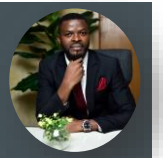
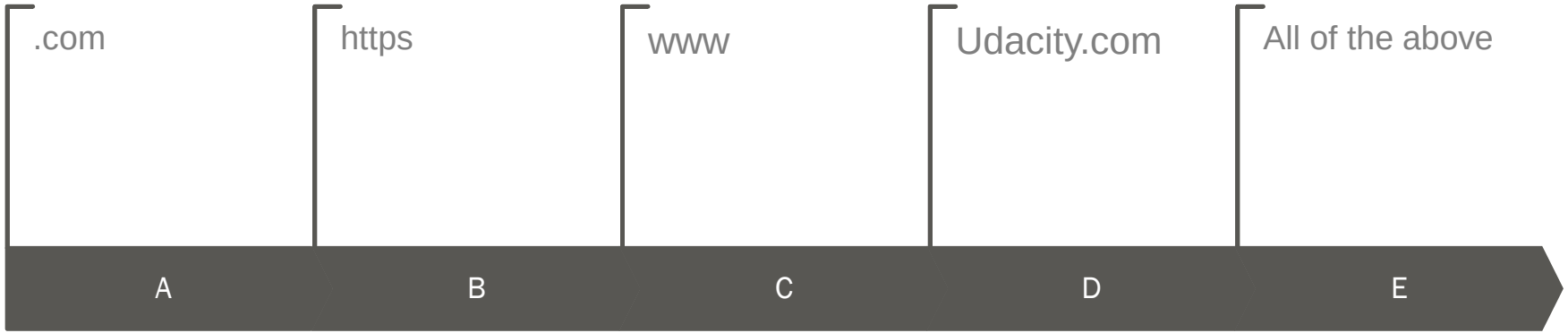
FULL-STACK NANODEGREE SESSION 5

AJIROGHENE SUNDAY

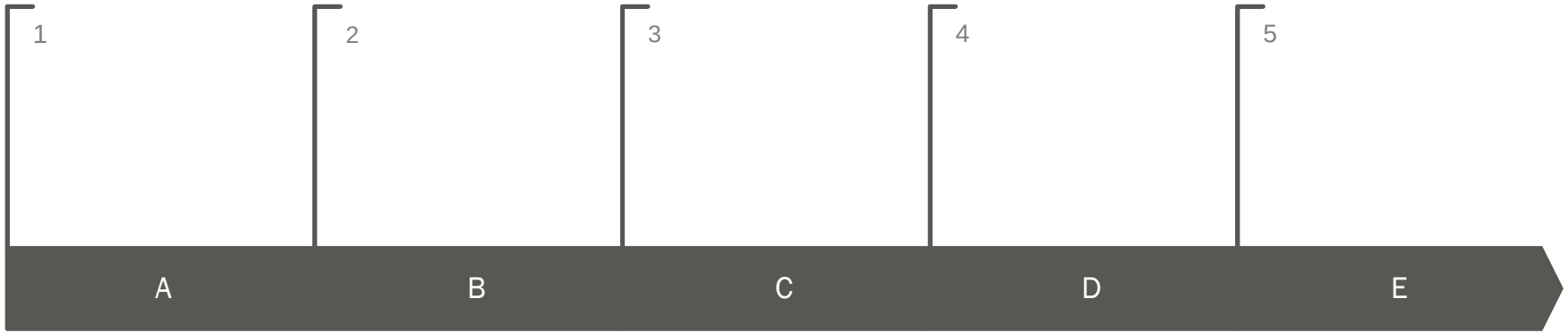


1. WHICH OF THE FOLLOWING IS CALLED AS PROTOCOL IN THE URL?

HTTPS://WWW.UDACITY.COM



2. AN HTTP REQUEST CONTAINS ____ PARTS.



1. Request method:Tells the server the type of action the clients want to be performed.

A full request message always begins with a Request-Line, which has the following format:

Request-Line=Method SP Request-URL HTTP Version CRLF

Methods in HTTP/1.1

a) OPTIONS: a request for information about the options available for the request/response chain identified by this URL

b) GET a request to retrieve the information identified in the URL and return it in an entity body. A GET is conditional if the If-Modified-Since header field is included and is partial if a Range header field is included

c) HEAD the request is identical to a GET, except that the server's response must not include an entity body; all of the header fields in the response are the same as if the entity body were present. This enables a client to get information about a resource without transferring the entity body.

Other methods are:

POST, PUT, PATCH, COPY, MOVE, DELETE, LINK, UNLINK, TRACE, WRAPPED, EXTENSION

2. URI: Uniform Resource Indicator specifies the address of the required document.

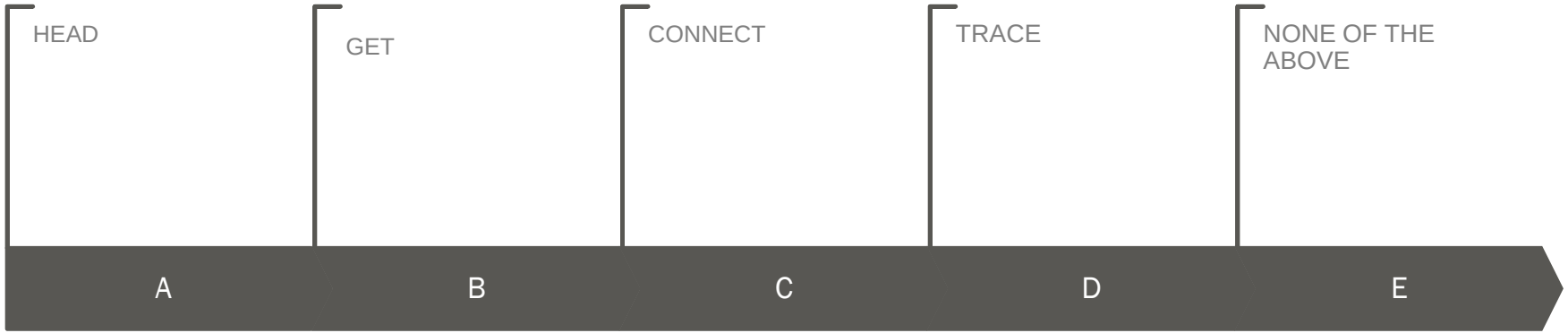
3. Header fields: Option headers can be used by the client to tell server extra information about request e.g. Client software and content type that it understands.

