**Introduction**

nexl-js is :

1. REST server
2. Scripting language

nexl REST server hosts JavaScripts files and exposes their primitives, arrays and objects via REST API. You might need it to store a distributed configuration/data of different servers, clients etc. in a centralized place.

nexl scripting language is intended to perform a wide variety of data manipulations with JavaScript primitives, arrays and objects hosted by nexl REST server. For example you can merge two or more JavaScript objects and produce an XML of them; or to join few arrays, eliminate duplicate elements, sort the rest elements and join them with comma. All those actions can be performed in a single nexl expression while accessing data on a nexl REST server.

In the other words you store your distributed configuration or data in a native JavaScript file(s) and access it by using scripting language via REST API.

What can nexl scripting language do :

* to resolve object’s key set and values
* to resolve object key by value ( object key reverse resolution )
* to produce an XML, YAML and key-value pairs ( property file ) from JavaScript objects
* to sort arrays, to remove/leave duplicate array elements
* to get array elements by indexes ( including negative indexes and special characters )
* to eliminate items from arrays and objects
* to merge objects, to join arrays
* to cast data types
* to perform string operations
* to use nested nexl expressions ( i.e. expression in expression, … )
* to use \_this\_ and \_parent\_ special keywords for JavaScript objects
* to use JavaScript functions to perform additional data manipulations including user and system functions
* and more…

**Products and installation**

nexl-js has two following products :

1. nexl-server ( is a nexl REST server )
2. nexl-client ( GUI application to interact with nexl-server and to develop nexl expressions locally )

nexl-js is written in JavaScript and requires nodejs to run. On the one hand nexl-js depends on nodejs, on the other hand you get a cross platform application.

First download and install a latest version of nodejs. After that use a command line to install, run or upgrade nexl-js products ( npm program is a part of nodejs installation kit )

|  |  |  |
| --- | --- | --- |
| **Product** | **Install latest** | **Run** |
| **nexl-server** | npm i nexl-server | nexl-server |
| **nexl-client** | npm i nexl-client | nexl |

Additionally you can

|  |  |  |  |
| --- | --- | --- | --- |
| **Product** | **Install specific version** | **Upgrade to latest** | **Uninstall** |
| **nexl-server** | npm i nexl-server@2.0.0 | npm update nexl-server -g | npm uninstall nexl-server -g |
| **nexl-client** | npm i nexl-client@2.0.0 | npm update nexl-client -g | npm uninstall nexl-client -g |

**nexl-server**

nexl-server hosts JavaScript files and exposes their primitives, arrays and objects via REST API. Those files are called nexl sources. By default nexl server is looking up for nexl sources in a ${HOME}/nexl-sources directory ( %userprofile%\nexl-sources in Windows ).

The following command explains how to change nexl server defaults :

nexl-server --help

General information about nexl-server

* nexl server accepts GET and POSTS requests ( the result is same )
* all requests are stateless
* nexl-server supports JSONP requests

Let's consider the following example. Run nexl-server, create nexl-sources directory ( ${HOME}/nexl-sources ) and put there the following JavaScript file named example.js with the following content

**distanceToMoon** = 384400;

**fruits** = [**'Mango'**, **'Banana'**, **'Orange'**, **'Annona'**, **'Grape'**];

**person** = {

**name**: **'Alex'**,

**age**: 25,

**country**: **'Canada'**

};

Here we have 3 items : distanceToMoon primitive, fruits array and person object. Those items are exposed now via REST API. They are accessible by the following URLs :

[http://localhost:8080/example.js?expression=${distanceToMoon}](http://localhost:8080/example.js?expression=$%7BdistanceToMoon%7D)

[http://localhost:8080/example.js?expression=${fruits}](http://localhost:8080/example.js?expression=$%7Bfruits%7D)

[http://localhost:8080/example.js?expression=${person}](http://localhost:8080/example.js?expression=$%7Bperson%7D)

The URL consists of the following

|  |  |
| --- | --- |
| localhost:8080 | Hostname and port of your nexl server |
| /example.js | Relative path to a nexl-source file.  This path is relative to a ${HOME}/nexl-sources directory ( %userprofile%\nexl-sources in Windows ).  You can put as much files as you need in a nexl-sources directory. Also you can use subdirectories ( path is changed according to a directories structure )  You can’t access files in upper level of nexl-source directory |
| expression=${distanceToMoon}  expression=${fruits}  expression=${person} | expression tells us what exactly we need to get from nexl-source file : distanceToMoon, fruits, person variables.  ${distanceToMoon}, ${fruits}, ${person} are nexl expressions  See more about nexl expressions below |

**nexl expressions**

nexl expressions describe what exactly is to get from the nexl source files and what kind of data manipulations is to perform.

