MarkLogic®

Unit 7: Search Application Fundamentals

Learning Objectives

- Describe the following search concepts:
  - Relevancy
  - Grammar
  - Stemming
  - Filtered vs. unfiltered search
  - Constraints
- Use the Node.js client API to:
  - Generate search response data
  - Build string, query by example (QBE), and structured queries.

MarkLogic®

Relevance Order

DOCUMENT 1

```
{
  "description":
  101 Dalmatians is
  a fun story about
  dogs escaping
  from Cruella
  Deville."
}
```

DOCUMENT 2

```
{
  "description":
  Fun, Fun, Fun by
  the Beach Boys.
}
```

DOCUMENT 3

```
{
  "description":
  Dogs are fun.
}
```

DOCUMENT 3

```
{
  "description":
  Raining cats and
  dogs.
}
```

Query: Find documents that contain the word “fun”.

Which documents match?
Which documents are the ***best*** matches?

Relevance Order

- The default score calculation:

Score = LOG (TF) X IDF

  - Score
    - A calculated value for each item that is returned in the search query result set
  - TF
    - Term Frequency
  - IDF
    - Inverse Document Frequency

Relevance Order

Score = LOG (TF) X IDF

- Term Frequency
  - How often a term occurs within a specific fragment (document) of the result set.
  - Normalized relative to total words in document: ***term density***.
  - Example: A search across the database for documents containing the word “dog”

Document #1
- “dog” occurs 10 times
- 100 total words in document
- Density 10%

Document #2
- “dog” occurs 100 times
- 10,000 total words in doc
- Density 1%

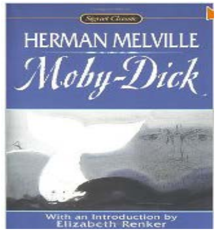
Relevance Order

Score = LOG (TF) X IDF

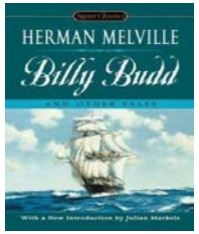
- Inverse Document Frequency
  - IDF = (1/DF)
    - DF = Document Frequency
  - Example: Search for “cat” OR “dog”
    - 100 documents contain “cat”. IDF = 1/100 = .01
    - 2 documents contain “dog”. IDF = 1/2 = .5
    - Assume a cat and dog document both have the same LOG(TF) value.
    - Which document would receive the highest overall score?

Impacting Relevance: Quality

- Quality is…
  - A factor to increase a documents relevancy score relative to other matching documents
  - Set upon ingestion (or modified later)
  - Default is 0
  - log(tf)/idf + (Quality Weight * Quality)



Q = 3



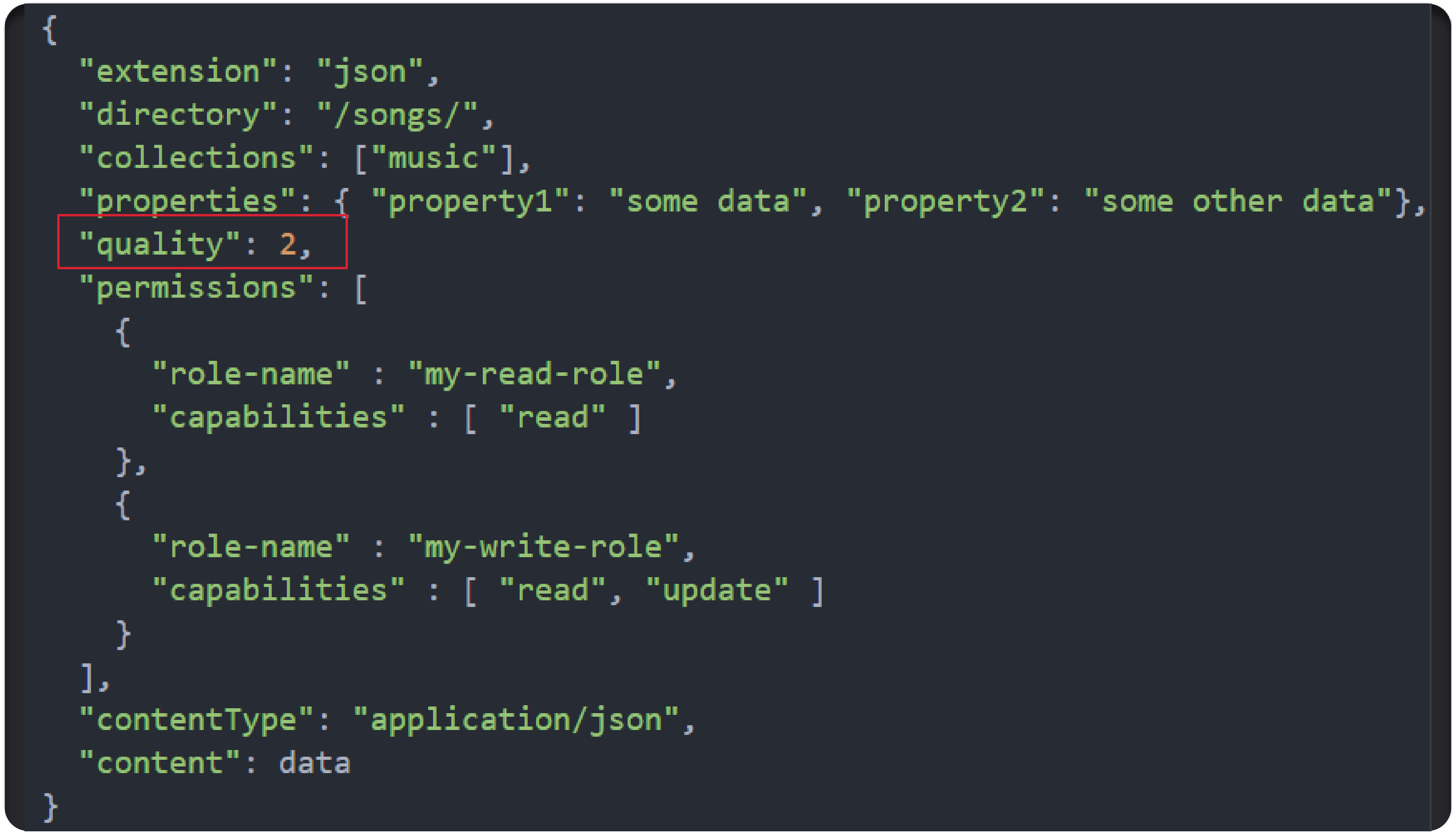
Q = 0

Find books written by the author Herman Melville.

