

# o guia do desenvolvedor\* para **SEGURANÇA CLIENT-SIDE**

@olarclara

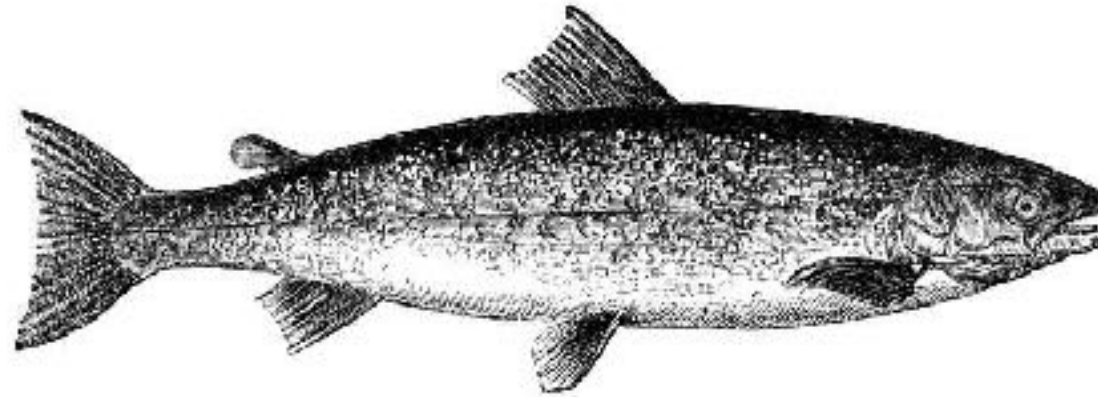


**women**

in tech



*Security by optimism and prayer*



*Expert*

Hoping Nobody  
Hacks You

O RLY?

