How to Create a Beautiful, Dynamic Shiny App

Yonas Kbrom

12/3/2017

Introduction

As an experienced programmer, once I learned about shiny apps, I was always curious of how exactly the app was rendered, or more specifically, how the visuals were processed so that we can actually see them. As I researched a little more into it, I later found out that every function in the UI portion of the app is basically broken down into HTML and CSS elements, those of which can be used within the UI. With this understanding in mind, it was much easier for me to understand how shiny apps are created and the more I read into shiny apps, the more interested I became in learning more about shiny apps and their capabilities. It even came to the point where I truly enjoyed the topic and has caused me to introduce you all to the great and wonderful things I have learned about shiny apps! Throughout this blog post, I will go through how to create a beautiful, functional shiny app using mostly HTML and CSS elements and outside packages.

Motivation

After working with shiny apps through out this course, I felt as if we were just scratching the surface of what could possibly be done with a shiny app. Learning how to create a shiny app was fun and exciting, but I always had this urge to go beyond the basics of what we learned about shiny apps in the class and create a beautiful shiny app that actually looked great. So I explored a bit into what exactly I wanted to include in my shiny app and came across many great resources that would amplify my current knowledge of shiny apps.

Set-up

Before we jump into actually creating a shiny app, we must first make sure we have everything required to start. First install the following packages:

```
install.packages("data.table")
install.packages("DT")
```

Once your have installed the two packages, create an shiny web application file with any title and include the following lines at the top:

```
library(DT)
library(data.table)
mydat <- fread("https://github.com/ucb-stat133/stat133-fall-2017/raw/master/data/nba2017-players.csv")
```

We will be using the statitics of NBA players as our data. We are now ready to start creating our app!

Creating the App

Navigation Bar

If your would like you could clear the entire file except for the lines I included above but not necessary. The first part of a shiny app that is essential is the UI. In our course, the main layout we used where sidebar layouts, but instead I am going to introduce the navigation bar. To include a navbar, put the following lines as your starting UI:

```
ui <- shinyUI(fluidPage(navbarPage("NBA",
                           tabPanel("About"),
                           tabPanel("Players"),
                           tabPanel("Teams"),
                           navbarMenu("More",
                                      tabPanel("References"),
                                      tabPanel("Github"),
                                      tabPanel("Facebook"),
                                      tabPanel("Email"))
                                    )
                         )
              )
```

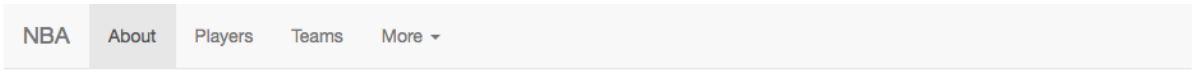
Along with the UI, the server is also required, in order to take inputs and use them to display different data. For now for the server, you can leave it empty like this:

```
# Define server logic required to draw a histogram
server <- function(input, output) {

}

# Run the application
shinyApp(ui = ui, server = server)
```

Once you have done all this, you can now run it! Once it is run it should like something like this:



Images

The next thing we would like to do is include an image within your “About” page. In order to do this, within the UI portion of the app, change the “About” tab to make it look like this.

```
tabPanel("About",
         img(style="display: block; margin-left: auto; margin-right: auto;",
         src="http://www.pngmart.com/files/5/NBA-Transparent-Background.png")
```

The “img” method correlates directly to the HTML tag and takes in arguments as does the tag. For those who are not familiar with HTML, the style argument allows you to change the image in any way you would like: make it bigger, small, left-aligned, right-aligned, centered. In this case, the style is made so that the image is centered by auto aligned the left and right margins. The “src” serves as the source of your image, or the location of it. For this, I chose an image from the web. Once your have made this change, your app should now look like this:

DT Tables

The next thing we will to this shiny app are data frames. The “DT” library allows your to efficiently include data frames into your UI in a nice way. Along with the table, the DT package includes a mini search bar to filter your table with characters that match that in the search bar and also includes an input so your can choose how many entires are shown on one page. This is the point where we will actually get to use our table “mydat” that we imported in the set-up. First we should add 2 lines to our server that allow us to actually access our table in the UI. Make sure your server looks like this:

```
server <- function(input, output) {
   output$players = DT::renderDataTable({DT::datatable(mydat)})
   output$teams = DT::renderDataTable({DT::datatable(unique(mydat[,2]))})
}
```