1. Moby Dick
2. Billy Budd

Impacting Relevance: Quality

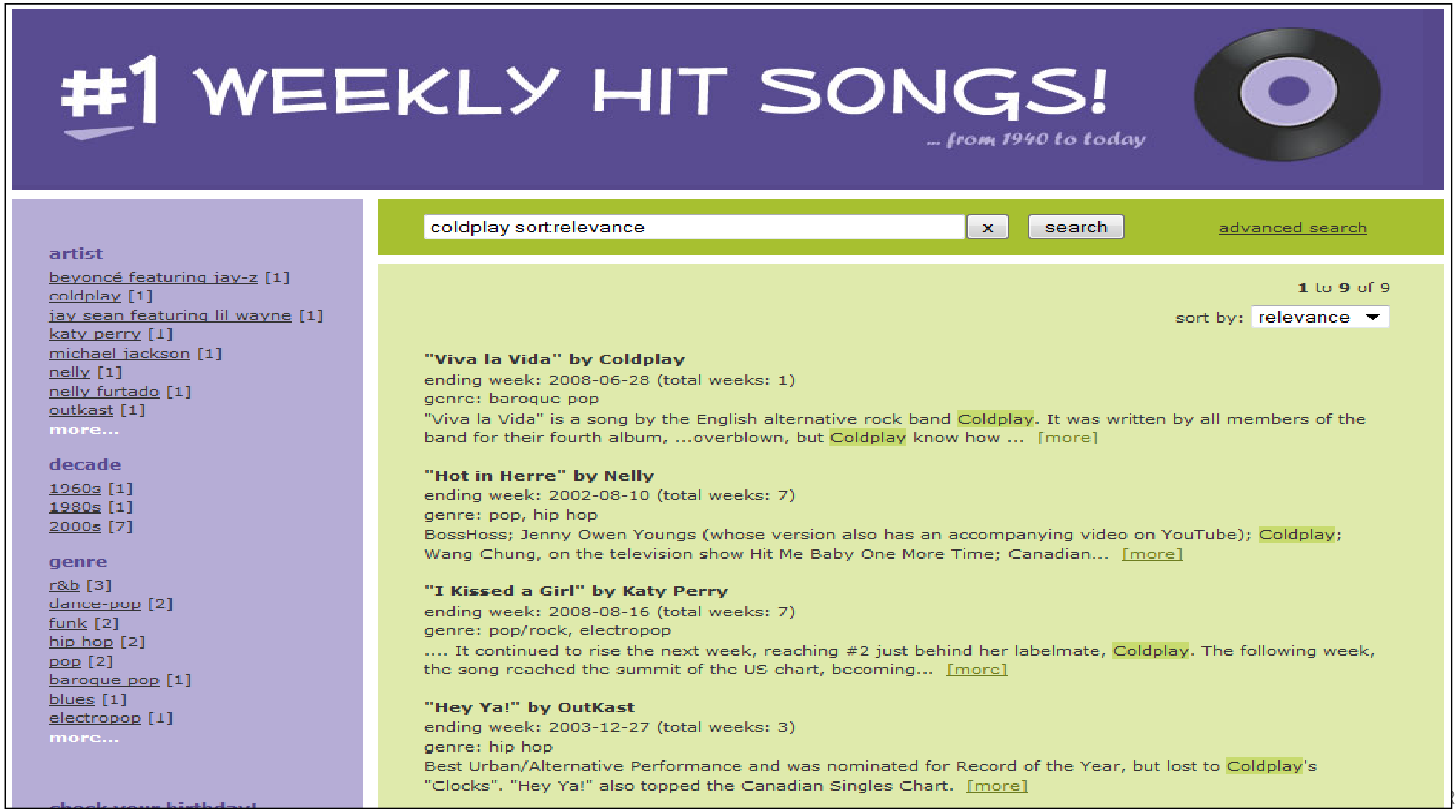
- Quality is document metadata that can be managed:



```
{
  "extension": "json",
  "directory": "/songs/",
  "collections": ["music"],
  "properties": { "property1": "some data", "property2": "some other data"},
  "quality": 2,
  "permissions": [
    {
      "role-name" : "my-read-role",
      "capabilities" : [ "read" ]
    },
    {
      "role-name" : "my-write-role",
      "capabilities" : [ "read", "update" ]
    }
  ],
  "contentType": "application/json",
  "content": data
}
```

Impacting Relevance: Word Query

- Why does the Coldplay song appear first?



coldplay sort:relevance

artist
beyoncé featuring jay-z [1]
coldplay [1]
jay sean featuring lil wayne [1]
katy perry [1]
michael jackson [1]
nelly [1]
nelly furtado [1]
outkast [1]
more...

decade
1960s [1]
1980s [1]
2000s [7]

genre
r&b [3]
dance-pop [2]
funk [2]
hip hop [2]
pop [2]
baroque pop [1]
blues [1]
electropop [1]
more...

check your birthday!

1 to **9** of 9

sort by: relevance

"Viva la Vida" by Coldplay
ending week: 2008-06-28 (total weeks: 1)
genre: baroque pop
"Viva la Vida" is a song by the English alternative rock band Coldplay. It was written by all members of the band for their fourth album, ...overblown, but Coldplay know how ... [more]

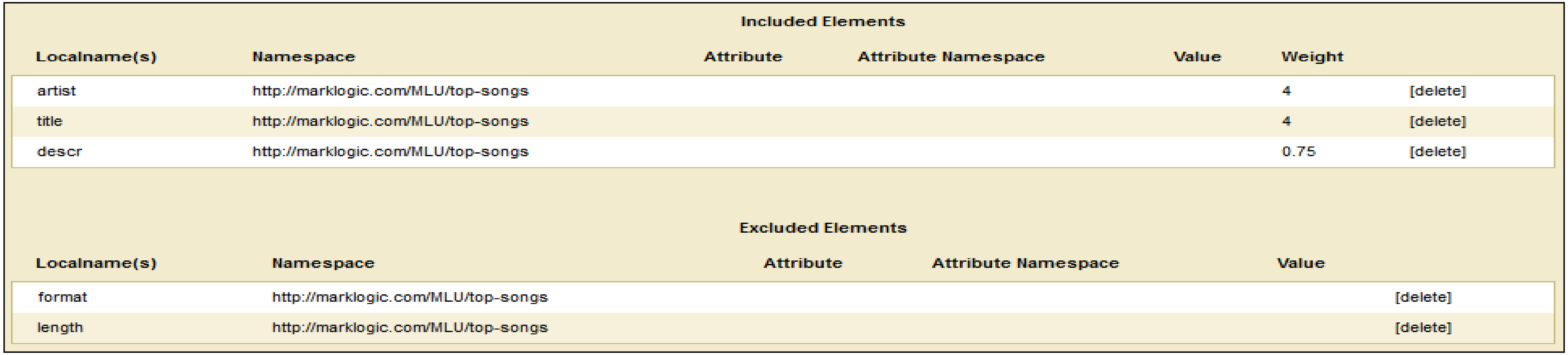
"Hot in Herre" by Nelly
ending week: 2002-08-10 (total weeks: 7)
genre: pop, hip hop
BossHoss; Jenny Owen Youngs (whose version also has an accompanying video on YouTube); Coldplay; Wang Chung, on the television show Hit Me Baby One More Time; Canadian... [more]

"I Kissed a Girl" by Katy Perry
ending week: 2008-08-16 (total weeks: 7)
genre: pop/rock, electropop
.... It continued to rise the next week, reaching #2 just behind her labelmate, Coldplay. The following week, the song reached the summit of the US chart, becoming... [more]

"Hey Ya!" by OutKast
ending week: 2003-12-27 (total weeks: 3)
genre: hip hop
Best Urban/Alternative Performance and was nominated for Record of the Year, but lost to Coldplay's "Clocks". "Hey Ya!" also topped the Canadian Singles Chart. [more]

Impacting Relevance: Word Query

- Why does the Coldplay song appear first?
  - Custom Word Query defined on the database adjusts weight based on where in the document the desired term was found.
  - Finding the word in the <artist> is weighted higher than the <descr>.



Included Elements

Localname(s)	Namespace	Attribute	Attribute Namespace	Value	Weight	
artist	http://marklogic.com/MLU/top-songs				4	[delete]
title	http://marklogic.com/MLU/top-songs				4	[delete]
descr	http://marklogic.com/MLU/top-songs				0.75	[delete]

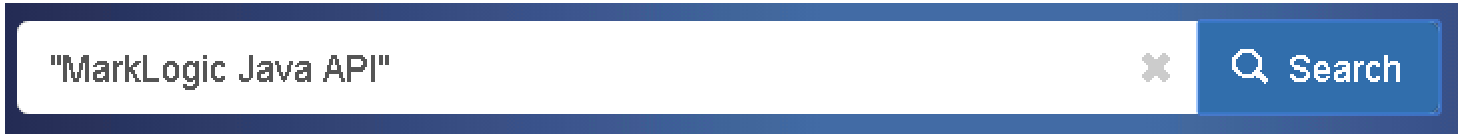
Excluded Elements

Localname(s)	Namespace	Attribute	Attribute Namespace	Value	
format	http://marklogic.com/MLU/top-songs				[delete]
length	http://marklogic.com/MLU/top-songs				[delete]

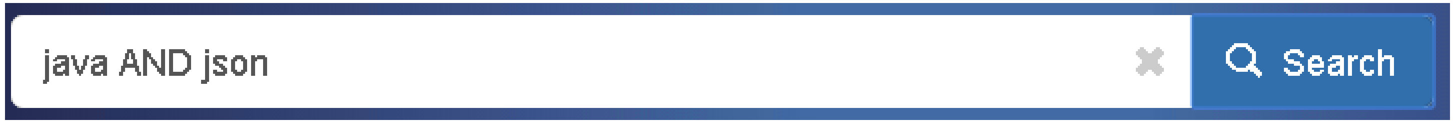
MarkLogic®

Search Grammar

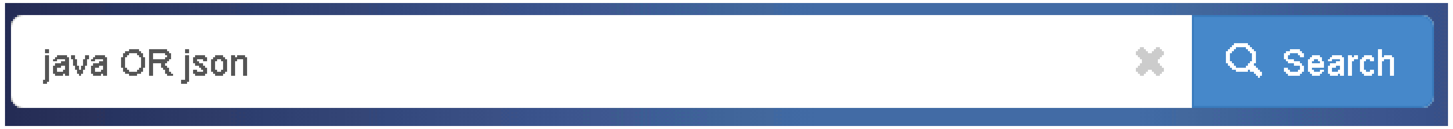
- “Phrase Search”


