# JSON - DATATYPES

 

There are following datatypes supported by JSON format:

| Type | Description |
|---|---|
| Number | double- precision floating-point format in JavaScript |
| String | double-quoted Unicode with backslash escaping |
| Boolean | true or false |
| Array | an ordered sequence of values |
| Value | it can be a string, a number, true or false, null etc |
| Object | an unordered collection of key:value pairs |
| Whitespace | can be used between any pair of tokens |
| 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 number types:

| Type | Description |
|---|---|
| Integer | Digits 1-9, 0 and positive or negative |
| Fraction | Fractions like .3, .9 |
| 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 string types:

| Type | Description |
|---|---|
| " | double quotation |
| \ | reverse solidus |
| / | solidus |
| b | backspace |
| f | form feed |
| n | new line |
| r | carriage return |
| t | horizontal tab |
| 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 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.
- 
- Object are enclosed in curly braces that is it starts with '{' and ends with '}'.
- 
- Each name is followed by ':'(colon) and the name/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 code more readable. Example shows declaration with and without whitespace:

## Syntax:

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

## Example:

```
var i= "   sachin";
var j = "  saurav"
```

## 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;
```