Exercises: Async Programming and Promises

Problems for exercises and homework for the "JavaScript Applications" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/360/.

1. 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

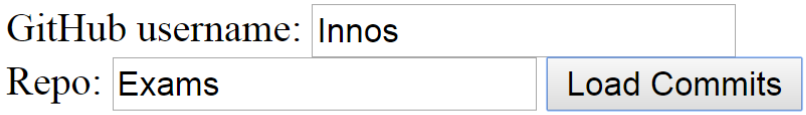
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Github Commits</title>
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
</head>
<body>
GitHub username:
<input type="text" id="username" value="nakov" /> <br>
Repo: <input type="text" id="repo" value="nakov.io.cin" />
<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"

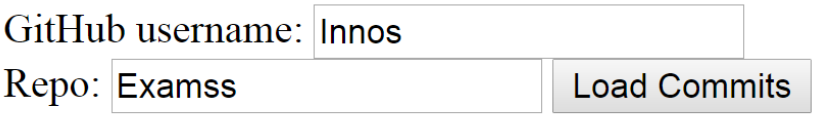
Swap **<username>** and **<repository>** with the ones from the HTML:

- 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>)"

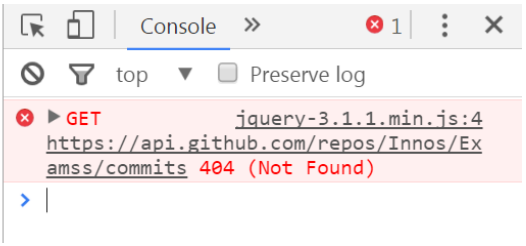
Screenshots:



- Innos: Merge branch 'master' of https://github.com/Innos/Exams
- Innos: added exam problems
- Viktor Kazakov: Update README.md
- Viktor Kazakov: Initial commit



- Error: 404 (Not Found)



Submit in the Judge only the **`loadCommits`** function.

2. 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

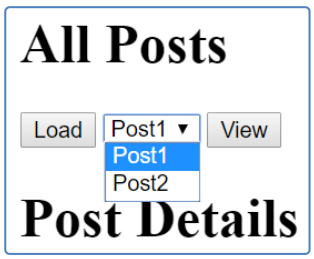
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Blog</title>
  <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
</head>
<body>
<h1>All Posts</h1>
<button id="btnLoadPosts">Load Posts</button>
<select id="posts"></select>
<button id="btnViewPost">View</button>
<h1 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**".




```
<select id="posts">
  <option value="582cde77209db9d9730bab03">Post1</option>
  <option value="582ce30adb630ca5056856d6">Post2</option>
</select>
```

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:

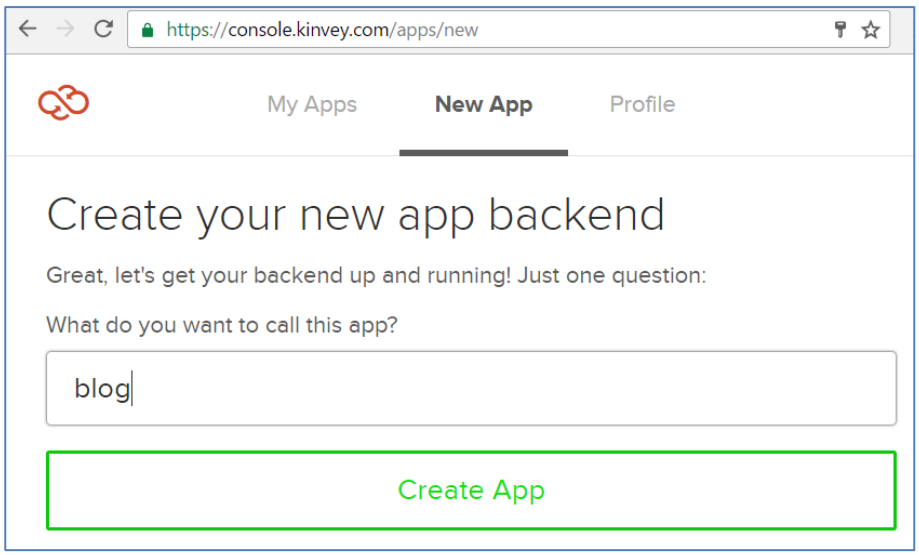
```
{ _id: "commentId",
  text: "commentCOntent",
  post_id: "postId"}
```

