OFFLINE-FIRST DESIGN & WEB COMPONENTS

AMAHDY ABDELAZIZ
WWW.AMAHDY.NET
@AMAHDY7

vaadin

vaadin

Massive Tech. Specs

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

vaadin

vaadin

€10000

vaadin

USER

vaadin

DESIGN NOTE #00

TECH SPECS

VS.

USABILITY

vaadin

AMAZON

vaadin

AMAZON

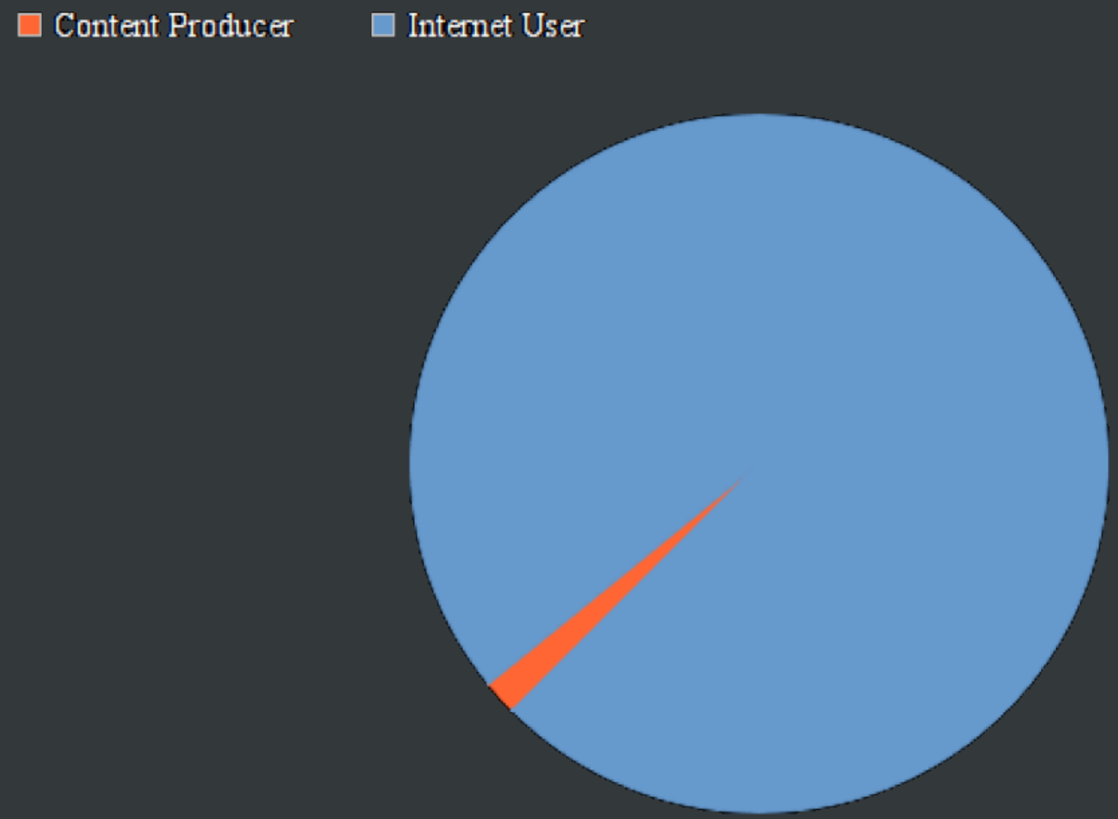
vaadin

AMAZON

vaadin

AMAZON

vaadin



vaadin

DESIGN NOTE #01

FAMILIAR TOOLS

VS.

TARGET AUDIENCE

vaadin

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 @manekinekko @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/6I5JEFzhd

vaadin

Twitter

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

vaadin

Twitter

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

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

vaadin

NATIVE?

OR

CROSS/MULTI PLATFORM?

vaadin

USER

vaadin

vaadin


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


THAT'S CALLED
INTERNET

vaadin

Technology

‘CLOUD’ IS THE DEFAULT

‘MOBILE’ IS TAKING OVER

vaadin

DESIGN NOTE #10

FANCY TECH

VS.

USER EXPERIENCE

vaadin

Progressive Web Apps

(PWA)

vaadin

Progressive Web Apps

FAST

OFFLINE-FIRST DESIGN

ENGAGING (WEB NOTIFICATIONS)

SERVICE WORKER

vaadin}>

Progressive Web Apps

FAST

OFFLINE-FIRST DESIGN

ENGAGING (WEB NOTIFICATIONS)

