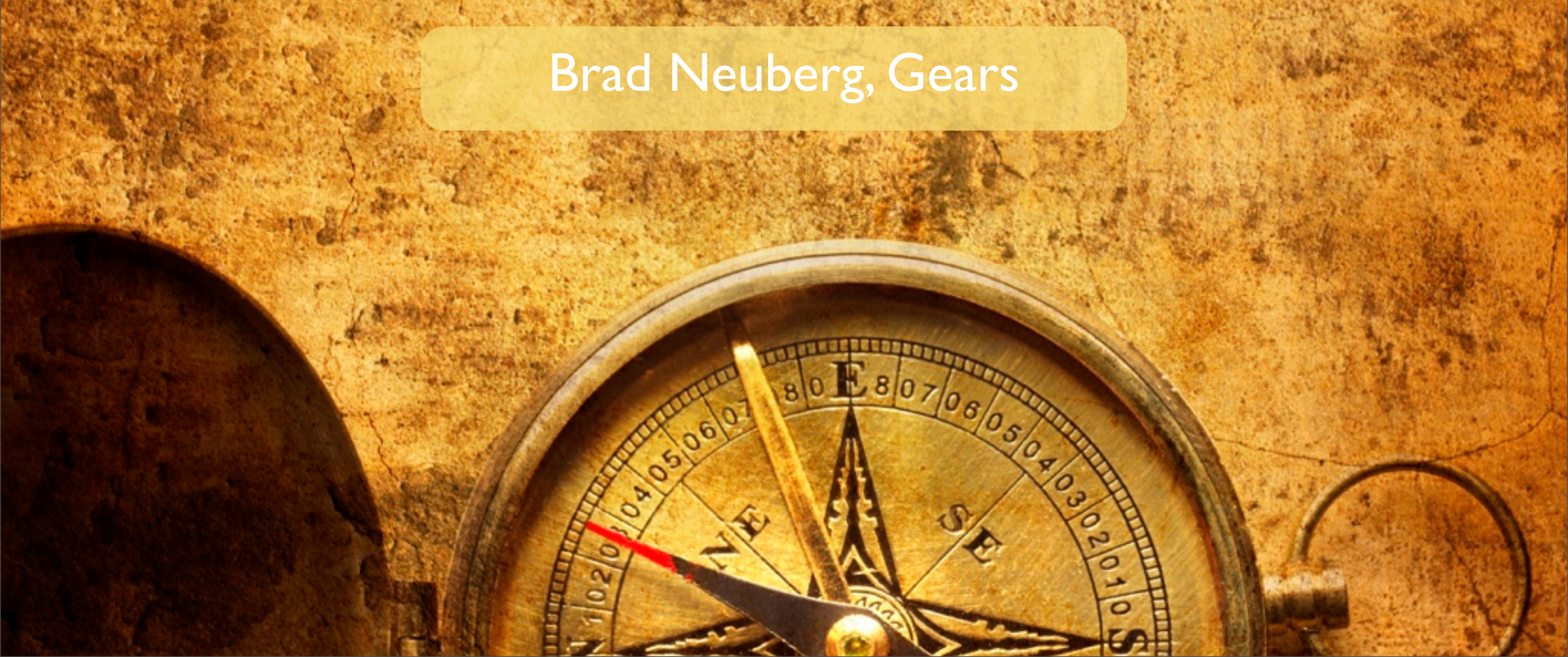
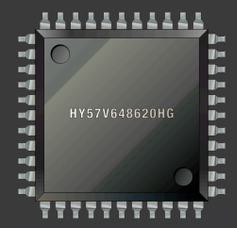
Creating a Client-Side Search Engine with Gears

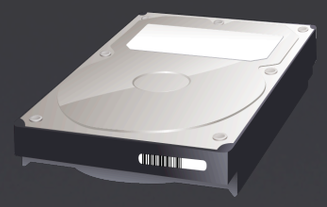
Brad Neuberg, Gears



Agenda

- Gears
- PubTools Search
- Gears Modules
- Search Architecture
- Dojo
- Tips, Tricks, and Code

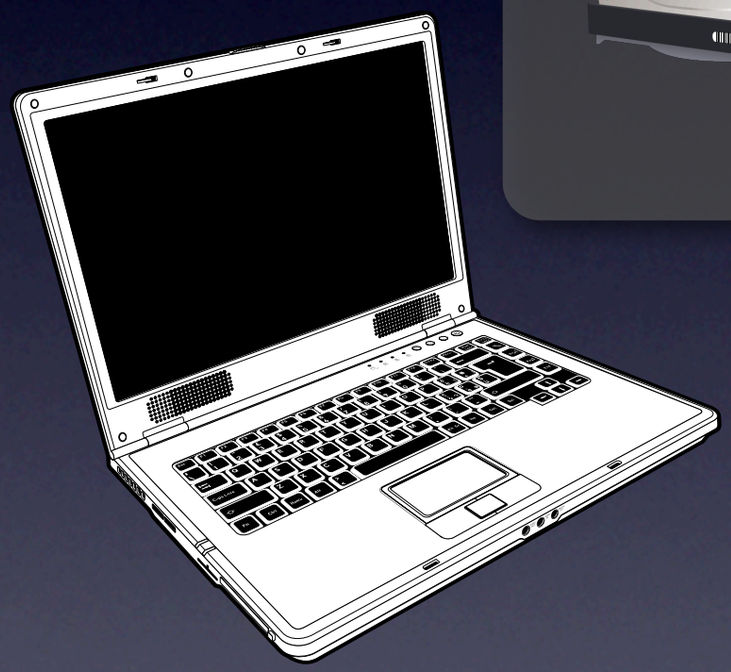




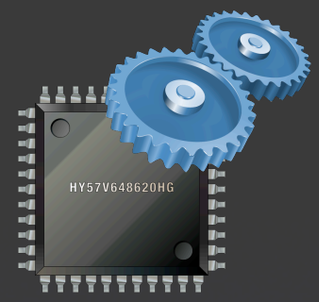
JavaScript

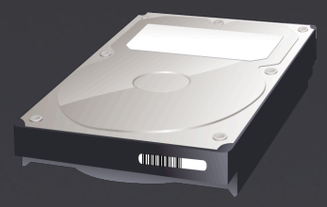
CSS

HTML



What is Gears?

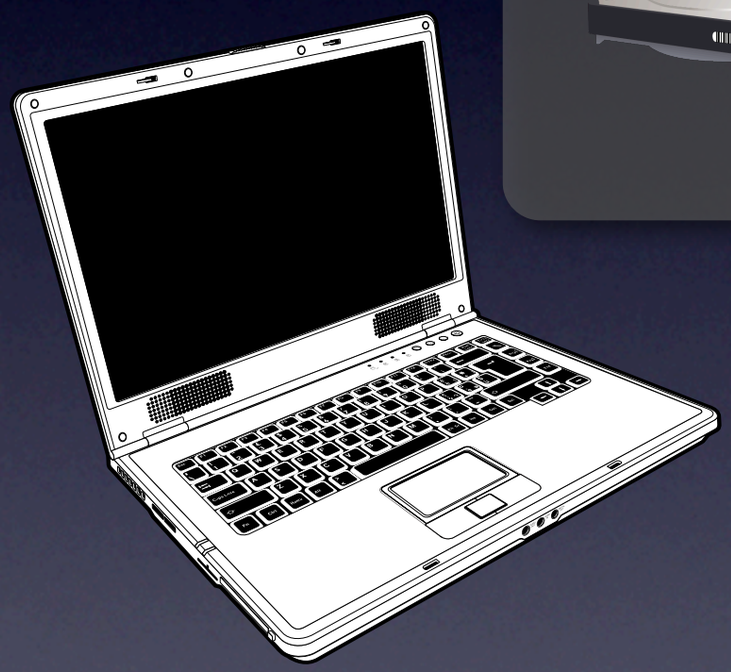




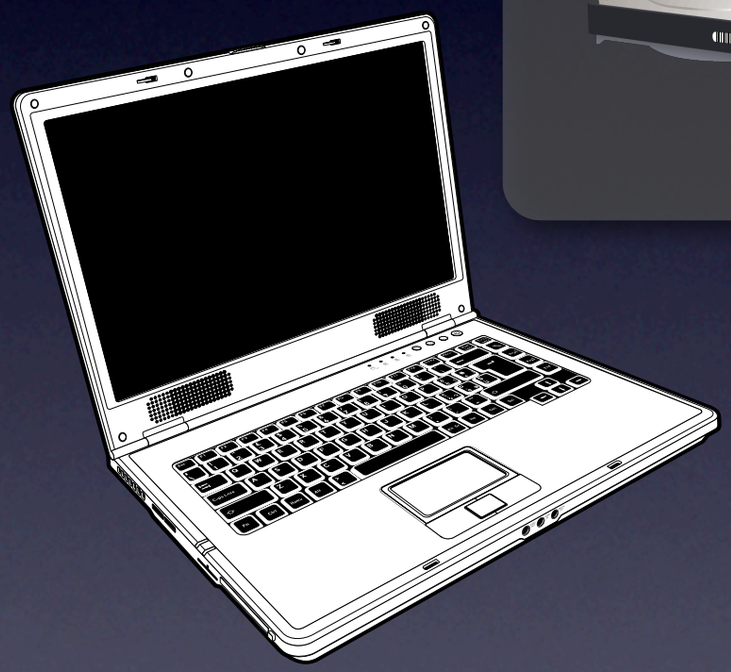
JavaScript

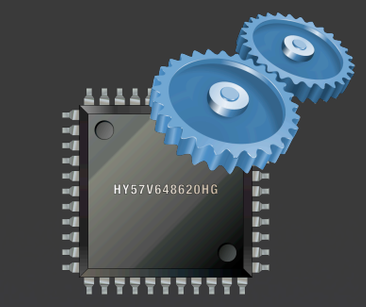
CSS

HTML



What is Gears?



