Serving CSV from the Browser

Ben Foxall,@benjaminbenben,@pusher

CSV

Things I really like about CSV:

- Accessibility - data for everyone
- It's the start of something, not the end

The thing I’m going to talk about:

Your data ➡ ☁

As a Person, I would like my data in a csv file, so that I can do some cool stuff with it

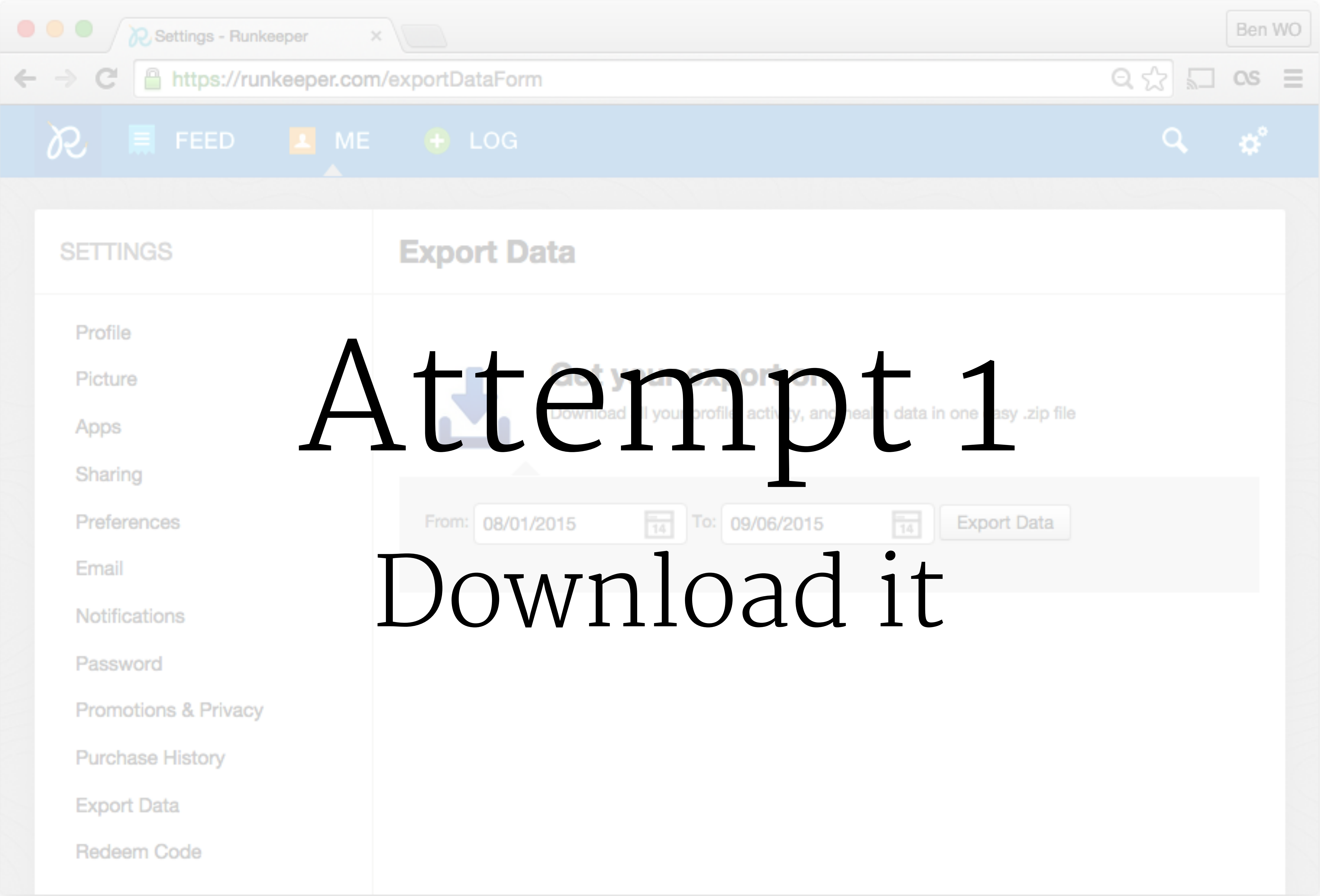
- Runkeeper – *gps routes of running activities*

How do we get our data back?

