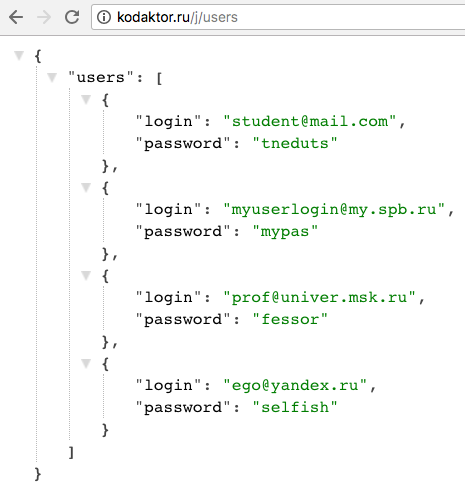
**Курс: Javascript. Событийно-ориентированное программирование**

**Тема 6. Создание объектов. Лабораторная/самостоятельная работа.**

Такие инструменты как https://jsonlint.com/ позволяют проверить корректность формы JSON-документа. Если же мы хотим зафиксировать определённую структуру и потребовать, чтобы те или иные ключи могли иметь те или иные типы значений, мы должны создать схему. Сверку документа с его схемой будем называть валидацией.

Рассмотрим документ JSON по адресу kodaktor.ru/j/users



мы видим, что значением ключа users является массив объектов вида {"login":..., "password":...}

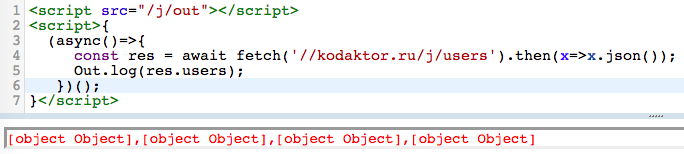
Он соответствует схеме:



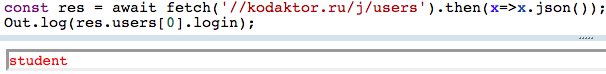
Находящиеся в массиве объекты можно в некотором смысле описать как «безымянные», потому что в той структуре, которая представлена выше, у них нет какого-либо явного различающего признака, гарантирующего уникальность. В массиве могут оказаться объекты с полностью совпадающими ключами и значениями.

***Примечание***

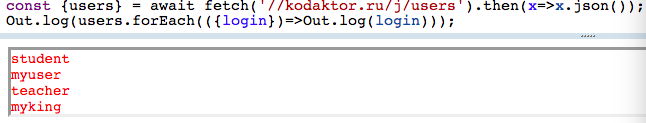
*Значения полей «логин» и «пароль» в документе по указанному адресу могут изменяться. Например, логином может быть просто слово student или teacher, а может быть простой адрес электронной почты. Это никак не меняет задачу и суть дела.*



Но так как это массив, то мы, разумеется, можем обратиться к его элементам по индексам



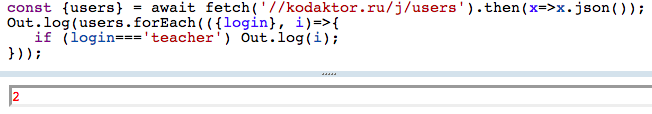
или с помощью итерации:



Однако что, если мы хотим устроить поиск по такой совокупности данных? Как ответить на вопросы: «Есть ли такой логин?» и «Какой пароль у пользователя с таким логином?»?

Предположим, что ключ login является первичным ключом и однозначно определяет каждый объект; а пароли могут и совпадать.

Мы можем сравнивать перебираемые логины:



Недостатком такого подхода является то, что перебор продолжается и после нахождения логина - т.е. независимо от того, найден он или нет.

