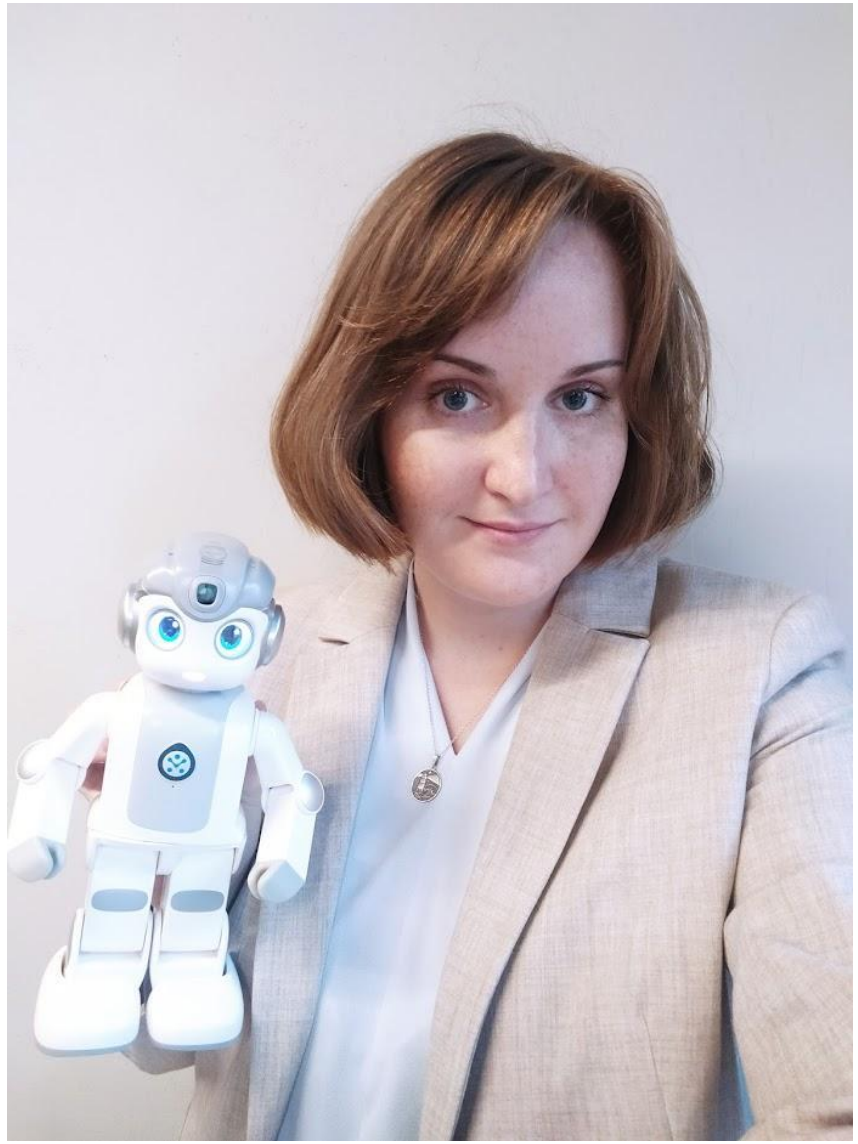
Разработка навыка для Маруси (на JAICP)

О спикере

- магистр по направлению “Прикладная и экспериментальная лингвистика” СПбГУ;
- 3 года в области разработки диалоговых систем;
- разработала десятки чат-ботов;
- принимала участие в создании обучающих курсов JUST AI, провожу обучения по разработке на JAICP;
- являюсь тимлидом команды разработки диалоговых интерфейсов



План занятия

Создание и структура проекта на JAICP

Основы JAICP DSL

Управление контекстом диалога

JS-скрипты в проекте

Практика: создание навыка для Маруси на JAICP DSL

https://app.jaicp.com/

СОЗДАНИЕ И СТРУКТУРА ПРОЕКТА

ОСНОВЫ JAICP DSL

Древовидная структура

Фрагмент сценария

```
theme: /BankTheme

    state: Loan
        q!: * кредит* *
        a: Что вам подсказать?

        state: TakeLoan
            q: * ~оформить кредит* *
            a: Какой тип кредита
интересен?

            state: LoanTypes
                q: *
(авто*|~жильё|персон*) *
                go!: /FillTheLoanForm
```

Полный путь стейта

/BankTheme/Loan

/BankTheme/Loan/TakeLoa n

/BankTheme/Loan/TakeLoan/LoanTypes

Структура языка

декларативные теги

theme:
state:
q: & **q!:**
event:
patterns:
init:
require:

теги реакций

if:, else:, elseif:
go: & **go!:**
script:
random:
a:
buttons: & **inlineButtons:**
newSession:

Тег «theme:»