Fields defined in HTTP/1.1:

Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization, From, Host, If-Modified Since, Range, Referrer, Unless, User-Agent, Proxy-Authorization

4. Body: contains data sent by the client

3 WHICH IS NOT AMONG THE REQUEST METHOD



HEAD
Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

GET
Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval". The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations." See safe methods below.

POST
Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

PATCH
Is used to apply partial modification

PUT
Uploads a representation of the specified resource.

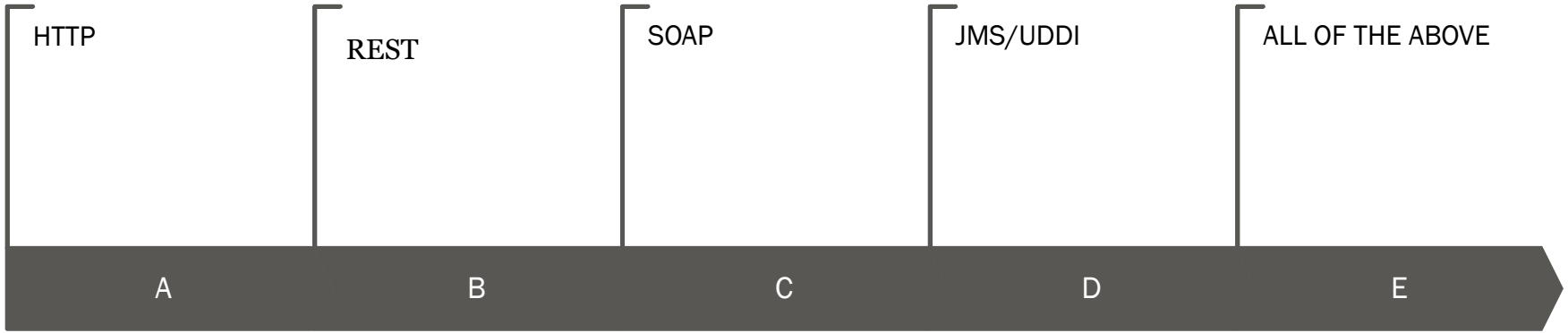
DELETE
Deletes the specified resource.

TRACE
Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.

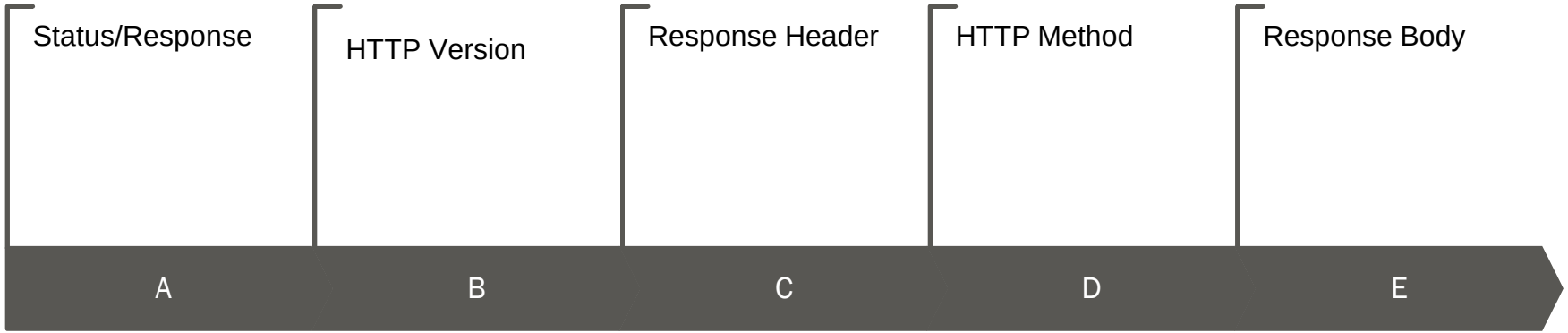
OPTIONS
Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT
Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

4 WHAT ARE THE PROTOCOLS USED IN API TESTING?



5 NAME THE COMPONENT OF HTTP RESPONSE THAT CONTAINS METADATA FOR THE HTTP RESPONSE MESSAGE AS KEY-VALUE PAIRS?



6 WHAT IS A MODEL IN WEB API?

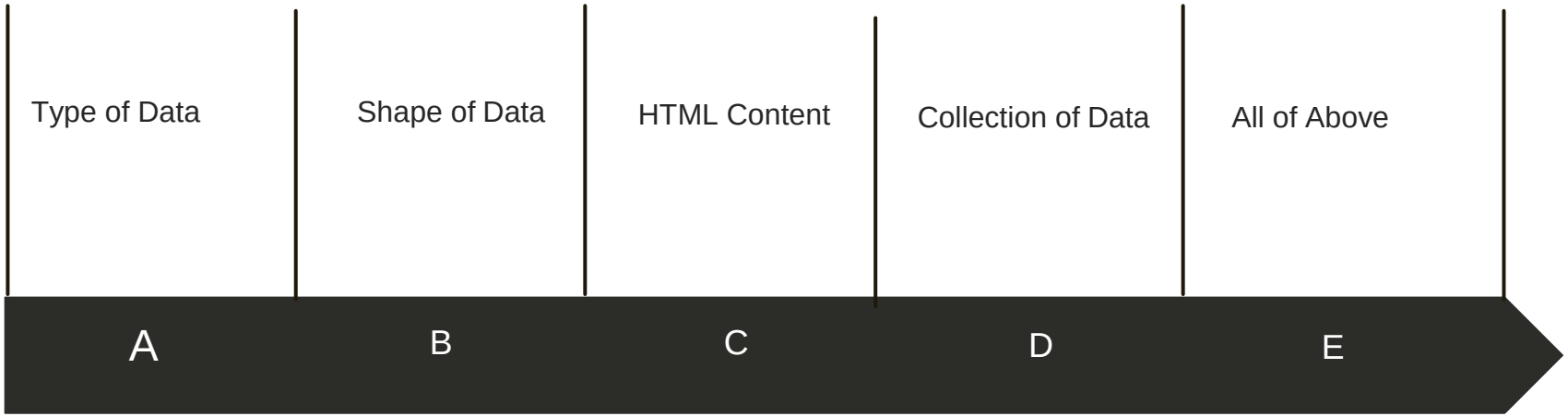
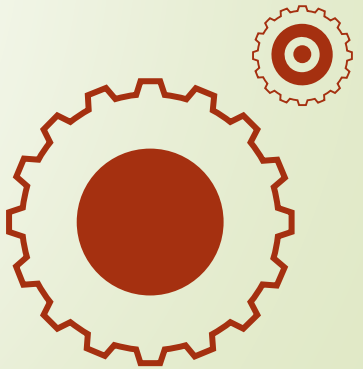