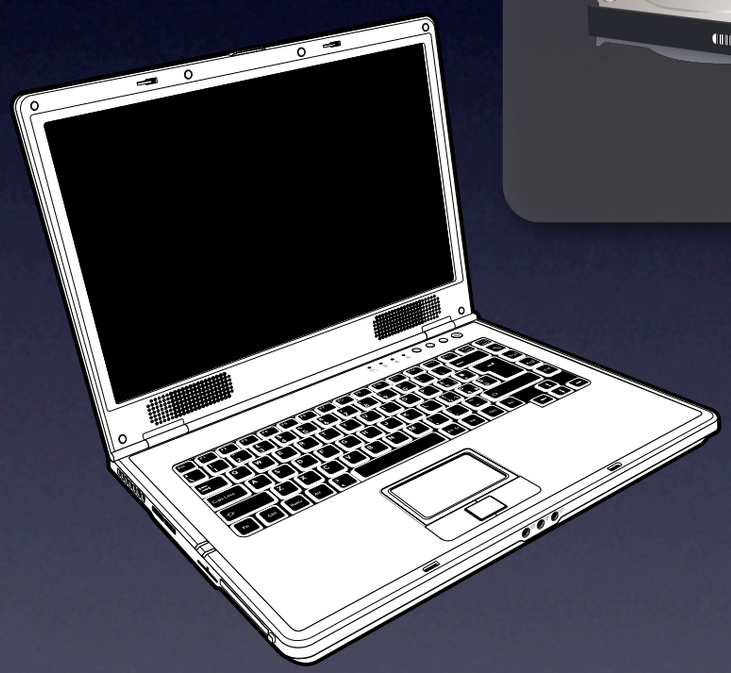


JavaScript

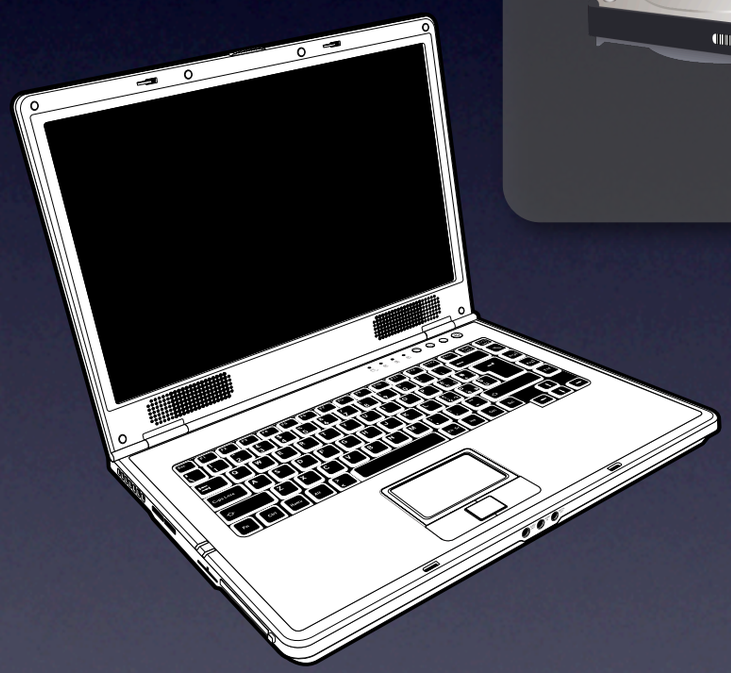
CSS

HTML

What is Gears?

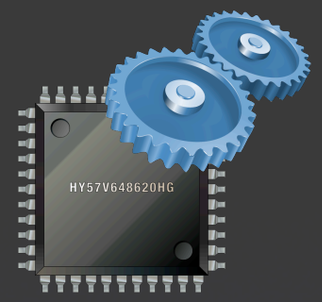


What is Gears?



HTML 5

What is Gears?

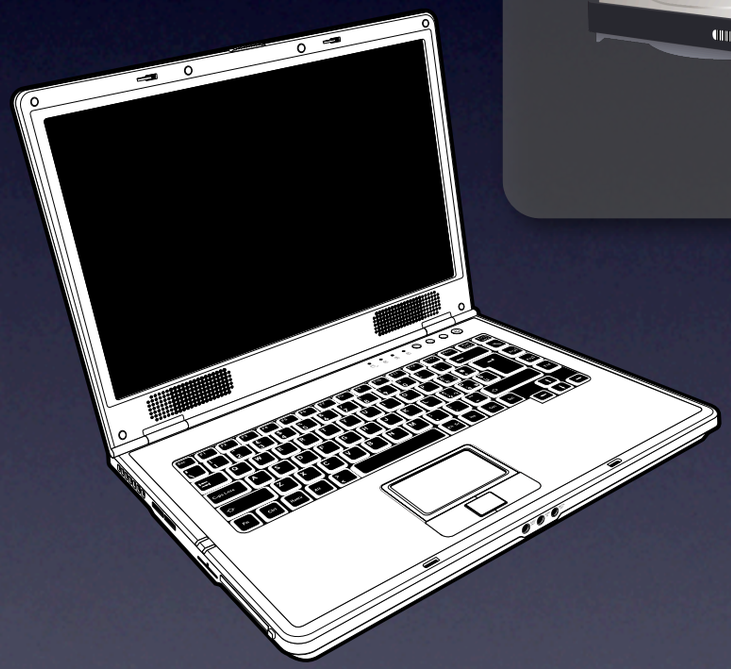


JavaScript

CSS

HTML

HTML 5



Open Source!

What is Gears?

JavaScript

CSS

HTML

Ajax++

Database

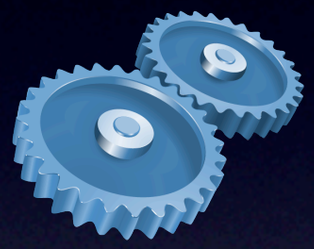
JavaScript

CSS

HTML

Ajax++

Database



Client-Side Search

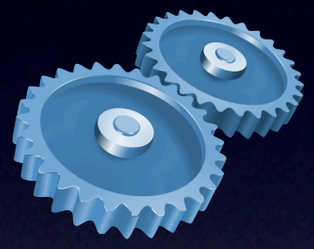
JavaScript

CSS

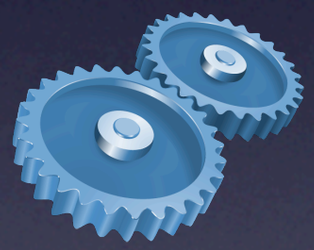
HTML

Ajax++

Database



Client-Side Search



Worker Pool

JavaScript

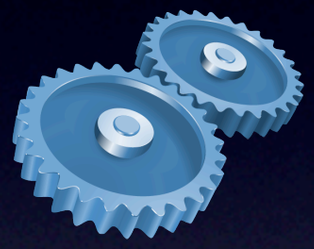
CSS

HTML

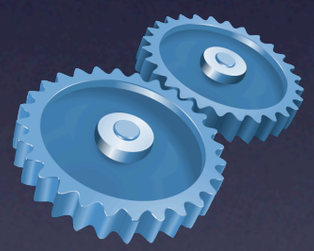
Ajax++

Database

X-Domain Mashups



Client-Side Search



Worker Pool

JavaScript

CSS

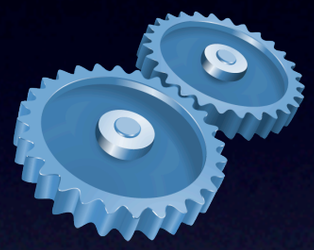
HTML

Ajax++

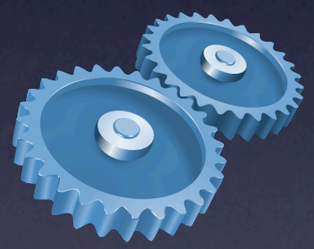
Database

X-Domain Mashups

Client-Side Search



Desktop API



Worker Pool

JavaScript

CSS

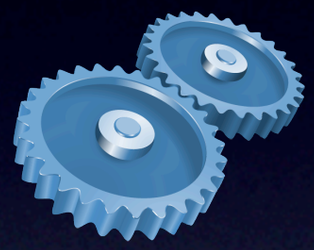
HTML

Ajax++

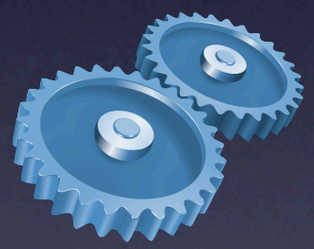
Database

X-Domain Mashups

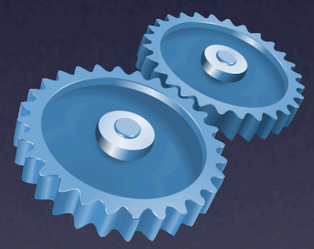
Client-Side Search



Desktop API



Worker Pool



Local Server

JavaScript

CSS

HTML

Ajax++

PubTools Search

- Not part of Gears
- Independent open source project
- Simple scripts
- Sprinkle HTML
- Client-side search engine

Demo

Why?

