

OFFLINE-FIRST DESIGN & WEB COMPONENTS

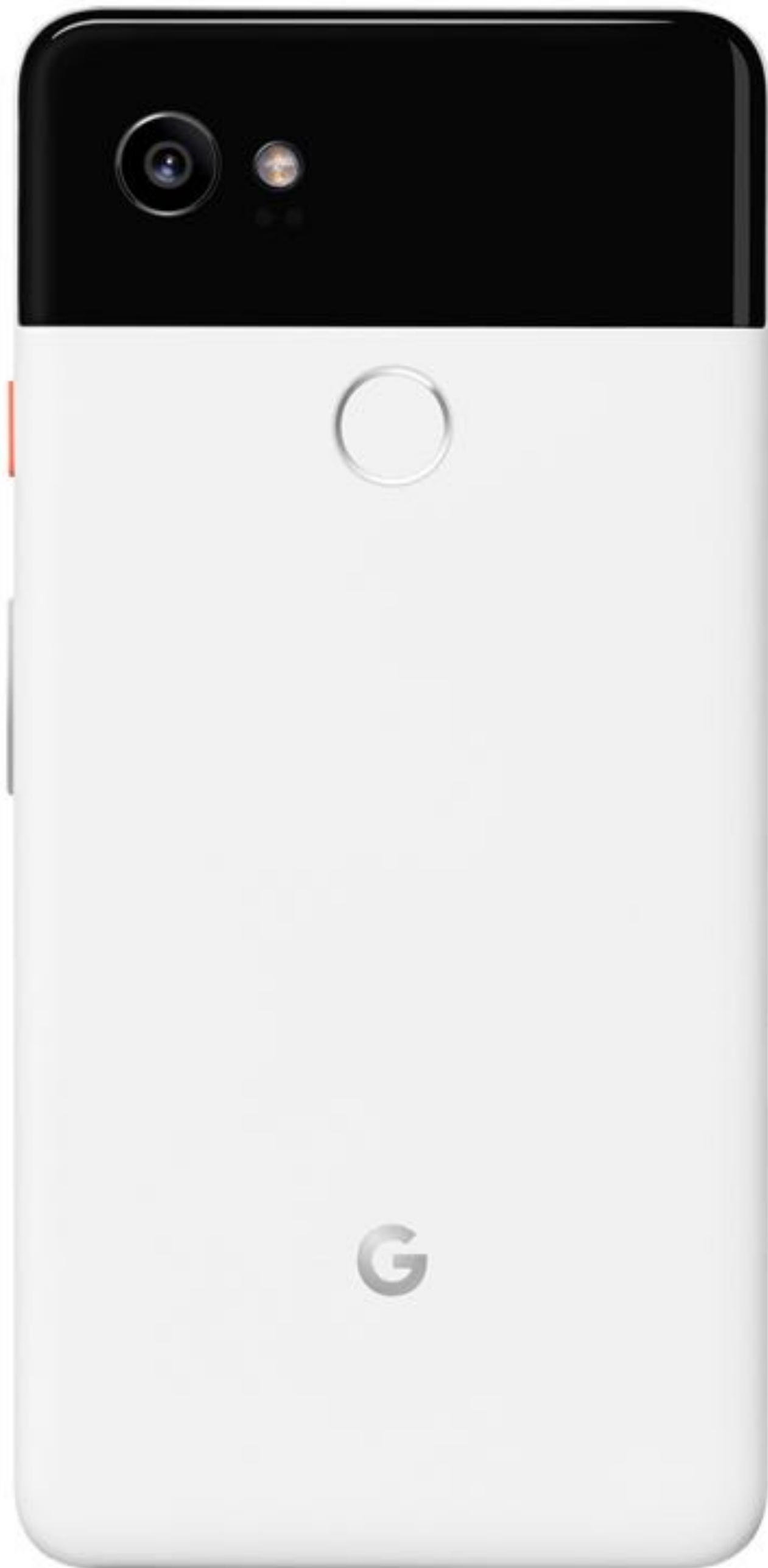


AMAHDY ABDELAZIZ

WWW.AMAHDY.NET

@AMAHDY7





Massive Tech. Specs

- 88 Gigabyte of RAM
- Multi-Hexa Processor
- 2 Tera Pixel Camera
- Convertible into white Goblin



€1000



USER

DESIGN NOTE #00

TECH SPECS

VS.

USABILITY

AMAZON

AMAZON



AMAZON



AMAZON



Content Producer Internet User



FAMILIAR TOOLS

VS.

TARGET AUDIENCE

Twitter

← AMahdy Abdelaziz

Tweets

Media

Likes



AMahdy Abdelaziz @amahdy7 · 6h

The best @oracle team with the @vaadin reindeer! #javaone



1

8



AMahdy Abdelaziz @amahdy7 · 7h

The best @oracle team with the @vaadin reindeer! #javaone

3



AMahdy Abdelaziz @amahdy7 · Oct 3

Replying to @manekinekkko @Sara_harkousse and @GeertjanW

Same technologie but not same technology :P Those French nerds..

