Домашнее задание к лекции 4.3 «Асинхронные функции и промисы»

Перед началом работы

1.Активируйте строгий режим соответствия.

2.Добавьте в редактор следующий фрагмент кода:

```
function rand(from, to) {
  return Math.floor(Math.random() * (to - from + 1)) + from;
}

function getTagAsync(title, cb) {
  setTimeout(() => {
    const tags = JSON.parse('[{"id":17,"title":"перемещение"},
{"id":32,"title":"гаджеты"}]');
    const tag = tags.find(tag => tag.title === title);
    if (!tag) {
      return cb(`Тег #${title} не найден`);
    }
    cb(null, tag);
  }, rand(2000, 5000));
}

function getTag(title) {
  return new Promise((done, fail) => {
    getTagAsync(title, (err, tag) => {
      if (err) {
        return fail(err);
      }
      done(tag);
    })
  });
}

function getTagItemsAsync(tagId, cb) {
  setTimeout(() => {
    const tagItems = new Map(JSON.parse('[
    [17,[{"title":"Телепорт бытовой VZHIH-
101","price":10000,"discount":7,"available":3},
```

```
    {"title":"Ховерборд Mattel
2016","price":9200,"discount":4,"available":14}]],
    [32,[{"title":"Ховерборд Mattel
2016","price":9200,"discount":4,"available":14},
    {"title":"Меч световой FORCE (синий
луч)","price":57000,"discount":0,"available":1}]]
    ]'));
    if (tagItems.has(tagId)) {
      cb(null, tagItems.get(tagId));
    } else {
      cb(new Error(`Тег #ID${tagId} не найден`));
    }
  }, rand(2000, 5000));
}

function getTagItems(tagId) {
  return new Promise((done, fail) => {
    getTagItemsAsync(tagId, (err, items) => {
      if (err) {
        return fail(err);
      }
      done(items);
    })
  });
}

function getCurrencyRateAsync(code, cb) {
  setTimeout(() => {
    const rates = new Map(JSON.parse('[["AUD", 44.95],["AZN", 33.73],["GBP",
73.42],
    ["AMD", 0.12],["BYN",30.96],["BGN",32.01],["BRL",18.8],["HUF",0.2],
["DKK",8.42],
    ["USD",58.85],["EUR",62.68],["INR",0.88],["KZT",0.18],["CAD",44.74],
["KGS",0.85],
    ["CNY",8.55],["MDL",2.94],["NOK",7.02],["PLN",14.55],["RON",13.92],
["ZZZ",79.91],
    ["SGD",41.36],["TJS",7.43],["TRY",15.97],["TMT",16.84],["UZS",0.02],
["UAH",2.16],
    ["CZK",2.32],["SEK",6.6],["CHF",58.69],["ZAR",4.4],["KRW",0.05],
["JPY",0.52]]'));
```

```
    if (!rates.has(code)) {
      return cb(new Error(`Валюта с кодом ${code} не найдена`));
    }
    cb(null, rates.get(code));
  }, rand(2000, 5000));
}

const tags = [
  { id: 17, title: 'перемещение' },
  { id: 32, title: 'гаджеты' }
];
const badTag = { id: 54, title: 'ошибка' };
const tagTitle = 'гаджеты';
```

Задача № 1. Вывод товаров по тегам

Наш новый маркетолог предложил внедрить тегирование всех товаров в каталоге и реализовать поиск по тегам.

Один из разработчиков уже реализовал базу данных тегов и функцию, возвращающую товары по заданному тегу. Вам нужно использовать эту функцию и реализовать вывод информации о тегах.

Описание функции

Реализуйте функцию `showTagInfo`, которая запрашивает информацию о теге из базы данных с помощью функции `getTagItemsAsync` и выводит информацию о тегах и список товаров в консоль. Принимает один аргумент:

1. `tag` — информация о теге, объект, обязательно должен иметь свойства `id` и `title`.