Why?

- Fast, Fast, Fast

Why?

- Fast, Fast, Fast
  - No network latency

Why?

- Fast, Fast, Fast
  - No network latency
  - No server

Why?

- Fast, Fast, Fast
  - No network latency
  - No server
  - Your documents

Why?

- Fast, Fast, Fast
  - No network latency
  - No server
  - Your documents
  - Simple

Why?

- Fast, Fast, Fast
  - No network latency
  - No server
  - Your documents
  - Simple
  - Good Gears example

How to Use

Create text file with URLs to index

search.txt:

```
version=0.1
resources/descartes.txt
resources/goethe.txt
resources/goldman.txt
resources/machiavelli.txt
resources/montaigne.txt
resources/kafka.html
resources/plato.html
```

How to Use

Pull in CSS and JavaScript:

```
<link rel="stylesheet" type="text/css"
      href="searchtools.css"></link>

<script src="pubtools-util.js"></script>
<script src="searchtools.js"></script>
```

How to Use

Point to search file:

```
<link rel="search.urls" href="search.txt"></link>
```

How to Use

Add DIV to HTML:

```
<div id="st-widget"></div>
```

How to Use

Update search file:

search.txt:

```
version=0.2
resources/descartes.txt
resources/goethe.txt
resources/goldman.txt
resources/machiavelli.txt
resources/montaigne.txt
resources/kafka.html
resources/plato.html
resources/woolf.html
```

Done!

Gears Modules

Database

Database

- Local SQL storage
- SQLite: Open source, mature, small (343K), fast
- Full-featured relational database
- Gigabytes of storage capacity
- Strict same-origin security model

Database Code

Database Code

```
var db = google.gears.factory.create('beta.database');
```

Database Code

```
var db = google.gears.factory.create('beta.database');

db.open('database-test');
```

Database Code

```
var db = google.gears.factory.create('beta.database');

db.open('database-test');

db.execute('CREATE TABLE IF NOT EXISTS Test' +
```

Database Code

```
var db = google.gears.factory.create('beta.database');

db.open('database-test');

db.execute('CREATE TABLE IF NOT EXISTS Test' +
           ' (Phrase TEXT, Timestamp INT)');
```

Database Code

```
var db = google.gears.factory.create('beta.database');

db.open('database-test');

db.execute('CREATE TABLE IF NOT EXISTS Test' +
           ' (Phrase TEXT, Timestamp INT)');

db.execute('INSERT INTO Test VALUES (?, ?)', ['Monkey!',
```

Database Code

```
var db = google.gears.factory.create('beta.database');

db.open('database-test');

db.execute('CREATE TABLE IF NOT EXISTS Test' +
           ' (Phrase TEXT, Timestamp INT)');

db.execute('INSERT INTO Test VALUES (?, ?)', ['Monkey!',
           new Date().getTime()]);
```

Database Code

Database Code

```
var rs;
```

Database Code

```
var rs;

try {
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
  }
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
  }
} finally {
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
  }
} finally {
  rs.close();
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
  }
} finally {
  rs.close();
  db.close();
```

Database Code

```
var rs;

try {
  rs = db.execute('SELECT * FROM Test ORDER BY Timestamp DESC');

  while (rs.isValidRow()) {
    console.log(rs.fieldByName(‘Phrase’) + '@'
                + rs.fieldByName(‘Timestamp’));
    rs.next();
  }
} finally {
  rs.close();
  db.close();
}
```

Full-Text Search

Full Text Search

- Gears added FTS2 to SQLite

- Create the database

  db.execute('CREATE VIRTUAL TABLE recipe USING fts2 (dish, ingredients)');

- Search the database

  db.execute('SELECT dish FROM recipe WHERE recipe MATCH ?', ['tomatoes']);

  Fun queries: *dish:stew tomatoes*

  Find rows with 'stew' in the dish field, and 'tomatoes' in any field.

Worker Pool

Worker Pool

JavaScript needs threads after all? Brendan!

Worker Pool Code

```
var pool = google.gears.factory.create('beta.workerpool');

pool.onmessage = function(message) {
  alert('next prime is: ' + message);
}

function nextPrime(n) {
  // TODO: Prime-finding algorithm goes here.
  google.gears.workerPool.sendMessage(result);
}

var runMe = String(nextPrime) + '; nextPrime()';

var worker = pool.createWorker(runMe);
```

Architecture

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

UI

constructor(id)
status(msg)
showResults(results)

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

UI

constructor(id)
status(msg)
showResults(results)

SearchManifest

constructor(url)

fetch_(url)
parse_()
process_()
getDBVersion_()
putDBVersion_(version)
getFinishedIndexing_()

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

UI

constructor(id)
status(msg)
showResults(results)

SearchManifest

constructor(url)

fetch_(url)
parse_()
process_()
getDBVersion_()
putDBVersion_(version)
getFinishedIndexing_()

Documents

constructor(urls)

filter_(urls, callback)
download_(urls)

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

UI

constructor(id)
status(msg)
showResults(results)

SearchManifest

constructor(url)

fetch_(url)
parse_()
process_()
getDBVersion_()
putDBVersion_(version)
getFinishedIndexing_()

Documents

constructor(urls)

filter_(urls, callback)
download_(urls)

Indexer

constructor(numDocs)
index(url, mimeType, doc)

indexWorker_(message)
getTitle_(url, mimeType,
doc)

SearchTools

search(query)
handleError(err)

ready_()
getDatabaseName_()
createDatabaseTables_()

UI

constructor(id)
status(msg)
showResults(results)

SearchManifest

constructor(url)

fetch_(url)
parse_()
process_()
getDBVersion_()
putDBVersion_(version)
getFinishedIndexing_()

Documents

constructor(urls)

filter_(urls, callback)
download_(urls)

Indexer

constructor(numDocs)
index(url, mimeType, doc)

indexWorker_(message)
getTitle_(url, mimeType,
doc)

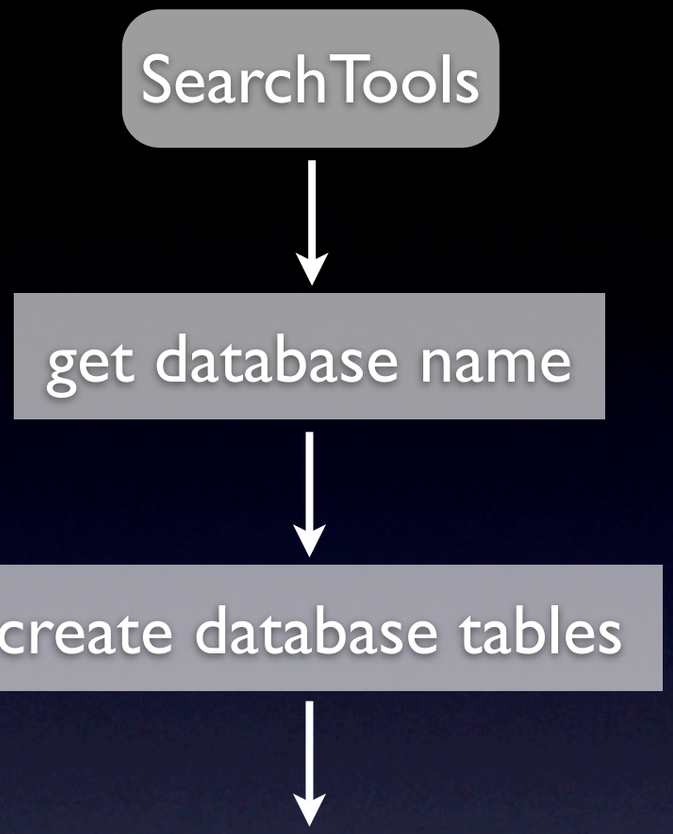
Searcher

search(query, callback)

getSnippet_(query,
mimeType,
content)
escapeString_(str)

Indexing

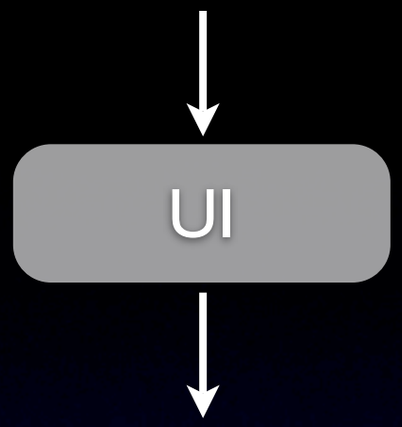
Indexing



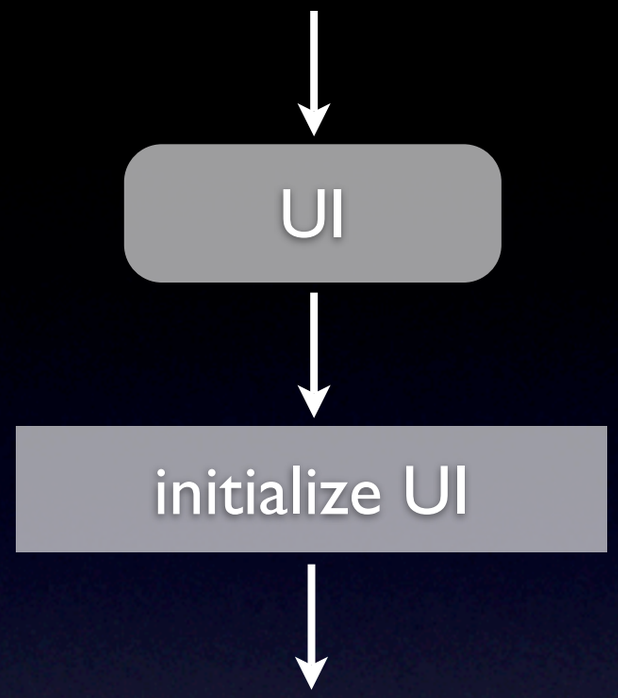
Indexing


SearchTools
get database name
create database tables
wait for page to finish loading
Async!

