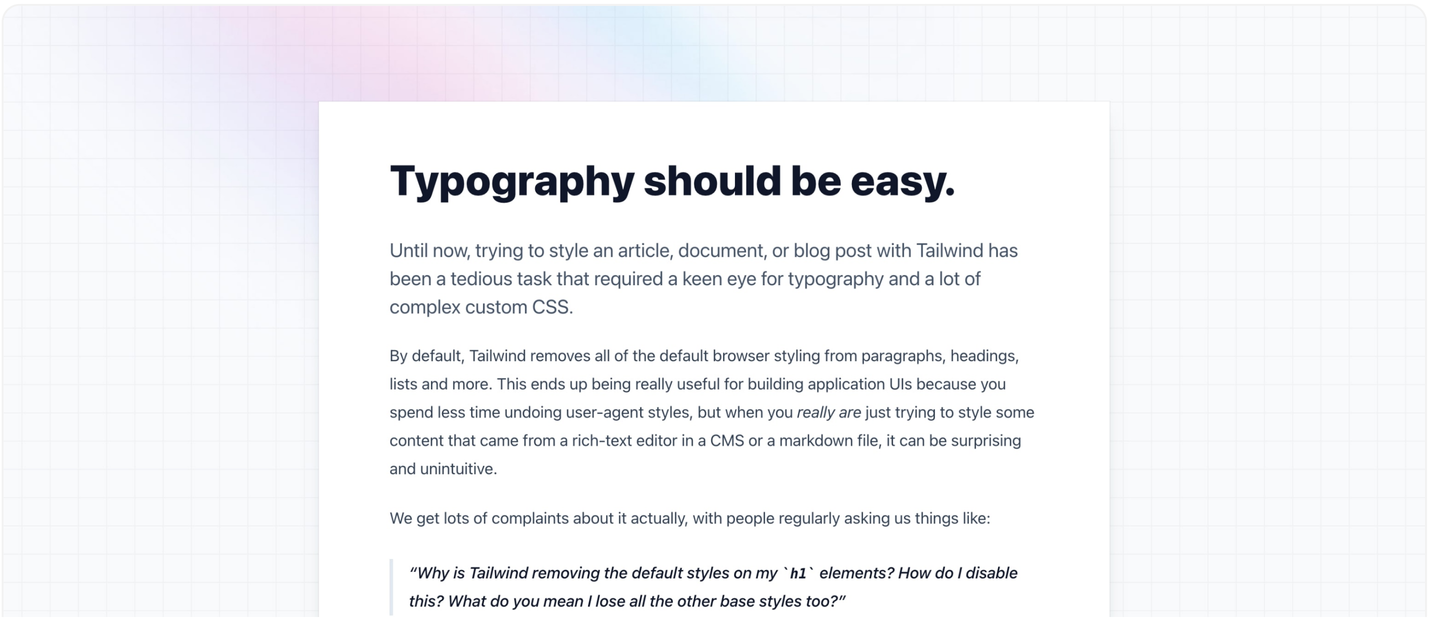
Official Plugins

@tailwindcss/typography

Beautiful typographic defaults for HTML you don't control.

The official Tailwind CSS Typography plugin provides a set of `prose` classes you can use to add beautiful typographic defaults to any vanilla HTML you don't control, like HTML rendered from Markdown, or pulled from a CMS.



Typography should be easy.

Until now, trying to style an article, document, or blog post with Tailwind has been a tedious task that required a keen eye for typography and a lot of complex custom CSS.

By default, Tailwind removes all of the default browser styling from paragraphs, headings, lists and more. This ends up being really useful for building application UIs because you spend less time undoing user-agent styles, but when you *really are* just trying to style some content that came from a rich-text editor in a CMS or a markdown file, it can be surprising and unintuitive.

We get lots of complaints about it actually, with people regularly asking us things like:

> *"Why is Tailwind removing the default styles on my `h1` elements? How do I disable this? What do you mean I lose all the other base styles too?"*

```
<article class="prose lg:prose-xl">
  {{ markdown }}
</article>
```

To see what it looks like in action, check out our **live demo** on Tailwind Play.

Installation

Install the plugin from npm:

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

Then add the plugin to your `` `tailwind.config.js` `` file:

tailwind.config.js

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

Basic usage

Now you can use the `` `prose` `` classes to add sensible typography styles to any vanilla HTML:

```
<article class="prose lg:prose-xl">
  <h1>Garlic bread with cheese: What the science tells us</h1>
  <p>
    For years parents have espoused the health benefits of eating garlic bread w
```

```
      children, with the food earning such an iconic status in our culture that ki
      up as warm, cheesy loaf for Halloween.
    </p>
    <p>
      But a recent study shows that the celebrated appetizer may be linked to a se
      springing up around the country.
    </p>
    <!-- ... -->
  </article>
```

Choosing a gray scale

This plugin includes a modifier class for each of the five gray scales Tailwind includes by default so you can easily style your content to match the grays you're using in your project.

```
<article class="prose prose-slate">
  {{ markdown }}
</article>
```

Here are the classes that are generated using a totally default Tailwind CSS v2.0 build:

Class	Gray scale
`` `prose-gray` `` *(default)*	Gray
`` `prose-slate` ``	Slate
`` `prose-zinc` ``	Zinc
`` `prose-neutral` ``	Neutral
`` `prose-stone` ``	Stone

Modifier classes are designed to be used with the **multi-class modifier pattern** and must be used in conjunction with the base `` `prose` `` class.

