Distributed CouchApps – Embracing Eventual Consistency

International PHP Conference – Spring Edition

Kore Nordmann (@koredn, kore@qafoo.com)

05.06.2012

Part I

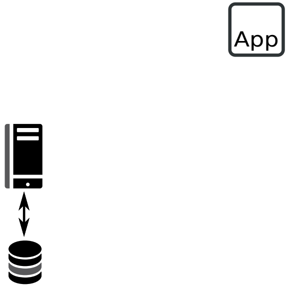
Introduction

Outline

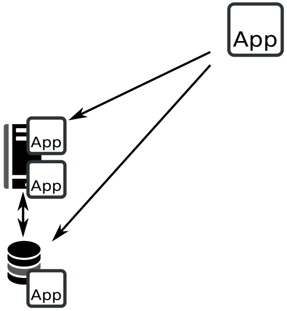
Couch Apps

The idea

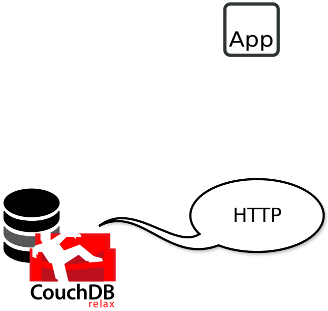
The idea



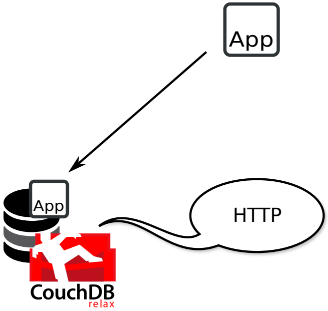
The idea



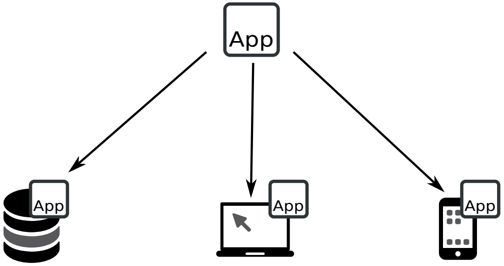
The idea



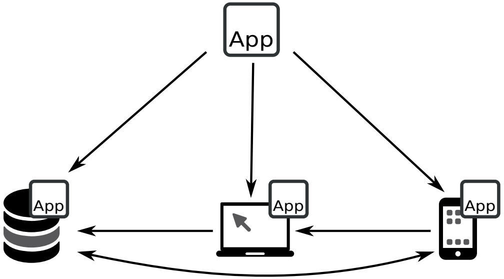
The idea



The idea



The idea



Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization
  - Servers will *eventually* be consistent

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization
  - Servers will *eventually* be consistent
- Applications
  - Mirror database into userspace

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization
  - Servers will *eventually* be consistent
- Applications
  - Mirror database into userspace
  - Offline usage and synchronization of Browser applications

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization
  - Servers will *eventually* be consistent
- Applications
  - Mirror database into userspace
  - Offline usage and synchronization of Browser applications
  - TouchDB developed by CouchBase
    - `https://github.com/couchbaselabs/TouchDB-Android`
    - `https://github.com/couchbaselabs/TouchDB-iOS`

Eventual consistency

- Eventual consistency
  - CouchDB does not enforce relation integrity
  - CouchDB servers may use delayed synchronization
  - Servers will *eventually* be consistent
- Applications
  - Mirror database into userspace
  - Offline usage and synchronization of Browser applications
  - TouchDB developed by CouchBase
    - `https://github.com/couchbaselabs/TouchDB-Android`
    - `https://github.com/couchbaselabs/TouchDB-iOS`
  - PouchDB – JavaScript CouchDB implementation based on HTML 5 Indexed DB `https://github.com/mikeal/pouchdb`

Design documents

- Defining the “logic” of a CouchApp

```
{
  _id                 : "_design/app",
  _rev                : "1-0139ca4e37f873b846fd37714a191e1a",
  _attachments        : { ... },
  views               : { ... },
  rewrites            : [ ... ],
  filters             : { ... },
  lists               : { ... },
  shows               : { ... },
  modules             : { ... },
  validate_doc_update : "...",
}
```

