Authorizing Access to the API

Kevin Dockx

@KevinDockx | http://blog.kevindockx.com/

Client Credentials Flow

Machine to machine communication

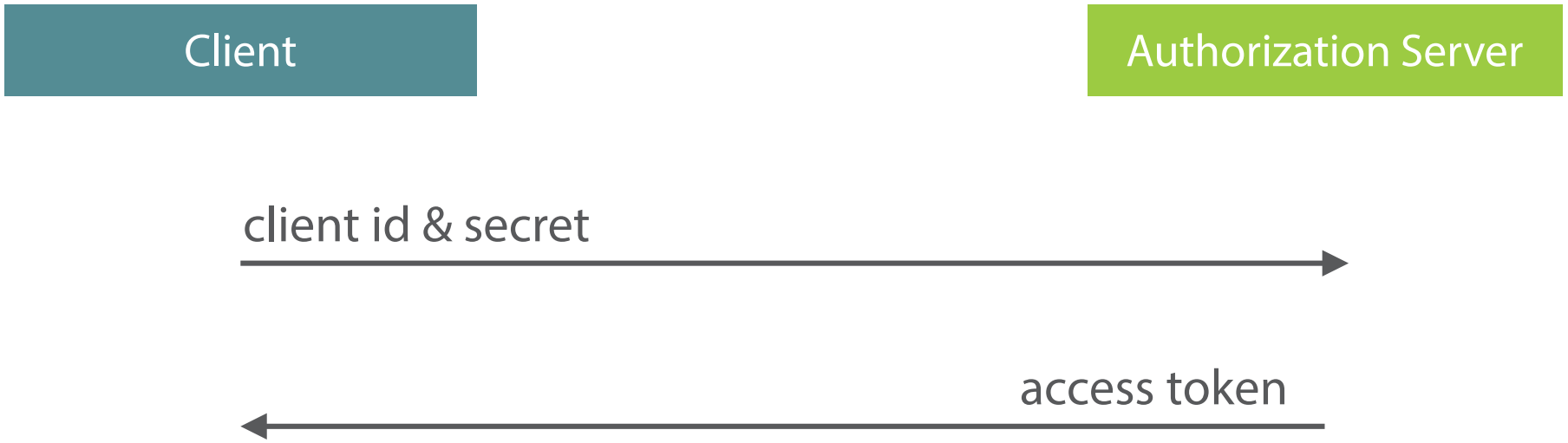
No human involved: no username/password

Can be used to obtain access tokens using client credentials

Must only be used by **confidential** clients

(because a public client cannot safely store the client secret)

Client Credentials Flow



Client Credentials Flow

Learn how to use the Client Credentials flow from ASP .NET MVC

Can't We Use This in an Angular Application?

The thing with security is that a lot of approaches will work, but most are not a good idea