Attempt 1

Attempt 1

Download it

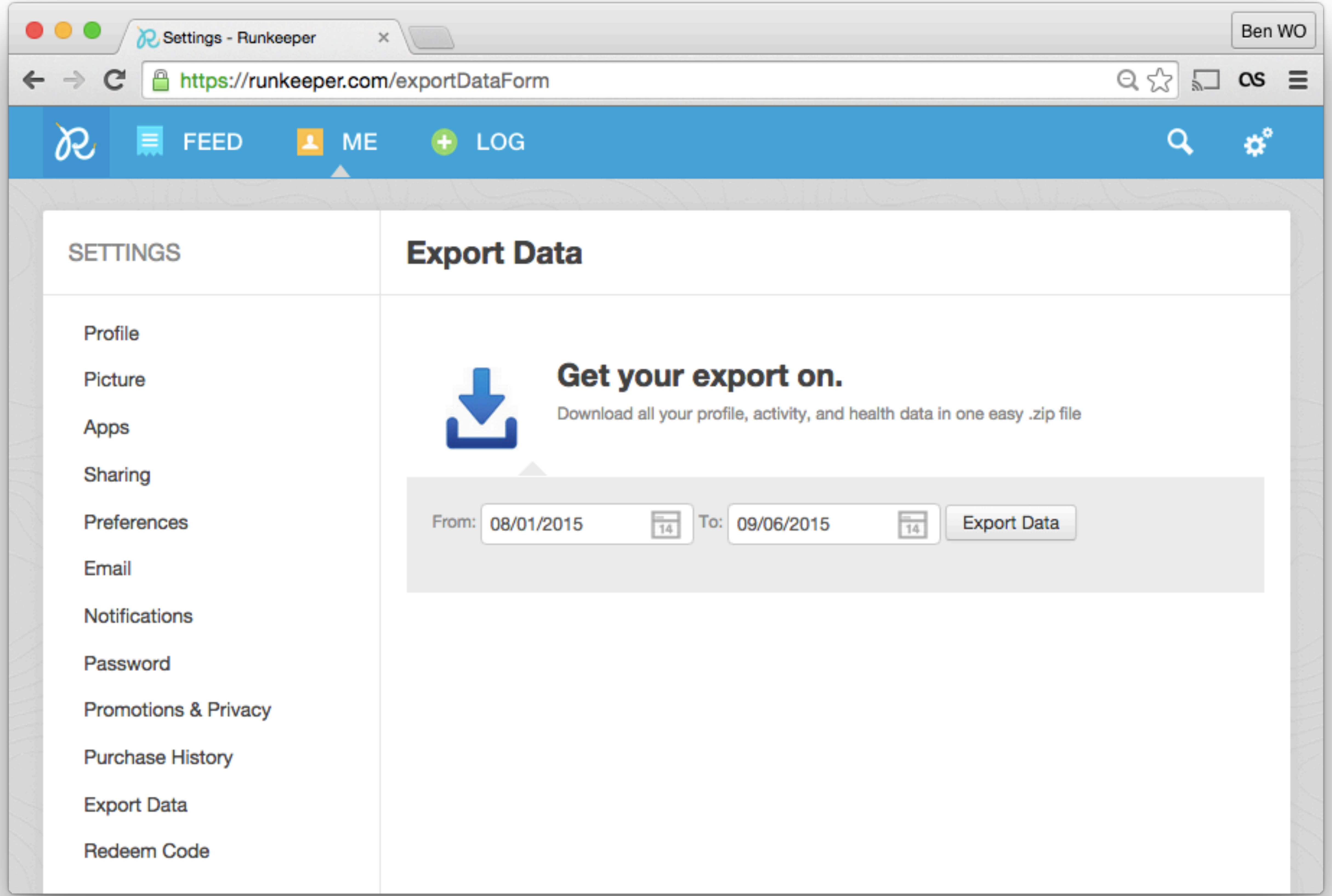


Export Data

Get your export on

Download all your profile, activity, and health data in one easy .zip file

From: 08/01/2015 To: 09/06/2015 Export Data



SETTINGS

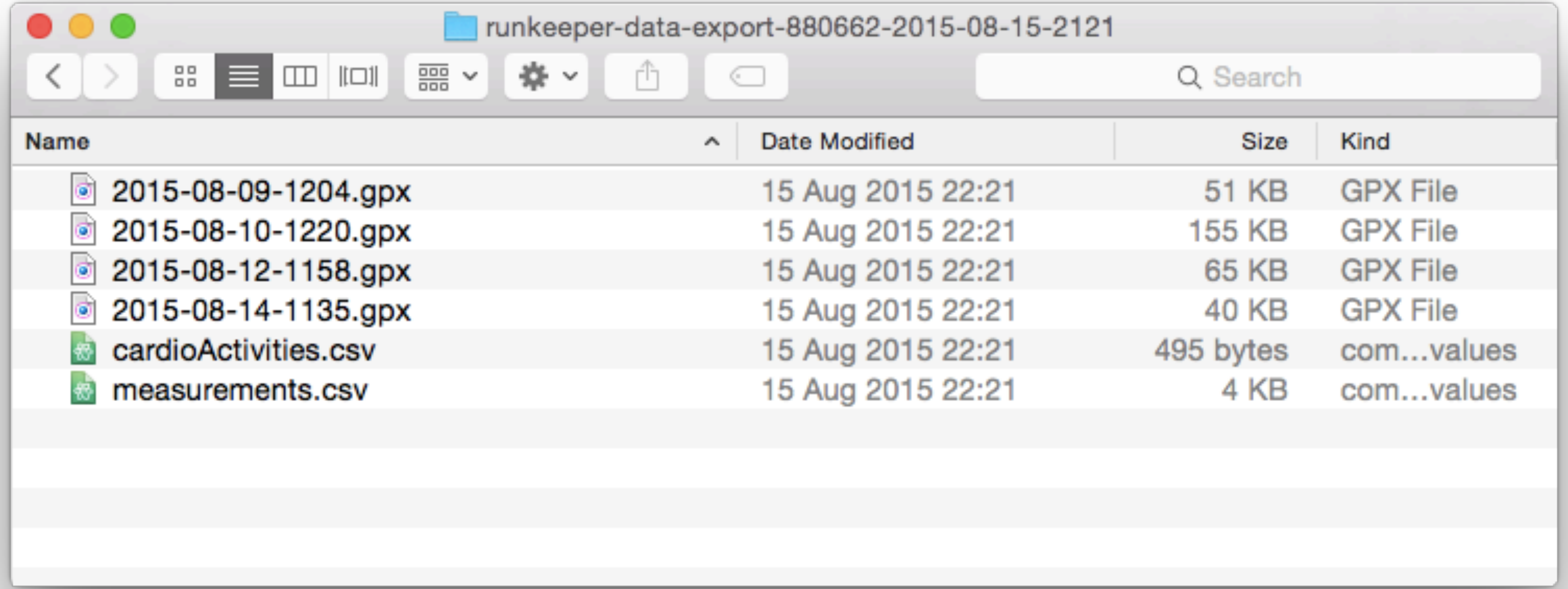
- Profile
- Picture
- Apps
- Sharing
- Preferences
- Email
- Notifications
- Password
- Promotions & Privacy
- Purchase History
- Export Data
- Redeem Code

Export Data

Get your export on.

Download all your profile, activity, and health data in one easy .zip file

From: 08/01/2015 To: 09/06/2015 Export Data



runkeeper-data-export-880662-2015-08-15-2121

Name	Date Modified	Size	Kind
2015-08-09-1204.gpx	15 Aug 2015 22:21	51 KB	GPX File
2015-08-10-1220.gpx	15 Aug 2015 22:21	155 KB	GPX File
2015-08-12-1158.gpx	15 Aug 2015 22:21	65 KB	GPX File
2015-08-14-1135.gpx	15 Aug 2015 22:21	40 KB	GPX File
cardioActivities.csv	15 Aug 2015 22:21	495 bytes	com...values
measurements.csv	15 Aug 2015 22:21	4 KB	com...values

👌

- Job Done

❌

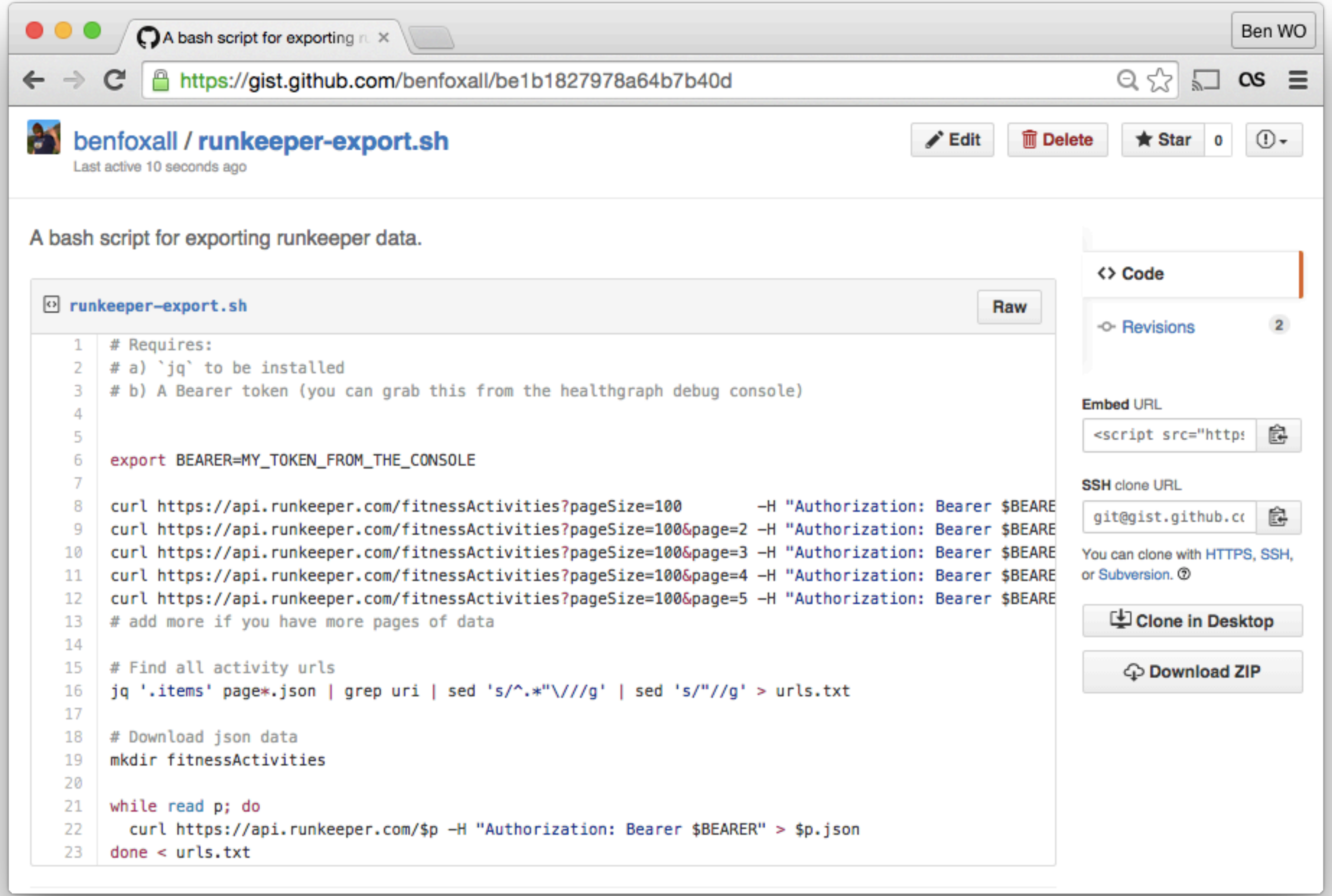
- No format control
- Functionality might change/disappear
- Need to go online to retrieve different times

