

# OfflineFirst Apps With PouchDB and CouchDB

Lorna Mitchell, IBM

# OfflineFirst Apps

Offline is not an error condition

*"Offline capability is a key characteristic of modern Progressive Web Applications"*

- <http://offlinefirst.org>

# Being Offline

Could include:

- being in a place without internet infrastructure
- being without a data package on your phone
- being on the tube or a plane

# Being Offline

Could include:

- being in a place without internet infrastructure
- being without a data package on your phone
- being on the tube or a plane
- getting the train from Huddersfield to ... really anywhere

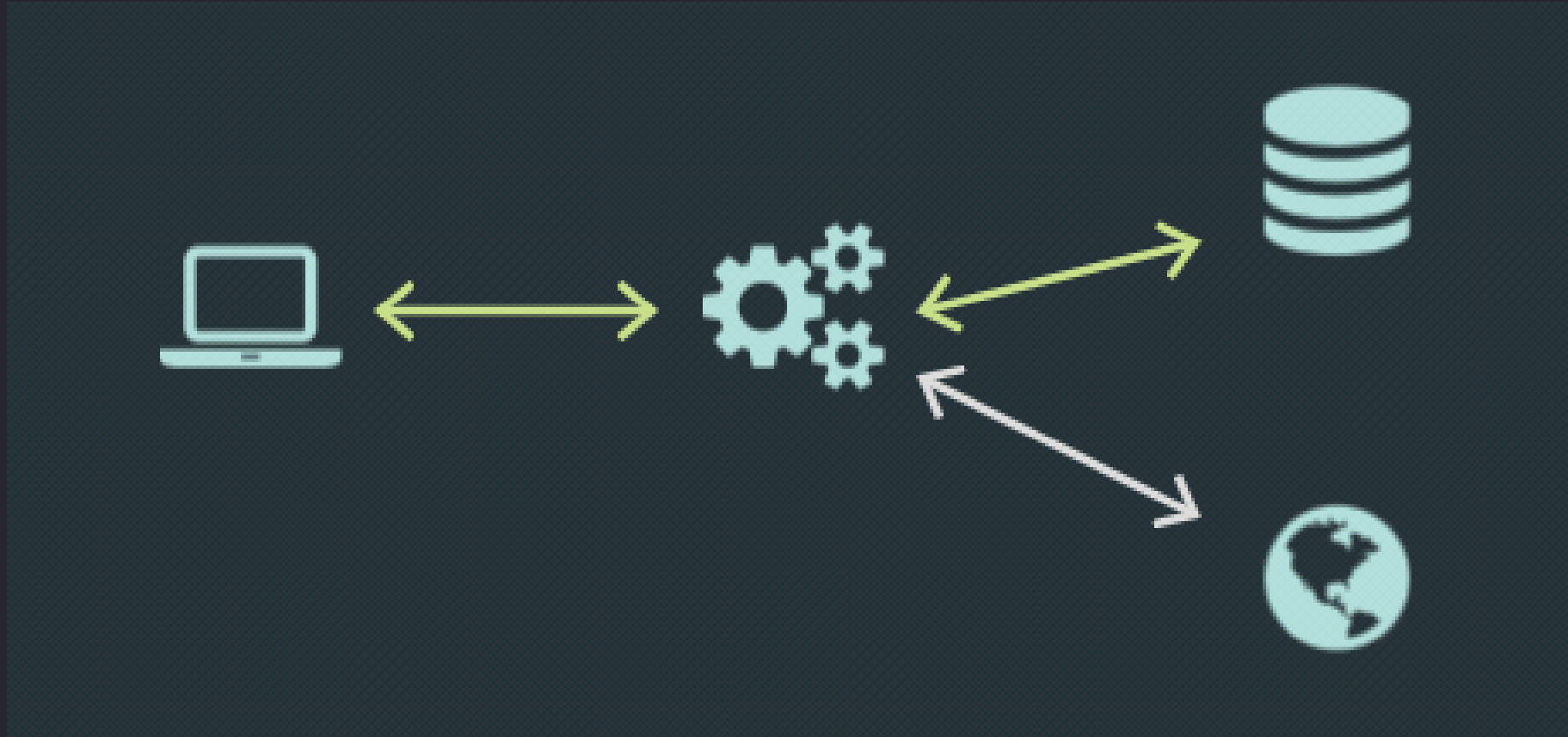
# OfflineFirst

OfflineFirst means failing gracefully

You'll also hear PWA which is a Progressive Web App, covering more than just network failures

