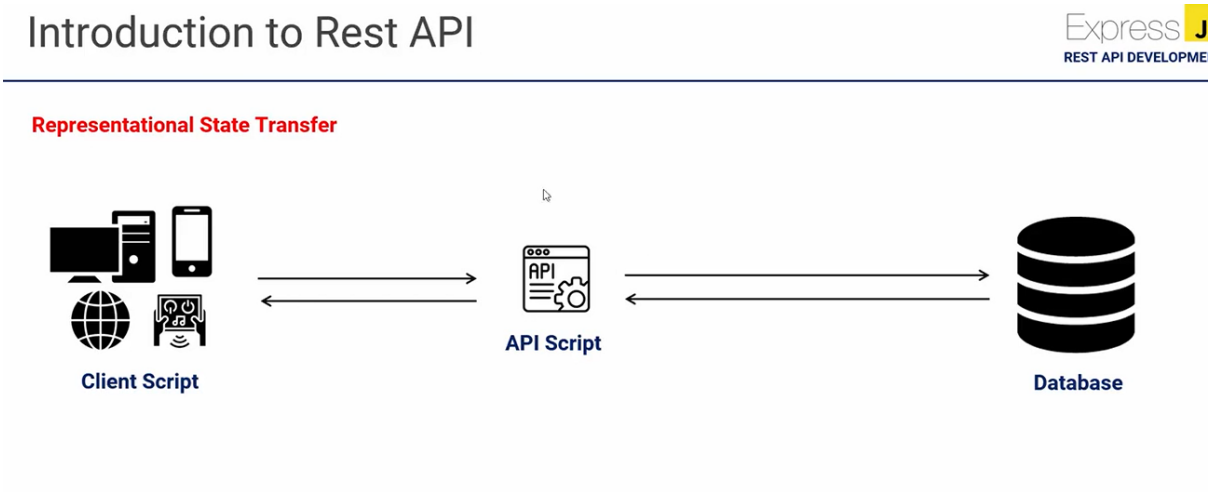
# Rest API Best Practices




## 1) First, What is a REST API?

=> REST stands for Representational State Transfer

## 2) URLS as resources as nouns

=> one of the most recognizable characteristics of REST in the predominant use of nouns in URLs. Restful URLs should not indicate any kind of CRUD functionality.

Example:/users/{id} instead of /getUser

## 2) Forward slashes for hierarchy

=> As shown in the examples above, forward slashes are conventionally used to show the hierarchy between individual resources and collections

> *Example:* `/users/{id}/address` clearly falls under the `/users/{id}` resource which falls under the `/users` collection.

## 3) Punctuation for lists

= When there is no hierarchical relationship (such as in lists), punctuation marks such as the semicolon, or, more frequently, the comma should be used.

> *Example:* `/users/{id1},{id2}` to access multiple user resources

## 4) Query parameters where necessary:

= In order to sort or filter a collection, a REST API should allow query parameters to be passed in the URI.

*Example:* `/users?location=USA` to find all users living in the United States

## 5) Lowercase letters and dashes:

= By convention, resource names should use exclusively lowercase letters. Similarly, dashes (-) are conventionally used in place of underscores (_).

*Example:* `/users/{id}/pending-orders` instead of `/users/{id}/Pending_Orders`

## 6) No file extensions:

= Leave file extensions (such as .xml) out of your URIs. We're sorry to say it, but they're ugly and add length to URIs. If you need to specify the format of the body, instead use the Content-Type header

## 7) No trailing forward slash:

= Similarly, in the interests of keeping URIs clean, do not add a trailing forward slash to the end of URIs.

## 8) Use JSON as the Format for Sending and Receiving Data

= In the past, accepting and responding to API requests were done mostly in XML and even HTML. But these days, JSON (JavaScript Object Notation) has largely become the de-facto format for sending and receiving API data.

## 9) Use Filtering, Sorting, and Pagination to Retrieve the Data Requested

= Sometimes, an API's database can get incredibly large. If this happens, retrieving data from such a database could be very slow.

Filtering, sorting, and pagination are all actions that can be performed on the collection of a REST API. This lets it only retrieve, sort, and arrange the necessary data into pages so the server doesn't get too occupied with requests.

An example of a filtered endpoint is the one below:

`https://mysite.com/posts?tags=javascript`

## 10) Use SSL for Security:

= SSL stands for secure socket layer. It is crucial for security in REST API design. This will secure your API and make it less vulnerable to malicious attacks.

`https://mysite.com/posts` runs on SSL.