Always include the `prose` class when adding a gray scale modifier

```html
<article class="prose prose-stone">
  {{ markdown }}
</article>
```

To learn about creating your own color themes, read the **adding custom color themes** documentation.

Applying a type scale

Size modifiers allow you to adjust the overall size of your typography for different contexts.

```html
<article class="prose prose-xl">
  {{ markdown }}
</article>
```

Five different typography sizes are included out of the box:

Class	Body font size
`` `prose-sm` ``	0.875rem *(14px)*
`` `prose-base` `` *(default)*	1rem *(16px)*
`` `prose-lg` ``	1.125rem *(18px)*
`` `prose-xl` ``	1.25rem *(20px)*
`` `prose-2xl` ``	1.5rem *(24px)*

These can be used in combination with Tailwind's **breakpoint modifiers** to change the overall font size of a piece of content at different viewport sizes:

```html
<article class="prose md:prose-lg lg:prose-xl">
  {{ markdown }}
</article>
```

Everything about the provided size modifiers has been hand-tuned by professional designers to look as beautiful as possible, including the relationships between font sizes, heading spacing, code block padding, and more.

Size modifiers are designed to be used with the **multi-class modifier pattern** and must be used in conjunction with the base `prose` class.

Always include the `prose` class when adding a size modifier

```
<article class="prose prose-lg">
  {{ markdown }}
</article>
```

To learn about customizing the included type scales, read the documentation on **customizing the CSS**.

Adapting to dark mode

Each default color theme includes a hand-designed dark mode version that you can trigger by adding the `prose-invert` class:

```
<article class="prose dark:prose-invert">
  {{ markdown }}
</article>
```

To learn about creating your own color themes, read the **adding custom color themes** documentation.

Element modifiers

Use element modifiers to customize the style of individual elements in your content directly in your HTML:

```
<article class="prose prose-img:rounded-xl prose-headings:underline prose-a:text
  {{ markdown }}
</article>
```

This makes it easy to do things like style links to match your brand, add a border radius to images, and tons more.

Here's a complete list of available element modifiers:

Modifier	Target
\`prose-headings:{utility}\`	\`h1\`, \`h2\`, \`h3\`, \`h4\`, \`th\`
\`prose-lead:{utility}\`	\`[class~="lead"]\`
\`prose-h1:{utility}\`	\`h1\`
\`prose-h2:{utility}\`	\`h2\`
\`prose-h3:{utility}\`	\`h3\`
\`prose-h4:{utility}\`	\`h4\`
\`prose-p:{utility}\`	\`p\`
\`prose-a:{utility}\`	\`a\`
\`prose-blockquote:{utility}\`	\`blockquote\`
\`prose-figure:{utility}\`	\`figure\`
\`prose-figcaption:{utility}\`	\`figcaption\`
\`prose-strong:{utility}\`	\`strong\`
\`prose-em:{utility}\`	\`em\`
\`prose-code:{utility}\`	\`code\`
\`prose-pre:{utility}\`	\`pre\`
\`prose-ol:{utility}\`	\`ol\`
\`prose-ul:{utility}\`	\`ul\`

Modifier	Target
`prose-li:{utility}`	`li`
`prose-table:{utility}`	`table`
`prose-thead:{utility}`	`thead`
`prose-tr:{utility}`	`tr`
`prose-th:{utility}`	`th`
`prose-td:{utility}`	`td`
`prose-img:{utility}`	`img`
`prose-video:{utility}`	`video`
`prose-hr:{utility}`	`hr`

When stacking these modifiers with other modifiers like `hover`, you most likely want the other modifier to come first:

```
<article class="prose prose-a:text-blue-600 hover:prose-a:text-blue-500">
  {{ markdown }}
</article>
```

Read the Tailwind CSS documentation on **ordering stacked modifiers** to learn more.

Overriding max-width

Each size modifier comes with a baked in `max-width` designed to keep the content as readable as possible. This isn't always what you want though, and sometimes you'll want the content to just fill the width of its container.

