因为http协议是无状态的，所以不能进行身份验证，就出现的cookie等身份验证方式

cookie

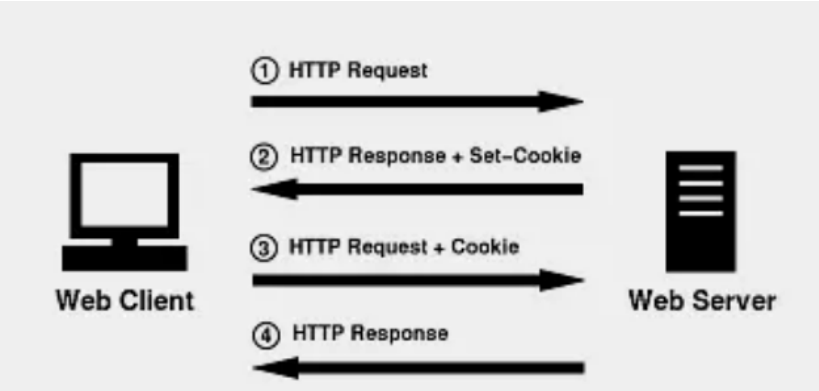
信息存储在浏览器上，通过访问cookie 来设置cookile 服务器响应时会设置cookie 的信息头，浏览器再次传输数据会带上设置的cookie用于信息校验，确认身份

```
app.post('/login', (req, res) => {
  // maxAge表示cookie存在的有效时长，单位毫秒
  // httpOnly:true表示 前端不能够通过js代码document.cookie去获取cookie的值
  res.cookie('username', 'rose', {maxAge: 10000,httpOnly:true}).send('cookie set');
//Sets name = express
})
```

session

和cookie基本一样 基于cookie 设置的头信息是加密的 传输的是一个sid 信息存储在服务端，

- 在客户端发起的第二次请求，假如服务器给了set-Cookie，浏览器会自动在请求头中添加cookie
- 服务器接收请求，分解cookie，验证信息，核对成功后返回response给客户端



token

token分为三个部分

- **Header（头部）**
- **Payload（负载）**
- **Signature（签名）**

-

只有 中间的部分是添加的用户信息 整个过程是加密的

客户端第二次送数据会带上这个tuken，当然tuken的数据是存储在浏览器上的，并且整个数据是加密的

第二次发送数据要加Bearer

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

此后，客户端每次与服务器通信，都要带上这个 JWT。你可以把它放在 Cookie 里面自动发送，但是这样不能跨域，所以更好的做法是放在 HTTP 请求的头信息Authorization字段里面。