Attempt 2

Attempt 2

Write a script

```
export BEARER=MY_TOKEN_FROM_THE_CONSOLE

curl https://api.runkeeper.com/fitnessActivities?pageSize=100        -H "Authorization: Bearer $BEARER" > page1.json
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=2 -H "Authorization: Bearer $BEARER" > page2.json
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=3 -H "Authorization: Bearer $BEARER" > page3.json
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=4 -H "Authorization: Bearer $BEARER" > page4.json
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=5 -H "Authorization: Bearer $BEARER" > page5.json

jq '.items' page*.json | grep uri | sed 's/^.*"\///g' | sed 's/"//g' > urls.txt

mkdir fitnessActivities

while read p; do
  curl https://api.runkeeper.com/$p -H "Authorization: Bearer $BEARER" > $p.json
done < urls.txt
```



benfoxall / runkeeper-export.sh

Last active 10 seconds ago

A bash script for exporting runkeeper data.

runkeeper-export.sh

```
# Requires:
# a) `jq` to be installed
# b) A Bearer token (you can grab this from the healthgraph debug console)


export BEARER=MY_TOKEN_FROM_THE_CONSOLE

curl https://api.runkeeper.com/fitnessActivities?pageSize=100        -H "Authorization: Bearer $BEARE
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=2 -H "Authorization: Bearer $BEARE
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=3 -H "Authorization: Bearer $BEARE
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=4 -H "Authorization: Bearer $BEARE
curl https://api.runkeeper.com/fitnessActivities?pageSize=100&page=5 -H "Authorization: Bearer $BEARE
# add more if you have more pages of data

# Find all activity urls
jq '.items' page*.json | grep uri | sed 's/^.*"\///g' | sed 's/"//g' > urls.txt

# Download json data
mkdir fitnessActivities

while read p; do
  curl https://api.runkeeper.com/$p -H "Authorization: Bearer $BEARER" > $p.json
done < urls.txt
```

You can clone with HTTPS, SSH, or Subversion.

