STATS 220

Web scraping

Web technology

I thank Dr Emi Tanaka for this part, adapted from her "Communicating with Data" course.

World Wide Web (WWW)

WWW (or the **Web**) is the information system where documents (web pages) are identified by Uniform Resource Locators (**URL**s)

A web page consists of:

- **HTML** provides the basic structure of the web page
- **CSS** controls the look of the web page (optional)
- **JS** is a programming language that can modify the behaviour of elements of the web page (optional)

Hypertext Markup Language (HTML)

- with the extension `.html`.
- rendered using a web browser via an URL.
- text files that follows a special syntax that alerts web browsers how to render it.

HTML structure

```
<!DOCTYPE html>

<html>
  <!--This is a comment and ignored by web client.-->
  <head>
    <!--This section contains web page metadata.-->
    <title>STATS 220 Data Technology</title>
    <meta name="author" content="Earo Wang">
    <link rel="stylesheet" href="css/styles.css">
  </head>

  <body>
    <!--This section contains what you want to display on your web page.-->
    <h1>I'm a first level header</h1>
    <p>This is a <b>paragraph</b>.</p>
  </body>
</html>
```


HTML syntax

`<span style="color:blue;">Author content</span>` → Author content

start tag:	`<span style="color:blue;">`
end tag:	`</span>`
content:	`Author content`
element name:	`span`
attribute:	`style="color:blue;"`
attribute name:	`style`
attribute value:	`color:blue;`

Not all HTML tags have an end tag:

`<img height="40px" src="https://tinyurl.com/rlogo-svg">` →

HTML elements

block element:	`<div>content</div>`
inline element:	`<span>content</span>`
paragraph:	`<p>content</p>`
header level 1:	`<h1>content</h1>`
header level 2:	`<h2>content</h2>`
italic:	`<i>content</i>`
emphasised text:	`<em>content</em>`
strong importance:	`<strong>content</strong>`
link:	`<a href="https://stats220.earo.me/">content</a>`
unordered list:	`<ul>`
	`<li>item 1</li>`
	`<li>item 2</li>`
	`</ul>`

Cascading Style Sheet (CSS)

- with the extension `.css`
- 3 ways to style elements in HTML:
  - **inline** by using the `style` attribute inside HTML start tag:

    ```
    <h1 style="color:blue;">Blue Header</h1>
    ```

  - **externally** by using the `<link>` element:

    ```
    <link rel="stylesheet" href="styles.css">
    ```

  - **internally** by defining within `<style>` element:

    ```
    <style type="text/css">
    h1 { color: blue; }
    </style>
    ```

By convention, the `<style>` and `<link>` elements tend to go into the `<head>` section of the HTML document.

CSS syntax

```
<style type="text/css">
h1 { color: blue; }
</style>
<h1>This is a header</h1>
```

This is a header

selector: `h1 { color: blue; }`

property: `h1 { color: blue; }`

property name: `h1 { color: blue; }`

property value: `h1 { color: blue; }`

You may have multiple properties for a single selector.➡️

```
h1 {
  color: blue;
  font-size: 16pt;
}
```


CSS properties

```
<div>Sample text</div>
```

background color: `div { background-color: yellow; }` Sample text

text color: `div { color: purple; }` Sample text

border: `div { border: 1px dashed brown; }` Sample text

left border only: `div { border-left: 10px solid pink; }` Sample text

text size: `div { font-size: 10pt; }` Sample text

padding: `div { background-color: yellow; padding: 10px; }` Sample text

margin: `div { background-color: yellow; margin: 10px; }` Sample text

CSS selector

`.classname` selects all elements with the attribute `class="classname"`.

`.c1.c2` selects all elements with *both* `c1` and `c2` within its class attribute.

`.c1 .c2` selects all elements with class `c2` that is a descendant of an element with class `c1`.