"MarkLogic Java API"

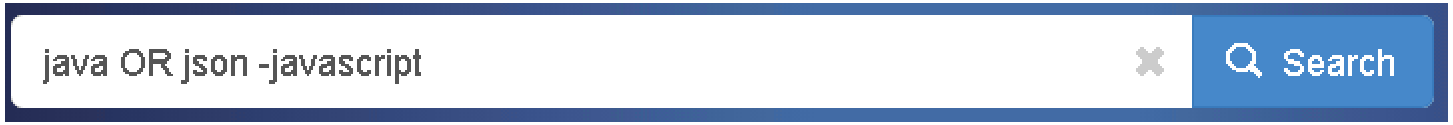
- AND


java AND json

- OR


java OR json

- NOT


java OR json -javascript

Stemming

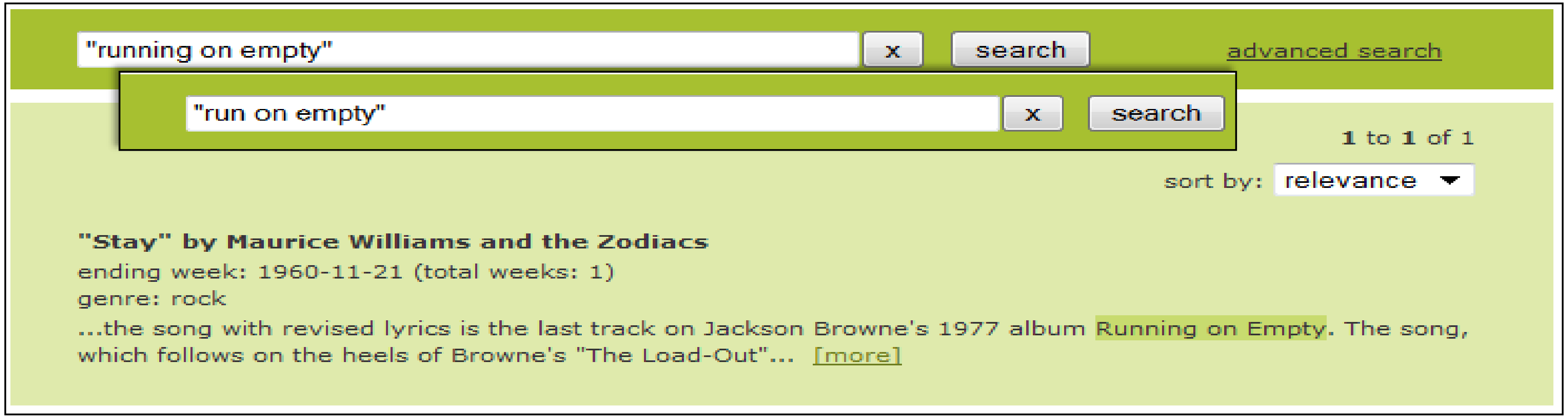
```
DOCUMENT 1
{
  "description":
  101 Dalmatians is
  a fun story about
  dogs escaping
  from Cruella
  Deville."
}
```

```
DOCUMENT 2
{
  "description":
  Dogs are fun.
}
```

Query	Matches	Why?
dogs AND escape	Doc1	dogs = dog, escaping = escape
“dog is fun”	Doc 1, Doc 2	dogs = dog, (is | are | be) = be, fun = fun
“dogs be awesome”	None	“awesome” does not stem to “fun”

Stemming

- Stemming rules take language and part of speech into account
  - Custom dictionaries and thesaurus can be implemented



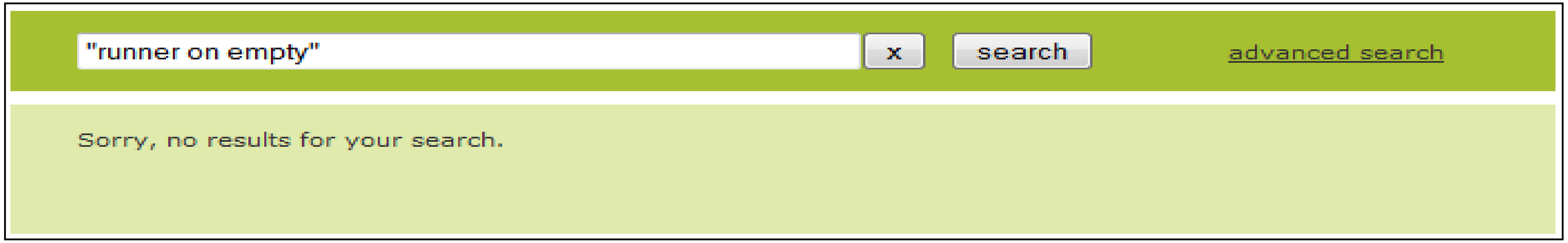
1 to 1 of 1

"Stay" by Maurice Williams and the Zodiacs

ending week: 1960-11-21 (total weeks: 1)

genre: rock

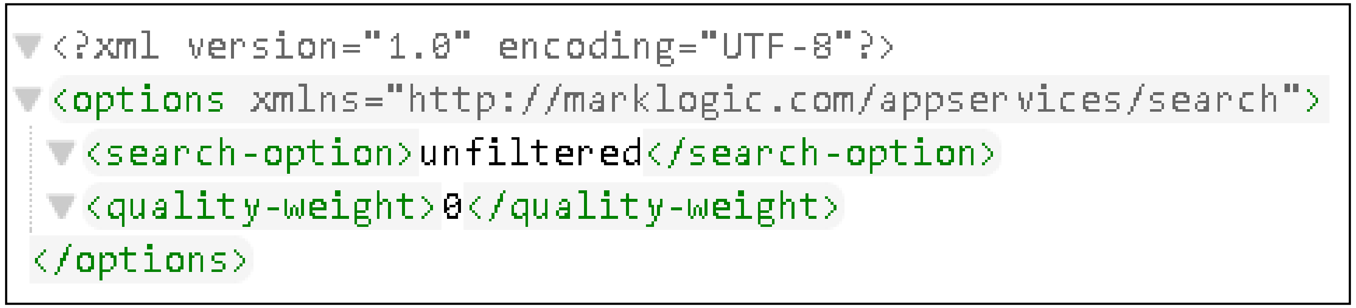
...the song with revised lyrics is the last track on Jackson Browne's 1977 album Running on Empty. The song, which follows on the heels of Browne's "The Load-Out"... [more]



Sorry, no results for your search.

Filtered vs. Unfiltered

- Default options used by Node.js / REST API is **UNFILTERED** search
- This can be seen if you explore the modules database of your REST instance.
- You will notice that the default configuration options specify unfiltered search:



```xml
<?xml version="1.0" encoding="UTF-8"?>
<options xmlns="http://marklogic.com/appservices/search">
  <search-option>unfiltered</search-option>
  <quality-weight>0</quality-weight>
</options>
```

Filtered vs. Unfiltered

- Filtered:
  - Resolves queries from indexes + document parsing
  - Emphasis on accuracy
- Unfiltered:
  - Resolves queries from index alone
  - Emphasis on speed
- With properly configured indexes, unfiltered search is both **fast** and **accurate**

Filtered vs. Unfiltered Example

- Docs loaded into a DB with **case sensitive indexing = false**
- Default query options in place (**UNFILTERED**)

```
DOCUMENT 1
{
  "description":
  101 Dalmatians is
  about dogs.
}
```

```
DOCUMENT 2
{
  "description":
  Dogs are fun.
}
```

TERM	DOCUMENTS	
property:value	Doc1	
property:value		Doc2
101	Doc1	
dalmatians	Doc1	
be	Doc1	Doc2
about	Doc1	
dog	Doc1	Doc2
fun		Doc2

Filtered vs. Unfiltered Example

- Example query:
  - Find docs containing the word “dogs”, case sensitive, unfiltered

