

Strongly-typed CSS with Vanilla-extract



Modern TS Web App



Today

Today



Today

- 🔎🤔
- 😍🤢

Today

- 🔎🤔
- 😍🤢
- 🌬🥧

Today

- 🔎🤔
- 😍🤢
- ☁🥧
- 🍰✓

Hello!



Georgia Loper



@developerLoper

 Large enterprises

 Tiny startups

 Quickly scaling companies

 Legacy systems

 Static landing pages

 SPAs

 Web applications



Patina Design System @ Upstart

TODO: QR code for slides/code



Markup

```
<div>
  <h1>🧁 Hello, CodeMash!</h1>
</div>
```

Styles

Static

```
{  
  background-color: "navy";  
  color: "plum";  
  padding: 1rem;  
}
```

Parent component

Static

```
// App.tsx
<Heading>
  🎂 Hello, CodeMash!
</Heading>
```

Parent component

Dynamic

```
// App.tsx
<Heading background="aqua" color="indigo" padding="large" >
  🎂 Hello, CodeMash!
</Heading>
```

Styles

Themed

```
{  
  background-color: var(--color-primary);  
  color: var(--color-secondary);  
  padding: var(--space-small);  
  
  :root {  
    --color-primary: "navy";  
    --color-secondary: "plum";  
    --space-small: "1rem";  
  }  
}
```

Red Yellow Green

css2022

English ▾

T-shirt »

> Introduction

[T-shirt](#)

[Data Explorer](#)

[Demographics](#)

[Features](#)

[CSS Frameworks](#)

[CSS-in-JS](#)

[Other Tools](#)

[CSS Usage](#)

[Resources](#)

[Awards](#)

[Conclusion](#)

[About](#)

STATE OF

2022

CSS FRAMEWORKS

Libraries that give you pre-made components and styles.

RATIOS OVER TIME

Retention, interest, usage, and awareness ratio over time.

Percentages Rankings

The chart displays the retention, interest, usage, and awareness ratio for ten CSS frameworks over four years. The Y-axis represents the ratio percentage, ranging from 0 to 100. The X-axis shows the years 2019, 2020, 2021, and 2022. Most frameworks show a general downward trend in their ratios over the period, except for Ant Design which remains relatively stable around 60%.

Framework	2019	2020	2021	2022
Bootstrap	80	85	78	80
Materialize CSS	55	52	48	42
Ant Design	60	60	58	50
Semantic UI	58	52	48	48
Bulma	75	62	55	48
Tachyons	68	52	48	42
Primer	45	48	45	42
Tailwind CSS	52	50	42	38
PureCSS	68	70	70	70
Foundation	45	32	28	25
Halfmoon	65	70	70	70



css2022

CSS FRAMEWORKS

[Introduction](#)

[T-shirt](#)

[Data Explorer](#)

[Demographics](#)

[Features](#)

[> CSS Frameworks](#)

[CSS-in-JS](#)

[Other Tools](#)

[CSS Usage](#)

[Resources](#)

[Awards](#)

[Conclusion](#)

[About](#)



Libraries that give you pre-made components and styles.

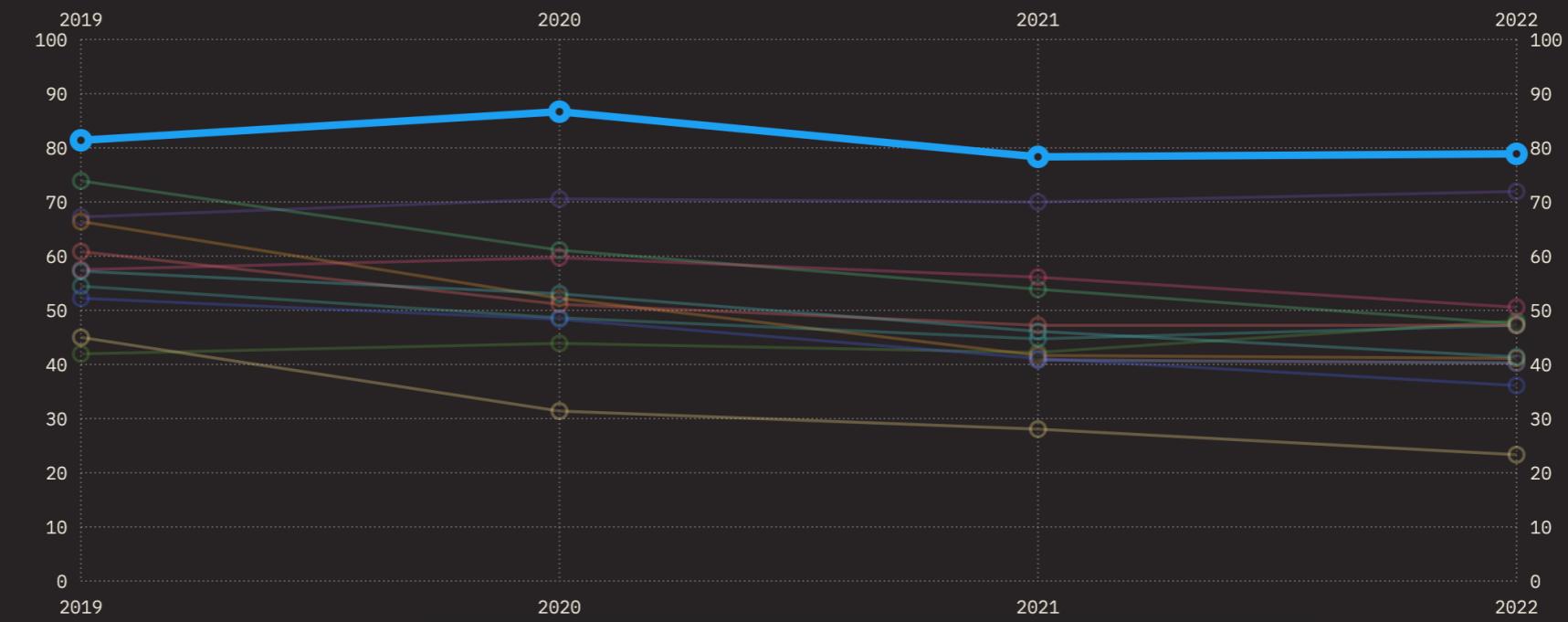
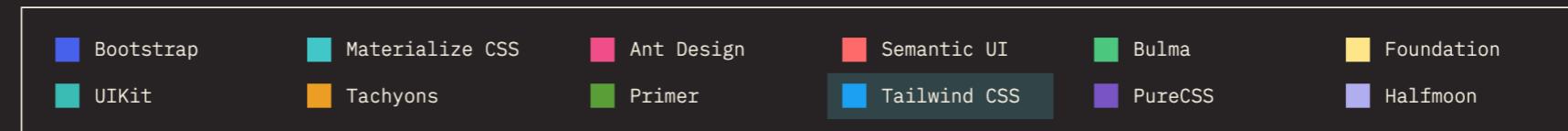
RATIOS OVER TIME



Percentages

Rankings

Retention, interest, usage, and awareness ratio over time.



css2022

CSS-IN-JS

Libraries that help integrate CSS into JavaScript code.

RATIOS OVER TIME 🎃

Retention, interest, usage, and awareness ratio over time.