# Achieving OfflineFirst: Code

Service Worker caches on first page load



# Achieving OfflineFirst: Data

Client-side app and storage, background sync



# PouchDB and CouchDB

- <https://pouchdb.com/>
  - A database that your client-side javascript can use
  - Can also sync to CouchDB (-compatible) databases
- <https://couchdb.apache.org>
  - A NoSQL document database
  - Best replication on the planet (probably)
  - HTTP API = good support in all languages



# Example App: Shopping List

- Client-side JavaScript with PouchDB
- Works locally
- If connected, syncs to Cloudant/CouchDB
- Code here:  
<https://github.com/lornajane/robust-shopping-list>

# PouchDB in Action

In index.html:

```
<script src="/js/pouchdb-6.1.2.min.js"></script>  
<script src="/js/shopping.js"></script>
```

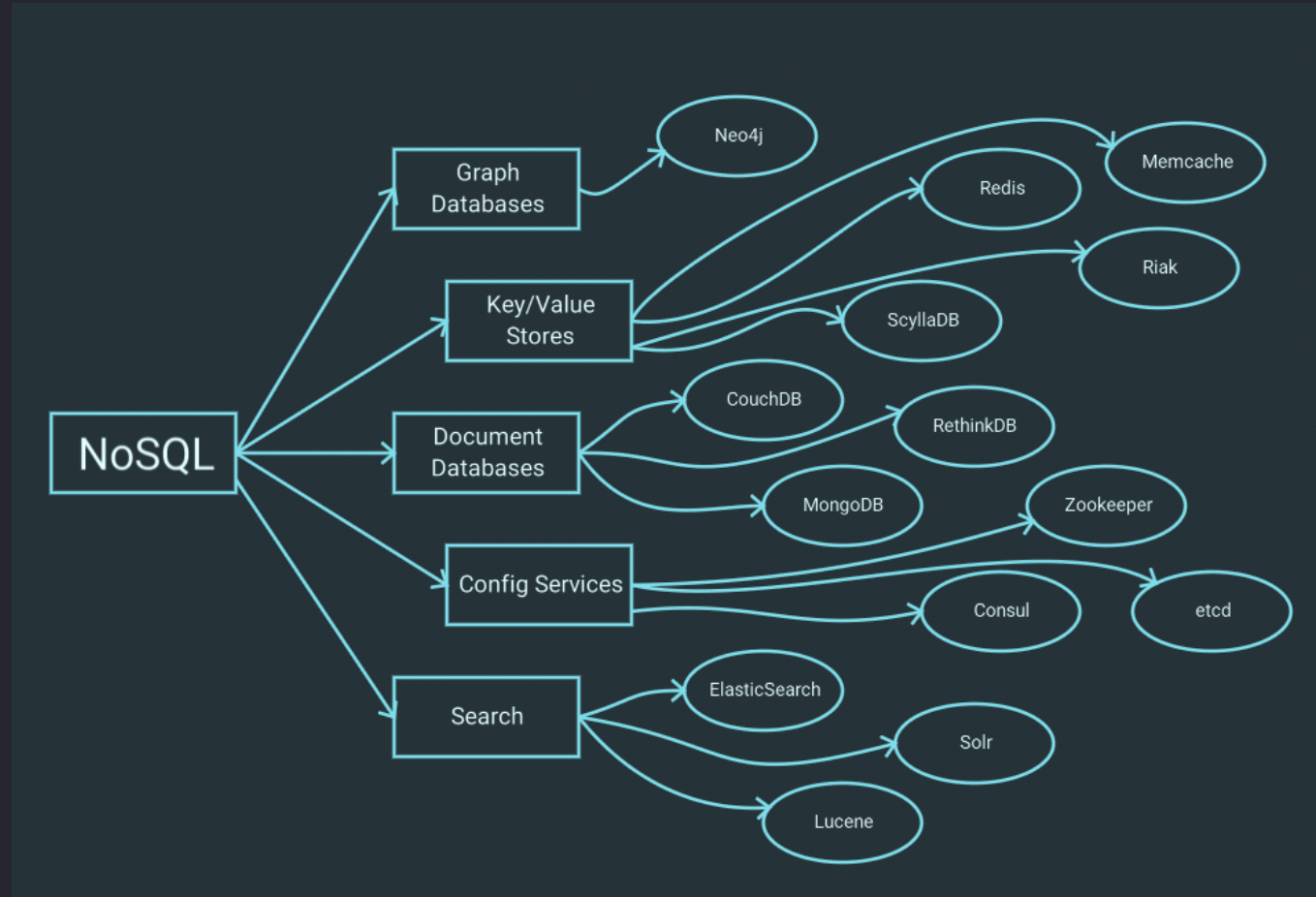
shopping.js is where my client-side JavaScript lives