Part II

Consistency

Outline

Consistency

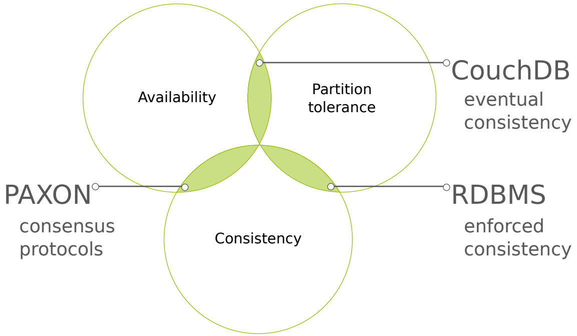
Replication

Remember:

- Multi-Version Concurrency Control (MVCC)
  - All documents in the database are versioned
- There is no ensured inter document consistency in CouchDB (relational integrity)

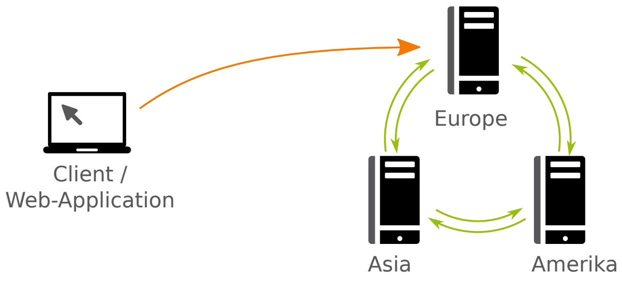
Scaling: The CAP theorem

- The CAP theorem, read more in “CouchDB: The Definitive Guide” [JCA09]

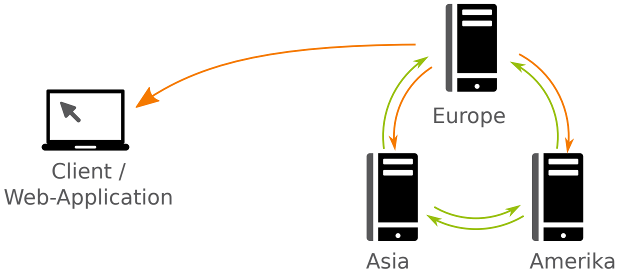


- CouchDB employs “Eventual Consistency” [Vog09]

Eventual consistency

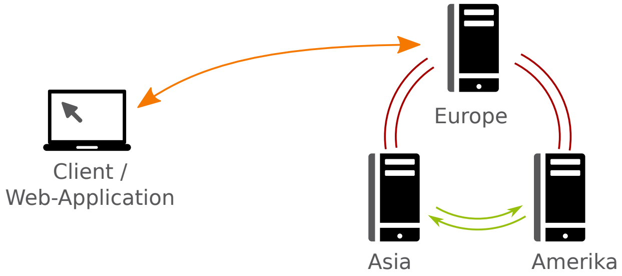


Eventual consistency



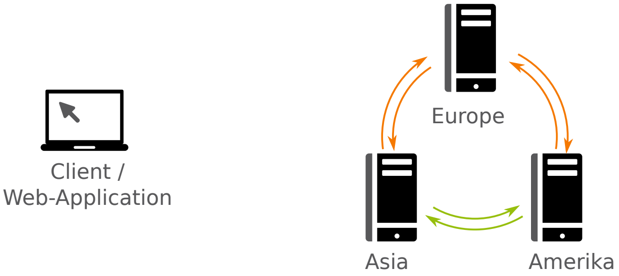
- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes

Eventual consistency



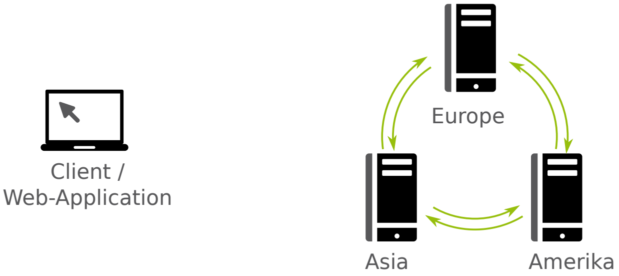
- Delayed, triggered synchronization (push, pull)
    - Deterministic (manual) conflict resolution on replication on all nodes

Eventual consistency



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
- Scales well for seldom concurrent writes

Eventual consistency



- Delayed, triggered synchronization (push, pull)
  - Deterministic (manual) conflict resolution on replication on all nodes
- Scales well for seldom concurrent writes
  - Structure your documents accordingly

