Description	Understanding what are HTTP requests and responses
Time	30 minutes
Learning Goals	• Understanding what are HTTP requests and responses

Lets try building an API endpoint to figure out what APIs are all about!

We previously mentioned most online APIs communicate via the HTTP protocol. But what does that mean???

The HTTP Protocol

The Internet consists of billions of computers that communicate to each other via ethernet cables/undersea cables/satellite uplinks.

HTTP is a communications standard to ensure all Internet communications are done in a specific format. Its basically the lingua franca of the Internet.

Without HTTP, every computer might communicate in a different way, e.g. some in Ruby, some in Python, some in PHP, some in Javascript…

HTTP is an *asynchronous request-response client-server protocol*. Heres a breakdown of what that means

1. client-server
    - The entire HTTP process is between two parties, a client and a server.
2. request-response
    - The client sends the request in a specific format to the server
    - After processing the request, the server sends a response in a specific format back to the client
3. asynchronous
    - The server can take as long as it wants to process the request
    - The server may choose NOT to send a response

Anatomy of a HTTP Request/Response

We've mentioned that HTTP requests/responses have a specific format several times already. Lets see an actual HTTP request/response to understand the format!

Request Example:

```
POST /bin/login HTTP/1.1
```

```
Host: 127.0.0.1:8000
Accept: image/gif, image/jpeg, */*
Referer: http://127.0.0.1:8000/login.html
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Length: 37
Connection: Keep-Alive
Cache-Control: no-cache

User=Peter+Lee&pw=123456&action=login
```

Response Example:

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug
```

```
<html><body><h1>It works!</h1></body></html>
```

It looks… really complicated. Which is why programmers almost never compose raw HTTP requests directly! Instead, we use a client library to handle it for us and only fill in the important parts. However, it is important to note that both request and response are basically long strings. You can read more about this at this link

HTTP: The Important Parts

So what are the important parts that programmers need to specify in any HTTP request?

Request

1. Method
    - Is it a GET, POST, PATCH, PUT request
    - For a GET request, the server will ignore the request body
2. Path
    - Where to send the request to? Either a URL or actual IP address
    - Examples of valid path: `localhost:3000`, `https://google.com`, `127.0.0.1:3000`
3. Headers
    - Optional
    - Usually used for authorization purposes/other info about this request
    - usually in a key-value format, like a hash
    - E.g. `authorization_token: 1234dasd123123`, `Content-Accept: application/json`
    - Special Note:
        - `Content-Accept: application/json` means the client expects the server to respond with a JSON
        - `Content-Type: application/json` means the request body is in a JSON format, other formats are possible, e.g. uploading a file -> `application/multipart`
4. Body
    - Not needed for a GET request
    - Usually contains the data to be sent, e.g. data from a form, uploading a file, etc.
    - Basically one BIG string

Response

1. Status Code
    - a 3 digit code to indicate the status of the request
    - e.g. `404 -> Not Found`, `302 -> Redirected`, `200 -> Success!`
    - complete list of codes can be found here
2. Headers
    - Optional
    - Usually handled by client library
    - Uses `Content-Type` in a similar way to Request
3. Body
    - Contains response body, e.g. string containing entire HTML of a website, a json containing specific data

Sending HTTP requests in Ruby

Now that you know the theory of HTTP requests, lets try sending a request to Google in Ruby using the rest-client gem.

```
require 'rest-client`


RestClient.get 'https://google.com'
```

Thats all! `RestClient` handles everything else for you. Lets see an example of a POST request this time.

```
# RestClient.post <url>, <body>, <headers>
RestClient.post(
 "http://example.com/resource",
 {'x' => 1}.to_json ,
 {content_type: :json, accept: :json}
)
```

A bit more complicated, but far easier than writing the following:

```
POST /resource HTTP/1.1
Host: example
Cf-Ipcountry: MY
Cf-Ray: 35507b52c06f2ccf-KUL
Cf-Connecting-Ip: 121.121.81.104
Via: 1.1 vegur
Content-Length: 7
Accept-Encoding: gzip
Cf-Visitor: {"scheme":"http"}
Connection: close
Content-Type: application/json
Accept: application/json
Connect-Time: 1
User-Agent: Ruby
X-Request-Id: 920174e7-167a-4f29-be6e-fd356c48f602
Total-Route-Time: 0


"\{\"x\" : 1\}"
```