WKW SCHOOL

2019

CI6206 Internet Programming

JSON

Wong Twee Wee

Ver1.1

JSON (JAVASCRIPT OBJECT NOTATION)

- A lightweight data-interchange format
- Text-based Open Standards
- A JSON string must be enclosed by double quotes.
- Media type for JSON is **application/json**
- filename extension is **.json**
- See http://json.org/ for the detailed syntax of JSON.
- JSON Quick Reference Guide
- JSON Useful Resources
- JSON

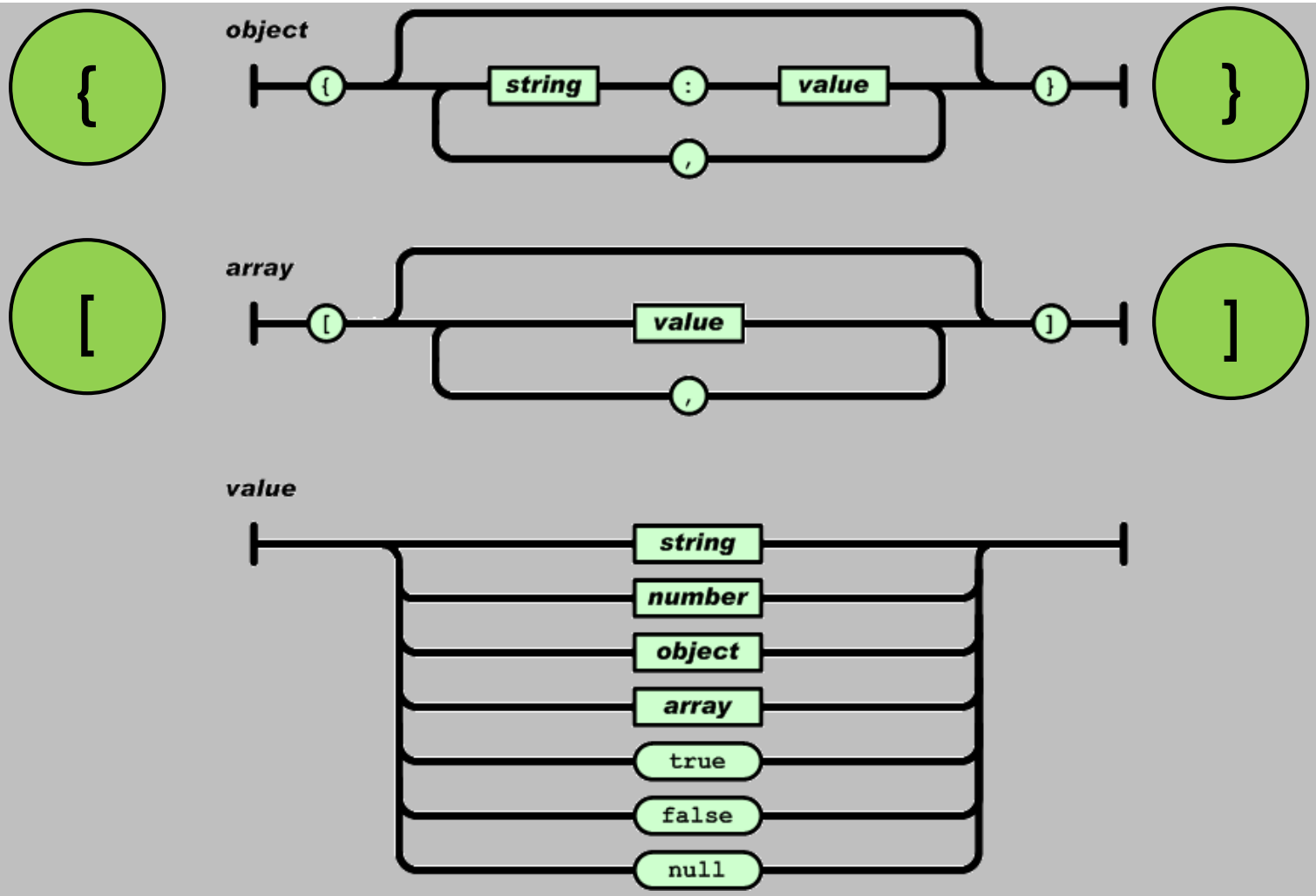
SYNTAX RULES

- JSON syntax is a set of the JavaScript object notation syntax.
- Data is in name/value pairs
- Data is separated by comma
- Curly brackets hold objects
- Square bracket holds arrays

JSON SYNTAX

- Data is represented in name/value pairs
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- *{ Name : Value , Name : Value }*
- Square brackets hold arrays and values are separated by ,(comma).