The chart displays the retention, interest, usage, and awareness ratios for ten CSS-in-JS libraries over four years. The Y-axis represents the ratio scale, ranging from 20 to 100. The X-axis shows the years 2019, 2020, 2021, and 2022. Most libraries show a general decline in their ratios over time, except for Fela which shows a slight increase.

Library	2019	2020	2021	2022
Styled Components	85	82	78	75
JSS	60	58	50	48
Stitches	68	70	68	65
Fela	40	42	45	36
vanilla-extract	88	88	88	85
Linaria	65	68	70	60
Emotion	88	80	75	70
Astroturf	58	58	58	38
CSS Modules	88	88	88	85
Windi CSS	88	88	85	75
Styled System	85	85	82	78
Twin	70	72	70	65
Theme UI	70	72	70	65

Legend:

- Styled Components
- JSS
- Stitches
- Fela
- vanilla-extract
- Linaria
- Emotion
- Astroturf
- CSS Modules
- Windi CSS
- Styled System
- Twin
- Theme UI

Icons on the right side of the chart area:

- Bar chart icon
- Grid icon
- Share icon

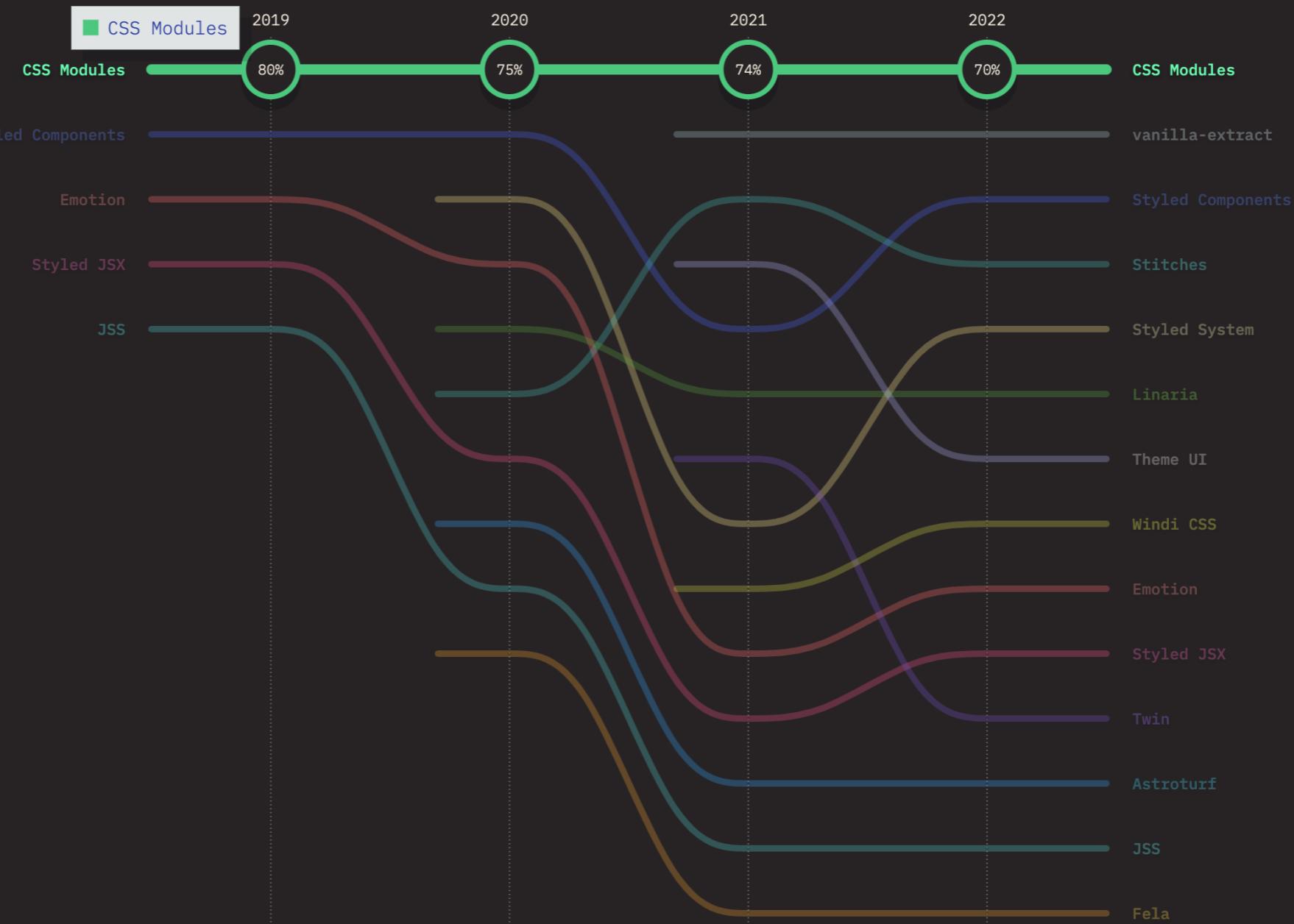
Social media icons at the bottom left:

- Twitter
- Facebook
- LinkedIn
- Email

**CSS2022**

RANKINGS OVER TIME

Retention, interest, usage, and awareness ratio rankings.



Runtime CSS-in-JS

Runtime CSS-in-JS

- Author styles in JS or TS
- Styles interpreted and applied at runtime

```
// @emotion/react (css prop), with object styles
export const Heading = ({ children }) =>(
  <div
    css={{
      background: "navy",
      color: "plum",
      padding: "1rem"
    }}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// @emotion/react (css prop), with object styles
export const Heading = ({ children }) =>(
  <div
    css={{
      background: "navy",
      color: "plum",
      padding: "1rem"
    }}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
export const Heading = ({ background, color, padding, children }) => (
  <div
    css={{
      background,
      color,
      padding
    }}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
export const Heading = ({ background, color, padding, children }) => (
  <div
    css={{
      background,
      color,
      padding
    }}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
export const Heading = ({ background, color, padding, children }) => (
  <div
    css={{
      background,
      color,
      padding
    }}
  >
    <h1>{ children }</h1>
    </div>
)
```

```
.css-asdfgh {  
background: "navy";  
color: "plum";  
space: "1rem";  
}
```

```
.css-asdfgh {  
background: "navy";  
color: "plum";  
space: "1rem";  
}
```



- DevX
- Co-location
- Locally scoped
- *Can be made typesafe*



- Runtime overhead
- SSR complexity
- Bundle size



Sam Magura
Posted on Oct 16, 2022 • Updated on Oct 25, 2022

Why We're Breaking Up with CSS-in-JS

#javascript #react #css #typescript

Hi, I'm Sam — software engineer at [Spot](#) and the 2nd most active maintainer of [Emotion](#), a widely-popular CSS-in-JS library for React. This post will delve into what originally attracted me to CSS-in-JS, and why I (along with the rest of the Spot team) have decided to shift away from it.

We'll start with an overview of CSS-in-JS and give an overview of its pros & cons. Then, we'll do a deep dive into the performance issues that CSS-in-JS caused at Spot and how you can avoid them.

What is CSS-in-JS?