`http://mysite.com/posts` does not run on SSL.

# Http Methods Best Practices

## 1) HTTP Request :

= HTTP Request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

## 2) Http Request Segments:

| Request Area | Standard Data Type |
|---|---|
| Body | Simple String, JSON, Download, Redirect, XML |
| Header | Key Pair Value |
| URL Parameter | String |

## 3) HTTP Request Methods:

| Method Name | Responsibilities |
|---|---|
| GET() | The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| Head() | Same as GET, but transfers the status line and header section only. |
| POST() | A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| PUT() | Replaces all current representations of the target resource with the uploaded content. |
| DELETE() | Removes all current representations of the target resource given by a URI. |

## 4) Http Request Throttling:

= Throttle Request refers to a process in which a user is allowed to hit the application maximum time in per second or per minute. Throttling is also known as request rate limiting.

Essential component of Internet security, as DoS attacks can tank a server with unlimited requests.

Rate limiting also helps make your API scalable by avoid unexpected spikes in traffic, causing severe lag time.

## 5) HTTP Response:

Http response is the final step of request-response communication. Every response is a combination of response header, body and cookies.

**6) Http Response Segments:**

| Response Area | Standard Data Type |
|---|---|
| Body | Simple String, JSON, Download, Redirect, XML |
| Header | Key Pair Value |
| Cookies | Key Pair Value |

## 7) HTTP Response status messages

| Code | Meaning | Description |
|---|---|---|
| 200 | OK | The request is OK (this is the standard response for successful HTTP requests) |
| 201 | Created | The request has been fulfilled, and a new resource is created |
| 202 | Accepted | The request has been accepted for processing, but the processing has not been completed |
| 203 | Non-Authoritative Information | The request has been successfully processed, but is returning information that may be from another source |
| 204 | No Content | The request has been successfully processed, but is not returning any content |
| 205 | Reset Content | The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view |
| | | |

| Code | Meaning | Description |
|---|---|---|
| 206 | Partial Content | The server is delivering only part of the resource due to a range header sent by the client |
| 400 | Bad Request | The request cannot be fulfilled due to bad syntax |
| 401 | Unauthorized | The request was a legal request, but the server is refusing to respond to it. |
| 403 | Forbidden | The request was a legal request, but the server is refusing to respond to it |
| 404 | Not Found | The requested page could not be found but may be available again in the future |
| 405 | Method Not Allowed | A request was made of a page using a request method not supported by that page |

| Code | Meaning | Description |
|---|---|---|
| 408 | Request Timeout | Request Timeout |
| 500 | Internal Server Error | A generic error message, given when no more specific message is suitable |
| 502 | Bad Gateway | The server was acting as a gateway or proxy and received an invalid response from the upstream server |
| 503 | Service Unavailable | The server is currently unavailable (overloaded or down) |
| | | |
| | | |

# JSON Best Practices

1

## JavaScript Object Notation (JSON)

- JSON is a lightweight data-interchange format that is completely language independent.
- It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data
- The official Internet media type for JSON is application/json.
- It was designed for human-readable data interchange.
- The filename extension is .json.

2) Key: Value Pair double quotes

```
{'id': '1','name':File} is not right ✗

{"id": 1,"name":"File"} is okay  ✓

{"id": "1","name":"File"} is the best  ✓
```

3)Never Use Hyphens you key fields

```
{"first-name":"Rachel","last-name":"Green"}  is not right. ✗

{"first_name":"Rachel","last_name":"Green"} is okay ✓

{"firstname":"Rachel","lastname":"Green"} is okay ✓

{"firstName":"Rachel","lastName":"Green"} is the best. ✓
```

4)Always Create Roo element

5) Provide META sample:

=> The idea of JSON is flexibility so you don't have to restrict your data feed within few columns.

But at the same time, if you are providing large data set with nested levels, the consumer will go crazy. Provide them with the meta / sample, so it helps them to understand what data to look for and what to skip.

# Request-Response Best Practices

1)Don’t Use Verbs in URLs:

= If you understood the APIs' basics, you would know that inserting verbs in the URL isn’t a good idea. The reason behind this is that HTTP has to be self-sufficient to describe the purpose of the action.

`GET: /articles/:slug/generateBanner/`

The GET method is only able to say here that you just want to retrieve a banner. So, using this syntax might be beneficial:

`GET: /articles/:slug/banner/`

Similarly, for the endpoint, it might generate the new article, as shown in this example.

Don’t use

`POST: /articles/createNewArticle/`

Do use

`POST: /articles/`

2) PostMan http Client :

= Postman is an HTTP Client application, used to test request-response communication. Postman is widely used for API testing and generating documentation.

- Quickly and easily send REST, SOAP, and GraphQL requests directly within Postman.
- Generate and publish beautiful, machine-readable API documentation.
- Checking performance and response times at scheduled intervals.
- Communicate the expected behavior of an API by simulating endpoints and their responses

2) Use Plural Nouns to Name a Collection:

When you have to develop the collection in REST API, just go with plural nouns. It makes it easier for humans to understand the meaning of collection without actually opening it. Let’s go through this example:

```
GET /cars/123

POST /cars

GET /cars
```

3) HTTP Request:

= HTTP Request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

4) Http Request Segments:

| Request Area | Standard Data Type |
|---|---|
| **Body** | Simple String, JSON, Download, Redirect, XML |
| **Header** | Key Pair Value |
| **URL Parameter** | String |

5) HTTP Request Methods:

| Method Name | Responsibilities |
|---|---|
| GET() | The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| Head() | Same as GET, but transfers the status line and header section only. |
| POST() | A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| PUT() | Replaces all current representations of the target resource with the uploaded content. |
| DELETE() | Removes all current representations of the target resource given by a URI. |

6) Request Compare GET vs. POST:

| Key Points | GET | POST |
|---|---|---|
| BACK button/Reload | Harmless | Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted) |
| Bookmarked | **Can be bookmarked** | Cannot be bookmarked |
| Cached | Can be | Never |
| Encoding type | application/x-www-form-urlencoded | application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data |
| History | Parameters remain in browser history | Parameters are not saved in browser history |
| Restrictions on data length | Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters) | No restrictions |

7) Http Request Throttling:

Throttle Request refers to a process in which a user is allowed to hit the application maximum time in per second or per minute. Throttling is also known as request rate limiting. Essential component of Internet security, as DoS attacks can tank a server with unlimited requests.Rate limiting also helps make your API scalable by avoid unexpected spikes in traffic, causing severe lag time.

8) HTTP Response:

= Http response is the final step of request-response communication. Every response is a combination of response header, body and cookies.

9) Http Response Segments:

| Response Area | Standard Data Type |
|---|---|
| Body | Simple String, JSON, Download, Redirect, XML |
| Header | Key Pair Value |
| Cookies | Key Pair Value |

10) HTTP Response status messages:

| Code | Meaning | Description |
|---|---|---|
| 200 | OK | The request is OK (this is the standard response for successful HTTP requests) |
| 201 | Created | The request has been fulfilled, and a new resource is created |
| 202 | Accepted | The request has been accepted for processing, but the processing has not been completed |
| 203 | Non-Authoritative Information | The request has been successfully processed, but is returning information that may be from another source |
| 204 | No Content | The request has been successfully processed, but is not returning any content |
| 205 | Reset Content | The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view |
| | | |

| Code | Meaning | Description |
|---|---|---|
| 206 | Partial Content | The server is delivering only part of the resource due to a range header sent by the client |
| 400 | Bad Request | The request cannot be fulfilled due to bad syntax |
| 401 | Unauthorized | The request was a legal request, but the server is refusing to respond to it. |
| 403 | Forbidden | The request was a legal request, but the server is refusing to respond to it |
| 404 | Not Found | The requested page could not be found but may be available again in the future |
| 405 | Method Not Allowed | A request was made of a page using a request method not supported by that page |

# Web Security Practices

1) **Output Validation**

Output Header:

Provide proper http response status code.

Provide proper content type, file type if any.

Provide cache status if any.

Authentication token should provide via response header.

Only string data is allowed for response header.

Provide content length if any.

Provide response date and time.

Follow request-response model described before.

Output Body:

Avoid providing response status, code, message via response body

Use JSON best practices for JSON response body.

For single result, can use String, Boolean directly.

Provide proper JSON encode-decode before writing JSON Body.

Follow discussion on JSON described before.

2) Request Rate limit- Throttling:

= We need to make sure our APIs are running as efficiently as possible. Otherwise, everyone using your database will suffer from slow performance. Performance isn't the only reason to limit API requests, either. API limiting, which also known as rate is limiting, is an essential component of Internet security, as DoS attacks can tank a server with unlimited API requests.

Rate limiting also helps make your API scalable. If your API blows up in popularity, there can be unexpected spikes in traffic, causing severe lag time.

3) CSRF/XSRF Protection :

Cross-site request forgery attacks (CSRF or XSRF for short) are used to send malicious requests from an authenticated user to a web application.

Use request-response header to pass CSRF token

CSRF token should be unique for every session For self API CSRF token works well.

| Language | Platform | Library name | Library Sources |
|---|---|---|---|
| C# | ASP.NET | AntiCSRF | Nuget package manager |
| PHP | Laravel | Laravel Default | Packagist |
| JS | Node/Express JS | npm i csrf | NPM |

4) User Agent Protection:

User agent is a request header property, describe client identity like operating system, browser details, device details etc. Moreover every web crawler like Google crawler, Facebook crawler has specific user-agent name.

- Using user agent we can prevent REST API from search engine indexing, social media sharing.
- Can stop subspecies request from who is hiding his identity.
- We can add user agent along with REST API usage history.
- We can add device/OS usage restriction.

Exmaple:

| Platform | Example User Agent Like |
|---|---|
| Android web browser | Mozilla/5.0 (Linux; Android 6.0.1; RedMi Note 5 Build/RB3N5C; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/68.0.3440.91 Mobile Safari/537.36 |
| IOS web browser | Mozilla/5.0 (iPhone; CPU iPhone OS 12_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1 Mobile/15E148 Safari/604.1 |
| Windows | Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36 |
| Mac | Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1.2 Safari/605.1.15 |
| Google BOT | Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html) |
| Facebook BOT | facebookexternalhit/1.0 (+http://www.facebook.com/externalhit_uatext.php) |

5) Bearer Authentication/ Auth 2.0 :

= Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens, passes through request-response header. In General JSON Web Tokens JWT used for this purposes.

| Language | Platform | Library name | Library Sources |
|---|---|---|---|
| C# | ASP.NET | JwtBearer, jose-jwt | Nuget package manager |
| PHP | Laravel | firebase / php-jwt | GitHub |
| JS | Node/Express JS | npm i jsonwebtoken | NPM |

6) JWT (JSON WEB TOKEN):

- Compact and self-contained way for securely transmitting information between parties as a JSON object.
- Information can be verified and trusted because it is digitally signed.

**USES:**

Authorization: Allowing the user to access routes, services, and resources

Information Exchange: Way of securely transmitting information between parties.

7) JSON WEB TOKEN STRUCTURE :

= Header, Payload, Signature

8) JSON WEB TOKEN STRUCTURE :

- Type of the token
- Signing algorithm

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

9) JSON WEB TOKEN PAYLOAD:

Registered claims: iss (issuer), exp (expiration time), sub (subject), aud (audience)

Public claims: These can be defined at will by those using JWTs.

Private claims: These are the custom claims created to share information between parties.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022,
  "exp":1576239022
}
```

10) JSON WEB TOKEN SIGNATURE:

To create the signature part -

- Take the encoded header
- Take the encoded payload, a secret
- The algorithm specified in the header


# Web Back-End Development

1: What is typeof operator?

2: Explain Null and Undefined in JavaScript

3: When should we use generators in ES6?

4: Explain equality in JavaScript

5: What is Scope in JavaScript?

6: Explain what is Linear (Sequential) Search and when may we use one?

7: Explain Values and Types in JavaScript

8: What is let keyword in JavaScript?

9: Explain what is Binary Search

10: Explain the same-origin policy with regards to JavaScript.

11: What is the difference between == and ===?

12: Is there anyway to force using strict mode in Node.js?

13: What is export default in JavaScript?

14: What is the new keyword in JavaScript?

15: Explain Prototype Inheritance in JavaScript?

16: How does concurrency work in Node.js?

17: What is a Blocking Code in Node.js?

18:What is an Aggregation Pipeline in MongoDB?

19:Where does MongoDB stand in the CAP theorem?

20:Why should you separate Express app and server?

21:What are the key features of Node.js?

22:What is the purpose of ExpressJS?

23:What is the function of Node.js?

24:What is Mongoose?

25:Define DATA modeling?

26:Highlight the difference between Node.js, AJAX, and jQuery?

27:What purpose do Indexes serve in MongoDB?

28:What is meant by “Callback” in Node.js?

29: What is the middleware used in redux?

30:What do you understand by NoSQL databases?

31:What is a Covered Query in MongoDB?

32:What is TTL Collection in MongoDB?

33:Is MongoDB schema-less?

34:What are the differences between MongoDB and MySQL?

35:How do I create a Compound Index in MongoDB?

36:What is data modeling?

37:Point the difference between NodeJS, AJAX, and jQuery?

38:What is containerization?

39:State the meaning of NPM in NodeJS?