SERVICE WORKER

vaadin}>

OfflineFirst

vaadin

OfflineFirst

WebComponents

vaadin

OfflineFirst

WebComponents

Java

Jfokus

vaadin

WebComponents

vaadin

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>
```

vaadin

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

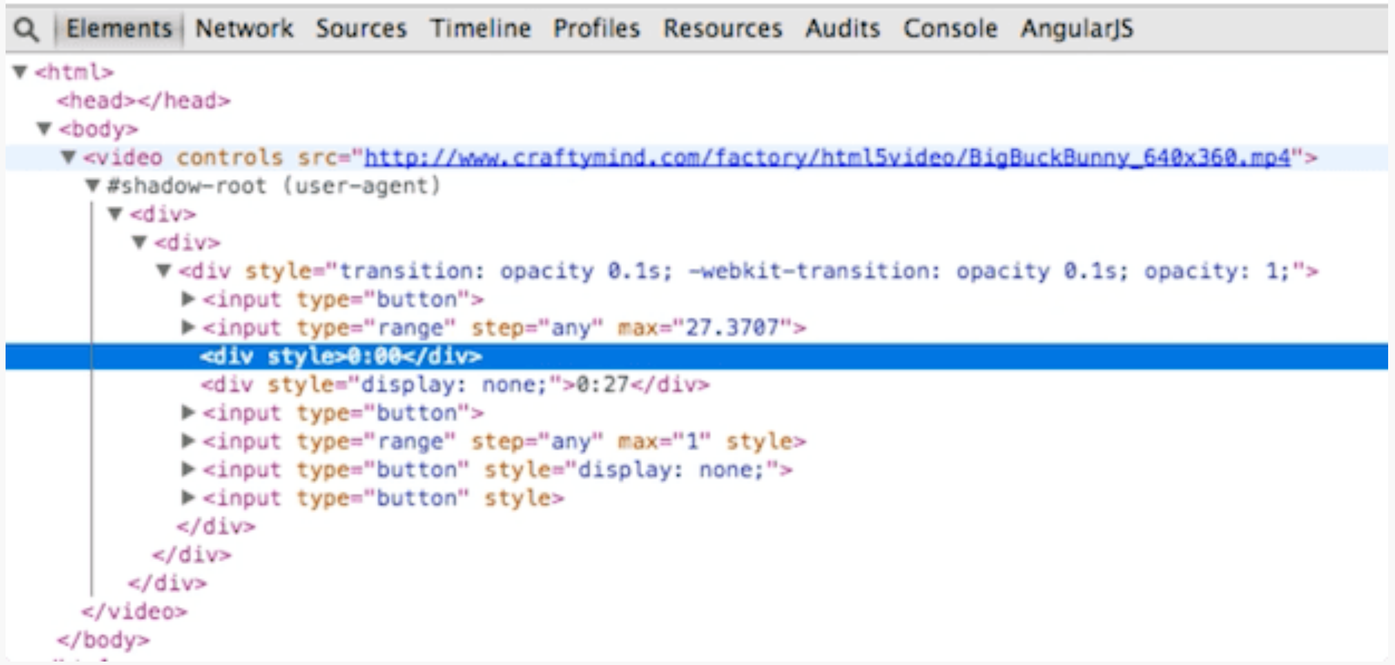
The HTML template element specification defines how to