Developing a Shiny WebApp User Interface

Fu-Sheng Chou, MD, PhD

Last month, we introduced different basic layouts available for choosing when designing a Shiny WebApp. The plain layout is easy to start. It is useful when you are developing simple WebApps for routine tasks. I used the *plain* layout to develop the NAS scoring WebApp (https://lluchnicu.org/NICU_Manual/NAS_scoring/). Because of its simplicity, the plain layout also works very well on mobile browsers. When the *sidebar* layout is used, it is typical to have all the input widgets on the sidebar panel and the output for display on the main panel, although it is just a general guide. On the other hand, the dashboard layout has more complete functions and appears more like typical websites we see and use routinely. It has a sidebar panel that hides when not in use. It also allows the construction of a header toolbar for notification, message, and task menus. In the main panel, the *dashboard* layout allows the use of “boxes” to group information for display. There are basic boxes, tabbed boxes, and infoboxes. (1) The *dashboard* layout uses AdminLTE (2), a fully responsive administrative template based on the Bootstrap 3 framework(3). All this means is that, for professional web developers, the potential is limitless.

“There are basic boxes, tabbed boxes, and infoboxes. (1) The dashboard layout uses AdminLTE (2), a fully responsive administrative template based on the Bootstrap 3 framework(3). All this means is that, for professional web developers, the potential is limitless.”

1. HTML Tags

HTML, or HyperText Markup Language, is the standard programming language for webpages. It is the most basic building block of the websites. Here, we will discuss how the *Shiny* package turns R codes into HTML. We will use the NAS scoring WebApp as an example. If you click on the above link to go to the NAS scoring WebApp, and open up the code webpage (<https://neonatologytoday.org/datascience/NAS.html>) side by side to compare, we can see that the first argument in the `ui <- fluidPage()` is `title="LLUCH NICU NAS Scoring WebApp"`. This argument will place the strings in the tab title of the webpage (Figure 1). The second argument is a sub-function `titlePanel("Modified Finnegan Scoring System")`. Now, if you type down the same codes in the console:

```
> titlePanel("Modified Finnegan Scoring System")
<h2>test</h2>
```

You see that the return is an HTML string. The basic HTML uses `<xxx></xxx>` to flank the content for display in a web browser. `h2` is a header tag: a standard HTML element used to define page headings. You can think of `h2` as the Heading 2 style in Microsoft Words® (Figure 2). The header tags go from `h1` to `h6`, with `h1` being the largest and `h6` being the smallest. So, what does this mean, and why do we care? This means that the *Shiny* package's functions turn the R codes into HTML codes, so people who know the R language do not have to re-learn a new programming language from the beginning before creating WebApps that are based on HTML. These *Shiny* functions also helped me learn HTML. Moreover, one can replace `titlePanel("Modified Finnegan Scoring System")` with `HTML("<h2>Modified Finnegan Scoring System</h2>")`, and still get the same result because they both result in the same output. The implication from this example is that people who know HTML well can further fine-tune the user interface with HTML. The possibility is enormous.

“The implication from this example is that people who know HTML well can further fine-tune the user interface with HTML. The possibility is enormous.”

Similarly, going down to the sixth argument, `h4("Central Nervous System Disturbances")` gives the output of `<h4>Central Nervous System Disturbances</h4>`. Because `h4()` aims to translate R codes into HTML codes, rules applicable to HTML codes will be applicable to the corresponding R codes. For example, `<h4 style="color:red;>Central Nervous System Disturbances</h4>` will turn the flanked words into red font color. The corresponding R code writes `h4("Central Nervous System Disturbances", style="color:red;")`. You can even add an `align="right"` argument to align the words to the right border of the web browser. Try it for yourself if you will. In R, `h4()` is called a function; in HTML, `<h4></h4>` is called a tag. The *Shiny* package has prepared numerous useful functions that will translate into corresponding HTML tags, available here: <https://shiny.rstudio.com/articles/tag-glossary.html>. I found that some functions require the prefix of `tags$`, others don't. A safe approach is always to add the prefix.

2. Widget Arrangement

As mentioned before, webpages are divided into twelve invisible stripes of equal size. If not specified, the widgets are lined up horizontally (by row) unless a line break is indicated. `<h4></h4>` (or `h4()` in R) automatically adds to line break at the end, so as `<hr/>` (or `hr()` in R, which is used to draw a horizontal line). Other don't, especially widgets. The function `fluidRow()` is used to force the arrangement of widgets or other items by column. The first and only argument I typically use inside `fluidRow()` function is a sub-function called `column()`, inside which the first argument

LLUCH NICU NAS Scoring x +

lluchnicu.org/NICU_Manual/NAS_scoring/

Modified Finnegan Scoring System

Total Score: 0

Central Nervous System Disturbances

High-Pitched Cry

☒ None

☐ Excessive high-pitched (or other) cry (2 points)

☐ Continuous high-pitched (or other) cry (3 points)

Sleeps

is used to define the number of stripes this “column” will occupy. Here I put 12 so it occupies the entire webpage. By doing this, it is guaranteed no widgets or items will be placed side by side. The arrangement can only be done vertically. `column()` takes an optional argument of `offset=n`. It should be straightforward to the readers that the items will be placed on the $n+1$ stripe. For example, `fluidRow(column(5, offset=3))` will place the widgets/items on stripes 4-8.

“Last month, we introduced that a Shiny WebApp has two parts, the user interface, and the server function. Practically, this means that designing the user interface is uncoupled from the backend computation.”