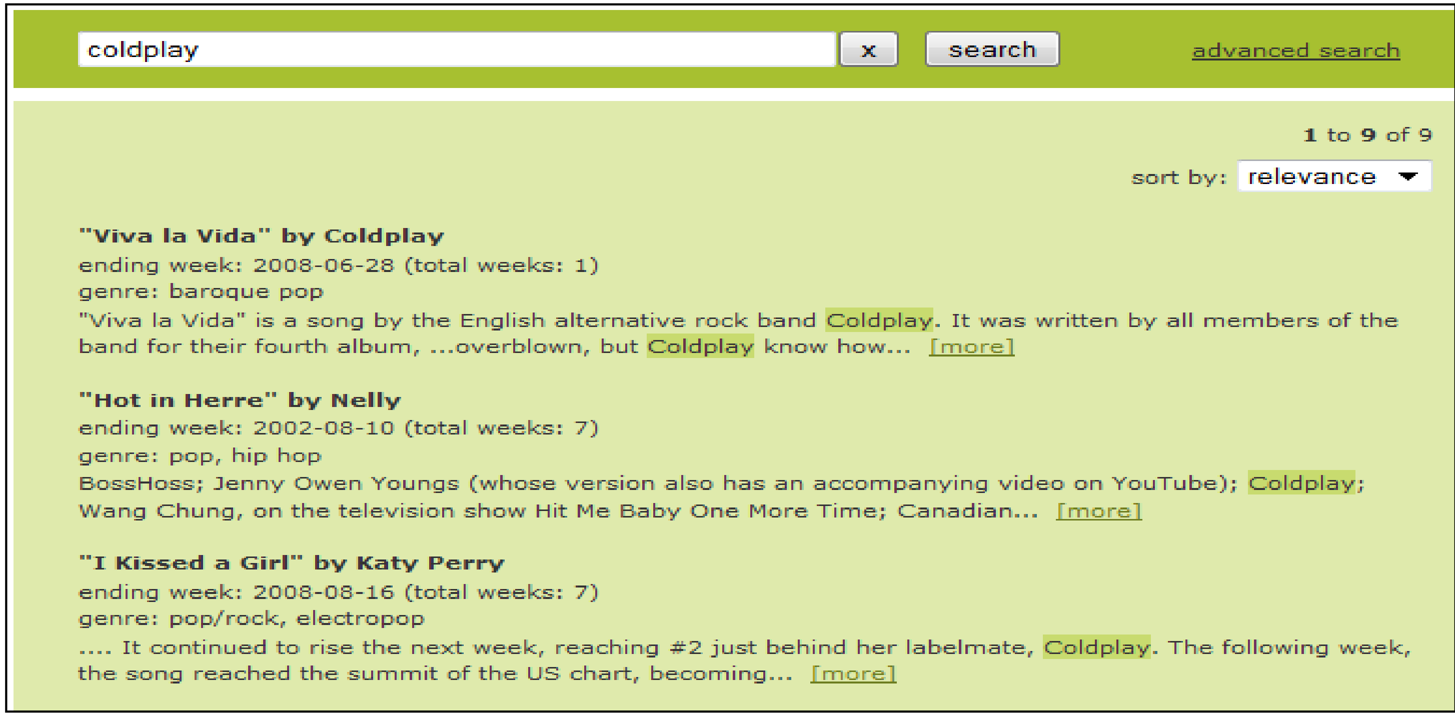
```
DOCUMENT 1
{
  "description":
  101 Dalmatians is
  about dogs.
}
```

```
DOCUMENT 2
{
  "description":
  Dogs are fun.
}
```

TERM	DOCUMENTS	
<element>:value	Doc1	
<element>:value		Doc2
101	Doc1	
dalmatians	Doc1	
be	Doc1	Doc2
about	Doc1	
dog	Doc1	Doc2
fun		Doc2

Constraints

- Constrain search results to subsets of content
- Designed to take advantage of index configuration
- Out of the box constraint options
- Create custom constraint options



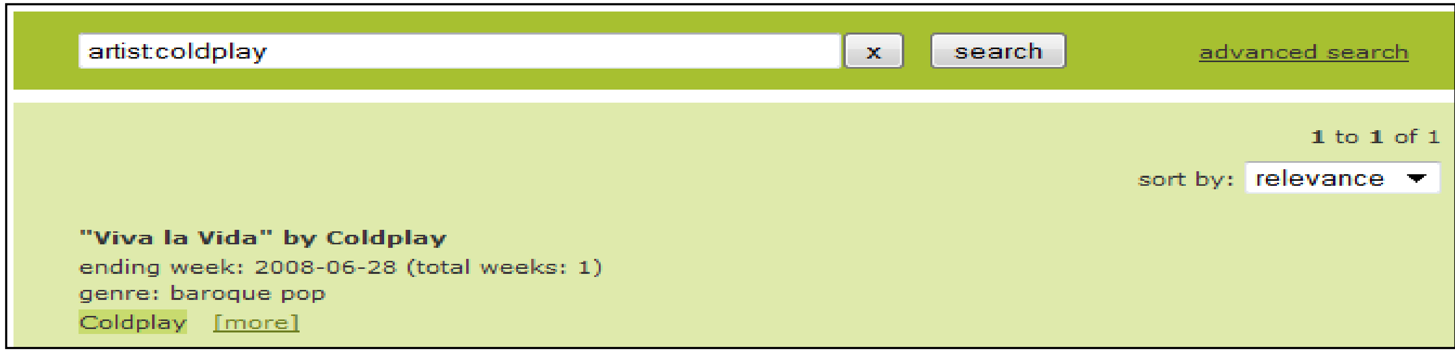
coldplay

1 to **9** of 9

"Viva la Vida" by Coldplay
ending week: 2008-06-28 (total weeks: 1)
genre: baroque pop
"Viva la Vida" is a song by the English alternative rock band Coldplay. It was written by all members of the band for their fourth album, ...overblown, but Coldplay know how... [more]

"Hot in Herre" by Nelly
ending week: 2002-08-10 (total weeks: 7)
genre: pop, hip hop
BossHoss; Jenny Owen Youngs (whose version also has an accompanying video on YouTube); Coldplay; Wang Chung, on the television show Hit Me Baby One More Time; Canadian... [more]

"I Kissed a Girl" by Katy Perry
ending week: 2008-08-16 (total weeks: 7)
genre: pop/rock, electropop
.... It continued to rise the next week, reaching #2 just behind her labelmate, Coldplay. The following week, the song reached the summit of the US chart, becoming... [more]



artist:coldplay

1 to **1** of 1

"Viva la Vida" by Coldplay
ending week: 2008-06-28 (total weeks: 1)
genre: baroque pop
Coldplay [more]

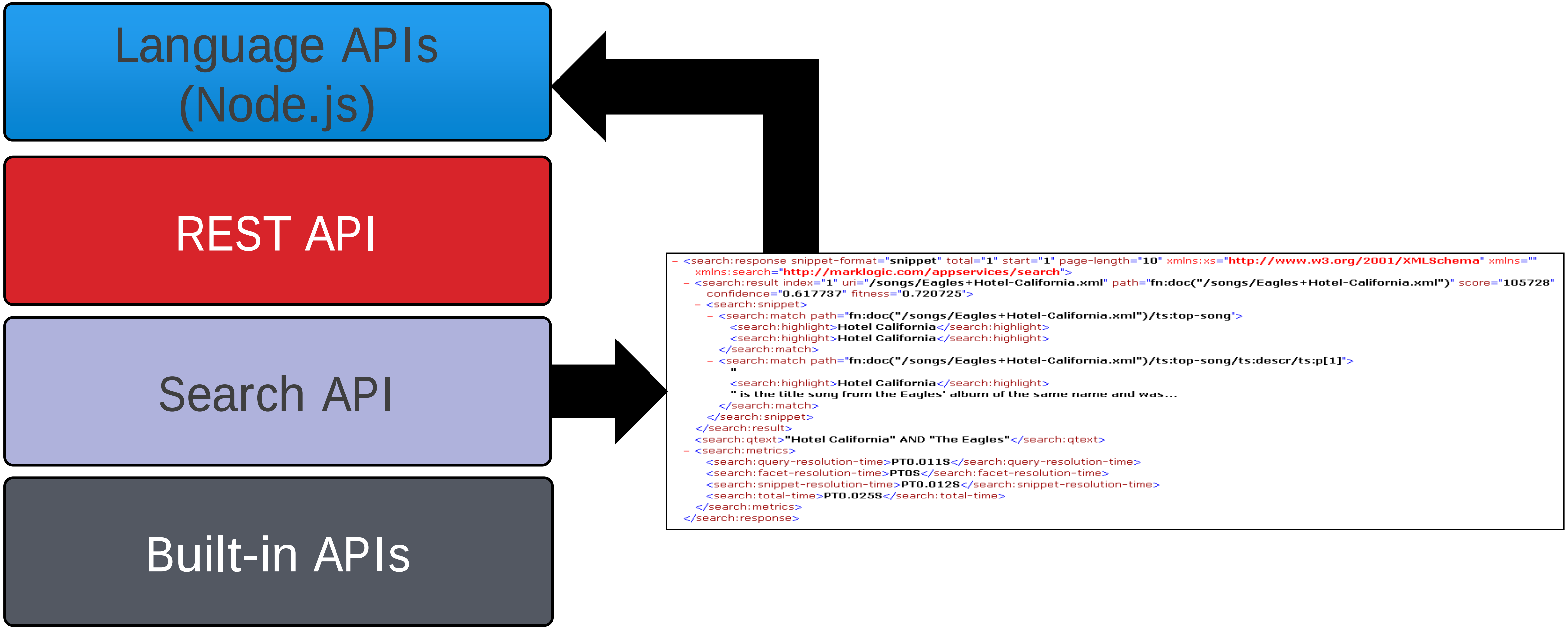
Constraint Examples

- VALUE
  - Constrain to a particular XML element or attribute
  - Constraint to a JSON property
- COLLECTION
  - “slice” data by collection, like “rock” or “hip-hop”
- RANGE
  - Find songs on the chart during the 1980s

