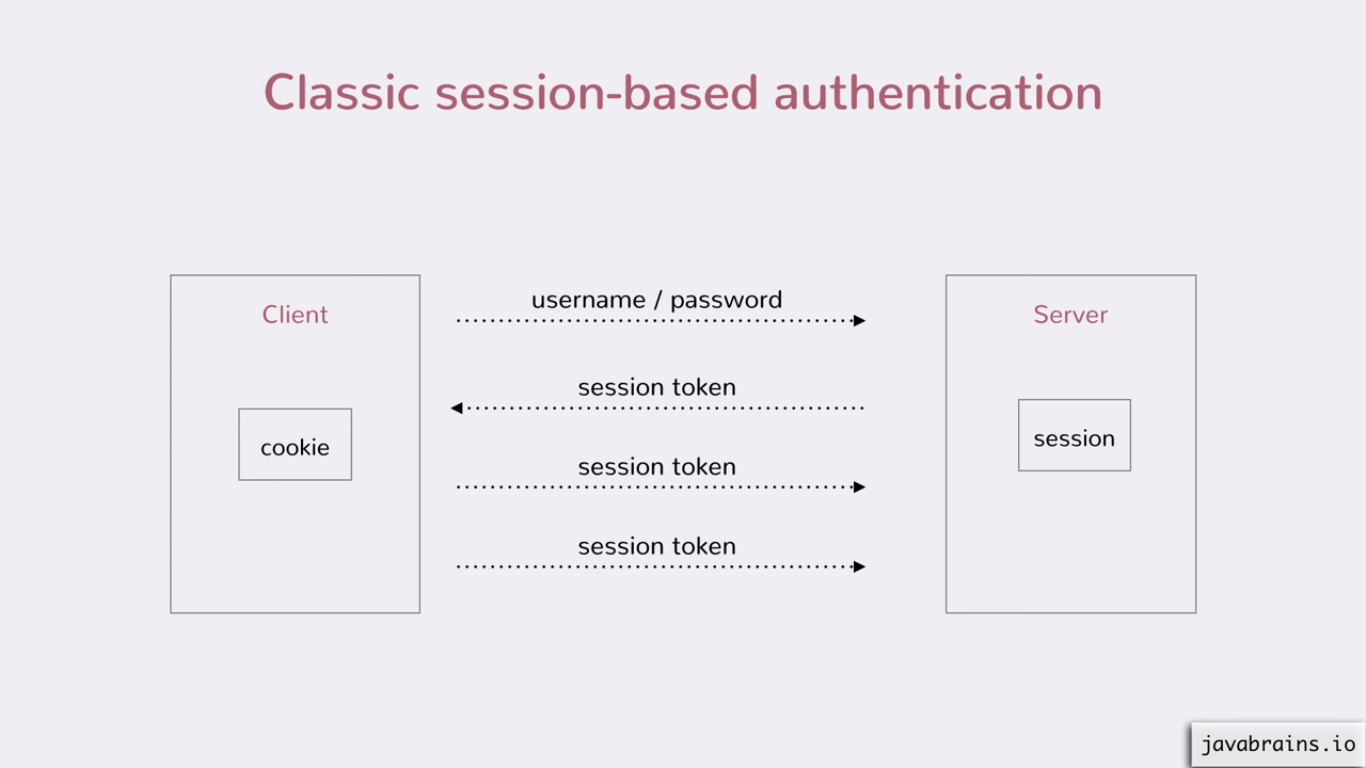
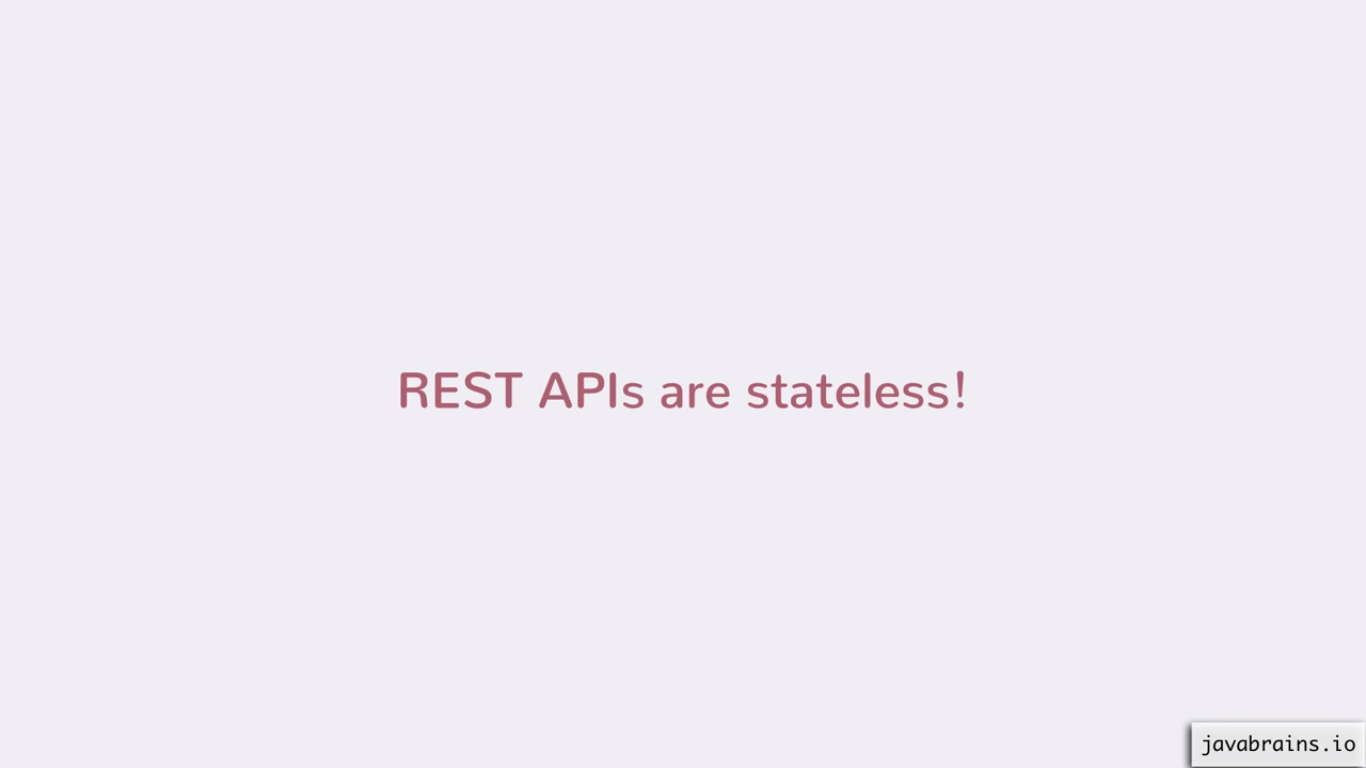


