

## **ASPIRE Web documentation**

Kathleen AshleyW

## Table of Contents

<b>I. Contents of Files</b>	<b>2</b>
A. /app :	2
a. all_census.RDS	2
b. all_schooldistricts.RDS	2
c. all_schoolna.RDS	2
d. Census_ASPIRE_Collapsed.CSV	3
e. Resources_ASPIRE_Collapsed.CSV	5
f. SchoolDistricts_ASPIRE_Collapsed.CSV	10
g. run2.R	11
h. global.R	11
i. server.R	12
j. ui.R	12
k. webpages.R	12
l. Google drive files	13
A. Website Fonts	13
B. OSU brand images	13
C. Past ASPIRE project files	13
D. Learning materials	13
E. ASP3IRE Prototype mockup	14
F. ASP3IRE conference poster	14
<b>II. Guide and Notes</b>	<b>15</b>
A. Dependencies for R Shiny	15
B. Instructions on how to run the application	16
C. Common Pitfalls	16
a. Fetching Data	17
b. Manipulating CSS of fetched data	18
D. Parameters in functions	19
E. Print Statements	19
<b>III. Format and Screenshots of Variables (from .csv file)</b>	<b>20</b>
A. Census Data Variables (all topics but environmental variable):	20
B. Environmental variable	21
C. Website Variables	22
D. School District Variables	23
<b>IV. Resources Used</b>	<b>25</b>
<b>V. Future Suggestions</b>	<b>27</b>
<b>VI. Contact</b>	<b>28</b>



## I. Contents of Files

The website is broken down into a few files as follows:

### A. */app* :

The files associated with the app are stored in this directory.

#### a. *all\_census.RDS*

- Census Leaflet maps and histograms under the “Census” website page are pulled from this RDS file.

#### b. *all\_schooldistricts.RDS*

- School district Leaflet maps and histograms under the “School districts” website page are pulled from this RDS file.

#### c. *all\_schoolna.RDS*

- Since the *all\_schooldistricts.RDS* file contains NA values, we have to extract those NA values out of the file so we can figure out the average (mean) of the values. Using the *all\_schooldistricts.RDS* to perform this would cause an error. The example below is used in *server.R*.

```
mean = switch(input$grades,
  "Kindergarten" = mean(all_schoolna$F2021_22_Kindergarten),
  "Grade one" = mean(all_schoolna$F2021_22_Grade_One),
  "Grade two" = mean(all_schoolna$F2021_22_Grade_Two),
  "Grade three" = mean(all_schoolna$F2021_22_Grade_Three),
  "Grade four" = mean(all_schoolna$F2021_22_Grade_Four),
  "Grade five" = mean(all_schoolna$F2021_22_Grade_Five),
  "Grade six" = mean(all_schoolna$F2021_22_Grade_Six),
  "Grade seven" = mean(all_schoolna$F2021_22_Grade_Seven),
  "Grade eight" = mean(all_schoolna$F2021_22_Grade_Eight),
  "Grade nine" = mean(all_schoolna$F2021_22_Grade_Nine),
  "Grade ten" = mean(all_schoolna$F2021_22_Grade_Ten),
  "Grade eleven" = mean(all_schoolna$F2021_22_Grade_Eleven),
  "Grade twelve" = mean(all_schoolna$F2021_22_Grade_Twelve))
```

d. *Census\_ASPIRE\_Collapsed.CSV*

- The contents of the 'Census' page is generated from this file.

Format of the variables:

1. **Indicator:** the indicator related to the *image\_url* and

*summary*

2. **Image\_url :** Done in html format such as below

- <https://images.unsplash.com/photo-1562016600-ece13e8ba570?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWZyY2h8M3x8d2F0ZXJ8ZW58MHx8MHx8fDA%3D&w=1000&q=80>

- The url above can be retrieved by looking on google images > press right click on mouse on the image selected > open image in new tab > go to the new tab retrieved > press right click on mouse on the image > copy image address

- Example on how image\_url gets called:

Below, the uiOutput calls the picture\_holder\_pov from server.R,

```
div(class = "census-flex-container",  
  div(class = "census-item census-item-1",  
    div(id = "expandBox1",  
      tags$div(class = "inside-textbox",  
        tags$h3(class = "topic-title",  
          "Poverty Status, Oregonian residents, 2021-2022",  
          tags$button(  
            id = "expandButton1",  
            class = "btn btn-link",  
            type = "button",  
            "Expand"  
          )  
        ),  
      ),  
      uiOutput("picture_holder_pov"),  
      tags$p(class = "topic-p topic-p-1",  
        textOutput("census_summary_pov")  
      )  
    )  
  ),  
  )  
,
```