- PROPERTIES
  - Search a documents metadata
- GEOSPATIAL
- For complete list of constraints:
  - docs.marklogic.com

The Search API Response

- Response can be JSON or XML

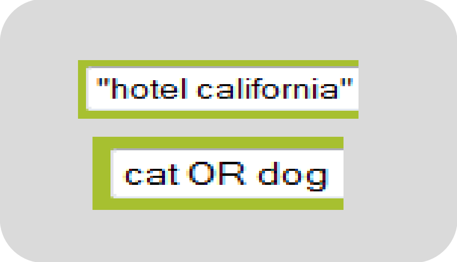


```
- <search:response snippet-format="snippet" total="1" start="1" page-length="10" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns=""
    xmlns:search="http://marklogic.com/appservices/search">
  - <search:result index="1" uri="/songs/Eagles+Hotel-California.xml" path="fn:doc("/songs/Eagles+Hotel-California.xml")" score="105728"
      confidence="0.617737" fitness="0.720725">
    - <search:snippet>
      - <search:match path="fn:doc("/songs/Eagles+Hotel-California.xml")/ts:top-song">
          <search:highlight>Hotel California</search:highlight>
          <search:highlight>Hotel California</search:highlight>
        </search:match>
      - <search:match path="fn:doc("/songs/Eagles+Hotel-California.xml")/ts:top-song/ts:descr/ts:p[1]">
          "
          <search:highlight>Hotel California</search:highlight>
          " is the title song from the Eagles' album of the same name and was...
        </search:match>
      </search:snippet>
    </search:result>
    <search:qtext>"Hotel California" AND "The Eagles"</search:qtext>
  - <search:metrics>
      <search:query-resolution-time>PT0.011S</search:query-resolution-time>
      <search:facet-resolution-time>PT0S</search:facet-resolution-time>
      <search:snippet-resolution-time>PT0.012S</search:snippet-resolution-time>
      <search:total-time>PT0.025S</search:total-time>
    </search:metrics>
  </search:response>
```

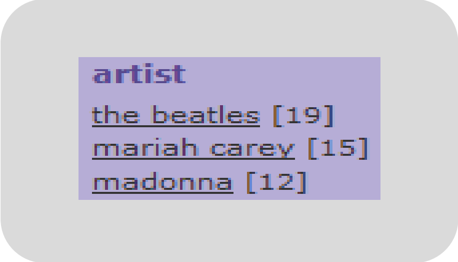
MarkLogic

Search API Response

Grammar



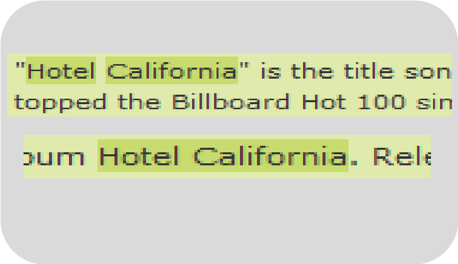
Facets



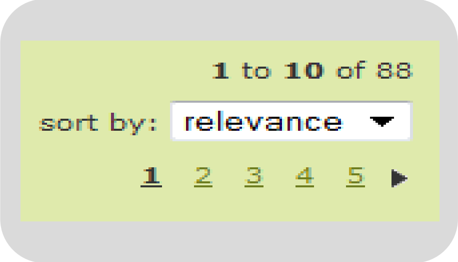
Snippets

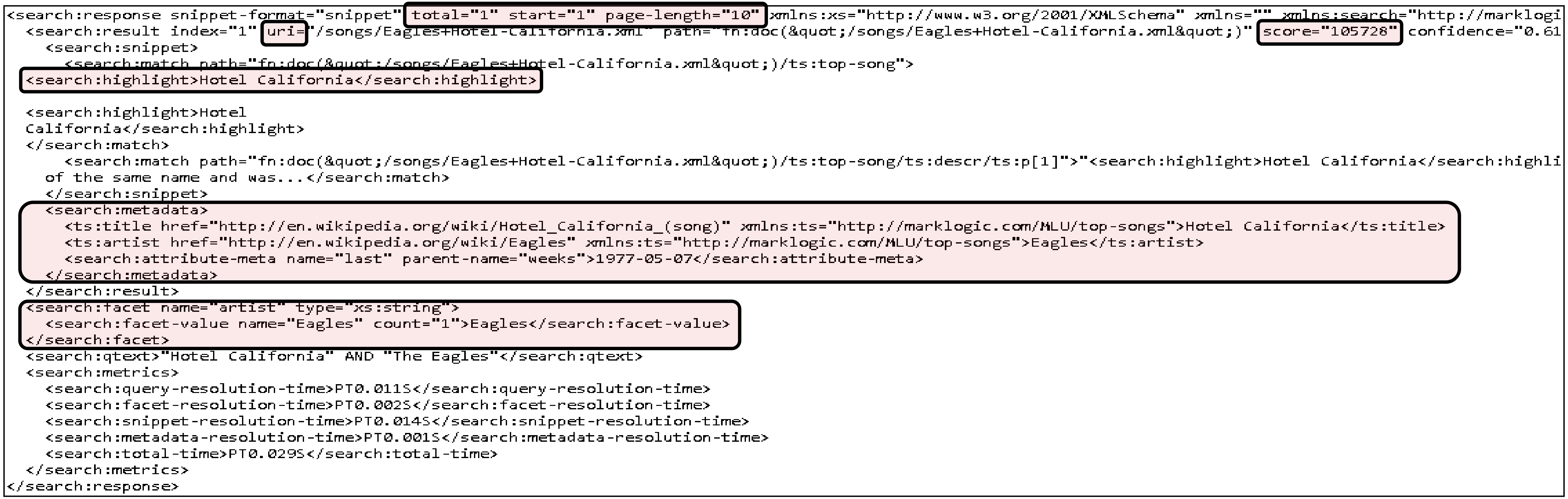
…text text text text
SEARCH TERM text
text text…

Highlights


"Hotel California" is the title son
topped the Billboard Hot 100 sin
oum Hotel California. Rele

Pagination



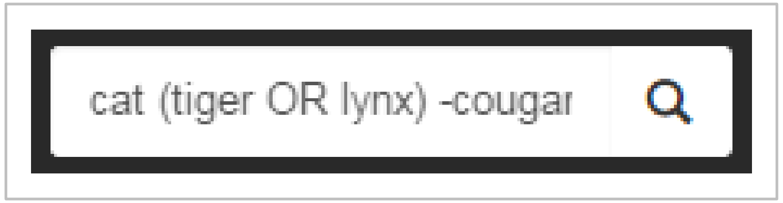


```
<search:response snippet-format="snippet" total="1" start="1" page-length="10" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="" xmlns:search="http://marklogi
  <search:result index="1" uri="/songs/Eagles+Hotel-California.xml" path="fn:doc(&quot;/songs/Eagles+Hotel-California.xml&quot;)" score="105728" confidence="0.61
    <search:snippet>
      <search:match path="fn:doc(&quot;/songs/Eagles+Hotel-California.xml&quot;)/ts:top-song">
  <search:highlight>Hotel California</search:highlight>

  <search:highlight>Hotel
  California</search:highlight>
  </search:match>
      <search:match path="fn:doc(&quot;/songs/Eagles+Hotel-California.xml&quot;)/ts:top-song/ts:descr/ts:p[1]">"<search:highlight>Hotel California</search:highli
    of the same name and was...</search:match>
    </search:snippet>
    <search:metadata>
      <ts:title href="http://en.wikipedia.org/wiki/Hotel_California_(song)" xmlns:ts="http://marklogic.com/MLU/top-songs">Hotel California</ts:title>
      <ts:artist href="http://en.wikipedia.org/wiki/Eagles" xmlns:ts="http://marklogic.com/MLU/top-songs">Eagles</ts:artist>
      <search:attribute-meta name="last" parent-name="weeks">1977-05-07</search:attribute-meta>
    </search:metadata>
  </search:result>
  <search:facet name="artist" type="xs:string">
    <search:facet-value name="Eagles" count="1">Eagles</search:facet-value>
  </search:facet>
  <search:qtext>"Hotel California" AND "The Eagles"</search:qtext>
  <search:metrics>
    <search:query-resolution-time>PT0.011S</search:query-resolution-time>
    <search:facet-resolution-time>PT0.002S</search:facet-resolution-time>
    <search:snippet-resolution-time>PT0.014S</search:snippet-resolution-time>
    <search:metadata-resolution-time>PT0.001S</search:metadata-resolution-time>
    <search:total-time>PT0.029S</search:total-time>
  </search:metrics>
</search:response>
```

