STYLED SYSTEM

Introduction

Styled System

The collection of utility functions that add style props to the design system based on React components and allows you to control styles based on a global theme object with typographic scales, colors, and layout properties.

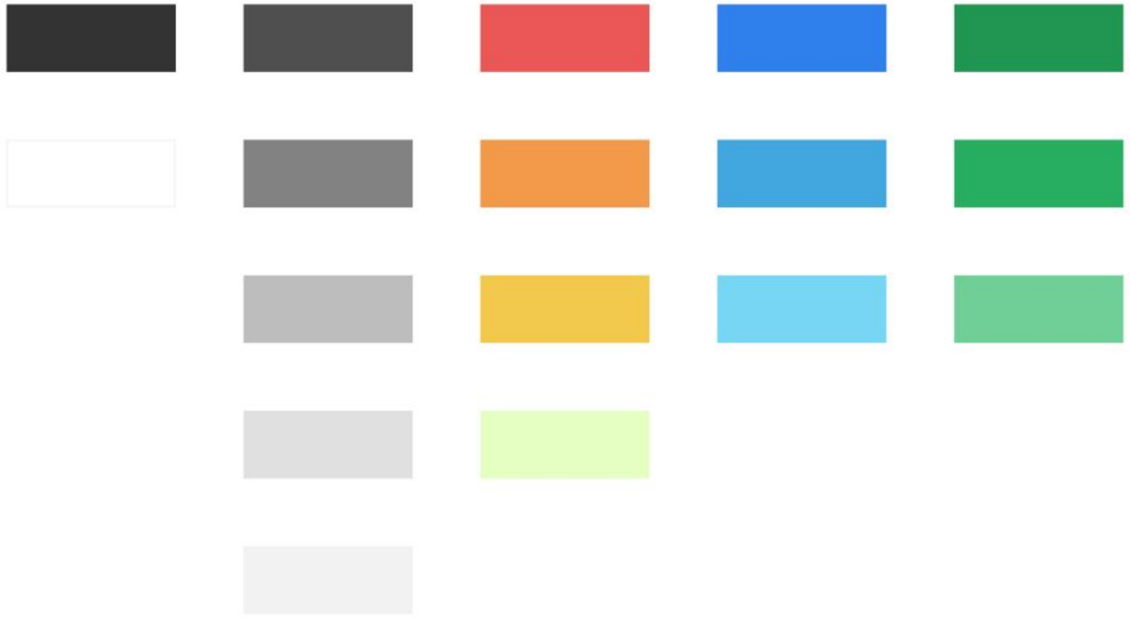
Features

- Add style props that hook into the theme
- Quickly set responsive font-size, margin, padding, width, and more with props
- Influenced by constraint-based design system principles
- Typographic scale
- Spacing scale for margin and padding
- Works with any color palette
- Works with most css-in-js libraries, including styled-components & emotion
- Used in Rebass, Reflexbox, and the Priceline Design System

