Scout-App (gratuit) : Mac, Windows et Linux
  - CodeKit, Hammer (payants) : Mac
  - Ghostlab (payant) : Mac, Windows
  - Sassmeister (gratuit) : En ligne
- Frameworks
  - Compass, Bourbon, Susy

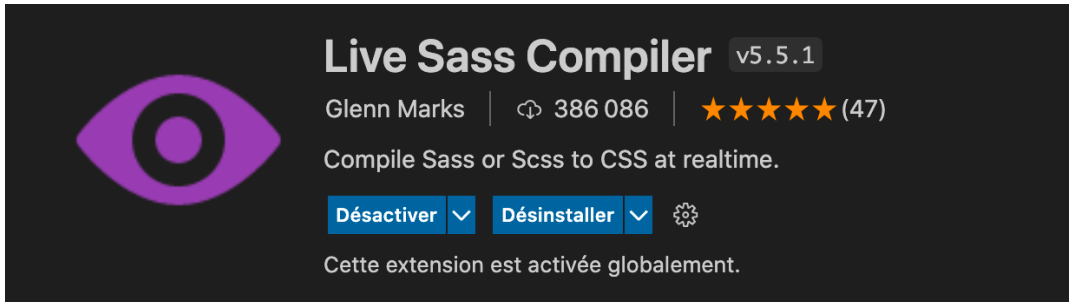
Sassmeister

- Préprocesseur en ligne : www.sassmeister.com/



- Site officiel de Sass : https://sass-lang.com/install

Extension VSC



- Extension Live Sass Compiler par Glenn Marks
  - Installation de l'extension
  - Création d'un fichier style.scss
  - Saisie du code .SCSS
  - F1 pour la 1ère génération puis génération automatique du fichier .CSS

Génération automatique du code CSS lors de la sauvegarde du fichier .SCSS

  - Le fichier .css.map fait le lien entre le css et le scss dans le navigateur

Commentaires

- Commentaire sur une ligne non inclus en CSS

```
// Commentaire présent dans le Sass mais non intégré dans le CSS
```

- Commentaire sur plusieurs lignes inclus en CSS : /* ..... */

```
/* Commentaire sur plusieurs lignes présent dans le Sass et intégré dans le CSS */
```

- Commentaire sur plusieurs lignes inclus en CSS y compris le CSS minimisé : /*! ..... */

```
/*! Commentaire sur plusieurs lignes présent dans le Sass, intégré dans le CSS et dans le CSS minimisé */
```

DRY - Don’t Repeat Yourself

- Imbrication : factoriser le code, éviter les répétitions
  - 2 ou 3 niveaux maximum
  - pas d’utilisation des id, trop spécifiques

- Utilisation des combinateurs :
  - “ “(espace) combinateur descendant
  - > combinateur enfant
  - + combinateur de frère adjacent
  - ~ combinateur de frère
  - & sélecteur de parent

Imbrication : combinateur descendant

- Définition Sass

```
table {
    border: 1px solid blue;
    caption {
        color: green;
    }
    tr {
        th {
            background: #333;
            color: white;
        }
        td {
            color: blue;
        }
    }
}
```

- Code CSS

```
table {
    border: 1px solid blue;
}
table caption {
    color: green;
}
table tr th {
    background: #333;
    color: white;
}
table tr td {
    color: blue;
}
```

Imbrication : enfant > adjacent +, ~

- Définition Sass

```
header {
    > img {
        display: block;
        margin: 20px auto;
    }
    + article {
        border: 2px solid red;
    }
    ~ blockquote {
        color: red;
    }
}
```

- Code CSS

```
header > img {
  display: block;
  margin: 20px auto;
}
header + article {
  border: 2px solid red;
}
header ~ blockquote {
  color: red;
}
```

Définition imbriquée des liens

- Exercice : Définition imbriquée des différents états des liens :
  - *a:link*
  - *a:visited*
  - et *a:hover*

Imbrication de propriétés

- Définition Sass

```
h1 {
    font: {
        size: 1.5em;
        weight: bold;
        variant: small-caps;
    }
    margin: 15px 40px;
}
```

Remarque : Espace entre : et { !