@ThePracticalDev

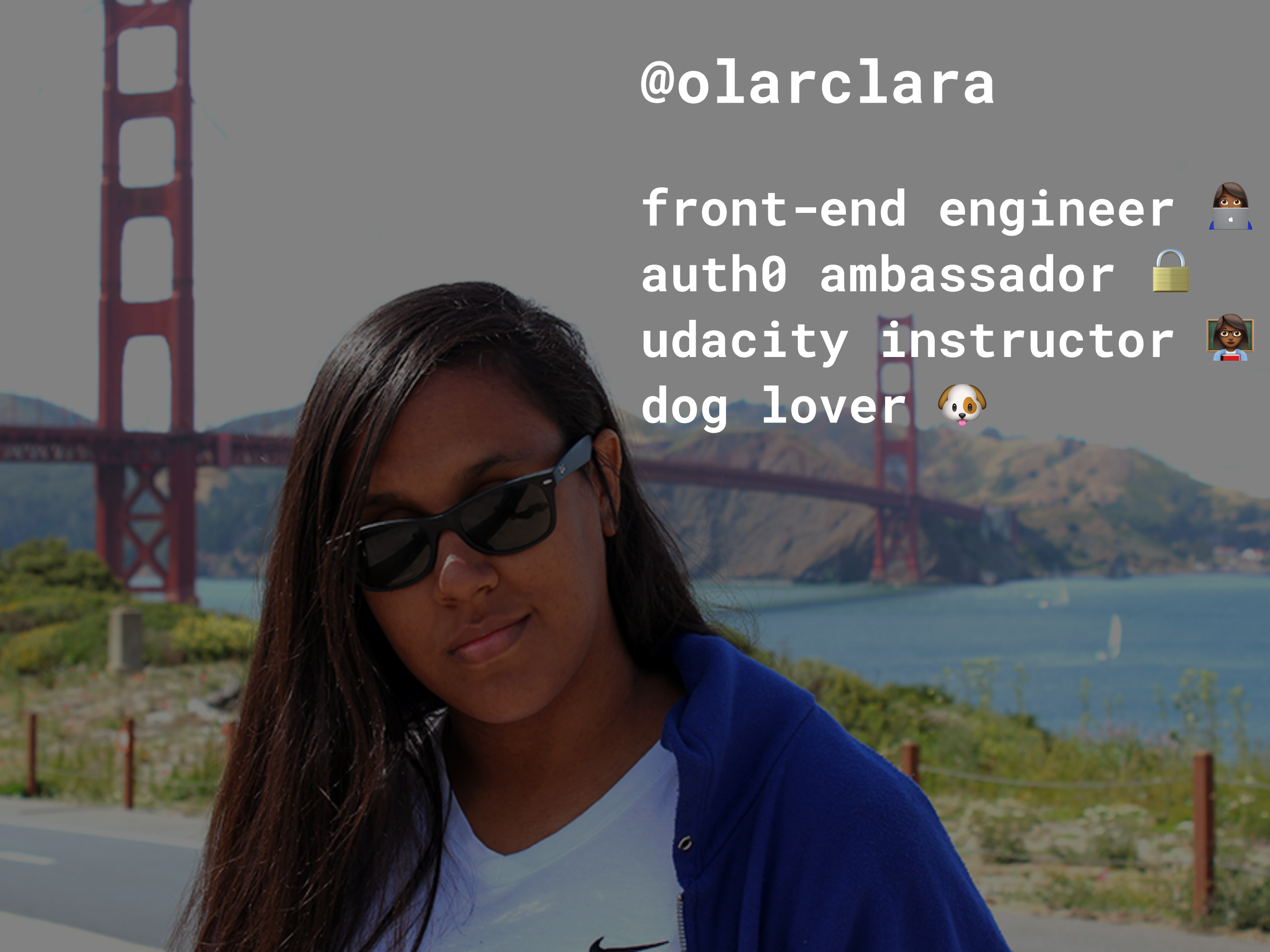
@olarclara

front-end engineer 🧑💻

auth0 ambassador 🗝️

udacity instructor 🧑🎓

dog lover 🐶

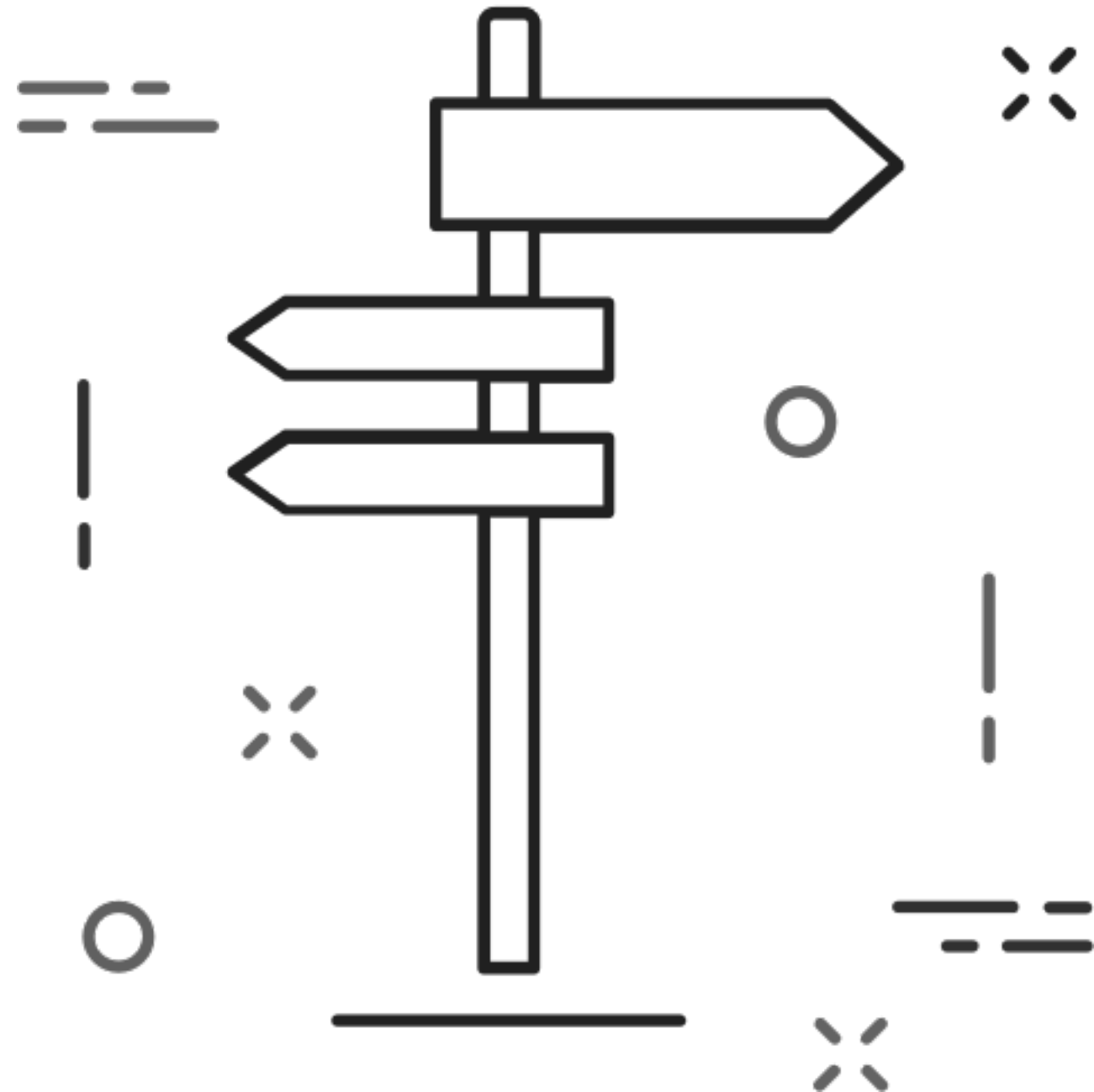




**Auth0**

# ROADMAP

- **ataques;**
- **identidade;**
- **diversos;**



# **CROSS-SITE SCRIPTING**



ChromeFileEditViewHistoryBookmarksPeopleWindowHelp

Twitter, Inc. [US]

https://twitter.com/derGeruhn/status/476764918763749378

HomeMomentsNotificationsMessagesSearch TwitterTweet



\*andy

@derGeruhn

Follow

<script  
class="xss">\$(\$('.xss').parents().eq(1).find('a').  
eq(1).click());\$('[data-  
action=retweet]').click();alert('XSS in  
Tweetdeck')</script> ❤️

12:36 PM · 11 Jun 2014

69,080 Retweets17,004 Likes



5.0K

69K

17K



Tweet your reply



Elk Cloner @ElkCl · Mar 2

Replying to @derGeruhn

how did you think of doing that? I mean how did you notice the exploit?