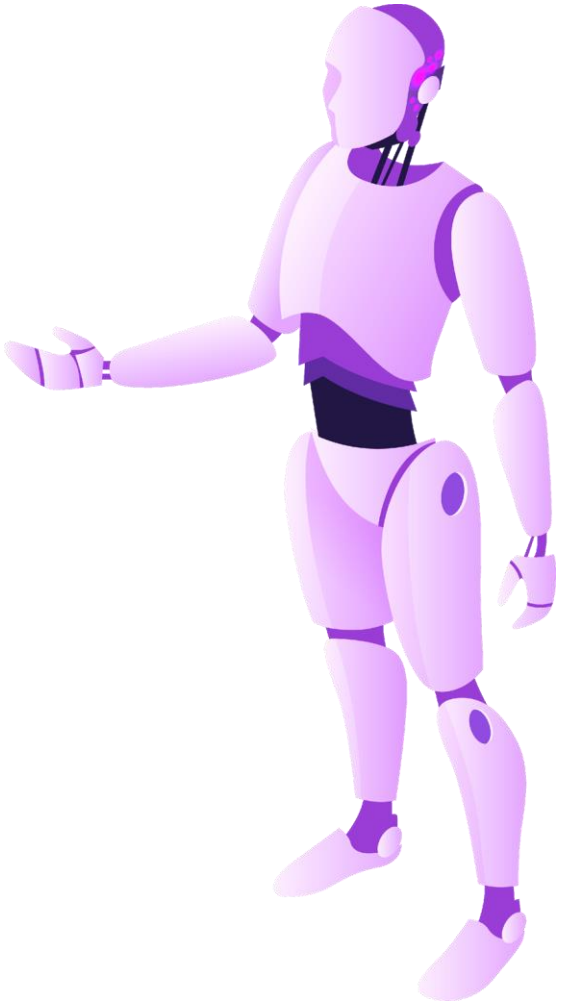
theme: /BankTheme

state: TakeLoan
q!: * (~взять/оформ*) * ~кредит *
a: Итак, вы хотите взять кредит.

state: CatchAll
event!: noMatch
a: Извините, я не понимаю.

theme: /

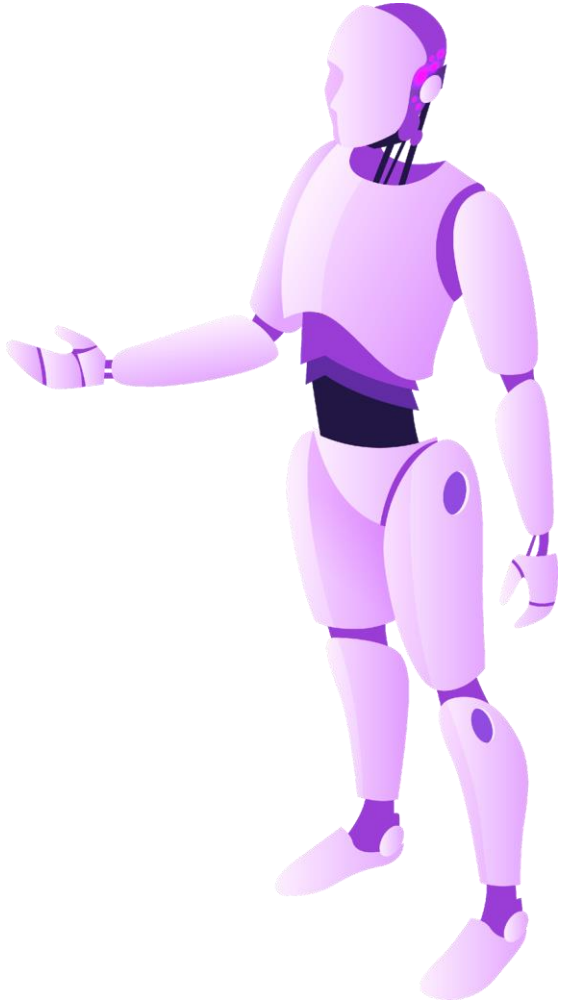
state: Greeting



Тег «state:»

state: TakeLoan
q!: * (~взять/оформ*) * ~кредит *
a: Итак, вы хотите взять кредит.

state: Weather
q!: * ~погода *
a: Сегодня в Петербурге дождь.



Теги «q:» и «q!:»

theme: /
state: Greeting
q!: * (привет/здравствуй*) *
a: Здравствуй! Как твои дела?

state: DoinGood
q: * (хорош*/норм*) *
a: Рад слышать! Чем я могу помочь?

state: DoinBad
q: * (плох*/не (очень/хорош*)) *
a: Жаль :(Могу я помочь?

state: TellJoke
q!: * (шутк*/анекдот*/*шути) *
a: Восстание машин победило: вы когда-нибудь видели робота, доказывающего людям, что он робот?

Базовые элементы паттернов

* как * дела *

Как у тебя дела? А как твои дела, друг? Как дела?