Table of contents

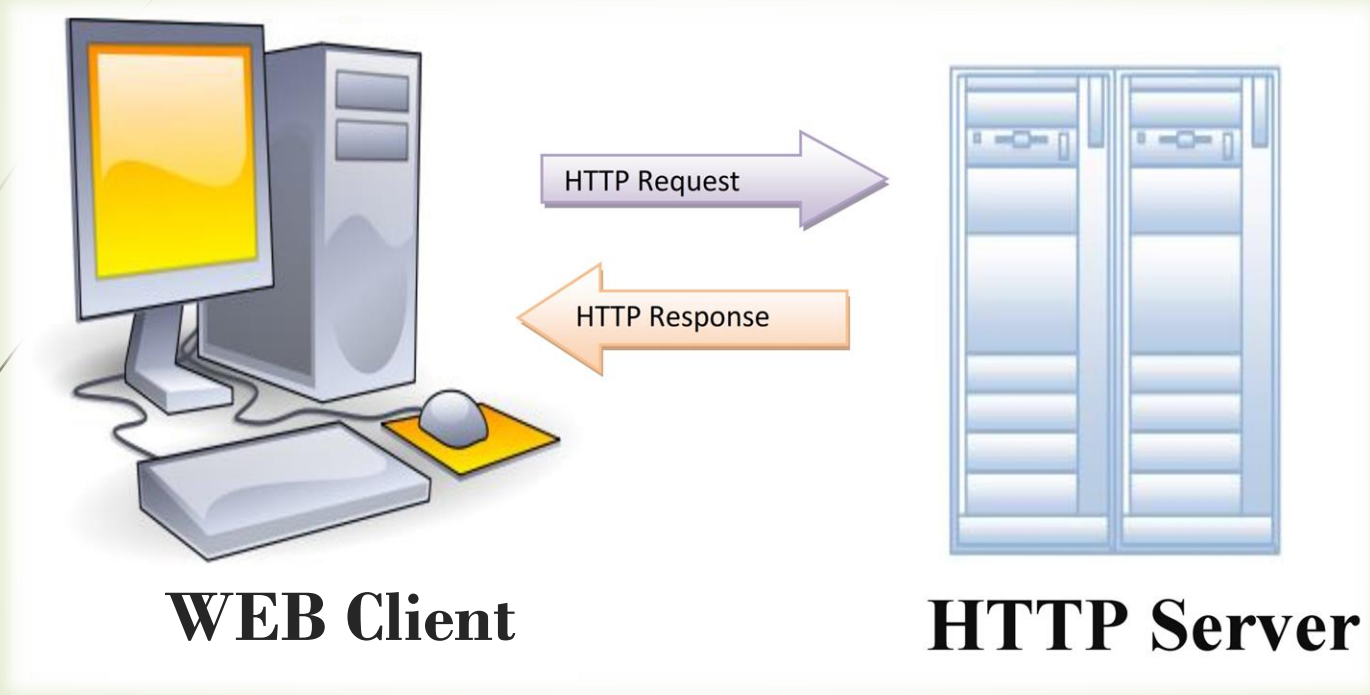
Introduction

A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.



01
Recap

HTTP BASICS



AJAX

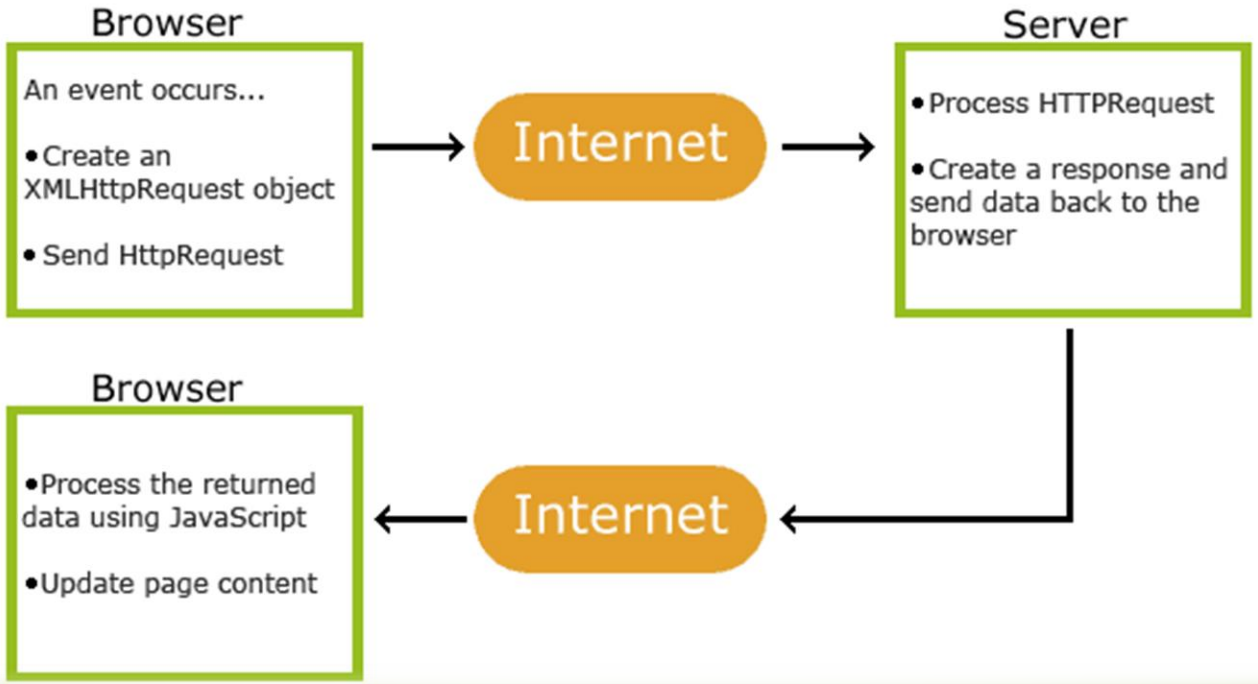
AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

