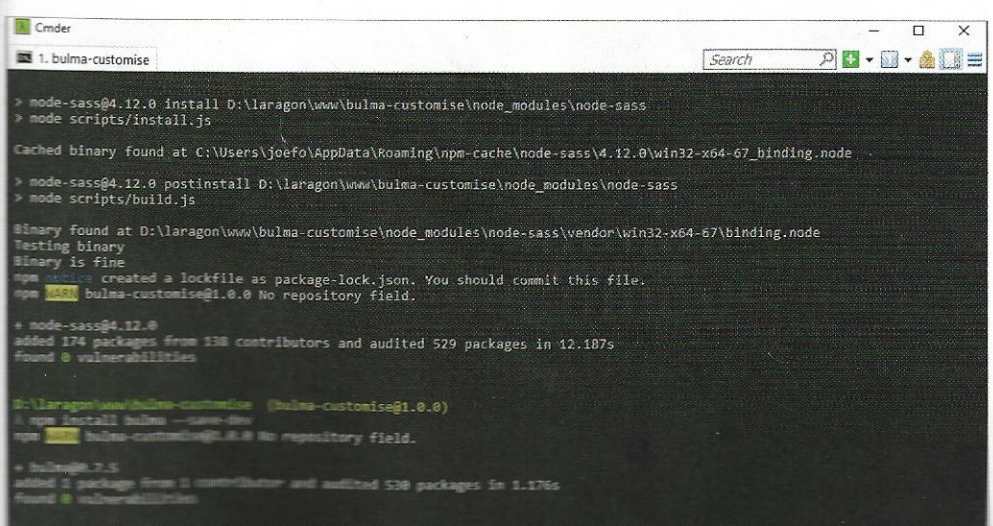
Dans un précédent numéro, le Framework CSS Bulma était sous le feu des projecteurs, et nous avons construit un site web classique en utilisant les classes CSS, les couleurs et les polices intégrées. La plupart de vos projets impliquent que vous suiviez les directives d'une marque, ou du moins que vous utilisiez autre chose que les couleurs Bulma standards. Heureusement, Bulma offre de grandes possibilités de personnalisation, et avec 415 variables Sass, vous pouvez modifier le thème entier d'un site en quelques lignes de code. Le framework est également modulaire, ce qui vous permet de n'importer que la partie de Bulma que vous voulez personnaliser. Si un projet ne requiert que les classes de colonne et un système de grille, c'est possible.

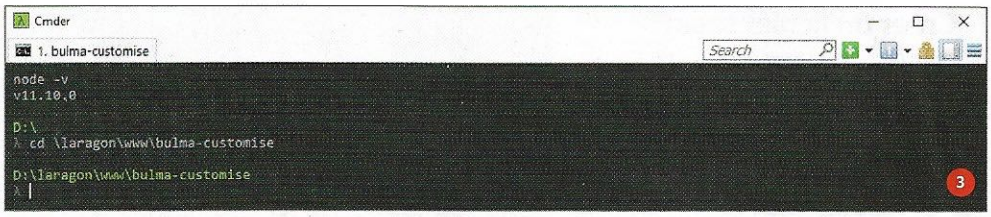
Mais il ne s'agit pas que de changer quelques lignes dans un fichier CSS comme vous en avez l'habitude. Bulma est basé sur Sass, une des extensions de CSS qui vous permet d'utiliser des variables et des importations inline. Sass est extrêmement utile, et peut vous faire gagner beaucoup de temps à long terme. Mais il peut aussi être un peu intimidant à apprendre, et le workflow est très différent pour ceux qui ont l'habitude de modifier directement les fichiers. Vous allez rapidement rattraper le temps passé à configurer un projet qui utilise Sass grâce à ses puissantes fonctionnalités.

