Storing Data in PouchDB



James Millar FREELANCE SOFTWARE DEVELOPER

@jamesmillar www.james-millar.co.uk

Overview



Introducing PouchDB

- Adapters

Setting up PouchDB

Storing and managing data

- Revisions

Asynchronous functions

Storing product data



Introducing PouchDB



Introducing PouchDB



JavaScript client-side database



Store complex objects



Abstraction layer over Indexed DB



Increased performance



Sync across devices



Relational Data

ld	Name	Description	UnitPrice



Relational Data

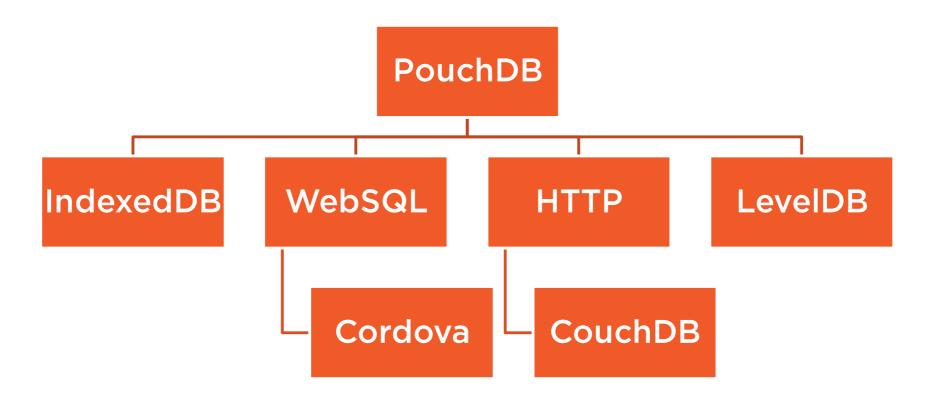
Id	Name	Description	UnitPrice
1	Columbian Black	Rich dark and aromatic coffee	4.95
2	Peruvian Gold	A light and energetic blend	6.00
3	Italian Sunshine	Classical Italian coffee	3.40



Unstructured Documents

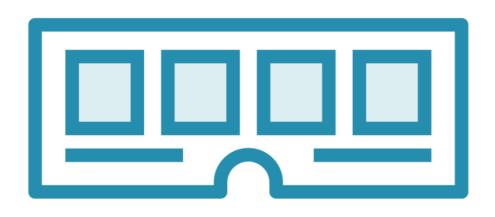
```
"_id": "productlist",
"LastUpdatedDate": "020-09-24T14:31:14.241Z",
"products": [
  "Columbian Black",
  "Peruvian Gold",
  "Italian Sunshine"
```

PouchDB Adapters

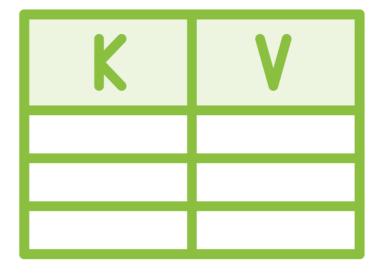




Advanced Adapters



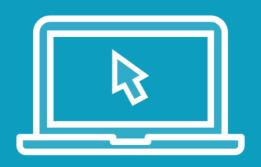
In memory adapter Ideal for unit tests



Local storage adapter Support old browsers



Demo



Add support for PouchDB

Find out what adapter is being used

Add the in memory adapter



Working with PouchDB



```
"_id": "columbianGold",
"stock": 400,
"deliveryDays":5
}
```

◄ Every document needs an ID

Creating a Document

```
var db = new PouchDB('ProductsDatabase');
var doc = {
   "_id": "columbianGold",
   "stock": 400,
   "deliveryDays":5
};
db.put(doc);
```



Getting a Document

```
db.get('columbianGold').then(function (doc)
{
    console.log(doc);
}
);
```

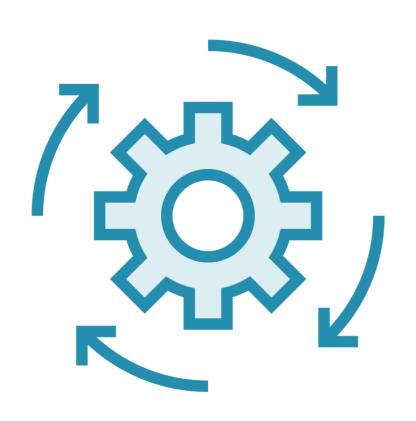


Getting a Document

```
{
   "_id": "columbianGold",
   "stock": 400,
   "deliveryDays":5,
   "_rev": "1-a58cb476d163f302e4f73d4d19ca2704"
}
```



Revisions



Randomly generated ID

Changes when documents are

- Created
- Updated

Updates require entire document including revision marker



Revision Trees

Git Revision Tree

Rev 4a Rev 4 Rev 3 Rev 2 Rev 1

PouchDB Revision Tree

```
Rev 5
Rev 4
Rev 3
Rev 2
Rev 1
```



Updating a Document

```
db.get('columbianGold').then(
function (doc) {
doc.stock = 350;
return db.put(doc);
});
```

Deleted Attribute

```
"_id": "columbianGold",

"stock": 400,

"deliveryDays":5,

"_rev": "1-a58cb476d163f302e4f73d4d19ca2704",

"_deleted": true
```



Deleting a Document

```
db.get('columbianGold').then(function (doc) {
  return db.remove(doc);
});
```



Asynchronous Functions



```
db.get('products', function (error, doc) {
  if (error)
    {// handle the error}
  else
    {// get operation was successful}
});
```

Callback Format

Avoid callbacks if possible



Promise

An object that will contain the result of a deferred operation



```
db.get('products').then(function (doc)
{ // get operation was successful}
).catch(function (err)
{ // handle the error}
);
```

Promise Format



Promises



Not all browsers support promises

- Pouch DB falls back to using Lie

Use custom 3rd party promise libraries

- Must meet the Promises A+ spec



```
db.get('products').then(function (doc)
{ // do this}
).then(function (doc)
{ // then do this}
).catch(function (err)
{ // handle the error}
);
```

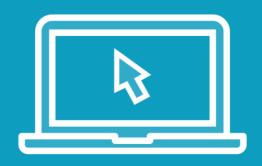
Promise format

Can chain multiple promises together

Cleaner error handling



Demo



Store API data in PouchDB



Summary



Introduced PouchDB

- Complex objects
- Asynchronous API
- Adapters

Documents

- Managing documents
- Revision id

Working with promises





James Millar FREELANCE SOFTWARE DEVELOPER

@jamesmillar www.james-millar.co.uk