Let’s consider examples of nexl expressions ( below will be explained how to test them )

|  |  |
| --- | --- |
| nexl expression | Explanation |
| ${person.name} | First resolves person object and then resolves a name property of that object |
| ${fruits#S} | Resolves a fruits array and sorts it |
| ${fruits#S&,} | First sorts fruits array and then joins all elements with comma |
| ${fruits[-1]} | Resolves a second element from the end of fruits array |
| ${person~K} | Resolves a key set of person object as array |
| ${person~V&,} | Resolves all values of person object as array and then joins all array elements with comma |
| ${person~K#s[$]} | Resolves a key set of person object as array, sorts them in descending order, resolves last array element |
| ${person.country[3..$]^U1} | Resolves a country property of person object, substrings it from fourth element to the end and then capitalizes a first letter |
| ${person~X}  ${person~Y}  ${person~P} | Produces an XML, YAML and key-value pairs ( property file ) from person object |
| ${person<Alex} | Resolves a key of person object by ‘Alex’ string value ( i.e. makes object property reverse resolution ).  The result is array |
| ${person<Alex[0]} | Resolves a key of person object by ‘Alex’ value as array and then resolves a first array element |
| ${person~K+${person~V}&\t} | Joins two arrays. The first array is a key set of a person object, the second array are values of a person object. Finally joins all array elements with tab character |
| ${distanceToMoon~O} | Converts a distanceToMoon primitive number to JavaScript object |
| ${distanceToMoon~O~P} | Converts a distanceToMoon primitive number to JavaScript object and then produces a key-value pair of it  distanceToMoon=384400 |
| ${distanceToMoon~O+${person}} | Converts a distanceToMoon primitive number to JavaScript object and then merges to him person object |
| ${Math.PI} | Resolves a PI property from Math object |
| ${Math.PI|Math.round()} | Resolves a PI property from Math object and pushes it to stack. Calles a Math.round() function which automatically gets a Math.PI argument from the stack |
| ${Math.PI|distanceToMoon|Math.max()} | Pushes a Math.PI to the stack, then pushes a distanceToMoon to the stack. Finally calls a Math.max() function which gets arguments from the stack |

Each nexl expression from examples above you can test by substituting them to the following URL

[http://localhost:8080/example.js?expression=${person.name}](http://localhost:8080/example.js?expression=$%7Bperson.name%7D)

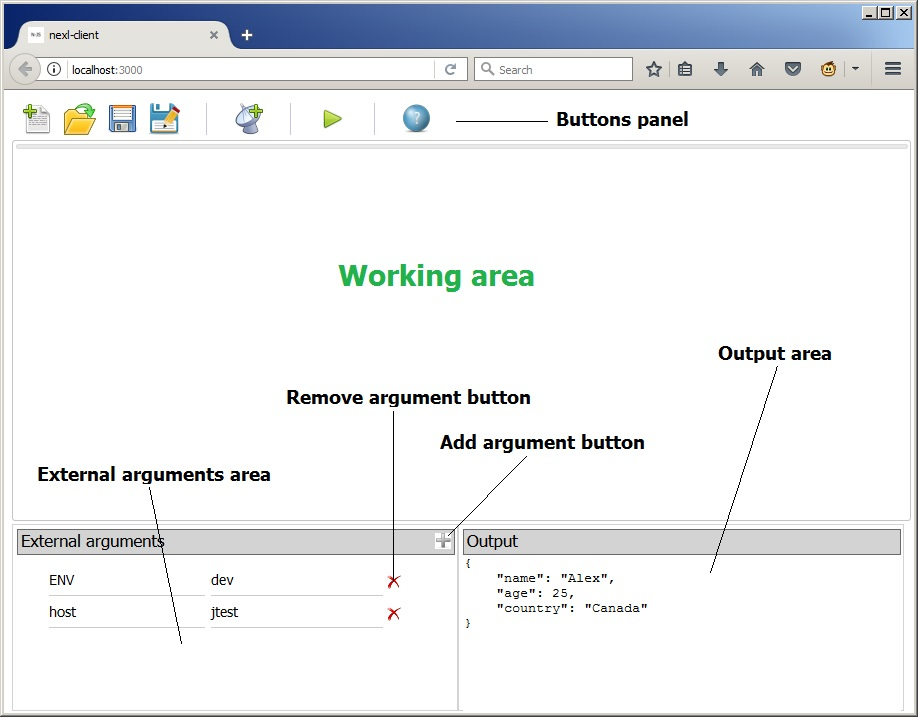
Please pay attention for HTTP GET requests you have to escape the following characters in URL & ( %26 ), # ( %23 ), + ( %2B ).