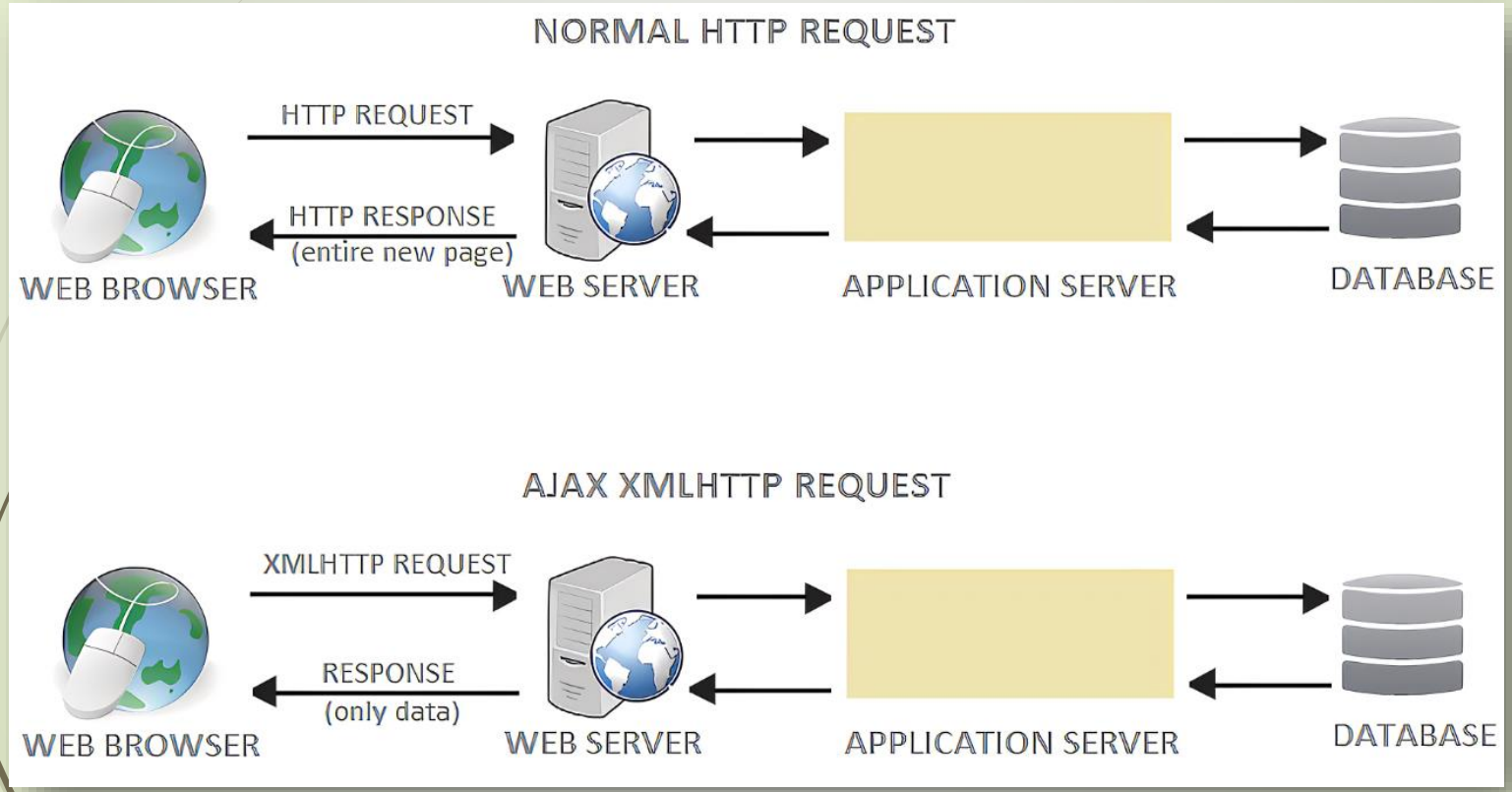
AJAX uses a combination of:

- ✓ A browser built-in XMLHttpRequest object (to request data from a web server)
- ✓ JavaScript and HTML DOM (to display or use the data)

