# Exercises: Async Programming and Promises

Problems for exercises and homework for the [“JavaScript Applications” course @ SoftUni](https://softuni.bg/courses/javascript-applications). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/360/>.

## Github Commits

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

### HTML Template

You are given the following HTML:

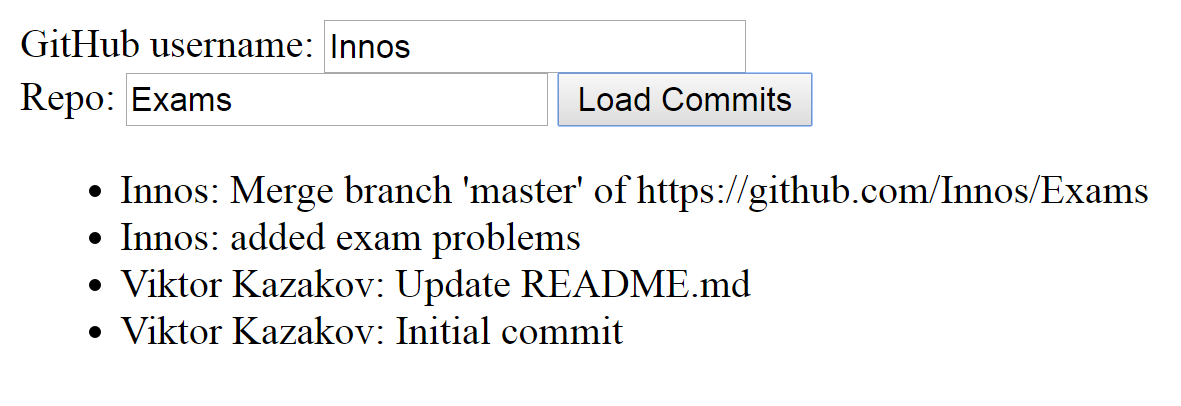
|  |
| --- |
| commits.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Github Commits</**title**>  <**script src="https://code.jquery.com/jquery-3.1.1.min.js"**></**script**> </**head**> <**body**> GitHub username: <**input type="text" id="username" value="nakov"** /> <**br**> Repo: <**input type="text" id="repo" value="nakov.io.cin"** /> <**button onclick="***loadCommits*()**"**>Load Commits</**button**> <**ul id="commits"**></**ul**> <**script**>  **function** *loadCommits*() {  *// AJAX call …* } </**script**> </**body**> </**html**> |

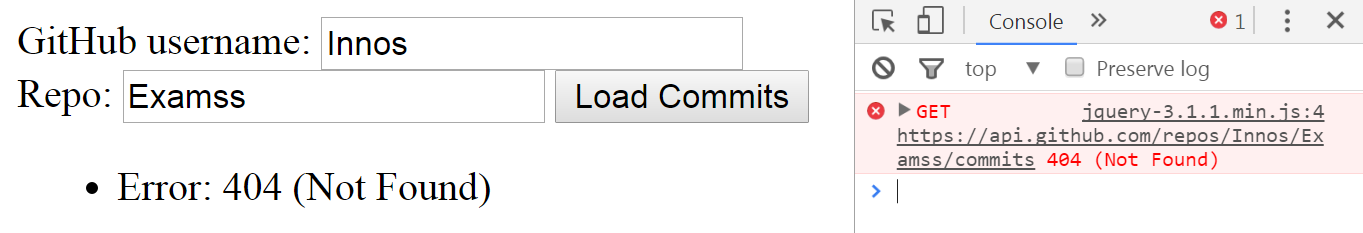
The loadCommits function should get the username and repository from the HTML textboxes with ids "username" and "repo" and make a GET request to the Github API:  
**"https://api.github.com/repos/<username>/<repository>/commits"**

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 format:  
  "<commit.author.name>: <commit.message>"
* In case of error and a single list item (li) with text in the format:  
  "Error: <error.status> (<error.statusText>)"

### Screenshots:





Submit in the Judge only the loadCommits function.