declare fragments of markup that go unused at page load, but can be instantiated later on at runtime.

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:

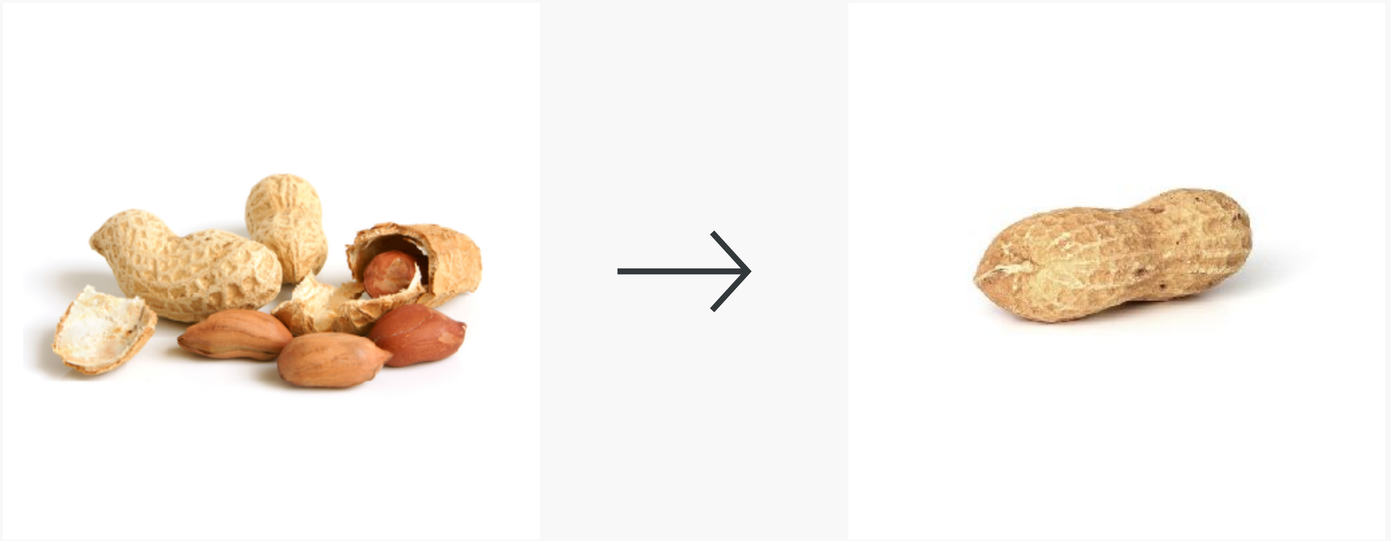
```
<link rel="import" href="../emoji-rain/emoji-rain.html">
...
<emoji-rain active></emoji-rain>
```



vaadin

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

vaadin