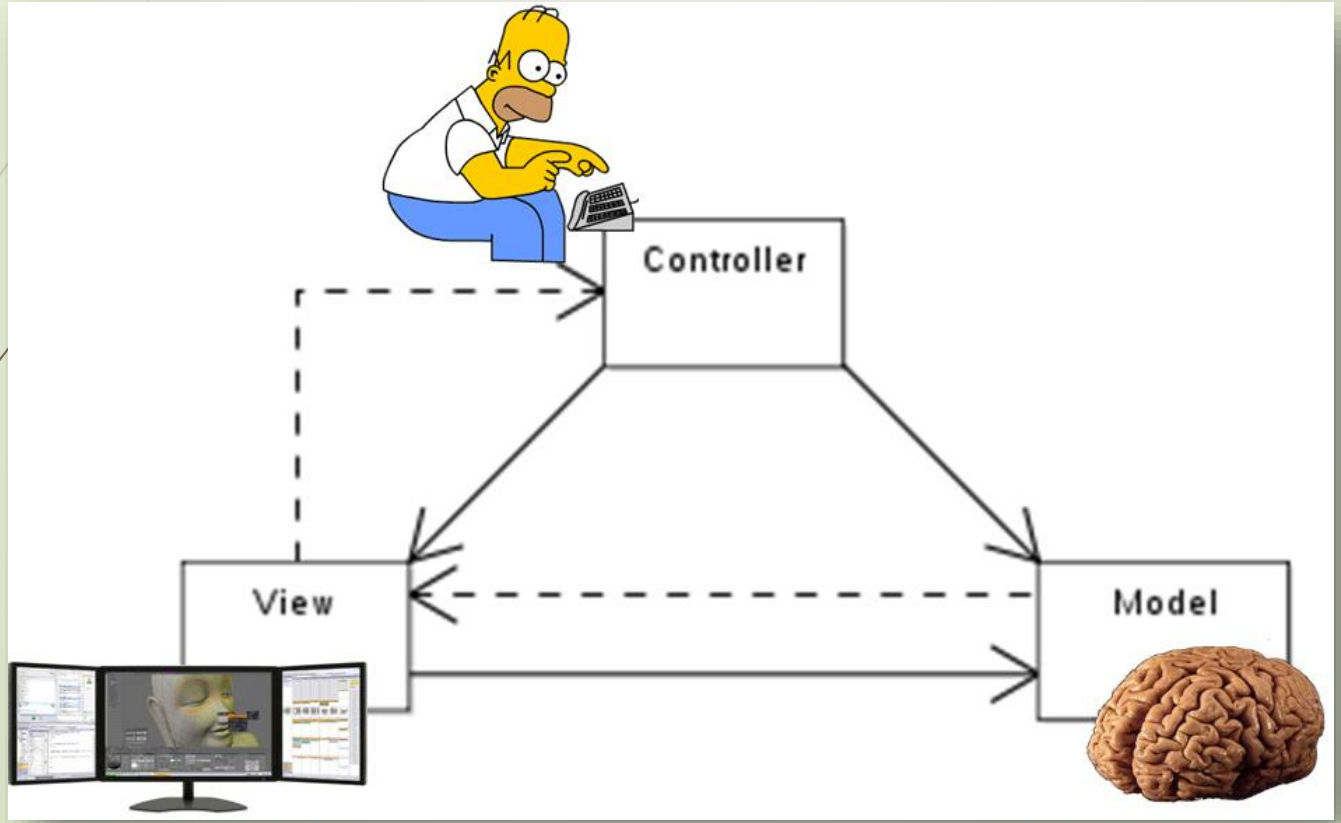
How AJAX Works



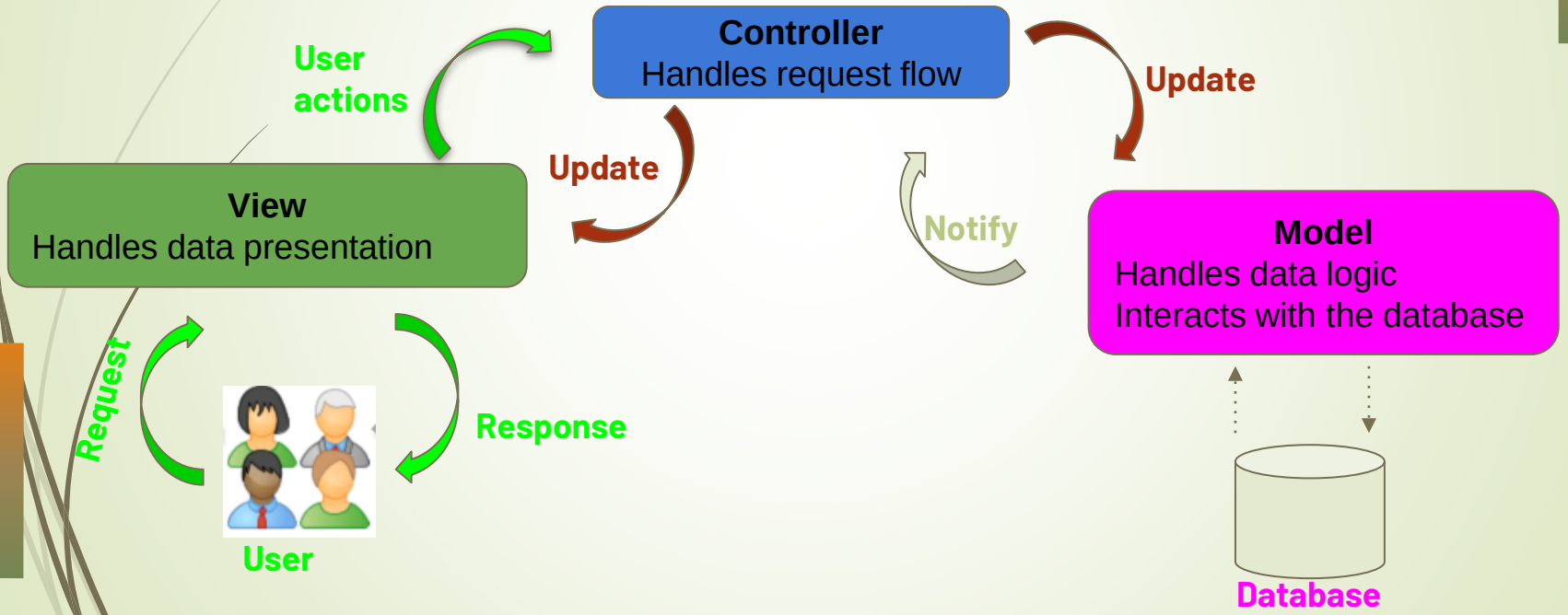
NORMAL HTTP REQUEST VS AJAX



Model, View, Controller



Model, View, Controller



02

Application **P**rogramming **I**nterface (API)

API stands for

Application **P**rogramming **I**nterface.

A Web API is an application programming interface for the Web.

A Browser API can extend the functionality of a web browser.

A Server API can extend the functionality of a web server.

An API is a set of programming code that enables data transmission between one software product and another

Browser APIs

All browsers have a set of built-in Web APIs to support complex operations, and to help accessing data.

For example, the Geolocation API can return the coordinates of where the browser is located.