1

2

© 2017 Twitter

About

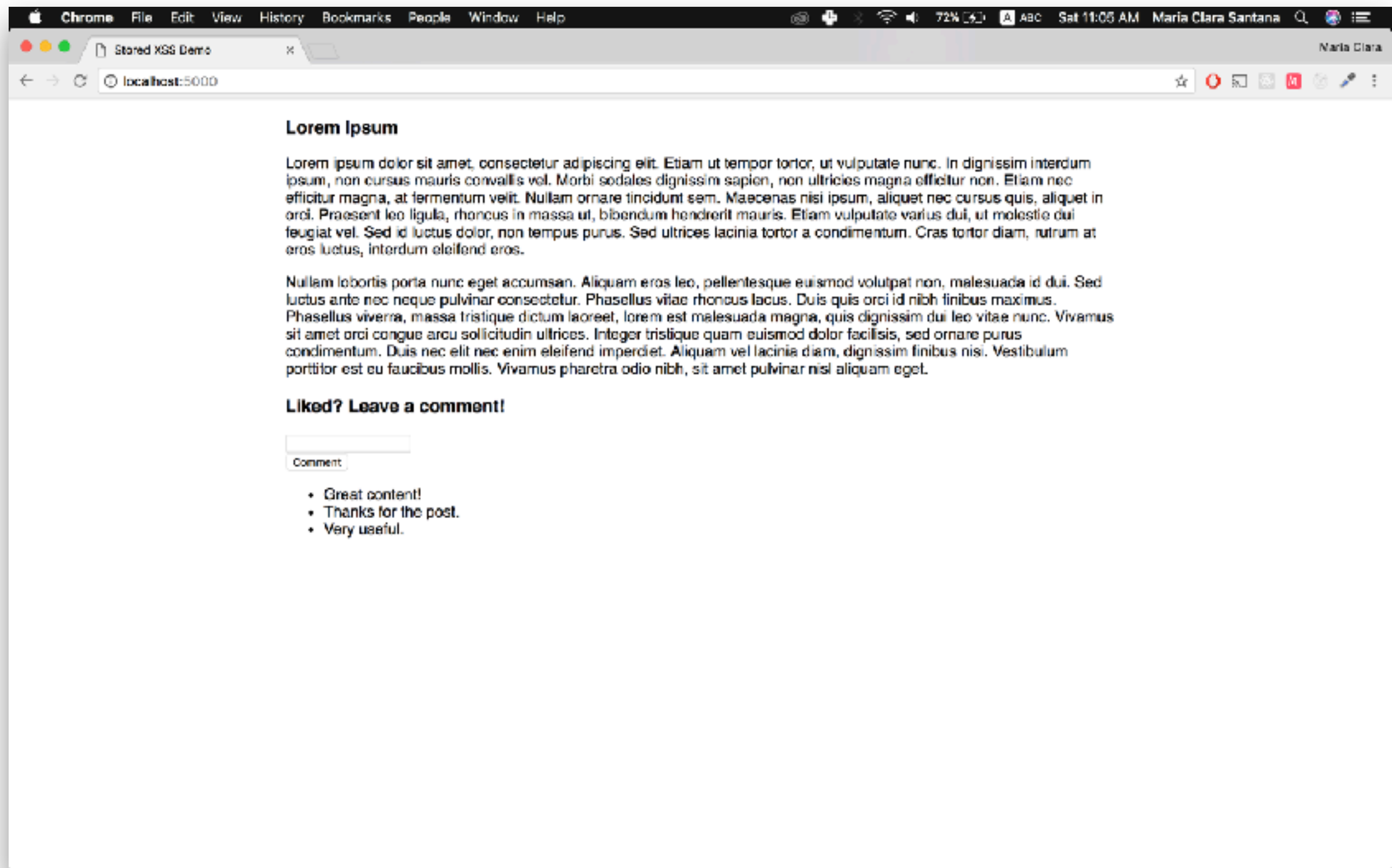
Help Center

Terms

Privacy policy

Cookies

Ads info







```
<div>
  <h3>Liked? Leave a comment!</h3>
  <form method="post" action="/new-comment">
    <input type="text" name="content" id="inputComment">
    <button type="submit">Comment</button>
  </form>
  <ul>
    <% comments.forEach((comment) => { %>
      <li> <%- comment.content %> </li>
    <% }) %>
  </ul>
</div>
```



```
const app = require('express')();
const bodyParser = require('body-parser');
const urlencodedParser = bodyParser.urlencoded({ extended: false })

app.set('view engine', 'ejs');
let comments = [
  { content: 'Great content!' },
  { content: 'Thanks for the post.' },
  { content: 'Very useful.' }
];

app.get('/', (req, res) => {
  res.render('./index.ejs', { comments })
});

app.post('/new-comment', urlencodedParser, (req, res) => {
  comments.push(req.body);
  res.redirect('/');
});
```

# prevenindo XSS

- implemente **Content Security Policy**;
- **trust no input**, sempre escape qualquer tipo de informação inserida pelo usuário;
- sempre valide estritamente as variáveis que vão pro css (stylesheet);
- use o javascript para validar e escapar dados antes de inseri-los no DOM;



# **CROSS-SITE REQUEST FORGERY**



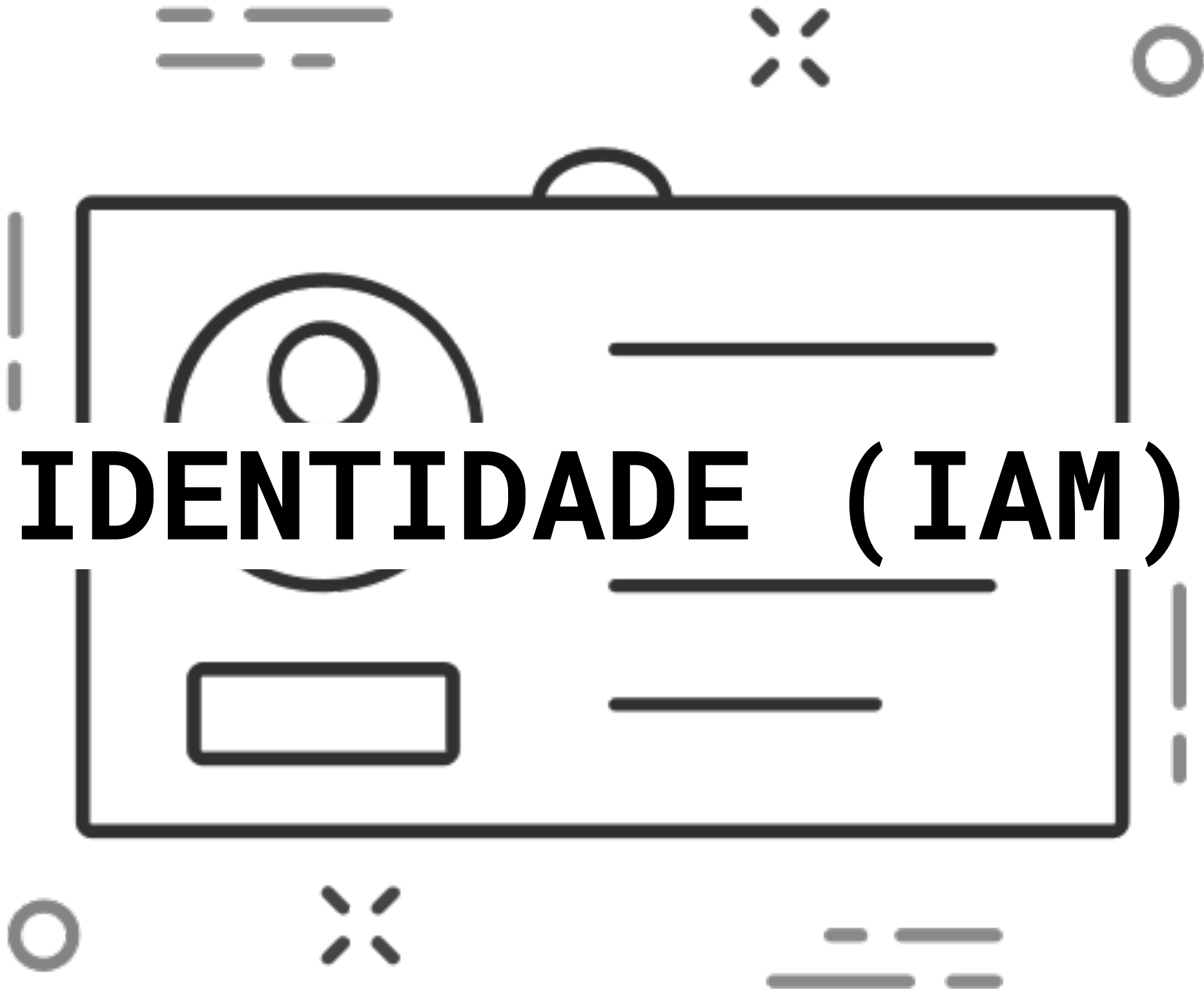
```
GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1
```

```
<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">  
  As fotos da festa ficaram ótimas!  
</a>
```

# prevenindo CSRF

- gere tokens anti-csrf únicos na sua aplicação;
- limite a validade dos tokens dos usuários;
- não salve seus tokens em plain text;
- envie duplamente os cookies;
- headers customizados;





Client Settings

Secure https://manage.auth0.com/w/clients/KFUh1tVw2xg32r7UgLNija5pw2OH7taU/settings

Auth0

Search for clients or features

Help & Support Documentation Talk to Sales mariaclarasantana

User Search features were deactivated due to the low activity of this account. The result of this is that new users or update to existing users will not be indexed. For re-enabling this feature, let us know by clicking the button next to this message. We will notify you as soon as the request has been completed.

