Цель урока:

Вы узнаете разницу между синхронным и асинхронным кодом. Также вы научитесь, как получать данные с сервера, показывать лоудер для загрузки данных, отображать полученные данные в html.

Содержание урока:

1. Отличие асинхронного и синхронного кода. Что такое Promise и как с ним работать.
2. Работа с функцией fetch. Получение данных и отрисовка их в HTML.
3. Promise.all
4. Promise.race

Дополнительные материалы:

1. Промисы в JavaScript: https://learn.javascript.ru/pr...
2. Fetch в JavaScript: https://learn.javascript.ru/fe...
3. Promise. Что это, как работает (+ пример): https://youtu.be/1idOY3C1gYU
4. Promise.all: https://developer.mozilla.org/...
5. Promise.race: https://developer.mozilla.org/...

Задание #1

Вам дан HTML код.

Ваши коллеги разработчики реализовали систему, благодаря которой можно получать список пользователей по url: https://jsonplaceholder.typicode.com/users.

Вам необходимо получить всех пользователей с помощью **fetch** и добавить данные о имени каждого пользователя внутрь html-элемента с **id** равным **“data-container”**.

Для удобства **необходимо добавить** элемент **span** с текстом **“Загрузка...”** перед загрузкой пользователей, и спрятать этот элемент после загрузки данных о пользователях.

Шаблон для HTML-элемента пользователя выглядит следующим образом:

```
<li><a href="#">Имя пользователя</a></li>
```

Примечание: обязательно не забывайте прописывать блоки **catch**. В них просто выводите ошибку в консоль при помощи **console.error**.

Задание #2

Вы молодец! Если вы дошли до этого задания, то вы умеете получать данные и отображать их в HTML. Сейчас же задача будет посложнее.

Вам необходимо создать функцию **getUsersByIds**, которая будет принимать массив с **id** пользователей. Вам нужно получить всех пользователей, у которых есть данные значения **id**. Используйте некоторый код из предыдущего задания и **Promise.all** для решения поставленной задачи.

Добавьте данные о имени каждого пользователя внутрь html-элемента с **id** равным **“data-container”** . **Также** для удобства **необходимо добавить** элемент **span** с текстом **“Загрузка...”** перед загрузкой пользователей, и спрятать этот элемент после загрузки данных о пользователях.

Шаблон для HTML-элемента пользователя выглядит следующим образом:

```
<li><a href="#">Имя пользователя</a></li>
```

Для тестирования функции getUsersByIds используйте данный код:

```
getUsersByIds([5, 6, 2, 1])
```

Примечание: обязательно не забывайте прописывать блоки **catch**. В них просто выводите ошибку в консоль при помощи **console.error**.

Задание #3

Представьте, что у вас появился новый проект, где есть следующая функциональность: отображение фотографии, которая быстрее всего загрузилась.

Создайте функцию getFastestLoadedPhoto, которая принимает в себя 1 параметр **ids**, являющийся массивом параметров **id** у объекта **photo**. Чтобы получить информацию о фото, **вам необходимо** использовать следующий url https://jsonplaceholder.typico... (1 - это id фотографии). С помощью массива **ids** получите данные о фотографии, которая быстрее всего загрузилась при **fetch** запросе. Для решения поставленной задачи используйте **Promise.race** .

Для удобства **необходимо добавить** элемент **span** с текстом **“Загрузка...”** перед загрузкой фотографии, и спрятать этот элемент после загрузки данных.

Для создания HTML-элемента фотографии используйте данный шаблон:

```
<li class="photo-item">
   <img class="photo-item__image"
src="https://via.placeholder.com/600/92c952">
   <h3 class="photo-item__title">
       accusamus beatae ad facilis cum similique qui sunt
   </h3>
</li>
```

В **src** у **img** должно быть подставлено значение свойства **url** у фотографии, а в тег **h3** - значение свойства **title**. Добавляйте конечную фотографию в элемент с **id** равным **“data-container”**.

Для теста функции getFastestLoadedPhoto используйте данный код:

```
getFastestLoadedPhoto([60, 12, 55])
```

Примечание: обязательно не забывайте прописывать блоки **catch**. В них просто выводите ошибку в консоль при помощи **console.error**.

Задание #4 (*дополнительное*)

Вы вместе с командой разработки создаете социальную сеть, где можно делится постами и оставлять комментарии к посту. Пока проект на ранней стадии и поэтому вам нужно поработать только с отрисовкой нужных элементов.

Вам необходимо создать функцию **renderPost**, которая будет принимать 1 параметр **postId**. С помощью **postId** вам нужно получить пост. Например, с помощью данного url https://jsonplaceholder.typico... вы получите данные о посте с **id** равным 1.

После того, как вы получите данные о посте, **требуется получить** все комментарии для конкретного поста. Чтобы, например, извлечь комментарии для поста с **id** равным 1, вам необходимо использовать следующий url https://jsonplaceholder.typico...

Для отображения информации о посте и комментариев на странице используйте данный шаблон:

```
<div id="post" class="post">
    <h1 class="post__title">Название Поста</h1>
    <p class="post__text">Текст Поста</p>
    <b class="post__comments-text">Комментарии</b>
    <div class="post__comments">
        <div class="post-comment">
            <span class="post-comment__author">
                maxim@gmail.com
            </span>
            <span class="post-comment__text">
                laudantium enim quasi est quidem magnam voluptate
ipsam eos\ntempora quo
                necessitatibus\ndolor quam autem quasi\nreiciendis et
nam sapiente accusantium
            </span>
        </div>
    </div>
</div>
```

Каждый класс отвечает за конкретную информацию:

“post__title” - значение свойства title у поста;

“post__text” - значение свойства body у поста;

“post-comment__author” значение свойства email у комментария;

“post-comment__text” значение свойства body у комментария;

Протестируйте функцию renderPost, вызвав данный код:

```
renderPost(1);
```

Примечание: обязательно не забывайте прописывать блоки **catch**. В них просто выводите ошибку в консоль при помощи **console.error**.