# INTERNET ENGINEERING

Basir University, 2020-2021

By: Prof. Dr. Mohammad Hajarian

موسسه آموزش عالی غیردولتی غیرانتفاعی بصیر آبیک

- Session 8

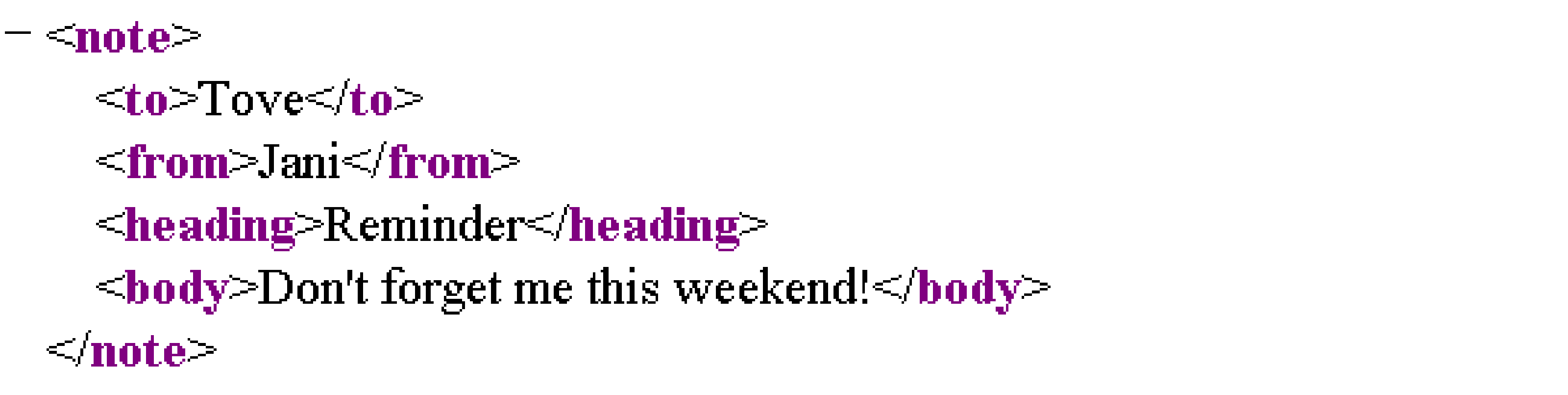
# JSON AND XML

# JSON AND XML

# XML

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive

# XML: Example



```
- <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
```

# XML: Well-formed

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must always be quoted

# XML: Validity

- Valid XML is An XML document that has an associated document type declaration and complies with the constraints expressed in it.
    - Document Type Definition (DTD)
    - XML Schema
    - Relax NG
    - ISO DSDL

# XML: Processing files

- Programming languages and SAX API (Simple API for XML)
- Programming Languages and DOM API
- And Others

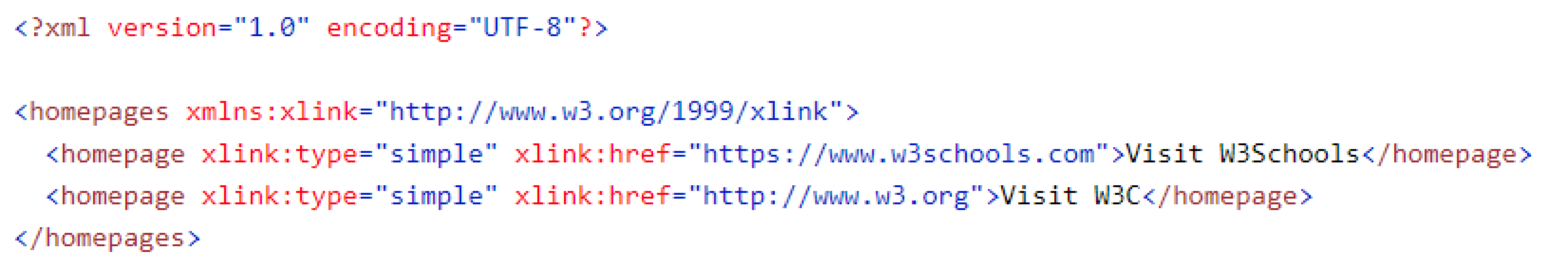
# XML: TRANSFORMATION ENGINES AND FILTERS

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.
- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT stylesheet as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.

