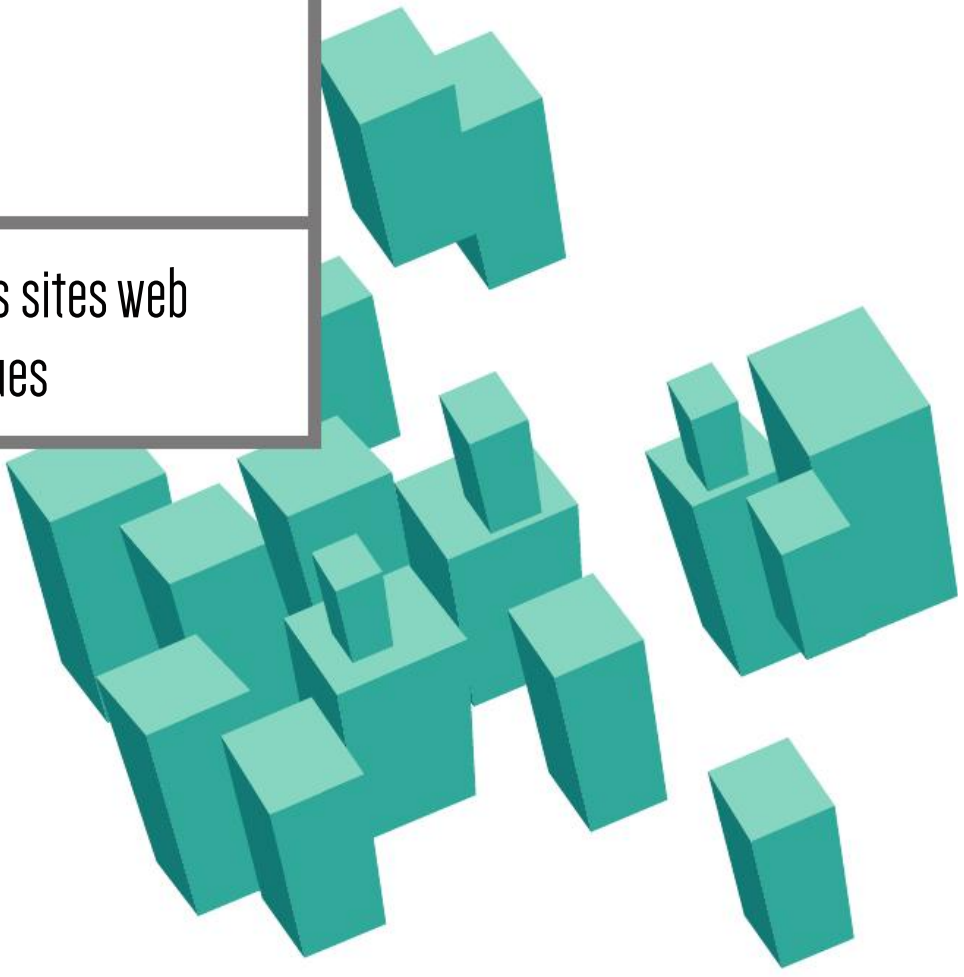
Sébastien Ausin

Tailwind CSS

Pour concevoir des sites web modernes et uniques



Éditions EYROLLES

Tailwind CSS

Un livre incontournable pour concevoir des applications web modernes

Découvrez l'univers de Tailwind CSS, le framework CSS de plus en plus populaire auprès des développeurs pour concevoir des sites ou des applications web modernes. Grâce à sa bibliothèque de classes réutilisables, il évite l'écriture des règles de style pour chaque élément HTML et permet ainsi aux utilisateurs de créer rapidement des mises en page élégantes.

Cet ouvrage vous apprendra les bases pour vous mener jusqu'aux fonctionnalités avancées de Tailwind CSS.

Vous commencerez par voir comment installer Tailwind CSS selon plusieurs méthodes, en fonction de vos projets. Puis, vous vous perfectionnerez sur les techniques de mise en page et apprendrez à agir sur la typographie, les couleurs, les dimensions et les espaces. Vous pourrez également pousser l'utilisation de Tailwind CSS encore plus loin, en le personnalisant selon vos besoins, pour concevoir des sites web uniques correspondant exactement à vos attentes. Enfin, un cas pratique vous permettra de mettre en application ce que vous aurez appris, en réalisant l'intégration d'une maquette d'un site e-commerce étape par étape.

Ce livre vous propose également des ressources supplémentaires et des astuces pratiques pour compléter votre boîte à outils et ainsi gagner en productivité.

Que vous soyez débutant ou développeur expérimenté, cet ouvrage est votre allié pour créer des interfaces modernes et élégantes.

À qui s'adresse cet ouvrage ?

- Aux étudiants en développement web qui souhaitent prendre en main un framework CSS
- Aux développeurs web qui veulent travailler plus rapidement
- Aux entreprises ou organisations qui souhaitent mettre à jour leur site pour le moderniser et ainsi proposer une meilleure expérience utilisateur

Compléments web

Les images et le code source du cas pratique de ce livre sont disponibles sur le site d'accompagnement :
https://www.editions-eyrolles.com/dl/0101421

Au sommaire

D'une formation initiale en informatique de gestion, **Sébastien Ausin** s'est forgé à travers plusieurs expériences professionnelles : e-commerce, webmarketing, développement web, formateur. Désormais, il est également auteur avec ce premier ouvrage consacré à Tailwind CSS aux éditions Eyrolles.

www.editions-eyrolles.com

Tailwind CSS

DANS LA MÊME COLLECTION

C. BLAESS. – **Solutions temps réel sous Linux.**
N° 67711, 3e édition, 2019, 320 pages.

C. PIERRE DE GEYER, J. PAULI, P. MARTIN, E. DASPET. – **PHP 7 avancé.**
N° 67720, 2e édition, 2018, 736 pages.

H.WICKHAM, G. GROLEMUND. – **R pour les data sciences.**
N° 67571, 2018, 496 pages.

F. PROVOST, T. FAWCETT. – **Data science pour l'entreprise.**
N° 67570, 2018, 370 pages.

J. CHOKOGOUE. – **Maîtrisez l'utilisation des technologies Hadoop.**
N° 67478, 2018, 432 pages.

H. BEN REBAH, B. MARIAT. – **API HTML 5 : maîtrisez le Web moderne !**
N° 67554, 2018, 294 pages.

W. MCKINNEY. – **Analyse de données en Python.**
N° 14109, 2015, 488 pages.

E. BIERNAT, M. LUTZ. – **Data science : fondamentaux et études de cas.**
N° 14243, 2015, 312 pages.

SUR LE MÊME THÈME

R. GOETTER. – **CSS 3 Grid Layout.**
N° 67683, 2019, 144 pages.

R. GOETTER. – **CSS 3 Flexbox.**
N° 14363, 2016, 152 pages.

Sébastien Ausin

Tailwind CSS

Pour concevoir des sites web modernes et uniques

2e édition

Éditions EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Depuis 1925, les éditions Eyrolles s'engagent en proposant des livres pour comprendre le monde, transmettre les savoirs et cultiver ses passions !

Pour continuer à accompagner toutes les générations à venir, nous travaillons de manière responsable, dans le respect de l'environnement. Nos imprimeurs sont ainsi choisis avec la plus grande attention, afin que nos ouvrages soient imprimés sur du papier issu de forêts gérées durablement. Nous veillons également à limiter le transport en privilégiant des imprimeurs locaux. Ainsi, 89 % de nos impressions se font en Europe, dont plus de la moitié en France.

© Éditions Eyrolles, 2024, ISBN : 978-2-416-01421-5

Table des matières

1

Introduction

Tailwind CSS est un outil de développement web. C'est un *framework* CSS de plus en plus populaire, comme l'atteste l'évolution des recherches sur Google France à son sujet au cours des dernières années.

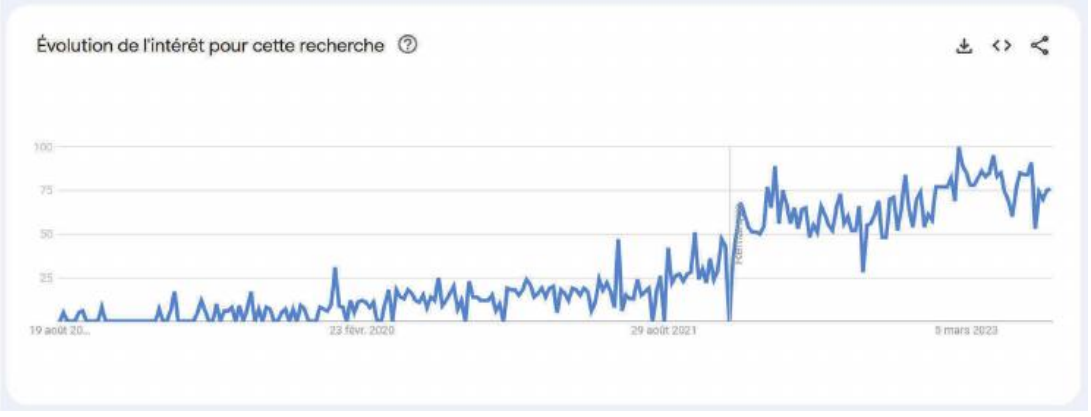


Figure 1–1 Évolution de la recherche sur Tailwind CSS.

Nous avons donc jugé opportun de lui consacrer un ouvrage afin de vous aider à prendre en main ce framework CSS.

Grâce à ce livre, vous apprendrez à installer Tailwind CSS, à le configurer selon vos besoins, et à manipuler ses classes afin de créer des sites modernes et uniques. Un cas pratique vous sera également proposé pour valider vos acquis.

À qui s'adresse cet ouvrage ?

Cet ouvrage s'adresse :

- aux étudiants en développement web qui souhaitent prendre en main un framework CSS ;
- aux formateurs qui enseignent le développement web et qui veulent faire découvrir un framework moderne à leurs élèves ;
- aux développeurs *front-end* qui veulent travailler plus vite ;
- aux entreprises ou organisations qui souhaitent mettre à jour leur site pour le moderniser et ainsi proposer une meilleure expérience utilisateur.

Définition d'un framework

Le terme *framework* est composé de *frame* qui veut dire « cadre » et de *work* qui signifie « travail ». Un framework se traduit ainsi en français par « cadre d'applications » ou encore « cadriciel ». Il s'agit d'un ensemble de composants logiciels réutilisables qui fournissent une structure constituée de bibliothèques de code et de conventions de codage afin de faciliter le développement.

L'utilisation d'un framework permet aux développeurs de gagner du temps en réutilisant du code existant plutôt que de « réinventer la roue ». De plus, les conventions de codage facilitent la maintenance et favorisent le travail en équipe.

Le framework CSS est un ensemble de fichiers dont les règles de styles sont déjà écrites, afin de faciliter et accélérer le processus de conception web. Ces règles de styles sont établies pour les éléments de base d'une page web, tels que les typographies, les couleurs, les mises en page, les boutons, les formulaires et autres éléments couramment utilisés.

En utilisant un framework CSS, les développeurs n'ont donc pas à réécrire entièrement les styles CSS. Au lieu de cela, ils utilisent les styles préconçus pour créer rapidement des sites web qui ont une apparence cohérente et professionnelle. Les frameworks CSS les plus populaires sont Bootstrap, Foundation, Bulma, Materialize et bien entendu Tailwind CSS.

Qu'est-ce que Tailwind CSS ?

Tailwind CSS est un framework CSS *utility-first*. Contrairement à d'autres frameworks comme Bootstrap ou Foundation qui fournissent des composants (bouton, barre de navigation…), Tailwind CSS se base sur des classes utilitaires et ne propose pas de composant.

Cette approche présente de nombreux avantages :

- faire du CSS sans quitter le HTML, ce qui évite les allers-retours entre les fichiers HTML et CSS ;
- créer et personnaliser nos propres composants ;
- développer la personnalisation pour concevoir un site unique.

Tailwind CSS présente également d'autres avantages, comme son système de purge qui génère une feuille de styles avec les classes utilisées dans le projet. Le fichier CSS est plus léger et favorise la vitesse de chargement du site web.

Les débuts de Tailwind CSS

Le framework a été créé par Adam Wathan, un développeur full-stack canadien. La première version 0.1 de Tailwind CSS est apparue au cours de la nuit d'Halloween en 2017. Au départ, Adam Wathan utilisait le framework pour ses projets personnels mais il s'est rendu compte que son outil avait été adopté par d'autres entreprises et individus. Il s'est alors consacré à temps plein au développement de Tailwind CSS en 2019.

Voici l'évolution des différentes versions majeures :

- v0.1 : 1er novembre 2017 (création de Tailwind CSS)
- v1.0 : 13 mai 2019
- v2.0 : 18 novembre 2020
- v3.0 : 9 décembre 2021

Au moment de la rédaction de cet ouvrage, la version en cours est la v3.3.5.

Prérequis pour commencer avec Tailwind CSS

Tailwind CSS étant un framework CSS, il va sans dire que vous devez avoir de bonnes connaissances en HTML et CSS pour l'utiliser. À noter que dans la documentation officielle de Tailwind, le nom des classes est souvent associé à la propriété CSS équivalente.

En ce qui concerne les outils, vous aurez besoin d'un éditeur de code. Je vous recommande le célèbre Visual Studio Code, et vous suggère d'installer quelques extensions pour travailler plus efficacement :

- Live Server, pour mettre en place un serveur local et pouvoir actualiser la page en temps réel ;
- Tailwind CSS IntelliSense, qui facilite l'écriture des noms de classes par autocomplétion.

Pour bien démarrer

L'objectif de cet ouvrage est de vous guider dans votre apprentissage de Tailwind CSS. Nous n'allons pas énumérer toutes les classes du framework, qui sont nombreuses, mais nous étudierons les principales.

En parallèle, je vous invite à consulter régulièrement la documentation officielle, disponible à l'adresse suivante : https://tailwindcss.com/docs/installation. Vous y trouverez de nombreux exemples de code qui vous permettront de vous exercer.

Avant de manipuler les exemples de code des chapitres qui vont suivre, je vous conseille de créer un dossier sur votre ordinateur et de l'ouvrir avec Visual Studio Code. Vous pourrez y ajouter les pages HTML créées au fur et à mesure.

Nous verrons au chapitre suivant comment installer Tailwind CSS simplement (via le CDN). Il vous suffira ensuite de cliquer sur vos fichiers HTML avec le bouton droit de la souris, et de sélectionner *Open with Live Server* dans le menu contextuel. Votre navigateur s'ouvrira alors pour afficher le contenu de la page.

Le site officiel de Tailwind CSS met à disposition une interface permettant de s'entraîner, disponible à l'adresse suivante : https://play.tailwindcss.com. Elle est également utile pour tester des portions de code sans avoir à installer un environnement de développement.

2

Installer Tailwind CSS

Dans ce chapitre, nous allons présenter les deux méthodes principales pour installer Tailwind CSS :

- installation via un CDN, qui constitue une méthode simple et rapide ;
- installation en ligne de commandes, qui est plus complexe mais nécessaire si vous souhaitez personnaliser Tailwind.

Nous aborderons rapidement les autres méthodes, que vous pourrez explorer davantage si besoin.

Installer Tailwind CSS via le CDN

Cette première méthode est la plus simple à mettre en place.

Pour rappel, un CDN *(Content Delivery Network)* est un réseau de distribution de contenu. Il permet le transfert rapide de ressources comme des feuilles de styles, des fichiers JavaScript ou encore des images et des vidéos. Un des avantages à utiliser un CDN est qu'il permet un chargement plus rapide des pages web.

Pour installer Tailwind CSS via le CDN, il suffit d'ajouter la balise `<script>` suivante dans le `<head>` de votre page HTML.

```
<script src="https://cdn.tailwindcss.com"></script>
```

Et c'est tout ! Vous pouvez maintenant utiliser Tailwind CSS.

Notez que cette méthode n'est pas recommandée pour déployer un site en production pour des raisons de performances, mais aussi pour sa limitation à effectuer une configuration personnalisée du projet. On l'utilisera plutôt pour un site en développement, ou encore pour s'entraîner à manipuler Tailwind CSS. C'est d'ailleurs ce que je vous recommande de faire pour les exemples de cet ouvrage.

Comme nous terminerons avec un cas pratique, nous aurons également l'occasion d'installer Tailwind CSS avec la seconde méthode que nous allons aborder maintenant.

Installer Tailwind CSS via la CLI

Rappelons que la CLI *(Command-Line Interface)* est une interface de ligne de commandes.

Nous allons donc ici installer Tailwind CSS en tapant des commandes dans un terminal. Pour ce faire, vous devez au préalable installer l'environnement Node.js, disponible à l'adresse suivante : https://nodejs.org.

Figure 2–1
Télécharger Node.js.

Vous pouvez télécharger la version recommandée pour la plupart des utilisateurs (figure 2-1).

Après l'installation de Node.js, nous pourrons utiliser npm *(Node Package Manager)*, qui nous permettra d'installer des dépendances comme des librairies, des outils…

Ouvrez ensuite un dossier avec Visual Studio Code (ou votre éditeur préféré) afin de commencer un projet.

Lancez le terminal, vous devriez vous situer dans le dossier du projet. Dans le cas contraire, positionnez-vous dans le dossier du projet avant de lancer les commandes suivantes pour installer Tailwind CSS.

```
npm install -D tailwindcss
npx tailwindcss init
```

Après avoir exécuté la première commande, nous constatons qu'un dossier `node_modules` a été ajouté à notre projet. Nous avons également le fichier `package.json` qui confirme la version installée de Tailwind CSS.

La seconde commande va créer le fichier `tailwind.config.js` qui nous permettra de configurer et personnaliser Tailwind CSS.

Ce fichier devrait se présenter comme ceci :

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

La clé `content` contiendra les fichiers qui possèdent les classes utilitaires de Tailwind afin d'être analysés et compilés pour créer la feuille de styles. La clé `theme` servira à personnaliser Tailwind, tandis que la clé `plugins` permettra d'ajouter des extensions. Nous reviendrons sur ces deux dernières clés un peu plus tard. Restons pour le moment sur la clé `content`.

Après avoir créé un dossier `public` et y avoir ajouté notre première page HTML (`index.html`), nous renseignerons la clé `content` de la manière suivante :

```
content: ["./public/*.html"],
```

Nous demandons ainsi à Tailwind CSS d'analyser tous les fichiers, avec l'extension `.html`, qui se trouvent dans le dossier `public`. C'est donc ce dossier qui contiendra tous les fichiers nécessaires à la conception du site web.

À la racine de notre projet, nous allons créer un nouveau dossier `src` qui contiendra le fichier d'entrée CSS, soit `input.css` pour notre exemple.

Nous y ajoutons les trois directives de Tailwind CSS :

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

La directive `@tailwind base` permet d'effectuer un *reset*, une remise à zéro de certaines propriétés.

`@tailwind components` nous autorise à créer des composants afin de les réutiliser.

Enfin, la directive `@tailwind utilities` contient toutes les classes utilitaires.

Tout est désormais prêt pour effectuer une première compilation et générer la feuille de styles.

```
npx tailwindcss -i ./src/input.css -o ./public/style.css
```

Le paramètre `-i` de cette commande permet d'analyser le fichier en entrée. Ensuite, le paramètre `-o` va créer un fichier en sortie qui est le résultat de la compilation.

Maintenant que le fichier `style.css` est créé, nous pouvons l'ajouter à notre page `index.html`.

```
<link rel="stylesheet" href="style.css">
```

Voilà à quoi pourrait ressembler le code de votre page `index.html` :

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Installation de Tailwind CSS via la CLI</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <h1>Installation de Tailwind CSS via la CLI</h1>
</body>
</html>
```

Si vous ouvrez cette page dans votre navigateur, vous constaterez que le titre `<h1>` a désormais une taille de police plus petite. Cela vient du fait que Tailwind a procédé à un *reset* des propriétés, ce qui nous confirme qu'il est bien installé.

J'attire votre attention sur un point important. Imaginons que nous souhaitions ajouter une classe à la balise `<h1>` pour augmenter la taille de la police.

```
<h1 class="text-2xl">Installation de Tailwind CSS via la CLI</h1>
```

Nous enregistrons le fichier, et pourtant nous ne constatons aucun changement dans le navigateur. Il faut alors relancer la commande pour générer la feuille de styles.

Si nous vérifions le fichier `style.css`, Tailwind a bien ajouté la classe suivante :

```
.text-2xl {
   font-size: 1.5rem;
   line-height: 2rem;
}
```

C'est ce qui fait la force du framework Tailwind CSS : il ajoute simplement les classes utilisées dans le projet. De cette manière, la feuille de styles est plus légère et améliore la vitesse de chargement de vos pages web.

Vous pourriez me dire que c'est plutôt contraignant de lancer la commande à chaque fois. Rassurez-vous, il suffit d'ajouter `--watch` à la fin. Ainsi, la commande se lancera de manière automatique dès que vous enregistrerez votre fichier HTML.

```
npx tailwindcss -i ./src/input.css -o ./public/style.css --watch
```

Bravo ! Vous avez effectué votre première installation de Tailwind CSS en ligne de commandes.

Autres méthodes

En fonction de vos besoins, d'autres méthodes sont disponibles. Nous allons les survoler ici, mais je vous invite à consulter la documentation officielle si vous souhaitez en savoir plus.

Installation en tant que plugin PostCSS

Cette installation est sensiblement identique à l'installation via la CLI. Cependant, elle ajoute le plugin autoprefixer qui se chargera d'ajouter des préfixes à la feuille de styles pour une meilleure compatibilité des navigateurs.

En effet, si certaines propriétés CSS sont mal supportées par un navigateur, nous pouvons avoir recours aux préfixes.

Par exemple, si nous regardons du côté de Flexbox, sur le site https://caniuse.com/flexbox, nous constatons que d'anciennes versions de navigateurs peuvent supporter partiellement ou complètement la propriété grâce au préfixe.

Voici les préfixes utilisés en fonction des navigateurs :

- -webkit- (Chrome et nouvelles versions d'Opera) ;
- -moz- (Firefox) ;
- -o- (anciennes versions d'Opera) ;
- -ms- (Internet Explorer et Edge).

Installation en fonction d'un framework

La documentation de Tailwind CSS propose des guides d'installation en fonction du framework avec lequel vous avez l'habitude de travailler.

Voici quelques environnements populaires :

- Laravel
- Symfony

- Next.js
- Angular
- Ruby on Rails
- etc.

Si vous ne trouvez pas votre environnement dans les guides d'installation proposés dans la documentation officielle, nous vous recommandons de procéder à une installation via la CLI ou en tant que plugin PostCSS.

3

Principes de base

Avant d'entrer dans le vif du sujet, nous allons nous intéresser dans ce chapitre au principe de fonctionnement de Tailwind CSS. Nous le comparerons à un framework populaire tel que Bootstrap.

Principe des classes utilitaires

En principe, si vous souhaitez ajouter un style particulier à un élément, vous devez le spécifier en CSS. Dans l'exemple suivant, nous créons une classe `myButton` que nous ajoutons à l'élément `<button>`. Nous pouvons ainsi indiquer que nous souhaitons un bouton avec un fond bleu, le texte en blanc et des bords arrondis.