👌

- format choices
- sharable
- offline

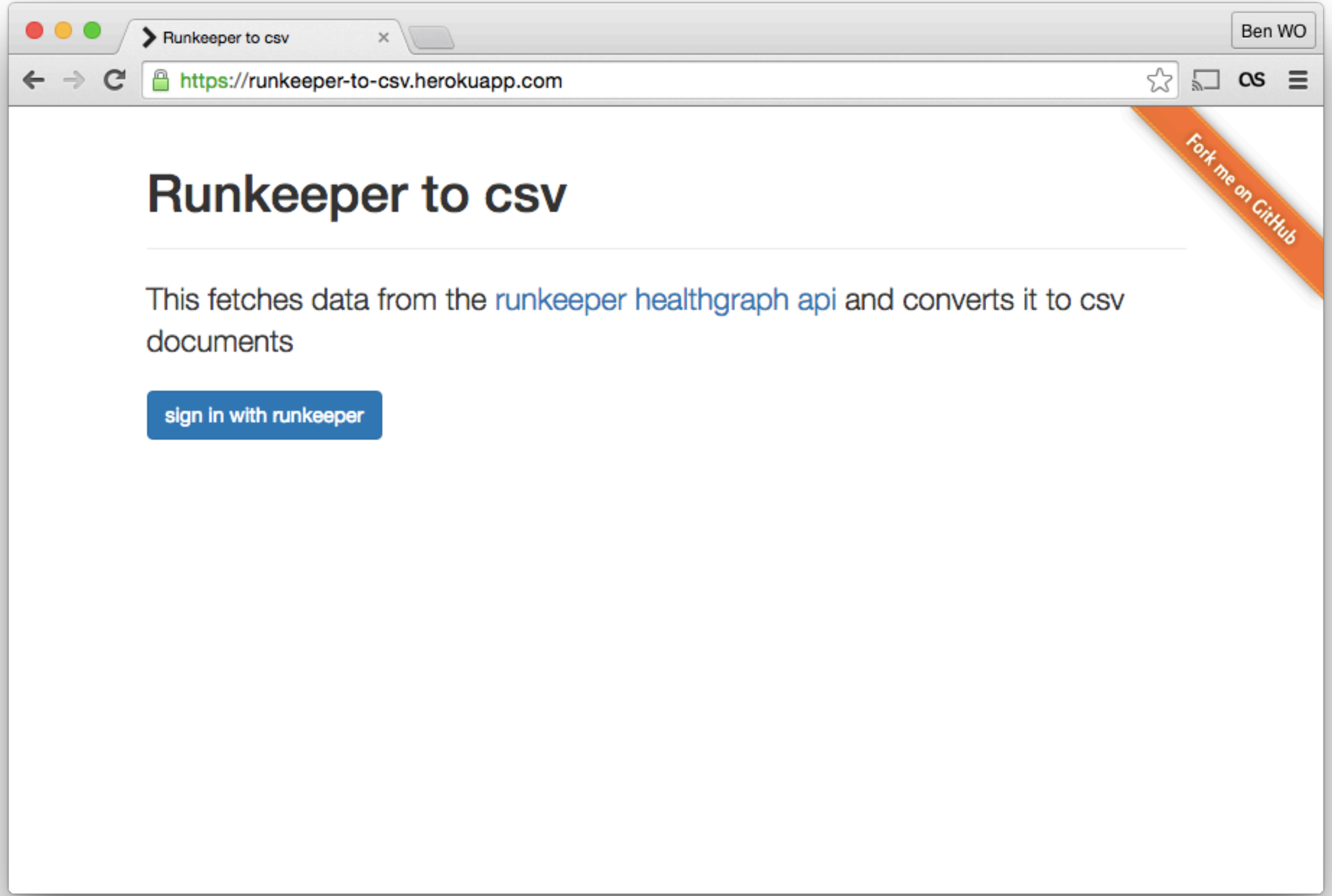
❌

- Inaccessible

Attempt 3

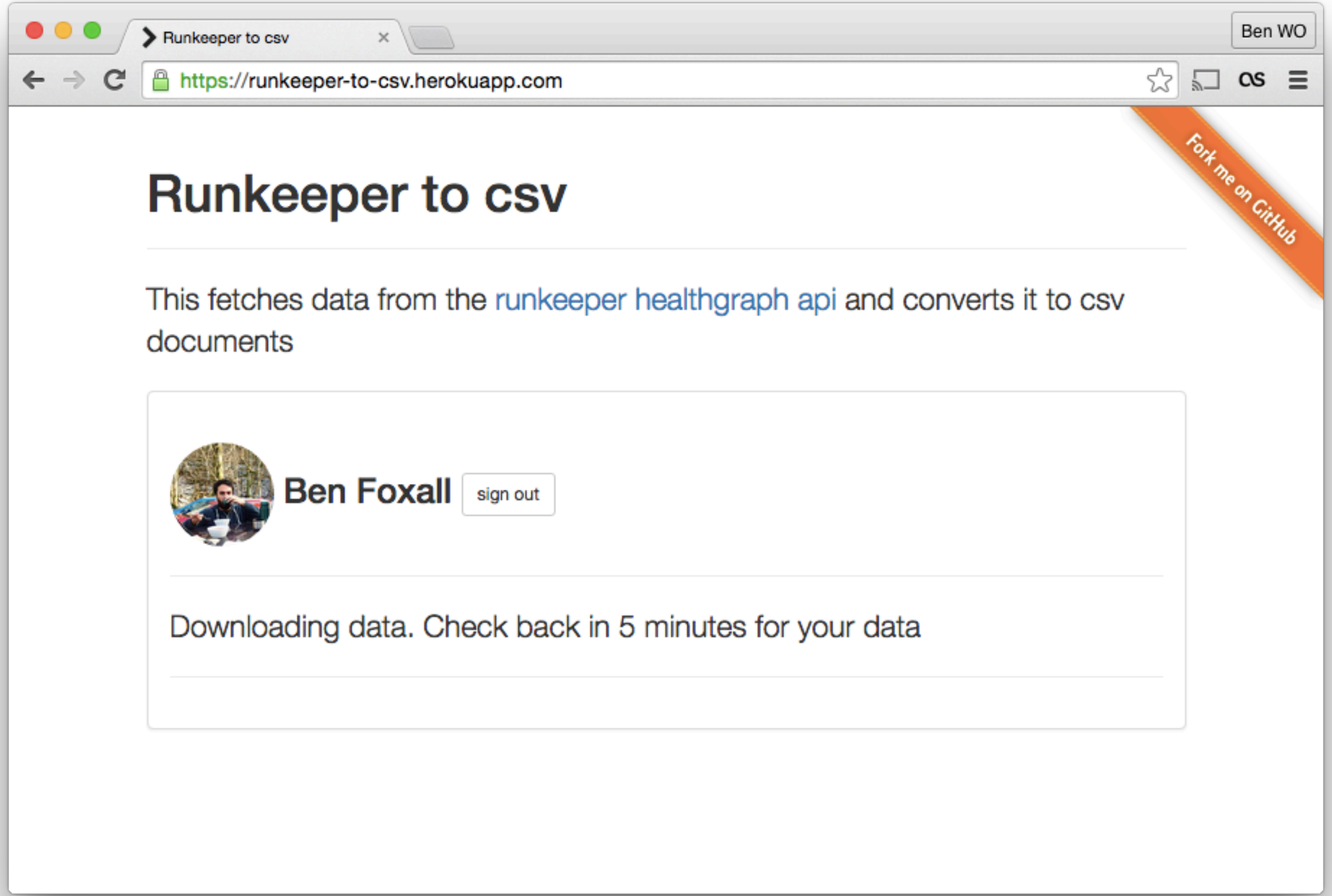
Attempt 3

Make a web service



Runkeeper to csv

This fetches data from the runkeeper healthgraph api and converts it to csv documents



Runkeeper to csv

This fetches data from the runkeeper healthgraph api and converts it to csv documents

