# Modern Architectures with Spring and JavaScript

Martin Lippert, VMware
mlippert@vmware.com, @martinlippert

# Were do we come from?

http://www.nasa.gov

# Servlet Specification

mostly static HTML created on server
Template Engines

# JSP Specification

mostly static HTML created on server
no template engines necessary anymore

# Web Frameworks

mostly static HTML created on server
various framework, supporting:
authentication, session-handling, page flows, etc.

# JavaScript

only used to do some kid‘s stuff

Maestro

File Edit Window Downlink Help

Orbital View

Bonneville

Site: 17 Site: 18 Site: 16 Site: 14 Site: 22 Site: 24 Site: 11 Site: 9 Site: 8 Site: 3 Site: 10 Site: 25 Site: 26 Site: 28 Site: 1 Site: 29 Site: 33 Site: 30 Site: 31 Site: 34 Site: 5 Site: 2 Site: 4

Or from here?

Desktop?

SWT?

Swing?

EDR Search View

Sol Range

Start End

Instruments

FRONT_HAZCAM_LEFT
FRONT_HAZCAM_RIGHT

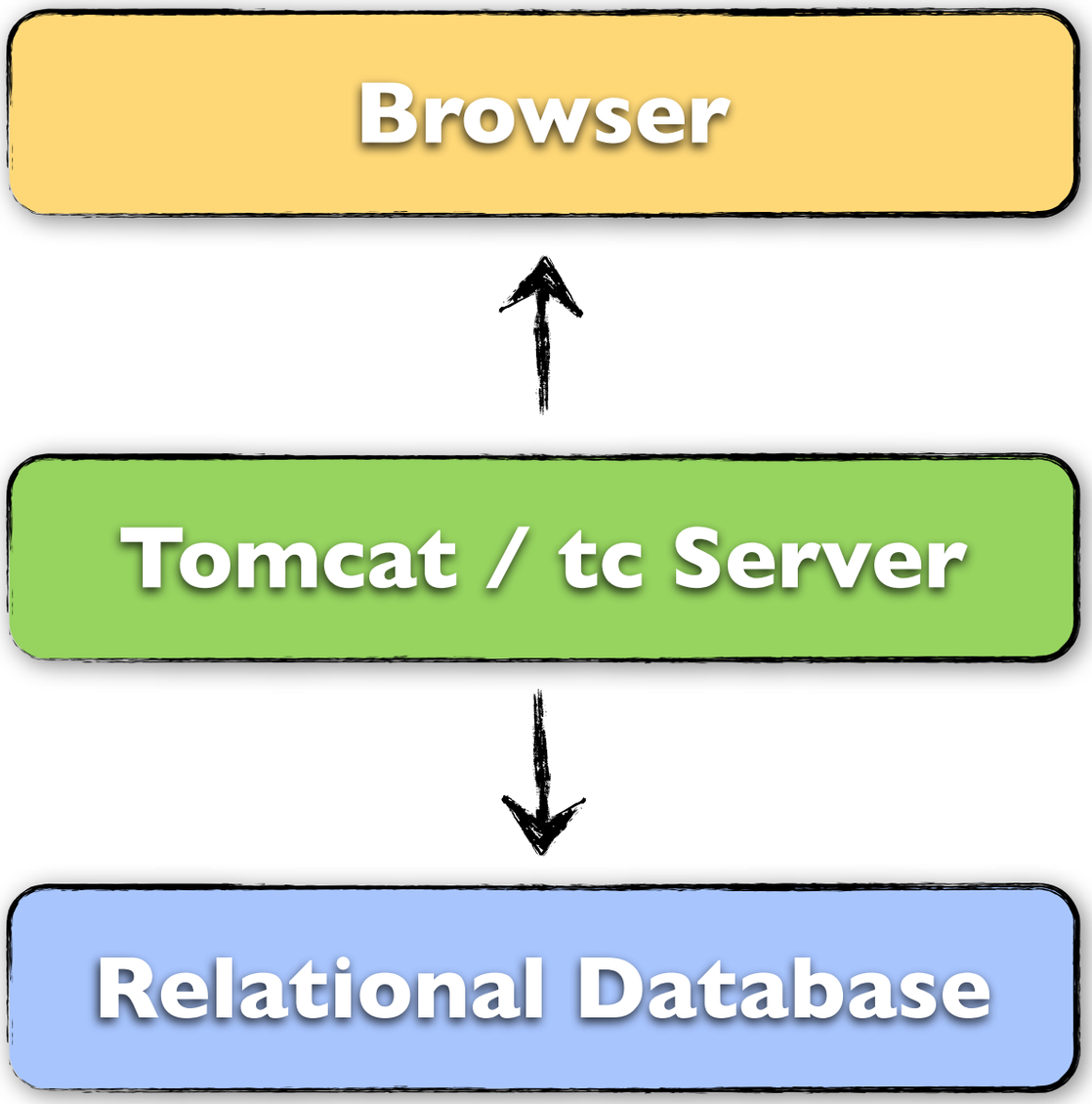
Use Selected Orbital R

Go

| Product ID | Instrument |
|---|---|
| 2F134356767EFF2600P1212L0M1 | FRONT_HAZ |
| 2F134449644EFF2700P1212L0M1 | FRONT_HAZ |
| 2F134614869EFF2700P1403L0M1 | FRONT_HAZ |
| 2F134615161EDN2700P1131L0M1 | FRONT_HAZ |
| 2F135147950EDN2700P1131L0M1 | FRONT_HAZ |
| 2F135148174ESF2700P1127L0M1 | FRONT_HAZ |
| 2F135149189EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135149794EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135150380EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135150997EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135151610EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135152416EDN2700P1141L0M1 | FRONT_HAZ |
| 2F135152602EFF2700P1212L0M1 | FRONT_HAZ |
| 2F135153765EDN2700P1111L0M1 | FRONT_HAZ |