THE BNF IS SIMPLE



JSON IS BUILT ON TWO STRUCTURES

- A collection of name/value pairs.
  - E.g An object with two properties named “id”, “fName”

Property ‘id’ Property ‘fName’

  - { “id":”333” , “fName":”Jayson” }

- An ordered list of values.
  - E.g An array of three integers and one string value

    [1, 2, 3, "value #4"]

JSON - OBJECT

JSON Objects

Values in JSON can be objects.

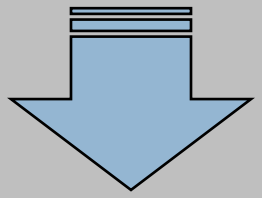
Example

```
{
"employee":{ "name":"John", "age":30, "city":"New York" }
}
```

JSON – ARRAY OF OBJECTS

- { “id":”111” , “fName":”Jayson” }
- { “id":”222” , “fName":”Johnson” }
- { “id":”333” , “fName":”Jackie” }

List of student information ?



- { “Students” : [
  - { “id":”111” , “fName":”Jayson” },
  - { “id":”222” , “fName":”Johnson” },
  - { “id":”333” , “fName":”Jackie” },
-]}

DATATYPE

Valid Data Types

In JSON, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- *null*

JSON values **cannot** be one of the following data types:

- a function
- a date
- *undefined*

DATATYPE

JSON Strings

Strings in JSON must be written in double quotes.

Example

```
{ "name":"John" }
```

JSON Numbers

Numbers in JSON must be an integer or a floating point.

Example

```
{ "age":30 }
```

DATATYPE

JSON Booleans

Values in JSON can be true/false.

Example

```
{ "sale":true }
```

JSON null

Values in JSON can be null.

Example

```
{ "middlename":null }
```

XML EQUIVALENT

```
<Students>
   <student>
      <id>111</id> <fName>Jayson</fName>
   </student >
   <student >
      <id>222</id> <fName>Johnson</fName>
   </student >
   <student >
      <id>333</id> <fName>Jackson</fName>
   </student >
</Students>
```

```
{ "Students" : [
     { "id":"111" , "fName":"Jayson" },
     { "id":"222" , "fName":"Johnson" },
     { "id":"333" , "fName":"Jackie" },
]}
```

XML VS JSON

JSON

- **Pro:**
  - Simple syntax, which results in less "markup" overhead compared to XML.
  - Easy to use with JavaScript as the markup is a subset of JS object literal notation and has the same basic data types as JavaScript.
- **Con:**
  - Simple syntax, only a handful of different data types are supported.

XML VS JSON

XML

- **Pro:**
  - Generalized markup; it is possible to create "dialects" for any kind of purpose
  - XML Schema for datatype, structure validation. Makes it also possible to create new datatypes
  - XSLT for transformation into different output formats
  - XPath/XQuery for extracting information (which makes getting information in deeply nested structures much easier then with JSON)
- **Con:**
  - Relatively wordy compared to JSON (results in more data for the same amount of information).

DATA EXCHANGE

- A common use of JSON is to exchange data to/from a web server.
- When receiving data from a web server, the data is always a string.
- Parse the data with JSON.parse(), and the data becomes a JavaScript object.

PARSING JSON

Example - Parsing JSON

Imagine we received this text from a web server:

```
'{ "name":"John", "age":30, "city":"New York"}'
```

Use the JavaScript function JSON.parse() to convert text into a JavaScript object:

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

Make sure the text is written in JSON format, or else you will get a syntax error.

JSON IN HTML

- HTML Delivery.

- JSON data is built into the page.

```
<html>...
  <script>
      var data = { ... JSONdata ... };
  </script>...
</html>
```

JSON IN HTML

- HTML Delivery.

- JSON data is built into the page.

```
<html>...
  <script>
      var data1 = { ... JSONdata ... };
      var data2 = {
          "ItemName" : "iWatch",
          "price":500 };
  </script>...
</html>
```

EXAMPLE

```
<html>
  <head>
    <title>Creating Object JSON with JavaScript</title>
    <script language = "javascript" >

      var JSONObj = { “Code" : “CI6206", “Year"  : 2015 };

      document.write("<h2>Learn JSON</h2>");
      document.write("<br>");
      document.write("<h3>Module Code = "+JSONObj.Code +"</h3>");
      document.write("<h3>Year = "+JSONObj.Year+"</h3>");

    </script>
  </head>
  <body> … </body>
</html>
```

