Lecture 21: Oct 26, 2018

Web APIs

- *HTTP(S) Requests*
- *Web + APIs*
- *httr with REST*
- *JSON*
- *Resources*

James Balamuta
STAT 385 @ UIUC

Announcements

- **hw07** is due **Friday, Nov 2nd, 2018** at **6:00 PM**
- **Group project** member choice phase **has begun!**
  - https://github.com/stat385-fa2018/disc/issues/75
- **Quiz 10** covers Week 9 contents @ CBTF.
  - Window: Oct 30th - Nov 1st
  - Sign up: https://cbtf.engr.illinois.edu/sched
- Want to review your homework or quiz grades?
  Schedule an appointment.

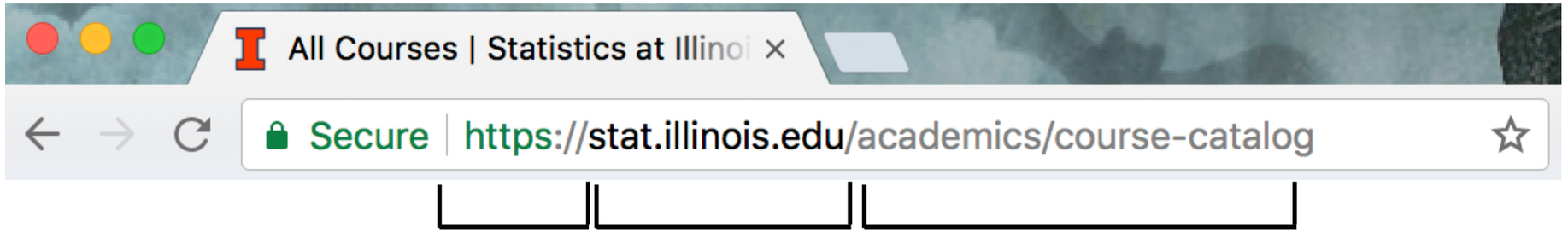
Lecture Objectives

- **Summarize** how an HTTP/HTTPS request is formed
- **Communicate** with a Web API
- **Understand** the semi-structured form of JSON
- **Extract** data from Web API Response

HTTP(S) Requests

HTTP(S) Request

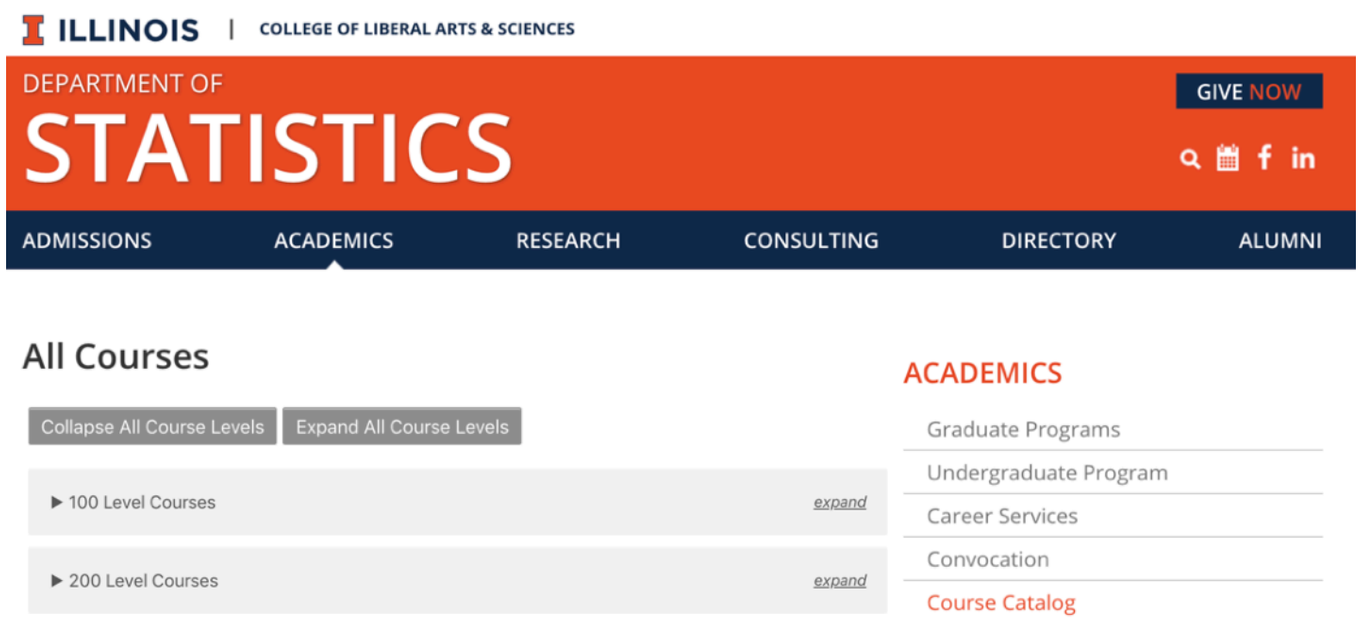
… **H**yper**T**ext **T**ransfer **P**rotocol (**S**ecure) …



Protocol/Scheme
Communication medium for request

Domain
Web Address

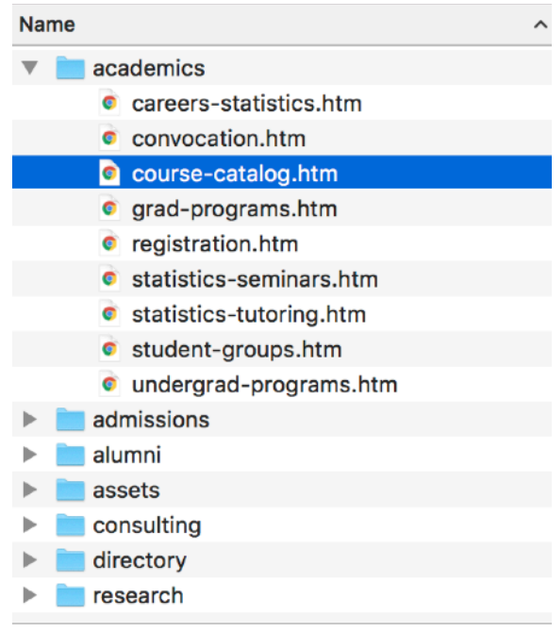
Path
Location of Resource

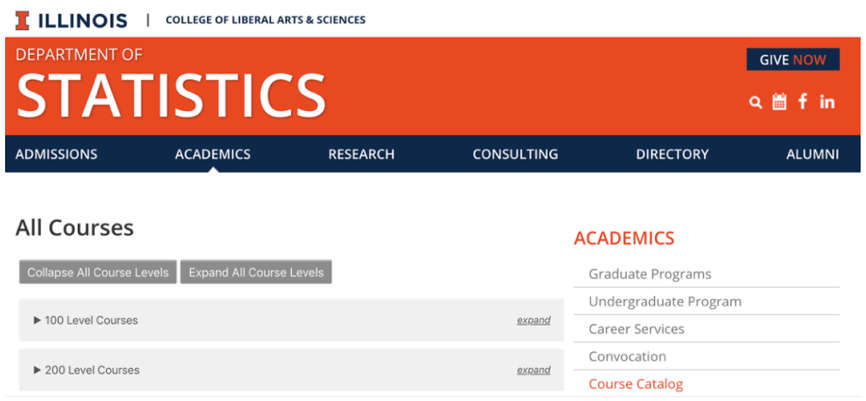


Hello stat.illinois.edu,
I'd like to see the
course catalog

Hello person's web
browser, let me pull
that up...

