File: client/src/components/ui/carousel.tsx

Type: tsx | Size: 6210 bytes

```tsx
import * as React from "react"
import useEmblaCarousel, {
  type UseEmblaCarouselType,
} from "embla-carousel-react"
import { ArrowLeft, ArrowRight } from "lucide-react"

import { cn } from "@/lib/utils"
import { Button } from "@/components/ui/button"

type CarouselApi = UseEmblaCarouselType[1]
type UseCarouselParameters = Parameters<typeof useEmblaCarousel>
type CarouselOptions = UseCarouselParameters[0]
type CarouselPlugin = UseCarouselParameters[1]

type CarouselProps = {
  opts?: CarouselOptions
  plugins?: CarouselPlugin
  orientation?: "horizontal" | "vertical"
  setApi?: (api: CarouselApi) => void
}

type CarouselContextProps = {
  carouselRef: ReturnType<typeof useEmblaCarousel>[0]
  api: ReturnType<typeof useEmblaCarousel>[1]
  scrollPrev: () => void
  scrollNext: () => void
  canScrollPrev: boolean
  canScrollNext: boolean
} & CarouselProps

const CarouselContext = React.createContext<CarouselContextProps | null>(null)

function useCarousel() {
  const context = React.useContext(CarouselContext)

  if (!context) {
    throw new Error("useCarousel must be used within a <Carousel />")
  }

  return context
}

const Carousel = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement> & CarouselProps
>(
  (
    {
      orientation = "horizontal",
      opts,
      setApi,
      plugins,
      className,
      children,
      ...props
    },
    ref
  ) => {
    const [carouselRef, api] = useEmblaCarousel(
      {
        ...opts,
        axis: orientation === "horizontal" ? "x" : "y",
      },
      plugins
    )
    const [canScrollPrev, setCanScrollPrev] = React.useState(false)
    const [canScrollNext, setCanScrollNext] = React.useState(false)

    const onSelect = React.useCallback((api: CarouselApi) => {
      if (!api) {
```

```
    return
  }

  setCanScrollPrev(api.canScrollPrev())
  setCanScrollNext(api.canScrollNext())
}, [])

const scrollPrev = React.useCallback(() => {
  api?.scrollPrev()
}, [api])

const scrollNext = React.useCallback(() => {
  api?.scrollNext()
}, [api])

const handleKeyDown = React.useCallback(
  (event: React.KeyboardEvent<HTMLDivElement>) => {
    if (event.key === "ArrowLeft") {
      event.preventDefault()
      scrollPrev()
    } else if (event.key === "ArrowRight") {
      event.preventDefault()
      scrollNext()
    }
  },
  [scrollPrev, scrollNext]
)

React.useEffect(() => {
  if (!api || !setApi) {
    return
  }

  setApi(api)
}, [api, setApi])

React.useEffect(() => {
  if (!api) {
    return
  }

  onSelect(api)
  api.on("reInit", onSelect)
  api.on("select", onSelect)

  return () => {
    api?.off("select", onSelect)
  }
}, [api, onSelect])

return (
  <CarouselContext.Provider
    value={{
      carouselRef,
      api: api,
      opts,
      orientation:
        orientation || (opts?.axis === "y" ? "vertical" : "horizontal"),
      scrollPrev,
      scrollNext,
      canScrollPrev,
      canScrollNext,
    }}
  >
    <div
      ref={ref}
      onKeyDownCapture={handleKeyDown}
      className={cn("relative", className)}
      role="region"
      aria-roledescription="carousel"
      {...props}
    >
      {children}
```

```
        </div>
      </CarouselContext.Provider>
    )
  }
)
Carousel.displayName = "Carousel"

const CarouselContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { carouselRef, orientation } = useCarousel()

  return (
    <div ref={carouselRef} className="overflow-hidden">
      <div
        ref={ref}
        className={cn(
          "flex",
          orientation === "horizontal" ? "-ml-4" : "-mt-4 flex-col",
          className
        )}
        {...props}
      />
    </div>
  )
})
CarouselContent.displayName = "CarouselContent"

const CarouselItem = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { orientation } = useCarousel()

  return (
    <div
      ref={ref}
      role="group"
      aria-roledescription="slide"
      className={cn(
        "min-w-0 shrink-0 grow-0 basis-full",
        orientation === "horizontal" ? "pl-4" : "pt-4",
        className
      )}
      {...props}
    />
  )
})
CarouselItem.displayName = "CarouselItem"

const CarouselPrevious = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<typeof Button>
>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
  const { orientation, scrollPrev, canScrollPrev } = useCarousel()

  return (
    <Button
      ref={ref}
      variant={variant}
      size={size}
      className={cn(
        "absolute  h-8 w-8 rounded-full",
        orientation === "horizontal"
          ? "-left-12 top-1/2 -translate-y-1/2"
          : "-top-12 left-1/2 -translate-x-1/2 rotate-90",
        className
      )}
      disabled={!canScrollPrev}
      onClick={scrollPrev}
      {...props}
    >
```

```
      <ArrowLeft className="h-4 w-4" />
      <span className="sr-only">Previous slide</span>
    </Button>
  )
})
CarouselPrevious.displayName = "CarouselPrevious"

const CarouselNext = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<typeof Button>
>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
  const { orientation, scrollNext, canScrollNext } = useCarousel()

  return (
    <Button
      ref={ref}
      variant={variant}
      size={size}
      className={cn(
        "absolute h-8 w-8 rounded-full",
        orientation === "horizontal"
          ? "-right-12 top-1/2 -translate-y-1/2"
          : "-bottom-12 left-1/2 -translate-x-1/2 rotate-90",
        className
      )}
      disabled={!canScrollNext}
      onClick={scrollNext}
      {...props}
    >
      <ArrowRight className="h-4 w-4" />
      <span className="sr-only">Next slide</span>
    </Button>
  )
})
CarouselNext.displayName = "CarouselNext"

export {
  type CarouselApi,
  Carousel,
  CarouselContent,
  CarouselItem,
  CarouselPrevious,
  CarouselNext,
}
```