Traditionally this used to happen. Client sends username and password to server on first request. Server stores them in session object on server. Sends the session token in response which is stored in cookies object on client side. For subsequent requests to server, this session token is used. This holds good for stateful services. But REST services are stateless.





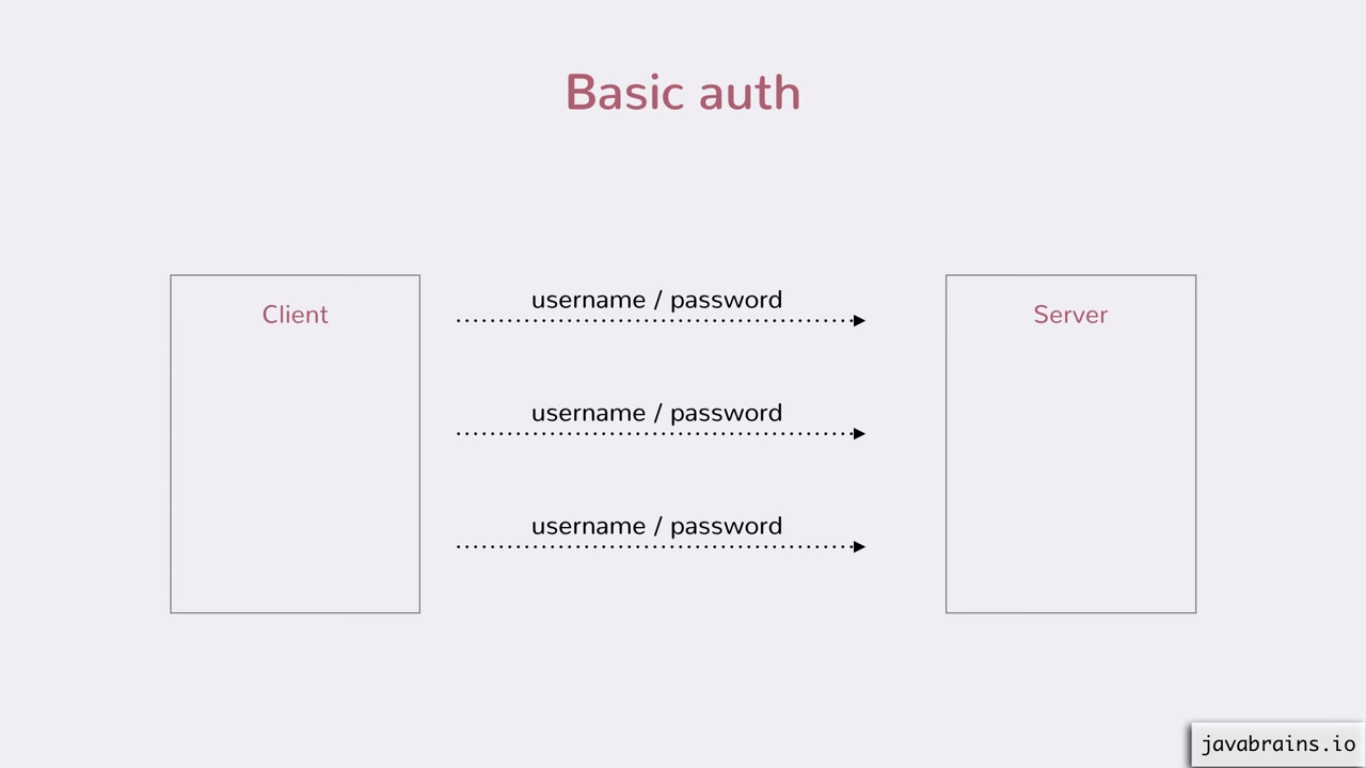
This means applications running on server does not maintain state of the users. No user information is persisted. That’s how REST is designed.

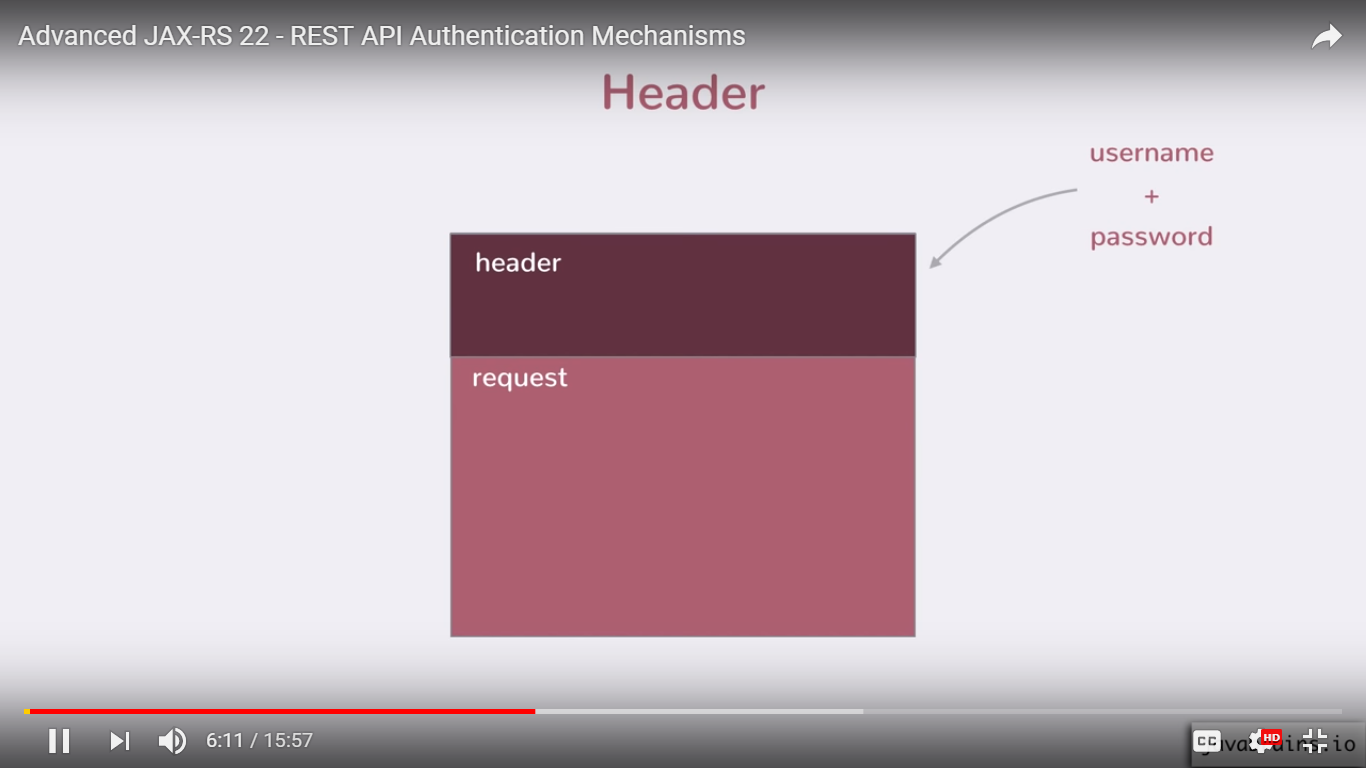
There are many ways to handle the stateless authentication for REST services.

One of them is Basic Authentication (auth) – send the username and password for each request in request headers.

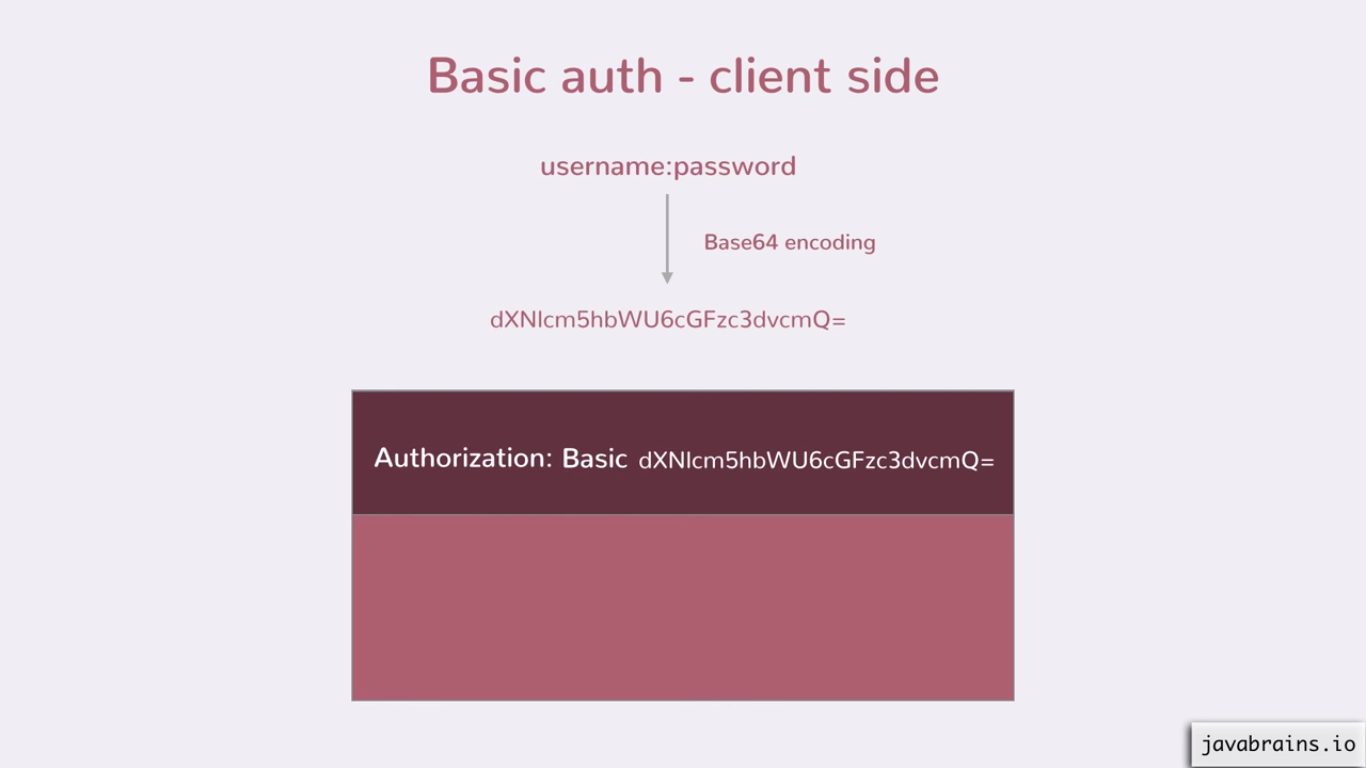
Basic authentication –

Can be implemented using filters in java.





Addition of username and password is in a protocol (in a specific way).

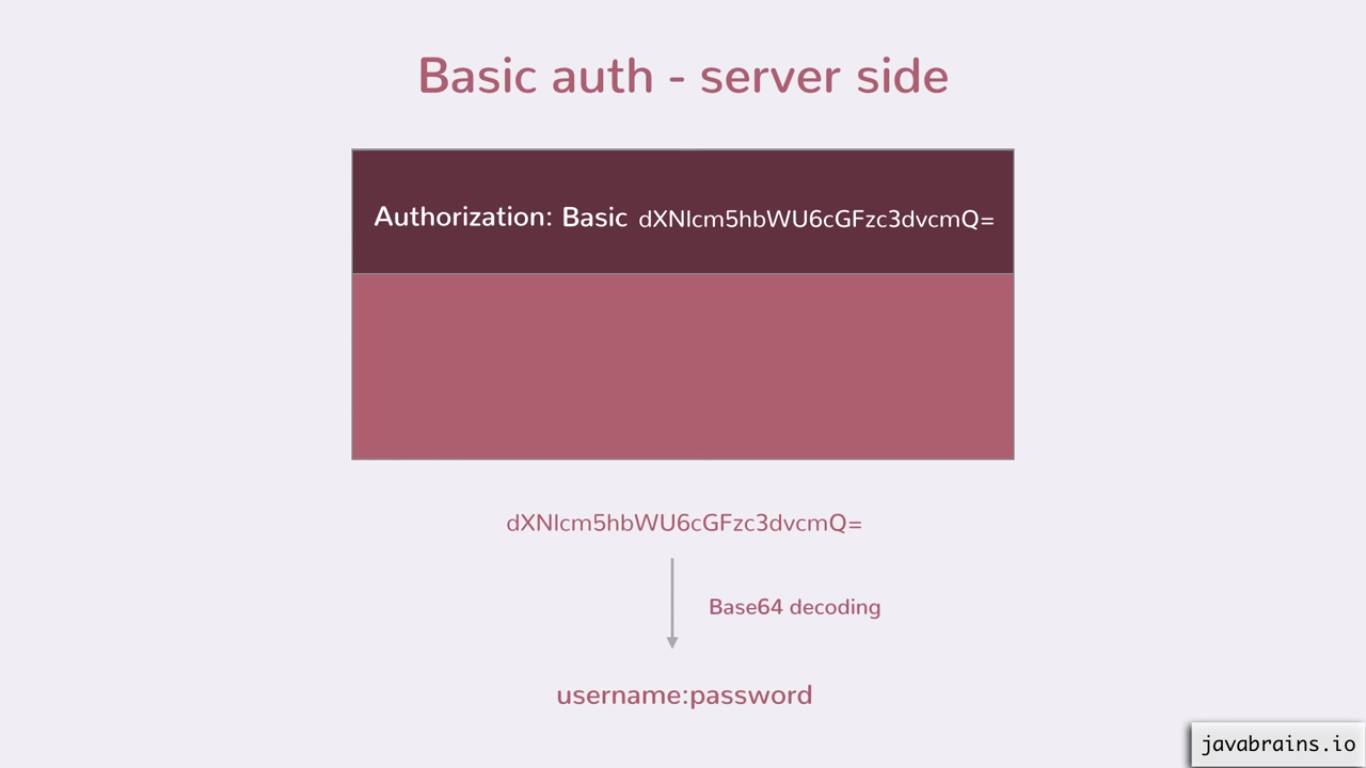


username:password – encode this string using Base64 encoding. Pass this string as a request header with key-value pair as shown above. Encoding is not equals Encrypted. It is not secure.

It looks like -

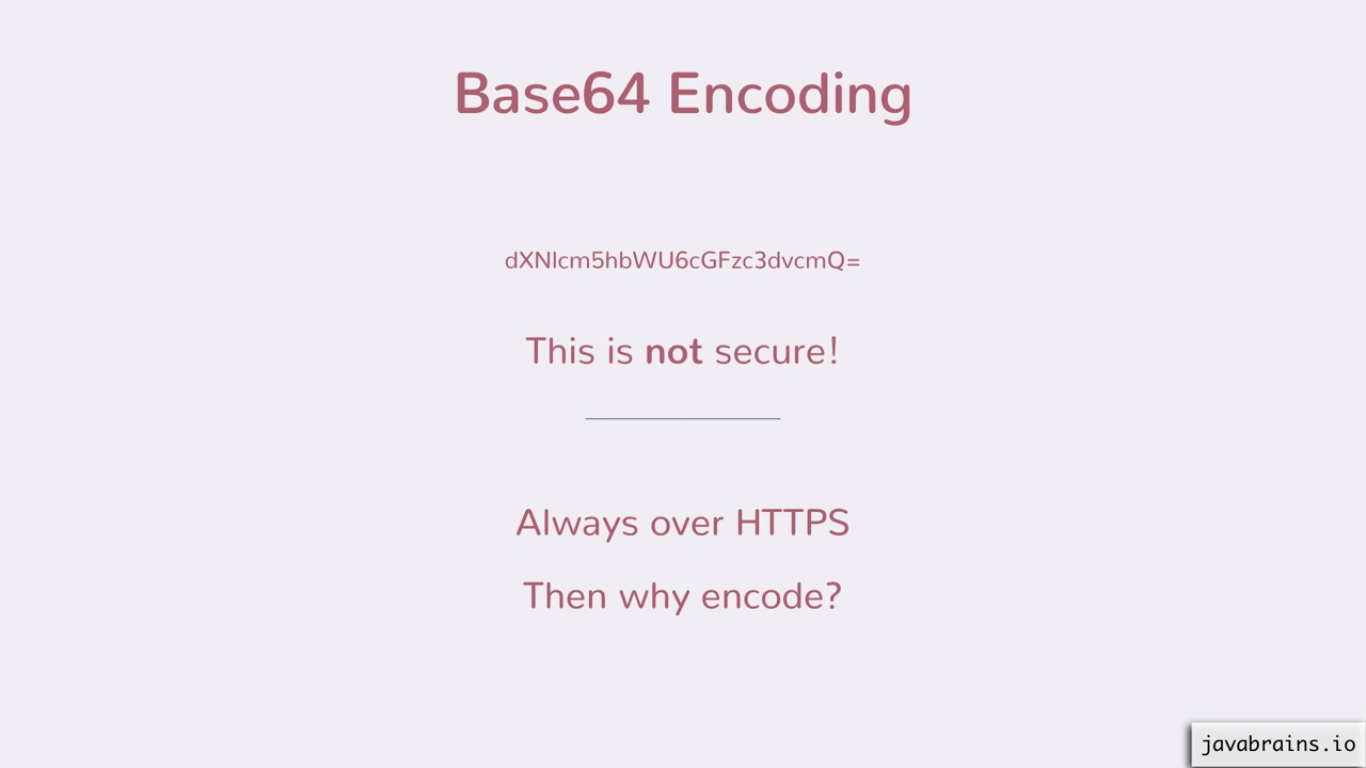
E.g. Authorization (this is key) : Basic (must add) (space) someEncodedString

On server side – it is reverse – it decodes this string –



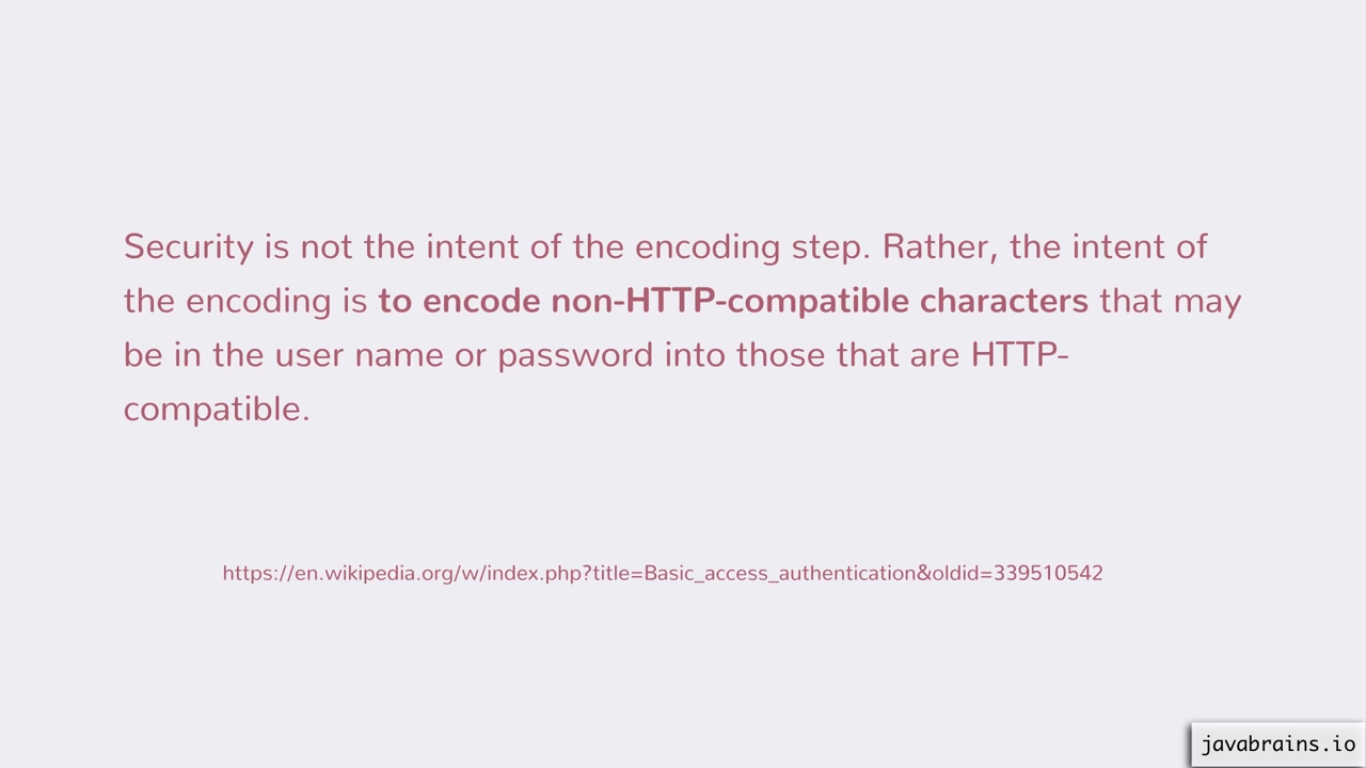
After decoding the Basic auth credentials sent by user, server should validate it.

Base64 encoded string can easily be decoded, so always send it over https protocol; as request headers are not visible in https.

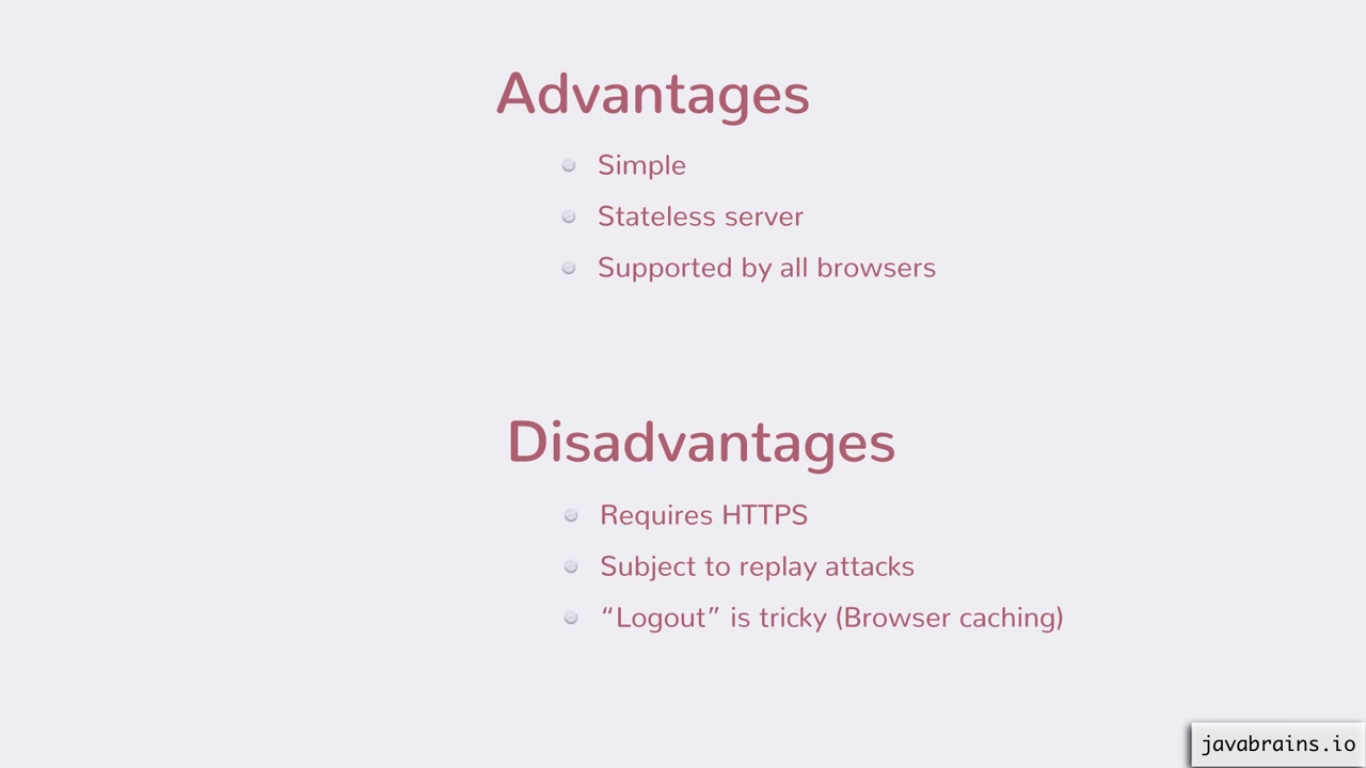


Why not send it directly without encoding?