Contents

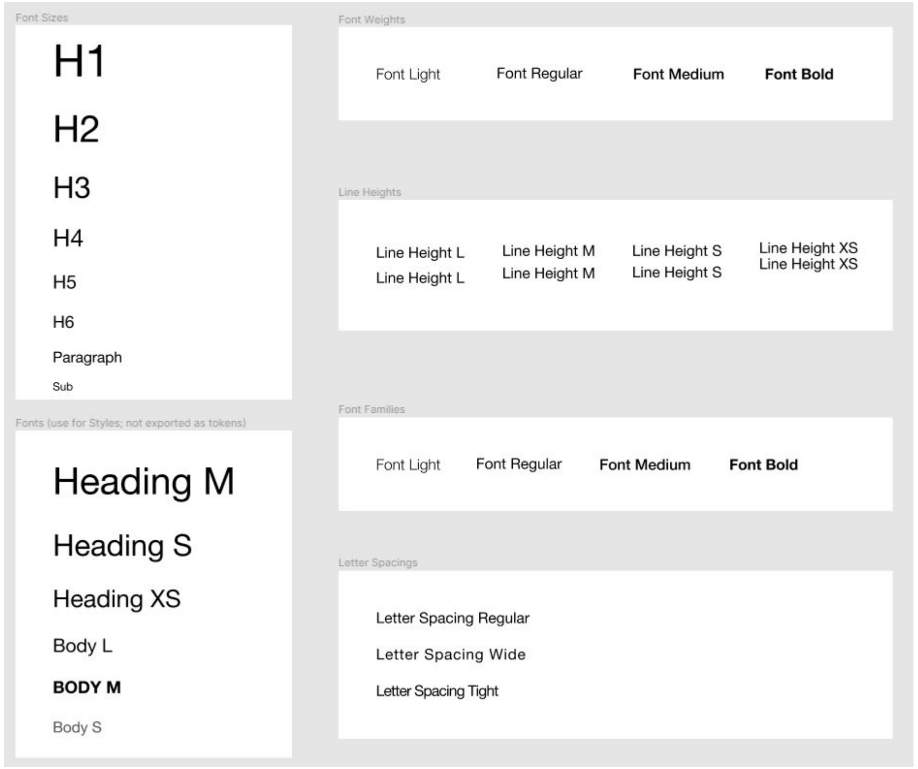
Design System

Color Palette



Design System

Typographic Scale



Font Sizes

H1

H2

H3

H4

H5

H6

Paragraph

Sub

Fonts (use for Styles; not exported as tokens)