- Code CSS

```
h1 {
    font-size: 1.5em;
    font-weight: bold;
    font-variant: small-caps;
    margin: 15px 40px;
}
```

Imbrication : regroupement de sélecteur

- L'imbrication des règles supporte le regroupement des sélecteurs avec la virgule ,

- Définition Sass

```
article, .introduction {
    em, strong {
        background-color: #333;
        color: white;
    }
}
```

- Code CSS

```
article em,
article strong,
.introduction em,
.introduction strong {
    background-color: #333;
    color: white;
}
```

Imbrication de média query

- Regroupement des règles d'un même élément
- Définition Sass :

```
h1 { font-size: 3em;
     color: red;
     @media screen and (max-width: 960px) {
          font-size: 2em; }
}
```

- Code CSS :

```
h1 { font-size: 3em;
     color: red }
@media screen and (max-width: 960px) {
     h1 { font-size: 2em; }
}
```

Variable

- Définition de variables pour éviter les répétitions

  $nomvar: valeur;

- Stockage de couleur, nombre, chaîne ou liste

```
$coul: red;                 // rgb(255,0,0) ou #ff0000
$trait: 5px;                // avec l'unité
$policeDef: arial;          // possibilité d'imbriquer $policeDef dans $police
$police: ("Trebuchet MS", $policeDef, sans-serif);    // valeurs multiples
h1 {
        color:    $coul;
        border:   $trait  so
        font-family: $poli
        padding: $trait+1
}
```

```
h1 {
        color: red;
        border: 5px solid red;
        font-family: "Trebuchet MS", arial, sans-serif;
        padding: 15px;          // Addition de 5px + 10
}
```

Var. *globale* vs var. *locale*

- 2 types de variables
  - globale définie hors bloc
  - locale définie dans un bloc

```
$coul1: blue;        // variable globale
$coul2: $coul1;      // affectation d'une variable à une autre variable
h1 {
    color: $coul2;
}
p {
    $coul3: red;     // variable locale
    border: 1px solid $coul3;
}
```

Variable

- Variable et sigle de concaténation
  - Ajout de l'unité grâce à la concaténation +

```
$coul: blue;
$taille: 3;

h1 {
    color: $coul;
    font-size: $taille + em;
}
```

Interpolation de variable

- L'interpolation permet d'intégrer le résultat d'une expression SASS dans un morceau de code CSS
- L'interpolation s'insère avec les caractères #{… }

SCSS

```
$path: "../fonts/roboto";
$fam: family;
@font-face {
    src: url("#{$path}/roboto.woff2") format("woff2");
    font-#{$fam}: "Roboto";
    font-weight: #{75 + 25};
}
```

CSS

```
@font-face {
    src: url("../fonts/roboto/roboto.woff2") format("woff2");
    font-family: "Roboto";
    font-weight: 100;  }
```

Exercice Interpolation de variable

- Définir des variables ($mobile, $tablette) pour chaque dimension d'écran (680px, 960px) afin de rendre plus lisible les Media queries.

```
h1 {                                    // Code CSS à réécrire en SASS
    font-size: 3em;
    font-weight: bold;
}
@media screen and (max-width: 960px) {
    h1 {
            font-size: 2em;
            font-weight: normal;  }
}
```

Liste

- Définition d'une liste

$nomliste1: val1 val2;
$nomliste2: val1, val2;
$nomliste3: (val1 val2);
$nomliste4: (val1, val2);
$listevide: ();

```
$marge: 50px 10px 0;          // équiv. $marge: (50px 10px 0);
$nb: length($marge);          // 3 éléments dans $marge
h1 {
      margin: $marge;
}
```

Attention au séparateur !

– 2 séparateurs possibles : comma , ou space
– length() : retourne le nombre d'éléments d'une liste

Liste

- Accès à un élément
  - *nth(liste, indice) :* retourne la valeur située au numéro d'indice indiqué

```
$marge: 50px 10px 0;
h1 {
    margin: {
            right: nth($marge, 2);        // valeur ?  10px
            left: nth($marge, -1);        // valeur ?   0
}
```

Remarque : les indices débutent à 1 !