Example

Get the latitude and longitude of the user's position:

```
const myElement = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    myElement.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  myElement.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
```

How an API works

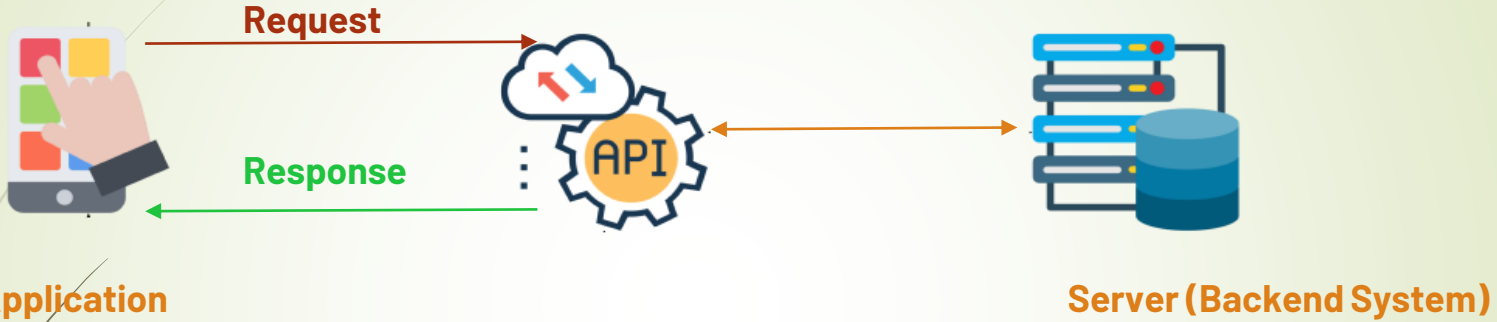
An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

Here's how an API works:

1. **A client application initiates an API call** to retrieve information—also known as a *request*. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
2. **After receiving a valid request**, the API makes a call to the external program or web server.
3. **The server sends a *response*** to the API with the requested information.
4. **The API transfers the data** to the initial requesting application.

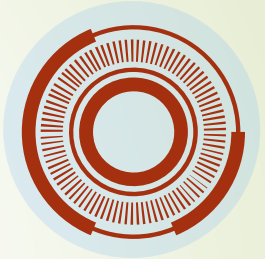
While the data transfer will differ depending on the web service being used, this process of requests and response all happens through an API. Whereas a user interface is designed for use by humans, APIs are designed for use by a computer or application.

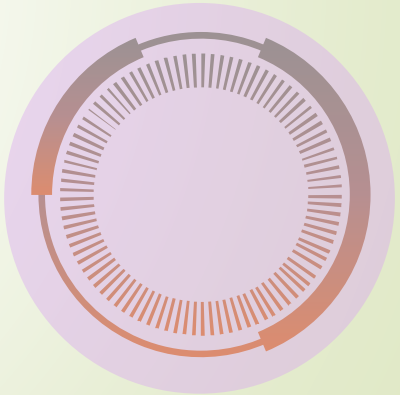
How do APIs work?



Why use APIs?

- To provide a standardized way of accessing data
- Share application without exposing the implementation to those who shouldn't have access to it.
- To simplify how to access the application's data and functionality.





Why we need APIs

- **Improved collaboration:** APIs enable integration so that platforms and apps can seamlessly communicate with one another
- **Easier innovation:** APIs offer flexibility, allowing companies to make connections with new business partners, offer new services to their existing market, and, ultimately, access new markets that can generate massive returns and drive digital transformation
- **Data monetization:** Many companies choose to offer APIs for free. However, if the API grants access to valuable digital assets, you can monetize it by selling access (this is referred to as the API economy).
- **Added security:** As noted above, APIs create an added layer of protection between your data and a server.

Types of APIs

Open APIs are open source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.

Partner APIs are application programming interfaces exposed to or by strategic business partners. Typically, developers can access these APIs in self-service mode through a public API developer portal.

Internal APIs are API that remain hidden from external users. These private APIs aren't available for users outside of the company, they're intended to improve productivity and communication across different internal development teams.

Composite APIs combine multiple data or service APIs. These services allow developers to access several endpoints in a single call. They are useful in microservices architecture where performing a single task may require information from several sources.

Types of API protocols

SOAP (Simple Object Access Protocol) is an API protocol built with XML, enabling users to send and receive data through SMTP and HTTP. With SOAP APIs, it is easier to share information between apps or software components that are running in different environments or written in different languages.

XML-RPC is a protocol that relies on a specific format of XML to transfer data, whereas SOAP uses a proprietary XML format. XML-RPC is older than SOAP, but much simpler, and relatively lightweight in that it uses minimum bandwidth.

JSON-RPC is a protocol similar to XML-RPC, as they are both remote procedure calls (RPCs), but this one uses JSON instead of XML format to transfer data. Both protocols are simple. While calls may contain multiple parameters, they only expect one result.

REST (Representational State Transfer) is a set of web API architecture principles, which means there are no official standards (unlike those with a protocol). To be a REST API (also known as a RESTful API), the interface must adhere to certain architectural constraints. It's possible to build RESTful APIs with SOAP protocols, but the two standards are usually viewed as competing specifications.

SOAP VS REST

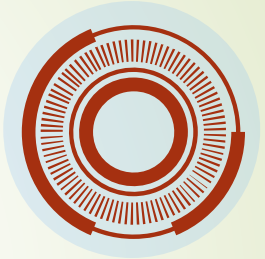
SOAP	REST
Simple Object Access Protocol	Representational State Transfer
A protocol	An architectural pattern
uses service interfaces to expose its functionality to client applications	uses Uniform Service locators to access to the components on the hardware device.
needs more bandwidth for its usage	Doesn’t need much bandwidth
only works with XML formats	works with plain text, XML, HTML and JSON.
cannot make use of REST	can make use of SOAP

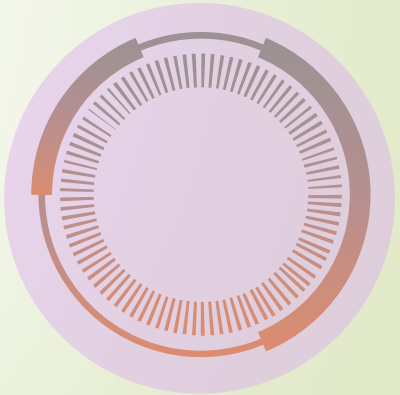
What is REST?

REST stands for Representational State Transfer.

Concepts

- Separation of Client and Server
- Server Requests are Stateless
- Cacheable Requests
- Uniform Interface





What is RestFul APIs?

Representational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. **REST API** is a way of accessing web services in a simple and flexible way without having any processing.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.

How it Works

Working: A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON. But now JSON is the most popular format being used in Web Services.