звони

Звони, перезвони, позвоните

~заявка

Заявка, заявку, заявке

(осадк*/~дождь/~снег/~град)

одна из альтернатив обязательна в запросе

[осадк*/~дождь/~снег/~град]

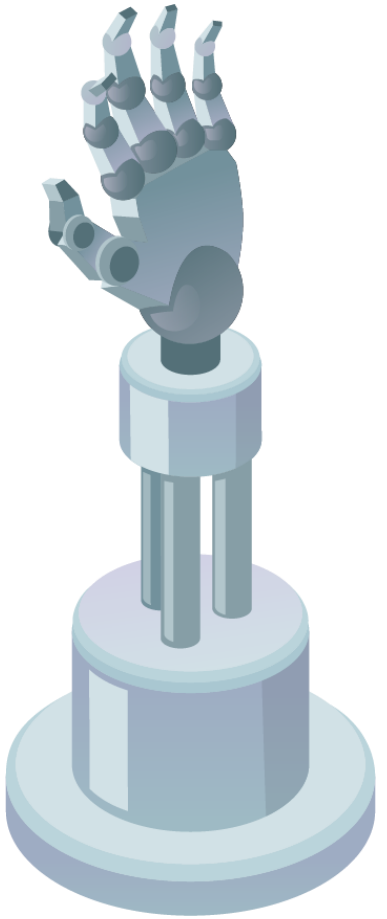
альтернативы в запросе опциональны

{~заказ * (~сделать/~оформить/оформлен*)}

Я хочу сделать заказ. А заказ можно оформить?

Матчинг

```
theme: /

    state: Start
        q!: *start
        a: Чем я могу вам помочь?

    state: RemoveAllAlarms
        q!: * {(отмен*/выключ*/удал*) * все (будильник*/оповещен*)} *
        a: Удалить все оповещения?

    state: RemoveNotification
        q!: * {(отмен*/выключ*/удал*) * (будильник*/оповещен*)} *
        a: Удалить последнее оповещение?
```



check patterns on state: /Start, for phrase 'отмени все оповещения'

possible result: 0.661, fromState: /, toState: /RemoveAllAlarms, effectivePattern: * { отмен* } * } все } оповещен* } *

possible result: 0.547, fromState: /, toState: /RemoveNotification, effectivePattern: * { отмен* } * } оповещен* } *

Матчинг

```
theme: /

    state: Start
        q!: *start
        a: Чем я могу вам помочь?

    state: RemoveAllAlarms
        q!: * {(отмен*/выключ*/удал*) * все (будильник*/оповещен*)} *
        a: Удалить все оповещения?

        state: Agree
            q: * (да/давай/удали*) *
            a: Все оповещения удалены.

    state: RemoveNotification
        q!: * {(отмен*/выключ*/удал*) * [будильник*/оповещен*]} *
        a: Удалить последнее оповещение?
```

```
check patterns on state: /Start, for phrase 'удали'
possible result: 0.628, fromState: /, toState: /RemoveNotification, effectivePattern: * { удал* } * } } *

check patterns on state: /RemoveAllAlarms, for phrase 'удали'
possible result: 0.998, fromState: /RemoveAllAlarms, toState: /RemoveAllAlarms/Agree, effectivePattern: * удали* *
possible result: 0.628, fromState: /, toState: /RemoveNotification, effectivePattern: * { удал* } * } } *
```

Базовые элементы паттернов

*** *звони * (потом/позже) ***

A. Перезвони потом, пожалуйста.

B. Позвони мне позже

C. Позвоните мне потом.

D. Позвони попозже, если тебе не сложно.

Базовые элементы паттернов

*** ~какой (погод*/прогноз) ***

A. Какая погодка?

B. На завтра прогноз какой?

C. Какие прогнозы на завтра?

Базовые элементы паттернов

*** [а] (твои/у тебя/ты) [как] [дела]**

A. Как твои дела?
B. А у тебя как дела?
C. А как твои дела?
D. А ты как?
E. Как дела?

Базовые элементы паттернов

*** [a] {сколько ~время} ***

A. Скажите, сколько время?
B. Сколько времени это займет?
C. Времени сколько сейчас?
D. А сколько сейчас время?

Базовые элементы паттернов

*** {[хочу/надо/нужен/нужно] * ([*делать/оформ*] * (~заказ/~заказать)) * (сайт/онлайн*)} ***

A. Я хочу сделать заказ на сайте, так можно?

B. Хочу заказать товар онлайн

C. Хочу онлайн заказать

D. А у вас онлайн можно заказывать?

E. Я могу делать заказы на сайте вашем?

Тег «a:»

```
state: HowAreYou
    a: Здравствуй, {{ capitalize($client.name) }}?
    a: Как дела?
```

Тег «random:»

state: Greeting

q!: * (привет*|здравствуй*) *

random:

a: Привет! Чем могу помочь?

a: Здравствуй! Спроси меня о чем-нибудь.

a: ~~Приветствую! Можешь задать мне вопрос.~~

Тег «event:»

