Session vs Token Based Authentication

By Sherry Hsu

Published: 2018-07-27 · Archived: 2026-04-05 12:50:03 UTC

Why do we need session or token for authentication?

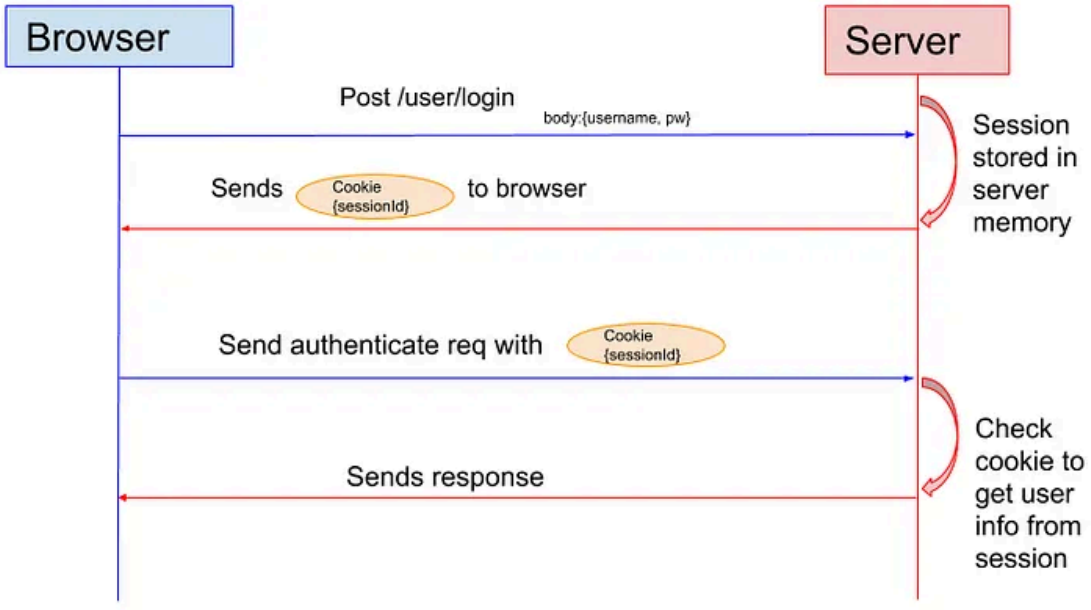
HTTP is stateless. All the requests are stateless. However, there are situations where we would like our states to be remembered. For example, in a on-line shop, after we put bananas in a shopping cart, we don't want our bananas to disappear when we go to another page to buy apples. ie. we want our purchase state to be remembered while we navigate through the on-line shop!

To overcome the stateless nature of HTTP requests, we could use either a session or a token.

Session Based Authentication

In the session based authentication, the server will create a session for the user after the user logs in. The session id is then stored on a cookie on the user's browser. While the user stays logged in, the cookie would be sent along with every subsequent request. The server can then compare the session id stored on the cookie against the session information stored in the memory to verify user's identity and sends response with the corresponding state!

Press enter or click to view image in full size

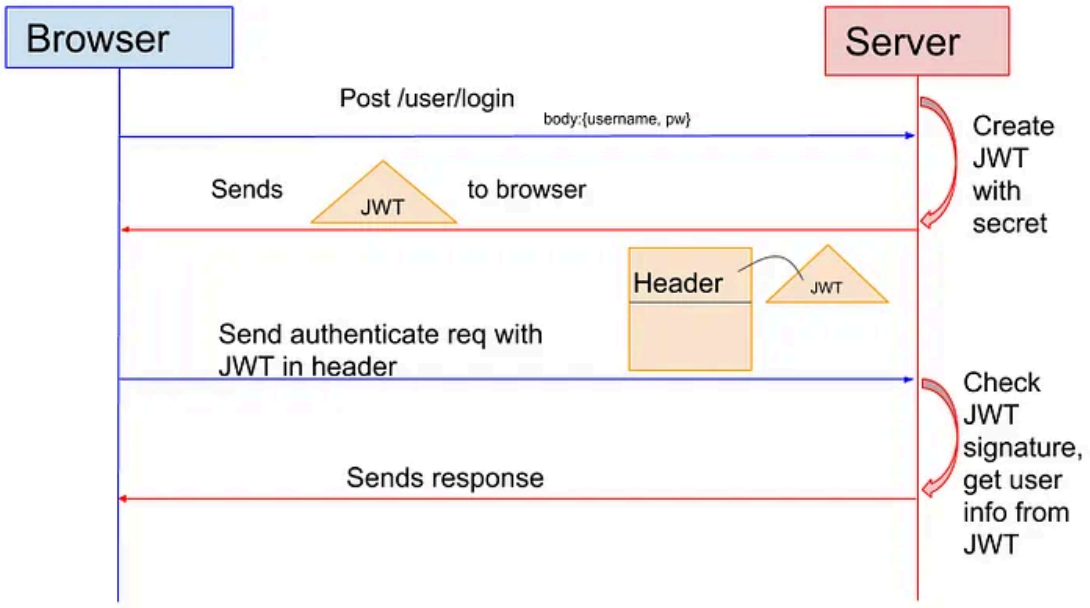


Session Based Authentication flow

Token Based Authentication

Many web applications use JSON Web Token (JWT) instead of sessions for authentication. In the token based application, the server creates JWT with a secret and sends the JWT to the client. The client stores the JWT (usually in local storage) and includes JWT in the header with every request. The server would then validate the JWT with every request from the client and sends response.

Press enter or click to view image in full size



Token Based Authentication flow

The biggest difference here is that the user's state is not stored on the server, as the state is stored inside the token on the client side instead. Most of the modern web applications use JWT for authentication for reasons including scalability and mobile device authentication.

Node Modules for JWT

`jsonwebtoken` library can be used to created the JWT token on the server. Once the user is logged in, the client passes the JWT token back on the header.authorization.bearer attribute.

```
{
  method: "GET",
  headers:{
    "Authorization": "Bearer ${JWT_TOKEN}"
  }
}
```

Middleware, `express-jwt`, can be used to validate the JWT token by comparing the secret.

Scalability

Session based authentication: Because the sessions are stored in the server's memory, scaling becomes an issue when there is a huge number of users using the system at once.

Token based authentication: There is no issue with scaling because token is stored on the client side.

Multiple Device

Session based authentication: Cookies normally work on a single domain or subdomains and they are normally disabled by browser if they work cross-domain (3rd party cookies). It poses issues when APIs are served from a different domain to mobile and web devices.

Token based authentication: There is no issue with cookies as the JWT is included in the request header.

Token Based Authentication using JWT is the more recommended method in modern web apps. One drawback with JWT is that the size of JWT is much bigger comparing with the session id stored in cookie because JWT contains more user information. Care must be taken to ensure only the necessary information is included in JWT and sensitive information should be omitted to prevent XSS security attacks.

Reference
