## 01.Match Full Name

Write a JavaScript function to **match full names** from a list of names and **print** them on the console.

### Writing the Regular Expression

First, write a regular expression to match a valid full name, according to these conditions:

* A valid full name has the following characteristics:
  + It consists of **two words**.
  + Each word **starts** with a **capital letter**.
  + After the first letter, it **only contains lowercase letters afterward**.
  + **Each** of the **two words** should be **at least two letters long**.
  + The **two words** are **separated** by a **single space**.

To help you out, we've outlined several steps:

1. Use an online regex tester like <https://regex101.com/>
2. Check out how to use **character sets** (denoted with square brackets - "[]")
3. Specify that you want **two words** with a space between them (the **space character** ' ', and **not** any whitespace symbol)
4. For each word, specify that it should begin with an uppercase letter using a **character set**. The desired characters are in a range – **from** 'A' **to** 'Z'.
5. For each word, specify that what follows the first letter are only **lowercase letters**, one or more – use another character set and the correct **quantifier**.
6. To prevent capturing of letters across new lines, put "\b" at the beginning and the end of your regex. This will ensure that what precedes and what follows the match is a word boundary (like a new line).

To check your RegEx, use these values for reference (paste all of them in the **Test String** field):

|  |  |
| --- | --- |
| **Match ALL of these** | **Match NONE of these** |
| Ivan Ivanov | ivan ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Ivan IvAnov, Ivan Ivanov |

By the end, the matches should look something like this:

Graphical user interface, application

Description automatically generated

After you’ve constructed your regular expression, it’s time to write the solution in JavaScript.

### Implementing the Solution in JavaScript

Create a new JavaScript file and copy your **regular expression** into a variable:



Note: You should put **"/"** before and after the pattern so that it is interpreted as a RegEx pattern. Also, place the '**g'** (global) flag after it, so that you get all the matches in the text.

Now, it’s time to **read the input**, **extract the matches** from it and push them into an array. For this we can use exec():

Graphical user interface, text

Description automatically generated

The execmethod matches the string and the pattern keeps the first index after the match. This way the next time exec runs it starts looking after the last match. If there are no more matches, it will return **null**.

We are declaring a variable in the while loop's condition because we need to check every time if there are any more matches.

Now we have an array (**validNames**), which holds all of the valid names in the input. All that is left is to **join** it by **space** and **print** it (do this by using **join()**):



### Examples

|  |
| --- |
| **Input** |
| "Ivan Ivanov, Ivan ivanov, ivan Ivanov, IVan Ivanov, Test Testov, Ivan Ivanov" |
| **Output** |
| Ivan Ivanov Test Testov |