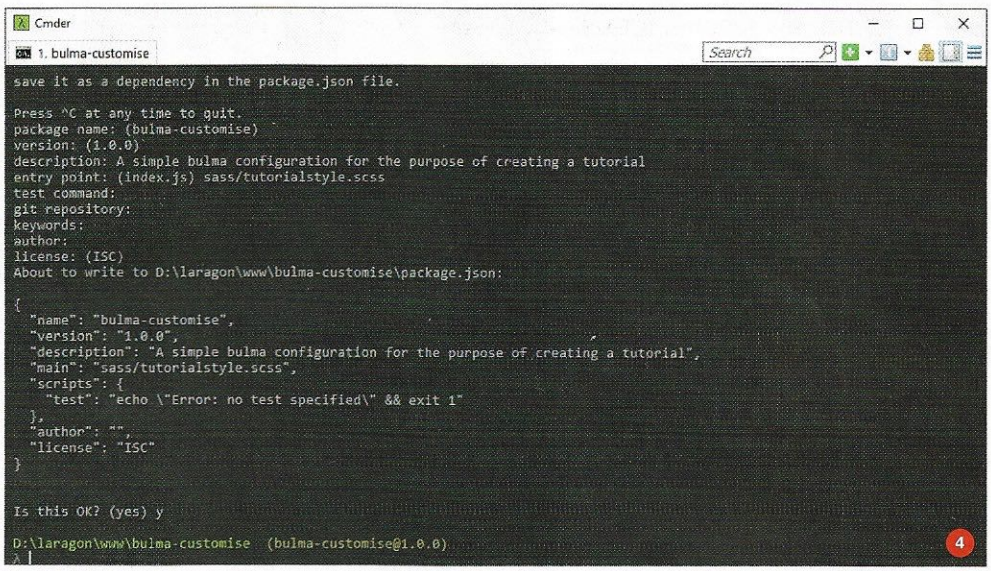
Dans ce tutoriel, nous allons paramétrer un nouveau projet Bulma avec npm, puis utiliser les commandes pour définir un environnement de travail paramétrable qui nous donnera non seulement le contrôle total sur les nombreuses variables de Bulma, mais qui servira aussi d'introduction à des outils comme node et npm.

1. Paramétrer le projet

Créez un nouveau dossier, puis donnez-lui un nom. C'est ici que nous allons placer tous les fichiers du projet et exécuter les commandes npm. N'oubliez pas le chemin d'accès du dossier, parce que vous allez en avoir besoin.







2. Télécharger node

Nous allons utiliser node et npm pour installer Bulma et pour gérer les tâches automatisées comme la compilation du Sass en CSS quand nous détectons un changement. Allez sur le site de node, nodejs.org/en/download et téléchargez le programme d'installation pour votre système d'exploitation.

3. Installer node et npm

Une fois que node est téléchargé, lancez l'installation. Vous pouvez vérifier que node est bien installé en ouvrant l'invite de commandes ou, si vous êtes sur Mac, le terminal. Tapez node -v et vous devriez obtenir un numéro de version.

4. npm init, une configuration pas-à-pas

Toujours dans l'invite de commandes ou dans le terminal, remplacez le répertoire par le dossier créé pendant l'étape 1. Exécutez npm init pour lancer un processus de configuration. La première invite est le nom de votre projet, suivi par la version et la description, qui sont facultatives. Quand on vous demande un point d'accès, saisissez sass/tutorialstyles.scss.

5. Installer les dépendances

Pour inclure Bulma dans notre projet et nous permettre de compiler des fichiers .scss en CSS automatiquement, nous avons besoin de deux dépendances. Saisissez-les dans cmd ou dans le terminal comme précédemment. Une fois qu'elles sont installées, vérifiez votre package.json, qui devrait désormais avoir une section appelée dev dependencies avec les deux paquets.

```
npm install node-sass --save-dev
npm install bulma --save-dev
```

Les couleurs de Bulma sont stockées en hsl

Les couleurs dans Bulma sont stockées en hsl par défaut, mais les formats hex et RGB sont aussi supportés. Indépendamment du format utilisé, les variables de couleurs peuvent être écrasées dans