OfflineFirst

WebComponents

Java

Jfokus

vaadin

OfflineFirst Design

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

*ASSUMING THE DEVICE IS RELIABLE

vaadin

Solution

CACHING

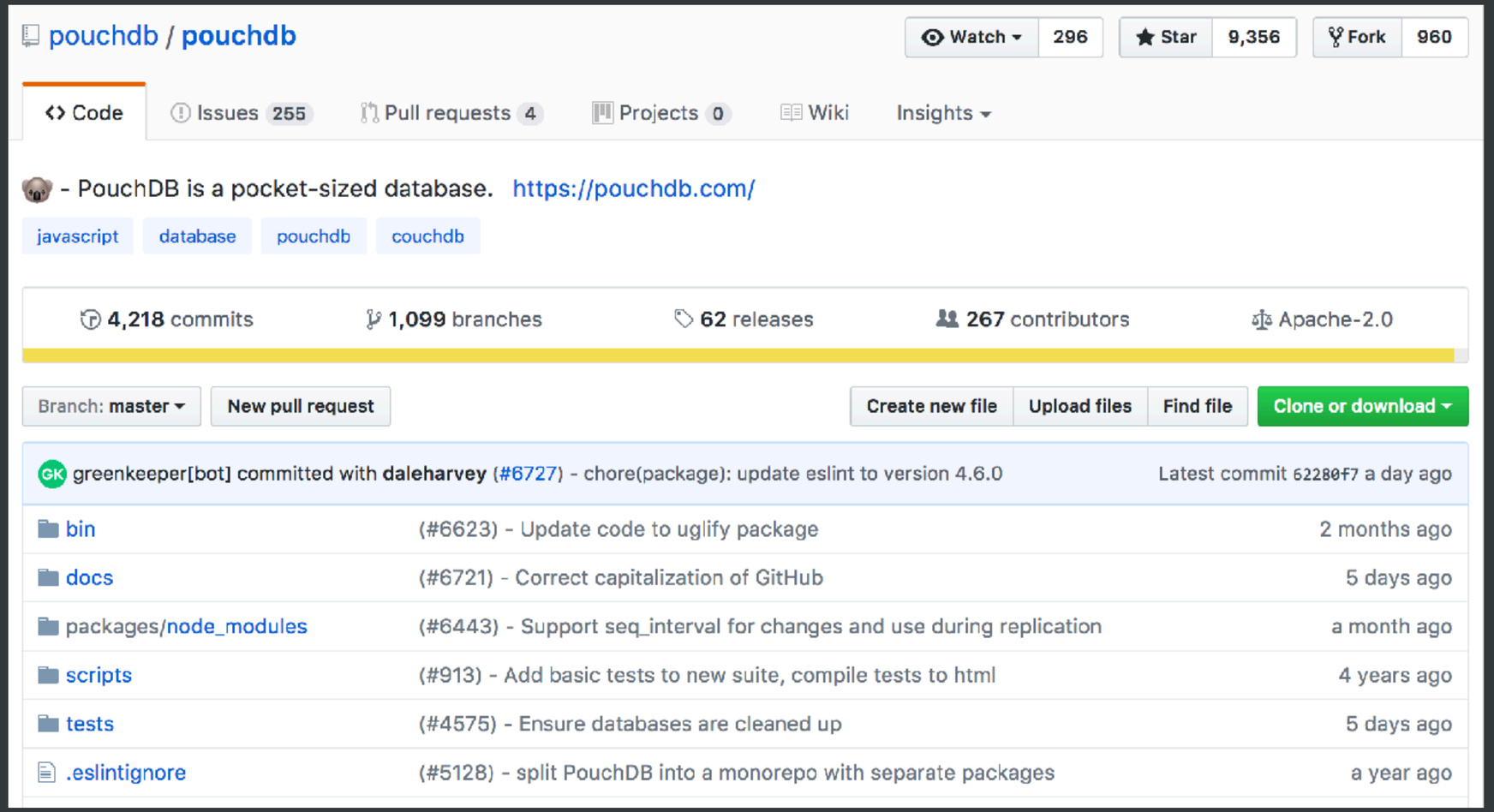
OFFLINE STORAGE

DATA REPLICATION

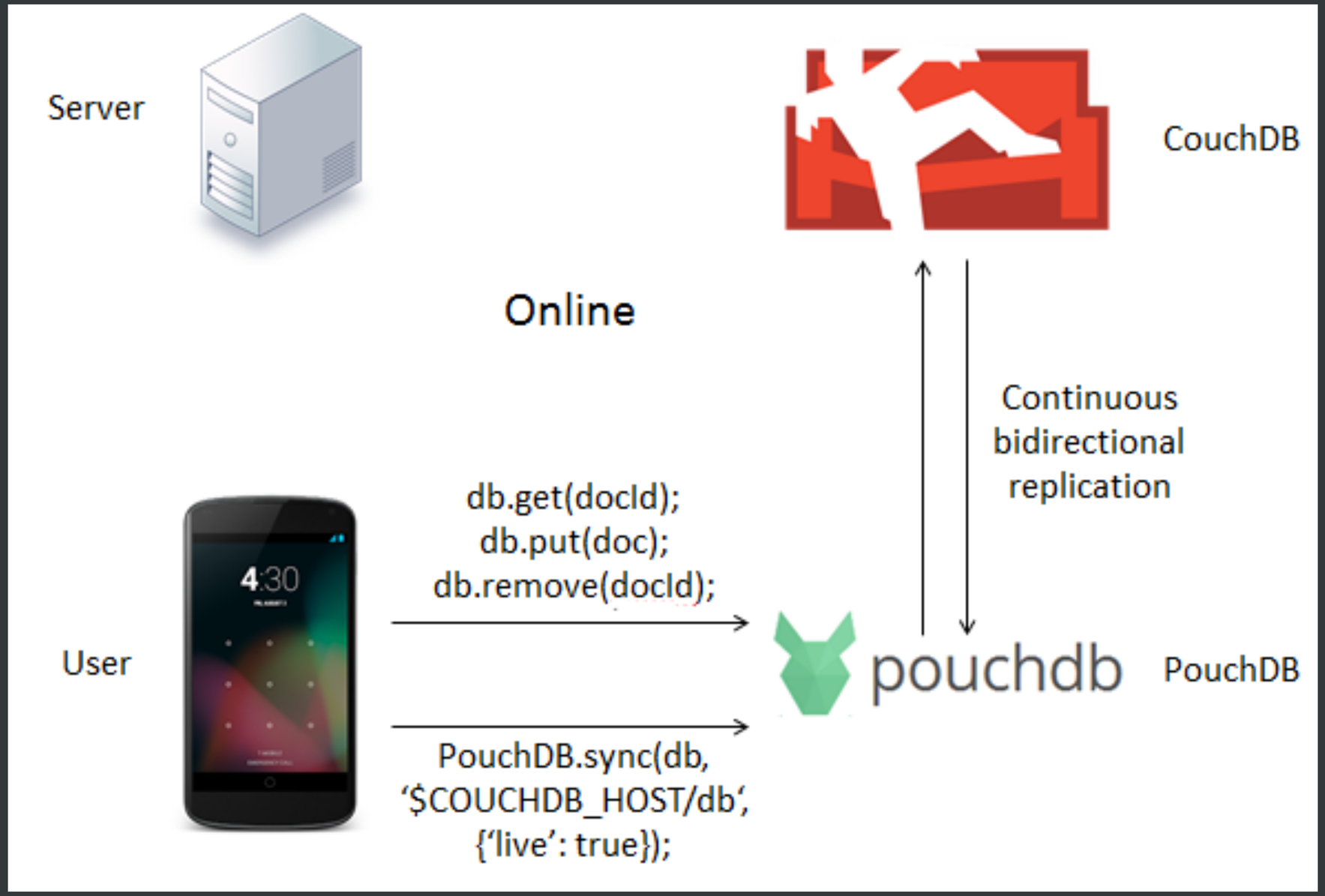
FIREBASE

vaadin}>

Solution



vaadin


Server
CouchDB
Online
Continuous
bidirectional
