Json Web Tokens
• What is JSON Web Token?
• When should you use JSON Web Tokens?
• What is the JSON Web Token structure?
• How do JSON Web Tokens work?
• Why should we use JSON Web Tokens?
What is JSON Web Token?

- An open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
  
  - Compact: Because of its smaller size, JWTs can be sent through an URL, POST parameter, or inside an HTTP header.
  
  - Self-contained: The payload contains all the required information about the user, avoiding the need to query the database more than once.
When should you use JSON Web Tokens?

- Authentication: 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.

- Because of its small overhead it can be used across different domains (single signon).

- Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties, because as they can be signed.

- as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.
What is the JSON Web Token structure?

- Three parts separated by dots (.), which are:
  - Header
  - Payload
  - Signature

- A JWT typically looks like the following.
  - xxxxx.yyyyy.zzzzz
JWT Structure: Header

- Typically consists of two parts:
  - hashing algorithm being used, such as HMAC SHA256 or RSA.
  - type of the token, which is JWT,

- This JSON is Base64Url encoded to form the first part of the JWT.

```json
{
  "alg": "HS256",
  "typ": "JWT"
}
```
JWT Structure: Payload

- Payload contains the claims - statements about an entity (typically, the user) and additional metadata. There are three types of claims:

  - **Reserved claims**: A set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Examples: `iss` (issuer), `exp` (expiration time), `sub` (subject), `aud` (audience).

  - **Public claims**: These can be defined at will by those using JWTs. But to avoid collisions they should be defined in the IANA JSON Web Token Registry or be defined as a URI that contains a collision resistant namespace.

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

```json
{
    "sub": "1234567890",
    "name": "John Doe",
    "admin": true
}
```
JWT Structure: Signature

- Take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign it.

- 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 along the way.

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

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

- The output is three Base64 strings separated by dots that can be easily passed in HTML and HTTP environments,

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

```text
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4g
grRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```
Encoded

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImltYWdlLmF1ZGxhbiIsImV4cCI6MTUxMDY1MjIyMCwiYXVkIjoiQ29tcHJvbnRMaXN0b3IifQ.JVA95OyvpaG4gRG90REU4iOnRyD4W9.TJVA95OR7E2cBab30RMHrHDcEfxj0YfGhQ

Decoded

HEADER:

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

PAYLOAD:

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

VERIFY SIGNATURE

```javascript
HMACSHA256(
  base64UrlEncode(header) + "," +
  base64UrlEncode(payload),
  secret
) == secret base64 encoded
```

_signature verified_
Another Example

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

Header

```json
{
  "iss":"http://trustyapp.com/",
  "exp": 1300819380,
  "sub": "users/8983462",
  "scope": "self api/buy"
}
```

Body (‘Claims’)

Cryptographic Signature

`:8H...μ€U...SZu^-nîv+~O%à—™tβ`
The Claims

{
  "iss": "http://trustyapp.com/",
  "exp": 1300819380,
  "sub": "users/8983462",
  "scope": "self api/buy"
}
How do JSON Web Tokens work?

- When the user successfully logs in using their credentials, a JSON Web Token will be returned and must be saved locally, perhaps in local storage in a browser.

- If user wants to access a protected route or resource, the JWT is sent, typically in the Authorization header using the Bearer schema

Authorization: Bearer <token>
Stateless APIs

- The server's protected routes will check for a valid JWT in the Authorization header, and if it's present, the user will be allowed to access protected resources.

- Token contains all the necessary information.

- Token may even make requests to downstream services.

**Stateless APIs**
Why should we use JSON Web Tokens?

- **Compact**: Less verbose than XML, more compact than Security Assertion Markup Language Tokens (SAML).

- **Security**: JWT tokens can use a public/private key pair in the form of a X.509 certificate for signing. Signing XML can introducing obscure security holes compared to the simplicity of signing JSON.

- **Convenience**: 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.