### Status of Project

The project's goal is to support any recommendation to the ECMA or the W3C for the introduction of the inline JSON standard as long as it conforms to the inline JSON standard as it is outlined in this document.  
  
Current Status: In progress  
To check this projects latest status or make a comment, please visit the project forum page [here](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\inline_json.html).

### Preface

This page is dedicated to Kip's Inline JSON project. The Inline JSON project is an effort to document the rules for inline JSON. This project follows guide lines laid out by the [ECMA-404](http://ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf) although the original work done by Mr. Crockford was submitted as [RFC 4627](https://tools.ietf.org/html/rfc7159) in July 2006.  
  
Please keep in mind that inline JSON is not implemented as an allowed format in any popular browser, if you want to use inline JSON you will have to find a library or you can use Kip's [djBase.js](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\djBase.html) library. If you use another Javascript library it’s not guaranteed that they followed all the rules and guide lines that are stated here, please be safe and check their documentation.  
  
Inline JSON's intent is to implement a data format that is compatible across multiple platforms inside of an HTML document. It can be considered a part of a family of syntaxes that are allowed to be used as values inside HTML attributes. Inline JSON unlike the other syntax inside of an html doc is not a technology that is used mainly or primarily for styling or scripting logic. The JSON format can be used for information systems or more specifically in a data exchange process between two servers or programs.  
  
Inline JSON is a working project and is currently looking for reviewers, if you're interested in volunteering please contact kipOmaha@gmail.com.

### Introduction to JSON

JSON is a syntax for sending and receiving information. Its compatibility with multiple programming languages makes it appealing for cross platform development. Languages that are compatible include:

1. Javascript \*
2. PHP
3. Python

This list doesn't mean that the language you use doesn't support JSON. To be sure that your language is compatible you should check the documentation of the language that you are currently programming with.  
  
JSON was developed by [Douglas Crockford](https://en.wikipedia.org/wiki/Douglas_Crockford), you can learn more about Douglas Crockford and his career on Wikipedia. His most current works and information about him can be found at [www.crockford.com](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\www.crockford.com). Mr. Crockford, keeps all JSON schema data at [www.json.org](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\www.json.org), kipOmaha.org does not claim to have any affiliation with the JSON.org name, brand, ect, this is an independent JSON project being conducted for a valid reason within the bounds of the License Agreement set forth at [www.json.org/license.html](http://www.json.org/license.html).   
The license in its current words:  
  
Copyright (c) 2002 JSON.org  
  
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:  
  
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.  
  
The Software shall be used for Good, not Evil.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### Syntax Rules of Inline JSON

The most current rules for the JSON syntax can be downloaded as a pdf file [here](http://ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf).  
  
(This document declares each time it refers to inline JSON by its full name to deter confusion)  
  
This projects goal is to add the current JSON standard to the data-json HTML attribute. The data-json attribute was chosen because it conforms to the HTML data- attribute rules and is an explicit description of what the attribute stores. Although, the intent is not to change the JSON syntax, the following are some recommendations for an inline variety of JSON; I am pushing for the following changes and additions to be made for the implementation of Inline JSON:

1. Single or double quotes allowed.
2. White space or commas between property and value pairs.
3. Add the following global properties:
   1. id: [ string value same as id attr ]
   2. class: [ string value same as id attr ]
   3. name: [ string value same as id attr ]
   4. html: [ string value same as tag name ]
4. Setting the JSON global properties are the same as setting the html element attributes and tag name values.

The point to these changes is to create a JSON syntax that has overlapping rules with the html document making it easier and quicker to code with JSON on an html page.  
  
Below is an example of a valid inline json:  
  
<div id="myElementId" class="myElementClass" name="myElementName" data-json="{  
  'id': 'myElementId'  
  'class': 'myElementClass'  
  'name': 'myElementName'  
  'html': 'div'  
 }"> </div>   
The standard should also include a Javascript method to retrieve the JSON data. For example:  
  
document.getJsonById("myElementId");  
// returns JSON  
  
document.getJsonByClassName("myElementClass");  
// returns JSON node list  
  
document.getJsonByName("myElementName");  
// returns JSON node list  
  
document.getJsonByTagName("myElementTagName");  
// returns JSON node list  
  
  
  
Recommendation 3 from this document outlines the global properties that would enable the Javascript methods to work for JSON.  
  
The Javascript retrieval methods for JSON would mimic the element retrieval methods. Ideally the implementations would be set up so that the JSON and the element attribute values would be the same. With the exception of the html property of the inline JSON, since there is no tag name attribute for the element.  
  