```
<style>
.myButton {
    background-color: #3b82f6;
    color: #fff;
    font-size: 18px;
    padding: 8px 16px;
    border: none;
    border-radius: 8px;
}
</style>
<button class="myButton">Bouton</button>
```

Avec Tailwind CSS, nous allons styler les éléments en appliquant des classes préexistantes. Nous écrirons ces classes utilitaires directement dans le HTML, c'est le principe même de Tailwind CSS.

Ainsi, pour avoir un bouton identique au précédent, nous écrirons le code suivant :

```
<button class="bg-blue-500 text-white px-4 py-2 text-lg rounded-lg">Bouton</button>
```

Le bouton correspondant à ce style apparaît ainsi :

Figure 3–1
Bouton réalisé avec Tailwind CSS.

Sans trop entrer dans les détails pour le moment, voici ce que nous avons spécifié :

- la couleur de fond avec la classe `bg-blue-500` ;
- la couleur du texte avec la classe `text-white` ;
- des marges internes avec les classes `px-4 py-2` ;
- la taille de la police avec la classe `text-lg` ;
- des bords arrondis avec `rounded-lg`.

Cette approche nous permet de styler les éléments HTML sans écrire de code CSS. Certains pourront dire que le code HTML risque d'être surchargé et illisible. Mais le développeur de Tailwind a pensé à tout ! En effet, il est possible de créer des composants, dont l'emploi permet de construire des éléments d'interface utilisateur pour les réutiliser facilement dans le projet. Cette façon de faire permet d'avoir une meilleure organisation, production et maintenance du code. Nous verrons au chapitre 10 comment créer une classe personnalisée afin d'y placer les classes utilitaires de Tailwind CSS.

Différences avec Bootstrap

Bootstrap est l'un des frameworks CSS les plus utilisés pour la création de sites web. Bien qu'il présente certaines similitudes avec Tailwind CSS, il diffère néanmoins sur quelques points.

Bootstrap offre une gamme complète de composants, tels que des boutons, barres de navigation, accordéons… Les développeurs l'apprécient pour sa facilité d'utilisation et sa grande communauté d'utilisateurs.

Tailwind CSS, quant à lui, est un framework CSS qui se concentre sur l'utilisation de classes utilitaires pour créer des styles personnalisés. Au lieu de fournir des classes CSS prédéfinies, il met à disposition des classes utilitaires pour chaque propriété CSS (couleur, taille de la police, hauteur, largeur, marges…). Les développeurs l'apprécient pour sa flexibilité et pour le degré de personnalisation qu'il propose.

En effet, Tailwind CSS permet de concevoir des sites web uniques, alors que les sites réalisés avec Bootstrap ont tendance à se ressembler.

Pour reprendre l'exemple du bouton, nous aurions un code comme celui-ci avec Bootstrap :

```
<button class="btn btn-primary">Bouton</button>
```

La classe `btn` indique que nous souhaitons avoir un composant `Bouton`, et la classe `btn-primary` permet de spécifier sa couleur. Comme vous pouvez le constater, nous allons vite être limités en termes de personnalisation, à moins d'ajouter encore du code CSS personnalisé, ce qui alourdirait la feuille de styles déjà bien chargée de Bootstrap.

Avec Tailwind, nous pouvons reprendre la logique de Bootstrap et créer un composant `Bouton`. Nous verrons plus loin comment personnaliser des classes (voir chapitre 10, page 80).

De la même manière, nous pouvons créer une classe `btn` et lui appliquer les classes utilitaires suivantes : `px-4 py-2 text-lg rounded-lg`. Nous pouvons également créer une classe couleur, `btn-blue`, qui possédera les classes `bg-blue-500 text-white`.

Ainsi, nos boutons pourront s'écrire comme ceci avec Tailwind CSS :

```
<button class="btn btn-blue">Bouton</button>
```

Comme évoqué précédemment, cette approche permet d'alléger le code HTML en classes utilitaires. Il sera ainsi plus lisible et facilement maintenable.

4

Typographie

Dans une page web, le contenu le plus important est bien évidemment le texte. Nous verrons dans ce chapitre comment modifier les propriétés concernant la typographie grâce à Tailwind CSS.

Taille de la police

Pour changer la taille de la police, nous utiliserons la classe utilitaire `text-{taille}`. La taille pourra prendre les valeurs suivantes : `xs`, `sm`, `base`, `lg`, `xl` et de `2` à `9xl`.

Par défaut, la taille de la police possède la classe `text-base`.

Le tableau 4-1 récapitule les classes utilisées par Tailwind ainsi que les propriétés CSS équivalentes.

Tableau 4–1 Taille de la police.

Classes	Propriétés (taille)	Propriétés (hauteur de ligne)
`text-xs`	`font-size: 0.75rem; /* 12px */`	`line-height: 1rem; /* 16px */`
`text-sm`	`font-size: 0.875rem; /* 14px */`	`line-height: 1.25rem; /* 20px */`
`text-base`	`font-size: 1rem; /* 16px */`	`line-height: 1.5rem; /* 24px */`
`text-lg`	`font-size: 1.125rem; /* 18px */`	`line-height: 1.75rem; /* 28px */`
`text-xl`	`font-size: 1.25rem; /* 20px */`	`line-height: 1.75rem; /* 28px */`
`text-2xl`	`font-size: 1.5rem; /* 24px */`	`line-height: 2rem; /* 32px */`
`text-3xl`	`font-size: 1.875rem; /* 30px */`	`line-height: 2.25rem; /* 36px */`
`text-4xl`	`font-size: 2.25rem; /* 36px */`	`line-height: 2.5rem; /* 40px */`
`text-5xl`	`font-size: 3rem; /* 48px */`	`line-height: 1;`

Tableau 4–1 Taille de la police. *(suite)*

Classes	Propriétés (taille)	Propriétés (hauteur de ligne)
text-6xl	font-size: 3.75rem; /* 60px */	line-height: 1;
text-7xl	font-size: 4.5rem; /* 72px */	line-height: 1;
text-8xl	font-size: 6rem; /* 96px */	line-height: 1;
text-9xl	font-size: 8rem; /* 128px */	line-height: 1;

Voici un exemple avec trois paragraphes de tailles différentes :

```
<p class="text-sm">Taille SM</p>
<p class="text-lg">Taille LG</p>
<p class="text-xl">Taille XL</p>
```

Notez que, d'après le tableau 4-1, nous pourrions avoir besoin d'une taille de police de `22px` (entre `xl` et `2xl`) qui n'existe pas…

Nous pourrons alors avoir recours à des valeurs arbitraires qui consistent à ajouter une valeur entre crochets comme dans l'exemple suivant :

```
<p class="text-[22px]">Taille de la police de 22 pixels</p>
```

Nous pouvons utiliser comme valeur arbitraire une autre unité de mesure comme le `rem`, le `em`, etc.

Par ailleurs, ces classes modifient la taille de la police avec la propriété `font-size`, mais également la hauteur de ligne avec la propriété `line-height`.

En fonction de la taille de la police, si la valeur par défaut de l'interlignage ne vous convient pas, vous pouvez ajouter une classe `leading-{valeur}` pour modifier la hauteur de ligne.

Tableau 4–2 Hauteur de ligne.

Classes	Propriétés CSS
leading-3	line-height: .75rem; /* 12px */
leading-4	line-height: 1rem; /* 16px */
leading-5	line-height: 1.25rem; /* 20px */
leading-6	line-height: 1.5rem; /* 24px */
leading-7	line-height: 1.75rem; /* 28px */
leading-8	line-height: 2rem; /* 32px */
leading-9	line-height: 2.25rem; /* 36px */
leading-10	line-height: 2.5rem; /* 40px */
leading-none	line-height: 1;
leading-tight	line-height: 1.25;
leading-snug	line-height: 1.375;
leading-normal	line-height: 1.5;
leading-relaxed	line-height: 1.625;
leading-loose	line-height: 2;

Utilisez les classes `leading-none`, `leading-tight`, `leading-snug`, `leading-normal`, `leading-relaxed` et `leading-loose` pour spécifier une hauteur de ligne proportionnelle à la taille de police actuelle.

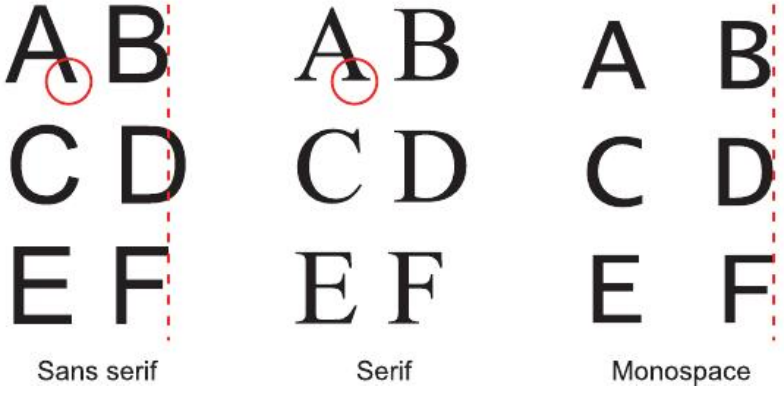
Vous pouvez également utiliser les classes de `leading-3` à `leading-10` pour ajouter une hauteur de ligne fixe.

Familles de polices

Les polices de caractères sont des éléments importants de la mise en page d'un document ou d'un site web. Il existe trois principales familles de polices, à savoir Sans serif, Serif et Monospace.

- Les polices Sans serif ont une apparence nette et moderne, car elles ne possèdent pas d'empattement à l'extrémité de chaque lettre. Elles sont souvent utilisées pour des textes conséquents afin d'éviter une fatigue visuelle. Parmi ces polices, les plus connues sont Arial, Helvetica et Verdana.
- Les polices Serif, inversement, présentent des empattements aux extrémités des lettres. Elles ont une apparence plus classique, élégante, traditionnelle, et sont généralement utilisées pour les titres. Les plus populaires sont Times New Roman, Georgia et Baskerville.
- Les polices Monospace, quant à elles, ont des caractères de largeur égale, ce qui signifie que chaque lettre occupe la même quantité d'espace horizontal. Elles sont souvent utilisées pour la programmation et l'écriture de code. Parmi ces polices, citons Courrier, Consolas et Lucida Console.

Figure 4–1
Les familles de polices.



En général, les polices Sans serif sont adaptées pour une meilleure lisibilité à l'écran. Les polices Serif, quant à elles, peuvent être utilisées avec des tailles plus élevées, par exemple dans des titres.

Il me semblait intéressant de faire ce petit rappel sur les familles de polices, car Tailwind CSS dispose des trois classes utilitaires suivantes :

```
<p class="font-sans">Sans serif</p>
<p class="font-serif">Serif</p>
<p class="font-mono">Monospace</p>
```

À noter qu'il est aussi possible d'ajouter des polices personnalisées que vous pourriez trouver sur Google Fonts (https://fonts.google.com). Dans ce cas, nous utiliserons les valeurs abstraites en ajoutant le nom de la police entre crochets.

```
<p class="font-['Arvo']">Arvo</p>
```

Dans cet exemple, nous avons ajouté la police `Arvo`. Au nom de la classe `font`, nous spécifions le nom de la police. Attention toutefois, les noms de polices composés de plusieurs mots se verront attribuer un nom de classe avec un underscore (_) comme séparateur au lieu d'un espace.

Voici un exemple avec la police `Josefin Sans` :

```
<p class="font-['Josefin_Sans']">Josefin Sans</p>
```

Style

Un texte peut être mis en valeur par un soulignement, des caractères gras ou encore italiques. Ces différents styles sont également pris en charge par Tailwind CSS.

Italique

Comme son nom l'indique, la classe `italic` va mettre le texte en italique avec la propriété CSS `font-style: italic;`. La classe `not-italic` mettra le texte en style normal avec la propriété `font-style: normal;`.

Dans l'exemple qui suit, le premier paragraphe est mis en italique. Dans le second paragraphe, un élément `<em>` a été ajouté. Cette balise met le texte en italique, qui a été remis en normal avec la classe `not-italic`.

```
<p class="italic">Un texte en italique</p>
<p>Un paragraphe avec un <em class="not-italic">texte normal</em>.</p>
```

Le résultat de ce code s'affiche ainsi :

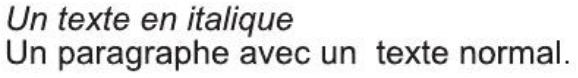


Figure 4–2
Utilisation des classes « italic » et « not-italic ».

Graisse de la police

En typographie, la graisse est l'épaisseur d'un caractère. Les titres apparaissent généralement en caractères gras, mais il est aussi possible d'augmenter la graisse d'un mot ou groupe de mots dans un texte afin d'insister sur son importance.

La propriété CSS qui gère la graisse de la police est `font-weight`, qui peut prendre une valeur allant de 100 à 900, par pas de cent.

Pour rappel, `font-weight: 400` équivaut à `font-weight: normal` et `font-weight: 700` est identique à `font-weight: bold`.

À noter que toutes les polices ne possèdent pas des variations de graisse, donc veillez à en choisir une qui le permet si vous avez besoin de mettre des caractères en gras.

Le tableau 4-3 récapitule les différentes classes de Tailwind CSS pour modifier le style des caractères.

Tableau 4–3 Graisse de la police.

Classes	Propriétés CSS
font-thin	font-weight: 100;
font-extralight	font-weight: 200;
font-light	font-weight: 300;
font-normal	font-weight: 400;
font-medium	font-weight: 500;
font-semibold	font-weight: 600;
font-bold	font-weight: 700;
font-extrabold	font-weight: 800;
font-black	font-weight: 900;

Voici un exemple de syntaxe :

```
<p class="font-thin">Font Thin</p>
<p class="font-semibold">Font Semibold</p>
<p class="font-black">Font Black</p>
```

Ainsi que son rendu dans le navigateur :

Figure 4–3
Affichage des différentes graisses de police.

Soulignement

En CSS, nous avons la possibilité de « décorer » le texte avec une ligne en dessous (soulignement), au-dessus ou en travers (texte barré). Nous utilisons alors la propriété `text-decoration-line` dont voici les classes équivalentes de Tailwind CSS.

Tableau 4–4 Soulignement de texte.

Classes	Propriétés CSS
underline	text-decoration-line: underline;
overline	text-decoration-line: overline;
line-through	text-decoration-line: line-through;
no-underline	text-decoration-line: none;

Voici un exemple pour illustrer ce style :

```
<p class="underline">Texte souligné</p>
<p class="overline">Ligne au-dessus</p>
<p class="line-through">Texte barré</p>
```

Ainsi que son rendu dans le navigateur :

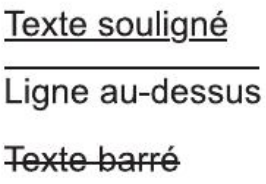


Figure 4–4
Aperçu des types de soulignement du texte.

Casse du texte

Tailwind CSS propose des classes pour transformer la casse d'un texte et ainsi le mettre en majuscule, en minuscule ou encore mettre uniquement la première lettre d'un mot en majuscule.

Tableau 4–5 Casse du texte.

Classes	Propriétés CSS
uppercase	text-transform: uppercase;
lowercase	text-transform: lowercase;
capitalize	text-transform: capitalize;
normal-case	text-transform: none;

Voici un exemple pour illustrer cela :

```
<p class="uppercase">Texte en majuscule</p>
<p class="lowercase">Texte en minuscule</p>
<p class="capitalize">La première lettre du mot en majuscule</p>
```

Ainsi que son rendu dans le navigateur :

Figure 4–5
Affichage des différentes casses du texte.

Couleurs

Tailwind CSS dispose d'une large palette de couleurs composée de 22 couleurs de base comportant 10 nuances chacune, soit 220 couleurs au total (sans compter le blanc et le noir).

Il est également possible de spécifier des couleurs personnalisées, comme nous le verrons au chapitre 10 consacré à la personnalisation de Tailwind CSS.

Pour mettre un texte en couleur, le nom de la classe utilisera la syntaxe `text-{couleur}-{nuance}`.

Voici la liste des 22 couleurs de la palette :

- slate
- gray
- zinc
- neutral
- stone
- red
- orange
- amber
- yellow
- lime
- green
- emerald
- teal
- cyan
- sky
- blue
- indigo
- violet
- purple
- fuchsia
- pink
- rose

La nuance peut prendre les valeurs `50`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800` et `900`. La valeur `50` donnera une couleur très claire, tandis qu'avec la valeur `900`, nous obtiendrons une couleur très foncée. Avec une nuance de `500`, nous obtiendrons une couleur vive.

```
<p class="text-orange-500">Couleur orange</p>
<p class="text-blue-900">Couleur bleu foncé</p>
<p class="text-cyan-100">Couleur cyan clair</p>
```

Le tableau 4-6 présente les classes utilisées par Tailwind CSS pour spécifier la couleur.

Tableau 4–6 Couleurs du texte.

Classes	Propriétés CSS
text-inherit	color: inherit;
text-current	color: currentColor;
text-transparent	color: transparent;
text-black	color: rgb(0 0 0);
text-white	color: rgb(255 255 255);
text-slate-50	color: rgb(248 250 252);
text-slate-100	color: rgb(241 245 249);
text-slate-200	color: rgb(226 232 240);
text-slate-300	color: rgb(203 213 225);
text-slate-400	color: rgb(148 163 184);
text-slate-500	color: rgb(100 116 139);
text-slate-600	color: rgb(71 85 105);
text-slate-700	color: rgb(51 65 85);
text-slate-800	color: rgb(30 41 59);
text-slate-900	color: rgb(15 23 42);
…	…

Comme son nom l'indique, la classe `text-transparent` ajoute du texte transparent. Les classes `text-black` et `text-white` n'ont pas de nuance puisqu'il s'agit d'un noir et d'un blanc purs.

Notez que vous pouvez ajouter une valeur arbitraire pour définir une couleur personnalisée :

```
<p class="text-[#FA8072]">Couleur saumon</p>
```

Il est également possible de modifier l'opacité du texte en ajoutant une fraction à la valeur de la nuance.

```
<p class="text-blue-600/100">Opacité à 100 %</p>
<p class="text-blue-600/75">Opacité à 75 %</p>
<p class="text-blue-600/50">Opacité à 50 %</p>
<p class="text-blue-600/25">Opacité à 25 %</p>
<p class="text-blue-600/0">Opacité à 0 %</p>
```

Si une valeur ne vous convient pas, vous pouvez aussi utiliser une valeur arbitraire pour spécifier une opacité de 60 %, par exemple :

```
<p class="text-blue-600/[0.6]">Opacité à 60 %</p>
```

Alignement du texte

Tailwind CSS propose différentes classes pour spécifier l'alignement du texte, horizontal ou vertical.

Alignement horizontal

Le flux du texte peut être aligné à gauche ou à droite, centré et justifié.

Le tableau 4-7 présente les classes permettant de contrôler l'alignement horizontal du texte.

Tableau 4–7 Alignement horizontal du texte.

Classes	Propriétés CSS
`text-left`	`text-align: left;`
`text-center`	`text-align: center;`
`text-right`	`text-align: right;`
`text-justify`	`text-align: justify;`
`text-start`	`text-align: start;`
`text-end`	`text-align: end;`

Notez que `text-start` est l'équivalent de `text-left` si la direction va de gauche à droite. En revanche, si la direction va de droite à gauche, la classe sera égale à `text-right`. Enfin, `text-end` est l'équivalent de `text-right` si la direction va de gauche à droite et inversement.

Alignement vertical

Nous pouvons gérer l'alignement vertical des éléments `inline`, `inline-block` et `table-cell`.

Tableau 4–8 Alignement vertical du texte.

Classes	Propriétés CSS
`align-baseline`	`vertical-align: baseline;`
`align-top`	`vertical-align: top;`
`align-middle`	`vertical-align: middle;`
`align-bottom`	`vertical-align: bottom;`
`align-text-top`	`vertical-align: text-top;`
`align-text-bottom`	`vertical-align: text-bottom;`
`align-sub`	`vertical-align: sub;`
`align-super`	`vertical-align: super;`

Par exemple, il arrive souvent que l'on ait besoin d'aligner une image par rapport au texte. Les classes présentées dans le tableau 4-8 viendront ajuster cet alignement vertical.

Listes

Tailwind CSS propose différentes classes pour gérer les listes non ordonnées (à puces) et les listes ordonnées (numérotées).

Tableau 4–9 Types de listes.

Classes	Propriétés CSS
`list-none`	`list-style-type: none;`
`list-disc`	`list-style-type: disc;`
`list-decimal`	`list-style-type: decimal;`

Voici un exemple d'utilisation de ces classes :

```
<p>Liste à puces</p>
<ul class="list-disc">
    <li>un élément</li>
    <li>un autre élément</li>
</ul>
<p>Liste ordonnée</p>
<ol class="list-decimal">
    <li>un premier élément</li>
    <li>un second élément</li>
</ol>
```

Tailwind CSS permet de positionner les puces ou les chiffres à l'intérieur ou à l'extérieur de l'élément de liste. Par défaut, les puces se trouvent à l'extérieur (classe `list-outside`). Il faudra ajouter la classe `list-inside` pour les positionner à l'intérieur.

5

Modèle de boîte

Dans ce chapitre, nous aborderons la gestion des dimensions, marges, bordures et couleurs de fond d'un élément HTML dans Tailwind CSS. Nous verrons également qu'il est possible d'ajouter des filtres et des effets sur les éléments.

Dimensions

Chaque élément HTML occupe un espace rectangulaire dans le navigateur. Il est possible d'afficher un aperçu du modèle de boîte de l'élément HTML sélectionné à l'aide des outils de développement web du navigateur.

La figure 5-1 montre une représentation d'un modèle de boîte.

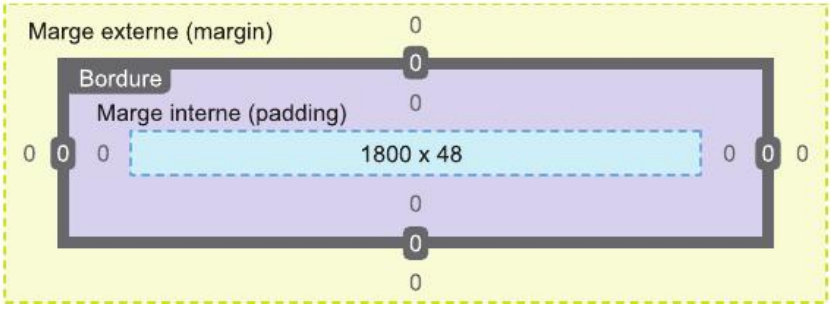


Figure 5–1
Représentation du modèle de boîte.

Tailwind CSS fournit de nombreuses classes utilitaires pour gérer la largeur et la hauteur des éléments.

Largeur

Le nom de la classe utilitaire pour définir une largeur se présente sous la forme `w-{valeur}`.

Tableau 5–1 Largeurs en unités de mesure.

Classes	Propriétés CSS
w-0	width: 0px;
w-px	width: 1px;
w-auto	width: auto;
w-screen	width: 100vw;
w-min	width: min-content;
w-max	width: max-content;
w-fit	width: fit-content;
w-0.5	width: 0.125rem; /* 2px */
w-1	width: 0.25rem; /* 4px */
w-1.5	width: 0.375rem; /* 6px */
w-2	width: 0.5rem; /* 8px */
w-2.5	width: 0.625rem; /* 10px */
w-3	width: 0.75rem; /* 12px */
w-3.5	width: 0.875rem; /* 14px */
w-4	width: 1rem; /* 16px */
…	…
w-72	width: 18rem; /* 288px */
w-80	width: 20rem; /* 320px */
w-96	width: 24rem; /* 384px */

Mis à part les classes spéciales (`w-screen`, `w-auto`…), Tailwind CSS définit 34 valeurs numériques : 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 72, 80, 96.

Comme on peut le constater dans le tableau 5-1, la largeur peut prendre une valeur absolue qui s'exprime en rem. Par ailleurs, nous pouvons ajouter une valeur de 0 à 96 au préfixe `w`.

Il est possible de multiplier une valeur par 0,25 rem pour obtenir la dimension. Par exemple, `w-72` a pour dimension : 72 × 0,25 = 18 rem. Si vous préférez utiliser le pixel comme unité, il faudra multiplier par 4 : 72 × 4 = 288 pixels.

Notez que si vous souhaitez une largeur de 1 pixel, il conviendra d'utiliser la classe `w-px` comme indiqué dans le tableau 5-1.

Pour un affichage *responsive design*, nous favoriserons des largeurs avec une unité relative. Tailwind CSS propose des classes sous forme de fraction pour définir une largeur en pourcentage (voir tableau 5-2).

Tableau 5–2 Largeurs en pourcentage.

Classes	Propriétés CSS
w-1/2	width: 50%;
w-1/3	width: 33.333333%;
w-2/3	width: 66.666667%;
w-1/4	width: 25%;
w-2/4	width: 50%;
w-3/4	width: 75%;
…	…
w-full	width: 100%;

Si vous ne trouvez pas votre bonheur dans toutes ces classes pour ajuster une largeur à vos besoins, vous pouvez utiliser les valeurs arbitraires.