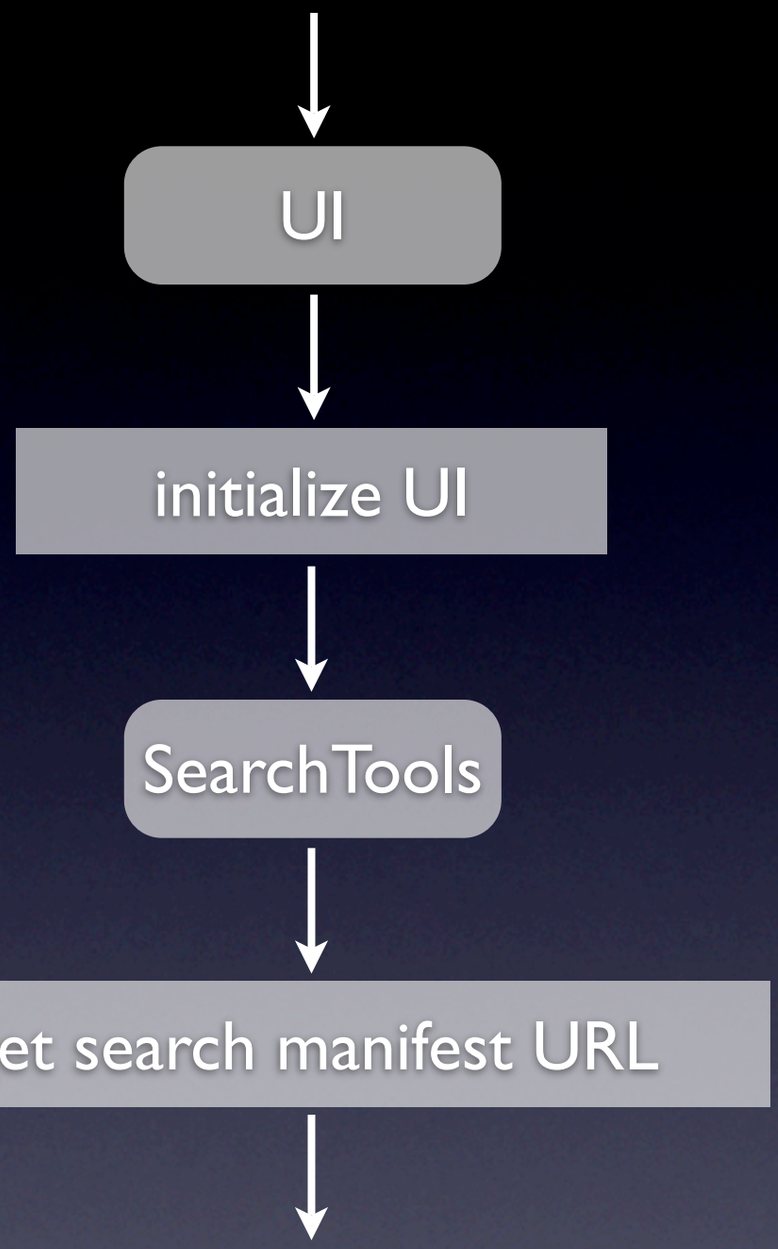
Indexing



Indexing



Indexing

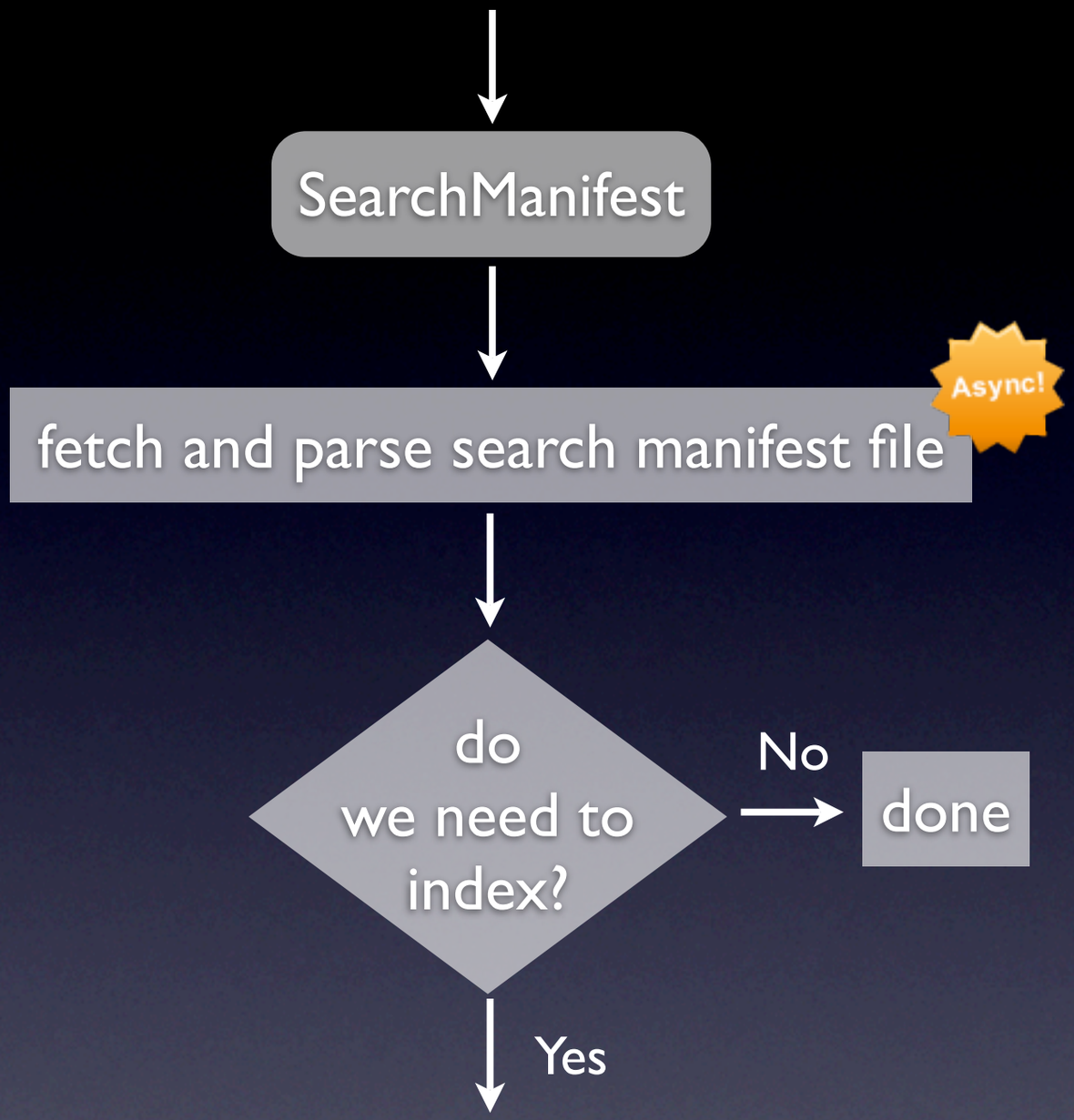


Indexing

Indexing

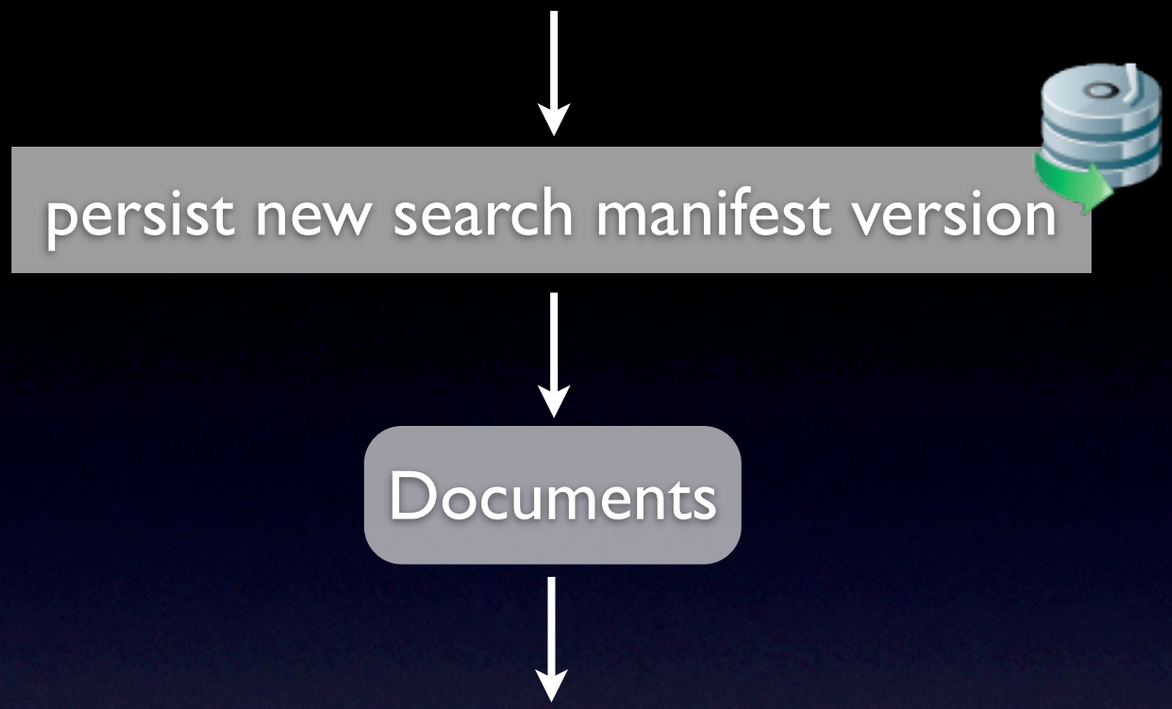

SearchManifest
fetch and parse search manifest file
Async!