Интерфейс функции `getTagItemsAsync`

Функция реализована. Делает вид, что она запрашивает список товаров из базы данных. При этом она имеет идентичный интерфейс с оригинальной функцией. Принимает два аргумента:

`id` — идентификатор тега, число;
`callback` — функция обратного вызова, функция, должна принимать ошибку первым аргументом и список товаров, массив, вторым.

Пример использования

```
tags.forEach(showTagInfo);
showTagInfo(badTag);
```

Если все реализовано верно, вы получите такой вывод:

```
Товары по тегу #гаджеты:
 * Ховерборд Mattel 2016: 9200Q
 * Меч световой FORCE (синий луч): 57000Q
```

```
Товары по тегу #перемещение:
 * Телепорт бытовой VZHIH-101: 10000Q
 * Ховерборд Mattel 2016: 9200Q
[Error: Тег #ID54 не найден]
```

Процесс реализации

Создайте функцию `showTagInfo`. Вызовите внутри `showTagInfo` функцию `getTagItemsAsync`, передав ей идентификатор тега и функцию обратного вызова. Если функция обратного вызова будет вызвана с ошибкой, то выведите ошибку с помощью `console.error`. Если функция обратного вызова будет вызвана без ошибки, то отобразите информацию о теге в формате:

```
Товары по тегу #перемещение:
 * Телепорт бытовой VZHIH-101: 10000Q
 * Ховерборд Mattel 2016: 9200Q
```

Проверьте работу вашего кода на примере использования. Также протестируйте его, используя свои примеры.

Задача № 2. Обещания для курса валют

В системе есть функция `getCurrencyRateAsync`, которая асинхронно возвращает курс валюты по её коду. У вас есть заглушка, которая симулирует работу функции `getCurrencyRateAsync`. Так как мы переводим весь асинхронный код на промисы, то требуется реализовать функцию-обертку getCurrencyRate, которая будет возвращать промис, который разрешится курсом запрошенной валюты.
Также реализуйте удобную функцию конвертации из одной валюты в другую — `convertCurrency`. Эта функция должна использовать только что созданную функцию getCurrencyRate и возвращать промис, который разрешится сконвертированной суммой.

Описание функции `getCurrencyRate`

Функция должна вернуть промис, который разрешится курсом запрошенной валюты, числом. Принимает один аргумент:

`code` — код валюты, строка.
Функция должна использовать функцию `getCurrencyRateAsync` для получения курса валюты. Если при получении возникнет ошибка, промис должен быть переведен в статус rejected. Сама функция никак не должна обрабатывать ошибки.

Описание функции `convertCurrency`

Функция должна вернуть промис, который разрешится суммой в производной валюте, число. Принимает три аргумента:

`amount` — сумма в исходной валюте, которую необходимо конвертировать, число;
`fromCode` — код исходной валюты, строка;
`toCode` — код производной валюты, строка.

Интерфейс функции `getCurrencyRateAsync`

Функция уже реализована и делает вид, что получает курс валюты из удаленного веб-сервиса. При этом имеет идентичный интерфейс с оригинальной функцией. Принимает два аргумента:

code — код валюты, курс которой требуется получить, строка;
callback — функция обратного вызова, функция, должна принимать ошибку первым аргументом и курс запрошенной валюты, число, вторым.

Пример использования

```
const amount = 42;
convertCurrency(amount, 'ZZZ', 'USD')
  .then(result => console.log(`${amount}Q = $${result}`))
  .catch(err => console.error(err));

convertCurrency(amount, 'XXX', 'USD')
  .then(result => console.log(`${amount}X = $${result}`))
  .catch(err => console.error(err));
```

Если всё реализовано верно, вы получите такой вывод:

```
[Error: Валюта с кодом XXX не найдена]
42Q = $57
```

Процесс реализации

1. Объявите функцию `getCurrencyRate`.
2. Создайте промис, передав в него функцию обратного вызова [1].

