

NGINX Conf 2018

The official event for all things NGINX



NGINX JavaScript in Your Web Server Configuration

Dmitry Volynsev
NGINX, Inc.



Outline

1

2

3

4

5



Outline

1 History of scripting in NGINX

2

3

4

5



Outline

1 History of scripting in NGINX

2 Project goals

3

4

5



Outline

1 History of scripting in NGINX

2 Project goals

3 njs interpreter

4

5



Outline

1 History of scripting in NGINX

2 Project goals

3 njs interpreter

4 Using njs in NGINX

5



Outline

1 History of scripting in NGINX

2 Project goals

3 njs interpreter

4 Using njs in NGINX

5 Plans for the future and available functionality



History of scripting in NGINX



Ideal scripting



Ideal scripting

- **Fast**

- Otherwise you can use much more advanced alternatives like node.js.



Ideal scripting

- **Fast**
 - Otherwise you can use much more advanced alternatives like node.js.
- **Integrate well with asynchronous nature of NGINX**



Ideal scripting

- **Fast**
 - Otherwise you can use much more advanced alternatives like node.js.
- **Integrate well with asynchronous nature of NGINX**
- **Modular**
 - So people who do not need it, can disable it to squeeze out performance.



Ideal scripting

- **Fast**
 - Otherwise you can use much more advanced alternatives like node.js.
- **Integrate well with asynchronous nature of NGINX**
- **Modular**
 - So people who do not need it, can disable it to squeeze out performance.
- **Popular scripting language**
 - To help people to write their scripts faster.



Perl scripting



Perl scripting

- Existing perl libraries can be used.



Perl scripting

- Existing perl libraries can be used.
- **Perl code can be embedded into nginx conf file.**



Perl scripting

- Existing perl libraries can be used.
- Perl code can be embedded into nginx conf file.
- **No support for non-blocking IO.**



Perl scripting

- Existing perl libraries can be used.
- Perl code can be embedded into nginx conf file.
- No support for non-blocking IO.
- Perl interpreter can exit the worker process.



Project goals



Goals

Modern, fast, high level scripting tailored to NGINX runtime.



Goals

Modern, fast, high level scripting tailored to NGINX runtime.

- **Fast and lightweight**

- njs should not degrade NGINX performance too much.
- memory/cpu overhead should not be substantial.



Goals

Modern, fast, high level scripting tailored to NGINX runtime.

- **Fast and lightweight**

- njs should not degrade NGINX performance too much.
- memory/cpu overhead should not be substantial.

- **Security/Robustness**

- Each request should be isolated from others.



Goals

Modern, fast, high level scripting tailored to NGINX runtime.

- **Fast and lightweight**

- njs should not degrade NGINX performance too much.
- memory/cpu overhead should not be substantial.

- **Security/Robustness**

- Each request should be isolated from others.

- **Popular scripting language**



Why JavaScript?



Why JavaScript?

- **Modern lingua-franca**
 - So, people can quickly understand it.



Why JavaScript?

- **Modern lingua-franca**
 - So, people can quickly understand it.
- **C-like syntax**
 - Good match for nginx config files.



Why JavaScript?

- **Modern lingua-franca**
 - So, people can quickly understand it.
- **C-like syntax**
 - Good match for nginx config files.
- **Event-driven paradigm is natural for JavaScript.**
 - Good match for NGINX runtime.



Why own interpreter?



Why own interpreter?

- **V8/SpiderMonkey are too heavy to be used inside NGINX.**
 - Sophisticated engines, too much overhead not needed in NGINX.



Why own interpreter?

- **V8/SpiderMonkey are too heavy to be used inside NGINX.**
 - Sophisticated engines, too much overhead not needed in NGINX.
- **Duktape is not fast enough for tasks inside NGINX.**
 - Has different sets of priorities. Values memory footprint and ECMAScript specs conformance more than performance.