Indexing


SearchManifest
fetch and parse search manifest file
Async!
do
we need to
index?
No
done
Yes

Indexing

Indexing



Indexing

persist new search manifest version

Documents

filter URLs by mime type

Async!

Indexing

persist new search manifest version

Documents

filter URLs by mime type

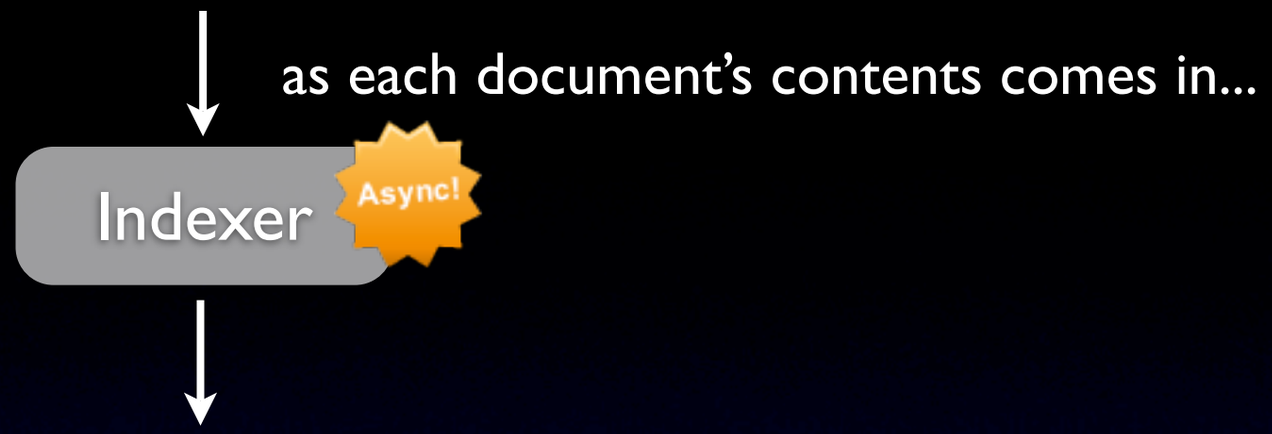
Async!

for HTML, XML, and text documents...

download documents

Async!