# PouchDB in Action

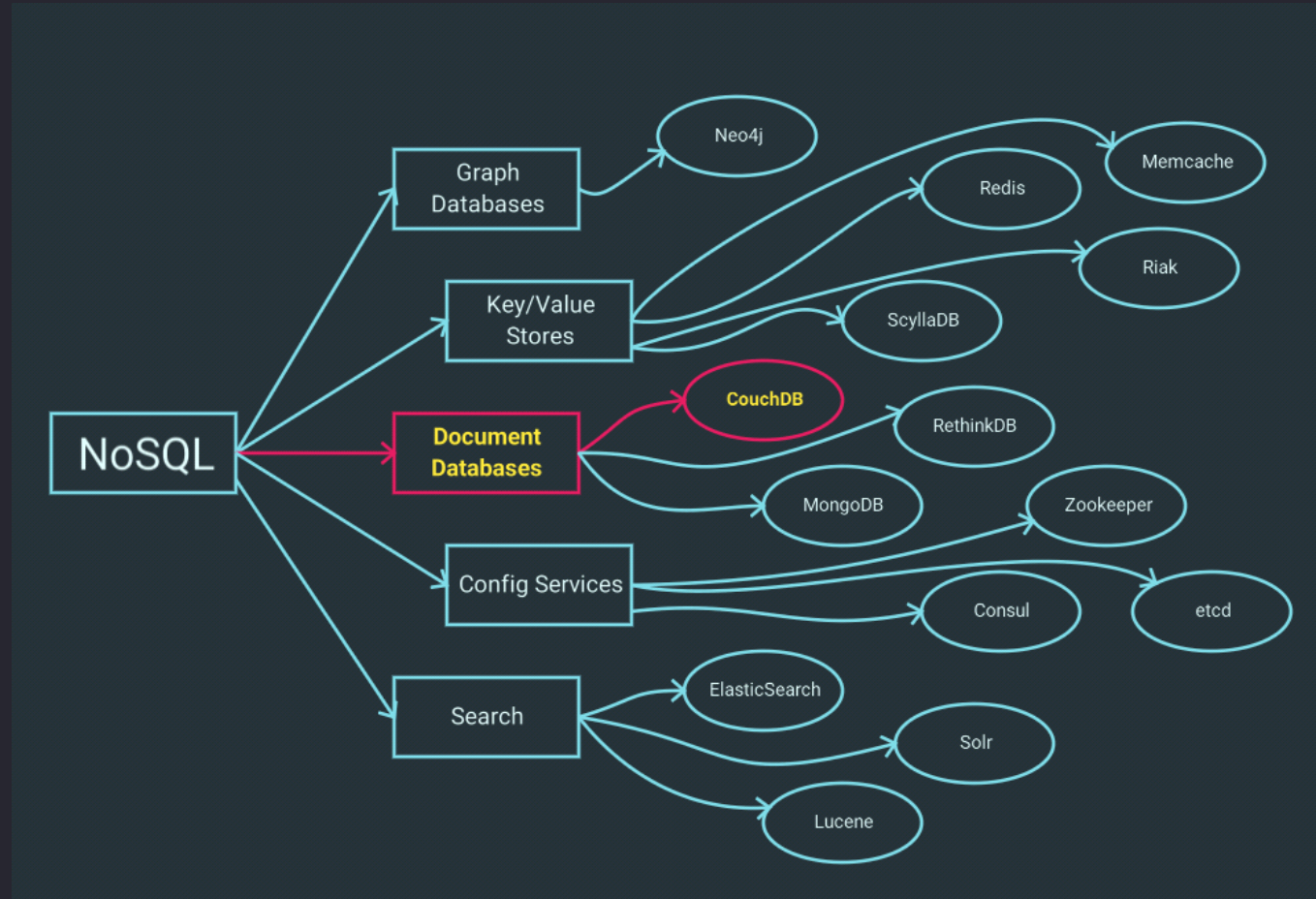
```
1 var db = new PouchDB('shopping');
2 var remoteDB = new PouchDB('http://localhost:5984/shopping');
3 window.onload = function() {
4     db.sync(remoteDB, { live: true, retry: true }
5     ).on('change', function (change) {
6         return getItemList().then(function (contents) {
7             document.getElementById('itemList').innerHTML = con
8         })
9     }).on('active', function (info) {
10        return getItemList().then(function (contents) {
11            document.getElementById('itemList').innerHTML = con
12        });
13    });
```