```
<p class="w-[100px]">Largeur de 100 pixels.</p>
```

Le tableau 5-3 présente les classes utilitaires permettant de définir la largeur minimale d'un élément.

Tableau 5–3 Largeur minimale.

Classes	Propriétés CSS
min-w-0	min-width: 0px;
min-w-full	min-width: 100%;
min-w-min	min-width: min-content;
min-w-max	min-width: max-content;
min-w-fit	min-width: fit-content;

Le tableau 5-4 présente, quant à lui, les classes permettant de définir la largeur maximale d'un élément.

Tableau 5–4 Largeur maximale.

Classes	Propriétés CSS
max-w-0	max-width: 0rem; /* 0px */
max-w-none	max-width: none;
max-w-xs	max-width: 20rem; /* 320px */
max-w-sm	max-width: 24rem; /* 384px */
max-w-md	max-width: 28rem; /* 448px */
max-w-lg	max-width: 32rem; /* 512px */
max-w-xl	max-width: 36rem; /* 576px */
max-w-2xl	max-width: 42rem; /* 672px */
…	…
max-w-7xl	max-width: 80rem; /* 1280px */

Tableau 5–4 Largeur maximale. *(suite)*

Classes	Propriétés CSS
max-w-full	max-width: 100%;
max-w-min	max-width: min-content;
max-w-max	max-width: max-content;
max-w-fit	max-width: fit-content;
max-w-screen-sm	max-width: 640px;
max-w-screen-md	max-width: 768px;
max-w-screen-lg	max-width: 1024px;
max-w-screen-xl	max-width: 1280px;
max-w-screen-2xl	max-width: 1536px;

Ces classes sont utiles pour fixer des largeurs maximales aux conteneurs dans la mise en page.

Hauteur

En ce qui concerne la hauteur d'un élément, les classes utilitaires sont sensiblement les mêmes que celles utilisées pour spécifier la largeur à la différence que le préfixe est ici `h`, auquel nous pourrons ajouter une valeur.

Nous pouvons exprimer la hauteur en valeur absolue (`h-1`, `h-2`, …, `h-96`) ou en pourcentage à l'aide de fractions (`h-1/2`, `h-1/4`…).

Espacements

Tailwind CSS nous propose de nombreuses classes pour gérer les espacements des éléments. Nous pourrons agir sur les marges internes et externes, mais également sur la répartition des éléments entre eux.

Marges internes et externes

Les marges internes *(padding)* et externes *(margin)* sont des propriétés importantes en CSS pour spécifier l'espace entre les éléments d'une page web. Les classes utilitaires de Tailwind CSS permettent de définir ces propriétés facilement et rapidement.

La syntaxe pour gérer les marges internes ou externes est la suivante : `{propriété}{côté}-{taille}`.

La propriété peut prendre les valeurs suivantes :

- `m` : pour la propriété `margin` ;
- `p` : pour la propriété `padding`.

À cette propriété, nous pouvons ajouter un côté :

- `t` : pour définir un `padding-top` ou un `margin-top` ;
- `b` : pour définir un `padding-bottom` ou un `margin-bottom` ;
- `l` : pour définir un `padding-left` ou un `margin-left` ;
- `r` : pour définir un `padding-right` ou un `margin-right` ;
- `x` : pour définir une marge à gauche et à droite ;
- `y` : pour définir une marge en haut et en bas.

Notez que le côté est facultatif si l'on souhaite gérer les quatre directions simultanément.

Enfin, la valeur de la taille peut être un nombre de 0 à 96, parmi les 34 valeurs que nous avons présentées précédemment pour les dimensions.

Le tableau 5-5 présente quelques classes supplémentaires pour annuler une marge, la définir à 1 pixel ou encore fixer des marges externes avec une valeur `auto`.

Tableau 5–5 Autres classes disponibles pour les marges internes et externes.

Classes	Propriétés CSS
p-0	padding: 0px;
p-px	padding: 1px;
px-px	padding-left: 1px; padding-right: 1px;
py-px	padding-top: 1px; padding-bottom: 1px;
pt-px	padding-top: 1px;
pr-px	padding-right: 1px;
pb-px	padding-bottom: 1px;
pl-px	padding-left: 1px;
m-0	margin: 0px;
m-px	margin: 1px;
mx-px	margin-left: 1px; margin-right: 1px;
my-px	margin-top: 1px; margin-bottom: 1px;
mt-px	margin-top: 1px;
mr-px	margin-right: 1px;
mb-px	margin-bottom: 1px;
ml-px	margin-left: 1px;
m-auto	margin: auto;
mx-auto	margin-left: auto; margin-right: auto;
my-auto	margin-top: auto; margin-bottom: auto;
mt-auto	margin-top: auto;
mr-auto	margin-right: auto;
mb-auto	margin-bottom: auto;
ml-auto	margin-left: auto;

Voici quelques exemples d'utilisation :

```
<p class="pt-4">Un padding vers le haut de 1rem.</p>
<p class="p-6">Un padding tout autour de 1.5rem.</p>
<p class="px-8">Un padding à gauche et à droite de 2rem.</p>
<p class="mb-6">Un margin en bas de 1.5rem.</p>
<p class="mx-auto">Un margin auto à gauche et à droite.</p>
```

Si besoin, vous pouvez également utiliser les valeurs arbitraires :

```
<p class="p-[5px]">Un padding de 5 pixels tout autour.</p>
```

Espaces entre éléments

Tailwind CSS permet d'ajouter des espaces entre les éléments enfants. Cette répartition d'espaces peut se faire horizontalement ou verticalement.

La classe utilitaire s'écrit de la manière suivante : `space-{x|y}-{valeur}`. Elle doit être ajoutée au niveau de l'élément parent.

- `x` : permet une répartition horizontale ;
- `y` : permet une répartition verticale ;
- `valeur` : est un nombre compris entre 0 et 96, parmi les 34 valeurs numériques de Tailwind CSS.

Voici un exemple avec la réalisation d'une barre de navigation :

```
<nav class="space-x-4">
    <a href="#">Lien 1</a>
    <a href="#">Lien 2</a>
    <a href="#">Lien 3</a>
</nav>
```

Bordures

Les bordures sont importantes pour délimiter les éléments et améliorer la lisibilité. Nous allons voir ici comment créer des bordures et comment les personnaliser avec Tailwind CSS afin de répondre aux besoins de vos projets.

Épaisseur du trait

Pour ajouter des bordures, la classe utilitaire se présente sous la forme suivante : `border-{côté}-{taille}`.

La classe `border` seule permet d'ajouter une bordure tout autour de l'élément avec une épaisseur de 1 pixel.

L'attribut `côté` est facultatif. Il peut prendre les valeurs suivantes :

- `t` : pour *top*, pour définir une bordure en haut ;
- `b` : pour *bottom*, pour définir une bordure en bas ;
- `l` : pour *left*, pour définir une bordure à gauche ;
- `r` : pour *right*, pour définir une bordure à droite ;
- `x` : pour définir une bordure à gauche et à droite ;
- `y` : pour définir une bordure en haut et en bas.

En ce qui concerne les tailles, nous pourrons définir les valeurs 0, 2, 4 et 8. Attention, l'unité de mesure ici est le pixel et non le rem.

Tableau 5–6 Bordure et épaisseur du trait.

Classes	Propriétés CSS
`border`	`border-width: 1px;`
`border-0`	`border-width: 0px;`
`border-2`	`border-width: 2px;`
`border-4`	`border-width: 4px;`
`border-8`	`border-width: 8px;`

Voici quelques exemples de bordures :

```
<p class="border-x-4">Une bordure à gauche et à droite.</p>
<p class="border-y-2">Une bordure en haut et en bas.</p>
```

Dans l'exemple qui suit, nous souhaitons ajouter une bordure tout autour de l'élément sauf en haut. L'épaisseur du trait est fixée à 4 pixels.

```
<p class="border-l-4 border-r-4 border-b-4">Une bordure tout autour sauf en
haut.</p>
```

L'élément a maintenant une bordure sur ces trois côtés. Mais peut-être est-il plus simple d'ajouter une bordure tout autour et de supprimer ensuite celle du haut ? Le code suivant présente comment faire ceci :

```
<p class="border-4 border-t-0">Une bordure tout autour sauf en haut.</p>
```

Les valeurs fixées par Tailwind CSS se limitent à 8 pixels pour l'épaisseur du trait, mais nous pouvons toujours ajouter une valeur arbitraire.

```
<p class="border-[10px]">Un trait de 10 pixels d'épaisseur.</p>
```

Couleur du trait

Par défaut, le trait de la bordure est gris clair. Il est possible de préciser une autre couleur via la syntaxe suivante : `border-{couleur}`.

L'exemple suivant applique une couleur verte sur un trait d'une épaisseur de 2 pixels.

```
<p class="border-2 border-green-500">Un contour vert.</p>
```

Et comme tout est possible avec Tailwind CSS, spécifions une couleur différente pour chaque côté :

```
<p class="border-2 border-t-red-500 border-b-lime-500 border-l-blue-500
border-r-amber-600">Une couleur différente de chaque côté.</p>
```

Comme pour la couleur attribuée au texte (voir chapitre 4, page 22), nous pouvons appliquer un pourcentage d'opacité à la couleur du trait à l'aide d'une fraction.

```
<p class="border-4 border-indigo-500/50">Un trait d'une opacité de 50 %.</p>
```

Style du trait

Par défaut, le trait de la bordure est continu, mais il est possible de modifier cela. En effet, Tailwind CSS propose des classes qui permettent d'obtenir un double trait, des tirets ou des pointillés.

Notez que les bordures peuvent également être désactivées.

Tableau 5–7 Styles de bordure.

Classes	Propriétés CSS
border-solid	border-style: solid;
border-dashed	border-style: dashed;
border-dotted	border-style: dotted;
border-double	border-style: double;
border-hidden	border-style: hidden;
border-none	border-style: none;

Coins arrondis

Les coins des éléments peuvent être arrondis grâce à neuf classes standards.

Tableau 5–8 Coins arrondis.

Classes	Propriétés CSS	Taille
`rounded-none`	`border-radius: 0px;`	0px
`rounded-sm`	`border-radius: 0.125rem;`	2px
`rounded`	`border-radius: 0.25rem;`	4px
`rounded-md`	`border-radius: 0.375rem;`	6px
`rounded-lg`	`border-radius: 0.5rem;`	8px
`rounded-xl`	`border-radius: 0.75rem;`	12px
`rounded-2xl`	`border-radius: 1rem;`	16px
`rounded-3xl`	`border-radius: 1.5rem;`	24px
`rounded-full`	`border-radius: 9999px;`	9999px

Il est possible de spécifier des arrondis sur un côté seulement : en haut, en bas, à droite ou à gauche. La classe s'écrit sous la forme suivante : `rounded-{côté}-{taille}`.

L'attribut `côté` peut prendre les valeurs suivantes :

- `t` : pour *top*, en haut ;
- `b` : pour *bottom*, en bas ;
- `r` : pour *right*, à droite ;
- `l` : pour *left*, à gauche.

La taille prend, quant à elle, l'une des valeurs présentées au tableau 5-8 : `sm`, `md`, `lg`…

Voici des exemples de classes avec leur illustration.

Figure 5–2
Coins arrondis selon un côté.

Nous pouvons spécifier un arrondi sur un coin précis :

- `tl` : pour *top left*, en haut à gauche ;
- `tr` : pour *top right*, en haut à droite ;
- `br` : pour *bottom right*, en bas à droite ;
- `bl` : pour *bottom left*, en bas à gauche.

Figure 5–3
Coin arrondi sur un angle précis.

Séparateurs

Il est possible de placer un séparateur entre les éléments HTML d'une page web. Pour cela, il convient d'ajouter la classe à l'élément parent à l'aide de la syntaxe suivante : `divide-{axe}-{taille}`.

L'attribut `axe` prendra `x` ou `y` comme valeur en fonction de la disposition des éléments en ligne ou en colonne. Quant à l'attribut `taille`, sa valeur est facultative (trait de 1 pixel par défaut) et peut être 0, 2, 4 et 8 pixels.

Voici un exemple avec des éléments sur une ligne, séparés par une ligne verticale d'une épaisseur de 4 pixels :

```
<p class="divide-x-4">
    <span>Un</span>
    <span>Deux</span>
    <span>Trois</span>
</p>
```

Et voici un autre exemple avec deux paragraphes qui sont séparés par une ligne horizontale d'une épaisseur de 2 pixels.

```
<div class="divide-y-2">
    <p>Un paragraphe</p>
    <p>Un autre paragraphe</p>
</div>
```

Pour changer la couleur du séparateur (gris clair par défaut), il faut ajouter la classe `divide-{couleur}`. Et comme pour les bordures, il est possible de modifier le style de trait du séparateur.

Tableau 5–9 Styles de séparateurs.

Classes	Propriétés CSS	Style
divide-solid	border-style: solid;	trait continu (par défaut)
divide-dashed	border-style: dashed;	trait avec tirets
divide-dotted	border-style: dotted;	trait avec pointillés
divide-double	border-style: double;	double trait
divide-none	border-style: none;	suppression du trait

Bordure en anneau ou « ring »

La classe `ring` ajoute une bordure qui utilise la propriété CSS `box-shadow` au lieu de `border`. Ceci a pour avantage de ne pas tenir compte de l'épaisseur du trait dans la taille totale de l'élément.

La classe `ring` s'écrit suivant la syntaxe `ring-{taille}`. L'attribut `taille` est facultatif et prend par défaut une valeur de 3 pixels pour l'épaisseur de la bordure. Si la taille est précisée, nous pouvons ajouter 0, 1, 2, 4 ou 8 pixels.

Il est possible de placer la bordure à l'intérieur de l'élément avec la classe `ring-inset` et de spécifier sa couleur avec la classe `ring-{couleur}`.

```
<p class="ring ring-red-600 ring-inset">Une bordure utilisant "box
shadow".</p>
```

Si vous souhaitez effectuer un décalage de la bordure, vous utiliserez la classe `ring-offset-{taille}`.

Couleur de fond

Dans cette section, nous allons explorer l'utilisation des couleurs de fond dans Tailwind CSS. Nous découvrirons comment utiliser les classes utilitaires pour appliquer rapidement et facilement des couleurs aux éléments HTML. En plus des couleurs unies, nous verrons qu'il est possible de créer des dégradés de couleurs pour réaliser un design percutant.

Couleur unie

Nous passerons rapidement cette section étant donné que nous avons déjà abordé les couleurs précédemment. Pour appliquer une couleur de fond à un élément, il suffit d'ajouter le préfixe `bg` suivi du nom de la couleur.

N'oubliez pas que vous avez la possibilité d'agir sur l'opacité en ajoutant une fraction.

```
<p class="bg-red-500">Un fond de couleur rouge.</p>
<p class="bg-red-500/20">Un fond de couleur rouge d'une opacité de 20 %.</p>
```

Il est possible également de définir une couleur personnalisée avec une valeur arbitraire exprimée en hexadécimal.

```
<p class="bg-[#00eb8e]">Un fond de couleur vert.</p>
```

Dégradé de couleurs

Pour ajouter un fond en dégradé de couleurs à un élément, il faut d'abord lui donner une direction avec la classe `bg-gradient-{direction}`.

Tableau 5–10 Dégradés de couleur selon une direction.

Classes	Direction
bg-none	Aucun dégradé
bg-gradient-to-t	Dégradé vertical vers le haut
bg-gradient-to-tr	Dégradé en diagonale vers le coin en haut à droite
bg-gradient-to-r	Dégradé horizontal vers la droite

Tableau 5–10 Dégradés de couleur selon une direction. *(suite)*

Classes	Direction
bg-gradient-to-br	Dégradé en diagonale vers le coin en bas à droite
bg-gradient-to-b	Dégradé vertical vers le bas
bg-gradient-to-bl	Dégradé en diagonale vers le coin en bas à gauche
bg-gradient-to-l	Dégradé horizontal vers la gauche
bg-gradient-to-tl	Dégradé en diagonale vers le coin en haut à gauche

Une fois la direction donnée, nous déterminerons une couleur de départ et une couleur d'arrivée grâce aux classes `from-{couleur}` et `to-{couleur}`.

```
<p class="bg-gradient-to-r from-orange-100 to-orange-500">Un dégradé de
couleurs vers la droite.</p>
<p class="bg-gradient-to-l from-orange-100 to-orange-500">Le même dégradé
vers la gauche.</p>
```

À noter que nous pouvons réaliser un autre effet intéressant. Que diriez-vous de faire un fondu d'une couleur vers un fond transparent ? Rien de plus facile, il suffit pour cela de ne pas spécifier de couleur de destination.

```
<p class="bg-gradient-to-r from-slate-300">Un fondu en transparence.</p>
```

Si vous le souhaitez, vous pouvez également ajouter au dégradé une couleur intermédiaire avec la classe `via-{couleur}`.

```
<p class="bg-gradient-to-tr from-yellow-200 via-orange-600 to-pink-800 h-
48">Un dégradé avec 3 couleurs.</p>
```

Image d'arrière-plan

Tailwind CSS fournit des classes utilitaires pour gérer la manière d'afficher une image d'arrière-plan dans un élément HTML (taille, position…).

Notez qu'il ne permet pas d'intégrer une image provenant d'une URL. Aussi, pour ajouter une image d'arrière-plan, nous disposons de trois méthodes.

- Ajouter l'attribut `style` dans l'élément HTML.

```
<div class="w-60 h-60" style="background-image:url('images/kayak.jpg');"></div>
```

- Utiliser une classe avec une valeur arbitraire (attention à ne pas laisser d'espace au niveau de la syntaxe entre crochets).

```
<div class="bg-[url('images/kayak.jpg')] w-60 h-60"></div>
```

- Créer une classe.

