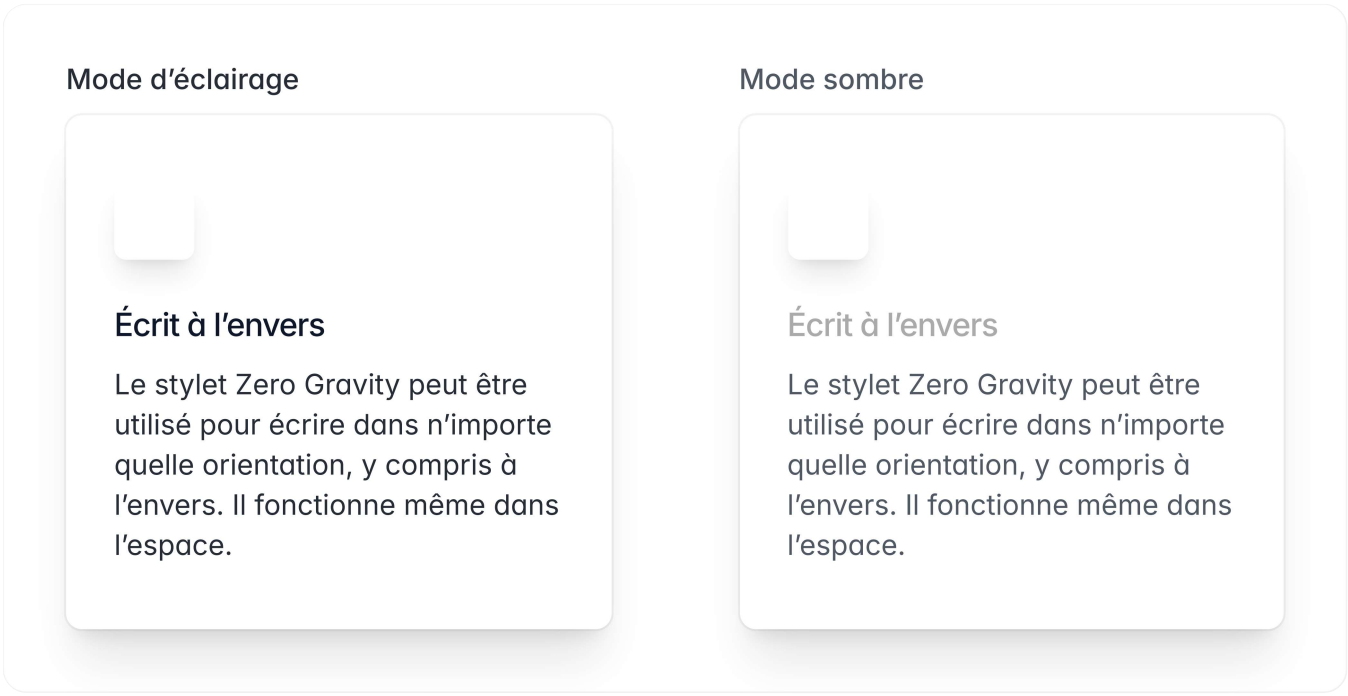
Concepts de base

Mode sombre

Utilisation de Tailwind CSS pour styliser votre site en mode sombre.

Maintenant que le mode sombre est une caractéristique de premier ordre de nombreux systèmes d'exploitation, il devient de plus en plus courant de concevoir une version sombre de votre site Web pour accompagner le design par défaut.