Heading M

Heading S

Heading XS

Body L

BODY M

Body S

Font Weights

Font Light Font Regular **Font Medium** **Font Bold**

Line Heights

Line Height L
Line Height L

Line Height M
Line Height M

Line Height S
Line Height S

Line Height XS
Line Height XS

Font Families

Font Light Font Regular **Font Medium** **Font Bold**

Letter Spacings

Letter Spacing Regular

Letter Spacing Wide

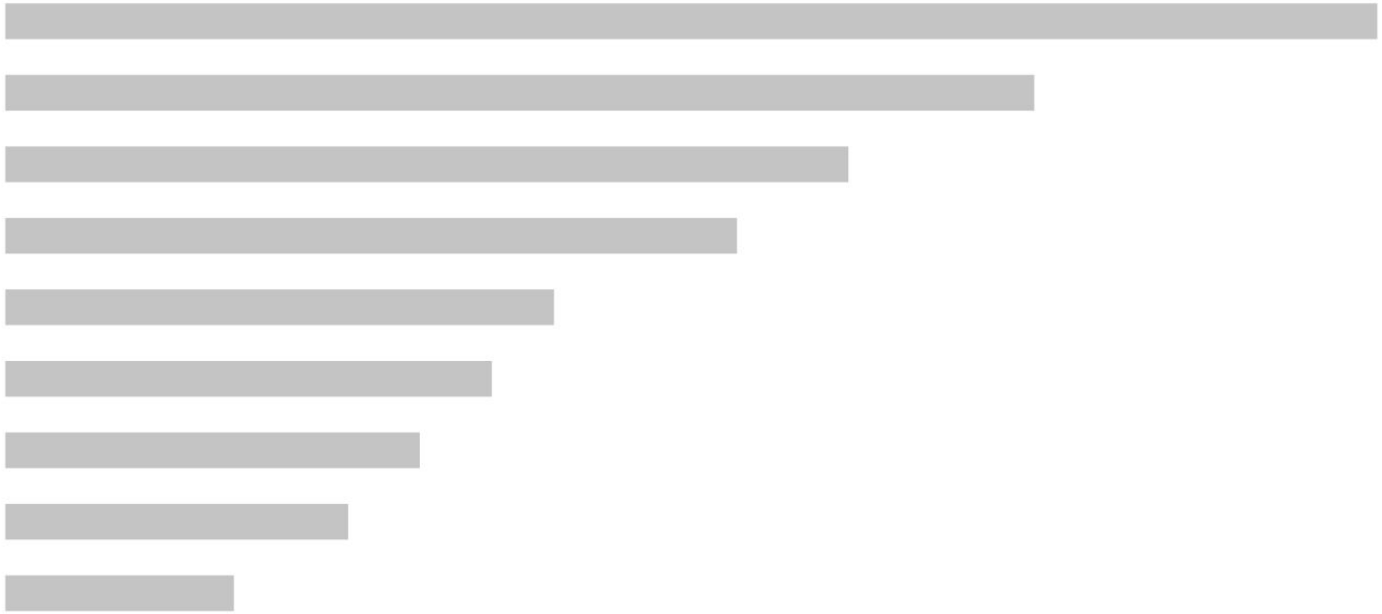
Letter Spacing Tight

Design System

Spacing

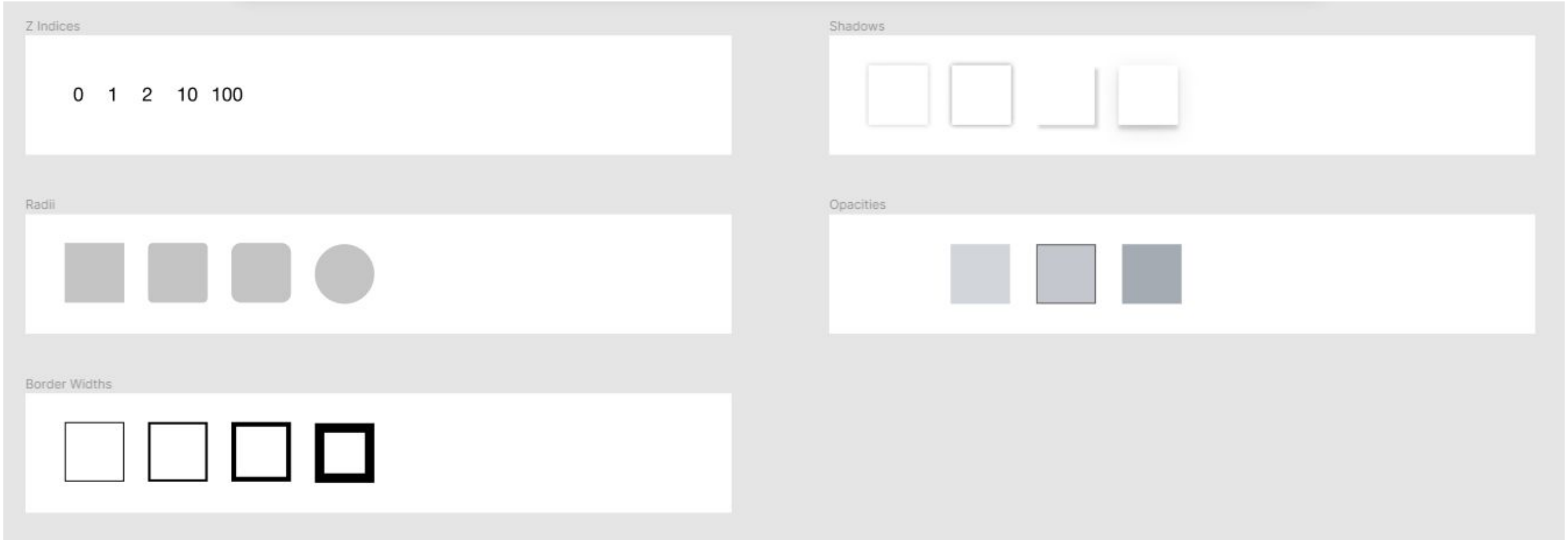
Design System

Layout



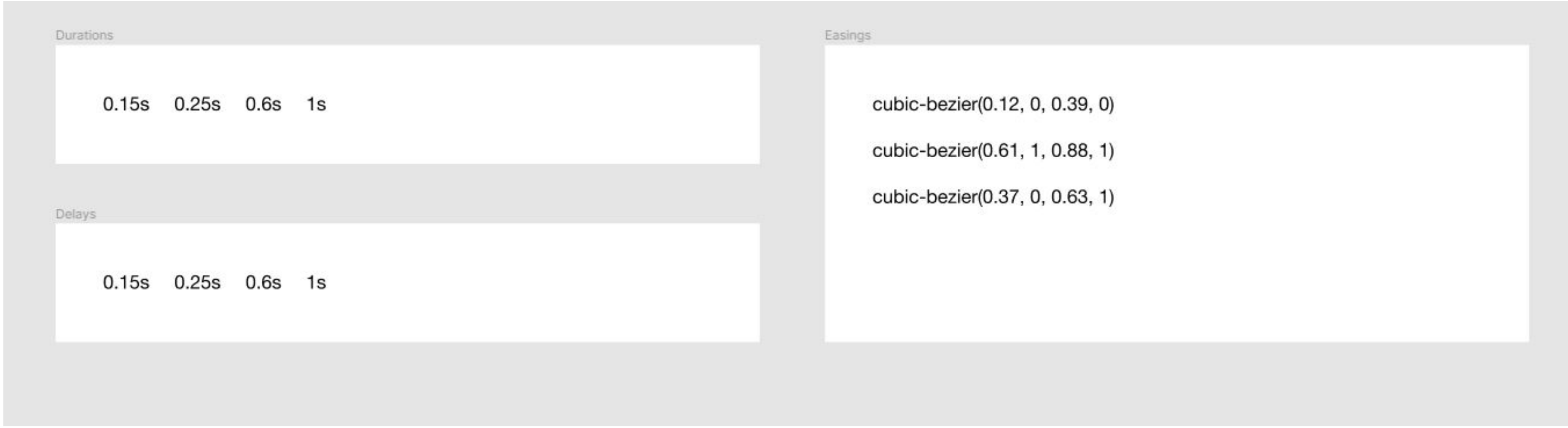
Design System

Others



Design System

Others



Durations

0.15s 0.25s 0.6s 1s

Delays

0.15s 0.25s 0.6s 1s

Easings

cubic-bezier(0.12, 0, 0.39, 0)

cubic-bezier(0.61, 1, 0.88, 1)

cubic-bezier(0.37, 0, 0.63, 1)

Design System

To define the theme to be used in the Styled System

- Custom UI
- Easy to develop
- High Quality of Code
- Expressive, consistent UI Components
- Multi-Theme

Theming

Theme Specification

Space	**margin, padding, grid-gap**	sizes	**Width, height**
fontSizes	**font-size**	borders	**border**
colors	**color, background-color, border-color**	borderWidths	**border-width**
fonts	**font-family**	borderStyles	**border-style**
fontWeights	**font-weight**	radii	**border-radius**
lineHeights	**line-height**	shadows	**box-shadow, text-shadow**
letterSpacings	**letter-spacing**	zIndices	**z-index**

Theme Specification

```
export default {
 breakpoints: ['768px', '1440px'],
 space: [0, 2, 4, 8, 16, 24, 32, 48],
 colors,
 fontSizes,
 fontWeights,
 lineHeights,
 letterSpacings,
 fonts,
 borderWidths,
 radii,
 Shadows,
}
```

Margin & Padding

Margin Props

`m`	margin
`mt`	margin-top
`mr`	margin-right
`mb`	margin-bottom
`ml`	margin-left
`mx`	margin-left and margin-right
`my`	margin-top and margin-bottom

Padding Props

`p`	padding
`pt`	padding-top
`pr`	padding-right
`pb`	padding-bottom
`pl`	padding-left
`px`	padding-left and padding-right
`py`	padding-top and padding-bottom

```
// theme.js
export default {
  space: [0, 4, 8, 16, 32, 64, 128, 256, 512],
}
```

```
// responsive margin, padding, fontSize
