JSON: JavaScript Object Notation

JSON
Google Protocol Buffers
JSON and REST

Instructor: Shel Finkelstein

Not in our textbook.

Important Notices

- Final Exam is on Wednesday, March 22, noon-3pm in our usual classroom.
  - Final is Cumulative, with more focus on second half of quarter.
    - Final may include material in Lecture 14 (XML); no JSON or NOSQL.
  - Please bring a red Scantron sheet (ParSCORE form number f-1712) sold at the Bookstore, and #2 pencils. You'll answer multiple choice on Scantron.
    - Ink and #3 pencils don't work.
  - You may bring a single two-sided 8.5" x 11" sheet of paper with as much info written (or printed) on it as you can fit and read unassisted, just as for the Midterm.
    - No sharing of these sheets will be permitted.
  - You must show your UCSC id when you turn in your Final and Scantron.
  - The Final from Fall 2016 and Answers to that Final have been posted on Piazza (Resources → Exams).

More Important Notices

- Gradiance Assignment #5 (on Functional Dependencies and Normal Forms) is due by Friday, March 17, 11:59pm.
- There will be Lab Sections during the last week of classes.
  - These Lab Sections are an opportunity go over the answers to Lab4 and other Labs, or ask questions about other course material.
  - Solution to Lab4 was posted on Piazza on Monday, March 13.
- Online course evaluations began Monday, March 5, and run through Sunday, March 19 at 11:59pm.
  - Instructors are not able to identify individual responses.
  - Constructive responses help improve future courses.

JSON: The Basics

Jeff Fox
@jfox015

Built in Fairfield County:
Front End Developers Meetup
Tues. May 14, 2013

Overview

- What is JSON?
- Comparisons with XML
- Syntax
- Data Types
- Usage

What is JSON?

JSON is…

- A lightweight text based data-interchange format
- Completely language independent
- Based on a subset of the JavaScript Programming Language
- Easy to understand, manipulate and generate

JSON is NOT…

- Overly Complex
- A “document” format
- A markup language
- A programming language

Why use JSON?

- Straightforward syntax
- Easy to create and manipulate
- Can be natively parsed in JavaScript using **eval()**
- Supported by all major JavaScript frameworks
- Supported by most backend technologies

JSON vs. XML

Much Like XML

- Plain text formats
- “Self-describing“ (human readable)
- Hierarchical (Values can contain lists of objects or values)

Not Like XML

- Lighter and faster than XML
- JSON uses typed objects. All XML values are type-less strings and must be parsed at runtime.
- Less syntax, no semantics
- Properties are immediately accessible to JavaScript code

Knocks against JSON

- Lack of namespaces
- No inherent validation (XML has DTD and templates, but there is JSONlint)
- Not extensible
- It’s basically just ***not*** XML

Syntax

JSON Object Syntax

- Unordered sets of name/value pairs
- Begins with **{** (left brace)
- Ends with **}** (right brace)
- Each name is followed by **:** (colon)
- Name/value pairs are separated by **,** (comma)

JSON Example

```
var employeeData = {
  "employee_id": 1234567,
  "name": "Jeff Fox",
  "hire_date": "1/1/2013",
  "location": "Norwalk, CT",
  "consultant": false
};
```

Arrays in JSON

- An ordered collection of values
- Begins with **[** (left bracket)
- Ends with **]** (right bracket)
- Name/value pairs are separated by **,** (comma)

JSON Array Example

```
var employeeData = {
  "employee_id": 1236937,
  "name": "Jeff Fox",
  "hire_date": "1/1/2013",
  "location": "Norwalk, CT",
  "consultant": false,
   "random_nums": [ 24,65,12,94 ]
};
```

Data Types

Data Types: Strings

- Sequence of zero or more Unicode characters
- Wrapped in "double quotes"
- Backslash escapement

Data Types: Numbers

- Integer
- Real
- Scientific
- No octal or hex
- No NaN (Not a Number) or Infinity – Use **null** instead.