What this means is that setting the JSON id property value would be the same as setting the id attribute value and vice versa. This would eliminate the need to duplicate attribute values. If you look at the example below you can see what the recommendation is explaining:  
  
<div data-json="{  
  'id': 'myElementId'  
  'class': 'myElementClass'  
  'name': 'myElementName'  
  'html': 'div'  
 }"> </div>  
  
// Notice that the global attributes of the div element have been omitted. This was done intentionally,   
// because the property values are set and will automatically set the values of the global attributes.

### Syntax Rules Conclusion

There are 4 recommended changes for the JSON format meant to have a minimal effect on the JSON format itself, but still enable the developer to have a useful set of tools to manipulate and retrieve JSON from an html document. I have personally found that many rich relationships can be formed with this method, for instance the JSON.stringify() method could be manipulated to retrieve the inline string of the json if available. This could be useful if your JSON uses some sort of dynamic stamping process to manipulate JSON strings.  
Below is a review the recommendations from the previous section:

1. Single or double quotes allowed.
2. White space or commas between property and value pairs.
3. Add the following global properties:
   1. id: [ string value same as id attr ]
   2. class: [ string value same as id attr ]
   3. name: [ string value same as id attr ]
   4. html: [ string value same as tag name ]
4. Setting the JSON global properties are the same as setting the html element attributes and tag name values.

Other formatting requirements are based off of the ECMA-404 or the json.org documentation page. If you would like to see the full documentation we have the pdf available as an embed please see the Important Project Notes section.

### Important Project Notes

#### Standard References:

1. [www.json.org](http://www.json.org/) JSON.org outlines all the allowed characters and data types of the JSON format. It also contains a list of resources concerning JSON and the JSON license agreement. In this document we have copy and pasted the license agreement for your convenience.
2. <https://tools.ietf.org/html/rfc7159>The Internet Engineering Task Force puts out requests for comments (rfc) document for web technologies including JSON. If you would like to view the embedded ietf rfc pdf please click the button. Embedded JSON Documentation PDF file
3. [www.ECMA-International.org/publications/standards/Ecma-404.html](http://www.ecma-international.org/publications/standards/Ecma-404.html) ECMA is a private members standardization group that supports a global standard for communications and information systems. Please click the button if you would like to see the embedded pdf of the ecma 404 standard. Embedded JSON Documentation PDF file
4. [www.w3.org](http://www.w3.org/) The World Wide Web Consortium is a standards body that exclusively supports modern and contemporary web standards.

#### Potential Applications:

The application's for an inline format such as JSON are primarily for web application development or widget development. The thought driving this recommendation is that you can use JSON in place of html attributes for web applications; this is useful because JSON is compatible with javascript and many other languages and as a result much of the logic can be used in other scripting languages that may not support html directly. In essence it gives html a more robust and richer way of communicating with other platforms.  
  
Kip Omaha is currently developing a web application library of inline JSON web apps. Many projects are going to be open source and some are closed source until a certain amount of money is raised, please visit our application library by clicking [JSON Application Library](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\json_application_library.html).

#### djBase.js

Earlier we mentioned the djBase.js library. Before you decide to use the library be sure that you are fully aware of the instructions, warnings, notes, and licensing information regarding its use. Please click the link below if you would like to download djBase.js.  
[Javascript Library](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\javascript.html)  
  
The djBase.js library enables you to use inline JSON in an unobtrusive way. The script loads using the async attribute and searches for inline JSON that you request to be looked for VIA the update() function. The update() function offers one required parameter and two other optional parameters. The parameters:

1. \_obj: Element node or element node list
2. \_array: An array object
3. deep: Bolean value

The 1st parameter is the html node or node list you want to be updated, the 2nd parameter gives you control over where your JSON gets stored for each update() used, and the 3rd allows you to easily update() any json nested inside the target element. If you’re interested in learning more about the library you can view the reference through the [djBase.html](file:///C:\Users\KIPPIK\Desktop\Kip-Master\Kip\djBase.html) documentation page.  
  
If you would like to be a reviewer, commenter, sponsor, affiliate, donor, or developer of the djBase.js library please contact, kipOmaha@gmail.com.

#### Inline JSON Project's Next Step

As this is an amateur science club, forum, and effort Kip Omaha feels it’s necessary to submit a formal proposal to a standards body, but first we need to collect feedback and write a more formal documentation concerning inline JSON. If you’re interested in helping with the formal write up or being a commenter please contact kipOmaha@gmail.com and please put, "IN RESPONSE TO CALL FOR COMMENTERS OR DOC WRITERS".