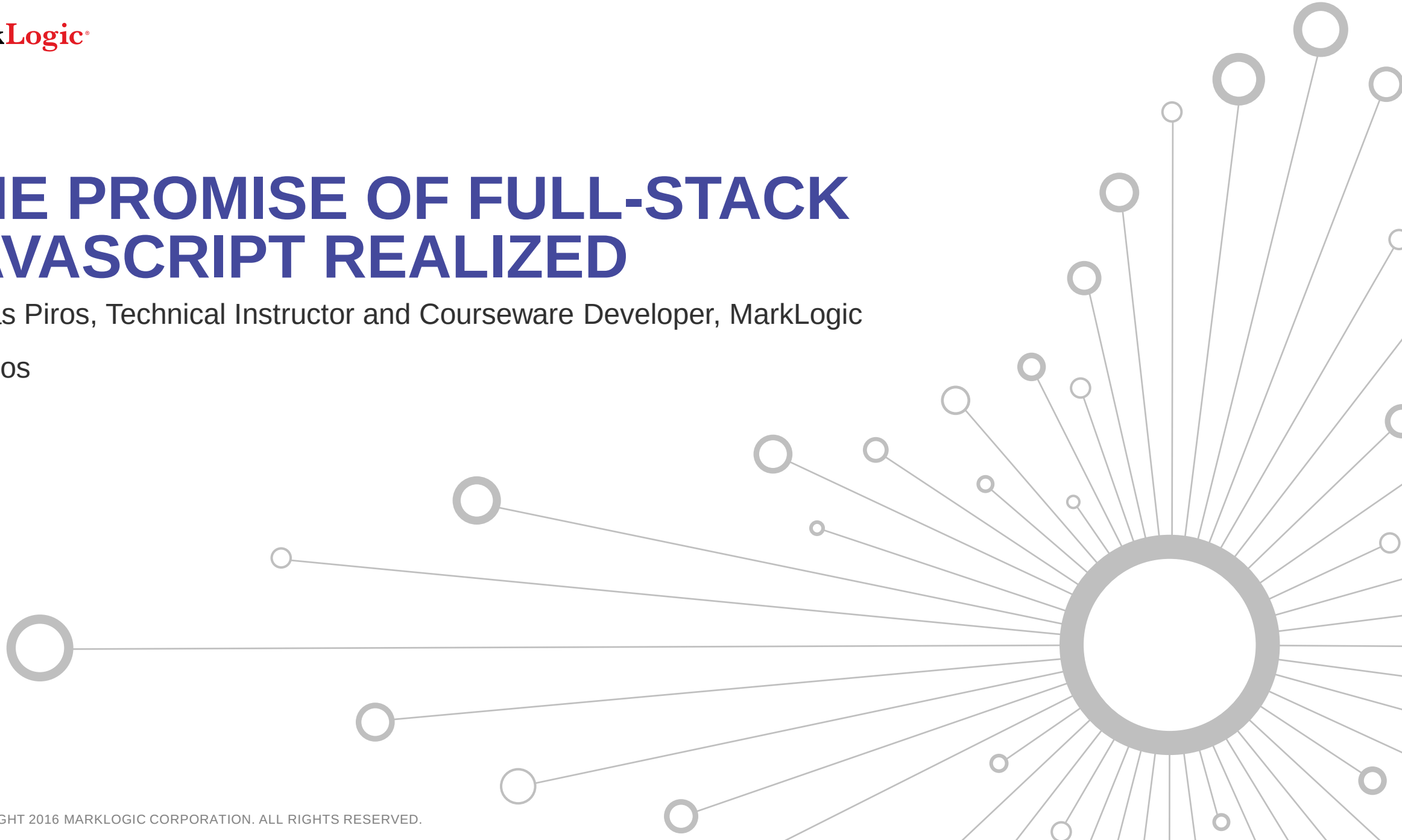
MarkLogic

THE PROMISE OF FULL-STACK JAVASCRIPT REALIZED

Tamas Piros, Technical Instructor and Courseware Developer, MarkLogic

@tpiros



MarkLogic

/me

- Senior Technical Instructor @ MarkLogic
- 10+ years of full stack web development
- 5+ years of technical training experience
- Prolific blogger on the “latest and greatest” web tech

