

# CouchApps

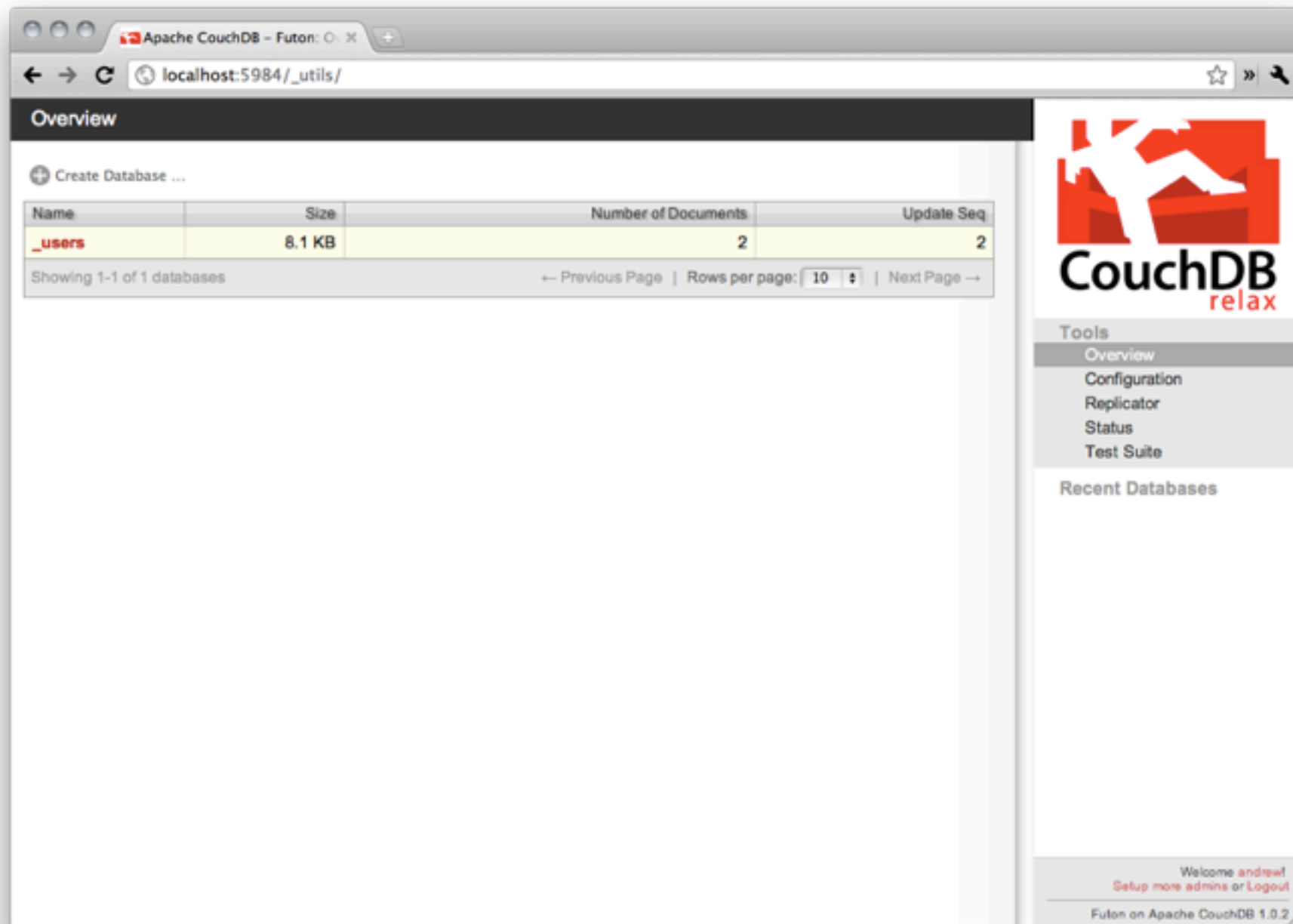
---

a2div / 02.24.2011

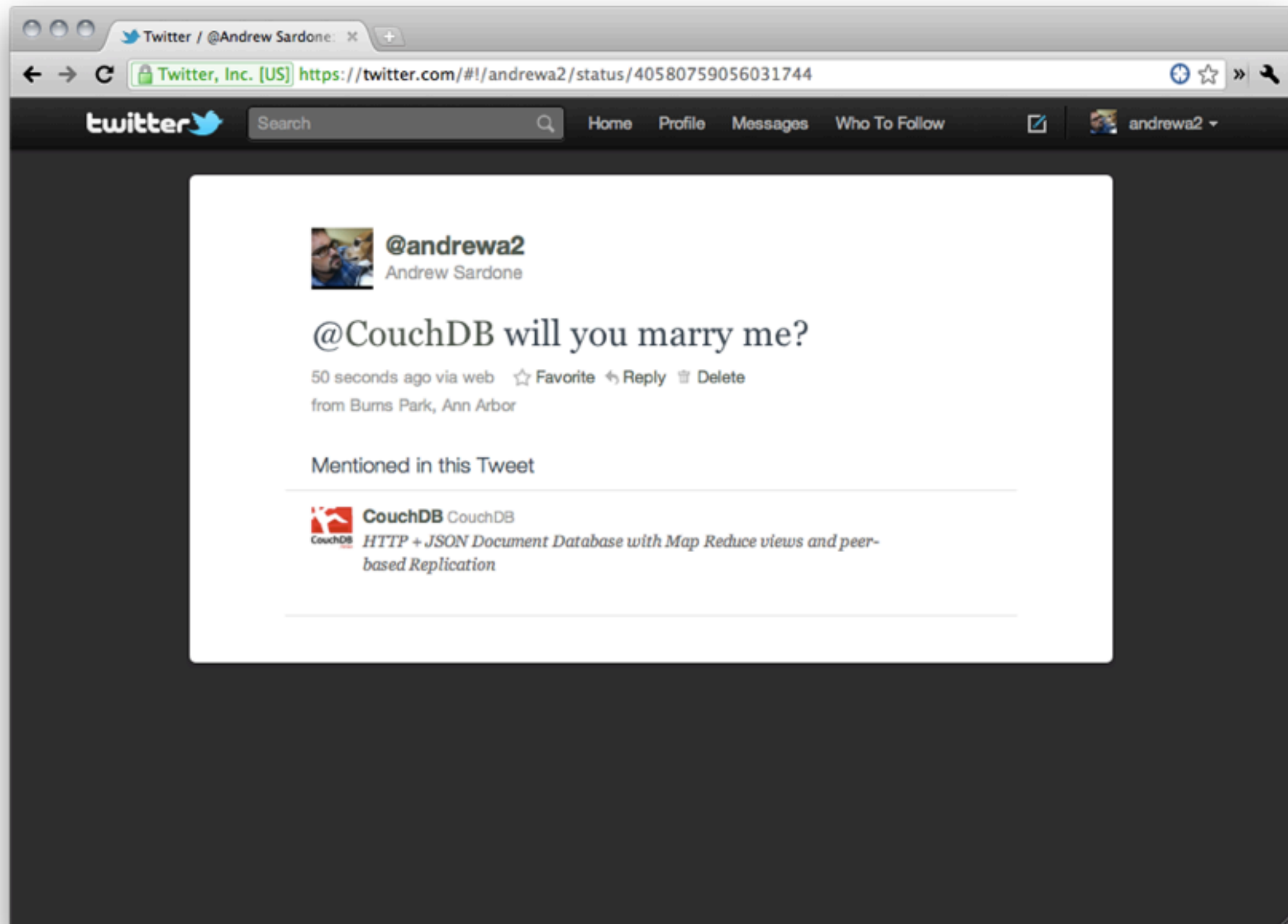
Andrew Sardone / @andrewa2



**[www.nutshell.com](http://www.nutshell.com)**  
**@nutshellcrm**



# CouchDB



- Document-Oriented Database
- RESTful JSON API
- Map-Reduce
- Distributed
- Fault-tolerant
- Open Source



i.e., webscale  
(but seriously, it's cool)

# Documents

```
{  
  "_id": "andrew",  
  "_rev": "3-013d308d28c9f1996148de9eff5cf054",  
  "type": "human",  
  "occupation": "Aspiring Polymath",  
  "number_of_eyes": 4,  
  "favorite_drinks": ["coffee", "iced tea"]  
}
```