Let's end with an example JSON

```
{     "firstName": "John",
      "lastName": "Smith",
      "age": 25,
      "address":     {
             "streetAddress": "21 2nd Street",
             "city": "New York",
             "state": "NY",
             "postalCode": "10021"
             },
      "phoneNumber": [
                 {
                 "type": "home",
                 "number": "212 555-1234"
                 },
                 {
                 "type": "fax",
                 "number": "646 555-4567"
                 }
             ]
}
```

The same Example in XML

```
<Object>
<Property><Key>firstName</Key> <String>John</String></Property>
<Property><Key>lastName</Key> <String>Smith</String></Property>
<Property><Key>age</Key> <Number>25</Number></Property>
<Property><Key>address</Key> <Object> <Property><Key>streetAddress</Key>
<String>21 2nd Street</String></Property>
<Property><Key>city</Key> <String>New York</String></Property>
<Property><Key>state</Key> <String>NY</String></Property>
<Property><Key>postalCode</Key> <String>10021</String></Property>
</Object>
</Property> <Property><Key>phoneNumber</Key>
<Array> <Object> <Property><Key>type</Key> <String>home</String></Property>
<Property><Key>number</Key> <String>212 555-1234</String></Property></Object>
<Object>
<Property><Key>type</Key> <String>fax</String></Property> <Property><Key>number</
Key> <String>646 555-4567</String></Property> </Object> </Array>
</Property>
</Object>
```

JSON Usage

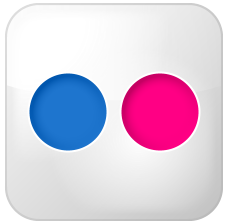
How & When to use JSON

- Transfer data to and from a server
- Perform asynchronous data calls without requiring a page refresh
- Working with data stores
- Compile and save form or user data for local storage

Where is JSON used today?

- Anywhere and everywhere **(even in 2013, much more now)**!





And many, many more!

Resources

- Simple Demo on Github: https://github.com/jfox015/BIFC-Simple-JSON-Demo
- Another JSON Tutorial: http://iviewsource.com/codingtutorials/getting-started-with-javascript-object-notation-json-for-absolute-beginners/
- JSON.org: http://www.json.org/

Google Protocol Buffers

from

F1: A Distributed SQL Database That Scales

Protocol Buffer Column Types

Google

Protocol Buffers

- Structured data types with optional and repeated fields
- Open-sourced by Google, APIs in several languages

Column data types are mostly Protocol Buffers

- Stored like blobs in Spanner
- SQL syntax extensions for reading nested fields
- Coarser schema with fewer tables - inlined objects instead

Why useful?

- Protocol Buffers pervasive at Google -> no impedance mismatch
- Simplified schema and code - apps use the same objects
  - Don't need foreign keys or joins if data is inlined

SQL on Protocol Buffers

Google

```
SELECT CustomerId, Whitelist
FROM Customer
```

CustomerId	Whitelist

123	feature {
	  feature_id: 18
	  status: ENABLED
	}
	feature {
	  feature_id: 269
	  status: ENABLED
	}
	feature {
	  feature_id: 302
	  status: ENABLED
	}

```
SELECT CustomerId, f.*
FROM Customer c
PROTO JOIN c.Whitelist.feature f
WHERE f.feature_id IN (269, 302)
  AND f.status = 'ENABLED'
```

CustomerId	feature_id	status
123	**269**	**ENABLED**
123	302	ENABLED

JSON and REST

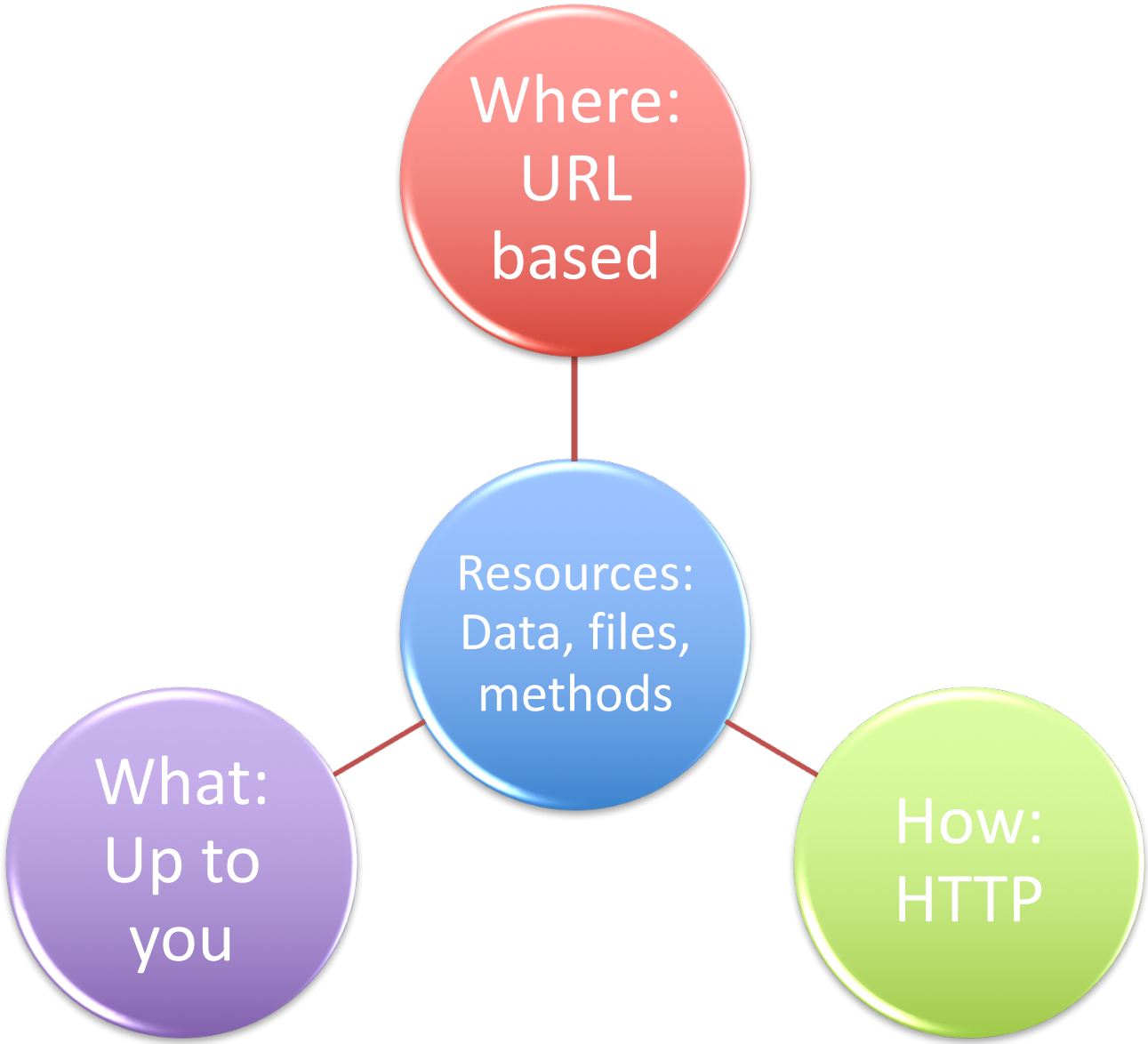
Robert MacLean
Barone, Budge & Dominick
2009

The New Kids on the Data Block
(new in 2009, prevalent now)

REST

- **What does it stand for?:**
- **R**epresentational **S**tat**e** **T**ransfer
- **What Is it?**
- A **style** of software architecture for distributed systems
- **Who/Where/When?**
- Came about in 2000 doctoral dissertation of Roy Fielding – but it's been used for much longer

REST – Core Principle



REST – Where/How: Simple Example

- Premise:
  - Data in a table could be a resource we want to read
- Database server called *bbddb01*
- Database called *northwind*
- Table called *users*

- **http://bbddb01/northwind/users**

What, What, What?

- What type of content you return is up to you.
- Compare to SOAP where you must return XML.
- Most common are XML or JSON. You could return complex types like a picture.