1

1

← AMahdy Abdelaziz

401 Tweets

Tweets

Tweets et réponses

Médias

J'aime



AMahdy Abdelaziz @amahdy7 · 7 h

The best @oracle team with the @vaadin reindeer! #javaone



1

7



AMahdy Abdelaziz @amahdy7 · 7 h

The best @oracle team with the @vaadin reindeer! #javaone

3



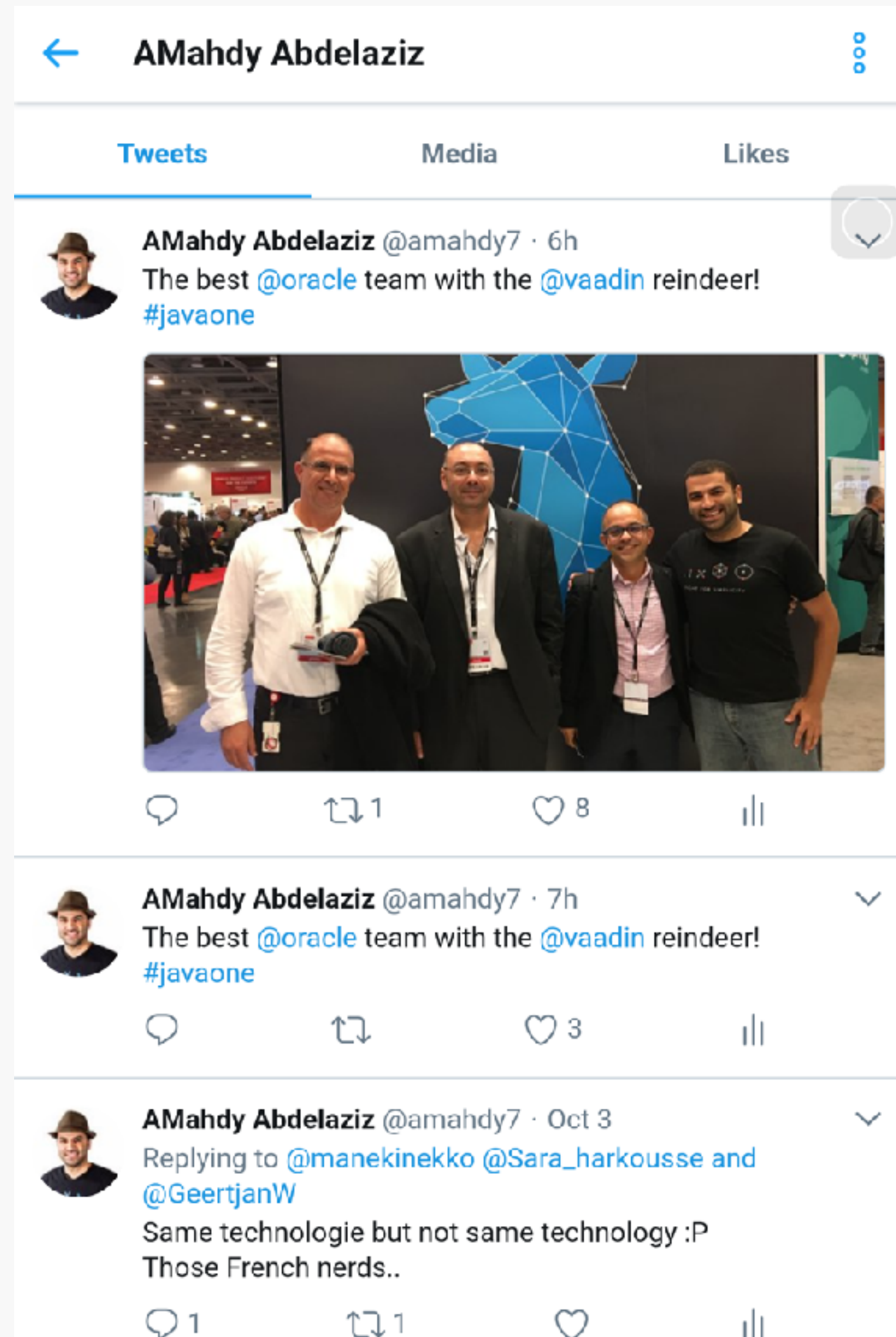
AMahdy Abdelaziz @amahdy7 · 1j

Retweeted Matti Tahvonen (@MattiTahvonen):

JS vs Java presentation starting by my good friends @GeertjanW and... fb.me/615JEFzhd

vaadin}>

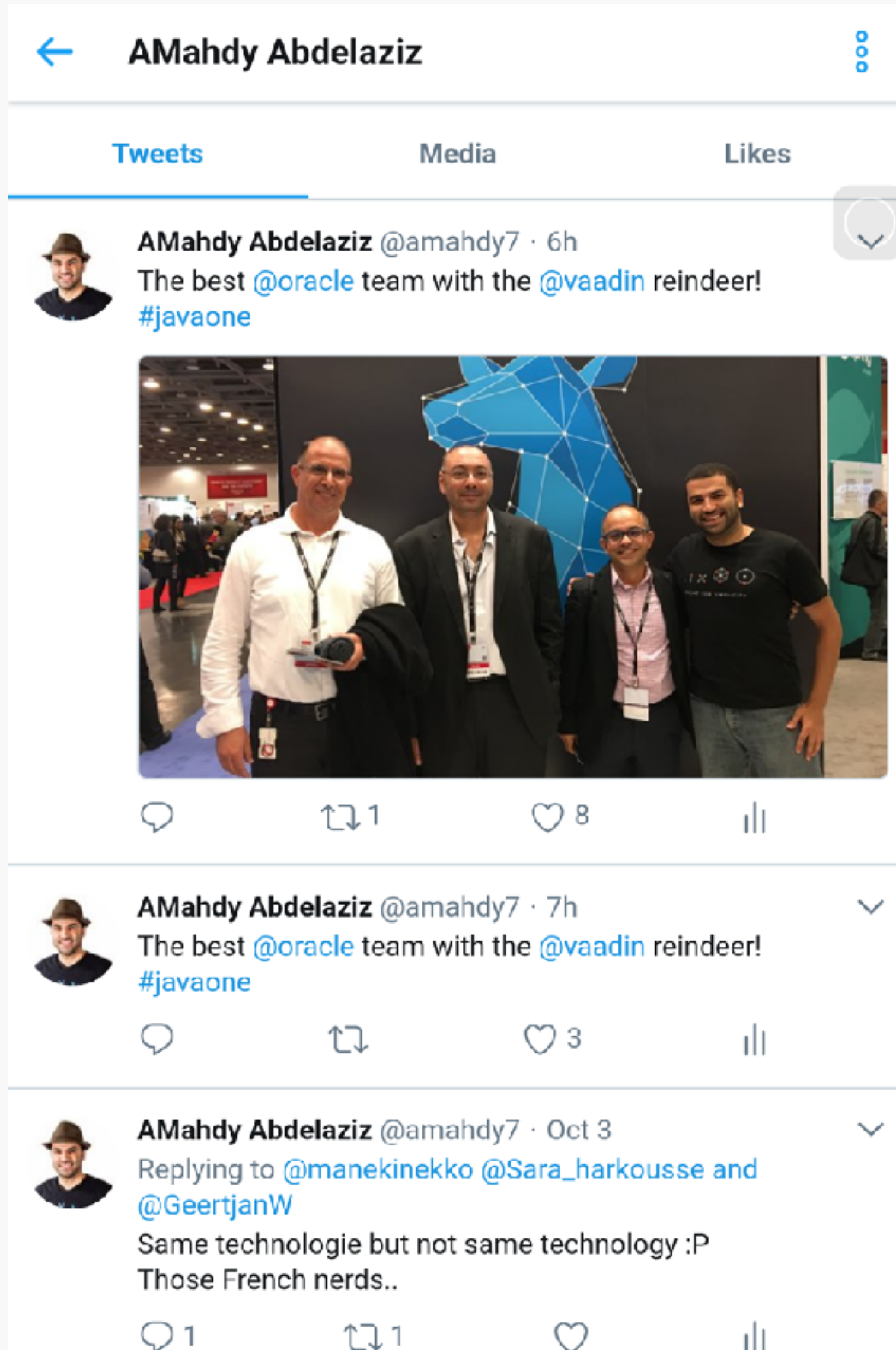
Twitter



- 24 MB
- ? Permission
- ? Autofill
- ? Remember
- ? Updates
- ? Cost



Twitter



- 0.24 MB
- Permission
- Autofill
- Remembers
- Latest
- The Platform

- 24 MB
- ? Permission
- ? Autofill
- ? Remember
- ? Updates
- ? Cost



NATIVE?

