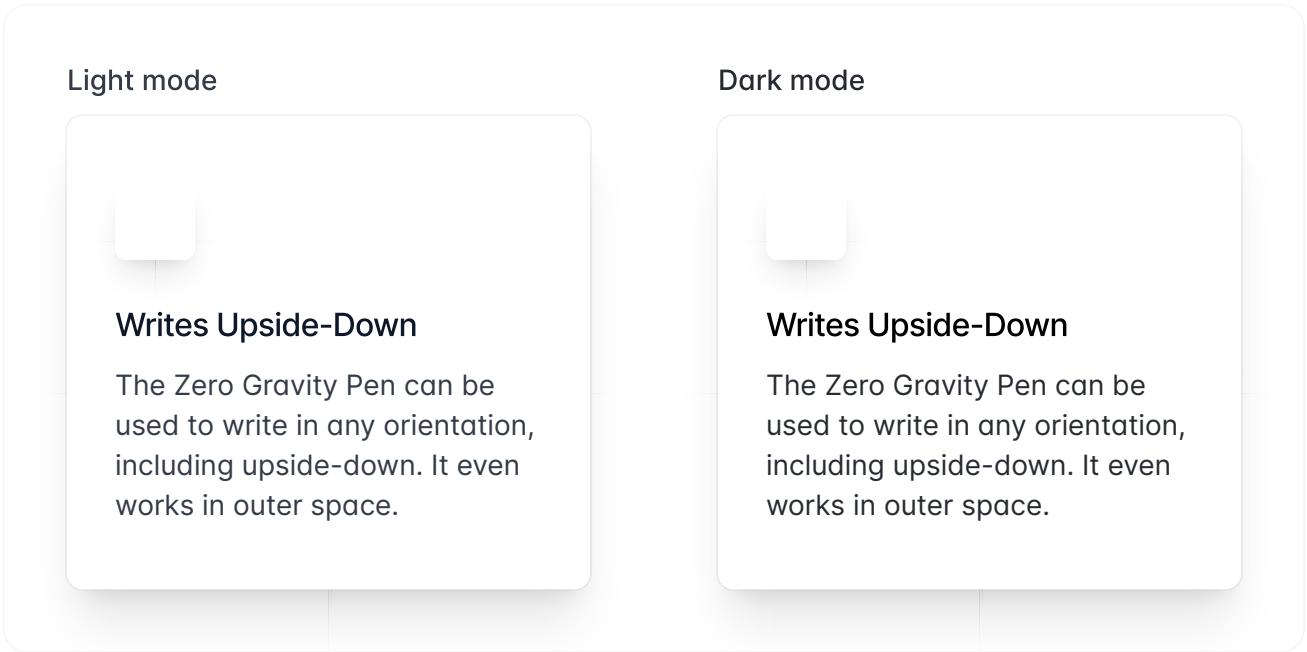
Core Concepts

Dark Mode

Using Tailwind CSS to style your site in dark mode.

Now that dark mode is a first-class feature of many operating systems, it's becoming more and more common to design a dark version of your website to go along with the default design.

To make this as easy as possible, Tailwind includes a `dark` variant that lets you style your site differently when dark mode is enabled:



```
<div class="bg-white dark:bg-slate-800 rounded-lg px-6 py-8 ring-1 ring-s
  <div>
    <span class="inline-flex items-center justify-center p-2 bg-indigo-50
      <svg class="h-6 w-6 text-white" xmlns="http://www.w3.org/2000/svg"
    </span>
  </div>
  <h3 class="text-slate-900 dark:text-white mt-5 text-base font-medium tr
  <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, includi
  </p>
</div>
```

By default this uses the `prefers-color-scheme` CSS media feature, but you can also build sites that support toggling dark mode manually using the **'class' strategy**.

Toggling dark mode manually

If you want to support toggling dark mode manually instead of relying on the operating system preference, use the `class` strategy instead of the `media` strategy:

tailwind.config.js

```
module.exports = {
  darkMode: 'class',
  // ...
}
```

Now instead of `dark:{class}` classes being applied based on `prefers-color-scheme`, they will be applied whenever `dark` class is present earlier in the HTML tree.

```
<!-- 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 `dark` class. For example, if you have a prefix of `tw-`, you'll need to use the `tw-dark` class to enable dark mode.

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

Supporting system preference and manual selection

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

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
if (localStorage.theme === 'dark' || (!('theme' in localStorage) && windo
  document.documentElement.classList.add('dark')
} else {
  document.documentElement.classList.remove('dark')
}

// 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.

Customizing the class name

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 name by setting `darkMode` to an array with your custom selector as the second item:

tailwind.config.js

```
module.exports = {
  darkMode: ['class', '[data-mode="dark"]'],
  // ...
}
```

‹ Responsive Design Reusing Styles ›