# XML: TRANSFORMATION ENGINES AND FILTERS

- **XQuery** is a W3C language for querying, constructing and transforming XML data.
- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.
- **Xlink** - is an XML markup language used for creating hyperlinks in XML documents

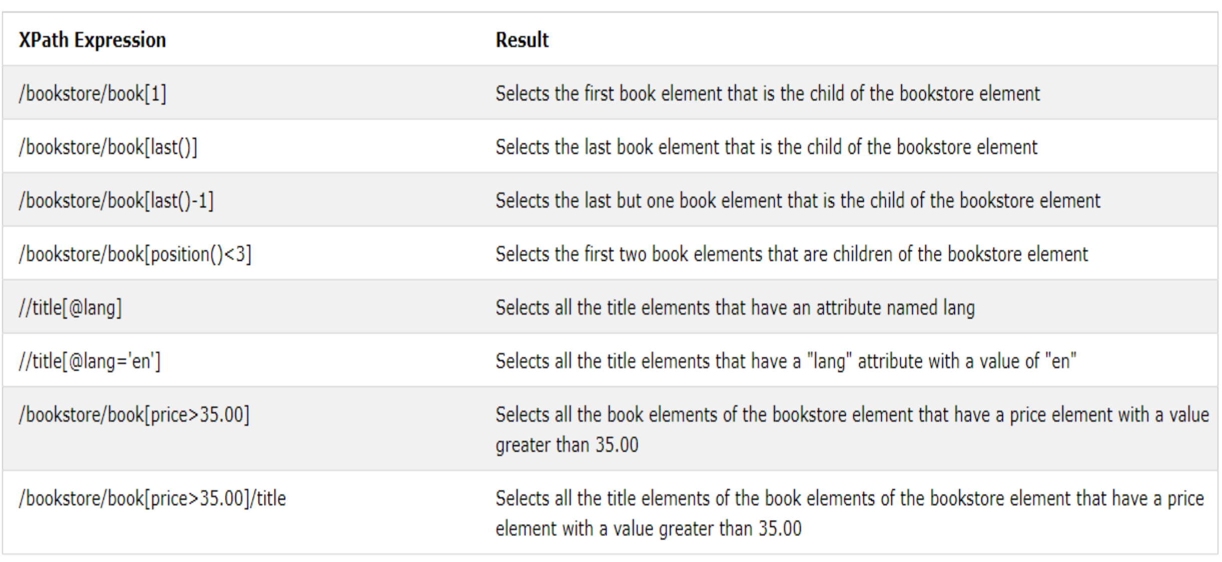
# XLINK SYNTAX



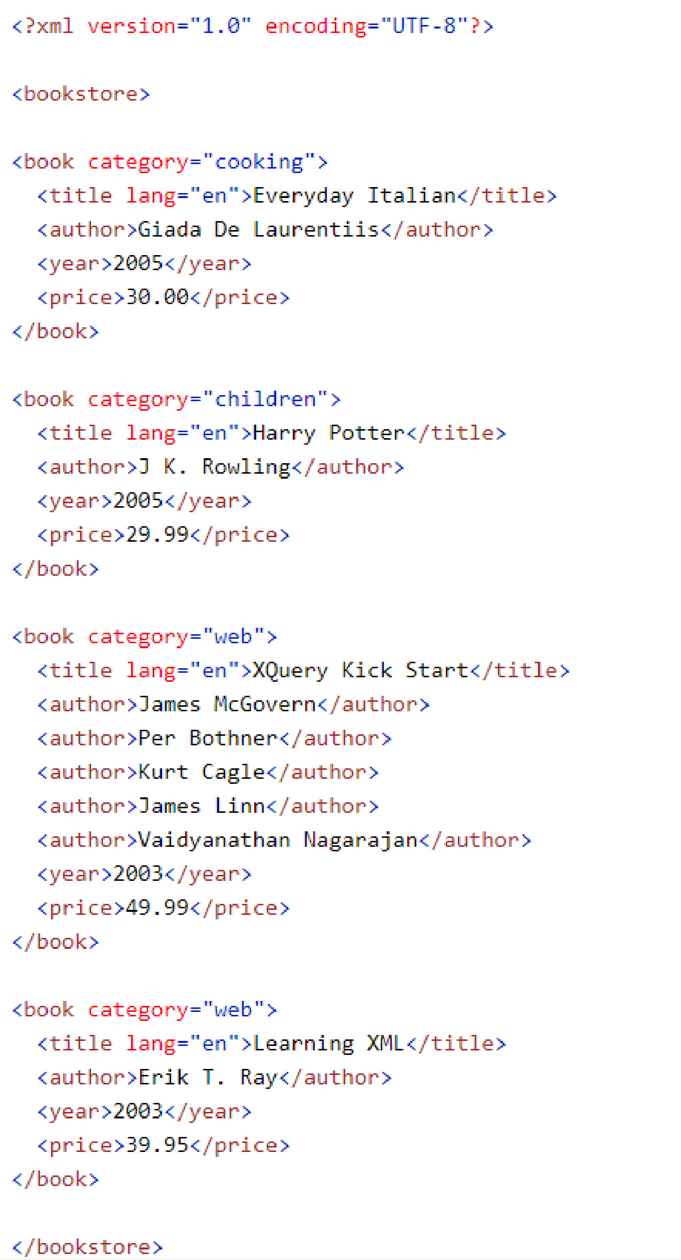
```xml
<?xml version="1.0" encoding="UTF-8"?>

<homepages xmlns:xlink="http://www.w3.org/1999/xlink">
  <homepage xlink:type="simple" xlink:href="https://www.w3schools.com">Visit W3Schools</homepage>
  <homepage xlink:type="simple" xlink:href="http://www.w3.org">Visit W3C</homepage>
</homepages>
```

