

Zephyr

A deep dive into our design system at Air

Goals

- [Zephyr](#) is a source of truth for all teams
- Designers work closer to implementation
- Developers move faster

Organization

Zephyr consumes our monorepo packages the same way our web app does.

- **packages**
 - **constants** - Cross-platform foundations (colors, typography)
 - **components** - Shared components
 - **icons** - Cross-platform icons
 - **web** - Web app
 - **zephyr** - Documentation site

Primitives

All our Zephyr components are built on smaller, foundational components, called primitives.

Box is our most primitive primitive.

```
<Box mx="auto" bg="jay9" />
```

Primitives

Other primitives (that are built with the **Box** primitive) include:

```
<Flex />  
<Grid />  
<NewText />
```

Components

Zephyr includes a component library that has things like:

```
<Avatar />  
<Banner />  
<Popover />  
<NewButton />  
<ZeroState />
```

...and many more

Components

Let's take a look at a stripped-down version of our [Banner component](#) that shows how primitives are being used.

```
const flavorMap: {
  [key in BannerFlavor]: { color: ColorName; bg: ColorName } } = {
  danger: { color: 'white', bg: 'flamingo6' },
  caution: { color: 'black', bg: 'canary3' },
  success: { color: 'jay9', bg: 'peacock4' },
};

const Banner: FC<BannerProps> = ({ flavor, message, action }) => {
  return (
    <Box as="section" p={12} {...flavorMap[flavor]}>
      <NewText>{message}</NewText>
      <NewButton variant={14}>{action.label}</NewButton>
    </Box>
  );
};
```


Theme

Our global [theme](#) includes our specific branded and chosen constraints used to construct our apps.

```
theme.space[8];  
theme.colors.jay5;  
theme.fontFamilies.display;
```

Theme

Types are built into our theme!

```
<Box color=""
```

You, a few seconds ago • Uncommitted changes

- canary3 canary3
- canary4
- canary5
- canary6
- canary7
- canary8
- canary9
- dropbox
- flamingo0
- flamingo1
- flamingo2
- flamingo3

Theme

Every theme value can be used as a responsive value.

Add the values to an array where the index represents our [mobile-first breakpoints](#), starting with 0px.

```
<Box py={[16, 20, 32, 64]} />  
<NewText variant={['ui-16', 'ui-20']}>
```

Props

```
<Box p={4} /> === <Box padding={4} />;  
<Box px={8} /> === <Box paddingRight={8} paddingLeftt={8} />  
<Box mt={16} /> === <Box marginTop={16} />
```

Props

Types are built into our props!

You, a few seconds ago • Uncommitted changes

- alignContent
- alignItems
- alignSelf
- textAlign
- verticalAlign

```
(JSX attribute) AlignItemsProps.a  
s?: string | (string | null)[] |  
[key: string]: string;  
} | undefined
```

The CSS align-items property sets the alignment of all direct children as a group. The align-self sets the alignment of an item within its container.

In Flexbox it controls the alignment of items on the Cross Axis, in Grid Layout it controls the alignment of items on the Block Axis within their grid area.

TypeScript

One factor that sets our design system apart is that both our props and values are typed.

Foundations

- [Colors](#)
- [Typography](#)
- [Iconography](#)

Testing

Tests should be written using [Testing Library](#).

Currently [tests](#) are written using [Enzyme](#), but will be updated.

Testing

```
describe(`[state]: Viewing`, () => {
  describe(`Container`, () => {
    it(`should be visible and have the correct "danger" colors`, () => {
      buildWrapper({ flavor: 'danger', message: "Here's a banner message." });
      expect(ui.getContainer().exists()).toBe(true);
      expect(ui.getContainer()).toHaveStyleRule('color', 'white');
      expect(ui.getContainer()).toHaveStyleRule('background-color', 'flamingo6');
    });

    it(`should be visible and have the correct "caution" colors`, () => {
      buildWrapper({ flavor: 'caution', message: "Here's a banner message." });
      expect(ui.getContainer().exists()).toBe(true);
      expect(ui.getContainer()).toHaveStyleRule('color', 'black');
      expect(ui.getContainer()).toHaveStyleRule('background-color', 'canary3');
    });

    it(`should be visible and have the correct "success" colors`, () => {
      buildWrapper({ flavor: 'success', message: "Here's a banner message." });
      expect(ui.getContainer().exists()).toBe(true);
      expect(ui.getContainer()).toHaveStyleRule('color', 'jay9');
      expect(ui.getContainer()).toHaveStyleRule('background-color', 'peacock4');
    });
  });
});
```

Resources

- [Zephyr](#)
- [Download these slides as a PDF](#)