

# Oauth 2

## Easy access delegation



# OAuth 2 rfc6749

## Si basa su un principio molto semplice:

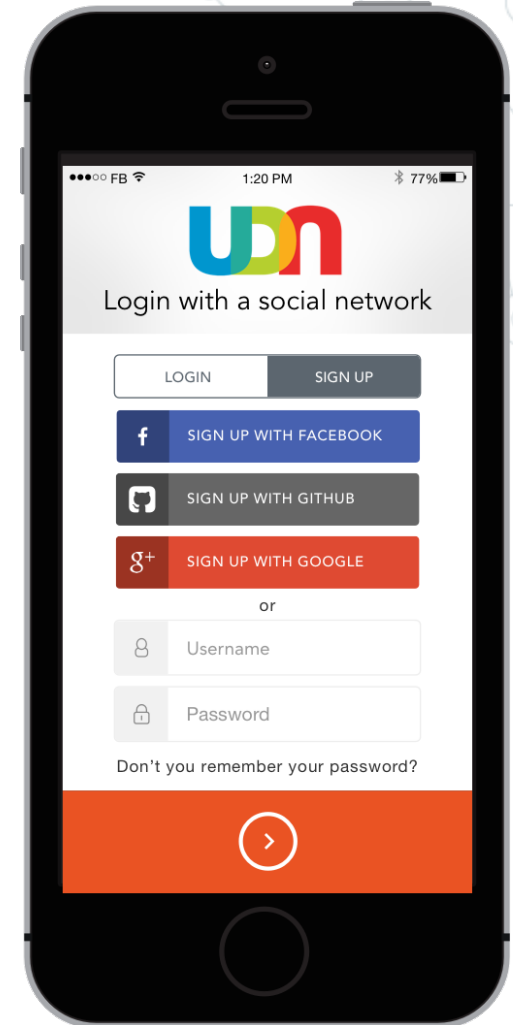
Garantire l'accesso ad applicazioni terze a delle risorse private senza condividere la propria password

## Perché non condividere la propria password?

- non si possono gestire livelli di autorizzazione differenti
- non si può garantire che l'autorizzazione venga utilizzata nel contesto scelto
- per revocare il permesso sono obbligato a cambiare password

OAuth è nato quindi con il presupposto di garantire l'accesso delegato ad un client specifico per determinate risorse sul server per un tempo limitato, con possibilità di revoca.

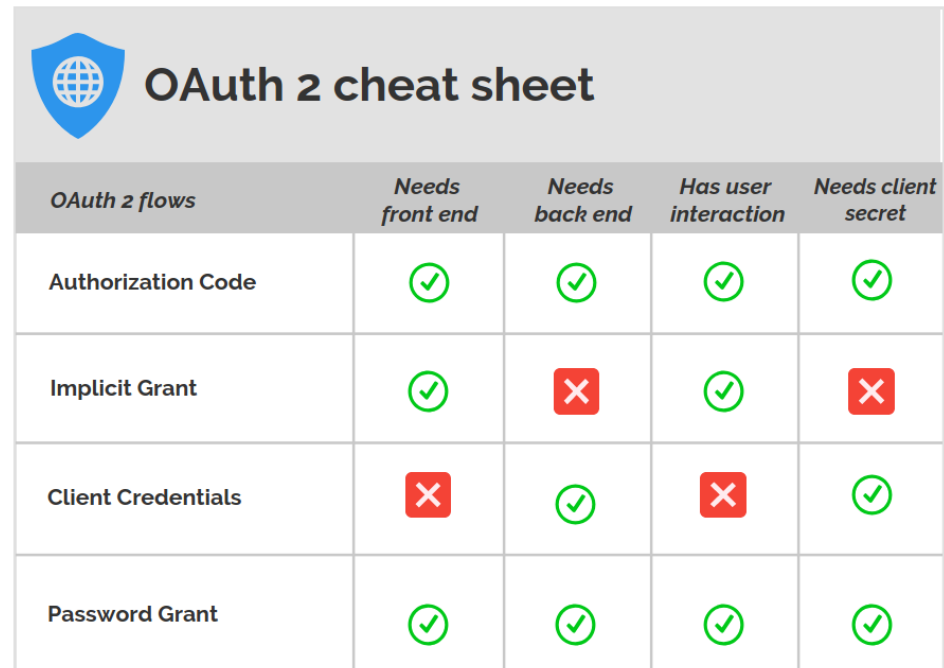
<https://it.wikipedia.org/wiki/OAuth>



# Oauth 2 rfc6749

Esistono differenti «flow» di autorizzazione descritti dal protocollo:

- **Authorization Code Grant**
- **Implicit Grant**
- **Client Credential Grant**
- **Password Grant**

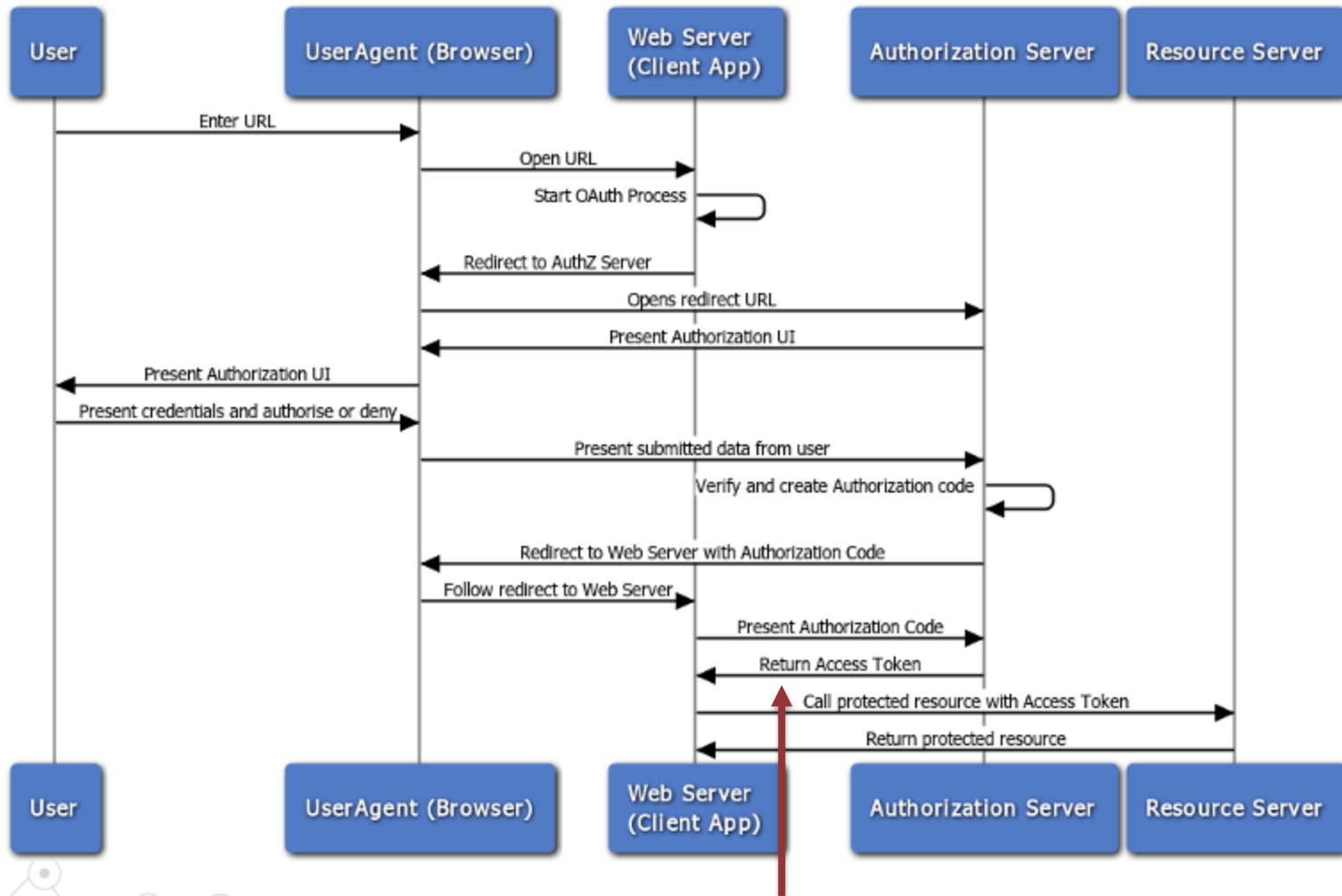


The image shows a cheat sheet for OAuth 2 flows. It features a blue shield icon with a globe inside, followed by the title "OAuth 2 cheat sheet". Below the title is a table with five columns: "OAuth 2 flows", "Needs front end", "Needs back end", "Has user interaction", and "Needs client secret". The rows represent different grant types: Authorization Code, Implicit Grant, Client Credentials, and Password Grant. Each cell in the table contains a green checkmark (✓) or a red X (✗) to indicate the requirements for each flow.