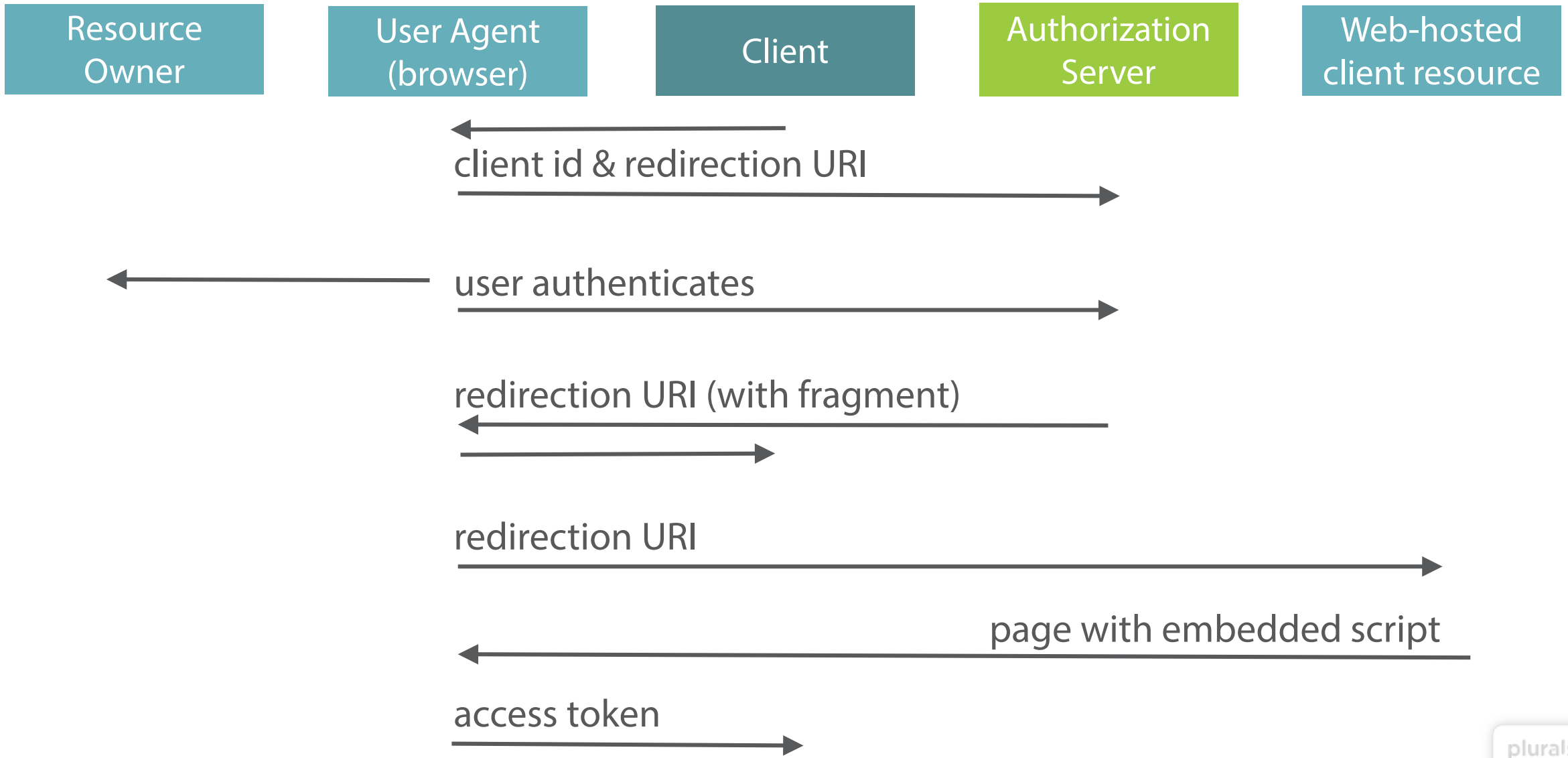
Implicit Flow

Optimized for **public** clients at predefined redirection uri
(but it can be used by confidential clients as well)

Can be used to obtain access tokens
… but no refresh tokens

No client authentication
(a public client can't safely store the secret anyway)

