Q. What is JSON?

JSON stands for **J**ava**S**cript **O**bject **N**otation

JSON is plain text written in JavaScript object notation

JSON is a lightweight data-interchange format for storing and transporting data

JSON is used to send data between computers

JSON is often used when data is sent from a server to a web page

JSON is "self-describing" and easy to understand

JSON is language independent

Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.

You can **receive** pure text from a server and use it as a JavaScript object.

You can **send** a JavaScript object to a server in pure text format.

You can work with data as JavaScript objects, with no complicated parsing and translations.

JSON Files

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

A JSON string can be stored in its own file, which is basically just a text file with an extension of `.json.`

JSON Example

This example defines an employees object:

an array of 3 employee records (objects): Each object is a record of a person (with a first name and a last name).

```
{
"employees":
[
    {"firstName":"Raj", "lastName":"Patel"},
    {"firstName":"Jay", "lastName":"Desai"},
    {"firstName":"Manoj", "lastName":"Rana"}
]
}
```

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Name/value is separted by :
- Curly braces hold objects
- objects can contain multiple name/value pairs
- Square brackets hold arrays

- JSON is purely a string with a specified data format — it contains only properties, no methods.
- JSON requires double quotes to be used around strings and property names. Single quotes are not valid other than surrounding the entire JSON string.
- Even a single misplaced comma or colon can cause a JSON file to go wrong, and not work.

parse(): Accepts a JSON string as a parameter, and returns the corresponding JavaScript object.

When receiving data from a web server, the data is always a string.

stringify():

Accepts an object as a parameter, and returns the equivalent JSON string.

When sending data to a web server, the data has to be a string.

Converting a JSON Text to a JavaScript Object

A common use of JSON is to read data from a web server, and display the data in a web page.

First, create a JavaScript string containing JSON syntax:

```
var text = '{
"employees":
[
    {"firstName":"Raj", "lastName":"Patel"},
    {"firstName":"Jay", "lastName":"Desai"},
    {"firstName":"Manoj", "lastName":"Rana"}
]
}
';
```

Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

```
var obj = JSON.parse(text);
```

Finally, use the new JavaScript object in your page:

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

<script>
```

```
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " +
obj.employees[1].lastName;
</script>
```

Stringify a JavaScript Object

Imagine we have this object in JavaScript:

```
const obj = {name: "Raj", age: 30, city: "Surat"};
```

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

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

`myJSON` is now a string, and ready to be sent to a server:

JSON

The JSON format is almost identical to JavaScript objects.

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

```
{"name":"Raj"}
```

JSON Values

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

- a string
- a number
- an object
- an array
- a boolean
- null

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

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":"Raj", "lastName":"Desai" },
  { "firstName":"Jay", "lastName":"Patel" },
  { "firstName":"Manoj", "lastName":"Rana" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>Raj</firstName> <lastName>Desai</lastName>
  </employee>
  <employee>
    <firstName>Jay</firstName> <lastName>Patel</lastName>
  </employee>
  <employee>
    <firstName>Manoj</firstName> <lastName>Rana</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 - parse().

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 - parse the JSON string

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*

JSON Strings

Strings in JSON must be written in double quotes.

```
{"name":"Raj"}
```

JSON Numbers

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

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

JSON Objects

Values in JSON can be objects.

```
{
"employee":{"name":"Raj", "age":30, "city":"Surat"}
}
```

JSON Arrays

Values in JSON can be arrays.

```
{
"employees":["Raj", "Jay", "Manoj"]
}
```

JSON Booleans

Values in JSON can be true/false.

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

JSON null

Values in JSON can be null.

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

Array as JSON

Stringify a JavaScript Array

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