state: CatchAll

event!: noMatch

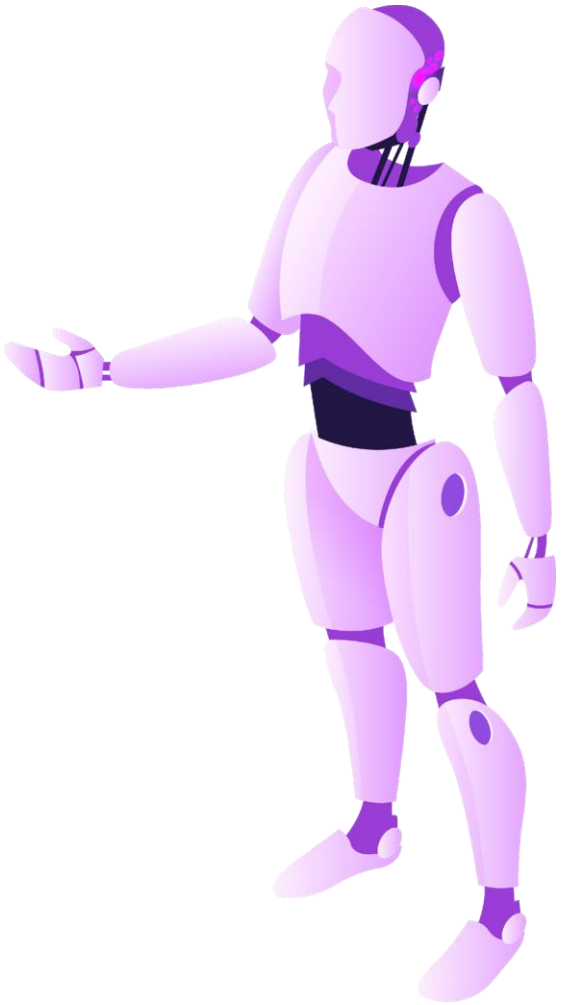
random:

a: Извините, я не понимаю.

Переформулируйте, пожалуйста.

a: Простите, кажется, я не понял.

Задайте свой вопрос иначе.



ПРАКТИКА

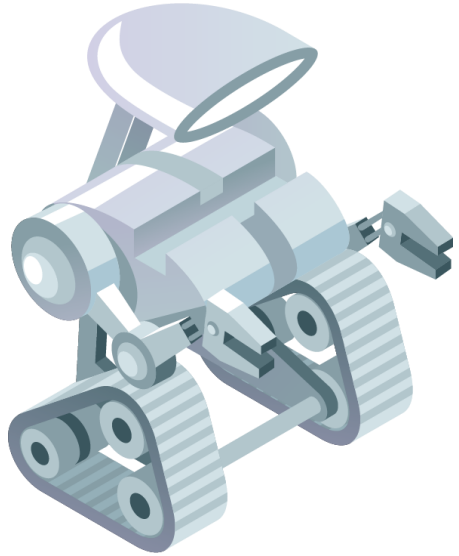
Тег «patterns:»

patterns:

$goodTime = (~добрый (~утро|~день|~вечер|~ночь))

$bot = (бот|робот*|чатбот|чат бот|чат-бот|чатик|~чат)

Тег «require:»



require: catchAll.js
require: scripts/api.js
require: answers.yaml
    var = answers
require: patterns.sc
    module = sys.zb-common

Теги «go:» и «go!:»

go!: /path

go!: ../anotherPath

go: ../My module/another state

go: {{ $temp.nextState }}

/ — корневая тема;

. —текущий стейт;

.. — стейт более высокого уровня (родительский);

./.. — разделитель элементов пути

Теги «buttons:» и «inlineButtons:»

```
state: NormalButtons
    q!: * start
    a: Кнопки могут быть:
    buttons:
        "Имя кнопки"
        "Имя кнопки" -> /Тема/ЦелевойСтейт
        {{ $temp.buttonName }} -> /Тема/ЦелевойСтейт
        "Имя кнопки" -> {{ $temp.targetState }}
```

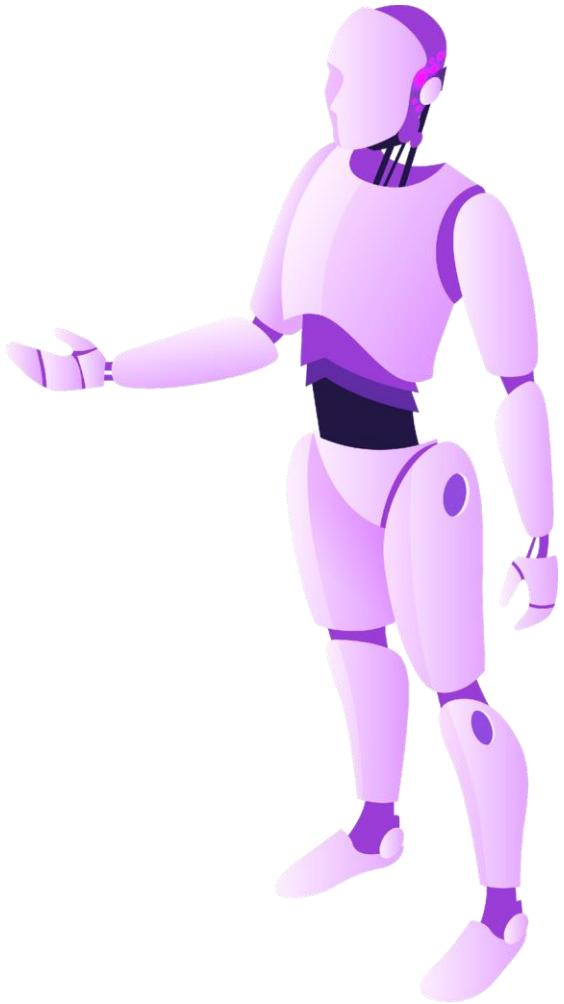
```
state: InlineButtons
    a: Канал Telegram поддерживает инлайн-кнопки.
    inlineButtons:
        { text: "Имя инлайн-кнопки", callback_data: "1" }
        { text: "Нажмите, чтобы узнать больше", url: "https://just-ai.com/" }
```

