JSON - Introduction

❮ Previous Next ❯

JSON

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

JSON is a **text format** for storing and transporting data

JSON is "self-describing" and easy to understand

JSON Example

This example is a JSON string:

```
'{"name":"John", "age":30, "car":null}'
```

It defines an object with 3 properties:

- name
- age
- car

Each property has a value.

If you parse the JSON string with a JavaScript program, you can access the data as an object:

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *

*
The JSON syntax is derived from JavaScript object notation, but the JSON format is text only.

Code for reading and generating JSON exists in many programming languages.

The JSON format was originally specified by Douglas Crockford.

Why Use JSON?

The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

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

JavaScript has a built in function for converting JSON strings into JavaScript objects:

`JSON.parse()` JSON -> JavaScript

JavaScript also has a built in function for converting an object into a JSON string:

`JSON.stringify()` JavaScript -> JSON

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.

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.

❮ Previous Log in to track progress Next ❯

JSON Syntax

‹ Previous Next ›

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 (aka key/value pairs).

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example

```
"name":"John"
```

JSON names require double quotes.

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"}
```

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 **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

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

JavaScript

```
{name:'John'}
```

JavaScript Objects

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

```
person = {name:"John", age:31, city:"New York"};
```

You can access a JavaScript object like this:

Example

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

Try it Yourself »

It can also be accessed like this:

Example

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

Data can be modified like this:

Example

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

Try it Yourself »

It can also be modified like this:

Example

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

Try it Yourself »

You will learn how to convert JavaScript objects into JSON later in this tutorial.

JavaScript Arrays as JSON

The same way JavaScript objects can be written as JSON, JavaScript arrays can also be written as JSON.

You will learn more about objects and arrays later in this tutorial.

JSON Files

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

JSON Data Types

❮ Previous Next ❯

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.

Example

```
{"name":"John"}
```

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 JSON syntax.

JSON Arrays

Values in JSON can be arrays.

Example

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

Example

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

JSON null

Values in JSON can be null.

Example

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

Previous Log in to track progress Next

JSON Object Literals

❮ Previous Next ❯

This is a JSON string:

```
'{"name":"John", "age":30, "car":null}'
```

Inside the JSON string there is a JSON object literal:

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

JSON object literals are surrounded by curly braces {}.

JSON object literals contains key/value pairs.

Keys and values are separated by a colon.

Keys must be strings, and values must be a valid JSON data type:

- string
- number
- object
- array
- boolean
- null

Each key/value pair is separated by a comma.

It is a common mistake to call a JSON object literal "a JSON object".

JavaScript Objects

You can create a JavaScript object from a JSON object literal:

Example

```
myObj = {"name":"John", "age":30, "car":null};
```

Try it Yourself »

Normally, you create a JavaScript object by parsing a JSON string:

Example

```
myJSON = '{"name":"John", "age":30, "car":null}';
myObj = JSON.parse(myJSON);
```

Try it Yourself »

Accessing Object Values

You can access object values by using dot (.) notation:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj.name;
```

You can also access object values by using bracket ([]) notation:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj["name"];
```

Try it Yourself »

Looping an Object

You can loop through object properties with a for-in loop:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += x + ", ";
}
```

Try it Yourself »

In a for-in loop, use the bracket notation to access the property *values*:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
```