


 This repository Search

Pull requests Issues Marketplace Gist

metatribal / xmlToJson


Watch 14 Star 145 Fork 73

<> Code ! Issues 11 Pull requests 0 Projects 0 Wiki Insights


simple javascript utility for converting xml into json

🕒 110 commits 1 branch 9 releases 6 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

 metatribal Added minified version Latest commit 761a30c on 29 Feb 2016

lib	Added minified version	a year ago
test	Fixed version test	2 years ago
LICENSE	License change from Apache to MIT	2 years ago
README.md	Update README.md	2 years ago
bower.json	Ignore test folder in bower installations	a year ago
package.json	version bump	2 years ago

 README.md

xmlToJson

A simple javascript module for converting XML into JSON within the browser.

## Features

- no external dependencies
- small (~3kb minified)
- simple parsing. pass either a string or xml node and get back a javascript object ( use `JSON.stringify(obj)` to get the string representation )
- supports attributes, text, cdata, namespaces, default namespaces, attributes with namespaces... you get the idea
- lots of rendering of options
- consistent, predictable output
- browser support - it works on IE 9+, and nearly every version of Chrome, Safari, and Firefox as well as iOS, Android, and Blackberry. (xmlToJson will work for IE 7/8 as well if you set the `xmlns` option to false)

Parsing XML (esp. with namespaces) with javascript remains one of the great frustrations of writing web applications. Most methods are limited by such things as poor browser support, poor or non-existent namespace support, poor attribute handling, incomplete representation, and bloated dependencies.

xmlToJson may not solve all of your woes, but it solved some of mine :)

## Usage

---

Include the src

```
<script type="text/javascript" src="path/xmlToJson.js"></script>
```

and enjoy! xmlToJson is packaged as a simple module, so use it like this

```
testString = '<xml><a>It Works!</a></xml>';    // get some xml (string or document/node)
result = xmlToJson.parseString(testString);    // parse
```

The (prettified) result of the above code is

```
{
  "xml": {
    "a": [
      {
        "text": "It Works!"
      }
    ]
  }
}
```

## Options

```
// These are the option defaults
var options = {
  mergeCDATA: true,      // extract cdata and merge with text nodes
  grokAttr: true,        // convert truthy attributes to boolean, etc
  grokText: true,        // convert truthy text/attr to boolean, etc
  normalize: true,       // collapse multiple spaces to single space
  xmlns: true,           // include namespaces as attributes in output
  namespaceKey: '_ns',   // tag name for namespace objects
  textKey: '_text',      // tag name for text nodes
  valueKey: '_value',    // tag name for attribute values
  attrKey: '_attr',      // tag for attr groups
  cdataKey: '_cdata'     // tag for cdata nodes (ignored if mergeCDATA is true)
  attrsAsObject: true,   // if false, key is used as prefix to name, set prefix to '' to merge child
  stripAttrPrefix: true, // remove namespace prefixes from attributes
  stripElemPrefix: true, // for elements of same name in diff namespaces, you can enable namespaces
```

```
    childrenAsArray: true // force children into arrays
};

// you can change the defaults by passing the parser an options object of your own
var myOptions = {
    mergeCDATA: false,
    xmlns: false,
    attrsAsObject: false
}

result = xmlToJson.parseString(xmlString, myOptions);
```

A more complicated example (with xmlns: true)

```
<?xml version="1.0" encoding="UTF-8"?>
<xml xmlns="http://default.namespace.uri">
  <a>
    <b id="1">one</b>
    <b id="2"><![CDATA[some <cdata>]]>two</b>
    <ns:c xmlns:ns="http://another.namespace" ns:id="3">three</ns:c>
  </a>
</xml>
```

results in

```
{
  "xml": [{
    "attr": {
      "xmlns": {
        "value": "http://default.namespace.uri"
      }
    },
    "a": [{
```

```
    "b": [{
      "attr": {
        "id": {
          "value": 1
        }
      },
      "text": "one"
    }, {
      "attr": {
        "id": {
          "value": 2
        }
      },
      "text": "some <CDATA>two"
    }],
    "c": [{
      "attr": {
        "xmlns:ns": {
          "value": "http://another.namespace"
        },
        "id": {
          "value": 3
        }
      },
      "text": "three"
    }
  ]
}
```