Source





Hello stat.illinois.edu,
thank you!

Hello person's web
browser, I found it.
Here you go!

URI*

Uniform **R**esource **I**dentifier

Source

---

* **URL**s (**U**niform **R**esource **L**ocators) or web links are actual a subset of **URI**s. They capture the **scheme** required to access the resource.

Downloading a File

… retrieving a file using a web browser …

Student Enrollment by Curriculum and Class Level

This report uses the final enrollment statistics from the official tenth day of on-campus terms. (Preliminary enrollment figures may be posted before the tenth day.) Reports for terms before Fall 2004 are sorted by college and curriculum code, with degree-granting and advising departments listed for each curriculum. Beginning in Fall 2004, reports are sorted by college, department code, and major code. Only students taking at least one on-campus, credit-bearing class are included in these reports. The following categories of students are excluded: auditors (students taking **only** non-credit classes); students taking **only** extramural or off-campus classes; Medical Scholars taking **no** on-campus, non-MSP classes. (Note: Illini Center MBA students **are** included in these statistics.)

NOTE: As of Spring 2017, these enrollment reports no longer separate students into "on-campus" and "extramural" categories. All students enrolled in classes on the 10th day are reported.

Student system	Fall - HTML	Spring - HTML	Summer 1 - HTML	Summer 2 - HTML	Fall - Excel	Spring - Excel	Summer - Excel
Banner: All Students	Fall, 2017	Spring, 2018 NEW Spring, 2017	Summer, 2017		Fall, 2017		
Banner: On-Campus Students	Fall, 2016 Fall, 2015 Fall, 2014 Fall, 2013	Spring, 2016 Spring, 2015 Spring, 2014 Spring, 2013	Summer, 2016 Summer, 2015 Summer, 2014 Summer, 2013	Starting Summer, 2005, summer 1 and summer 2 terms were consolidated	Fall, 2016 Fall, 2015 Fall, 2014 Fall, 2013		

Website: http://www.dmi.illinois.edu/stuenr/index.htm#class

CSV File: http://www.dmi.illinois.edu/stuenr/class/enrfa17.xls

Downloading a File

… retrieving a file online using R …

Uniform **R**esource **L**ocator
Website link to the file

Save Destination
Place to store file on computer

```
download.file(url = <address>, destfile = <save-to>)
```

Downloading a File

… retrieving a file online from R …

```
# URL of file to retrieve
url = "http://www.dmi.illinois.edu/stuenr/class/enrfa17.xls"

# Save this file as...
destfile = "enrfa17.xls"

# Download the file
download.file(url = url, destfile = destfile)

# Read the file into R
enrollfa17 = readxl::read_excel(x = destfile)
```

How can we:

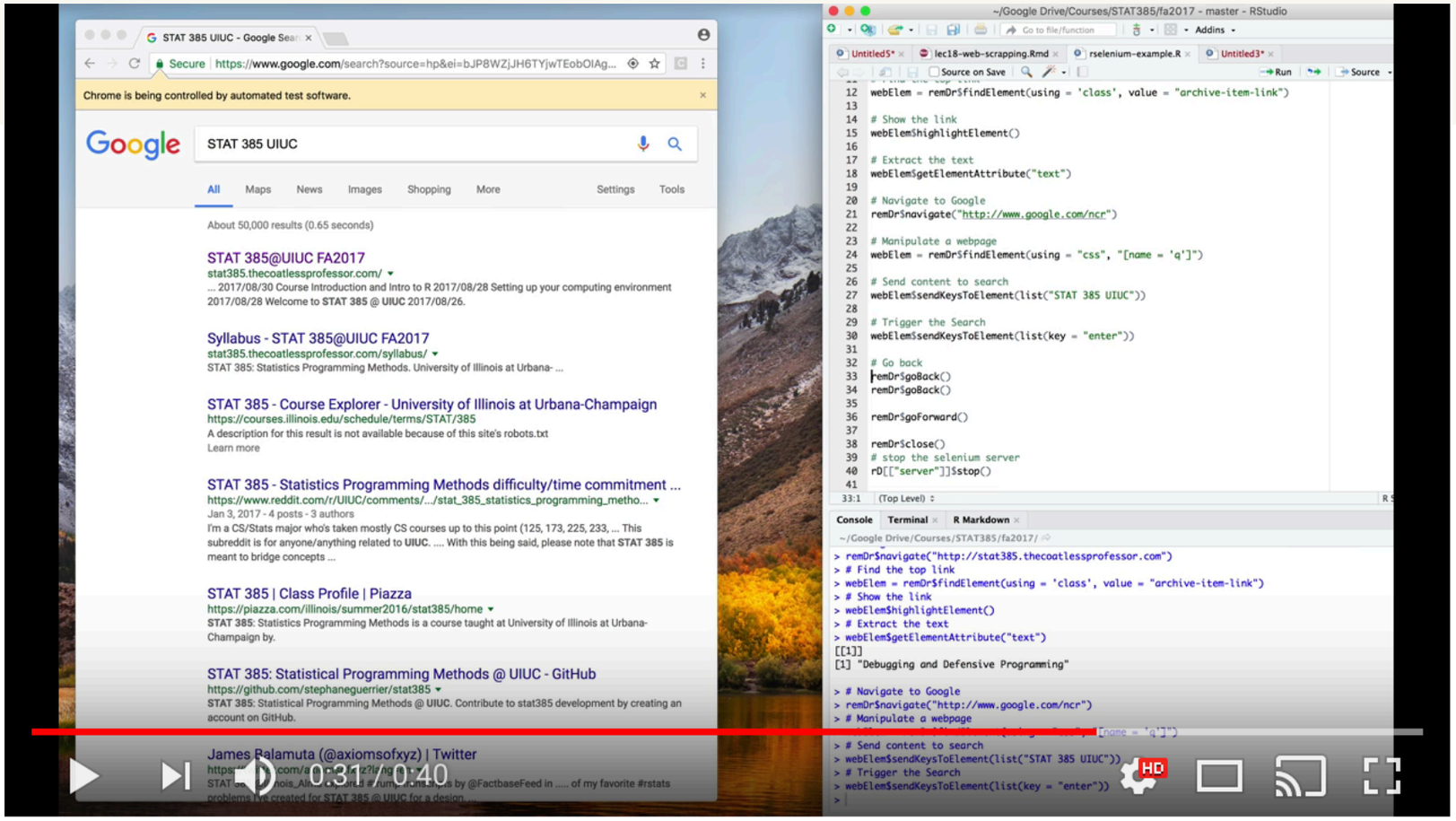
1. **obtain a list of files**; and
2. **download them automatically**?

More on this next time when we cover web scraping …

Web + APIs

Definition:

Web scraping (*web harvesting* or *web data extraction*) is a computer software technique of extracting information from websites. Source



https://www.youtube.com/watch?v=Vt6f8A35-1w

Why not just Web Scrape?