```
<style>
    .bg-kayak {
        background-image: url('images/kayak.jpg');
    }
</style>
<div class="bg-kayak w-60 h-60"></div>
```

Tailwind CSS propose différentes classes pour positionner l'image de fond dans l'élément HTML, à l'aide de la syntaxe suivante : `bg-{position}`.

Tableau 5–11 Positionnement de l'image.

Classes	Propriétés CSS
bg-bottom	background-position: bottom;
bg-center	background-position: center;
bg-left	background-position: left;
bg-left-bottom	background-position: left bottom;
bg-left-top	background-position: left top;
bg-right	background-position: right;
bg-right-bottom	background-position: right bottom;
bg-right-top	background-position: right top;
bg-top	background-position: top;

La figure 5-4 montre différents placements de l'image à l'intérieur de l'élément.

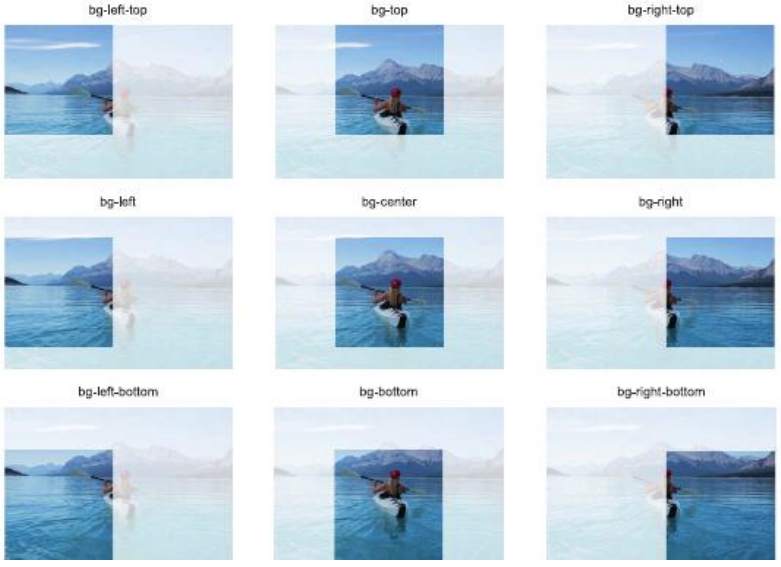


Figure 5–4
Position de l'image par rapport à l'élément.

Nous pouvons forcer l'image à s'adapter à la taille de l'élément avec les classes suivantes :

- `bg-cover` : l'image va couvrir la surface de l'élément ;
- `bg-contain` : l'image va s'adapter à l'élément en hauteur ou en largeur sans être recadrée ni étirée.

La classe `bg-auto` est la valeur par défaut et permet de conserver la taille initiale de l'image. Si l'image est plus petite que l'élément, elle se répétera sur l'axe horizontal et vertical par défaut. Il est possible de contrôler ce comportement grâce aux classes suivantes :

- `bg-repeat` : comportement par défaut dans lequel l'image se répète horizontalement et verticalement ;
- `bg-no-repeat` : l'image n'est pas répétée ;
- `bg-repeat-x` : l'image est répétée horizontalement ;
- `bg-repeat-y` : l'image est répétée verticalement.

Effets et filtres

Avec l'arrivée du CSS 3, nous pouvons désormais appliquer des effets et des filtres sur les éléments HTML de nos pages web. Il est possible de créer des ombres portées ou encore de réaliser des modes de fusion, comme nous pourrions le faire avec un logiciel de retouche photo.

Tailwind CSS propose différentes classes pour intégrer ces effets dans nos réalisations.

Effets

Ombres portées

Un effet que l'on retrouve dans de nombreuses mises en page modernes est l'ombre portée.

La propriété CSS `box-shadow` se verra attribuer les classes Tailwind suivantes : `shadow-sm`, `shadow`, `shadow-md`, `shadow-lg`, `shadow-xl` ou encore `shadow-2xl`.

Le suffixe indique la taille de l'ombre, de la plus petite à la plus grande dans la liste précédente.

Il est également possible d'ajouter une ombre interne grâce à la classe `shadow-inner`.

Si vous souhaitez ajouter de la couleur à vos ombres, vous utiliserez la classe `shadow-{couleur}`.

Par défaut, la couleur de l'ombre possède une opacité de 100 %, que vous pouvez diminuer pour réaliser de meilleurs effets.

Voici un exemple avec une ombre de couleur cyan et une opacité de 50 % :

```
shadow-lg shadow-cyan-500/50
```

Opacité

Il est possible de modifier l'opacité d'un élément avec la classe `opacity-{valeur}`.

L'attribut `valeur` peut prendre un nombre entre 0 et 100, par pas de 5 ou 10 (voir la documentation sur le site de Tailwind CSS pour obtenir le détail de ces valeurs : https://tailwindcss.com/docs/opacity).

Mode de fusion

La propriété CSS `mix-blend-mode` définit la façon dont le contenu d'un élément se mélange avec son arrière-plan.

La syntaxe de la classe Tailwind est la suivante : `mix-blend-{mode}`.

Voici une liste de quelques modes :

- `mix-blend-multiply`
- `mix-blend-screen`
- `mix-blend-overlay`
- `mix-blend-darken`
- `mix-blend-lighten`
- etc.

De la même manière, la propriété CSS `background-blend-mode` permet de contrôler la façon dont l'image d'arrière-plan d'un élément fusionne avec sa couleur d'arrière-plan. La syntaxe de la classe Tailwind est `bg-blend-{mode}`.

Filtres

La propriété CSS `filter` permet d'ajouter un effet graphique sur un élément tel que le flou, le contraste, la saturation…

Les classes Tailwind s'écrivent avec le nom de l'effet :

- `blur-{valeur}` : applique un effet de flou ;
- `brightness-{valeur}` : augmente ou diminue la luminosité ;
- `contrast-{valeur}` : applique un effet de contraste ;
- `grayscale` ou `grayscale-0` : convertit un élément en niveaux de gris ou annule l'effet ;
- `hue-rotate-{valeur}` : modifie la teinte d'un élément ;
- `invert` ou `invert-0` : applique un effet de négatif ou l'annule ;
- `saturate-{valeur}` : contrôle le niveau de saturation ;
- `sepia` ou `sepia-0` : ajoute le filtre sépia ou l'annule.

Sur le même principe, la propriété CSS `backdrop-filter` est utilisée pour appliquer un effet graphique à la zone située derrière un élément.

Ces effets sont associés aux classes Tailwind suivantes :

- `backdrop-blur-{valeur}`
- `backdrop-brightness-{valeur}`
- `backdrop-contrast-{valeur}`
- `backdrop-grayscale` ou `backdrop-grayscale-0`
- `backdrop-hue-rotate-{valeur}`
- `backdrop-invert` ou `backdrop-invert-0`
- `backdrop-opacity-{valeur}`
- `backdrop-saturate-{valeur}`
- `backdrop-sepia` ou `backdrop-sepia-0`

Consultez la documentation de Tailwind CSS pour en savoir plus sur les valeurs que peuvent prendre ces filtres.

6

Mise en page

La mise en page est incontournable en design web. En effet, une mise en page correctement effectuée améliore l'expérience utilisateur. Elle facilite la navigation et met en avant le contenu important. Un visiteur qui trouve facilement l'information restera plus longtemps sur le site.

Tailwind CSS vous permettra de créer facilement des designs qui seront à la fois élégants et fonctionnels.

Dans ce chapitre, nous allons aborder certains points essentiels pour concevoir et structurer efficacement vos pages web avec Tailwind CSS.

Nous étudierons la classe `container`, élément clé pour créer des mises en page fluides et adaptatives ; les classes qui permettent de créer des dispositions en colonnes ; la propriété `display` pour changer le comportement d'un élément, ainsi que la propriété `float` pour positionner les éléments dans la page.

Conteneur

La classe `container` permet de fixer la largeur du contenu de la page en fonction de la taille d'écran.

Par exemple, si un écran possède une taille comprise entre 640 et 768 pixels, alors le point de rupture est fixé pour une largeur maximale de 640 px. Si la largeur d'écran dépasse 768 pixels, alors la largeur du conteneur est fixée à 768 px.

Pour rappel, un point de rupture (*breakpoint* en anglais) définit le moment où le contenu et la mise en page vont s'adapter en fonction d'une largeur d'écran prédéfinie. Lorsque la largeur de

l'écran est supérieure ou égale à une valeur spécifiée, des règles CSS viendront modifier la mise en page (en réduisant la taille de certains éléments, en modifiant le nombre de colonnes, etc.).

Le tableau 6-1 indique les différents points de rupture. Pour les écrans dont la taille est inférieure à 640 px, la largeur du conteneur est de 100 %.

Tableau 6–1 Les points de rupture.

Classe	Point de rupture	Propriétés CSS
container	Aucun	width: 100%;
	sm (640px)	max-width: 640px;
	md (768px)	max-width: 768px;
	lg (1024px)	max-width: 1024px;
	xl (1280px)	max-width: 1280px;
	2xl (1536px)	max-width: 1536px;

Il est à noter que certains frameworks vont automatiquement centrer le conteneur. Ce n'est pas le cas de Tailwind CSS. Nous devons donc ajouter la classe `mx-auto` si nous souhaitons spécifier ce comportement par défaut.

```
<div class="container mx-auto"></div>
```

Colonnes

Comme pour un magazine, il est possible de présenter le texte sous forme de colonnes.

Pour cela, nous utiliserons la classe `columns-{nombre}` ou encore `columns-{largeur}`.

L'attribut `nombre` peut prendre une valeur comprise entre 1 et 12 pour définir le nombre de colonnes. L'attribut `largeur` peut, quant à lui, prendre les valeurs 3xs, 2xs, xs, sm, md, lg, xl, 2xl, 3xl, 4xl, 5xl, 6xl et 7xl pour définir une largeur de colonne.

```
<div class="container mx-auto columns-4">
    <p class="mb-2">Paragraphe 1 - Lorem ipsum, dolor sit amet consectetur
adipisicing elit. In quaerat ipsam vel delectus nulla exercitationem cum
ipsa doloribus minima ad? Officiis sed hic quae natus quis optio perferendis
laboriosam qui?</p>
    <p class="mb-2">Paragraphe 2 - Lorem ipsum, dolor sit amet consectetur
adipisicing elit. In quaerat ipsam vel delectus nulla exercitationem cum
ipsa doloribus minima ad? Officiis sed hic quae natus quis optio perferendis
laboriosam qui?</p>
    <p class="mb-2">Paragraphe 3 - Lorem ipsum, dolor sit amet consectetur
adipisicing elit. In quaerat ipsam vel delectus nulla exercitationem cum
ipsa doloribus minima ad? Officiis sed hic quae natus quis optio perferendis
laboriosam qui?</p>
</div>
```

Tableaux

Au début des années 1990, alors que le CSS n'avait pas encore fait son apparition, on utilisait beaucoup les tableaux pour les mises en page. Mais ils pouvaient vite devenir complexes et, dans cette avalanche de balises `<TR>` et `<TD>`, il était parfois pénible de s'y retrouver pour maintenir un site.

Heureusement, le CSS est apparu et a grandement simplifié la mise en page. Les tableaux ont donc retrouvé leur fonction principale, à savoir l'affichage des données.

Voyons rapidement les classes de Tailwind CSS consacrées aux tableaux.

Border-collapse

La propriété CSS `border-collapse` peut prendre les valeurs `collapse` ou `separate`. Les classes sont définies ainsi avec Tailwind :

- `border-collapse` : permet de combiner les bordures de cellules adjacentes en une seule bordure ;
- `border-separate` : force chaque cellule à afficher ses propres bordures.

Border-spacing

La propriété CSS `border-spacing` permet de contrôler l'espacement entre les bordures des tableaux.

Utilisez les classes `border-spacing-{valeur}` ou `border-spacing-{x|y}-{valeur}` pour contrôler les espacements. L'attribut `valeur` peut prendre un nombre entre 0 et 96.

```
<table class="border-separate border-spacing-2 border border-slate-400">
    <thead>
        <tr>
            <th class="border border-slate-300">Régions</th>
            <th class="border border-slate-300">Départements</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td class="border border-slate-300">Auvergne-Rhône-Alpes</td>
            <td class="border border-slate-300">Ain</td>
        </tr>
        <tr>
            <td class="border border-slate-300">Hauts-de-France</td>
            <td class="border border-slate-300">Aisne</td>
        </tr>
    </tbody>
</table>
```

Table-layout

La propriété CSS `table-layout` peut prendre les valeurs `auto` ou `fixed`. Elle permet de spécifier la largeur des colonnes du tableau.

Avec Tailwind CSS, les classes s'écrivent ainsi :

- `table-auto` : permet de dimensionner automatiquement la largeur des colonnes en fonction du contenu des cellules ;
- `table-fixed` : permet d'ignorer le contenu des cellules afin de spécifier une largeur de colonne fixe. C'est la largeur de la première ligne qui détermine la largeur des colonnes de l'ensemble du tableau. Notez que vous pouvez définir manuellement la largeur de certaines colonnes. La largeur restante pour les autres colonnes sera répartie uniformément.

Caption-side

La propriété CSS `caption-side`, qui peut prendre comme valeur `top` ou `bottom`, permet de positionner une légende (balise `<CAPTION>`) au-dessus ou en dessous du tableau.

La classe Tailwind CSS s'écrit simplement `caption-top` ou `caption-bottom`.

Propriété display

Certaines classes utilitaires de Tailwind CSS permettent de modifier la propriété `display`. Ces classes sont faciles à retenir, puisque leur nom prend la valeur de la propriété `display`. Le tableau 6-2 mentionne les plus courantes.

Tableau 6–2 La propriété display.

Classes	Propriétés CSS
`block`	`display: block;`
`inline-block`	`display: inline-block;`
`inline`	`display: inline;`
`flex`	`display: flex;`
`inline-flex`	`display: inline-flex;`
`table`	`display: table;`
…	…
`grid`	`display: grid;`
`inline-grid`	`display: inline-grid;`
`hidden`	`display: none;`

Attention toutefois à la classe `hidden` qui équivaut ici à `display: none`. Nous pourrions la confondre avec `visibility: hidden`.

Le tableau 6-3 présente, quant à lui, les classes qui permettent de gérer la visibilité d'un élément.

Tableau 6–3 La propriété visibility.

Classes	Propriétés CSS
`visible`	`visibility: visible;`
`invisible`	`visibility: hidden;`
`collapse`	`visibility: collapse;`

Propriété float

Si vous souhaitez vous servir de cette propriété, utilisez les classes de Tailwind CSS présentées dans le tableau 6-4.

Tableau 6–4 La propriété float.

Classes	Propriétés CSS
`float-right`	`float: right;`
`float-left`	`float: left;`
`float-none`	`float: none;`

Pour annuler un flottement, utilisez les classes du tableau 6-5.

Tableau 6–5 La propriété clear.

Classes	Propriétés CSS
`clear-left`	`clear: left;`
`clear-right`	`clear: right;`
`clear-both`	`clear: both;`
`clear-none`	`clear: none;`

Notez que l'utilisation de la propriété `float` peut rendre votre mise en page difficile à gérer et à maintenir. Même si nous pouvons l'utiliser de manière ponctuelle, cette propriété est devenue obsolète et il est donc recommandé d'utiliser d'autres techniques de mise en page : les boîtes flexibles *(flexbox)* ou l'affichage en grille *(grid)*, que nous étudierons aux chapitres 7 et 8.

7

Flexbox

Tailwind CSS fournit différentes classes utilitaires qui permettent d'appliquer facilement la propriété `flex` à des éléments HTML. Vous pourrez ainsi créer des mises en page flexibles et réactives qui s'adapteront à différents appareils et tailles d'écran.

Flexbox est une technique qui permet de disposer les éléments horizontalement ou verticalement et d'en contrôler les espacements et les alignements. Pour ce faire, nous avons besoin d'un élément parent – le *flex container* – et d'éléments enfants – les *flex items*.

Propriétés du flex container

Dans cette section, nous allons aborder les propriétés qui s'appliquent au flex container.

Propriété flex

Comme nous l'avons vu précédemment avec la propriété `display`, la classe de Tailwind CSS qui permet d'appliquer un `display: flex` est tout simplement `flex`.

Dans cet exemple, nous appliquons `display: flex` sur un élément parent.

```
<div class="container mx-auto flex">
    <p class="p-3">Paragraphe 1</p>
    <p class="p-3">Paragraphe 2</p>
    <p class="p-3">Paragraphe 3</p>
</div>
```

Le résultat de ce code est l'affichage des paragraphes, les éléments enfants, en colonnes (figure 7-1).

Figure 7–1
Affichage des éléments avec la classe flex.

Direction

Comme nous venons de le voir, si nous ajoutons la classe `flex` à un élément parent, les éléments enfants se placent alors les uns à côté des autres sur une seule ligne. Il s'agit du comportement par défaut.

Le tableau 7-1 présente les classes utilisées par Tailwind CSS pour spécifier une direction pour les éléments.

Tableau 7–1 La propriété flex-direction.

Classes	Propriétés CSS	Alignement
`flex-row`	`flex-direction: row;`	Sens horizontal de la gauche vers la droite
`flex-row-reverse`	`flex-direction: row-reverse;`	Sens horizontal de la droite vers la gauche
`flex-col`	`flex-direction: column;`	Sens vertical du haut vers le bas
`flex-col-reverse`	`flex-direction: column-reverse;`	Sens vertical du bas vers le haut

Voici le même exemple que précédemment, mais cette fois nous spécifions une direction en sens inverse (en partant de la droite de l'écran).

```
<div class="container mx-auto flex flex-row-reverse">
    <p class="p-3">Paragraphe 1</p>
    <p class="p-3">Paragraphe 2</p>
    <p class="p-3">Paragraphe 3</p>
</div>
```

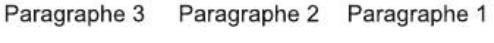


Figure 7–2
Affichage des éléments en sens inverse.

Un autre exemple d'utilisation de `flex-row-reverse` consiste à afficher des profils en alternant l'ordre d'affichage (photo et texte puis texte et photo) selon les lignes paires ou impaires (figure 7-3).

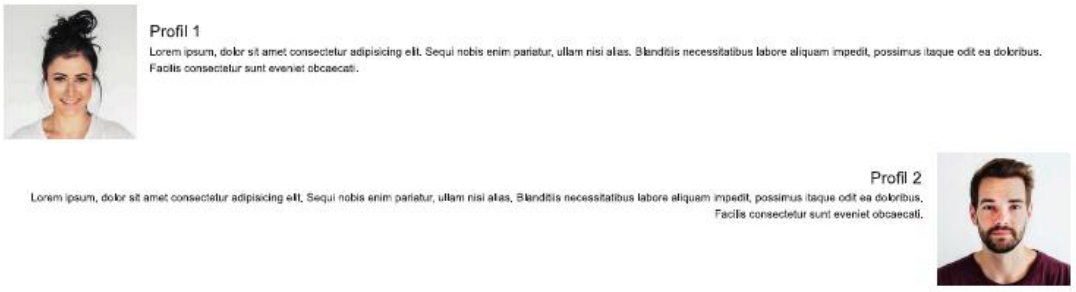


Figure 7–3 Un exemple de mise en page.

Voici le code permettant une telle disposition :

```
<div class="container mx-auto">
    <div class="flex mt-5 even:flex-row-reverse even:text-right">
        <img class="w-48 h-48" src="images/profil1.jpg" >
        <div class="p-5">
            <h3 class="text-2xl">Profil 1</h3>
            <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
Sequi nobis enim pariatur, ullam nisi alias. Blanditiis necessitatibus
labore aliquam impedit, possimus itaque odit ea doloribus. Facilis
consectetur sunt eveniet obcaecati.</p>
        </div>
    </div>
    <div class="flex mt-5 even:flex-row-reverse even:text-right">
        <img class="w-48 h-48" src="images/profil2.jpg" >
        <div class="p-5">
            <h3 class="text-2xl">Profil 2</h3>
            <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
Sequi nobis enim pariatur, ullam nisi alias. Blanditiis necessitatibus
labore aliquam impedit, possimus itaque odit ea doloribus. Facilis
consectetur sunt eveniet obcaecati.</p>
        </div>
    </div>
</div>
```

Bien que nous ayons conservé l'image à gauche et le texte à droite dans le flux du HTML, les classes `even:flex-row-reverse` et `even:text-right` nous ont permis d'alterner la position du texte, puis de l'image sur les lignes paires. Notez l'utilisation de la pseudo classe `even` qui se traduit en CSS par `:nth-child(even)`.

Le fait de définir une direction en ligne ou en colonne va directement déterminer comment les éléments vont être espacés entre eux. Nous allons voir dans la section suivante comment répartir les éléments dans leur conteneur.

Propriété justify-content

La classe utilitaire `justify-{espacement}` est utilisée pour contrôler l'espacement des éléments entre eux selon une direction.

Tableau 7–2 La propriété justify-content.

Classes	Propriétés CSS	Disposition
`justify-start`	`justify-content: flex-start;`	Les éléments sont placés au début du conteneur, sans espace.
`justify-end`	`justify-content: flex-end;`	Les éléments sont placés à la fin du conteneur, sans espace.
`justify-center`	`justify-content: center;`	Les éléments sont centrés dans le conteneur, sans espace.
`justify-between`	`justify-content: space-between;`	Tous les éléments sont espacés autant que possible, le premier élément étant placé au début et le dernier à la fin du conteneur (sans marge).
`justify-around`	`justify-content: space-around;`	L'espace avant et après les éléments est deux fois moins important que l'espace entre les éléments.
`justify-evenly`	`justify-content: space-evenly;`	Les espaces avant, après et entre les éléments sont identiques.

Voici un exemple de répartition de nos trois paragraphes :

```
<div class="bg-slate-300 container mx-auto flex justify-between">
    <p>Paragraphe 1</p>
    <p>Paragraphe 2</p>
    <p>Paragraphe 3</p>
</div>
```

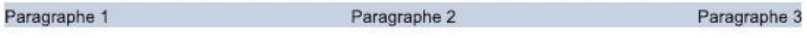


Figure 7–4
Un exemple de répartition des éléments avec « justify-between ».

Propriété flex-wrap

Par défaut, les éléments vont se « compresser » afin de tenir sur la même ligne. Parfois, il pourra être nécessaire de forcer un retour à la ligne à l'aide de la classe `flex-wrap`.

Tableau 7–3 La propriété flex-wrap.

Classes	Propriétés CSS
`flex-wrap`	`flex-wrap: wrap;`
`flex-wrap-reverse`	`flex-wrap: wrap-reverse;`
`flex-nowrap`	`flex-wrap: nowrap;`

Dans l'exemple suivant, nous avons fixé la largeur des paragraphes à 50 % avec la classe `w-1/2`. Le troisième paragraphe passe à la ligne grâce à l'ajout de la classe `flex-wrap`. Sans l'ajout de cette classe, les éléments seraient disposés sur la même ligne et la largeur des paragraphes se serait adaptée à 33,33 %.

```
<div class="container mx-auto flex flex-wrap">
    <p class="bg-slate-300 w-1/2">Paragraphe 1</p>
    <p class="bg-slate-300 w-1/2">Paragraphe 2</p>
    <p class="bg-slate-300 w-1/2">Paragraphe 3</p>
</div>
```

Figure 7–5
Forcer un retour à la ligne avec flex-wrap.

Propriété align-items

La classe utilitaire `items-{alignement}` permet d'aligner les éléments selon l'axe transversal (perpendiculaire à l'axe principal). Autrement dit, si l'axe principal est horizontal, alors la propriété `align-items` effectue une répartition des éléments verticalement. Au contraire, si l'axe principal est vertical, alors l'alignement des éléments se fait horizontalement.

Tableau 7–4 La propriété align-items.

Classes	Propriétés CSS	Disposition
`items-start`	`align-items: flex-start;`	Les éléments sont alignés au début du conteneur.
`items-end`	`align-items: flex-end;`	Les éléments sont alignés à la fin du conteneur.
`items-center`	`align-items: center;`	Les éléments sont centrés.
`items-baseline`	`align-items: baseline;`	Les éléments sont alignés sur la ligne de base.
`items-stretch`	`align-items: stretch;`	Les éléments sont étirés afin de remplir le conteneur.

Propriété align-content

La classe utilitaire `content-{alignement}` agit sur les éléments disposés en plusieurs lignes.

Tableau 7–5 La propriété align-content.

Classes	Propriétés CSS	Disposition
`content-center`	`align-content: center;`	Les éléments sont regroupés au centre du conteneur.
`content-start`	`align-content: flex-start;`	Les éléments sont regroupés au début du conteneur.
`content-end`	`align-content: flex-end;`	Les éléments sont regroupés à la fin du conteneur.

Tableau 7–5 La propriété align-content. *(suite)*

Classes	Propriétés CSS	Disposition
`content-between`	`align-content: space-between;`	Les éléments sont répartis dans le conteneur (sans marges).
`content-around`	`align-content: space-around;`	Les éléments sont répartis dans le conteneur (avec marges).
`content-evenly`	`align-content: space-evenly;`	Les espaces de répartition et les marges sont égaux.
`content-stretch`	`align-content: stretch;`	Les éléments s'étirent afin de s'adapter à la taille du conteneur.

Propriétés des flex items

Voyons maintenant quelles propriétés peuvent être utilisées pour les éléments enfants.

Propriété flex-grow

La largeur par défaut d'un élément `flex` est la largeur de son contenu. La propriété `flex-grow` permet d'élargir un élément de manière à remplir le conteneur.

Les classes de Tailwind CSS sont :

- `grow-0` : équivaut à `flex-grow: 0`, propriété par défaut ;
- `grow` : équivaut à `flex-grow: 1`.

Voici un exemple pour illustrer cette propriété.