- Get in touch via Twitter (@tpiros) or via email tamas.piros@marklogic.com

MarkLogic

Agenda

- A little bit on JavaScript
- Discussion on various application architectures
- Demo

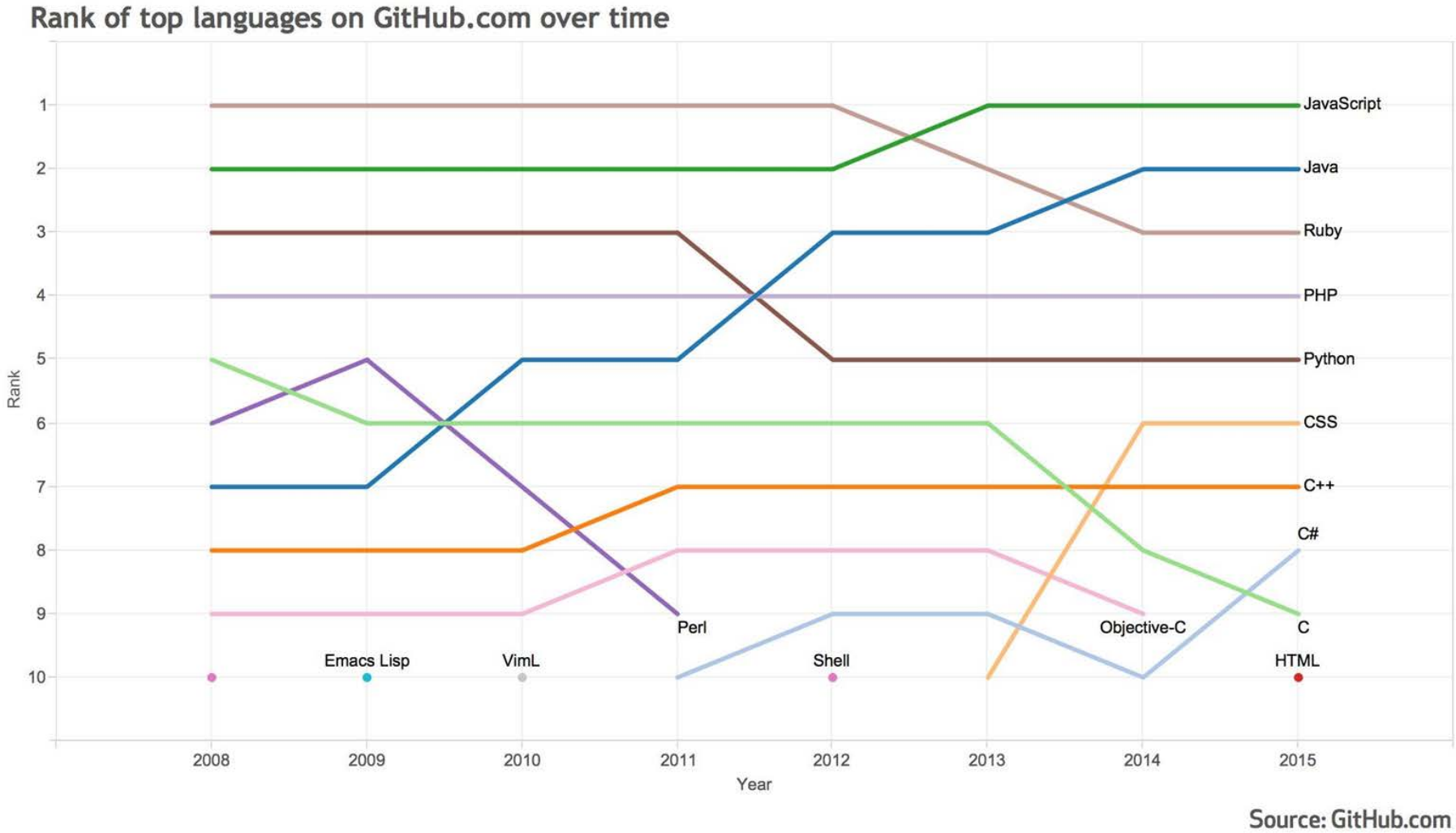
The Whys and the Wherefores

- Building a web app means dozens of architectural decisions
- Using the MarkLogic database is a natural choice
- Save time and effort by applying rapid development
- Constant iteration for testing and improvements
- Maximum efficiency, speed and robustness
- Be lean, be agile

