# Lab: AJAX

Problems for exercises for the ["JavaScript For Front-End Course@SoftUni"](https://softuni.bg/trainings/2699/javascript-for-front-end-december-2019).

## Darth Vader Response

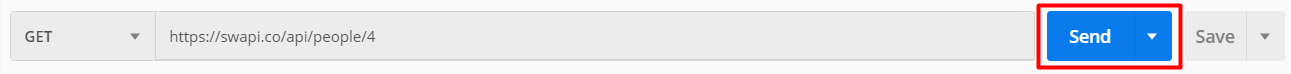
**NOTE: Install** "[Postman](https://www.getpostman.com/)" REST Client to **ease** your tasks.

Your first task is to get detailed information about Darth Vader.

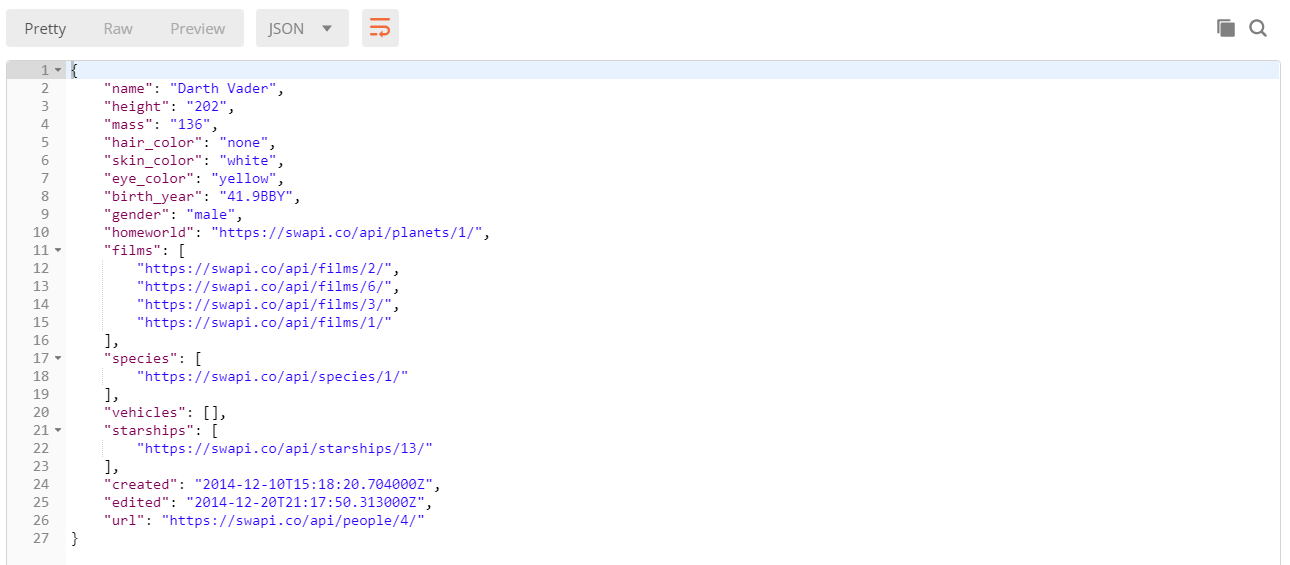
* Send a "**GET**" request to the link given below.
* **Copy** the response in JSON format.

**REQUEST**:

https://swapi.co/api/people/4



**RESPONSE**:

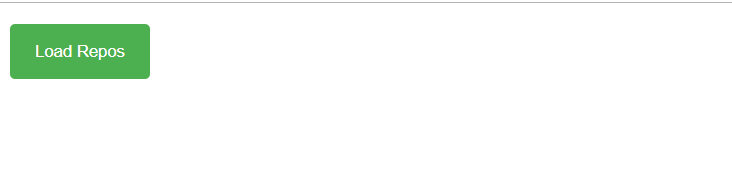


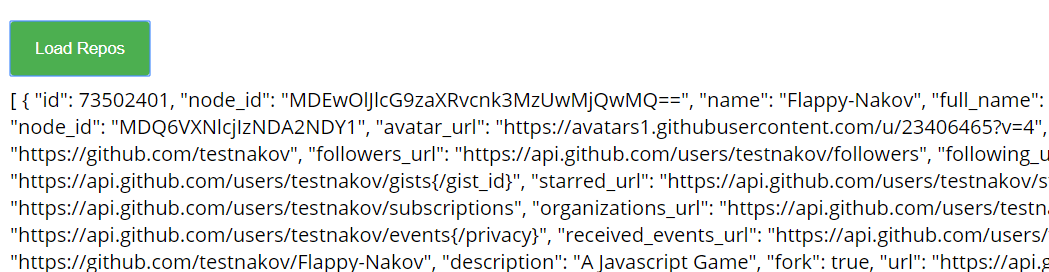
## XHR (XmlHttpRequest)

Your task is to write а 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). In your event handler, when the **readyState** attribute reaches a value of **4** (it is ready), replace the text content of adiv element with **id="res"** with the value of the **responseText** property of the request. **Do NOT format** the response in any way.

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

### Examples





## Bus Stop

Write a program that displays arrival times for all buses by a **given** **bus** **stop** **ID** when a button is **clicked**.