`#p1` selects all elements with the attribute `id="p1"`.

```
<h1>This is a sample html</h1>

<blockquote>
<p>Maybe stories are just d
<footer>—Brene Brown</foote
</blockquote>

<div id="p1" class="parent">
Hmm
<p>Hii!</p>
How are you?
<div class="child nice">
  <p>Hello!</p>
</div>
</div>

<p>Household 1</p>

<div class="parent">
<p>Hii!</p>
<blockquote class="child rebel">
  <p>Don't talk to me!</p>
</blockquote>
</div>

<span class="child">
<span class="parent child rebel">
  <p>Clean your room!</p>
</span>
</span>

<p>End of households</p>
```

⚠ Unlike `class`, you can only have one `id` value and must be unique in the whole HTML document.

JavaScript (JS)*

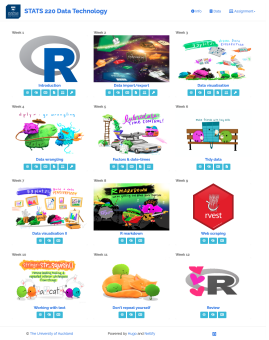
- JS is a programming language and enable interactive components in HTML documents.
- 2 ways to insert JS into a HTML document:
  - **internally** by defining within `<script>` element:

```
<script>
document.getElementById("p1").innerHTML = "content";
</script>
```

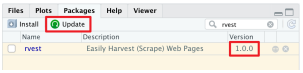
  - **externally** by using the `src` attribute to refer to the external file:

```
<script src="js/myjs.js"></script>
```

Web scraping



Use {rvest} `>= v1.0.0` (if not, update)




```
library(rvest)
course <- "https://stats220.earo.me"
stats220 <- read_html(course)
stats220
```

```
#> {html_document}
#> <html>
#> [1] <head>\n<meta http-equiv="Content ...
#> [2] <body>\n\n    <header class="site ...
```


Inspect elements

CSS selectors

`html_element()` select element

```
navbar <- stats220 %>%
  html_element(".navbar-right")
navbar
```

```
#> {html_node}
#> <ul class="nav navbar-nav navbar-right">
#> [1] <li><a href="/pages/info/"><i cla ...
#> [2] <li><a href="/pages/data/"><i cla ...
#> [3] <li class="dropdown">\n           ...
```

`html_name()` get element name

```
navbar %>%
  html_name()
```

```
#> [1] "ul"
```

`html_children()` get element children

```
navbar %>%
  html_children()
```

```
#> {xml_nodeset (3)}
#> [1] <li><a href="/pages/info/"><i cla ...
#> [2] <li><a href="/pages/data/"><i cla ...
#> [3] <li class="dropdown">\n           ...
```

`html_name()` get element name

```
navbar %>%
  html_children() %>%
  html_name()
```

```
#> [1] "li" "li" "li"
```


`html_text2()` get element text

```
navbar %>%
  html_children() %>%
  html_text2()
```

```
#> [1] "Info"
#> [2] "Data"
#> [3] "Assignment\nAssignment 1\nAssignment 2
```

`html_attr()` get element attributes

```
navbar %>%
  html_elements("a") %>%
  html_attr("href")
```

```
#> [1] "/pages/info/"
#> [2] "/pages/data/"
#> [3] "#"
#> [4] "/assignments/assignment1/"
```


`url_absolute()`: turn relative urls to absolute urls

```
navbar %>%
  html_elements("a") %>%
  html_attr("href") %>%
  url_absolute(course)
```

```
#> [1] "https://stats220.earo.me/pages/info/"
#> [2] "https://stats220.earo.me/pages/data/"
#> [3] "https://stats220.earo.me#"
#> [4] "https://stats220.earo.me/assignments/assignment1/"
#> [5] "https://stats220.earo.me/assignments/assignment2/"
#> [6] "https://stats220.earo.me/assignments/assignment3/"
#> [7] "https://stats220.earo.me/assignments/assignment1-sol/"
#> [8] "https://stats220.earo.me/assignments/assignment2-sol/"
```

