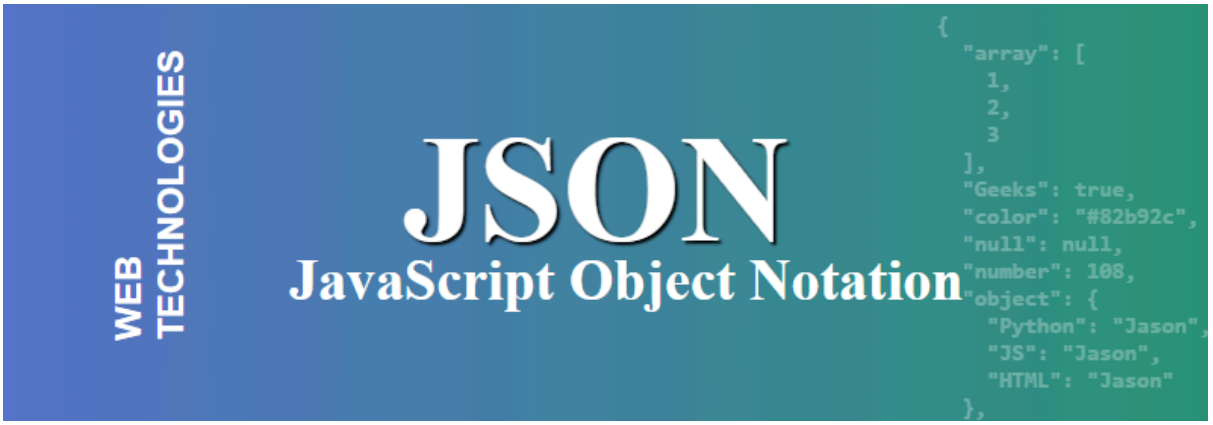
JSON Introduction

JSON stands for **J**ava**S**cript **O**bject **N**otation. It is a format for structuring data. This format is used by different web applications to communicate with each other. JSON is the replacement of the XML data exchange format in JSON. It is easy to struct the data compare to XML. It supports data structures like arrays and objects and the JSON documents that are rapidly executed on the server. It is also a Language-Independent format that is derived from JavaScript. The official media type for the JSON is application/json and to save those file **.json** extension.



Features of JSON:

- **Easy to understand:** JSON is easy to read and write.
- **Format:** It is a text-based interchange format. It can store any kind of data in an array of video, audio, and image anything that you required.
- **Support:** It is light-weighted and supported by almost every language and OS. It has a wide range of support for the browsers approx each browser supported by JSON.
- **Dependency:** It is an Independent language that is text-based. It is much faster compared to other text-based structured data.

JSON Syntax Rules: Data is in name/value pairs and they are separated by commas. It uses curly brackets to hold the objects and square brackets to hold the arrays.

Example:

- Javascript

```
{
    "Courses": [
        {
```

```
            "Name" : "Java Foundation",
            "Created by" : "Hexa",
            "Content" : [ "Java Core", "JSP",
                          "Servlets", "Collections" ]
        },

        {
            "Name" : "Data Structures",
            "also known as" : "Interview Preparation Course",
            "Topics" : [ "Trees", "Graphs", "Maps" ]
        }
    ]
}
```

Advantages of JSON:

- JSON stores all the data in an array so data transfer makes easier. That's why JSON is the best for sharing data of any size even audio, video, etc.
- Its syntax is very easy to use. Its syntax is very small and light-weighted that's the reason that it executes and response in a faster way.
- JSON has a wide range for the browser support compatibility with the operating systems, it doesn't require much effort to make it all browser compatible.
- On the server-side parsing the most important part that developers want, if the parsing will be fast on the server side then the user can get the fast response, so in this case JSON server-side parsing is the strong point compare tot others.

Disadvantages of JSON:

- The main disadvantage for JSON is that there is no error handling in JSON, if there was a slight mistake in the JSON script then you will not get the structured data.
- JSON becomes quite dangerous when you used it with some unauthorized browsers. Like JSON service return a JSON file wrapped in a function call that has to be executed by the browsers if the browsers are unauthorized then your data can be hacked.
- JSON has limited supported tools that we can use during JSON development.