Outline

Consistency

Replication

Replication

- Replication is trivial

```
$ curl -X POST http://localhost:5984/_replicate \
    -H 'Content-Type:␣application/json' \
    -d '{"source":"ipc",␣\
␣␣␣␣␣␣␣␣␣␣"target":"http://user:pass@192.168.1.3:5984/ipc"}'

{ "ok":                 true,
  "no_changes":         true,
  "session_id":         "73d69e7b5cdaea059e55ed1db7802151",
  "source_last_seq":    141,
  "history": [ {
    "session_id":           "73d69e7b5cdaea059e55ed1db7802151",
    "start_time":           "Thu,␣23␣Sep␣2010␣08:47:05␣GMT",
    "end_time":             "Thu,␣23␣Sep␣2010␣08:51:53␣GMT",
    "start_last_seq":       135,
    "end_last_seq":         141,
    "recorded_seq":         141,
    "missing_checked":      0,
    "missing_found":        1,
    "docs_read":            1,
    "docs_written":         1,
    "doc_write_failures":   0
  } ]
}
```

Replication details

- Source and target can be any combination of remote and local URLs

Replication details

- Source and target can be any combination of remote and local URLs
- Set {`"continuous": true`} to have CouchDB keeping the replication alive.

Replication details

- Source and target can be any combination of remote and local URLs
- Set {`"continuous": true`} to have CouchDB keeping the replication alive.
  - Persisted during restarts since CouchDB 1.1

Replication details

- Source and target can be any combination of remote and local URLs
- Set {`"continuous": true`} to have CouchDB keeping the replication alive.
  - Persisted during restarts since CouchDB 1.1
  - Can be cancelled again using {`"cancel": true`}

Replication details

- Source and target can be any combination of remote and local URLs
- Set {`"continuous": true`} to have CouchDB keeping the replication alive.
  - Persisted during restarts since CouchDB 1.1
  - Can be cancelled again using {`"cancel": true`}
- Potential replication failures:
  - Crashed node
  - Network failure
  - `validate_docs_update` does not allow writing

Replication details

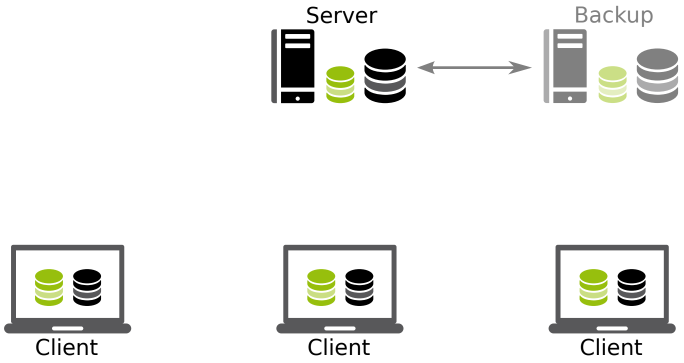
- Source and target can be any combination of remote and local URLs
- Set `{"continuous": true}` to have CouchDB keeping the replication alive.
  - Persisted during restarts since CouchDB 1.1
  - Can be cancelled again using `{"cancel": true}`
- Potential replication failures:
  - Crashed node
  - Network failure
  - `validate_docs_update` does not allow writing
    - Replication will be resumed once the error is fixed.

Checking replication status

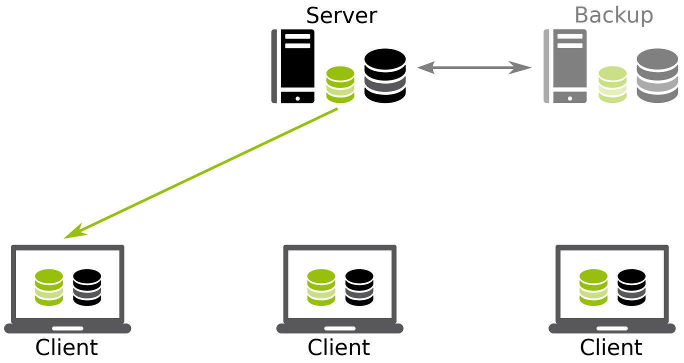
```
$ curl -X GET http://user:pass@localhost:5984/_active_tasks

[ { "type":    "Replication",
    "task":    "228689: http://koredn:*****@192.168.1.3:5984/ipc/ -> ipc",
    "status": "W Processed source update #20",
    "pid":     "<0.273.0>"
  }, {
    "type":    "Replication",
    "task":    "0444e5: ipc -> http://koredn:*****@192.168.1.3:5984/ipc/",
    "status": "MR Processed source update #141",
    "pid":     "<0.292.0>"
  }
]
```

