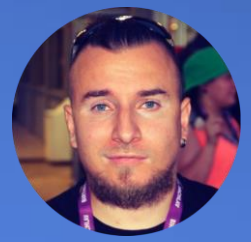


Тихон Усков

ZABBIX Инженер команды интеграции

РЕШАЕМ ПРАКТИЧЕСКИЕ ЗАДАЧИ МОНИТОРИНГА

С ПОМОЩЬЮ JAVASCRIPT

НЕМНОГО ИСТОРИИ

02.04.2019
Zabbix 4.2 - Появилась предобработка на JavaScript

07.10.2019
Zabbix 4.4 - Появился Webhook media type

**Администраторы Zabbix после релиза 4.2*

ПОЧЕМУ JAVASCRIPT И DUKTAPE?

Оценивались следующие языки / движки

- Lua – Lua 5.1
- Lua – LuaJIT
- Javascript – Duktape
- Javascript – JerryScript
- Embedded Python
- Embedded Perl

Выбран JavaScript на движке Duktape:

- Распространенность JS
- Простота встраивания
- Низкое потребление ресурсов и производительность
- Безопасность

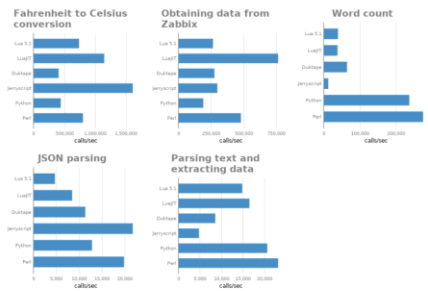
Особенности Duktape:

- Стандарт ECMAScript E5/E5.1
- Доработки Zabbix:
  - Zabbix.log()
  - CurlHttpRequest()
  - atob() и btoa()

*Zabbix blog: Javascript support in item preprocessing

	Lua		Javascript (ECMAScript)		Python	Perl
	Lua 5.1	LuaJIT	Duktape	Jerryscript	CPython 2.7	Perl 5.x
Language features	Might need third party libraries (json), no native regular expression support (lua patterns are similar, but less powerful).	Might need third party libraries (json), no native regular expression support (lua patterns are similar, but less powerful).	No out of the box support for file access, network functionality. Additonal features must be provided with third party or custom modules.	No out of the box support for file access, network functionality. Additonal features must be provided with third party or custom modules.	Wide collection of various modules.	Wide collection of various modules.
Language feature limitation	It's possible to disable 'dangerous' modules and functions, but might still be accessible if engine internals are known.	It's possible to disable 'dangerous' modules and functions, but might still be accessible if engine internals are known.	No file/network access is available by default.	No file/network access is available by default.	Disabling modules is possible by overriding the default import function, however, it would still be possible to import those modules by using a low level python howto.	Disabling modules is possible by overriding the default import function, however, it would still be possible to import those modules by using a low level python howto.
Engine integration	Link to lua and a few common libraries.	Link to lua and a few common libraries.	Distributed as single C and two header files to be included in project.	Must compile jerryscript libraries from sources.	Link to python and a few common libraries.	Link to perl and a few common libraries.
Resource limits	Can limit memory usage with custom allocator and CPU usage by limiting CPU cycles.	Can limit CPU usage by limiting CPU cycles. Custom memory allocator is not supported.	Can limit memory usage with custom allocator and CPU usage by setting script timeout.	JerryScript does not provide means to limit memory usage, but it seems to have default 512KB heap limit. Execution timeouts are not supported yet.	The memory and CPU usage can be limited with resources module. However that seems to be *nix specific, based on process ulimits.	Only OS based process resource limits can be applied.
Script precompilation and loading	Supports bytecode precompilation and loading.	Supports bytecode precompilation and loading.	Supports bytecode precompilation and loading.	Supports bytecode precompilation and loading.	Supports function marshaling/unmarshaling.	No bytecode precompilation/ loading is supported.
Notes				Crashed when used an insufficient sized buffer when compiling the script.	Indent based code structure, hard to write more complex code in just text area. Would need more advanced editor.	
Simple script performance						
Complex script performance						
Engine initialization performance						
Script compilation performance						

Performance (precompiled code):



МАГИЯ JAVASCRIPT

Вся магия заключена в динамической типизации и приведении типов.

```
var obj = { toString() { return "200" } }

obj +  1  // '2001'

obj + 'a' // ‘200a'
```