replication
db.get(docId);
db.put(doc);
db.remove(docId);
User
4:30
pouchdb
PouchDB
PouchDB.sync(db,
'$COUCHDB_HOST/db',
{'live': true});

vaadin

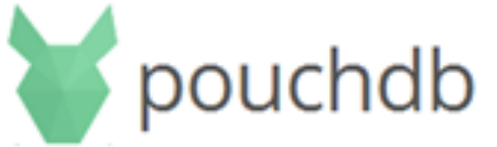
Server

CouchDB

Offline

User

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

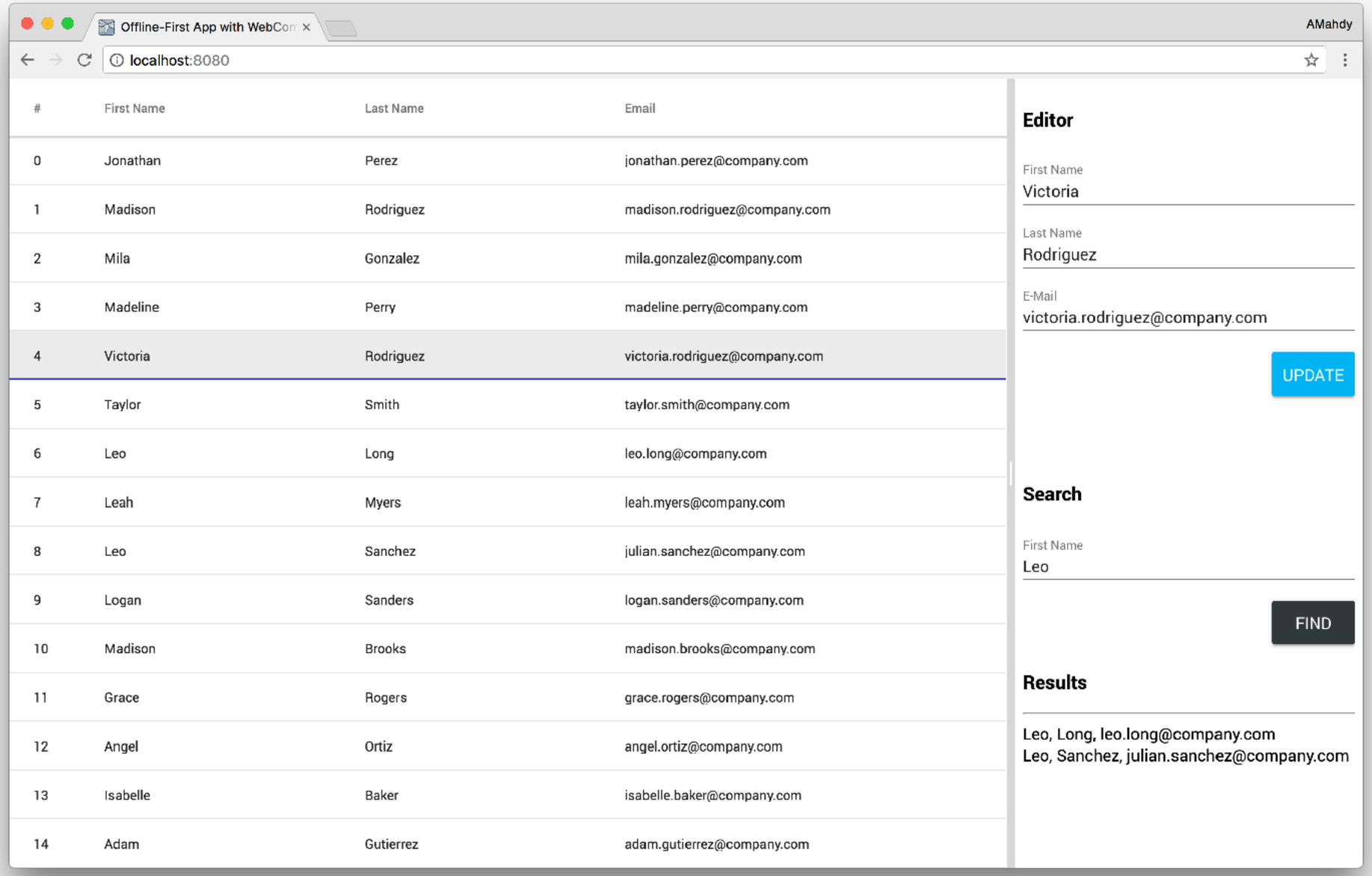


PouchDB

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