Outline

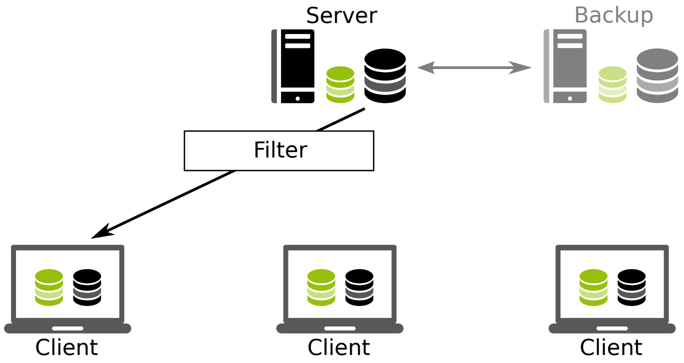
Filtered Replication



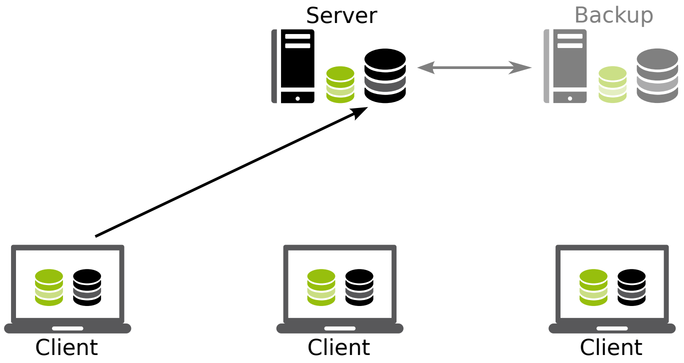
Filtered Replication



Filtered Replication



Filtered Replication



Filtered Replication

- Append a filter function to design document

```
{ "_id":       "_design/app",
  "language": "javascript",
  "filters": {
    "for_user": "function( doc, req ) { return false; }",...

  }...

}
```

Filtered Replication

- Common filtering function

```
function( doc, req ) {
  if( !req.userCtx.name ) {
    throw( "Unauthorized!" );
  }

  if( doc.recipients &&
      doc.recipients.indexOf( req.userCtx.name ) !== -1 )
  {
    return true;
  }

  return false;
}
```

Filtered Replication

- Usage during replication

```
$ curl -X POST http://localhost:5984/_replicate \
    -H 'Content-Type: application/json' \
    -d '{"source":"ipc", \
          "target":"http://user:pass@192.168.1.3:5984/ipc", \
          "filter":"app/for_user"}'...
```

Part III

Mechanics

Outline

Restrospect

vHosts & Rewrites

Documents

- CouchDB stores arbitrary JSON documents
  - Deep structures are fine
  - You can index on properties in deep structures

Attachments

- CouchDB allows you to attach files to documents

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated

Attachments

- CouchDB allows you to attach files to documents
- Files are replicated
- You can serve full Web-Applications from a CouchDB
- Deploy using PUSH-replication

Views

- Index the documents in the database
  - Provide structured / filtered access to documents
  - Can calculate statistics on documents
  - Can provide JOIN-Views on documents

Views

- Index the documents in the database
  - Provide structured / filtered access to documents
  - Can calculate statistics on documents
  - Can provide JOIN-Views on documents
- Usually written in JavaScript / Erlang

Views

- Index the documents in the database
  - Provide structured / filtered access to documents
  - Can calculate statistics on documents
  - Can provide JOIN-Views on documents
- Usually written in JavaScript / Erlang
- Small simple functions executed for every document in database
  - Results are stored in an interatively built append-only disk-serialized B-Tree

Views

- Index the documents in the database
  - Provide structured / filtered access to documents
  - Can calculate statistics on documents
  - Can provide JOIN-Views on documents
- Usually written in JavaScript / Erlang
- Small simple functions executed for every document in database
  - Results are stored in an interatively built append-only disk-serialized B-Tree
- Emplyoing the map-reduce pattern

