Image Carousel

Frame 2

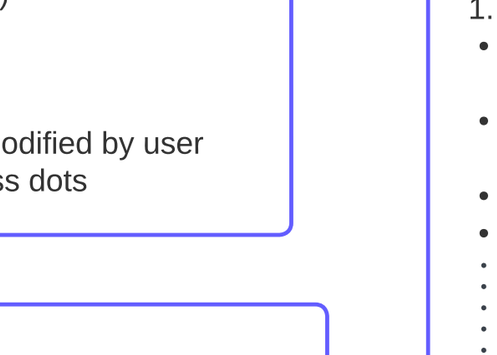
Requirements

1. List of images should be able be to specified.
2. Cycle through the images infinitely

Non Functional
- Responsive for all devices
- Animation effect while transitioning from images

Frame 3

Image carousel



Frame 1

Component responsibilities

1. ViewModel + Model : Contains configuration state of the component. orchestrates the events between the components. Informs the image component which image to render.
2. Image: Displays the currently selected image
3. Prev/next buttons: Tell the view model to change the image, based on the button clicked
4. Progress dots: Tells the view model, which image to show when the respective dot/page is being clicked or selected

Frame 10

Data Model

Only the view model will hold any state and data, the other components are part of the view and won't hold any data.

Fields:

1. configuration
   - List of images(image url and alt value)
   - transition duration
   - size (height and width of the image)
2. State
   - Index of the current image - can be modified by user interaction with the buttons or progress dots

Frame 6

Interface Definition

1. Basic API
   - List of images: An array of image urls with any associated meta data
   - Transition duration (ms): Duration for the transition animation during image transitions
   - Height (px)
   - Width (px)

```
<ImageCarousel
  images={[
     { src: 'https://example.com/images/foo.jpg', alt: 'A foo' },
     { src: 'https://example.com/images/bar.jpg', alt: 'A bar' },
     /* More images if desired. */
   ]}
  transitionDuration={300}
  height={500}
  width={800}
/>
```

2. Advanced API
   - Autoplay: Whether the carousel willautomatically progress to the next image.
   - A timer state value will be needed to keep incrementing the image
   - The timer should be cancelled / reset if the current image was manually changed by the user.
   - Delay between transitions, only needed if the carousel is in autoplay mode
   - Event listeners (onload -when the first image is done loading on the carousal, onPageselect -when a page is selected, onNextClick -when the next button is triggered, onPrevClick -when the prev image button is triggered
   - Theming - style options
   - Loop Enable looping behaviour where click next on the last image will return to first image, similarly with clicking prev in the first image to show last image.

Frame 7

OPtimizations

1. A simple version to achieve the layout is to use `display: flex` to make the images render in a horizontal row and programatically change the horizontal scroll offset to show the various images.

```
 div class="carousel-images">
  <img class="carousel-image" alt="..." src="..." />
  <img class="carousel-image" alt="..." src="..." />
  <img class="carousel-image" alt="..." src="..." />
  <!-- More images -->
</div>
.carousel-images {
  display: flex;
  height: 400px;
  width: 600px;
  overflow: hidden;
}

.carousel-image {
  height: 400px;
  width: 600px;
}

document.querySelector('.carousel-images').scrollTo({
  left: selectedIndex * 600,
  behavior: 'smooth',
});

2. CSS Background-size: Contain and cover, but requires us to use
background-image instead of img tags
3. CSS object fit - Same as above but with the img tags
4. Vertically - center buttons
<div class="carousel-image-container">
  <div class="carousel-images">
    <img class="carousel-image" alt="..." src="..." />
    <img class="carousel-image" alt="..." src="..." />
    <!-- More images -->
  </div>
  <button class="carousel-button carousel-button-prev">...</button>
  <button class="carousel-button carousel-button-next">...</button>
</div>
.carousel-image-container {
  height: 400px;
  width: 600px;
  position: relative; /* So that position: absolute will be relative to
this element. */
}

.carousel-button {
  position: absolute;
  top: 50%;
  transform: translateY(-50%); /* Shifts the button up by half its
height. */
}

.carousel-button-prev {
  left: 30px;
}

.carousel-button-next {
  right: 30px;
}

User Experience:

1. Scroll snapping: If halfway scrolled, snap smoothly to show full
   images
2. Interactive elements should be visible and clickable
3. having prev and next button nearby, speeds up the navigation
4. Defer loading of the images that aren't on screen
5. preemptively load the next image
const preloadImage = new Image();
preloadImage.src = url;
```

1. Initially, only the first image is loaded (the remaining images will be lazily loaded).
2. The second image is preloaded when the user shows possible intent of viewing more images:
   - Cursor hovers over the image carousel.
   - Focuses on the "Next" button via tabbing.
   - Image carousel comes into view (on mobile devices).
3. If the user does view the second image (which signals high intent to browse even more images), the next three images (3rd to 5th) are preloaded.
4. As the user clicks "Next" again to browse more images, the (n + 3)th image is preloaded.

```
Device specific images - The quality according to the device
Virtualized lists - Only visual images in the DOM
Accessibility of the progressive dots is extremely notorious
Have meaningful aria labels for the screen readers
Keyboard navigation for prev and next, the buttons should be focussable
```