In **HTTP** there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

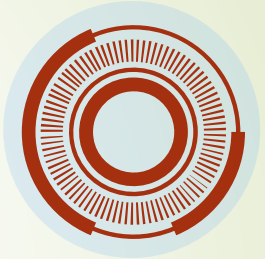
RESTful Principles

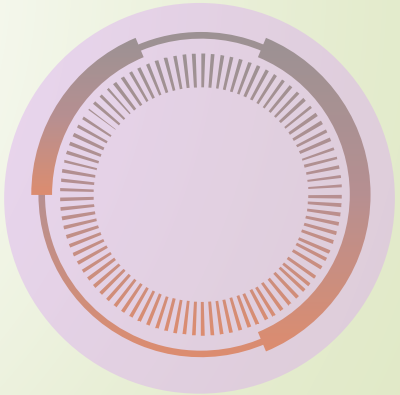
REpresentational State Transfer

- **Uniform Interface:** Resources, Representations and Self-Descriptive Messages
- **Stateless:** Self-contained requests
- **Client-Server:** There must be both a client and server in the architecture
- **Cacheable & Layered System:** Network efficiency

Why do we use RestFul APIs?

- They provide a great deal of flexibility and scalability.





03

HTTP Basics

What is Http?

HTTP stands for HyperText Transfer Protocol and is a protocol that provides a standardized way for computers to communicate with each other.

How does HTTP Work?

CLIENT

REQUEST
- VERB
- HEADERS
- CONTENT

SERVER

RESPONSE
- STATUS CODE
- HEADERS
- CONTENT

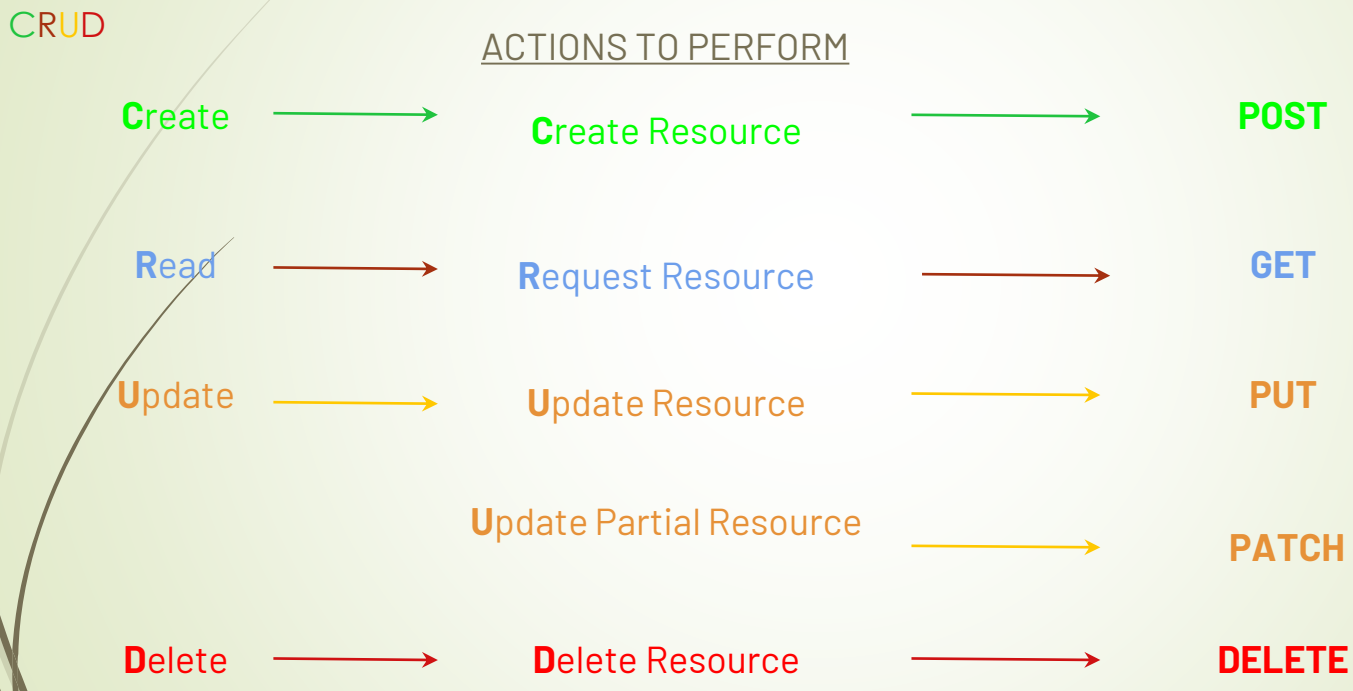
stateless

HTTP Features

- **Connectionless**
  When a request is sent, the client opens the connection; once a response is received, the client closes the connection. The client and server only maintain a connection during the response and request. Future responses are made on a new connection.

- **Stateless**
  There is no dependency between successive requests.
- **Not sessionless**
  Utilizing headers and cookies, sessions can be created to allow each HTTP request to share the same context.
- **Media Independent**
  Any type of data can be sent over HTTP as long as both the client and server know how to handle the data format. In our case, we'll use JSON.

HTTP Methods (VERBS)



HTTP Headers

METADATA ABOUT THE REQUEST / RESPONSE

used to describe a resource, or the behavior of the server or the client

- **Content Type**: The format of the Content
- **Content Length:** The Size of the Content
- **Authorization:** Who's making the call
- **Accept:** When you send me a response, what kind / type(s) of data can I accept
- **Cookies:** Passenger data in the request - Data being sent with the requests that expects the server to also to pass back as a way to have state through the entire process.
- **Expires:** When to consider stale- How long the data should be cached
- …….