Здесь мы можем либо

(а) использовать **функциональный** подход на примере метода *find*

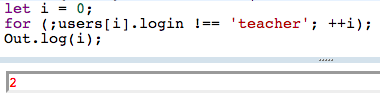
../../Скриншоты/2018-01-22_00-59-08.png

(получим найденный элемент ../../Скриншоты/2018-01-22_01-05-50.png)

или *filter*:

(получим массив найденных элементов, в том числе пустой при ненахождении)

либо (б) перебирать **императивно** - т.е. циклом



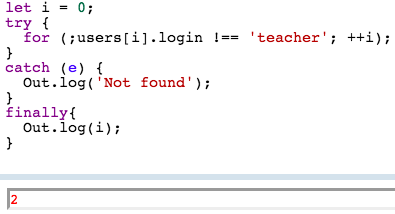
этот подход небезопасен: если искомого логина не окажется, мы выйдем за пределы размерности массива, т.к. счётчик будет увеличен и достигнет значения, на единицу большего, чем последний индекс (или равного длине массива, что то же самое):

/Users/eliasgoss/Yandex.Disk.localized/Скриншоты/2018-01-21_00-23-49.png

/Users/eliasgoss/Yandex.Disk.localized/Скриншоты/2018-01-21_00-24-22.png

Эту проблему можно разрешить, включив обработку исключения:

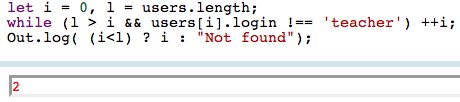
(https://kodaktor.ru/json\_f0bfc)



но лучше сделать так, чтобы исключение не возникало

В нижеследующем фрагменте может быть выведен только один из двух вариантов: корректный индекс найденного элемента или вердикт о ненайденности:

https://kodaktor.ru/json\_57e2b



цикл продолжается пока верны оба условия.

если элемент присутствует, то первое условие (l > i) выполняется автоматически, а второе условие перестаёт выполняться при нахождении элемента и приращение ++i тоже не выполняется, т.е. индекс верный.

если элемент отсутствует, то второе словие выполняется автоматически, а первое перестаёт выполняться когда весь массив перебран и i становится равным его длине. Иными словами, если цикл закончился и при этом неверно что i<l, значит элемент не найден, в противном случае выводится верный индекс.

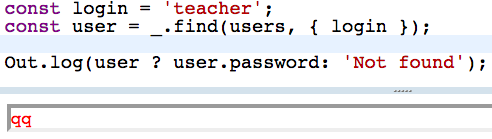
Массив перебирается целиком только в одном случае — когда элемента в нём нет.

===

**Поиск с помощью lodash**

Эта библиотека появилась до того, как в массивах JavaScript был реализован метод find. Если в нативном методе используется функция-предикат, то в lodash критерий поиска задаётся объектом

https://kodaktor.ru/json\_ceb5b



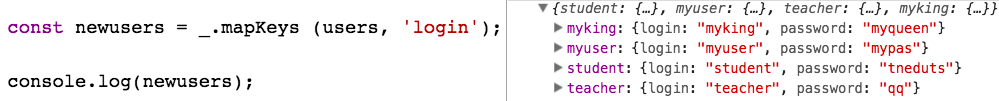
Чтобы удалить объект с логином teacher, например, нужно выполнить:

../../Скриншоты/2018-01-21_01-29-35.png

стрелочная лямбда играет роль предиката-отсеивателя, или фильтра: если она возвращает true, то такой объект удаляется.

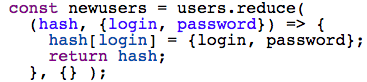
При этом метод \_remove возвращает массив удалённых объектов, который окажется пустым, если подходящих для удаления объектов в массиве не нашлось.

Если мы рассматриваем ключ как первичный, т.е. например считаем, что логины уникальны, то можем сгенерировать объект, состоящий из объектов, у которых будут уникальные ключи, имена которых будут равны значениям этого первичного ключа:



Можно построить своё решение на чистом JavaScript с использованием метода reduce:

https://kodaktor.ru/json\_c1656



Получив такой объект, можно использовать оператор in:

../../Скриншоты/2018-01-21_01-49-40.png

**Итак, создайте страницу со сценарием, который ищет в указанном массиве запись по значению указанного логина и удаляет её, после чего выводит на страницу список оставшихся логинов/паролей.**