The better way to test/develop nexl expressions is to use a nexl GUI client ( it escapes everything you need automatically )

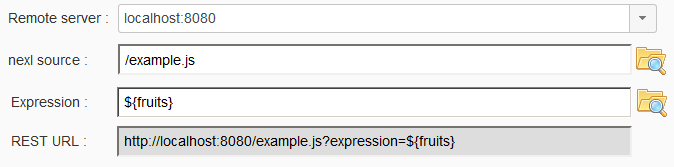
More about nexl scripting language [here](http://www.nexl-js/doc/nexl-expressions)

**nexl-client GUI application**

nexl-client GUI application is intended to interact with nexl-server and to simulate his work locally. nexl-client is a Web application and working in a browser. Type nexl in command line to run nexl-client. It will open your default browser with nexl-client application in it ( it’s recommended to use Chrome browser )



To interact with remote nexl-server click on “New remote nexl source” button 



Remote server is an IP address and port of remote nexl-server.

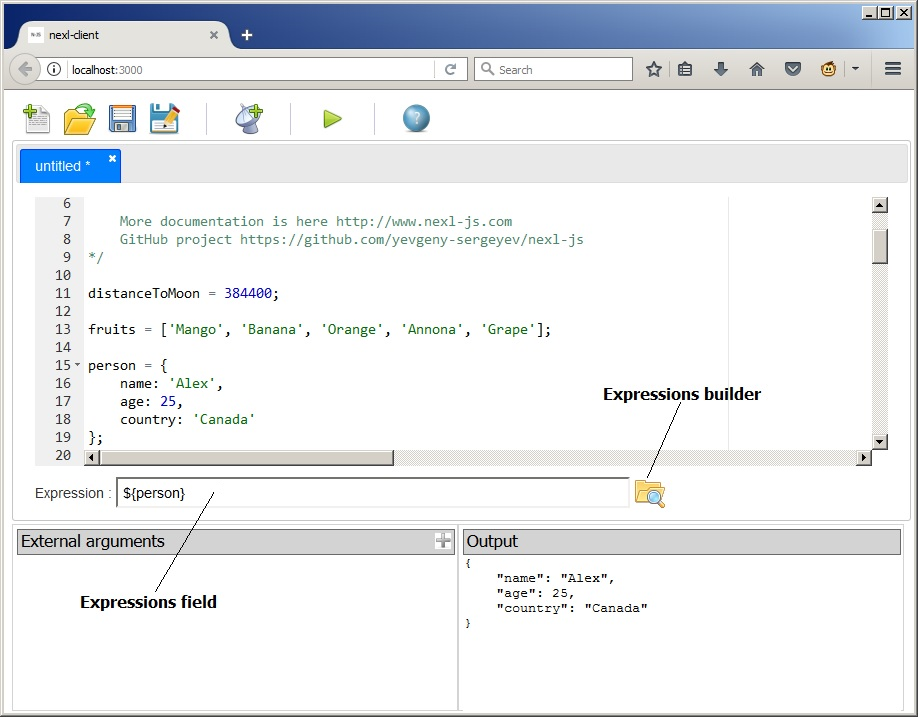
nexl source is a name of JavaScript file hosted and exposed by nexl-server. Specify relative path and file name manually or use “Explore nexl sources” button.

Expression tells us what exactly we need to resolve from the nexl source file. You can specify the expression manually or use “Expression builder” button

REST URL is automatically assembled according to remote server, nexl source, expression and external arguments ( see below ).

If you need to develop you nexl source file without running nexl server, click on “New nexl source file” button 

It will open a JavaScript editor.



Here you can edit your nexl source file and test nexl expressions.

You can save that file in your local file system. Also you can open existing JavaScript file.

To evaluate nexl expression click on “Evaluate nexl expression” button  ( or press F9 )

Evaluation result is shown in output area.

nexl-client will be improved in future versions.

**nexl expressions are viral**

nexl engine performs deep resolution for all nexl expressions are appeared in JavaScript strings. String items in arrays and objects ( including object keys ) which contains nexl expressions are also evaluated and substituted.