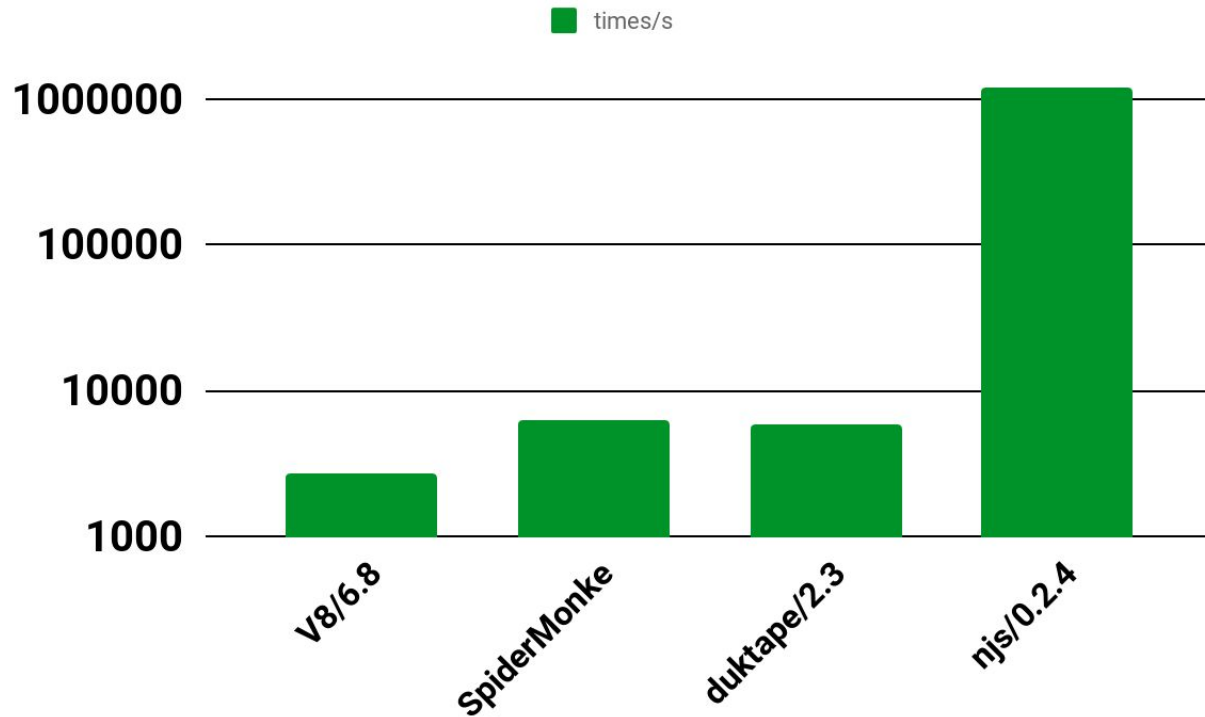


Why own interpreter?

- **V8/SpiderMonkey are too heavy to be used inside NGINX.**
 - Sophisticated engines, too much overhead not needed in NGINX.
- **Duktape is not fast enough for tasks inside NGINX.**
 - Has different sets of priorities. Values memory footprint and ECMAScript specs conformance more than performance.
- **Custom interpreter can be tailored to NGINX runtime.**



Created contexts/sec



njs interpreter



Why njs is fast?

NGINX modules



Why njs is fast?

NGINX modules

- **Bytecode compilation at start time.**

- `>> 1+1*2`
- `00000 MULTIPLY 1652F30 1652E10 1652F20`
- `00040 ADD 1652F30 1652E10 1652F30`
- `00080 STOP 1652F30`



Why njs is fast?

NGINX modules

- Bytecode compilation at start time.
 - `>> 1+1*2`
 - `00000 MULTIPLY 1652F30 1652E10 1652F20`
 - `00040 ADD 1652F30 1652E10 1652F30`
 - `00080 STOP 1652F30`
- **Copy-on-write cloning of compiled VM for each request.**
 - Fast creation and destroying of VMs.



Why njs is fast?

NGINX modules

- **Bytecode compilation at start time.**
 - `>> 1+1*2`
 - `00000 MULTIPLY 1652F30 1652E10 1652F20`
 - `00040 ADD 1652F30 1652E10 1652F30`
 - `00080 STOP 1652F30`
- **Copy-on-write cloning of compiled VM for each request.**
 - Fast creation and destroying of VMs.
- **No JIT Compilation**



Why njs is fast?

Interpreter



Why njs is fast?

Interpreter

- **Register based VM**
 - Small memory footprint.



