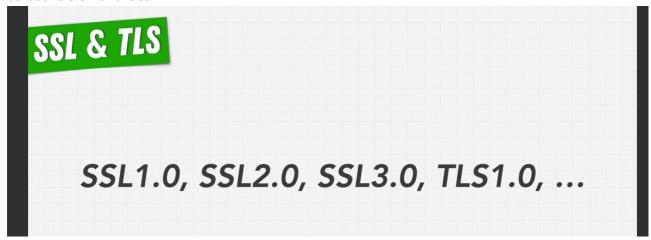
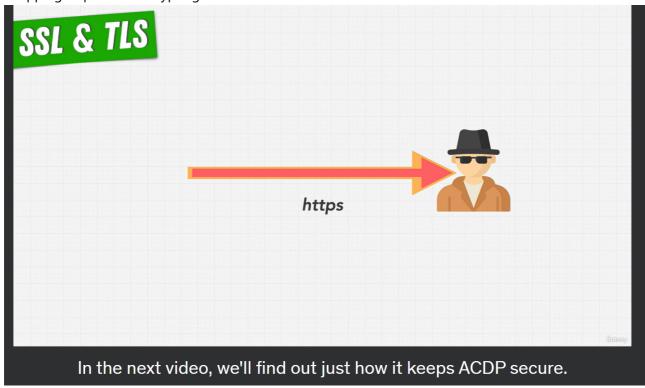
Fundamentals Node Security + Authentication

1. tls last version of the ssl



2. wrapping request tls encrypting



3. If need created ssl or tls certificate

Authentication

when need now who this user

Authorization

checks whether that user has permission to access a specific resource once they've been authenticated access control

The server could not understand the request due to invalid syntax.

401 Unauthorized

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested response.

402 Payment Required



This response code is reserved for future use. The initial aim for creating this code was using it for digital payment systems, however this status code is used very rarely and no standard conventior exists.

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. Unlike 401, the client's identity is known to the server.

404 permission 401 authenticate

Api key

It's a string

passing as either a query parameter or as a header in http request

429 Too Many Requests

The HTTP 429 Too Many Requests response status code indicates the user has sent too many requests in a given amount of time ("rate limiting").

A Retry-After header might be included to this response indicating how long to wait before making a new request.

Status

429 Too Many Requests

goole maps

Adding the API key to your request

You must include an API key with every Maps JavaScript API request. In the following example, replace YOUR_API_KEY with your API key.

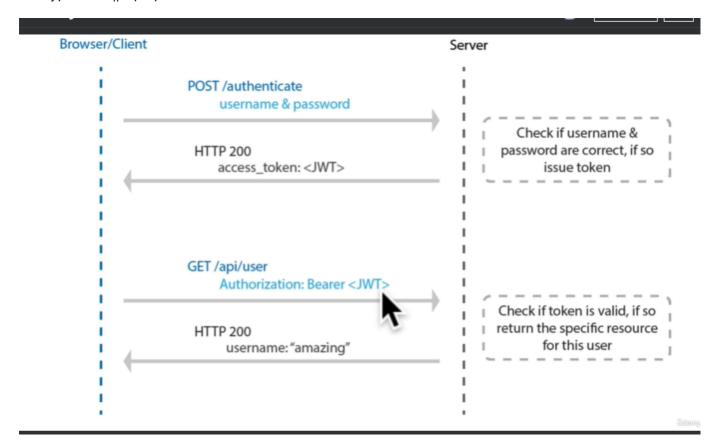
```
<script async defer src="https://maps.googleapis.com/maps/api/
type="text/javascript"></script>

com/maps/api/
```

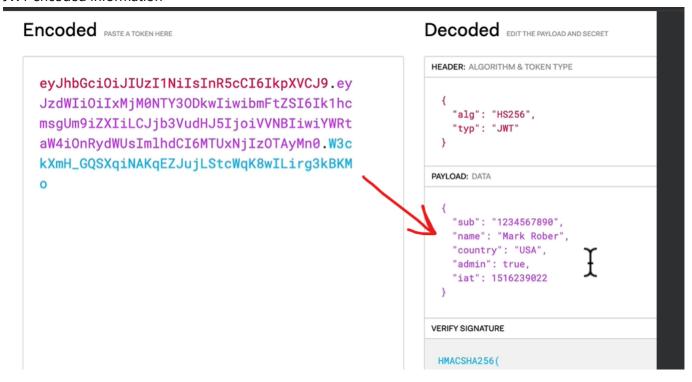
HTTPS is required for requests that use an API key, and recommended for requests that use a client ID. HTTPS is also required for applications that include sensitive user data - such as a user's location - in requests.

JSON Web token

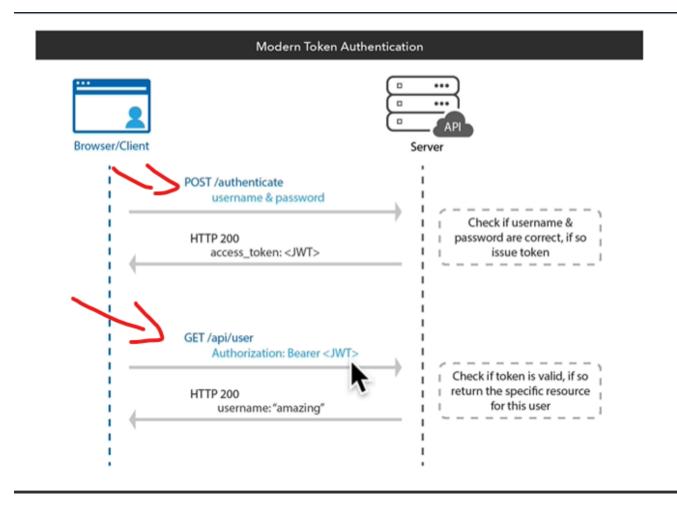
two types JWT || Opaque tokes



JWT encoded information



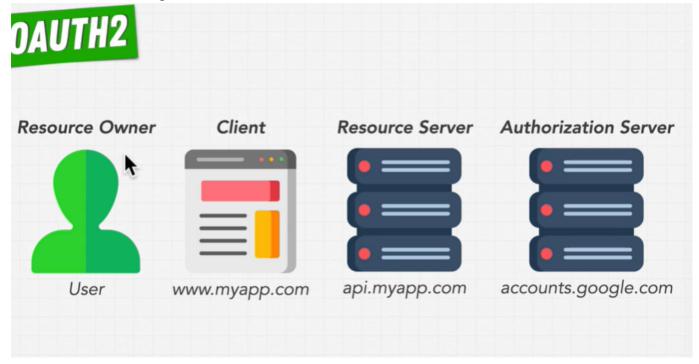
split to three part header -> alg (HS256) and type payload -> signature ->



data that's constantly being passed around from client to the server.

The OAUTH 2.0 authentication standard

resource owner -> user client -> my website resource server -> your backend in your application Authorization server -> goole service



1. has difference flows

Server-side (AKA web)

The table shows you which OAuth 2.0 flow to use for the type of application that you are building.

Type of Application	OAuth 2.0 flow
Server-side (AKA Web)	Authorization Code flow
Single- Page Application	Authorization Code flow with PKCE or Implicit flow when the SPA that you are building runs in older browsers that don't support Web Crypto for PKCE
Native	Authorization Code flow with PKCE
Trusted	Resource Owner Password flow

