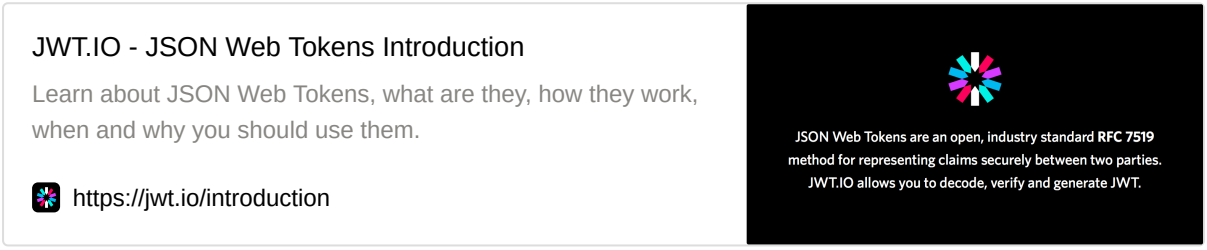
17강 - JWT 구조 이해



JWT의 구조

- `Header`
- `Payload`
- `Signature`
- `xxxxx.yyyyy.zzzzz`
  - xx부분은 Header
  - yy부분은 `Payload`
  - zz부분은 `Signature`

헤더

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

- 사용 알고리즘
- 타입 : `JWT`

- 그런 다음 이 JSON은 **Base64Url로** 인코딩되어 `JWT`의 첫 번째 부분을 형성합니다.

Base64Url

- 암호화와 복호화가 가능한 방식

- cf) 해싱은 암호화는 가능하지만 복호화는 불가능하다.

Payload

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

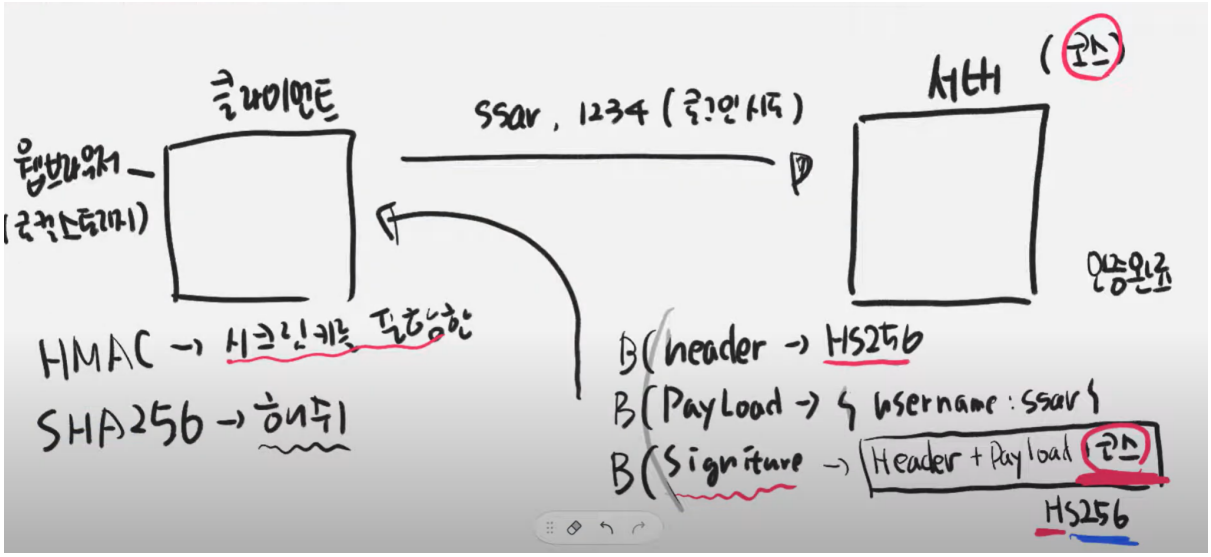
- 등록된 클레임, 개인 클레임 뭐 이런게 있는데 일단 요구사항이라 생각하면 된다.

서명

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

- 서명은 도중에 메시지가 변경되지 않았는지 확인하는 데 사용되며
- 개인 키로 서명된 토큰의 경우 `JWT` 발신자가 누구인지 확인할 수도 있습니다.
- `secret`은 서버만 알고 있는 개인키