A CouchDB is a flat collection of JSON documents with a unique ID

# RESTful HTTP Interface

```
// create database 'people'
```

```
> curl -X PUT http://couch/people
```

```
{"ok":true}
```

```
// create document in database 'people'
```

```
> curl -X PUT http://couch/people/andrew \  
-d '{"type": "human", "number_of_eyes": 4, ... }'
```

```
{"ok":true,"id":"andrew", "rev":"1-d6960bf0c..."}
```

```
// fetch document 'andrew' in database 'people'
```

```
> curl -X GET http://couch/people/andrew
```

```
{"_id": "andrew", "rev":"1-d6960bf0c...", "type": ... }
```



# Map-Reduce

Fault-tolerance

by @jrecursive



# Map-Reduce

Build queries / indexes across your documents

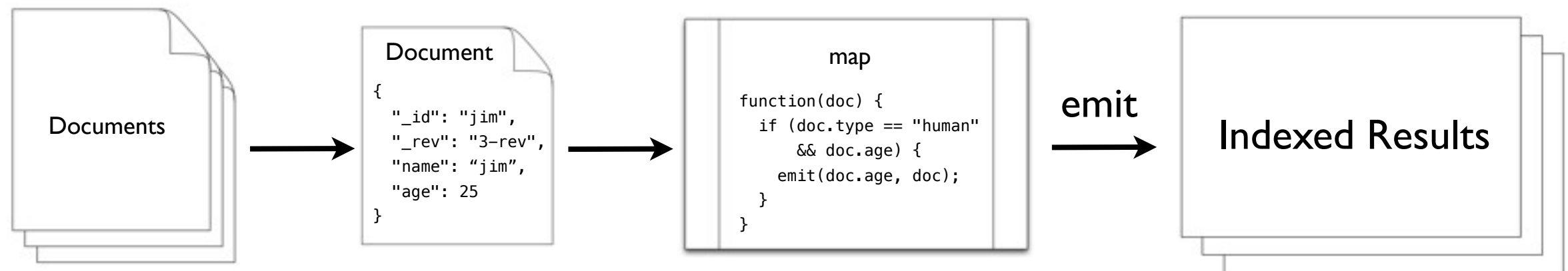
Map

```
function(doc) {  
  if (doc.type == "human" && doc.age) {  
    emit(doc.age, doc);  
  }  
}
```

Reduce

```
function(keys, values, rereduce) {  
  return sum(values);  
}
```

# View

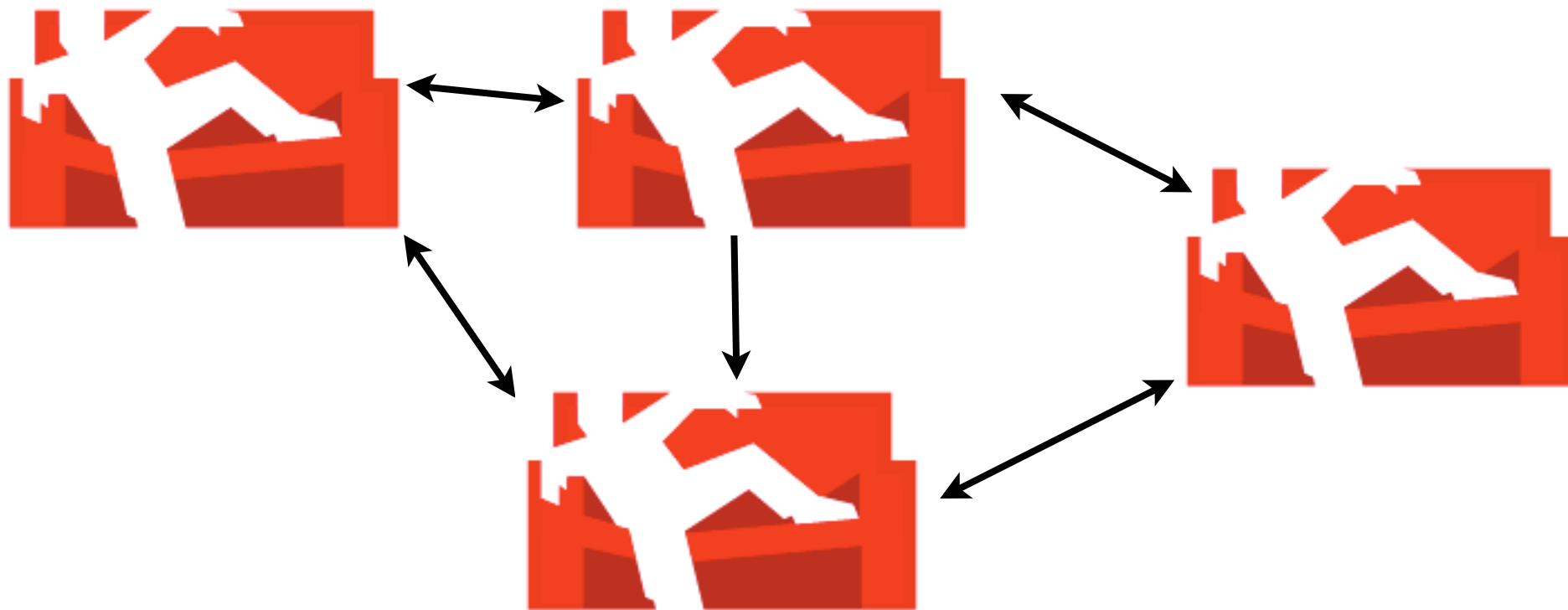


```
> curl -X GET http://couch/db/_design/doc/_view/human-ages
{
  "total_rows":2,
  "offset":0,
  "rows": [
    { "_id": "frank", "key": 25, "value": { value } },
    ...
  ]
}
```