Last month, we introduced that a Shiny WebApp has two parts, the *user interface*, and the *server function*. Practically, this means that designing the user interface is uncoupled from the backend computation. Therefore, when I start working on a new WebApp, I like to start with outlining the placement of the input widgets and other items, including the logo, text, icon, etc., and just run the App so I can appreciate it visually. I then create simple placeholders for the outputs. By doing this, I can visually assess the WebApp and write the codes for the server functions accordingly, ensuring that the WebApp is intuitive to the users and is easy to use. I don't focus

too much on making the WebApp fast and computation-efficient initially, which probably goes against the dogma that software engineers hold firmly. Having easy-to-use WebApps that allow me to be more efficient with my busy life as a clinician outweighs the importance of computational efficiency. This goal is achieved by having WebApps that are user-friendly. After all, these WebApps I develop for daily use don't contain high-level logic. I can't tell if a task takes 2 vs. 4 microseconds, even though that means doubling the computation time and may terrify engineers.

“After all, these WebApps I develop for daily use don't contain high-level logic. I can't tell if a task takes 2 vs. 4 microseconds, even though that means doubling the computation time and may terrify engineers.”

3. CSS and Javascript

It is worth mentioning that the developers can add custom cascading style sheets (CSS) to the R codes to polish up the webpages. Additionally, JavaScript can be added to improve the user experience of WebApps further. Several R packages have been developed to wrap JavaScript into R functions for easy adoption, such as *shinyjs*, *shinyBS*.^(4,5) I used the *dashboard* layout for the Neonatology Today website, with the sidebar placed on the right side. The current and past issues reside in their own boxes, such as

the search function, leaderboard advertisement, editorial board, and authors' art. The carousals for the announcement (below today's date in the title area), the advertisement leaderboard, and the authors' art were made possible with additional CSS codes. The closing of the sidebar panel after a button is clicked and the hiding and appearance of the input widgets, for example, were made possible by using the JavaScript wrapper functions from the aforementioned packages. I have to admit that I am not all that fluent with CSS, and I don't know how to write JavaScript, but I would also say that, with some effort, you can always find example codes on the internet. Once you see an example, it's not that difficult to extrapolate and customize it to your own needs.

To summarize, in this article, we discussed how the functions in the *Shiny* package translate R codes into HTML codes. We also discussed the function used to force the alignment of the widgets.

Next month, we will further discuss the *server function's reactivity*, which is why I love *Shiny* WebApps so much.

References

1. *Shiny Dashboard* [Internet]. Available from: <https://rstudio.github.io/shinydashboard/>
2. *AdminLTE* [Internet]. Available from: adminlte.io
3. *Bootstrap 3* [Internet]. Available from: <https://getbootstrap.com/docs/3.3/>
4. *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds* [Internet]. Available from: <https://cran.r-project.org/web/packages/shinyjs/index.html>
5. *shinyBS: Twitter Bootstrap Components for Shiny* [Internet]. Available from: <https://cran.r-project.org/web/packages/shinyBS/index.html>

Acknowledgment: The author would like to thank Dr. Shaina Lodhi in the Division of Neonatology, Department of Pediatrics at Loma Linda University School of Medicine for her insightful comments and constructive feedback on this article's content.

Disclosure: The author identifies no conflict of interest

NT

Corresponding Author

Fu-Sheng Chou, MD, PhD -
Senior Associate Editor,
Director, Digital Enterprise
Neonatology Today
Assistant Professor of Pediatrics
Division of Neonatology, Department of Pediatrics
Loma Linda University Children's Hospital
FChou@llu.edu

Neonatology Today's Digital Presence

Neonatology Today's now has a digital presence. The site is operational now and defines the future look of our digital web presence. By clicking on this <https://www.neonatologytoday.org/web/>, researchers can download individual manuscripts both in digital format and as part of the original PDF (print journal). While the PDF version of Neonatology Today will continue in its present form, we envision that the entire website will be migrated to this format in the next several months. We encourage you to take a look, "kick the wheels," and let us know where we still need to improve.. We are working towards making the website more functional for subscribers, reviewers, authors and anyone else. Although we have not yet applied for inclusion in the National Library of Medicine Database (Pub-Med), this new format meets several of the important metrics for this ultimate goal. As of December, 2020, NT has its own account with CrossRef and will assign DOI to all published material.

As we indicated last month, we look forward to a number of new features as well.

1. An online submission portal: Submitting a manuscript online will be easier than before. Rather than submitting by email, we will have a devoted online submission portal that will have the ability to handle any size manuscript and any number of graphics and other support files. We will have an online tracking system that will make it easier to track manuscripts in terms of where they are in the review process.
2. Reviewers will be able to review the manuscript online. This portal will shorten the time from receipt of review to getting feedback to the submitting authors.
3. An archive search will be available for journals older than 2012.
4. A new section called news and views will enable the submission of commentary on publications from other journals or news sources. We anticipate that this will be available as soon as the site completes the beta phase
5. Sponsors will be able to sign up directly on the website and submit content for both the digital and PDF issues of Neonatology Today.

Neonatology Today will continue to promote our Academic True Open Model (ATOM), never a charge to publish and never a charge to subscribe.

If there are any questions about the new website, please email Dr. Chou directly at:

fu-sheng.chou@neonatologytoday.net

New subscribers are always welcome!

NEONATOLOGY TODAY

To sign up for a free monthly subscription,
just click on this box to go directly to our
subscription page

Readers can also follow

NEONATOLOGY TODAY

via our Twitter Feed

@NEOTODAY