… reasons to not scrape …

1. **Websites change…**
   Scraping is not robust.

2. **Legality of scraping…**
   Check the Terms of Service (TOS)

3. **Lack of Documentation**
   Not as structured as an API

Definition:
Application programming interfaces (APIs) describe different ways to interact with software.

Source

Interface

Function Name
Actual name of the function that can be called e.g. add(1, 2)

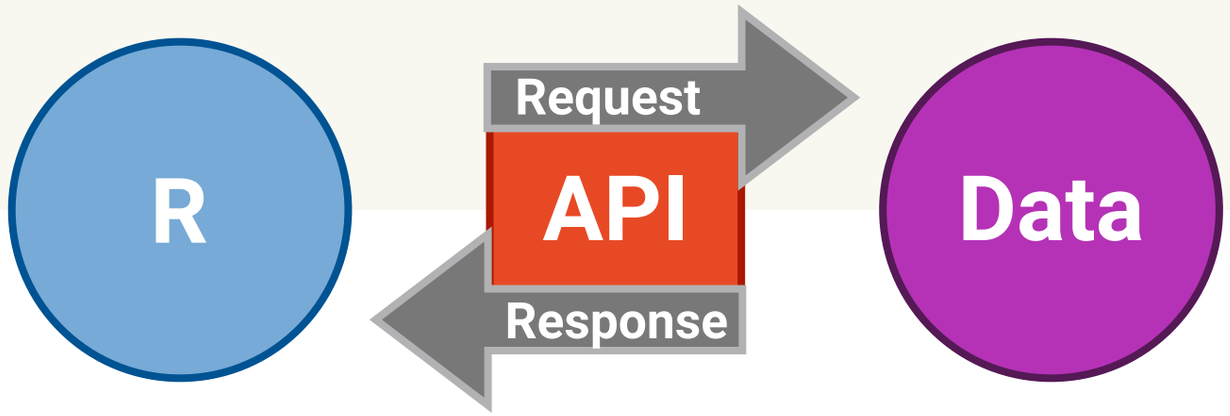
Parameters
Variables that receive expressions that can be used in the function's body

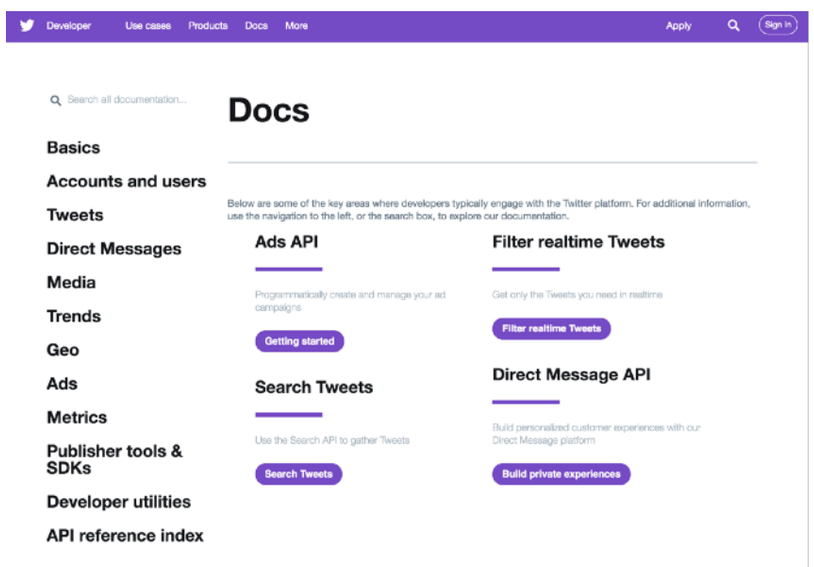
```
add = function(a, b) {
  summed = a + b
  return(summed)
}
```

Body
Statements in between {} that are run when the function is called

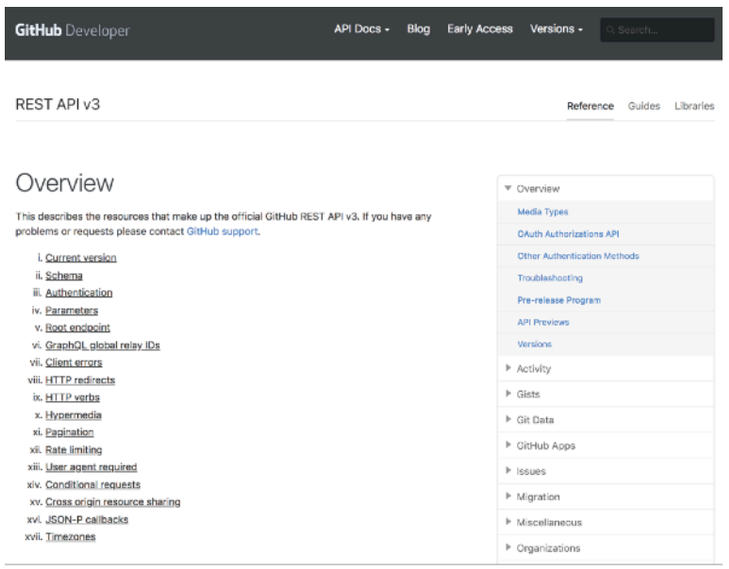
Return Value
Result made available from running the body statements

Definition:
Web APIs refer to an *Application Programming Interface (API)* for a *web server* or *web browser*.





https://developer.twitter.com/en/docs



https://developer.github.com/v3/

Definition:
Endpoints indicate where resources are on the server that can be requested by users.

https://api.website.com/v1/**resource**/

first API version

Endpoint

URI for APIs

… setting up ways to connect …

Base URI

https://api.github.com

Variables

+ **Endpoint**

/user/repos
/repos/**:**owner/**:**repo
/orgs/**:**org/repos

https://developer.github.com/v3/repos/#repositories

=

https://api.github.com**/user/repos**
https://api.github.com**/repos/:owner/:repo**
https://api.github.com**/orgs/:org/repos**

httr with REST

REST

Representation **S**tate **T**ransfer

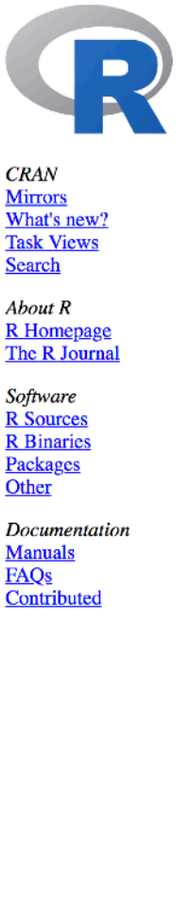
defines how networked resources are accessed