Atomic CSS

Atomic CSS

- ~1:1 class to CSS property
- Classes are composed together in markup

```
export const Heading = ({ children }) => (
  <div className="bg-primary text-secondary p-4">
    <h1>{ children}</h1>
  </div>
)
```

```
export const Heading = ({ children }) => (
  <div className="bg-primary text-secondary p-4">
    <h1>{ children }</h1>
  </div>
)
```

```
.bg-primary {  
    background: "navy";  
}  
  
.text-secondary {  
    color: "plum";  
}  
  
.p-4 {  
    padding: 1rem;  
}
```

```
.bg-primary {  
    background: var(--color-primary);  
}  
  
.text-secondary {  
    color: var(--color-secondary);  
}  
  
.p-4 {  
    padding: 1rem;  
}  
  
:root {  
    --color-primary: "navy";  
    --color-secondary: "plum";  
}
```

```
import cx from 'classnames'

export const Heading = ({ background, color, space, children }) => (
  <div
    className={cx(
      backgroundColorMap[background],
      colorMap[color],
      spaceMap[space]
    )}
    >
    <h1>{ children }</h1>
  </div>
)
```

```
import cx from 'classnames'

export const Heading = ({ background, color, space, children }) => (
  <div
    className={cx(
      backgroundColorMap[background],
      colorMap[color],
      spaceMap[space]
    )}
    >
    <h1>{ children }</h1>
  </div>
)
```

```
import cx from 'classnames'

export const Heading = ({ background, color, space, children }) => (
  <div
    className={cx(
      backgroundColorMap[background],
      colorMap[color],
      spaceMap[space]
    )}
    >
    <h1>{ children }</h1>
  </div>
)
```

```
import cx from 'classnames'

export const Heading = ({ background, color, space, children }) => (
  <div
    className={cx(
      backgroundColorMap[background],
      colorMap[color],
      spaceMap[space]
    )}
    >
    <h1>{ children }</h1>
  </div>
)
```

```
const backgroundMap = {  
  primary: "bg-primary",  
  secondary: "bg-secondary",  
  tertiary: "bg-tertiary",  
  ...  
}  
  
const colorMap = {  
  ...  
}  
  
const spaceMap = {  
  ...  
}
```



- Static Stylesheet
- Finite CSS
- Bundle size



- Global scope
- Mental overhead
- DevX

CSS Modules

CSS Modules

- CSS-like syntax
- Build time CSS-in-JS

```
/* Heading.module.scss */
.root {
    background: "navy";
    color: "plum";
    padding: 1rem;
}
```

```
// Heading.tsx
import styles from './Heading.module.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ children }</h1>
  </div>
)
```

```
// Heading.tsx
import styles from './Heading.module.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ children }</h1>
  </div>
)
```

```
// Heading.tsx
import styles from './Heading.module.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ className }</h1>
  </div>
)
```

```
.Heading-root-cnq0T {  
    background: "navy";  
    color: "plum";  
    padding: 1rem;  
}
```



- Locally scoped
- Explicit dependencies
- Familiar syntax



- DevX
- Runtime theming
- Typesafety





- Performance



- Performance
- DevX



- Performance
- DevX
- Scoped styles



- Performance
- DevX
- Scoped styles
- Typesafe



- Performance
- DevX
- Scoped styles
- Typesafe
- Runtime theming



- Performance
- DevX
- Scoped styles
- Typesafe
- Runtime theming
- Atomic styles



***Zero-runtime
Stylesheets in
TypeScript.***

Use TypeScript as your preprocessor. Write type-safe,
locally scoped classes, variables and themes, then
generate static CSS files at build time.

```
$ npm install @vanilla-extract/css
```

[Get started](#)  ★ 7.1k

Integrations

- Vite
- esbuild
- Webpack
- Next.js
- Rollup
- Gatsby



style

Creates a style rule with a locally scoped class name.



New to styling with vanilla-extract? Make sure you've read the [styling overview](#) first.

This class can then be imported directly into your consuming application code, creating a first-class contract between your CSS and JavaScript.

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

● ○ ●

TS Heading.css.ts U X

src > components > Heading > TS Heading.css.ts > ...

```
1 import { style } from "@vanilla-extract/css";
2
3 export const root = style({
4   backgroundColor: "navy",
5   (property) backgroundColor?: CSSVarFunction |
6   Property.BackgroundColor | (CSSVarFunction |
7   Property.BackgroundColor | undefined)[] | undefined
8 })
```

The `background-color` CSS property sets the background color of an element.

Syntax: `<color>`

Initial value: `transparent`

Chrome Firefox Safari Edge IE

1	1	1	12	4
---	---	---	----	---

@see — <https://developer.mozilla.org/docs/Web/CSS/background-color>

TS Heading.css.ts 1, U X

src > components > Heading > TS Heading.css.ts > [e] root

```
1 import { style } from "@vanilla-extract/css";
2
3 export const root = style({
4   backgroundColor: "navy",
5   color: "plum",
6   padding: "1rem",
7 });
8
```

```
// ./components/Heading.tsx
import React from 'react'

export const Heading = ({ children }) => (
  <div>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import styles from './Heading.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import styles from './Heading.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import styles from './Heading.css'

export const Heading = ({ children }) => (
  <div className={ styles.root }>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { root } from './Heading.css'

export const Heading = ({ children }) => (
  <div className={ root }>
    <h1>{ children }</h1>
  </div>
)
```



Dev

```
.Heading_root_asdfg0 {  
background-color: "navy";  
color: "plum";  
padding: "1rem";  
}
```

Prod

```
._asdfg0 {  
background-color: "navy";  
color: "plum";  
padding: "1rem";  
}
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

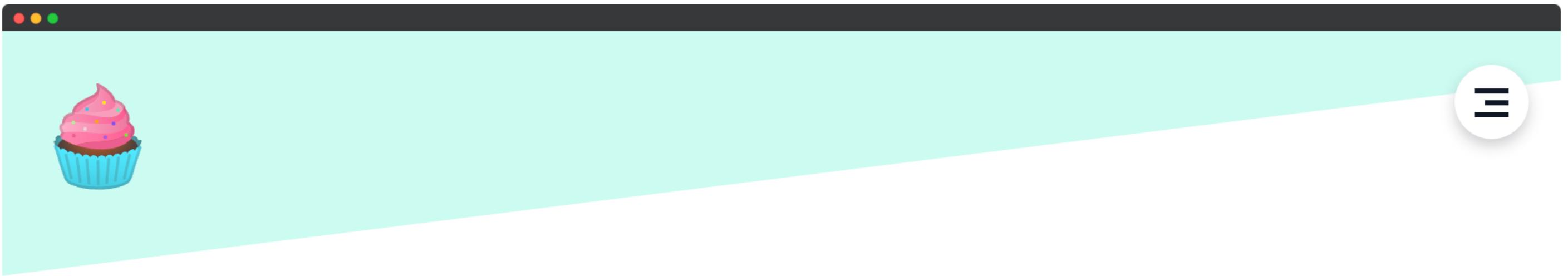
export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
    selectors: {
        "&:hover": {
            backgroundColor: "indigo"
        }
    }
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'

export const root = style({
  backgroundColor: "navy",
  color: "plum",
  padding: "1rem"
  selectors: {
    "&:hover": {
      backgroundColor: "indigo"
    }
  }
})
```



globalStyle

Creates styles attached to a global selector.