Why njs is fast?

Interpreter

- **Register based VM.**
 - Small memory footprint.
- **UTF8 strings, bytes strings optimizations.**
 - ECMAScript specs require UTF-16.



Why njs is fast?

Interpreter

- **Register based VM.**
 - Small memory footprint.
- **UTF8 strings, bytes strings optimizations.**
 - ECMAScript specs require UTF-16.
- **Disabled garbage collection.**
 - Instead cloned VM is destroyed at once.



What njs is not



What njs is not

- **nginx + njs is not an application server.**
 - not “Node.js” replacement



What njs is not

- **nginx + njs is not an application server.**
 - not “Node.js” replacement
- **Strict ECMAScript specs conformance (in progress).**
 - Huge amount of work to do. Pareto principle.



Using njs



Installation



Installation

Adding nginx repo (Ubuntu/16.04)

```
wget http://nginx.org/keys/nginx\_signing.key -O nginx_signing.key  
sudo apt-key add nginx_signing.key
```

```
echo "deb http://nginx.org/packages/mainline/ubuntu/ xenial nginx" | sudo tee  
-a /etc/apt/sources.list
```



Installation

Adding nginx repo (Ubuntu/16.04)

```
wget http://nginx.org/keys/nginx\_signing.key -O nginx_signing.key  
sudo apt-key add nginx_signing.key
```

```
echo "deb http://nginx.org/packages/mainline/ubuntu/ xenial nginx" | sudo tee  
-a /etc/apt/sources.list
```

Installing package

```
apt-get update  
sudo apt-get install nginx nginx-module-njs
```



Installation

Adding nginx repo (Ubuntu/16.04)

```
wget http://nginx.org/keys/nginx_signing.key -O nginx_signing.key  
sudo apt-key add nginx_signing.key
```

```
echo "deb http://nginx.org/packages/mainline/ubuntu/ xenial nginx" | sudo tee  
-a /etc/apt/sources.list
```

Installing package

```
apt-get update  
sudo apt-get install nginx nginx-module-njs
```

Examples

<https://github.com/xeioex/njs-examples>



Hello world

`nginx.conf:`

```
load_module  
modules/nginx_http_js_module.so;
```



Hello world

nginx.conf:

```
load_module
modules/ngx_http_js_module.so;
...
http {
    js_include example.njs;
```



Hello world

nginx.conf:

```
load_module
modules/nginx_http_js_module.so;
...
http {
    js_include example.njs;

    server {
        listen 8000;

        location /hello {
            js_content hello;
        }
    }
}
...
```



Hello world

nginx.conf:

```
load_module
modules/nginx_http_js_module.so;
...
http {
    js_include example.njs;

    server {
        listen 8000;

        location /hello {
            js_content hello;
        }
    }
}
...
```

example.njs:

```
function hello(r) {
}
```



Hello world

nginx.conf:

```
load_module
modules/nginx_http_js_module.so;
...
http {
    js_include example.njs;

    server {
        listen 8000;

        location /hello {
            js_content hello;
        }
    }
...

```

example.njs:

```
function hello(r) {
    r.return(200, "Hello world!");
}

```



Injecting HTTP header

`nginx.conf:`

```
load_module  
modules/nginx_stream_js_module.so;
```



Injecting HTTP header

nginx.conf:

```
load_module  
modules/nginx_stream_js_module.so;  
...
```

```
stream {  
    js_include stream.js;
```

```
}
```



Injecting HTTP header

nginx.conf:

```
load_module
modules/nginx_stream_js_module.so;
...

stream {
    js_include stream.js;

    server {
        listen 12345;

        proxy_pass 127.0.0.1:8000;
        js_filter header_inject;
    }
}
```



Injecting HTTP header

nginx.conf:

```
load_module
modules/ngx_stream_js_module.so;
...
```

```
stream {
    js_include stream.js;

    server {
        listen 12345;

        proxy_pass 127.0.0.1:8000;
        js_filter header_inject;
    }
}
```

stream.js:

```
var my_header = 'Foo: foo';

function header_inject(s) {
    var req = '';
    s.on('upload', function(data, flags) {
        req += data;
    });
}
```