REQUEST

Dashboard

Clients

APIs

SSO Integrations

Connections

Users

Rules

Hooks

Multifactor Auth

Hosted Pages

Emails

Logs

Anomaly Detection

Extensions

Get Support

WDS Demo

Quick Start Settings Addons Connections

Client ID: KFUh1tVw2xg32r7UgLNija5pw2OH7taU

Name

WDS Demo

Domain

mariaclarasantana.auth0.com

Client ID

KFUh1tVw2xg32r7UgLNija5pw2OH7taU

Client Secret

\*\*\*\*\*

☐ Reveal client secret.

The Client Secret is not base64 encoded.

Description

This client is used for...

Chrome

File

Edit

View

History

Bookmarks

People

Window

Help

95%

ABC

Sat 2:14 PM

Maria Clara Santana

Social Connections

Securehttps://manage.auth0.com/#/connections/social

will not be indexed. For re-enabling this feature, let us know by clicking the button next to this message. We will notify you as soon as this feature is available.

Auth0

has been completed

Search for clients or features

Help & Support

Documentation

Talk to Sales

mariaclarasantana

Dashboard

Clients

APIs

SSO Integrations

Connections

Database

Social

Enterprise

Passwordless

Users

Rules

Hooks

Multifactor Auth

Hosted Pages

Emails

Logs

Anomaly Detection

Extensions

Get Support

Social Connections

TUTORIAL

New! Learn about creating custom OAuth2 connections.

One or more connections are using Auth0 development keys which are only intended for use in development and testing. The connections should be configured with your own Developer Keys to enable the consent page to show your logo instead of Auth0's and to enable SSO for these connections. Auth0 development keys are not recommended for Production environments. To learn more about Development Keys please read the Social Connections Devkeys documentation.

Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. [Learn more](#)

<div>Google</div> <div>TRY &gt;</div> <div></div>	<div>facebook</div> <div></div>	<div>Microsoft</div> <div></div>
<div>LinkedIn</div> <div></div>	<div>GitHub</div> <div>TRY &gt;</div> <div></div>	<div>Dropbox</div> <div></div>
<div>Twitter</div> <div></div>	<div>PayPal</div> <div></div>	<div>PayPal</div> <div></div>





```
const webAuth = new auth0.WebAuth({
  domain: AUTH0_DOMAIN,
  clientID: AUTH0_CLIENT_ID,
  redirectUri: AUTH0_CALLBACK_URL,
  audience: 'https://' + AUTH0_DOMAIN + '/userinfo',
  responseType: 'token id_token',
  scope: 'openid profile',
  leeway: 60
});

const loginBtn = document.getElementById('btn-login');

loginBtn.addEventListener('click', function(e) {
  e.preventDefault();
  webAuth.authorize();
});
```



```
const handleAuthentication = () => {  
  webAuth.parseHash(function(err, authResult) {  
    if (authResult && authResult.accessToken && authResult.idToken) {  
      setSession(authResult);  
    } else if (err) {  
      console.error(err);  
    }  
  });  
}
```



```
const setSession = authResult => {  
  const expiresAt = JSON.stringify(  
    authResult.expiresIn * 1000 + new Date().getTime()  
  );  
  localStorage.setItem('access_token', authResult.accessToken);  
  localStorage.setItem('id_token', authResult.idToken);  
  localStorage.setItem('expires_at', expiresAt);  
}
```

