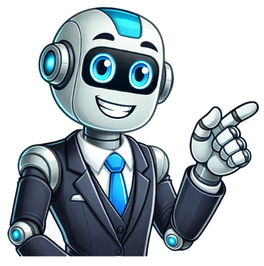
Click to prove you're human



Json array examples

The Internet Engineering Task Force (IETF) has published a new standard for the JavaScript Object Notation (JSON) Data Interchange Format, denoted as RFC 8259. The document aims to address inconsistencies and errors in previous JSON specifications, while providing guidance on interoperability. The revised specification of JSON defines a set of formatting rules for portable representation of structured data. This update is intended to improve the overall usability and reliability of JSON, which is widely used for exchanging data between different systems and applications. The document has received public review and was approved by the Internet Engineering Steering Group (IESG). It represents the consensus of the IETF community and provides information on how to obtain feedback and report any errata. JSON (JavaScript Object Notation) is a format for serializing structured data, derived from JavaScript object literals. It supports four primitive types (strings, numbers, booleans, and null) and two complex types (objects and arrays). A string in JSON is a sequence of Unicode characters, while an object is a collection of name-value pairs, where each name is a string and each value can be a string, number, boolean, or other type. An array is a list of values. The document outlines the structure and syntax of JSON, including its design goals, which were to make it minimal, portable, textual, and compatible with JavaScript. It also defines the key words used in the document and explains the grammatical rules. This document updates RFC 7159, which has been obsoleted. It includes information on character encoding, Unicode characters, string comparison, parsing, generators, IANA considerations, security concerns, examples, and references to other relevant documents. This document was initially designed to describe JSON and formally register the media type "application/json". It also references ECMA-404, which provides additional details on the format. Note that although the two documents use different descriptions, their grammatical rules are intended to be identical. If any discrepancies are found, ECMA and IETF will collaborate to update both specifications accordingly. The goal of this document is to incorporate errata, address inconsistencies with other JSON standards, and highlight potential issues that may cause interoperability problems. A JSON text consists of a series of tokens, including six structural characters and three literal names. It can take the form of an object, array, number, string, or one of the predefined literal values (false, null, true). These literals must be written in lowercase. Additionally, insignificant whitespace is permitted before or after specific structural characters, allowing for flexibility in formatting. Names are allowed with specific values such as %x66.61.6c.73.65 for false, %x6e.75.6c.6c for null, and %x74.72.75.65 for true. Objects are represented by curly brackets containing unique name-value pairs, where a single colon separates the name from its value, and commas separate values from subsequent names. However, when multiple objects have non-unique names, software behavior is unpredictable due to different implementations' handling of such cases. Arrays are enclosed within square brackets and contain elements separated by commas, with no requirement for all elements to be of the same type. Numbers are represented using decimal digits in base 10, including an optional minus sign, fraction part, and exponent part, subject to certain rules regarding representation and precision. JSON Number Precision and String Representation JSON numbers like 1E400 or 3.14159265358979323846264338327 9 might cause interoperability issues because they imply software can handle larger numbers and precision than is common. Integers within the range [-(2**53)+1, (2**53)-1] are considered interoperable as implementations will agree exactly on their numeric values. Strings start and end with quotation marks. All Unicode characters except quotation mark, reverse solidus, and control characters can be placed inside the quotes. Some characters may be escaped or represented in a six-character sequence. Extended characters outside the Basic Multilingual Plane are represented using a 12-character sequence encoding UTF-16 surrogate pairs. Strings containing extended characters like the G clef (U+1D11E) may be represented as "\uD834\uDD1E". Character escape representations include two-character sequences for some popular characters, and 12-character sequences for non-BMP characters. UTF-8 encoding is required for JSON text exchanged between systems not part of a closed ecosystem. Implementations that prioritize interoperability can opt to ignore byte order marks in JSON texts, rather than treating them as errors. This approach allows for greater flexibility and reduces potential issues with software parsing these texts. When a JSON text is composed entirely of Unicode characters (except when escaped), it is considered interoperable, meaning all software implementations will agree on the contents of names and string values. However, some strings may contain bit sequences that cannot encode Unicode characters, leading to unpredictable behavior in receiving software. Software implementations typically compare object member names for equality by transforming textual representations into sequences of Unicode code units and performing numerical comparisons. This approach ensures interoperability, but may incorrectly identify "a\\b" and "a\u005Cb" as unequal strings if comparing unconverted escaped characters. A JSON parser must accept all texts conforming to the JSON grammar, while a JSON generator produces text that strictly adheres to this grammar. Implementations may set limits on text size, maximum depth of nesting, number range and precision, string length, and character contents. The media type for JSON text is application/json, with no required or optional parameters. Security considerations are addressed in Section 12 of RFC 8259, while interoperability considerations are described within the same specification. The JSON Data Interchange Format has a range of extensions, including .json. The Macintosh file type code for this format is TEXT. For further information, contact IESG at . The intended usage is COMMON, with no restrictions on usage. Douglas Crockford is the author, and can be contacted at . The change controller is also IESG at . Security Considerations note that JSON's syntax is similar to JavaScript, but excludes assignment and invocation. This makes it possible for scripting languages to parse most JSON texts using the "eval()" function. However, this poses a security risk if the text contains executable code or data declarations. The format has been used in various examples, including a JSON object with multiple levels of nesting, as well as an array containing two objects and three small JSON texts containing only values. The RFC 8259 JSON December 2017 document references several other documents, including Ecma International's "The JSON Data Interchange Format" standard, IEEE's "IEEE Standard for Floating-Point Arithmetic", Bradner's "Key words for use in RFCs to Indicate Requirement Levels", Yergeau's "UTF-8, a transformation format of ISO 10646", and Crocker's "Augmented BNF for Syntax Specifications: ABNF". RFC 8259 JSON was published in December 2017, referencing Unicode Consortium's "The Unicode Standard". This document is an update to RFC 7159 by Tim Bray, which in turn built upon RFC 4627 written by Douglas Crockford. The changes include updates to sections on informative references, errata filed against RFC 7159, and security risks associated with using the ECMAScript "eval()" function. The document also reflects the removal of a JSON specification from ECMA-262 and requires the use of UTF-8 when transmitted over a network. Discover the editors' top picks from Motorsport Images Collections, featuring over a century's worth of racing moments. From vintage to modern-day events, this curated selection showcases the most compelling and must-see content. Explore now and experience the power of AI-driven creativity in shaping these exceptional images.