Three Ways to Search

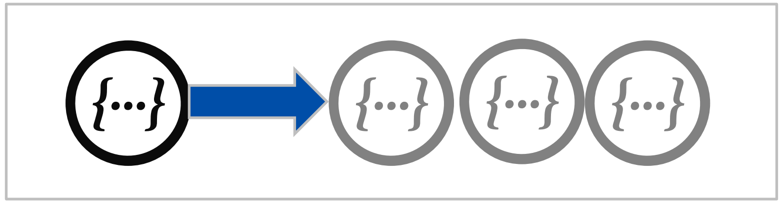
String Query



cat (tiger OR lynx) -cougar

- Simple:
  - – Pass in the string.
- Intuitive:
  - – Grammar understood.
- Less expressive.

Query By Example (QBE)



- Pattern / prototype based:
  - – Find others like this.
- More expressive.

Structured Query

FLL TXT
RANGE

- Combine together:
  - – Full text, geospatial, range and semantic queries.
- Most expressive.

Approach to Search

Define Criteria

- What are you looking for?
  - Define this with various query builders.

Refine Result Set

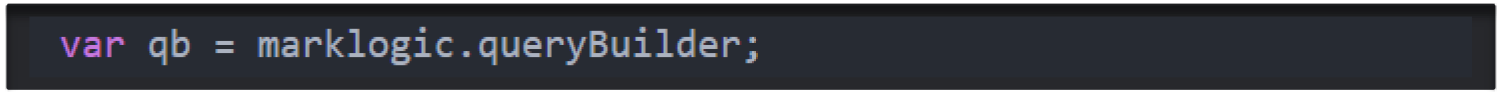
- What do you want back?
  - How many results?
  - Sort orders?
  - Facet data?

Execute Search

- Perform the search and return the response.

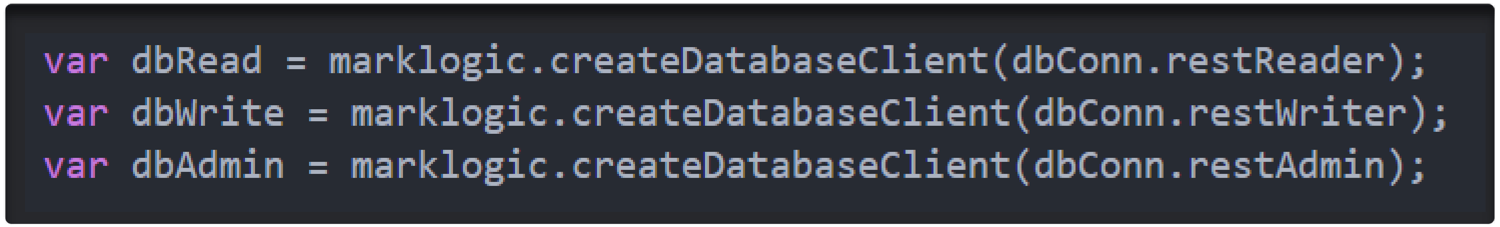
Building a Search

- You’ll need a query builder instance:



```
var qb = marklogic.queryBuilder;
```

- You’ll need your database clients:

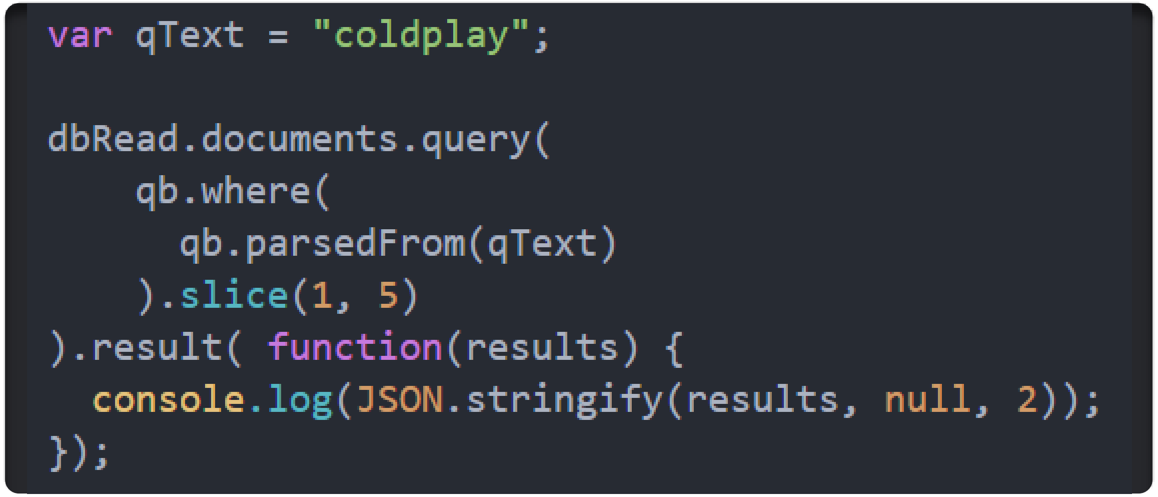


```
var dbRead = marklogic.createDatabaseClient(dbConn.restReader);
var dbWrite = marklogic.createDatabaseClient(dbConn.restWriter);
var dbAdmin = marklogic.createDatabaseClient(dbConn.restAdmin);
```

- Assume these things are in place for the next few code examples.

String Query Example 1

- As simple as it gets:
  - **Criteria:** Documents that have the word “coldplay” anywhere in the document.
  - **Result set:** bring data back for 5 most relevant matches.
  - **Execute:** using callback result handler.

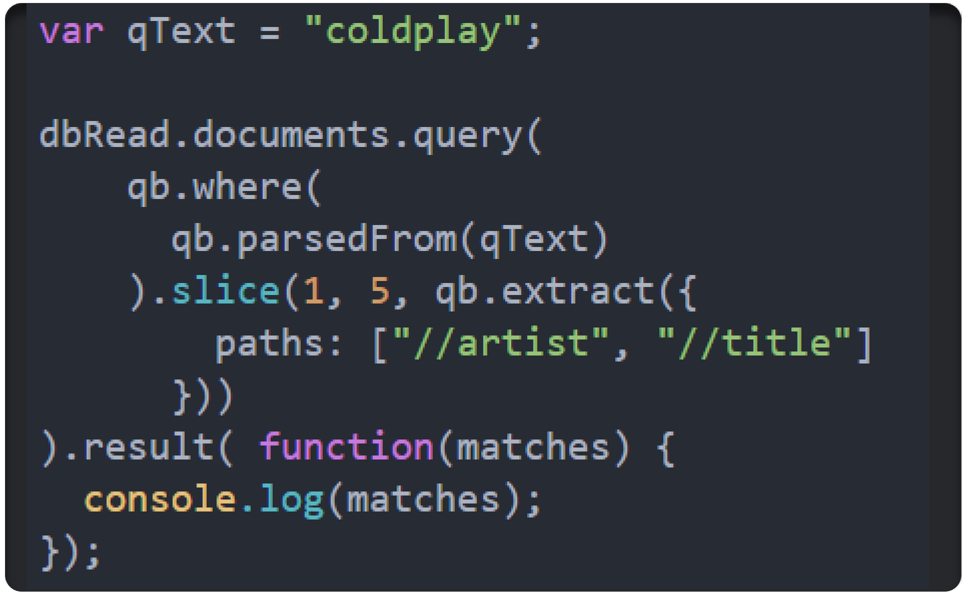


```
var qText = "coldplay";

dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    ).slice(1, 5)
).result( function(results) {
  console.log(JSON.stringify(results, null, 2));
});
```

String Query Example 2

- Controlling the response data:
  - Like before, we limit the response to the 5 most relevant matches.
  - We also ask for the title and artist data to be extracted and returned.