# NoSQL Document Database

# NoSQL Document Database

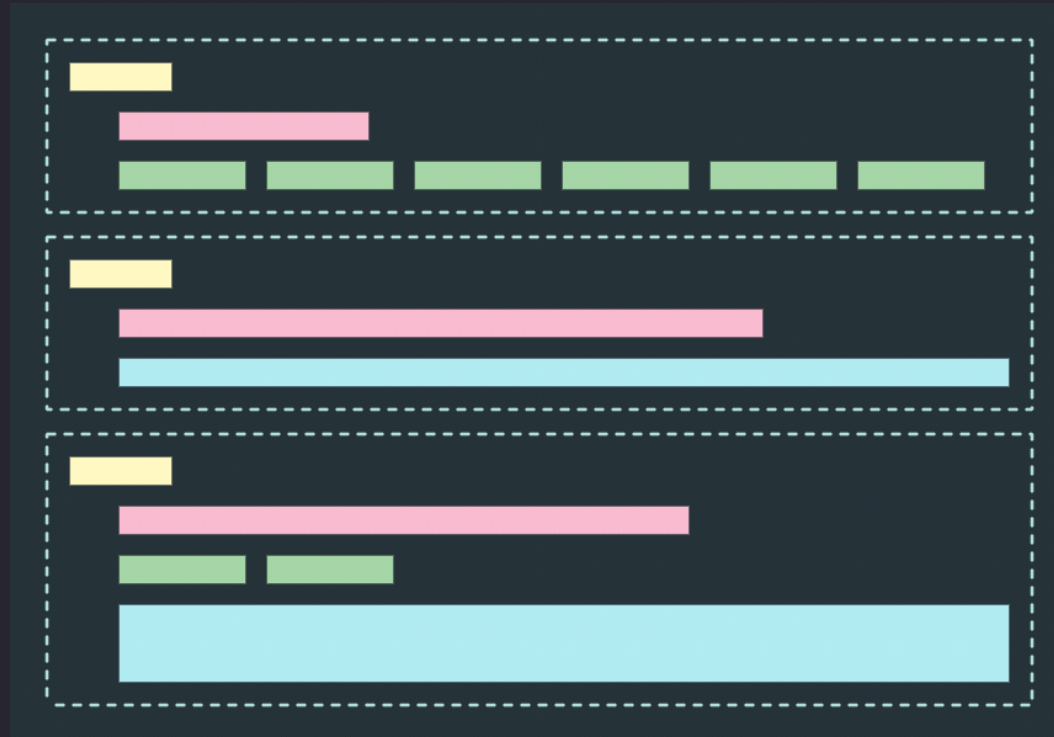


# NoSQL Document Database



# Document Databases

Store collections of schemaless documents



# Document Databases

Choose a document database if:

- the records you store don't have the same structure as one another
- you need to change data structures without downtime
- you like high availability



# Data Design for PouchDB

- data structure: include nested data/array, omit empty fields
- identifiers: pick a meaningful ID where appropriate
- beware updating/ appending data: these cause conflicts

# CouchDB

Cluster Of Unreliable Commodity Hardware

- HTTP API
- JSON data format
- Performant views use JavaScript and MapReduce
- Ad-hoc queries with a JSON structure using Mango