# XPATH EXAMPLE

- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT and in XQuery
- XPath is a W3C recommendation



| XPath Expression | Result |
|---|---|
| /bookstore/book[1] | Selects the first book element that is the child of the bookstore element |
| /bookstore/book[last()] | Selects the last book element that is the child of the bookstore element |
| /bookstore/book[last()-1] | Selects the last but one book element that is the child of the bookstore element |
| /bookstore/book[position()<3] | Selects the first two book elements that are children of the bookstore element |
| //title[@lang] | Selects all the title elements that have an attribute named lang |
| //title[@lang='en'] | Selects all the title elements that have a "lang" attribute with a value of "en" |
| /bookstore/book[price>35.00] | Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00 |
| /bookstore/book[price>35.00]/title | Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00 |

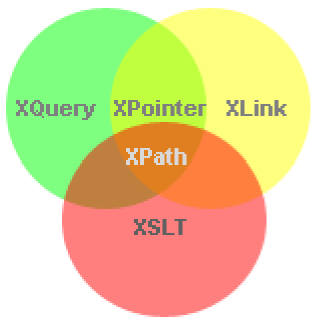


```xml
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="web">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="web">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

# XQUERY

- XQuery is ***the*** language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all major databases
- XQuery is a W3C Recommendation

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```



# XML: Advantages

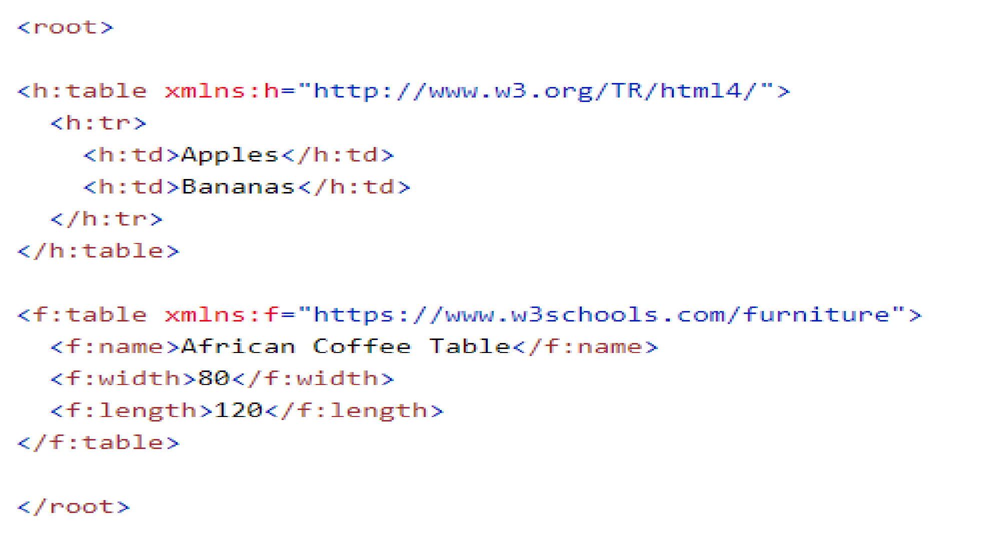
- **XML** provides a basic syntax that can be used to share information between different kinds of computers, different applications, and different organizations.
- **With XML**, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc)
- **XML** provides a gateway for communication between applications, even applications on wildly different systems. As long as applications can share data (through HTTP, file sharing, or another mechanism)
- It supports **Unicode**, allowing almost any information in any written human language to be communicated.
- **It** can represent common computer science data structures: records, lists and trees.
- **Its** self-documenting format describes structure and field names as well as specific values.
- **It** is based on international standards.

# XML: DisAdvantages

- **It** is difficult for the end-user to understand its capabilities.
- **XML** syntax is redundant or large relative to binary representations of similar data, especially with tabular data.
- **The** redundancy may affect application efficiency through higher storage, transmission and processing costs.
- **XML** syntax is verbose, especially for human readers, relative to other alternative 'text-based' data transmission formats.
- **The** hierarchical model for representation is limited in comparison to an object oriented graph.
- **Expressing** overlapping (non-hierarchical) node relationships requires extra effort.
- **XML** namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.

# XML NAMESPACES

- In the example, there will be no conflict because the two <table> elements have different names.

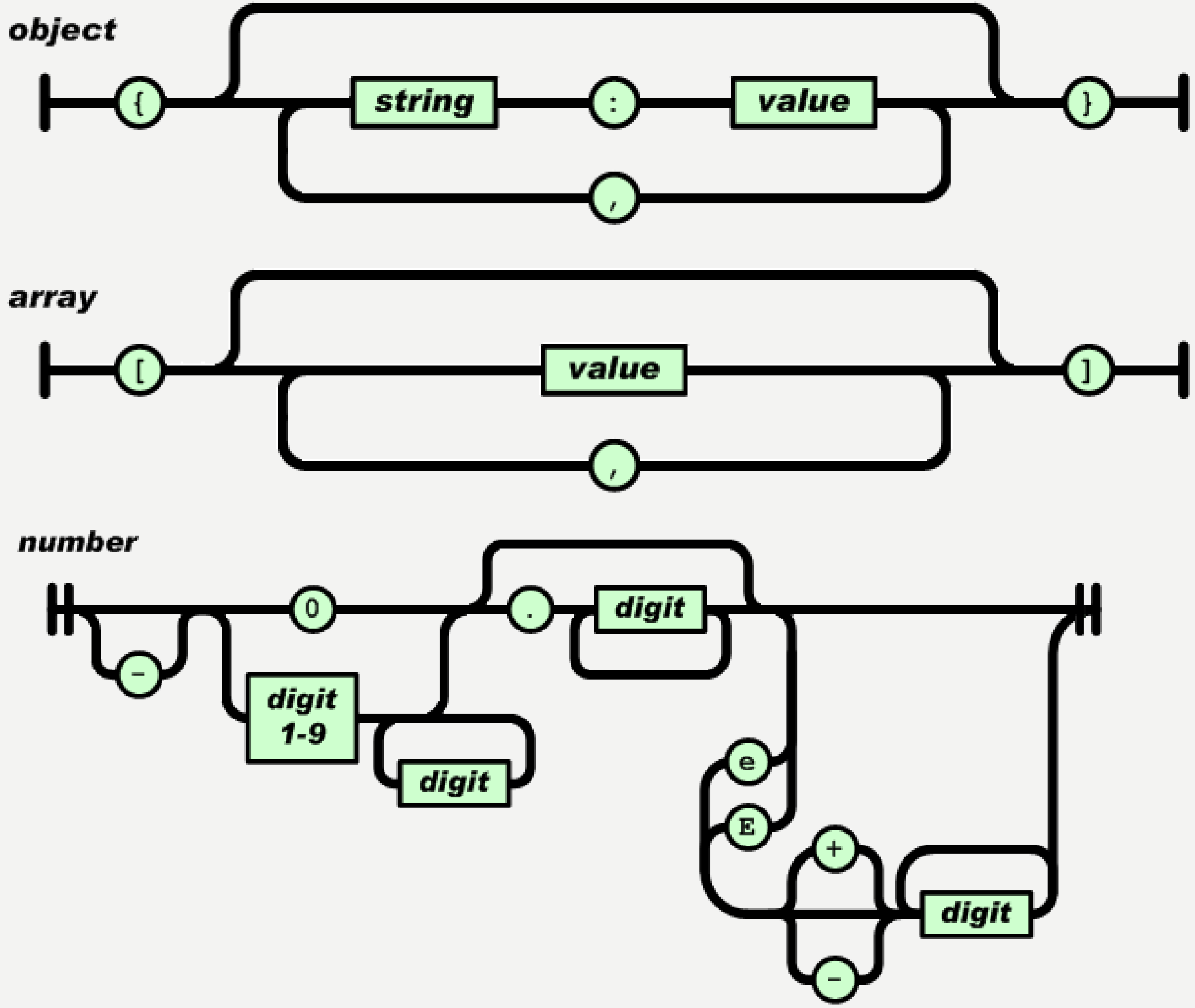