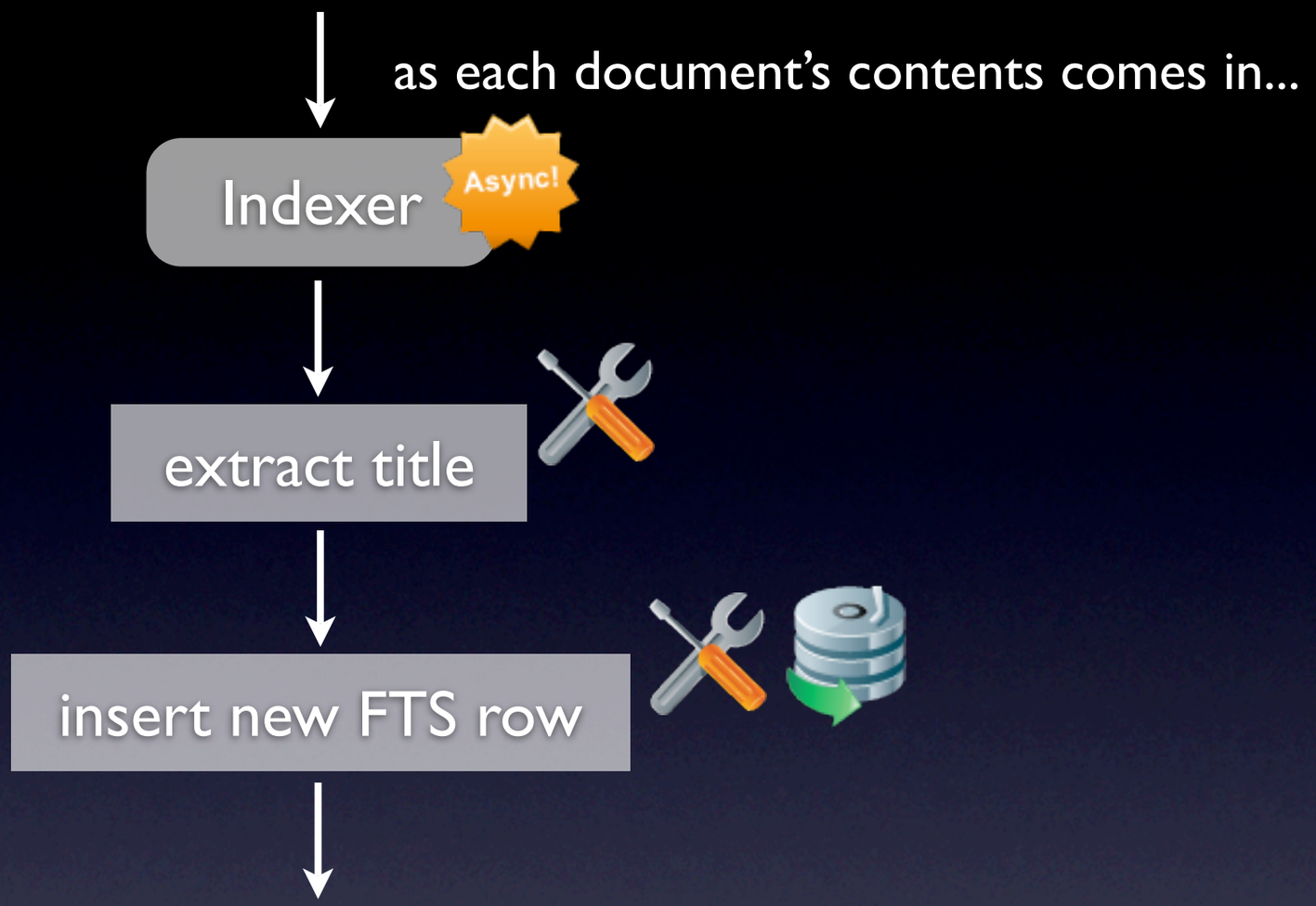
Indexing



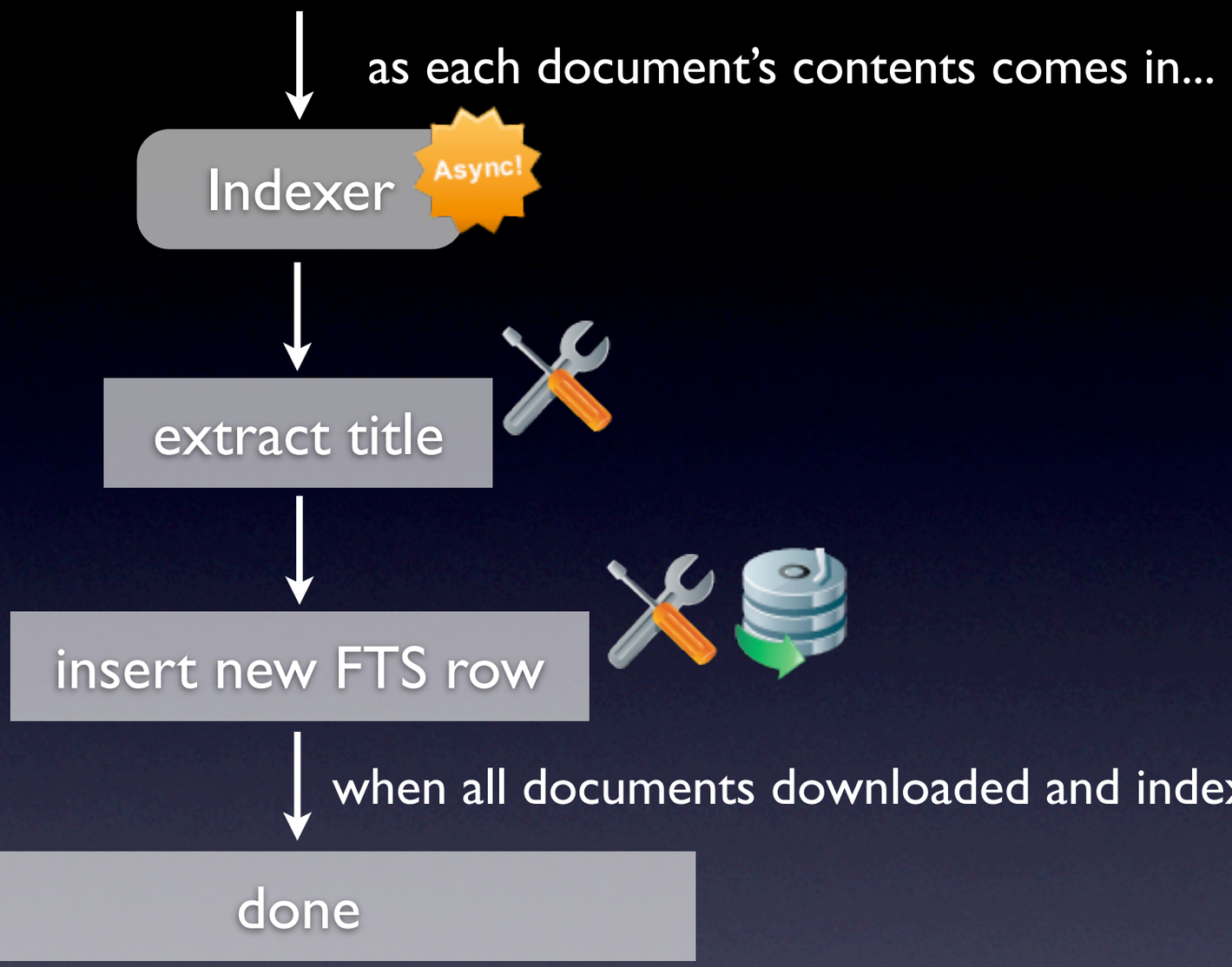
Indexing


as each document's contents comes in...
Indexer
Async!
extract title

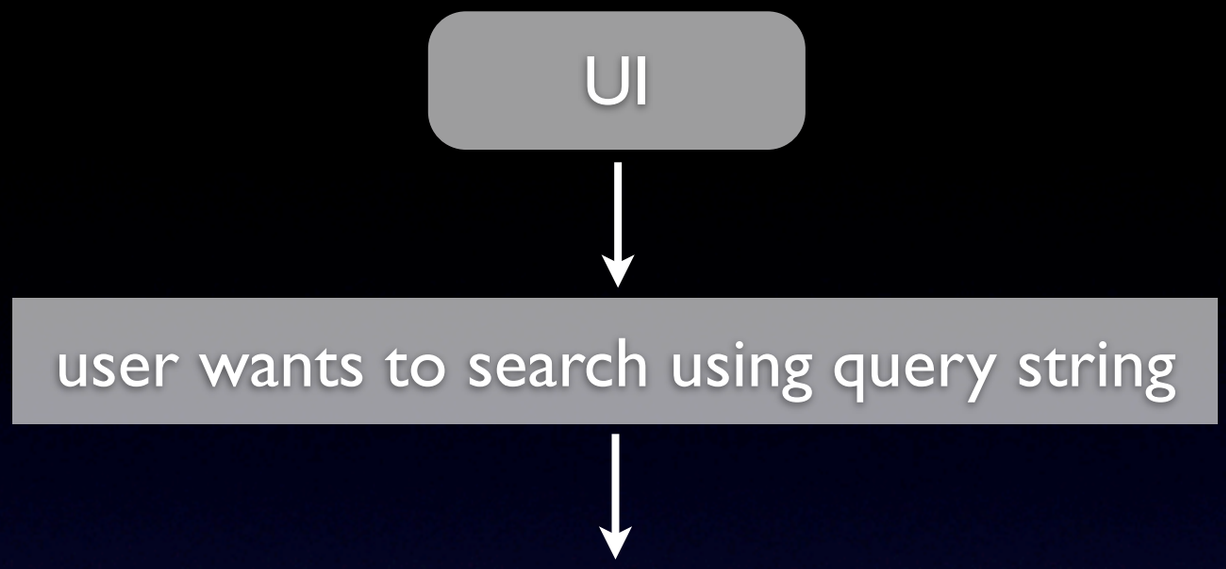
Indexing


as each document's contents comes in...
Indexer
Async!
extract title
insert new FTS row

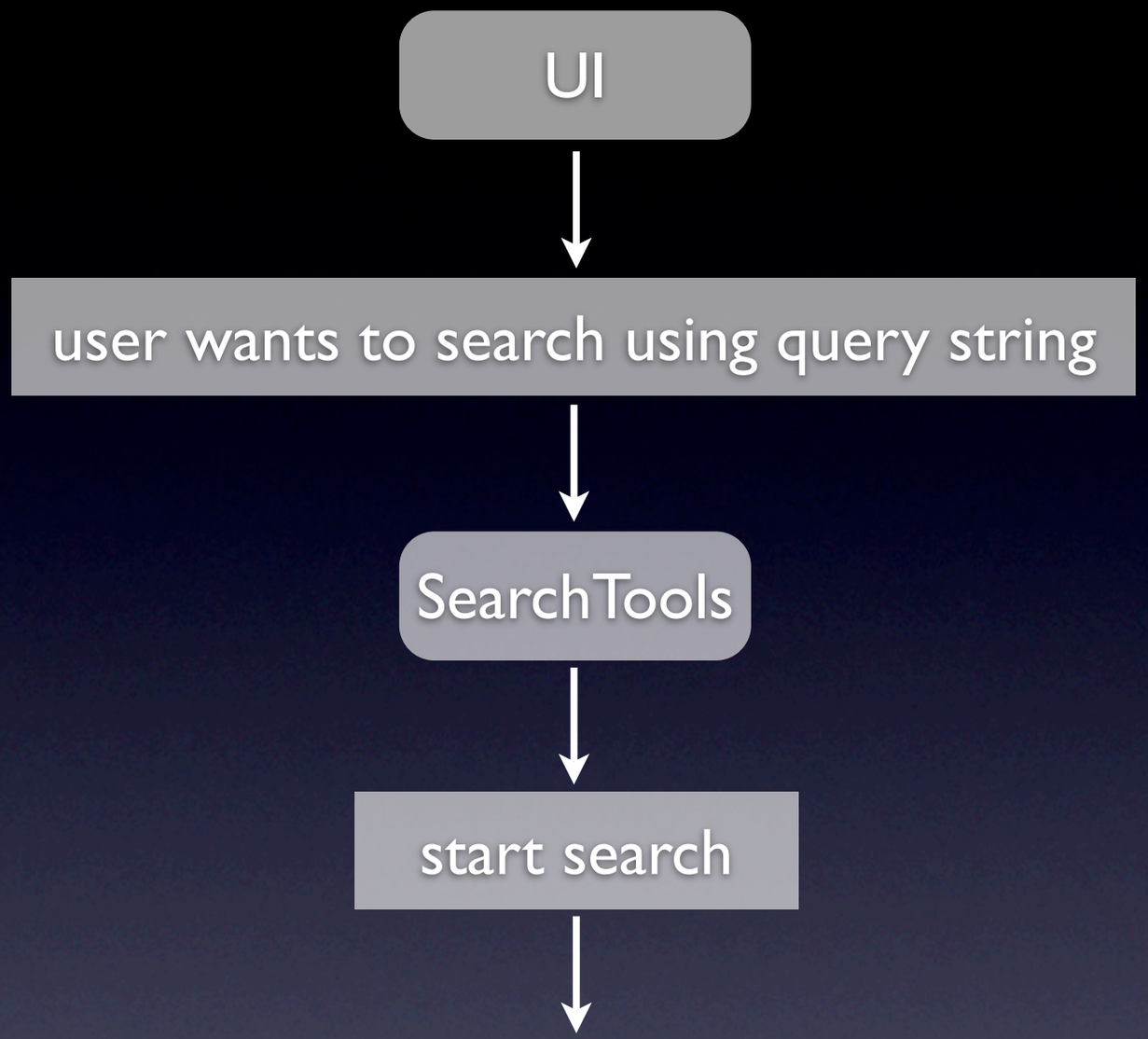
Indexing


as each document's contents comes in...
Indexer
Async!
extract title
insert new FTS row
when all documents downloaded and indexed....
done