```
<div class="w-52 flex text-2xl">
    <p class="bg-red-400 p-3">1</p>
    <p class="bg-indigo-400 p-3 grow">2</p>
    <p class="bg-cyan-400 p-3">3</p>
</div>
```

Figure 7–6
Élargissement de l'élément 2 avec grow.

Propriété flex-shrink

Le comportement par défaut d'un élément flexible est de se rétrécir de manière que l'ensemble des éléments puisse tenir dans le conteneur (à condition, bien sûr, qu'aucun retour à la ligne ne soit spécifié via `flex-wrap`). Néanmoins, cela peut parfois provoquer des effets indésirables sur une mise en page (figure 7-7).

Figure 7–7
Effet indésirable de l'affichage de l'heure.

L'affichage de l'heure n'est pas très beau. Nous allons donc utiliser `shrink-0` pour empêcher un élément flexible de se rétrécir.

```
<div class="w-64 flex text-xl bg-slate-100">
    <h3 class="p-2">Votre prochain rendez-vous</h3>
    <p class="p-2 bg-red-400 shrink-0">10h30</p>
</div>
```

Figure 7–8
Affichage correct de l'heure.

Voilà qui est mieux, n'est-ce pas ?

Les classes de Tailwind CSS sont :

- `shrink-0` : équivaut à `flex-shrink: 0` ;
- `shrink` : équivaut à `flex-shrink: 1`, propriété par défaut.

Propriété flex-basis

La propriété `flex-basis` détermine la taille d'un élément avant même que l'espace disponible soit réparti entre les éléments flexibles du conteneur.

Cette taille peut s'exprimer en pixels (ou rem), en pourcentage ou encore avec une valeur comme `auto`.

La classe utilitaire est `basis-{valeur}` et correspond à la propriété CSS `flex-basis`.

Le tableau 7-6 présente les différentes classes de Tailwind CSS.

Tableau 7–6 La propriété flex-basis.

Classes	Propriétés CSS
basis-0	flex-basis: 0px;
basis-1	flex-basis: 0.25rem; /* 4px */
basis-2	flex-basis: 0.5rem; /* 8px */
...	...
basis-96	flex-basis: 24rem; /* 384px */
basis-auto	flex-basis: auto;
basis-px	flex-basis: 1px;

Tableau 7–6 La propriété flex-basis. *(suite)*

Classes	Propriétés CSS
basis-0.5	flex-basis: 0.125rem; /* 2px */
basis-1.5	flex-basis: 0.375rem; /* 6px */
basis-2.5	flex-basis: 0.625rem; /* 10px */
basis-3.5	flex-basis: 0.875rem; /* 14px */
basis-1/2	flex-basis: 50%;
basis-1/3	flex-basis: 33.333333%;
...	...
basis-11/12	flex-basis: 91.666667%;
basis-full	flex-basis: 100%;

Propriété order

Il est possible de modifier l'ordre des éléments avec la classe `order-{ordre}`.

Tableau 7–7 La propriété order.

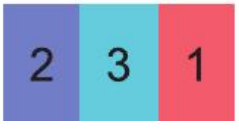
Classes	Propriétés CSS	Commentaires
order-1	order: 1;	Peut prendre une valeur de 1 à 12 : order-2, order-3...
order-first	order: -9999;	Place l'élément au début.
order-last	order: 9999;	Place l'élément à la fin.
order-none	order: 0;	Propriété par défaut.

Par défaut, l'ordre est à 0. Donc, le simple fait d'ajouter `order-1` à un élément suffit pour le placer à la fin.

Dans l'exemple suivant, sans toucher le flux HTML, nous positionnons l'élément `1` en dernier.

```
<div class="flex text-2xl">
    <p class="bg-red-400 p-3 order-1">1</p>
    <p class="bg-indigo-400 p-3">2</p>
    <p class="bg-cyan-400 p-3">3</p>
</div>
```

Figure 7–9
Manipuler l'ordre des éléments.



Propriété align-self

Les classes utilitaires `self-{alignement}` pour un élément `flex` sont similaires aux classes `items-{alignement}` mentionnées précédemment. Les éléments vont être alignés sur l'axe transversal.

Notez que les classes `self` s'appliquent à un élément alors que les classes `items` s'appliquent au conteneur afin d'aligner tous ses éléments.

Tableau 7–8 La propriété align-self.

Classes	Propriétés CSS	Disposition
`self-start`	`align-self: flex-start;`	L'élément est aligné au début du conteneur.
`self-end`	`align-self: flex-end;`	L'élément est aligné à la fin du conteneur.
`self-center`	`align-self: center;`	L'élément est centré.
`self-stretch`	`align-self: stretch;`	L'élément s'étire afin de remplir le conteneur.
`self-baseline`	`align-self: baseline;`	L'élément est aligné sur la ligne de base.

8

Grid

Comme nous venons de le voir dans le chapitre précédent, les boîtes flexibles sont bien pratiques pour réaliser des mises en page. Sachez néanmoins que le système de grille CSS est aussi une alternative très intéressante pour réaliser des mises en page flexibles et réactives.

En effet, alors que les boîtes flexibles permettent de disposer les éléments dans une seule dimension (en ligne ou en colonne), la grille permet de disposer et d'aligner les éléments dans les deux dimensions.

Comme pour l'utilisation de Flexbox, nous aurons ici aussi besoin d'un élément parent – le *grid container* – et d'éléments enfants – les *grid items* – pour mettre en place le système de grille.

Propriétés du grid container

Propriété grid-template-columns

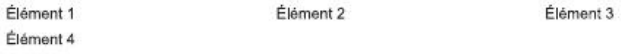
Pour appliquer un `display: grid` à notre élément parent, nous utiliserons la classe `grid` de Tailwind CSS.

Utilisée toute seule, cette classe ne suffit pas pour modifier l'affichage dans la page. Il faut ajouter la classe `grid-cols-{valeur}` afin de spécifier un nombre de colonnes avec une valeur comprise entre 1 et 12.

L'exemple suivant dispose les éléments dans une grille de trois colonnes :

```
<div class="container grid grid-cols-3">
    <div>Élément 1</div>
    <div>Élément 2</div>
    <div>Élément 3</div>
    <div>Élément 4</div>
</div>
```

Figure 8–1
Affichage des éléments en grille sur 3 colonnes.



Les colonnes ont ici la même largeur. Dans certaines mises en page, vous pourriez cependant vouloir spécifier des largeurs différentes.

Prenons l'exemple d'une page qui dispose d'une barre latérale à gauche et du contenu principal à droite. Nous pourrions recourir aux valeurs arbitraires pour avoir une barre latérale avec une largeur de 30 % et le contenu principal sur les 70 % restants. Dans le code, veillez à ne pas mettre d'espace après la virgule dans les valeurs arbitraires.

```
<div class="container grid grid-cols-[30%,70%]">
    <div class="bg-red-200 p-3">Barre latérale</div>
    <div class="bg-sky-200 p-3">Contenu principal</div>
</div>
```

Figure 8–2
Une mise en page avec une barre latérale à gauche.

Propriété grid-template-rows

Semblable à la propriété précédente, la propriété `grid-template-rows` agit sur les lignes de la grille. La classe Tailwind CSS est `grid-rows-{valeur}` où `valeur` peut être un nombre de 1 à 6.

Il est possible de spécifier des valeurs arbitraires pour déterminer la hauteur de ligne.

Voici un exemple de gabarit de page avec l'en-tête, le contenu principal et le pied de page.

```
<body class="container grid grid-rows-[60px,1fr,60px]">
    <header class="bg-red-200 p-3">Header</header>
    <div class="p-3">
        <h1>Contenu principal</h1>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Porro
eius asperiores reiciendis harum inventore, aliquam culpa doloremque. Minus,
iure. Nulla aliquam inventore velit obcaecati nam ipsum debitis facilis
tempore nihil.</p>
    </div>
    <footer class="bg-sky-200 p-3">Footer</footer>
</body>
```

Figure 8–3
Exemple de gabarit de page.

Propriété gap

La classe utilitaire `gap` de Tailwind CSS permet de définir des gouttières entre les lignes et les colonnes d'une grille.

Notez que cette classe s'applique également aux boîtes flexibles que nous avons vues dans le chapitre précédent.

Nous pouvons définir une taille de gouttière entre les colonnes avec la classe `gap-x-{valeur}`. Les espaces entre les lignes s'effectueront, quant à eux, avec la classe `gap-y-{valeur}`. Ceci s'avérera utile si nous souhaitons avoir une gouttière différente en fonction des lignes ou des colonnes. Dans le cas contraire, nous utiliserons plutôt `gap-{valeur}` pour obtenir des espaces identiques entre les lignes et les colonnes.

L'attribut `valeur` peut prendre un nombre compris entre 0 et 96, parmi les 34 valeurs proposées par Tailwind CSS.

Voici un exemple d'une grille avec des tailles de gouttières différentes pour les lignes et les colonnes.

```
<div class="container grid grid-cols-3 gap-x-5 gap-y-2">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

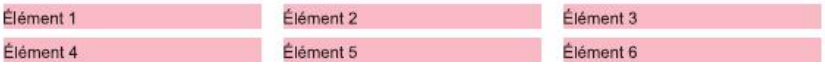


Figure 8–4
Exemple d'utilisation de la propriété gap.

Propriété justify-content

Nous avons déjà vu la propriété `justify-content` au chapitre précédent sur les boîtes flexibles. Dans le système de grille, cette propriété aura le même effet.

Pour rappel, les classes de Tailwind CSS sont : `justify-start`, `justify-end`, `justify-center`, `justify-between`...

La différence avec les boîtes flexibles est que la propriété agit dans la grille sur l'axe horizontal (la répartition des éléments peut s'effectuer dans le sens horizontal ou vertical pour les boîtes flexibles, en fonction de la direction qui a été donnée).

Attention, si les éléments sont flexibles, la propriété `justify-content` n'aura pas d'effet. Il faut que la taille totale des éléments soit inférieure à la taille du conteneur pour que nous puissions spécifier une répartition.

Dans l'exemple suivant, la grille se compose de quatre colonnes de largeur `auto` (qui s'adapte au contenu).

```
<div class="container bg-slate-200 grid grid-cols-[repeat(4,auto)] gap-2
justify-end">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–5
Largeur des colonnes qui s'adaptent au contenu.

Si nous avions laissé `grid-cols-4` au lieu de `grid-cols-[repeat(4,auto)]`, nous aurions obtenu l'affichage suivant :

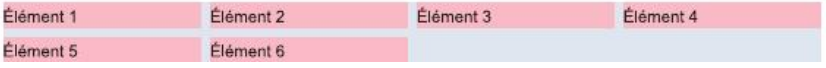


Figure 8–6
Les quatre colonnes ont la même largeur.

Les éléments de la grille possèdent la même largeur et nous pouvons constater que la classe `justify-end` n'a pas d'effet.

Propriété align-content

Nous avons également déjà étudié la propriété `align-content` dans le chapitre précédent sur les boîtes flexibles.

Pour rappel, Tailwind CSS propose les classes suivantes : `content-center`, `content-start`, `content-end`, `content-between`...

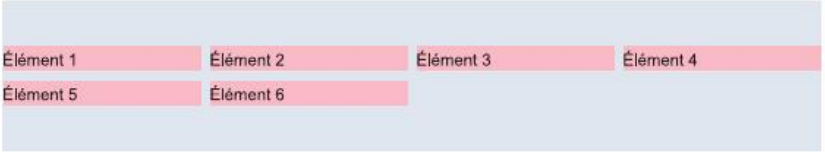
Contrairement à `justify-content`, la propriété `align-content` va répartir les éléments verticalement.

Pour que la propriété puisse avoir un effet, il faudra que la hauteur totale des éléments soit inférieure à la hauteur du conteneur.

Dans l'exemple suivant, une hauteur a été fixée au conteneur pour pouvoir apprécier la classe `content-center`.

```
<div class="container h-36 bg-slate-200 grid grid-cols-4 gap-2 content-
center">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–7
Les éléments sont centrés verticalement dans le conteneur.



Propriété place-content

La propriété `place-content` est une propriété raccourcie pour agir à la fois sur `justify-content` et `align-content`. Autrement dit, elle agit sur la répartition horizontale et verticale des éléments dans la grille.

Cette propriété s'applique également aux boîtes flexibles.

Tableau 8–1 La propriété *place-content*.

Classes	Propriétés CSS
`place-content-center`	`place-content: center;`
`place-content-start`	`place-content: start;`
`place-content-end`	`place-content: end;`
`place-content-between`	`place-content: space-between;`
`place-content-around`	`place-content: space-around;`
`place-content-evenly`	`place-content: space-evenly;`
`place-content-baseline`	`place-content: baseline;`
`place-content-stretch`	`place-content: stretch;`

Voici un exemple de répartition des éléments dans leur conteneur.

```
<div class="container h-36 bg-slate-200 grid grid-cols-[repeat(4,auto)]
gap-2 place-content-between">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–8
Exemple de répartition des éléments avec la classe place-content-between.

Propriété justify-items

Si la propriété `justify-content` permet de contrôler l'espace entre les colonnes dans le conteneur, la propriété `justify-items` va aligner horizontalement le contenu à l'intérieur des colonnes.

Tableau 8–2 La propriété justify-items.

Classes	Propriétés CSS
justify-items-start	justify-items: start;
justify-items-end	justify-items: end;
justify-items-center	justify-items: center;
justify-items-stretch	justify-items: stretch;

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2 justify-items-
center">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–9
Le contenu est centré dans la colonne avec la classe justify-items-center.

Propriété align-items

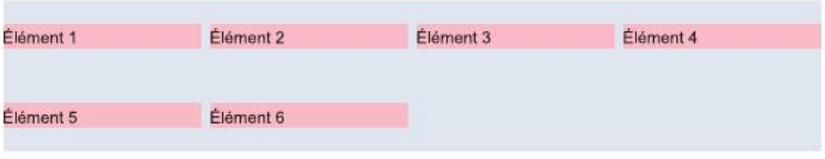
La propriété `align-items` agit de la même manière que `justify-items` à la différence que le contenu, à l'intérieur des éléments, est aligné verticalement.

Tableau 8–3 La propriété align-items.

Classes	Propriétés CSS
items-start	align-items: flex-start;
items-end	align-items: flex-end;
items-center	align-items: center;
items-baseline	align-items: baseline;
items-stretch	align-items: stretch;

```
<div class="container h-36 bg-slate-200 grid grid-cols-4 gap-2
items-center">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–10
Le contenu est centré verticalement avec la classe items-center.



Propriété place-items

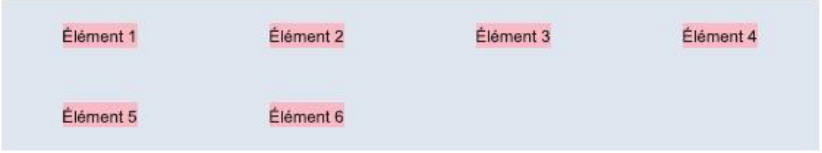
Tout comme `place-content`, la propriété `place-items` est une propriété raccourcie. Elle agit à la fois sur `justify-items` et `align-items`.

Tableau 8–4 La propriété place-items.

Classes	Propriétés CSS
place-items-start	place-items: start;
place-items-end	place-items: end;
place-items-center	place-items: center;
place-items-baseline	place-items: baseline;
place-items-stretch	place-items: stretch;

```
<div class="container h-36 bg-slate-200 grid grid-cols-4 gap-2
place-items-center">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–11
Le contenu est centré verticalement et horizontalement avec la classe place-items-center.



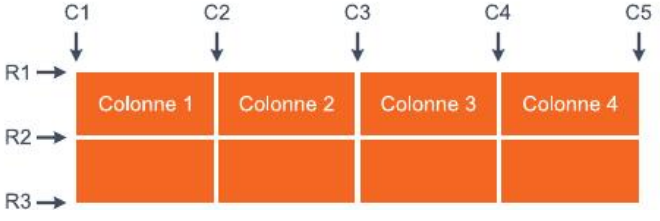
Nous venons de voir de nombreuses propriétés du grid container afin de répartir les éléments en lignes et en colonnes. Nous allons maintenant étudier les propriétés des grid items.

Propriétés des grid items

Propriété grid-column-{start | end}

Pour bien comprendre le fonctionnement des propriétés `grid-column-start` et `grid-column-end`, observons le schéma de la figure 8-12.

Figure 8–12
Comprendre comment s'organise une grille.



Comme nous pouvons le constater, les lignes et les colonnes sont numérotées. Par exemple, la colonne 2 commence au point C2 et se termine au point C3.

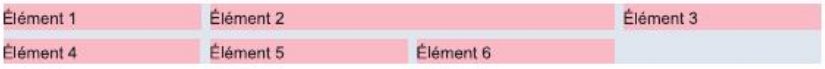
Gardons à l'esprit cette numérotation pour utiliser les propriétés `grid-column-{start | end}`. En indiquant une colonne de départ et une colonne d'arrivée, nous pourrons fusionner un élément sur plusieurs colonnes si besoin.

La syntaxe de la classe Tailwind CSS pour la propriété `grid-column-start` est la suivante : `col-start-{valeur | auto}`, où `valeur` est un nombre de 1 à 13. Pour la propriété `grid-column-end`, la classe Tailwind CSS est `col-end-{valeur | auto}` , où `valeur` est également un nombre de 1 à 13.

Dans l'exemple suivant, la largeur de l'élément 2 s'étale sur deux colonnes.

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200 col-start-2 col-end-4">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–13
L'élément 2 prend sa largeur sur deux colonnes.

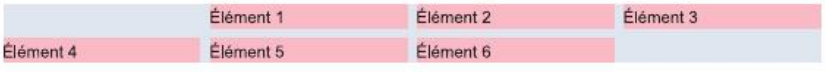


Rien ne nous empêche de réaliser un décalage d'un élément, comme nous pouvons le constater dans l'exemple ci-dessous.

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200 col-start-2 col-end-3">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Le premier élément commence sur la colonne 2 pour terminer sur la colonne 3. On procède ainsi au décalage (figure 8-14).

Figure 8–14
Décalage d'une colonne de l'élément 1.

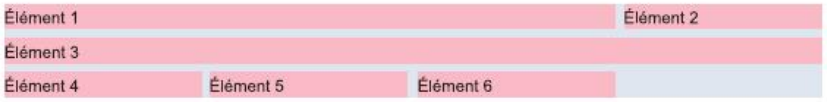


Une autre méthode pour fusionner plusieurs colonnes consiste à utiliser la classe utilitaire `col-span-{1 à 12 | full}`.

En voici un exemple :

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200 col-span-3">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200 col-span-full">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–15
Les éléments 1 et 3 fusionnent sur plusieurs colonnes.



Propriété grid-row-{start | end}

Cette propriété fonctionne de la même manière que la précédente, à la différence que cette fois nous agissons sur les lignes de la grille.

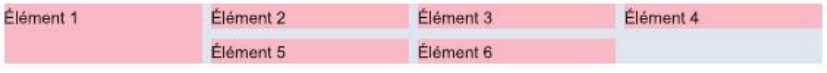
Selon le schéma de la figure 8-12, la première ligne commence au point R1 et se termine au point R2.

Nous pouvons fusionner des cellules sur plusieurs lignes en utilisant les classes suivantes de Tailwind CSS : `row-start-{valeur | auto}` et `row-end-{valeur | auto}`, où `valeur` est un nombre de 1 à 7.

Voici un exemple dont le premier élément va s'étendre sur deux lignes.

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200 row-start-1 row-end-3">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

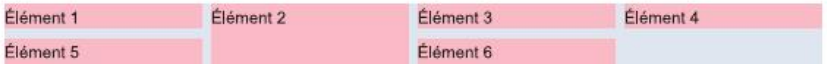
Figure 8–16
L'élément 1 prend une hauteur de deux lignes.



Il est également possible d'utiliser la classe utilitaire `row-span-{valeur | full | auto}`, où `valeur` est un nombre de 1 à 6. Voici un exemple :

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200">Élément 1</div>
    <div class="bg-red-200 row-span-2">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

Figure 8–17
L'élément 2 fusionne sur deux rangées.



Propriété order

Comme pour les boîtes flexibles, la propriété `order` peut être utilisée pour les éléments de grille. Veuillez vous référer au chapitre précédent, page 54, si vous avez besoin d'y revenir.

Propriété justify-self

La propriété `justify-self` aligne un élément de la grille à l'intérieur d'une cellule sur l'axe horizontal. Elle s'exprime par la classe `justify-self-{auto | start | end | center | stretch}` de Tailwind CSS.

Voici un exemple d'alignement centré du premier élément de la grille.

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
    <div class="bg-red-200 justify-self-center">Élément 1</div>
    <div class="bg-red-200">Élément 2</div>
    <div class="bg-red-200">Élément 3</div>
    <div class="bg-red-200">Élément 4</div>
    <div class="bg-red-200">Élément 5</div>
    <div class="bg-red-200">Élément 6</div>
</div>
```

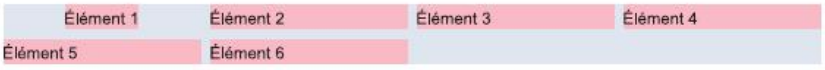


Figure 8–18
L'élément 1 est centré horizontalement.

Propriété align-self

Nous avons déjà vu la propriété `align-self` lors du chapitre précédent. Pour rappel, la classe de Tailwind CSS se présente sous la forme `self-{auto | start | end | center | stretch | baseline}`.

Cette propriété aligne le contenu d'un élément sur l'axe vertical.

Voici un exemple d'utilisation.

```
<div class="container bg-slate-200 grid grid-cols-4 gap-2">
   <div class="bg-red-200 row-span-2 self-center">Élément 1</div>
   <div class="bg-red-200">Élément 2</div>
   <div class="bg-red-200">Élément 3</div>
   <div class="bg-red-200">Élément 4</div>
   <div class="bg-red-200">Élément 5</div>
   <div class="bg-red-200">Élément 6</div>
</div>
```

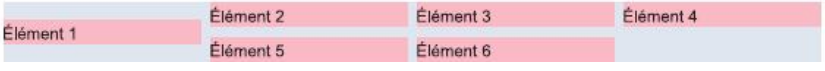


Figure 8–19
L'élément 1 est centré verticalement.

Comme nous venons de le voir dans ce chapitre, le système de grille permet de réaliser des mises en page complexes. De plus, il est tout à fait possible de combiner les grilles et les boîtes flexibles.

Grâce à ces éléments, vous serez capable de réaliser de belles interfaces web qui s'adapteront aux différentes tailles d'écran. C'est d'ailleurs le sujet du prochain chapitre : le responsive design.