<i>OAuth 2 flows</i>	<i>Needs front end</i>	<i>Needs back end</i>	<i>Has user interaction</i>	<i>Needs client secret</i>
Authorization Code	✓	✓	✓	✓
Implicit Grant	✓	✗	✓	✗
Client Credentials	✗	✓	✗	✓
Password Grant	✓	✓	✓	✓

<https://itnext.io/an-oauth-2-0-introduction-for-beginners-6e386b19f7a9>

# Oauth 2: Authorization Code Flow



Bearer Tokens are the predominant type of access token used with OAuth 2.0. A Bearer Token is an opaque string, not intended to have any meaning to clients using it.

## Oauth 2: Complicato?



<https://auth0.com/pricing/>



Firebase Authentication

<https://firebase.google.com/pricing>

<https://auth0.com/blog/ionic-framework-how-to-get-started/>

<https://auth0.com/docs/quickstart/spa/angular2/01-login>

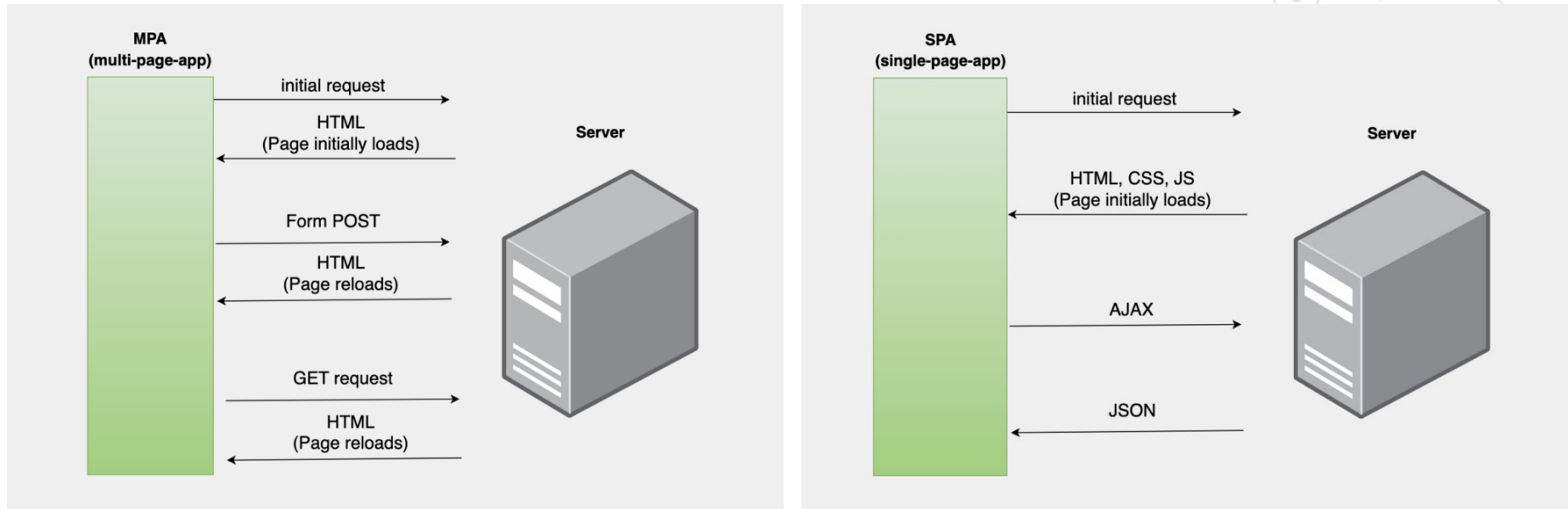
<https://github.com/angular/angularfire>

<https://github.com/angular/angularfire/blob/master/docs/auth/getting-started.md>