-1 fait référence au dernier élément

-2 à l'avant-dernier…

Liste

- Ajout d'un élément
  - *append(liste, valeur) :* ajoute une nouvelle valeur à la fin de la liste

```
$marge: (50px 10px 0);
$marge: append($marge, 100px);
h1 {
    margin: $marge;  // 50px 10px 0 100px
}
```

Liste

- Recherche d'un élément
  - *index(liste, valeur) :* retourne l'index où se trouve la 1ère valeur trouvée, null sinon

```
$marge: (50px auto 40px);
$position: index($marge, auto); // auto est en  2ème position
```

- Remplacement d'un élément
  - *set-nth(liste, indice, valeur) :* remplace l'élément situé à l'indice spécifié par la nouvelle valeur

```
$marge: (50px auto 40px);
$marge: set-nth($marge, 2, 10px);     // 50px 10px 40px
```

Liste

- Fusion de listes
  - *join(liste1, liste2) :* concatène liste1 avec liste2

```
$liste1: (50px 10px);
$liste2: (40px 10px);
$liste: join($liste1, $liste2);      // 50px 10px 40px 10px
```

  - *join(liste1, liste2, $separator: separateur) : précise le séparateur à utiliser : comma (,) ou space*

```
$liste3: join((blue, red), (#ff0, #00f), $separator: space);
p {
    border-color: $liste3;           // blue red #ff0 #00f
}
```

Exercice Liste

- Soit les deux listes suivantes :

  $GdeMg(40px,20px) et *$PteMg(10px,5px)*

  Fusionner ces 2 listes dans *$Mg*.

  A quel indice est située la valeur 20px ?

  Ajouter à la fin de $Mg une valeur (en px) égale à l'indice trouvé précédemment.

  Quelle est la longueur de la nouvelle liste constituée ?

  Définir avec uniquement $Mg les marges externes suivantes:

  - main : 40px ↑ ↓ 20px →←
  - main section et main article : 10px ↑ ↓ 5px →←
  - main p : la longueur de $Mg (en px) ↑ ↓ 2px →←

@debug, @warn, @error

- @debug permet, sans arrêter le programme, de visualiser le contenu de variables

```
@debug "Liste: #{$liste}";        // non fonctionnel avec VSC
```

- @warn n'arrête pas le programme mais permet d'être averti sur certains problèmes

```
@warn "Liste: #{$liste}";
```

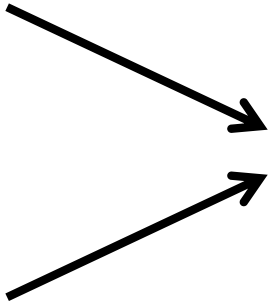
- @error stoppe le programme mais peut néanmoins fournir des informations

```
@error "Liste: #{$liste}";
```

Héritage

- @extend : permet d'indiquer qu'un élément hérite des propriétés d'un autre élément
- Définition Sass :

```
selecteur1 {
    prop1: valeur1;
    prop2: valeur2;
}
selecteur2 {
    @extend selecteur1;
    prop3: valeur3;
}
```



- Code CSS :

```
selecteur1, selecteur2 {
    prop1: valeur1;
    prop2: valeur2;
}
selecteur2 {
    prop3: valeur3;
}
```

Exercice héritage

- Définir la classe bouton avec :
  - les angles arrondis de 10px pour 2 angles sur 4
  - une ombre noire de 5px avec un flou de 2px
- Définir une figure et un titre principal avec :
  - les mêmes propriétés que la classe bouton
  - des marges automatiques →←
  - une largeur de 50%

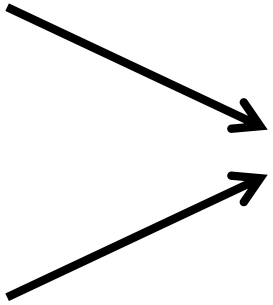
Mixin

- Mixin : Ensemble de codes CSS réutilisable
- Définition Sass :

```
@mixin nom-mixin {
    prop1: valeur1;
    prop2: valeur2;
}
```

- Utilisation Sass :

```
selecteur {
    @include nom-mixin;
}
```



- Code CSS :

```
selecteur {
    prop1: valeur1;
    prop2: valeur2;
}
```

Exercice Mixin

- Programmer un mixin *posit-f* pour définir :
  - une bordure de 1px avec un trait continu noir
  - une marge externe de 5px en haut
- Tester le mixin sur :
  - une image
  - un titre principal

Mixin avec argument

- Paramétrisation d'un mixin
- Définition Sass :

```
@mixin nom-mixin($arg) {
    prop1: $arg;
}
```