- Which then calls the function below that is defined

```

fetch_text_census = function(type, data, inp_type)                                in server.r:
{
  oregonr_csv <- read_csv("Census_ASPIRE_Collapsed.csv")
  indicator = inp_type
  oregon_resources = oregonr_csv[oregonr_csv$Indicator == indicator, ]

  value = ""
  if(type == "url")
  {
    value = oregon_resources$image_url
  }
  if(type == "summary")
  {
    value = oregon_resources$Summary
  }
  return(value)
}

turn_to_list = function(args)
{
  ul= tags$ul()

  if(args == "resources")
  {
    oregon_r_names = fetch_text_wildfire("resources", "names")
    oregon_r_links = fetch_text_wildfire("resources", "links")

    ul$children = lapply(oregon_r_names, function(x) {
      tags$li(oregon_r_names[index])
    })
  }

  return(ul)
}

```

- Depending on the parameter, the function  
fetch\_text\_census extracts values from the .CSV  
file.

```

output$picture_holder_pov = renderUI({
  img_url = fetch_text_census("url", "", input$age)
  generatePic(img_url)
})

```

- 
- After fetching is done, *generatePic* will generate the  
picture according to the photo\_url passed as  
parameter, and will format it under “img-topic”  
class. This CSS can be found in the *ui.R* file.

```
generatePic <- function(photo_url)
{
  tags$img(src = photo_url, height = 170, class = "img-topi
}
```

○

3. **Summary** : Write a summary on the variable selected
4. **Topic**: Which topic the indicator belongs to
5. **Topic\_Desc**: Description of the topic
6. **Indicator\_Title**: The Indicator title, look at the position of this variable in the [third section](#).

e. **Resources\_ASPIRE\_Collapsed.CSV**

The variables in this file are related to the “Resources” topic under the census page.

■ Format of the variables:

1. **Indicator**: The name of the variables related to the other information stored in the csv file (Summary, Image\_url, etc)
2. **Summary**: The summary related to the Indicator (The indicator Wildfire Smoke will have wildfire smoke summary)
3. **Image\_url** :
  - The url is written as this format:
 

<https://images.unsplash.com/photo-1562016600-ece13e8ba570?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFrY2h8M3x8d2F0ZXJ8ZW58MHx8MHx8fDA%3D&w=1000&q=80>
  - The url above can be retrieved by looking on google images > press right click on mouse on the image

selected > open image in new tab > go to the new  
tab retrieved > press right click on mouse on the  
image > copy image address

4. ***Resources\_Num:*** The number of resources that is listed under “*Resource\_Names*” and “*Resource\_Links*”. **The amount of Resources listed under the variables listed above must be the same.**

5. ***Resource\_Names:*** The list of names of resources, separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_ ).

- Newline character can be created in csv file through:

- i. Option + Enter in Macbook
- ii. Alt + Enter on Windows

6. ***Resource\_Links:***

- The list of links of resources, separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_ ).

- i. Newline character can be created in csv file through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows



7. ***Oregon\_Programs\_Num*** : The number of Oregon programs available for a particular indicator (this is the amount of Oregon\_Programs\_Name, Oregon\_Programs\_Links and Oregon\_Programs\_Desc).

**The amount of Oregon programs listed under the variables listed above must be the same.**

8. ***Oregon\_Programs\_Name***

- The list of Oregon Program names separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_).

- i. Newline character can be created in csv file through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows

9. ***Oregon\_Program\_Links***

- The list of links of Oregon Program Links, separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_).

- i. Newline character can be created in csv file through:

1. Option + Enter in Macbook

2. Alt + Enter on Windows

#### ***10. Oregon\_Program\_Desc***

- The list of Oregon Program Desc, separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_).

- i. Newline character can be created in csv file through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows

***11. Policies\_Num:*** the amount of policies listed under policies\_desc and policies\_links **(they both must be the same amount).**

#### ***12. Policies\_Desc***

- The list of Policies description separated by newline character (Must be separated with newline character as it is retrieved in that way from the files \_ and \_).

- i. Newline character can be created in csv file through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows

#### ***13. Policies\_Links***

- The list of Oregon Program names separated by  
newline character (Must be separated with newline  
character as it is retrieved in that way from the files  
\_ and \_).

- i. Newline character can be created in csv file  
through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows

#### ***14. LR\_Num***

- The amount of local resources listed under  
LR\_Links and LR\_Desc.

#### ***15. LR\_Links***

- The list of Local Resource Links separated by  
newline character (Must be separated with newline  
character as it is retrieved in that way from the files  
\_ and \_).

- i. Newline character can be created in csv file  
through:

1. Option + Enter in Macbook
2. Alt + Enter on Windows

#### ***16. LR\_Desc***

- The description of local resources listed on LR\_Links. Separated by newline character as it is retrieved that way from file \_ and \_.