Implicit Flow



Implicit Flow

Learn how to use the Implicit Flow from Angular

Automatically Adding an Authorization Header in Angular

Learn how to improve on the previous demo by using an interceptor

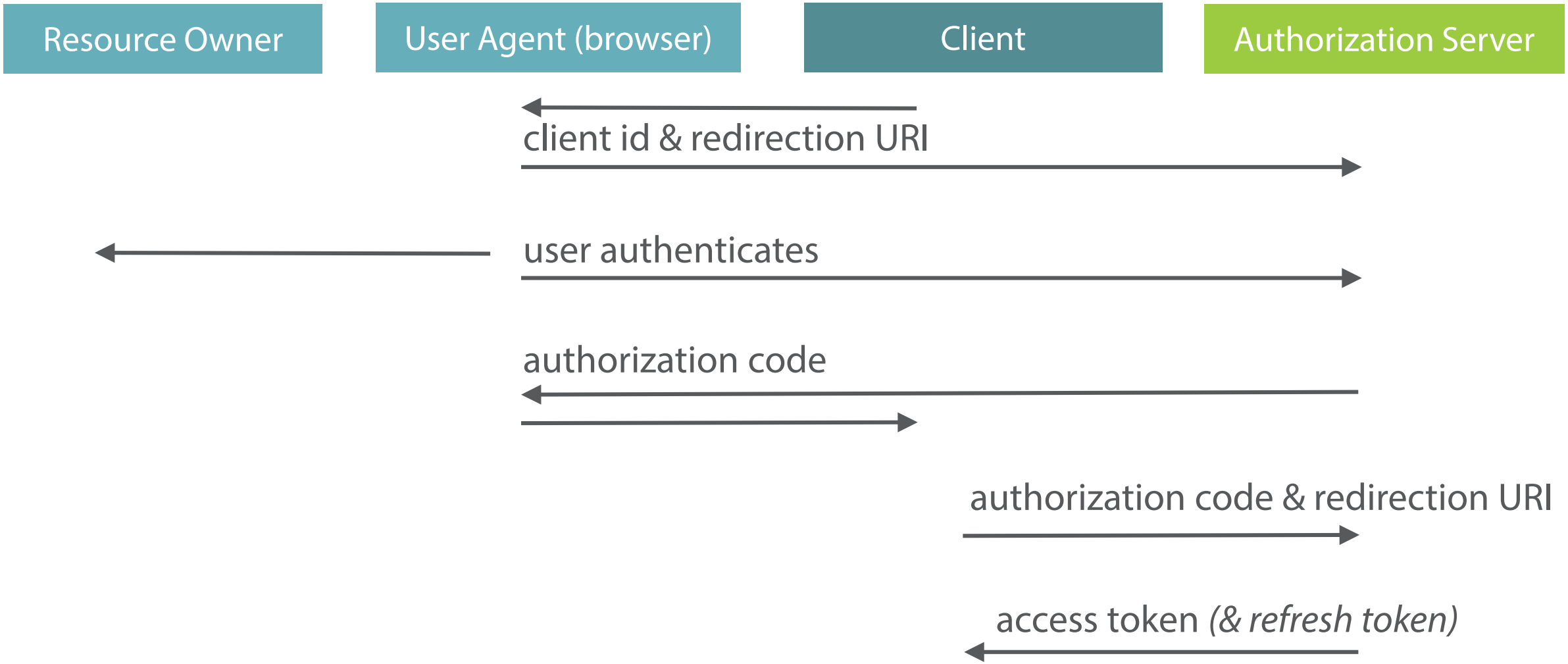
Authorization Code Flow

Optimized for **confidential** clients
(but it can be used by public clients as well)

Can be used to obtain access tokens and refresh tokens

Includes a client authentication step

Authorization Code Flow



Authorization Code Flow

Learn how to use the Authorization Code Flow from ASP .NET MVC

Resource Owner Password Credentials Flow

The client must be capable of obtaining the **resource owner's credentials**
(typically through an in-app login screen)