vaadin

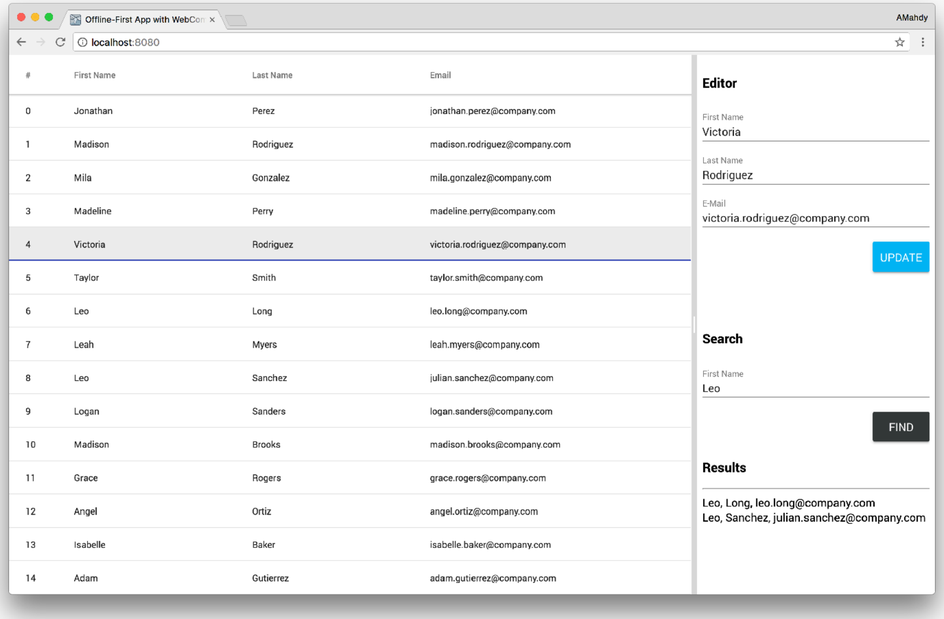
Demo


Offline-First App with WebCon
AMahdy
localhost:8080
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

vaadin

Demo

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



vaadin

Challenges

INITIAL LOAD TIME

SECURITY OF STORED DATA

RACE CONDITION

vaadin}>

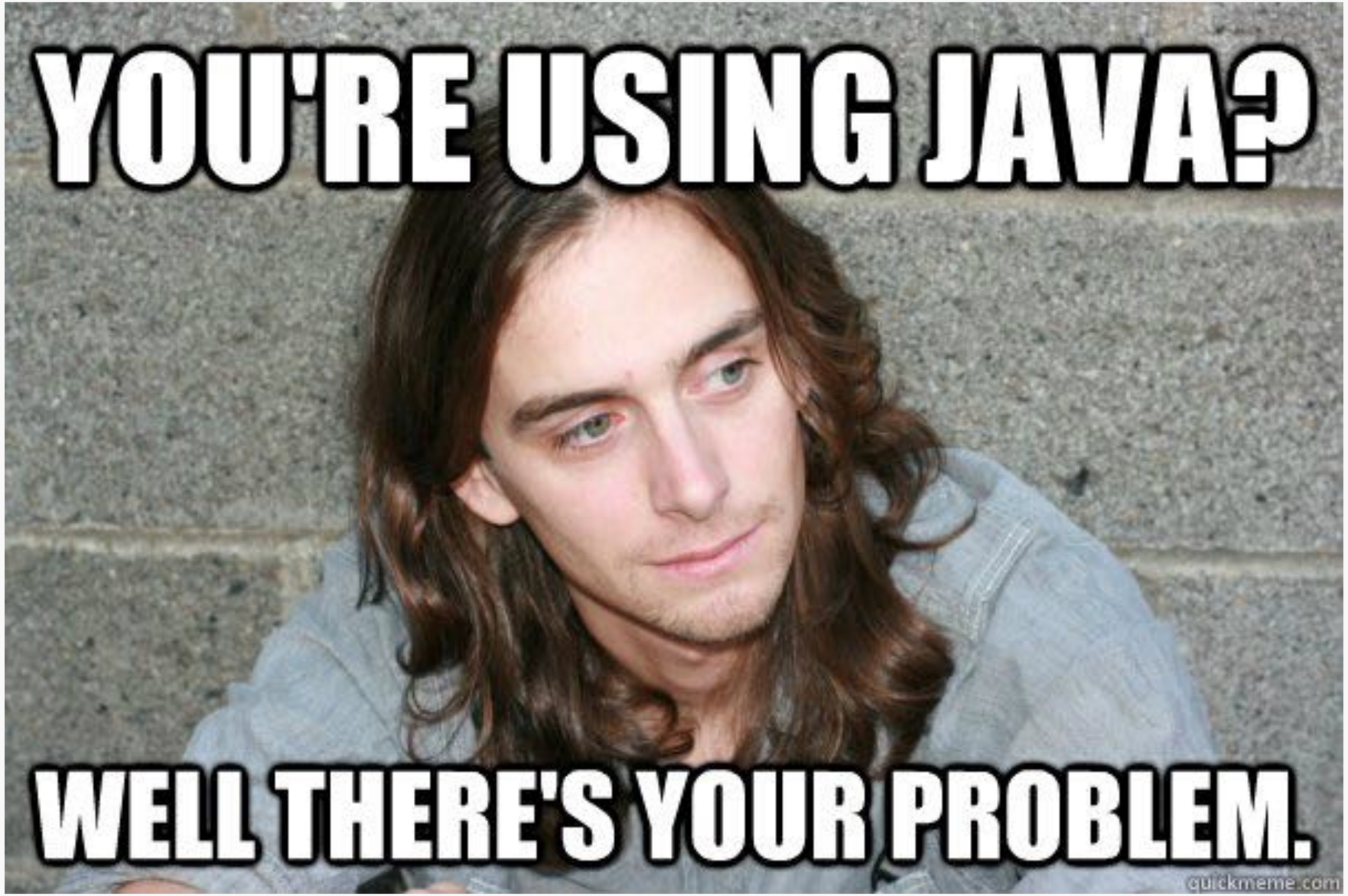
OfflineFirst