*f. SchoolDistricts\_ASPIRE\_Collapsed.CSV*

The variables in this file are related to the “School Districts” page.

■ Format of the variables:

1. **Indicator:** The name of the variables related to the other information stored in the csv file (Indicator, Summary, Image\_url, etc)
2. **Summary:** The summary related to the Indicator (The indicator Water will have water summary)
3. **Image\_url :**
  - The url is written as this format:  
<https://images.unsplash.com/photo-15620166600-ece13e8ba570?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWZyY2h8M3x8d2F0ZXJ8ZW58MHx8MHx8fDA%3D&w=1000&q=80>
  - The url above can be retrieved by looking on google images > press right click on mouse on the image selected > open image in new tab > go to the new tab retrieved > press right click on mouse on the image > copy image address
4. **Topic:**  
  
Topic will be the topic the indicator is related to. For instance, Resources will be the topic for water.
5. **Topic\_Desc:**

The description of the topic. With the example above,  
Topic\_Desc will be the description for Resources.

**6. Indicator\_Title:**

This will be the title for the indicator, to know where this is  
located in the UI/Webpage, go to the [third section](#).

**g. run2.R**

The file used by heroku to run the application together.

**h. global.R**

Where global values or global functions are kept. Some examples:

1. generateListItems take the array of names and links, and for  
each name and link create a list of them in the format  
underneath the function picture below.

```
generateListItems <- function(array, linksArr) {  
  tags$sul(class="bullet-points",  
    lapply(seq_along(array), function(i) {  
      tags$li(class="bullet-li",  
        tags$a(class="blue-link", href = linksArr[i], array[i]))  
    })  
}
```

- 2.



*i. server.R*

Server contains the function that gets called from webpages.R or ui.R. In this case, usually those functions get called with statements that end with Output. For instance:

- uiOutput()
- textOutput()
- leafletOutput()
- plotlyOutput()

Make sure to read one of the common pitfalls for these type of calls in [this section](#).

*j. ui.R*

Contains the css, HTML element or layout for the header, mobile-header, and footer.

- Media Queries
  1. Media queries are used in the development of this website to make the website accessible across many platforms (To get more information on media queries, [click here](#)).

*k. webpages.R*

Webpages contain the HTML structure of each of the webpages. Listed on this file should be the home\_page, censusdata, schooldistricts.

- As a note, the router of the website is located on global.R:

```
26 # The router used to navigate to each of the website.
27 router <- make_router(
28   route("/", home_page),
29   route("censusdata", censusdata),
30   route("schooldistricts", schooldistricts)
31 )
```

## I. [Google drive files](#)

### A. Website Fonts

The fonts ‘Stratum2WebBold.woff’ and ‘Stratum2WebBold.woff2’ was used for some fonts in the website such as the titles for the indicators and topics in the school district page and the census page. This font has been included in the repo underneath /www folder.

### B. OSU brand images

This file was given by the OSU lead developer which contains the OSU logos that can be used throughout the website to keep the websites more uniform.

### C. Past ASPIRE project files

Included in this file is schooldistrict-form.R that was a previous project given by Perry and Andy. If it’s needed, this file can still be used and bug-free.

### D. Learning materials

I have also included some of my notes and my side-projects while reading the R book that was also put in this file.

#### E. ASP3IRE Prototype mockup

The final prototype for the website that was designed. This could help give visual to what the end-product would look like. This was designed through Figma.

- If there are some additional adjustments, the changes can be made here on [Figma](#). This link should enable you to edit the file.

#### F. ASP3IRE conference poster

To understand the project a bit more, I have also included the ASPIRE conference poster which includes the features of the website and along with the purpose.



