YUI Library: JSON Utility

2011-3-21 **v2.9**

Simple Use Case: Parse a JSON string

One of the core use cases for the JSON Utility is to take string data formatted in JavaScript Object Notation and to validate the string as genuine JSON before evaluating it and processing it in script. `YAHOO.lang.JSON.parse()` provides this functionality:

```
var jsonString = '{"productId":1234,
  "price":24.5, "inStock":true, "bananas":null}';
// Parsing JSON strings can throw a SyntaxError
// exception, so wrap in a try catch block
try {
    var prod=YAHOO.lang.JSON.parse(jsonString);
}
catch (e) {
    alert("Invalid product data");
}
// We can now interact with the data
if (prod.price < 25) {
    prod.price += 10; // Price increase!
}
```

Usage: YAHOO.lang.JSON.parse()

`YAHOO.lang.JSON(str JSON[, fn reviver])`

Arguments:

(1) **JSON:** A string containing JSON-formatted data that you wish to validate and parse.

(2) **reviver**: A function that will be executed on each member of the JSON object; see the Solutions box for more.

Returns:

JavaScript representation: The returned value of the evaluated JSON string (if no exception was thrown in its evaluation).

Usage: YAHOO.lang.JSON.stringify()

`YAHOO.lang.JSON.stringify(obj object[, arr whitelist | fn replacer[, n depth]])`

Arguments:

(1) **object:** The JavaScript object you want to stringify.

(2) **whitelist:** An optional array of acceptable keys to include.

(2*) **replacer:** A function to offer a replacement value for serializing. Executed on each member of the input object.

(3) **depth:** An optional number specifying the depth limit to which stringify should recurse in the object structure (there is a practical minimum of 1).

Returns:

JSON string: A JSON string representing the object.

Using the JSON Format

JSON data is characterized as a collection of objects, arrays, booleans, strings, numbers, and null. The notation follows these guidelines:

1. Objects begin and end with curly braces (`{}`).
2. Object members consist of a string key and an associated value separated by a colon (`"key" : VALUE`).
3. Objects may contain any number of members, separated by commas (`{ "key1" : VALUE1, "key2" : VALUE2 }`).
4. Arrays begin and end with square braces and contain any number of values, separated by commas (`[ VALUE1, VALUE2 ]`).
5. Values can be a string, a number, an object, an array, or the literals true, false, and null.
6. Strings are surrounded by double quotes and can contain Unicode characters and common backslash escapes (`"new\nline"`).

JSON.org has helpful format diagrams and specific information on allowable string characters.

Solutions: Using the *reviver* argument

You can filter out and/or reformat specific pieces of data while applying the `parse` method by passing in a second (optional) argument, `reviver`. Reviver is a function that is passed the key and value of the item it is filtering; based on the key and value, the filter can return a formatted value for the item or return `undefined` to omit the key altogether.

```
var currencySymbol = "$"
function myFilter(key,val) {
  // format price as a string
  if (key == "price") {
    var f_price = currencySymbol + (val % 1 ? val + "0" :
        val + ".00");
    return f_price.substr(0,f_price.indexOf('.') + 3);
  }
  // omit keys by returning undefined
  if (key == "bananas") {
    return undefined;
  }
}
var formattedProd = YAHOO.lang.JSON.parse(jsonString,
   myFilter);
// key "bananas" is not present in the formattedProd object
if (YAHOO.lang.isUndefined(formattedProd.bananas)) {
  alert("We have no bananas today");
}
// and the price is the formatted string "$24.50"
alert("Your price: " + formattedProd.price)
```

YAHOO.lang.JSON Methods

parse(str *JSON*[,fn *reviver*]) see usage at left

stringify(obj *object*[, arr *whitelist* | fn *replacer*[, n *depth*]]) see usage at left

isValid(str *JSON*) see online docs

Dependencies

The JSON Utility requires only the YAHOO Global Object.