OR

CROSS/MULTI PLATFORM?



USER



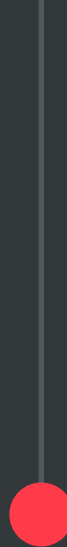
WHY MOBILE PHONE
DOES NOT COME WITH
ALL APPS PRE-INSTALLED?



WHY MOBILE PHONE
DOES NOT COME WITH
ALL APPS PRE-INSTALLED?



THAT'S CALLED
INTERNET



Technology

‘CLOUD’ IS THE DEFAULT

‘MOBILE’ IS TAKING OVER

FANCY TECH

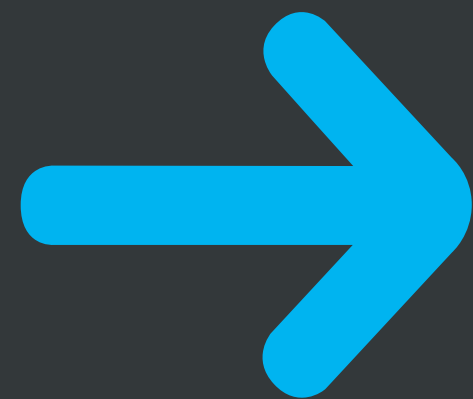
VS.

USER EXPERIENCE

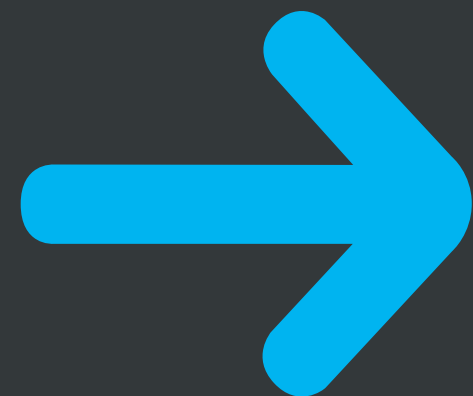
Progressive Web Apps

(PWA)

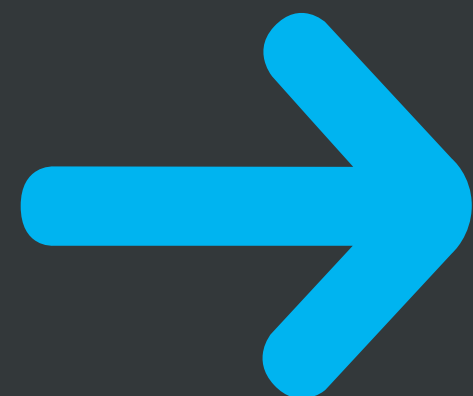
Progressive Web Apps



FAST



OFFLINE-FIRST DESIGN

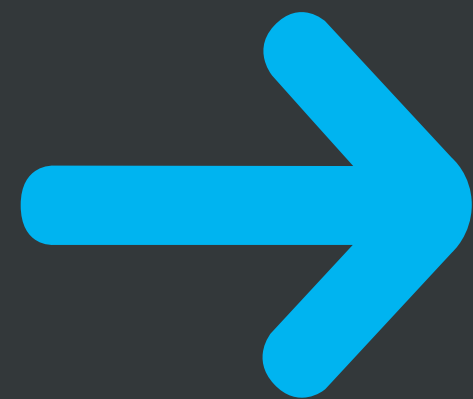


ENGAGING (WEB NOTIFICATIONS)

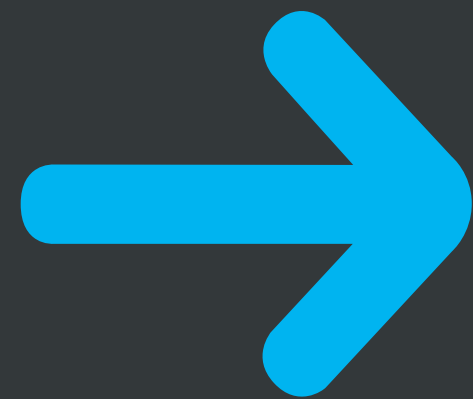


SERVICE WORKER

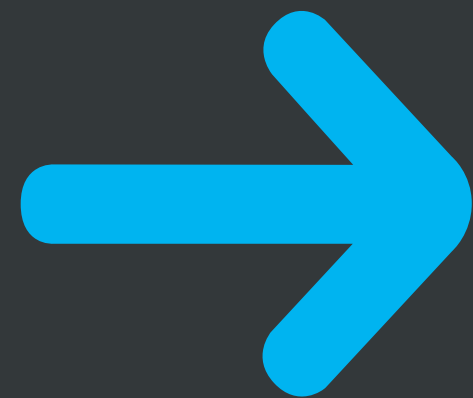
Progressive Web Apps



FAST



OFFLINE-FIRST DESIGN



ENGAGING (WEB NOTIFICATIONS)



SERVICE WORKER

OfflineFirst



OfflineFirst



WebComponents



OfflineFirst



WebComponents



Java

Jfokus

WebComponents



Introduction

What are web components?

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps. Custom components and widgets build on the Web Component standards, will work across modern browsers, and can be used with any JavaScript library or framework that works with HTML.

Web components are based on existing web standards. Features to support web components are currently being added to the HTML and DOM specs, letting web developers easily extend HTML with new elements with encapsulated styling and custom behavior.

Specifications

Web components are based on four main specifications:

Custom Elements

The [Custom Elements specification](#) lays the foundation for designing and using new types of DOM elements.

Shadow DOM

The [shadow DOM specification](#) defines how to use encapsulated style and markup in web components.

HTML imports

The [HTML imports specification](#) defines the inclusion and reuse of HTML documents in other HTML documents.

HTML Template

The [HTML template element specification](#) defines how to declare fragments of markup that go unused at page load, but can be instantiated later on at runtime.

How do I use a web component?

The components on this site provide new HTML elements that you can use in your web pages and web applications.

Using a custom element is as simple as importing it, and using the new tags in an HTML document. For example, to use the [Emoji Rain element](#):

```
<link rel="import" href="../emoji-rain/emoji-rain.html">
...
<emoji-rain active></emoji-rain>
```



Introduction

What are web components?

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps. Custom components and widgets build on the Web Component standards, will work across modern browsers, and can be used with any JavaScript library or framework that works with HTML.

Web components are based on existing web standards. Features to support web components are currently being added to the HTML and DOM specs, letting web developers easily extend HTML with new elements with encapsulated styling and custom behavior.