Image View Image View Image View Image View

# Typical Runtime Structures



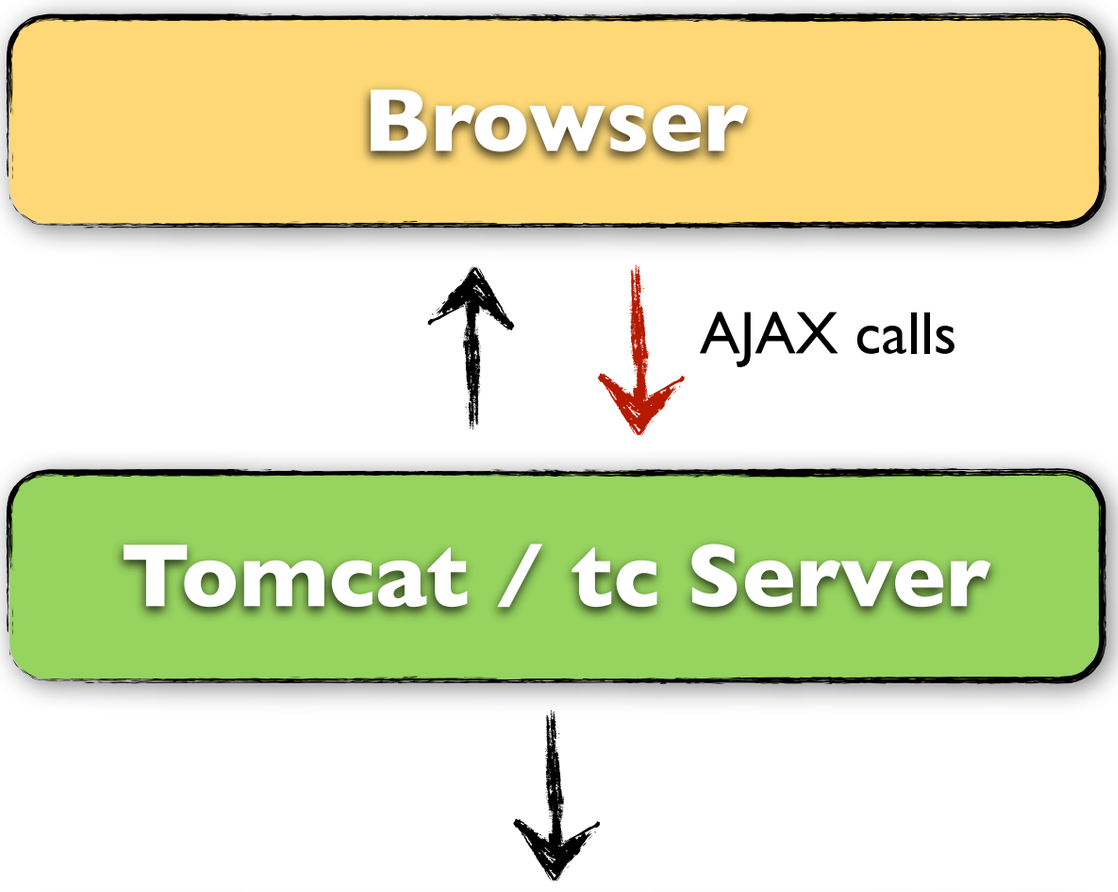

render HTML

business logic &
page rendering

contains data

# And today...

# What happens?

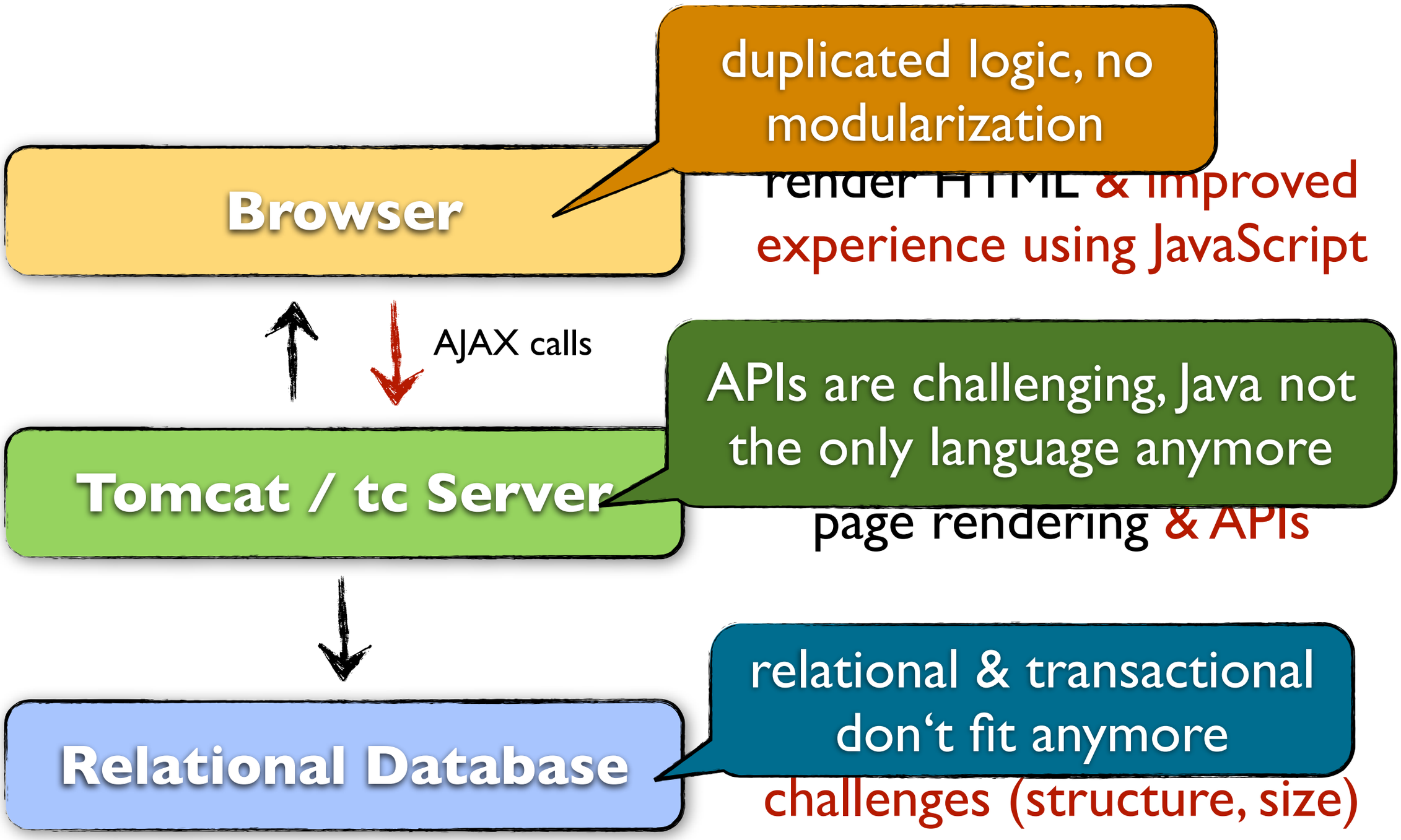



render HTML & improved experience using JavaScript

business logic & page rendering & APIs

contains data & new challenges (structure, size)

# A few observations



# Different pictures

AWS

node.js

NoSQL

JavaScript

Hadoop

CoffeeScript

## Startups

Scala

Ruby/Rails

Clojure

PaaS

HTML5/CSS3

HTML/css

Java

JavaScript

## „old economy“

Application Server

RDBMS

# Where do we go from here?

# The Client Side

# Innovation happens here


# The JavaScript Story


http://www.maztek.com/blog/wp-content/uploads/javascript.jpg

# My assumptions
**- on the client side -**

Browser only (HTML5/CSS3)
JavaScript only

**„The browser-based application written in JavaScript becomes the new rich client architecture“**

# Existing JavaScript libs are UI centric

(focus on making life with the DOM easier)

most prominent:

**jquery**

# JavaScript versions of „good old rich client patterns“ begin to appear

(and are highly necessary)