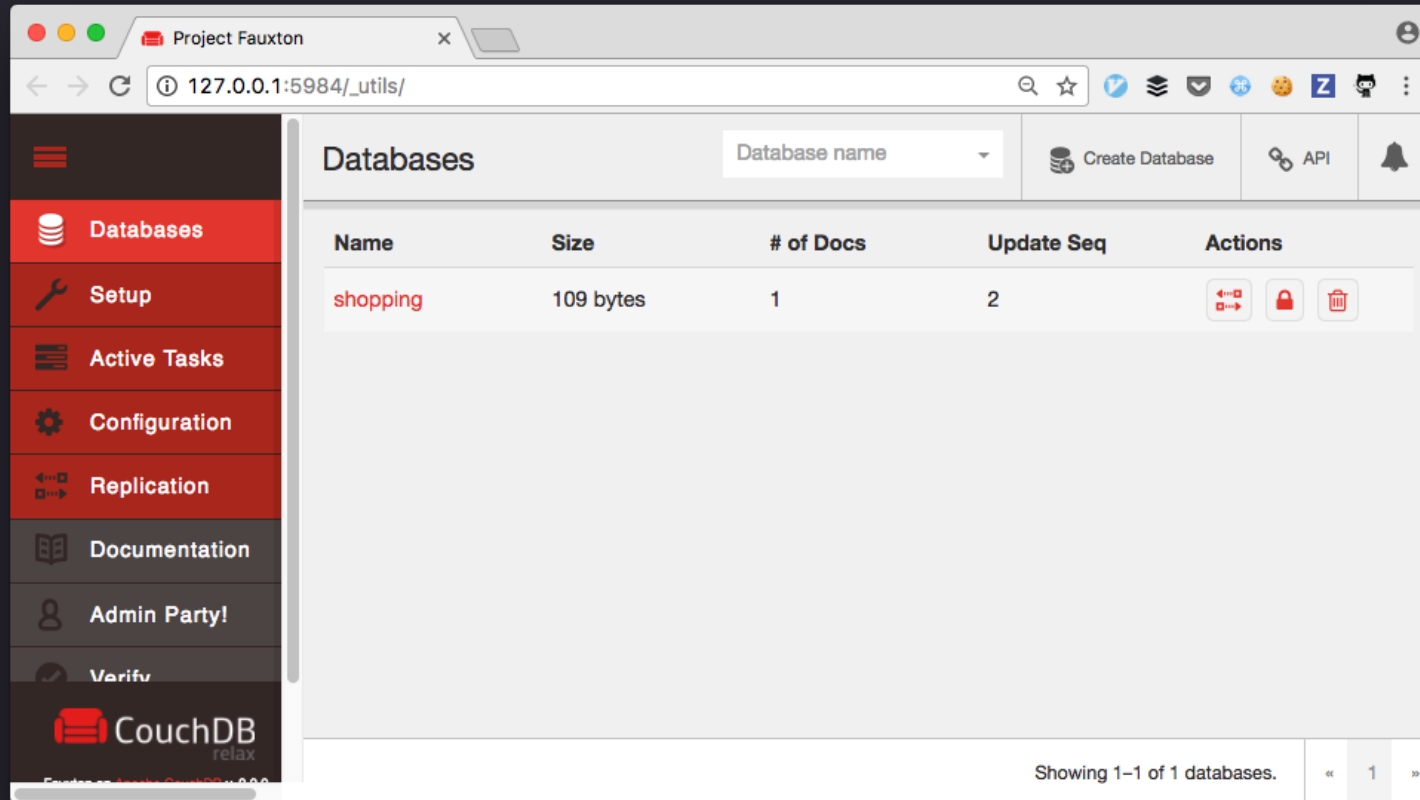
# Curl and Not-Curl

- love curl? (<https://curl.haxx.se/>)
  - try jq (<https://stedolan.github.io/jq/>)
- hate curl? Try one of these
  - http-console <https://github.com/cloudhead/http-console>
  - Postman <https://www.getpostman.com/>
- for more, try this HTTP Tools post (and comments):  
<http://lornajane.net/posts/2017/http-tools-roundup>




# Fauxton

Friendly web interface

# Fauxton



The screenshot shows the Fauxton web interface in a browser window. The browser tab is titled "Project Fauxton" and the address bar shows "127.0.0.1:5984/\_utils/". The interface has a dark sidebar on the left with a menu containing "Databases", "Setup", "Active Tasks", "Configuration", "Replication", "Documentation", "Admin Party!", and "Verify". The main content area is titled "Databases" and features a search box for "Database name", a "Create Database" button, and an "API" button. Below this is a table with the following data:

Name	Size	# of Docs	Update Seq	Actions
shopping	109 bytes	1	2	  

At the bottom right of the interface, it says "Showing 1-1 of 1 databases." with a pagination control showing "1" between "«" and "»" symbols.

# Fauxton

The screenshot shows the Fauxton web interface in a browser window. The browser's address bar displays the URL: `127.0.0.1:5984/_utils/#/database/shopping/_all_docs?include_docs=true&conflicts...`. The interface is divided into several sections:

- Left Sidebar:** A vertical navigation menu with the following items: Databases, Setup, Active Tasks, Configuration, Replication, Documentation, Admin Party!, and Verifu. At the bottom of the sidebar is the CouchDB logo with the tagline "relax".
- Header:** The word "shopping" is displayed in a large font, indicating the current database. To its right are tabs for "JSON" (selected) and "Table", along with a "Docs" checkbox and various utility icons.
- Main Content Area:** A list of database actions is visible, including "All Documents", "Run A Query with Mango", "Permissions", "Changes", and "Design Documents".
- Document View:** A document with the ID "hat" is displayed in a dark-themed editor. The document's content is a JSON object:

```
{
  "_id": "hat",
  "_rev": "2-15b01d3c5cd2ed475e6ce4cf84b51990",
  "value": {
    "rev": "2-15b01d3c5cd2ed475e6ce4cf84b51990"
  },
  "key": "hat",
  "doc": {
    "_id": "hat",
    "_rev": "2-15b01d3c5cd2ed475e6ce4cf84b51990",
    "colour": "blue"
  }
}
```
- Footer:** At the bottom of the document view, it says "Showing document 1 - 1." and "Documents per page: 20".