Pour rendre cela aussi facile que possible, Tailwind inclut une variante qui vous permet de styliser votre site différemment lorsque le mode sombre est activé : `dark`




```
<div class="bg-white dark:bg-slate-800 rounded-lg px-6 py-8 ring-1 ring-slat
  <div>
    <span class="inline-flex items-center justify-center p-2 bg-indigo-500 r
      <svg class="h-6 w-6 text-white" xmlns="http://www.w3.org/2000/svg" fil
    </span>
  </div>
  <h3 class="text-slate-900 dark:text-white mt-5 text-base font-medium track
  <p class="text-slate-500 dark:text-slate-400 mt-2 text-sm">
```

```
    The Zero Gravity Pen can be used to write in any orientation, including
  </p>
</div>
```

Par défaut, il utilise la fonction multimédia CSS, mais vous pouvez également créer des sites qui prennent en charge le basculement manuel du mode sombre à l'aide de la **stratégie « sélecteur ».**`prefers-color-scheme`

Basculer manuellement en mode sombre

Si vous souhaitez prendre en charge le basculement manuel du mode sombre au lieu de vous fier à la préférence du système d'exploitation, utilisez la stratégie au lieu de la stratégie :`selector``media`

La stratégie a remplacé la stratégie dans Tailwind CSS v3.4.1.`selector``class`

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: 'selector',
  // ...
}
```

Désormais, au lieu d'appliquer des classes sur la base de , elles seront appliquées chaque fois que la classe est présente plus tôt dans l'arborescence HTML.`dark:{class}``prefers-color-scheme``dark`

```
<!-- Dark mode not enabled -->
<html>
<body>
  <!-- Will be white -->
  <div class="bg-white dark:bg-black">
    <!-- ... -->
  </div>
```

```
</body>
</html>

<!-- Dark mode enabled -->
<html class="dark">
<body>
  <!-- Will be black -->
  <div class="bg-white dark:bg-black">
    <!-- ... -->
  </div>
</body>
</html>
```

If you've set **a prefix** in your Tailwind config, be sure to add that to the class. For example, if you have a prefix of , you'll need to use the class to enable dark mode.`dark` `tw-` `tw-dark`

How you add the class to the element is up to you, but a common approach is to use a bit of JavaScript that reads a preference from somewhere (like) and updates the DOM accordingly.`dark` `html` `localStorage`

Customizing the selector

Some frameworks (like NativeScript) have their own approach to enabling dark mode and add a different class name when dark mode is active.

You can customize the dark mode selector by setting to an array with your custom selector as the second item:`darkMode`

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: ['selector', '[data-mode="dark"]'],
  // ...
}
```

Tailwind will automatically wrap your custom dark mode selector with the pseudo-class to make sure the specificity is the same as it would be when using the strategy:`:where()``media`

```
.dark\:underline:where([data-mode="dark"], [data-mode="dark"] *){
  text-decoration-line: underline
}
```

Supporting system preference and manual selection

The strategy can be used to support both the user's system preference *or* a manually selected mode by using the `window.matchMedia()` API.`selector`

Here's a simple example of how you can support light mode, dark mode, as well as respecting the operating system preference:

spaghetti.js

```
// On page load or when changing themes, best to add inline in `head` to avo
document.documentElement.classList.toggle(
  'dark',
  localStorage.theme === 'dark' || (!('theme' in localStorage) && window.mat
)

// Whenever the user explicitly chooses light mode
localStorage.theme = 'light'

// Whenever the user explicitly chooses dark mode
localStorage.theme = 'dark'

// Whenever the user explicitly chooses to respect the OS preference
localStorage.removeItem('theme')
```

Again you can manage this however you like, even storing the preference server-side in a database and rendering the class on the server — it's totally up to you.

Overriding the dark variant

If you'd like to replace Tailwind's built-in dark variant with your own custom variant, you can do so using the dark mode strategy:`variant`

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: ['variant', '&:not(.light *)'],
  // ...
}
```

When using this strategy Tailwind will not modify the provided selector in any way, so be mindful of it's specificity and consider using the pseudo-class to ensure it has the same specificity as other utilities.`:where()`

Using multiple selectors

If you have multiple scenarios where dark mode should be enabled, you can specify all of them by providing an array:

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: ['variant', [
    '@media (prefers-color-scheme: dark) { &:not(.light *) }',
    '&:is(.dark *)',
  ]],
  // ...
}
```