Стажировка 2022: фронтенд

E. Асинхронное получение данных

Ваш друг Валера работает аналитиком, ему каждый день нужно собирать отчеты с нескольких сайтов, у которых есть открытый API. Чтобы не делать это вручную, Валера просит вас помочь написать ему функцию для получения всех данных разом. Так как на время пандемии Валера уехал работать в деревню, а интернет работает с переменным успехом, целостность получаемых данных может быть нарушена, а еще возможны проблемы с сетью.

Для решения этих проблем Валера предлагает следующее:

- Для обхода проблем с сетью реализовать логику перезапроса данных.
- Во избежание проблем с целостностью данных сравнивать хеш-сумму, полученную из запроса, с той, которую мы вычислим на нашей стороне с помощью асинхронной функции `getHashByData`. Если хеш-суммы не совпадут, попробуйте перезапросить данные.
- Чтобы не перегружать сервера постоянными перезапросами, нужно уметь устанавливать лимит на количество перезапросов к одному API, который можно будет задать как параметр функции.
- Если в итоге информацию получить не удалось, хотя бы из одного API, выдавать ошибку с текстом: "Не удалось получить данные".

Примечания

Заранее определены 2 функции: `fetchData` и `getHashByData`, которые понадобятся вам для решения и находятся в глобальной области видимости.

Функция fetchData

Это асинхронная функция, которую следует использовать для запросов к API:

- принимает один аргумент: url
- возвращает `Promise`, который при успешном запросе вернет вам объект с двумя полями `data` и `hashSum`.

Типы функции на TypeScript

```
// структура возвращаемых данных из запроса
type FetchResult = {
  data: string,
  hashSum: string,
}

function fetchData(url: string): Promise<FetchResult>;
```

Функция getHashByData

Это асинхронная функция, которую следует использовать для получения хеша от данных:

- принимает 2 аргумента: данные и функцию `callback`.
- как только будет вычислен хеш от данных, вызывает функцию `callback` и передает его первым параметром.
- **ничего не возвращает.**

Типы функции на TypeScript

```
/**
 * @param hash - результат вычисления хеш-суммы от переданных данных
 */
type CallbackFunction = (hash: string) => void;

/**
 * @param data - данные от которых следует вычислить хеш.
 * @param callbackFn - функция, в которую следует передать вычисленный хеш.
 */
function getHashByData(data: string, callbackFn: CallbackFunction): void;
```

Шаблон решения для отправки

Решите эту задачу на JavaScript (ES2017) и оформите решение по шаблону:

```
module.exports = async function(urls, retryCount) {
    // code here
    return result;
}
```

Формат ввода

Функции `fetchData` **и** `getHashByData` **находятся в глобальной области видимости.**

Ваша функция должна принимать 2 параметра:

- массив строк url (`string[]`)
- количество попыток получить данные (`number`)

Формат вывода

Полученные данные должны возвращаться в виде массива.

Если не удалось получить данные, выдавайте ошибку с текстом: «Не удалось получить данные».

Пример

```
/**
 * Описание возвращаемых функцией fetchData данных. Хеш всегда корректен.
 *
 * fetchData('metrika.ru/api/analitics') -> Promise<{data: 'Metrika data', hashSum: '#correctHash'}>
 * fetchData('google.ru/api/analitics') -> Promise<{data: 'Google analytics data', hashSum: '#correctHash'}>
 *
 * fetchData('badhost-analitics.com/api/analitics') -> Сервер постоянно не доступен
 *
 * solution - ваше решение.
 */


// Позитивный
solution(['metrika.ru/api/analitics', 'google.ru/api/analitics'], 3)
 .then(data => console.log(data)) // ['Metrika data', 'Google analytics data']
 .catch(error => console.log(error.message))

// Негативный
solution(['metrika.ru/api/analitics', 'badhost-analitics.com/api/analitics'], 3)
 .then(data => console.log(data))
 .catch(error => console.log(error.message)) // "Не удалось получить данные"
```

Вердикт **IL** означает, что вы неоптимально выполняете запросы и не укладываетесь в ограничение по времени.