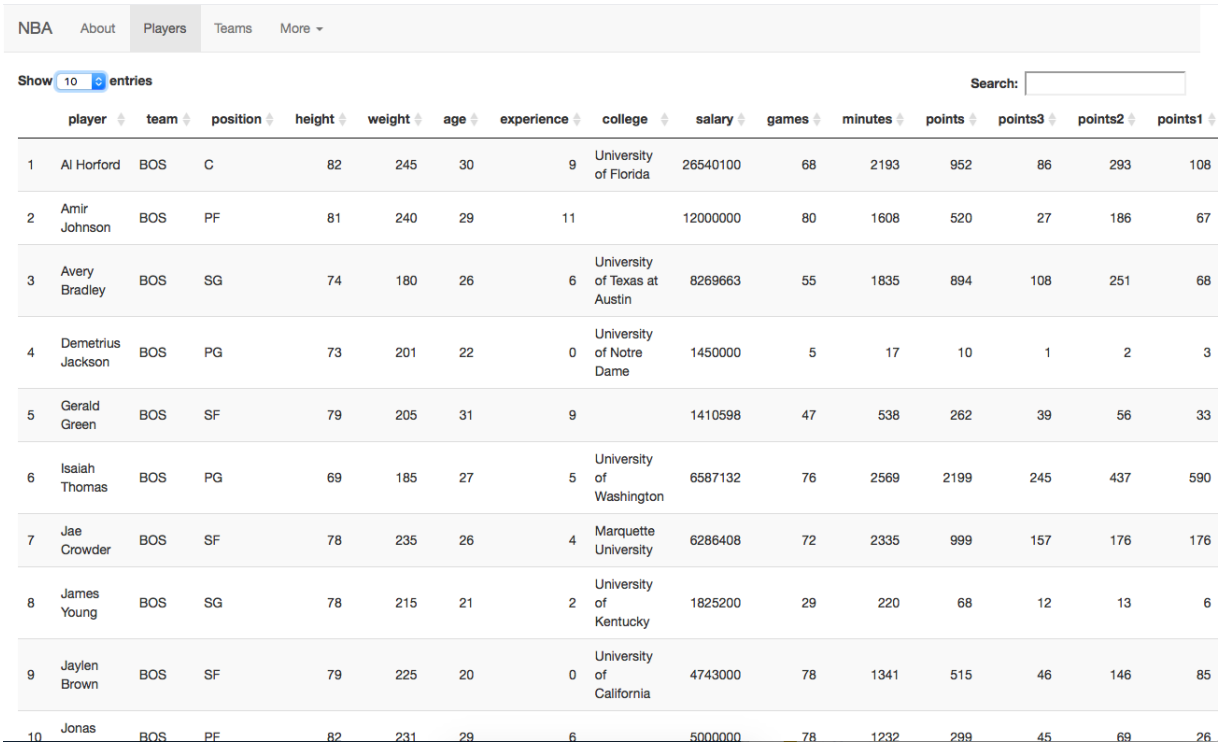
The first line stores the mydat table into the output variable “players”, so now in order to reference mydat in our UI, we must use “players”. Same goes for the teams variable except it is a list of teams that exist in the NBA. Once these lines are included, we can now move onto our UI to incorporate our tables. The first thing we will change is our “Players” tab and make it so that it looks somewhat like this:

```
tabPanel("Players",
        DT::dataTableOutput("players")),
```

