Styling Grafana

Emotion is our default-to-be approach to styling React components. It provides a way for styles to be a consequence of properties and state of a component.

Usage

Basic styling

For styling components, use Emotion's `css` function.

```
import React from 'react';
import { css } from '@emotion/css';

const ComponentA = () => (
  <div
    className={css`
      background: red;
    `}
  >
    As red as you can get
  </div>
);
```

Styling with theme

To access the theme in your styles, use the `useStyles` hook. It provides basic memoization and access to the theme object.

> *Please remember to put `getStyles` function at the end of the file!*

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

const Foo: FC<FooProps> = () => {
  const styles = useStyles(getStyles);

  // Use styles with classNames
  return <div className={styles}>...</div>;
};

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

Styling complex components

In more complex cases, especially when you need to style multiple DOM elements in one component, or when using styles that depend on properties and/or state, you should create a helper function that returns an object of styles. This function should also be wrapped in the `stylesFactory` helper function, which will provide basic memoization.

Let's say you need to style a component that has a different background depending on the `isActive` property :

```
import React from 'react';
import { css } from '@emotion/css';
import { GrafanaTheme } from '@grafana/data';
import { selectThemeVariant, stylesFactory, useTheme } from '@grafana/ui';

interface ComponentAProps {
  isActive: boolean;
}

const ComponentA: React.FC<ComponentAProps> = ({ isActive }) => {
  const theme = useTheme();
  const styles = getStyles(theme, isActive);

  return (
    <div className={styles.wrapper}>
      As red as you can get
      <i className={styles.icon} />
    </div>
  );
};

// Mind, that you can pass multiple arguments, theme included
const getStyles = stylesFactory((theme: GrafanaTheme, isActive: boolean) => {
  const backgroundColor = isActive ? theme.colors.red : theme.colors.blue;

  return {
    wrapper: css`
      background: ${backgroundColor};
    `,
    icon: css`
      font-size: ${theme.typography.size.sm};
    `,
  };
});
```

For more information about themes at Grafana please see the themes guide.

Composing class names

For class composition, use Emotion's `cx` function.

```
import React from 'react';
import { css, cx } from '@emotion/css';
```

```tsx
interface Props {
  className?: string;
}

const ComponentA: React.FC<Props> = ({ className }) => {
  const finalClassName = cx(
    className,
    css`
      background: red;
    `
  );

  return <div className={finalClassName}>As red as you can ge</div>;
};
```