WebComponents

Java

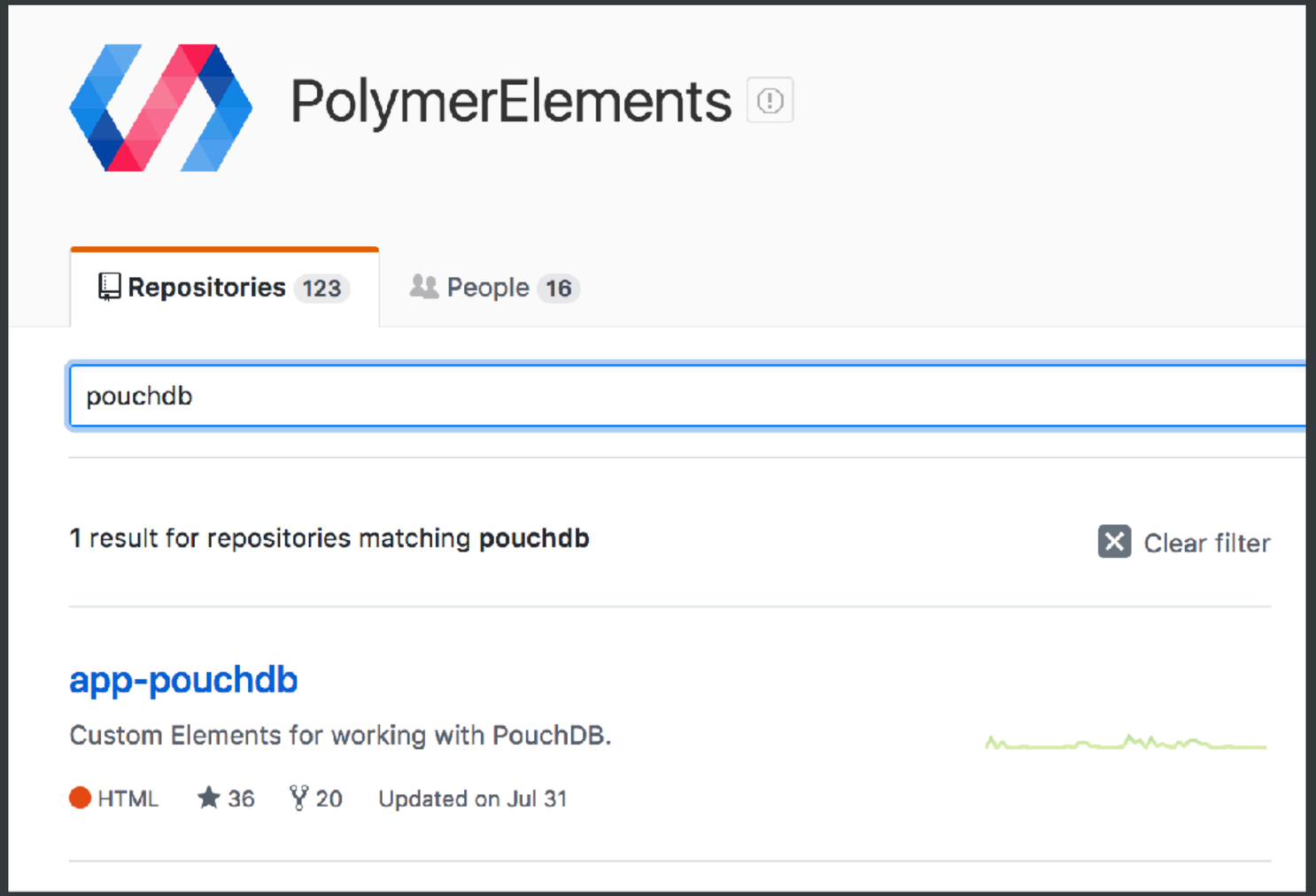
Jfokus

vaadin



vaadin

Solution



vaadin

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

Step Step 4: Wrap the Element in a

Define Implement

tep 5: Implement a test UI

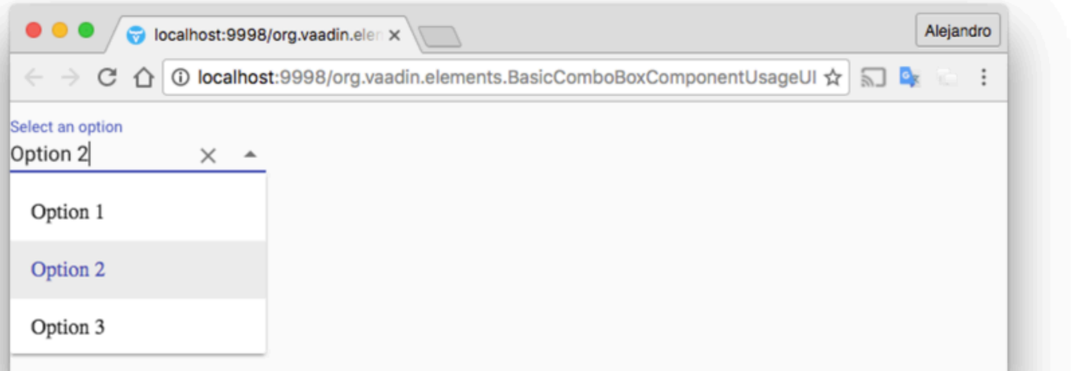
Step 6: Run the Test Application

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

