

## D. Lacoviello

### Candidate

E-mail:  
hr@epages.com

### Session

ID: BHBV2F-8A8  
Time limit: 120 min.  
Report recipients: hr@epages.com  
Accessed from: 84.59.97.177,  
84.59.97.177  
Invited by: hr@epages.com

### Status: completed

Invitation: [sent](#)  
Created on: 2023-06-20 10:39 UTC  
Started on: 2023-06-21 07:38 UTC  
Finished on: 2023-06-21 09:23 UTC

### Notes:

N/A

### Similarity Check

Status: not found

No similar solutions have been detected.

Test score

# 46%

### Tasks in test

- 1 | JsBrands  
Submitted in: JavaScript
- 2 | CountryCapital  
Submitted in: JavaScript

### Score

36%

55%

### Canvas Details

The canvas board was not used during that session.

### Tasks Details

Hard

**1. JsBrands**

Make a list of top brand names, using Promises.

**Task Score**

36

**Correctness**

36

**Performance**

Not assessed

**Task description**

You would like to make a list of N top brand names for a specific user based on his/her preferences. The method of compiling the list is as follows:

1. Every user has a list of brands he/she likes most. If there are at least N liked brands, take the first N brand names from the list.
2. There are lists of brands which are most popular among the users of each gender. If the user's list does not have enough liked brands, then the rest of the result list should be filled up with top brands from list for the user's gender.
3. If the user's individual preference list and the list for that user's gender combined do not provide enough brands, you should finish with an error.

Write a function:

```
function solution(U, N);
```

that, given user U and the number of brand names N, returns a Promise that should either be:

- resolved with an array of exactly N top brand names for the given user, in the following format: ["Some Brand Name", "Other Brand Name", ...]; or
- rejected with a CustomError with the message "Not enough data" (if there are fewer than N brand names to be listed, or both Promises `getLikedBrands(id)` and `getTopBrandsForGender(gender)` are rejected).

**Technical details:**

Accessing data:

- The user parameter is an object of the following structure: { id: 123132, gender: "FEMALE" }, where id is an integer and gender is a string containing either "FEMALE" or "MALE".
- The brand names liked by a specific user can be accessed by calling the function `getLikedBrands(id)`.
- The list of brands for a gender can be obtained by calling the function `getTopBrandsForGender(gender)`.
- The functions return Promises, that will be rejected or resolved with data in the following format:

```
[
  { id: 123, name: "Some Brand Name" },
  { id: 456, name: "Other Brand Name" },
  ...
]
```

The result:

- The order of the brand names in the result list should be the same as the order in the lists produced by the functions, with brand names returned by `getLikedBrands(id)` listed first.
- Brand names returned by both functions `getLikedBrands(id)` and `getTopBrandsForGender(gender)` in combination, should appear in the result list only once.

**Hints:**

- Please note that the solution should use the implementations of CustomError, getLikedBrands and getTopBrandsForGender that were defined in the global scope (as described in the initial solution comment: global CustomError, getLikedBrands, getTopBrandsForGender).

**Examples:**

Given user U, assume that `getLikedBrands(U.id)` returns [{id: 1, name: "Logestyx"}, {id: 10, name: "Gladlear"}] and `getTopBrandsForGender(U.gender)` returns [{id: 6, name: "Burylaze Slapgalt"}, {id: 1, name: "Logestyx"}, {id: 7, name: "Izarpure"}].

1. For N=1, your function should return a Promise which resolves with an array ["Logestyx"].
2. For N=3, Promise should be resolved with an array ["Logestyx", "Gladlear", "Burylaze Slaggalt"].
3. For N=4, Promise should be resolved with an array ["Logestyx", "Gladlear", "Burylaze Slaggalt", "Izarpure"].
4. For N=5, Promise should be rejected with a CustomError.

The solution should be optimized for client-side (browser-based) performance, rather than back-end load. The expectation is that the developer will call both functions in parallel. Note that the example test cases do not check for this, but the evaluation tests will check it.

