PHP and JSON Lecture-8

By: Ashiqullah Alizai
Herat University
Computer Science Faculty

Alizai.csf@hotmail.com

What is JSON?

- JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human readable data interchange.
- JSON stands for JavaScript Object Notation.
- JSON is lightweight text-data interchange format
- JSON is "self-describing" and easy to understand
- JSON was designed for human-readable data interchange
- JSON filename extension is .json

Usage of JSON

- JSON is used when writing JavaScript based application which includes browser extension and websites.
- JSON format is used for serializing & transmitting structured data over network connection.
- JSON is primarily used to transmit data between server and web application.
- Web Services and API.s use JSON format to provide public data.
- JSON can be used with modern prog ramming lang uag es.

Characteristics of JSON

- It is easy to read and write JSON.
- JSON is lightweight text based interchange format
- JSON is language independent.

- *JSON uses JavaScript syntax for describing data objects, but JSON is still language and platform independent. JSON parsers and JSON libraries exists for many different programming languages.

Why JSON?

- For AJAX applications, JSON is faster and easier than XML:
- Using XML
  - Fetch an XML document
  - Use the XML DOM to loop through the document, Extract values and store in variables
- Using JSON
  - Fetch a JSON string
  - eval() the JSON string

JSON - SYNTAX

- An *object* is an unordered set of name/value pairs
  - The pairs are enclosed within braces, { }
  - There is a colon between the name and the value
  - Pairs are separated by commas
  - Example: { "name": "html", "years": 5 }
- An *array* is an ordered collection of values
  - The values are enclosed within brackets, [ ]
  - Values are separated by commas
  - Example: ["html", "xml", "css"]

JSON is built on two structures

- A collection of name/value pairs.
- In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
  - e.g.: An object with three properties named "a", "b", and "c"
    - { "a":1,"b":2,"c":3 }
- An ordered list of values.
  - In most languages, this is realized as an array, vector,list, or sequence.
    - e.g.: An array of three integers and one string value
      - [1, 2, 3, "value #4 with"]

JSON Name value pare

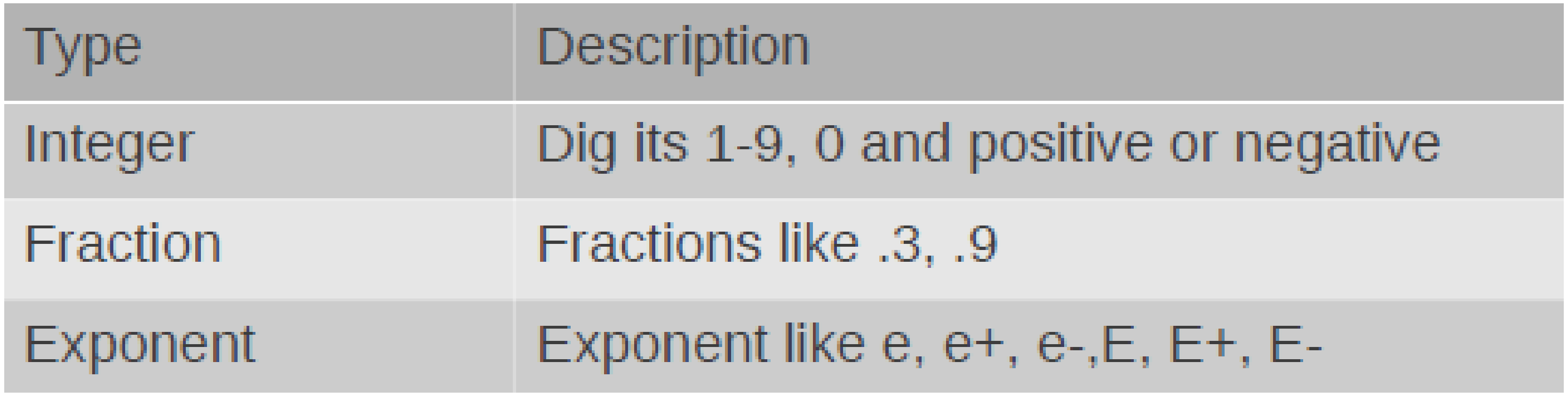
- A *value* can be: A string, a number, true, false, null, an object, or an array
- Values can be nested
- *Strings* are enclosed in double quotes, and can contain the usual assortment of escaped characters
- *Numbers* have the usual C/C++/Java syntax, including exponential (E) notation
- All numbers are decimal--no octal or hexadecimal
- *Whitespace* can be used between any pair of tokens

JSON - DATATYPES

Type	Description
Number	double- precision floating -point format in JavaScript
String	double-quoted Unicode with backslash escaping
Boolean	true or false
Value	it can be a string , a number, true or false, null etc
Array	an ordered sequence of values
Object	an unordered collection of key:value pairs
Whitespace	can be used between any pair of tokens
null	empty

JSON Name value pare

- A double precision floating -point format in JavaScript
  - Octal and hexadecimal formats are not used.
- No NaN or Infinity is used in



Type	Description
Integer	Dig its 1-9, 0 and positive or negative
Fraction	Fractions like .3, .9
Exponent	Exponent like e, e+, e-,E, E+, E-

String

- It is a sequence of zero or more double quoted Unicode characters with backslash escaping.
- Character is a single character string i.e. a string with length 1.
  - var obj = {'name': 'Amit'}
  - var obj = {'name': 'Amit', 'marks': 97, 'distinction': true}