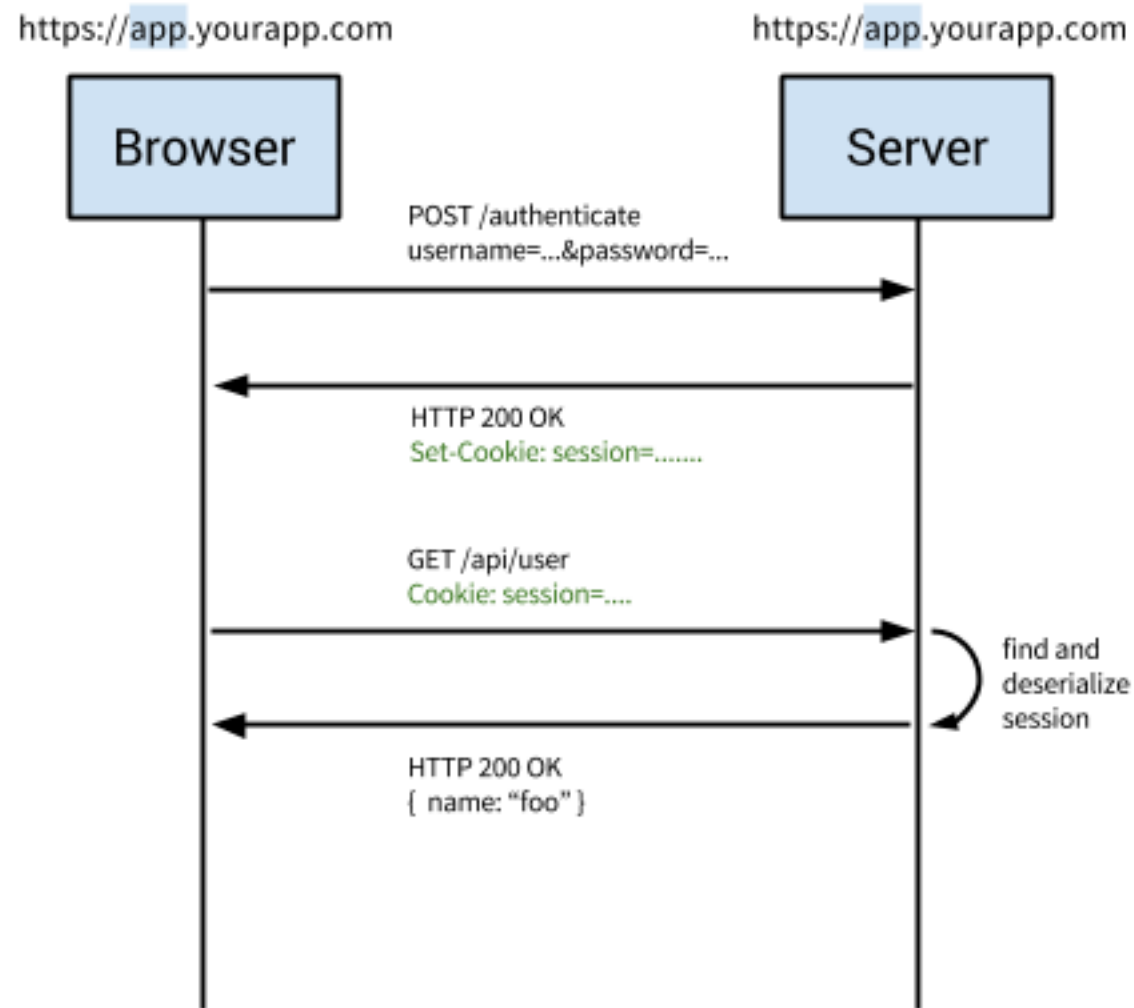




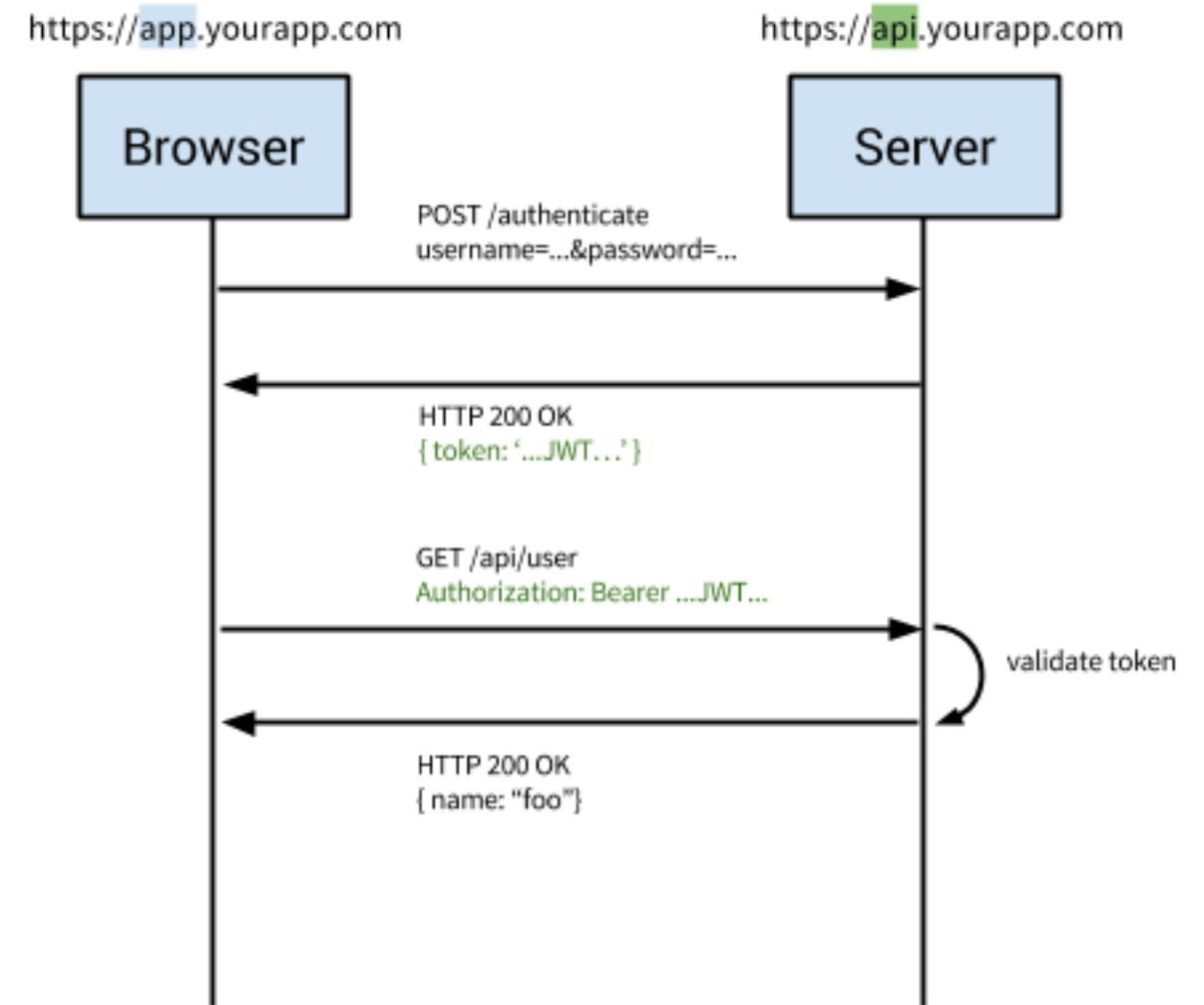
```
const getProfile = () => {  
  if (!userProfile) {  
    const accessToken = localStorage.getItem('access_token');  
  
    if (!accessToken) console.log('Could not find a token to fetch profile');  
  
    webAuth.client.userInfo(accessToken, ((err, profile) => {  
      if (profile) {  
        userProfile = profile;  
        displayProfile();  
      }  
    }));  
  } else {  
    displayProfile();  
  }  
}
```

