What is JSON?

* JSON stands for **J**ava**S**cript **O**bject **N**otation
* JSON is a lightweight **data-interchange** format
* JSON is "self-describing" and easy to understand
* JSON is language independent **\***

\*  
JSON uses JavaScript syntax, but the JSON format is text only.  
Text can be read and used as a data format by any programming language.

Why use JSON?

Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.

JavaScript has a **built in function** to convert a string, written in JSON format, into native JavaScript objects:

**JSON.parse()**

So, if you receive data from a server, in JSON format, you can use it like any other JavaScript object.

# **JSON Syntax**

The JSON syntax is a subset of the JavaScript syntax.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

JSON names require double quotes. JavaScript names don't.

## JSON Values

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

* a string { "name":"John" }
* a number { "age":30 }
* an object (JSON object)

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

* an array

{  
"employees":[ "John", "Anna", "Peter" ]  
}

* a Boolean { "sale":true }
* null { "middlename":null }

In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:

* **a function**
* **a date**
* **undefined**

## JSON Files

* The **file type** for JSON files is **".json"**
* The **MIME type** for JSON text is "**application/json**"

# **JSON vs XML**

Both JSON and XML can be used to receive data from a web server.

The following JSON and XML examples both define an employees object, with an array of 3 employees:

### **JSON Example**

{"employees":[  
  { "firstName":"John", "lastName":"Doe" },  
  { "firstName":"Anna", "lastName":"Smith" },  
  { "firstName":"Peter", "lastName":"Jones" }  
]}

### **XML Example**

<employees>  
  <employee>  
    <firstName>John</firstName> <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName> <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName> <lastName>Jones</lastName>  
  </employee>  
</employees>

## 

## JSON is Like XML Because

* Both JSON and XML are "**self describing"** (human readable)
* Both JSON and XML are **hierarchical** (values within values)
* Both JSON and XML **can be parsed** and used by lots of programming languages
* Both JSON and XML can be fetched with an XMLHttpRequest

## JSON is Unlike XML Because

* JSON **doesn't** use **end tag**
* JSON is shorter
* JSON is quicker to read and write
* **JSON can use arrays**

**The biggest difference is:**

 XML has to be parsed with an **XML parser**. JSON can be parsed by a standard JavaScript function.

**Why JSON is Better Than XML**

XML is much more difficult to parse than JSON.  
JSON is parsed into a ready-to-use JavaScript object.

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
* JSON.Parse the JSON string