Specifications

Web components are based on four main specifications:

Custom Elements

The [Custom Elements specification](#) lays the foundation for designing and using new types of DOM elements.

Shadow DOM

The [shadow DOM specification](#) defines how to use encapsulated style and markup in web components.

HTML imports

The [HTML imports specification](#) defines the inclusion and reuse of HTML documents in other HTML documents.

HTML Template

The [HTML template element specification](#) defines how to declare fragments of markup that go unused at page load, but can be instantiated later on at runtime.

How do I use a web component?

The components on this site provide new HTML elements that you can use in your web pages and web applications.

Using a custom element is as simple as importing it, and using the new tags in an HTML document. For example, to use the [Emoji Rain element](#):

```
<link rel="import" href="../emoji-rain/emoji-rain.html">
...
<emoji-rain active></emoji-rain>
```



```
Q Elements Network Sources Timeline Profiles Resources Audits Console AngularJS
▼ <html>
  <head></head>
  ▼ <body>
    ▼ <video controls src="http://www.craftymind.com/factory/html5video/BigBuckBunny_640x360.mp4">
      ▼ #shadow-root (user-agent)
        ▼ <div>
          ▼ <div>
            ▼ <div style="transition: opacity 0.1s; -webkit-transition: opacity 0.1s; opacity: 1;">
              ▶ <input type="button">
              ▶ <input type="range" step="any" max="27.3707">
              <div style=0:00</div>
              <div style="display: none;">0:27</div>
              ▶ <input type="button">
              ▶ <input type="range" step="any" max="1" style>
              ▶ <input type="button" style="display: none;">
              ▶ <input type="button" style>
            </div>
          </div>
        </div>
      </video>
    </body>
```


• Benefits of using Web Components

- Encapsulation
- Reusability



Vaadin Framework

Before WC

- Works only with Vaadin Framework
- Limited to own add-ons
- Challenging to modify/ test/ debug

After WC

- Standard: works anywhere
- Can work with any Standard WC
- Separation of concern

OfflineFirst



WebComponents



Java

Jfokus

OfflineFirst Design





OFFLINE-FIRST IS
THE ONLY WAY TO
ACHIEVE A TRUE 100%
ALWAYS-ON USER
EXPERIENCE.*

*ASSUMING THE DEVICE IS
RELIABLE

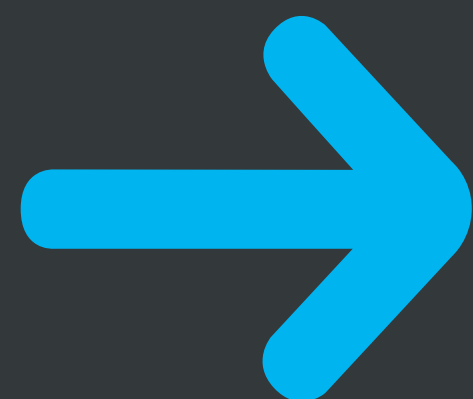
Solution



CACHING



OFFLINE STORAGE



DATA REPLICATION



FIREBASE

Solution

pouchdb / pouchdb

Watch 296 Star 9,356 Fork 960

Code Issues 255 Pull requests 4 Projects 0 Wiki Insights

👤 - PouchDB is a pocket-sized database. <https://pouchdb.com/>

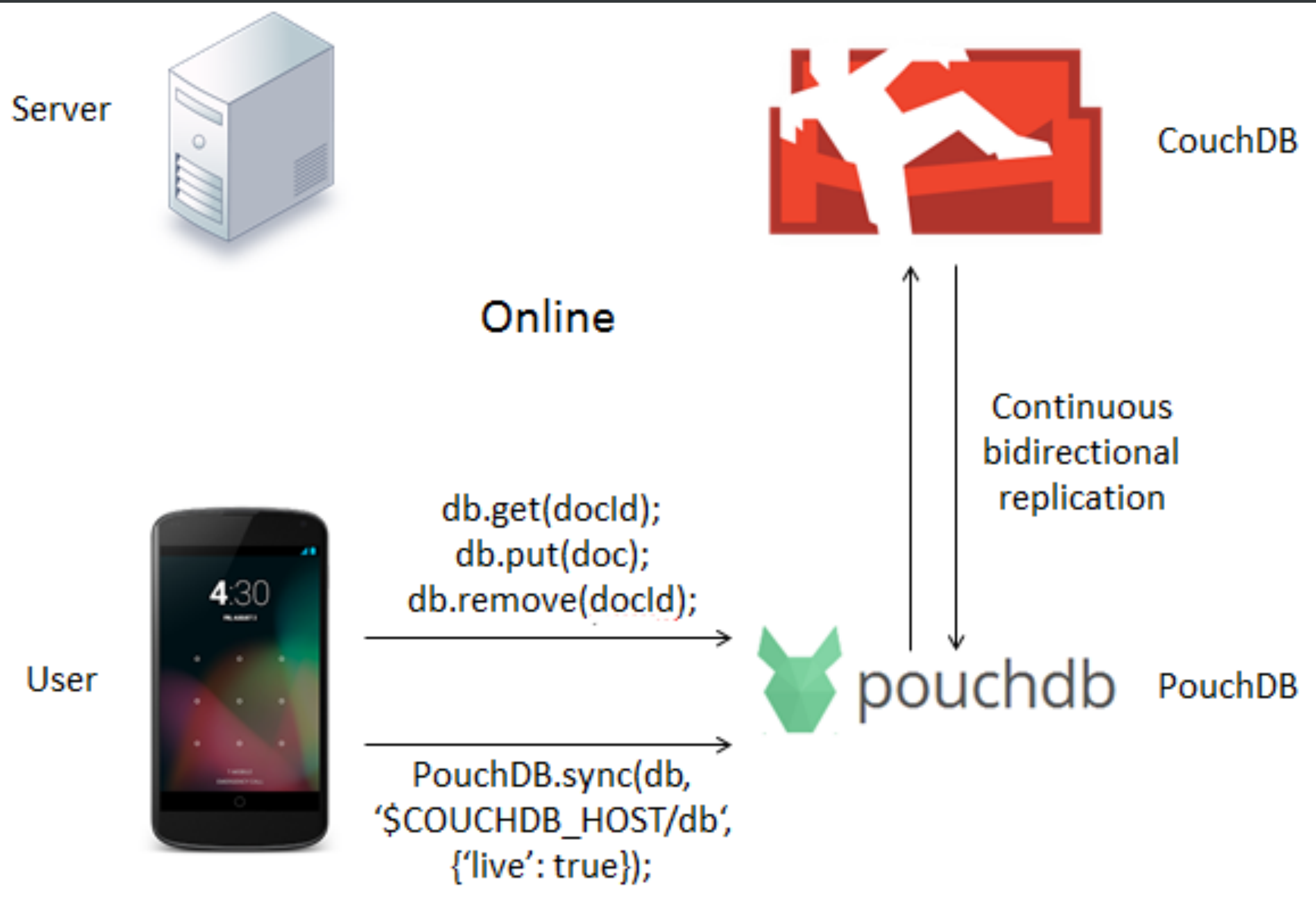
javascript database pouchdb couchdb

4,218 commits 1,099 branches 62 releases 267 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

