Introducing: The X3D JSON Loader (2.0 beta), X3DOM JSON Prototype Expander (2.0 beta), X3DOM JSONScript (0.1 alpha) – Your solution for XMLDOM -> X3D JSON -> XMLDOM JavaScript development for X3DOM and X_ITE.

For a Quick Start, edit flowers2.html and replace ../data/flowers2.json with your X3D JSON URL and put flowers2.html, following JavaScript on your web server. Then open flowers2.html in your web browser.

```
<script type="text/javascript" src="../node/X3DJSONLD.js"></script>
<script type="text/javascript" src="../node/Script.js"></script>
<script type="text/javascript" src="../node/PrototypeExpander.js"></script>
<script type="text/javascript" src="../node/Flattener.js"></script>
<script type="text/javascript" src="../node/loaderJQuery.js"></script>
```

[also you may want to add my versions of x_ite (maybe not—should be the same) and x3dom (definitely) for full functionality]

There are more complex cases of multi JSON files in the same scene in prototypes.html.

Good luck.

I still have extra stuff in the global scope of JavaScript—let people know this. Pull requests and forks are welcome, as long as you agree to the license.

License is here: https://github.com/coderextreme/X3DJSONLD/blob/master/LICENSE

Repository is here: https://github.com/coderextreme/X3DJSONLD/

For a web browser, a live, development version of the X3D JSON loader (I recommend

downloading locally or forking) in your HTML, put:

```
<script type="text/javascript"
src="https://raw.githubusercontent.com/coderextreme/X3DJSONLD/master/src/main/node/X3DJSONLD.js"></script>
```

somewhere in the script (see index.html),

call

```
loadX3DJS(document.implementation, json, url, xml, NS, loadSchema, doValidate, function(element, xmlDoc) {
      Then append the element to your DOM:
      document.querySelector(selector).appendChild(element);
      x3dom.reload();
}
```

`selector` is the CSS selector which you want to append the X3DOM HTML code to.

`json` is the X3D JSON you want to display.

`url` is used for resolving URLs in the X3D JSON. Should be similar or the same as the URL you passed to retrieve the JSON from the server.

`xml` is the array or LOG for inclusion into X_ITE via createX3DFromString, this would normally work something like:

```
      var browser = X3D.getBrowser("X3D");
      browser.replaceWorld(browser.createX3DFromString(xml.join("\n")));
```

"X3D" is the CSS query selector.

`NS` is the namespace to use when creating elements in the DOM for the XML Serializer. `http://www.w3.org/1999/xhtml` normally works for X3DOM and

`http://www.web3d.org/specifications/x3d-namespace` normally works for X_ITE. Leaving NS off is also acceptable, but may lead to results you don't like.

Sample code for X_ITE where #x_ite is the id of your X3DCanvas :

```
var content = xml.join("\n");
X3D(function() {
      var browser = X3D.getBrowser("#x_ite");
      browser.replaceWorld(browser.createX3DFromString(content);
      });
```

For the prototype expander a live, development version (I recommend downloading locally or forking) in your HTML, put:

```
<script type="text/javascript"
src="https://raw.githubusercontent.com/coderextreme/X3DJSONLD/master/src/main/node/PrototypeExpander.js"></script>
```

then call (does not modify extern protos yet, use the included server as `node app.js`--works in some cases—does this on the server):

```
json = protoExpander.prototypeExpander(url, json);
json = flattener(json);
```

`json` is the X3DJSON you want to expand protos for (also modifies the parameter as output)

The server-side code for running the prototype expander is (you may have to add the flattener as well):

```
var PROTOS = require('./src/main/node/PrototypeExpander')

PROTOS.setLoadURLs(loadURLs);

json = PROTOS.externalPrototypeExpander(url, json);
```

You may wish to try this on the client to see if it works. I haven't gotten it to work yet in all cases.

There is a lot of useful code in loaderJQuery.js. index.html (for protos), flipper.html (for the base loader, subscenes), prototypes.html (scripts, prototypes, subscenes), prototypes2.html (JSON loading into X_ITE) are good examples.

To enable JSONScript scripting, put this on your web page:

```
<span style="display:none"> Use JSONScript in X3DOM? <input id="scripting" type="checkbox" checked></input></span>
```

Add before calling any loading any scripts with loadScripts():

```
if ($('#scripting').is(':checked')) {

    initializeScripts();

}
```

And put this in your JavaScript where the selector is an X3DOM scene:

```
if ($('#scripting').is(':checked')) {

    loadScripts(json, selector, url);

}
```

To run XML -> JSON conversions, put your .x3d file in src/main/data, and cd to src/main/shell and run `sh several.sh ../data/file.x3d` You will find output in data, nashorn, java, and python folders (way down low for the latter).

To run the proto expander on the server, put your .json and .x3d files in src/main/data and cd to src/main/shell and run `sh runppp.sh` The XML will be in src/main/shell/data and the JSON will be in src/main/ppp. Good luck!

You may start a local file browser by cd'ing to X3DJSONLD and running `node app.js` This will allow you to use ExternProtoDeclare in JSON at least, and search for JSON, WRL, X3D, STL, and PLY files from the web browser.

TODO: Bring X_ITE and XML selectors to API.

John Carlson