Validate doc updates

- JavaScript function, which accepts or rejects document updates

Outline

Restrospect

vHosts & Rewrites

Rewrites

- Allows to rewrite URLs
- Specified in design documents

```
{ "_id":       "_design/app",
  "rewrites": [
    { "from": "/blog",
      "to":    "../../blog/index.html",
    },...

  ]
}
```

Rewrites

- Allows to rewrite URLs
- Specified in design documents

```
{ "_id":      "_design/app",
  "rewrites": [
    { "from": "/blog",
      "to":   "../../blog/index.html",
    },...

  ]
}
```

- Requested as: `http://localhost:5984/db/_design/app/_rewrite/blog`
- Rewrites to: `http://localhost:5984/db/blog/index.html`

Rewrites

- Parameters are possible

```
{ "_id":       "_design/app",
  "rewrites": [
    { "from": "/images/:image",
      "to":    "../../images/:image",
    },...

  ]
}
```

Rewrites

- Parameters are possible

```
{ "_id":      "_design/app",
  "rewrites": [
    { "from": "/images/:image",
      "to":   "../../images/:image",
    },...

  ]
}
```

- Requested as: `http://localhost:5984/db/_design/app/_rewrite/images/favicon.png`
- Rewrites to: `http://localhost:5984/db/images/favicon.png`

Rewrites

- Match anything, including query parameters
  - Especially useful for views, will be covered later

```
{ "_id":       "_design/app",
  "rewrites": [
    { "from": "/*",
      "to":    "../../blog/404.html",
    },...

  ]
}
```

Rewrites

- Match anything, including query parameters
  - Especially useful for views, will be covered later

```
{ "_id":      "_design/app",
  "rewrites": [
    { "from":  "/*",
      "to":    "../../blog/404.html",
    },...

  ]
}
```

- Requested as: `http://localhost:5984/db/_design/app/_rewrite/something`
- Rewrites to: `http://localhost:5984/db/blog/404.html`

vHosts

- Those are still pretty ugly URLs. . .

vHosts

- Those are still pretty ugly URLs…
- vHosts to the rescue:

```
$ cat /etc/couchdb/local.ini
[...]
myhost:5984 = /db/_design/app/_rewrite
[...]
```

vHosts

- Those are still pretty ugly URLs. . .
- vHosts to the rescue:

```
$ cat /etc/couchdb/local.ini
[...]
myhost:5984 = /db/_design/app/_rewrite
[...]
```

- Requested as:
  `http://myhost:5984/blog`
- Rewrites to:
  `http://localhost:5984/db/blog/index.html`

vHosts

- Those are still pretty ugly URLs. . .
- vHosts to the rescue:

```
$ cat /etc/couchdb/local.ini
[...]
myhost:5984 = /db/_design/app/_rewrite
[...]
```

- Requested as:
  `http://myhost:5984/blog`
- Rewrites to:
  `http://localhost:5984/db/blog/index.html`
  - A rewrite rule for "/" is also possible, of course. . .

Tampering

Demo

Part IV

Graceful degration

Outline

Server side formatting

Server side formatting

- Dataformatting usually happens in the client using JavaScript

Server side formatting

- Dataformatting usually happens in the client using JavaScript
- . . . but we *can* also do this on the server
  - *show* functions display a single document
  - *list* functions display a view result

Show function

- Displays a specified document
- Extend your design document:

```
{ "_id":        "_design/app",
  "language": "javascript",
  "shows": {
    "wiki": "function( doc, req ) { /*...*/ return responseObject; }",
  },...

}
```

Show function

- Displays a specified document
- Extend your design document:

```
{ "_id":       "_design/app",
  "language": "javascript",
  "shows": {
    "wiki": "function( doc, req ) { /* ... */ return responseObject; }",
  },...

}
```

- Request:
  `/db/_design/app/_show/wiki/someDocId`

Show function

- Displays a specified document
- Extend your design document:

```
{ "_id":       "_design/app",
  "language": "javascript",
  "shows": {
    "wiki": "function( doc, req ) { /* ... */ return responseObject; }",
  },...

}
```

- Request:
  `/db/_design/app/_show/wiki/someDocId?param=value`

Simple show function

```
function( doc, req ) {
  if ( doc ) {
    return {
      body: "Hello World"
    }
  } else { // document not found
    if( req.id ) {
      // handle unused doc id
    } else {
      // handle unspecified doc id
    }
  }
}
```

