



`<?xml?> + {"json"}`  
in

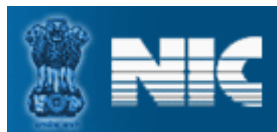
XForms

- ❖ Products & Services:
  - ❑ XSLTForms, tXs, html2xml, ...
  - ❑ Web Apps, Support & Extensions
  - ❑ Trainings

- ❖ Customers:



- ❖ Organizations/Companies just using/looking at?



...

# Summary

- XForms Context
- JSON Objects Internal Storage In Browsers
- JSONP Support
- The Wikipedia Search Demo

# XForms Context

- MVC Design
- XPath in XForms
- Server Exchanges
- JSON vs. XML for browsers
- Other Data Formats

# XForms Context

## MVC Design

- models: instances, bindings, submissions
- controls: input, output, upload, range, trigger,...
- events: xforms-value-changed, ev:DOMActivate,...
- actions: dispatch, send, refresh,...

# XForms Context

## XPath in XForms

- XForms 1.1: XPath 1.0 + extra functions (if statement, encryption, dates, credit cards,...)
- Used for nodeset association, calculations,
- Evaluation according to current context (document element by default)
- Dependencies to be maintained for performance at refresh

# XForms Context

## Server Exchanges

- XRX with Native XML Databases but not only
- AJAX-based exchanges
- Only XML serialization and flat GET parameters list in XForms 1.1

# XForms Context

## JSON vs. XML for browsers

- Lighter: no namespace, no element vs. attribute
- Shorter: minimal closing tags (})
- Richer: names, arrays and datatypes
- Natively evaluated and stored by Javascript
- Query support???



# XForms Context

## Other Data Formats

- vCard
- CSV with or without titles
- Formatted log files
- ...
  
- Efficient XML Interchange

# JSON Objects Internal Storage

- Constraints
- Elements, Attributes and Namespaces
- Extra Document Element
- JSON Names
- JSON Datatypes
- JSON Named Arrays
- JSON Anonymous Arrays
- XPath Engine Proposed Enhancements

# JSON Objects Internal Storage Constraints

- Reversibility:
  - from JSON to XML and back
- XPath Full Support:
  - XPath extension functions
  - Minimal number of extra elements
  - Array [] notation almost preserved
- XML Schema Conformance:
  - xsi:type
  - xsi:nil
  - xsd:maxOccurs

# JSON Objects Internal Storage

## Elements, Attributes and Namespaces

### General Principles:

- JSON Values:  
Elements within the empty namespace
- Metadata (datatypes, structures):  
Attributes within a specific namespace
- JSON Structures (when required):  
Elements within a specific namespace

# JSON Objects Internal Storage

## Extra Document Element

### Proposed Rule #1:

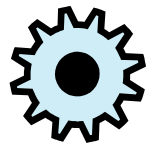
Add a Document Element for each JSON object such as **exml:anonymous**.

# JSON Objects Internal Storage

## Extra Document Element

```
{  
  a: "stringA",  
  b: {  
    c: "stringC",  
    d: "stringD"  
  }  
}
```

```
<exml:anonymous>  
  <a>stringA</a>  
  <b>  
    <c>stringC</c>  
    <d>stringD</d>  
  </b>  
</exml:anonymous>
```



"a" equals 'stringA'  
"b/c" equals 'stringC'  
"b/d" equals 'stringD'

# JSON Objects Internal Storage

## JSON Names

### Proposed Rule #2:

For each JSON name which cannot be used as an XML name, create an element in the empty namespace with a name such as "\_\_\_\_\_" and add an attribute such as **exml:fullname**.

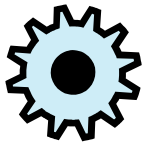
Create two new XPath functions: **fullname()** and **local-fullname()**.

# JSON Objects Internal Storage

## JSON Names

```
{
  "a & b": "A+B",
  "déjà": "already",
  "_____":
    "underscores"
}
```

```
<exml:anonymous>
  <_____ exml:fullname="a & b">A+B</_____>
  <_____ exml:fullname="déjà">already</_____>
  <_____ exml:fullname="_____">underscores</_____>
</exml:anonymous>
```



```
"*[fullname() = 'a & b']"
```

```
equals 'A+B'
```

```
"local-fullname(*[1])"
```

```
equals 'a & b'
```

```
"*[fullname() = 'déjà']"
```

```
equals 'already'
```

```
"local-fullname(*[2])"
```

```
equals 'déjà'
```

```
"_____ [fullname() = '_____' ]"
```

```
equals 'underscores'
```



# JSON Objects Internal Storage

## JSON Datatypes

### Proposed Rule #3:

For each Javascript datatype, convert data into the corresponding XSD datatype and add an **xsi:type** attribute.

JS number becomes XSD double.

JS boolean becomes XSD boolean.

JS Date becomes XSD dateTime.

# JSON Objects Internal Storage

## JSON Datatypes

```
{  
  a: "string"+"A",  
  b: 42,  
  c: new Date(2011,3,26),  
  d: true  
}
```

```
<exml:anonymous>  
  <a>stringA</a>  
  <b xsi:type="xsd:double">42</b>  
  <c xsi:type="xsd:dateTime">2011-03-26T00:00:00Z</c>  
  <d xsi:type="xsd:boolean">true</d>  
</exml:anonymous>
```

# JSON Objects Internal Storage

## JSON Named Arrays

### Proposed Rule #4:

Iterate an element for each named array item and add an attribute such as **exsi:maxOccurs** to indicate it is an array item.

Add an attribute such as **xsi:nil** to distinguish an empty array from an empty element.