6. Créer un fichier sass

Créez un nouveau dossier dans le répertoire de votre projet et nommez-le sass. Dans ce dossier, créez un nouveau fichier appelé tutorialstyles.scss, en vérifiant bien que le nom de fichier est le même que le point d'accès défini dans l'étape 4. Ouvrez le fichier et tapez :

```
@charset "utf-8";
@import "../node_modules/bulma/bulma.sass";
```

7. Créer une page d'index

Nous devons maintenant créer une page qui utilise Bulma pour appliquer des styles. Créez un index.html dans le dossier du projet. Écrivez ou importez du HTML qui utilise Bulma. Il suffit d'utiliser une balise link vers tutorialstyles.css. Ce fichier n'existe pas pour l'instant, mais c'est là que nous allons compiler notre scss.

```
<link rel="stylesheet" href="css/
tutorialstyles.css">
```

8. Ajouter des scripts Node

Ouvrez le fichier package.json dans le dossier du projet. Dans la zone Scripts, ajoutez le code suivant. Css-build exécute node-sass qui lit le fichier sass que nous avons créé et le compile en CSS. Css-watch exécute la commande build à chaque fois que nous sauvegardons le fichier. Bulma-watch est un raccourci facile à mémoriser qui exécute la commande watch.

```
"scripts": {
   "css-build": "node-sass --omit-source-
map-url sass/tutorialstyles.scss css/
tutorialstyles.css",
   "css-watch": "npm run css-build --
--watch",
   "bulma-watch": "npm run css-watch"
 },
```

9. Compiler et tester le code CSS

Ouvrez index.html dans un navigateur. Vous allez remarquer que, comme on pouvait s'y attendre, il n'a absolument aucun style. Dans l'invite de commande ou dans le terminal, exécutez la commande build, pour effectuer un passage unique sur le fichier tutorialstyles.scss et le compiler en tutorialstyles.css. Si vous rencontrez des erreurs ici, vérifiez que nos noms de fichiers sont bien cohérents.

```
npm run css-build
```

10. Choisir et définir des couleurs

En rechargeant index.html dans le navigateur, vous allez voir que le CSS Bulma par défaut a été importé. Nous devons maintenant définir des variables de couleurs. Ouvrez tutorialstyles.scss, où vous allez devoir paramétrer un nom et un code hex pour chaque couleur à inclure au-dessus de la ligne importe de Bulma.

```
// ersonnalisation des couleurs
$pink: #FF3562;
$purple: #693668;
$maroon: #a74482;
$dark-blue:#1b1b3a;
// Importe bulma et applique nos changements
@import "../node_modules/bulma/bulma.sass"
```

11. Importer et définir les polices

Importez une police au-dessus des variables de couleur, puis attribuez à chaque police une variable facile à mémoriser. Pour l'instant, importez une police pour les titres et une pour le corps du texte. Nous pouvons aussi définir la taille de la police globale ainsi que la hauteur de ligne et d'autres propriétés en utilisant des variables.