JWT가 적용된 서버에서 로그인 시도



- 클라이언트가 로그인 시도를 했는데 인증이 완료가 되었다.
- 서버는 더이상 세션을 만들지 않고 `JWT`를 만든다

서버는 JWT 토큰을 만든다. - secret : 코스

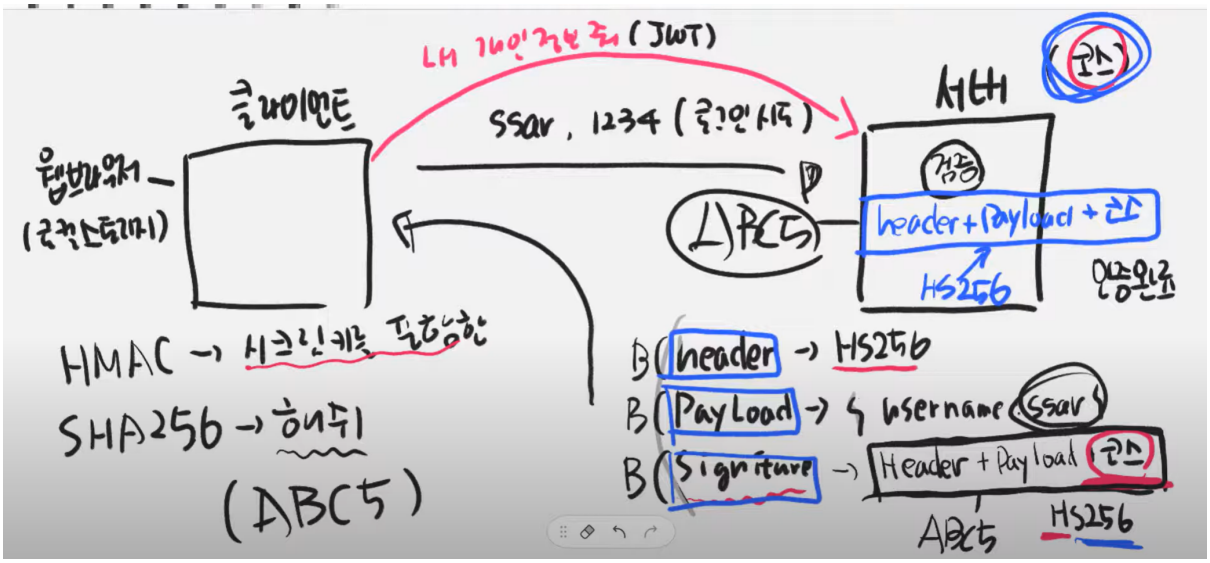
- `header : HS256`
- `Paylad : {username : ssar}`
- `Signature :`
  - `HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), 코스 )`

- 만든 `JWT`를 클라이언트에 전달한다.
- 클라이언트는 웹 브라우저의 **로컬 스토리지**에 저장한다.

HMACSHA256 간략 설명

- `secret key`를 포함한 해쉬 방식

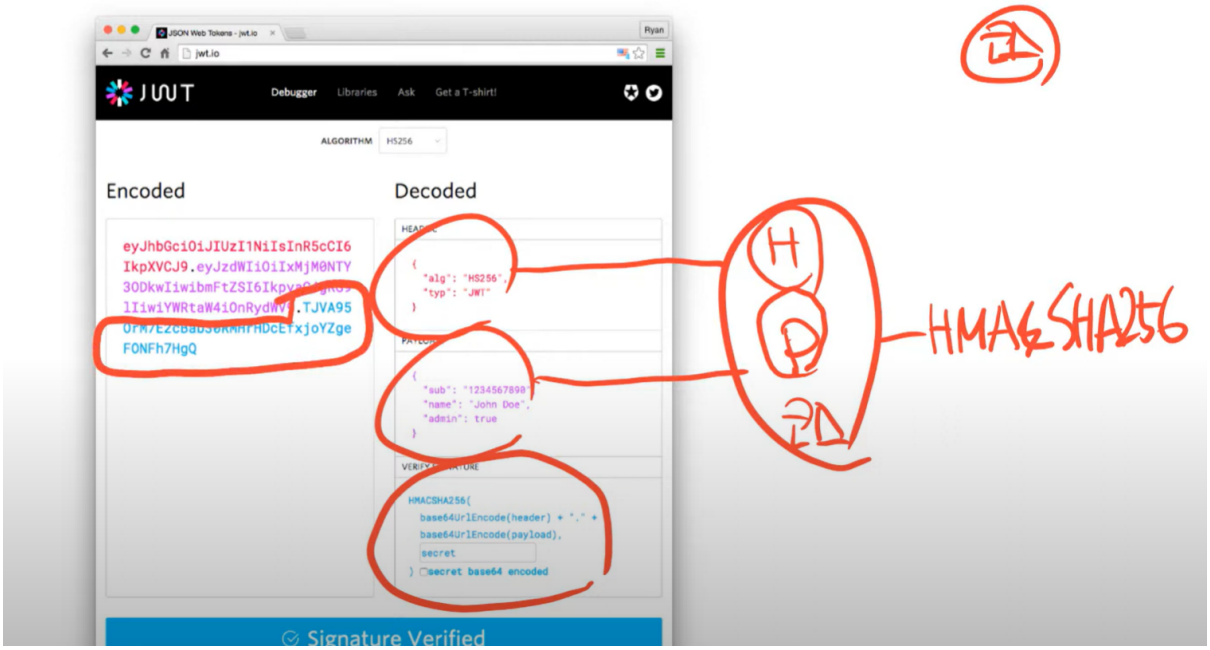
로그인 완료 후 개인정보 요청



- 클라이언트는 `JWT`를 들고 요청을 한다.
  - `JWT : ABC5`
- 서버는 검증을 해야 한다.
  - 서명 부분만 확인해보면 된다.