# Replication

## Master-Master Distributed

```
> curl -X POST http://couch/_replicate -d \  
'{"source":"http://couch1/mydb", "target":"http://couch2/mydb"}'
```



# /\_changes

- Realtime notifications about changes to the database
- CouchDB keeps a record of the order in which operations were applied to a given database. This way, you can always ask it "what's happened since the last time I asked?"
- replicator can listen to \_changes, keep 2 databases in sync

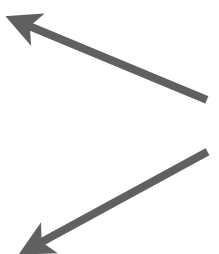


CouchApps

# What is a CouchApp?

Applications served directly from  
CouchDB over HTTP

```
{
  "_id": "special_assets",
  "_rev": "22-3f6fd2c7e03ac5cfe6384cb9d3cfbfff9",
  "_attachments": {
    "kick-ass.png": {
      "content_type": "image/png",
      "revpos": 14,
      "length": 16526,
      "stub": true
    },
    "index.html": {
      "content_type": "text/html",
      "revpos": 11,
      "length": 12324,,
      "stub": true
    }
  }
}
```



Documents can store  
arbitrary binary attachments

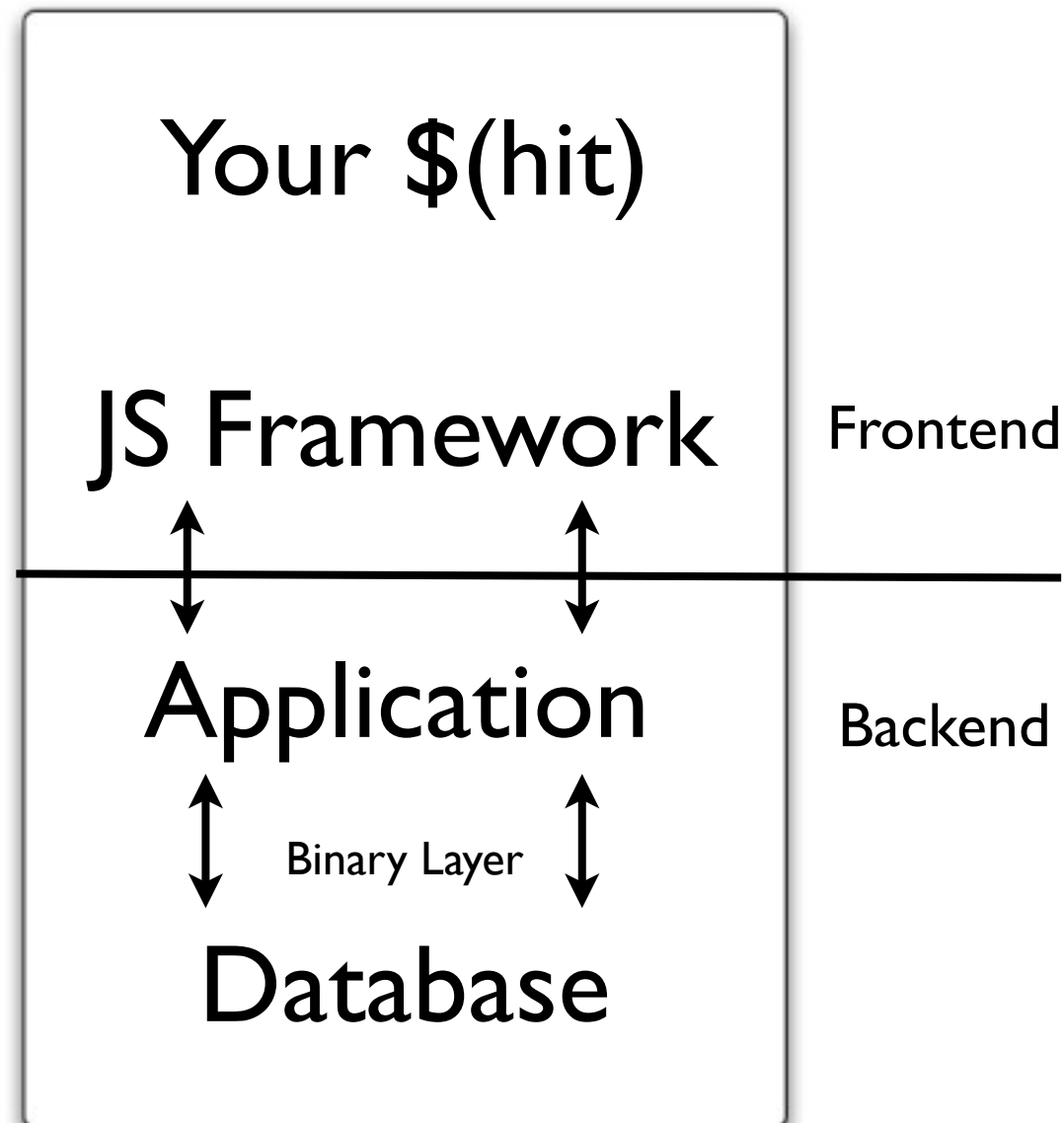
HTML, JavaScript, CSS, PNGs, etc.