// и совсем другой результат

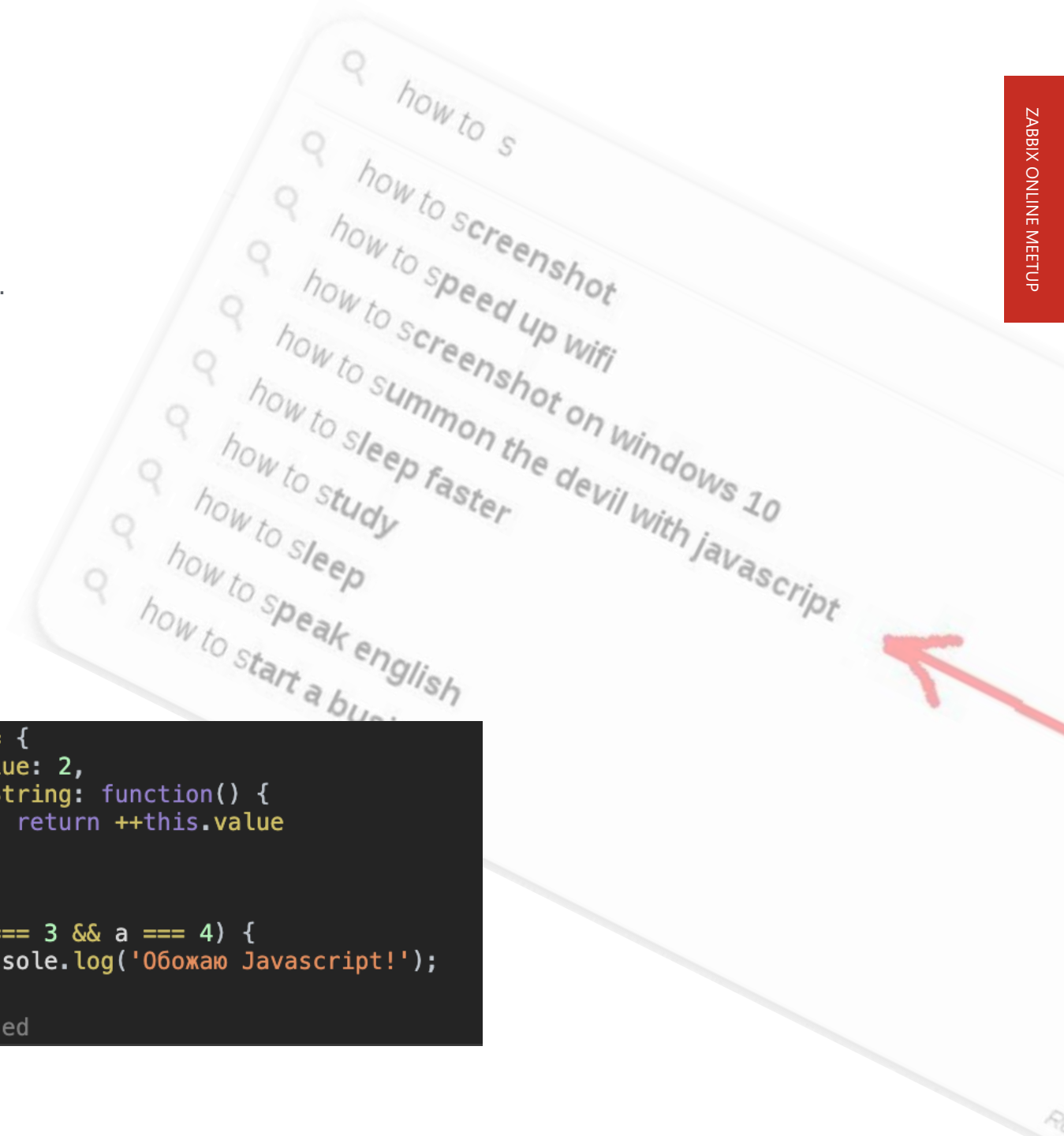
```
var obj = { toString() { return 200 } }
```

```
> var a = {
      value: 2,
      toString: function() {
          return ++this.value
      }
  }

  if (a == 3 && a == 4) {
      console.log('Обожаю Javascript!');
  }
  Обожаю Javascript!
```

```
> var a = {
      value: 2,
      toString: function() {
          return ++this.value
      }
  }

  if (a === 3 && a === 4) {
      console.log('Обожаю Javascript!');
  }
< undefined
```



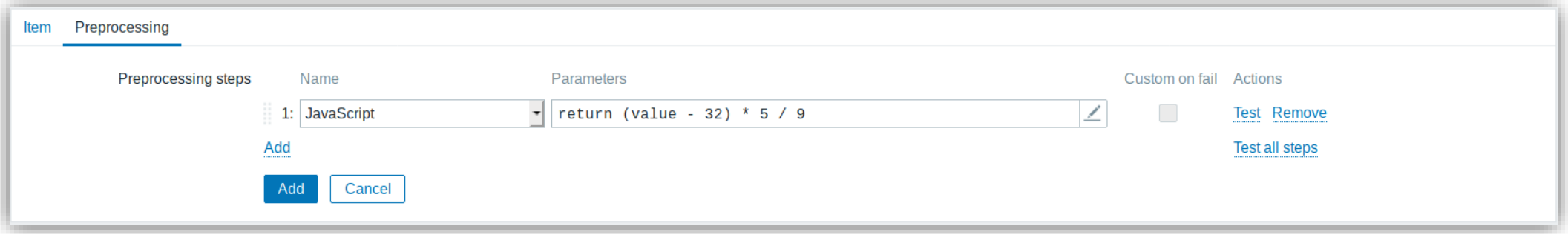
WEBHOOK MEDIA

- Discord
- Jira
- Jira Service Desk
- Mattermost
- Microsoft Teams
- Opsgenie
- OTRS
- Pagerduty
- Pushover
- Redmine
- ServiceNow
- SIGNL4
- Slack
- Telegram
- Zammad
- Zendesk

https://www.zabbix.com/documentation/current/manual/config/notifications/media/webhook