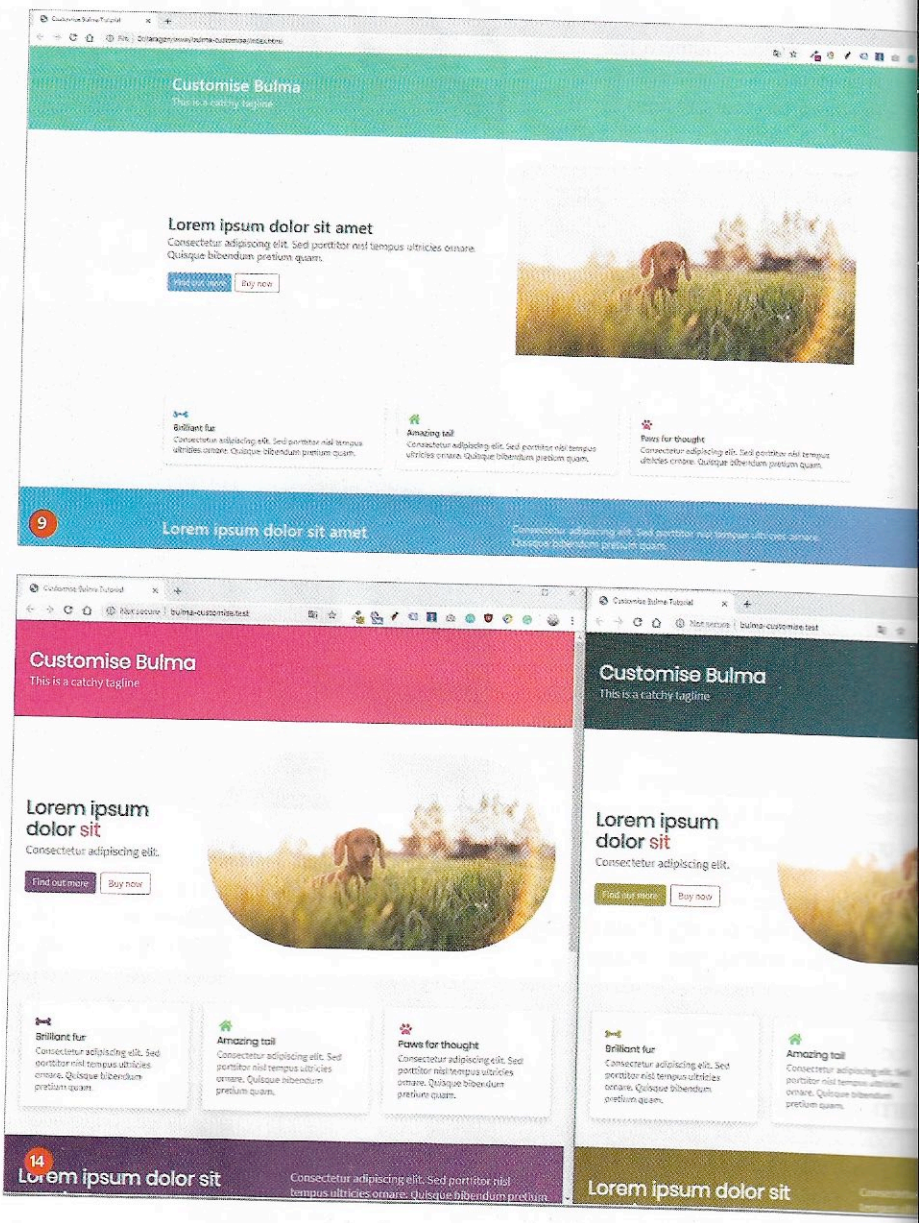
```
@import url('HYPERLINK "https://fonts.
googleapis.com/css?family=Poppins%7CSou
ans+Pro&display=swap" \hhttps://fonts.
googleapis.com/css?family=Poppins|Sour
s+Pro&display=swap');

// Personnalisation des couleurs

...
//Personnalisation des polices
$poppins: "poppins", sans-serif;
$source-sans: "Source Sans Pro", sans-
```