- Utilisation Sass :

```
selecteur1 {
    @include nom-mixin(param1); }
selecteur2 {
    @include nom-mixin(param2); }
```

- Code CSS :

```
selecteur1 {
    prop1: param1; }
selecteur2 {
    prop1: param2; }
```

Exercice Mixin avec argument

- Modifier le mixin *posit-f* pour définir :
  - une bordure de 1px avec un trait continu noir
  - un positionnement flottant à droite/gauche
  - une marge externe
    - de 5px en haut
    - et 10px du côté droit/gauche
- Tester le mixin sur :
  - une image avec un positionnement à droite
  - un titre principal avec un positionnement à gauche

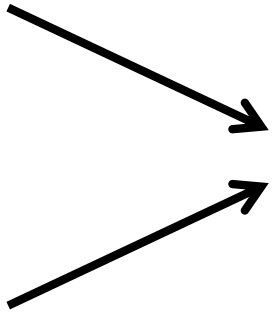
Mixin, argument et valeur par défaut

- Appel du mixin avec ou sans argument
- Définition Sass :

```
$var: val;
@mixin nom-mixin($arg: $var) {
    prop1: $arg;
}
```

- Utilisation Sass :

```
selecteur1 {
    @include nom-mixin(param); }
selecteur2 {
    @include nom-mixin; }
```



- Code CSS :

```
selecteur1 {
    prop1: param; }
selecteur2 {
    prop1: val; }
```

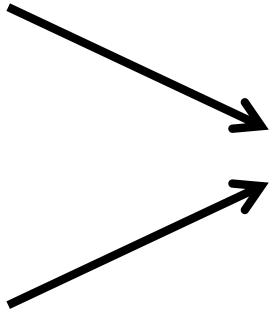
Mixin, argument et valeur par défaut

- Appel du mixin avec 0, 1 ou 2 arguments
- Définition Sass :

```
$var1: val1; $var2: val2;
@mixin nom-mixin($arg1: $var1, $arg2: $var2) {
    prop1: $arg1;
    prop2: $arg2; }
```

- Utilisation Sass :

```
selecteur1 {
    @include nom-mixin(par1); }
selecteur2 {
    @include nom-mixin($arg2: par2); }
```



- Code CSS :

```
selecteur1 {
    prop1: par1;
    prop2: val2; }
selecteur1 {
    prop1: val1;
    prop2: par2; }
```

Exemple Mixin, argument et valeur

- Programmer un mixin *ombre* pour définir :
  - un ombrage de bloc :
    - avec un décalage vers le haut et vers la droite de 10px (ou d'une autre valeur indiquée)
    - avec un flou de 5px (ou d'une autre valeur indiquée)
    - avec une ombre rouge
- Tester le mixin sur :
  - une image avec une ombre de base
  - un titre h1 pour une ombre de 8px dans les 2 dimensions et un flou de 3px

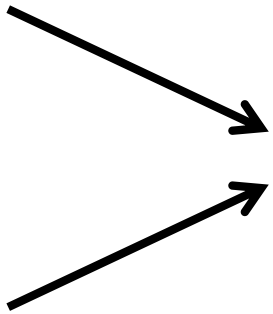
Extension/Placeholder

- Placeholder: factorisation de code dans des classes abstraites non converties après génération du CSS
- Définition Sass :

```
%placeholder {
    prop1: val1;
    prop2: val2; }
```

- Utilisation Sass :

```
selecteur1 {
    @extend %placeholder;
    prop3: val3; }
selecteur2 {
    @extend %placeholder;
    prop4: val4; }
```



- Code CSS :

```
selecteur1,
selecteur2 {
    prop1: val1;
    prop2: val2;
}
selecteur1 {
    prop3: val3;
 }
selecteur2 {
    prop4: val4;
 }
```

Exercice Extension/Placeholder

- Programmer un placeholder *marge* pour définir :
  - une marge gauche et droite de 10px
  - une marge haute et basse de 5px
- Tester le placeholder sur :
  - une image qui possède également une bordure solide et noire de 1px
  - un titre principal

Mixin *vs* Placeholder

- Définition avec @mixin et @include :

```
@mixin btn($color) {
    padding: 10px;
    border-radius: 5px;
    color: $color; }
.btn-primary {
    @include btn(blue); }
.btn-secondary {
    @include btn(red); }
```