# Server side rendering



## Approcci «classici»:



Un nuovo “approccio” sta emergendo:  
Server Side Rendering

<https://pretius.com/blog/angular-ssr/>

# Web vitals / SEO su Single Page Application

https://pagespeed.web.dev/

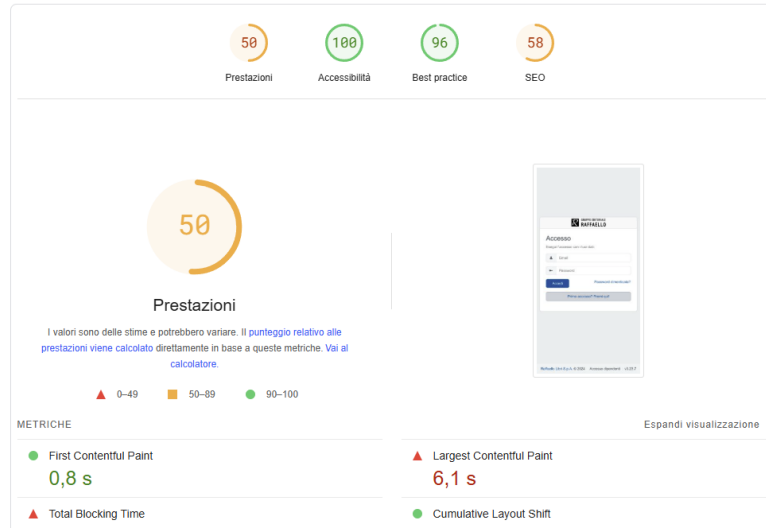
Report di 1 dic 2024, 13:22:21

https://conc.grupporaffaello.it/ Analizza

Dispositivi mobili  Computer

Scopri com'è l'esperienza dei tuoi utenti reali Nessun dato

## Diagnostica i problemi di prestazioni

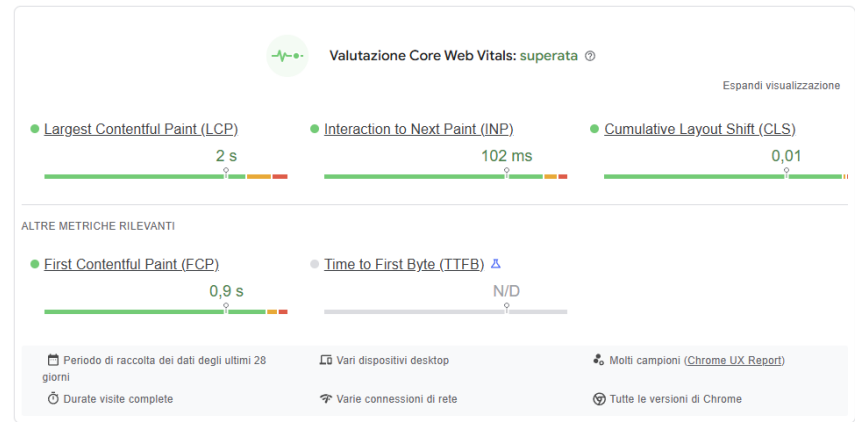


Report di 1 dic 2024, 13:22:21

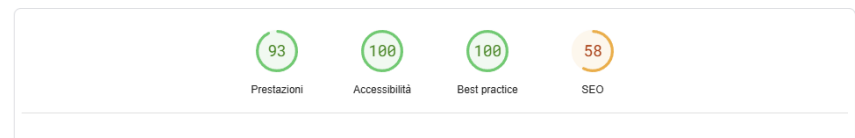
https://conc.grupporaffaello.it/ Analizza