9

Responsive design

Le responsive design est aujourd'hui un élément clé de la conception web. En effet, il permet aux sites web de s'adapter automatiquement à différents appareils et tailles d'écran.

Avec l'essor des appareils mobiles, il est désormais essentiel de créer des sites web qui s'adaptent à tous les écrans, afin d'offrir une expérience utilisateur cohérente et optimale.

Grâce à sa vaste bibliothèque de classes prédéfinies, Tailwind CSS nous facilite la création de mises en page réactives, sans avoir besoin d'écrire du CSS personnalisé en fonction des différentes tailles d'écran.

Dans ce chapitre, nous allons explorer les principes de base du responsive design avec Tailwind CSS.

Tailles d'écran et point de rupture

En CSS, nous avons la possibilité de modifier ou de faire évoluer des règles de style en fonction de la taille d'écran, grâce aux requêtes de média ou *media queries*. La requête de média fonctionne de la même manière qu'une condition en programmation. Si une largeur d'écran spécifique est atteinte ou dépassée, alors on pourra ajouter ou modifier des règles de styles CSS. Le fait d'atteindre ou de dépasser une largeur donnée est appelé le point de rupture. C'est le moment où nous pourrons faire évoluer le CSS.

Tailwind CSS s'est inspiré des tailles d'écran les plus courantes pour proposer cinq points de rupture par défaut.

Tableau 9–1 Les points de rupture.

Point de rupture	Largeur minimale
sm	640px
md	768px
lg	1024px
xl	1280px
2xl	1536px

Toutes les classes utilitaires de Tailwind CSS peuvent se voir attribuer un point de rupture à l'aide des préfixes indiqués dans le tableau 9-1, selon la syntaxe suivante : `préfixe:classe` (veillez à ne pas laisser d'espace entre le préfixe, les deux points et la classe utilitaire).

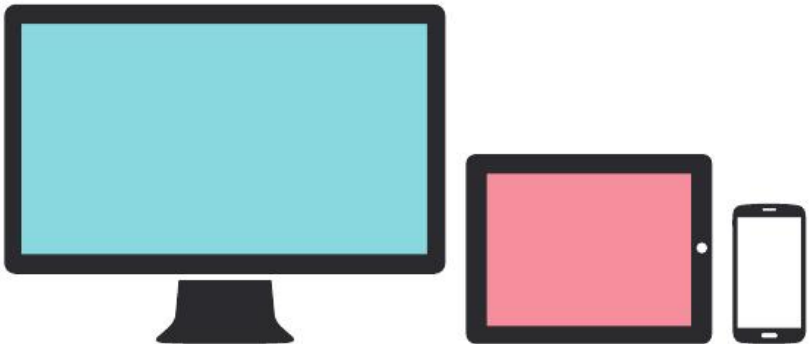
Examinons l'exemple suivant :

```
<body class="md:bg-red-300 lg:bg-cyan-300">
```

Il a été défini deux points de rupture sur l'élément `<body>`. Sur des écrans dont la taille est supérieure ou égale à 768 pixels (`md`), une couleur de fond rouge a été appliquée. La couleur de fond passera au cyan sur des écrans dont la résolution est supérieure ou égale à 1 024 pixels (`lg`).

La figure 9-1 montre le résultat en fonction de la taille d'écran.

Figure 9–1
Affichage de couleurs différentes en fonction de la taille d'écran.



Affichage optimisé pour les mobiles

Sur la figure 9-1, nous constatons que la couleur de fond est blanche sur la version mobile, par défaut.

Le code vu précédemment a ajouté un premier point de rupture pour une taille d'écran supérieure ou égale à 768 pixels. Mais qu'en est-il pour la version mobile dont les résolutions sont inférieures à 640 pixels ?

Tailwind CSS utilise une approche dite *mobile first* comme la plupart des frameworks CSS tels que Bootstrap, pour ne citer que lui. Autrement dit, l'affichage est déjà optimisé pour les

mobiles. Lorsque nous ajoutons une classe (sans préfixe), elle s'affiche sur toutes les tailles d'écran. Si un préfixe est ajouté, alors la classe prend effet à partir du point de rupture spécifié.

Reprenons notre exemple précédent et ajoutons une classe `bg-green-300`.

```
<body class="bg-green-300 md:bg-red-300 lg:bg-cyan-300">
```

Nous avons maintenant un fond vert par défaut, sur petits écrans, qui change de couleur sur les écrans plus grands.

Notez que lorsque nous utilisons ce genre de framework CSS, il est recommandé de concevoir la mise en page de l'écran le plus petit vers le plus grand.

Sur le même principe, et pour une meilleure lisibilité de votre code, ajoutez d'abord les classes utilitaires sans préfixe, puis les classes avec préfixes par ordre de grandeur.

```
<p class="m-2 p-2 text-lg md:m-4 md:p-4 md:text-2xl lg:m-8 lg:p-8 lg:text-
4xl">Lorem ipsum</p>
```

Gérer un intervalle de point de rupture

Toujours sur le même exemple avec les couleurs de fond, imaginons le petit exercice suivant : nous souhaitons maintenant revenir sur la couleur verte pour un écran large (au lieu du cyan). La première idée pourrait être de modifier la couleur au niveau du préfixe `lg`.

```
<body class="bg-green-300 md:bg-red-300 lg:bg-green-300">
```

Cette méthode est correcte, mais sachez qu'il est possible de créer un intervalle avec les modificateurs suivants : `max-sm`, `max-md`, `max-lg`, `max-xl` et `max-2xl`.

Le code suivant montre la syntaxe pour créer un intervalle. La couleur rouge s'applique sur les écrans dont la taille se situe entre 768 et 1 023 pixels.

À partir de 1 024 pixels, la couleur de fond est verte.

```
<body class="bg-green-300 md:max-lg:bg-red-300">
```

Exemples d'utilisation

Maintenant que vous vous êtes familiarisé avec les bases du responsive design avec Tailwind CSS, voyons quelques cas d'utilisation.

Afficher une ou plusieurs colonnes

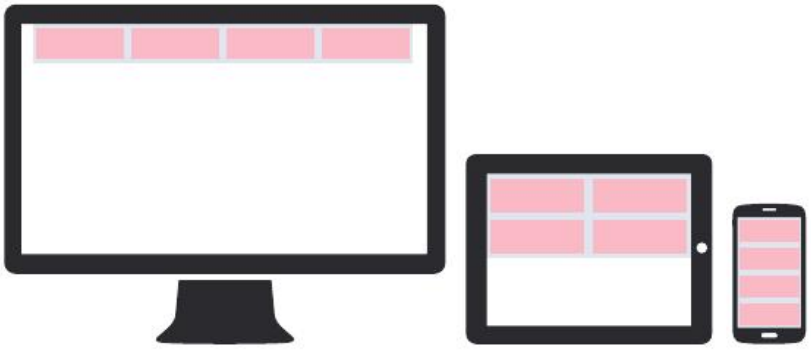
Dans l'exemple suivant, nous utilisons les boîtes flexibles pour gérer l'affichage des colonnes en fonction de la taille d'écran.

Pour commencer, nous effectuons un affichage sur une colonne (par défaut) pour un écran de smartphone. Nous allons ensuite disposer le contenu sur deux colonnes, pour un écran moyen de type tablette, et enfin sur quatre colonnes pour les écrans larges.

```
<div class="container mx-auto bg-slate-200 md:flex md:flex-wrap">
    <div class="md:w-1/2 lg:w-1/4 p-4">
        <div class="bg-red-200 h-32"></div>
    </div>
    <div class="md:w-1/2 lg:w-1/4 p-4">
        <div class="bg-red-200 h-32"></div>
    </div>
    <div class="md:w-1/2 lg:w-1/4 p-4">
        <div class="bg-red-200 h-32"></div>
    </div>
    <div class="md:w-1/2 lg:w-1/4 p-4">
        <div class="bg-red-200 h-32"></div>
    </div>
</div>
```

La figure 9-2 présente l'affichage obtenu dans un navigateur.

Figure 9–2
Disposition des colonnes en fonction de la taille d'écran.



Grille réactive

Dans cet exemple, nous allons utiliser le système de grille. La grille est plus élaborée et change d'aspect en fonction des tailles d'écran.

Les éléments sont disposés les uns en dessous des autres pour l'affichage sur un smartphone. Puis, leur disposition change selon que la largeur d'écran est moyenne ou large.

```
<div class="container mx-auto p-2">
    <div class="grid gap-4 md:grid-cols-3 md:grid-rows-2">
        <div class="p-4 bg-teal-200 md:max-lg:col-span-2 lg:row-span-2">A</
div>
        <div class="p-4 bg-cyan-200 md:row-span-2">B</div>
        <div class="p-4 bg-red-200">C</div>
        <div class="p-4 bg-purple-200">D</div>
    </div>
</div>
```

La figure 9-3 présente les différents aspects de la grille.

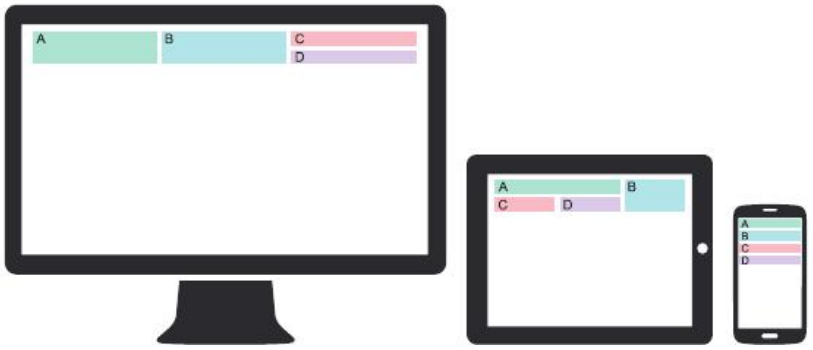


Figure 9–3
Différentes dispositions de la grille en fonction de la taille d'écran.

10

Personnaliser Tailwind CSS

Afin de vous familiariser avec Tailwind CSS, vous avez peut-être effectué une ou plusieurs installations via le CDN, ce qui vous aura permis de réaliser les différents exemples de l'ouvrage.

Il est temps à présent de voir comment nous pouvons personnaliser Tailwind CSS. Pour cela, nous devrons l'installer via la CLI (voir chapitre 2, page 6).

Bien que Tailwind CSS dispose déjà d'un grand nombre de classes utilitaires, le fait de le personnaliser vous permettra d'obtenir de nombreux avantages.

En effet, Tailwind CSS est très flexible et permet de personnaliser de nombreux aspects de la conception (couleurs, typographies, espacements, bordures, etc.). Vous pourrez ainsi créer des styles uniques qui répondront à des besoins spécifiques.

De plus, du point de vue de la performance, Tailwind CSS propose un système de purge qui allège considérablement le fichier CSS, car les classes inutilisées ne sont pas ajoutées à la feuille de styles.

Fichier de configuration

Lors de l'installation, vous avez saisi la commande `npx tailwindcss init` qui va créer le fichier de configuration `tailwind.config.js` situé à la racine du dossier de votre projet.

Ce fichier contient la clé `content` que nous avons déjà étudiée. C'est ici que nous devons indiquer les fichiers qui seront analysés par Tailwind CSS lors de la compilation pour créer la feuille de styles.

Le fichier de configuration doit ressembler à celui-ci :

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./public/**/*.html"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

N'oubliez pas de créer un dossier `public` à la racine de votre projet. La clé `content` indique ici que Tailwind CSS va analyser les fichiers HTML qui se trouvent dans le dossier et les sous-dossiers de `public` en indiquant `/**/`.

Si votre projet comporte en plus des fichiers PHP (ou autres), vous pouvez renseigner la clé `content` de cette manière :

```
content: ["./public/**/*.{html,php}"],
```

Au fur et à mesure du développement d'un projet, nous serons amenés à lancer régulièrement la commande pour compiler le CSS.

Pour lancer plus simplement la commande qui va exécuter la compilation, je vous recommande d'ajouter la clé `scripts` dans le fichier `package.json` qui doit se présenter de la manière suivante :

```
{
  "devDependencies": {
    "tailwindcss": "^3.3.1"
  },
  "scripts": {
    "dev": "npx tailwindcss -i ./src/input.css -o ./public/style.css --
watch"
  }
}
```

En tapant la commande `npm run dev`, la ligne `npx tailwindcss -i ./src/input.css -o ./public/style.css --watch` va s'exécuter pour compiler le CSS.

Cela facilite les choses et évitera les erreurs de saisie à chaque fois que vous aurez besoin de compiler le CSS.

Si vous lancez la commande, le fichier `style.css` est créé dans le répertoire `public`. Sans reprendre les étapes d'installation, n'oubliez pas de lier cette feuille de styles à votre page HTML.

Nous sommes maintenant prêts pour personnaliser Tailwind CSS.

Couleurs

En renseignant le fichier de configuration de Tailwind CSS, nous avons la possibilité de personnaliser notre palette de couleurs.

Ainsi, au sein de la clé `theme`, nous pouvons ajouter une clé `colors` pour définir nos couleurs.

```
  theme: {
    colors: {
      'chocolate': '#D2691E',
      'brown': '#A52A2A'
    },
    extend: {}
  },
…
```

Dans cet exemple, nous avons ajouté les couleurs `chocolate` et `brown`. Nous pourrons ensuite les utiliser comme classes utilitaires `text-chocolate`, `bg-brown`…

Mais attention, en procédant ainsi, la palette de Tailwind CSS est écrasée. Vous n'avez donc plus accès à l'ensemble des couleurs de base qu'il propose.

Pour y remédier, il est préférable d'ajouter le contenu de la clé `colors` à l'intérieur de la clé `extend`. Cette dernière permet d'ajouter ou de mettre à jour des valeurs sans pour autant écraser les autres.

```
theme: {
    extend: {
        colors: {
        'chocolate': '#D2691E',
        'brown': '#A52A2A'
      }
    }
},
…
```

Il est également possible de définir différentes nuances pour une couleur, comme dans l'exemple suivant :

```
        'rouge': {
        50: '#fff0f0',
        100: '#ffdddd',
        200: '#ffc0c0',
        300: '#ff9494',
        400: '#ff5757',
        500: '#ff2323',
        600: '#ff0000',
        700: '#d70000',
        800: '#b10303',
        900: '#920a0a',
        950: '#500000',
        }
```

Typographie

Nous avons vu au chapitre 4 que Tailwind CSS propose les classes `font-sans`, `font-serif` et `font-mono` et qu'il est possible de personnaliser une police grâce aux valeurs abstraites.

Cette fois, nous allons utiliser le fichier de configuration pour personnaliser les polices. Si vous avez repéré une ou plusieurs polices qui vous plaisent dans celles proposées sur Google Fonts, voici comment les mettre en place.

Nous prenons ici pour exemple la police Open Sans et copions dans notre page HTML les balises `link` fournies par Google Fonts.

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/
css2?family=Open+Sans:ital,wght@0,300;0,400;0,500;0,600;0,700;0,800;1,300;1
,400;1,500;1,600;1,700;1,800&display=swap" rel="stylesheet">
```

Dans le fichier de configuration, il suffit d'ajouter la clé `fontFamily` comme ceci :

```
      fontFamily: {
        'open-sans': ["'Open Sans'", 'sans-serif']
      }
```

Veillez à bien ajouter des guillemets si le nom de la police se compose de plusieurs mots.

Vous pouvez également échapper l'espace :

```
['Open\\ Sans', …]
```

Nous pouvons désormais utiliser la classe `font-open-sans` et l'ajouter à l'élément `<body>` pour que la police s'affiche sur l'ensemble de la page.

Dans la section suivante, nous verrons qu'il est également possible de l'intégrer dans une feuille de styles avec votre CSS personnalisé.

Voici d'autres clés que vous pouvez ajouter au fichier de configuration si vous souhaitez personnaliser d'autres propriétés CSS concernant la typographie :

- `fontSize` : pour gérer la taille de la police ;
- `fontWeight` : pour gérer la graisse de la police ;
- `letterSpacing` : pour gérer l'espacement des lettres ;
- `lineClamp` : pour tronquer du texte sur un certain nombre de lignes ;
- `lineHeight` : pour gérer l'interlignage ;
- `textIndent` : pour gérer l'indentation du texte.

Espacements

Pour personnaliser ou redéfinir les valeurs des espaces, il suffit d'ajouter la clé `spacing` au thème du fichier de configuration.

Notez qu'il est possible de redéfinir complètement les valeurs en écrasant le thème par défaut comme dans l'exemple suivant :

```
theme: {
  spacing: {
    '1': '8px',
    '2': '12px',
    '3': '16px',
    '4': '24px',
    '5': '32px',
    '6': '48px',
  }
}
```

Nous pouvons également ajouter des nouvelles valeurs, en plus de celles déjà définies par Tailwind CSS. Pour cela, nous ajouterons la clé `spacing` à l'intérieur de `extend`.

```
theme: {
  extend: {
    spacing: {
      '13': '3.25rem',
      '15': '3.75rem',
      '128': '32rem',
      '144': '36rem',
    }
  }
}
```

Ces espacements seront alors accessibles pour modifier les classes `margin`, `padding`, `width`, `height`...

Personnaliser les autres propriétés

Il est possible de personnaliser n'importe quelle propriété du thème Tailwind.

Pour des propriétés telles que `border-radius` ou `box-shadow`, vous devez utiliser l'écriture en Camel Case, qui consiste à écrire les mots sans espaces. Chaque mot commence par une lettre majuscule, à part le premier. Par exemple, la propriété « border-radius » s'écrira « borderRadius ».

Ainsi, les clés s'écriront de la manière suivante dans le fichier de configuration de Tailwind CSS :

```
extend: {
   borderRadius: {},
   boxShadow: {},
}
```

Plugins

Tailwind CSS permet d'ajouter des plugins. Parmi les plugins officiels, nous prendrons ici comme exemple celui des formulaires.

En effet, il n'est pas toujours aisé de mettre en forme un formulaire. L'affichage peut varier d'un navigateur à un autre et les champs de type `input` et `select` ne sont pas toujours homogènes.

Le plugin de formulaire peut être intéressant si vous ne souhaitez pas perdre trop de temps à mettre en forme vos formulaires. Il appliquera un style minimal qu'il vous suffira d'améliorer ensuite avec les classes de Tailwind CSS.

Pour le mettre en place, saisissez la commande suivante :

```
npm install -D @tailwindcss/forms
```

Ajoutez ensuite une clé `plugins` dans le fichier `tailwind.config.js` :

```
module.exports = {
  theme: {
    // ...
  },
  plugins: [
    require('@tailwindcss/forms'),
    // ...
  ],
}
```

Les éléments du formulaire sont maintenant réinitialisés et normalisés avec des styles simples par défaut. Il ne vous reste plus qu'à les personnaliser avec les classes utilitaires.

Fichier CSS personnalisé

En plus de pouvoir personnaliser un certain nombre d'éléments dans le fichier de configuration de Tailwind CSS, nous pouvons bien évidemment ajouter un fichier CSS personnalisé.

Pour cela, nous l'ajouterons dans le fichier `src/input.css` que nous avons créé lors de l'installation. Ce dernier contient actuellement les trois directives suivantes :

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Il est tout à fait possible de créer nos propres classes et d'écrire notre CSS. Voici un exemple :

```
.title {
  font-size: 24px;
  color: #3b82f6;
}
```

Cependant, la bonne pratique veut que l'on ajoute la directive `@layer` pour ajouter du style aux couches `base`, `components` et `utilities` de Tailwind CSS.

```
@layer base {
    .title {
      @apply text-2xl text-blue-500;
    }
}
```

La directive `@apply` permet d'intégrer les classes utilitaires de Tailwind CSS dans votre propre feuille de styles personnalisée.

Nous pouvons en profiter pour ajouter la police que nous avons personnalisée dans la section précédente.