```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

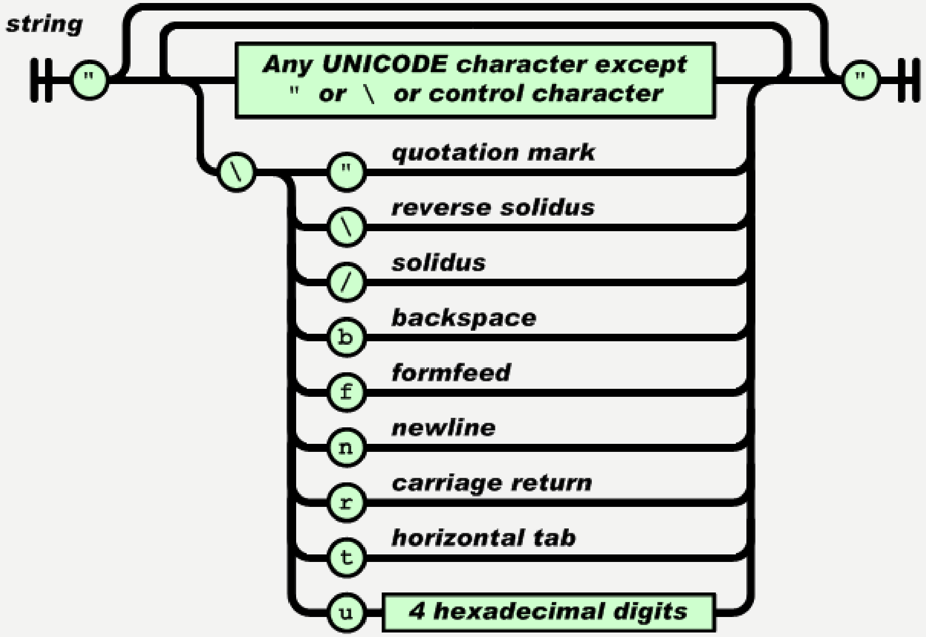
# JSON

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
- **JSON** code is valid JavaScript
- **JSON** is a text format that is completely language independent
- **JSON** is built on two structures:
  - A collection of name/value pairs
  - An ordered list of values
- **JSON** file extension: *.json

# JSON: Object, Array, Number



# JSON: STRING, VALUE



string

"