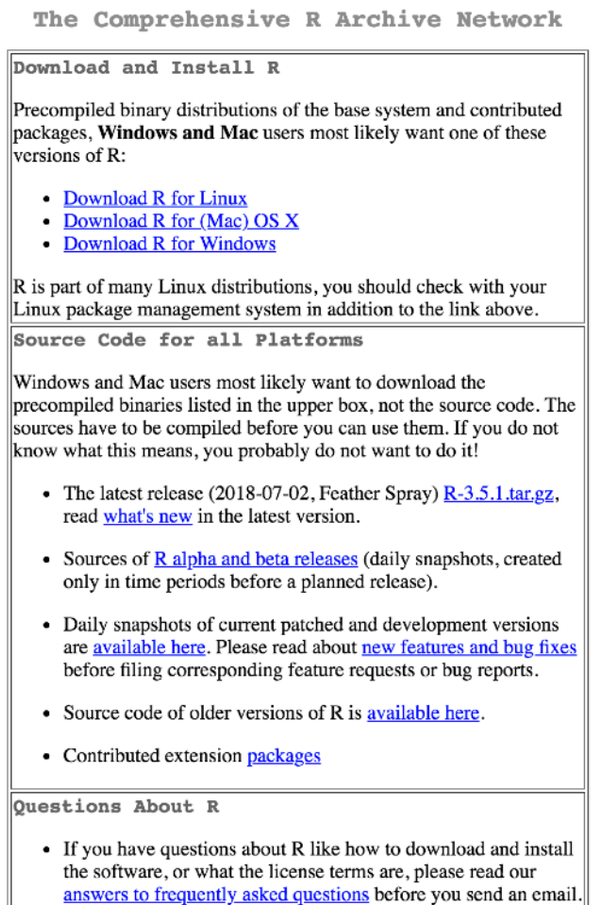
HTTP Information

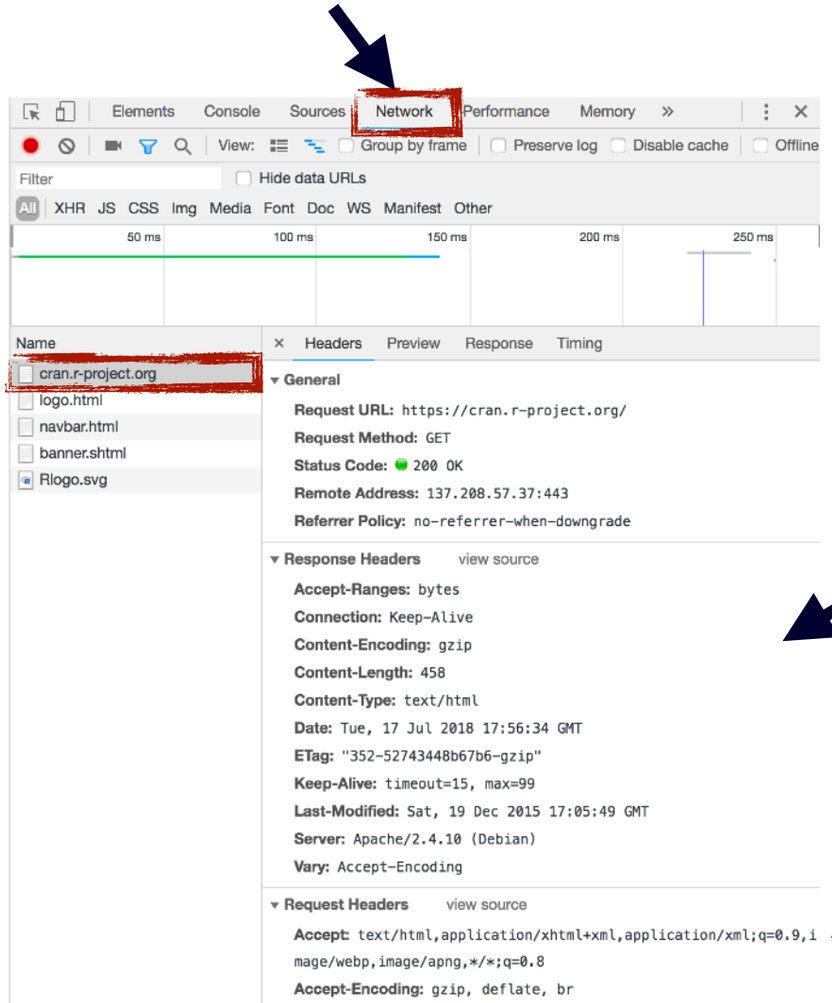
… when visiting a website …

Open Chrome's Web Developer Tools

- Windows: **Ctrl** + **Shift** + **I**
- macOS: **Command** + **Option** + **I**





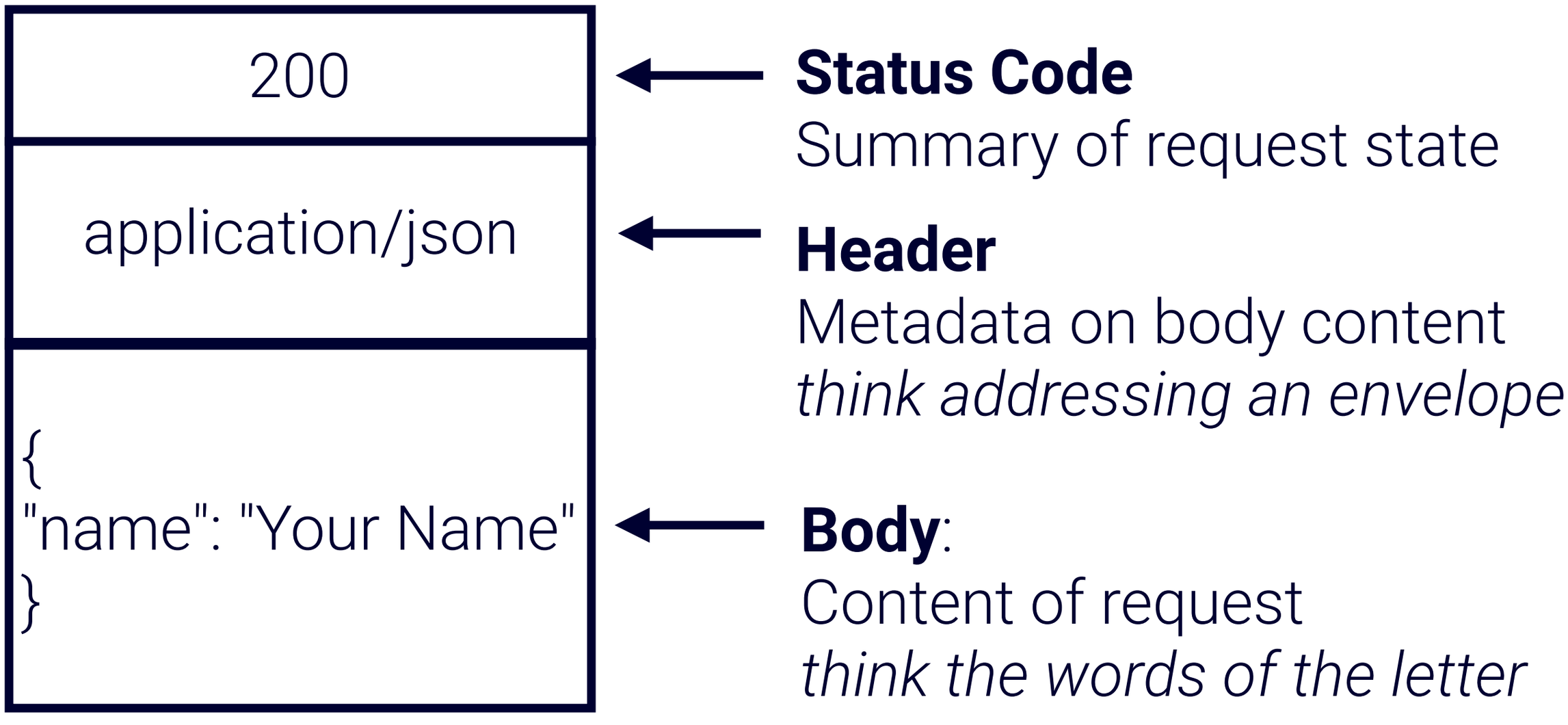


Response

Request

HTTP Response

… breaking down the 3 components …



HTTP Verbs

… navigating APIs …

Verb	Description
POST	*C*reate a new resource given data
GET	*R*etrieve a resource from collection.
PUT	*U*pdate an existing resource with new data.
DELETE	*D*elete a resource from collection.

