JSON

JSON stands for JavaScript Object Notation. It is a format for structuring data. This format is used by different web applications to communicate with each other. It is the replacement of the XML data exchange format. 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.

Why JSON?

It is a Language-Independent format that is derived from JavaScript. It is Human-readable and writable. It is a lightweight text-based data interchange format which means, it is simpler to read and write when compared to XML. Though it is derived from a subset of JavaScript, yet it is Language independent. Thus, the code for generating and parsing JSON data can be written in any other programming language.

```
{
    "Data Structures": [
        {
            "Name" : "Trees",
            "Course" : "Intoduction of Trees",
            "Content" : [ "Binary Tree", "BST",
                          "Generic Tree"]
        },
        {
            "Name" : "Graphs",
            "Topics" : [ "BFS", "DFS", "Topological Sort" ]
        }
    ]
}
```

Advantages of JSON:

It stores all the data in an array so that data transfer makes easier. That’s why it is the best for sharing data of any size even audio, video, etc.

Its syntax is very small, easy, and light-weighted that’s the reason it executes and responds in a faster way.

It has a wide range for browser support compatibility with the operating systems. It doesn’t require much effort to make it all browser compatible.

On the server-side parsing is the most important part that developers want. If the parsing will be fast on the server side then the user can get a fast response, so in this case, JSON server-side parsing is the strong point compared to others.

Disadvantages of JSON:

The main disadvantage is that there is no error handling. If there was a slight mistake in the script then you will not get the structured data.

It 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.

It has limited supported tools that we can use during the development.

Syntax

The following syntax shows the fetch api method using JavaScript for the reading response of the url.

```
1. fetch('url')
2. .then(response => response.json())
3. .then(data = > console.log(data));
```

Parameters:

This technique accepts two arguments, but one parameter is essential for the method.

- URL: This is the URL that the request should be sent to.
- Options: The set of properties is an array. It is a choice-based parameter.

Return Value:

- Whether it is resolved or not, it returns a promise.
- The returned information may be in XML or JSON formats.
- It could be a collection of objects or just one object.

`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()`.

```
<h2>Create a JSON string from a JavaScript object.</h2>

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


<script>

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

const myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

</script>
```

`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

```
<h2>Creating an Object from a JSON String</h2>


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


<script>

const txt = '{"name":"John", "age":30, "city":"New York"}'

const obj = JSON.parse(txt);

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

</script>