# Exercises: ReactJS

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/366/>.

## List Towns

Create a web application in which you have one input field and a button. In the input field should be **elements** **separated** by comma and whitespace (“**,** ”). Your task is to display every **element**, separated with **a line**. Use the following HTML for testing your solution:

|  |
| --- |
| listtown.html |
| <!DOCTYPE **html**>  <**html lang="en"**>  <**head**>  <**meta charset="UTF-8"**>  <**title**>List Town</**title**>  <**script src="https://unpkg.com/react/dist/react.js"**></**script**>  <**script src="https://unpkg.com/react-dom/dist/react-dom.js"**></**script**>  <**script src="https://unpkg.com/babel-standalone"**></**script**>  </**head**>  <**body**>  <**div id="root"**></**div**>  </**body**>  <**script type="text/babel"**>  **function** listTowns(selector) {  *//* ***TODO: define and generate*** component  ReactDOM.**render**(  component,  **document**.getElementById(selector)  );  }  </**script**>  </**html**> |

Submit just the listTowns() function, which takes one argument and contains all program logic, including rendering the components.

Create two Components: **App** and **Form**. Use **App** in order to store and display the data from the **Form** component. In the **Form** component render the html form and make sure to send data to the other component on submit.

### Screenshots

This is how the html look like:

## Title Bar

Write a **React component** that generates a title bar for a webpage. A title bar is composed of title text, a button and a menu list of navigation links. Clicking the button toggles the visibility of the menu. Use the following HTML for testing your solution:

|  |
| --- |
| titlebar.html |
| <!DOCTYPE **html**>  <**html lang="en"**>  <**head**>  <**meta charset="UTF-8"**>  <**title**>Titlebar</**title**>  <**script src="https://unpkg.com/react/dist/react.js"**></**script**>  <**script src="https://unpkg.com/react-dom/dist/react-dom.js"**></**script**>  <**script src="https://unpkg.com/babel-standalone"**></**script**>  <**style**>  .**header** { **background-color**: **#5555ff**; **color**: **white**; **height**: 60**px**; **position**: **relative**; **left**: 0; **top**: 0; }  .**button** { **cursor**: **pointer**; **display**: **block**; **width**: 50**px**; **height**: 50**px**; **margin**: 4**px**; **position**: **absolute**; **text-align**: **center**; **line-height**: 50**px**; **font-size**: 24**px**; }  .**title** { **display**: **block**; **position**: **absolute**; **left**: 80**px**; **font-size**: 32**px**; **font-family**: **"Helvetica"**, **sans-serif**; **line-height**: 60**px**; }  .**drawer** { **left**:0; **top**:55**px**; **float**:**left**; **position**: **absolute**; **background-color**: **#eeeeff**; **width**: 180**px**; **display**: **none**; }  .**menu-link** { **color**: **black**; **font-family**: **"Helvetica"**, **sans-serif**; **text-decoration**: **none**; **font-weight**: 600; **display**: **block**; **padding**: 4**px**; **padding-left**: 12**px**; }  .**menu-link**:**hover** {: **white**; **background-color**: **#5555ff**; }  .**post** { **margin**: 48**px**; }  </**style**>  </**head**>  <**body**>  <**div id="container"**>  <**div id="head"**></**div**>  <**div id="content"**>  <**article class="post"**><**p**>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</**p**></**article**>  </**div**>  </**div**>  </**body**>  <**script type="text/babel"**>  **class** TitleBar **extends** React.Component {  *//* ***TODO***  }  </**script**>  </**html**> |

Your component needs to take two properties – the title of the webpage, named title and a list of navigation links, named links. The render method **composes** the element and returns it.

The resulting HTML shown below has the following elements:

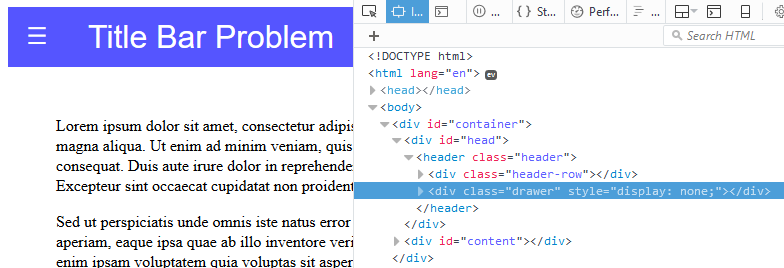
* <span> with class "**title**" which holds the **title** passed in as a property
* **<a>** with class "**button**" which **toggles** the visibility of the **menu**; you need to attach a **click event** to this element
* <div> with class "**drawer**" which contains the **navigation links**; it’s display css property needs to be **toggled** between block and hidden via the button listed above (save this internally as **state** – component properties cannot be changed at run time)
* <a>’s with class "**menu-link**" for each of the **navigation links**

The generated HTML needs to match this **structure** exactly.