## II. Guide and Notes

### A. Dependencies for R Shiny

After running “sessionInfo()” on R to get information on base packages, R version, and other loaded packages, here are the results as reference when developing this project:

```
> sessionInfo()
R version 4.0.3 (2020-10-10)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS 13.5.1

Matrix products: default
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats      graphics  grDevices  utils      datasets  methods   base

other attached packages:
[1] shinyscreenshot_0.2.0  fresh_0.2.0      plotly_4.10.0     plyr_1.8.7      raster_3.5-21
[6] tigris_1.6.1           sf_1.0-7         htmltools_0.5.2   htmlwidgets_1.5.4  leaflet_2.1.1.9000
[11] shinyjs_2.1.0          shinythemes_1.2.0  forcats_0.5.1     stringr_1.4.0     dplyr_1.0.9
[16] purrr_0.3.4           readr_2.1.2      tidyr_1.2.0       tibble_3.1.7      ggplot2_3.3.6
[21] tidyverse_1.3.1       shiny.router_0.2.3  shiny_1.7.1       mapview_2.11.0    sp_1.5-0

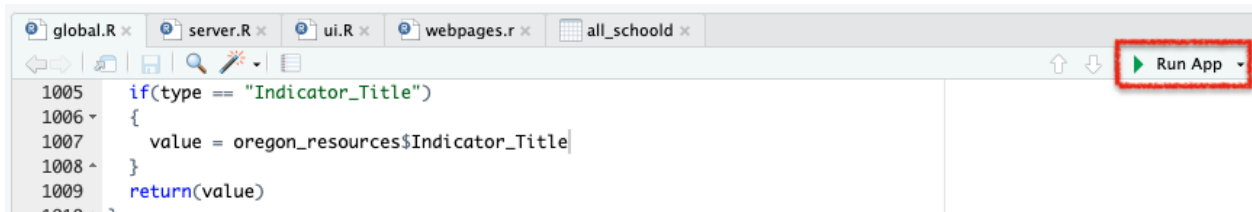
loaded via a namespace (and not attached):
[1] leafem_0.2.0           colorspace_2.0-3    ellipsis_0.3.2      class_7.3-20
[5] rsconnect_0.8.26       rgdal_1.5-29        satellite_1.0.4      base64enc_0.1-3
[9] fs_1.5.2               rstudioapi_0.13     proxy_0.4-27         farver_2.1.0
[13] bit64_4.0.5           fansi_1.0.3         lubridate_1.8.3      xml2_1.3.3
[17] codetools_0.2-18      cachem_1.0.6        jsonlite_1.8.0       broom_1.0.0
[21] dbplyr_2.2.1          png_0.1-7           compiler_4.0.3       httr_1.4.3
[25] backports_1.4.1       assertthat_0.2.1    fastmap_1.1.0        lazyeval_0.2.2
[29] cli_3.3.0             later_1.3.0         leaflet.providers_1.9.0 tools_4.0.3
[33] gtable_0.3.0          glue_1.6.2          rappdirs_0.3.3       Rcpp_1.0.8.3
[37] cellranger_1.1.0      jquerylib_0.1.4     vctr_0.4.1           crosstalk_1.2.0
[41] rvest_1.0.2           mime_0.12           lifecycle_1.0.1      terra_1.5-21
[45] scales_1.2.0          vroom_1.5.7         hms_1.1.1            promises_1.2.0.1
[49] parallel_4.0.3        RColorBrewer_1.1-3  yaml_2.3.5           memoise_2.0.1
[53] sass_0.4.1            stringi_1.7.6       maptools_1.1-4       e1071_1.7-11
[57] rlang_1.0.3           pkgconfig_2.0.3     lattice_0.20-45      labeling_0.4.2
[61] bit_4.0.4             tidyselect_1.1.2    magrittr_2.0.3       R6_2.5.1
[65] generics_0.1.2        DBI_1.1.3           pillar_1.7.0         haven_2.5.0
[69] foreign_0.8-82        withr_2.5.0         units_0.8-0          modelr_0.1.8
[73] crayon_1.5.1          uuid_1.1-0          KernSmooth_2.23-20   utf8_1.2.2
[77] tzdb_0.3.0           grid_4.0.3          readxl_1.4.0         data.table_1.14.2
[81] reprex_2.0.1          digest_0.6.29       classInt_0.4-3       webshot_0.5.3
[85] xtable_1.8-4          httpuv_1.6.5        stats4_4.0.3         munsell_0.5.0
[89] viridisLite_0.4.0     bslib_0.4.0
```

## B. Instructions on how to run the application

1. After downloading R, make sure to have the packages needed installed by running this command:

```
library(c("shiny", "tidyverse", "shiny.router",  
"shinythemes", "shinyjs", "leaflet", "htmlwidgets", "tigris",  
"raster", "plyr", "dplyr", "plotly", "fresh",  
"shinyscreenshot"))
```

2. Afterwards, open the following files for easy access to manipulating access to the website itself:
  - a. global.R
  - b. server.R
  - c. ui.R
  - d. webpages.R
3. Navigate to either the ui.R, global.R or server.R and then run the program by clicking the 'Run app' button on the right upper side of the screen.



## C. Common Pitfalls

While creating this document, I kept on running into the same issue and forgetting about how I resolved it. To make it easier on you, I have compiled a few things to check or make sure it's correct when you are running into issues!

### a. Fetching Data

When fetching some data from the ui.R to server.R make sure several things are correct:

#### 1. The type of item being fetched/ generated

I kept on making the same issue regarding this. For instance, imagine this scenario:

- You're trying to fetch text from the file

Website\_ASPIRE.csv, so you make the following call to project\_1\_desc by the following line on ui.R:

```
uiOutput("project_1_title"),
```

- The definition of project\_1\_desc is as follows:

```
output$project_1_desc = renderUI({  
  text = fetch_text_webpage("Project_1_Desc")  
  generateWebText("Project_1_Desc", text)  
})
```

- You have to double check on what the end goal of calling this function is, are you trying to fetch an HTML element or purely text?