Examples

**backbone.js**

**angular.js**

**ember.js**

**...**

# The Server Side

# My assumptions

**- server side languages -**

many different languages in use

choose the right language for the right job

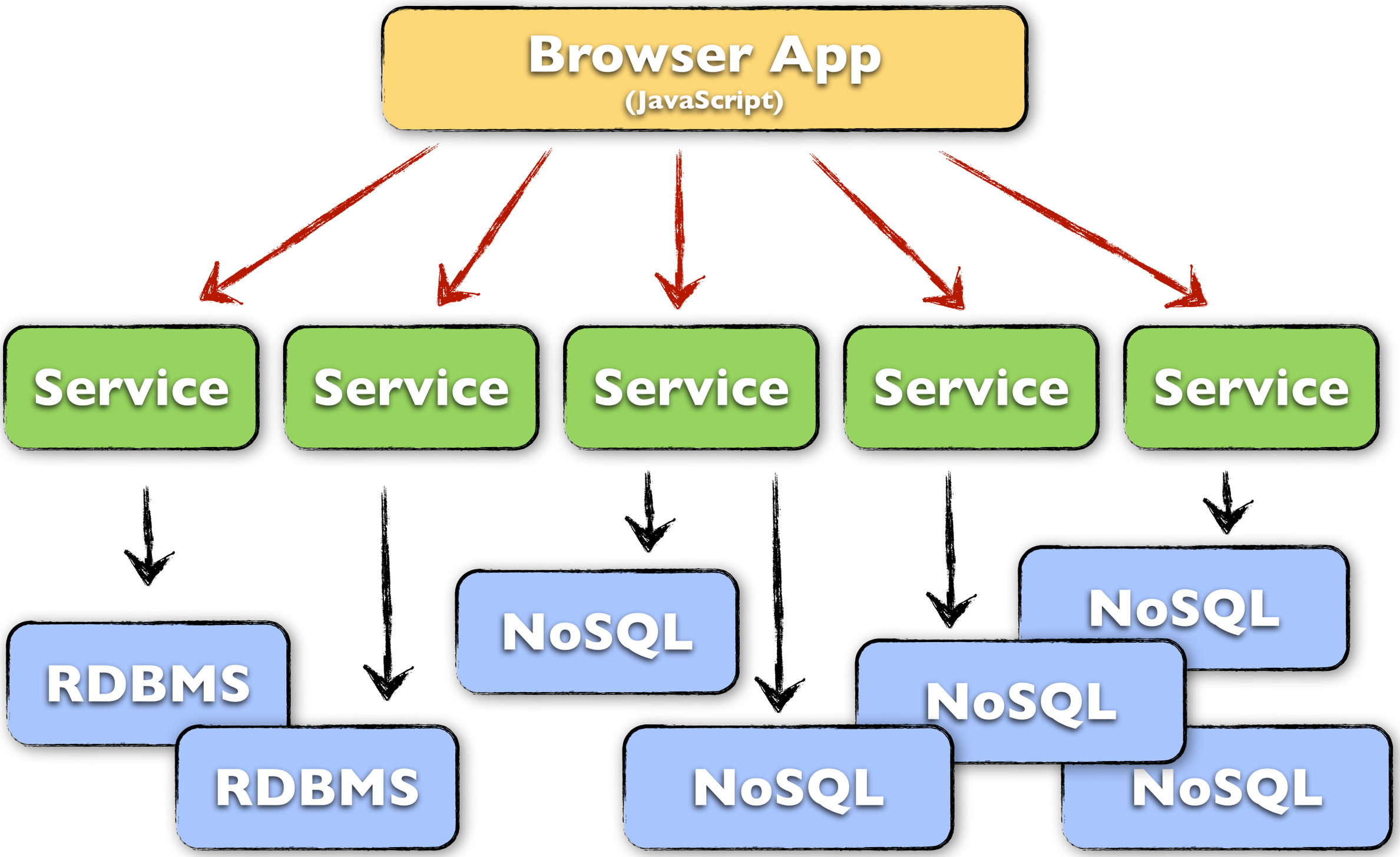
don‘t use a new language for fun

# My assumptions

**- data storage -**

more and more data (big data)
different storage techniques combined
(rdbms, nosql, graph databases)
scalability is important

# The landscape



**Browser App**
**(JavaScript)**

**rich client application written in JavaScript**
(a lot bigger than what we do today in JavaScript within the browser)

Servic

ervice

RDBMS

RDBMS

NoSQL

NoSQL

NoSQL

NoSQL

NoSQL

**services are provided by a PaaS**
**or are hand-written** (in a language of your choice)
**this is where Spring is really powerful**
**ready to run „in the cloud“ (scalability)**
**(no client-side rendering or logic)**

Service | Service | Service | Service | Service

RDBMS | RDBMS | NoSQL | NoSQL | NoSQL | NoSQL | NoSQL

Browser App

Serv

**RDBMS and NoSQL datastores are provided by the PaaS**
**the PaaS takes care of scalability**
**access managed by Spring** (e.g. Spring Data)