JSON stands for JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package

called JSON. To use this feature, we import the Python JSON package into Python script. The text in JSON is done through quoted-string which contains a value in key-value mapping within { }. It is similar to the dictionary in Python.

Function Used

json.load(): json.load() function is present in Python built-in 'JSON' module. This function is used to parse the JSON string.

json.loads(): json.loads() function is present in Python built-in 'json' module. This function is used to parse the JSON string.

Convert JSON String to Dictionary Python

In this example, we are going to convert a JSON string to Python Dictionary using json.loads() method of JSON module in Python. Firstly, we import json module and then define JSON string after that converting JSON string to Python dictionary by passing it to json.loads() in parameter. We have print the dictionary and their values using the keys as seen in the output.

- Python3

```
# Import JSON module

import json


# Define JSON string

jsonString = '{ "id": 121, "name": "Naveen", "course": "MERN Stack"}'


# Convert JSON String to Python

student_details = json.loads(jsonString)


# Print Dictionary

print(student_details)


# Print values using keys

print(student_details['name'])

print(student_details['course'])
```

Output

```
{'id': 121, 'name': 'Naveen', 'course': 'MERN Stack'}

Naveen

MERN Stack
```

Convert JSON File to Python Object

Below is the JSON file that we will convert to Python dictionary using **json.load()** mehtod.

In the below code, firstly we open the “data.json” file using file handling in Python and then convert the file to Python object using the json.load() method we have also print the type of data after conversion and print the dictionary.

- Python3

```
# Python program to demonstrate

# Conversion of JSON data to

# dictionary


# importing the module

import json


# Opening JSON file

with open('data.json') as json_file:

    data = json.load(json_file)


    # Print the type of data variable

    print("Type:", type(data))


    # Print the data of dictionary

    print("\nPeople1:", data['people1'])

    print("\nPeople2:", data['people2'])
```

Output :

Convert Nested JSON Object to Dictionary

In this example, we will convert the nested JSON into a Python dictionary. For JSON data we will use the same JSON file used in the above example.

- Python3

```
# importing the module
import json


# Opening JSON file
with open('data.json') as json_file:
    data = json.load(json_file)


    # for reading nested data [0] represents
    # the index value of the list
    print(data['people1'][0])


    # for printing the key-value pair of
    # nested dictionary for loop can be used
    print("\nPrinting nested dictionary as a key-value pair\n")
    for i in data['people1']:
        print("Name:", i['name'])
        print("Website:", i['website'])
        print("From:", i['from'])
        print()
```

Output :

Convert JSON String to Dictionary in Python

In this example, we will convert the json string into Python dictionary using json.loads() method. Firstly, we will import JSON module. Create a json string and store it in a variable ‘json_string’ after that we will convert the json string into dictionary by passing ‘json_string’ into json.loads() as argument and store the converted dictionary in ‘json_dict’. Finally, print the Python dictionary.

- Python3

```
import json


# JSON string

json_string = '{"Name": "Suezen", "age": 23, "Course": "DSA"}'


# Convert JSON string to dictionary

json_dict = json.loads(json_string)


print(json_dict)
```

Output

```
{'Name': 'Suezen', 'age': 23, 'Course': 'DSA'}
```

JSON is a lightweight data format for data interchange that can be easily read and written by humans, and easily parsed and generated by machines. It is a complete language-independent text format. To work with JSON data, Python has a built-in package called JSON.

Example of JSON String

```
s = '{"id":01, "name": "Emily", "language": ["C++", "Python"]}'
```

The syntax of JSON is considered a subset of the syntax of JavaScript including the following:

- **Name/Value pairs:** Represents Data, the name is followed by a colon(:), and the Name/Value pairs are separated by a comma(,).
- **Curly braces:** Holds objects.
- **Square brackets:** Hold arrays with values separated by a comma (,).

Keys/Name must be strings with double quotes and values must be data types amongst the following:

- String
- Number
- Object (JSON object)
- array
- Boolean
- Null

Example of JSON file:

```
{
   "employee": [
     {
           "id": "01",
           "name": "Amit",
          "department": "Sales"
     },
     {
           "id": "04",
           "name": "sunil",
          "department": "HR"
     }
   ]
}
```

Python Parse JSON String

In the below code, we are going to convert JSON to a Python object. To parse JSON string Python firstly we import the JSON module. We have a JSON string stored in a variable **'employee'** and we convert this JSON string to a Python object using **json.loads()** method of JSON module in Python. After that, we print the name of an employee using the key 'name' .