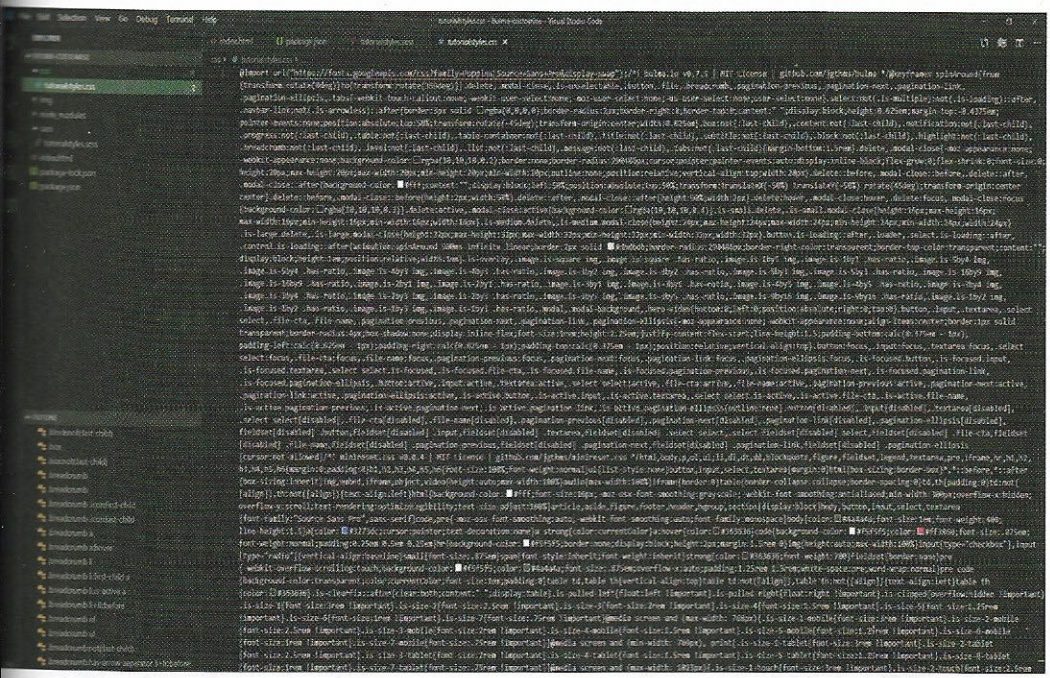
12. Assigner les variables

Les variables donnent beaucoup de puissance à Bulma. La couleur principale du design peut non seulement être modifiée en une seule ligne, mais



Bulma est modulaire

Si le projet ne nécessite que certaines parties de Bulma, comme les colonnes et la grille, cherchez les feuilles de styles de chaque module que vous voulez importer dans le répertoire /bulma/sass plutôt que d'importer /node_modules/bulma/bulma.sass.

