Quintrix
A MINDLANCE COMPANY

Amazon Connect Integration

MODULE 4 – JSON

1095 Morris Ave, SUITE 101, Union NJ 07083
646-762-0305 | INFO@QUINTRIXSOLUTIONS.COM

Amazon Connect - JSON

When you transfer files out of Amazon Connect and other latforms you will notice that those files are in a JSON format.

JSON: **J**ava**S**cript **O**bject **N**otation.

What are the key features of JSON?

- syntax for storing and exchanging data.
- text, written with JavaScript object notation.
- lightweight data-interchange format
- "self-describing" and easy to understand
- language independent *

You can validate, format and edit JSON code through the following tool:

https://jsononline.net/

Amazon Connect - JSON

Exchanging Data

When exchanging data between a browser and a server, the data can only be text.

JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.

We can also convert any JSON received from the server into JavaScript objects.

This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Amazon Connect - JSON

Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

Example

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

If you receive data in JSON format, you can convert it into a JavaScript object:

Example

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Amazon Connect - JSON

Storing Data

When storing data, the data has to be a certain format, and regardless of where you choose to store it, text is always one of the legal formats.

JSON makes it possible to store JavaScript objects as text.

Example
Storing data in local storage

```
// Storing data:
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

Quintrix
A MINDLANCE COMPANY

Amazon Connect - JSON

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()

Amazon Connect - JSON

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 Data - A Name and a Value

JSON data is written as name/value pairs. JSON names require double quotes. JavaScript names don’t. A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example
"name":"John"

Amazon Connect - JSON

JSON - Evaluates to JavaScript Objects

The JSON format is almost identical to JavaScript objects.

In JSON, keys must be strings, written with double quotes:

JSON
{ "name":"John" }
In JavaScript, keys can be strings, numbers, or identifier names:

JavaScript
{ name:"John" }

Amazon Connect - JSON

JSON Values

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

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

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

- a function
- a date
- undefined

In JSON, string values must be written with double quotes:

JSON	JavaScript
{ "name":"John" }	{name: ‘John’}

In JavaScript, you can write string values with double or single quotes:

Amazon Connect - JSON

JSON Uses JavaScript Syntax

Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.

With JavaScript you can create an object and assign data to it, like this:

Example
```
var person = { name: "John", age: 31, city: "New York" };
```
You can access a JavaScript object like this:

Example
```
// returns John
person.name;
```

It can also be accessed like this:

Example
```
// returns John
person["name"];
```

Amazon Connect - JSON

JSON Uses JavaScript Syntax

Data can be modified like this:

Example

```
person.name = "Gilbert";
```

It can also be modified like this:

Example

```
person["name"] = "Gilbert";
```

Amazon Connect - 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>
```

Amazon Connect - JSON

JSON vs XML

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.

Amazon Connect - JSON

JSON vs XML

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

Amazon Connect - JSON

JSON Data Types

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

Amazon Connect - JSON

JSON Data Types

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 }

JSON Objects
Values in JSON can be objects.
Example

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

Objects as values in JSON must follow the same rules as JSON objects.

Amazon Connect - JSON

JSON Data Types

JSON Arrays
Values in JSON can be arrays.
Example

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

JSON Booleans
Values in JSON can be true/false.
Example - { "sale":true }

JSON null
Values in JSON can be null.
Example - { "middlename":null }

Amazon Connect - JSON

JSON.parse()

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.

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.

Quintrix
A MINDLANCE COMPANY

Amazon Connect - JSON

JSON From the Server

You can request JSON from the server by using an AJAX request

As long as the response from the server is written in JSON format, you can parse the string into a JavaScript object.

Example
Use the XMLHttpRequest to get data from the server:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj.name;
  }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

Amazon Connect - JSON

Array as JSON

When using the JSON.parse() on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

Example
The JSON returned from the server is an array:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myArr = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myArr[0];
  }
};
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

Amazon Connect - JSON

Exceptions

Parsing Dates

Date objects are not allowed in JSON. If you need to include a date, write it as a string. You can convert it back into a date object later:

Example

Convert a string into a date:

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text);
obj.birth = new Date(obj.birth);

document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;
```

Amazon Connect - JSON

Parsing Functions

Functions are not allowed in JSON.

If you need to include a function, write it as a string.

You can convert it back into a function later:

Example
Convert a string into a function:

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

document.getElementById("demo").innerHTML = obj.name + ", " + obj.age();
```

Amazon Connect - JSON

JSON.stringify()

A common use of JSON is to exchange data to/from a web server. When sending data to a web server, the data has to be a string. Convert a JavaScript object into a string with JSON.stringify().

Stringify a JavaScript Object

Imagine we have this object in JavaScript:

```
var obj = { name: "John", age: 30, city: "New York" };
```

Use the JavaScript function JSON.stringify() to convert it into a string.

```
var myJSON = JSON.stringify(obj);
```

Amazon Connect - JSON

JSON Objects

Object Syntax

Example
{ "name":"John", "age":30, "car":null }

- JSON objects are surrounded by curly braces {}.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.

Accessing Object Values
You can access the object values by using dot (.) notation:

Example
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;

Amazon Connect - JSON

JSON Arrays

Arrays as JSON Objects

Example
["Ford", "BMW", "Fiat"]

Arrays in JSON are almost the same as arrays in JavaScript. In JSON, array values must be of type string, number, object, array, boolean or null.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and undefined.

Amazon Connect - JSON

JSON Arrays

Arrays in JSON Objects

Arrays can be values of an object property:

Example
```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accessing Array Values

You access the array values by using the index number:

Example
```
x = myObj.cars[0];
```

Amazon Connect - JSON

JSON Arrays

Looping Through an Array

You can access array values by using a for-in loop:

Example
```
for (i in myObj.cars) {
  x += myObj.cars[i];
}
```

Or you can use a for loop:

Example
```
for (i = 0; i < myObj.cars.length; i++) {
  x += myObj.cars[i];
}
```

Quintrix
A MINDLANCE COMPANY

Amazon Connect - JSON

JSON Arrays

Nested Arrays in JSON Objects

Values in an array can also be another array, or even another JSON object:

```
myObj = {
  "name":"John",
  "age":30,
  "cars": [
    { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
    { "name":"BMW", "models":[ "320", "X3", "X5" ] },
    { "name":"Fiat", "models":[ "500", "Panda" ] }
  ]
 }
```

To access arrays inside arrays, use a for-in loop for each array:

```
for (i in myObj.cars) {
  x += "<h1>" + myObj.cars[i].name + "</h1>";
  for (j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j];
  }
}
```

Amazon Connect - JSON

Exercise:

Review the Okta blog post

https://aws.amazon.com/blogs/contact-center/configure-single-sign-on-for-amazon-connect-using-okta/