🟢 greenkeeper[bot] committed with daleharvey (#6727) - chore(package): update eslint to version 4.6.0 Latest commit 62280f7 a day ago

bin	(#6623) - Update code to uglify package	2 months ago
docs	(#6721) - Correct capitalization of GitHub	5 days ago
packages/node_modules	(#6443) - Support seq_interval for changes and use during replication	a month ago
scripts	(#913) - Add basic tests to new suite, compile tests to html	4 years ago
tests	(#4575) - Ensure databases are cleaned up	5 days ago
.eslintignore	(#5128) - split PouchDB into a monorepo with separate packages	a year ago



Server



CouchDB

Offline

User



```
db.get(docId);  
db.put(doc);  
db.remove(docId);
```



pouchdb

PouchDB

```
PouchDB.sync(db,  
'$COUCHDB_HOST/db',  
{'live': true});
```


Demo

Offline-First App with WebCom x

localhost:8080

AMahdy

#	First Name	Last Name	Email
0	Jonathan	Perez	jonathan.perez@company.com
1	Madison	Rodriguez	madison.rodriguez@company.com
2	Mila	Gonzalez	mila.gonzalez@company.com
3	Madeline	Perry	madeline.perry@company.com
4	Victoria	Rodriguez	victoria.rodriguez@company.com
5	Taylor	Smith	taylor.smith@company.com
6	Leo	Long	leo.long@company.com
7	Leah	Myers	leah.myers@company.com
8	Leo	Sanchez	julian.sanchez@company.com
9	Logan	Sanders	logan.sanders@company.com
10	Madison	Brooks	madison.brooks@company.com
11	Grace	Rogers	grace.rogers@company.com
12	Angel	Ortiz	angel.ortiz@company.com
13	Isabelle	Baker	isabelle.baker@company.com
14	Adam	Gutierrez	adam.gutierrez@company.com

Editor

First Name
Victoria

Last Name
Rodriguez

E-Mail
victoria.rodriguez@company.com

UPDATE

Search

First Name
Leo

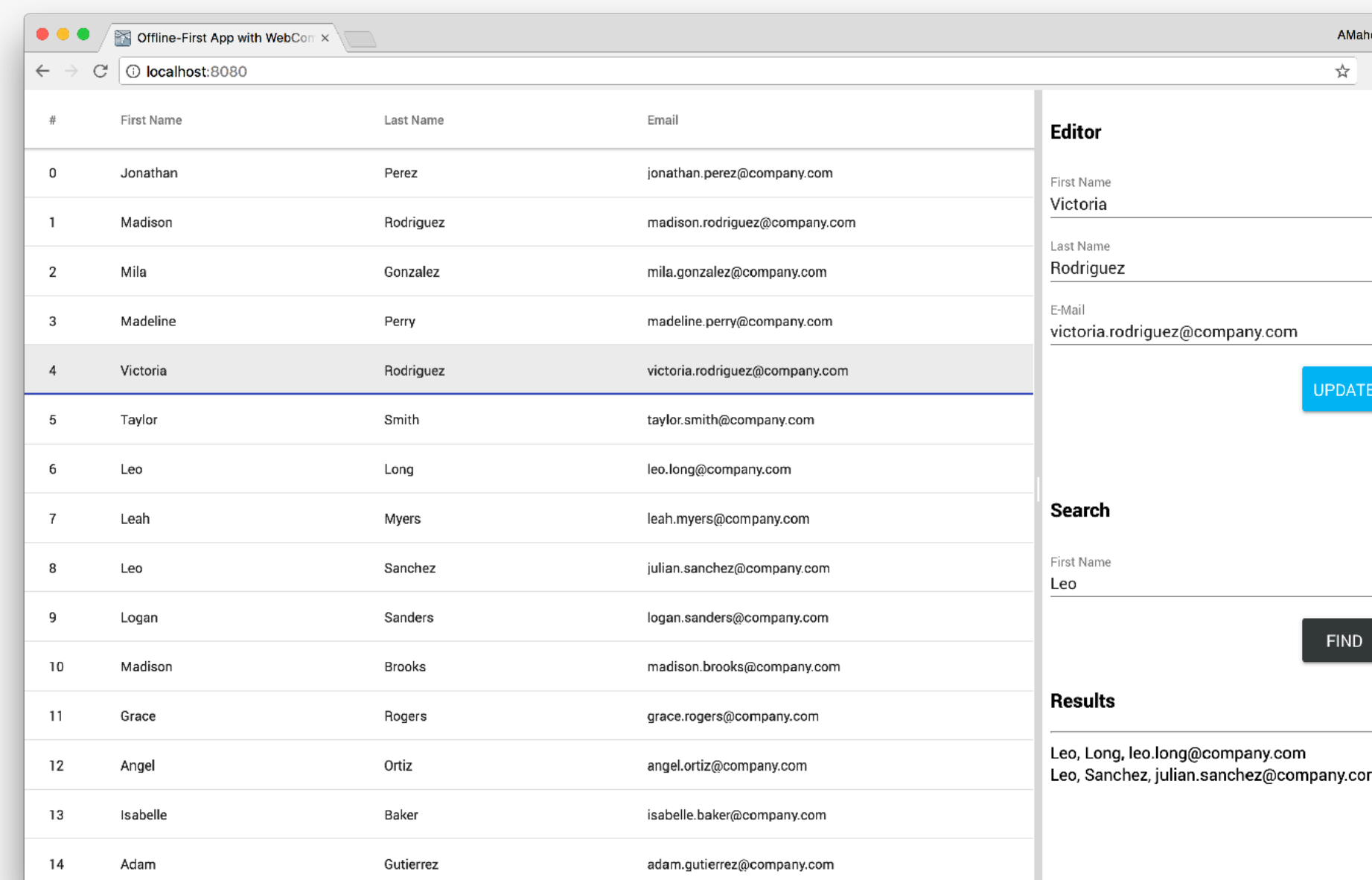
FIND

Results

Leo, Long, leo.long@company.com
Leo, Sanchez, julian.sanchez@company.com

Demo

<https://github.com/amahdy/offline-first-app>



The screenshot shows a web application interface. On the left, there is a table with 15 rows of employee data. The right side of the interface contains an 'Editor' form for editing the selected record (index 4, Victoria Rodriguez) and a 'Search' section with a text input and a 'FIND' button. Below the search section, the 'Results' of the search are displayed.