PREPROCESSING

- Мощнейший инструмент для преобразования полученных значений
- Принимает значение в параметр *‘value’. Value* - всегда строка
- В конце скрипта обязателен return
- Позволяет использовать пользовательские макросы в коде
- Имеет таймаут 10 секунд
- Имеет лимит на 10Мб heap memory



Проверить свой код можно через тест предобработки или с помощью утилиты **zabbix_js**

Usage

```
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V
```

PREPROCESSING: ПРАКТИКА

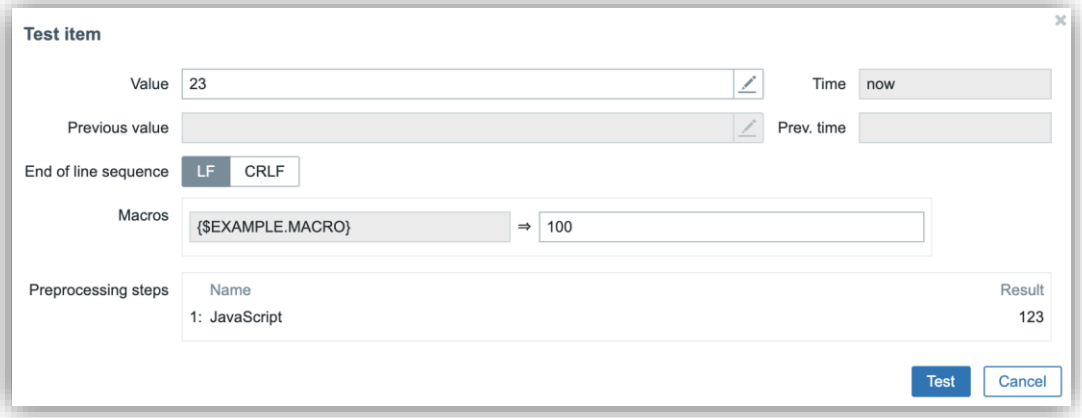
Задача: заменить вычисляемый элемент данных предобработкой.
Условие: получаем с датчика температуру в градусах Фаренгейта, а хранить хотим в градусах Цельсия.

Решение:

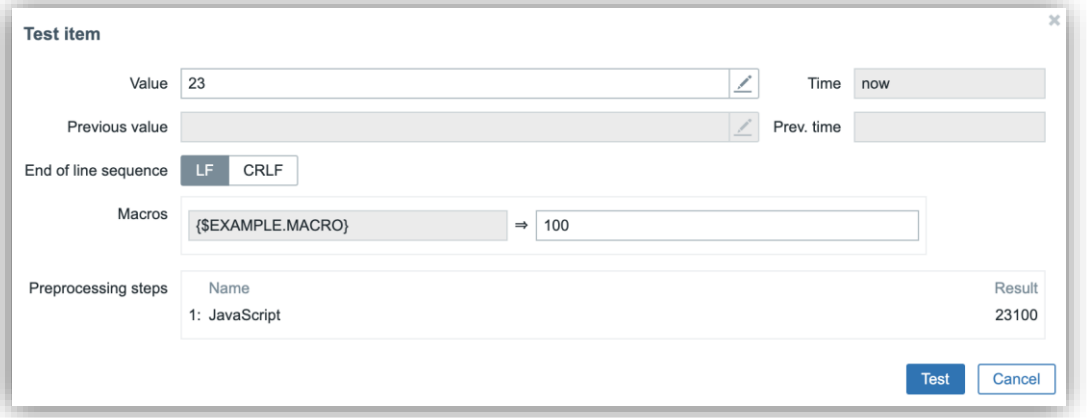
```
return (value - 32) * 5 / 9;
```

Но если нам требуется сложить значение с постоянным значением, определяемым макросом:

```
return (parseInt(value) + parseInt("{$EXAMPLE.MACRO}"));
```




```
return (value + "{$EXAMPLE.MACRO}");
```



PREPROCESSING: ПРАКТИКА

Задача: получить время в секундах до окончания сертификата.
Условие: некий сервис отдает дату окончания сертфиката «Feb 12 12:33:56 2022 GMT».

> В ECMAScript5 Date.parse() принимает дату в формате ISO 8601 (YYYY-MM-DDTHH:mm:ss.sssZ). Необходимо привести к нему строку в фомате MMM DD YYYY HH:mm:ss ZZ

Решение:

```
var split = value.split(' '),
    MONTHS_LIST = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    month_index = ('0' + (MONTHS_LIST.indexOf(split[0]) + 1)).slice(-2),
    ISOdate = split[3] + '-' + month_index + '-' + split[1] + 'T' + split[2],
    now = Date.now();

return parseInt((Date.parse(ISOdate) - now) / 1000);
```

THANK YOU!