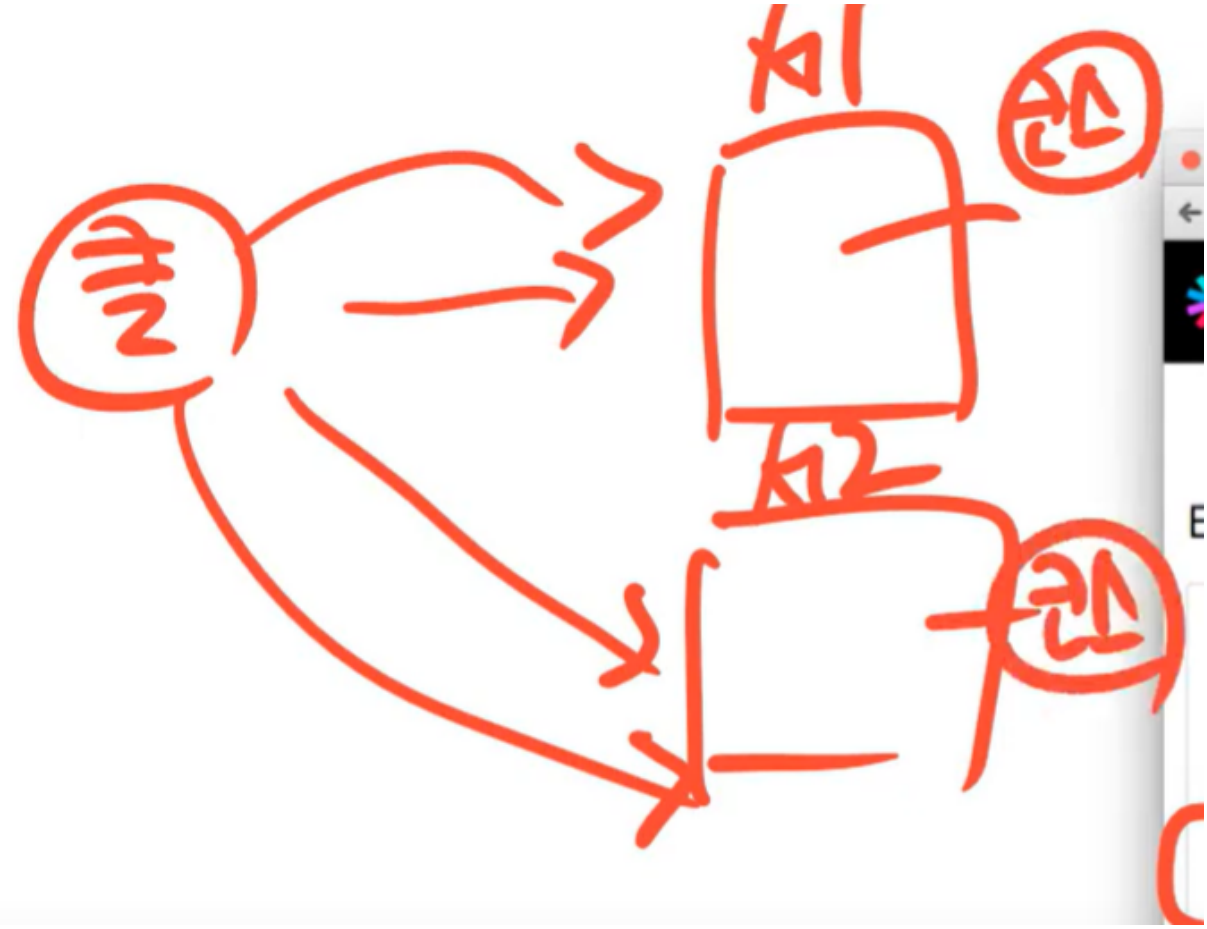
    - 헤더와 페이로드와 시크릿 키로 `HS256` 암호화를 했을 때 `ABC5`가 나왔다면
    - 인증에 성공한 것이다.
- 이때는 페이로드의 `username`으로 조회해서 클라이언트에 응답해준다.

다시 설명



- 인코딩된 `JWT`는 `Base64Url`로 암호화되어있기 때문에 복호화가 가능하다.
- 오른쪽으로 디코딩할 수 있다.
- 디코딩된 **헤더**와 **페이로드**와 서버가 알고 있는 `secret key`로 `HMACSHA256`으로 암호화를 해봤을 때
- 왼쪽에 파랑 부분과 동일하다면! 인증이 성공한 것이다.

이제 무얼 할 수 있냐면



- 이제 더이상 세션을 안 만들기 때문에
- 각각의 서버들은 `secret key`만 알고 있으면 `JWT` 검증을 할 수 있다.