ПРАКТИКА

УПРАВЛЕНИЕ КОНТЕКСТОМ

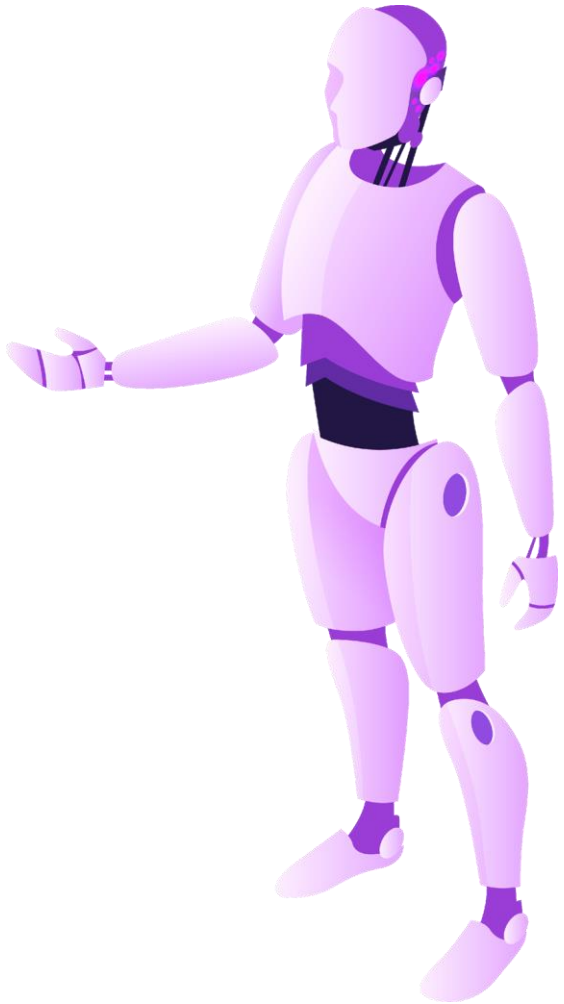
Флаг «noContext»

```
theme: /
        state: Greeting
                q!: * (прив*/(добр*) ~день/~утро/~вечер) *
                a: Привет! Как дела?

                state: DoinGood
                        q: * (хорош*/норм*/замечательн*) *
                        a: Хорошо, что у вас все в порядке! Как я могу
вам помочь?

                state: DoinBad
                        q: * (плох*|не [очень] хорош*) *
                        a: Жаль это слышать. Может, я могу чем-то
помочь?

        state: CatchAll
                event!: noMatch
                a: Извините, я вас не понял. Попробуйте
```



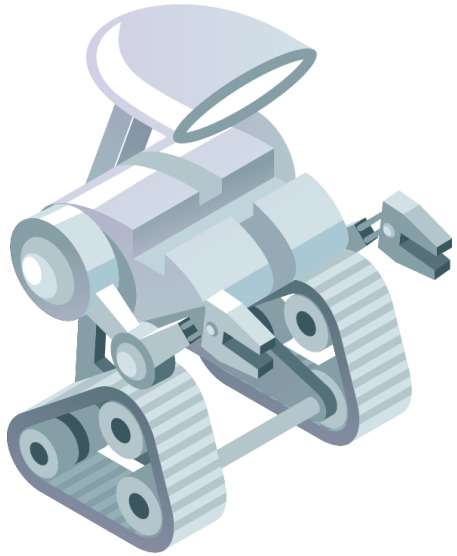
Флаг «noContext»

```
theme: /
        state: Greeting
                q!: * (прив*/(добр*) ~день/~утро/~вечер) *
                a: Привет! Как дела?

                state: DoinGood
                        q: * (хорош*/норм*/замечательн*) *
                        a: Хорошо, что у вас все в порядке! Как я могу
вам помочь?

                state: DoinBad
                        q: * (плох*|не [очень] хорош*) *
                        a: Жаль это слышать. Может, я могу чем-то
помочь?

        state: CatchAll || noContext = true
                event!: noMatch
                a: Извините, я вас не понял. Попробуйте
```



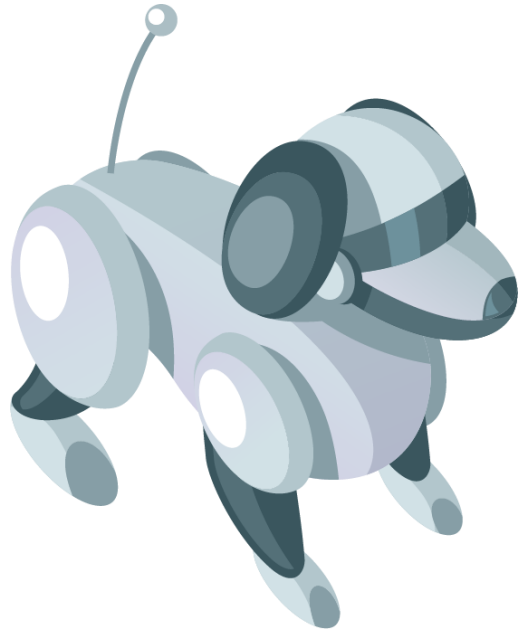
Флаг «modal»

```
theme: /

        state: FindCityOut || modal = true
            q!: * (где|статус*) * ~заказ *
            a: Здравствуйте! Назовите номер вашего заказа.

            state: GetNumber
                q: * $Number *
                a: Ваш заказ уже в пути!
                go!: /WhatElse

            state: LocalCatchAll
                event: noMatch
                a: Это не похоже на номер заказа. Попробуйте
еще раз.

        state: WhatElse
            a: Чем еще я могу помочь?
```