# CouchDB: Lovely Doc DB

I could stop here:

- JSON format
- HTTP interface and nice web UI
- Scales well
- Modern, performant document database

# Changes Feed

A feed containing all database changes.

```
GET /_changes
```



# Replication

# Replication

- Replication can be in either direction - or both
- Can be one-off, or continuous
- Other CouchDB-compatible storage also exists
  - e.g. PouchDB, a JavaScript implementation

# Conflicts

Change docs in both places, replicate again:

```
87bf-bluemix.cloudant.com:443/shopping> GET /hat?conflicts=true
```

```
{
  _id: '123',
  _rev: '4-ecbc38075f9a8535c123e523519613b9',
  item: 'cheese',
  _conflicts: [ '3-0bb689d59034fb769d99dcf697ae2de7' ]
}
```

CouchDB will always choose the same "winning" doc

# Conflicts

Fetch the "losing" doc(s) with ?rev= parameter

```
87bf-bluemix.cloudant.com:443/shopping> GET /123?rev=3-0bb689d5903
```

```
{
  _id: 123,
  _rev: '3-0bb689d59034fb769d99dcf697ae2de7',
  item: 'cheddar cheese'
}
```

CouchDB doesn't store old revisions forever

# Mango

# Mango: CouchDB Queries

Mango is a mongo-like query language, useful for ad-hoc querying

It is a JSON structure containing:

- **Selector:** the criteria to match records on
- **Fields:** which fields to return
- **Sort:** what order you'd like that in (use with **Skip**)
- **Limit:** how many records (default = 25)

# Mango: Example Query

Use a query like this with the `_find` endpoint

```
{
  "selector": {
    "Year": {"$eq": "2012"}
  },
  "fields": ["Quarter", "Product line"],
  "limit": 5
}
```

# Mango: Example Query

```
$ curl -X POST -H Content-Type:application/json \  
http://localhost:5984/products/_find --data @mango.json
```

```
{ "warning": "no matching index found, create an index to optimize q  
"docs": [  
  { "Quarter": "Q1 2012", "Product line": "Mountaineering Equipment" },  
  { "Quarter": "Q1 2012", "Product line": "Mountaineering Equipment" },  
  { "Quarter": "Q1 2012", "Product line": "Mountaineering Equipment" },  
  { "Quarter": "Q1 2012", "Product line": "Mountaineering Equipment" },  
  { "Quarter": "Q1 2012", "Product line": "Mountaineering Equipment" }  
]}
```



# Mango: Indexes

Describe the index in JSON, then use the `_index` endpoint

```
{  
  "index": {  
    "fields": ["Year"]  
  },  
  "name": "Year"  
}
```

# Mango: Indexes

```
$ curl -X POST -H Content-Type:application/json \  
http://localhost:5984/products/_index --data @index.json
```

```
{  
  "result": "created",  
  "id": "_design/e9b54f2ac34b8823ccbe8aaf6f406d464f50f521",  
  "name": "Year"  
}
```

Check which indexes are used by putting `_explain` where the `_find` normally goes!

# Views

# Views

- Written in Javascript
- Use MapReduce
- The map results are stored
- Can be used either for filtering, or for aggregation

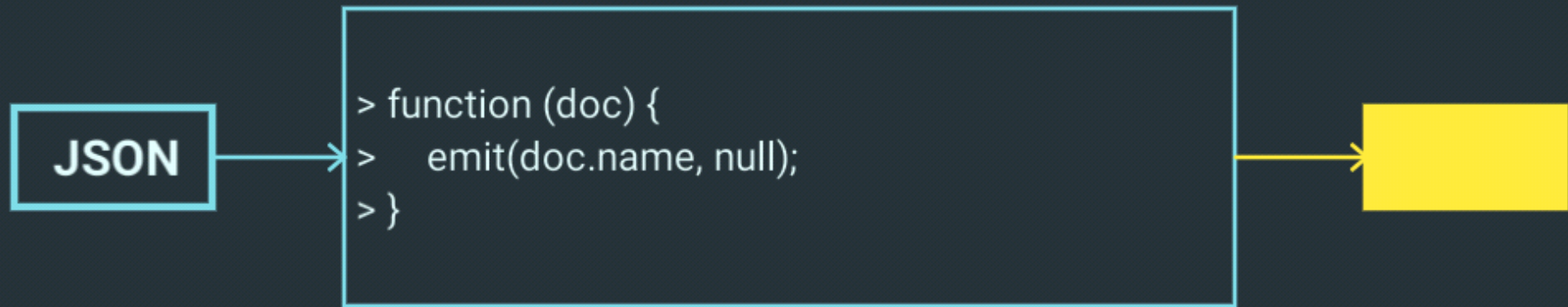
# MapReduce Primer: Map

- Examine each document, "emit" 0+ keys/value pairs
- Scales well because each document is independent
- To filter a collection of documents, use map step only