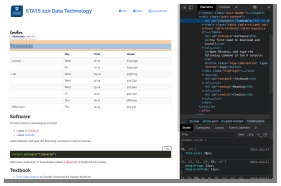

`html_elements()` select elements

```
navbar %>%
  html_elements("i")
```

```
#> {xml_nodeset (8)}
#> [1] <i class="fas fa-info-circle"></i>
#> [2] <i class="fas fa-database"></i>
#> [3] <i class="fas fa-laptop-code"></i>
#> [4] <i class="fas fa-laptop-code"></i>
#> [5] <i class="fas fa-laptop-code"></i>
#> [6] <i class="fas fa-laptop-code"></i>
#> [7] <i class="fas fa-key"></i>
#> [8] <i class="fas fa-key"></i>
```

fontawesome icons

```
stats220_info <- read_html(
  "https://stats220.earo.me/pages/info/")
```



select all `<h3>` elements

```
stats220_info %>%
  html_elements("h3") %>%
  html_text()
```

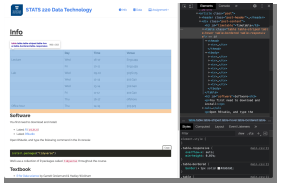
```
#> [1] "Timetable" "Software"  "Textbook"
#> [4] "Reading"   "Credits"
```

select `#timetable` id

```
stats220_info %>%
  html_elements("#timetable") %>%
  html_text()
```

```
#> [1] "Timetable"
```


select the first `<table>` element



```
stats220_info %>%
  html_element("table") %>%
  html_table()
```

```
#> # A tibble: 8 x 4
#>   ``             Day   Time  Venue
#>   <chr>          <chr> <chr> <chr>
#> 1 "Lecture"      Wed   16-17 Eng1.439
#> 2 ""             Fri   10-11 Eng1.439
#> 3 "Lab"          Wed   09-10 303S.175
#> 4 ""             Wed   12-13 302.G40
#> 5 ""             Wed   13-14 302.G40
#> 6 ""             Fri   11-12 302.G40
#> 7 ""             Thu   16-17 offshore
#> 8 "Office hour" Thu   14-15 303.323
```


Download pdf slides at once

```
stats220_urls <- stats220 %>%
  html_elements(".panel-body .btn") %>%
  html_attr("href")
stats220_urls
```

```
#>  [1] "/objs/obj01"
#>  [2] "/R/01-intro.R"
#>  [3] "/01-intro.Rmd"
#>  [4] "/01-intro.pdf"
#>  [5] "/labs/lab01"
#>  [6] "/labs/lab01-sol"
#>  [7] "/objs/obj02"
#>  [8] "/R/02-import-export.R"
#>  [9] "/02-import-export.Rmd"
#> [10] "/02-import-export.pdf"
#> [11] "/labs/lab02"
#> [12] "/labs/lab02-sol"
#> [13] "/objs/obj03"
#> [14] "/R/03-data-vis.R"
#> [15] "/03-data-vis.Rmd"
#> [16] "/03-data-vis.pdf"
```


Download pdf slides at once

```
library(stringr) # manipulate strings in week 10
(pdf_urls <- stats220_urls[str_detect(stats220_urls, "pdf")])
```

```
#> [1] "/01-intro.pdf"
#> [2] "/02-import-export.pdf"
#> [3] "/03-data-vis.pdf"
#> [4] "/04-data-wrangle.pdf"
#> [5] "/05-fcts-dates.pdf"
#> [6] "/06-tidy-data.pdf"
#> [7] "/07-data-vis2.pdf"
#> [8] "/08-rmd.pdf"
#> [9] "/09-web-scrape.pdf"
```

```
pdf_files <- str_remove(pdf_urls, "/")
purrr::walk2( # below for week 11
  url_absolute(pdf_urls, course), pdf_files,
  ~ download.file(url = .x, destfile = .y))
```

REST API

Github REST API

- Each URL is called a **request**.
- The data sent back to you is called an HTTP **response** that consists of headers and a body.