Injecting HTTP header

nginx.conf:

```
load_module
modules/ngx_stream_js_module.so;
...

stream {
    js_include stream.js;

    server {
        listen 12345;

        proxy_pass 127.0.0.1:8000;
        js_filter header_inject;
    }
}
```

stream.js:

```
var my_header = 'Foo: foo';

function header_inject(s) {
    var req = '';
    s.on('upload', function(data, flags) {
        req += data;
        var n = req.search('\n');
        if (n != -1) {
            req = req.substr(0, n);
            req += my_header + '\n';
        }
    });
}
```



Injecting HTTP header

nginx.conf:

```
load_module
modules/nginx_stream_js_module.so;
...

stream {
    js_include stream.js;

    server {
        listen 12345;

        proxy_pass 127.0.0.1:8000;
        js_filter header_inject;
    }
}
```

stream.js:

```
var my_header = 'Foo: foo';

function header_inject(s) {
    var req = '';
    s.on('upload', function(data, flags) {
        req += data;
        var n = req.search('\n');
        if (n != -1) {
            var rest = req.substr(n + 1);
            req = req.substr(0, n + 1);
            s.send(req + my_header + '\r\n' + rest,
                flags);
        }
    });
}
```



Injecting HTTP header

nginx.conf:

```
load_module
modules/ngx_stream_js_module.so;
...

stream {
    js_include stream.js;

    server {
        listen 12345;

        proxy_pass 127.0.0.1:8000;
        js_filter header_inject;
    }
}
```

stream.js:

```
var my_header = 'Foo: foo';

function header_inject(s) {
    var req = '';
    s.on('upload', function(data, flags) {
        req += data;
        var n = req.search('\n');
        if (n != -1) {
            var rest = req.substr(n + 1);
            req = req.substr(0, n + 1);
            s.send(req + my_header + '\r\n' + rest,
                flags);
            s.off('upload');
        }
    });
}
```



Joining subrequests

nginx.conf:

```
location /join {  
    js_content join;  
}
```

```
location /foo {  
    proxy_pass http://backend1;  
}
```

```
location /bar {  
    proxy_pass http://backend2;  
}
```



Joining subrequests

nginx.conf:

```
location /join {  
    js_content join;  
}
```

```
location /foo {  
    proxy_pass http://backend1;  
}
```

```
location /bar {  
    proxy_pass http://backend2;  
}
```

example.js:

```
function join(r) {  
    join_subrequests(r, ['/foo', '/bar']);  
}
```



Joining subrequests

nginx.conf:

```
location /join {
    js_content join;
}

location /foo {
    proxy_pass http://backend1;
}

location /bar {
    proxy_pass http://backend2;
}
```

example.js:

```
function join(r) {
    join_subrequests(r, ['/foo', '/bar']);
}

function join_subrequests(r, subs) {

    for (var i in subs) { r.subrequest(subs[i], done);}
};
```



Joining subrequests

nginx.conf:

```
location /join {
    js_content join;
}

location /foo {
    proxy_pass http://backend1;
}

location /bar {
    proxy_pass http://backend2;
}
```

example.js:

```
function join(r) {
    join_subrequests(r, ['/foo', '/bar']);
}

function join_subrequests(r, subs) {
    var parts = [];

    function done(reply) {
        parts.push({ uri: reply.uri,
                    body: reply.responseBody });
    }

    for (var i in subs) { r.subrequest(subs[i], done); }
};
```



Joining subrequests

nginx.conf:

```
location /join {
    js_content join;
}

location /foo {
    proxy_pass http://backend1;
}

location /bar {
    proxy_pass http://backend2;
}
```

example.js:

```
function join(r) {
    join_subrequests(r, ['/foo', '/bar']);
}

function join_subrequests(r, subs) {
    var parts = [];

    function done(reply) {
        parts.push({ uri: reply.uri,
                    body: reply.responseBody });
    }

    if (parts.length == subs.length) {
        r.return(200, JSON.stringify(parts));
    }

    for (var i in subs) { r.subrequest(subs[i], done);}
};
```



Command line

```
docker run -i -t nginx:mainline /usr/bin/njs
```



Command line

```
docker run -i -t nginx:mainline /usr/bin/njs
```

```
>> [{a:[Date()]}]  
[  
  {  
    a: 'Sun Sep 23 2018 19:15:17 GMT+0000 (UTC)'  
  }  
]
```