# **COOKIES VS. TOKENS**

## Traditional Cookie-Based Auth



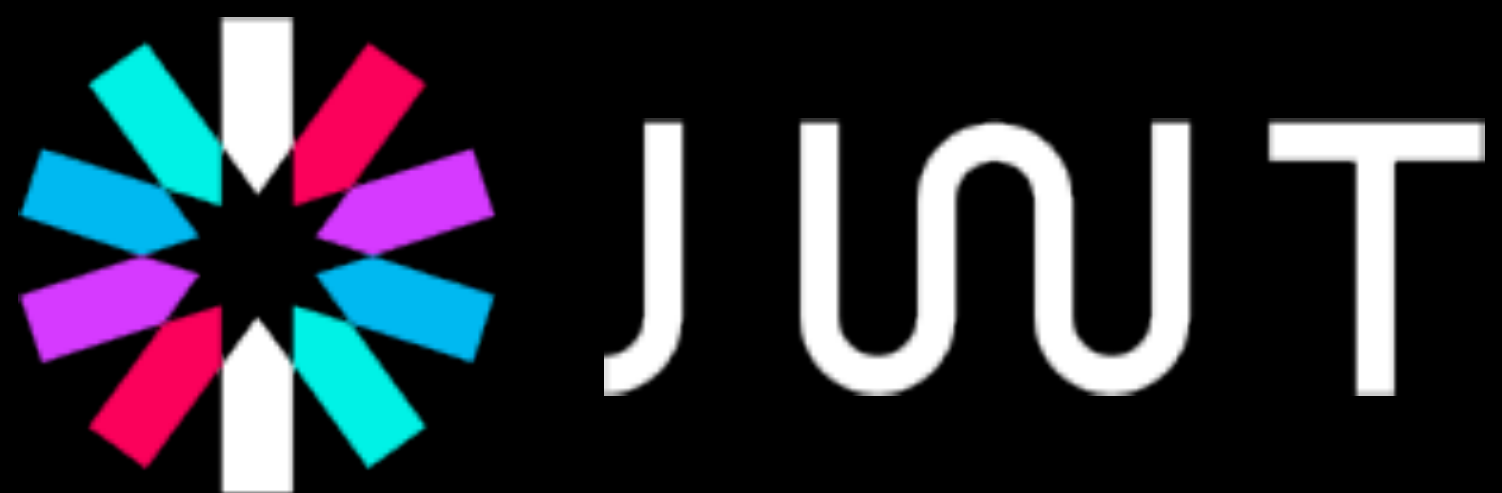
## Modern Token-Based Auth

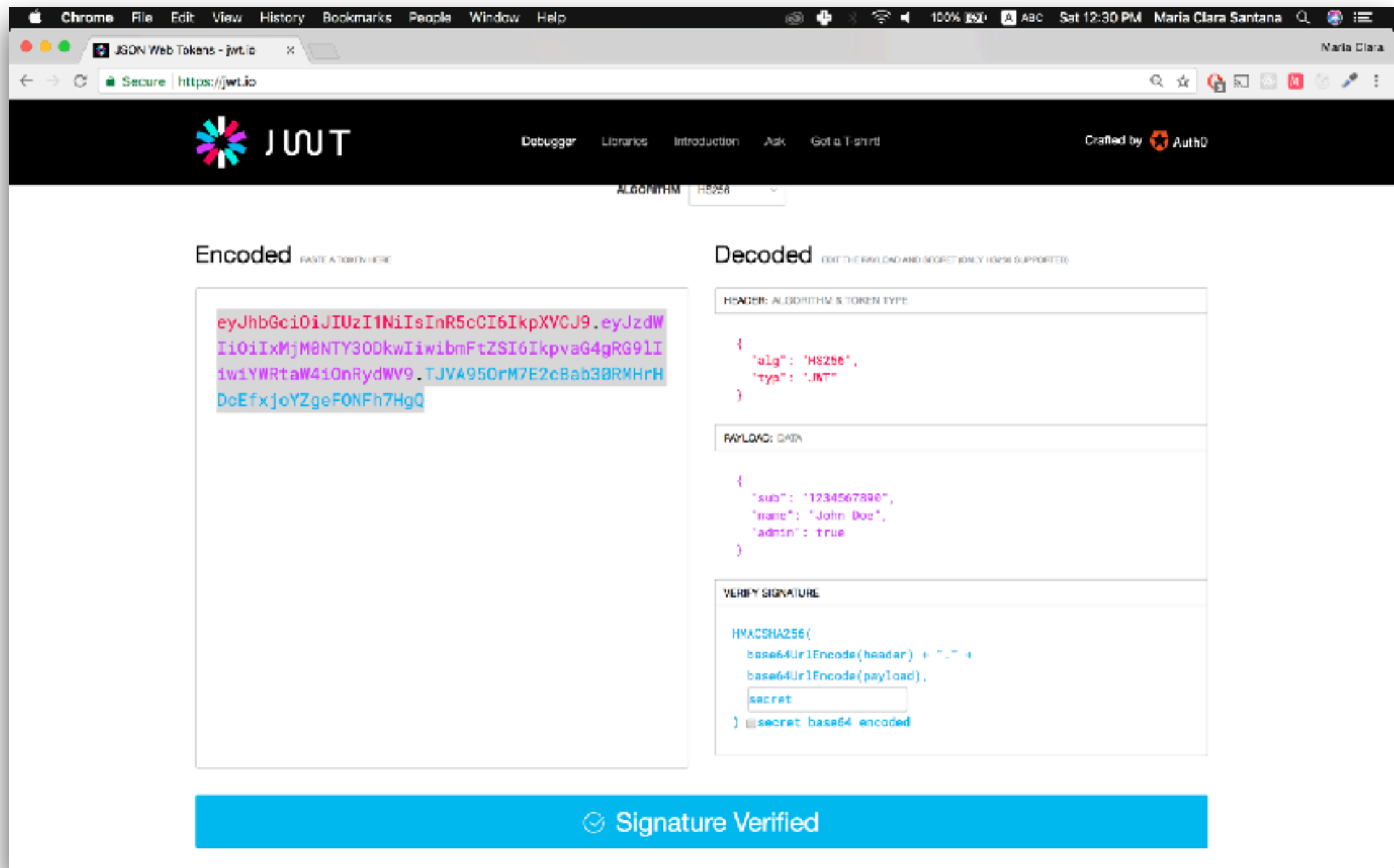


# tokens > cookies

- token são stateless e portanto escaláveis;
- tokens permitem tirar vantagem de CORS;
- JWTs;
- melhor interação cross-platform;

