LES VARIABLES CSS

Puisque cette fonction est toujours en développement dans certains navigateurs, veuillez consulter le tableau de compatibilité pour les préfixes à utiliser selon les navigateurs.
Il convient de noter qu'une fonctionnalité expérimentale peut voir sa syntaxe ou son comportement modifié dans le futur en fonction des évolutions de la spécification.
Les variables CSS sont des entités définies par les développeurs ou les utilisateurs d'une page Web, contenant des valeurs spécifiques utilisables à travers le document. Elles sont initialisées avec des propriétés personnalisées et accessibles en utilisant la notation spécifique var().

Utilisation simple

Voici comment on déclare une variable :

```
element {
  --main-bg-color: brown;
}
```

Et voici comment on l'utilise

```
element {
  background-color: var(--main-bg-color);
}
```

Problématique

Lors de l'élaboration de sites de grande envergure, leurs auteurs font parfois face à des soucis de maintenabilité. De grandes feuilles de styles sont utilisées et de nombreuses informations se répètent. Par exemple, maintenir un thème de couleurs à travers un document nécessite la réutilisation des valeurs des couleurs à plusieurs endroits dans les fichiers CSS. Modifier un thème, en changeant une couleur ou en le récrivant entièrement, devient alors une tâche complexe demandant de la précision, là où un simple trouver et remplacer ne suffit pas.

Le problème peut s'aggraver en utilisant les *frameworks* CSS puisque modifier une couleur demande de modifier le framework lui-même. Les pré-processeurs comme LESS ou Sass peuvent faciliter cette tâche, mais peuvent également complexifier le processus de création en ajoutant une

étape de compilation. Les variables CSS permettent d'utiliser une des principales fonctionnalités des pré-processeurs, sans cette étape de compilation.

Le deuxième avantage de ces variables vient du fait que le nom lui-même contient des informations sémantiques. Les fichiers CSS deviennent alors plus facile à lire et à comprendre : écrire main-text-color permet de mieux s'y retrouver au fur et à mesure de la lecture qu'une valeur hexadécimale comme #00ff00, surtout si la même couleur est utilisée dans un autre contexte.

Utilité des variables CSS

Dans les langages de programmation impératifs, comme Java, C++, ou encore JavaScript, le fonctionnement peut être tracé à travers la notion de variables. Les variables sont des noms symboliques associés à une valeur donnée pouvant varier avec le temps.

Dans un langage déclaratif comme CSS, les valeurs changeant dans le temps et le concept de variables ne sont pas habituels.
Toutefois, CSS introduit la notion de variables en cascade afin de résoudre le problème de la maintenabilité. Cela permet de faire référence symboliquement à une valeur à travers l'arborescence CSS.

Définition

Les variables CSS ont actuellement deux formes :

- les variables, qui sont des associations entre un identifiant et une valeur utilisables à la place de n'importe quelle valeur normale, en utilisant la notation fonctionnelle `var()` : `var(--example-variable)` retourne la valeur de `--example-variable`.
- les propriétés personnalisées, qui sont des propriétés spéciales notées `--*` où * représente le nom de la variable. Elles sont utilisées pour définir la valeur d'une variable donnée : `--example-variable: 20px;` est une déclaration en CSS, utilisant la propriété personnalisée `--*` pour initialiser la valeur de la variable CSS `--example-variable` à `20px`.

Note : Le préfixe de propriété personnalisée était noté `var-` dans les précédentes spécifications, mais a ensuite été changé pour `--`. Firefox 31 et supérieurs respectent cette nouvelle notation. (bug 985838)

Les propriétés personnalisées sont similaires aux propriétés ordinaires. Elles sont sujettes à cascade et héritent leur valeur de leur parent si elles ne sont pas redéfinies.

Premiers pas avec les variables CSS

Commençons avec cette feuille CSS simple colorant les éléments de différentes classes avec la même couleur :

```
.un {
  color: white;
  background-color: brown;
  margin: 10px;
  width: 50px;
  height: 50px;
  display: inline-block;
}
.deux {
  color: white;
  background-color: black;
  margin: 10px;
  width: 150px;
  height: 70px;
  display: inline-block;
}
.trois {
  color: white;
  background-color: brown;
  margin: 10px;
  width: 75px;
}
.quatre {
  color: white;
  background-color: brown;
```

```
  margin: 10px;
  width: 100px;
}
.cinq {
  background-color: brown;
}
```

Appliquons-le à ce code HTML :

```
<div>
      <div class="un"></div>
      <div class="deux">

            Texte <span class="cinq">- encore du texte</span>

      </div>
    <input class="trois">
    <textarea class="quatre">Lorem Ipsum</textarea>
</div>
```

ce qui donne ceci :