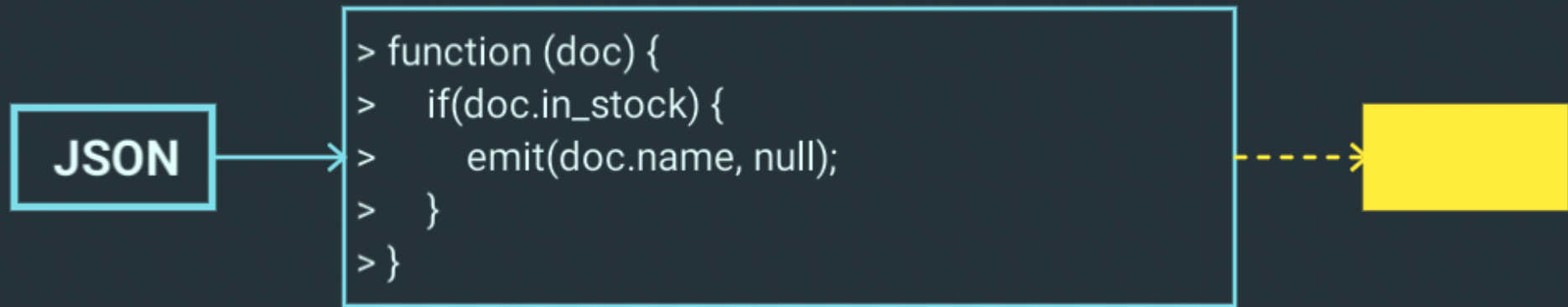
# MapReduce Primer: Map

```
> {  
>   product_name: shoe77,  
>   departments: [outdoor, sports, womens],  
>   in_stock: true,  
>   date_added: 2017-03-16  
> }
```