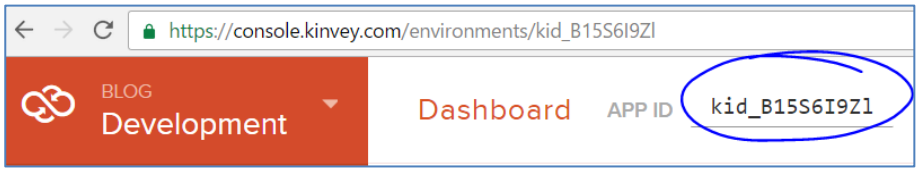
Display the post title inside "#post-title" and the post content inside "#post-body". Display each comment as a <li> inside "#post-comments" and don't forget to clear its contents beforehand.


```
<h1 id="post-title">Post1</h1>
<ul id="post-body">Post #1 body</ul>
<h2>Comments</h2>
<ul id="post-comments">
  <li>Com1a</li>
  <li>Com1b</li>
</ul>
```

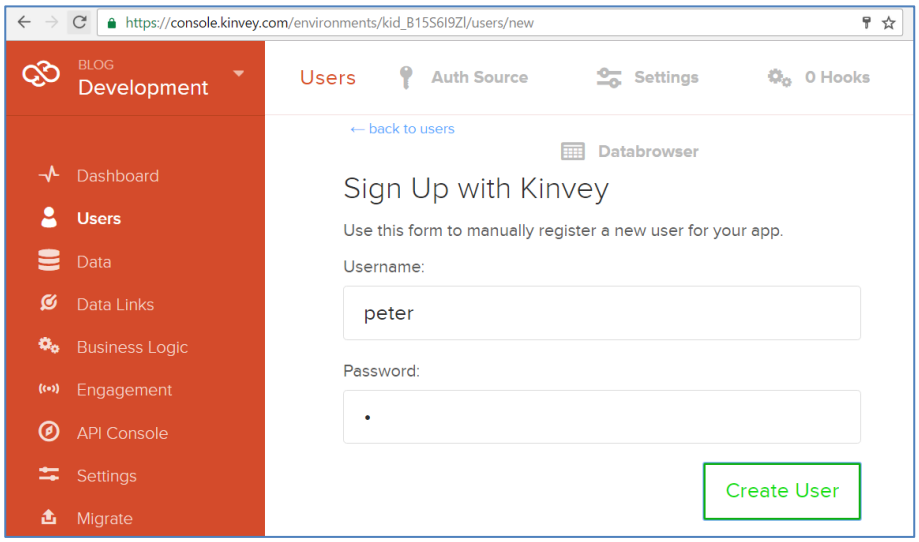
Hints

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:

```
POST /appdata/{appId}/posts/ HTTP/1.1
Host: baas.kinvey.com
Authorization: Basic {base64(user:pass)}
Content-Type: application/json

{ "title":"Post1", "body":"Post #1 body" }
```

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.

```
POST /appdata/{appId}/comments/ HTTP/1.1
Host: baas.kinvey.com
Authorization: Basic {base64(user:pass)}
Content-Type: application/json

{ "text":"Com1a", "post_id":"{postId}" }
```

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

_id	_acl	_kmd	title	body
582ce30adb630ca5056856d6	{"creator":"582cde6b..	{"lmt":"2016-11-16T22:57:09.7..	"Post2"	"Post #2 body"
582cde77209db9d9730bab03	{"creator":"582cde6b..	{"lmt":"2016-11-16T22:57:13.9..	"Post1"	"Post #1 body"

_id	_acl	_kmd	text	post_id
582ce37b209db9d9730be951	{"creator":"582cde6b..	{"lmt":"2016...	"Com2c"	"582ce30adb630ca5056856d6"
582ce375db630ca505685855	{"creator":"582cde6b..	{"lmt":"2016...	"Com2b"	"582ce30adb630ca5056856d6"
582ce36d209db9d9730be92a	{"creator":"582cde6b..	{"lmt":"2016...	"Com2a"	"582ce30adb630ca5056856d6"
582ce2b76ce7ea9e15ca61e3	{"creator":"582cde6b..	{"lmt":"2016...	"Com1b"	"582cde77209db9d9730bab03"
582ce154d474d37548133d04	{"creator":"582cde6b..	{"lmt":"2016...	"Com1a"	"582cde77209db9d9730bab03"