MarkLogic

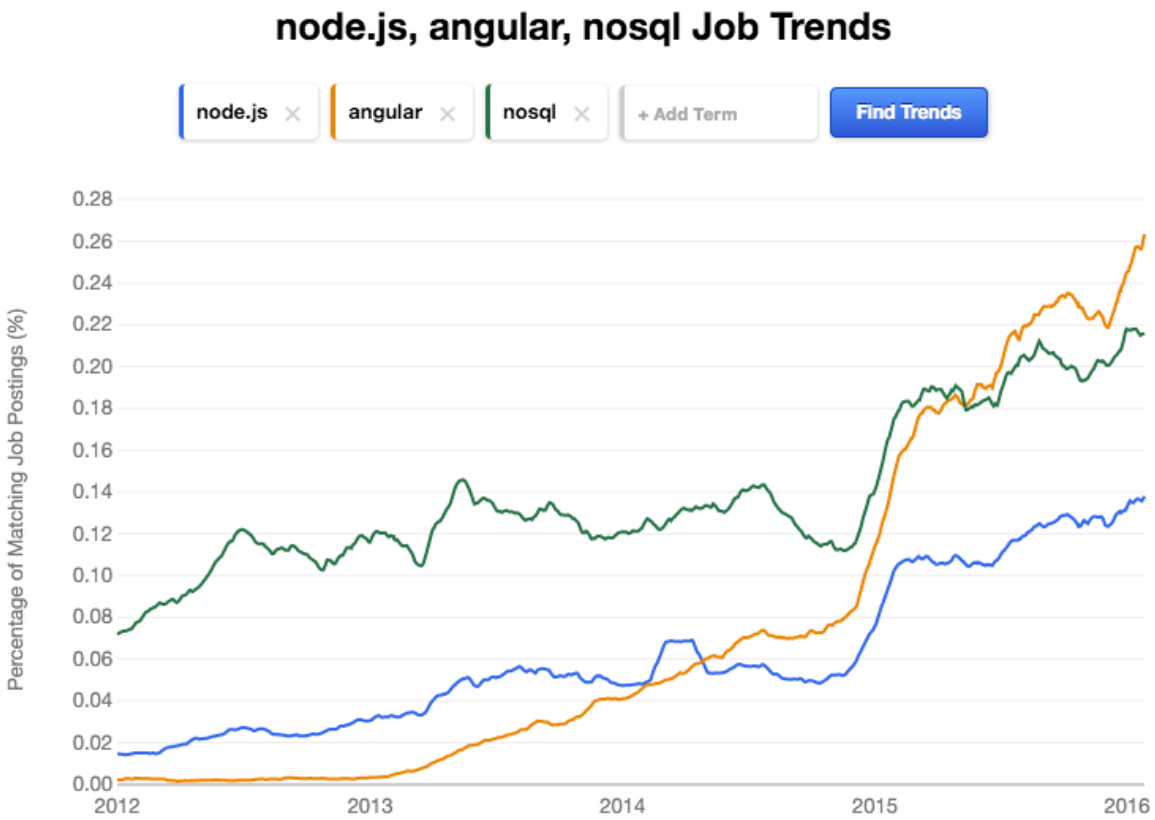
Why JavaScript?