<Text m={[ 0, 1, 2 ]} p={[ 2, 3, 4 ]} />
```

Layout

The layout function adds props for widths, heights, display, and more. Widths and heights can use values defined in theme.sizes to help ensure consistency in layout styles.

```
<Box
  width={[
    1, // 100% below the smallest breakpoint (all viewports)
    1 / 2, // 50% from the next breakpoint and up
    1 / 4, // 25% from the next breakpoint and up
  ]}
/>
```

Layout

```
<Box width={[1, 1 / 2, 1 / 4]} />

.Box-hash {
  width: 100%;
}
@media screen and (min-width: 40em) {
  .Box-hash {
    width: 50%;
  }
}
@media screen and (min-width: 52em) {
  .Box-hash {
    width: 25%;
  }
}
```

Text

```
// example theme
export default {
  // base theme values...
  // custom button variants
  text: {
    h1: {
      fontSize: 'h1',
    },
    bodyText: {
      color: 'white',
      fontSize: 2,
    },
  }
}
```

```
const fontSizes = {
  h1: '2.4375rem',
  h2: '1.9375rem',
  paragraph: '1rem',
  sub: '0.8125rem',
  tiny: '0.625rem',
};


<Text variant=’h1’>Heading</Text>
```

Buttons

```
// example theme
export default {
  // base theme values...
  // custom button variants
  buttons: {
    primary: {
      color: 'white',
      bg: 'red',
    },
    secondary: {
      color: 'white',
      bg: 'tomato',
    },
  }
}
```

```
<Button variant='primary'>Primary</Button>

<Button variant='secondary'>Secondary</Button>
```

Forms

```
const inputStyle = {
  color: 'black',
  backgroundColor: 'white',
  height: 48,
};

const forms = {
  input: inputStyle,
  textarea: inputStyle,
  select: inputStyle,
  label: {
    color: 'black',
    fontSize: 'bodyText',
    mb: 4,
  },
};
```

```
// example theme
export default {
  // base theme values...
  // custom forms variants
  forms
}

<input type='text' />
```

Theme

```
export default {
 breakpoints: ['768px', '1440px'],
 space: [0, 2, 4, 8, 16, 24, 32, 48],
 colors,
 fontSizes,
 fontWeights,
 lineHeights,
 letterSpacings,
 fonts,
 borderWidths,
 radii,
 shadows,
 buttons,
 text,
 forms
}
```

“

Most CSS-in-JS libraries include a ThemeProvider to provide values through React context.