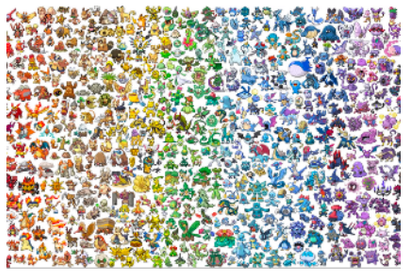
* These verbs are collectively known as "*CRUD*"

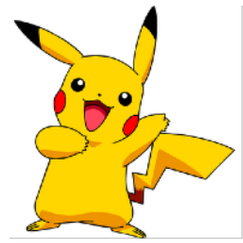
** There are additional verbs that are less frequently used:
CONNECT, TRACE, OPTIONS, HEAD, PATCH

API and Verbs

… how an API relates to HTTP Verbiage…

Verb	Path	Description
GET	/v2/pokemon/	Retrieve all Pokemon
GET	/v2/pokemon/{id}	Retrieve a specific Pokemon
POST	/v2/pokemon/	Add a new pokemon
PUT	/v2/pokemon/{id}	Update an existing Pokemon
DELETE	/v2/pokemon/{id}	Delete an existing Pokemon





httr

… making *HTTP* requests in *R* …


```
# install.packages("httr")
library("httr")

# Form a GET request to obtain the
STAT department website
web_page =
    GET("https://stat.illinois.edu/")
# Response [https://stat.illinois.edu/]
#   Date: 2018-07-10 18:37
#   Status: 200
#   Content-Type: text/html;
charset=UTF-8
#   Size: 60.3 kB
# …

# Check status of HTTP GET
request
status_code(web_page)
# [1] 200
```

A status of **200** means everything went smoothly

HTTP Status Code Breakdown

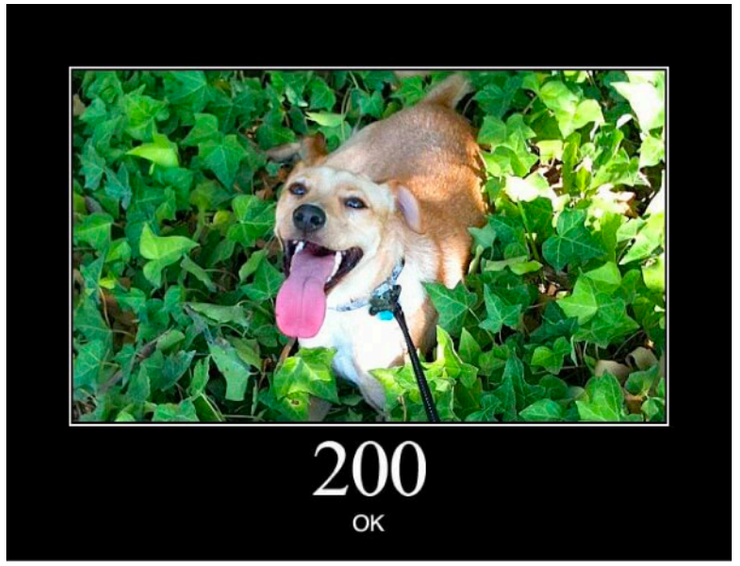
… classes of the status codes …

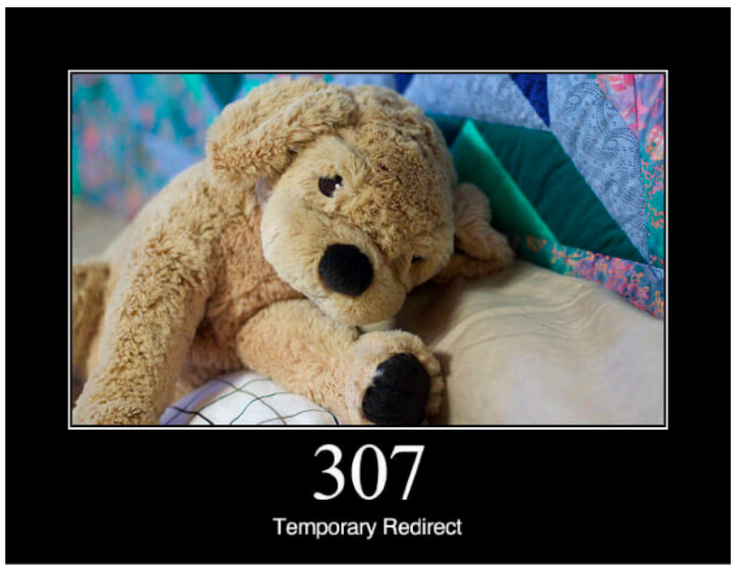
Code	Meaning
1xx	Informational
2xx	Successful
3xx	Redirection
4xx	Client Error
5xx	Server Error

---

* Status codes may *not* always be correct. May need info in header.
** Breakdown of Common Status Codes