#	First Name	Last Name	Email
0	Jonathan	Perez	jonathan.perez@company.com
1	Madison	Rodriguez	madison.rodriguez@company.com
2	Mila	Gonzalez	mila.gonzalez@company.com
3	Madeline	Perry	madeline.perry@company.com
4	Victoria	Rodriguez	victoria.rodriguez@company.com
5	Taylor	Smith	taylor.smith@company.com
6	Leo	Long	leo.long@company.com
7	Leah	Myers	leah.myers@company.com
8	Leo	Sanchez	julian.sanchez@company.com
9	Logan	Sanders	logan.sanders@company.com
10	Madison	Brooks	madison.brooks@company.com
11	Grace	Rogers	grace.rogers@company.com
12	Angel	Ortiz	angel.ortiz@company.com
13	Isabelle	Baker	isabelle.baker@company.com
14	Adam	Gutierrez	adam.gutierrez@company.com

Editor

First Name
Victoria

Last Name
Rodriguez

E-Mail
victoria.rodriguez@company.com

UPDATE

Search

First Name
Leo

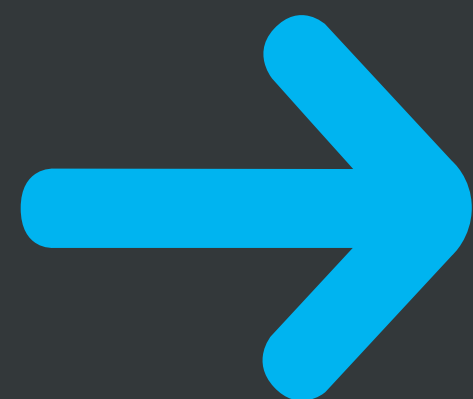
FIND

Results

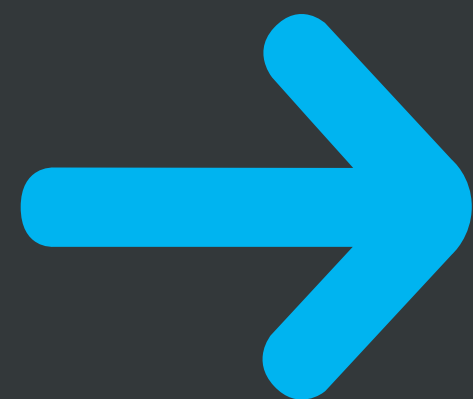
Leo, Long, leo.long@company.com
Leo, Sanchez, julian.sanchez@company.com