Ben Foxall

Downloading data. Check back in 5 minutes for your data

👌

- Accessible

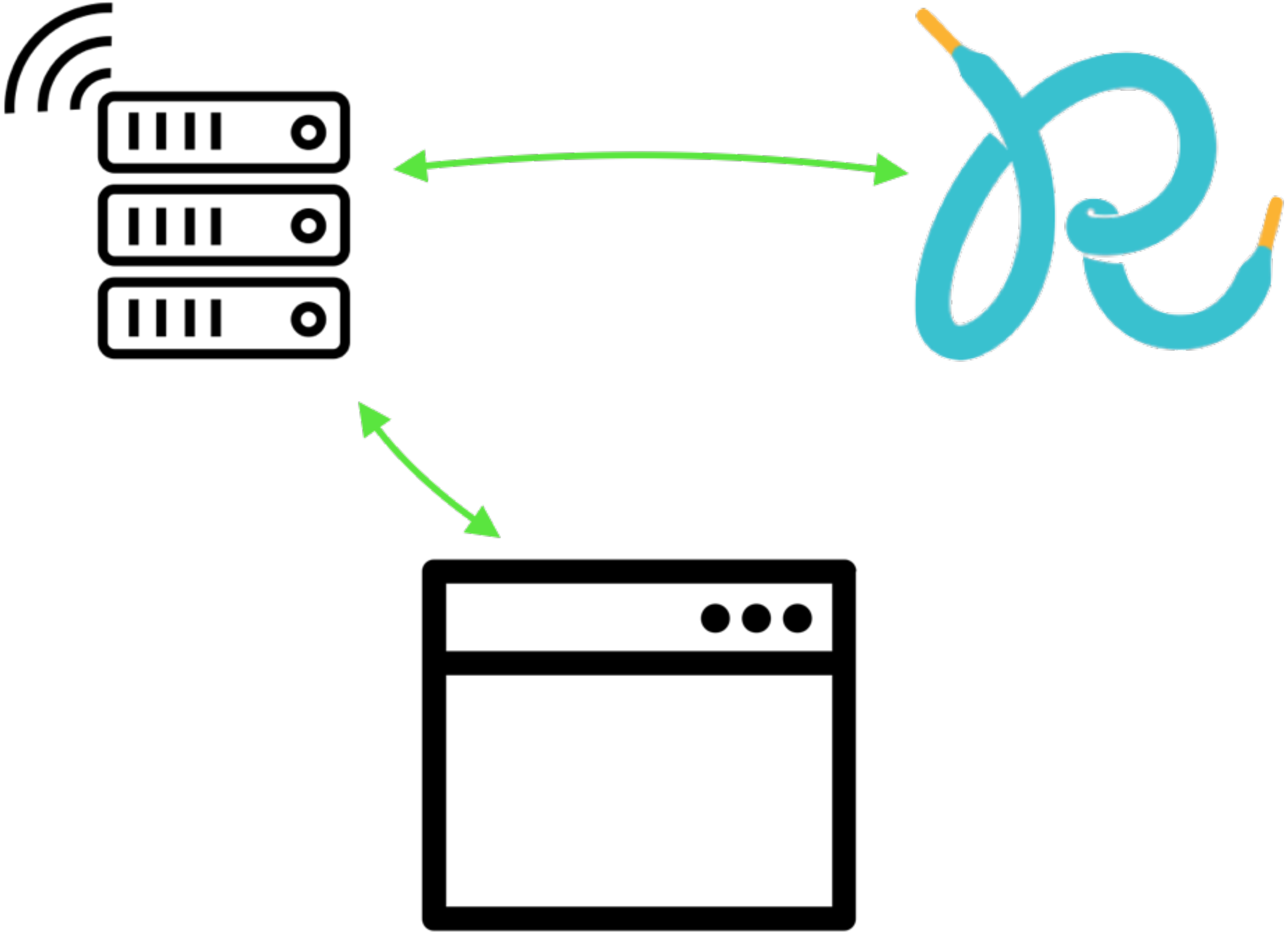
❌

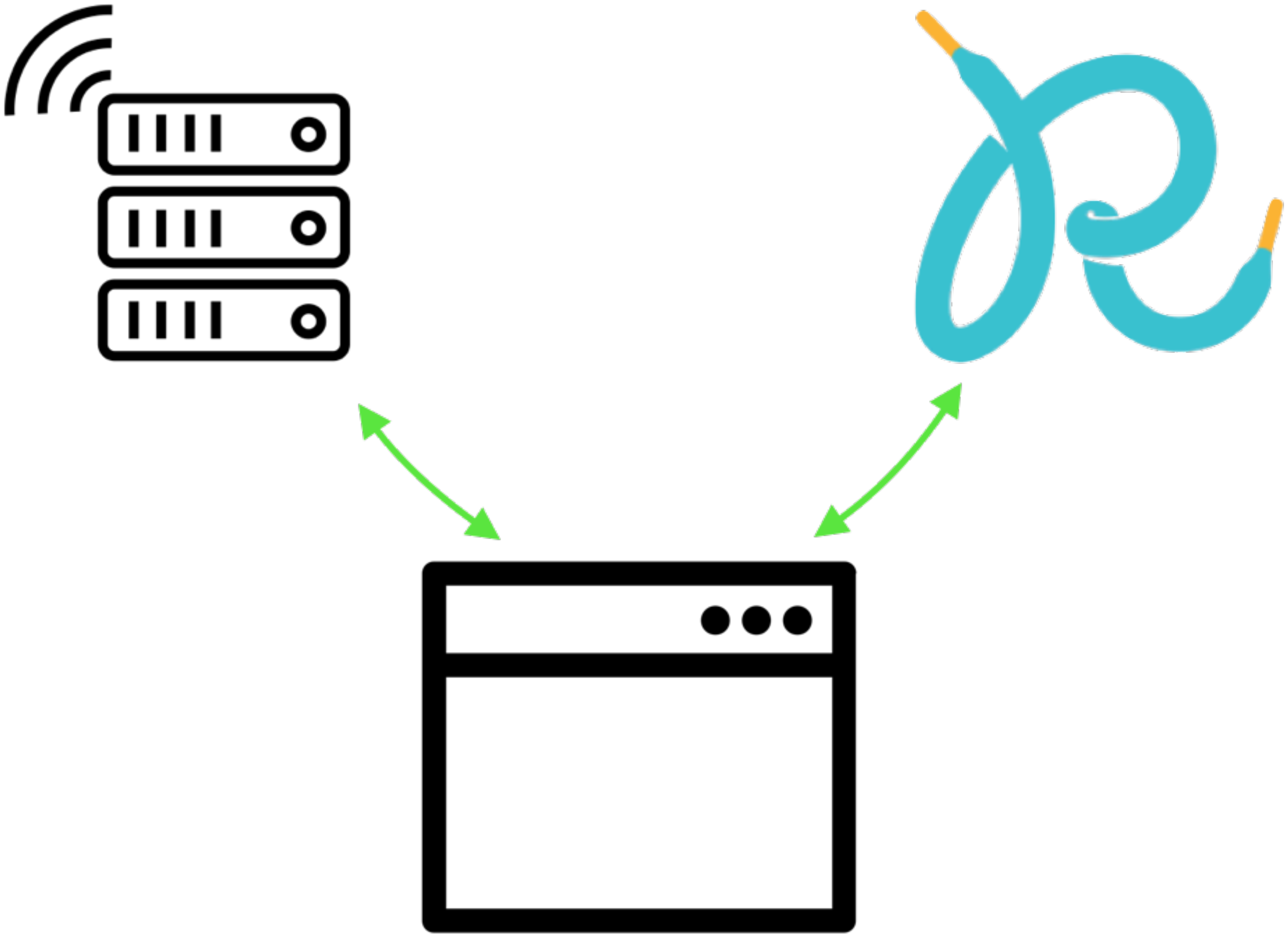
- Non-trivial backend
- Handling sensitive data
- Online only

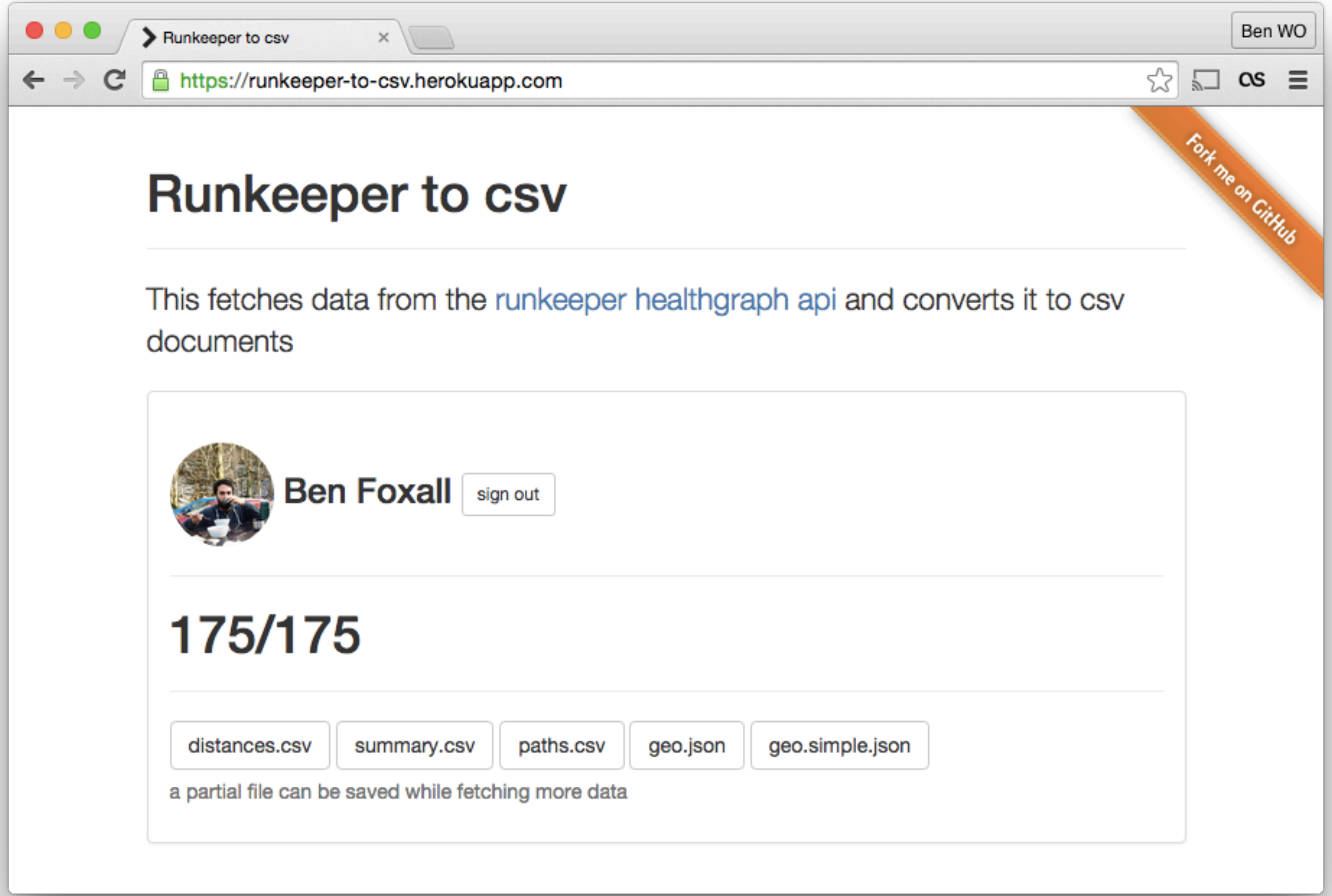
Attempt 4

Attempt 4

Serve from the browser







Runkeeper to csv

This fetches data from the runkeeper healthgraph api and converts it to csv documents

Ben Foxall

175/175

a partial file can be saved while fetching more data

👌

- Accessible
- Data stored locally

❌

≫ ...

