Theming Grafana

Overview

Themes are implemented in Typescript. That's because our goal is to share variables between Grafana TypeScript and Sass code. Theme definitions are located in the following files:

- packages/grafana-data/src/themes/createTheme.ts
- packages/grafana-data/src/themes/createColors.ts

Usage

This section provides usage guidelines.

Using themes in React components

Here's how to use Grafana themes in React components.

useStyles2 hook `useStyles2` memoizes the function and provides access to the theme.

```
import React, { FC } from 'react';
import { GrafanaTheme2 } from '@grafana/data';
import { useStyles2 } from '@grafana/ui';
import { css } from '@emotion/css';

const getComponentStyles = (theme: GrafanaTheme2) => css`
  padding: ${theme.spacing.md};
`;

const Foo: FC<FooProps> = () => {
  const styles = useStyles2(getComponentsStyles);
  // Use styles with className
};
```

Get the theme object

```
import React, { FC } from 'react';
import { useTheme2 } from '@grafana/ui';

const Foo: FC<FooProps> = () => {
  const theme = useTheme2();

  // Your component has access to the theme variables now
};
```

Picking the right variable

The rich color object and the state colors

The theme.colors object has 6 rich color objects for `primary`, `secondary`, `info`, `success`, `warning` and `error`. These all have the same sub colors that have different use cases.

Property	When to use
main	For backgrounds
shade	For hover highlight
text	For text color
border	For borders, currently always the same as text color
contrastText	Text color to use for text placed on top of the main color

Example use cases:

- Want a `red` background? Use `theme.colors.error.main`
- Want `green` text? Use `theme.colors.success.text`
- Want text to be visible when placed inside a background that uses `theme.colors.error.main` then use `theme.colors.error.contrastText`.

Text colors

Property	When to use
theme.colors.text.primary	The default text color
theme.colors.text.secondary	Text color for things that should be a bit less prominent
theme.colors.text.disabled	Text color for disabled / faint things
theme.colors.text.link	Text link color
theme.colors.text.maxContrast	Maximum contrast (absolute white in dark theme, absolute black in white theme)

Background colors

Property	When to use
theme.colors.background.canvas	The dashboard background. A background surface for panels and panes that use primary background
theme.colors.background.primary	The default content background for content panes and panels
theme.colors.background.secondary	For cards and other surfaces that need to stand out when placed on top of the primary background

Borders

Property	When to use
theme.colors.border.weak	Primary border for panels and panes and other subtle borders
theme.colors.border.medium	For stronger borders like inputs
theme.colors.border.strong	For even stronger border like hover highighted border

Actions

Property	When to use
theme.colors.action.hover	Background color for hover on card, menu or list item
theme.colors.action.focus	Background color for focused card, menu or list item
theme.colors.action.selected	Background color for selected card, menu or list item

Paddings and margins

Example	Result
theme.spacing(1)	8px
theme.spacing(1, 2)	8px 16px
theme.spacing(1, 2, 0.5, 4)	8px 16px 4px 32px

Border radius

Example	Result
theme.shape.borderRadius(1)	2px
theme.shape.borderRadius(2)	4px

Typography

For font family, font sizes and line heights use the variables under `theme.typography`.

Using `ThemeContext` directly

```
import { ThemeContext } from '@grafana/ui';

<ThemeContext.Consumer>{(theme) => <Foo theme={theme} />}</ThemeContext.Consumer>;
```

Using `withTheme` higher-order component (HOC) With this method your component will be automatically wrapped in `ThemeContext.Consumer` and provided with current theme via `theme` prop. Components used with `withTheme` must implement the `Themeable` interface.

```
import  { ThemeContext, Themeable } from '@grafana/ui';

interface FooProps extends Themeable2 {}

const Foo: React.FunctionComponent<FooProps> = () => ...

export default withTheme2(Foo);
```

Using theme in tests

If you need to pass a theme object to a function under test just import `createTheme` and call it without any arguments.

```
import { createTheme } from '@grafana/data';

describe('MyComponent', () => {
  it('should work', () => {
    result = functionThatNeedsTheme(createTheme());
    expect(result).toBe(true);
  });
});
```

FAQ

This section provides insight into frequently-asked questions.

How can I modify Sass variable files?

If possible, migrate styles to Emotion

For the following to apply you need to run `yarn dev` task.

`[_variables|_variables.dark|_variables.light].generated.scss` files are the ones that are referenced in the main Sass files for Sass variables to be available. **These files are automatically generated and should never be modified by hand!**

If you need to modify the sass variable files be sure to update the files that end with `.tmpl.ts` and not the `.generated.scss` files.