```
@layer base {
    body {
      @apply font-open-sans;
    }
    .title {
      @apply text-2xl text-blue-500;
    }
}
```

Voici un exemple de l'utilisation des différentes couches avec la directive `@layer` :

```
@layer base {
  h1 {
    @apply text-2xl;
  }
  h2 {
    @apply text-xl;
  }
}
@layer components {
  .btn-blue {
    @apply bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded;
  }
}
@layer utilities {
  .filter-none {
    filter: none;
  }
  .filter-grayscale {
    filter: grayscale(100%);
  }
}
```

La couche `base` permet de réinitialiser des règles ou d'appliquer des styles aux éléments HTML de base.

La couche `components` est prévue pour définir des composants réutilisables tels que les boutons, les cartes, les barres de navigation…

La couche `utilities`, quant à elle, définit les classes utilitaires qui viennent affiner des réglages concernant les marges, les couleurs, la taille du texte…

Par ailleurs, la directive `@layer` dans le fichier source présente plusieurs avantages. En effet, elle permet de réorganiser les différentes parties de votre code CSS plus facilement en les classant par fonctionnalité. Votre code sera ainsi mieux géré, car vous pourrez aisément déplacer et regrouper vos classes.

Par ailleurs, lorsque vous ajoutez votre propre CSS à un fichier Tailwind, le code peut être répété plusieurs fois. En utilisant la directive `@layer`, Tailwind CSS peut regrouper toutes les classes de la même fonctionnalité dans un seul bloc, ce qui permet d'optimiser la taille du fichier CSS final.

Enfin, il sera facile de maintenir le code CSS à mesure que l'application évolue. Si vous devez ajouter de nouvelles fonctionnalités ou supprimer des éléments existants, vous pourrez le faire rapidement et simplement, sans avoir à passer en revue l'ensemble du fichier CSS.

En résumé, le fait d'utiliser la directive `@layer` permet de mieux organiser, optimiser et maintenir le code CSS avec Tailwind.

11

Cas pratique : le projet « Burger Xpress »

Il est maintenant temps de mettre en pratique tout ce que nous avons appris dans cet ouvrage.

Pour cela, nous allons réaliser, étape par étape, l'intégration HTML/CSS d'une maquette. Vous pourrez télécharger les images et le code source sur le site des Éditions Eyrolles : https://www.editions-eyrolles.com/dl/0101421

Ces éléments seront également disponibles via le lien GitHub suivant : https://github.com/sausinfr/Burger-Xpress-Tailwind-CSS

Présentation du projet

Le projet consiste à réaliser la page d'accueil d'une entreprise fictive, « Burger Xpress », qui propose un service de restauration rapide.

La figure 11-1, page suivante, présente la maquette dont nous allons réaliser l'intégration.

Notez que cette intégration proposera une certaine façon de faire, mais plusieurs méthodes permettent d'arriver à un même résultat. Vous êtes donc libre de travailler comme vous le souhaitez et d'explorer votre propre méthode.

Nous allons ici découper la page en plusieurs sections :

- une section « Hero », qui comprend la barre de navigation et affiche un visuel de hamburger ;
- une section « Nos suggestions » sur trois colonnes ;

- une section « Newsletter », qui comporte un formulaire ;
- une section « Témoignages » sur quatre colonnes ;
- un pied de page sur trois colonnes.

Bien entendu, nous allons gérer un affichage responsive design. Les éléments en colonnes viendront s'afficher les uns en dessous des autres pour la version mobile.

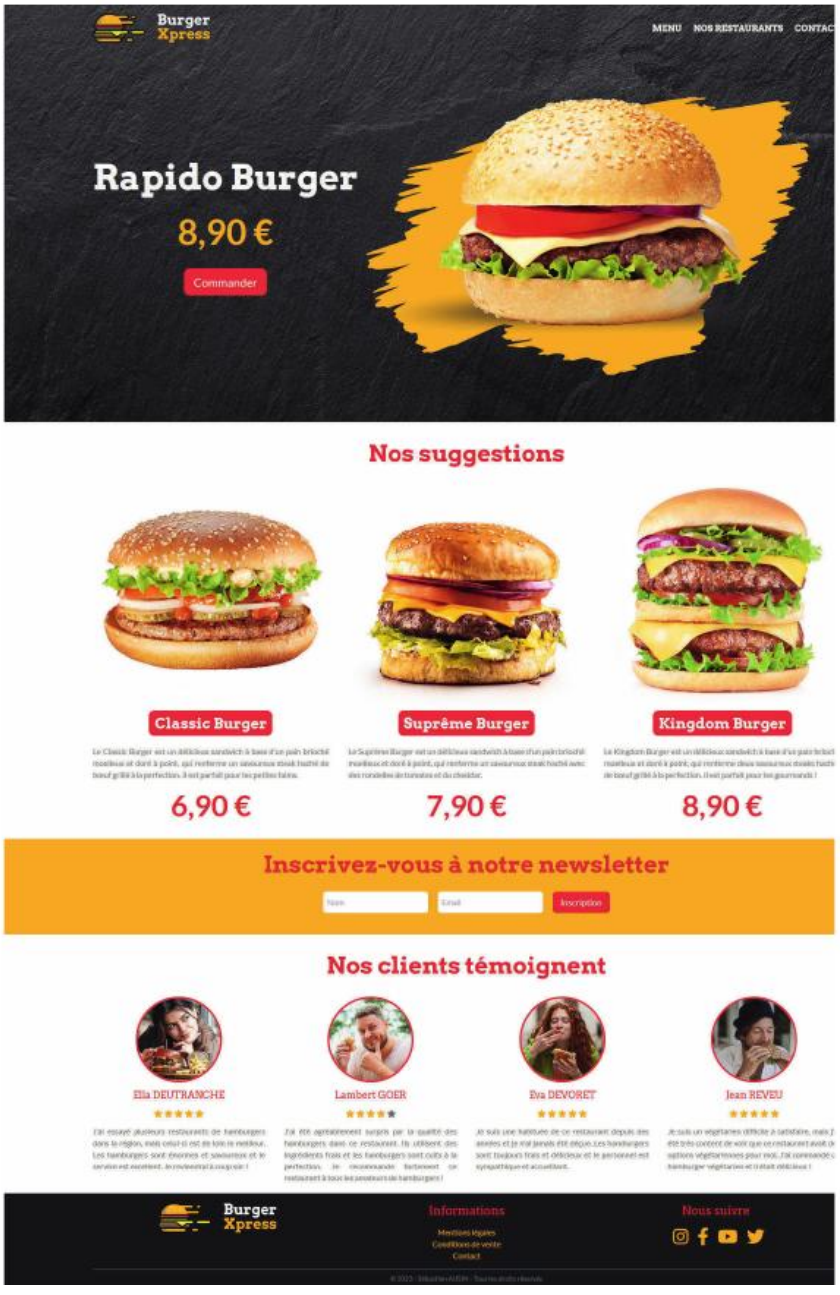


Figure 11–1 Maquette du projet « Burger Xpress ».

Mise en place du projet

Installation de Tailwind CSS

Nous allons passer rapidement sur cette étape. Si besoin, vous pouvez relire le chapitre 2 consacré à l'installation de Tailwind CSS.

1. Créez le dossier de votre projet et ouvrez-le avec Visual Studio Code.
2. Ouvrez le terminal et saisissez les commandes suivantes :

```
npm install -D tailwindcss
npx tailwindcss init
```

3. Créez un dossier `public`, puis ajoutez la clé suivante dans votre fichier `tailwind.config.js`.

```
content: ["./public/*.html"],
```

4. Dans le dossier `public`, ajoutez un fichier `index.html` qui aura la structure suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Burger Xpress</title>
</head>
<body>

</body>
</html>
```

5. Ajoutez ensuite un dossier `src`, à la racine du projet, dans lequel nous allons créer le fichier `input.css` qui contiendra les directives de Tailwind CSS :

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

6. Effectuez une compilation à l'aide de la commande suivante :

```
npx tailwindcss -i ./src/input.css -o ./public/style.css
```

Si tout s'est bien passé, le fichier `style.css` a été créé dans le dossier `public`.

7 Vous pouvez maintenant lier votre feuille de styles à votre fichier `index.html` :

```
<link rel="stylesheet" href="style.css">
```

Il est conseillé d'ajouter la clé suivante dans le fichier `package.json` :

```
  "scripts": {
    "dev": "npx tailwindcss -i ./src/input.css -o ./public/style.css --
watch"
  }
```

De cette manière, il vous suffira de saisir la commande `npm run dev` pour compiler.

L'installation de Tailwind est terminée, passons à la suite.

Réglages du thème

Selon la charte graphique, nous utiliserons les polices suivantes :

- Lato, pour le corps du texte et les prix ;
- Arvo, pour les niveaux de titres.

1 Rendez-vous sur le site de Google Fonts (https://fonts.google.com), sélectionnez ces polices et veillez à activer Regular et Bold, cela suffira pour ce projet.

Vous obtenez alors le code suivant :

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/
css2?family=Arvo:wght@400;700&family=Lato:wght@400;700&display=swap"
rel="stylesheet">
```

2 Ajoutez-le au fichier `index.html`.

3 Côté CSS, ajoutez le code suivant dans le fichier `src/input.css` :

```
@layer base {
    body {
        font-family: 'Lato', sans-serif;
    }
    h1, h2, h3, h4, .arvo {
        font-family: 'Arvo', serif;
    }
}
```

Les polices sont prêtes à l'emploi, nous allons maintenant configurer les couleurs de la charte graphique en définissant une couleur primaire et une couleur secondaire.

4 Ajoutez une clé `colors` dans le fichier `tailwind.config.js` qui se présente comme ceci :

```
theme: {
  extend: {
    colors: {
      primary:"#E7272D",
      secondary:"#FBB217",
    },
  },
},
```

5 Ajoutez également une clé `backgroundImage`, juste après la clé `colors`. C'est ici que vous indiquerez une image de fond pour la section Hero.

```
    backgroundImage: {
      "ardoise": "url('images/background-slate.jpg')",
    },
```

Bien entendu, veillez à bien ajouter le dossier `images` dans le dossier `public`.

Intégration de la maquette

Après avoir effectué quelques réglages concernant la charte graphique, nous allons commencer notre intégration HTML/CSS.

Section Hero et barre de navigation

Nous allons ici ajouter une balise `<section>` dans laquelle nous placerons la barre de navigation et une bannière « tape à l'œil ». Cette section, appelée Hero, doit inciter l'internaute à aller plus loin sur le site.

En voici le code :

```
<!-- section hero -->
<section class="bg-ardoise bg-cover bg-center p-5">
    <header>
        <div class="container mx-auto flex flex-col items-center space-y-6
md:flex-row md:justify-between md:space-y-0">
            <div class="w-60">
                <a href="index.html">
                    <img src="images/logo.png" alt="Burger Xpress">
                </a>
            </div>
            <ul class="arvo text-white text-lg uppercase font-bold space-y-2
md:flex md:space-x-6 md:space-y-0">
                <li><a class="menu-link" href="#">Menu</a></li>
```

```
                <li><a class="menu-link" href="#">Nos restaurants</a></li>
                <li><a class="menu-link" href="#">Contact</a></li>
            </ul>
        </div>
    </header>
    <div class="container mx-auto flex flex-col space-y-8 md:flex-row md:py-
16">
        <div class="flex flex-col items-center justify-center space-y-10
order-2 md:order-1">
            <h1 class="text-white text-7xl font-bold text-center">Rapido
Burger</h1>
            <p class="text-secondary text-7xl font-bold">8,90 €</p>
            <a class="btn-lg" href="#">Commander</a>
        </div>
        <div class="order-1 md:order-2">
            <img src="images/rapido-burger.png" alt="Rapido Burger">
        </div>
    </div>
</section>
```

Au niveau de la balise `<section>`, nous avons ajouté la classe `bg-ardoise`. Il s'agit de l'image de fond qui a été paramétrée dans le fichier de configuration de Tailwind CSS. Cette image vient couvrir la surface de la section (`bg-cover`) et est centrée (`bg-center`).

À l'intérieur de la section, nous ajoutons l'en-tête `<header>` puis un élément `<div>` de classe `container`. Ce dernier utilise une boîte flexible pour disposer le logo et les liens du menu.

Notez que, sur la ligne suivante, nous utiliserons d'abord une approche mobile first pour aller vers des largeurs d'écran plus élevées.

```
<div class="container mx-auto flex flex-col items-center space-y-6
md:flex-row md:justify-between md:space-y-0">
```

Sur mobile, nous utilisons `flex-col` pour une disposition en colonnes, puis `md:flex-row` pour une disposition en lignes, pour des largeurs supérieures ou égales à 768 pixels.

La classe `space-y-6` répartit les éléments sur l'axe vertical. Une fois la disposition en ligne, nous l'annulons avec `md:space-y-0` pour éviter des décalages indésirables.

Les liens du menu sont réalisés à l'aide d'une liste. La balise `<ul>` sert de flex container pour disposer les éléments. Nous avons recours aux classes `space-y-[valeur]` et `space-x-[valeur]` pour répartir les éléments.

Pour la balise `<a>`, nous avons créé une classe `menu-link` afin d'éviter des répétitions et de surcharger inutilement la balise de classes utilitaires.

Nous ajouterons donc une couche `components` dans le fichier `src/input.css` :

```
@layer components {
    .menu-link {
        @apply transition duration-300 ease-in-out hover:text-secondary;
    }
}
```

Après avoir réalisé l'en-tête, nous ajoutons du contenu avec le nom du hamburger, le prix, le bouton Commander et une photo.

La subtilité ici consiste à changer l'ordre des éléments flexibles. En effet, sur smartphone, nous affichons d'abord l'image du hamburger (`order-1`), puis le contenu (`order-2`). Sur des écrans plus larges, le flux normal affiche d'abord le contenu (`md:order-1`), puis l'image (`md:order-2`).

Nous avons créé une classe `btn-lg` pour le bouton, que nous devons déclarer dans le fichier `src/input.css`, dans la couche `components` à la suite de `.menu-link`.

```
.btn-lg {
      @apply text-white bg-primary text-2xl px-5 py-3 rounded-xl transition
duration-300 ease-in-out hover:bg-secondary hover:text-primary;
}
```

Si vous souhaitez par la suite modifier l'apparence du bouton, par exemple sa couleur, il sera plus facile d'effectuer la modification ici plutôt que dans les pages HTML en recherchant les occurrences.

Section Nos suggestions

Dans cette section, notre entreprise fictive Burger Xpress veut suggérer à ses clients trois types de hamburgers.

Pour la version mobile, les produits seront affichés les uns en dessous des autres en suivant le flux normal du HTML. Pour les écrans plus larges, d'une taille supérieure ou égale à 1 024 pixels, nous utiliserons une boîte flexible pour disposer les produits sur une ligne de trois colonnes.

Voici le code de cette section :

```
<!-- section suggestions -->
<section class="p-5">
    <h2>Nos suggestions</h2>
    <div class="container mx-auto lg:flex lg:space-x-10">
        <div class="text-center lg:w-1/3">
          <img class="mx-auto" src="images/classic-burger.jpg" alt="Classic
Burger">
           <h3 class="bg-primary text-3xl font-bold text-white inline-block
px-3 py-2 rounded-xl mb-5">Classic Burger</h3>
```

```
            <p class="text-justify">Le Classic Burger est un délicieux
sandwich à base d'un pain brioché mœlleux et doré à point, qui renferme un
savoureux steak haché de bœuf grillé à la perfection. Il est parfait pour les
petites faims.</p>
            <p class="text-6xl text-primary font-bold py-5">6,90 €</p>
        </div>
        <div class="text-center lg:w-1/3">
          <img class="mx-auto" src="images/supreme-burger.jpg" alt="Suprême
Burger">
           <h3 class="bg-primary text-3xl font-bold text-white inline-block
px-3 py-2 rounded-xl mb-5">Suprême Burger</h3>
            <p class="text-justify">Le Suprême Burger est un délicieux
sandwich à base d'un pain brioché mœlleux et doré à point, qui renferme un
savoureux steak haché avec des rondelles de tomates et du cheddar.</p>
            <p class="text-6xl text-primary font-bold py-5">7,90 €</p>
        </div>
        <div class="text-center lg:w-1/3">
          <img class="mx-auto" src="images/kingdom-burger.jpg" alt="Kingdom
Burger">
           <h3 class="bg-primary text-3xl font-bold text-white inline-block
px-3 py-2 rounded-xl mb-5">Kingdom Burger</h3>
            <p class="text-justify">Le Kingdom Burger est un délicieux
sandwich à base d'un pain brioché mœlleux et doré à point, qui renferme deux
savoureux steaks hachés de bœuf grillé à la perfection. Il est parfait pour
les gourmands !</p>
            <p class="text-6xl text-primary font-bold py-5">8,90 €</p>
        </div>
    </div>
</section>
```

Nous allons définir un léger gris pour la couleur par défaut du texte. Pour cela, il suffit d'ajouter une directive `@apply text-neutral-500;` au niveau de la balise `<body>` de la couche `base` du fichier `src/input.css`.

Toujours dans la couche `base`, nous allons également définir la balise `<h2>` comme ceci :

```
h2 {
       @apply text-3xl font-bold text-center text-primary py-5 md:text-5xl;
}
```

Sur la version mobile, les titres de niveau 2 ont une taille de 30 pixels (`text-3xl`). Sur des écrans plus larges, la taille du texte est fixée à 48 pixels (`md:text-5xl`). Les titres seront centrés (`text-center`) et de couleur rouge (`text-primary`).

Dans cette section, la syntaxe n'est pas compliquée. Pour disposer nos colonnes, nous appliquerons un `display: flex` pour les écrans de taille supérieure ou égale à 1 024 pixels.

Au niveau de notre flex container, nous ajoutons donc les classes `lg:flex lg:space-x-10`. Rappelons que `space-x-10` vient ajouter des espaces entre les colonnes.

Les éléments `<div>`, les éléments enfants de notre flex container, se verront ajouter la classe `lg:w-1/3` pour obtenir une largeur de colonne de 33,33 %.

Section Newsletter

Cette section nous permettra de réaliser un petit formulaire dont voici le code :

```
<section class="bg-secondary px-5 pt-2 pb-8">
    <h2>Inscrivez-vous à notre newsletter</h2>
    <form class="py-3 flex flex-col space-y-4 md:flex-row md:space-y-0
md:space-x-5 md:justify-center" action="">
        <input class="p-2 rounded-md" type="text" name="name" id="name"
placeholder="Nom">
        <input class="p-2 rounded-md" type="email" name="email" id="email"
placeholder="E-mail">
        <button class=" text-white bg-primary text-lg px-4 py-2 rounded-lg"
type="submit">Inscription</button>
    </form>
</section>
```

La section se voit appliquer une couleur de fond à l'aide de la classe `bg-secondary`. Rappelez-vous, la couleur secondaire a été définie dans notre fichier `tailwind.config.js`.

Nous n'avons pas besoin d'ajouter de classe au niveau de la balise `<h2>` dont le style a été ajouté précédemment dans le fichier `input.css`.

Là encore, nous aurons recours aux boîtes flexibles pour positionner les éléments du formulaire. La balise `<form>` sera notre flex container, tandis que les balises `<input>` et `<button>` seront les flex items.

Nous affichons les éléments les uns en dessous des autres dans la version mobile `flex flex-col`. Nous utilisons la classe `space-y-4` pour ajouter un peu d'espace en hauteur entre les éléments.

À partir d'une taille d'écran moyenne, nous disposons les éléments en ligne grâce à `md:flex-row`. Nous annulons les espaces en hauteur définis pour la version mobile `md:space-y-0`. Nous ajoutons des espaces entre les éléments sur l'axe horizontal avec `md:space-x-5`.

Section Témoignages

Cette section permet d'afficher les témoignages des clients. Sur grand écran, nous les présenterons sur quatre colonnes tandis que, sur un écran de type tablette, ils seront affichés sur deux colonnes. Sur la version smartphone, les témoignages seront placés les uns en dessous des autres sur une seule colonne.

Pour changer un peu notre manière de faire, nous utiliserons cette fois le système de grille. Cela vous permettra de mettre en pratique une autre technique de mise en page.

Analysons le code de cette section :

```
<section class="p-5">
    <h2>Nos clients témoignent</h2>
    <div class="mt-5 container mx-auto grid grid-cols-1 gap-10 md:grid-cols-
2 lg:grid-cols-4">
        <div class="text-center">
            <img class="rounded-full w-1/2 mx-auto border-solid border-4
border-primary" src="images/avatar1.jpg" alt="Ella DEUTRANCHE">
            <h3 class="text-primary text-xl my-3">Ella DEUTRANCHE</h3>
            <p>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
            </p>
          <p class="mt-3 text-justify">J'ai essayé plusieurs restaurants de
hamburgers dans la région, mais celui-ci est de loin le meilleur. Les
hamburgers sont énormes et savoureux et le service est excellent. Je
reviendrai à coup sûr !</p>
        </div>
        <div class="text-center">
            <img class="rounded-full w-1/2 mx-auto border-solid border-4
border-primary" src="images/avatar2.jpg" alt="Lambert GOER">
            <h3 class="text-primary text-xl my-3">Lambert GOER</h3>
            <p>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star"></i>
            </p>
          <p class="mt-3 text-justify">J'ai été agréablement surpris par la
qualité des hamburgers dans ce restaurant. Ils utilisent des ingrédients
frais et les hamburgers sont cuits à la perfection. Je recommande fortement
ce restaurant à tous les amateurs de hamburgers !</p>
        </div>
        <div class="text-center">
            <img class="rounded-full w-1/2 mx-auto border-solid border-4
border-primary" src="images/avatar3.jpg" alt="Eva DEVORET">
            <h3 class="text-primary text-xl my-3">Eva DEVORET</h3>
            <p>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
            </p>
```

```
            <p class="mt-3 text-justify">Je suis une habituée de ce
restaurant depuis des années et je n'ai jamais été déçue. Les hamburgers
sont toujours frais et délicieux et le personnel est sympathique et
accueillant.</p>
        </div>
        <div class="text-center">
            <img class="rounded-full w-1/2 mx-auto border-solid border-4
border-primary" src="images/avatar4.jpg" alt="Jean REVEU">
            <h3 class="text-primary text-xl my-3">Jean REVEU</h3>
            <p>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
                <i class="fa-solid fa-star text-secondary"></i>
             </p>
             <p class="mt-3 text-justify">Je suis un végétarien difficile à
satisfaire, mais j'ai été très content de voir que ce restaurant avait des
options végétariennes pour moi. J'ai commandé un hamburger végétarien et il
était délicieux !</p>
         </div>
    </div>
</section>
```

Pour mettre en place notre système de grille, nous avons besoin d'un grid container et grid items. Le code suivant spécifie le grid container :

```
<div class="mt-5 container mx-auto grid grid-cols-1 gap-10 md:grid-cols-2
lg:grid-cols-4">
```

La syntaxe est simple, la classe `grid` met en place le système de grille. Il suffit ensuite de préciser le nombre de colonnes souhaité à l'aide des classes suivantes :

- `grid-cols-1` dispose les éléments sur une colonne pour les petits écrans tels que les smartphones ;
- `md:grid-cols-2` affiche les éléments sur deux colonnes pour les écrans de tailles moyennes ;
- `lg:grid-cols-4` affiche les éléments sur quatre colonnes pour les grands écrans.

Les grid items se composent d'éléments `<div>` dans lesquels on ajoute le contenu concernant les témoignages clients (image, nom du client, notation et commentaire).

Nous n'allons pas nous attarder sur les classes concernant le contenu, ce sont des choses que nous avons déjà vues. En revanche, certains d'entre vous se demanderont peut-être comment ajouter les icônes.

Pour ajouter les icônes sous forme d'étoiles, nous utiliserons la bibliothèque Font Awesome, disponible à l'adresse suivante : https://fontawesome.com/.

Notez que son utilisation nécessite de vous inscrire sur le site. Si vous souhaitez éviter cela, récupérez la bibliothèque d'icônes via un CDN, par exemple le site CDNJS : https://cdnjs.com/libraries/font-awesome.

Pour ajouter la bibliothèque à votre page HTML, il suffit de copier le code fourni dans le `<head>` de votre page.

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" integrity="sha512-iecdLmaskl7CVkqkXNQ/ZH/
XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw=="
crossorigin="anonymous" referrerpolicy="no-referrer" />
```

Vous pouvez maintenant chercher des icônes sur le site de Font Awesome et recopier le code HTML de l'icône souhaitée.

Dans notre exemple, il s'agit de l'icône `star`.

```
<i class="fa-solid fa-star text-secondary"></i>
```

Nous ajoutons la classe `text-secondary` pour mettre notre icône en jaune.

Pied de page

Nous avons pratiquement terminé notre intégration, il nous reste uniquement le pied de page à réaliser.

Il n'y a pas de nouveautés ici, étant donné que nous avons vu la plupart des cas précédemment.

Voici le code du pied de page :

```
<footer class="bg-neutral-900 pt-5">
    <div class="container mx-auto">
        <div class="flex flex-col gap-y-10 md:flex-row">
            <div class="md:w-1/3">
                <img class="mx-auto" src="images/logo.png" alt="Burger
Xpress">
            </div>
            <div class="md:w-1/3 text-center">
                <h3 class="text-2xl text-primary mb-4">Informations</h3>
                <ul>
                   <li><a class="text-secondary" href="#">Mentions légales</
a></li>
                    <li><a class="text-secondary" href="#">Conditions de
vente</a></li>
                    <li><a class="text-secondary" href="#">Contact</a></li>
                </ul>
            </div>
```

```
            <div class="md:w-1/3 text-center">
                <h3 class="text-2xl text-primary mb-4">Nous suivre</h3>
                <p class="text-4xl">
                    <a href="#" class="text-secondary ml-3"><i class="fa-
brands fa-instagram"></i></a>
                    <a href="#" class="text-secondary ml-3"><i class="fa-
brands fa-facebook-f"></i></a>
                    <a href="#" class="text-secondary ml-3"><i class="fa-
brands fa-youtube"></i></a>
                    <a href="#" class="text-secondary ml-3"><i class="fa-
brands fa-twitter"></i></a>
                </p>
            </div>
        </div>
        <hr class="mt-4 border-1 border-slate-500">
        <p class="text-center text-sm py-2">&copy; 2023 - Sébastien AUSIN -
Tous les droits réservés</p>
    </div>
</footer>
```

Nous utilisons ici aussi les boîtes flexibles pour mettre en place nos colonnes.