Import the ThemeProvider in the root of your application and pass the theme to the theme prop.

ThemeProvider

```
import React from 'react'
import StyledSystem from 'styled-system'
import { ThemeProvider } from 'styled-components'
import theme from './theme'

const App = props => (
  <ThemeProvider theme={theme}>
    {/* application elements */}
  </ThemeProvider>
)


export default App
```

Application Elements

@styled-system/space	space, margin, padding
@styled-system/color	color
@styled-system/layout	layout
@styled-system/typography	typography
@styled-system/flexbox	flexbox
@styled-system/border	border
@styled-system/background	background
@styled-system/position	position
@styled-system/grid	grid
@styled-system/shadow	shadow
@styled-system/variant	variant, textStyle, buttonStyle, colorStyle

<Text>

<Box>

<Flex>

<Grid>

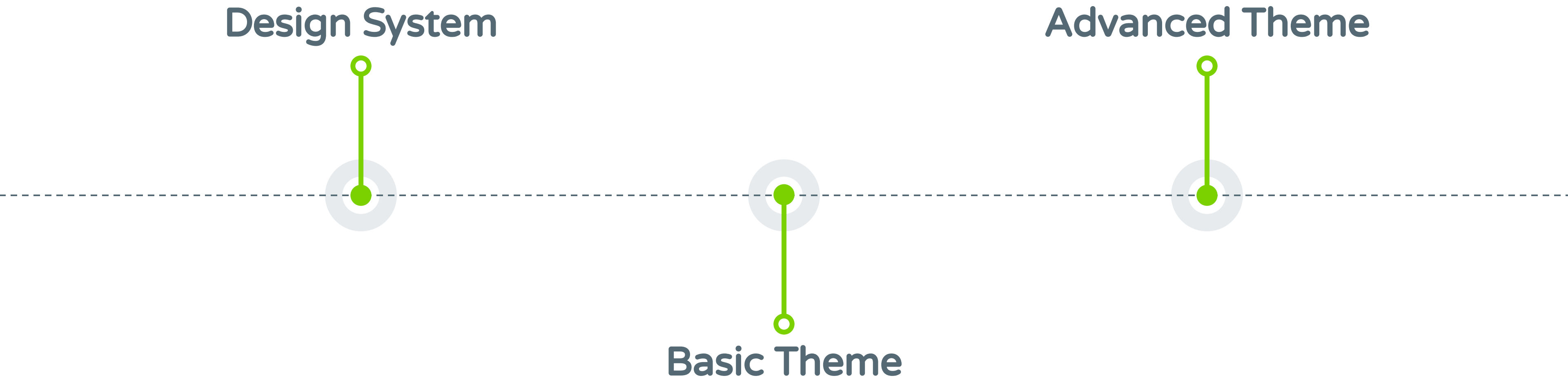
...

Custom Elements

```
// usage with the css prop
import React from 'react'
import css from '@styled-system/css'

const Beep = props =>
  <div
    {...props}
    css={css({
      fontSize: [4, 5, 6],
      color: 'primary',
    })}
  />
```