common HTTP STATUS CODES





https://httpstatusdogs.com/

Supplying Query Terms

… writing a query term …

https://www.google.com/search**?q=uiuc+stat+department&sourceid=chrome**

Separator

Term Encoded

Delimiter

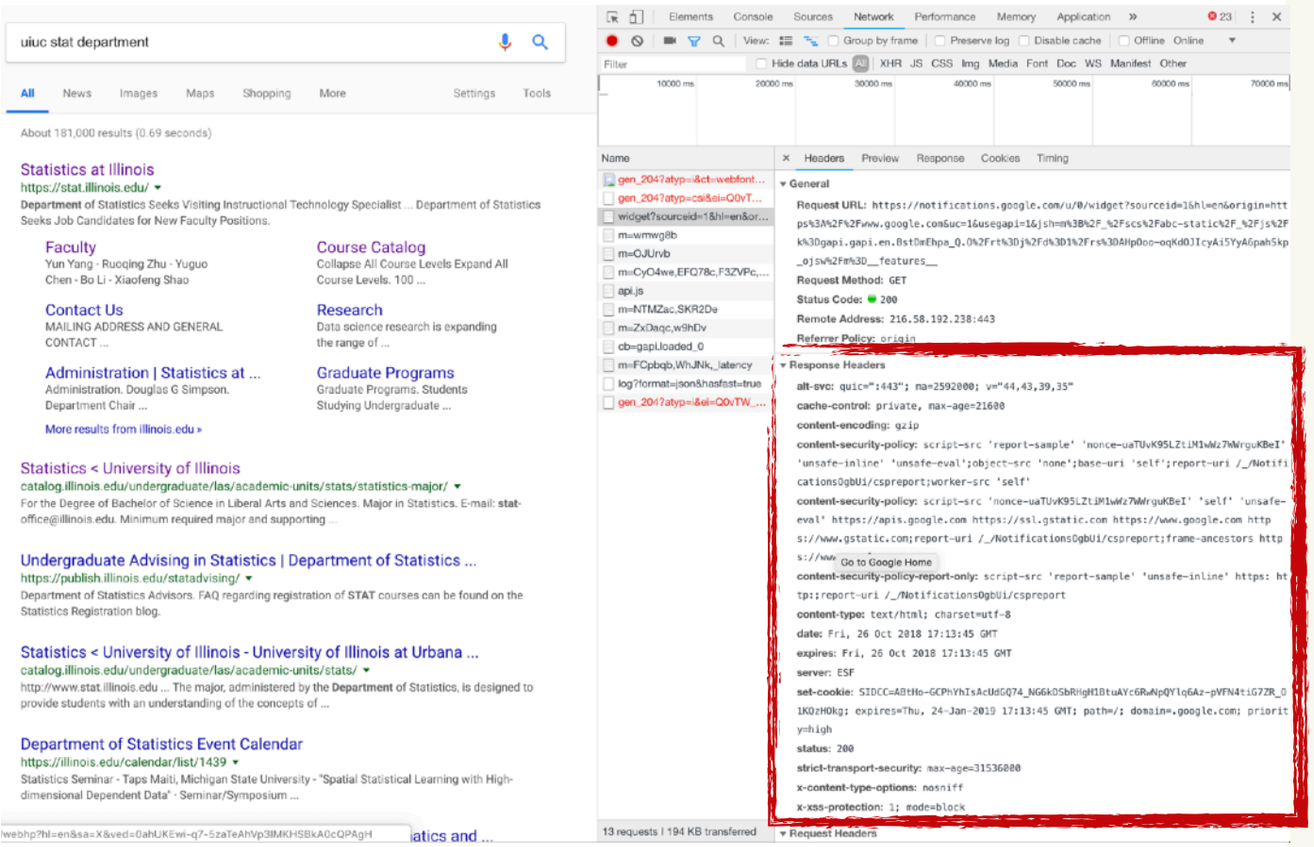
More in-depth formatting: https://en.wikipedia.org/wiki/Query_string

```
# Provide a query term
query_terms = list(q = "uiuc stat department")

# Form a GET request to search google
google_search = GET("https://www.google.com/search",
                    query = query_terms)
```

Retrieving Header Contents

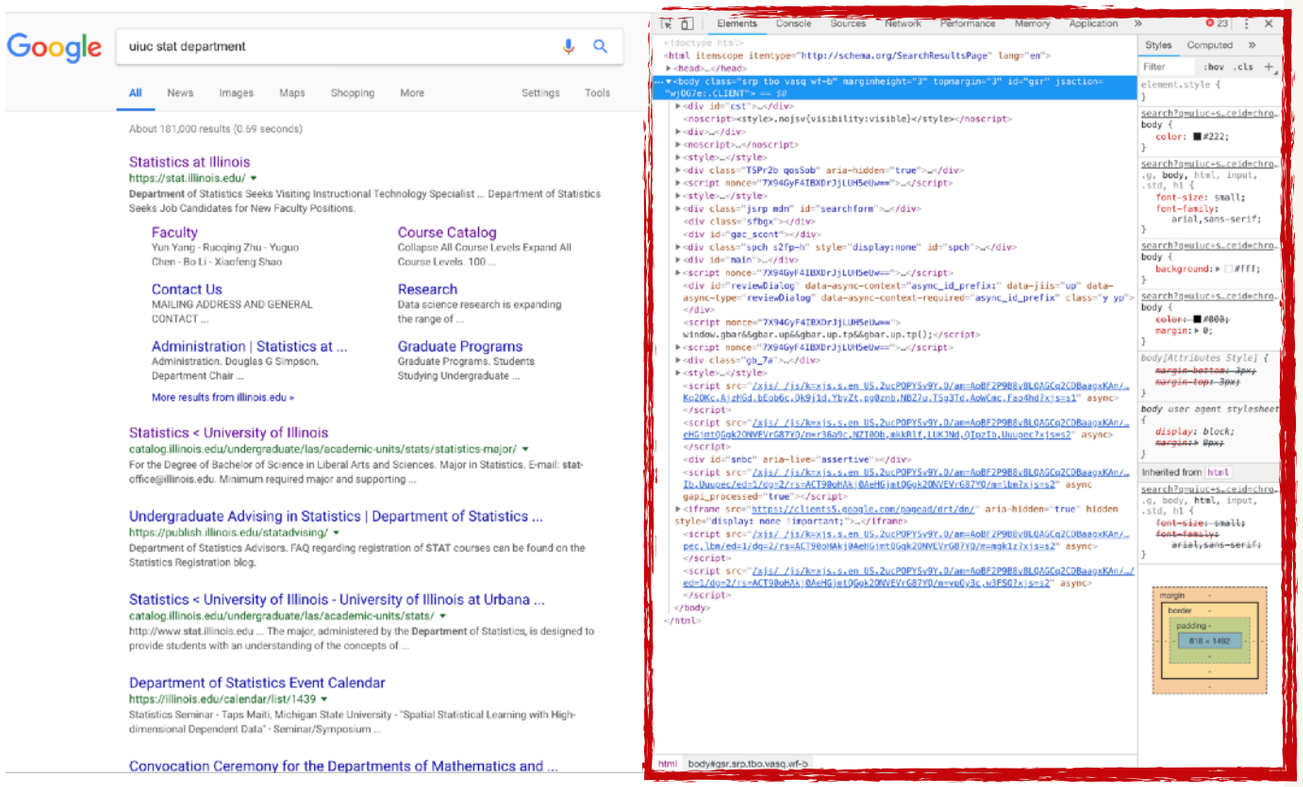
… metadata on the request …




```
# Retrieve the header
headers(google_search)
# $date
# [1] "Tue, 10 Jul 2018 12:17:39
GMT"
# $expires
# [1] "-1"
# $`cache-control`
# [1] "private, max-age=0"
# $`content-type`
# [1] "text/html;
charset=ISO-8859-1"
# …
```

Retrieving Body Contents

… grabbing web data …




```
# Retrieve body of request as text
content(google_search, "text")
# [1] "<!doctype html><html
itemscope=\"\" itemtype=\"http://
schema.org/SearchResultsPage\"
lang=\"en\"><head><meta
content=\"text/html; charset=UTF-8\"
http-equiv=\"Content-Type\"><meta
content=\"/images/branding/googleg/
1x/googleg_standard_color_128dp.png\"
itemprop=\"image\"><link href=\"/
images/branding/product/ico/
googleg_lodp.ico\" rel=\"shortcut
icon\"><title>uiuc stat department -
Google Search</title>

# Default content is either:
# xml document or raw vector
content(google_search)
# {xml_document}
# <html itemtype="http://schema.org/
SearchResultsPage" lang="en">
# …
```