Nous affichons une colonne sur smartphone avec la classe `flex-col`, puis nous affichons les trois colonnes pour les écrans de tailles moyennes et supérieures grâce à `md:flex-row`.

Nous appliquons une dimension aux flex items avec `md:w-1/3`.

N'oubliez pas de récupérer les icônes des réseaux sociaux sur le site de Font Awesome.

Nous en avons maintenant terminé avec l'intégration de cette maquette. J'espère qu'elle vous aura plu et qu'elle vous aura permis de valider vos acquis concernant l'apprentissage de Tailwind CSS.

Pour terminer cet ouvrage, nous allons voir quelques ressources supplémentaires qui pourront vous aider dans votre production.

12

Ressources supplémentaires

La communauté d'utilisateurs de Tailwind CSS étant de plus en plus importante, il existe sur le Web de nombreuses ressources pour gagner en efficacité.

Dans ce chapitre, vous trouverez une liste d'outils complémentaires qui pourront vous être utiles lors de vos conceptions web.

Thèmes et composants

Si vous ne souhaitez pas partir de zéro, vous trouverez sur le Web des thèmes et des composants qui vous permettront de démarrer un projet rapidement.

Tailwind UI

Bien que Tailwind CSS soit *open source*, vous trouverez sur le site officiel des composants et des thèmes payants.

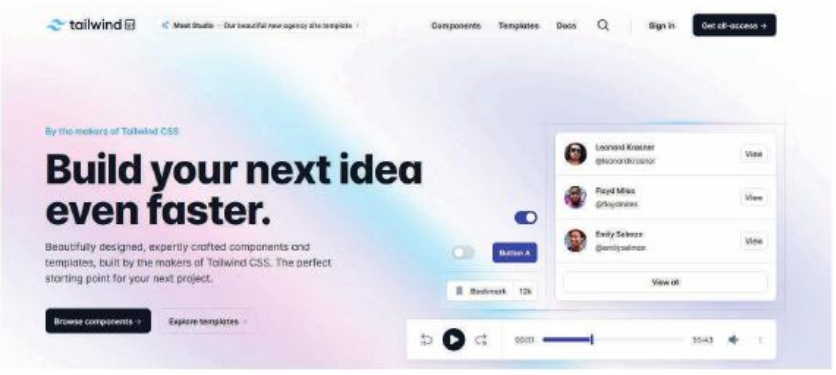
Taiwind UI est accessible à l'adresse suivante : https://tailwindui.com/.

Les thèmes proposés peuvent être un bon point de départ pour votre projet. Il vous en coûtera 89 € pour un thème.

Si vous le souhaitez, vous pouvez acquérir tous les composants et tous les thèmes au prix de 249 € (ou 749 € pour une agence). Il s'agit d'un paiement unique pour un accès à vie.

Notez que certains composants sont gratuits. Vous pouvez récupérer le code au format HTML, React ou Vue.

Figure 12–1
Capture d'écran du site Tailwind UI.



TailGrids

TailGrids propose également de nombreux composants et thèmes. Vous disposerez d'un grand nombre d'éléments pour créer toutes sortes de sites : landing pages, boutiques en ligne, espaces d'administration avec tableau de bord, etc.

Il est accessible à l'adresse suivante : https://tailgrids.com/.

Figure 12–2
Capture d'écran du site TailGrids.

Après avoir installé un projet Tailwind CSS, vous pouvez installer les paquets de TailGrids avec la commande suivante :

```
npm i tailgrids
```

Ensuite, ajoutez TailGrids en tant que plugin dans le fichier `tailwind.config.js` comme ceci :

```
plugins: [require("tailgrids/plugin")],
```

Il ne vous reste plus qu'à parcourir les composants, à sélectionner ceux qui vous intéressent et à recopier le code dans votre projet.

Preline UI

Preline UI dispose d'une vaste bibliothèque de composants construite avec Tailwind CSS. Cerise sur le gâteau, le kit est open source. Vous allez pouvoir profiter gratuitement de tous ces composants pour vos projets personnels ou professionnels.

La documentation est bien renseignée et rappelle celle du site de Tailwind CSS.

Preline UI dispose de plus de 300 exemples de composants fournissant une collection complète d'éléments d'interface utilisateur. Ces composants, entièrement personnalisables, répondront à tous vos besoins en matière de développement de sites web.

Une autre caractéristique de Preline UI est que les composants incluent une variante pour le mode sombre. Cela vous permettra de styliser sans effort votre site web pour répondre aux besoins de vos utilisateurs qui pourront choisir entre une interface claire ou sombre.

Preline UI est compatible HTML, React et Vue.

Vous pouvez parcourir le site via l'adresse suivante : https://preline.co/.

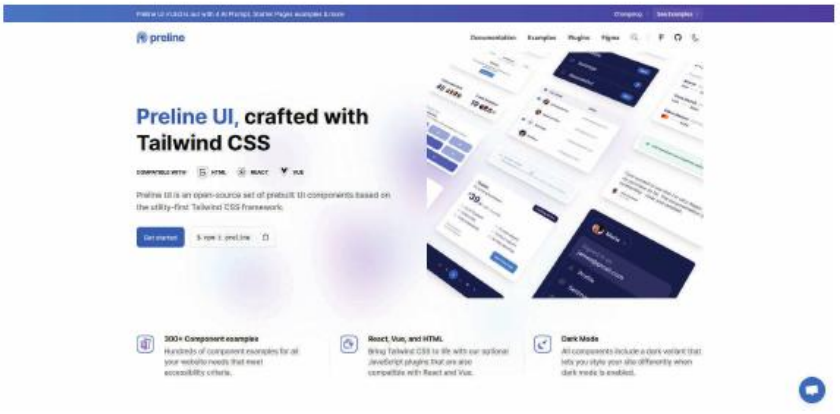


Figure 12–3
Capture d'écran du site Preline UI.

Même les designers UI y trouveront leur compte. En effet, Preline UI propose des sources pour Figma, permettant le prototypage d'applications ou de sites web.

Mais ce n'est pas tout ! En plus de la gamme étendue de composants, Preline UI fournit également des pages de démarrage et des exemples.

Vous pouvez démarrer rapidement un projet en choisissant parmi les exemples : sections Hero, tables de prix, FAQ, articles de blog, galeries, témoignages clients…

Pour installer Preline UI, il faut installer les paquets avec la commande `npm` et ajouter Preline UI en tant que plugin dans le fichier de configuration de Tailwind CSS.

Pour plus de détails, veuillez consulter la documentation en ligne à l'adresse suivante : https://preline.co/docs/index.html.

DaisyUI

DaisyUI est une autre ressource gratuite qui propose une bibliothèque de composants pour Tailwind CSS.

Si vous souhaitez en savoir plus sur cette bibliothèque, rendez-vous sur https://daisyui.com/.

La page d'accueil du site est bien réalisée et on peut dire qu'elle vend plutôt bien son produit étant donné qu'elle conseille de ne pas réinventer la roue.

Figure 12–4
Capture d'écran du site daisyUI.

Par exemple, si vous souhaitez styliser un bouton avec Tailwind CSS, l'élément `<button>` risque d'être rapidement surchargé de classes utilitaires, comme dans l'exemple suivant :

```
<button class="bg-indigo-600 px-4 py-3 text-center text-sm font-semibold
inline-block text-white cursor-pointer uppercase transition duration-200
ease-in-out rounded-md hover:bg-indigo-700 focus-visible:outline-none focus-
visible:ring-2 focus-visible:ring-indigo-600 focus-visible:ring-offset-2
active:scale-95">Tailwind Button</button>
```

La force de DaisyUI est de pouvoir utiliser des noms de classes sémantiques. Le code est alors plus léger et maintenable.

Le même bouton s'écrit de la manière suivante :

```
<button class="btn btn-primary">daisyUI Button</button>
```

Si vous connaissez Bootstrap, le code doit vous rappeler quelque chose !

DaisyUI est capable de nous proposer d'écrire 80 % de classes utilitaires en moins, ce qui allège de 70 % la taille du fichier HTML.

En plus des composants, DaisyUI propose un outil appelé Theme Generator dont l'interface permet de visualiser les composants et les couleurs (figure 12-5).

Cet outil s'avérera bien pratique pour sélectionner les couleurs. Il vous suffira ensuite de recopier le code dans le fichier de configuration de Tailwind CSS.

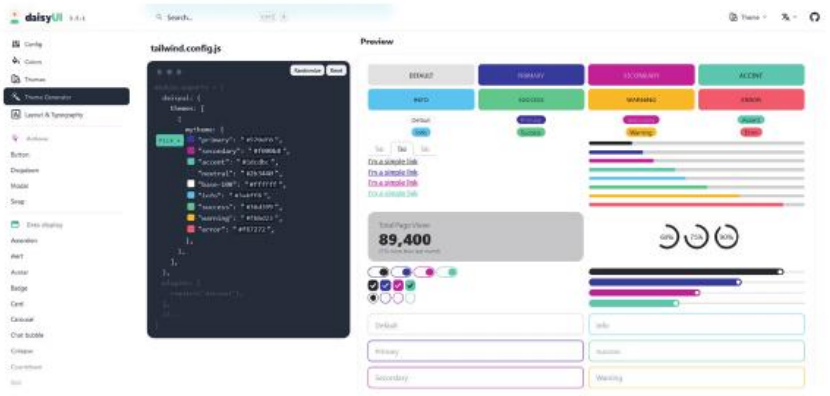


Figure 12–5
Interface de création de thème avec daisyUI.

HyperUI

Cet outil open source propose des composants gratuits pour vos projets Tailwind CSS.

Vous pouvez découvrir HyperUI à l'adresse suivante : https://www.hyperui.dev/.

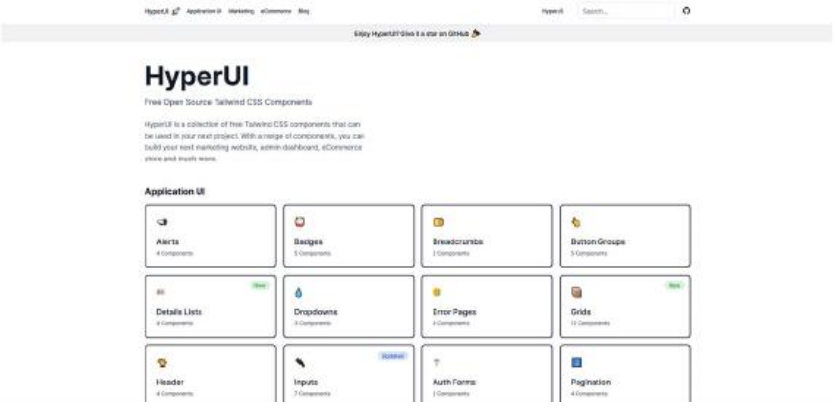


Figure 12–6
Capture d'écran du site HyperUI.

La page d'accueil affiche les différentes catégories de composants disponibles. Cliquez sur l'une d'entre elles pour afficher les composants.

L'interface vous permet de consulter chaque composant en fonction des tailles d'écran. Une fois votre choix effectué, vous n'aurez qu'à copier son code.

Flowbite

Terminons avec un dernier outil qui propose des composants et des éléments dynamiques. Il s'agit de Flowbite, accessible à l'adresse suivante : https://flowbite.com/.

Le site propose une solution payante et gratuite. Vous pouvez obtenir 60 composants open source.

Vous y trouverez également différents blocs pour intégrer rapidement dans vos pages des barres de navigation, des formulaires, des tableaux, etc.

Figure 12–7
Capture d'écran du site Flowbite.

De plus, Flowbite propose une bibliothèque d'icônes au format SVG.

Palettes de couleurs et générateurs de dégradés

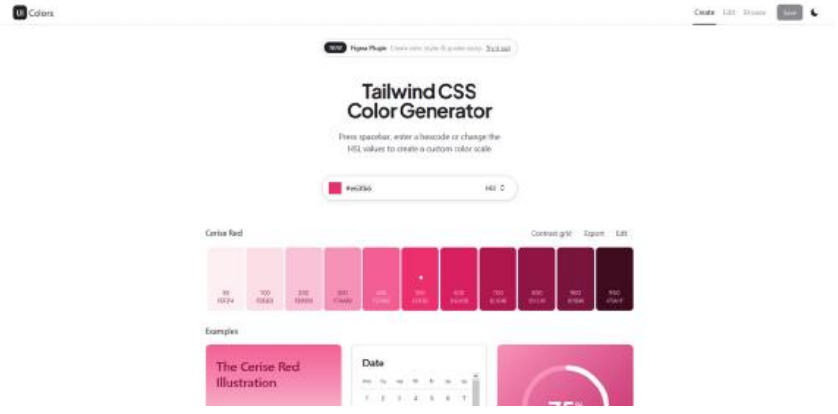
Vous trouverez sur Internet de nombreux outils qui vous aideront à créer des palettes de couleurs ou faciliteront l'écriture des classes pour réaliser des dégradés de couleurs sur les éléments.

UIColors

À partir d'une couleur de base, cet outil va en créer le nuancier.

Pour l'utiliser, rendez-vous à l'adresse suivante : https://uicolors.app/create.

Figure 12–8
Capture d'écran du site UIColors.



Il suffit de saisir la valeur hexadécimale d'une couleur et de laisser l'outil créer automatiquement le nuancier. Les noms des couleurs sont également indiqués.

Cliquez ensuite sur le lien *Export* pour récupérer le code à intégrer dans votre fichier de configuration de Tailwind CSS.

Voici l'exemple du code exporté :

```
'cerise-red': {
    '50': '#fef2f4',
    '100': '#fde6e9',
    '200': '#fbd0d9',
    '300': '#f7aab9',
    '400': '#f27a93',
    '500': '#e63f66',
    '600': '#d42a5b',
    '700': '#b21e4b',
    '800': '#951c45',
    '900': '#801b40',
    '950': '#470a1f',
},
```

En cliquant sur le lien *Edit*, vous pouvez modifier les valeurs du nuancier individuellement si celui-ci ne vous convient pas totalement.

Tailwind Shades

Voici un autre générateur de palettes de couleurs, similaire au précédent. Il est disponible à l'adresse suivante : https://www.tailwindshades.com/.

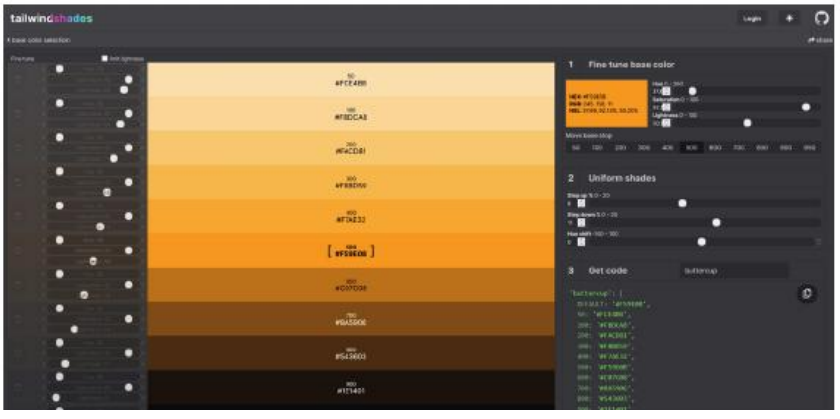


Figure 12–9
Capture d'écran du site Tailwind Shades.

Créez votre palette à partir d'une couleur de base. De nombreux réglages sont disponibles pour affiner les nuances.

Intéressons-nous à présent aux générateurs de dégradés.

Gradient Designer

Gradient Designer est un outil permettant de générer des dégradés de couleurs. Il est disponible à l'adresse suivante : https://gradient-designer.csspost.com/.

Figure 12–10
Capture d'écran du site Gradient Designer.

Le code généré est celui des classes utilitaires à ajouter à votre élément HTML.

En voici l'exemple :

```
bg-gradient-to-r from-cyan-400 to-blue-600
```

En plus des classes Tailwind, l'outil exporte également le code en CSS.

HyperColor

HyperColor est accessible à l'adresse suivante : https://hypercolor.dev/.

Dès sa page d'accueil, l'outil propose un large choix de dégradés parmi lesquels vous pouvez choisir celui qui vous convient le mieux et en récupérer le code.

Bien entendu, vous pouvez concevoir votre dégradé en effectuant vos propres réglages.

La petite particularité du site est qu'il propose également des drapeaux de pays, réalisés grâce aux dégradés. Voilà un concept plutôt original et qui peut être utile.

Figure 12–11
Capture d'écran du site HyperColor.

Voici un aperçu des outils que l'on peut trouver en ligne. Vous en trouverez probablement d'autres au fur et à mesure de vos investigations.

Cheat sheets

Les *cheat sheets* désignent les aide-mémoires ou antisèches. En un coup d'œil, vous pourrez y retrouver les classes utilitaires de Tailwind CSS dont vous avez besoin.

- https://tailwindcomponents.com/cheatsheet/

  Ce site regroupe les propriétés CSS et classes Tailwind correspondantes, sous forme de catégories. Un champ de recherche est disponible pour vous aider à trouver une propriété plus rapidement.
- https://umeshmk.github.io/Tailwindcss-cheatsheet/

  Ce site est également organisé en catégories. Il est plus visuel concernant les espacements et les couleurs. En effet, nous pouvons visualiser les largeurs avec une représentation graphique en barre. Concernant les couleurs, toutes les nuances de la palette de Tailwind sont représentées.

Outils divers

En plus des outils que nous avons évoqués précédemment, citons d'autres outils qui pourront également vous aider dans vos projets.

Tailwind Grid Generator

Disponible à l'adresse https://www.tailwindgen.com/, cet outil est un générateur de grille.

Il vous aidera à créer des grilles avec Tailwind CSS afin de pouvoir réaliser des mises en page complexes.

Pour cela, il suffit d'indiquer le nombre de lignes, de colonnes et l'espacement entre les cellules (la gouttière).

Vous pouvez fusionner des cellules en lignes et colonnes.

La figure 12-12 présente un exemple de réalisation.

Figure 12–12
Capture d'écran du site Tailwind Grid Generator.

Voici le code correspondant à cette grille :

```
<div class="grid grid-cols-5 grid-rows-3 gap-4">
    <div class="col-span-2">1</div>
    <div class="col-span-2 row-span-2 col-start-3">2</div>
    <div class="row-span-2 col-start-5">3</div>
    <div class="row-start-2">4</div>
    <div class="row-start-2">5</div>
    <div class="col-span-3 row-start-3">6</div>
    <div class="col-span-2 col-start-4 row-start-3">7</div>
</div>
```

Cette interface graphique est simple à utiliser et facilite grandement la création de grilles complexes.

Tailwind CSS Animated

Tailwind CSS Animated, comme son nom l'indique, est un outil qui vous aidera à réaliser des animations sur les éléments, qu'il s'agisse de textes ou d'images. L'interface propose de nombreux effets parmi lesquels vous pourrez choisir celui qui vous convient le mieux.

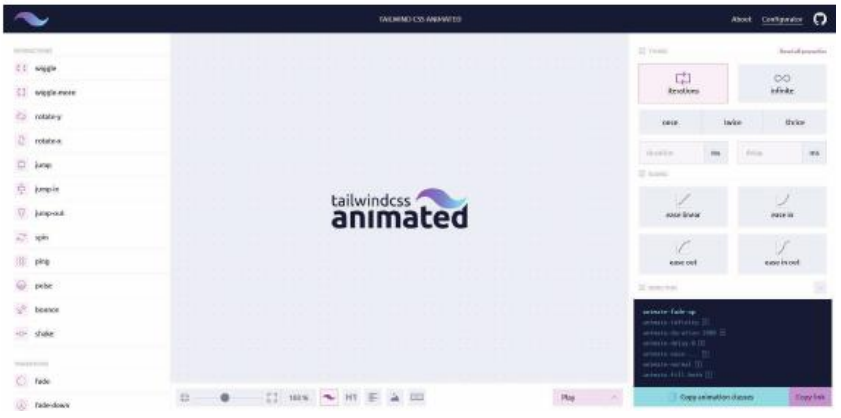


Figure 12–13
Capture d'écran du site Tailwind CSS Animated.

Tailwind CSS Animated est disponible à l'adresse suivante : https://www.tailwindcss-animated.com/.

Nous avons vu dans ce chapitre un certain nombre d'outils qui vous aideront dans votre production. Même si beaucoup d'entre eux génèrent du code afin de nous faciliter la tâche, je vous encourage à réaliser quelques projets sans ces outils pour bien assimiler les classes de Tailwind CSS lors de votre apprentissage du framework.

Par la suite, lorsque vous copierez les différents codes obtenus via ces outils, vous les comprendrez mieux et pourrez éventuellement les modifier pour arriver au résultat souhaité.

Conclusion

Nous arrivons au terme de cet ouvrage. Au fil de votre lecture, vous avez pris en main Tailwind CSS et vous êtes familiarisé avec ses principes fondamentaux pour concevoir des sites web modernes.

Vous avez eu l'occasion de mettre en pratique votre apprentissage au travers du projet « Burger Xpress ». J'espère que vous avez apprécié cette expérience et qu'elle vous aura permis de valider les connaissances acquises durant la lecture de ce livre.

Bien évidemment, votre apprentissage ne fait que commencer. Je vous encourage à poursuivre votre entraînement ! Mais surtout, si je peux vous donner un dernier conseil, évitez de copier/coller des codes trouvés sur Internet. Il est important de les écrire par vous-même. En effet, cela stimulera votre mémoire visuelle et vous amènera à connaître les classes utilitaires de Tailwind CSS sur le bout des doigts.

Vous trouverez une communauté grandissante autour de Tailwind CSS, des designers et des développeurs du monde entier qui partagent leur passion. Ils contribuent à proposer des ressources en ligne, comme celles que nous avons vues dans le dernier chapitre. Ils sauront vous aider en répondant à vos questions sur des forums ou les réseaux sociaux dédiés à Tailwind CSS.

La technologie évoluant rapidement, je vous recommande de vous tenir informé des dernières nouveautés de Tailwind CSS sur le blog https://tailwindcss.com/blog. Proposé en version 3.3.5 au moment de la rédaction de cet ouvrage, la version 3.4 devrait sortir très prochainement, avec son lot d'améliorations.

D'ici là, je vous souhaite une bonne continuation dans l'univers de Tailwind CSS.

Index