Theme UI

Theme UI is a library for styling React apps with user-configurable design constraints. It allows you to style any component in your application with typographic, color, and layout values defined in a shared theme object. Theme UI is currently used in Gatsby's official themes, but it can be used in any Gatsby site or React application. It includes the Emotion CSS-in-JS library along with additional utilities for styling MDX and using configurations and themes from Typography.js.

Using Theme UI in Gatsby

Theme UI includes the `gatsby-plugin-theme-ui` package to better integrate with your Gatsby project.

Install the following packages to add Theme UI.

```
npm install theme-ui gatsby-plugin-theme-ui
```

After installing the dependencies, add the following to your `gatsby-config.js`.

```
module.exports = {
  plugins: ["gatsby-plugin-theme-ui"],
}
```

Theme UI uses a `theme` configuration object to provide color, typography, layout, and other shared stylistic values through React context. This allows components within your site to add styles based on a predefined set of values.

There are two ways to add a theme to your site: using configuration options in `gatsby-plugin-theme-ui`, or local shadowing.

`gatsby-plugin-theme-ui` configuration

`gatsby-plugin-theme-ui` allows for optional configuration. One of the options you can pass in is `preset`. Preset is a base theme object you'd like your site to use.

The `preset` can be a JSON object or a string package name that the plugin will require for you.

If you're passing in a string package name you need to make sure that the dependency is installed.

```
npm install @theme-ui/preset-funk
```

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-theme-ui`,
      options: {
        preset: "@theme-ui/preset-funk",
      },
    },
  ],
}
```

Alternatively, you could do the same thing and require the package yourself. This is necessary if you're referencing a local package that isn't available publicly.

```
npm install @theme-ui/preset-funk
```

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-theme-ui`,
      options: {
        preset: require("my-local-preset"),
      },
    },
  ],
}
```

`preset` operates as your base theme and any local shadowed styles will automatically merge with it. Local styles are considered more specific and will override base styles where necessary.

Local shadowing

The Theme UI plugin uses the component shadowing API to add the theme object context to your site. Shadowing is the ability to create a file in the same location as the theme you're leveraging and override the original file's content.

In this case, that location is `src/gatsby-plugin-theme-ui`. So, create that directory in your project and add an `index.js` file to export a theme object.

```
mkdir src/gatsby-plugin-theme-ui
```

```
export default {}
```

Creating a theme object

Create a `src/gatsby-plugin-theme-ui` directory in your project and add an `index.js` file to export a theme object. Add a `colors` object to the file to store the color palette for your site.

```
export default {
  colors: {
    text: "#333",
    background: "#fff",
    primary: "#639",
    secondary: "#ff6347",
  },
}
```

Next add some base typographic values.

```
export default {
  colors: {
    text: "#333",
    background: "#fff",
    primary: "#639",
    secondary: "#ff6347",
  },
  // highlight-start
  fonts: {
    body: "system-ui, sans-serif",
    heading: "system-ui, sans-serif",
    monospace: "Menlo, monospace",
  },
  fontWeights: {
    body: 400,
    heading: 700,
    bold: 700,
  },
  lineHeights: {
    body: 1.5,
    heading: 1.125,
  },
  fontSizes: [12, 14, 16, 20, 24, 32, 48, 64, 72],
  // highlight-end
}
```

Next add values for use in margin and padding.

```
export default {
  colors: {
    text: "#333",
```

```
    background: "#fff",
    primary: "#639",
    secondary: "#ff6347",
  },
  fonts: {
    body: "system-ui, sans-serif",
    heading: "system-ui, sans-serif",
    monospace: "Menlo, monospace",
  },
  fontWeights: {
    body: 400,
    heading: 700,
    bold: 700,
  },
  lineHeights: {
    body: 1.5,
    heading: 1.125,
  },
  fontSizes: [12, 14, 16, 20, 24, 32, 48, 64, 72],
  // highlight-start
  space: [0, 4, 8, 16, 32, 64, 128, 256, 512],
  // highlight-end
}
```

Feel free to include as many additional values as you'd like in your theme. Read more about theming in the Theme UI documentation.

Adding styles to elements

Theme UI uses a custom function to add support for the Theme UI `sx` prop in JSX. This custom function is enabled by including a comment on the top of the file:

```
/** @jsx jsx */
import { jsx } from "theme-ui"
```

The `sx` prop is used to style elements by referencing values from the theme object.

```
/** @jsx jsx */
import { jsx } from "theme-ui"

export default function Header(props) {
  return (
    <header
      sx={{
        // this uses the value from `theme.space[4]`
        padding: 4,
```

```
        // these use values from `theme.colors`
        color: "background",
        backgroundColor: "primary",
      }}
    >
      {props.children}
    </header>
  )
}
```

Using Theme UI in a Gatsby theme

When using Theme UI in a Gatsby theme, it's important to understand how the `gatsby-plugin-theme-ui` package handles theme objects from multiple Gatsby themes and sites.

If multiple themes are installed in the same site, the one defined last in your `gatsby-config.js` file's `plugins` array will take precedence.

To extend an existing Theme UI configuration from a theme you'll need to find out how the base theme is passing styles. If it's using a `preset`, as the `gatsby-theme-blog` package does, any shadowed files are merged automatically. This can be discovered by looking at the theme source code in `node_modules` or on GitHub.

```
export default {
  colors: {
    text: "#222",
    primary: "tomato",
  },
}
```

If the theme itself is shadowing styles without a preset, you'll want to merge those styles with your local ones. Here's an example using Theme UI's `merge` API:

```
import baseTheme from "gatsby-theme-unknown/src/gatsby-plugin-theme-ui"
import { merge } from "theme-ui"

// merge will deeply merge custom values with the
// unknown theme's defaults
export default merge(baseTheme, {
  colors: {
    text: "#222",
    primary: "tomato",
  },
})
```

Styling MDX content

Theme UI includes a way to style elements in MDX documents without the need to add global CSS to your site. This is useful for authoring Gatsby themes that can be installed alongside other themes.

In your Theme UI configuration, add a `styles` object to target elements rendered from MDX.

```
export default {
  // base theme values...
  styles: {
    // the keys used here reference elements in MDX
    h1: {
      // the style object for each element
      // can reference other values in the theme
      fontFamily: "heading",
      fontWeight: "heading",
      lineHeight: "heading",
      marginTop: 0,
      marginBottom: 3,
    },
    a: {
      color: "primary",
      ":hover, :focus": {
        color: "secondary",
      },
    },
    // more styles can be added as needed
  },
}
```

With the example above, any `<h1>` or `<a>` elements rendered from an MDX file will include these base styles.

To learn more about using Theme UI in your project, see the official Theme UI website.