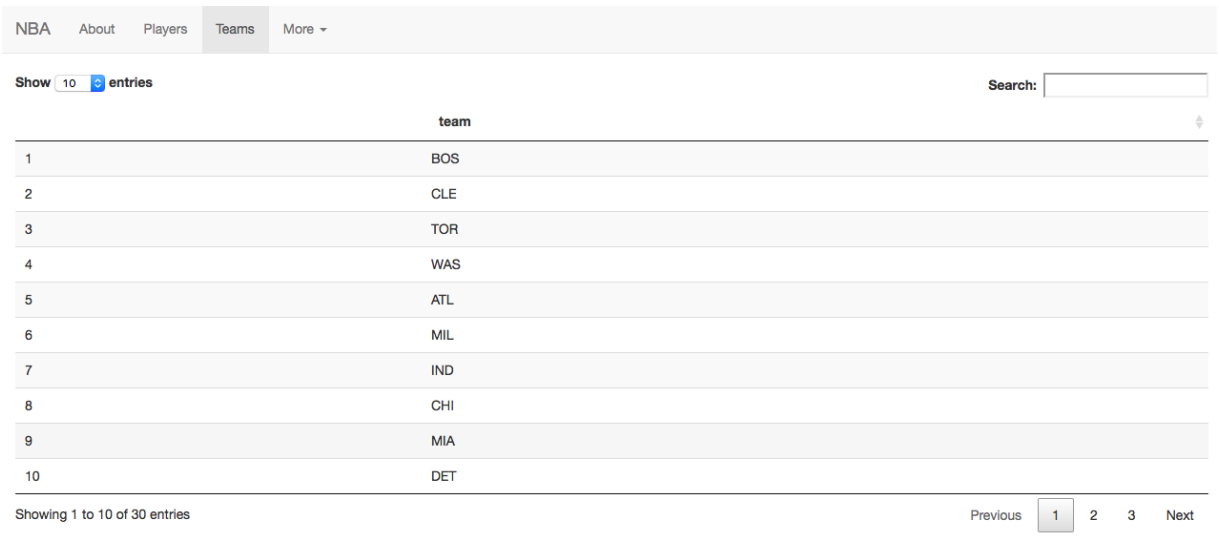
We will also do the same for our teams tab:

```
tabPanel("Teams",
        DT::dataTableOutput("teams")),
```

Once these are changed, you can now see these changes in the UI! Restart the app and click on each of the tabs, they should look like the following respectively:



	player	team	position	height	weight	age	experience	college	salary	games	minutes	points	points3	points2	points1
1	Al Horford	BOS	C	82	245	30	9	University of Florida	26540100	68	2193	952	86	293	108
2	Amir Johnson	BOS	PF	81	240	29	11		12000000	80	1608	520	27	186	67
3	Avery Bradley	BOS	SG	74	180	26	6	University of Texas at Austin	8269663	55	1835	894	108	251	68
4	Demetrius Jackson	BOS	PG	73	201	22	0	University of Notre Dame	1450000	5	17	10	1	2	3
5	Gerald Green	BOS	SF	79	205	31	9		1410598	47	538	262	39	56	33
6	Isaiah Thomas	BOS	PG	69	185	27	5	University of Washington	6587132	76	2569	2199	245	437	590
7	Jae Crowder	BOS	SF	78	235	26	4	Marquette University	6286408	72	2335	999	157	176	176
8	James Young	BOS	SG	78	215	21	2	University of Kentucky	1825200	29	220	68	12	13	6
9	Jaylen Brown	BOS	SF	79	225	20	0	University of California	4743000	78	1341	515	46	146	85
10	Jonas	BOS	PF	82	231	29	6		5000000	78	1232	299	45	69	26



	team
1	BOS
2	CLE
3	TOR
4	WAS
5	ATL
6	MIL
7	IND
8	CHI
9	MIA
10	DET

Showing 1 to 10 of 30 entries

Previous 1 2 3 Next

Menu items

Now for the last tab we will work with the menu items. We will not be adding much depth to these tabs, but rather an introduction to the idea of the menu of the navbar. When you click on any of the items, such as “References” or “Github”, it just shows a blank page. Lets change that now! For the first one, we will just include a bulleted list of a couple of references I have used with this assignment. Add the following to your “References” tab:

```
tabPanel("References",
        div(style="text-align: center;", "References"),
        br(),br(),br(),br(),
        tags$ul(
            tags$li("https://shiny.rstudio.com/articles/layout-guide.html"),
tags$li("https://stat.ethz.ch/R-manual/R-devel/library/base/html/unique.html"),
            tags$li("http://rstudio.github.io/shiny/tutorial/#run-and-debug"),
            tags$li("https://shiny.rstudio.com/articles/css.html"),
            tags$li("https://developer.mozilla.org/en-US/docs/Web/CSS/font-family")
        )),
```