Simple show function

```
function( doc, req ) {
  if ( doc ) {
    return {
      body: "Hello World"
    }
  } else { // document not found
    if( req.id ) {
      // handle unused doc id
    } else {
      // handle unspecified doc id
    }
  }
}
```

- Request and response:
  `https://wiki.apache.org/couchdb/ExternalProcesses`

List function

- Displays a specified view
- Extend your design document:

```
{ "_id":       "_design/app",
  "language": "javascript",
  "views": {
    "by_name": {
      "map":     "function(doc) { if (doc.type == 'user') emit(doc.name, null) }",
    },
  },
  "lists": {
    "posts": "function( head, req ) { /*...*/ }",
  },...

}
```

List function

- Displays a specified view
- Extend your design document:

```
{ "_id":       "_design/app",
  "language": "javascript",
  "views": {
    "by_name": {
      "map":     "function(doc) { if (doc.type == 'user') emit(doc.name, null) }",
    },
  },
  "lists": {
    "posts": "function( head, req ) { /*...*/ }",
  },...

}
```

- Request:
  `/db/_design/app/_list/posts/by_name`

List function

- Displays a specified view
- Extend your design document:

```
{ "_id":       "_design/app",
  "language": "javascript",
  "views": {
    "by_name": {
      "map":     "function(doc) { if (doc.type == 'user') emit(doc.name, null) }",
    },
  },
  "lists": {
    "posts": "function( head, req ) { /*...*/ }",
  },...

}
```

- Request:
  `/db/_design/app/_list/posts/by_name?viewParams`

List function example

```
function(head, req) {
  start( {
    "headers": {
      "Content-Type": "text/html"
     }
  } );
  while( row = getRow() ) {
    send( row.value );
  }
}
```

List function example

```
function(head, req) {
  start( {
    "headers": {
      "Content-Type": "text/html"
    }
  } );
  while( row = getRow() ) {
    send( row.value );
  }
}
```

- `start()` call to send headers

List function example

```
function(head, req) {
  start( {
    "headers": {
      "Content-Type": "text/html"
    }
  } );
  while( row = getRow() ) {
    send( row.value );
  }
}
```

- `start()` call to send headers
- `getRow()` call to receive next row from view

List function example

```
function(head, req) {
  start( {
    "headers": {
      "Content-Type": "text/html"
    }
  } );
  while( row = getRow() ) {
    send( row.value );
  }
}
```

- `start()` call to send headers
- `getRow()` call to receive next row from view
- `send()` echo something to the client

Outline

- Server side formatting
  - Modules

Server side code reuse

- It is possible to reuse code on server side
- CouchDB can use JavaScript code from CommonJS modules, attached to your design document

```
{ _id:"_design/test",
  language: "javascript",
  modules: {
    jquery    : "exports.jquery = /* Source code */;",
    something: "exports.answer = 42;",
  },
  shows: {
    post: "function() { var lib = require('modules/something'); return lib.answer; };"
  },
}
```

- The name `modules` ist arbitrary

Server side code reuse

- It is possible to reuse code on server side
- CouchDB can use JavaScript code from CommonJS modules, attached to your design document

```
{ _id:"_design/test",
  language: "javascript",
  modules: {
    jquery    : "exports.jquery = /* Source code */;",
    something: "exports.answer = 42;",
  },
  shows: {
    post: "function() { var lib = require('modules/something'); return lib.answer; };"
  },
}
```

- The name `modules` ist arbitrary
  - You can even put the library below `views` or `shows`
  - `https://wiki.apache.org/couchdb/CommonJS_Modules`

Summary

- Eventually Consistent Offline Replicated JavaScript Frontend, Backend & Data

Summary

- Eventually Consistent Offline Replicated JavaScript Frontend, Backend & Data
- Example application: `http://github.com/Qafoo/Lounge`
- Application stub: `http://github.com/Qafoo/Stub`

Thanks for listening

- More:
  - `http://kore-nordmann.de`
  - `http://qafoo.com`
  - `http://github.com/Qafoo`
- Feedback:
  - `https://joind.in/6650`

Bibliography I

[JCA09] Noah Slater J. Chris Anderson, Jan Lehnardt, *Couchdb: The definitive guide*, O'Reilly Media, Inc., 2009.

[Vog09] Werner Vogels, *Eventually consistent - revisited*, `http://www.allthingsdistributed.com/2008/12/eventually_consistent.html`, December 2009.