Extracting Data from the Web Part 1: **APIs**

Garrett Grolemund

Data Scientist, Educator

November 2016

1. **APIs** (Today)
    1. What is an API?
    2. The role of HTTP
    3. `httr` package
2. **Web Scraping** (November 30)

httr QuickStart Vignette

1500 words, <10 minutes to read

cran.r-project.org/web/packages/httr/vignettes/quickstart.html

Extracting Data from the Web

Scott Chamberlain, Karthik Ram, Me

github.com/ropensci/user2016-tutorial

APIs

What is an API?

Instructions for how a program should interact with a piece of software.

Can be an interface to a:

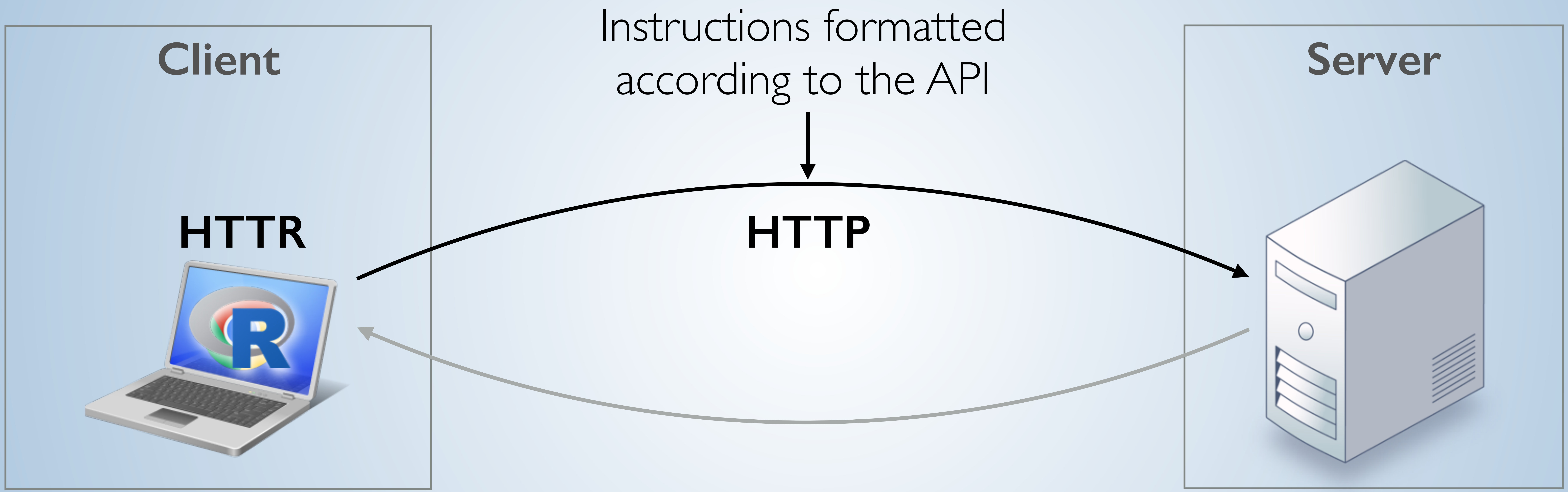
Database

OS

Software package

Web Application

API Architecture



HTTP

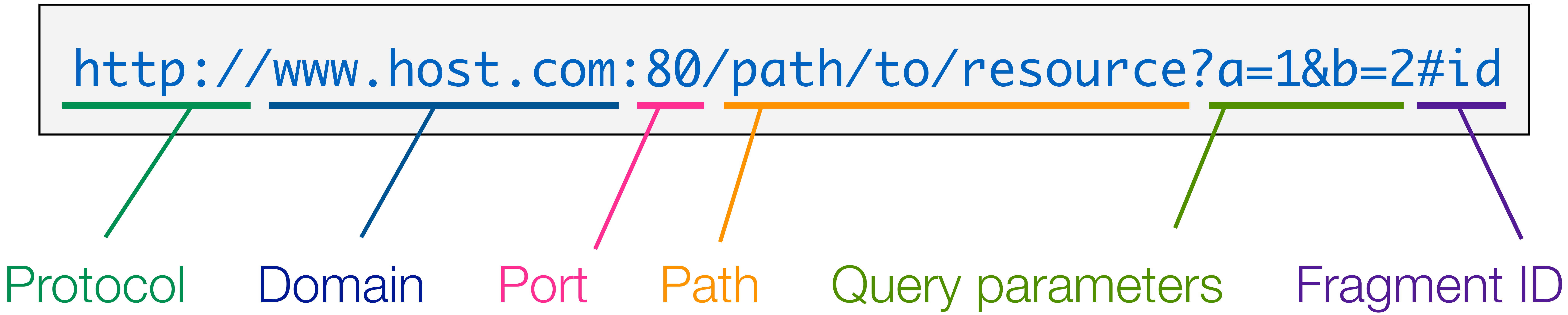
HTTP

Hyper**T**ext **T**ransfer **P**rotocol

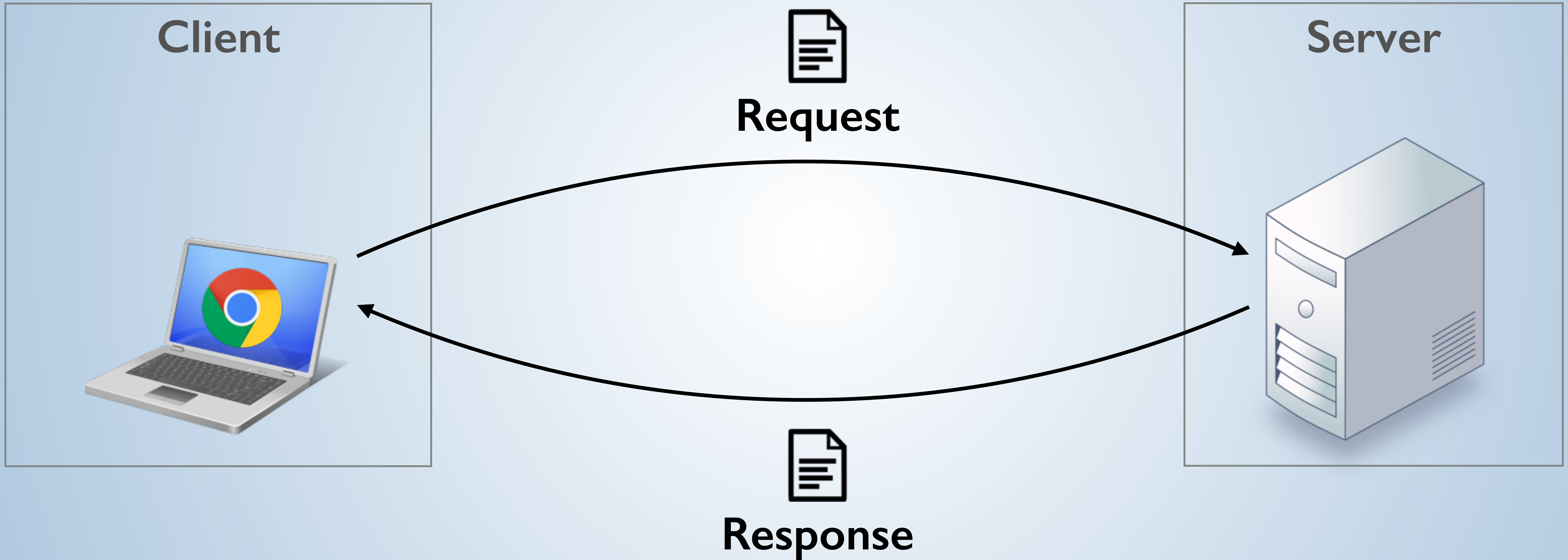
URL

Uniform **R**esource **L**ocator

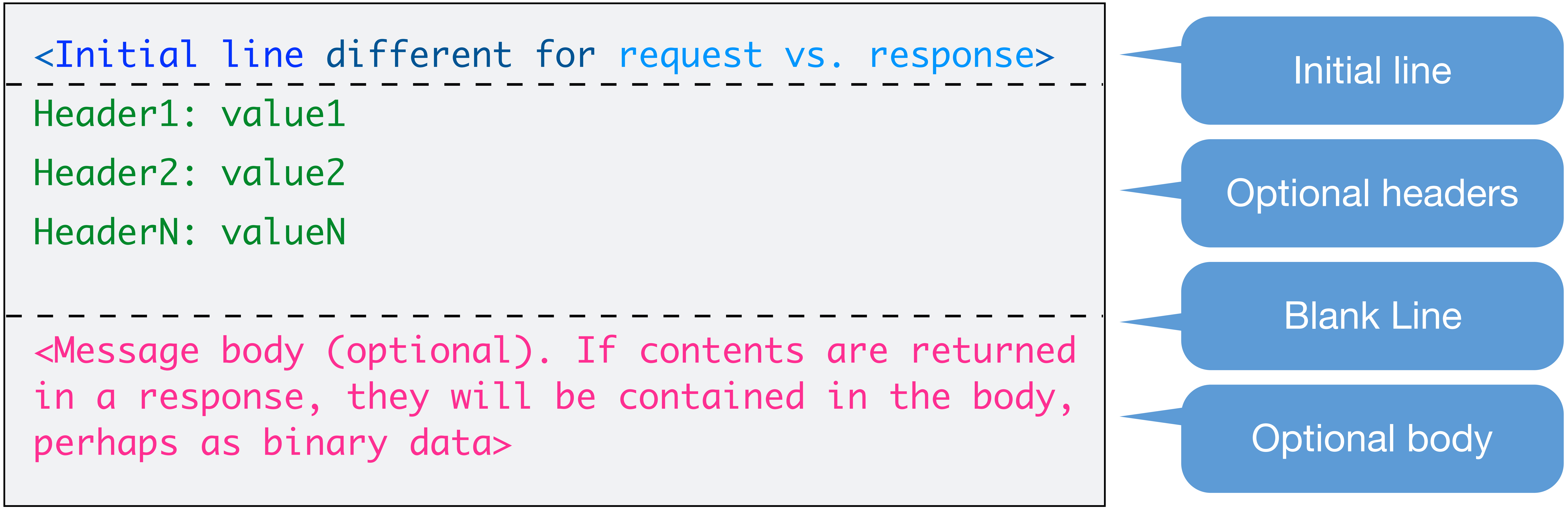
URL


http://www.host.com:80/path/to/resource?a=1&b=2#id
Protocol
Domain
Port
Path
Query parameters
Fragment ID

HTTP



HTTP message structure



```
<Initial line different for request vs. response>
Header1: value1
Header2: value2
HeaderN: valueN

<Message body (optional). If contents are returned
in a response, they will be contained in the body,
perhaps as binary data>
```

Sample Request

HTTP Verb

URL path

HTTP version

```
GET /resources/webinars/ HTTP/1.1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Host: www.rstudio.com
User-Agent: libcurl/7.43.0 r-curl/2.1 httr/1.2.1
Accept-Encoding: gzip, deflate
Accept: application/json, text/xml, application/xml, */*
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

HTTP Verbs

GET	**Retrieve** whatever is specified by the URL
POST	**Create** resource at URL with data in body
PUT	**Update** resource at URL with data in body
DELETE	**Delete** resource at URL

HTTR

httr Request

```r
library(httr)
r <- GET("http://httpbin.org/get")
```

Server response returned by function

HTTP Verb function (all CAPS)

Complete URL

headers

Add headers with **add_headers()**

Key

Value

```
r <- GET("http://httpbin.org/get",
             add_headers(Name = "Garrett"))
```

```
GET /get HTTP/1.1
Name: Garrett
```

body

Add content with the **body** and **encode** arguments.

```
url <- "http://httpbin.org/post"
body <- list(a = 1, b = 2, c = 3)
r <- POST(url, body = body, encode = "form")
r <- POST(url, body = body, encode = "multipart")
r <- POST(url, body = body, encode = "raw")
r <- POST(url, body = body, encode = "json")
```

content as list or file

How to encode (if list)

```
url <- "http://httpbin.org/post"
body <- list(a = 1, b = 2, c = 3)
r <- POST(url, body = body, encode = "json")
```

```
POST /post HTTP/1.1
Host: httpbin.org
User-Agent: libcurl/7.43.0 r-curl/2.1 httr/1.2.1
Accept-Encoding: gzip, deflate
Accept: application/json, text/xml, application/xml, */*
Content-Type: application/json
Content-Length: 19

{"a":1,"b":2,"c":3}
```

list contents as json

JSON

Java**S**cript **O**bject **N**otation

Collection of key: value pairs. Becoming standard data format for web APIs.

```
{
  "Title": "Frozen",
  "Year": "2013",
  "Rated": "PG",
  "Released": "27 Nov 2013"
}
```

jsonlite

Package for working with JSON data in R.

```r
library(jsonlite)
toJSON(list(a = 1, b = 2, c = 3))
# {"a":[1],"b":[2],"c":[3]}

fromJSON('{"a":[1],"b":[2],"c":[3]}')
# $a
# [1] 1
# $b
# [1] 2
# $c
# [1] 3
```

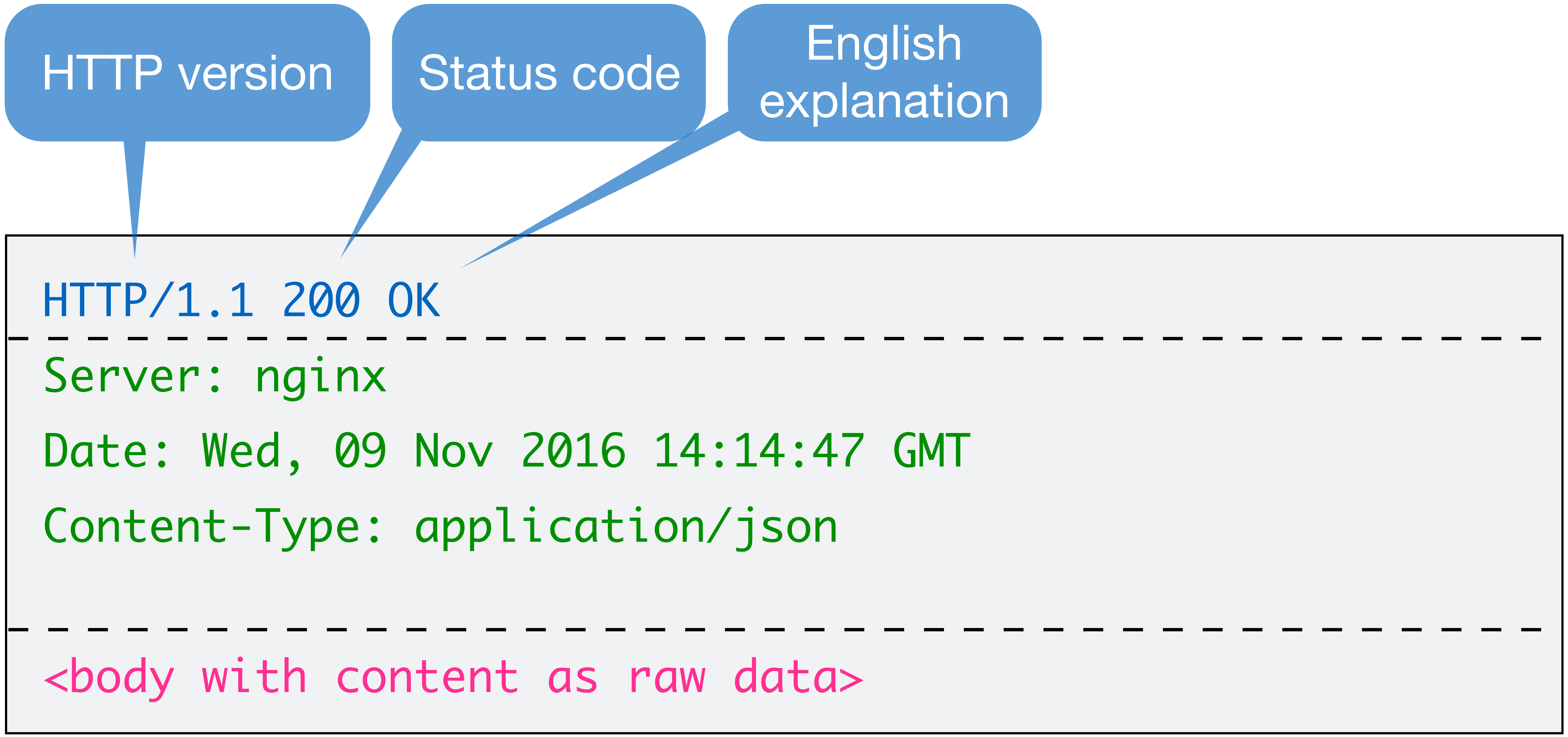
verbose()

To view the HTTP request that httr sends.

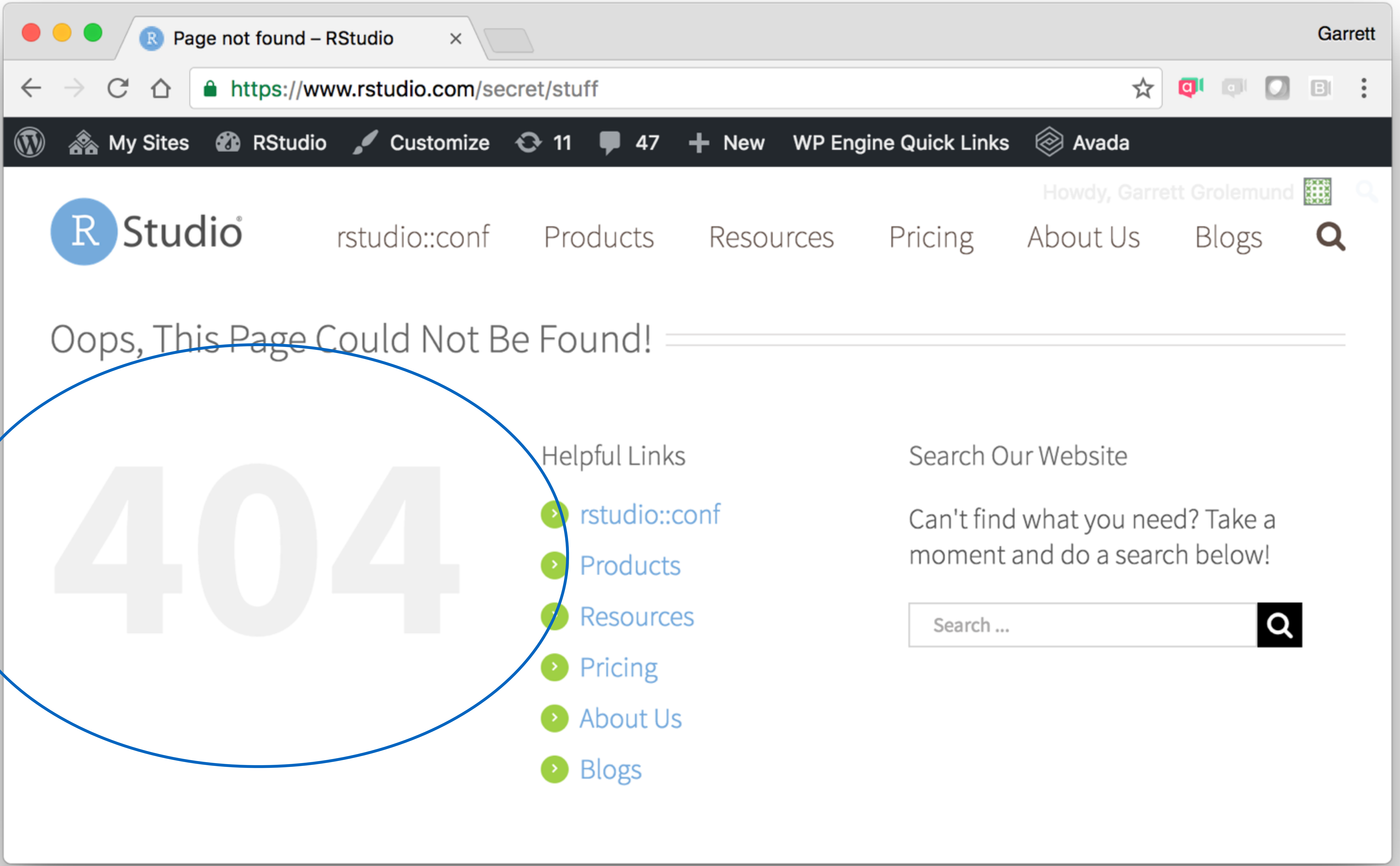
```
r <- GET("http://httpbin.org/get", verbose())
# -> GET /get HTTP/1.1
# -> Host: httpbin.org
# -> User-Agent: libcurl/7.43.0 r-curl/2.1 httr/1.2.1
# -> Accept-Encoding: gzip, deflate
# -> Accept: application/json, text/xml, application/
xml, */*
```

Responses

Sample Response



```
HTTP/1.1 200 OK
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Server: nginx
Date: Wed, 09 Nov 2016 14:14:47 GMT
Content-Type: application/json

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
<body with content as raw data>
```


Page not found – RStudio

Garrett

https://www.rstudio.com/secret/stuff

My Sites RStudio Customize 11 47 New WP Engine Quick Links Avada

Howdy, Garrett Grolemund

rstudio::conf Products Resources Pricing About Us Blogs

Oops, This Page Could Not Be Found!

404

Helpful Links

- rstudio::conf
- Products
- Resources
- Pricing
- About Us
- Blogs

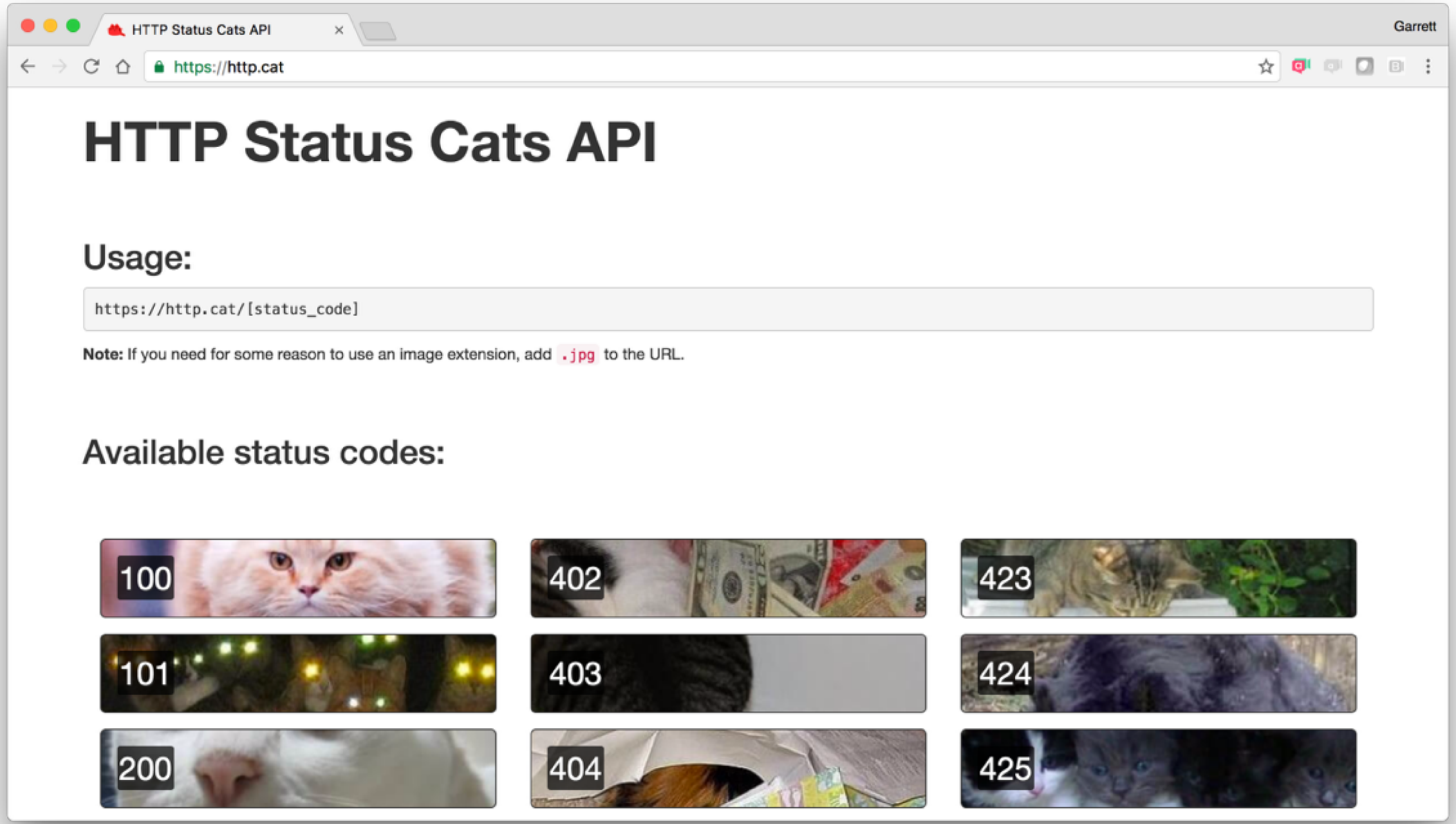
Search Our Website

Can't find what you need? Take a moment and do a search below!

Search ...

Status codes

Cute dictionary at http://http.cat



HTTP Status Cats API

Usage:

```
https://http.cat/[status_code]
```

Note: If you need for some reason to use an image extension, add `.jpg` to the URL.

Available status codes:

```
r <- GET("http://httpbin.org/get")
r
# Response [http://httpbin.org/get]
#   Date: 2016-11-09 14:30
#   Status: 200
#   Content-Type: application/json
#   Size: 295 B
# {
#   "args": {},
#   "headers": {
#     "Accept": "application/json, text/xml, applicat...
#     "Accept-Encoding": "gzip, deflate",
#     "Host": "httpbin.org",
#     "User-Agent": "libcurl/7.43.0 r-curl/2.1 httr/1...
#   },
#   "origin": "50.154.53.9",
#   "url": "http://httpbin.org/get"
# …
```

Status

Extract status with **$status_code** or **http_status()**

```
r$status_code
# 200
http_status(r)
# $category
# [1] "Success"
# $reason
# [1] "OK"
# $message
# [1] "Success: (200) OK"
```

Program defensively with **warn_for_status()** and **stop_for_status()**

```
r2 <- r
r2$status_code <- 404
warn_for_status(r2)
# Warning message:
# Not Found (HTTP 404).

stop_for_status(r2)
# Error: Not Found (HTTP 404).
```

Headers

Extract headers with **headers()**

```
headers(r)
# $server
# [1] "nginx"
# $date
# [1] "Wed, 09 Nov 2016 14:30:53 GMT"
# $`content-type`
# [1] "application/json"
# …
headers(r)$server
# [1] "nginx"
```

Content

Extract content from the body with **$content** or **content()**

```
r$content
#   [1] 7b 0a 20 20 22 61 72 67 73 22 3a 20 7b 7d 2c 20 0a 20 20 22 68 65 61 64
#  [25] 65 72 73 22 3a 20 7b 0a 20 20 20 20 22 41 63 63 65 70 74 22 3a 20 22 61
#  [49] 70 70 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 2c 20 74 65 78 74 2f 78 6d
#  [73] 6c 2c 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 2c 20 2a 2f 2a 22
#  [97] 2c 20 0a 20 20 20 20 22 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 22
# [121] 3a 20 22 67 7a 69 70 2c 20 64 65 66 6c 61 74 65 22 2c 20 0a 20 20 20 20
# [145] 22 48 6f 73 74 22 3a 20 22 68 74 74 70 62 69 6e 2e 6f 72 67 22 2c 20 0a
# [169] 20 20 20 20 22 55 73 65 72 2d 41 67 65 6e 74 22 3a 20 22 6c 69 62 63 75
# [193] 72 6c 2f 37 2e 34 33 2e 30 20 72 2d 63 75 72 6c 2f 32 2e 31 20 68 74 74
# [217] 72 2f 31 2e 32 2e 31 22 0a 20 20 7d 2c 20 0a 20 20 22 6f 72 69 67 69 6e
# [241] 22 3a 20 22 35 30 2e 31 35 34 2e 35 33 2e 39 22 2c 20 0a 20 20 22 75 72
# [265] 6c 22 3a 20 22 68 74 74 70 3a 2f 2f 68 74 74 70 62 69 6e 2e 6f 72 67 2f
# [289] 67 65 74 22 0a 7d 0a
```

Raw bytes

Content

Extract content from the body with **$content** or **content()**

```
content(r, "raw")
#   [1] 7b 0a 20 20 22 61 72 67 73 22 3a 20 7b 7d 2c 20 0a 20 20 22 68 65 61 64
#  [25] 65 72 73 22 3a 20 7b 0a 20 20 20 20 22 41 63 63 65 70 74 22 3a 20 22 61
#  [49] 70 70 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 2c 20 74 65 78 74 2f 78 6d
#  [73] 6c 2c 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 2c 20 2a 2f 2a 22
#  [97] 2c 20 0a 20 20 20 20 22 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 22
# [121] 3a 20 22 67 7a 69 70 2c 20 64 65 66 6c 61 74 65 22 2c 20 0a 20 20 20 20
# [145] 22 48 6f 73 74 22 3a 20 22 68 74 74 70 62 69 6e 2e 6f 72 67 22 2c 20 0a
# [169] 20 20 20 20 22 55 73 65 72 2d 41 67 65 6e 74 22 3a 20 22 6c 69 62 63 75
# [193] 72 6c 2f 37 2e 34 33 2e 30 20 72 2d 63 75 72 6c 2f 32 2e 31 20 68 74 74
# [217] 72 2f 31 2e 32 2e 31 22 0a 20 20 7d 2c 20 0a 20 20 22 6f 72 69 67 69 6e
# [241] 22 3a 20 22 35 30 2e 31 35 34 2e 35 33 2e 39 22 2c 20 0a 20 20 22 75 72
# [265] 6c 22 3a 20 22 68 74 74 70 3a 2f 2f 68 74 74 70 62 69 6e 2e 6f 72 67 2f
# [289] 67 65 74 22 0a 7d 0a
```

Raw bytes

Content

Extract content from the body with **$content** or **content()**

```
content(r, "text")
# No encoding supplied: defaulting to UTF-8.
# [1] "{\n  \"args\": {}, \n  \"headers\": {\n
\"Accept\": \"application/json, text/xml,
application/xml, */*\", \n    \"Accept-Encoding\":
\"gzip, deflate\", \n    \"Host\": \"httpbin.org\",
\n    \"User-Agent\": \"libcurl/7.43.0 r-curl/2.1
httr/1.2.1\"\n  }, \n  \"origin\": \"50.154.53.9\",
\n  \"url\": \"http://httpbin.org/get\"\n}\n"
```

select format

Content

Extract content from the body with **$content** or **content()**

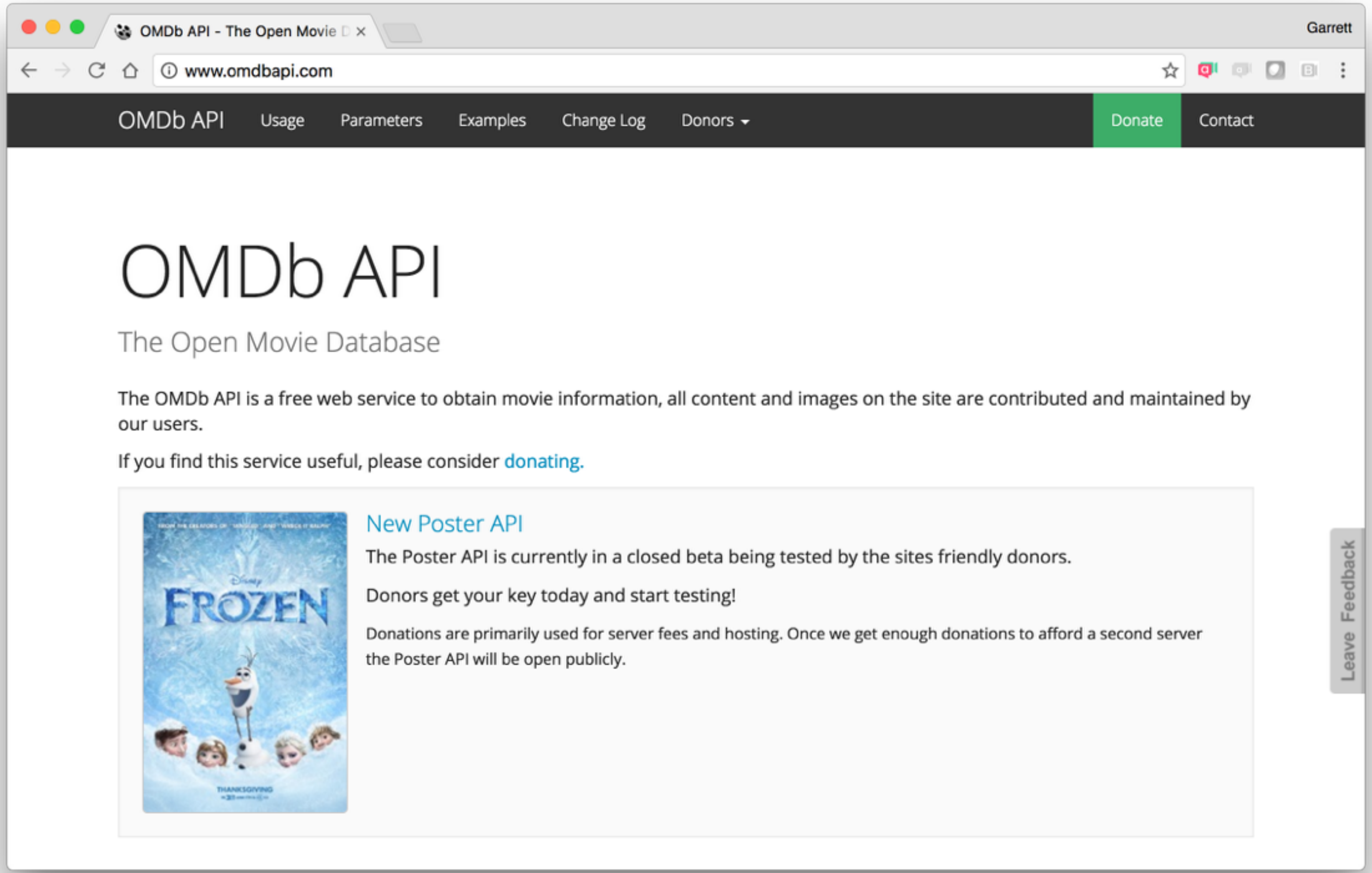
```
content(r, "parse")
# $args
# named list()

# $headers
# $headers$Accept
# [1] "application/json, text/xml, application/xml, */*"

# $headers$`Accept-Encoding`
# [1] "gzip, deflate"

# …
```

named list

Can parse: html, xml, csv, tsv, json, jpeg, png, forms

Example

www.omdbapi.com



OMDb API

The Open Movie Database

The OMDb API is a free web service to obtain movie information, all content and images on the site are contributed and maintained by our users.

If you find this service useful, please consider donating.

New Poster API

The Poster API is currently in a closed beta being tested by the sites friendly donors.

Donors get your key today and start testing!

Donations are primarily used for server fees and hosting. Once we get enough donations to afford a second server the Poster API will be open publicly.

Thank You