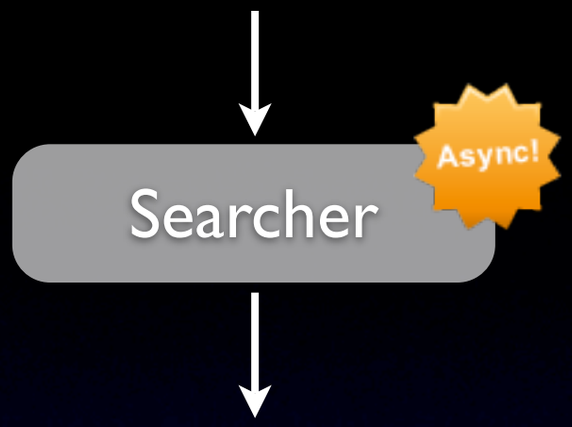
Indexing



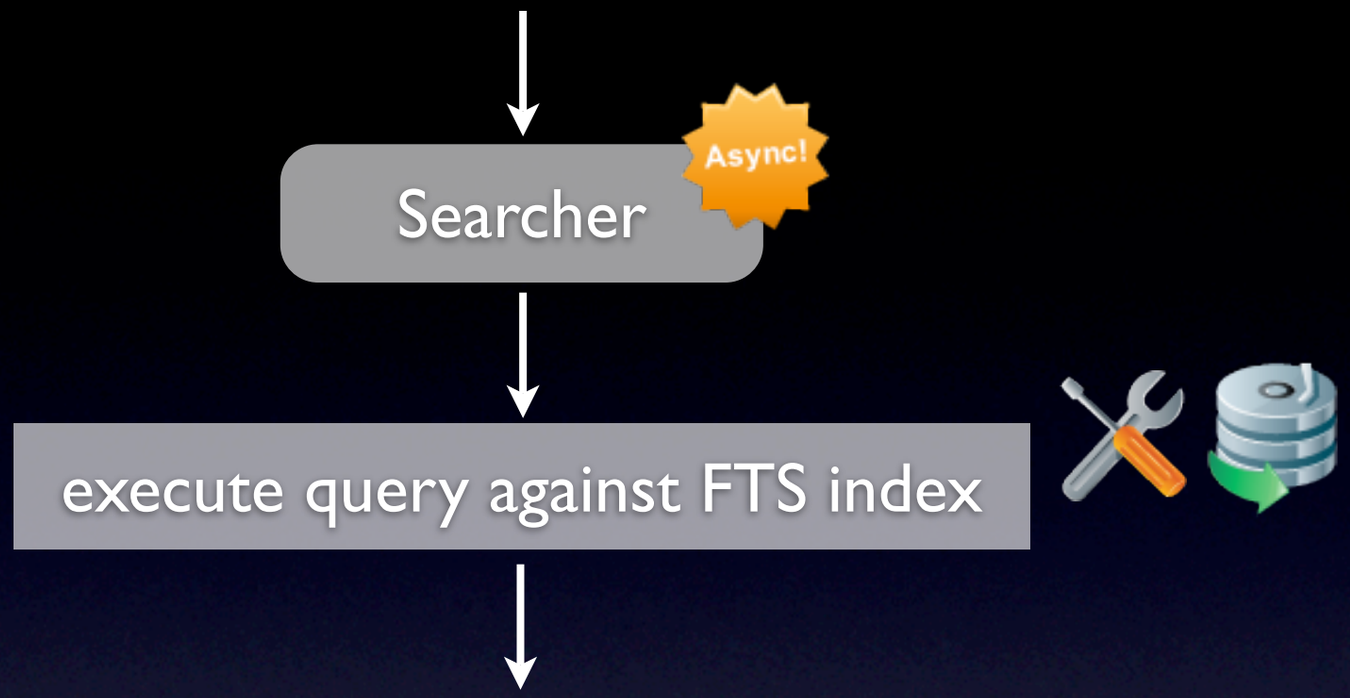
Searching


UI
user wants to search using query string
SearchTools
start search

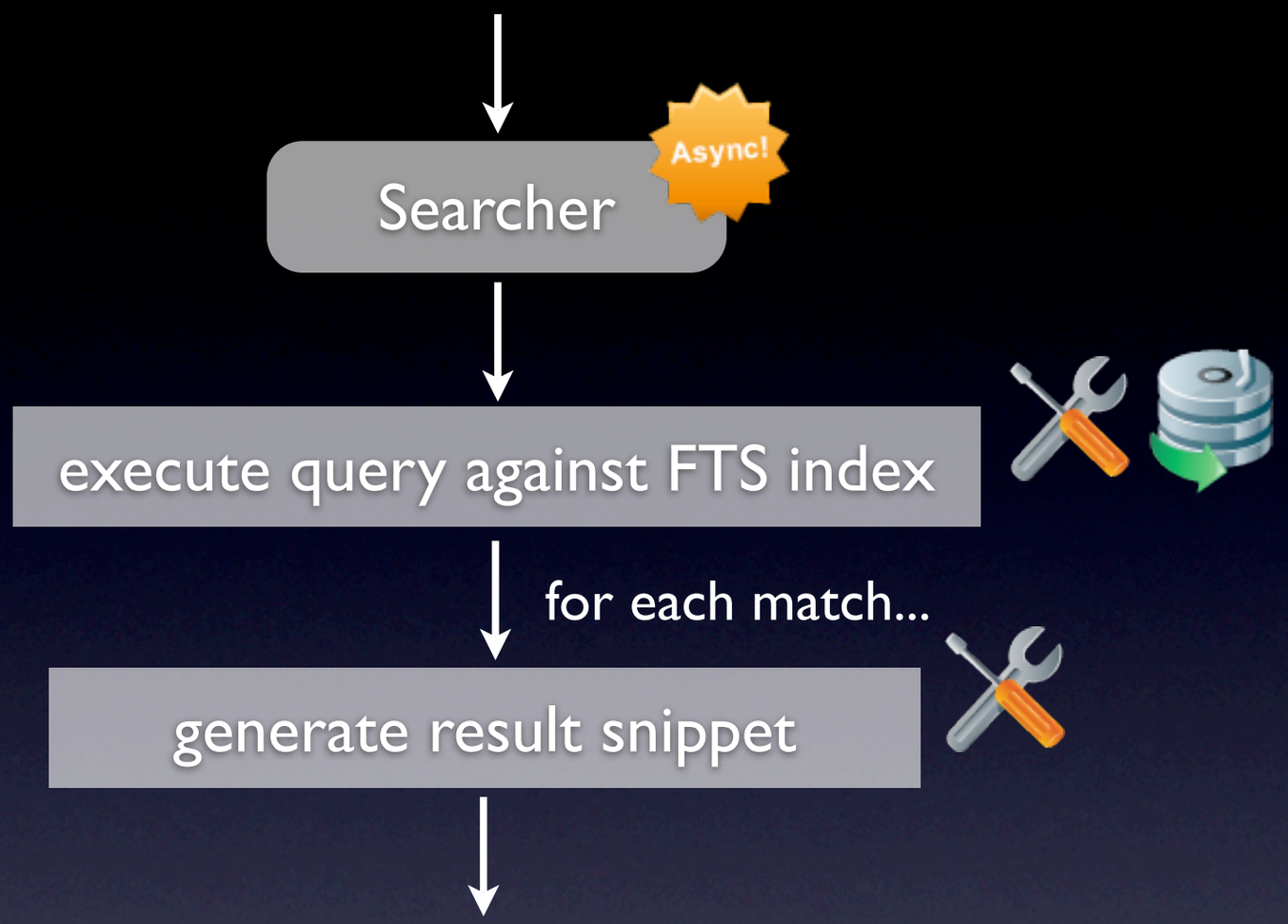
Searching



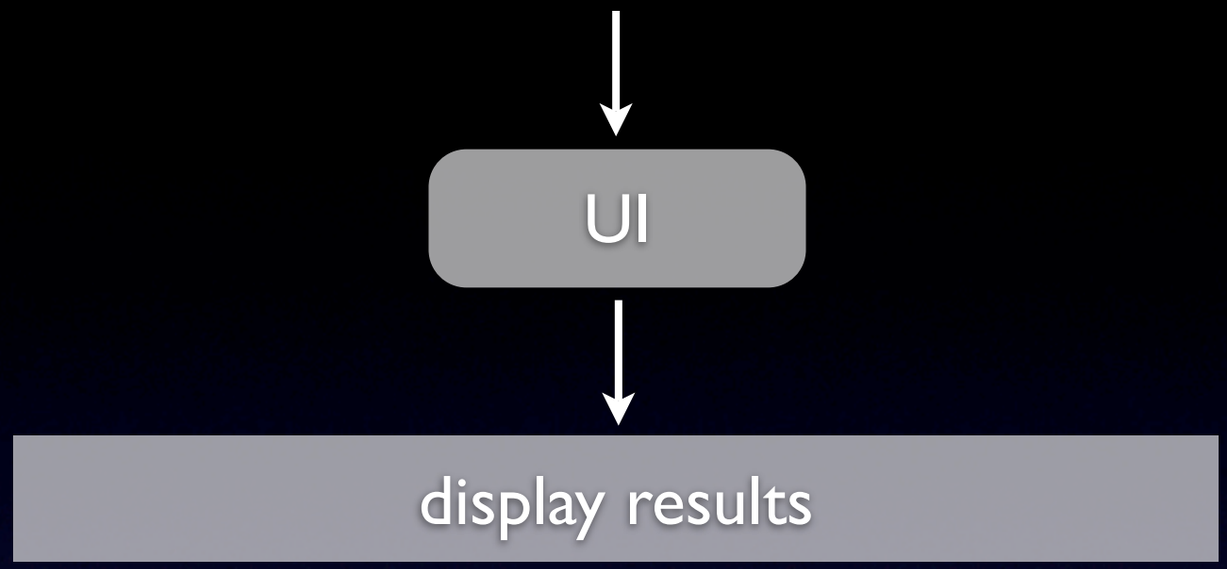
Searching



Searching


Searcher
Async!
execute query against FTS index
for each match...
generate result snippet