## Blog

Write a JS program for reading blog content. It needs to make requests to the server and display all blog posts and their comments. Use the following HTML to test your solution:

|  |
| --- |
| blog.html |
| <!DOCTYPE **html**>  <**html**>  <**head**>  <**meta charset="UTF-8"**>  <**title**>Blog</**title**>  <**script src="https://code.jquery.com/jquery-3.1.1.min.js"**></**script**>  </**head**>  <**body**>  <**h1**>All Posts</**h1**>  <**button id="btnLoadPosts"**>Load Posts</**button**>  <**select id="posts"**></**select**>  <**button id="btnViewPost"**>View</**button**>  <**h1 id="post-title"**>Post Details</**h1**>  <**ul id="post-body"**></**ul**>  <**h2**>Comments</**h2**>  <**ul id="post-comments"**></**ul**>  <**script src="solution.js"**></**script**>  <**script**>  *attachEvents*();  </**script**>  </**body**>  </**html**> |

Submit only the attachEvents() function that attaches events to the buttons and contains all program logic. You will need to create a **Kinvey** database to test your code (instructions bellow).

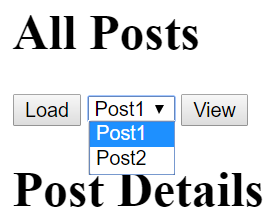
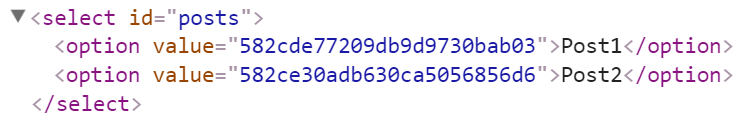
The button with ID "**btnLoadPosts**" should make a GET request to "**/posts**". The response from the server will be an **array** of objects with format:

{ \_id: "postId",

title: "postTitle",

body: "postContent" }

Create an <option> for each post using its \_id as value and title as text inside the node with ID "**posts**".

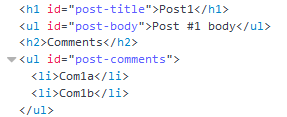
When the button with ID "**btnViewPost**" is clicked should make a GET request to "**/posts/{postId}**" to obtain just the selected post (from the dropdown menu with ID "**posts**") and another request to "**/comments/?query={"post\_id":"{postId}"}**" to obtain all comments (replace highlighted parts with the relevant value). The first request will return a single object as described above, while the second will return an array of objects with format:

{ \_id: "commentId",

text: "commentCOntent",

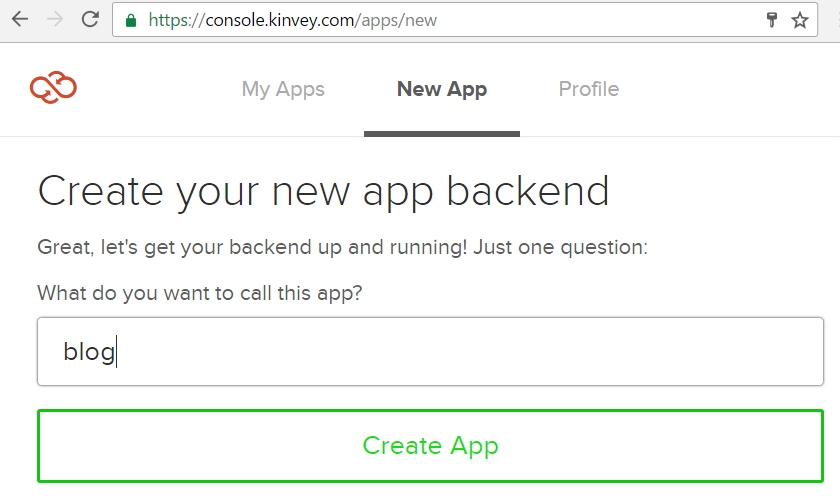
post\_id: "postId"}

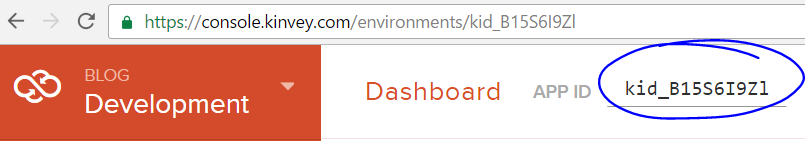
Display the post title inside "#post-title" and the post content inside "#post-body". Display each comment as a <li> inside "#post-comments" and don’t forget to clear its contents beforehand.