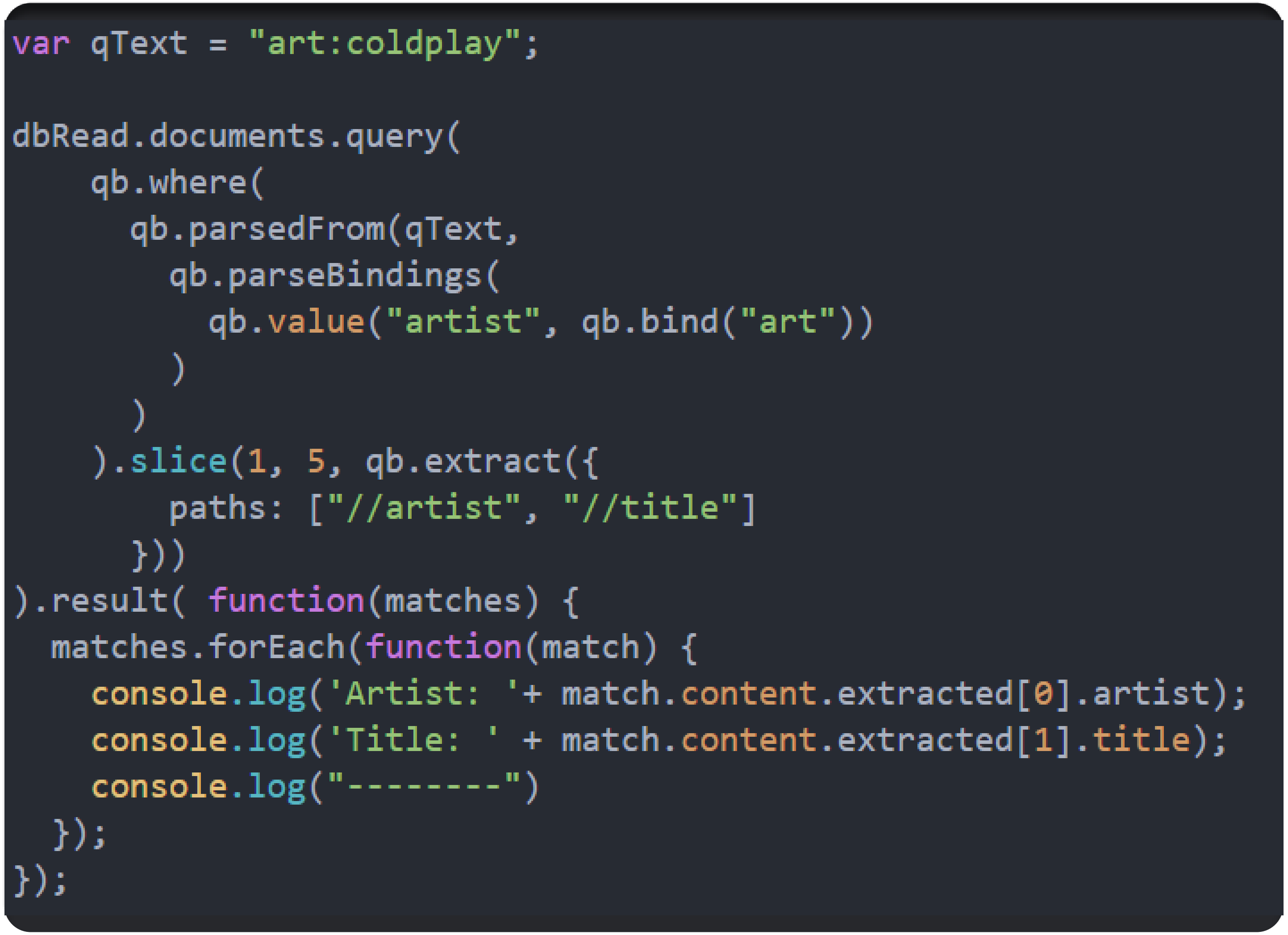


```
var qText = "coldplay";

dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    ).slice(1, 5, qb.extract({
        paths: ["//artist", "//title"]
      }))
).result( function(matches) {
  console.log(matches);
});
```

String Query Example 3

- Implementing custom grammar:
  - Now anytime “art:term” is present in the grammar it means something specific.
  - Look for the term in the artist property.

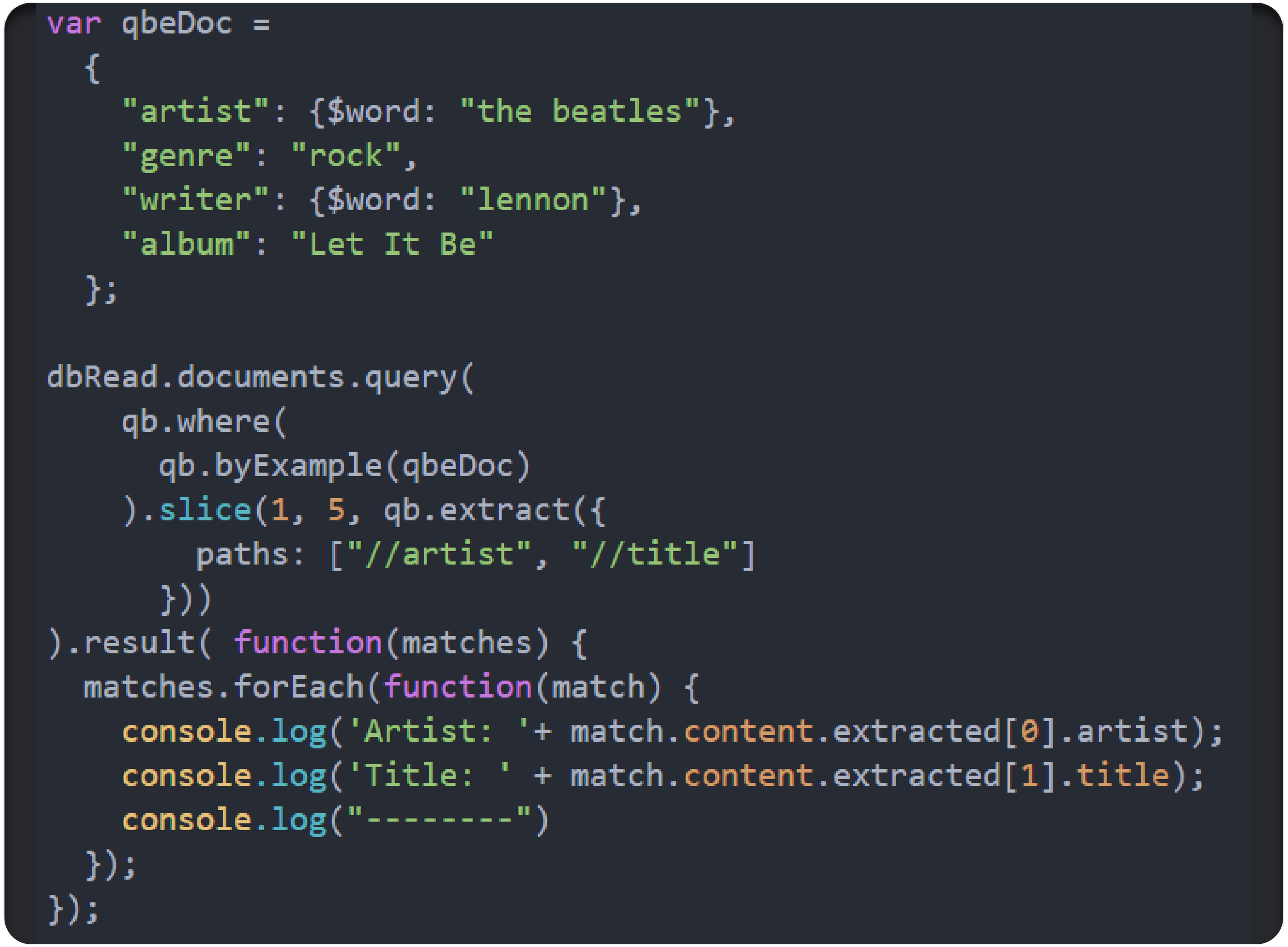


```
var qText = "art:coldplay";

dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText,
        qb.parseBindings(
          qb.value("artist", qb.bind("art"))
        )
      )
    ).slice(1, 5, qb.extract({
        paths: ["//artist", "//title"]
      }))
).result( function(matches) {
  matches.forEach(function(match) {
    console.log('Artist: '+ match.content.extracted[0].artist);
    console.log('Title: ' + match.content.extracted[1].title);
    console.log("--------")
  });
});
```