Requires a selector string as the first parameter, followed by a style object.

```
// global.css.ts
import { globalStyle } from '@vanilla-extract/css'

globalStyle('html, body', {
  margin: 0
})
```

```
html, body {  
    margin: 0;  
}
```

```
// global.css.ts
import { style, globalStyle } from '@vanilla-extract/css'

export const rootClass = style({})

globalStyle(`#${rootClass} > a`, {
  color: "navy"
})
```

```
// global.css.ts
import { style, globalStyle } from '@vanilla-extract/css'

export const rootClass = style({})

globalStyle(`#${rootClass} > a`, {
  color: "navy"
})
```

```
// global.css.ts
import { style, globalStyle } from '@vanilla-extract/css'

export const rootClass = style({})

globalStyle(`#${rootClass} > a`, {
  color: "navy"
})
```

```
// global.css.ts
import { style, globalStyle } from '@vanilla-extract/css'

export const rootClass = style({})

globalStyle(`#${rootClass} > a`, {
  color: "navy"
})
```

```
// global.css.ts
import { style, globalStyle } from '@vanilla-extract/css'

export const rootClass = style({})

globalStyle(`#${rootClass} > a`, {
  color: "navy"
})
```

```
.app_rootClass_asdfg0 {  
    color: "navy";  
}
```

```
// app.tsx
import { rootClass } from '@styles/global.css'

export const App = () => (
  <div className={rootClass}>
    ...
  </div>
)
```

```
// app.tsx
import { rootClass } from '@styles/global.css'

export const App = () => (
  <div className={rootClass}>
    ...
  </div>
)
```

```
// app.tsx
import { rootClass } from '@styles/global.css'

export const App = () => (
  <div className={rootClass}>
    ...
  </div>
)
```

```
// app.tsx
import { rootClass } from '@styles/global.css'

export const App = () => (
  <div className={rootClass}>
    ...
  </div>
)
```



styleVariants

Creates a collection of named style rules.

This is useful for mapping component props to styles, for example:

```
<button className={styles.background[props.variant]}>
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.tsx
import React from 'react'
import { variants } from './Heading.css'

export const Heading = ({ variant, children }) => (
  <div className={ variants[variant] }>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { variants } from './Heading.css'

export const Heading = ({ variant, children }) => (
  <div className={ variants[variant] }>
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { variants } from './Heading.css'

export const Heading = ({ variant, children }) => (
  <div className={ variants[variant] }>
    <h1>{ children }</h1>
  </div>
)
```

```
// app.tsx
import { Heading } from './components/Heading'

export const App = () => (
  <>
    <Heading variant="primary">🧁 Hello, CodeMash!</Heading>
  </>
);
```

```
// app.tsx
import { Heading } from './components/Heading'

export const App = () => (
  <Heading variant="primary">🧁 Hello, CodeMash!</Heading>
);


```

```
// app.tsx
import { Heading } from './components/Heading'

export const App = () => (
  <>
    <Heading variant="secondary">🧁 Hello, CodeMash!</Heading>
  </>
);
```



```
// ./components/Heading.tsx
import React from 'react'
import { variants } from './Heading.css'

type HeadingProps = {
  variant: keyof typeof variants
}

export const Heading = ({ variant, children }: HeadingProps) => (
  <div className={ variants[variant] }>
    <h1>{ children }</h1>
  </div>
)
```



TS App.tsx 1 ×

src > TS App.tsx > ...

```
1 import { Heading } from './components/Heading'  
2  
3 export const App = () => (  
4   <>  
5   | <Heading variant='tertiary'>🧁 Hello, CodeMash!</Heading>  
6   |</>  
7   | (property) HeadingProps.variant: "primary" | "secondary"  
8 );  
| variant: keyof typeof variants
```

Type '"tertiary"' is not assignable to type '"primary" | "secondary"'. ts(2322)

[Heading.tsx\(6, 3\):](#) The expected type comes from property 'variant' which is declared here on type 'IntrinsicAttributes & HeadingProps'

[View Problem](#) No quick fixes available

```
// ./components/Heading.css.ts
import { styleVariants } from '@vanilla-extract/css'

export const variants = styleVariants({
  primary: {
    backgroundColor: "navy",
    color: "plum",
    padding: "1rem"
  },
  secondary: {
    backgroundColor: "plum",
    color: "navy",
    padding: "1rem"
  },
})
```

```
// ./components/Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
// ./components/Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
// ./components/Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
// ./components/Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

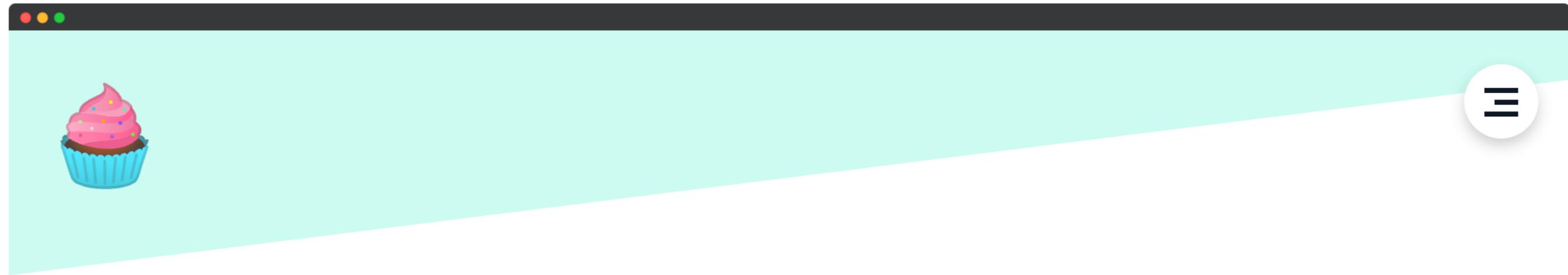
export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
// ./components/Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
.Heading_variants_primary_asdfg1 {  
    background-color: navy;  
    color: plum;  
}  
  
.Heading_space_asdfg0 {  
    padding: 1rem;  
}
```



createGlobalTheme

Creates a theme attached to a global selector, but with locally scoped variable names.



