Стажировка в Тинькофф

Предлагаем вам выполнить несколько тестовых заданий с возврастающим уровнем сложности. Всего таких уровня три, где 1 - самый простой и 3 - самый сложный.

В каждом представленном задании указан уровень сложности, есть описание и пример.

Если на каком-то этапе выполнения любого задания у вас возникли непреодолимые сложности, то напишите комментарий на каком моменте вы остановились и почему.

Просто прибавь

Сложность: 1

Описание

Необходимо реализовать метод, который будет принимать два аргумента. Первый аргумент - массив, состоящий из цифр от 0 до 9. Второй - любое целое положительное число. Метод должен возвращать массив из цифр от 0 до 9, который получается путем прибавления второго аргумента к числу, составленного из элементов массива. Если в массиве есть элементы, не являющиеся числами от 0 до 9, метод должен возвращать `null`.

Пример

```
function justAddOne(array, number) {
  // Реализация метода
}

justAddOne([1, 0, 9], 2); // 109 + 2 = 111; => [1, 1, 1]
justAddOne([2, 5, 1], 5); // 251 + 5 = 256; => [2, 5, 6]
justAddOne([1], 4020); // 1 + 4020 = 4021; => [4, 0, 2, 1]
justAddOne([1, '4', 11, null], 1); // '4' - строка и есть null => null
```

Обратите внимание

- Корректными в массиве могут быть только числа от 0 до 9.
- Число, которое можно прибавить, может быть любым целым положительным.
- Если значение в массиве не является числом от 0 до 9, то должен возвращаться `null`.

Форматирование операций

Сложность: 2

Описание

Необходимо реализовать метод, который принимает данные в формате JSON и возращает отформатированную определенным образом строку.

Пример

Имеются следующие данные:

```
[
  {
    "name": "Ashlynn Hartmann",
    "cardNumber": "4929289137092267",
    "date": "2019-01-24T17:39:07.347Z",
    "amount": "579.63",
    "currency": "$"
  },
  {
    "name": "Philip Stoltenberg",
    "cardNumber": "4916258329158678",
    "date": "2018-09-07T02:21:03.144Z",
    "amount": "10472.99",
    "currency": "$"
  }
]
```

Необходимо получить следующий результат:

```
Имя покупателя: Ashlynn Hartmann
Номер карты: 4929 **** **** 2267
Дата и время операции: 24.01.2019 22:39
Сумма операции: $579.63

Имя покупателя: Philip Stoltenberg
Номер карты: 4916 **** **** 8678
Дата и время операции: 07.09.2018 07:21
Сумма операции: $10,472.99
```

Обратите внимание:

- 8 цифр номера карты по середине скрыты.
- Сумма операции показана по американской системе.
- Необходимо соблюсти форматирование, куда входит корректное расставление пробелов, формата операции оплаты и даты.

Параллельные вычисления

Сложность: 2

Описание

Необходимо реализовать метод, который принимает два аргумента. Первый аргумент - массив синхронных функций, а второй - функция любого типа. Метод должен вызывать функции из первого аргумента и затем, когда они все выполнятся, запустить функцию, переданную вторым аргументом. В фукнции из массива необходимо передать `callback`, который вызывается по окончании выполнения.

Пример

```
function first(callback) {
  setTimeout(callback, 1000);
```

```
}

function second(callback) {
  setTimeout(_ => {
    // Пример. Здесь можент быть логика метода.
    callback();
  });
}

function mainFunction() {}

function parallelComputing([first, second], mainFunction) {
  // Реализация метода
  // После того, как first и second будут выполнены
  mainFunction();
}
```

Обратите внимание

- В методы, которые передаются в массиве, обязательно должен быть передан метод обратного вызова `callback`.
- Количество методов в массиве может быть любым. И в каждом есть `callback`.
- Метод `parallelComputing` должен после выполнения вызвать метод, переданный вторым аргументом.

Создаем базу данных!

Сложность: 3

Описание

Необходимо сформировать маленькую базу данных. Поэтапно:

1. Получить 10 пользователей с помощью запроса `https://reqres.in/api/users/{ID юзера}`. В данном API ID юзеров начинаются с 1.
2. Модифицировать данные так, чтобы у каждого пользователя был свой новый уникальный ID, объединить его имя и фамилию.
3. Загрузить его аватар, сохранить в директорию и прописать в полученные данные корректный локальный адрес.
4. Сохранить сфомированную базу данных в отдельный файл в формате JSON.

Пример

Допустим, мы получаем двух пользователей:

```
const users = [
  {
    id: 2,
    first_name: 'Janet',
    last_name: 'Weaver',
    avatar: 'https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg'
  },
  {
```

```
    id: 12,
    first_name: 'Rachel',
    last_name: 'Howell',
    avatar:
      'https://s3.amazonaws.com/uifaces/faces/twitter/hebertialmeida/128.jpg'
  }
];
```

На выходе мы должны получить следующую структуру данных:

```
db/
  images/
    Rachel_Howell.jpg
    Janet_Weaver.jpg
  data.json
```

Содержимое файла `data.json`:

```
[
  {
    "id": "uiasd2135ksad",
    "name": "Janet Weaver",
    "avatar": "./images/Janet_Weather.jpg"
  },
  {
    "id": "askd12pnjkkj213",
    "name": "Rachel Howell",
    "avatar": "./images/Rachel_Howell.jpg"
  }
]
```

Обратите внимание

Исходя из способа реализации, это может быть как отдельный скрипт на Node.js, так и страница на фронтенде, но в этом случае стоит сделать запуск скрипта по кнопке.

- ID у пользователей должен быть уникальным.
- Реализовать можно как на Node.js, так и на фронтенде.
- Необходимо пользоваться либо VanillaJS, либо стандартными библиотеками Node.js.

Если не получится сделать все этапы, то в этом нет ничего страшного. Сделайте все, что сможете по максимуму. Можете написать комментарии к своему коду.