Any UNICODE character except
" or \ or control character

\

" quotation mark

\ reverse solidus

/ solidus

b backspace

f formfeed

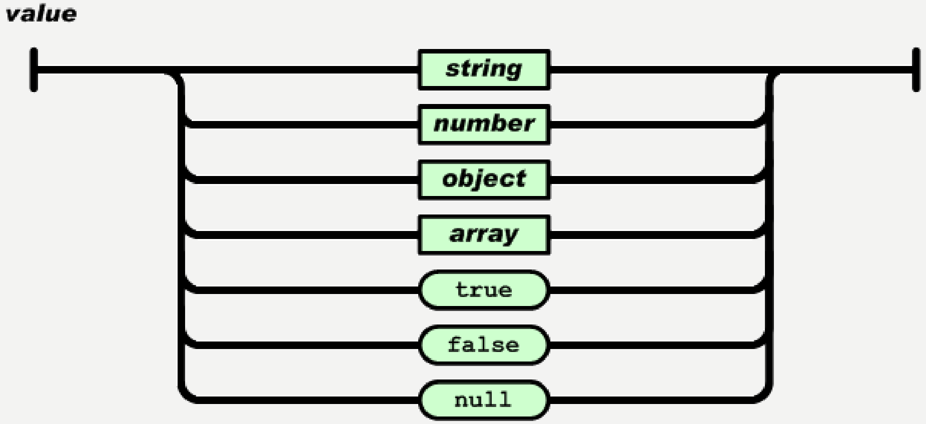
n newline

r carriage return

t horizontal tab

u 4 hexadecimal digits

"