- Python3

```
# Python program to convert JSON to Python

import json



# JSON string

employee ='{"id":"09", "name": "Nitin", "department":"Finance"}'



# Convert string to Python dict

employee_dict = json.loads(employee)

print(employee_dict)



print(employee_dict['name'])
```

Output

```
{'id': '09', 'name': 'Nitin', 'department': 'Finance'}

Nitin
```

Read, Write and Parse JSON using Python

Python read JSON file

Let's suppose we have a JSON file that looks like this.

Here, we have used the open() function to read the JSON file. Then, the file is parsed using json.load() method which gives us a dictionary named data.

- Python3

```
import json


# Opening JSON file
f = open('data.json',)


# returns JSON object as
# a dictionary
data = json.load(f)


# Iterating through the json
# list
for i in data['emp_details']:
    print(i)


# Closing file
f.close()
```

Output:

Convert Python Dict to JSON

In the below code, we are converting a Python dictionary to a JSON object using json.dumps() method of JSON module in Python. We first import the JSON module and then make a small dictionary with some key-value pairs and then passed it into json.dumps() method with 'indent=4' to convert this Python dictionary into a JSON object. As we have given the value of indent to 4 there are four whitespaces before each data as seen in the output.

- Python3

```
# Python program to convert

# Python to JSON

import json


# Data to be written

dictionary = {

  "id": "04",

  "name": "sunil",

  "department": "HR"

}


# Serializing json

json_object = json.dumps(dictionary, indent = 4)

print(json_object)
```

Output

```
{

      "id": "04",
```

```
    "name": "sunil",
    "department": "HR"
}
```

The following types of Python objects can be converted into JSON strings:

- dict
- list
- tuple
- string
- int
- float
- True
- False
- None

Python objects and their equivalent conversion to JSON:

Python	JSON Equivalent
dict	object
list, tuple	array
str	string
int, float	number
True	true

False	false
None	null

Writing JSON to a file in Python

We can write JSON to file using json.dump() function of JSON module and file handling in Python. In the below program, we have opened a file named sample.json in writing mode using **'w'**. The file will be created if it does not exist. Json.dump() will transform the Python dictionary to a JSON string and it will be saved in the file sample.json.

- Python3

```
# Python program to write JSON

# to a file

import json



# Data to be written

dictionary ={

    "name" : "sathiyajith",

    "rollno" : 56,

    "cgpa" : 8.6,

    "phonenumber" : "9976770500"

}
```

```
with open("sample.json", "w") as outfile:

    json.dump(dictionary, outfile)
```

Output:

Python Pretty Print JSON

When we convert a string to JSON the data is in a less readable format. To make it more readable we can use pretty printing by passing additional arguments in json.dumps() function such as **indent** and **sort_keys** as used in the below code.

- Python3

```
# Python program to convert JSON to Python

import json


# JSON string

employee ='{"id":"09", "name": "Nitin", "department":"Finance"}'


# Convert string to Python dict

employee_dict = json.loads(employee)


# Pretty Printing JSON string back

print(json.dumps(employee_dict, indent = 4, sort_keys= True))
```

Output

```
{
```

```
    "department": "Finance",
    "id": "09",
    "name": "Nitin"
}
```

An example of a simple JSON file:

A simple JSON representation

As you can see in the example, a single key-value pair is separated by a colon (:) whereas each key-value pairs are separated by a comma (,). Here, “name”, “profile”, “age”, and “location” are the key fields while the corresponding values are “*Amit Pathak*“, “*Software Engineer*“, “24”, “London, UK” respectively.

A nested JSON is a structure where the value for one or more fields can be an another JSON format. For example, follow the below example that we are going to use to convert to CSV format.

An example of a nested JSON file:

A nested JSON example

In the above example, the key field “*article*” has a value which is another JSON format. JSON supports multiple nests to create complex JSON files if required.