REST – Is it used?

- Web sites are RESTful
- RSS is RESTful
- Twitter, Flickr and Amazon expose data using REST

- Some things are “accidentally RESTful” in that they offer limited support.

Real Life: Flickr API

- Resource: Photos
- Where:
  - http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}.jpg
  - http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}_[mstb].jpg
  - http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{o-secret}_o.(jpg|gif|png)
- What: JPEG, GIF or PNG (defined in the URL)
- **http://farm1.static.flickr.com/2/1418878_1e92283336_m.jpg**

REST – Methods

- HTTP Methods are a key corner stone in REST.
- They define the action to be taken with a URL.
- Proper RESTful services expose all four – “accidental” expose less.
- Nothing stopping you doing some Mix & Match
  - ❖ Some URL’s offering all of them and others a limited set

What are the four methods and what should they do?

REST	CRUD (Create, Read, Update, Delete)
POST	Create
GET	Read
PUT	Update or Create
DELETE	Delete

REST – Methods Example

http://bbddb01/northwind/users[firstname="rob%"]

+ POST = Error
+ GET = Returns everyone who begins with rob
+ PUT = Error
+ DELETE = Deletes everyone who begins with rob

http://bbddb01/northwind/users

& we add some input data

+ POST = Creates a new user
+ GET = Returns everyone who meets criteria
+ PUT = Creates/Updates a user (based on data)
+ DELETE = Deletes everyone who meets criteria

REST – Methods Example

http://bbddb01/northwind/users[firstname="rob%"]

- **+ POST = Error**
- **+ PUT = Error**

What would the error be?

HTTP 400 or 500 errors are normally used to indicate problems – same as websites

REST – Commands

You can associate commands with a resource.

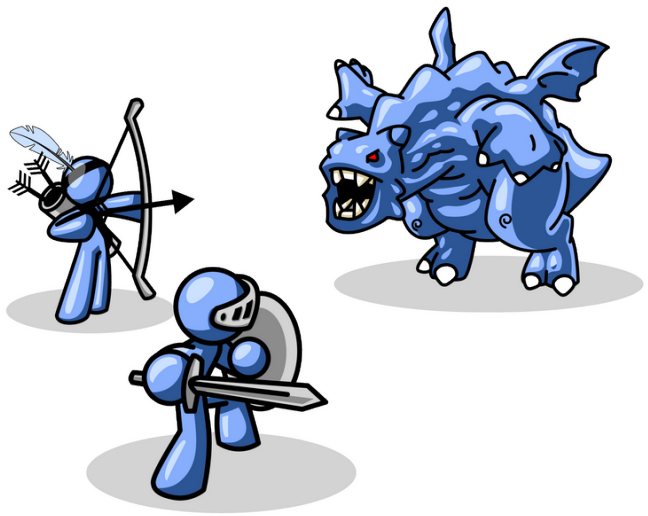
Commands can replace the need for using HTTP methods and can provide a more familiar way of dealing with data.

Example:

```
userResource = new Resource('http://example.com/users/001')
userResource.delete()
```

FIGHT:
REST vs. SOAP

Comparing apples and oranges



REST vs. SOAP – Part I: Technology

REST	SOAP
A STYLE	A Standard
Proper REST: Transport must be HTTP/HTTPS	Normally transport is HTTP/HTTPS but can be something else
Response data is normally transmitted as JSON/XML, can be something else. ❖On average the lighter of the two as does not have SOAP header overhead	Response data is transmitted as XML
Request is transmitted as URI ❖Exceptionally light compared to web services ❖Limit on how long it can be ❖Can use input fields	Request is transmitted as XML
Analysis of method and URI indicate intent	Must analyse message payload to understand intent
…	WS* initiatives to improve problems like compression or security

REST vs. SOAP – Part II: Languages

REST	t
Easy to be called from JavaScript	JavaScript can call SOAP but it is hard, and not very elegant.
If JSON is returned it is very powerful (keep this in mind)	JavaScript parsing XML is slow and the methods differ from browser to browser.
C# (Visual Studio) parsing of REST means using HttpWebRequest and parsing the results (string/xml) or normal service consumption (.NET 3.5 SP 1 and later).	C# (Visual Studio) makes consuming SOAP very easy and provides nice object models to work with.
C# version 4 should make this easier thanks to new dynamic methods.	...
There are 3rd party add-on's for parsing JSON with C# so that may make it easier.	...