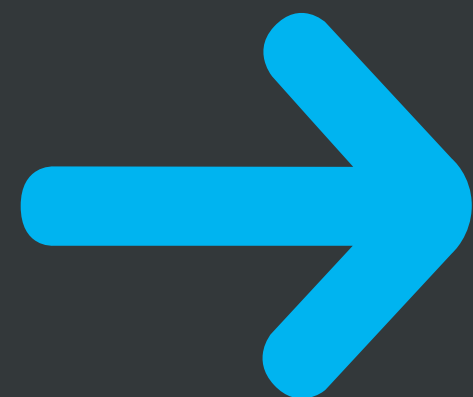
Challenges



INITIAL LOAD TIME



SECURITY OF STORED DATA



RACE CONDITION

OfflineFirst

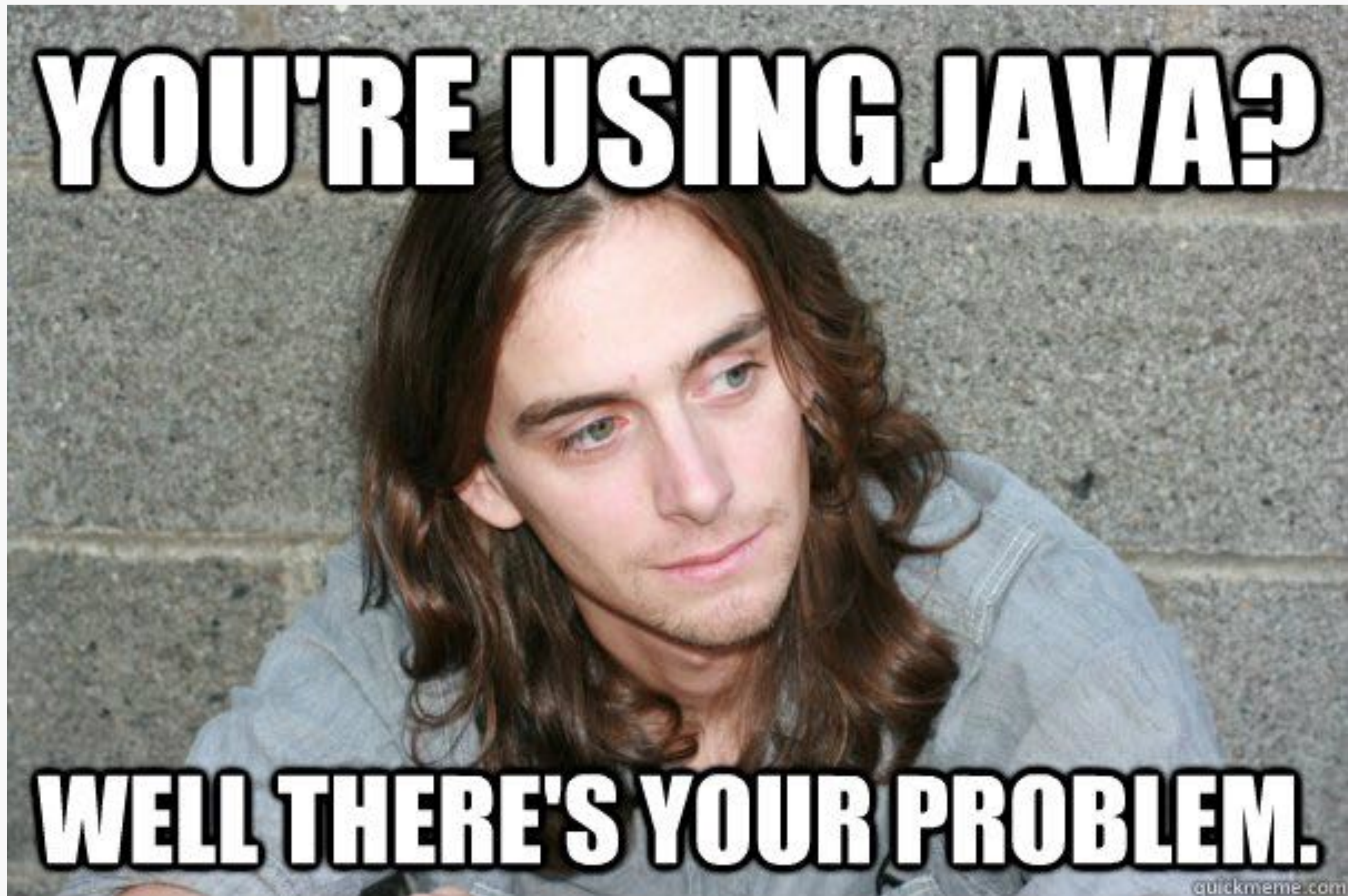


WebComponents

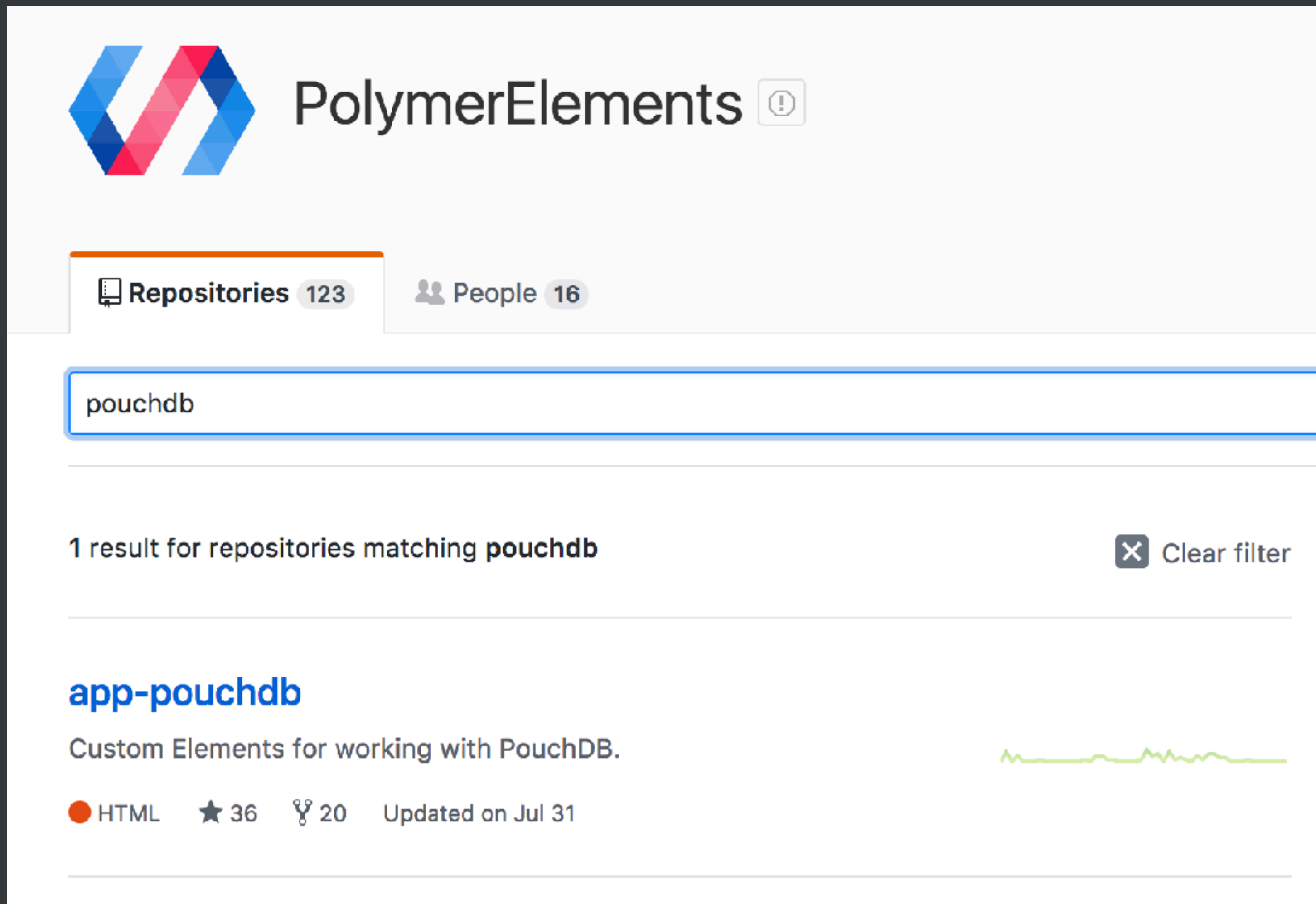


Java

Jfokus



Solution



The screenshot shows the GitHub interface for the PolymerElements repository. At the top, the PolymerElements logo and name are displayed. Below this, there are two tabs: 'Repositories' (123) and 'People' (16). A search bar contains the text 'pouchdb'. Below the search bar, it says '1 result for repositories matching pouchdb'. A button labeled 'Clear filter' is visible. The search result is for the repository 'app-pouchdb', which is described as 'Custom Elements for working with PouchDB'. Below the description, there are icons for HTML (a red circle), stars (36), forks (20), and the last update date (Jul 31). A green line graph is also visible on the right side of the repository entry.

 PolymerElements 

 **Repositories** 123  **People** 16

pouchdb

1 result for repositories matching **pouchdb**  Clear filter

app-pouchdb
Custom Elements for working with PouchDB.

 HTML  36  20 Updated on Jul 31

Vaadin Framework

Before WC

<https://dzone.com/articles/using-web-components-in-plain-java>

Step 1: Create a New Vaadin Add-On Project

Step 2: Add the Required Dependencies

Add the [Elements add-on](#) dependency.