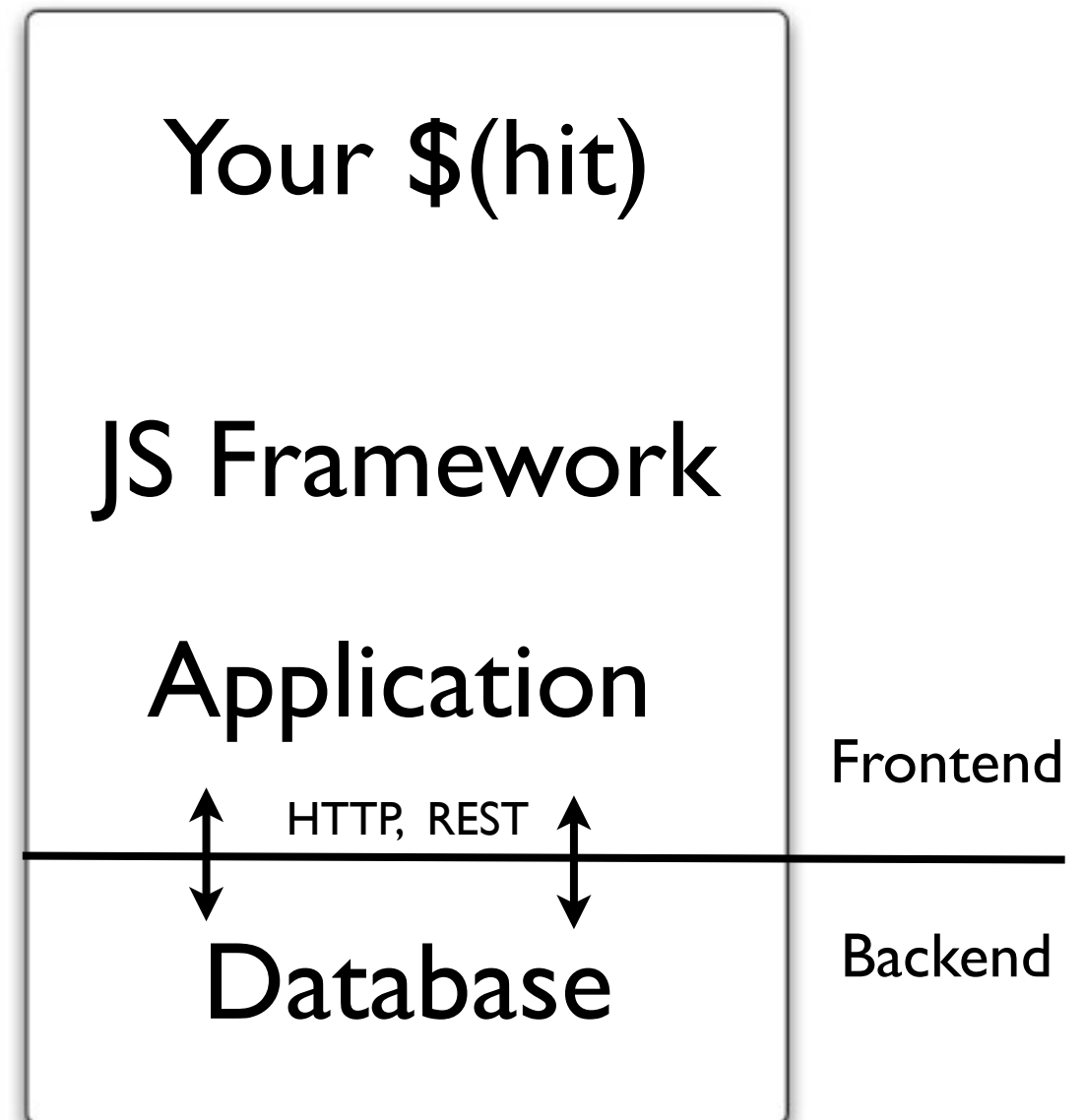
# Simplified Stack

*Entire API is HTTP (no binary layer)*

Before



After



Ripped off diagram from Aaron Quint / @aq

# Design Document

## *A CouchApp's Glue*

Documents whose `_id` contains a `_design/` prefix

# Design Document

- Attachments served directly from Couch
  - HTML, JavaScript, CSS, Images, etc.
  - Avoid same origin policy
- Validation functions
- Map-Reduce Views
- List / Show functions
- Templates
- URL Rewrites

```
{
  "_id": "_design/blog",
  "_rev": "1-97642f500f52df4026cc7ce9010c99ca",
  "_attachments": {
    "script/app.js": {
      "content_type": "application/javascript",
      "revpos": 1,
      "length": 183,
      "stub": true
    },
  },
  "views": {
    "recent-posts": {
      "map": "function(doc) { ... }"
    },
  },
  "validate_doc_update": "function (newDoc, oldDoc, userCtx, secObj) {...}",
  "lists": {
    "index": "function(head, req) { ... }"
  },
  "templates": {
    "index": "<!DOCTYPE html>\n<html>\n  <head>\n..."
  }
}
```

```
> curl -X GET http://couch/db/_design/blog/_view/recent-posts
{
  "total_rows":2,
  "offset":0,
  "rows": [
    { "_id": "3a8f29cc3",
      "key": "2011-02-03T20:06:44Z",
      "value": { blog post }
    },
    ...
  ]
}
```