Process

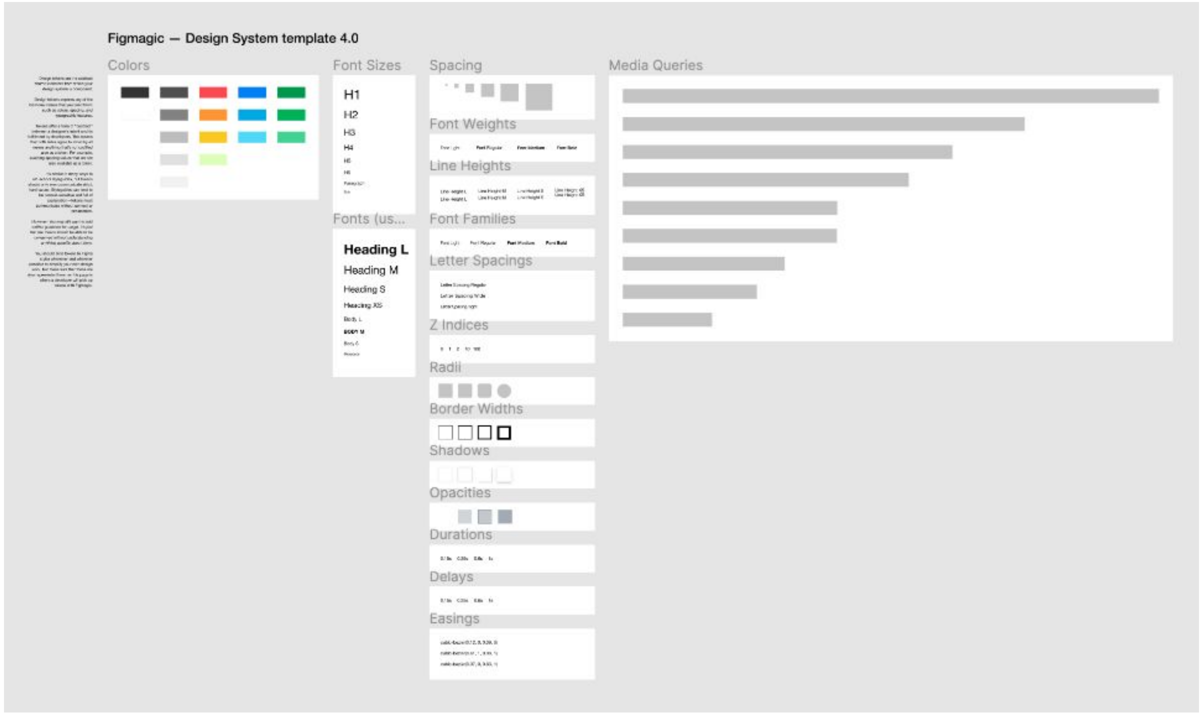


Design System => Basic Theme

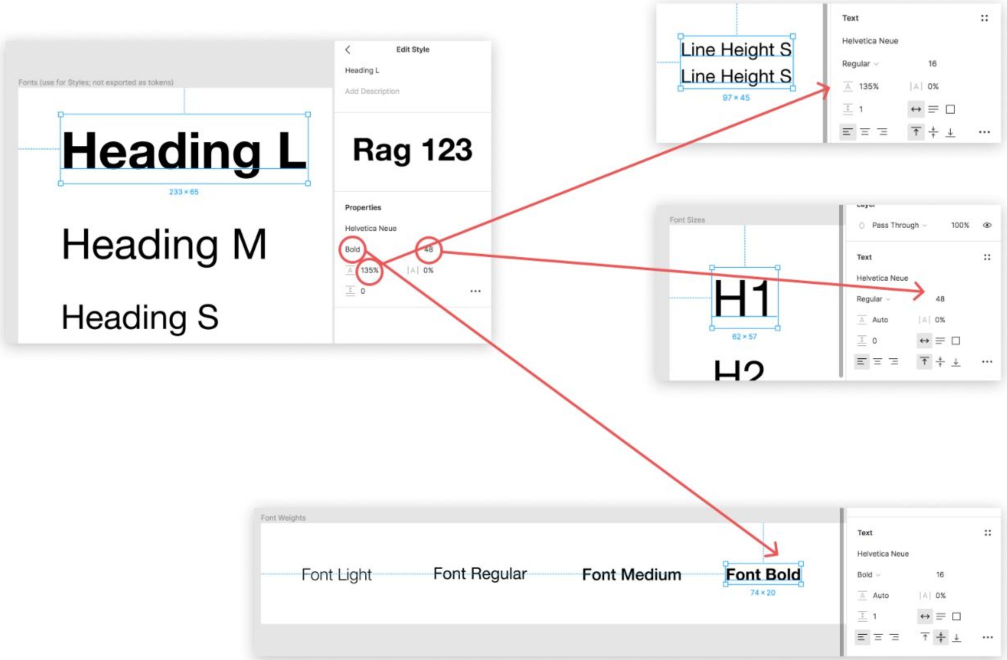
```
npm install -g figmagic

"scripts": {
      "figmagic": "node ./node_modules/figmagic/build/index.js"
}
```

Design System => Basic Theme



Design System => Basic Theme



Thanks!