What we’ll implement in the browser:

Request ▸ Process ▸ Serve

Request

A small Runkeeper API

```
const apiRequest = (path, token) =>
    fetch(`http://runkeeper.com/${path}`, {
        headers: {
            Authorisation: `Bearer: ${token}`
        }
    })
    .then(response => response.json())
```

A small Runkeeper API

```
function apiRequest (path, token) {
    return fetch('http://runkeeper.com/' + path, {
        headers: {
            Authorisation: 'Bearer: ' + token
        }
    })
    .then(function(response) {return response.json()})
}
```

dataForUser(…)

```
const dataForUser => token =>
  apiRequest('/activityIndex', token)
    .then(extractUrls)
    .then(makeRequests)
```

Process

json ➡ csv

A few ways of doing this…

- papaparse.com/docs#json-to-csv
- csv-formatter
- jwerle/to-csv

...another way

```
// pull out a row of keys
const row = (keys, obj) =>
  keys.map(k => obj[k])

// create a csv row from an array
const csv = array =>
  array.map(item =>
    typeof item === 'string'
        ? item.replace(/[\",]/g,'')
        : item
  ).join(',') + "\n"
```

```
const columns = ['distance', 'climb', '…']

const csvForUser = (token) =>
  dataForUser(token)
    .then(pages =>
      pages.map(page =>
        csv(row(columns , page))
      ).join('\n')
    )
```

Serve

Data URIs

```
data:[<mediatype>][;base64],<data>
```

```
data:text/html,<h1>Hello World
```

```
data:text/csv,a,b,c%0A1,2,3
```

Put it all together

```
csvForUser(token)
  .then(csv =>
    document.querySelector('#csv-link').href = toDataURI(csv, 'text/csv')
  )
```

…

… supporting bigger files

```
.then(csv =>
  … toDataURI(csv, 'text/csv')
)
```

```
Blob(blobParts[, options])
```

Blob

```
const csvForUser = (token) =>
  dataForUser(token)
    .then(pages =>
      pages.map(page =>
        page.items.map(to_csv).join('')
      ).join('\n')
    )
```

Blob

```
const csvForUser = (token) =>
  dataForUser(token)
    .then(pages =>
      pages.map(page =>
        new Blob(page.items.map(to_csv))
      )
```

Blob

```
csvForUser(token)
  .then(csv =>
    document.querySelector('#csv-link').href = toDataURI(csv, 'text/csv')
  )
```

Blob

```
csvForUser(token)
  .then(blobs =>
    const csv = new Blob(blobs,  {type: 'text/csv'})

    document.querySelector('#csv-link').href = URL.createObjectURL(csv)

    // "blob:https%3A//example.com/8ffbb9df-...-4fffe9160979"
  )
```

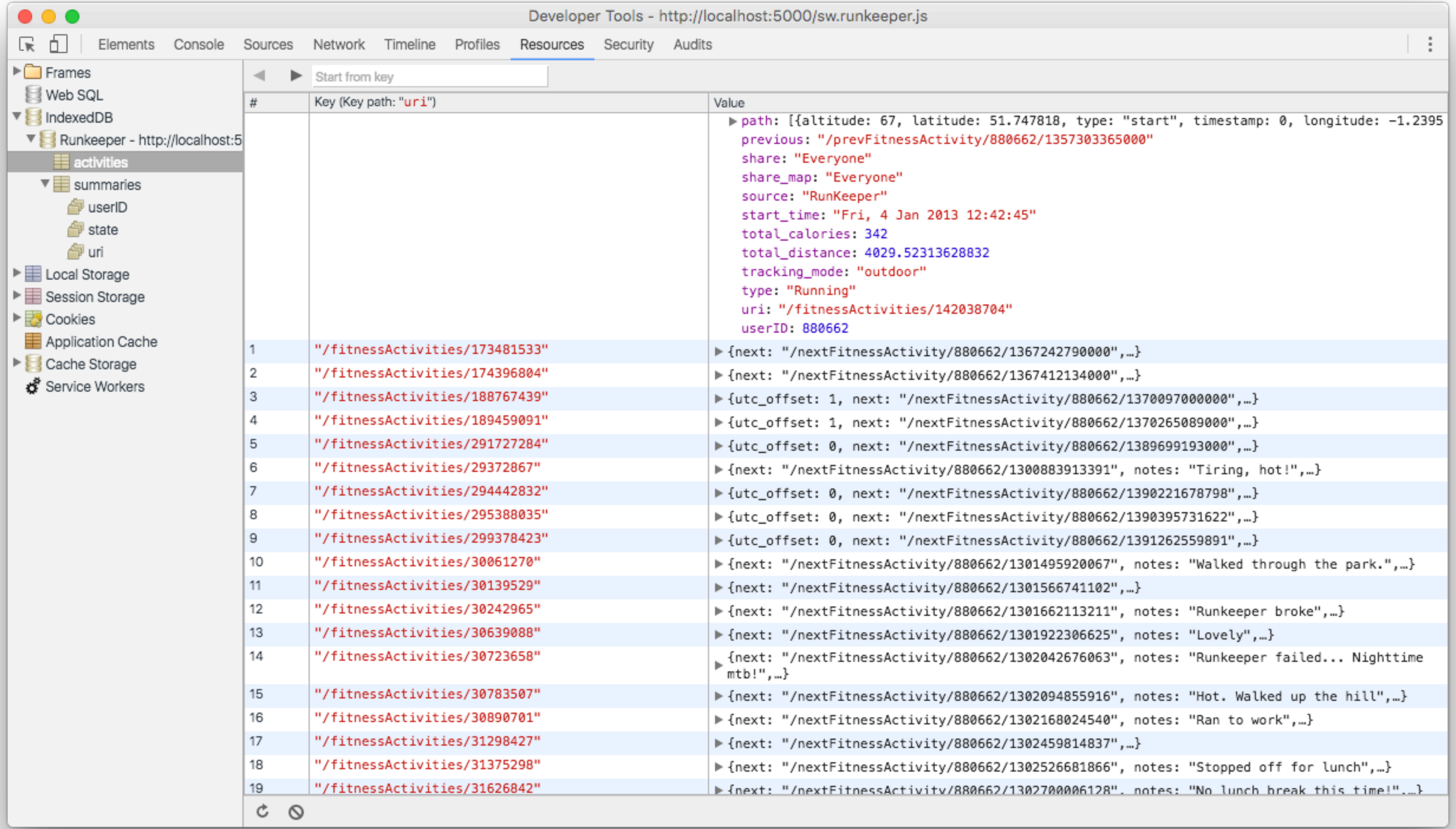
…

…no persistance

IndexedDB

IndexedDB (*+Dexie*)

```
const db = new Dexie('Runkeeper')
db.version(1).stores({
  activities: '&uri'
})
db.open()


fetch(uri)
  .then(res => res.json() )
  .then(data => db.activities.put(data))
```



#	Key (Key path: "uri")	Value
		▶ path: [{altitude: 67, latitude: 51.747818, type: "start", timestamp: 0, longitude: -1.2395 previous: "/prevFitnessActivity/880662/1357303365000" share: "Everyone" share_map: "Everyone" source: "RunKeeper" start_time: "Fri, 4 Jan 2013 12:42:45" total_calories: 342 total_distance: 4029.52313628832 tracking_mode: "outdoor" type: "Running" uri: "/fitnessActivities/142038704" userID: 880662
1	"/fitnessActivities/173481533"	▶ {next: "/nextFitnessActivity/880662/1367242790000",…}
2	"/fitnessActivities/174396804"	▶ {next: "/nextFitnessActivity/880662/1367412134000",…}
3	"/fitnessActivities/188767439"	▶ {utc_offset: 1, next: "/nextFitnessActivity/880662/1370097000000",…}
4	"/fitnessActivities/189459091"	▶ {utc_offset: 1, next: "/nextFitnessActivity/880662/1370265089000",…}
5	"/fitnessActivities/291727284"	▶ {utc_offset: 0, next: "/nextFitnessActivity/880662/1389699193000",…}
6	"/fitnessActivities/29372867"	▶ {next: "/nextFitnessActivity/880662/1300883913391", notes: "Tiring, hot!",…}
7	"/fitnessActivities/294442832"	▶ {utc_offset: 0, next: "/nextFitnessActivity/880662/1390221678798",…}
8	"/fitnessActivities/295388035"	▶ {utc_offset: 0, next: "/nextFitnessActivity/880662/1390395731622",…}
9	"/fitnessActivities/299378423"	▶ {utc_offset: 0, next: "/nextFitnessActivity/880662/1391262559891",…}
10	"/fitnessActivities/30061270"	▶ {next: "/nextFitnessActivity/880662/1301495920067", notes: "Walked through the park.",…}
11	"/fitnessActivities/30139529"	▶ {next: "/nextFitnessActivity/880662/1301566741102",…}
12	"/fitnessActivities/30242965"	▶ {next: "/nextFitnessActivity/880662/1301662113211", notes: "Runkeeper broke",…}
13	"/fitnessActivities/30639088"	▶ {next: "/nextFitnessActivity/880662/1301922306625", notes: "Lovely",…}
14	"/fitnessActivities/30723658"	▶ {next: "/nextFitnessActivity/880662/1302042676063", notes: "Runkeeper failed... Nighttime mtb!",…}
15	"/fitnessActivities/30783507"	▶ {next: "/nextFitnessActivity/880662/1302094855916", notes: "Hot. Walked up the hill",…}
16	"/fitnessActivities/30890701"	▶ {next: "/nextFitnessActivity/880662/1302168024540", notes: "Ran to work",…}
17	"/fitnessActivities/31298427"	▶ {next: "/nextFitnessActivity/880662/1302459814837",…}
18	"/fitnessActivities/31375298"	▶ {next: "/nextFitnessActivity/880662/1302526681866", notes: "Stopped off for lunch",…}
19	"/fitnessActivities/31626842"	▶ {next: "/nextFitnessActivity/880662/1302700006128", notes: "No lunch break this time!",…}