The root-endpoint of Github's API is https://api.github.com.

```
library(httr)
endpoint <- "https://api.github.com"
GET(endpoint)
```

```
#> Response [https://api.github.com]
#>   Date: 2021-05-12 00:20
#>   Status: 200
#>   Content-Type: application/json; charset=u
#>   Size: 2.31 kB
#> {
#>   "current_user_url": "https://api.gi...
#>   "current_user_authorizations_html_u...
#>   "authorizations_url": "https://api....
#>   "code_search_url": "https://api.git...
#>   "commit_search_url": "https://api.g...
#>   "emails_url": "https://api.github.c...
#>   "emojis_url": "https://api.github.c...
#>   "events_url": "https://api.github.c...
#>   "feeds_url": "https://api.github.co...
#> ...
```


HTTP methods

- `GET` to **retrieve** resource data/information only, and NO change in state of the resource
- `POST` to **create** new subordinate resources, e.g. upload a file
- `PUT` to **update/replace** an existing resource in its entirety
- `DELETE` to **delete** resources
- `PATCH` to make **partial update** on a resource (not all browsers support `PATCH`)

Path

The **path** determines the resource you're requesting for.

`GET /repos/{owner}/{repo}`

```
path <- "/repos/STATS-UOA/stats220"
resp <- GET(modify_url(endpoint,
  path = path))
resp
```

```
#> Response [https://api.github.com/repos/STAT
#>   Date: 2021-05-12 00:20
#>   Status: 200
#>   Content-Type: application/json; charset=u
#>   Size: 6.41 kB
#> {
#>   "id": 242031925,
#>   "node_id": "MDEwOlJlcG9zaXRvcnkyNDI...
#>   "name": "stats220",
#>   "full_name": "STATS-UOA/stats220",
#>   "private": false,
#>   "owner": {
#>     "login": "STATS-UOA",
#>     "id": 62915494,
#>     "node_id": "MDEyOk9yZ2FuaXphdGlvb...
#> ...
```

Parse the response

```
http_type(resp)
```

```
#> [1] "application/json"
```

Content type of a response

- `"image/png"`
- `"application/text"`
- `"application/csv"`
- `...`

```
content(resp)
```

```
#> $id
#> [1] 242031925
#>
#> $node_id
#> [1] "MDEwOlJlcG9zaXRvcnkyNDIwMzE5MjU="
#>
#> $name
#> [1] "stats220"
#>
#> $full_name
#> [1] "STATS-UOA/stats220"
#>
#> $private
#> [1] FALSE
#>
#> $owner
#> $owner$login
#> [1] "STATS-UOA"
#>
```


Status code

```
status_code(resp)
```

```
#> [1] 200
```

```
http_status(200) # OK
http_status(201) # Created
http_status(204) # NO CONTENT
```

```
http_status(400) # BAD REQUEST
http_status(403) # Forbidden
http_status(404) # NOT FOUND
```

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

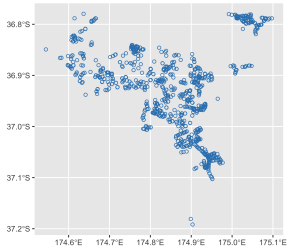
Auckland Transport Open GIS Data: bus stop

```
endpoint <- "https://services2.arcgis.com"
path <- "/JkPEgZJGxhSjYOo0/arcgis/rest/services/BusService/FeatureServer/0/"
query <- "query?where=1%3D1&outFields=*&outSR=4326&f=geojson"
bus_url <- parse_url(paste0(endpoint, path, query))
resp <- GET(bus_url)
cnt <- geojson::as.geojson(content(resp))
cnt
```

```
#> <geojson>
#>   type:  FeatureCollection
#>   features (n): 1000
#>   features (geometry / length) [first 5]:
#>     Point / 2
#>     Point / 2
#>     Point / 2
#>     Point / 2
#>     Point / 2
```



```
library(tidyverse)
library(sf)
bus_sf <- geojsonsf::geojson_sf(cnt)
ggplot() +
  geom_sf(data = bus_sf, pch = 1,
    colour = "#3182bd")
```



Reading

- Get started with {rvest}
- {httr} quickstart guide