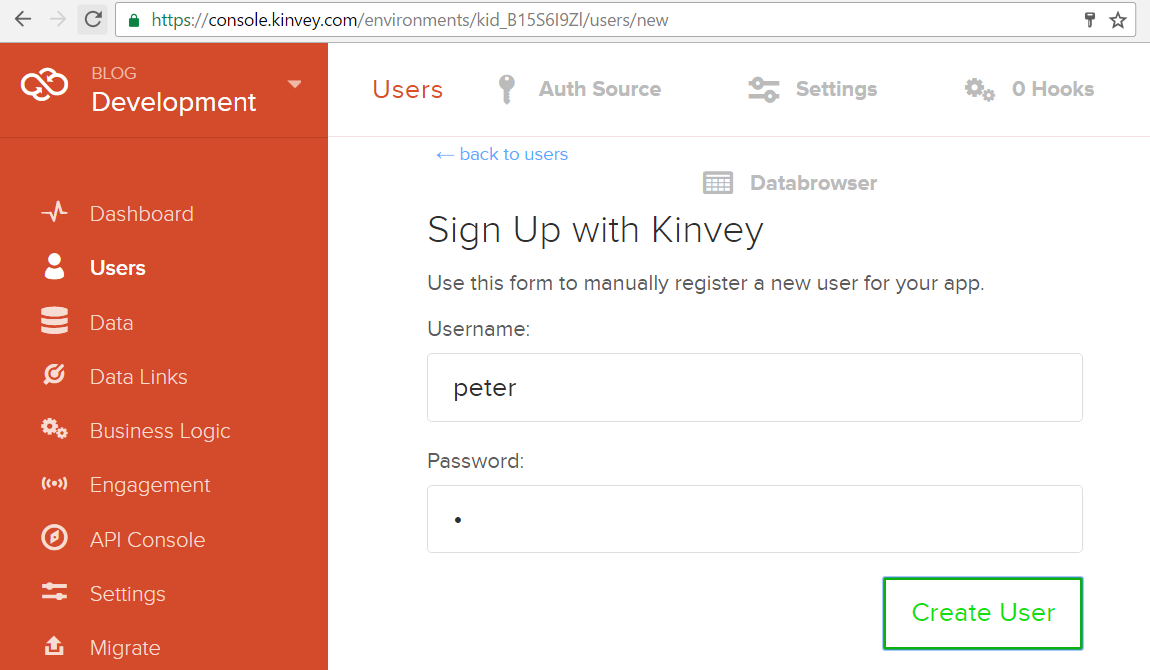
### Hints

To create a Kinvey database with the required content, you need to register an account and create a new backend app.

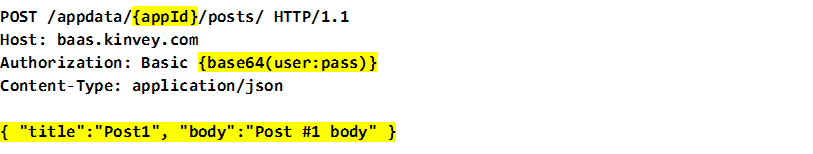




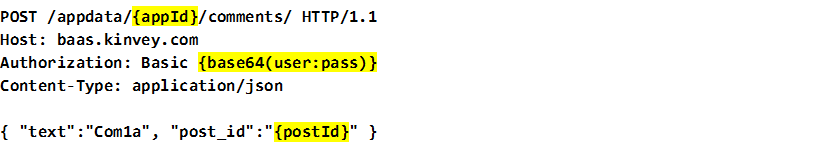
Create a user and a password. You will need these, along with your **app ID** to authenticate with the server from your JS program.



Use the following POST request trough Postman to create blog posts:



Note the **empty line** between the header and the content, the request won’t work without it. Replace the highlighted parts with the relevant info. The authorization string is your **username** and **password** appended together with a **colon** between them as string, hashed with the btoa() function (built into the browser). The resulting post will have an \_id automatically assigned by Kinvey. You will then use this ID when creating comments for each blog post.



After the posts and comments are created, your database should look like this:

