Курс по JavaScript. 2021

19 июн 2022, 18:07:40
старт: 27 сен 2021, 14:53:11
начало: 1 сен 2020, 00:00:00

> Мы очень хотим, чтобы код вы написали сами, а не пользовались внешними библиотеками.

Аркадий – серьезный человек и в его телефонной книге очень много записей. Чтобы как можно меньше отрываться от дел, Аркадий делает голосовую систему "Василиса", которая будет заполнять телефонную книгу за него. К счастью, Василиса уже умеет распознавать голос и превращать его в текст, осталось научить её понимать и выполнять команды. Общеизвестный **SQL** использует английский язык, а Аркадий и Василиса говорят на русском. Так появился язык, который Аркадий назвал **pbQL** (Phone Book Query Language).

Аркадий придумал спецификацию **pbQL**, а за первой реализацией обратился к вам.

Синтаксис языка pbQL

Язык состоит из нескольких команд:

`Создай контакт <имя>`

- Создает новый контакт с именем `<имя>` с пустыми списками телефонов и почт
- Принимает `<имя>` содержащее любые символы, кроме `;`
- Не создает ничего, если контакт уже существует

Пример: `Создай контакт Григорий;`

`Удали контакт <имя>`

- Удаляет контакт с именем `<имя>`
- Принимает `<имя>` содержащее любые символы, кроме `;`
- Не удаляет ничего, если контакт не существует

Пример: `Удали контакт Григорий;`

`Добавь телефон <телефон> и почту <почта> для контакта <имя>`

- Добавляет `<телефон>` в спискок телефонов и `<почту>` в список почт для контакта `<имя>`
- Принимает телефоны только в формате 5556667788 (без кода), иначе это считается ошибкой синтаксиса
- Принимает почты без пробелов, поэтому через пробел ожидается следующее слово
- Принимает произвольное количество почт и телефонов, перечисленных через `и`
- Не добавляет ничего, если контакт не существует
- Не добавляет почту/телефон, если такая почта/телефон уже есть у контакта

Пример: `Добавь телефон 5556667788 и телефон 5556667787 и почту grisha@example.com для контакта Григорий;`

`Удали телефон <телефон> и почту <почта> для контакта <имя>`

- Удаляет `<телефон>` из списка телефонов и `<почту>` из списка почт контакта `<имя>`
- Принимает телефоны только в формате 5556667788 (без кода), иначе это считается ошибкой синтаксиса
- Принимает почты без пробелов, иначе это считается ошибкой синтаксиса
- Принимает произвольное количество почт и телефонов, перечисленных через `и`
- Не удаляет ничего, если контакт не существует
- Не удаляет почту/телефон, если такая почта/телефон отсутствует у контакта

Пример: `Удали телефон 5556667788 для контакта Григорий;`

`Покажи почты и телефоны для контактов, где есть <запрос>`

- Ищет вхождение `<запрос>` хотя бы в один из телефонов, либо в одну из почт, либо в имя контакта
- Принимает `<запрос>` содержащий любые символы, кроме `;`
- Принимает для перечисления произвольное количество типов полей, перечисленных через `и`, среди которых могут быть `имя`, `почты` и `телефоны`
- Возвращает список строк в формате `<почта 1>,<почта 2>;<телефон 1>` в соответствии с запращиваемыми полями, для подходящих под запрос контактов
- Возвращает контакты в порядке создания, а их почты/телефоны в порядке добавления
- Возвращает имена и электронные почты как есть, а телефоны в виде `+7 (555) 666-77-88`
- Не возвращает ничего на пустой запрос (`Покажи имя для контактов, где есть ;`)

Пример: `Покажи имя и почты и телефоны и почты для контактов, где есть Гр;`

`Удали контакты, где есть <запрос>`

- Ищет вхождение `<запрос>` хотя бы в один из телефонов, либо в одну из почт, либо в имя контакта
- Принимает `<запрос>` содержащий любые символы, кроме `;`
- Удаляет все подходящие контакты
- Не удаляет ничего на пустой запрос (`Удали контакты, где есть ;`)

Пример: `Удали контакты, где есть Гр;`

Другие особенности синтаксиса

- `pbQL` регистрозависимый язык, поэтому `покажи` не является правильной командой
- Команды в запросе разделяются и заканчиваются `;`. `Создай контакт Григорий;Удали контакт Григорий;` – правильный запрос
- Каждое слово отделяется ровно одним пробелом, поэтому большое количество подряд идущих пробелов это ошибка синтаксиса
- `<запрос>` не является частью языка, а является аргументом, поэтому может содержать любое количество подряд идущих пробелов

Ваша функция run должна

- Принимать на вход запрос в виде строки
- Возвращать массив строк с ответом
- Конкатенировать в один массив результаты всех `Покажи`
- Вызывать вспомогательную функцию **syntaxError** на первую обнаруженную ошибку синтаксиса
- Вызывать **syntaxError** с двумя аргументами – номер команды и номер символа, с которого начинается ошибка синтаксиса
- Считать команды и символы с единицы, а не с нуля

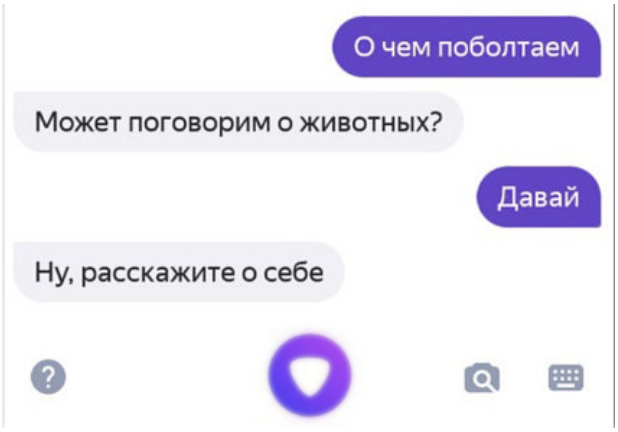
Примеры ошибок синтаксиса

- `покажи имя для контактов, где есть Гр;` – команда с маленькой буквы, ошибка с символа 1
- `Покжи имя для контактов, где есть Гр;` – опечатка в команде, ошибка с символа 1
- `Покажи имя для контактов, где есть Гр;` – лишний пробел, ошибка с символа 8 (второй пробел)
- `Удали телефон 55566677 для контакта Григорий;` – недостаточно цифр в телефоне, ошибка с символа 15 (первый символ телефона)
- `Покажи имя для контактов, где есть Гр` – отсутствует точка с запятой, ошибка с символа 38 (отсутствующий символ в конце команды)

Используйте заготовку для написания своего кода.

Полезные ссылки

- Знакомимся с массивами
- Пытаемся знакомиться с регулярными выражениями
- Перебираем ключи объектов
- Метод indexOf для строк
- Метод slice для строк
- Объект Map



Примеры:

```
const pbql = require('pbql.js');

// Пример 1
pbql.run('Покажи имя для контактов, где есть ий;')
/*
    []
*/
```

```
// Пример 2
pbql.run(
    'Создай контакт Григорий;' +
    'Создай контакт Василий;' +
    'Создай контакт Иннокентий;' +
    'Покажи имя для контактов, где есть ий;'
)
/*
    [
        'Григорий',
        'Василий',
        'Иннокентий'
    ]
*/

// Пример 3
pbql.run(
    'Создай контакт Григорий;' +
    'Создай контакт Василий;' +
    'Создай контакт Иннокентий;' +
    'Покажи имя и имя и имя для контактов, где есть ий;'
)
/*
    [
        'Григорий;Григорий;Григорий',
        'Василий;Василий;Василий',
        'Иннокентий;Иннокентий;Иннокентий'
    ]
*/

// Пример 4
pbql.run(
    'Создай контакт Григорий;' +
    'Покажи имя для контактов, где есть ий;' +
    'Покажи имя для контактов, где есть ий;'
)
/*
    [
        'Григорий',
        'Григорий'
    ]
*/

// Пример 5
pbql.run(
    'Создай контакт Григорий;' +
    'Удали контакт Григорий;' +
    'Покажи имя для контактов, где есть ий;'
)
/*
    []
*/

// Пример 6
pbql.run(
    'Создай контакт Григорий;' +
    'Добавь телефон 5556667787 для контакта Григорий;' +
    'Добавь телефон 5556667788 и почту grisha@example.com для контакта Григорий;' +
    'Покажи имя и телефоны и почты для контактов, где есть ий;'
)
/*
    [
        'Григорий;+7 (555) 666-77-87,+7 (555) 666-77-88;grisha@example.com'
    ]
*/

// Пример 7
pbql.run(
    'Создай контакт Григорий;' +
    'Добавь телефон 5556667788 для контакта Григорий;' +
    'Удали телефон 5556667788 для контакта Григорий;' +
```

```
    'Покажи имя и телефоны для контактов, где есть ий;'
)
/*
    [
        'Григорий;'
    ]
*/
```

Набрать здесь | Отправить файл

```
'use strict';

/**
 * Регулярное выражение задающее шаблон Email по стандарту RFC 5322.
 * @type {RegExp}
 */
const regExpEmail = /[^\s]+ /;

/**
 * Регулярное выражение задающее шаблон номера телефона.
 * @type {RegExp}
 */
const regExpPhoneNumber = /\d{10} /

/**
 * Регулярное выражение задающее шаблон имени в телефонной книге.
 * @type {RegExp}
 */
const regExpName = /([^;]*)/

/**
 * Функция, отвечающая является ли аргумент строкой.
 * @param {string} string - проверяемый аргумент.
 * @returns boolean - true если аргумент является строкой, иначе false.
 */
function isString(string) {
    return typeof string === "string";
}

/**
 * Функция, отвечающая является ли аргумент целым числом.
 * @param {string} string - проверяемый аргумент.
 * @returns boolean - true если аргумент является целым числом, иначе false.
 */
function isInteger(string) {
    return Number.isInteger(Number(string));
}

```

Отправить

Предыдущая | Следующая