- If you're trying to fetch only text, make sure that instead of renderUI, it says **renderText** on the function definition in the server.R. Use textOutput("[name\_of\_function]") on the ui.R.
- If you're trying to fetch an HTML content, make sure to use uiOutput on the ui.R as the example

above mentions and to use **renderUI** on the function definition in the server.R.

## b. Manipulating CSS of fetched data

Something to note is that texts that are fetched from the server will have their own class and it is a bit hard to manipulate that. To make the texts fetched wrapped in classes/CSS elements, do this instead:

```
output$project_1_desc = renderUI({
  text = fetch_text_webpage("Project_1_Desc")
  generateWebText("Project_1_Desc", text)
})
```

Fetch the text using `fetch_text_webpage`, and then generate an HTML element with that text. An example of the `generateWebText` function is shown below

```
generateWebText <- function(type, text) {
  if (type == "Mission_Text") {
    return(
      tags$p(class = "normal-p", text)
    )
  }
  if (type == "Project_1_Title" || type == "Project_2_Title" || type == "Project_3_Title" || type == "Project_4_Title") {
    # We're routing the titles to each perspective webpages. route_link will automatically link the webpages with
    # the webpages assigned from webpages.R.
    if(type == "Project_1_Title")
    {
      return(
        tags$h3(class = "flexbox-title",
          tags$a(class = "flexbox-title-link", href = route_link("censusdata"), text))
      )
    }
    if(type == "Project_2_Title")
    {
      return(
        tags$h3(class = "flexbox-title",
          tags$a(class = "flexbox-title-link", href = route_link("schoolistricts"), text))
      )
    }
    if(type == "Project_3_Title" || type == "Project_4_Title")
    {
      return(
        tags$h3(class = "flexbox-title", text)
      )
    }
  }
  if(type == "Project_1_Desc" || type == "Project_2_Desc" || type == "Project_3_Desc" || type == "Project_4_Desc")
  {
    return(
      tags$p(class = "flexbox-p", text)
    )
  }
}
```

## D. Parameters in functions

The functions that are declared in `server.r` or `global.R` have a function definition on top of them, make sure to read them to understand how the function works, what it returns, what the accepted parameters are and how the function behaves.

For instance, this particular function, `fetch_text_wildfire`, has so many requirements. Make sure to read the text to save some time!

```
#-----  
# fetch_text_wildfire  
#-----  
# This was created for the environment variables section of Census Data.  
# - Any variables associated here are already documented in the documentation  
# underneath the third section which is Format and Screenshots of Variables.  
# > Purpose:  
#   This fetches texts from the file "Resources_ASPIRE_Collapsed.csv"  
#   and uses the texts to dynamically populate the website.  
# > Parameters:  
#   - type: Character string indicating the type of data to retrieve ("summary", "title", "url", "resources", "policies", "localResources").  
#   - data: Character string specifying additional data retrieval ("num", "names", "links", "desc")  
#     based on the selected type (see note below)  
#   - inp_type: Character string specifying the indicator to filter the data.  
# > Returns:  
#   - A character string containing the requested data based on the specified type and data.  
# > Example:  
#   If you're trying to fetch Resource_Names for Wildfire Smoke, this is the call you'd make:  
#     array_name = as.array(fetch_text_wildfire("localResources", "names", input$env_var))  
#   If you're trying to fetch Policies Links for Well Water Contamination, this is the call:  
#     array_name = as.array(fetch_text_wildfire("policies", "names", input$env_var))  
#   If you're trying to get the img_url for an indicator, do this:  
#     img_url = fetch_text_wildfire("url", "", input$env_var)  
# *note:  
#   > Since some of the variables (Resource_Names, Resource_Links, Oregon_Program_Name has a list of  
#     sentences, this function would return as an array which is why the examples above require as.array()  
#     wrapping the response. Otherwise, the function could be called by doing simply fetch_text_wildfire()  
#     without as.array() wrapping it.  
#   > Depending on the call you're trying to make, the parameter "data" can either be "names", "links", "desc" or "num".  
#     at times when you don't need it (the example of the photo_url or summary), you could put "" or "0" on the  
#     data parameter.  
#-----
```

## E. Print Statements

I have left some print statements commented on the code to make it easier to debug the code or to know how the function itself works! Just simply remove the '#' to start!

### III. Format and Screenshots of Variables (from .csv file)

#### A. Census Data Variables (all topics but environmental variable):

The screenshot below corresponds to the *Census Data* page and the .csv file named *Census\_ASPIRE\_Collapsed.csv*. To modify any of the variables listed below, make sure to make changes in the *Census\_ASPIRE\_Collapsed.csv* file, and make sure to follow the requirements listed in [Content of Files](#).