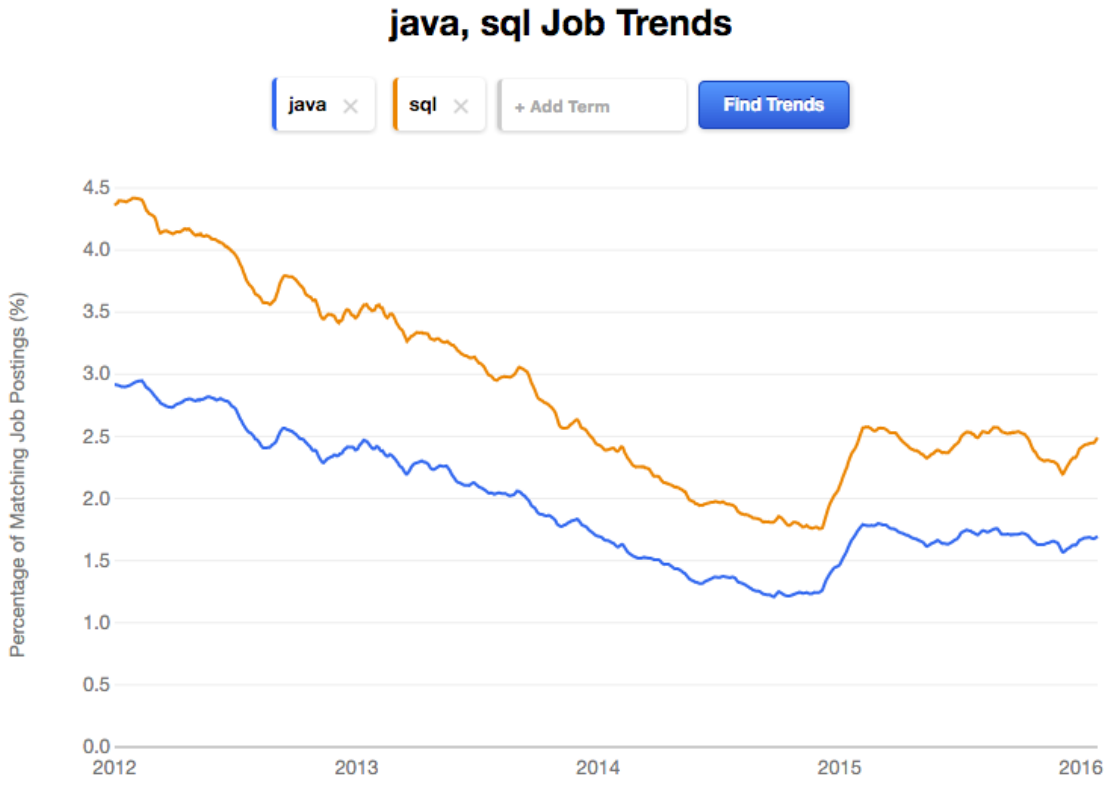
- JavaScript is everywhere
- JavaScript can be used both at the client well as at the server



MarkLogic

And If You Need More Convincing...





MarkLogic

Why JavaScript?

- JavaScript has been around since 1995
- It has gone through multiple iterations
- It has been used predominantly in the browser
  - Specific frameworks/libraries exist such as jQuery, Angular and Backbone.js
- It's lightweight and expressive
- Thanks to Google's V8 compiler JavaScript is blazingly fast
- As of 2009 it also runs at the server-side thanks to Node.js

Traditional Architecture

User Interface

- Data views
- User workflow
- Browser

Communication over HTTP

Middle-tier

- Business rules
- Application logic

- Two tiered architecture
- Separation of user interface (view) from the middle-tier (application logic)
- Missing database tier for persistent storage

Traditional Multi-tier Architecture

User Interface

- Data views
- User workflow
- Browser

Communication over HTTP

Middle-tier

- Business rules
- Application logic

Communication over HTTP

Database tier

- Persistent storage
- Stored procedures

- Three (multi) tiered architecture
- Adds database for persistent storage
- Presentation, application processing and data management functions are physically separated

MarkLogic

Traditional Multi-tier Architecture

Technology	Data Model	Environment
Struts	HTML	JSP / JavaScript

Spring	Java Object Graph	Java

Oracle	Tables, rows, columns	SQL, PL/SQL

Multi-tier Architecture

Challenges

- Different tiers use different technologies
- And use different data structures
- Difficult to map and configure data to domain model in application

Wouldn’t it be great if we could use a single programming language and a single data structure?

