JSON format supports the following data types −

S. No.	Type & Description
1	**Number** double- precision floating-point format in JavaScript
2	**String** double-quoted Unicode with backslash escaping
3	**Boolean** true or false
4	**Array** an ordered sequence of values
5	**Value** it can be a string, a number, true or false, null etc
6	**Object** an unordered collection of key:value pairs
7	**Whitespace** can be used between any pair of tokens
8	**null** empty

Number

- It is a double precision floating-point format in JavaScript and it depends on implementation.
- Octal and hexadecimal formats are not used.
- No NaN or Infinity is used in Number.

The following table shows the number types −

S. No.	Type & Description

1	**Integer** Digits 1-9, 0 and positive or negative
2	**Fraction** Fractions like .3, .9
3	**Exponent** Exponent like e, e+, e-, E, E+, E-

Syntax

```
var json-object-name = { string : number_value, .......}
```

Example

Example showing Number Datatype, value should not be quoted −

```
var obj = {marks: 97}
```

String

- It is a sequence of zero or more double quoted Unicode characters with backslash escaping.
- Character is a single character string i.e. a string with length 1.

The table shows various special characters that you can use in strings of a JSON document −

S. No.	Type & Description
1	**"** double quotation
2	\ backslash
3	**/** forward slash
4	**b** backspace

5	**f** form feed
6	**n** new line
7	**r** carriage return
8	**t** horizontal tab
9	**u** four hexadecimal digits

Syntax

```
var json-object-name = { string : "string value", .......}
```

Example

Example showing String Datatype –

```
var obj = {name: 'Amit'}
```

Boolean

It includes true or false values.

Syntax

```
var json-object-name = { string : true/false, .......}
```

Example

```
var obj = {name: 'Amit', marks: 97, distinction: true}
```

Array

- It is an ordered collection of values.
- These are enclosed in square brackets which means that array begins with .[. and ends with .]..
- The values are separated by , (comma).
- Array indexing can be started at 0 or 1.
- Arrays should be used when the key names are sequential integers.

Syntax

```
[ value, .......]
```

Example

Example showing array containing multiple objects –

```
{
   "books": [
      { "language":"Java" , "edition":"second" },
      { "language":"C++" , "lastName":"fifth" },
      { "language":"C" , "lastName":"third" }
   ]
}
```

Object

- It is an unordered set of name/value pairs.
- Objects are enclosed in curly braces that is, it starts with '{' and ends with '}'.
- Each name is followed by ':'(colon) and the key/value pairs are separated by , (comma).
- The keys must be strings and should be different from each other.
- Objects should be used when the key names are arbitrary strings.

Syntax

```
{ string : value, .......}
```

Example

Example showing Object –

```
{
   "id": "011A",
   "language": "JAVA",
   "price": 500,
}
```

Whitespace

It can be inserted between any pair of tokens. It can be added to make a code more readable. Example shows declaration with and without whitespace –

Syntax

```
{string:" ",....}
```

Example

```
var obj1 = {"name": "Sachin Tendulkar"}
var obj2 = {"name": "SauravGanguly"}
```

null

It means empty type.

Syntax

```
null
```

Example

```
var i = null;

if(i == 1){
   document.write("<h1>value is 1</h1>");
} else{
   document.write("<h1>value is null</h1>");
}
```

JSON Value

It includes –

- number (integer or floating point)
- string
- boolean
- array
- object
- null

Syntax

```
String | Number | Object | Array | TRUE | FALSE | NULL
```

Example

```
var i = 1;
var j = "sachin";
var k = null;
```