- **\*\*As a note**, the topic section of the *Census\_ASPIRE\_Collapsed* was not used in any of the files, it is just put in the .csv file to make it easier to make a correlation between the indicator and the topic.
- This format follows the following topics:
  - *Resources, Population in Household by Age, Health Insurance Coverage, Poverty Status*

Choose a topic  
Poverty Status


Topic

Choose a variable  
Under 18

Indicator

Health Coverage, Oregon, 2021-2022

Collapse



Image\_url

Topic Desc

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

About Male under 18 with health insurance coverage

Collapse


Indicator Title

Summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

## B. Environmental variable

The screenshot below corresponds to the topic environmental variable on the census data page and the .csv file named *Resources\_ASPIRE\_Collapsed.csv*. To modify any of the variables listed below, make sure to make changes in the *Resources\_ASPIRE\_Collapsed.csv* file, and make sure to follow the requirements listed in [Content of Files](#).

<b>Children's Blood Lead Levels</b>		<b>Indicator</b>	<a href="#">Collapse</a>
		<p>The total amount of lead detected in children's blood is representative of blood lead levels. Lead is a heavy metal that can be found in several different sources including lead dust, plumbing, and children's products, among others. Lead poses a threat to human health because it can enter the body via consumption and inhalation, where it can enter the blood. For children, lead can be particularly harmful because their growing bodies will absorb more lead than compared to adults, making them more vulnerable to its effects. Lead exposure as a child is harmful as it can disrupt neurological and physical development, leading to adverse developmental outcomes later in life, such as learning disorders and behavioral problems.</p>	
<b>Resources</b>		<a href="#">Collapse</a>	
<ul style="list-style-type: none"><li>• OHA Lead Poisoning and Exposure to Lead</li><li>• OHA Sources of Lead</li><li>• OHA Signs &amp; Effects of Lead</li><li>• UW &amp; PEHSU Pediatric Lead Exposure in Oregon</li><li>• Oregon School Drinking Water Lead Tests (2016)</li><li>• EPA Water Contaminants in Children's Drinking Water</li></ul>		<b>Links:</b> <b>Oregon_Program_Link</b> <b>Titles:</b> <b>Oregon_Program_Name</b>	
<b>Policies</b>	<b>Policies_Desc</b>	<b>Links:</b> <b>Policies_Links</b> <b>Titles:</b> <b>Policies_Titles</b>	<a href="#">Collapse</a>
• Oregon State Senate Bill 64	Oregon State Senate Bill 65 established lead-based paint regulations, and developed a plan for creating school-based health centers.		
• Oregon State Senate Bill 426	Oregon State Senate Bill 426 regulates the testing of water in schools as well as for lead-based paint.		
• Oregon State House Bill 2842	Oregon State House Bill 2842 relates to healthy housing in the state, including lead abatement.		
• Oregon State House Bill 2077	Oregon State House Bill 2077 created guidelines for lead-based paint inspection as well as developed programs for blood lead testing to reduce children's exposure to lead.		
• President Biden's Bipartisan Infrastructure Law Oregon	President Biden's Bipartisan Infrastructure law included funding specifically to Oregon, where \$37 million is dedicated to lead pipe replacement, and \$24 million invested in safe drinking water.		
• Oregon CDC Funding	The Centers for Disease Control and Prevention provided the state of Oregon with \$350,000 towards addressing childhood lead poisoning and exposure programs and research.		
• OSHA Reducing Lead Exposure in School-Aged Children	The Oregon Health Authority created a statewide plan for reducing lead exposure in school-aged children in the state.		



Community-Based Organizations (CBOs)

Collapse

- Coalition of Communities of Color

The Coalition of Communities of Color is a community-based organization that advocates for clean water for communities of color in Clackamas, Multnomah and Washington counties. They have various health clinics in Oregon, where individuals can access testing for lead.
- Northwest Environmental Advocates

The Northwest Environmental Advocates is a community-based organization that focuses on the health of Oregonians, specifically in regards to clean water.
- Confederate Tribes of Siletz Indians Healthcare Clinic (Federally-recognized Tribal Members)

The Confederate Tribes of Siletz Indians Healthcare Clinic offers healthcare to federally recognized tribal members, including laboratory testing for lead exposure.
- Verde PDX

The community-based organization, Verde PDX, aims to create healthier communities in the Portland metroplex. They are working to expand access to clean water in the state.
- Oregon Water Futures Project

The Oregon Water Futures Project was created by community-based organizations and academic institutions in Oregon to make advancements towards clean water for oppressed groups across the state.