Copyright 2009–2023 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

## Solution

[See Live Version](#)

Programming language used: JavaScript

Total time used: 45 minutes



Effective time used: 15 minutes



Notes: *not defined yet*

## Source code

Code: 08:23:31 UTC, js, final, score: 36

```
1'use strict';
2
3/* global CustomError, getLikedBrands, getTopBrandsForGender */
4
5function solution(U, N) {
6  return new Promise((resolve, reject) => {
7    Promise.all([getLikedBrands(U.id), getTopBrandsForGender(U.gender)])
8      .then(([likedBrands, genderBrands]) => {
9        const likedBrandsCount = likedBrands.length;
10       const result = likedBrands.map((brand) => brand.name);
11
12       // Add top brands from the gender list if needed
13       for (let i = 0; i < genderBrands.length; i++) {
14         const brandName = genderBrands[i].name;
15         if (!result.includes(brandName)) {
16           result.push(brandName);
17         }
18       }
19
20       // Check if enough brands were found
21       if (result.length >= N) {
22         resolve(result.slice(0, N));
23       } else {
24         reject(new CustomError('Not enough data'));
25       }
26     })
27   .catch(reject);
28 });
29}
```

## Analysis summary

The following issues have been detected: wrong answers.

## Analysis

Example tests		
example	first example test	✓ OK
example2	second example test	✓ OK
example3	third example test	✓ OK
example4	fourth example test	✓ OK
Correctness tests		
simple_test	liked brands succeeds with enough data	✗ <b>WRONG ANSWER</b> Expected the promise state to be resolved but it was pending
liked_success_gender_succeed	liked succeeds first, liked gender succeeds later	✓ OK
liked_success_gender_fails	liked succeeds first, liked gender fails later	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:607:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
liked_fails_gender_succeed	liked brands fails first, gender brands succeeds with enough brands later	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:600:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
liked_fails_gender_fails	both promises fail, liked first	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:600:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
gender_success_liked_succeed	gender succeeds first, liked brands succeeds later	✓ OK
gender_succeed_liked_fails	gender brands succeeds with enough brands, then liked brands fails	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:600:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
gender_fails_liked_succeed	gender fails first, liked brands succeeds with enough brands later	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:607:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
gender_fails_like_fails	both promises fail, gender first	✗ <b>WRONG ANSWER</b> Promise rejected: Error: Couldn't get brands at /tmp/exec_user_xfl3uyxb/exec.js:607:11 at /tmp/exec_user_xfl3uyxb/exec.js:523:42
brands_duplicates	brands on the results lists are duplicating	✓ OK
big_data	test with lot of not related promises	✓ OK

Medium

**2. CountryCapital**

Implement a country and capital matching game in React.

**Task Score**

55

**Correctness**

55

**Performance**

Not assessed

## Task description

Implement a game to match countries with their capitals in React.

## Requirements

Implement a React component that renders a simple game.

In the game, the player needs to match a country to its capital by clicking on appropriate buttons.

1. Your component should receive a `data` property in the following format (an object with the correct answer, where the keys are the names of the countries and the value of each key is the capital of the country):

```
<CountryCapitalGame data={{ Germany: "Berlin", Azerbaijan: "Baku" }} />
```

2. A button should be displayed for each country and each capital. So, the example above would return four buttons: Germany, Berlin, Azerbaijan and Baku.
3. The buttons should be displayed in a random order.
4. Clicking a button should set its background color to blue (`#0000ff`).
5. Clicking another button should:
  - remove both buttons if a correct country and capital pair has been selected;
  - set the background color of both buttons to red (`#ff0000`) if a wrong pair has been selected.
6. Assuming the previously selected pair was wrong, clicking another button should restore the default background color of the wrong pair and set the background color of the clicked button to blue (as in point 4).
7. When there are no buttons left, display a message: Congratulations.
8. Export your component as the **default** export.

## Assumptions

- Assume the provided data is correct (all the `data` object keys and values are strings).
- The look of the component won't be evaluated; only its specified functionalities will be tested.