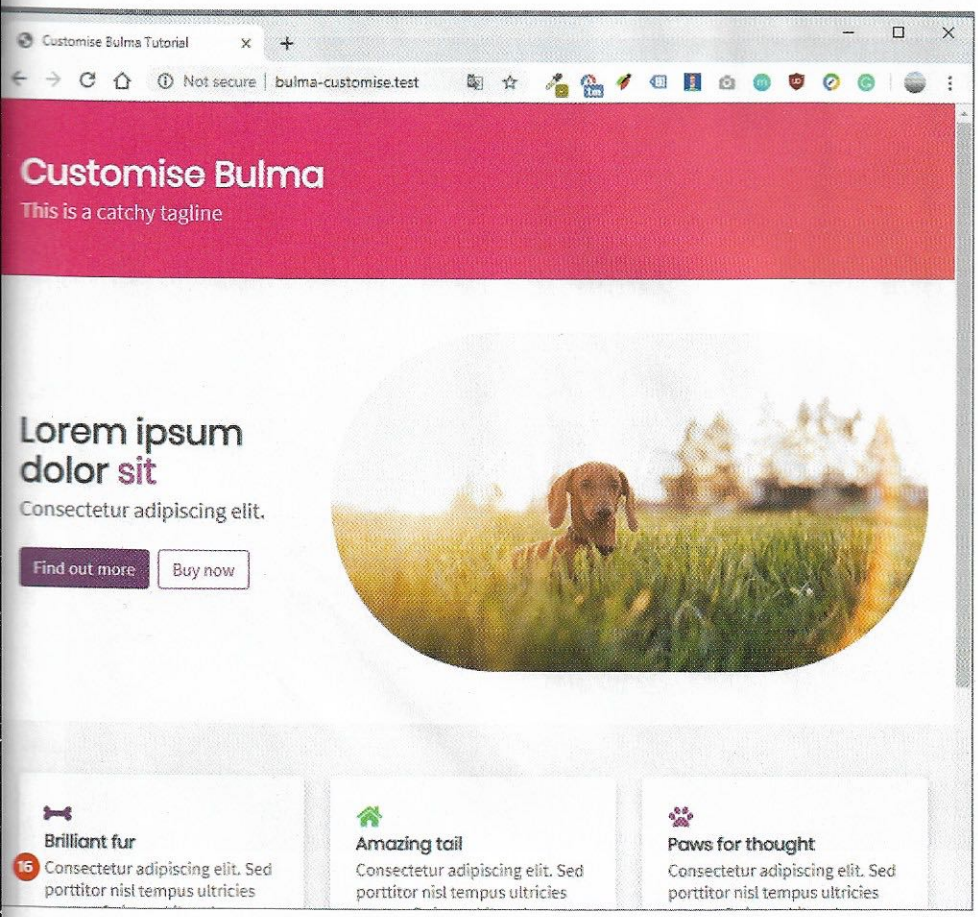


Minification automatique

Nous avons déjà npm qui nous attend pour enregistrer les fichiers tutorialstyles.scss et les compiler en CSS.

Comme Bulma couvre beaucoup de choses, tutorialstyles.css sera probablement assez gros. Pour gérer ce problème, vous pouvez minifier le fichier. La minification est le processus qui consiste à éliminer tous les retours à la ligne et le formatage d'un fichier CSS pour le rendre plus petit et plus rapide à charger. Le fichier sera illisible, mais travailler en scss élimine le besoin de lire le CSS compilé.

Node-sass inclut une option pour compresser le CSS, ce qui permet d'obtenir ce résultat. Pour l'activer, ouvrez package.json et modifier la ligne du script CSS-build. Après css/tutorialstyles.css et entre les guillemets, ajoutez --output-style compressed. Cela devrait ressembler au css-build suivant : "node-sass --omit-source-map- url sass/tutorialstyles.scss css/ tutorialstyles.css --output-style compressed". Si bulma-watch est déjà en cours d'exécution, annulez-le en utilisant control + c, enregistrez le package.json et exécutez-le à nouveau.



aiculs seront également automatiquement faits avec nouvelle couleur. Par exemple, la classe is-bold, qui oute un dégradé subtil sur la couleur d'arrière-plan, adapte à vos changements. Utilisez le rose défini plus t comme variable primary, puis passez la couleur info n purple, et modifiez le titre et les polices principales.

```
//Remplace des variables
$title-family: $poppins;
$family-primary: $source-sans;
$primary: $pink;
$info: $purple;
```

13. Surveiller les changements CSS

De retour dans le terminal, exécutez npm run bulma-watch et enregistrez tous les changements dans tutorialstyles.scss. Le terminal va détecter les changements et compiler les modifications en CSS. Rafraîchissez index.html dans le navigateur et le changement sera immédiatement visible.

```
npm run bulma-watch
```

14. Personnaliser les composants

Vous avez maintenant vu comme il est facile de faire de grands changements sur le site web avec peu de code. La plupart des composants et des éléments de Bulma sont accompagnés de variables sur leur page de documentation. Modifiez les variables suivantes pour que les boîtes aient plus de padding et une ombre portée plus tendance.

```
$box-padding: 1.5rem;
$box-shadow: 0 5px 10px #582d5733, 0 15px
10px #ff346200;
```

15. Ajouter du CSS non Bulma

L'étendue de Bulma ne sera pas toujours suffisante, et vous serez parfois amené à écrire votre propre code CSS, ce que vous pouvez faire après la dernière ligne @import de votre fichier tutorialstyles.scss.

16. Enregistrer une version de production

En production, les répertoires scss et modules node ne sont pas requis et peuvent donc être omis ou ignorés. Vérifiez que le site de production final inclut index.html, le dossier CSS contenant tutorialstyles.css, et les images ou le JavaScript.