New to theming in vanilla-extract? Make sure you've read the [theming overview](#) first.

```
// theme.css.ts
import { createGlobalTheme } from "@vanilla-extract/css";

export const vars = createGlobalTheme(":root", {
  color: {
    primary: "navy",
    secondary: "plum",
  },
  space: {
    s: ".5rem",
    m: "1rem",
    l: "2rem",
  },
});
```

```
// theme.css.ts
import { createGlobalTheme } from "@vanilla-extract/css";

export const vars = createGlobalTheme(":root", {
  color: {
    primary: "navy",
    secondary: "plum",
  },
  space: {
    s: ".5rem",
    m: "1rem",
    l: "2rem",
  },
});
```

```
// theme.css.ts
import { createGlobalTheme } from "@vanilla-extract/css";

export const vars = createGlobalTheme(":root", {
  color: {
    primary: "navy",
    secondary: "plum",
  },
  space: {
    s: ".5rem",
    m: "1rem",
    l: "2rem",
  },
});
```

```
// theme.css.ts
import { createGlobalTheme } from "@vanilla-extract/css";

export const vars = createGlobalTheme(":root", {
  color: {
    primary: "navy",
    secondary: "plum",
  },
  space: {
    s: ".5rem",
    m: "1rem",
    l: "2rem",
  },
});
```

```
:root {  
  --color-primary__asdfg0: "navy";  
  --color-primary__asdfg1: "plum";  
  --space-s__asdfg2: "0.5rem";  
  --space-m__asdfg3: "1rem";  
  --space-l__asdfg4: "2rem";  
}
```

```
:root {  
  --color-primary__asdfg0: "navy";  
  --color-primary__asdfg1: "plum";  
  --space-s__asdfg2: "0.5rem";  
  --space-m__asdfg3: "1rem";  
  --space-l__asdfg4: "2rem";  
}
```

```
:root {  
  --color-primary__asdfg0: "navy";  
  --color-primary__asdfg1: "plum";  
  --space-s__asdfg2: "0.5rem";  
  --space-m__asdfg3: "1rem";  
  --space-l__asdfg4: "2rem";  
}
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'

const space = style({
  padding: "1rem",
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: "navy",
      color: "plum",
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: "plum",
      color: "navy",
    },
  ],
})
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

const space = style({
  padding: vars.space.s,
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: vars.color.primary,
      color: vars.color.secondary,
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: vars.color.secondary,
      color: vars.color.primary,
    },
  ],
})
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

const space = style({
  padding: vars.space.s,
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: vars.color.primary,
      color: vars.color.secondary,
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: vars.color.secondary,
      color: vars.color.primary,
    },
  ],
})
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

const space = style({
  padding: vars.space.s,
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: vars.color.primary,
      color: vars.color.secondary,
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: vars.color.secondary,
      color: vars.color.primary,
    },
  ],
})
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

const space = style({
  padding: vars.space.s,
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: vars.color.primary,
      color: vars.color.secondary,
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: vars.color.secondary,
      color: vars.color.primary,
    },
  ],
})
```

```
// Heading.css.ts
import { style, styleVariants } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

const space = style({
  padding: vars.space.s,
});

export const variants = styleVariants({
  primary: [
    space,
    {
      backgroundColor: vars.color.primary,
      color: vars.color.secondary,
    },
  ],
  secondary: [
    space,
    {
      backgroundColor: vars.color.secondary,
      color: vars.color.primary,
    },
  ],
})
```

```
.Heading_variants_primary_asdfg1 {  
    background-color: var(--color-primary_asdfg0);  
    color: var(--color-secondary_asdfg1);  
}  
  
.Heading_space_asdfg0 {  
    padding: var(--space-s_asdfg2);  
}
```

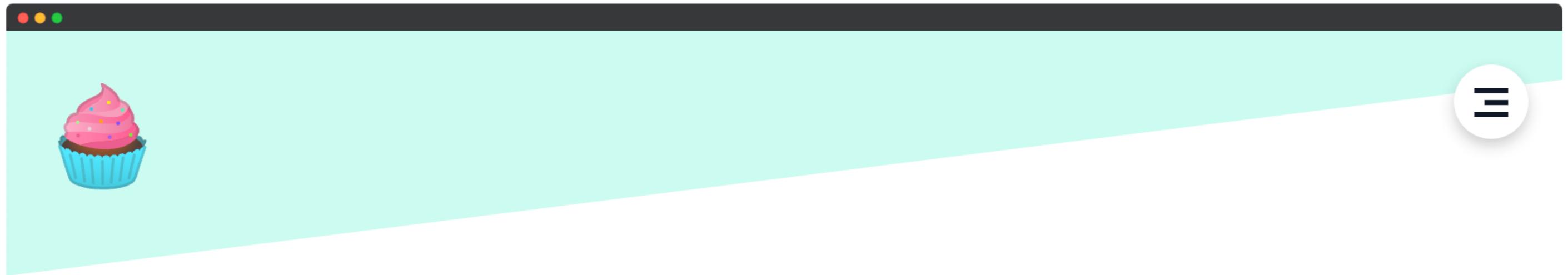
```
// theme.css.ts
import { createGlobalTheme } from "@vanilla-extract/css";

export const vars = createGlobalTheme(":root", {
  color: {
    primary: "navy",
    secondary: "plum",
  },
  space: {
    s: ".5rem",
    m: "1rem",
    l: "2rem",
  },
});
```



createTheme

Creates a locally scoped theme class and a theme contract which can be consumed within your styles.



Dynamic

A tiny (<1kB compressed) runtime for performing dynamic updates to scoped theme variables.

```
npm install @vanilla-extract/dynamic
```



- Performance ✓



- Performance ✓
- DevX ✓



- Performance ✓
- DevX ✓
- Scoped styles ✓



- Performance ✓
- DevX ✓
- Scoped styles ✓
- Typesafe ✓

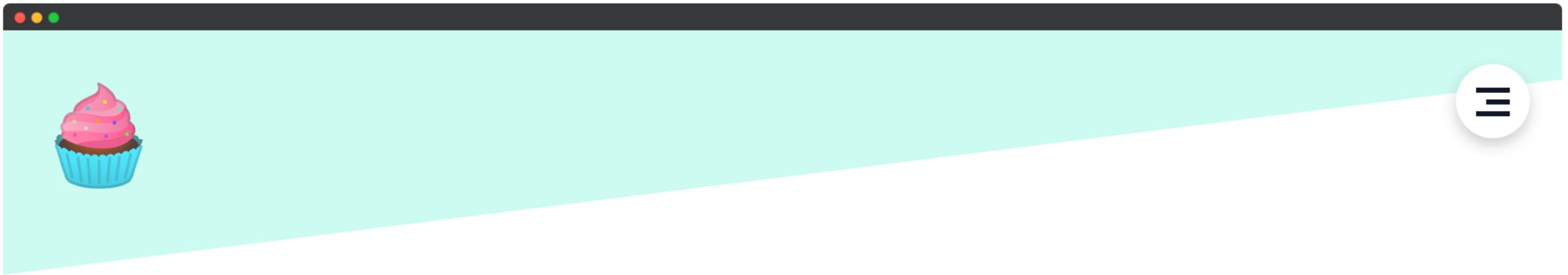


- Performance ✓
- DevX ✓
- Scoped styles ✓
- Typesafe ✓
- Runtime theming ✓



- Performance ✓
- DevX ✓
- Scoped styles ✓
- Typesafe ✓
- Runtime theming ✓
- Atomic styles ?





Sprinkles

A zero-runtime atomic CSS framework for vanilla-extract.

Generate a static set of custom utility classes and compose them either statically at build time, or dynamically at runtime, without the usual style generation overhead of CSS-in-JS.

Basically, it's like building your own zero-runtime, type-safe version of [Tailwind](#), [Styled System](#), etc.

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    mobile: {},
    tablet: { "@media": "screen and (min-width: 768px)" },
    desktop: { "@media": "screen and (min-width: 1024px)" },
  },
  defaultCondition: "mobile",
  properties: {
    color: vars.color,
    backgroundColor: vars.color,
    padding: vars.space,
  }
});
export const sprinklesFn = createSprinkles(properties);
```

```
.sprinkles_color_primary_mobile_asdfg0 {  
  color: var(--color-primary_asdfg0)  
}  
.sprinkles_color_secondary_mobile_asdfg0 {  
  color: var(--color-secondary_asdfg1)  
}  
.sprinkles_backgroundColor_primary_mobile_asdfg0 {  
  background-color: var(--color-primary_asdfg2)  
}  
.sprinkles_backgroundColor_secondary_mobile_asdfg0 {  
  background-color: var(--color-secondary_asdfg3)  
}  
.sprinkles_space_s_mobile_asdfg0 {  
  padding: var(--space-s_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}
```