OUVRIR DANS CODEPEN OUVRIR DANS JSFIDDLE

Remarquez la répétition dans le CSS. La couleur d'arrière-plan est définie à `brown` à plusieurs endroits. Certaines déclarations peuvent être faites plus haut dans la cascade et le problème se résoudra grâce à l'héritage. Mais pour des projets non-triviaux, cela n'est pas toujours possible. En déclarant une variable dans la pseudo-classe `:root`, un développeur CSS peut éviter certaines répétitions en utilisant cette variable.

```
:root {
  --main-bg-color: brown;
}
```

```css
.un {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 50px;
  height: 50px;
  display: inline-block;
}

.deux {
  color: white;
  background-color: black;
  margin: 10px;
  width: 150px;
  height: 70px;
  display: inline-block;
}
.trois {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 75px;
}
.quatre {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 100px;
}

.cinq {
  background-color: var(--main-bg-color);
}
```

Ce code donne le même résultat que précédemment mais permet de n'utiliser la propriété désirée qu'une seule fois.

Héritage des variables CSS et valeurs par défaut

Il y a un héritage des propriétés personnalisées. Cela signifie que si une propriété n'est pas définie sur un élément, la valeur prise en compte sera celle utilisée pour la propriété de l'élément parent. Le fragment de document suivant :

```
<div class="un">
  <div class="deux">
    <div class="trois">
    </div>
    <div class="quatre">
    </div>
  </div>
</div>
```

associé à cette feuille de style :

```
.un {
  --test: 10px;
}

.trois {
  --test: 2em;
}
```

Dans ce cas, les résultats de var(--test) seront :

- 10px pour l'élément avec class="deux"
- 2em pour l'élément avec class="trois"
- 10px pour l'élément avec class="quatre" : la valeur est héritée depuis le parent
- invalid value pour l'élément avec class="one", c'est la valeur par défaut utilisée pour les différentes propriétés personnalisées.

-

Avec var() on peut définir plusieurs valeurs par défaut lorsque la variable donnée n'est pas définie. Cela peut s'avérer utile lorsqu'on travaille avec des éléments personnalisés (Custom Elements) et le Shadow DOM.
Le premier argument passé à la fonction est le nom de la propriété personnalisée qui doit être substituée. Le deuxième argument, s'il est fourni, indique la valeur par défaut qui est utilisée lorsque la propriété personnalisée en question est invalide :

```
.deux {
  color: var(--my-var, red);
  /* Red si --my-var n'est pas définie */
}

.trois {
  background-color: var(--my-var, --my-background, pink);
  /* rose (pink) si --my-var et --my-background ne sont pas
définies */
}
```

Note : Des problèmes de performances ont pu être observés[source ?] car le navigateur doit analyser l'ensemble des variables pour voir si elles sont disponibles.

Validité et valeurs

Le concept classique de validité en CSS, lié à chaque propriété, n'est pas très utile en ce qui concerne les propriétés personnalisées. Quand la valeur d'une propriété personnalisée est lue, le navigateur ne sait pas à quel moment elle sera utilisée. Il doit donc considérer quasiment toutes les valeurs comme valides.

Malheureusement, ces valeurs valides peuvent être utilisées, via la notation fonctionnelle `var()`, dans un contexte où cela n'aurait pas de sens. Les propriétés et variables personnalisées peuvent mener à des déclarations CSS invalides, conduisant à un nouveau concept de *valide lors de l'exécution*.

Compatibilité avec les navigateurs

Fonctionnalité	Chrome	Firefox (Gecko)	Edge	Internet Explorer	Opera	Safari (WebKit)
Support simple	(Oui)-webkit 33.0 Pas de support 34.0[2] 49.0	29 (29)[3] 31 (31)	(Oui)	Pas de support	36.0	9.1

[1] Chrome a initialement implémenté cette fonctionnalité en utilisant une autre syntaxe, nécessitant de préfixer le nom des propriétés personnalisées avec `-webkit-var-` pour les définir. Elles pouvaient ensuite être utilisées sans le préfixe dans une fonction `-webkit-var()`. De plus, l'implémentation était dissimulée sous la préférence *Enable experimental WebKit features* dans `chrome://flags`, par la suite renommée *Enable experimental Web Platform features*.

[2] Chrome 34.0 a retiré cette fonctionnalité à cause de problèmes de performances.

[3] Cette fonctionnalité a été implémenté derrière la préférence `layout.css.variables.enabled`, dont la valeur par défaut est `false` et utilisant l'ancienne syntaxe `var-variablename` dans Gecko 29. Depuis Gecko 31, la préférence est activé par défaut et la nouvelle syntaxe `--variablename` est utilisée. À partir de Gecko 55, la préférence `layout.css.variables.enabled` est retirée et la fonctionnalité est donc nécessairement activée.

Mobile

Fonctionnalité	Android	Webview Android	Firefox Mobile (Gecko)	Edge	IE Phone	Opera Mobile	Safari Mobile	Chrome pour Android
Support simple	Pas de support	49.0	29	(Oui)	?	?	9.1	49.0