Rendering images

Overview

Having a proper set of tools to render images within your application is quite important for many reasons, of which the most obvious are the fact that they can heavily affect the layout, as well as possible performance hit caused by somewhat careless handling of image assets.

UDS provides components to help you soften the impact of using images on both layout and performance levels.

Image components

Out of the box, there are three components that can be used to render images:

- `Image`
- `ResponsiveImage`
- `OptimizeImage`

Image

`Image` component is essentially a wrapper around HTML `img` tag. However, on top of the basic functionality, it also provides some styling features: you can render circular images, as well as images with rounded corners using the `rounded` prop that can take either `circle` or `corners` as the value if defined.

Note that you can also pass `'lazy'` via the `loading` prop to make the image lazy-loading.

```
<Image
  alt="Coworkers looking at phone"
  loading="lazy"
  rounded="circle"
  src={testImage}
  width={120}
  height={120}
/>
```

ResponsiveImage

`ResponsiveImage` component allows providing different image sources for different screen sizes based on brand-specific breakpoints.

```
<ResponsiveImage
  alt="mountain background"
  fallbackSrc={mountainsDesktopImage}
  loading="lazy"
  lgSrc={mountainsDesktopImage}
  mdSrc={mountainsTabletImage}
  smSrc={mountainsMobileImage}
  xsSrc={mountainsMobileImage}
/>
```

Note that it also accepts the `'lazy'` via the `loading` prop to allow lazy-loading images.

OptimizeImage

`OptimizeImage` component should be used to optimize Contentful image files based on screen size. Note that you have to provide a Contentful asset URL here for this component to work properly:

```
<OptimizeImage

contentfulAssetUrl="https://images.ctfassets.net/3cqlnin176yn/1GfIHDOb3n3uO3wVnKjHZ2/3a3c59c5caa91a315da1b3
804cdf0b1b/Alpaca-lets-make-the-future-friendly-Hero-Banner-Tile.jpg"
```

```
  alt="alpacas"
/>
```

Skip to main content

Other components rendering images

On top of components directly used to render images, there are several components that render images in context of other information.

Cards with full bleed images

Cards (`Card` , `PreviewCard` , `StoryCard`) offer an option to render full bleed images with flexible positioning options:

Full Bleed Image Example

Resize the width of the window, and the content will be positioned accordingly.

```
<Card
  fullBleedContent={{
    alt: 'Full Bleed Image Example',
    position: { md: 'left', sm: 'right', xs: 'none' },
    align: { xl: 'end', lg: 'center' },
    src: '/universal-design-system/img/pigs-allium-lo.png'
  }}
>
  <Box vertical={2}>
    <Typography block variant={{ size: 'h1' }}>
      Full Bleed Image Example
    </Typography>
  </Box>
  <Typography>
    Resize the width of the window, and the content will be positioned accordingly.
  </Typography>
</Card>
```

By default, if `fullBleedContent` prop has `src` property set, this functionality will be using `ResponsiveImage` component to render the image, however you can use by passing some JSX in a `content` property of the `fullBleedContent` :

Custom Full Bleed Content

Example content only.

Skip to main content

```
<Card
  fullBleedContent={{
    alt: 'Full Bleed Image Example',
    position: 'bottom',
    content: (
      <OptimizeImage

contentfulAssetUrl="https://images.ctfassets.net/3cqlnin176yn/1GfIHDOb3n3uO3wVnKjHZ2/3a3c59c5caa91a315da1b3
804cdf0b1b/Alpaca-lets-make-the-future-friendly-Hero-Banner-Tile.jpg"
        alt="alpacas"
      />
    )
  }}
>
  <Box vertical={2}>
    <Typography block variant={{ size: 'h1' }}>
      Custom Full Bleed Content
    </Typography>
  </Box>
  <Typography>Example content only.</Typography>
</Card>
```

This feature allows you to use optimized components like NextJS `Image` in combination with Cards in order to provide better user experience.

Testimonial

The purpose of the `Testimonial` component is displaying a testimonial in a standalone, pre-styled container. As an optional element, testimonial can have an image rendered next to the main text:

“Nulla sit amet congue lorem. Interdum et malesuada fames ac ante ipsum primis fame.”

John Doe
Pinnacle Reforestation

```
<Testimonial
  testimonial="Nulla sit amet congue lorem. Interdum et malesuada fames ac      primis fame."
  showDivider
  title="John Doe"
  image={testImage}
  additionalInfo="Pinnacle Reforestation"
/>
```

Skip to main content

Similarly to the full bleed content in cards, you can display image content via a custom image component. All you need in this case is to pass some JSX via the `image` prop:

“Nulla sit amet congue lorem. Interdum et malesuada fames ac ante ipsum primis fame.”

John Doe
Pinnacle Reforestation

```
<Testimonial
  testimonial="Nulla sit amet congue lorem. Interdum et malesuada fames ac ante ipsum primis fame."
  showDivider
  title="John Doe"
  image={
    <OptimizeImage

contentfulAssetUrl="https://images.ctfassets.net/3cqlnin176yn/1GfIHDOb3n3uO3wVnKjHZ2/3a3c59c5caa91a315da1b3
804cdf0b1b/Alpaca-lets-make-the-future-friendly-Hero-Banner-Tile.jpg"
      alt="alpacas"
      xs={120}
      sm={120}
      md={120}
      lg={120}
      xl={120}
    />
  }
  additionalInfo="Pinnacle Reforestation"
/>
```

WaffleGrid

`WaffleGrid` is used to show items in a waffle-like manner with borders surrounding the element:

TELUS	TELUS	TELUS	TELUS	TELUS	TELUS

```
<WaffleGrid
  items={[1, 2, 3, 4, 5, 6].map((_, i) => ({
    image:

'https://images.ctfassets.net/3cqlnin176yn/6fbpuI7xGEQwM0sueuEMmy/8768c289e22006227aa1ebf64bd05cf4/stevie.j
pg',
    imageAltText: 'The image alt text',
    href: `//telus.com?item=${i}`,
    text: 'TELUS'
  }))}
```

```
/>
```

Here as well, you can display image content via a custom image component by passing som [Skip to main content] e `image` prop:

[TELUS](#) [TELUS](#) [TELUS](#) [TELUS](#) [TELUS](#) [TELUS](#)

```
<WaffleGrid
  items={[1, 2, 3, 4, 5, 6].map((_, i) => ({
    image: (
      <OptimizeImage

contentfulAssetUrl="https://images.ctfassets.net/3cqlnin176yn/1GfIHDOb3n3uO3wVnKjHZ2/3a3c59c5caa91a315da1b3
804cdf0b1b/Alpaca-lets-make-the-future-friendly-Hero-Banner-Tile.jpg"
        alt="alpacas"
      />
    ),
    imageAltText: 'The image alt text',
    href: `//telus.com?item=${i}`,
    text: 'TELUS'
  }))}
/>
```

Conclusion

Hopefully, this guide provides enough information for you to choose an appropriate way of rendering images. If you believe that something is missing here or needs a clarification, feel free to open a discussion in the [#ds-support](#) Slack channel.

[Edit this page](#)