Customizing Markdown Components

Using MDX, you can replace every HTML element that Markdown renders with a custom implementation. This allows you to use a set of design system components when rendering.

```
import { MDXProvider } from "@mdx-js/react"
import * as DesignSystem from "your-design-system"

export default function Layout({ children }) {
  return (
    <MDXProvider
      components={{
        // Map HTML element tag to React component
        h1: DesignSystem.H1,
        h2: DesignSystem.H2,
        h3: DesignSystem.H3,
        // Or define component inline
        p: props => <p {...props} style={{ color: "rebeccapurple" }} />,
      }}
    >
      {children}
    </MDXProvider>
  )
}
```

Note: you can also provide your own custom components to the `MDXProvider` that make them globally available while writing MDX. You can find more details about this pattern in the Importing and Using Components in MDX guide.

The following components can be customized with the MDXProvider:

Tag	Name	Syntax
`p`	Paragraph	
`h1`	Heading 1	`#`
`h2`	Heading 2	`##`
`h3`	Heading 3	`###`
`h4`	Heading 4	`####`

Tag	Name	Syntax
h5	Heading 5	#####
h6	Heading 6	######
thematicBreak	Thematic break	***
blockquote	Blockquote	>
ul	List	-
ol	Ordered list	1.
li	List item	
table	Table	--- | --- | --- | ---
tr	Table row	This | is | a | table row
td/th	Table cell	
pre	Pre	```js console.log()```
code	Code	`console.log()`
em	Emphasis	_emphasis_
strong	Strong	**strong**
delete	Delete	~~strikethrough~~
code	InlineCode	`console.log()`
hr	Break	---
a	Link	https://mdxjs.com or [MDX](https://mdxjs.com)
img	Image	![alt](https://mdx-logo.now.sh)

How does this work?

Components passed to the MDXProvider are used to render the HTML elements that Markdown creates. It uses React's Context API.

Related

- MDX components