Practical 7

Handling JSON

Introduction to JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

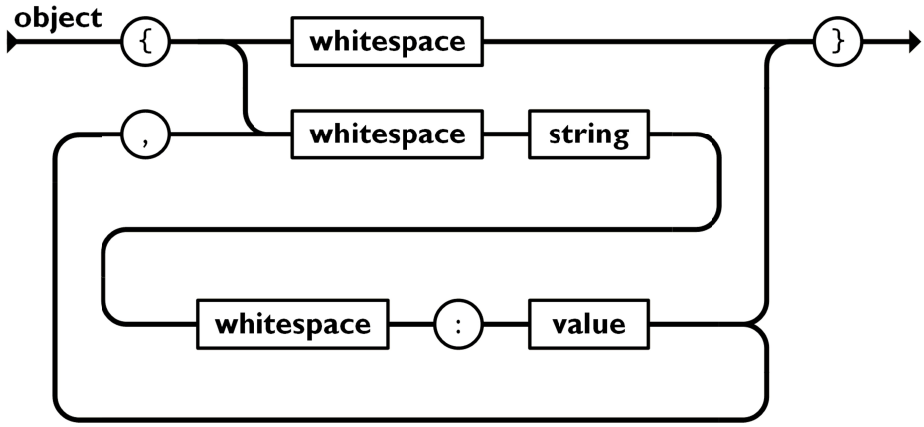
Data types, syntax and example
JSON's basic types are:

- Number (double precision floating-point format in JavaScript, generally depends on implementation)
- String (double-quoted Unicode, with backslash escaping)
- Boolean (true or false)
- Array (an ordered sequence of values, comma-separated and enclosed in square brackets; the values do not need to be of the same type)
- Object (an unordered collection of key:value pairs with the ':' character separating the key and the value, comma-separated and enclosed in curly braces; the keys must be strings and should be distinct from each other)
- null (empty)

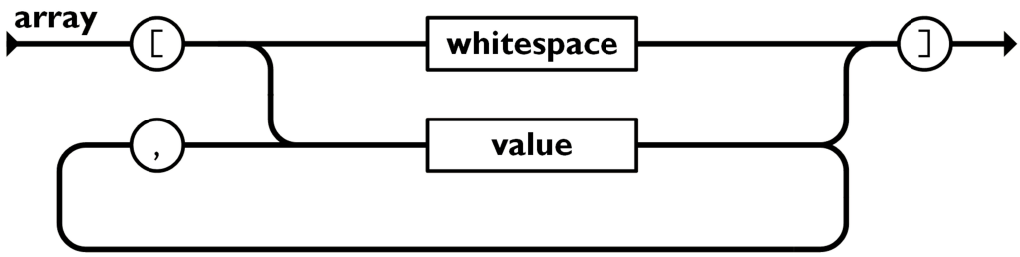
Non-significant white space may be added freely around the "structural characters" (i.e. the brackets "[{]}", colon ":" and comma ",").

In JSON, they take on these forms:

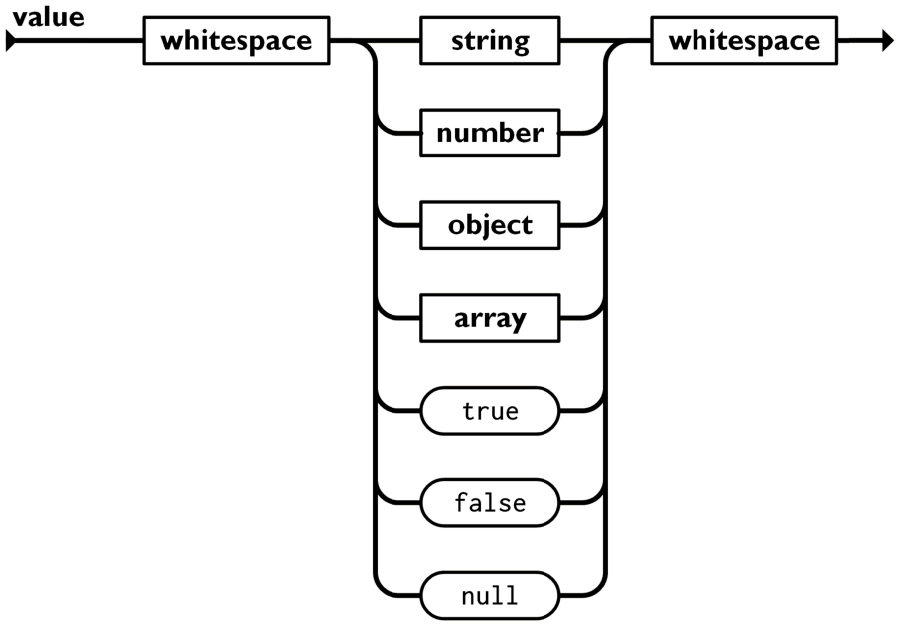
An *object* is an unordered set of name/value pairs. An object begins with `{` left brace and ends with `}` right brace. Each name is followed by `:` colon and the name/value pairs are separated by `,` comma.