Step 4: Wrap the Element in a Vaadin Component

Step 5: Implement a test UI

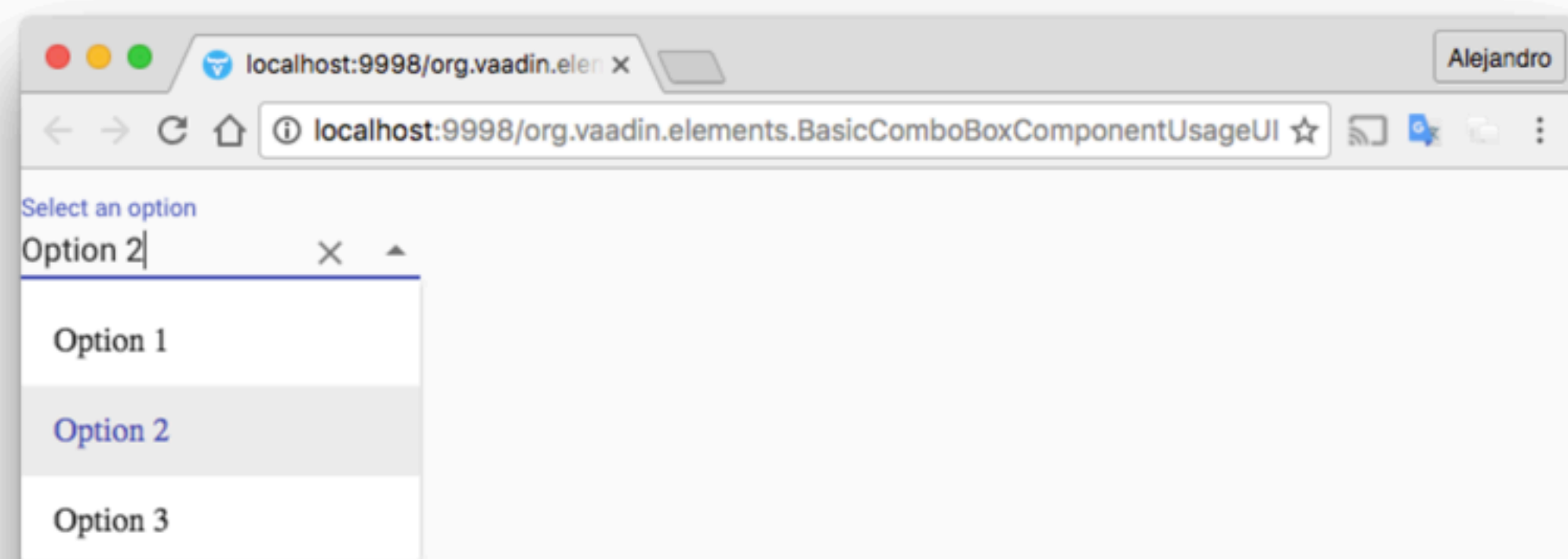
Step 6: Run the Test Application

Run the `Server.main()` method from your IDE or using Maven:

```
1 mvn package exec:java -Dexec.mainClass="org.vaadin.elements.uiserver.Server" -Dexec.classpathScope=test
```

Point your browser to <http://localhost:9998> and click the `BasicComboBoxComponentUsageUI` link.

The following is a screenshot of the application:



After WC

Vaadin Framework

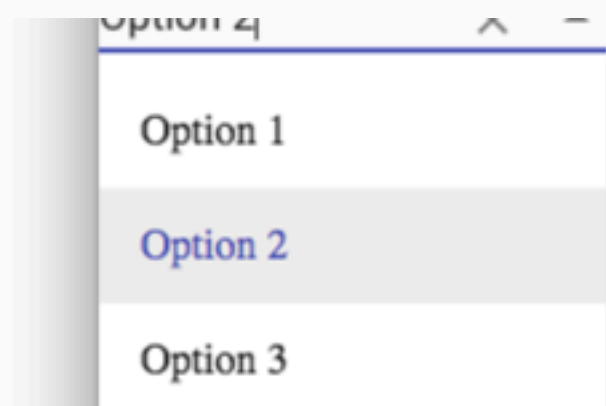
Before WC

After WC

USING WEB COMPONENTS INSIDE THE FRAMEWORK:

Since Vaadin announced a beautiful set of [elements](#), they have grown quite a lot with very active development and community contributions. And there have been some [blog posts](#) showing how to integrate Web Components inside Vaadin Framework. Thanks to the [Elements add-on](#), things are relatively easier.

I was interested in specifically trying to solve the variable row height problem, by using [vaadin-grid](#) instead of [Grid](#). And I have to confess that it was not that easy. Besides the fact of manually defining the element component in Java, I did not manage to define the `vaadin-grid-column` template easily to set up the columns.



Vaadin Framework

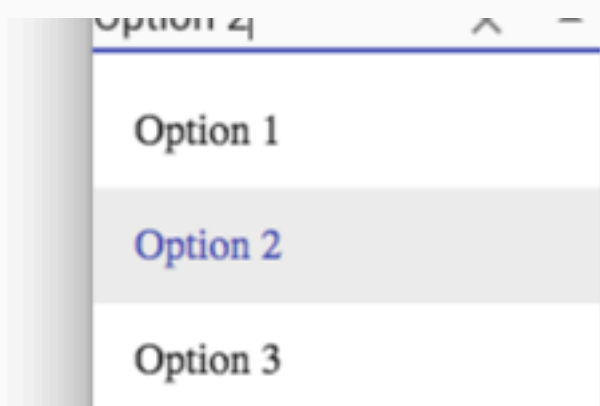
Before WC

After WC

USING WEB COMPONENTS INSIDE THE FRAMEWORK:

Since Vaadin announced a beautiful set of [elements](#), they have grown quite a lot with very active development and community contributions. And there have been some [blog posts](#) showing how to integrate Web Components inside Vaadin Framework. Thanks to the [Elements add-on](#), things are relatively easier.

I was interested in specifically trying to solve the variable row height problem, by using [vaadin-grid](#) instead of [Grid](#). And I have to confess that it was not that easy. Besides the fact of manually defining the element component in Java, I did not manage to define the `vaadin-grid-column` template easily to set up the columns.



Vaadin Framework

Before WC

<https://dzone.com/articles/using-web-components-in-plain-java>

Step 1: Create a New Vaadin Add-On Project

Step 2: Add the Required Dependencies

Add the [Elements add-on](#) dependency.

Step 4: Wrap the Element in a Vaadin Component

Step 5: Implement a test UI

Step 6: Run the Test Application

Run the `Server.main()` method from your IDE or using Maven:

```
1 mvn package exec:java -Dexec.mainClass="org.vaadin.elements.uiserver.Server" -Dexec.classpathScope=test
```

PO USING WEB COMPONENTS INSIDE THE FRAMEWORK:

geUI link.

Th Since Vaadin announced a beautiful set of [elements](#), they have grown quite a lot with very active development and community contributions. And there have been some [blog posts](#) showing how to integrate Web Components inside Vaadin Framework. Thanks to the [Elements add-on](#), things are relatively easier.

I was interested in specifically trying to solve the variable row height problem, by using [vaadin-grid](#) instead of [Grid](#). And I have to confess that it was not that easy. Besides the fact of manually defining the element component in Java, I did not manage to define the `vaadin-grid-column` template easily to set up the columns.

Option 1

Option 2

Option 3

After WC

```
@Tag("my-label") // declare which HTML element to use
public class Label extends Component {

    public void setText(String text) {
        getElement().setText(text);
    }

    public String getText() {
        return getElement().getText();
    }
}
```



Practical Test

1. Mobile-first design
2. Touch-first design
3. Coffee-first design

● THANK YOU!
- BUDDHA

DEMO APP

<https://github.com/amahdy/offline-first-app>

JAVA WITH PWA
[HTTP://J.MP/JAVAPWA](http://j.mp/javapwa)

@AMAHDY7

vaadin}>