In those cases, all you need to do is add `max-w-none` to your content to override the embedded max-width:

```
<div class="grid grid-cols-4">
  <div class="col-span-1">
```

```
    <!-- ... -->
  </div>
  <div class="col-span-3">
    <article class="prose max-w-none">
      {{ markdown }}
    </article>
  </div>
</div>
```

Advanced topics

Undoing typography styles

If you have a block of markup embedded in some content that shouldn't inherit the `prose` styles, use the `not-prose` class to sandbox it:

```
<article class="prose">
  <h1>My Heading</h1>
  <p>...</p>

  <div class="not-prose">
    <!-- Some example or demo that needs to be prose-free -->
  </div>

  <p>...</p>
  <!-- ... -->
</article>
```

Note that you can't nest new `prose` instances within a `not-prose` block at this time.

Adding custom color themes

You can create your own color theme by adding a new key in the `typography` section of your `tailwind.config.js` file and providing your colors under the `css` key:

tailwind.config.js

```
module.exports = {
  theme: {
    extend: {
      typography: ({ theme }) => ({
        pink: {
          css: {
            '--tw-prose-body': theme('colors.pink[800]'),
            '--tw-prose-headings': theme('colors.pink[900]'),
            '--tw-prose-lead': theme('colors.pink[700]'),
            '--tw-prose-links': theme('colors.pink[900]'),
            '--tw-prose-bold': theme('colors.pink[900]'),
            '--tw-prose-counters': theme('colors.pink[600]'),
            '--tw-prose-bullets': theme('colors.pink[400]'),
            '--tw-prose-hr': theme('colors.pink[300]'),
            '--tw-prose-quotes': theme('colors.pink[900]'),
            '--tw-prose-quote-borders': theme('colors.pink[300]'),
            '--tw-prose-captions': theme('colors.pink[700]'),
            '--tw-prose-code': theme('colors.pink[900]'),
            '--tw-prose-pre-code': theme('colors.pink[100]'),
            '--tw-prose-pre-bg': theme('colors.pink[900]'),
            '--tw-prose-th-borders': theme('colors.pink[300]'),
            '--tw-prose-td-borders': theme('colors.pink[200]'),
            '--tw-prose-invert-body': theme('colors.pink[200]'),
            '--tw-prose-invert-headings': theme('colors.white'),
            '--tw-prose-invert-lead': theme('colors.pink[300]'),
            '--tw-prose-invert-links': theme('colors.white'),
            '--tw-prose-invert-bold': theme('colors.white'),
            '--tw-prose-invert-counters': theme('colors.pink[400]'),
            '--tw-prose-invert-bullets': theme('colors.pink[600]'),
            '--tw-prose-invert-hr': theme('colors.pink[700]'),
            '--tw-prose-invert-quotes': theme('colors.pink[100]'),
            '--tw-prose-invert-quote-borders': theme('colors.pink[700]'),
            '--tw-prose-invert-captions': theme('colors.pink[400]'),
            '--tw-prose-invert-code': theme('colors.white'),
            '--tw-prose-invert-pre-code': theme('colors.pink[300]'),
            '--tw-prose-invert-pre-bg': 'rgb(0 0 0 / 50%)',
            '--tw-prose-invert-th-borders': theme('colors.pink[600]'),
            '--tw-prose-invert-td-borders': theme('colors.pink[700]'),
          },
        },
      }),
    },
  },
  plugins: [
    require('@tailwindcss/typography'),
    // ...
  ],
}
```

See our internal **style definitions** for some more examples.

Changing the default class name

If you need to use a class name other than `` `prose` `` for any reason, you can do so using the `` `className` `` option when registering the plugin:

tailwind.config.js

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

Now every instance of `prose` in the default class names will be replaced by your custom class name:

```html
<article class="wysiwyg wysiwyg-slate lg:wysiwyg-xl">
  <h1>My Heading</h1>
  <p>...</p>

  <div class="not-wysiwyg">
    <!-- Some example or demo that needs to be prose-free -->
  </div>

  <p>...</p>
  <!-- ... -->
</article>
```

Customizing the CSS

If you want to customize the raw CSS generated by this plugin, add your overrides under the `typography` key in the `theme` section of your `tailwind.config.js` file:

tailwind.config.js

```js
module.exports = {
  theme: {
    extend: {
      typography: {
        DEFAULT: {
          css: {
            color: '#333',
            a: {
              color: '#3182ce',
              '&:hover': {
                color: '#2c5282',
              },
            },
          },
        },
      },
    },
  },
  plugins: [
    require('@tailwindcss/typography'),
    // ...
  ],
}
```

Like with all theme customizations in Tailwind, you can also define the `typography` key as a function if you need access to the `theme` helper:

tailwind.config.js

```
module.exports = {
  theme: {
    extend: {
      typography: (theme) => ({
        DEFAULT: {
          css: {
            color: theme('colors.gray.800'),

            // ...
          },
        },
      }),
    },
  },
  plugins: [
    require('@tailwindcss/typography'),
    // ...
  ],
}
```

Customizations should be applied to a specific modifier like `DEFAULT` or `xl`, and must be added under the `css` property. Customizations are authored in the same **CSS-in-JS syntax** used to write Tailwind plugins.

See **the default styles** for this plugin for more in-depth examples of configuring each modifier.

‹ Screen Readers Forms ›