```
.sprinkles_color_primary_mobile_asdfg0 {  
  color: var(--color-primary_asdfg0)  
}  
.sprinkles_color_secondary_mobile_asdfg0 {  
  color: var(--color-secondary_asdfg1)  
}  
.sprinkles_backgroundColor_primary_mobile_asdfg0 {  
  background-color: var(--color-primary_asdfg2)  
}  
.sprinkles_backgroundColor_secondary_mobile_asdfg0 {  
  background-color: var(--color-secondary_asdfg3)  
}  
.sprinkles_space_s_mobile_asdfg0 {  
  padding: var(--space-s_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}
```

```
.sprinkles_color_primary_mobile_asdfg0 {  
  color: var(--color-primary_asdfg0)  
}  
.sprinkles_color_secondary_mobile_asdfg0 {  
  color: var(--color-secondary_asdfg1)  
}  
.sprinkles_backgroundColor_primary_mobile_asdfg0 {  
  background-color: var(--color-primary_asdfg2)  
}  
.sprinkles_backgroundColor_secondary_mobile_asdfg0 {  
  background-color: var(--color-secondary_asdfg3)  
}  
.sprinkles_space_s_mobile_asdfg0 {  
  padding: var(--space-s_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}
```

```
.sprinkles_color_primary_mobile_asdfg0 {  
  color: var(--color-primary_asdfg0)  
}  
.sprinkles_color_secondary_mobile_asdfg0 {  
  color: var(--color-secondary_asdfg1)  
}  
.sprinkles_backgroundColor_primary_mobile_asdfg0 {  
  background-color: var(--color-primary_asdfg2)  
}  
.sprinkles_backgroundColor_secondary_mobile_asdfg0 {  
  background-color: var(--color-secondary_asdfg3)  
}  
.sprinkles_space_s_mobile_asdfg0 {  
  padding: var(--space-s_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}
```

```
.sprinkles_color_primary_mobile_asdfg0 {  
  color: var(--color-primary_asdfg0)  
}  
.sprinkles_color_secondary_mobile_asdfg0 {  
  color: var(--color-secondary_asdfg1)  
}  
.sprinkles_backgroundColor_primary_mobile_asdfg0 {  
  background-color: var(--color-primary_asdfg2)  
}  
.sprinkles_backgroundColor_secondary_mobile_asdfg0 {  
  background-color: var(--color-secondary_asdfg3)  
}  
.sprinkles_space_s_mobile_asdfg0 {  
  padding: var(--space-s_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}  
.sprinkles_space_m_mobile_asdfg0 {  
  padding: var(--space-m_asdfg4)  
}
```

```
@media screen and (min-width: 768px) {  
  .sprinkles_color_primary_tablet_asdfg0 {  
    color: var(--color-primary_asdfg0)  
  }  
  .sprinkles_color_secondary_tablet_asdfg0 {  
    color: var(--color-secondary_asdfg1)  
  }  
  .sprinkles_backgroundColor_primary_tablet_asdfg0 {  
    background-color: var(--color-primary_asdfg2)  
  }  
  .sprinkles_backgroundColor_secondary_tablet_asdfg0 {  
    background-color: var(--color-secondary_asdfg3)  
  }  
  .sprinkles_space_s_tablet_asdfg0 {  
    padding: var(--space-s_asdfg4)  
  }  
  .sprinkles_space_m_tablet_asdfg0 {  
    padding: var(--space-m_asdfg4)  
  }  
  .sprinkles_space_m_tablet_asdfg0 {  
    padding: var(--space-m_asdfg4)  
  }  
}
```

```
@media screen and (min-width: 1024px) {  
  .sprinkles_color_primary_desktop_asdfg0 {  
    color: var(--color-primary_asdfg0)  
  }  
  .sprinkles_color_secondary_desktop_asdfg0 {  
    color: var(--color-secondary_asdfg1)  
  }  
  .sprinkles_backgroundColor_primary_desktop_asdfg0 {  
    background-color: var(--color-primary_asdfg2)  
  }  
  .sprinkles_backgroundColor_secondary_desktop_asdfg0 {  
    background-color: var(--color-secondary_asdfg3)  
  }  
  .sprinkles_space_s_desktop_asdfg0 {  
    padding: var(--space-s_asdfg4)  
  }  
  .sprinkles_space_m_desktop_asdfg0 {  
    padding: var(--space-m_asdfg4)  
  }  
  .sprinkles_space_m_desktop_asdfg0 {  
    padding: var(--space-m_asdfg4)  
  }  
}
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
    // ...
    properties: {
        // ...
        paddingTop: vars.space,
        paddingBottom: vars.space,
        paddingLeft: vars.space,
        paddingRight: vars.space,
    }
    shorthands: {
        padding: ["paddingTop", "paddingBottom", "paddingLeft", "paddingRight"],
        paddingX: ["paddingLeft", "paddingRight"],
        paddingY: ["paddingTop", "paddingBottom"],
    },
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
    // ...
    properties: {
        // ...
        paddingTop: vars.space,
        paddingBottom: vars.space,
        paddingLeft: vars.space,
        paddingRight: vars.space,
    }
    shorthands: {
        padding: ["paddingTop", "paddingBottom", "paddingLeft", "paddingRight"],
        paddingX: ["paddingLeft", "paddingRight"],
        paddingY: ["paddingTop", "paddingBottom"],
    },
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
    // ...
    properties: {
        // ...
        paddingTop: vars.space,
        paddingBottom: vars.space,
        paddingLeft: vars.space,
        paddingRight: vars.space,
    }
    shorthands: {
        padding: ["paddingTop", "paddingBottom", "paddingLeft", "paddingRight"],
        paddingX: ["paddingLeft", "paddingRight"],
        paddingY: ["paddingTop", "paddingBottom"],
    },
});
export const sprinklesFn = createSprinkles(properties);
```

```
// sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
    // ...
    properties: {
        // ...
        paddingTop: vars.space,
        paddingBottom: vars.space,
        paddingLeft: vars.space,
        paddingRight: vars.space,
    }
    shorthands: {
        padding: ["paddingTop", "paddingBottom", "paddingLeft", "paddingRight"],
        paddingX: ["paddingLeft", "paddingRight"],
        paddingY: ["paddingTop", "paddingBottom"],
    },
});
export const sprinklesFn = createSprinkles(properties);
```

```
// Heading.css.ts
import { style } from '@vanilla-extract/css'
import { vars } from '../..//styles/theme.css'