Searching



Searching

dōjō

dojo

dojo

dojo

dojo

dōjō

dōjō

- Open source

dojo

- Open source
- http://dojotoolkit.org

dōjō

- Open source
- http://dojotoolkit.org

- Three pieces

dōjō

- Open source
- http://dojotoolkit.org
- Three pieces
  - core - 24K gzipped

dōjō

- Open source
- http://dojotoolkit.org

- Three pieces
  - core - 24K gzipped
  - dijit

dōjō

- Open source
- http://dojotoolkit.org

- Three pieces
  - core - 24K gzipped
  - dijit
  - dojoX

dōjō

dōjō

- Includes build system

dōjō

- Includes build system
- Special build for PubTools

dōjō

- Includes build system
  - Special build for PubTools
    - Even smaller Dojo Core

dōjō

- Includes build system
  - Special build for PubTools
    - Even smaller Dojo Core
    - pubtools-util.js

dōjō

dōjō

- Re-namespaced ‘dojo’ element

dōjō

- Re-namespaced ‘dojo’ element
- ‘pu’

dōjō

- Re-namespaced ‘dojo’ element
- ‘pu’
- No collisions

dōjō

- Re-namespaced ‘dojo’ element
- ‘pu’
- No collisions
- Example:

dōjō

- Re-namespaced ‘dojo’ element
- ‘pu’
- No collisions
- Example:
  - dojo.hitch()

dōjō

- Re-namespaced ‘dojo’ element
- ‘pu’
- No collisions
- Example:
  - dojo.hitch()
  - pu.hitch()

XHR

dojo

```
download_: function(downloadMe) {
      var idx = new Indexer(downloadMe.length);

      pu.forEach(downloadMe, function(entry) {
        var url = entry.url;
        var mimeType = entry.mimeType;

        pu.xhrGet({
          url: url,

          load: function(data) {
            idx.index(url, mimeType, data);
          },

          error: function(err) {
            searchTools.handleError(err);
          }
        });
      });
  }
```

Declare

dōjō

```
pu.declare('Searcher', null, {
    search: function(query, callback) {
    },

    escapeString_: function(str) {
    },

    getSnippet_: function(query, mimeType, content) {
    }
});
```

Hitch

dōjō

```
pu.declare('Documents', null, {
    constructor: function(urls) {
      this.filter_(urls, pu.hitch(this, this.download_));
    },

    filter_: function(urls, callback) {
      // filter the URLs here asynchronously
      // ...
      callback(filteredURLs)
    },

    download_: function(downloadMe) {
      this.doSomething_();
    }
});
```

Query and byId

dojo

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
```

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
searchTools.search(query);
```

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
searchTools.search(query);
```

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
searchTools.search(query);

// ...
```

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
searchTools.search(query);

// ...
```

Query and byId

dojo

```
var query = pu.query('.gsc-input')[0].value;
searchTools.search(query);

// ...

var w = pu.byId('st-widget');
```

Tips & Tricks

- Do intensive operations on Workers!
  - Database
  - Network
  - Searching FTS

Tips & Tricks

- Creating database name:
  - Use window.location
  - Remove invalid characters
  - Add app specific label
  - Trim to length 64

Tips & Tricks

- Stringify workers:

```
index: function(url, mimeType, doc) {
      workerScript =
                'var getTitle = '
                + String(this.getTitle_) + '; '
                + 'google.gears.workerPool.onmessage = '
                + String(this.indexWorker_);
      // ...
},

indexWorker_: function(a, b, message) {
},

getTitle: function(url, mimeType, doc) {
}
```

Tips & Tricks

- Use JSON with workers (but NOT x-domain workers):

```
var childWorkerId = worker.createWorker(workerScript);

var msg = {url: url, mimeType: mimeType, doc: doc, dbName:
dbName};

msg = pu.toJson(msg);

worker.sendMessage(msg, childWorkerId);
```

Tips & Tricks

- Use a closure for encapsulation:

```
var searchTools = function() { // top of file
  pu.declare('SearchTools', null, { /* ... */ });
  pu.declare('UI', null, { /* ... */ });
  pu.declare('SearchManifest', null, { /* ... */ });
  pu.declare('Documents', null, { /* ... */ });
  pu.declare('Indexer', null, { /* ... */ });
  pu.declare('Searcher', null, { /* ... */ });

  searchTools = new SearchTools();
  return searchTools;
}(); // bottom of file
```

createDatabaseTables

createDatabaseTables

```
createDatabaseTables_: function() {
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

  try {
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

  try {
    db.execute('CREATE VIRTUAL TABLE ClientSearch '
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

  try {
    db.execute('CREATE VIRTUAL TABLE ClientSearch '
                    + 'USING fts2(url, title, mimeType, content)');
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

  try {
    db.execute('CREATE VIRTUAL TABLE ClientSearch '
                    + 'USING fts2(url, title, mimeType, content)');
  } catch (e) {} // just ignore if it exists
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

    try {
      db.execute('CREATE VIRTUAL TABLE ClientSearch '
                 + 'USING fts2(url, title, mimeType, content)');
    } catch (e) {} // just ignore if it exists
},
```

createDatabaseTables

```
createDatabaseTables_: function() {
    db = google.gears.factory.create('beta.database');
    db.open(dbName);

    db.execute('CREATE TABLE IF NOT EXISTS ClientMetadata '
               + '(version VARCHAR UNIQUE, '
               + ' schemaVersion VARCHAR, '
               + ' finishedIndexing INTEGER)');

    try {
      db.execute('CREATE VIRTUAL TABLE ClientSearch '
                 + 'USING fts2(url, title, mimeType, content)');
    } catch (e) {} // just ignore if it exists
},
```

indexWorker

indexWorker

```
indexWorker_: function(a, b, message) {
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }

    google.gears.workerPool.sendMessage('indexed:::' + args.url,
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }

    google.gears.workerPool.sendMessage('indexed:::' + args.url,
                                        message.sender);
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }

    google.gears.workerPool.sendMessage('indexed:::' + args.url,
                                        message.sender);
}
```

indexWorker

```
indexWorker_: function(a, b, message) {
    var args = eval('(' + message.text + ')');

    // extract metadata
    var title = getTitle(args.url, args.mimeType, args.doc);

    var db = google.gears.factory.create('beta.database');
    db.open(args.dbName);
    try {
      db.execute('INSERT INTO ClientSearch '
                 + '(url, title, mimeType, content) '
                 + 'VALUES (?, ?, ?, ?)',
                 [args.url, title, args.mimeType, args.doc]);
    } finally {
      db.close();
    }

    google.gears.workerPool.sendMessage('indexed:::' + args.url,
                                        message.sender);
}
```

search

search

```
search: function(query, callback) {
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];

        var db = google.gears.factory.create('beta.database');
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];

        var db = google.gears.factory.create('beta.database');
        db.open(dbName);
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];

        var db = google.gears.factory.create('beta.database');
        db.open(dbName);
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];

        var db = google.gears.factory.create('beta.database');
        db.open(dbName);

        var rs = db.execute('SELECT * FROM ClientSearch WHERE '
```

search

```
search: function(query, callback) {
      var workerScript = function(a, b, message) {
        var params = message.text.split(":::");
        var query = params[0];
        var dbName = params[1];

        var db = google.gears.factory.create('beta.database');
        db.open(dbName);

        var rs = db.execute('SELECT * FROM ClientSearch WHERE '
                            + 'content MATCH ?', [query]);
```

search

search

```
results = '[';
```

search

```
results = '[';
while (rs.isValidRow()) {
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
    + '"snippet": ' + escapeString(entry.snippet) + ', '
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
    + '"snippet": ' + escapeString(entry.snippet) + ', '
    + '"href": "' + entry.href + '"'
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
    + '"snippet": ' + escapeString(entry.snippet) + ', '
    + '"href": "' + entry.href + '"'
    + '}';
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
    + '"snippet": ' + escapeString(entry.snippet) + ', '
    + '"href": "' + entry.href + '"'
    + '}';
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);
```

```
var entryStr =
    '{'
    + '"title": ' + escapeString(entry.title) + ', '
    + '"snippet": ' + escapeString(entry.snippet) + ', '
    + '"href": "' + entry.href + '"'
    + '}';

results += entryStr;
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);

  var entryStr =
      '{'
      + '"title": ' + escapeString(entry.title) + ', '
      + '"snippet": ' + escapeString(entry.snippet) + ', '
      + '"href": "' + entry.href + '"'
      + '}';

  results += entryStr;
  rs.next();
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);

  var entryStr =
      '{'
      + '"title": ' + escapeString(entry.title) + ', '
      + '"snippet": ' + escapeString(entry.snippet) + ', '
      + '"href": "' + entry.href + '"'
      + '}';

  results += entryStr;
  rs.next();
}
```

search

```
results = '[';
while (rs.isValidRow()) {
  var entry = {};
  entry.title = rs.fieldByName('title');
  entry.href = rs.fieldByName('url');
  var mimeType = rs.fieldByName('mimeType');
  var content = rs.fieldByName('content');
  entry.snippet = getSnippet(query, mimeType, content);

  var entryStr =
      '{'
      + '"title": ' + escapeString(entry.title) + ', '
      + '"snippet": ' + escapeString(entry.snippet) + ', '
      + '"href": "' + entry.href + '"'
      + '}';

  results += entryStr;
  rs.next();
}
google.gears.workerPool.sendMessage(results, message.sender);
```

Status

- PubTools Search still in development
- Not released yet

Creating a Client-Side Search Engine with Gears

Brad Neuberg, Gears