Query By Example (Example)

- Find documents like…
  - Artist contains “the beatles”
  - Genre = “rock”
  - Writer contains “lennon”
  - Album = “Let It Be”

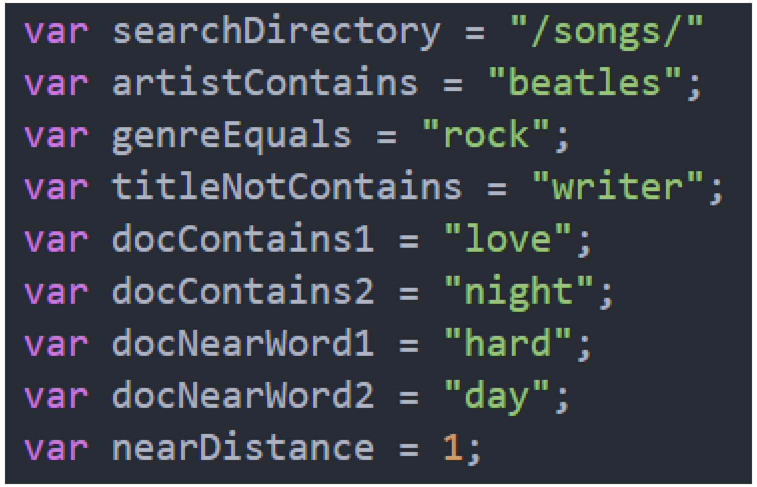


```
var qbeDoc =
  {
    "artist": {$word: "the beatles"},
    "genre": "rock",
    "writer": {$word: "lennon"},
    "album": "Let It Be"
  };

dbRead.documents.query(
    qb.where(
      qb.byExample(qbeDoc)
    ).slice(1, 5, qb.extract({
        paths: ["//artist", "//title"]
      }))
).result( function(matches) {
  matches.forEach(function(match) {
    console.log('Artist: '+ match.content.extracted[0].artist);
    console.log('Title: ' + match.content.extracted[1].title);
    console.log("--------")
  });
});
```

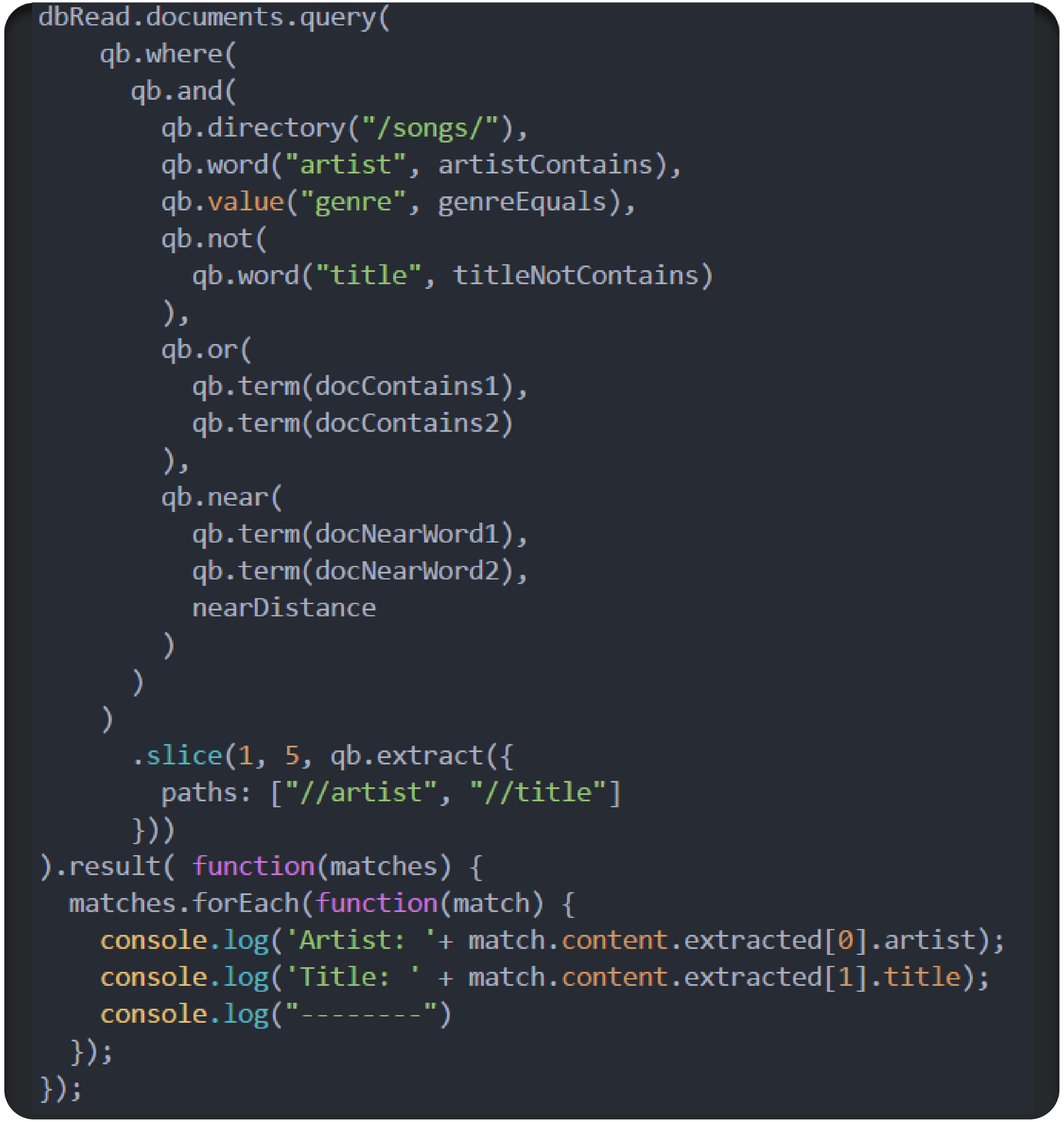
Structured Query Example

- Expressive and customizable.
- Assume these variables:



```
var searchDirectory = "/songs/"
var artistContains = "beatles";
var genreEquals = "rock";
var titleNotContains = "writer";
var docContains1 = "love";
var docContains2 = "night";
var docNearWord1 = "hard";
var docNearWord2 = "day";
var nearDistance = 1;
```

- What are we asking?



```
dbRead.documents.query(
  qb.where(
   qb.and(
    qb.directory("/songs/"),
    qb.word("artist", artistContains),
    qb.value("genre", genreEquals),
    qb.not(
     qb.word("title", titleNotContains)
    ),
    qb.or(
     qb.term(docContains1),
     qb.term(docContains2)
    ),
    qb.near(
     qb.term(docNearWord1),
     qb.term(docNearWord2),
     nearDistance
    )
   )
  )
   .slice(1, 5, qb.extract({
    paths: ["//artist", "//title"]
   }))
).result( function(matches) {
 matches.forEach(function(match) {
  console.log('Artist: '+ match.content.extracted[0].artist);
  console.log('Title: ' + match.content.extracted[1].title);
  console.log("--------")
 });
});
```

Learn More

- https://mlu.marklogic.com/ondemand/
  - Free On Demand tutorial covering QBE
- Node.js client API Developer Guide:
  - http://docs.marklogic.com/guide/node-dev
- Search Developer Guide:
  - http://docs.marklogic.com/guide/search-dev

Labs: Unit 7

Exercise 1 – Exercise 9
DIY: Search Star Wars

Unit Review Question 1:

Stemming rules are based on ___________ and ____________.

Unit Review Question 1:

Stemming rules are based on **language** and **part of speech**.

Unit Review Question 2:

Term frequency is based on?

1. The content of the matching document
2. The content of the entire database
3. All of the above
4. None of the above

Unit Review Question 2:

Term frequency is based on?

1. **The content of the matching document**
2. The content of the entire database
3. All of the above
4. None of the above

Unit Review Question 3:

Which of the following is true:

When performing a search…

1. You are searching every document in the database.
2. You are searching every document in the database for which you have read permissions.
3. You must filter documents in order to get results.
4. All of the above.

Unit Review Question 3:

Which of the following is true:

When performing a search…

1. You are searching every document in the database.
2. **You are searching every document in the database for which you have read permissions.**
3. You must filter documents in order to get results.
4. All of the above.