- Définition avec % et @extend :

```
%btn {
    padding: 10px;
    border-radius: 5px; }
.btn-primary {
    @extend %btn;
    color: blue; }
.btn-secondary {
    @extend %btn;
    color: red; }
```

Mixin *vs* Placeholder

- Code CSS avec Mixin

```
.btn-primary {
    padding: 10px;
    border-radius: 5px;
    color: blue; }
.btn-secondary {
    padding: 10px;
    border-radius: 5px;
    color: red; }
```

- Code CSS avec Placeholder

```
.btn-primary,
.btn-secondary {
    padding: 10px;
    border-radius: 5px; }
.btn-primary {
    color: blue; }
.btn-secondary {
    color: red; }
```

Mixin *vs* Placeholder

- Mixin
  - ⊕ flexibilité grâce à l'usage des arguments
  - ⊖ répétitions de code
- Placeholder
  - ⊕ CSS optimisé, moins de répétitions de code
  - ⊖ répétitions de sélecteur ➔ classification perturbée
- Solution : combiner les 2 en fonction des besoins

@use

- @use permet l'importation d'un fichier SASS dans un autre fichier de définition SASS.

```
// Fichier _variables.scss
$var1: (50px 100px);
$var2: purple;
```

```
// Fichier styles.scss
@use  "variables"          // importation de _variables.scss
h1 { margin: $var1; }
```

```
// Fichier styles.css généré
h1 { margin: 50px 100px; }
```

- @use est à utiliser au dépend de @import qui est obsolète

Modules Sass

- Sass dispose de modules qui intègrent des fonctions prédéfinies.

  Ces modules sont chargés avec la commande @use :

  - @use "sass:string"; fonctions relatives aux chaînes de caractères
  - @use "sass:list"; fonctions relatives aux listes
  - @use "sass:map"; fonctions relatives aux maps
  - @use "sass:color"; fonctions relatives aux couleurs
  - @use "sass:math"; fonctions mathématiques

Fcts liées aux chaînes sass:string

- @use "sass:string"; pour charger le module
- Manipulation des chaînes de caractères
  - *to-lower-case(chaîne), to-upper-case(chaîne)*
  - *length(chaîne) : nombre de caractères de la chaîne*
  - *slice(chaîne, deb, fin) : sous-chaîne issue de la chaîne entre les indices de début et de fin*
  - *index(chaîne, ss-chaîne) : indice en cas de présence de la sous-chaîne dans la chaîne*
  - *insert(chaîne, ss-chaîne, indice) : insertion d'une sous-chaîne dans la chaîne à l'indice indiqué*
  - *quote(chaîne)/unquote(chaîne) : ajout/retrait des " "*

Fonctions mathématiques sass:math

- @use "sass:math"; pour charger le module
- Fonctions de calcul
  - *max(val1, val2…), min(val1, val2…)*
  - *floor(val), ceil(val), round(val)*
  - *random() : valeur aléatoire comprise entre 0 et 1*
  - *pow(val), sqrt(val), log(val) ...*
- Fonctions liées aux unités
  - *compatible(val1, val2) : compatibilité des unités ? true, false*
  - *is-unitless(val) : valeur sans unité ? true, false*
  - *unit(val) : unité de la valeur ? px, em, %, ""*
    (utile pour le débogage)

Fcts liées aux couleurs sass:color

- @use "sass:color"; pour charger le module
- Fonction de définition :
  - *rgb(red, green, blue, alpha)*
  - *hsl(hue, saturation, lightness, alpha)*
  - *hwb(hue, whiteness, blackness, alpha)*
- Fonctions d'acquisition :
  - *red(coul), green(coul), blue(coul) : valeur entre 0 et 255*
  - *alpha(coul) : valeur d'opacité comprise entre 0 et 1*
  - *hue(coul) : valeur entre 0deg et 360deg*
  - *saturation(coul), lightness(coul) : valeur entre 0% et 100%*
  - *whiteness(coul), blackness(coul) : valeur entre 0% et 100%*