```
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

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

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.

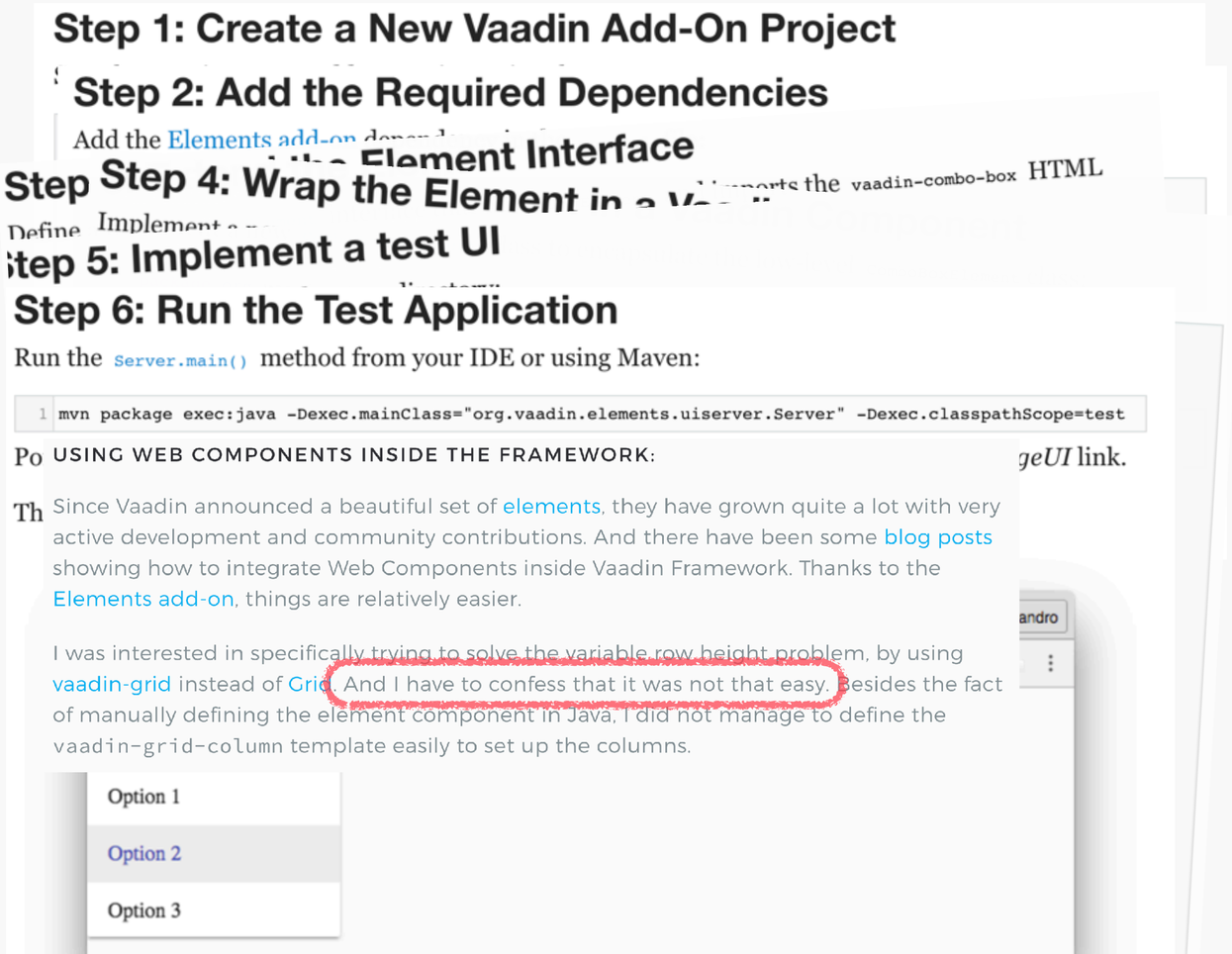
I was interested in specifically trying to solve the variable row height problem, by using vaadin-grid instead of Grid. And I have to confess that it was not that easy. Besides the fact of manually defining the element component in Java, I did not manage to define the `vaadin-grid-column` template easily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

vaadin

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

Step 4: Wrap the Element in a

Define Implement a

Step 5: Implement a test UI

Step 6: Run the Test Application

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

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

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.

Option 1

Option 2

Option 3

After WC

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

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

vaadin

Practical Test

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

vaadin

THANK YOU!

-BUDDHA

DEMO APP

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

JAVA WITH PWA

HTTP://J.MP/JAVAPWA

@AMAHDY7

vaadin