LR\_Desc

Links: LR\_Links  
Titles: LR\_Name

- *\*\*As a note, the variables listed below was not used in any of the server files, it was just to keep a track of the amount of said resources, etc.*
  - *Resources\_Num, Oregon\_Programs\_Num, Policies\_Num, LR\_Num*

### C. Website Variables

The screenshot below corresponds to the *homepage* and the .csv file

*Webpage\_ASPIRE.csv*.

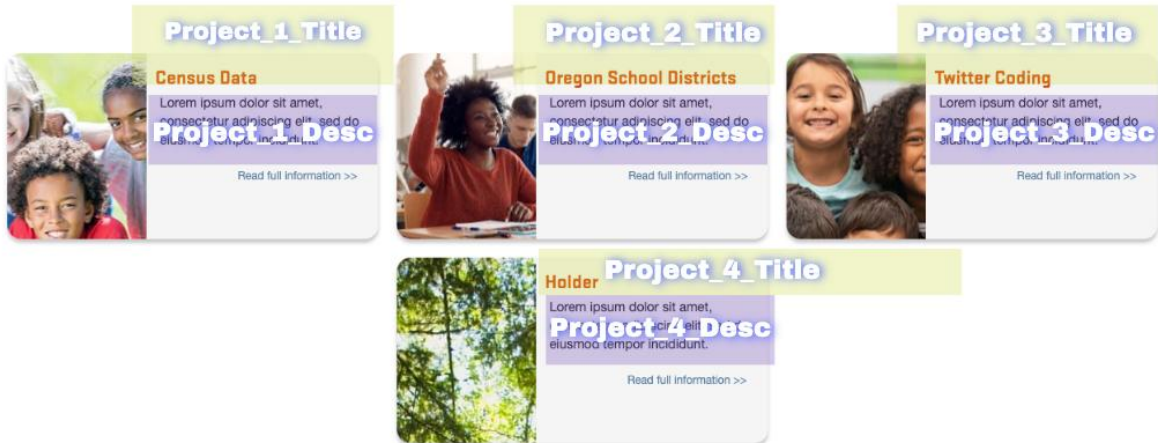
- Things to note:
  - The screenshot below shows the location of each variable of the website variables. Accessing them is shown in the `fetch_text_webpage` function in the `global.R` file (function definition is below)



## Our Mission

We are a dedicated team working together to improve children's health and wellbeing. Our efforts focus on accelerating the adoption of evidence informed policies, programs, and practices that reduce environmental exposures where children live, play, and go to school.

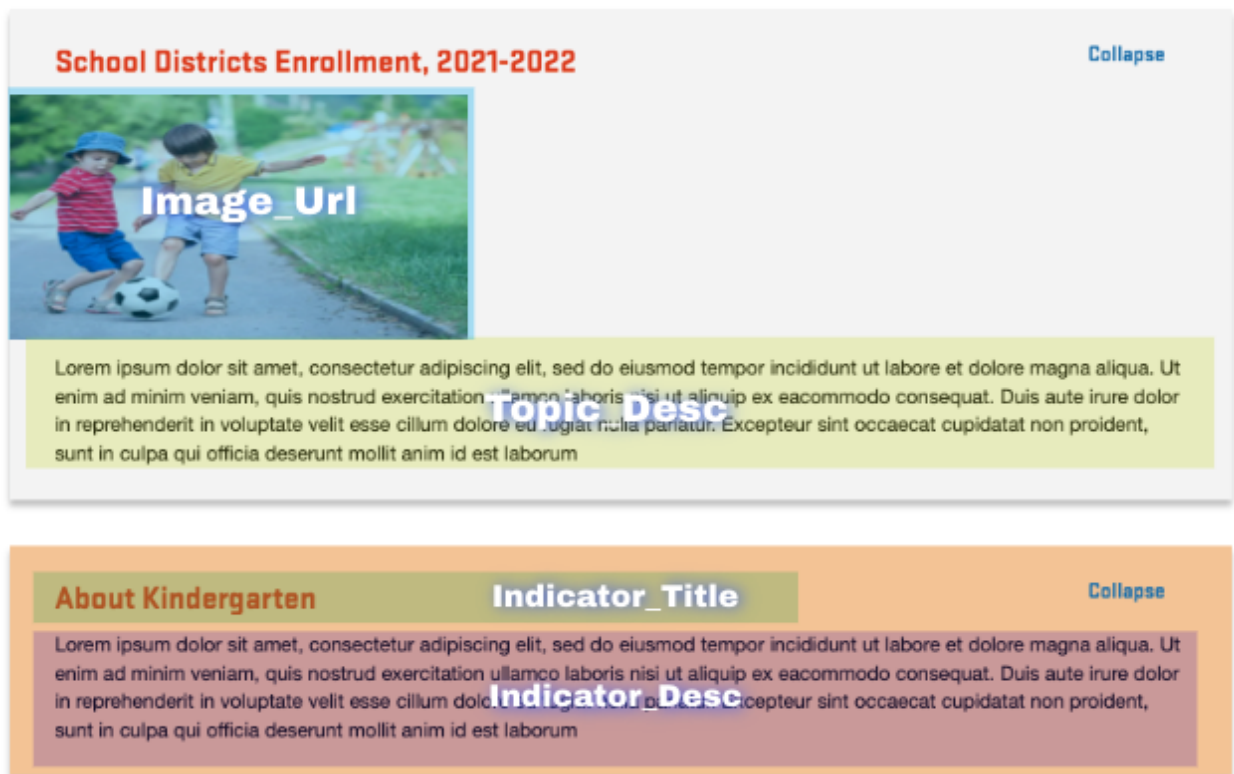
## Projects



```
fetch_text_webpage <- function(type) {  
  # Load the CSV file  
  webpage_text <- read.csv("Webpage_ASPIRE.csv")  
  
  oregon_wp = webpage_text[webpage_text$Variable == type, ]  
  
  return(oregon_wp$Text_Variable)  
}
```