Command line

```
docker run -i -t nginx:mainline /usr/bin/njs
```

```
>> [{a:[Date()]}]
```

```
[  
  {  
    a: 'Sun Sep 23 2018 19:15:17 GMT+0000 (UTC)'  
  }  
]
```

```
>> require('crypto').createHash("sha1").update("XX").digest("hex")  
'20026dc165c030fe3a5d9609a6e61ab26210cbc1'
```



Command line

```
docker run -i -t nginx:mainline /usr/bin/njs
```

```
>> [{a:[Date()]}]
[
  {
    a: 'Sun Sep 23 2018 19:15:17 GMT+0000 (UTC)'
  }
]
>> require('crypto').createHash("sha1").update("XX").digest("hex")
'20026dc165c030fe3a5d9609a6e61ab26210cbc1'
>> (function(o) {return o.a.a})();
TypeError: cannot get property 'a' of undefined
at anonymous (:1)
at main (native)
```



Available functionality



What is available

- **Object, Array, Number, String, Date, Regexp, Function**
- **JSON, Math**



What is available

- Object, Array, Number, String, Date, Regexp, Function
- JSON, Math
- **exceptions**



What is available

- Object, Array, Number, String, Date, Regexp, Function
- JSON, Math
- exceptions
- **closures, anonymous functions**



What is available

- Object, Array, Number, String, Date, Regexp, Function
- JSON, Math
- exceptions
- closures, anonymous functions
- **crypto, files ops and more**



What is available

- Object, Array, Number, String, Date, RegExp, Function
 - JSON, Math
 - exceptions
 - closures, anonymous functions
 - crypto, files ops and more
- `eval()`



What is available

- Object, Array, Number, String, Date, RegExp, Function
 - JSON, Math
 - exceptions
 - closures, anonymous functions
 - crypto, files ops and more
- `eval()`
 - `let, const`
 - `arrow functions`



What is available

- Object, Array, Number, String, Date, RegExp, Function
 - JSON, Math
 - exceptions
 - closures, anonymous functions
 - crypto, files ops and more
- `eval()`
 - `let, const`
 - `arrow functions`
 - `modules`



Plans



Plans

- **More integration with NGINX**
 - **Embedding njs into NGINX conf files directly.**



Plans

- **More integration with NGINX**
 - Embedding njs into NGINX conf files directly.
 - **Extending feature set of the modules.**



Plans

- **More integration with NGINX**
 - Embedding njs into NGINX conf files directly.
 - Extending feature set of the modules.
- **njs development**
 - **Extending ECMAScript specs conformance.**



Plans

- **More integration with NGINX**
 - Embedding njs into NGINX conf files directly.
 - Extending feature set of the modules.
- **njs development**
 - Extending ECMAScript specs conformance.
 - **Modules support.**



NGINX

Thank you

- **Github:** <https://github.com/nginx/njs>
- **Examples:** <https://github.com/xeioex/njs-examples>

Dmitry Volyntsev
xeioex@nginx.com



Execution model

`example.njs:`

```
var time = new Date();
```

```
function variable_handler(r) {  
    return (Date.now() - time).toString();  
}
```

```
function content_handler(r) {  
    r.return(200, "Delay: " + (Date.now() - time));  
}
```



Decode URI

nginx.conf:

```
...  
http {  
    js_include example.njs;
```



Decode URI

nginx.conf:

```
...  
http {  
    js_include example.njs;  
  
    js_set $dec_foo dec_foo;
```



Decode URI

nginx.conf:

```
...  
http {  
    js_include example.njs;  
  
    js_set $dec_foo dec_foo;  
  
    server {  
        listen 8000;  
  
        location /dec_foo {  
            return 200 $dec_foo;  
        }  
    }  
...  
...
```



Decode URI

nginx.conf:

```
...  
http {  
    js_include example.njs;  
  
    js_set $dec_foo dec_foo;  
  
    server {  
        listen 8000;  
  
        location /dec_foo {  
            return 200 $dec_foo;  
        }  
  
    }  
...  
}
```

example.njs:

```
function dec_foo(r) {  
    decodeURIComponent(r.args.foo);  
}
```