REST vs. SOAP – Part III: Tools

REST	SOAP
Basic support for REST in BizTalk	BizTalk and SOAP are made to be together.
WCF can consume REST.	WCF can consume SOAP.
WCF can serve REST with a bit of tweaking.	WCF can server SOAP.
The new routing feature in ASP.NET 3.5 SP1 makes building a RESTful service easy.	...

FAQ about Security?

Are RESTful services secure?

- It's a style, not a technology so that depends on how you implement it.

Are you open to SQL injection attacks?

- When you look at *http://bbddb01/northwind/users[firstname="rob%"]*, you may think so but you shouldn't be. Because:
  1) The parameter shouldn't be SQL.
  2) If it is SQL, why are you not filtering it?
  3) Remember the old rule: Do not trust user input.

FAQ about Security?

- How can I do authentication?
- It’s built on HTTP, so everything you have for authentication in HTTP is available
- PLUS
- You could encode your authentication requirements into the input fields

Building a RESTful database with ADO.NET Data Services

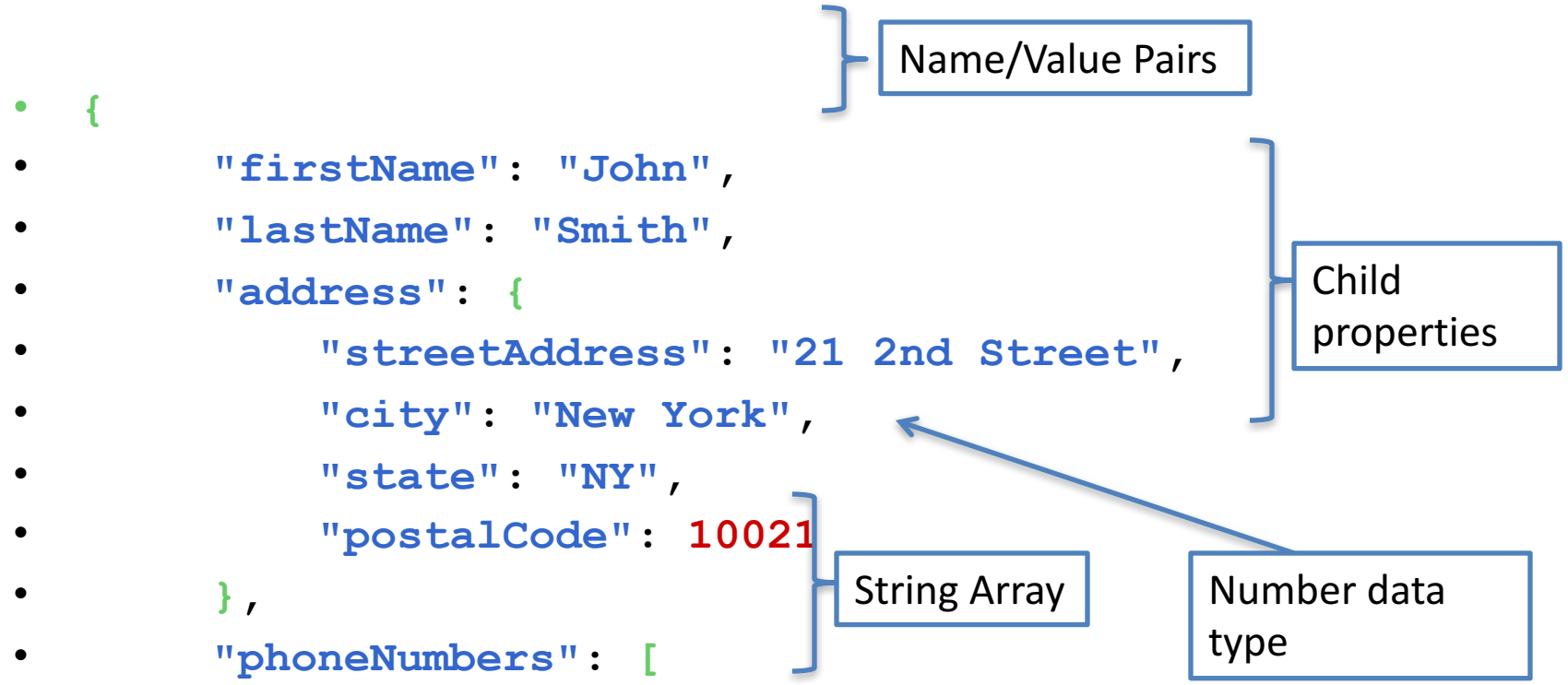
DEMO

JSON – What is it?

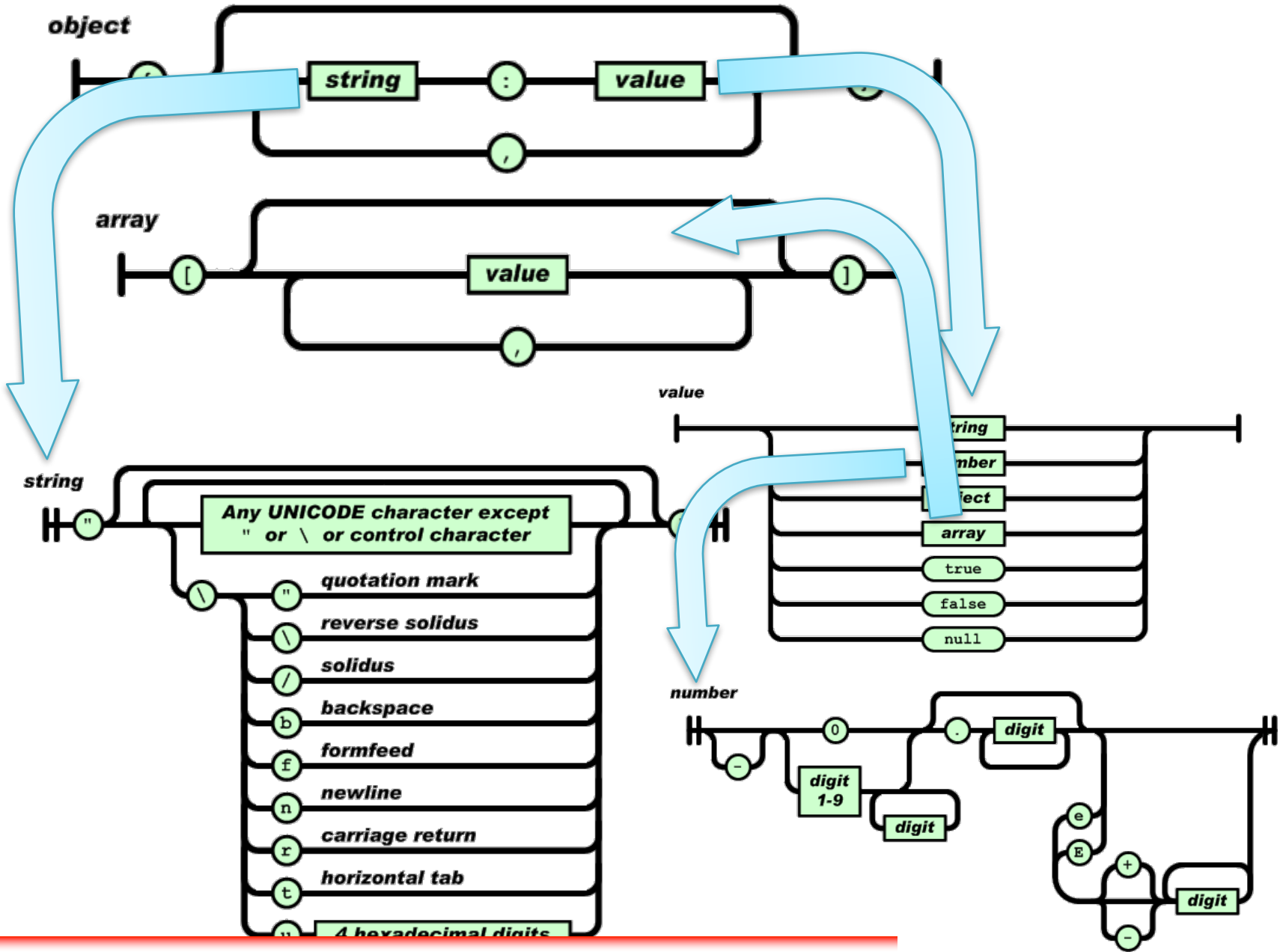
- *"JSON (JavaScript Object Notation) is a **lightweight data-interchange format**. It is easy for humans to read and write. It is easy for machines to parse and generate"* – JSON.org