Dispositivi mobili  Computer

Scopri com'è l'esperienza dei tuoi utenti reali Questo URL Origine

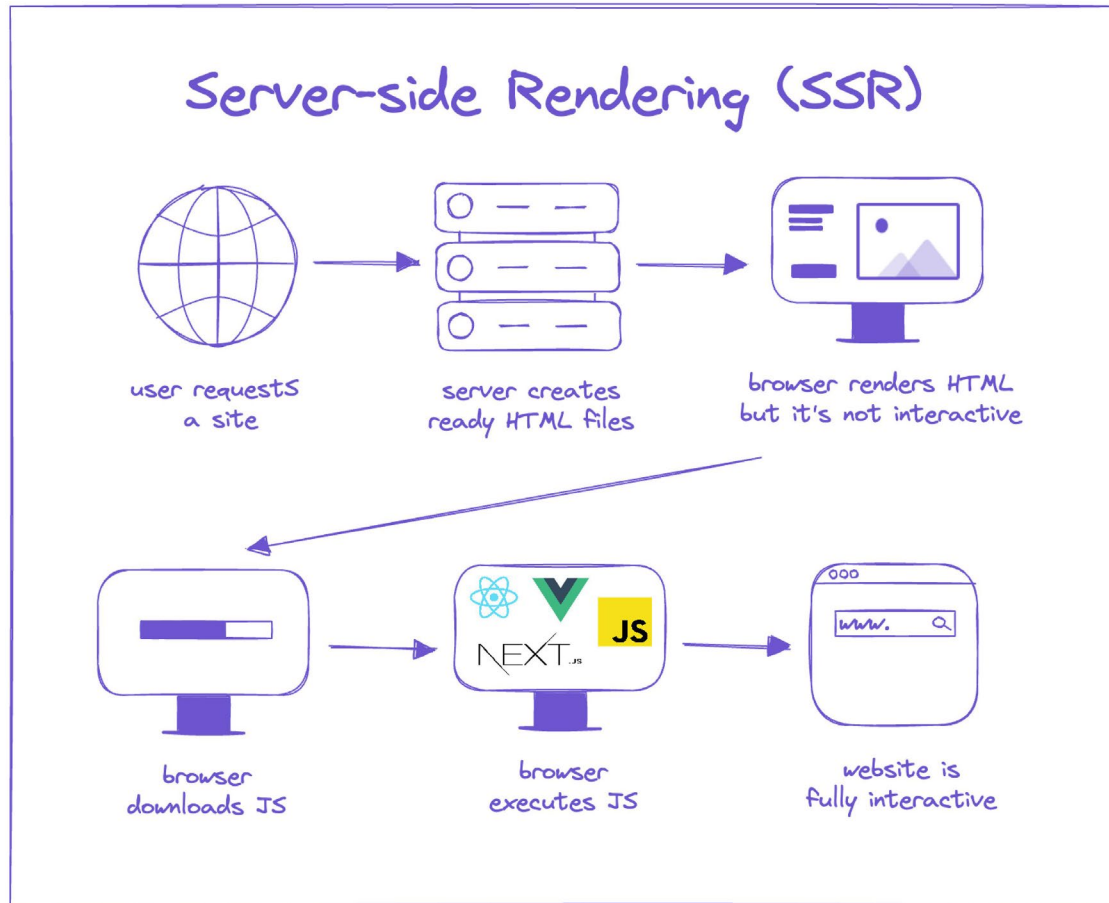


## Diagnostica i problemi di prestazioni



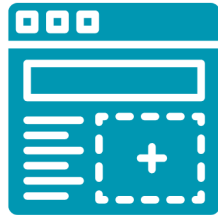


# Server Side Rendering:



<https://prismic.io/blog/what-is-ssr>

## What is 'Hydration'?



**Server Rendered HTML**



**Attaching  
interactive stuff**

(e.g. Reusing DOM Structures,  
Attaching Event Handler,  
Restoring Application State)



**"Booted"  
Single Page App on  
client side**

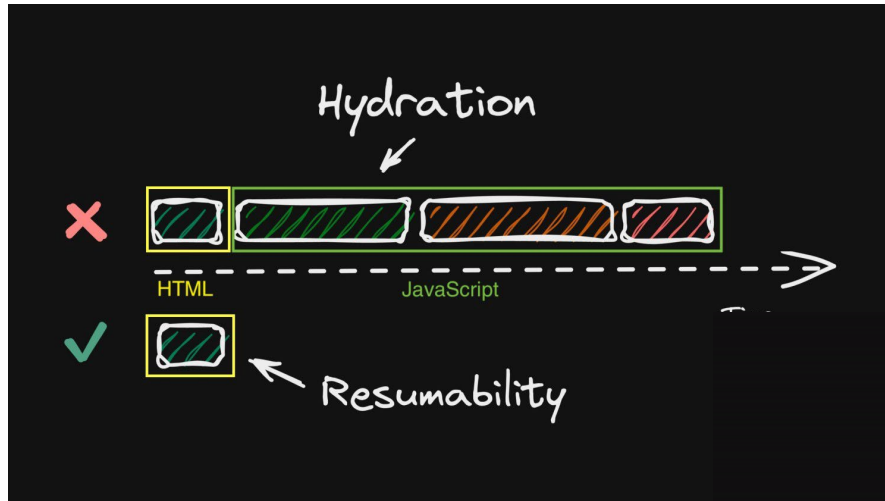


**PATRICK ROOS**  
TECH LEAD  
SOFTWARE ARCHITECT  
SOFTWARE ENGINEER



<https://www.workingsoftware.dev/understanding-ssr-with-hydration-for-software-architects/>

# Resumability (futuro?):



2. How does Resumability work?

<https://www.linkedin.com/pulse/hydration-resumability-what-sets-them-apart-website-usama-tahir-udjmf/>

<https://qwik.dev/examples/reactivity/counter/>