Creating Navigation Bars with shiny.router

To show how `shiny.router` works in practice, we'll develop a simple dashboard with a couple of routes
showing us which route we're on.

To start, we'll import both `shiny` and `shiny.router`:

```
library(shiny)
library(shiny.router)
```

Next, we will store content for three pages in three variables. Every page has a `shiny.titlePanel` an

```
home_page <- div(
  titlePanel("Dashboard"),
  p("This is a dashboard page")
)

settings_page <- div(
  titlePanel("Settings"),
  p("This is a settings page")
)

contact_page <- div(
  titlePanel("Contact"),
  p("This is a contact page")
)
```

We can then make a router and attach each of the pages with its corresponding route. The dashboard is
the first one you see:

```
router <- make_router(
  route("/", home_page),
  route("settings", settings_page),
  route("contact", contact_page)
)
```

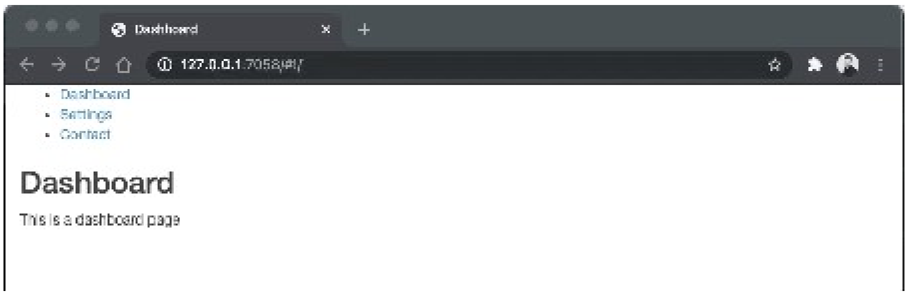
The rest of the Shiny app is more or less what you would expect. We have to declare the `UI`, which conta
navigate between pages. The `server` function passes input, output, and session data to the router. Fina
two components together.

```
ui <- fluidPage(
  tags$ul(
    tags$li(a(href = route_link("/"), "Dashboard")),
    tags$li(a(href = route_link("settings"), "Settings")),
    tags$li(a(href = route_link("contact"), "Contact"))
  ),
  router$ui
)

server <- function(input, output, session) {
  router$server(input, output, session)
}
```

```
shinyApp(ui, server)
```

As a result, we have the following web application:


Dashboard

127.0.0.1:7058/#!/

- Dashboard
- Settings
- Contact

Dashboard

This is a dashboard page

Unstyled Shiny Router App

The application gets the job done but is quite basic with regards to the styling. Let's fix that next.

Styling Navigation Bars

You can add styles to your Shiny applications with CSS. To do so, create a `www` folder where your R scrip
We've named ours `main.css`, but you can call yours whatever you want.

To link the created CSS file with the Shiny app, we have to add a `theme` to `shiny.fluidPage`. Here's h

```
ui <- fluidPage(
  theme = "main.css",
  tags$ul(
    tags$li(a(href = route_link("/"), "Dashboard")),
    tags$li(a(href = route_link("settings"), "Settings")),
    tags$li(a(href = route_link("contact"), "Contact"))
  ),
  router$ui
)
```

The value for the `theme` parameter must be identical to the name of the CSS file.

If you were to run the app now, everything would look the same as before. That's because we haven't ad
code snippet to your CSS file:

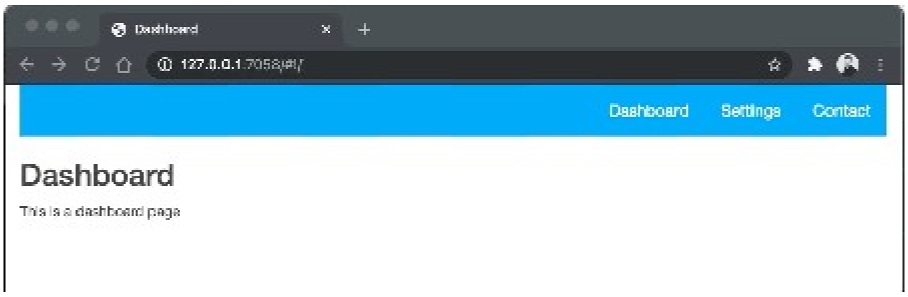
```
ul {
  background-color: #0099f9;
    display: flex;
  justify-content: flex-end;
  list-style-type: none;
}

ul li a {
  color: #ffffff;
    display: block;
  font-size: 1.6rem;
  padding: 1.5rem 1.6rem;
  text-decoration: none;
  transition: all, 0.1s;
}
```

```
a:link, a:visited, a:hover, a:active {
  color: #ffffff;
    text-decoration: none;
}

ul li a:hover {
  background-color: #1589d1;
    color: #ffffff;
}
```

Save and rerun the application. You will see the following:



Dashboard

This is a dashboard page

Styled Shiny Router App

And that’s how you can style `shiny.router` and Shiny apps in general.