**JSON 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 our output.

**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 don’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 ✓**

**No Hyphens please**

Never Never 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. ✓**

**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()"}  
 ]  
 }  
   
   
 }  
]

**Provide META sample**

The idea of JSON is flexibility so you don’t have to restrict your data feed within few columns.

But at the same time, if you are providing large data set with nested levels, the consumer will go crazy. Provide them with the meta / sample, so it helps them to understand what data to look for and what to skip.

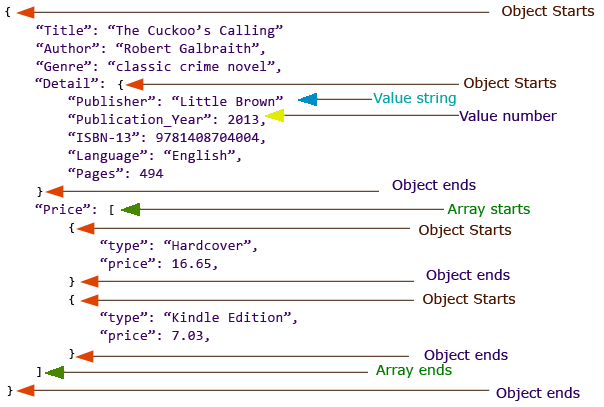
{  
“menu”: [  
 {  
 “id”: “1”,  
 “name”:”File”,  
 “value”: “F”,  
 “popup”: {  
 “menuitem”: [  
 {“name”:”New”, “value”: “1N”, “onclick”: “newDoc()”},  
 {“name”:”Open”, “value”: “1O”, “onclick”: “openDoc()”},  
 {“name”:”Close”, “value”: “1C”, “onclick”: “closeDoc()”}  
 ]  
 }  
 },  
 {  
 “id”: “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()”}  
 ]  
 }  
 }  
 ]  
}

Meta sample can be

menu.id : integer — unique identifier  
menu.name : string — name of the menu  
menu.value : string — internval id of the menu  
menu.popup.menuitem.name : string — name of the submenu  
menu.popup.menuitem.value : string — interlal id of the submenu  
menu.popup.menuitem.onclick : string — client event of the submenu

**Validating JSON output**

Using command line tools like ajv-cli / jsonlint (can be installed via any package manager) will eliminate trouble for consumers.



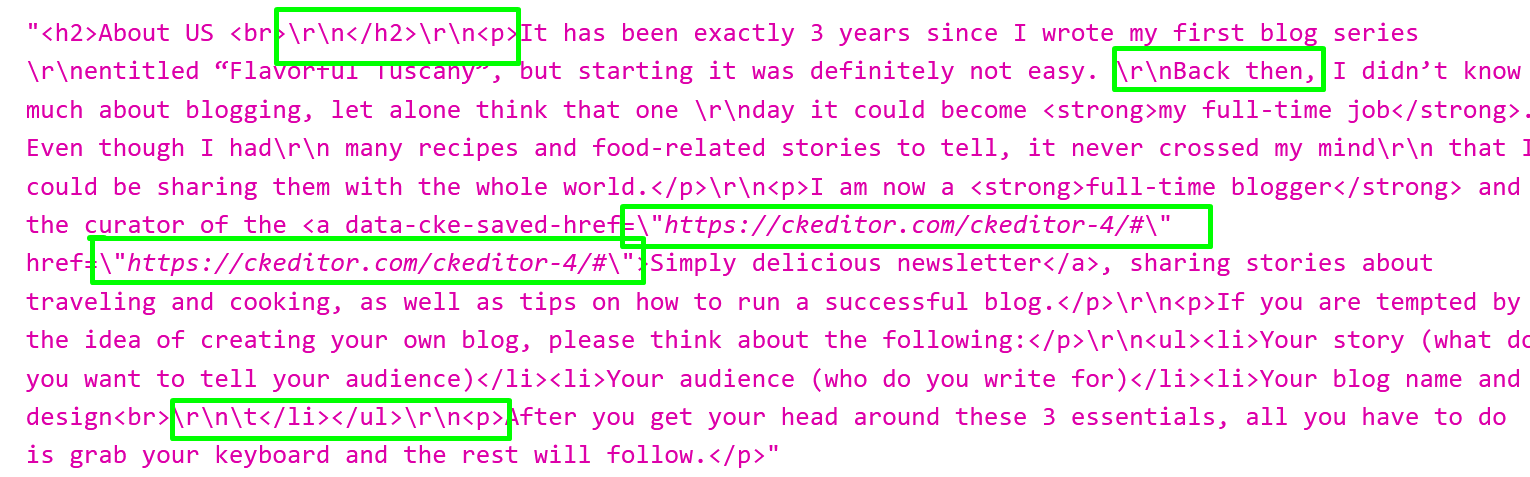
**JSON - Data Types**

|  |  |
| --- | --- |
| 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 |

**Bad Special Characters & Solution**

|  |  |
| --- | --- |
| Characters | Replace with |
| Backspace | \b |
| Form feed | \f |
| Newline | \n |
| Carriage return | \r |
| Tab | \t |
| Double quote | \" |
| Backslash | \\ |

* **Bad Special Characters And Solution**

****