Let’s consider the following example

**A** = 3;

**B** = **'${A}'**;

**C** = **'${B}'**;

**D** = [1, 2, **'${C}'**];

**E** = {

**key1**: **'test'**,

**key2**: **'${D}'**,

**'${A}'**: **'${C}'**,

**key3**: **'hello \\${world}'**

};

B is a string variable which has a ${A} nexl expression. It will be evaluated to 3.

C is also string variable which is referencing to B variable. Will be evaluated to 3.

D is a JavaScript array where last element is string and has a nexl expression. This array will be evaluated to **D** = [1, 2, 3];

E is a JavaScript object. All keys and string values which contains nexl expressions will be processed and evaluated by nexl engine except ‘hello \\${world}’ string, because nexl expression is escaped here. E object will be evaluated to :

**E** = {

**key1**: **'test'**,

**key2**: [1, 2, 3],

3: 3,

**key3**: **'hello ${world}'**

};

**Include directive @**

nexl sources can include another nexl sources by using a @ directive. Let’s say you have two JavaScript files : common.js and server.js

server.js can include a common.js file in the following way :

“@ ../commons/common.js”;

Path is relative to a path of server.js file. Path can be absolute. Duplicate includings are ignored

**External args**

JavaScript variables in nexl source file can overridden by external arguments when performing a request to nexl REST server. You can provide a variable name and his value in HTTP request. Variable value can be only a primitive. If you provide a variable which doesn’t exist in nexl source it will be added to nexl source for that HTTP request. Variables are provided by external arguments are strings by default. It’s possible to cast them to needed type by adding the following to the end of their values :num, :str, :bool, :null, :undefined

Examples

|  |
| --- |
| [http://localhost:8080/example.js?expression=${person}](http://localhost:8080/example.js?expression=$%7Bperson.name%7D)  Just resolves a person object |
| [http://localhost:8080/example.js?expression=${person}&person.age=26](http://localhost:8080/example.js?expression=$%7Bperson%7D&person.age=26)  This is evaluated to the following object  {  name: **'Emily'**,  age: **'26'**,  country: **'Canada'**  };  By providing a person.age=26 argument person’s age is overridden with '26' string value.  If we need age to be a numeric we need cast a string value to numeric in the following way :  [http://localhost:8080/example.js?expression=${person}&person.age=26:num](http://localhost:8080/example.js?expression=$%7Bperson%7D&person.age=26:num)  Now we will get the following result  {  name: **'Emily'**,  age: **26**,  country: **'Canada'**  };  age field become numeric |
| [http://localhost:8080/example.js?expression=${person.${field}}&field=name](http://localhost:8080/example.js?expression=$%7Bperson.$%7Bfield%7D%7D&field=name)  In this example we are resolving a ${person.${field}} expression where ${field} is internal nexl expression. field value is provided by field=name external argument.  Result is  'Alex' |

Also external arguments are available by the nexl.args reference. You can use it in nexl expressions and in the functions.

**nexl.defaultExpression and nexl.defaultArgs**

When you are querying nexl REST server you pass the following in the URL :

* nexl source
* nexl expression
* optional external args

By using of nexl.defaultExpression and nexl.defaultArgs you can embed your nexl expression and args into nexl source file. Just add following in your nexl source :

**nexl**.defaultExpression = **'${...}'**;

**nexl**.**defaultArgs** = {**person**: {**name**: **'Emily'**}};

After that you can don’t need to provide an expression and arguments by the URL. For example :

<http://localhost:8080/example.js>

nexl expression and arguments explicitly provided in HTTP request are always override nexl.defaultExpression and nexl.defaultArgs

**nexl-server additional REST API**

nexl servers has an additional REST API

/nexl-rest/list-nexl-sources

Returns list of all nexl sources under the ${HOME}/nexl-sources directory (or under a directory you provided as command line argument for nexl-server)

/nexl-rest/list-js-variables?nexlSource=example.js

Returns list of all JavaScript variables for nexl source provided by nexlSource URL parameter

**Compatibility with nexl 1.x version**

nexl 2.x and 1.x versions are pretty different. It’s strongly recommended not to use a 1.x version because of security issues and escaping bugs.

If you have a 1.x version uninstall it in command line :

npm uninstall nexl-server -g

npm uninstall nexl-client -g

And then install latest version

npm i nexl-server -g

npm i nexl-client -g