RDBMS

RDBMS

NoSQL

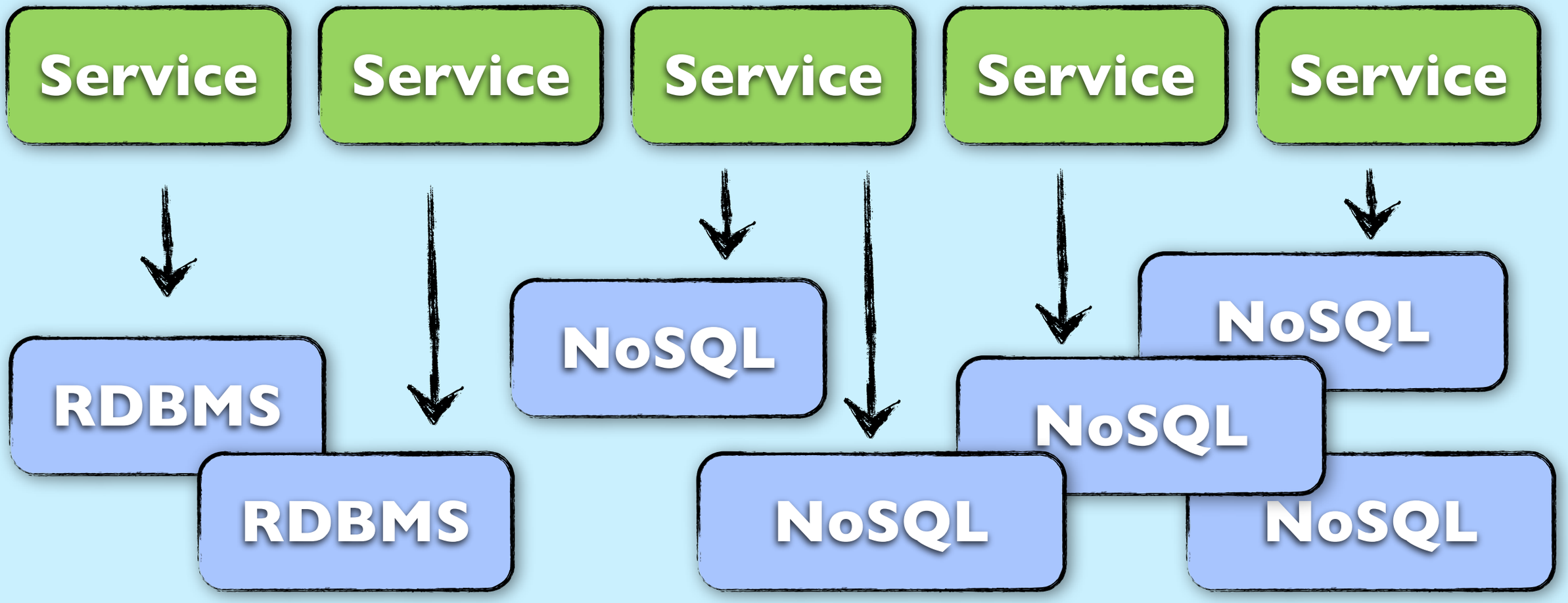
NoSQL

NoSQL

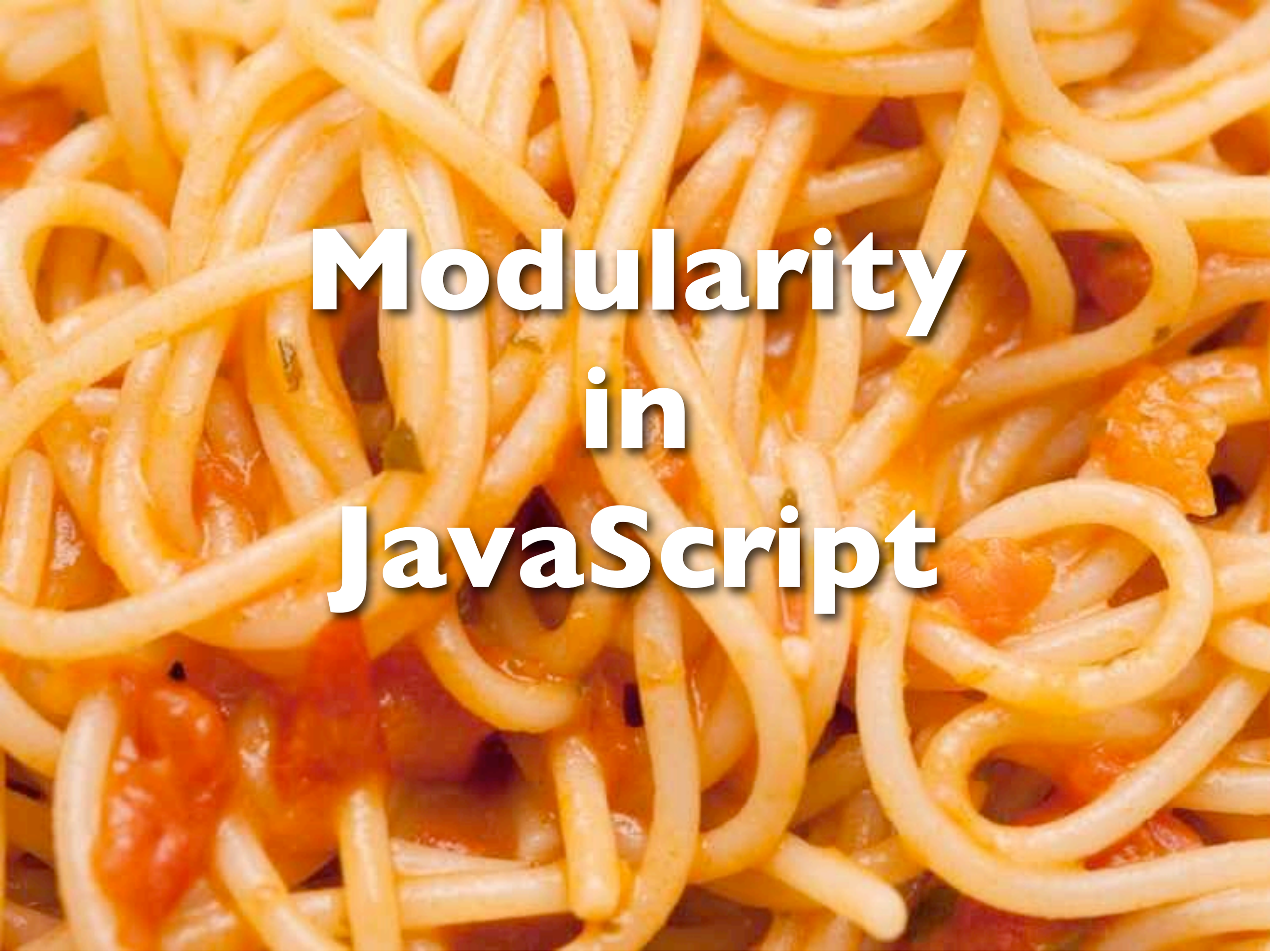
NoSQL

NoSQL

# Running in the cloud
## (on a PaaS)



# The Challenges


Modularity
in
JavaScript

**AMD**
(asynchronous module definition)

**wire.js**
(Dependency Injection for JavaScript)

**Micro Services for JavaScript**
(OSGi services written in JavaScript)

# More Challenges

**offline**
**cloud-ready services**
**define good APIs**
**versioned APIs**
**TDD for JavaScript**

# more information

Adrian Colyer on Application Development in the Cloud Era
http://www.youtube.com/watch?v=axOPJbrIjkY

Example app using Spring for providing RESTful APIs and JavaScript for a rich client and mobile app
https://github.com/SpringSource/html5expense

Asynchronous Module Definition for JavaScript (AMD)
https://github.com/amdjs/amdjs-api
http://requirejs.org/docs/whyamd.html

wire.js
https://github.com/cujojs/wire

hello world with wire.js
https://github.com/briancavalier/hello-wire.js

more advanced example for wire.js
https://github.com/briancavalier/piratescript

Cloud Foundry PaaS
http://www.cloudfoundry.com
http://www.cloudfoundry.org

# Q&A

**and thank you for your attention**

Martin Lippert, VMware
mlippert@vmware.com, @martinlippert