value

string

number

object

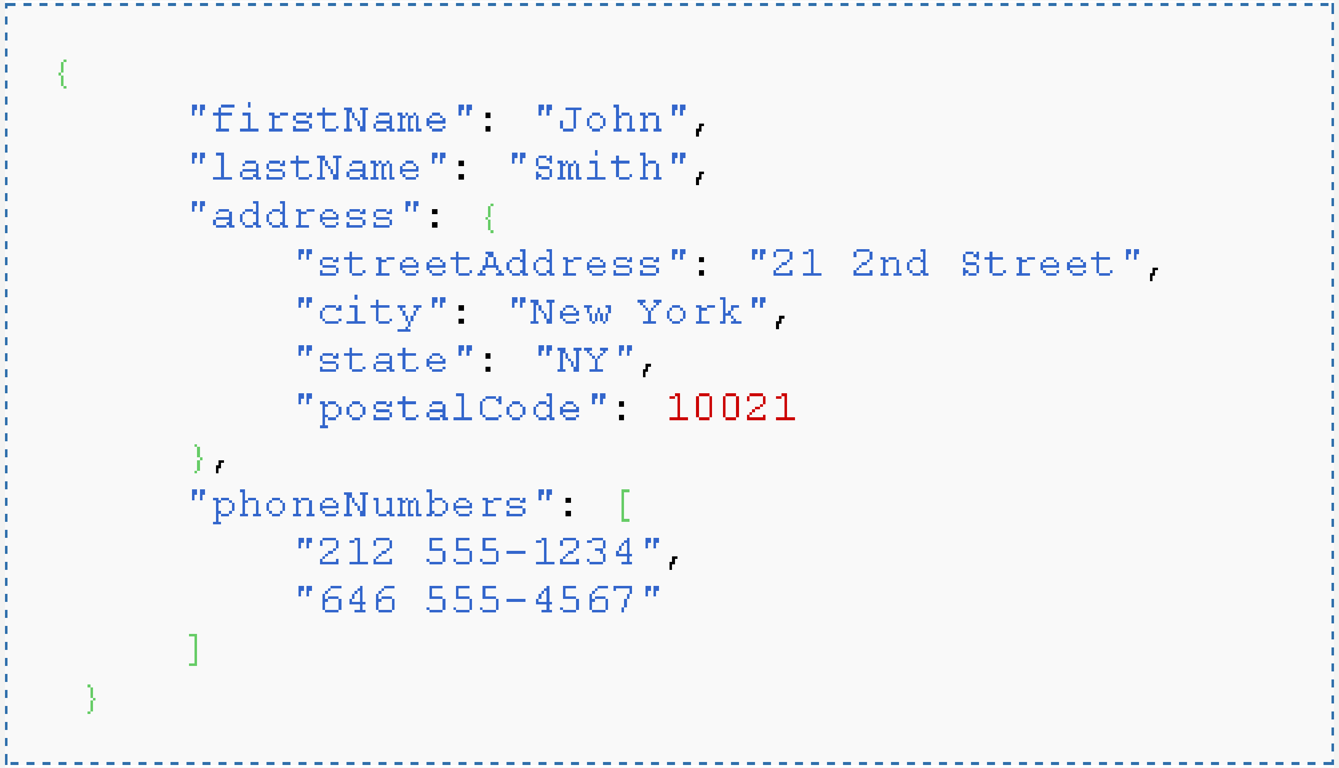
array

true

false

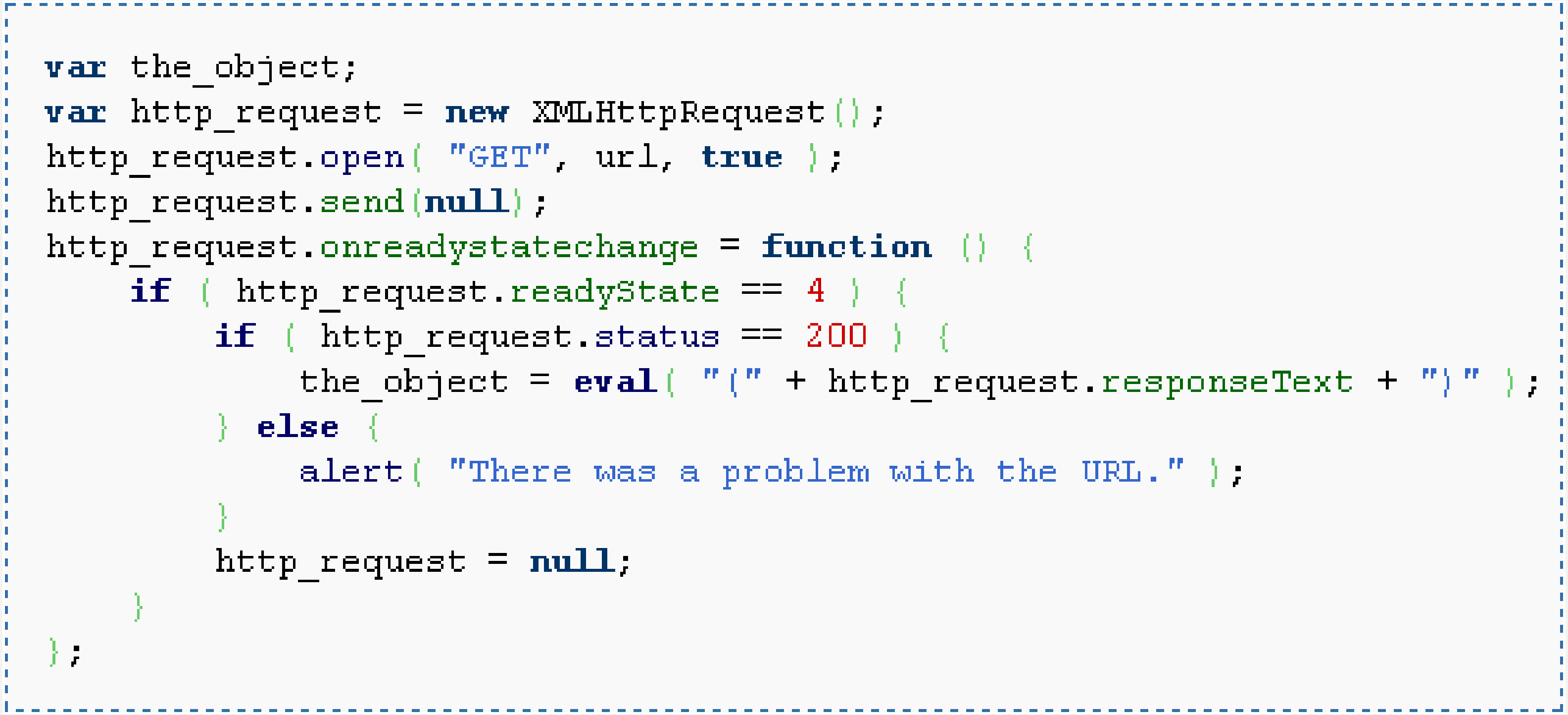
null

# JSON: EXAMPLE



```
{
     "firstName": "John",
     "lastName": "Smith",
     "address": {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": 10021
     },
     "phoneNumbers": [
         "212 555-1234",
         "646 555-4567"
     ]
 }
```