export const variants = style({
  backgroundColor: vars.color.primary,
  color: vars.color.secondary,
  padding: vars.space.s,
})
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const variants = style([
  sprinklesFn({
    backgroundColor: "primary",
    color: "secondary",
    padding: "s"
  })
])
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const variants = style([
  sprinklesFn({
    backgroundColor: "primary",
    color: "secondary",
    padding: "s"
  })
])
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const variants = style([
  sprinklesFn({
    backgroundColor: "primary",
    color: "secondary",
    padding: "s"
  })
])
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingRight_s_mobile_asdfg13 {
  padding-right: var(--space-s_asdfg3);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
.sprinkles_paddingBottom_s_mobile_asdfgl {
  padding-bottom: var(--space-s_asdfg3);
}
.sprinkles_paddingTop_s_mobile_asdfgc {
  padding-top: var(--space-s_asdfg3);
}
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingRight_s_mobile_asdfg13 {
  padding-right: var(--space-s_asdfg3);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
.sprinkles_paddingBottom_s_mobile_asdfgl {
  padding-bottom: var(--space-s_asdfg3);
}
.sprinkles_paddingTop_s_mobile_asdfgc {
  padding-top: var(--space-s_asdfg3);
}
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {  
    background-color: var(--color-primary_asdfg1);  
}  
.sprinkles_color_secondary_mobile_asdfg3 {  
    color: var(--color-secondary_asdfg2);  
}  
.sprinkles_paddingRight_s_mobile_asdfg13 {  
    padding-right: var(--space-s_asdfg3);  
}  
.sprinkles_paddingLeft_s_mobile_asdfgu {  
    padding-left: var(--space-s_asdfg3);  
}  
.sprinkles_paddingBottom_s_mobile_asdfgl {  
    padding-bottom: var(--space-s_asdfg3);  
}  
.sprinkles_paddingTop_s_mobile_asdfgc {  
    padding-top: var(--space-s_asdfg3);  
}
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const variants = style([
  sprinklesFn({
    backgroundColor: "primary",
    color: "secondary",
    paddingX: "s"
  })
])
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {  
    background-color: var(--color-primary_asdfg1);  
}  
.sprinkles_color_secondary_mobile_asdfg3 {  
    color: var(--color-secondary_asdfg2);  
}  
.sprinkles_paddingRight_s_mobile_asdfg13 {  
    padding-right: var(--space-s_asdfg3);  
}  
.sprinkles_paddingLeft_s_mobile_asdfgu {  
    padding-left: var(--space-s_asdfg3);  
}
```

```
// ./components/Heading.css.ts
import { style } from '@vanilla-extract/css'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const variants = style([
  sprinklesFn({
    backgroundColor: "primary",
    color: "secondary",
    paddingLeft: {
      mobile: "s",
      tablet: "m",
      desktop: "l"
    }
  })
])
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
@media screen and (min-width: 768px) {
  .sprinkles_paddingLeft_s_tablet_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
@media screen and (min-width: 1024px) {
  .sprinkles_paddingLeft_s_desktop_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
@media screen and (min-width: 768px) {
  .sprinkles_paddingLeft_s_tablet_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
@media screen and (min-width: 1024px) {
  .sprinkles_paddingLeft_s_desktop_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
@media screen and (min-width: 768px) {
  .sprinkles_paddingLeft_s_tablet_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
@media screen and (min-width: 1024px) {
  .sprinkles_paddingLeft_s_desktop_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
```

```
.sprinkles_backgroundColor_primary_mobile_asdfg6 {
  background-color: var(--color-primary_asdfg1);
}
.sprinkles_color_secondary_mobile_asdfg3 {
  color: var(--color-secondary_asdfg2);
}
.sprinkles_paddingLeft_s_mobile_asdfgu {
  padding-left: var(--space-s_asdfg3);
}
@media screen and (min-width: 768px) {
  .sprinkles_paddingLeft_s_tablet_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
@media screen and (min-width: 1024px) {
  .sprinkles_paddingLeft_s_desktop_asdfgu {
    padding-left: var(--space-s_asdfg3);
  }
}
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const Heading = ({ children }) => (
  <div
    className={sprinklesFn({
      backgroundColor: "primary",
      color: "secondary",
      padding: "s"
    })}
  >
  <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn } from '../..//styles/sprinkles.css'

export const Heading = ({ children }) => (
  <div
    className={sprinklesFn({
      backgroundColor: "primary",
      color: "secondary",
      padding: "s"
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn } from '../../styles/sprinkles.css'

export const Heading = ({ children }) => (
  <div
    className={sprinklesFn({
      backgroundColor: "primary",
      color: "secondary",
      padding: "s"
    })}
  >
  <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn } from '../../styles/sprinkles.css'

export const Heading = ({ children }) => (
  <div
    className={sprinklesFn({
      backgroundColor: "primary",
      color: "secondary",
      padding: "s"
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./styles/sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    ...
  },
  defaultCondition: "mobile",
  properties: {
    ...
  },
  shorthands: {
    ...
  },
});
export const sprinklesFn = createSprinkles(properties);
export type Sprinkles = Parameters<typeof sprinklesFn>[0];
```

```
// ./styles/sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    ...
  },
  defaultCondition: "mobile",
  properties: {
    ...
  },
  shorthands: {
    ...
  },
});
export const sprinklesFn = createSprinkles(properties);

export type Sprinkles = Parameters<typeof sprinklesFn>[0];
```

```
// ./styles/sprinkles.css.ts
import { defineProperties, createSprinkles } from "@vanilla-extract/sprinkles";
import { vars } from "./theme.css";

const properties = defineProperties({
  conditions: {
    ...
  },
  defaultCondition: "mobile",
  properties: {
    ...
  },
  shorthands: {
    ...
  },
});
export const sprinklesFn = createSprinkles(properties);

export type Sprinkles = Parameters<typeof sprinklesFn>[0];
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../../../../../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../..../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../../../../../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../../../../../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../../../../../styles/sprinkles.css'

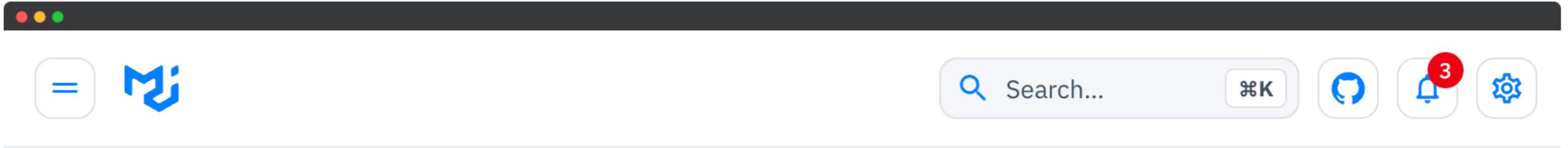
type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../..../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

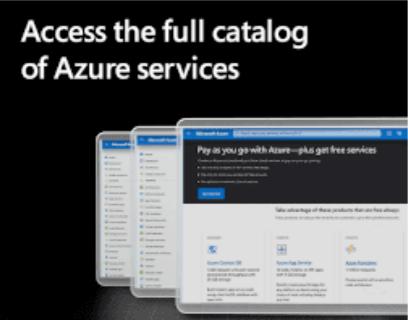
export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```



[Edit this page](#)

Box

The Box component serves as a wrapper component for most of the CSS utility needs.



Pay only for the services you use beyond free amounts. Cancel anytime.
ads via Carbon

The Box component packages [all the style functions](#) that are exposed in `@mui/system`.