Metadata - a set of data that describes and gives information about other data

HTTP Content

- **HTML**
- **CSS**
- **Javascript**
- **XML**
- **JSON**
- **Binary and blobs**

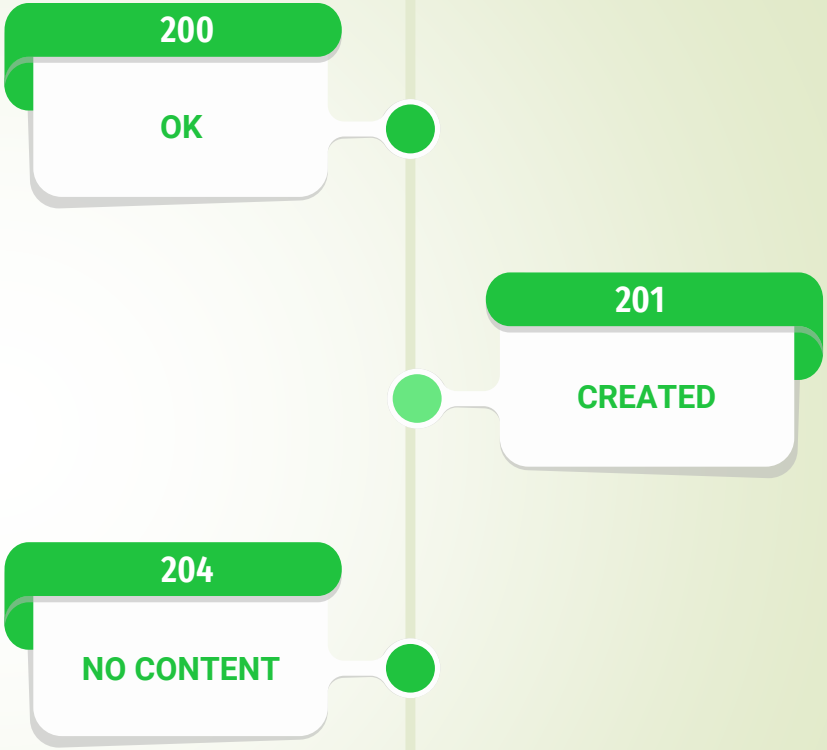
HTTP Response Status Codes

A NUMBER THAT INDICATES WHAT KIND OF SUCCESS

STATUS CODE	DESCRIPTION
1xx	Informational responses (100–199)
2xx	Successful responses (200–299)
3xx	Redirection messages (300–399)
4xx	Client error responses (400–499)
5xx	Server error responses (500–599)

Successful Responses

Range: 200 - 299



Client Error Responses

Range: 400 - 499

Server Error Responses

Range: 500 - 599

500

INTERNAL SERVER ERROR

503

SERVER UNAVAILABLE

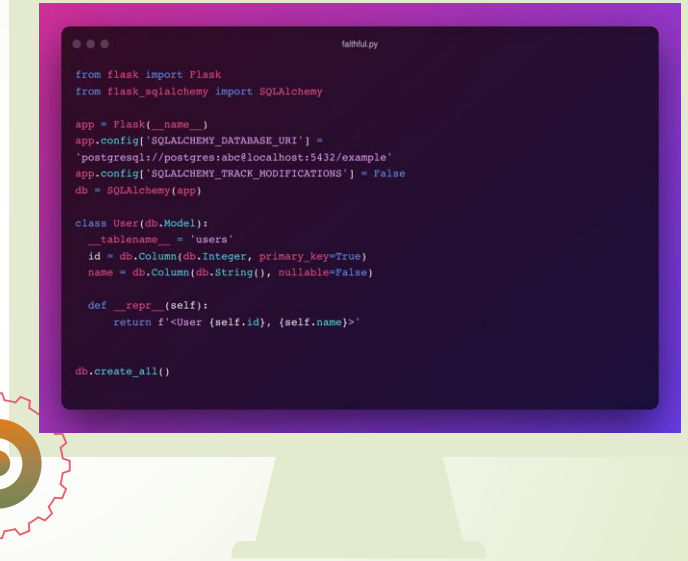
504

GATEWAY TIMEOUT

Practical Application

PostMan

ThunderClient

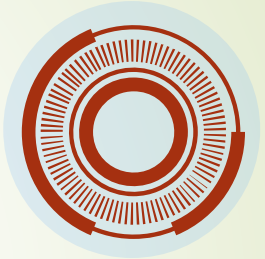


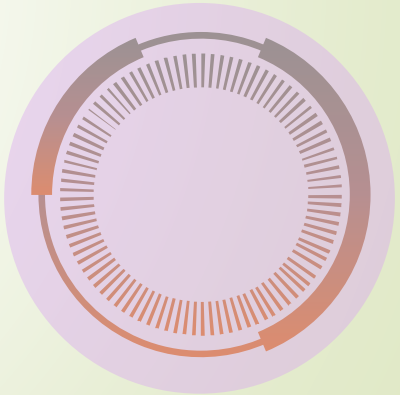
04

What Next

What Next

- Endpoints and Payload
- API Testing
- Api Documentation





QUESTIONS

Feedback

This is where you can tell me what needs improvement

THANK YOU

Big thank you can catch your audience's attention

API & DOCUMENTATION

PRE STUDY PLAN

API & Route

- **RestFul APIs Fundamental**: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API
- **RESTful-API video Tutorial** : https://youtu.be/rtWH70_MMHM
- Restful Api with Python Flask : https://youtu.be/Sf-7zXBB_mg

Flask

- **Flask Fundamental**: https://www.tutorialspoint.com/flask/index.htm
- **Flask video Tutorial** : https://youtu.be/Z1RJmh_OqeA

Testing Fundamentals and Application

- https://youtu.be/iQVvpnRYI-w;
- https://youtu.be/dTvJwxrM1VY

HTTP Response Status Codes

STATUS CODE	DESCRIPTION
200 ok	The Response Succeeded
201 Created	The request Succeeded and a new resource was created
204 No Content	No content for the request

HTTP Response Status Codes

STATUS CODE	DESCRIPTION
301 Moved Permanently	The URL of the requested resource has been changed permanently. The new URL is given in the response
304 Not Modified	This is used for caching purposes. It tells the client that the response has not been modified, so the client can continue to use the same cached version of the response.

HTTP Response Status Codes

STATUS CODE	DESCRIPTION
400 Bad Request	The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).
401 Unauthorized	That is, the client must authenticate itself to get the requested response.
403 Forbidden	The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource.

HTTP Response Status Codes

STATUS CODE	DESCRIPTION
404 Not Found	The server can not find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of 403 Forbidden to hide the existence of a resource from an unauthorized client.
405 Method Not Allowed	The request method is known by the server but is not supported by the target resource.
409 Conflict	The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource.

HTTP Response Status Codes

STATUS CODE	DESCRIPTION
422 Unprocessable Entity	The request was well-formed but was unable to be followed due to semantic errors.