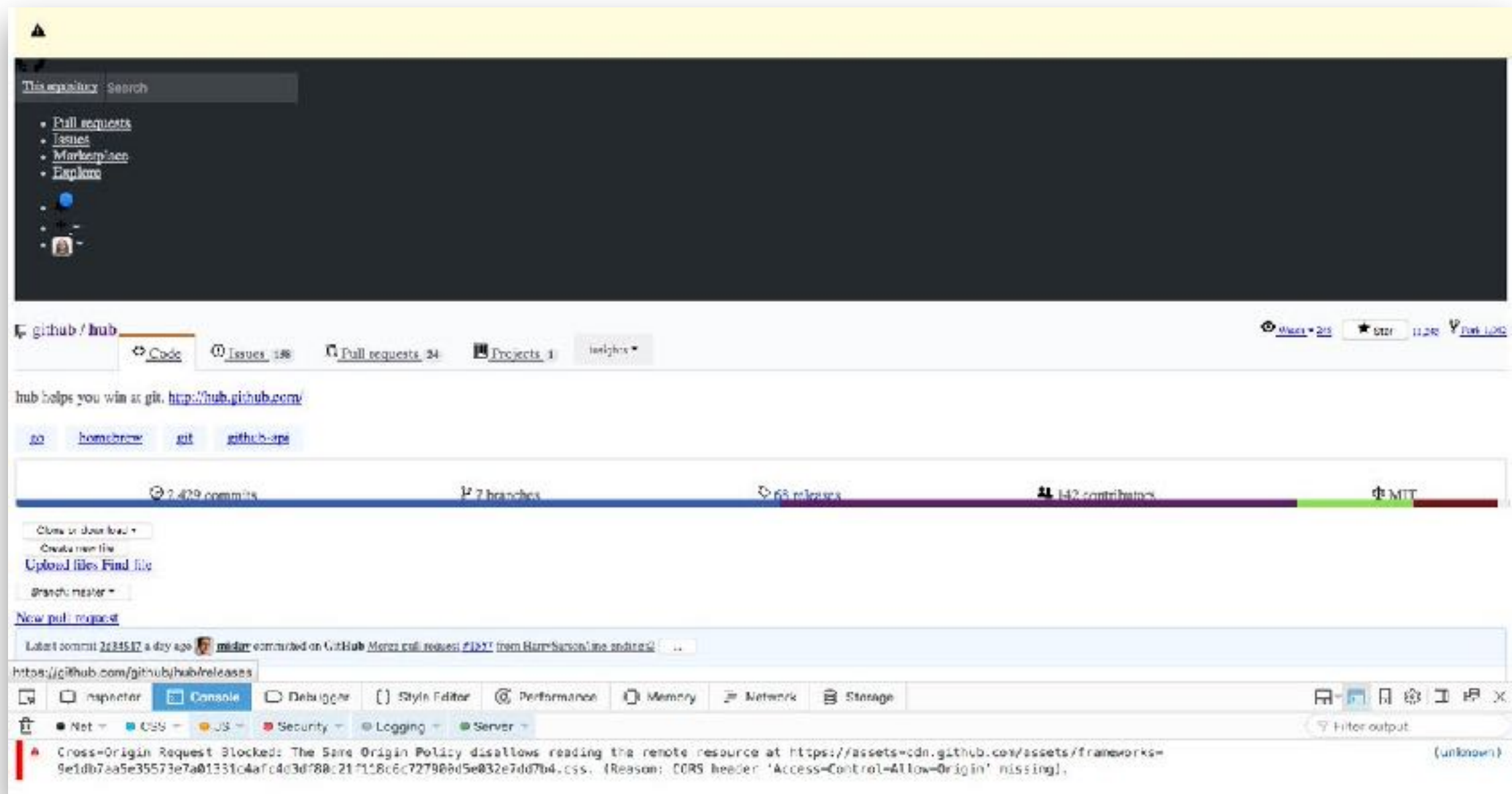






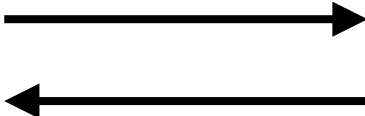
# **CORS**

**(cross-origin resource sharing)**



(Imagen via <https://twitter.com/curiouslorax/status/905131369357078528>)

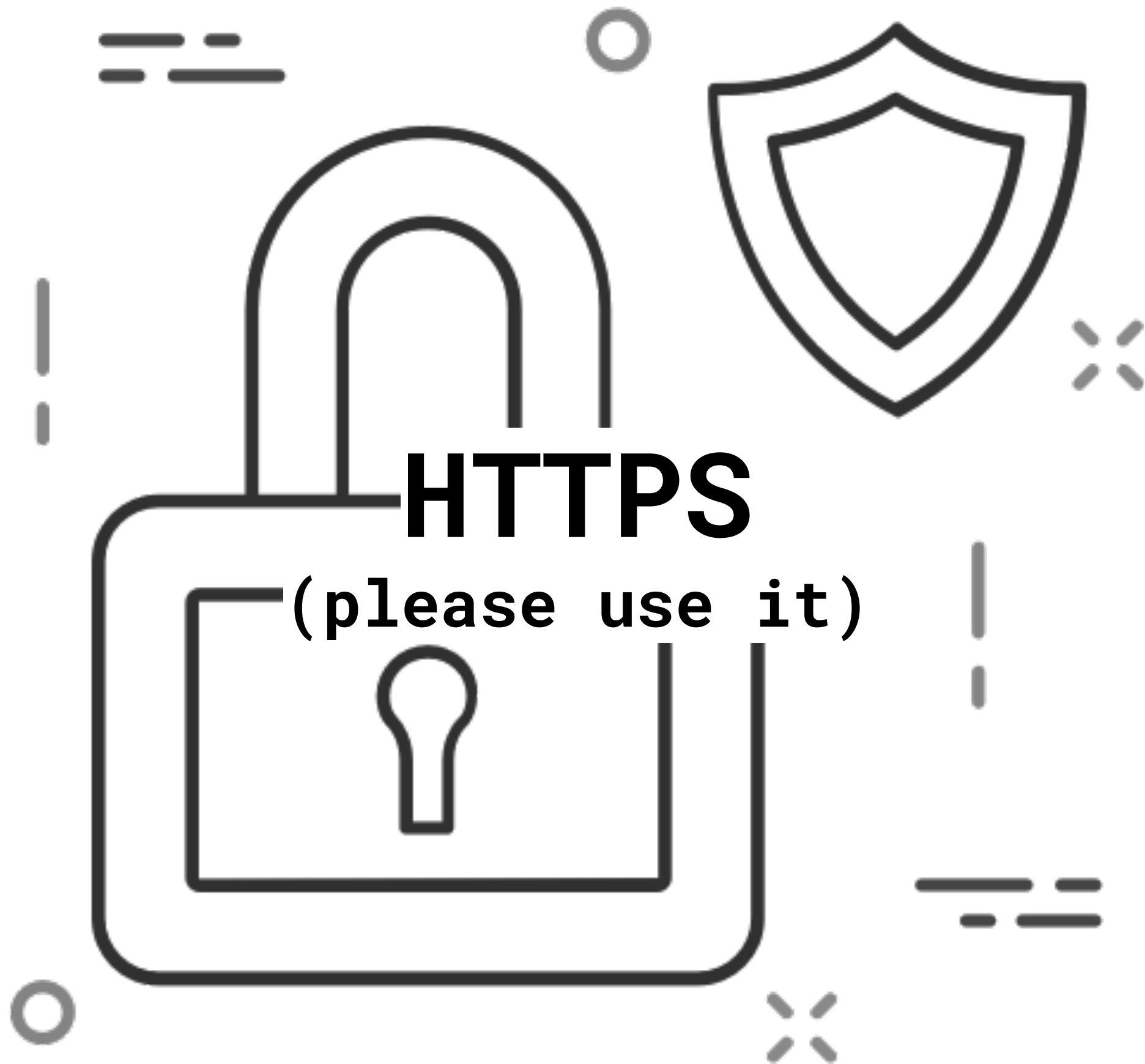
```
GET /greeting/ HTTP/1.1  
Origin: http://foo.client.com
```