IndexedDB *(+Dexie)*

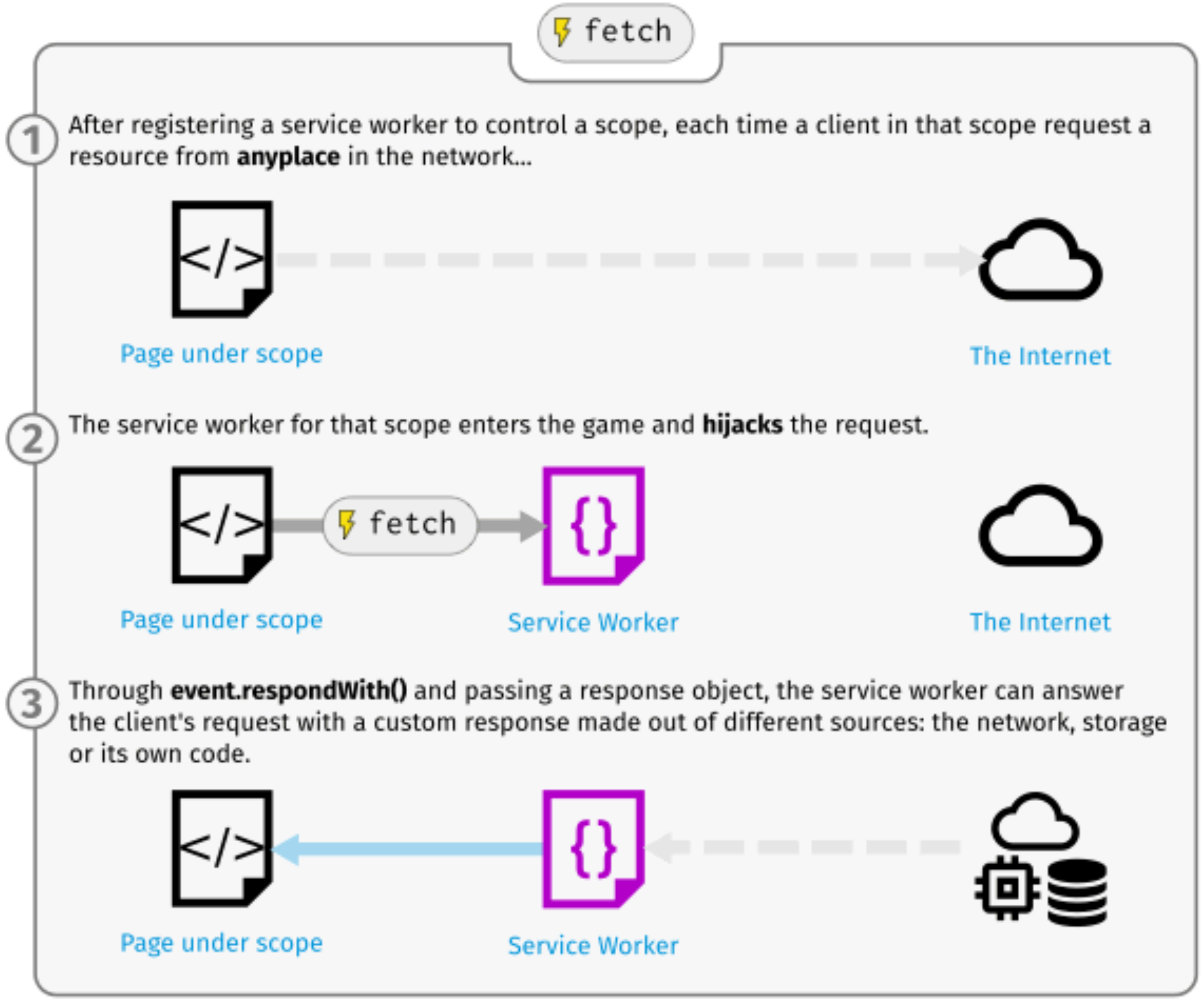
```
db.activities
  .each(activityToCSV)
```

…

…no urls

*The `URL.createObjectURL()` static method creates a DOMString containing an URL representing the object given in parameter. The URL **lifetime is tied to the document in the window** on which it was created. The new object URL represents the specified File object or Blob object.*

Service Workers



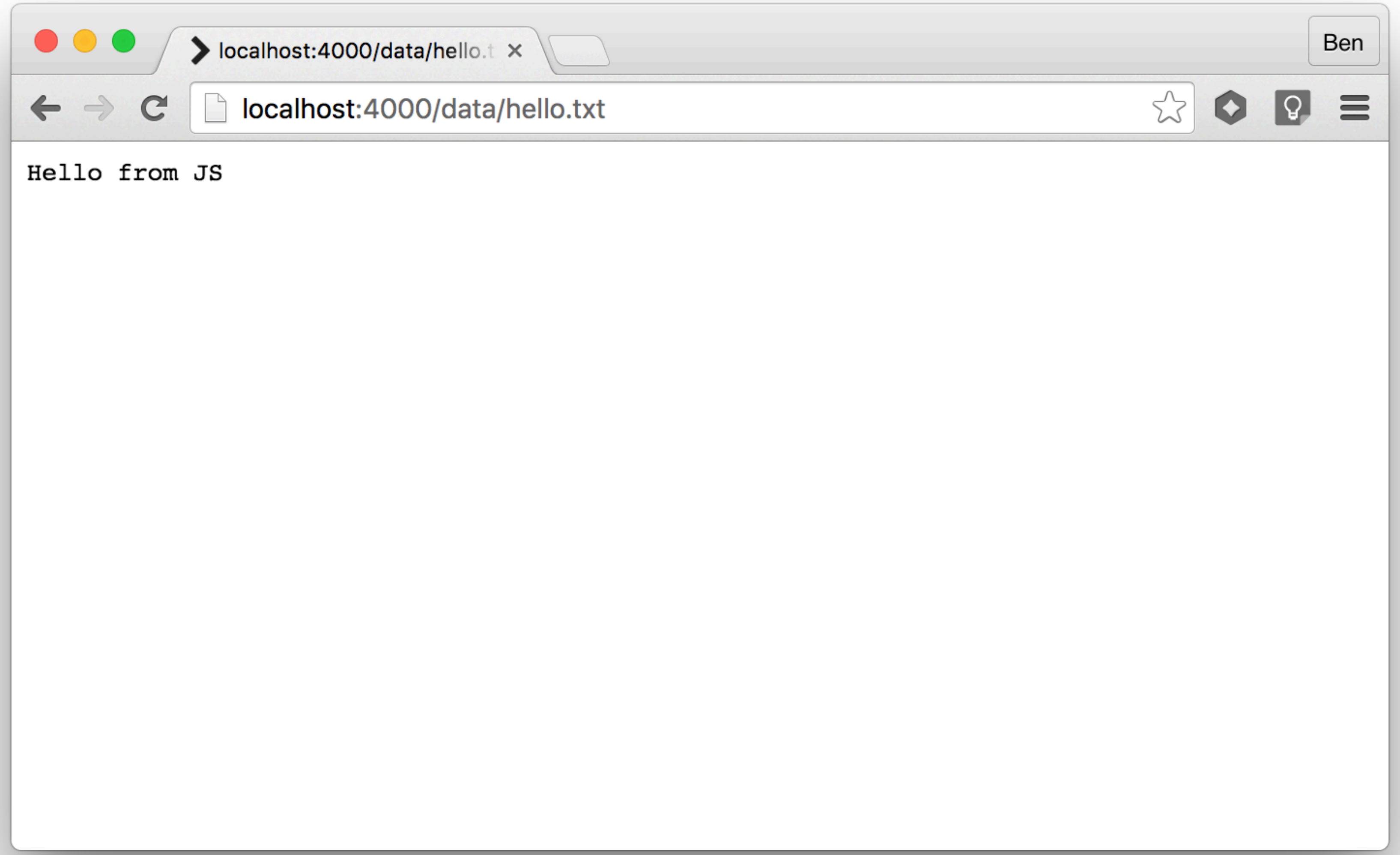
fetch
1
After registering a service worker to control a scope, each time a client in that scope request a resource from **anyplace** in the network...
Page under scope
The Internet
2
The service worker for that scope enters the game and **hijacks** the request.
Page under scope
fetch
Service Worker
The Internet
3
Through **event.respondWith()** and passing a response object, the service worker can answer the client's request with a custom response made out of different sources: the network, storage or its own code.
Page under scope
Service Worker

