Junior Front End Developer   
Test Assignment

© 2017 CUBE Team  
All rights reserved.

**THIS DOCUMENT IS CONFIDENTIAL. DO NOT SHARE WITH 3RD PARTIES. IF YOU ARE NOT THE INTENDED RECIPIENT OF THIS DOCUMENT, PLEASE DESTROY ALL COPIES AND CONTACT CUBE TEAM.**

The following assignment is designed to let us determine your skill level, and learn more about the way you reason about typical front end problems. Although the project is pretty straightforward, there is no one right answer. Feel free to use tools and techniques you feel are the most adequate for the job.

**This assignment should be completed within 5 days. The exact date by which you are to complete it is noted in the email.**

This document is part of a bundle. You should also have received 3 mockup files and a SVG icon along with it. If you are missing any files, please contact us immediately.

Before you begin, read through, and make sure all requirements are clear to you. If you have questions, or need an extension to the 5-day deadline, ask before you start.

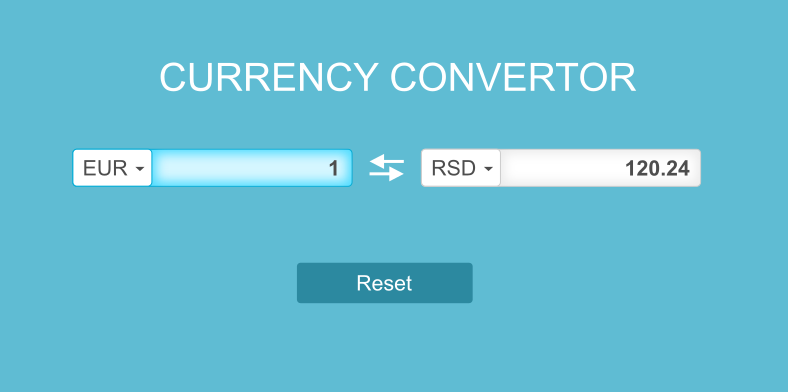
# Currency convertor

Your task is to build a simple currency convertor.

The user interface consists of a pair of **currency widgets**, each having:

1. **Select list** with a choice of currencies
2. **Input box** for entering the numeric value

The user interface also contains a **reset button**. The swap button between the two widgets is optional.



## Currency selection

* The widgets should initialize with EUR on the left, and RSD on the right.
* The selection supports RSD, USD, and EUR only.
* Changing the currency in one widget will automatically recalculate the result in the other widget.
* The choice of currencies in a widget *excludes* the currency currently selected in the other (for example, if left widget is set to RSD, the right widget only has EUR and USD options).

## Input behavior

* Widget should initialize with 1 on the left, and calculated value on the right.
* When a number is modified in one of the widgets, converted value is automatically shown in the other.
* Pasting a number in the widget’s input box should also cause the other to recalculate.
* If the user clears the value of a widget, the other widget should show the unit value (as if the widget contains 1).
* If the user enters a non-numeric value (or an invalid number), an error message “Please enter a number” should be shown next to the widget, and the other widget should recalculate to a unit value.
* Calculated numeric values should be rounded to 2 decimal places.
* The input user is editing should be highlighted.

## Reset button

* When the reset button is used, the interface is reset to its initial state.

## Appearance

* Mockups are provided as part of the assignment details.
* The appearance of your interface should be as close as possible but it does not need to be pixel-perfect.

## Source data

* Source data should be obtained from this server:  
   **https://api.kursna-lista.info/b7b80a59415046c33449b6a2a96bd4d8/kursna\_lista**
* Data on the server will update once every day at 6:00 UTC (8:00 CET).
* API documentation is not provided. You are expected to figure it out by yourself.
* The server is not on the same domain as the local deployment, but we will expect the program to work without throwing exceptions about cross-domain requests. For this purpose, you may use proxies such as the one built into Webpack dev serer, or node-http-proxy, and similar. You may, as last resort, download the data and save it as a JSON file, and serve it as a static asset. You may not embed the data in your application or HTML code. It must be fetched using an XHR request.

# Technical requirements

* Please track the code in git and provide a link to a public BitBucket or GitHub repository.
* Please track the time it took you to complete the assignment and report it when sending us the repository URL.
* Please provide sufficient documentation (in English) and/or automation so that we can deploy your code on our machines. **Keep in mind that we will not review code that cannot be deployed on our machines.**
* Support browsers IE10 and above, and latest versions of Google Chrome, and Firefox.

# Bonus requirements

These items are not critical, but we hope they will be fun.

* The interface should recall its last state (currency selection, values of the widgets) when the page is reopened or refreshed.
* A swap button between the two widgets should swap the states of the two widgets (currency selection and numeric values).
* The interface should be responsive, and work on both mobile and desktop browsers.
* Try doing this without using a view library or a framework.
* Write automated tests.

# Mockups

Mockups are provided as part of the assignment bundle.