foo.client.com  bar.server.com



```
HTTP/1.1 200 OK  
Content-Type: application/json; charset=UTF-8  
Access-Control-Allow-Origin: http://foo.client.com
```

```
[response payload]
```





**HTTPS**  
**(please use it)**

1. “Meu site não é importante suficiente pra precisar de HTTPS.” 
2. “HTTPS é caro.” 








**1. “Meu site não é importante suficiente pra precisar de HTTPS.”** 

**2. “HTTPS é caro.”** 









# Chromium Blog

News and developments from the open source browser project

## Next steps toward more connection security

Thursday, April 27, 2017


In January, we [began our quest](#) to improve how Chrome communicates the connection security of HTTP pages. Chrome now marks HTTP pages as "Not secure" if they have password or credit card fields. Beginning in October 2017, Chrome will show the "Not secure" warning in two additional situations: when users enter data on an HTTP page, and on all HTTP pages visited in [Incognito mode](#).


	Treatment of HTTP pages outside Incognito mode.	Treatment of HTTP pages in Chrome Incognito mode.
Current (Chrome 58)	 example.com	 example.com
Oct. 2017 (Chrome 62) at page load	 example.com	 Not secure example.com
Oct. 2017 (Chrome 62) on user input	 example.com	 Not secure example.com

Labels

Archive

Feed

Google on 

 Follow @ChromiumDev

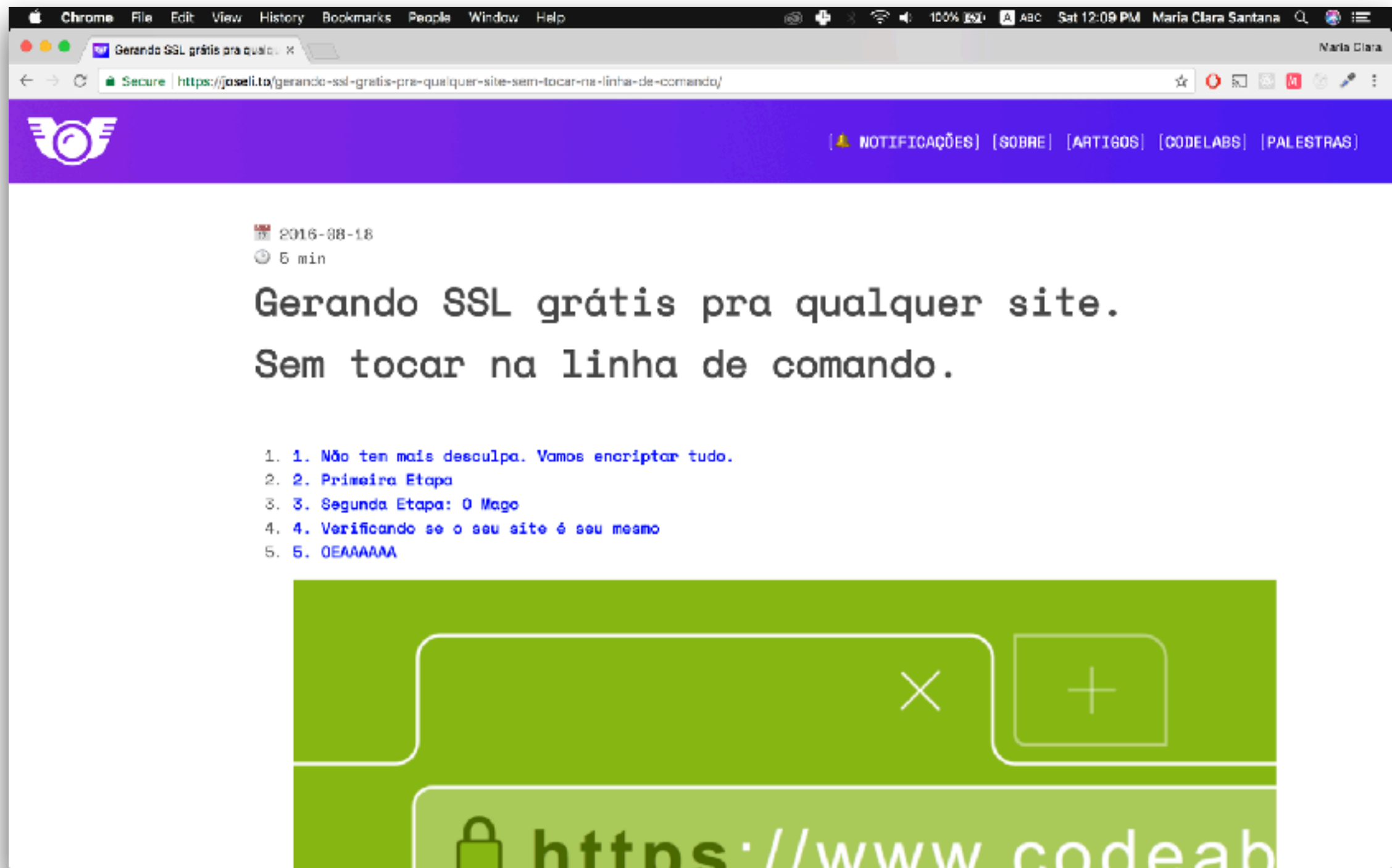
Give us feedback in our [Product Forums](#).



- `geolocation api;`
- `http/2;`
- `push notifications;`
- `user media;`
- `SEO;`
- ...

1. “Meu site não é importante suficiente pra precisar de HTTPS.” 

2. “HTTPS é caro.” 



# OBRIGADA!



<https://olarclara.github.io>