Search the docs

K

v2.4.6



Sponsor

Getting Started

Styled System

Components

Hooks

Community

Changelog

Blog

LAYOUT

AspectRatio

Box

Center

Container

Flex

Grid

SimpleGrid

Stack

Wrap

FORMS

Box

Box is the most abstract component on top of which all other Chakra UI components are built. By default, it renders a `div` element

Source

NPM @chakra-ui/layout

Usage

Props

Import

```
import { Box } from '@chakra-ui/react'
```

COPY

Usage

The Box component is useful because it helps with three common use cases:

- Create responsive layouts with ease.
- Provide a shorthand way to pass styles via props (`bg` instead of `backgroundColor`).
- Compose new component and allow for override using the `as` prop.

```
// ./components/Box.tsx
import React from 'react'

type BoxProps = {
    children: React.ReactNode;
};

export const Box = ({ children }: BoxProps) => (
    <div>{ children }</div>
);
```

```
// ./components/Box.tsx
import React from 'react'

type BoxProps = {
  as: React.ElementType
  children: React.ReactNode;
};

export const Box = ({ as: As, children }: BoxProps) => (
  <As>{ children }</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

type BoxProps = {
  as: React.ElementType
  children: React.ReactNode;
};

export const Box = ({ as: As, children }: BoxProps) => (
  <As>{ children }</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

type BoxProps = {
  as: React.ElementType
  children: React.ReactNode;
};

export const Box = ({ as: As, children }: BoxProps) => (
  <As>{ children }</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

import { sprinklesFn, Sprinkles } from "../../styles/sprinkles.css";

type BoxProps = {
  as: React.ElementType;
  children: React.ReactNode;
  sprinkles: Sprinkles;
};

export const Box = ({ as: As, sprinkles, children }: BoxProps) => (
  <As className={sprinklesFn(sprinkles)}>{children}</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

import { sprinklesFn, Sprinkles } from "../../styles/sprinkles.css";

type BoxProps = {
  as: React.ElementType;
  children: React.ReactNode;
  sprinkles: Sprinkles;
};

export const Box = ({ as: As, sprinkles, children }: BoxProps) => (
  <As className={sprinklesFn(sprinkles)}>{children}</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

import { sprinklesFn, Sprinkles } from "../../styles/sprinkles.css";

type BoxProps = {
  as: React.ElementType;
  children: React.ReactNode;
  sprinkles: Sprinkles;
};

export const Box = ({ as: As, sprinkles, children }: BoxProps) => (
  <As className={sprinklesFn(sprinkles)}>{children}</As>
);
```

```
// ./components/Box.tsx
import React from 'react'

import { sprinklesFn, Sprinkles } from "../../styles/sprinkles.css";

type BoxProps = {
  as: React.ElementType;
  children: React.ReactNode;
  sprinkles: Sprinkles;
};

export const Box = ({ as: As, sprinkles, children }: BoxProps) => (
  <As className={sprinklesFn(sprinkles)}>{children}</As>
);
```

```
// ./components/Heading.tsx
import React from 'react'
import { sprinklesFn, Sprinkles } from '../..../styles/sprinkles.css'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <div
    className={sprinklesFn({
      backgroundColor,
      color,
      padding
    })}
  >
    <h1>{ children }</h1>
  </div>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

```
// ./components/Heading.tsx
import React from 'react'
import { Sprinkles } from '../..//styles/sprinkles.css'
import { Box } from './Box'

type HeadingProps = {
  backgroundColor?: Sprinkles['backgroundColor'];
  color?: Sprinkles['color']
  padding?: Sprinkles['padding']
}

export const Heading = ({ backgroundColor, color, padding, children }: HeadingProps) => (
  <Box sprinkles={{ backgroundColor, color, padding }}>
    <h1>{ children }</h1>
  </Box>
)
```

The image shows a screenshot of a GitHub repository page for "TheMightyPenguin/dessert-box". The repository is public and has 4 issues, 4 pull requests, and 220 stars. The "Code" tab is selected. The main file shown is "dessert-box / README.md".

Code Issues 4 Pull requests 4 Actions Projects Security Insights

main dessert-box / README.md Go to file ...

 TheMightyPenguin Add escape hatch for Box Latest commit 1d99a5c on Jun 5, 2022 History

3 contributors 

269 lines (189 sloc) | 9.1 KB

 **Dessert Box**

A library to easily consume your design tokens from a React component, meant to be used with [vanilla-extract](#). This library will make consuming your [sprinkles](#) from a react component a breeze. It provides a zero-CSS-runtime `<Box />` component (similar to the one in [Braid](#) or [Chakra](#)). Try it on [CodeSandbox!](#) It works by consuming `atoms` created with [vanilla-extract](#) and [sprinkles](#). Shout out to the team at Seek for making these awesome libraries!

- Performance ✓
- DevX ✓
- Scoped styles ✓
- Typesafe ✓
- Runtime theming ✓
- Atomic styles ✓



Drawbacks

- Atomic classes purging
- Dynamic inline style assignments

vanilla-extract-css / vanilla-extract Public

Watch 35 Fork 191 Starred 7.2k

Code Issues 46 Pull requests 26 Discussions Actions Security Insights

feat: integration callbacks #942

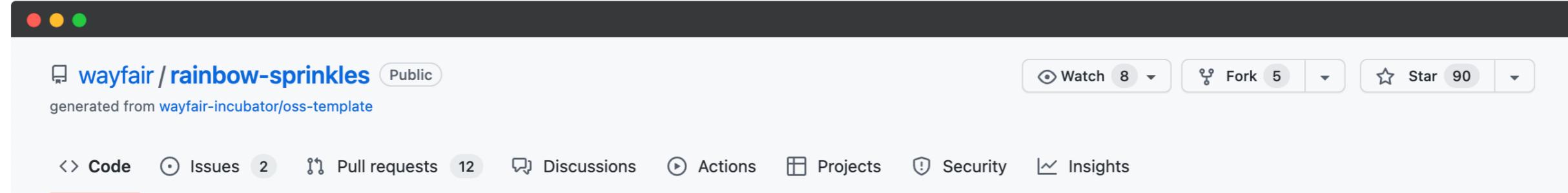
Open astahmer wants to merge 13 commits into [vanilla-extract-css:master](#) from [astahmer:feat/integration-callbacks-with-tgz](#)

Conversation 2 Commits 13 Checks 0 Files changed 16 +234 -57

astahmer commented on Dec 9, 2022 • edited

hey, I made a vite plugin that allows purging any unused style generated by vanilla-extract/sprinkles

Reviewers No reviews

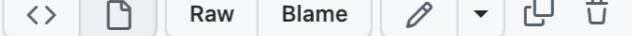

wayfair / rainbow-sprinkles Public
generated from [wayfair-incubator/oss-template](#)

[Code](#) [Issues 2](#) [Pull requests 12](#) [Discussions](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

 main  [rainbow-sprinkles / packages / rainbow-sprinkles / README.md](#) [Go to file](#) [...](#)

 [Robin Farrer](#) update readme ✓ Latest commit 51822b9 3 weeks ago [History](#)

 2 contributors 

 201 lines (160 sloc) | 6.67 KB 

Rainbow Sprinkles

Dynamic, theme-driven, style props for [vanilla-extract](#)

[release](#) [rainbow-sprinkles@0.15.1](#) [License](#) [MIT](#) [Contributor Covenant 2.0](#) [Maintainer](#) [Wayfair](#)

Rainbow sprinkles works similarly to [@vanilla-extract/sprinkles](#). Like sprinkles, it generates custom CSS utility classes at build time. While sprinkles requires a pre-defined list of available values, Rainbow Sprinkles uses CSS custom properties to allow dynamic values using inline style variable assignments.

Compared to sprinkles:

- **Rainbow sprinkles ships a fraction of the CSS.** For each property, Sprinkles produces CSS that's a factor of [\[pre-defined values\]](#) * [\[possible conditions\]](#). Rainbow sprinkles produces CSS that only scales with the number of conditions.
- **Supports dynamic values.** Rainbow Sprinkles uses dynamic inline style assignments to set the value of each property. You still get the TypeScript editor suggestions, but the ability to use any valid CSS value for that property.

Modern TS Web App

with Vanilla-extract