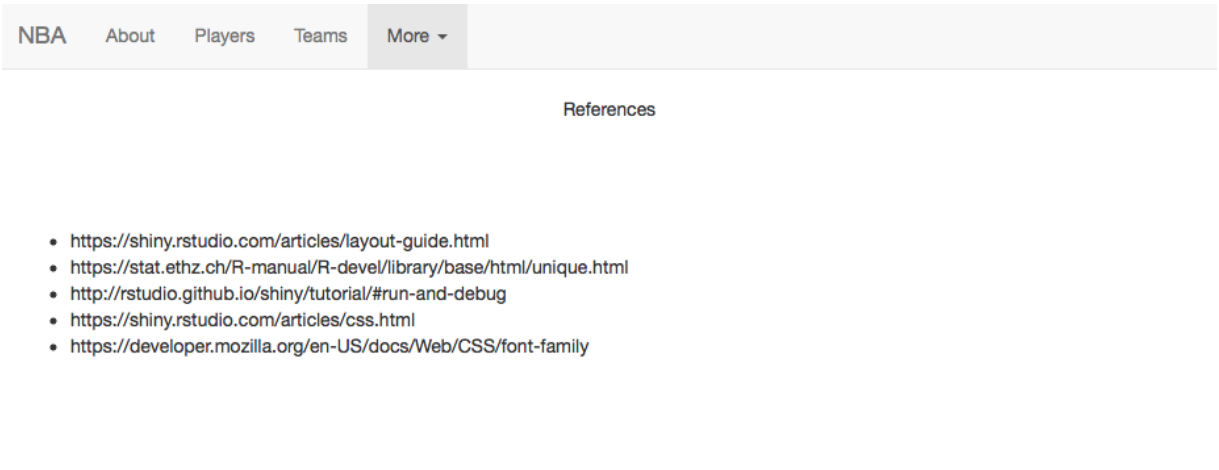
The “tags” is a reference to HTML tags. The “ul” element stands for “unordered list” otherwise known as bullet points and the “li” tags stands for “list item” and only exists within either an “ul” or “ol” (ordered list) tag. While we are at it, might as well change the other items. Make it so the other tabs look something like this:

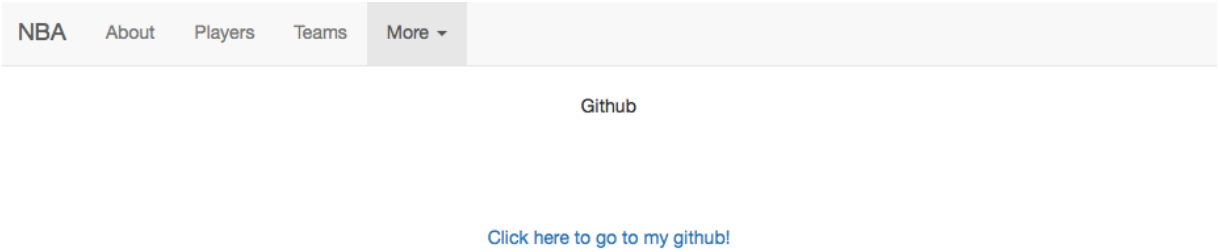
```
tabPanel("Github",
        div(style="text-align: center;", "Github"),
        br(),br(),br(),br(),
        div(style="text-align: center;", tags$a(href="https://github.com/yonas024", "Click here to go to my github
!")))
```