Encoding/Decoding Special Characters

Decoded	Encoded
space	%20
!	%21
"	%22
#	%23
$	%24
%	%25
&	%26
	%27
(	%28
)	%29
*	%2A
+	%2B
,	%2C
-	%2D
.	%2E
/	%2F
[	%5B
\	%5C
]	%5D
^	%5E
<	%3C
=	%3D
>	%3E
?	%3F
@	%40

Source

```
# Encoding a URL
URLencode("https://www.google.com/search?q=URL encoded")
# [1] "https://www.google.com/search?q=URL%20encoded"

# Decoding the URL
URLdecode("https://www.google.com/search?q=URL%20encoded")
# [1] "https://www.google.com/search?q=URL encoded"

# Encoding Special Characters
URLencode("+ * - \ ^ # $ % &")
# [1] "+%20*%20-%20%20%5E%20#%20$%20%25%20&"

# Decode Special Characters
URLdecode(
    URLencode("+ * - \ ^ # $ % &")
)
# [1] "+ * -  ^ # $ % &"
```

Your Turn

Send a **GET** request to https://httpbin.org/get

Check:

Status

Header

Body

What happens if a **POST** request is sent to https://httpbin.org/post?

JSON

Previously

Structures of Data

… how data is shaped …

Structured*

Rectangular

~5 - 10%

Semistructured**

key: value

~5 - 10%

```
---
title: "Untitled"
author: "JJB"
date: "1/27/2018"
output: html_document
---
```

Unstructured***

?????????

~80 - 90%

Pinky said,
"Gee, Brain. What are we going to do tonight?"
The Brain replied, "The same thing we do every night, Pinky. Try to take over the world."

---

* Typical form for scientific experiments and company databases
** *RMarkdown* Document Properties (YAML), JavaScript Object Notation (JSON), XML
*** Pure text documents, images, social media posts, and so on. No visible relationship.

Previously

What kind of data is this?

```
{
    "id": "file",
    "value": "File Menu",
    "showuser": true,
    "dropdown": {
        "menuitem": [
            {"label": "New Document", "onclick": "create_document()"},
            {"label": "Save Document", "onclick": "save_document()"},
            {"label": "Open Document", "onclick": "open_document()"},
            {"label": "Close Document", "onclick": "close_document()"}
        ]
    }
}
```

JSON

… **J**ava**S**cript **O**bject **N**otation …

most common response type from a Web API

```
{
    "name": "Your Name",
    "occupation": "Student",
    "eyecolor": "hazel",
    "salary": -14000,
    "starbucks": true,
    "music": {
        "band": "Johnny Cash",
        "song": "Ring of Fire"
    },
    "required": ["name", "occupation"]
}
```

Behind JSON

… **semi-structure** format …

Curly Braces

Colon separator

Key

Value

List

Vector

JSON

```
{
   "name": "Your Name",
   "occupation": "Student",
   "eyecolor": "hazel",
   "salary": -14000,
   "starbucks": true,
   "music": {
       "band": "Johnny Cash",
       "song": "Ring of Fire"
   },
   "required": ["name", "occupation"]
}
```

R

```
my_data = list(
   "name" = "Your Name",
   "occupation" = "Student",
   "eyecolor" = "hazel",
   "salary" = -14000,
   "starbucks" = true,
   "music" = list(
       "band" = "Johnny Cash",
       "song" = "Ring of Fire"
   ),
   "required" = c("name", "occupation")
)
```

```
# install.packages("jsonlite")
library("jsonlite")

fromJSON('{
"name": "Your Name",
"occupation": "Student",
"eyecolor": "hazel",
"salary": -14000,
"starbucks": true,
"music": {
      "band": "Johnny Cash",
       "song": "Ring of Fire"
       },
"required": ["name", "occupation"]
}')
```

Parsing JSON

Querying GitHub's API

... connecting to GitHub ...

```
# Specify base URL
base_url = "https://api.github.com"

# Specify user
gh_user = "tidyverse"

# Create a request for information
endpoint_resource = paste0("/users/", gh_user ,"/repos")

# Make a url
url = paste0(base_url, endpoint_resource)

# Form a GET request to obtain a list of repos on GitHub
gh_get_repos = GET(url)
```

Manipulating Results

… viewing data in an API response …

```
# Acquire results in a list
gh_list_repos = content(gh_get_repos)
# May generate a list OR a data frame
gh_data_repos = fromJSON(content(gh_get_repos, "text"))

# Check underlying structure
str(gh_data_repos)
# 'data.frame': 28 obs. of  72 variables:
# $ id              : int  72109608 23932217 86504302…
# …

# Directly verify if a data frame
is.data.frame(gh_data_repos)
# [1] TRUE
```