- Importantly: JSON is a subset of JavaScript

JSON – What does it look like?



JSON Data Structures



JSON Basics

Demo

ADO.NET Data Services & JSON

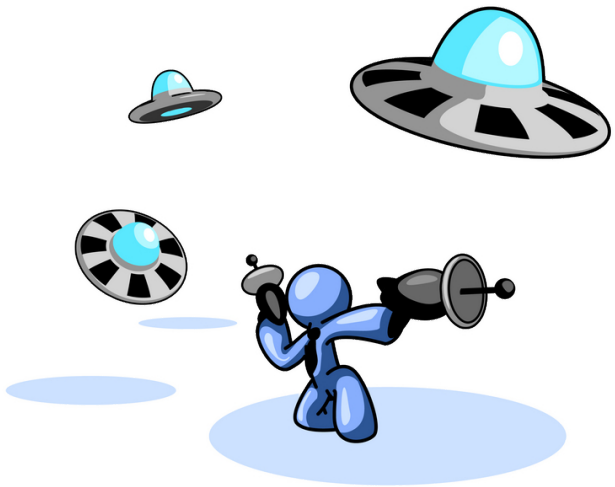
Demo

More on jQuery?

- What: jQuery TR
- Who: Rein
- Where: BB&D Collab
- When: 20th May 2009

FIGHT:
JSON vs XML

Aren’t they the same?



JSON vs. XML

JSON	XML
Data Structure	Data Structure
No validation system	XSD
No namespaces	Has namespaces (can use multiples)
Parsing is just an eval •Fast •Security issues	Parsing requires XML document parsing using things like XPath
In JavaScript you can work with objects – runtime evaluation of types	In JavaScript you can work with strings – may require additional parsing
Security: Eval() means that if the source is not trusted anything could be put into it. Libraries exist to make parsing safe(r)	Security: XML is text/parsing – not code execution.

JSON vs. XML which to use?

- Scenario 1: You have a website (say Twitter.com) and you want to expose a public API to build apps.

Issue	JSON	XML
The public will be parsing data in. You must make it secure.	Run checks against the data in the object to make sure it's secure. You are working on objects so you must also check for potential code access issues.	Run checks against the data to make sure it's secure.
Data must be in a specific format.	Build something that parses the objects.	XML Schema

JSON vs. XML which to use?

- Scenario 2: You have a website (say gmail.com) and your front end needs to show entries from a mailbox, but needs to be dynamic and so you will use a lot of JavaScript.

Issue	JSON	XML
Your in house developers know objects and would like to use them.	JSON is JavaScript objects.	Write JavaScript to parse the XML to objects.
The site is secure but you worry about people checking the page source.	You page has JavaScript in it and (maybe) code which communicates with a private backend server. No major issues.	You page has JavaScript in it and (maybe) code which communicates with a private backend server. No major issues.

JSON vs. XML

- **Which of them should you use?**
- Use Both – They both have strengths and weaknesses and you need to identify when one is stronger than the other.

Question and Answers