Create three new XPath functions: **is-array(*node*)**, **is-non-empty-array(*node*)** and **array-length(*node*)**.

# JSON Objects Internal Storage

## JSON Named Arrays

```
{
  a: ["stringA",
      42],
  b: [],
  c: [""]
}
```

```
<exml:anonymous>
  <a xsi:maxOccurs="unbounded">stringA</a>
  <a xsi:maxOccurs="unbounded"
    xsi:type="xsd:double">42</a>
  <b xsi:maxOccurs="unbounded" xsi:nil="true"/>
  <c xsi:maxOccurs="unbounded"/>
</exml:anonymous>
```



```
"is-array (a) " equals true ()
"a [1] " equals 'stringA'
"a [2] " equals '42'
"array-length (b) " equals 0
"c [1] " equals ''
```

# JSON Objects Internal Storage

## JSON Anonymous Arrays

### Proposed Rule #5:

For each anonymous array item, add an extra element such as **exml:anonymous** with an attribute such as **exsi:maxOccurs** to indicate it is an array item.

# JSON Objects Internal Storage

## JSON Anonymous Arrays

```
[
  ["stringA",
   42],
  []
]
```



```
"*[1]/*[2]"
```

```
equals '42'
```

```
"is-array(*[2])"
```

```
equals true()
```

```
<exml:anonymous>
  <exml:anonymous exsi:maxOccurs="unbounded">
    <exml:anonymous
      exsi:maxOccurs="unbounded">stringA</exml:anonymous>
    <exml:anonymous
      exsi:maxOccurs="unbounded"
      xsi:type="xsd:double">42</exml:anonymous>
  </exml:anonymous>
  <exml:anonymous exsi:maxOccurs="unbounded">
    <exml:anonymous exsi:maxOccurs="unbounded" xsi:nil="true"/>
  </exml:anonymous>
</exml:anonymous>
```

# JSON Objects Internal Storage

## XPath Engine Proposed Enhancements

- ` (backquote) to delimit names
- **name()** and **local-name()** to support all names
- **name()** and **local-name()** to return the empty string for **exml:anonymous**
- Instead of \* for **exml:anonymous**:
  - **/exml:anonymous/** just written **/** ?
  - **exml:anonymous** written **`** ?
  - **exml:anonymous[...]** written **[...]** ?

# JSONP Support

- Request Submission
- Response Processing



# JSONP Support

## Request Submission

- Just construct a **script** element with an **src** attribute containing request parameters + the name of the Javascript callback function
- No cross-domain limitation!

# JSONP Support

## Response Processing

- The generated script contains the call to the callback function with the JSON response as parameter
- The callback function has to convert the JSON response into an XML instance

# The Wikipedia Search Demo



```
http://en.wikipedia.org/w/api.php?  
action=opensearch&search=prague
```



```
[ "prague",  
  [ "Prague",  
    "Prague Spring",  
    "Prague Conservatory",  
    "Prague Ruzyn\u011b Airport",  
    "Prague University",  
    "Prague Castle",  
    "Prague Metro",  
    "Prague-East District",  
    "Prague-West District",  
    "Prague Offensive" ] ]
```

WIKIPEDIA



# The Wikipedia Search Demo

- Two instances:

```
<xf:instance id="isearch"  
mediatype="application/json">
```

```
{  
  action: "opensearch",  
  format: "json",  
  search: ""  
}
```

```
</xf:instance>
```

```
<xf:instance id="iresults"  
mediatype="application/json">
```

```
  []  
</xf:instance>
```

- One submission:

```
<xf:submission method="get" replace="instance"  
instance="iresults" separator="&" validate="false"  
action="http://en.wikipedia.org/w/api.php"/>
```

- One constraint:

```
<xf:bind nodeset="search"  
constraint="instance('iresults')/*[2]/*[upper-case(.) =  
upper-case(current())]"/>
```

# The Wikipedia Search Demo


```
<xf:input id="search" ref="search" incremental="true" delay="500">
  <xf:label>Subject: </xf:label>
  <xf:send ev:event="xforms-value-changed"/>
  <xf:toggle ev:event="DOMFocusIn" case="show-autocompletion" />
</xf:input>
<xf:switch>
  <xf:case id="show-autocompletion">
    <xf:repeat id="results" nodeset="instance('iresults')/*[2]/*[
      is-non-empty-array() and . != '&#x300;']">
      <xf:trigger appearance="minimal">
        <xf:label><xf:output value="."/></xf:label>
        <xf:action ev:event="DOMActivate">
          <xf:setvalue ref="instance('isearch')/search"
            value="current()" />
          <xf:toggle case="hide-autocompletion" />
        </xf:action>
      </xf:trigger>
    </xf:repeat>
  </xf:case>
  <xf:case id="hide-autocompletion" />
</xf:switch>
```

# The Wikipedia Search Demo

## WIKIPEDIA OpenSearch Test Form

Please enter a subject in the following field. The value is not case sensitive but it has to exist in the results of the corresponding search.

Subject :

- Prague
- Prayer
- Praying mantis
- Prairie
- Praetor
- Pratt Institute
- Prahova County
- Practice squad
- Practical joke
- Prague Spring

## WIKIPEDIA OpenSearch Test Form

Please enter a subject in the following field. The value is not case sensitive but it has to exist in the results of the corresponding search.

Subject :

- Prague
- Prague Spring
- Prague Conservatory
- Prague Ruzyně Airport
- Prague Castle
- Prague University
- Prague Metro
- Prague-East District
- Prague Offensive
- Prague-West District

# Conclusion

- XForms instances written in JSON
- Intuitive XPath for JSON
- Simple JSONP use within XForms
- XML+XPath extensions to consider?