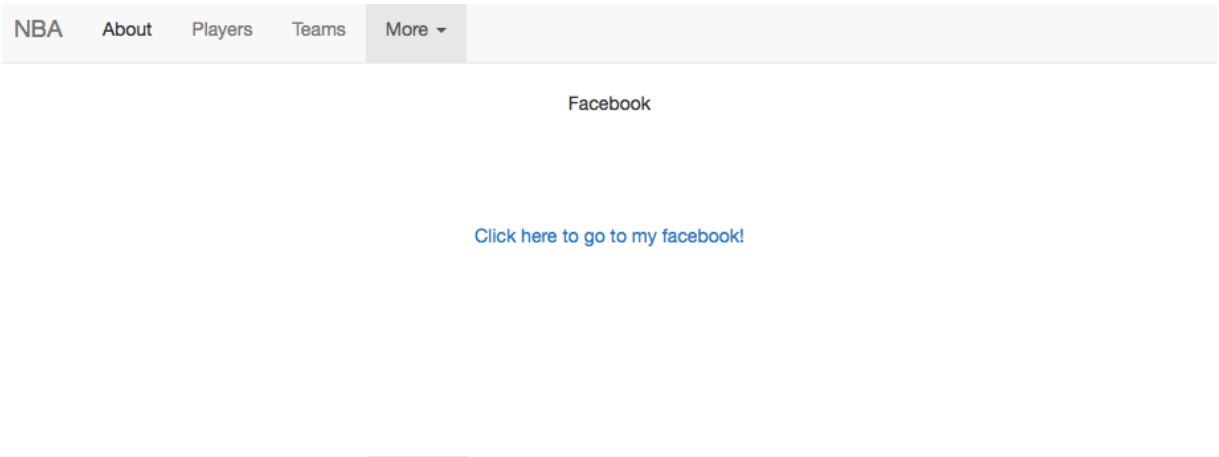
```
tabPanel("Facebook",
        div(style="text-align: center;", "Facebook"),
        br(),br(),br(),br(),
        div(style="text-align: center;", tags$a(href="https://www.facebook.com/yotukb", "Click here to go to my fa
cebook!")))
```

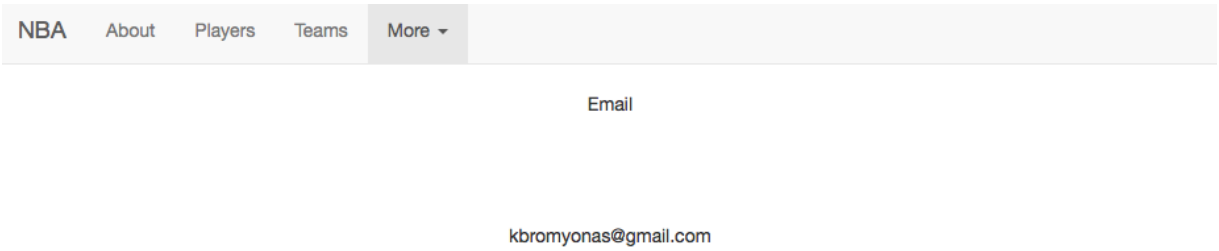
```
tabPanel("Email",
        div(style="text-align: center;", "Email"),
        br(),br(),br(),br(),
        div(style="text-align: center;", "kbromyonas@gmail.com"))
```

The tags with “a” and “href” are links where if you click on it, it will take you to that link. The following is the name of that link element, that shows up on the UI. Once you have changed these elements, your app should look like the following respectively:









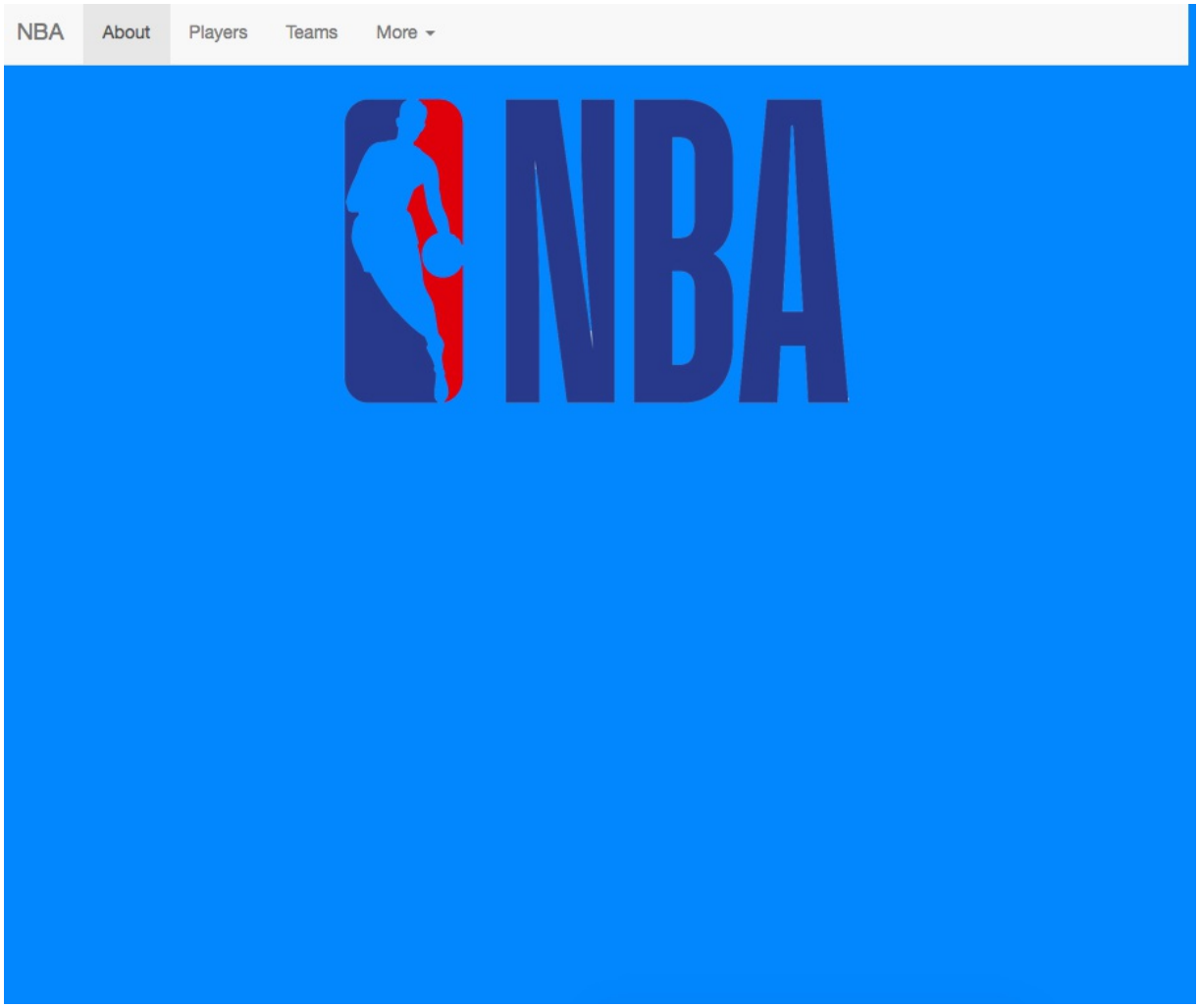
CSS Styling

CSS is a really handy tool in order to make your UI pop and look better that plain black and white. CSS stands for Cascading Style Sheets which is simply a way to style your HTML elements. We have already been using a bit of CSS by how we center our text and images to the center of the page, now we will try to change the shape, color, and fonts of our images. First of lets change the background color of our app, it seems a bit to white for me. In order to do this, we must add tags that allow us the change our element styles. Within the fluidPage, include the following after the closing parenthesis of the navbarPage (make sure you add a comma after the closing parenthesis of the navbarPage):

```
tags$head(
    tags$style(HTML("
            body {
              background-color: #1E90FF
            }

          "))
)
```