Fcts liées aux couleurs sass:color

- Fonctions de transformation :
  - *adjust(coul, $red: val, $green: val, $blue: val, $hue: val, $saturation: val, $lightness: val, $whiteness: val, $blackness: val, $alpha: val) : modifie la valeur d'une ou plusieurs propriétés de la couleur*
  - *scale(cf. adjust) : modification selon une échelle (-100% à 100%)*
  - *change(cf. adjust) : attribue une nouvelle valeur à une ou plusieurs propriétés de la couleur*
  - *grayscale(coul) : transforme la couleur en niveau de gris*
  - *invert(coul, val%) : inversion d'une couleur*
  - *complement(coul) : complément d'une couleur*

Attention :

- *lighten(), darken(), saturate(), desaturate(), opacify() changent la valeur d'une propriété à l'aide d'un montant*
  MAIS ajouter 30% à une couleur déjà sombre la transforme en noir !

Fcts liées aux couleurs sass:color

- Exemple d'utilisation de fonctions

```
@use "sass:color";

h1 {
// définition de couleurs selon différents modes colorimétriques
    $col: rgb(115, 255, 255, 0.8);
// ajustement de caractéristiques : rgb(235, 255, 165, 0.3)
    color: color.adjust($col, $red:+120, $blue:-90, $alpha:-0.5);
}
```

Fcts liées aux couleurs sass:color

- Exercice : utilisation de fonctions

```
@use "sass:color";

h1 {
// définition de couleurs selon différents modes colorimétriques
      $col: rgb(115, 255, 255, 0.8);
// modification selon un % :
// + 50% de rouge et – 80% de bleu
      background-color:
// attribution d'une valeur par caractéristique :
// redéfinition du rouge à 0 et d'alpha à 1
      border-color:
}
```

Fonction personnalisée

- Syntaxe :

  @function nomfct($arg) {... @return(val); }

- Possibilité d'utiliser :
  - 1 ou plusieurs arguments
  - 1 seule valeur de retour

```
@use "sass:color";
@function inverser($coul) {
    @return (color.complement($coul));
}
h1 { color: inverser(purple);}              // color: green;
```

Fonction personnalisée

- Exercice : Ecrire une fonction calcul() permettant d'appliquer un coefficient multiplicateur, passé en paramètre, à une valeur de base définie à 25px.

  La fonction retourne la valeur calculée.

  Appeler calcul() pour définir une marge avec un coefficient de 3 à un titre principal.

Condition @if… @else

- Syntaxe :

  @if (condition) { ... } @else { ... }

- Condition définie à l'aide des opérateurs :

  <, >, <=, >=, ==, !=

- Possibilité d'utiliser les opérateurs and et or

  @if ((cond1) and/or (cond2)) { ... } @else { ... }

```
@if ($coul == #fff) {
    color: #000;
} @else {
    color: #fff; }
```

Exercice @if… @else

- Programmer le mixin coulTxtFond permettant de :
  - définir une couleur de texte
  - d'adapter la luminosité du fond en fonction de la couleur du texte :
    - couleur de texte claire → fond assombri à 40%
    - couleur de texte sombre → fond éclairci à 40%
- Appliquer le mixin à h1

Boucle @each

- Syntaxe :

  @each $elem in $ensemble { ... }

```
SASS
$tailles: 200px 100px;
@each $taille in $tailles {
    .bloc-#{$taille} {
        height: $taille;
        width:  $taille;
    }
}
```

```
CSS généré
.bloc-200px{
    height: 200px;
    width:  200px;
}
.bloc-100px{
    height: 100px;
    width:  100px;
}
```

Boucle @for

- Syntaxe :

  @for $var from min through max { ... }

```
SASS
@for $cpt from 2 through 3 {
    .marge-#{$cpt} {
            margin: $cpt * 10px;
            padding: 5px;
    }
}
```

```
CSS généré
.marge-2 {
  margin: 20px;
  padding: 5px;
}
.marge-3 {
  margin: 30px;
  padding: 5px;
}
```

Exercice @each - @for

- Programmer à l'aide de Sass, 6 classes permettant de définir des boutons carrés de 50px et de 100px de côté contenant les images de ces 3 réseaux sociaux :
  - instagram
  - twitter/X
  - et linkedin

Exemple de code CSS avec twitter/X

```
.btn-twitter-50px {
  height: 50px;
  width: 50px;
  background-image: url(images/btn-twitter.png);
  background-repeat: no-repeat;
}
```

Les images sont stockées dans le repertoire images.

