JSON Best Practices

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 3rd Edition - December 1999. 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:

1. 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.

2.An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

Characteristics of JSON__

1. JSON is a lightweight data interchange format that is completely language independent.

2. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON format data.

3. The official internet media type for JSON is application/json.

4. It was designed for human readable data interchange.

5. The filename extension is .josn

6. It is easy to read and write.

7. It is text-based interchanging formates.

Uses of JSON__

1. It is used while writing JavaScript based application that includes browser extensions and websites.

2. JSON format is used for serializing and transmitting structured data over network connection.

3. It is primarily used to transmit data between a server and web applications.

4. Web services and APIs use JSON format to provide public data.

Understanding JSON structures__

Two types of json structure

1. Simple
2. Complex

JSON data types__

1. Number
2. String
3. Boolean
4. Array
5. Value
6. Object
7. Whitespace
8. Null

Bad special characters and solution__

Bad characters replace with__

Characters	Solution
1. Backspace	\b
2. Form feed	\f
3. Newline	\n
4. Carriage return	\r
5. Tab	\t
6. Double quote	\”
7. Backslash	\\

Best practices__

JSON is no doubt a flexible way to share data across systems. But that doesn’t mean JSON can be created any way.

Here are some best practices which will help your consumers to use consume your output.

1. **Enclose within DOUBLE Quotes**

Always enclose the **Key: Value** pair within **double quotes.** It may be convenient (not sure how) to generate with Single quotes, but JSON parser doesn't like to parse JSON objects with single quotes.

For numerical Values, quotes are optional but is a good practice to enclose them in double quote.

```
{'id': '1','name': File} is not right ✗
{"id": 1,"name":"File"} is okay ✓
{"id": "1","name":"File"} is the best ✓
```

2. **No Hyphens please**

Never use Hyphens in your Key fields. It breaks python, scala parser and developers have to escape it to use those fields.

Instead of Hyphens use underscores (_). But using alllower case or camel Case is the best. See samples below.

```
{"first-name":"Rachel","last-name":"Green"}  is not right. ✗
{"first_name":"Rachel","last_name":"Green"} is okay ✓
{"firstname":"Rachel","lastname":"Green"} is okay ✓
{"firstName":"Rachel","lastName":"Green"} is the best. ✓
```

3. Always create a Root element.

Creation of Root element is optional, but it helps when you are generating complicated JSON.

JSON with root element

```
{
"menu": [
    {
        "id": "1",
        "name":"File",
        "value": "F",
        "popup": {
            "menuitem": [
                {"name":"New", "value": "1N", "onclick": "newDoc()"},
                {"name":"Open", "value": "1O", "onclick":
"openDoc()"},
                {"name":"Close", "value": "1C", "onclick":
"closeDoc()"}
                ]
            }
    },
    {
        "id": "2",
        "name":"Edit",
        "value": "E",
        "popup": {
            "menuitem": [
                {"name":"Undo", "value": "2U", "onclick": "undo()"},
                {"name":"Copy", "value": "2C", "onclick": "copy()"},
                {"name":"Cut", "value": "2T", "onclick": "cut()"}
                ]
            }
    }
    ]
}
```

JSON without root element

```
[
    {
        "id": "1",
        "name":"File",
        "value": "F",
        "popup": {
            "menuitem": [
                {"name":"New", "value": "1N", "onclick": "newDoc()"},
                {"name":"Open", "value": "1O", "onclick":
"openDoc()"},
                {"name":"Close", "value": "1C", "onclick":
"closeDoc()"}
                ]
            }
    },
    {
        "id": "2",
        "name":"Edit",
        "value": "E",
        "popup": {
            "menuitem": [
                {"name":"Undo", "value": "2U", "onclick": "undo()"},
                {"name":"Copy", "value": "2C", "onclick": "copy()"},
                {"name":"Cut", "value": "2T", "onclick": "cut()"}
                ]
            }


   }
]
```