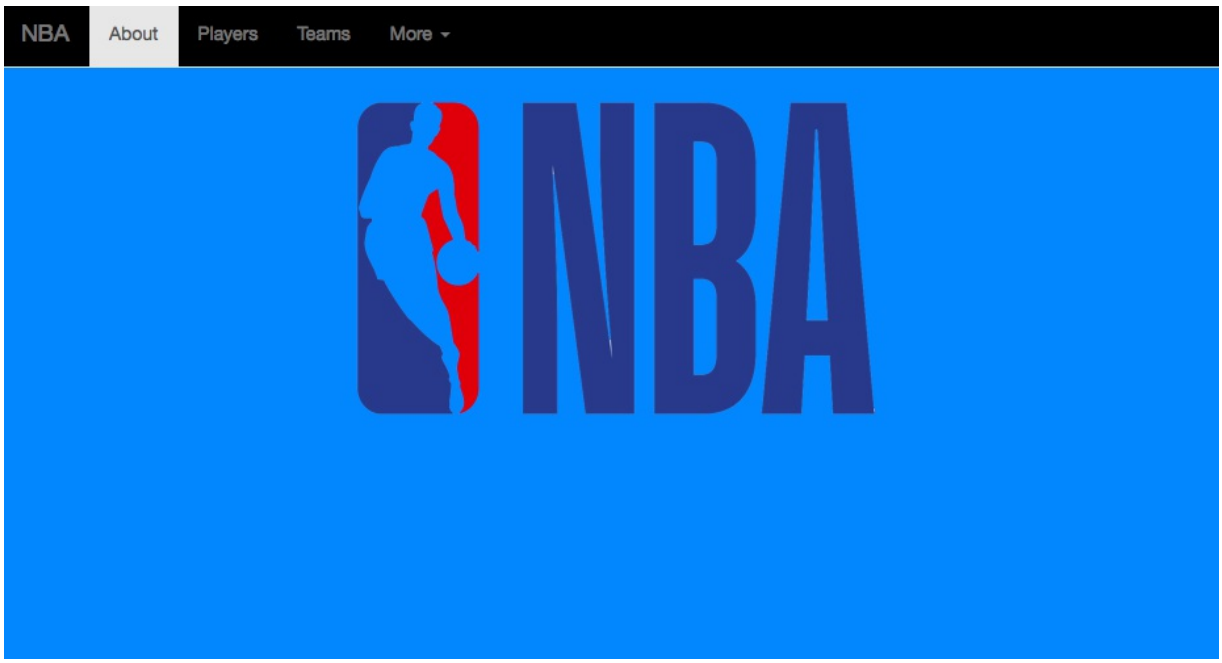
This code translates to CSS syntax that basically says change the background color of every “body” tage to “#1E90FF” which is hex (find more here https://www.w3schools.com/colors/colors_hex.asp) for a light blue color. This will change the background color of every page on the app. After adding this your “About” page should look somewhat like this:



You can also add the follow to the previous example to change the color of the navbar at the top, in my example, I have changed it to black:

```
tags$head(
    tags$style(HTML("
            body {
              background-color: #1E90FF
            }

            .navbar {
              background-color: black
            }
          "))
)
```



Lastly, you can change the text of the titles within the menu item pages. First we must add the following ".titles" to the previous CSS segment:

```
tags$head(
    tags$style(HTML("

            .titles {
              font-family: cursive;
              font-size: 50px;
              line-height: 1.1;
              color: black;
            }

            body {
              background-color: #1E90FF
            }

            .navbar {
              background-color: black
            }
          "))
)
```

Once this is added, add 'class = "titles"' to each of the div tags that include a title to change the text of it. Once this added, the titles should look somewhat like the following:

Take Home Message

You just created a beautiful shiny app with HTML and CSS! You were able to incorporate a navigation bar, images, tables, and style all of these in a fashionable manner. As one can see, shiny apps are incredible tools that can be using to display a variety of things, and although it is not the best way of doing so, you can even create your own website! There are much more about shiny apps that can be learned through the numerous online resources. All in all, going out of your way to learn more about shiny apps is an investment that would be well worth your time, and I will continue to do so as you should too!

Help

If you are struggling to make the app work properly, I have put up my working version at https://github.com/yonas024/stat133images/blob/master/ShinyApp/app.R for assistance.

References

- https://shiny.rstudio.com/articles/layout-guide.html
- https://stat.ethz.ch/R-manual/R-devel/library/base/html/unique.html
- http://rstudio.github.io/shiny/tutorial/#run-and-debug
- https://shiny.rstudio.com/articles/css.html
- https://developer.mozilla.org/en-US/docs/Web/CSS/font-family
- https://www.w3schools.com/colors/colors_hex.asp
- https://shiny.rstudio.com/articles/tag-glossary.html
- https://www.wikihow.com/Set-Background-Color-in-HTML
- http://rstudio.github.io/shiny/tutorial/#ui-and-server
- https://developer.mozilla.org/en-US/docs/Web/CSS/font-family