3. В функции обратного вызова [1] вызовите функцию `getCurrencyRateAsync`, передав туда код валюты и функцию обратного вызова. [2]
4. Если в функцию обратного вызова [2] передана ошибка (первый аргумент), то переведите промис в статус `rejected`.
5. Иначе в функции обратного вызова [2] разрешите промис полученным курсом валюты (второй аргумент).

6. Верните созданный промис из функции `getCurrencyRate`.
7. Проверьте работу функции `getCurrencyRate`, используя существующий код `ZZZ` и несуществующий код `XXX`.
8. Создайте функцию `convertCurrency`.
9. Получите промис для обеих валют с помощью функции `getCurrencyRate`.
10. Создайте с помощью `Promise.all` промис [3], который сработает, когда разрешатся оба промиса с курсом валют.
11. Когда разрешится промис [3], произведите конвертацию суммы.

12. Верните из функции `convertCurrency` промис [3].

Проверьте работу вашего кода на примере использования. Также протестируйте его, используя свои примеры.

Задача № 3. Количество товаров по тегу

Реализуйте асинхронную функцию `getTagItemsCountAsync`, принимающую название тега и возвращающую количество товаров, которое по этому тегу присутствует в базе данных.

Описание функции

Принимает два аргумента:

1. `title` — название тега, строка;
2. `callback` — функция обратного вызова, функция, должна ожидать два аргумента: `error` — ошибка, если не `null`, и `count` — количество товаров.

Для получения информации о теге по его названию должна использоваться функция `getTagAsync`, а для получения списка товаров по тегу — `getTagItemsAsync`.

Пример использования

```
getTagItemsCountAsync(tagTitle, (err, count) => {
  if (err) {
    return console.error(err);
  }
  console.log(`По тегу #${tagTitle} найдено товаров ${count} шт.`);
});
```

Если все реализовано верно, вы получите такой вывод:

```
По тегу #гаджеты доступно товаров 2 шт.
```

Процесс реализации

1. Создайте функцию `getTagItemsCountAsync`.
2. В теле функции вызовите `getTagAsync`, передав туда название тега из аргумента `title`.
3. Когда информация о теге будет получена, вызовите `getTagItemsAsync`, передав туда идентификатор тега из свойства `id`.
4. Когда список товаров будет получен, посчитайте количество товаров в списке и вызовите функцию обратного вызова из аргумента `callback`, передав туда `null` и количество товаров.
5. Не забудьте обработать возможные ошибки при вызове `getTagAsync` и `getTagItemsAsync`. Проверьте работу вашего кода на примере использования. Также протестируйте его, используя свои примеры.

Задача № 4. Количество товаров по тегу на промисах

Реализуйте асинхронную функцию `getTagItemsCount`, которая будет принимать название тега и возвращать промис, который разрешится количеством товаров, которое по этому тегу присутствует в базе данных.

Описание функции

Принимает один аргумент:

1. `title` — название тега, строка.

Для получения информации о теге по его названию должна использоваться функция `getTag`, а для получения списка товаров по тегу — `getTagItems`. Обе эти функции возвращают промисы.
Пример использования

```
getTagItemsCount(tagTitle)
  .then(count => console.log(`По тегу #${tagTitle} найдено товаров ${count} шт.`))
  .catch(err => console.error(err));
```

Если все реализовано верно, вы получите такой вывод:

```
По тегу #гаджеты доступно товаров 2 шт.
```

Процесс реализации

1. Создайте функцию `getTagItemsCount`.
2. Получите промис [1], вызвав `getTag` и передав туда название тега из аргумента `title`.
3. Когда промис [1] разрешится, получите промис [2], вызвав `getTagItems` и передав туда идентификатор полученного тега из свойства `id`.
4. Когда промис [2] разрешится, посчитайте количество товаров в полученном списке и верните его в качестве результата.

Проверьте работу вашего кода на примере использования. Также протестируйте его, используя свои примеры.