

Subscribe to more awesome content!

your@email.com



# Get Started with JSON Web Tokens

All you wanted to know about JSON Web Tokens but were afraid to ask.

[CONTACT US](#)

## WHAT IS JSON WEB TOKEN?

JSON Web Token (JWT) is an open standard ([RFC 7519](#)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with HMAC algorithm) or a public/private key pair using RSA.

Let's explain some concepts of this definition further.

- **Compact:** Because of its size, it can be sent through an URL, POST parameter, or inside an HTTP header. Additionally, due to its size its transfer is efficient.
- **Self-contained:** The payload contains all the relevant information, so you don't have to query the database more than once.

Hey, you're back! Things must be getting serious 😬 Do you have any product questions we can answer?

Interested in getting up-to-speed with JWTs as soon as possible?

Subscribe to more awesome content! your@email.com



DOWNLOAD THE FREE EBOOK



## WHEN SHOULD YOU USE JSON WEB TOKENS?

These are some scenarios where JSON Web Tokens are useful:

- **Authentication:** This is the typical scenario for using JWT, once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used among systems of different domains.
- **Information Exchange:** JWTs are a good way of securely transmitting information between parties, because as they can be signed, for example using a public/private key pair, you can be sure that the sender is who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't changed.

## WHICH IS THE JSON WEB TOKEN STRUCTURE?

JWTs consist of three parts separated by dots ( . )

- Header
- Payload

Hey, you're back! Things must be getting serious 😏 Do you have any product questions we can answer?

1

- Signature

Subscribe to more awesome content!

your@email.com



Therefore, a JWT typically looks like the following.

```
xxxxx.yyyyy.zzzzz
```

Let's break down the different parts.

## Header

The header **typically** consists of two parts: the type of the token, which is JWT, and the hashing algorithm such as HMAC SHA256 or RSA.

For example:

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

Then, this JSON is **Base64Url** encoded to form the first part of the JWT.

## Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional metadata. There are three types of claims: **reserved**, **public**, and **private** claims.

- **Reserved claims:** These are a set of predefined claims, which are not mandatory but recommended, thought to provide a set of useful, interoperable claims. Some of them are: **iss** (issuer), **exp** (expiration time), **sub** (subject), **aud** (audience), among others.

*Notice that the claim names are only three characters long as JWT is meant to be compact.*

- **Public claims:** These can be defined at will by the issuer, but they should be defined in the IANA JSON Web Token Registry or in a collision resistant namespace.

Hey, you're back! Things must be getting serious 😬 Do you have any product questions we can answer?

Is there a 100% solution?

- Private claims: These are the custom claims created to share information between parties that agree on using them.

Subscribe to more awesome content!

your@email.com



An example of payload could be:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

The payload is then **Base64Url** encoded to form the second part of the JWT.

## Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

The signature is used to verify that the sender of the JWT is who it says it is and to ensure that the message wasn't changed in the way.

## Putting all together

The output is three Base64 strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact compared to JSON.

The following shows a JWT that has the previous payload and a secret.

Hey, you're back! Things must be getting serious 😊 Do you have any product questions we can answer?

Subscribe to more awesome content!

your@email.com



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaXNTb2NpYWwiOnRydWV9.

4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4

You can browse to [jwt.io](https://jwt.io) where you can play with a JWT and put these concepts in practice. [jwt.io](https://jwt.io) allows you to decode, verify and generate JWT.

## HOW JSON WEB TOKENS WORK?

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required.

You also [should not store sensitive session data in browser storage due to lack of security](#).

Whenever the user wants to access a protected route, it should send the JWT, typically in the **Authorization** header using the **Bearer** schema. Therefore the content of the header should look like the following.