- Encoding is just to convert/ encode few special characters (no http chars) to http compatible. It is not to hide it or to make it secure.

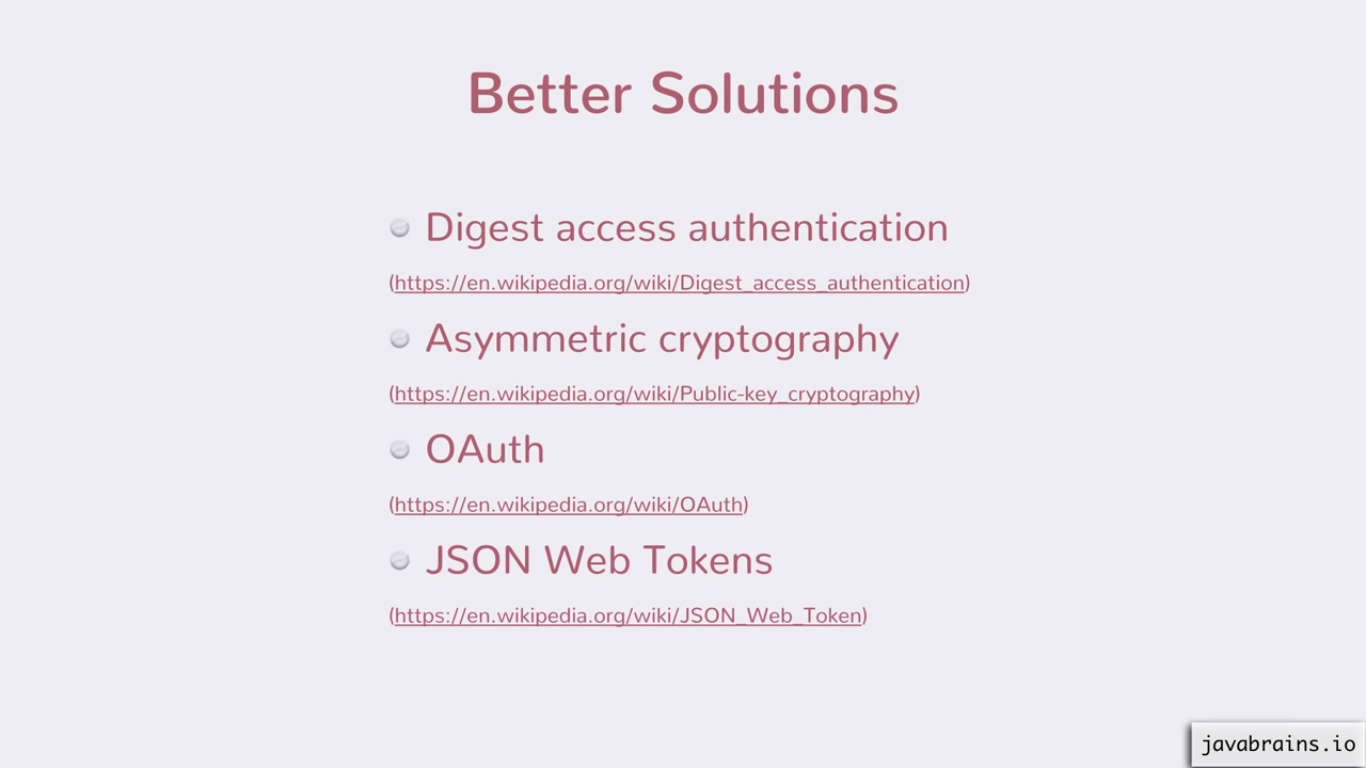


Advantages and disadvantages -



It is simple to use, quite basic, it serves the purpose of REST being stateless. And all browser support request headers.

Better Solutions -



Digest access authentication – Does encryption (secret) on top of username and password (?).

Asymmetric Cryptography – Public (made public) and private key (individual). Encrypts and decrypts username and pwd with help of it.

OAuth – OAuth 1.0 and 2.0 – different from each other

JWTs