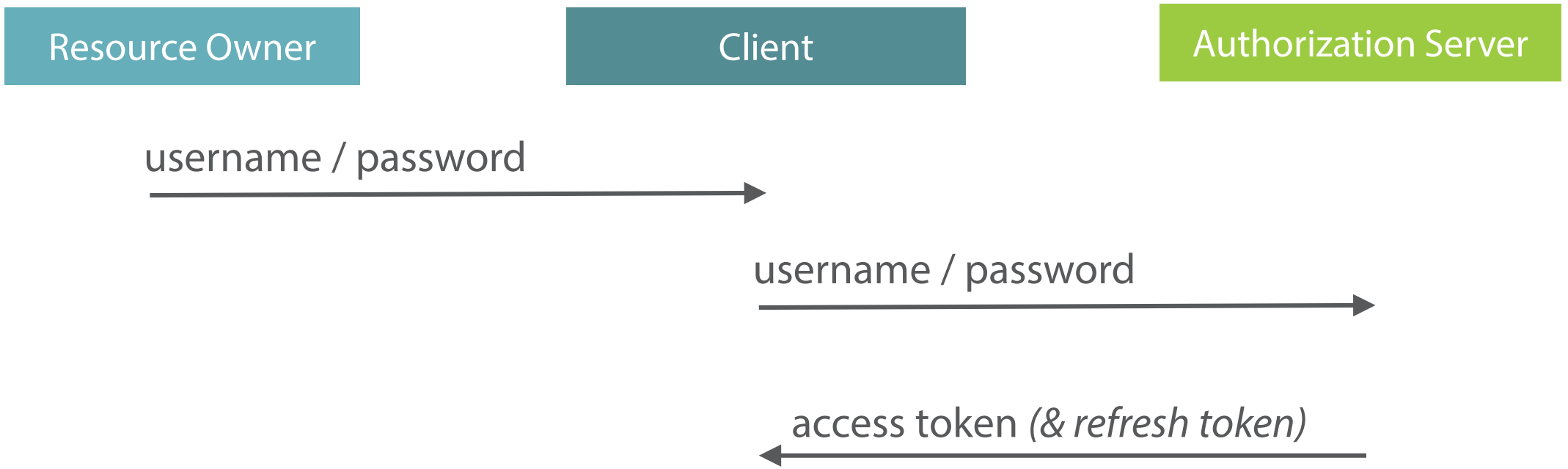
Only for **trusted** applications

Can be used to obtain access tokens and refresh tokens, and includes a client authentication step

Greater risk than the other flows

Only use this if other flows aren't viable!

Resource Owner Password Credentials Flow



Resource Owner Password Credentials Flow – Part 1

Learn how to use the Resource Owner Password Credentials Flow from Angular

Angular and Cross-Origin Resource Sharing

Browser security prevents a web page from making AJAX requests to **another domain**

Cross-origin Resource Sharing (**CORS**) is a W3C standard that allows a server to relax the same-origin policy

Origin: http://example.com

http://example.net
Different domain

http://example.com:9000/foo.html
Different port

https://example.com/foo.html
Different scheme

http://www.example.com/foo.html
Different subdomain

Resource Owner Password Credentials Flow – Part 2

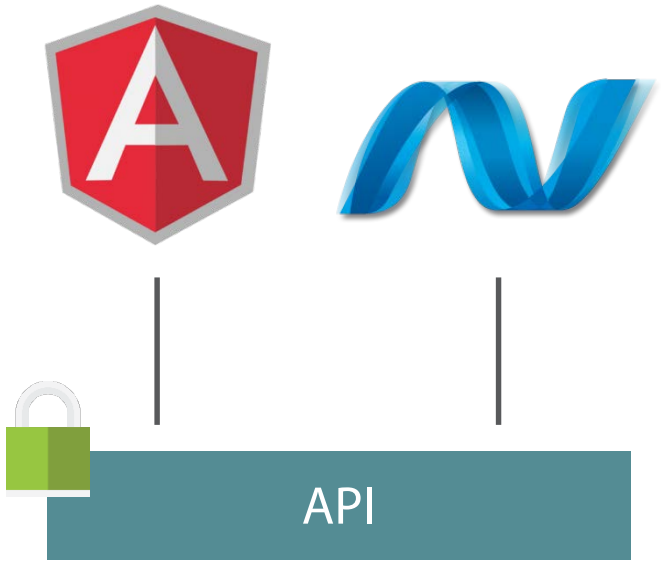
Learn how to use the Resource Owner Password Credentials Flow from Angular

Summary

OAuth 2.0 describes **4 different flows** to achieve authorization

- Client Credentials Flow
- Implicit Flow
- Authorization Code Flow
- Resource Owner Password Credentials Flow

Summary



We know nothing about the **user**

We are **not signed in**

…does signing in even **make sense**?

We need **a different kind of token**