## D. School District Variables

The screenshot below corresponds to the *School Districts* page and the .csv file *SchoolDistricts\_ASPIRE\_Collapsed.csv*.



- Note:
  - To actually change the title “School Districts enrollment, 2021-2022” as shown above, you would have to go to webpages.R where the statement is located.

```
div(class="census-flex-container",
  div(class = "census-item census-item-1", # Place expandButton here
    tags$div(id = "expandBox1",
      tags$div(class = "inside-textbox",
        tags$h3(class = "topic-title",
          "School Districts Enrollment, 2021-2022",
          tags$button(
            id = "expandButton1",
            class = "btn btn-link",
            type = "button",
            "Expand"
          )
        ),
      ),
    uiOutput("sd_image"),
    uiOutput("sd_topic_desc"),
  ),
),
```

Currently, the topic title is hard-coded due to the reason that the topic would always be the same for the variables in that section.

## IV. Resources Used

### A. Hands on Programming with R

This book helped me a lot in understanding the basic fundamentals of R during my first few months! If you have no knowledge of R at all, I recommend starting off here. The pdf of the book can be retrieved from this [link](#).

### B. Interactive Visualization tutorials

#### 1. [Interactive Map of 2019 Covid spread using R shiny tutorial](#)

- a) This tutorial is helpful as a step-by-step guide in creating an interactive map through R shiny! I recommend watching this if you want to understand how the interactive maps on the website work.
- b) Most of the maps on the website are made from trial and errors and suggestions from stackoverflow. I recommend to always google the questions you don't know the answer to because someone most likely has gone through it :)

### C. [Mastering Shiny](#)

This website contains a lot of step-by-step guides to making your first R shiny app. This was really helpful in getting myself to familiarize with the environment, how files are structured in R shiny, what packages to call or could be used for the project, etc.

### D. Media Queries

As mentioned above, media queries are used in the development of this website to make it compatible across many platforms. To get started on media queries, here are some links:

- [Using media queries](#)
- [Media queries examples](#)

## V. Future Suggestions

Upon commenting and revising my work, I have noticed that some things can be improved, such as:

- Making calls to functions more efficient
  - As of now, multiple calls are made from webpages.R regarding Indicator\_Title, Indicator\_Desc, and many more. For School districts page alone, each variable needs their own call such as below:

```
# for Grades
output$sd_topic_desc = renderUI({
  text = fetch_text_sd("Topic_Desc", input$grades)
  generateSchoolUI("Topic_Desc", text)
})

# for Race
output$sd_topic_desc_r = renderUI({
  text = fetch_text_sd("Topic_Desc", input$race_opt)
  generateSchoolUI("Topic_Desc", text)
})
```

- The reason for this was because for each topic, race and grades, they have different names for selectInput that need to be passed to fetch\_text\_sd. I was not able to add another argument from the webpages.R other than calling the function “sd\_topic\_desc\_r” since we could only do the following:

```
uiOutput("sd_topic_desc_r"),
```

- If we could cut down the number of functions to fetch different things from the fetch functions, and somehow detect which topic

it's needing information for and switch between `input$grades` and `input$race_opt`, it would make the app much more efficient.

- Separating the CSS from `ui.R`
  - To make the styling for the website more accessible, separating the CSS from `ui.R` would be great for readability.
- Store the images used throughout the website (banner, images on the website main screen) in the `Website_ASPIRE.csv`.

## **VI. Contact**

While this documentation has been made to my best of ability, I realize that there could be questions that cannot be answered through this document. If any developers ever want to reach out to ask questions, please don't hesitate! I can be reached out at [kathleenashley0402@gmail.com](mailto:kathleenashley0402@gmail.com).