## Available tools/packages

Use the browser console for debugging.

You are expected to use:

- React 17.0.1

## Examples

## Example in working app

Correct answers in the example are:

- Poland — Warsaw
- Germany — Berlin
- Azerbaijan — Baku
- Papua New Guinea — Port Moresby

Warszawa Port Moresby Poland  
Azerbaijan Papua New Guinea Berlin  
Germany Baku

Solution

[See Live Version](#)

Programming language used: JavaScript

Total time used: 101 minutes

?

Effective time used: 90 minutes

?

Notes: *not defined yet*

## Source code

Code: 09:23:06 UTC, js, final, score: 55

```
1import React, { useState, useEffect } from 'react';
2
3const CountryCapitalGame = ({ data }) => {
4  const [buttons, setButtons] = useState([]);
5  const [selectedPair, setSelectedPair] = useState([]);
6  const [message, setMessage] = useState('');
7
8  useEffect(() => {
9    const buttonList = [];
10   for (const country in data) {
11     buttonList.push({ type: 'country', name: country });
12     buttonList.push({ type: 'capital', name: data[country] });
13   }
14   // Randomize the button order
15   const shuffledButtons = buttonList.sort(() => Math.random() - 0.5);
16   setButtons(shuffledButtons);
17 }, [data]);
18
19 const handleClick = (button) => {
20   if (button.selected) {
21     // Same button clicked twice
22     return;
23   }
24
25   const [country, capital] = selectedPair;
26   if (selectedPair.length === 0) {
27     setSelectedPair([button.name, button.type]);
28     button.selected = true;
29   } else {
30     if (
31       (button.type === 'country' && selectedPair[1] === 'capital' && button.name === capital) ||
32       (button.type === 'capital' && selectedPair[1] === 'country' && button.name === country)
33     ) {
34       setSelectedPair([]);
35       setMessage('');
36       setButtons(buttons.filter((btn) => btn.name !== country && btn.name !== capital));
37     } else {
38       buttons.forEach((btn) => {
39         if (btn.selected) {
40           btn.selected = false;
41           btn.wrong = true;
42         }
43       });
44       button.selected = true;
45       setSelectedPair([button.name, button.type]);
46     }
47   }
48
49   setButtons([...buttons]);
50 };
51
52 useEffect(() => {
```

```

53   if (buttons.length === 0 && selectedPair.length === 0) {
54     setMessage('Congratulations!');
55   } else {
56     setMessage('');
57   }
58 }, [buttons, selectedPair]);
59
60 return (
61   <div>
62     {buttons.map((button, index) => (
63       <button
64         key={index}
65         onClick={() => handleClick(button)}
66         style={{
67           backgroundColor: button.selected ? '#0000ff' : button.wrong ? '#ff0000' : undefined,
68         }}
69       >
70         {button.name}
71       </button>
72     ))}
73     <p>{message}</p>
74   </div>
75 );
76 };
77
78 export default CountryCapitalGame;

```

## Analysis summary

The following issues have been detected: wrong answers.

## Analysis

Correctness tests	
CountryCapitalGame should render all buttons	✓ OK
CountryCapitalGame should handle data with spaces	✓ OK
CountryCapitalGame should render buttons in random order	✓ OK
CountryCapitalGame should set button background to blue (#0000ff) when clicked	✓ OK
CountryCapitalGame should keep blue (#0000ff) button background, when the same button is clicked twice	✓ OK
CountryCapitalGame should remove buttons when correct answer selected (country first)	✗ WRONG ANSWER
CountryCapitalGame should remove buttons when correct answer selected (capital first)	✗ WRONG ANSWER
CountryCapitalGame should set buttons backgrounds to red (#ff0000) when incorrect answer selected	✗ WRONG ANSWER
CountryCapitalGame should restore buttons default background when answer selected	✗ WRONG ANSWER
CountryCapitalGame should change color to blue when wrong answer selected again	✓ OK
CountryCapitalGame should display "Congratulations" when no answers left	✗ WRONG ANSWER