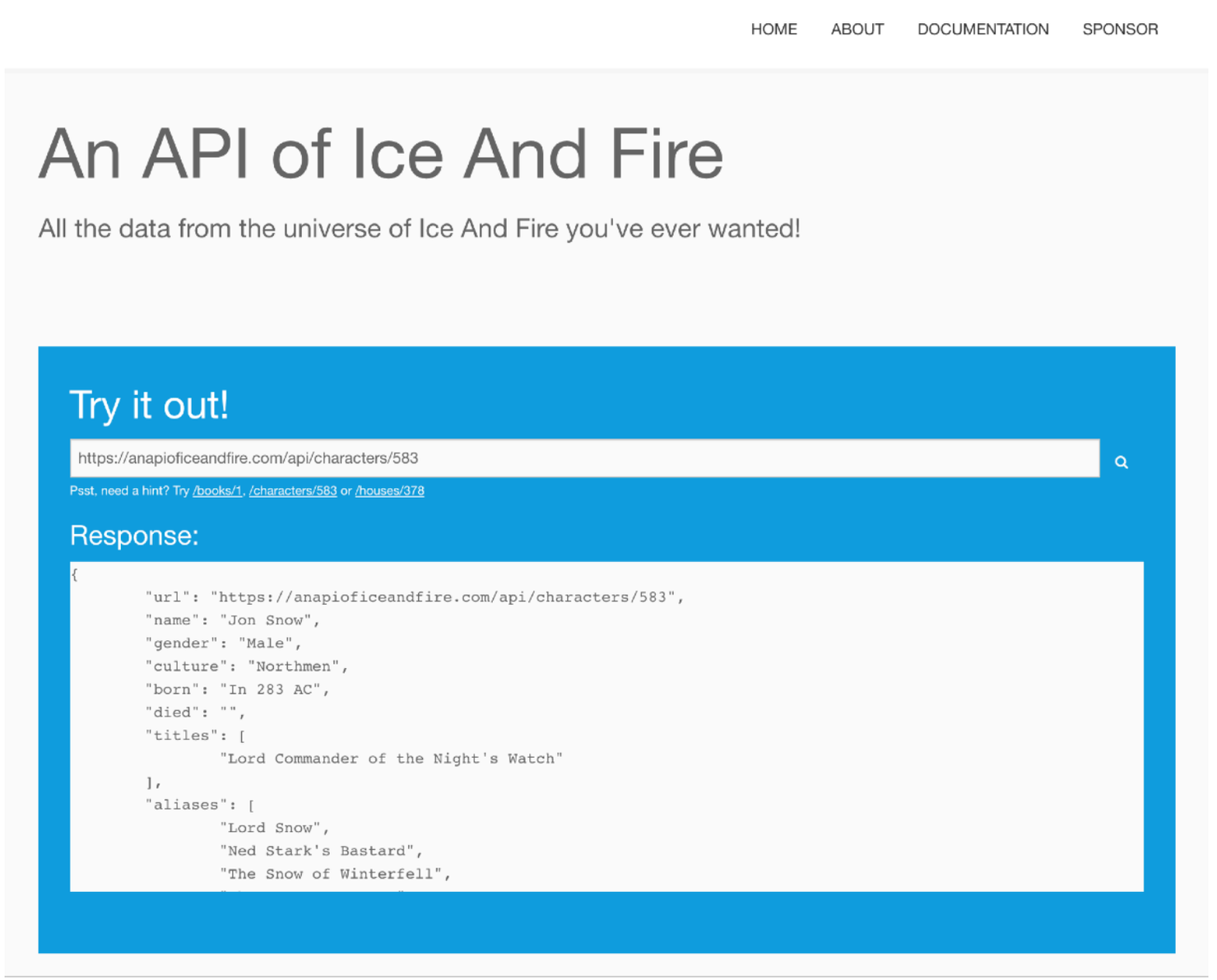
Your Turn

Retrieve the tag information for the **RcppCore/Rcpp** repository based on the details provided in:

https://developer.github.com/v3/repos/#list-tags

Your Turn

Obtain the list of houses from the Ice and Fire API



https://anapioficeandfire.com/Documentation#houses

Resources

Public APIs

… APIs for all !!!

README.md

Public APIs build passing

A collective list of free APIs for use in web development.

A public API for this project can be found here - thanks to DigitalOcean for helping us provide this service!

For information on contributing to this project, please see the contributing guide.

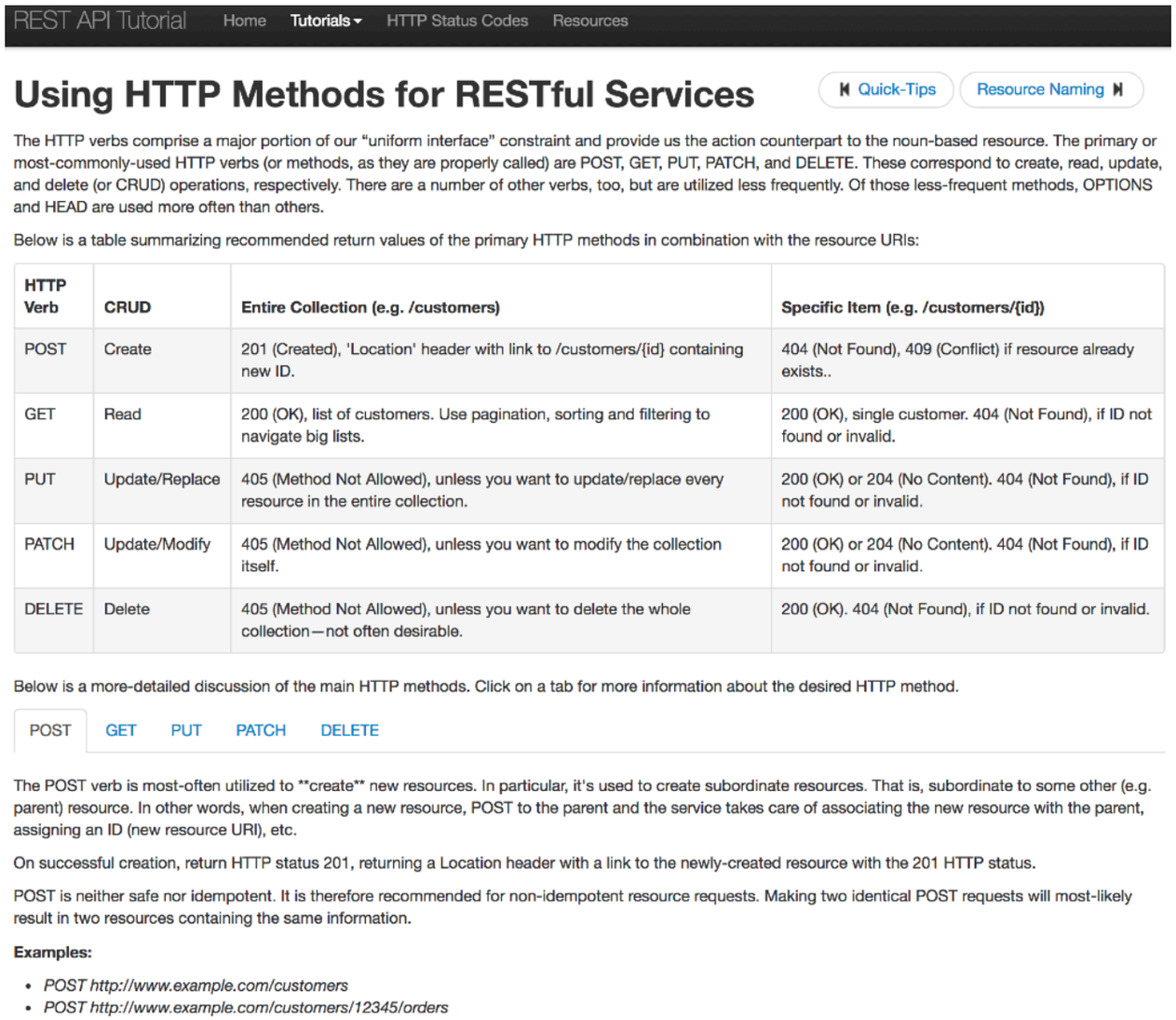
Please note a passing build status indicates all listed APIs are available since the last update. A failing build status indicates that 1 or more services may be unavailable at the moment.

Index

https://github.com/toddmotto/public-apis

REST API Tutorial

… Crux of the API Request …



REST API Tutorial Home Tutorials HTTP Status Codes Resources

Using HTTP Methods for RESTful Services

Quick-Tips Resource Naming

The HTTP verbs comprise a major portion of our "uniform interface" constraint and provide us the action counterpart to the noun-based resource. The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently. Of those less-frequent methods, OPTIONS and HEAD are used more often than others.

Below is a table summarizing recommended return values of the primary HTTP methods in combination with the resource URIs:

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Below is a more-detailed discussion of the main HTTP methods. Click on a tab for more information about the desired HTTP method.

POST GET PUT PATCH DELETE

The POST verb is most-often utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc.

On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests. Making two identical POST requests will most-likely result in two resources containing the same information.

Examples:

- *POST http://www.example.com/customers*
- *POST http://www.example.com/customers/12345/orders*

http://www.restapitutorial.com/lessons/httpmethods.html

Recap

- **HTTP(S)**
  - Communicating over the internet.
- **Web + APIs**
  - APIs provide structured interaction with software.
  - Web APIs are more stable than web pages
- **httr with REST**
  - Connect with a REST API using HTTP Verbs
  - Formatting of HTTP Response is: Status, Header, Body
- **JSON**
  - Semi-structured output based on key-value form
  - Common Web API response that is converted to an *R* list