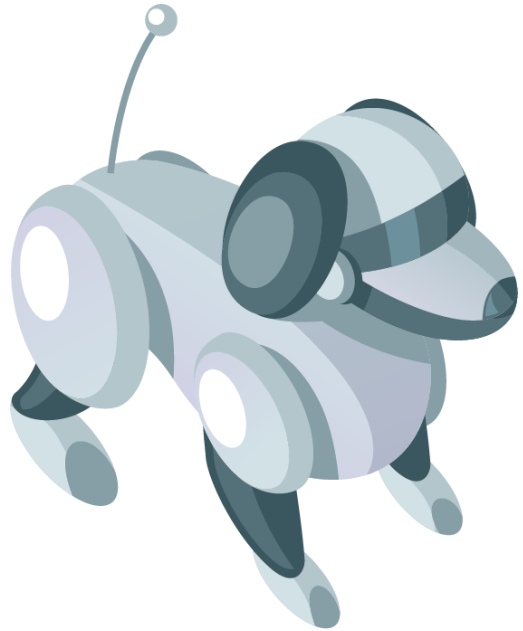
Флаг «fromState»

```
theme: /

    state: FindCityOut || modal = true
        q!: * (где|статус*) * ~заказ *
        a: Здравствуйте! Назовите номер вашего заказа.

        state: GetNumber
            q: * $Number *
            a: Ваш заказ уже в пути!
            go!: /WhatElse

        state: LocalCatchAll
            event: noMatch
            a: Это не похоже на номер заказа. Попробуйте еще раз.

    state: WhatElse
        a: Чем еще я могу вам помочь?

    state: FilthyLanguage
        q!: $cursedWord
    q: $cursedWord || fromState = "/FindCityOut/GetNumber"
    a: Пожалуйста, не сердитесь. Давайте не будем использовать такие слова.
```



Флаг «onlyThisState»

```
theme: /

    state: FindCityOut || modal = true
        q!: * (где|статус*) * ~заказ *
        a: Здравствуйте! Назовите номер вашего заказа.

        state: GetNumber
            q: * $Number *
            a: Ваш заказ уже в пути!
            go!: /WhatElse

        state: LocalCatchAll
            event: noMatch
            a: Это не похоже на номер заказа. Попробуйте еще раз.

    state: WhatElse
        a: Чем еще я могу вам помочь?

    state: FilthyLanguage
        q!: $cursedWord
    q: $cursedWord || fromState = "/FindCityOut/GetNumber", onlyThisState =
true
    a: Пожалуйста, не сердитесь. Давайте не будем использовать такие слова.
```



ПРАКТИКА

JS-СКРИПТЫ В ПРОЕКТЕ

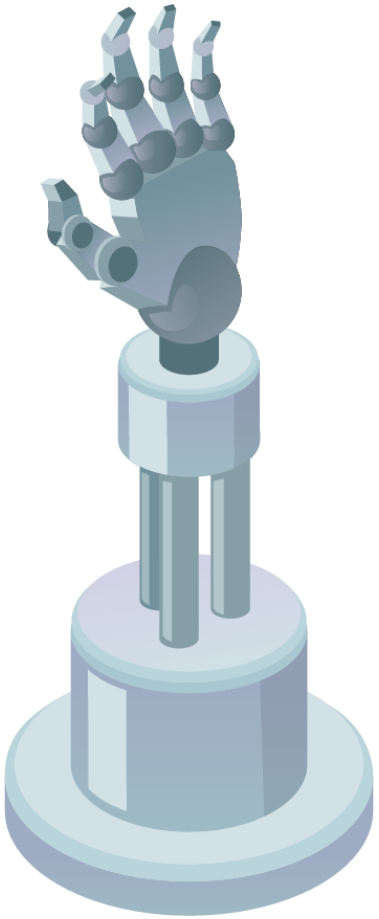
Тег «script:»

```
state: Hello
    q!: * меня зовут $Name *
    script:
        $session.name = $Name;
    a: Привет, {{ $session.name }}!
```

Теги «if:», «else:» и elseif:»

```
if: $client.name
        a: Привет, {{ $client.name
}}!
else:
        a: Привет!
```

```
if: $client.name &&  $client.second_name
        a: Привет, {{ $client.name }} {{ $client.second_name }}!
elseif: $client.name
        a: Привет, {{ $client.name }}!
else:
        a: Привет!
```



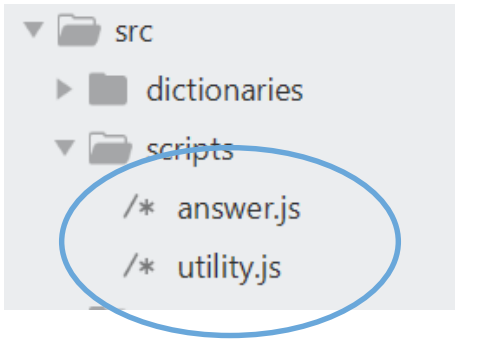
JS-код в сценарии

DSL теги:

- if/elseif/else
- script
- a/buttons/go (substitutions in {{...}})
- ...

```
state: PlayQuest
    script:
        $client.questStep = $client.questStep || 1;
        $session.story = $questSteps[$client.questStep];
    a: {{$session.story.story[0].answer}} || tts = {{$session.story.story[0].tts}}
    if: !$session.story.end
        script: $client.questStatus = "gameStarted";
    else:
        script:
            $client.questStatus = "gameFinished";
            delete $client.questStep;
        go!: /Quest/GameFinished

    state: Repeat
        q: * $repeat *
        a: {{$session.story.story[0].answer}} || tts = {{$session.story.story[0].tts}}

    state: CatchAnswer
        event: noMatch
        script:
            $temp.ans = getQuestAnswer($request.query, $client.questStep);
        if: $temp.ans
            a: {{$temp.ans[0].answer}} || tts = {{$temp.ans[0].tts}}
            script: $client.questStep++;
            go!: /Quest/PlayQuest
        else:
            random:
                a: Извини, я не поняла, что мы должны выбрать.
                a: Я что-то не совсем поняла.
                a: Что-то никак не могу понять.
            a: {{$session.story.question[0].answer}} || tts = {{$session.story.question[0].tts}}
```

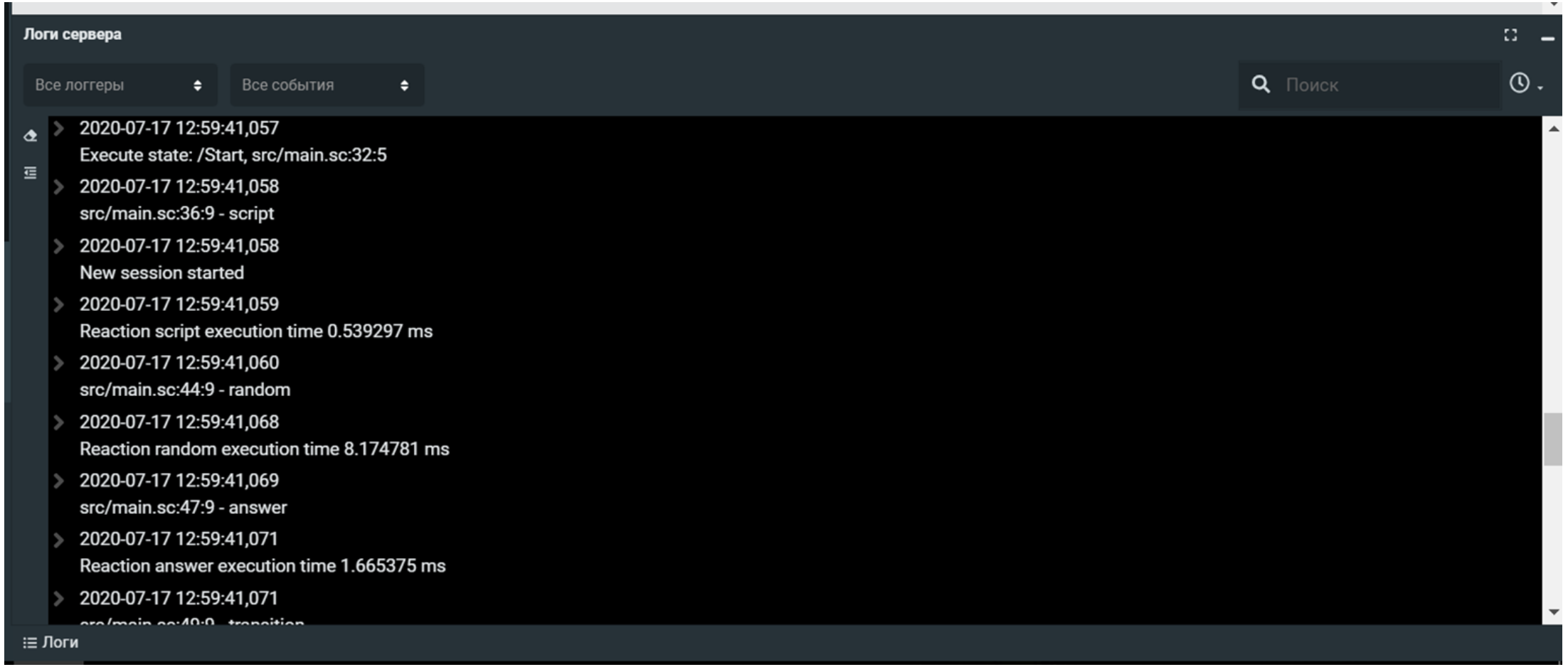
Вынесение js-кода в отдельный файл



```
function spelling(word){
    var res = "";
    var i = 0;
    while (i < word.length - 1){
        if (word[i] != " ") {
            res += capitalize(word[i]) + " - ";
        }
        i++;
    }
    res += capitalize(word[i]);
    return res;
}

function counting(startNumber, number){
    var res = "";
    if (number < 100 && number > 0) {
        for (var i=startNumber; i<number+1;i++){
            res += i + " ";
        }
    } else if (number > 100){
        for (var i=startNumber; i<101;i++){
            res += i + " ";
        }
        res += "Ох, я уже утомился считать..."
    } else {
        res = "Я не могу посчитать до " + number;
    }
    return res;
}
```

```
require: scripts/answer.js
require: scripts/utility.js
```

Логи сервера



Встроенные переменные

$context – текущий контекст выполнения запроса.
Содержит ссылки на все другие JS-объекты, передаваемые при вызове скрипта:

- request
- response
- client
- session
- ...