When the button with **id**=**'submit'** is clicked, the name of the bus stop appears and the list bellow gets filled with all the buses that are expected and their time of arrival. Take the **value** of the input field with **id** **'stopId'**. Submit a GET request to https://judgetests.firebaseio.com/businfo/{stopId}.json (replace the highlighted part with the correct value) and parse the response. You will receive a **JSON** object in the format:

stopId: {

name: stopName,

buses: { busId: time, … }

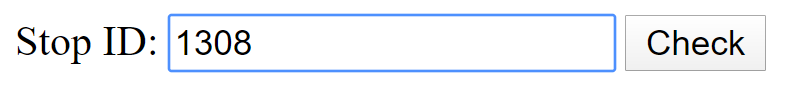
}

Place the name property as text inside the div with **id**=**'stopName'** and each bus as a list item with text:

"Bus {busId} arrives in {time}"

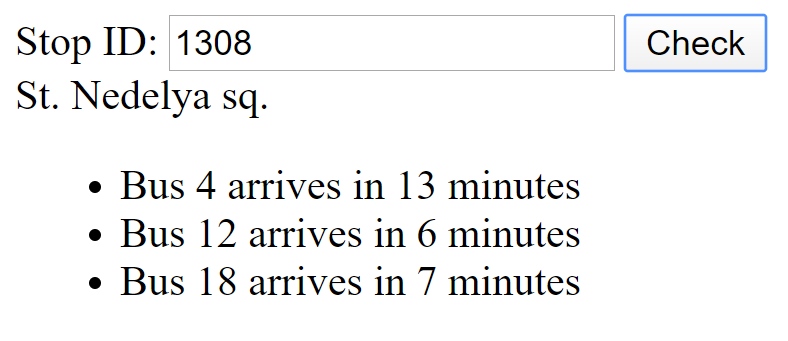
Replace all highlighted parts with the relevant value from the response. If the request is **NOT** successful, or the information is **NOT** in the expected format, display **"**Error**"** as stopName and nothing in the list. The list should be cleared before every request is sent.

### Examples



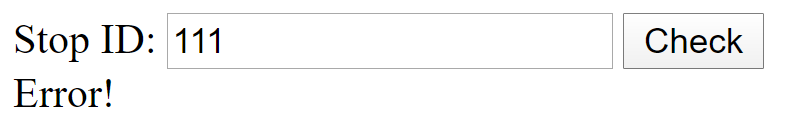
|  |
| --- |
| HTML |
| <div id="stopInfo" style="width:20em">      <div>          <label for="stopId">Stop ID: </label>          <input id="stopId" type="text">          <input id="submit" type="button" value="Check" onclick="getInfo()"></div>      <div id="result">          <div id="stopName"></div>          <ul id="buses"></ul>      </div>  </div> |

When the button is **clicked**, the results are displayed in the corresponding elements:



|  |
| --- |
| HTML |
| <div id="stopInfo" style="width:20em">      <div>          <label for="stopId">Stop ID: </label>          <input id="stopId" type="text">          <input id="submit" type="button" value="Check" onclick="getInfo()"></div>      <div id="result">          <div id="stopName">St. Nedelya sq.</div>          <ul id="buses">  <li>Bus 4 arrives in 13 minutes</li>  <li>Bus 12 arrives in 6 minutes</li>  <li>Bus 18 arrives in 7 minutes</li></ul>      </div>  </div> |

If an error occurs, the stop name changes to **Error**:



|  |
| --- |
| HTML |
| <div id="stopInfo" style="width:20em">      <div>          <label for="stopId">Stop ID: </label>          <input id="stopId" type="text">          <input id="submit" type="button" value="Check" onclick="getInfo()"></div>      <div id="result">          <div id="stopName">Error!</div>          <ul id="buses"></ul>      </div>  </div> |

### Hints

The webhost will respond with valid data to IDs **1287**, **1308**, **1327** and **2334**.

## \* Phonebook

Write a program that can **load**, **create** and **delete** entries from a **Phonebook**.

When the [Load] button is clicked, a GET request should be made to the server to get all phonebook entries. Each received entry should be in a li inside the ul with id="phonebook" in the following format with text "<person>: <phone> " and a [Delete] button attached. Pressing the [Delete] button should send a DELETE request to the server and delete the entry. The received response will be an object in the following format:  
{<key>:{person:<person>, phone:<phone>}, <key2>:{person:<person2>, phone:<phone2>,…} where <key> is an unique key given by the server and <person> and <phone> are the actual values.

When the [Create] button is clicked, a new POST request should be made to the server with the information from the Person and Phone textboxes, the Person and Phone textboxes should be cleared and the Phonebook should be automatically reloaded (like if the [Load] button was pressed).

The data sent on a POST request should be a valid JSON object, containing properties person and phone. Example format:   
{

"person": "<person>",

"phone": "<phone>"  
}

The url to which your program should make requests is:

'https://phonebook-nakov.firebaseio.com/phonebook'

GET and POST requests should go to

https://phonebook-nakov.firebaseio.com/phonebook.json,

while DELETE requests should go to

https://phonebook-nakov.firebaseio.com/phonebook/<key>.json,

where <key> is the unique key of the entry (you can find out the key from the key property in the GET request)

**You may create your own app** in **Firebase**.

### Screenshots:

