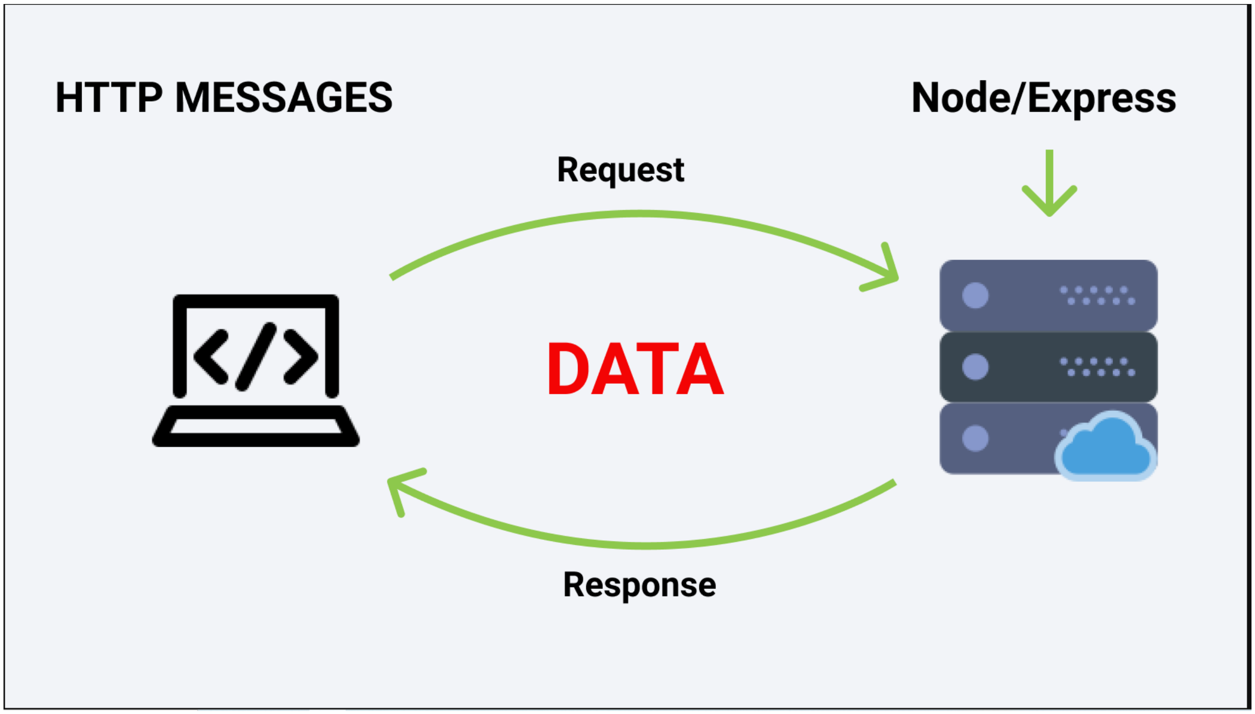
Express

HTTP Request/Response Cycle

Every time we open the browser and type the URL, we are performing a request to server that is responsible for serving a response. Now this is done using HTTP protocol and these are called HTTP messages.

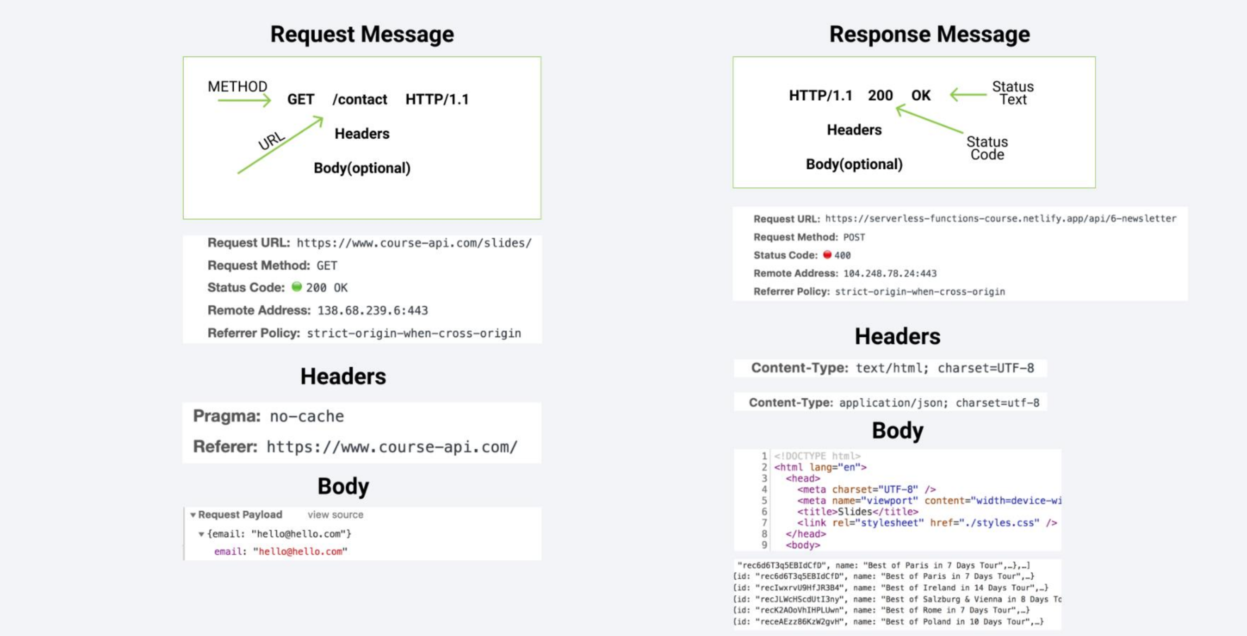


Suer sends a HTTP request message and then server sends an HTTP response message and that's how we exchange data on web.

We mostly use Node but in order make our work easier we use a framework named Express JS. A server job is always to make a resource available. A server doesn't have an GUI (Graphical User Interface). Cloud is nothing but a bunch of servers and computers connected.

HTTP Messages

Let’s see how HTTP messages are structured.



```
Request URL: https://www.course-api.com/slides/
Request Method: GET
Status Code: ● 200 OK
Remote Address: 138.68.239.6:443
Referrer Policy: strict-origin-when-cross-origin
```

```
Pragma: no-cache
Referer: https://www.course-api.com/
```

```
▾ Request Payload    view source
  ▾ {email: "hello@hello.com"}
      email: "hello@hello.com"
```

```
Request URL: https://serverless-functions-course.netlify.app/api/6-newsletter
Request Method: POST
Status Code: ● 400
Remote Address: 104.248.78.24:443
Referrer Policy: strict-origin-when-cross-origin
```

```
Content-Type: text/html; charset=UTF-8
```

```
Content-Type: application/json; charset=utf-8
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-wi
    <title>Slides</title>
    <link rel="stylesheet" href="./styles.css" />
  </head>
  <body>
```

```
 "rec6d6T3q5EBIdCfD", name: "Best of Paris in 7 Days Tour",…},…]
{id: "rec6d6T3q5EBIdCfD", name: "Best of Paris in 7 Days Tour",…}
{id: "recIwxrvU9HfJR3B4", name: "Best of Ireland in 14 Days Tour",…}
{id: "recJLWcHScdUtI3ny", name: "Best of Salzburg & Vienna in 8 Days Tc
{id: "recK2AOoVhIHPLUwn", name: "Best of Rome in 7 Days Tour",…}
{id: "receAEzz86KzW2gvH", name: "Best of Poland in 10 Days Tour",…}
```

General structure for both messages (request and response) is similar.

They both have a start line, they both have optional headers, a blank line that indicates that all meta info has been sent and effectively headers are that meta info as well as optional body.

Request Messages – messages sent by a user.

Response Messages – messages sent by a server.

In General, when we talk about **request message** in start line there’s going to be a method, then URL and then HTTP version as well.

Methods is the place where we communicate what we want to do.

> Ex: If we want to get the resource then we set it up as GET request.
>
> If we want to add the resource, then we set it up as POST request.
>
> ***GET request is the default request that the browser performs (since we open the browser and get some request from web, hence GET is the default request).***

URL is just the address. (Ex: freecodecamp.org)

Headers is essentially optional; it is meta information about our request. Headers have a key-value pair. We don’t need to add headers manually but in few in cases we need to add headers. (Basically, it an information about our message).

Body, if we just need the data from resource then there is no body but if we want to add a resource to the server then we are expected to provide a body and that is called request payload.

When we talk about response message, the Node JS developers will be creating the response. Start line has the HTTP version, then we have a status code and status text.

HTTP version – it is mostly going to be 1.1

Status Code, it just signals what is the result of the request.

Ex: Status Code: 200 – Request was successful.

Status Code: 400 – There was an error in the request.

Status Code: 404 – Resource was not found.

Headers, we provide info about our message. (It is a setup of key value pairs).

Content-Type: text/html; we are sending back the html.

Content-Type: application/json: we are sending back the data.

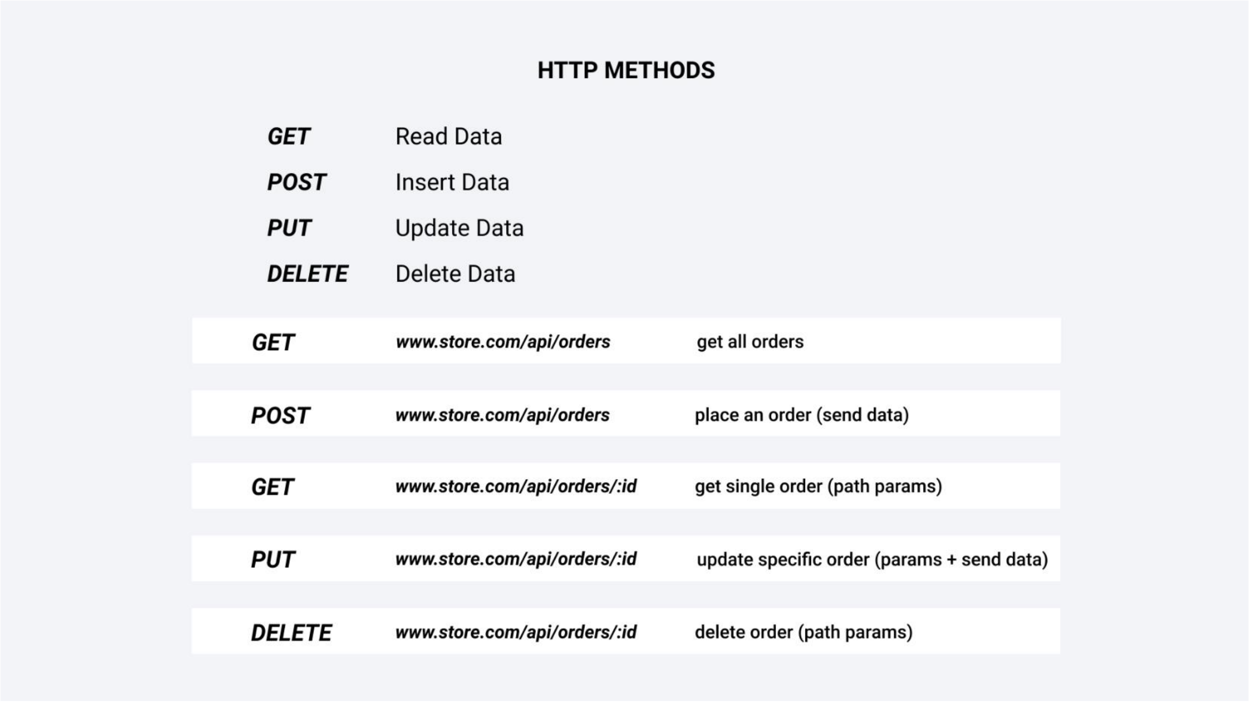
When we communicate with API, mostly we are getting back the JSON data because over the web effectively we just send over the string.

In our headers we indicate that we are sending the data in application/json and then that application (web application) which is requesting knows that they are receiving application/json from the server.

Starter Project Install

Clone projects from https://github.com/john-smilga/node-express-course

HTTP Methods



HTTP METHODS

GET	Read Data
POST	Insert Data
PUT	Update Data
DELETE	Delete Data

GET	***www.store.com/api/orders***	get all orders
POST	***www.store.com/api/orders***	place an order (send data)
GET	***www.store.com/api/orders/:id***	get single order (path params)
PUT	***www.store.com/api/orders/:id***	update specific order (params + send data)
DELETE	***www.store.com/api/orders/:id***	delete order (path params)