Contents of a design  
document exposed as  
resources

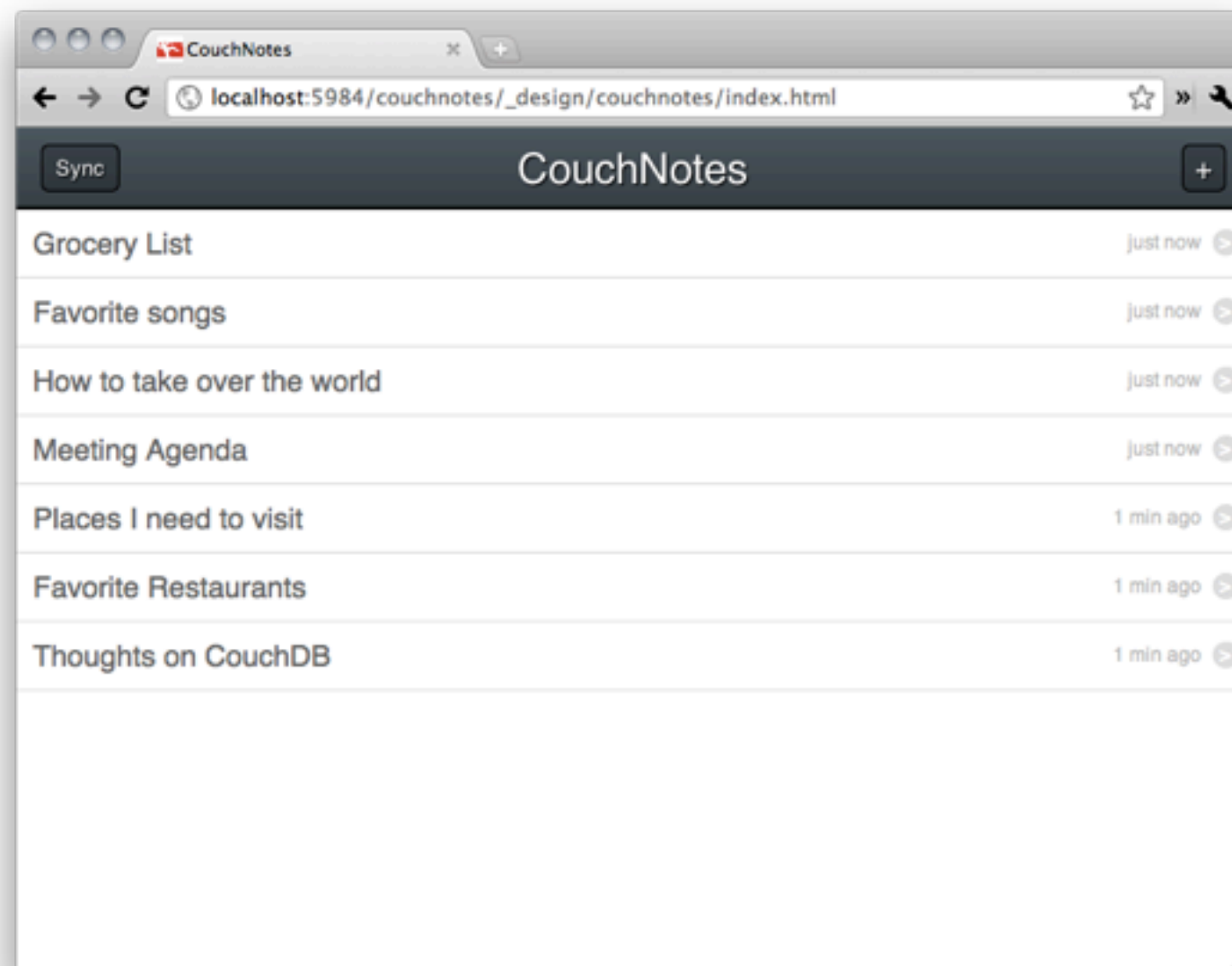
```
> curl -X GET http://couch/db/_design/blog/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>My Blog</title>
    ...
  