# MapReduce Primer: Map

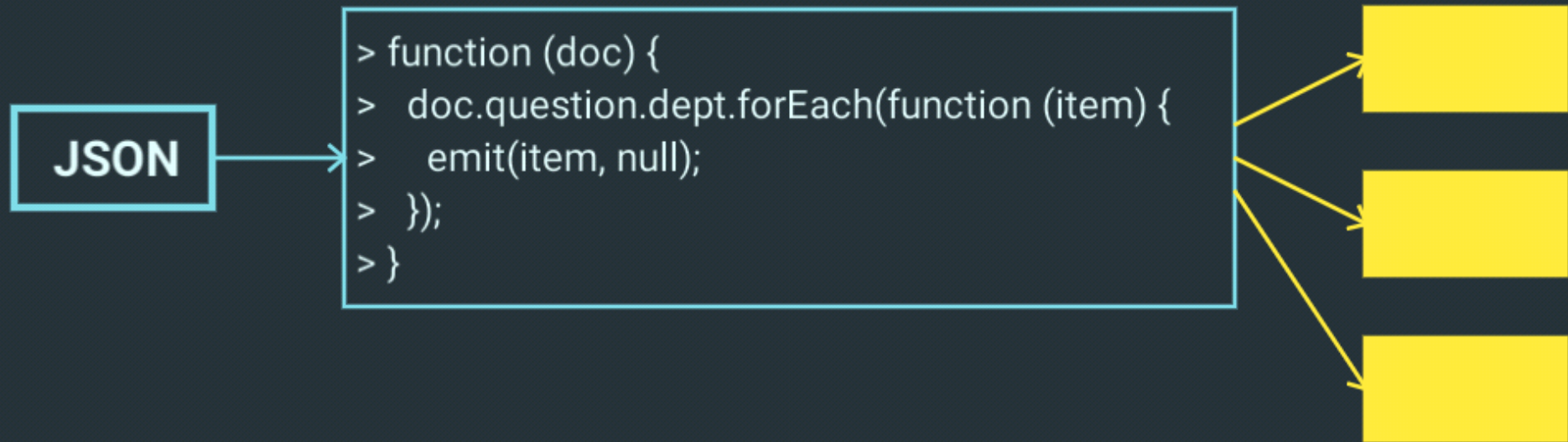


# MapReduce Primer: Map

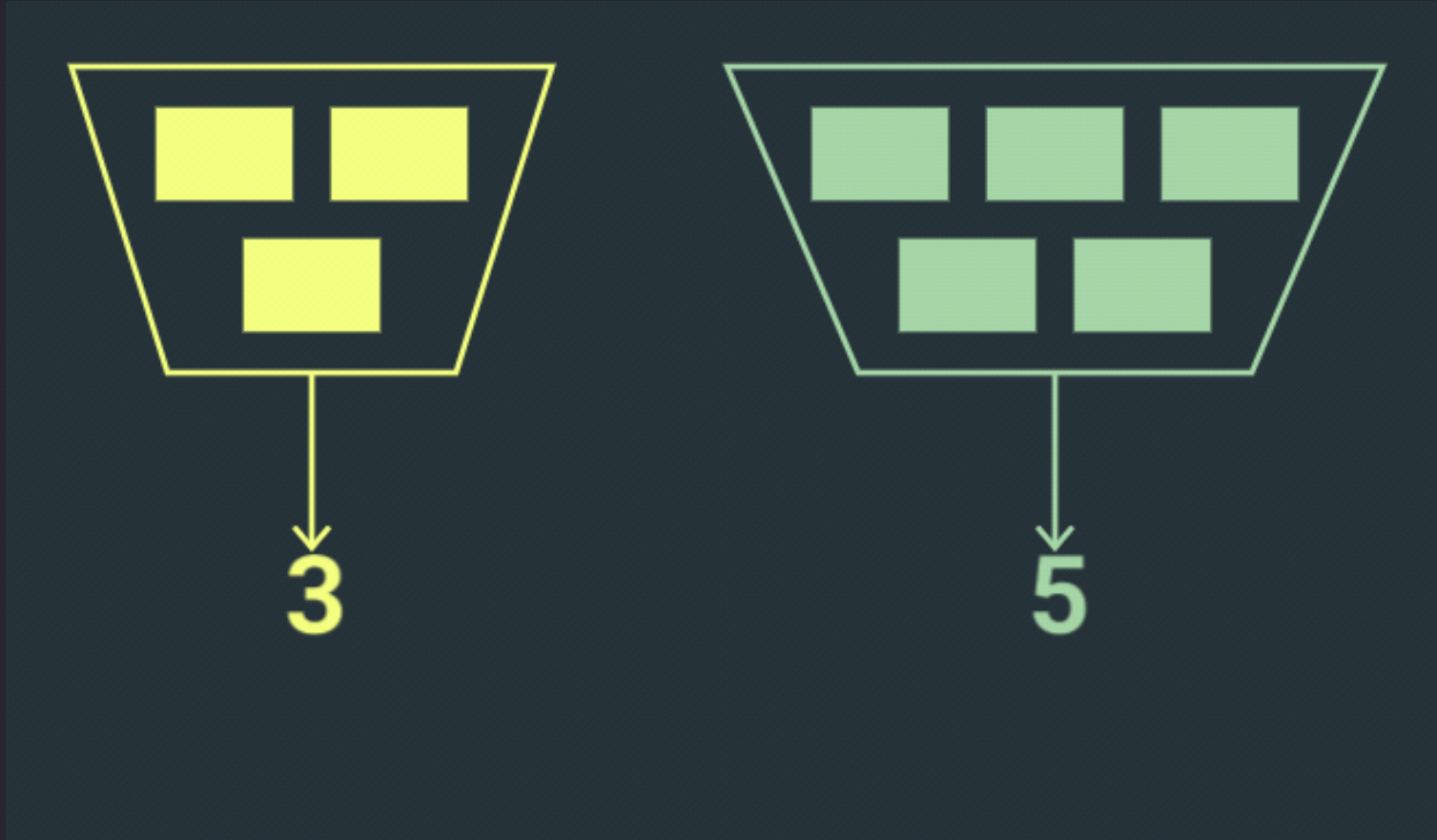




# MapReduce Primer: Map



# MapReduce Primer: Reduce



# MapReduce Primer: Reduce

- "Reduce" values in batches with the same key
- CouchDB has useful built in functions for most things
- Use reduce step when you want aggregate data
  - (SQL equivalent: a query with GROUP BY)

# Views Example

The screenshot shows the Project Fauxton web interface for a CouchDB database named 'products'. The left sidebar contains navigation options: Databases, Setup, Active Tasks, Configuration, Replication, Documentation, Admin Party!, and Verifu. The main content area is titled 'products' and lists options: All Documents, Run A Query with Mango, Permissions, Changes, and Design Documents. A 'New View' button is highlighted in the Design Documents section. The 'New View' configuration panel on the right includes a 'Design Document' dropdown set to 'New document', a text input for the design document name '\_design/newDesignDoc', an 'Index name' input set to 'new-view', and a 'Map function' input containing the following code:

```
1 function (doc) {  
2   emit(doc._id, 1);  
3 }
```

# OfflineFirst Apps With PouchDB and CouchDB

# Example Apps

Ready-made shopping list examples are available:

- VanillaJS and PouchDB (a more detailed example)
- Polymer and PouchDB
- React and PouchDB
- Vue.js and PouchDB
- React Native and PouchDB

<https://github.com/ibm-watson-data-lab/shopping-list>

# Resources

- <https://lornajane.net>
- <https://github.com/lornajane/robust-shopping-list>
- <https://github.com/ibm-watson-data-lab/shopping-list>
- <https://offlinefirst.org>
- <http://hood.ie/>