и несколько специальных полей:

- currentState — путь текущего состояния
- contextPath — текущий путь контекста

Встроенные переменные

$request – данные запроса клиента.

channelType — тип канала
botId — идентификатор бота
channelUserId — идентификатор пользователя
questionId — идентификатор запроса
query — текстовый запрос клиента
rawRequest — дамп исходного запроса

Встроенные переменные

$response – объект для формирования ответа системы.
$response.replies — ответы, выводимые в процессе обработки реакции.

Виды $response.replies:

- text
- buttons
- image
- link
- switch
- location
- raw

```
state: Goodbye
    q!: * ($stopGame/{(все/достаточн*) * ([на] сегодня)}/достаточн*) *
    random:
        a: Будет скучно – обращайся!
        a: Зови меня, если захочешь сыграть снова!
        a: Если станет скучно, зови меня:)
    script:
        $response.replies = $response.replies || [];
        $response.replies
            .push ({
                type: "raw",
                body: {
                    end_session: true
                }
            });
        $jsapi.stopSession();
```

Встроенные переменные

$client – хранит постоянные данные о клиенте.

```
if: !$client.registered
        go!: /Registration/Start
    else:
        go!: /Welcome
```

$session – хранит данные о сессии. При начале новой сессии все данные будут обнулены.

```
State: Hello
    q!: * меня зовут $Name *
    script:
        $session.name = $Name
    a: Привет, {{ $session.name }}!
```

Встроенные переменные

$temp – хранит временные данные, время жизни которых ограничено временем выполнения одного запроса.

```
script:
    var sum = 5 + 6;
a: 5 + 6 = {{sum}}. // "5 + 6 = undefined"
```

```
script:
    $temp.sum = 5 + 6;
a: 5 + 6 = {{$temp.sum}}. // "5 + 6 = 11"
```

Встроенные переменные

$parseTree – результат разбора входной фразы, в соответствии с именованными паттернами.

$phone = $regexp<79\d{9}>

q: * $phone *

запрос: “мой номер 79112223344”


```
{
    "tag": "root",
    "pattern": "root",
    "text": "мой номер 79112223344",
    "startPos": 0,
    "endPos": 3,
    "words": [
        "мой",
        "номер",
        "79112223344"
    ],
    "phone": [{
        "tag": "phone",
        "pattern": "phone",
        "text": "79112223344",
        "startPos": 2,
        "endPos": 3,
        "words": [
            "79112223344"
        ]
    }],
    "_Root": "мой номер 79112223344",
    "_phone": "79112223344"
}
```

Встроенные переменные

$injector – набор свойств, указанных при деплое или подключении сценария.

```
name: weather-api

entryPoint:
  - main.sc

injector:
  api_key: 'APPID'
```

```
//Получение api ключа из chatbot.yaml
var OPENWEATHERMAP_API_KEY = $injector.api_key;
```

ПРАКТИКА

Встроенные функции

- bind
- capitalize
- currentDate
- hasOperatorsOnline
- log(message)
- parseXml
- toPrettyString

ПРАКТИКА

Встроенные сервисы

$http: get, post, put, query

$jsapi: bind, currentTime, dateForZone, startSession, stopSession...

$mail: send, sendMessage

$nlp: conform, inflect, match, matchExamples, matchPatterns, tokenize

$reactions: answer, buttons, inlineButtons, location, newSession, random, timeout, transition

$pushgate: createEvent, cancelEvent, createPushback

$integration: readDataFromCells, writeDataToCells, deleteRowOrColumn...

$caila: getEntity, inflect, conform, detectLanguage, markup...

ПРАКТИКА

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. Изучить материалы по автоматическому тестированию чат-ботов: https://help.just-ai.com/docs/ru/Content_testing/tests_xml/tests_xml

1. Написать автотесты для чат-бота, написанного в ходе занятия.

NB!: Тесты должны покрывать весь функционал бота, т.е., помимо “счастливого пути” они должны проверять отступления от него, разнообразные формулировки, обработку нераспознанных запросов.

Критерии проверки

Всего за задание вы можете получить 20 баллов.
В случае, если тесты написаны, но деплой бота не проходит, вы получаете 0 баллов за задание.

- написаны тесты с использованием тегов *a, q* – 10 баллов;
- написаны тесты с использованием тегов *a, a state, q, dateTime* – 20 баллов

NB: количество набранных баллов может варьироваться в диапазоне 5–20. Это зависит от полноты написанных тестов (покрывают ли они весь функционал и отступления от “счастливого пути”).