Application Architecture

User Interface

- Data views
- User workflow
- Browser

JSON over HTTP

Middle-tier

- Business rules
- Application logic

Pros

- Same language throughout the stack
- Lightweight data format
- Data format ‘natively’ understood by JavaScript

Con(s)

- Missing persistent data storage

MarkLogic

Application Architecture

User Interface

- Data views
- User workflow
- Browser

JSON over HTTP

Middle-tier

- Business rules
- Application logic

Wouldn't it be nice to add a **database** to this architecture that can:

- Store JSON documents natively (along with XML, binary and RDF)?
- Allow you to construct queries using JavaScript?
- Have ACID properties instead of eventual consistency?
- Give you all the indexes you need and allow you to execute search out of the box?
- Apply role based, document level security?
- Execute SPARQL queries?
- Manage the database via REST API calls?

Application Architecture

User Interface

- Data views
- User workflow
- Browser

JSON over HTTP

Middle-tier

- Business rules
- Application logic

JSON/XML over HTTP

Database-tier

- Persistent storage

MarkLogic can:

- Store JSON documents natively (along with XML, binary and RDF)
- Allow you to construct queries using JavaScript
- Have ACID properties instead of eventual consistency
- Give you all the indexes you need and allow you to execute search out of the box
- Apply role based, document level security
- Execute SPARQL queries
- Manage the database via REST API calls

Multi-tier Architecture

	Technology	Data Model	Environment
	Angular	JSON	JavaScript
	Node.js	JSON	JavaScript
	MarkLogic	JSON	JavaScript

MarkLogic

JavaScript at the Middle-tier

- Run JavaScript at the server via Node.js
- It's fast – due to the event loop and asynchronous features of the language
- MarkLogic has a Node.js Client API
- Registered npm package
- Focus on application features rather than plumbing

```
var marklogic  = require('marklogic');
var db =
marklogic.createDatabaseClient(connect
ion);

var qb = marklogic.queryBuilder;
db.documents.query(
  qb.where(
    qb.collection('books')
  )
).result().then(function(response) {
  console.log(response);
});
```

MarkLogic

JavaScript in MarkLogic (Database Tier)

- Runs on Google's V8 engine
- Allows you to execute JavaScript code close to your data
  - Both native JavaScript (including some ES2015) and proprietary JavaScript
- Supports all geospatial query types

```
var items = [];
var results = cts.search(cts.andQuery(['skywalker',
cts.notQuery('force')]));
for (var result of results) {
  items.push({score: cts.score(result), uri: xdmp.nodeUri(result),
name: result.root.name});
}
items;
```

Data Hub Architecture

- Replace the User Interface with Web Service endpoints
- Beneficial for Data Hubs
- Consumption of endpoints happen at several layers later
- JavaScript HTTP services are useful even without a front-end
- Think along the lines of Service Oriented Architecture and microservices

MarkLogic

MarkLogic ♥ JavaScript

- Future of JavaScript is determined by Ecma committee
- Some ECMAScript 2015 features are already implemented in MarkLogic 8 with more coming in MarkLogic 9:
  - Iterators, generators
  - Arrow functions
  - Classes
  - Template strings
  - Spread operator and rest parameter
  - Let keyword
  - Object destructuring
  - Template literals
  - Arrow functions
  - Promises
  - Maps and Sets
- JavaScript is a first class language in MarkLogic and we will keep pace with the growing community

DEMO


```
{
  "filename": "05.JPG",
  "location": {
    "type": "Point",
    "coordinates": [
      37.809333,
      -122.475833
    ]
  },
  "make": "Apple",
  "model": "iPhone 4",
  "created": 1315246591000,
  "binary": "/binary/05.JPG"
}
```


{
"filename": "05.JPG",
"location": {
"coordinates": [
37.809333,
-122.475833
]
},
"make": "Apple",
"model": "iPhone 4",
"created": 1315246591000,
"binary": "/binary/05.JPG"
}
This is good...
node
This is better!!!