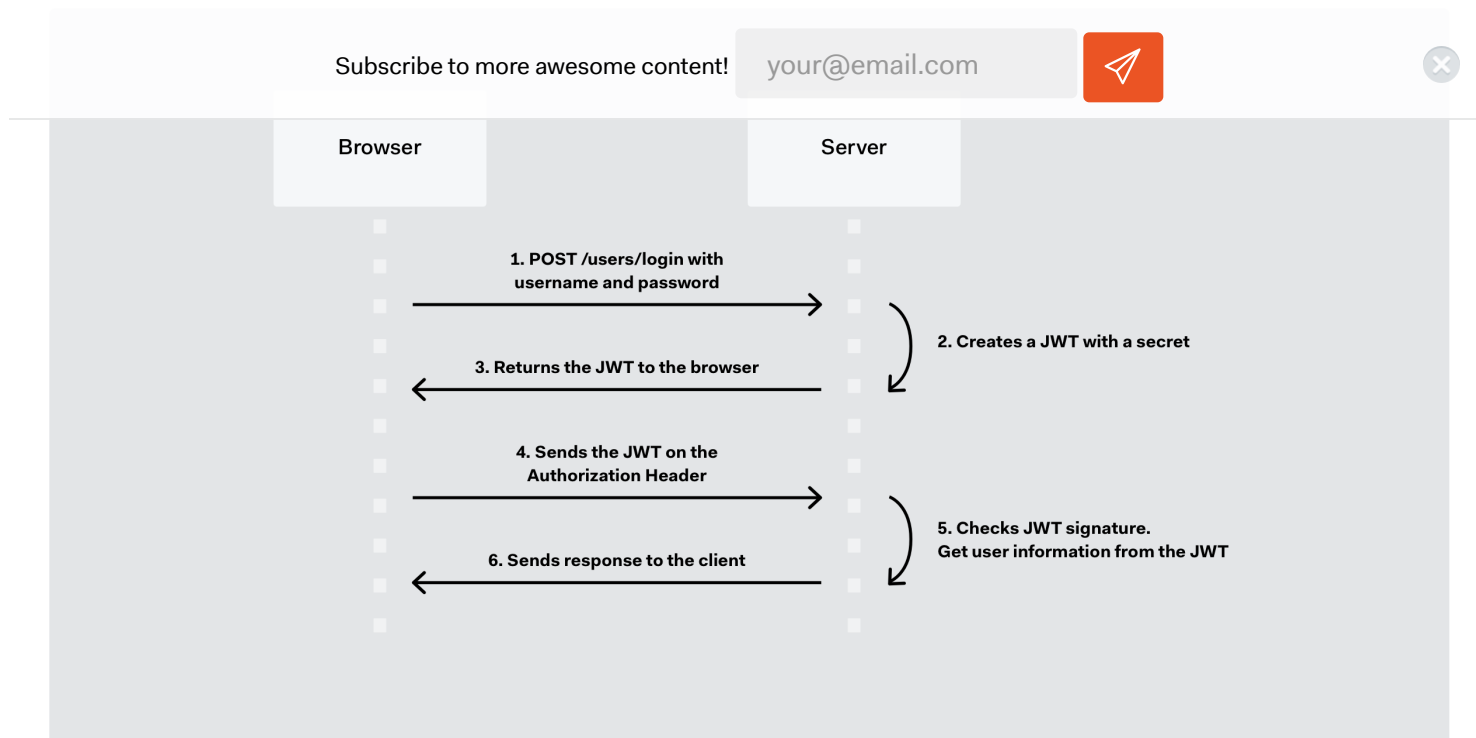
```
Authorization: Bearer <token>
```

This is a stateless authentication mechanism as the user state is never saved in the server memory. The server's protected routes will check for a valid JWT in the Authorization header, and if there is, the user will be allowed. As JWTs are self-contained, all the necessary information is there, reducing the need of going back and forward to the database.

This allows to fully rely on data APIs that are stateless and even make requests to downstream services. It doesn't matter which domains are serving your APIs, as Cross-Origin Resource Sharing (CORS) won't be an issue as it doesn't use cookies.

Hey, you're back! Things must be getting serious 😊 Do you have any product questions we can answer?

1



## WHY SHOULD YOU USE JSON WEB TOKENS?

Let's talk about the benefits of **JSON Web Tokens (JWT)** comparing it to **Simple Web Tokens (SWT)** and **Security Assertion Markup Language Tokens (SAML)**.

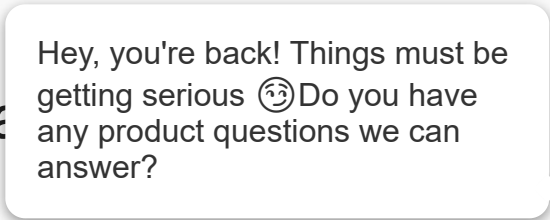
As JSON is less verbose than XML, when it is encoded its size is also smaller; making JWT more compact than SAML. This makes JWT a good choice to be passed in HTML and HTTP environments.

Security-wise, SWT can only be symmetric signed by a shared secret using the HMAC algorithm. While JWT and SAML tokens can also use a public/private key pair in the form of a X.509 certificate to sign them. However, signing XML with XML Digital Signature without introducing obscure security holes is very difficult compared to the simplicity of signing JSON.

JSON parsers are common in most programming languages, because they map directly to objects, conversely XML doesn't have a natural document-to-object mapping. This makes it easier to work with JWT than SAML assertions.

Regarding usage, JWT is used at an Internet scale of JWTs on multiple platforms, especially, mobile.


Hey, you're back! Things must be getting serious 😊 Do you have any product questions we can answer?




Name

Subscribe to more awesome content!

your@email.com





Company

Email

Role

Software Developer ▼

Message

SEND



## PRODUCT

Pricing

Why Auth0

How It Works

## COMPANY

Hey, you're back! Things must be getting serious 😬 Do you have any product questions we can answer?

1



[About Us](#)

Subscribe to more awesome content!

[Blog](#)[Jobs](#)[Press](#)

## SECURITY

[Availability & Trust](#)[Security](#)[White Hat](#)

## LEARN

[Help & Support](#)[Documentation](#)[Open Source](#)

## EXTEND

[Lock](#)[WordPress](#)[API Explorer](#)

## CONTACT

10800 NE 8th Street  
Suite 600  
Bellevue, WA 98004

Hey, you're back! Things must be getting serious 😬 Do you have any product questions we can answer?

888) 235-2699  
1  
42 521

SUPPORT

Follow @auth0

23K followers



Follow

27,422

Subscribe to more awesome content!

your@email.com



+1 (425) 559-9554



Like 14K

[Privacy Policy](#) [Terms of Service](#) © 2013-2016 Auth0® Inc. All Rights Reserved.

Hey, you're back! Things must be getting serious 😏 Do you have any product questions we can answer?