EXAMPLE - ARRAY

```
<html>
  <head>
    <title>Creating Object JSON with JavaScript</title>
    <script language = "javascript" >
      var Books = {
          "Classic" : [
             { "Name"  : "The Great Gatsby", "price" : 44.40 },
             { "Name"  : "To Kill a Mockingbird", "price" : 55.12 }
          ],
          "War"  : [
            { "Name"  : "American Sniper", "price" : 19 },
            { "Name"  : "Lone Survivor", "price" : 25 }
          ]
      }
    </script>
  </head>
  <body> … </body>
</html>
```

Example

Books.Classic[0].Name

Books.Classic[0].Price

EXAMPLE - ARRAY

```
<html>
  <head>
    <title>Creating Object JSON with JavaScript</title>
    <script language = "javascript" >
      var Books = { … }
      for(i = 0;i<Books.Classic.length;i++){
            document.writeln("<tr>");
        document.writeln("<td>Classic</td>");
        document.writeln("<td>" + Books.Classic[i].Name  + "</td>");
        document.writeln("<td>" + Books.Classic[i].price + "</td>");
        document.writeln("</tr>");
       }

    </script>
  </head>
  <body> … </body>
</html>
```

WORKING WITH JAVASCRIPT

Two methods are primarily used with JavaScript and JSON

- JSON.parse()
  - converts a JSON string into an Javascript object.
- JSON.stringify().
  - When sending data to a web server, the data has to be a string.
  - converts a Javascript object into a string with JSON.stringify()

JSON.PARSE()

Turning a JSON string into an Javascript object

```
var jsonString =
    '{"name":"ben","age":23,"skills":["php","css","javascript"]}';
var json = JSON.parse(jsonString);
var info = '<ul><li>Name:' + json.name;
console.log(json);
```

```
Object
    age: 23
    name: "ben"
    skills: Array[3]
        0: "php"
        1: "css"
        2: "javascript"
```

JSON.PARSE()

```
<!DOCTYPE html>
<html>
<body>

<h2>Create Object from JSON String</h2>

<p id="demo"></p>

<script>

var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;

</script>

</body>
</html>
```

Create Object from JSON String

John, 30

JSON.STRINGIFY()

Turning a Javascript object into a JSON string.

```
/define a json object
var data = {
    name: 'ben',
    age: 23,
    skills: [
        'php', 'css', 'javascript'
    ]
}
//use JSON.stringify to convert it to json string
var json = JSON.stringify(data);
$("#result").append('<p>json string: ' + json + '</p>')
console.log(json);
```

```
{"name":"ben","age":23,"skills":["php","css","javascript"]}
```

```
<?xml version='1.0' encoding='UTF-8'?>
<card>
   <fullname>Jayson Boswick</fullname>
   <org>WKWSCI</org>
   <emailaddrs>
      <address type='work'>jayson.b@wkwsci.ntu.edu.sg</address>
      <address type='home'>jayson@gmail.com</address>
   </emailaddrs>
   <telephones>
      <tel type='work'>+65 67483680 </tel>
      <tel type='fax'>+65 67483000 </tel>
      <tel type='mobile'>+65 93334488 </tel>
   </telephones>
   <addresses>
     <address type='work' format='sg'>31 Nanyang Link, 637718</address>
     <address type='home' format='sg'> 444 Hougang Ave 8, 530444</address>
   </addresses>
   <urls>
      <address type='work'>http://WKWSCI.ntu.edu.sg/</address>
      <address type='home'>http://JAYSON.bitly/</address>
   </urls>
</card>
```

Example: An address book data encoded in XML

```
{
   "fullname": "Jayson Boswick",
   "org": "WKWSCI",
   "emailaddrs": [
      {"type": "work", "value": " jayson.b@wkwsci.ntu.edu.sg"},
      {"type": "home", "value": " jayson@gmail.com"}
   ],
    "telephones": [
      {"type": "work", "value": "+65 67483680"},
      {"type": "fax", "value" : "+65 67483000"},
      {"type": "mobile", "value": "+65 93334488"}
   ],
   "addresses": [
      {"type": "work", "format": "sg",
       "value": "31 Nanyang Link, 637718"},
      {"type": "home", "format": "sg",
       "value": "444 Hougang Ave 8, 530444"}
   ],
    "urls": [
      {"type": "work", "value": "http:// WKWSCI.ntu.edu.sg/"},
      {"type": "home", "value": "http:// JAYSON.bitly/"}
   ]
}
```

Example: The same address book data encoded in JSON