(source: MDN)

Service Workers

```
self.addEventListener('fetch', event =>
  if(event.request.url.match(/data\/hello\.txt$/))
    event.respondWith(
      new Response('Hello from JS',
        {ContentType: 'text/plain'}
      )
    )
```

Service Workers



```
Hello from JS
```

Put it all together

```
new Response(blob, {ContentType: 'text/csv'})
```

👌

```
event.respondWith(
  cache(responder)
)
```

👌

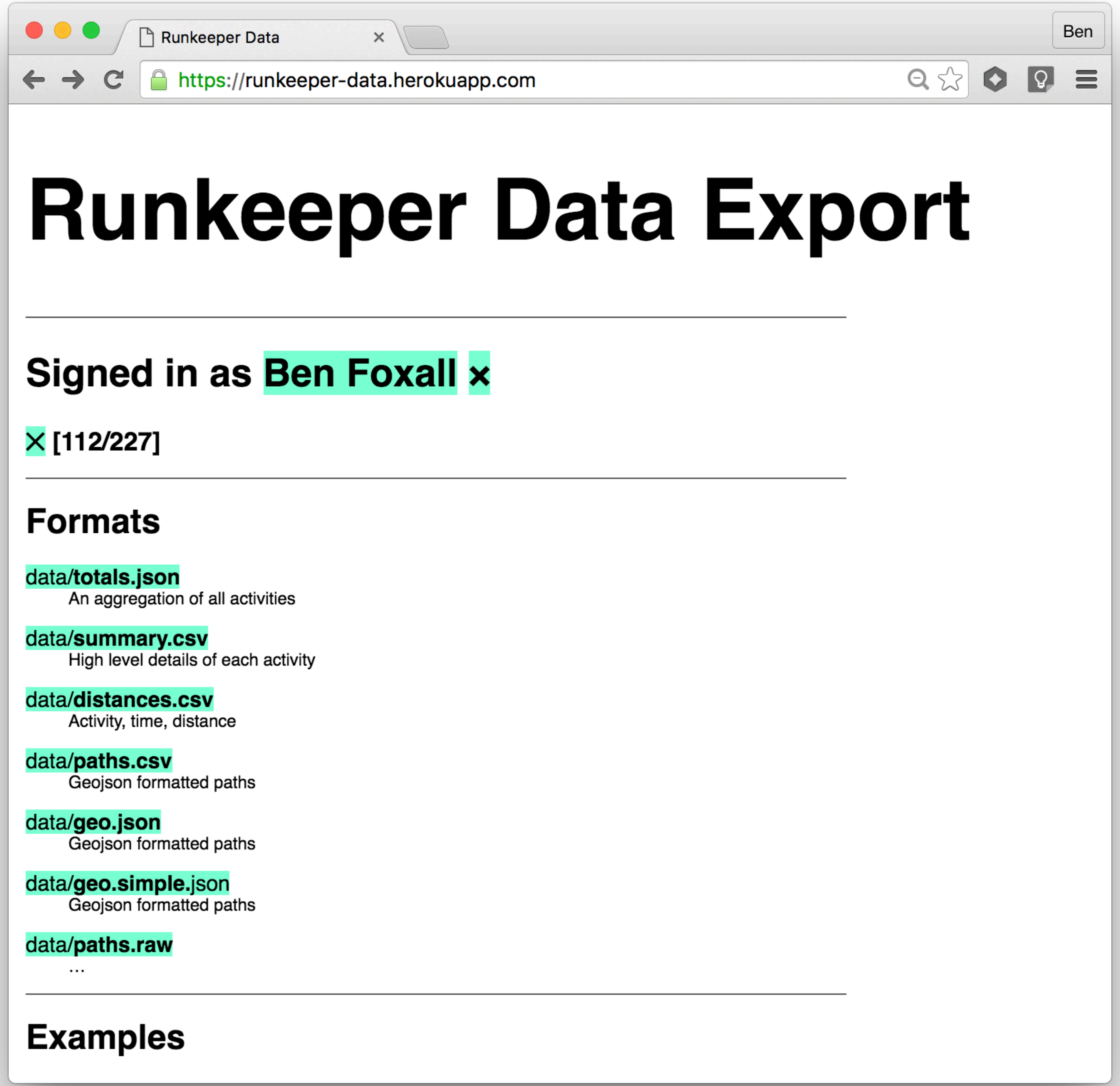
```
self.addEventListener('fetch', event =>  {

    if(event.request.url.match(/data\/summary\.csv$/))
      respond(event, summaryResponse)

    if(event.request.url.match(/data\/distances\.csv$/))
      respond(event, distancesResponse)

    if(event.request.url.match(/data\/paths\.csv$/))
      respond(event, pathsResponse)

    if(event.request.url.match(/data\/geo\.json$/))
      respond(event, geoJSONResponse)

    //…

})
```

👌

```
fetch('/data/foo.json')
d3.csv('/data/foo.csv')
$.get('/data/foo.html')
```

👌

Offline



Runkeeper Data Export

Signed in as Ben Foxall ×

✕ [112/227]

Formats

data/**totals.json**
An aggregation of all activities

data/**summary.csv**
High level details of each activity

data/**distances.csv**
Activity, time, distance

data/**paths.csv**
Geojson formatted paths

data/**geo.json**
Geojson formatted paths

data/**geo.simple**.json
Geojson formatted paths

data/**paths.raw**
...

Examples

Things I really like about this:

- Accessibility - data for everyone
- It's the start of something, not the end

🔗

github/benfoxall/runkeeper-data
runkeeper-data.herokuapp.com

Dankeschön

@benjaminbenben

Developer Relations at @pusher

slides: github/benfoxall/csv-browser-talk
Network + Browser icons: YuguDesign & Fritz – Noun Project