**Lab: Asynchronous Programming**

Problems for exercises and homework for the ["JavaScript Apps" course @ SoftUni.](https://softuni.bg/trainings/3847/js-applications-october-2022)  
The following tasks do not have tests in the Judge system. They are for practice.

**Working with Remote Data**

For the solution of some of the following tasks, you will need to use an up-to-date version of the **local REST service**, provided in the lesson’s resources archive. You can [read the documentation here](https://github.com/softuni-practice-server/softuni-practice-server).

* **XHR (XmlHttpRequest)**

Your task is to **write** a JS function that **loads** a github repository **asynchronously with AJAX**. You should **create** an instance of **XmlHttpRequest** and attach an **onreadystatechange** event to it. (An EventHandler that is called whenever the readyState attribute changes). Obtain the data by making a GET request to the following URL: [**https://api.github.com/users/testnakov/repos**](https://api.github.com/users/testnakov/repos)**.** In your event handler, when the **readyState** attribute reaches a value of **4** (it is ready), replace the text content of a **div** element with **id "res"** with the value of the **responseText** property of the request. **Do not format** the response in any way.

[HYPERLINK "https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open"More on XmlHttpRequest.open()](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open)

**Examples**





* **Github Repos**

Your task is to **write** a JS function that **executes** an **AJAX** request with **Fetch API** and loads all user **github repositories** by a given username (taken from an input field with **id "username"**) into a **list** (each repository as a **list-item**) with **id** "**repos**". Use the properties **full\_name** and **html\_url** of the returned objects to create a link to each repo’s GitHub page. If an **error** occurs (like 404 “Not Found”), **append** to the list a list-item with **text** the current instead. Clear the contents of the list before any new content is appended. See the **highlighted lines** of the skeleton for formatting details of each list item.

**Examples**





* **Github Commits**

Write a JS program that loads all commit messages and their authors from a github repository using a given HTML.

The **loadCommits()** function should get the **username** and **repository** from the HTML textboxes with IDs **"username"** and **"repo"** and make a **GET** request to the **Github API**:  
[**https://api.github.com/repos/<username>/<repository>/commits**](https://api.github.com/repos/%3Cusername%3E/%3Crepository%3E/commits)

Swap **<username>** and **<repository>** with the ones from the HTML:

* In case of **success**, for **each** entry add a **list item** (<**li>**) in the **unordered list** (<**ul>**) with **id "commits"** with text in the following format:

**"<commit.author.name>: <commit.message>"**

* In case of an **error**, add a single **list item** (<**li>**) with text in the following format:  
  **"Error: <error.status> (Not Found)"**

**Screenshots:**





* **Cookbook – Part 1**

The resources for this task are available in the following GitHub repository:

<https://github.com/viktorpts/js-apps-workshop>

You may check-out the repository or download the files via the green button labeled "Code" in the upper-right corner. Use the files located in **lesson-02/base** to begin the task. Before starting, make sure you have the most recent version of the repository. To see the solution, check the files inside **lesson-02/finished**.

Write a JS program that loads all recipies from the provided local server. You are **provided with skeleton** (**HTML & CSS**) for this task, also with **server**, which you will use as localhost. You will be able to load from the server "database" the needed recipies and other details.

**Load all recipies**

When the app is started, you need to **load all the recipies** from the server:



You have to make **"GET" request** to the server on this **URL**: <http://localhost:3030/jsonstore/cookbook/recipes>

**Load selected recipe**

By **clicking on a card** with recipe you need to make a **"GET" request** to the server, and **toggle** **the information** only for the **selected recipe**.

The **URL** for the details is: [http://localhost:3030/jsonstore/cookbook/details/ **HYPERLINK "http://localhost:3030/jsonstore/cookbook/details/:id":id**](http://localhost:3030/jsonstore/cookbook/details/:id)

Where **":id"** is the id of the selected recipe.