Array

- It is an ordered collection of values.
- These are enclosed square brackets which means that array begins with [and ends with].
- The values are separated by ,(comma).
- Array indexing can be started at 0 or 1.
- Arrays should be used when the key names are sequential integers.

```
{
 "books": [
 { "language":"Java" , "edition":"second" },
 { "language":"C++" , "lastName":"fifth" },
 { "language":"C" , "lastName":"third" }
 ]}
```

Object

- It is an unordered set of name/value pairs.
- Object are enclosed in curly braces that is it starts with '{' and ends with '}'.
- Each name is followed by ':'(colon) and the name/value pairs are separated by , (comma).
- The keys must be strings and should be different from each other.
- Objects should be used when the key names are arbitrary strings

```
{
"id": "011A", "language": "JAVA", "price": 500,
}
```

JSON - OBJECTS

- JSON objects can be created with Javascript.

Creation of an empty Object:

- var JSONObj = {};

Creation of new Object:

- var JSONObj = new Object();

Creation of an object with attribute bookname with value in string , attribute price with numeric value.

Attributes is accessed by using '.' Operator:

- var JSONObj = { "bookname ":"VB BLACK BOOK", "price":500 };

Example

```
<html> <head>
<title>Creating Object JSON with JavaScript</title>
<script language="javascript" >
var JSONObj = { "name" : "tutorialspoint.com", "year" : 2005
    };
document.write("<h1>JSON with JavaScript example</h1>");
document.write("<br>");
document.write("<h3>Website
    Name="+JSONObj.name+"</h3>");
document.write("<h3>Year="+JSONObj.year+"</h3>");
</script></head> </html>
```

JSON - SCHEMA

- JSON Schema is a specification for JSON based format for defining structure of JSON data. It was written under IETF draft which expired in 2011.
- JSON Schema describes your existing data format with:
  - Clear, human- and machine-readable documentation.
  - Complete structural validation, useful for automated testing .
  - Complete structural validation, validating client-submitted data.

JSON Schema Example

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"title": "Product", "description": "A product from Acme's catalog",
    "type": "object",
"properties": { "id": { "description": "The unique identifier for a
    product", "type": "integer"
}, "name": { "description": "Name of the product", "type": "string"
},"price": { "type": "number", "minimum": 0, "exclusiveMinimum":
    true
} },
"required": ["id", "name", "price“]}
```

Keywords	Description
$schema	The $schema keyword states that this schema is written according to the draft v4 specification.
title	You will use this to give a title to your schema
description	A little description of the schema
type	The type keyword defines the first constraint on our JSON data: it has to be a JSON Object.
properties	Defines various keys and their value types, minimum and maximum values to be used in JSON file.
required	This keeps a list of required properties.
minimum	This is the constraint to be put on the value and represents minimum acceptable value.
exclusiveMinimum	If "exclusiveMinimum" is present and has boolean value true, the instance is valid if it is strictly greater than the value of "minimum".
maximum	This is the constraint to be put on the value and represents maximum acceptable value.
exclusiveMaximum	If "exclusiveMaximum" is present and has boolean value true, the instance is valid if it is strictly lower than the value of "maximum".
multipleOf	A numeric instance is valid against "multipleOf" if the result of the division of the instance by this keyword's value is an integer.
maxLength	The length of a string instance is defined as the maximum number of its characters.
minLength	The length of a string instance is defined as the minimum number of its characters.

JSON Schema Example

```
[{
"id": 2,
"name": "An ice sculpture",
"price": 12.50,
},
{
"id": 3,
"name": "A blue mouse",
"price": 25.50,
}]
```

Comparison of JSON and XML

- **Similarities:**
  - Both are human readable
  - Both have very simple syntax
  - Both are hierarchical
  - Both are language independent
  - Both can be used by Ajax
- **Differences:**
  - Syntax is different
  - JSON is less verbose
  - JSON can be parsed by JavaScript's eval method
  - JSON includes arrays
  - Names in JSON must not be JavaScript reserved words
  - XML can be validated
  - JavaScript is not typically used on the server side

Support for JSON in PHP

- JSON functions
  - json_decode — Decodes a JSON string
  - json_encode — Returns the JSON representation ofa value
  - json_last_error — Returns the last error occurred

json_decode()

- json_decode (string $json , bool $assoc)
  - Takes a JSON encoded string and converts it into a PHP value.
- $json
  - The JSON string being decoded
- $assoc
  - false (default) return the value as an object
  - True return the value as an associative array

json_encode()

- string json_encode ($value)
  - Returns a string containing the JSON
  - representation of $value.
- $value
  - The value being encoded. Can be any type except a resource.
  - This function only works with UTF-8 encoded data.

JSON and—methods?

- In addition to instance variables, objects typically have methods
  - There is nothing in the JSON specification about methods
- However, a method can be represented as a string, and (when received by the client) evaluated with eval
  - Obviously, this breaks language-independence
  - Also, JavaScript is rarely used on the server side

JSON eval function

- The JavaScript eval(*string*) method compiles and executes the given string
  - The string can be an expression, a statement, or a sequence of statements
  - Expressions can include variables and object properties
  - eval returns the value of the last expression evaluated
- When applied to JSON, eval returns the described object

More reeding

www.tutorialspoint.com/json/

http://www.w3schools.com/json/

http://www.json.org/

http://developers.squarespace.com/what-is-json/

This is the end for this lecture