Le nom des fichiers est défini ainsi : btn-nomreseau.png

Map

- @use "sass:map"; pour charger le module
- Syntaxe :

  $nommap: (clé1: val1, clé2: val2, ...);

  Association d'une valeur à chaque clé

  – *get($map, clé) :* permet de retrouver la valeur associée à la clé passée en parameter

```
$taille: (texte: 1em, titre: 1.2em, grostitre: 2em);
h1 {
    font-size: map.get($taille, grostitre);          // 2em
    margin: map.get($taille, intertitre);            // null
}
```

Map

– *has-key($map, clé) :* retourne un booléen indiquant si la clé est présente ou non dans la map

```
$taille: (texte: 1em, titre: 1.2em, grostitre: 2em);
$presenceCle: map.has-key($taille, grostitre);        // true
$presenceCle: map.has-key($taille, intertitre);       // false
```

– *merge($map1, $map2) :* retourne une map indiquant si la clé est présente ou non dans la map

```
$taille1: (texte: 1em, titre: 1.2em);
$taille2: (intertitre: 1.5em, grostitre: 2em);
$taille: map.merge($taille1, $taille2);
 // (texte: 1em, titre: 1.2em, intertitre: 1.5em, grostitre: 2em
```

Map

– *values($map) :* retourne une liste contenant toutes les valeurs de la map

```
$taille: (texte: 1em, titre: 1.2em, grostitre: 2em);
$listevaleur: map.values($taille);        // (1em, 1.2em, 2em)
```

– *keys($map) :* retourne une liste contenant toutes les clés de la map

```
$taille: (texte: 1em, titre: 1.2em, grostitre: 2em);
$listecle: map.keys($taille);                  // (texte, titre, grostitre)
```

Map

– *set($map, clé, valeur) :* met à jour la valeur associée à la clé passée en parametre

```
$taille1: (texte: 1em, titre: 1.2em, grostitre: 2em);
$taille2: map.set($taille1, titre, 1.5em);
                        // (texte: 1em, titre: 1.5em, grostitre: 2em);
```

– *remove($map, clé1, clé2…) :* retourne une liste dont les couples clé:valeur sont supprimées

```
$taille1: (texte: 1em, titre: 1.5em, grostitre: 2em);
$taille2: map.remove($taille1, texte, titre);    // (grostitre: 2em)
```

BEM - Bloc Elément Modificateur

- Convention de nommage : modulaire, réutilisable
  - Bloc : composant parent/autonome
    - *menu de navigation, formulaire, article, bouton, tableau...*
  - Elément : partie d'un bloc, enfant lié à son parent
    - *élément du menu, ligne d'un tableau, item d'une liste...*
  - Modificateur : variation de l'aspect, de la fonction, de l'état
    - *bouton actif/inactif, case cochée/décochée, titre petit/grand...*
- Notation : .bloc__element--modificateur
  - Exemple : *.menu*
    .menu__item
    .menu__item--actif

Notation BEM et &

- Définition Sass

```
.block {
        …
        &__element1 {
                …
                &--modifier {
                        …
                }
        }
        &__element2 {
                …
        }
}
```

- CSS après compilation

```
.block {
        …
}
.block__element1 {
        …
}
.block__element1--modifier {
        …
}
.block__element2 {
        …
}
```

Exercice @each, map, BEM

- Soit des couleurs :
  - Couleur de base : #fff
  - Couleur primaire : #0e7
  - Couleur secondaire : #e56
  - Couleur tertiaire : #267

  et une liste de noms de bouton et leur couleur de fond :
  - Danger : Couleur secondaire
  - Aide : Couleur tertiaire
- On souhaite disposer :
  - d'une définition générale (.btn) pour un bouton :
    - Police de 1em avec une graisse de 900
    - Marges internes et angles arrondis à 15px
    - Couleur de base pour le texte et primaire pour le fond
  - d'une définition spécifique pour 2 boutons :
    - Danger (.btn--danger) et Aide (.btn--aide)

Exercice @each, map, BEM

- Choisir le bon mode de stockage pour les informations

- Définir l'aspect du bouton de base .btn

- Compléter cette définition pour créer automatiquement des classes pour la mise en forme de 2 autres boutons à partir des informations disponibles.