{
"filename": "05.JPG",
"location": {
"type": "Point",
"coordinates": [
37.809333,
-122.475833
],
"city": "San Francisco",
"country": "United States"
},
"make": "Apple",
"model": "iPhone 4",
"created": 1315246591000,
"binary": "/binary/05.JPG"
}
What do we know about...
node
js
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
<sem:triple>
<sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
<sem:predicate>http://xmlns.com/foaf/0.1/homepage</sem:predicate>
<sem:object>http://www.usa.gov/</sem:object>
</sem:triple>
<sem:triple>
<sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
<sem:predicate>http://dbpedia.org/ontology/anthem</sem:predicate>
<sem:object>http://dbpedia.org/resource/The_Star-Spangled_Banner</sem:object>
</sem:triple>
<sem:triple>
<sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
<sem:predicate>http://dbpedia.org/ontology/capital</sem:predicate>
<sem:object>http://dbpedia.org/resource/Washington,_D.C.</sem:object>
</sem:triple>

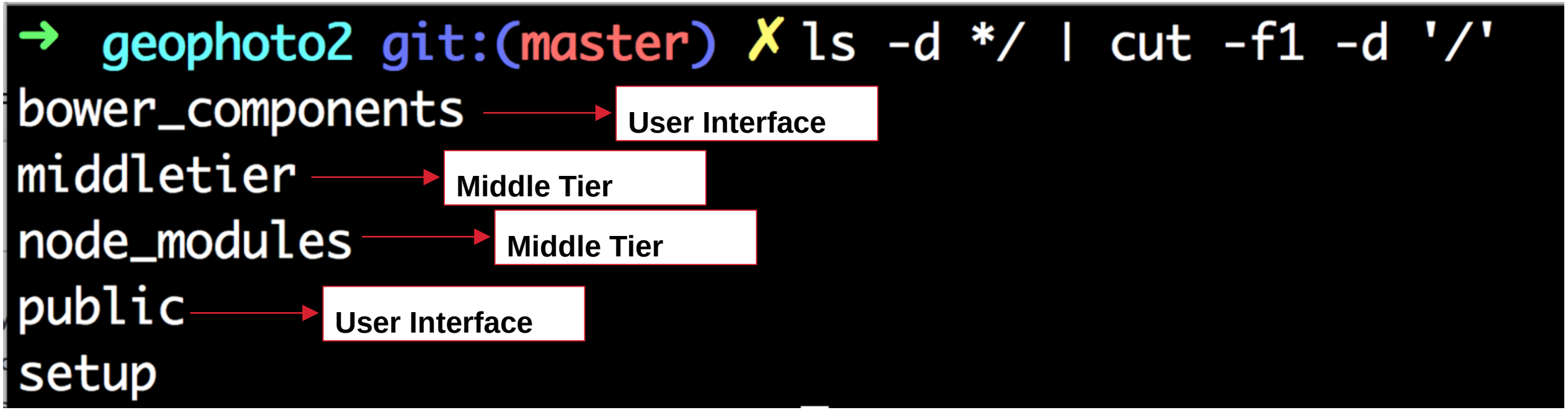
MarkLogic

GeoPhoto – Technical Details

- Three-tiered architecture
- Application that works with geospatial metadata (Exif) extracted from images
- Uses JSON documents, JPEG binaries, RDF triples
- Authentication and authorization via JWT (JSON Web Tokens)
- Document security using MarkLogic's built in security
- Single Page Application design
- Text and GeoSpatial search
- Uses AngularJS at the UI, Node.js at the middle-tier
- Server-side JavaScript is used at MarkLogic

MarkLogic

GeoPhoto – Technical Details



GeoPhoto – Technical Details

- Same data type means significant reduction in format mismatch between the tiers
- Share code and data models between the tiers
- Performance and application scalability

MarkLogic

Summary

- Benefits of using the same language throughout the stack
- Helps with reduction of data mismatch between tiers
- Helps with code portability
- JavaScript is a first class language in MarkLogic
- Node.js Client API saves you from plumbing

MarkLogic

Resources

- Get the database for free!
- GeoPhoto (GitHub)
- Samplestack (GitHub)
- Character Search v1 (GitHub)
- Character Search v2 (GitHub)
- MarkLogic Java API (GitHub)
- MarkLogic Node.js API (GitHub)
- How is MarkLogic different from MongoDB? (Article)
- Free Training

Q&A