```



Grab index.html

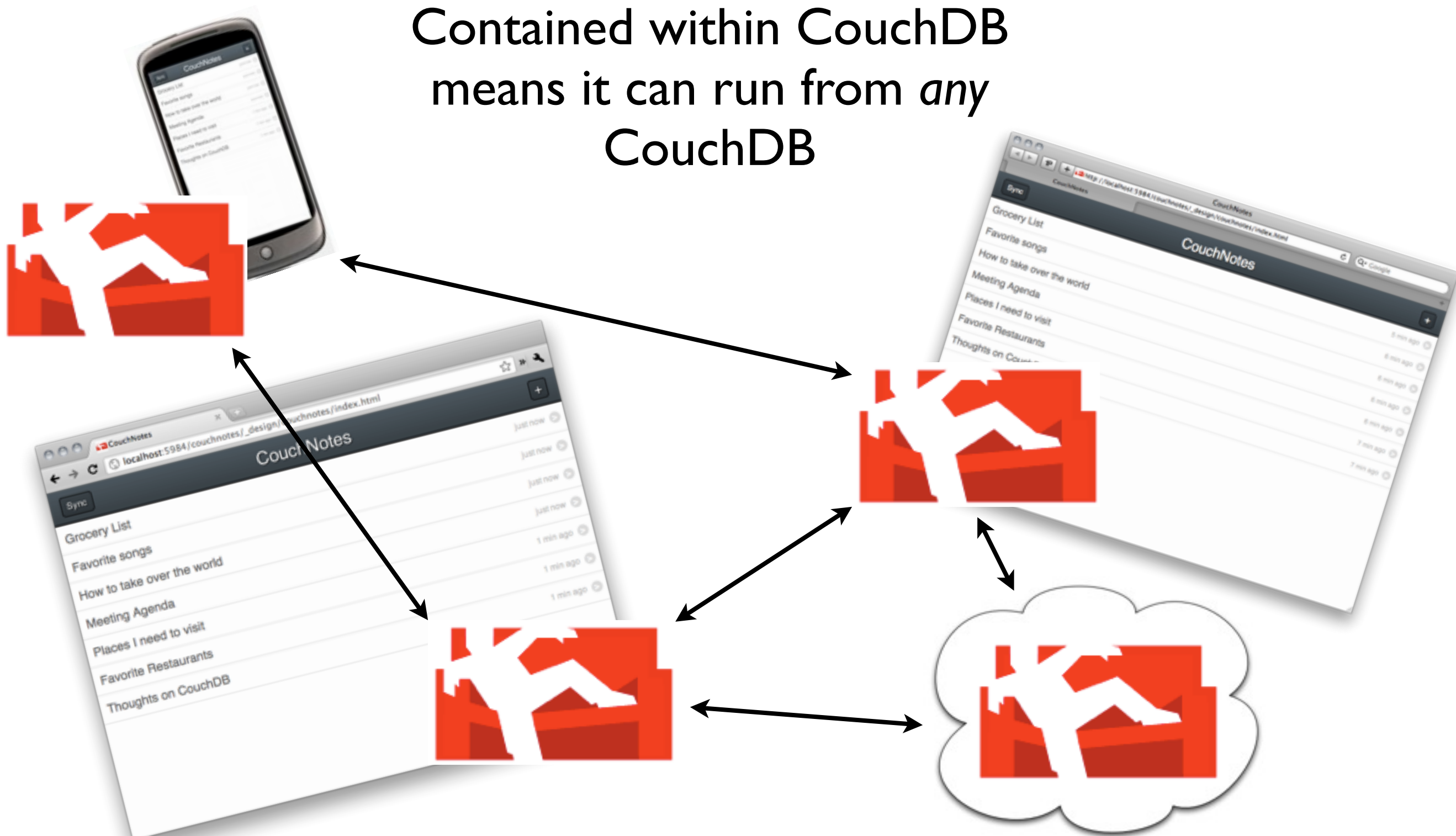
Query a view

Profit



# Replication

Contained within CouchDB  
means it can run from *any*  
CouchDB



# CouchApp stdlib

- jquery.couch.js & jquery.couch.app.js
- Handy jQuery wrappers around standard Couch APIs
- <http://couchapp.org/page/what-is-couchapp>



# DEMO

(a trivial app)

# Show Functions

- Transform a single document GET from JSON into the format of your choice
- Rendered server-side HTML, CSV, PDF etc.

# List Functions

- Analog of show functions, but for view results
- Serve plain-old HTML based on rows of JSON

[couchone.com/get](http://couchone.com/get)

# Reference Resources

- The Antepenultimate CouchDB Reference Card [bit.ly/couchref](http://bit.ly/couchref)
- CouchDB HTTP API Reference [bit.ly/couchhttpapi](http://bit.ly/couchhttpapi)

## CouchApp Tools

- <http://couchapp.org>
- <https://github.com/quirkey/soca>

## Front-End Frameworks

- <http://sammyjs.org>
- <http://janmonschke.github.com/backbone-couchdb>