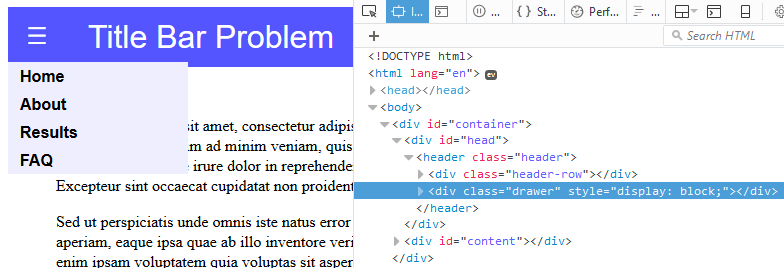
|  |
| --- |
| Titlebar Template |
| <**header class="header"**>  <**div class="header-row"**>  <**a class="button"**>**&#9776;**</**a**>  <**span class="title"**>Title</**span**>  </**div**>  <**div class="drawer"**>  <**nav class="menu"**>  <**a class="menu-link" href="href"**>Link name</**a**>  <**a class="menu-link" href="href"**>Link name</**a**>  <**a class="menu-link" href="href"**>Link name</**a**>  <**a class="menu-link" href="href"**>Link name</**a**>  </**nav**>  </**div**> </**header**> |

Submit only the class definition of the TitleBar component.

### Examples



↓



The title bar from the example can be generated using the following code:

|  |
| --- |
| Sample JavaScript |
| **let** links = [  [**'/'**, **'Home'**],  [**'about'**, **'About'**],  [**'results'**, **'Results'**],  [**'faq'**, **'FAQ'**]];  ReactDOM.**render**(  <**TitleBar title="Title Bar Problem" links=**{links} />,  **document**.getElementById(**'head'**)  ); |

## Dollar Football

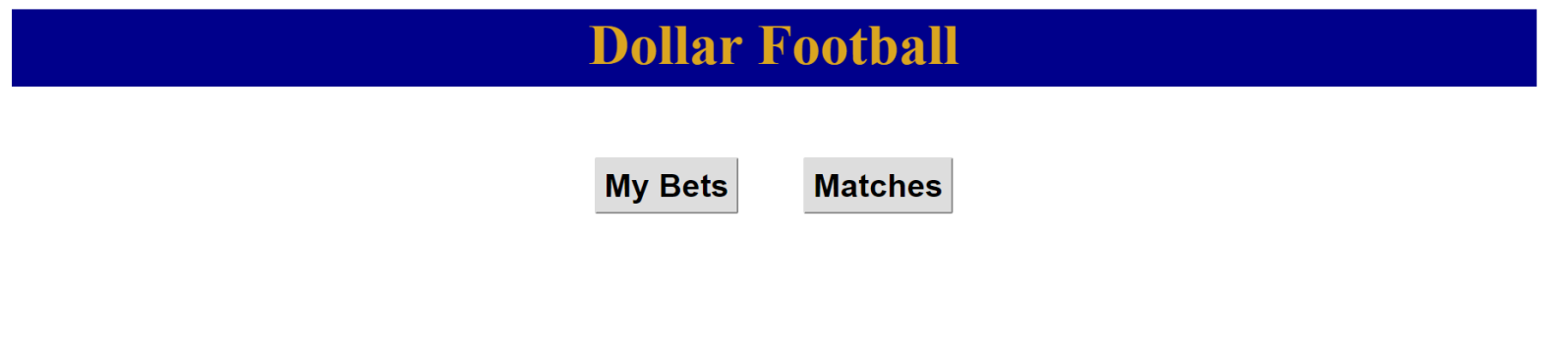
You will be given an HTML file and a JavaScript file, containing working code which almost works. Your task is to migrate it to ReactJS and render the HTML with it. The code simulates a simple application for betting on football matches. The matches are hardcoded into a database at Kinvey, and are extracted with a HTTP request.

Examine the code and see how it works, it should work perfectly, when you finish migrating it to ReactJS.

Keep all **HTML classes** the same way as they were before. You should have a main component called “AppView” which renders all other components.

Submit in the judge a function that returns **all Components**, including the **main component**. Do **NOT** make it an **IIFE**.

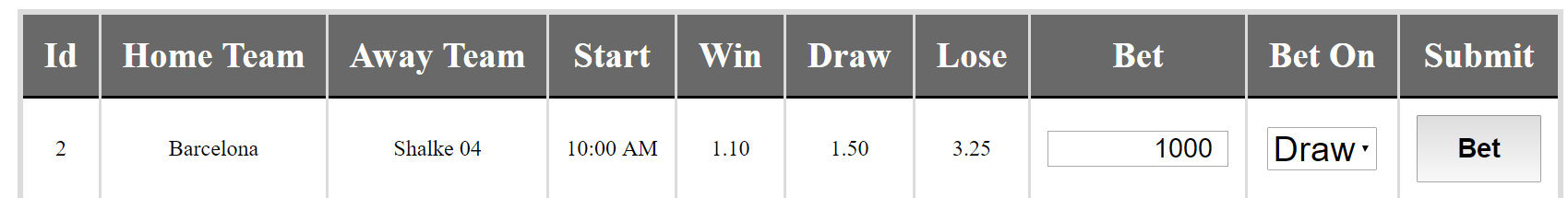
### Screenshots:



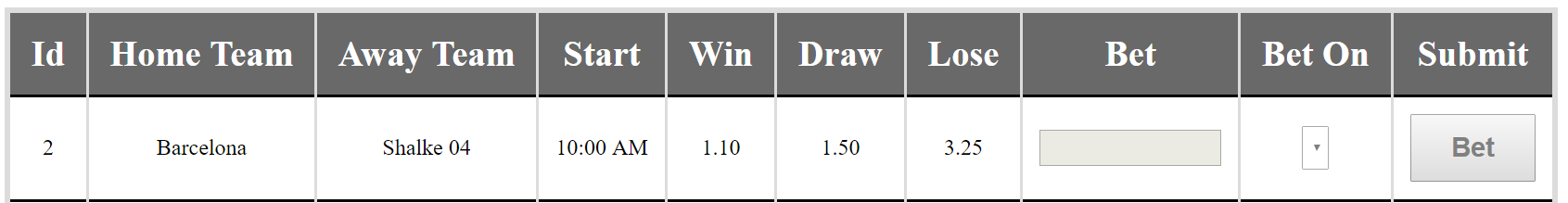
This is the main page, that we start our application with.



This is what we see when we click on [Matches]. We see all the available matches, **sorted** by **start time**.



As you see we can bet on a team, choose the value to bet, choose on who to bet and bet.



When we bet the fields become disabled so that we can’t bet for the same team twice.



Finally, when we click on [My Bets] button, we see all bets we’ve placed.

## \*\* Realtime Chat

In this task we are going to implement a functioning chat using React and Firebase.

### HTML and JavaScript Code

You are given the following **HTML** code:

|  |
| --- |
| chat.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Chat</**title**>  <**style**>  **#message** {  **width**: 100%;  **margin**: 10**px** 0 10**px** 0;  **display**: **block**;  **box-sizing**: **border-box**;  **font-size**: 1.2**rem**;  }  .**messageContainer**{ **border**: 1**px solid blue**; **margin**: 5**px**;}  **button**{ **margin**: 0 5**px**; }  **#root**{ **width**: 1080**px**; }  </**style**>  <**script src="https://fb.me/react-15.3.0.min.js"**></**script**>  <**script src="https://fb.me/react-dom-15.3.0.min.js"**></**script**>  <**script src="https://www.gstatic.com/firebasejs/3.3.0/firebase.js"**></**script**>  <**script src="https://cdn.firebase.com/libs/reactfire/1.0.0/reactfire.min.js"**></**script**>  <**script src="https://unpkg.com/babel-standalone"**></**script**>  <**script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"**></**script**> </**head**> <**body**> <**div id="root"**></**div**> <**script**>  **var *config*** = {  **apiKey**: **"AIzaSyBenpa0HaIOwkDdvb0SC9LfvJVsKGXvqos"**,  *// Only needed if using Firebase Realtime Database (which we will be in this example)* **databaseURL**: **"https://chat-25654.firebaseio.com/"** };   firebase.initializeApp(***config***); </**script**> <**script type="text/babel" src="chat.jsx"**></**script**> <**script type="text/babel"**>  ReactDOM.render(<**Chatbox message="Test message" author="Nameless Author"** />, **document**.getElementById(**"root"**)); </**script**> </**body**> </**html**> |

It comes together with the following **JavaScript** code:

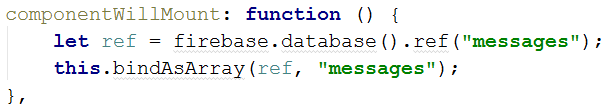
|  |
| --- |
| chat.jsx |
| **let *Chatbox*** = React.createClass({  *//****TODO*** }); |

We're going to be using some extra libraries for this task, as you can see in the HTML we're going to use Firebase's SDK and another library ReactFire which gives react components some extra capabilities.

We'll begin with adding the ReactFire capabilities to our Chatbox element:



Next we should setup the initial state of the element by using the getInitialState function, after we have our base state we'll use the functionality from the ReactFire and Firebase libraries to connect our state to a collection in Firebase:



With just these steps the messages property from our state will get updated every time the message collection in Firebase changes. Using this finish the Chatbox class so it has functionality for seeing all messages and their authors and posting a message with a given text and author.

### Screenshots



Here are some screenshots from the structure as shown from the React Add-on in Chrome:



And here's the structure of a message:



### Hints

* You can use variables in the render method of the element to remap the array of messages from the server to HTML elements.

### Bonus

Optionally you can implement extra functionality for deleting and updating entries.