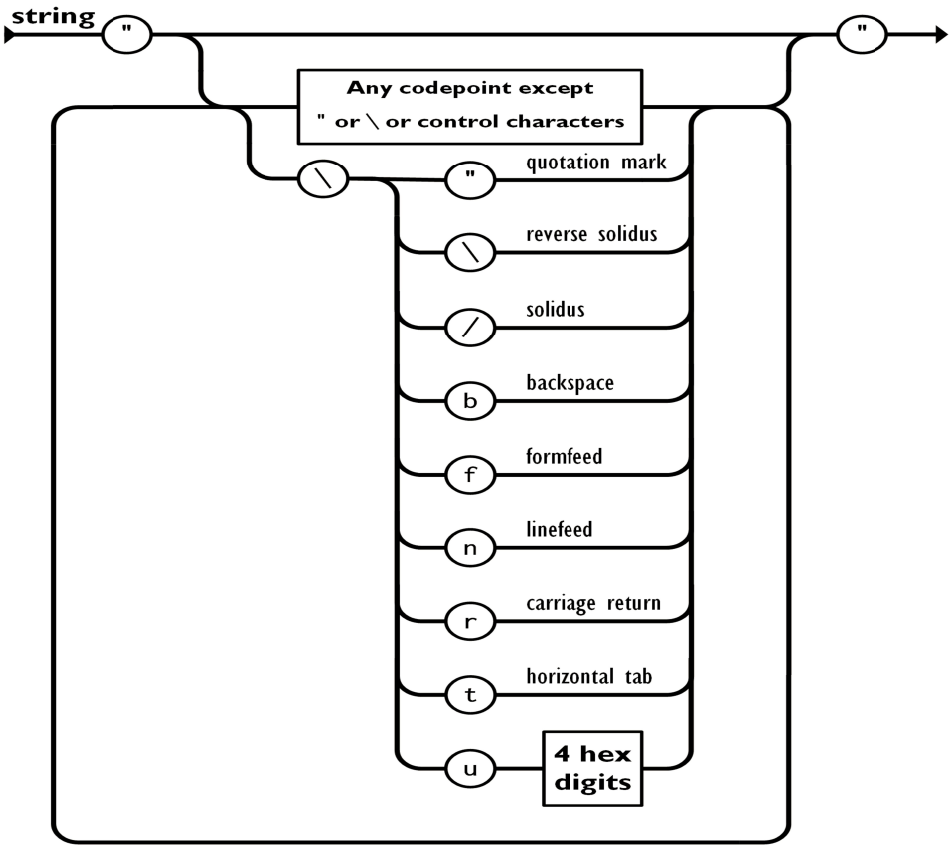
An *array* is an ordered collection of values. An array begins with `[` left bracket and ends with `]` right bracket. Values are separated by `,` comma.