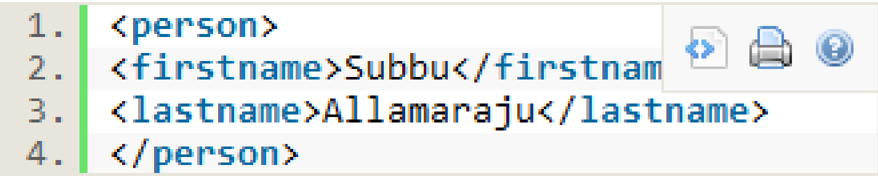
# JSON: IN AJAX



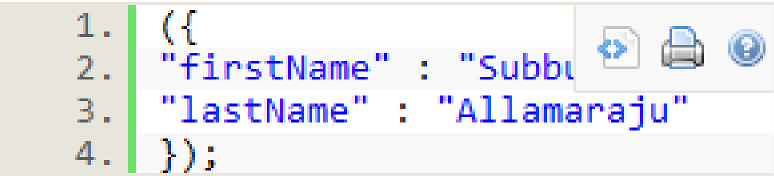
```
var the_object;
var http_request = new XMLHttpRequest();
http_request.open( "GET", url, true );
http_request.send(null);
http_request.onreadystatechange = function () {
    if ( http_request.readyState == 4 ) {
        if ( http_request.status == 200 ) {
            the_object = eval( "(" + http_request.responseText + ")" );
        } else {
            alert( "There was a problem with the URL." );
        }
        http_request = null;
    }
};
```

# Comparing Criteria: Human readability

| XML | JSON |
|---|---|



```
<person>
<firstname>Subbu</firstnam
<lastname>Allamaraju</lastname>
</person>
```



```
({
"firstName" : "Subbu
"lastName" : "Allamaraju"
});
```

Both  formats Human readability are almost in same degree

# Comparing Criteria: Speed

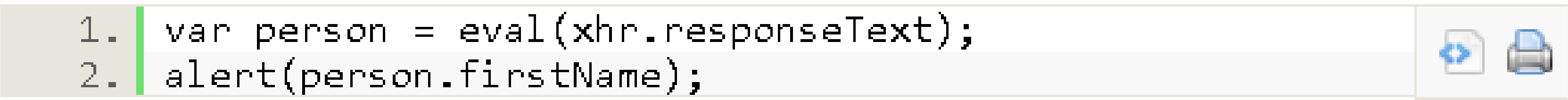
- Each  format has its advantages and disadvantages
- **JSON** is smaller than **XML**
- **JSON** is part of JavaScript code, It needs less time to parse data thank the time for **XML**
- **JSON** is faster than **XML** in **AJAX**

# Comparing Criteria: Fetching data from Server

- **XML** uses XMLHttpRequest to fetch data from Server
- **JSON** can fetch data from Server with and without XMLHttpRequest in AJAX, it also uses **JSONRequest**
- **Speed** fetching data is almost same, but speed parsing fetched data is different

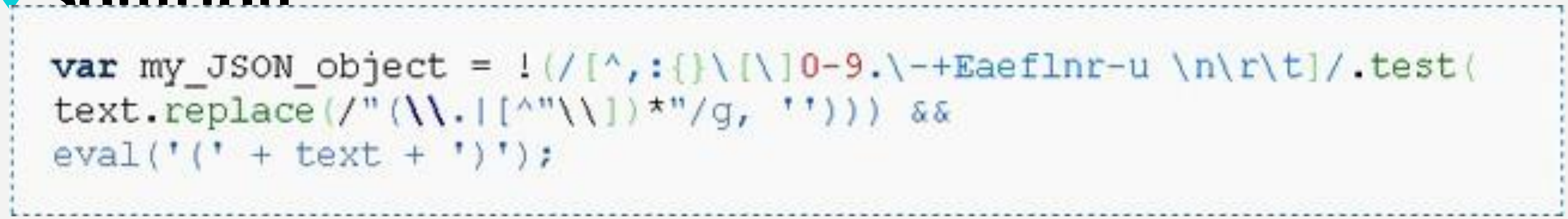
# Comparing Criteria: Security

- **JavaScript eval()** function does not check json code before running it, this is place for code injection attack



```
var person = eval(xhr.responseText);
alert(person.firstName);
```

- **Solution**



```
var my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"(\\.|[^"\\])*"/g, ''))) &&
eval('(' + text + ')');
```

# Comparing Criteria: Others

- JSON is thinner than Xml
- JSON is native to the client(most browsers support IE8,Firefox and others)
- JSON can do cross domain calls that XMLHttpRequest can not
- It is easier to encode / escape in JSON than XML

# CONCLUSION

What to use depends on the applications and your requirements. For data-oriented applications, I prefer JSON to XML due to its simplicity and ease of processing on the client side. XML may be great on the server side, but JSON is definitely easier to deal with on the client side.

# Q/A

- End of Session 8

# THANK YOU!