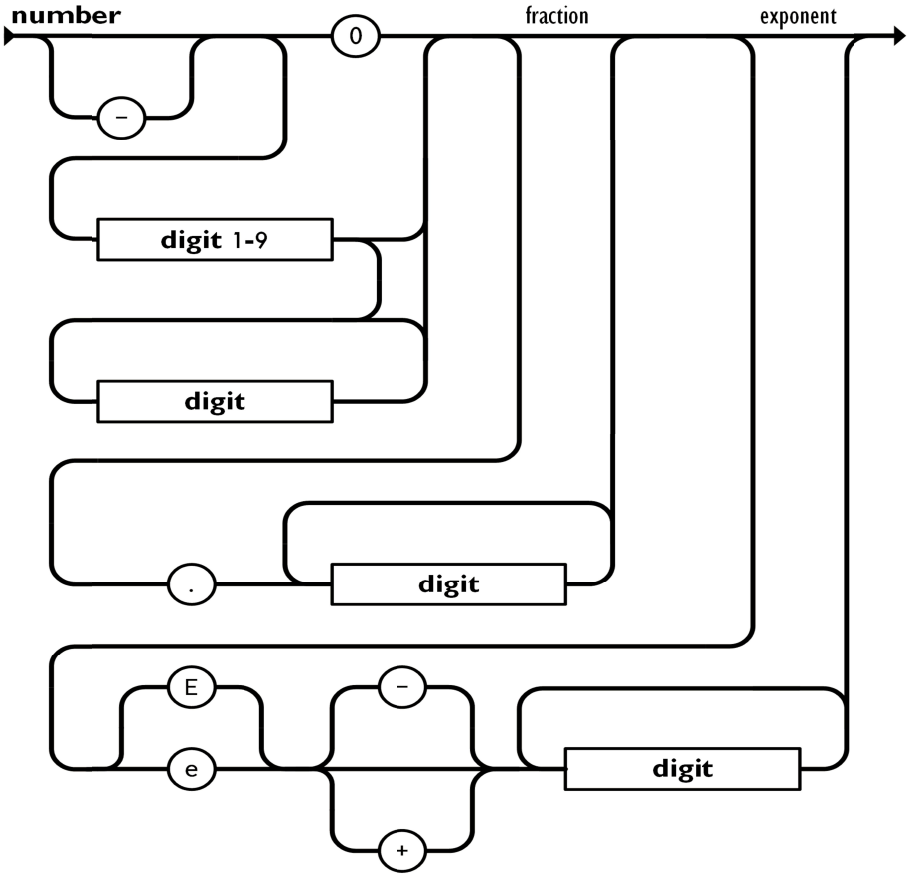
A *value* can be a *string* in double quotes, or a *number*, or `true` or `false` or `null`, or an *object* or an *array*. These structures can be nested.



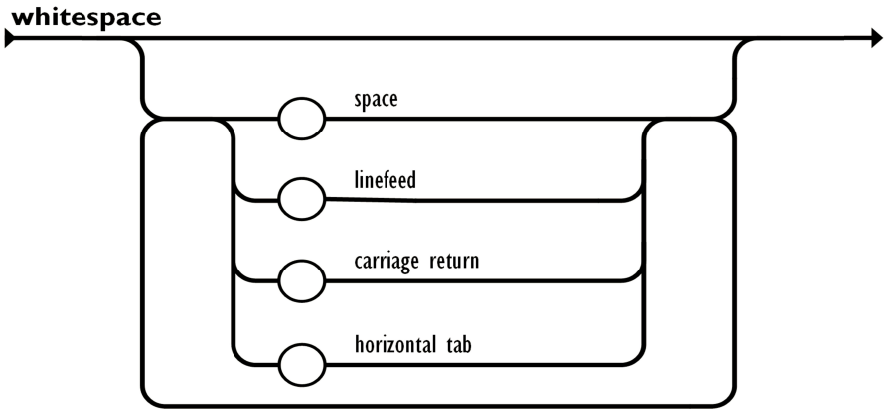
A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.



A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used.



Whitespace can be inserted between any pair of tokens. Excepting a few encoding details, that completely describes the language.



	Exercise
1	Write a script to write and display text from text to JSON object. Demonstrate parse function and array.
2	Accept book data from the user containing AccessNo, Title, Authors (May be one or more), Publisher, Edition details. Convert this data to JSON using PHP script. Write another PHP script to store JSON to PHP variable.
3	Using Using PHP and AJAX, get updated Player score from the server and display on the web page. (Player ID, Name, Runs, AvgRuns, IsHighest, Out) (IsHighest and Out are Booleans) The data should be taken using JSON from the PHP. The score should be updated in the interval of 2 minutes
4	Take students result data in a JSON object. Retrieve the same in Associative array and display all its values as independent strings. (Hint implode function)