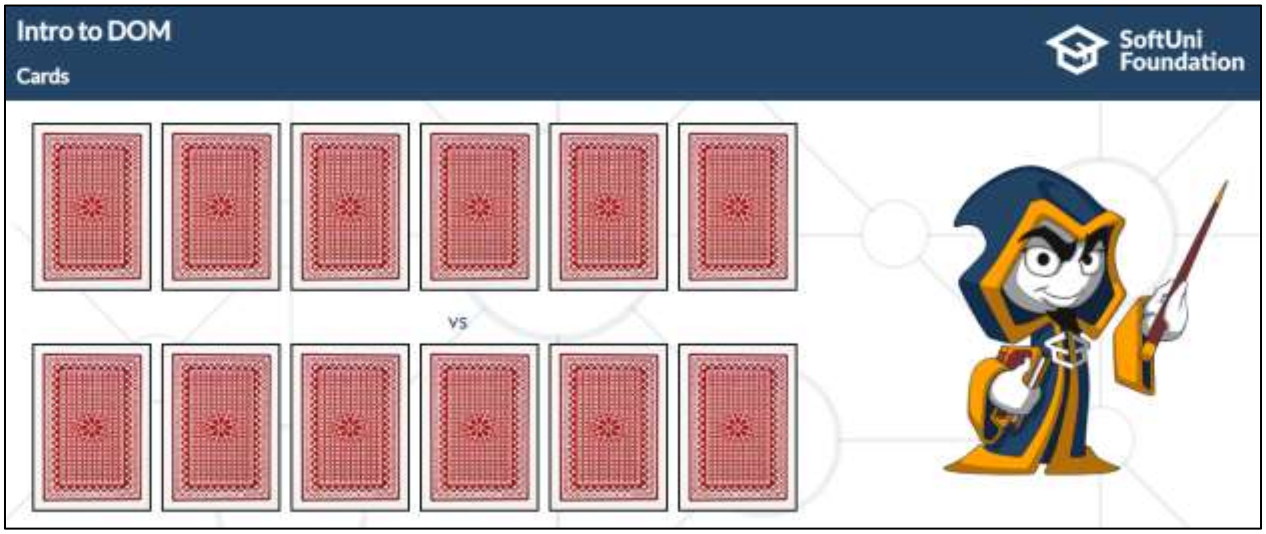
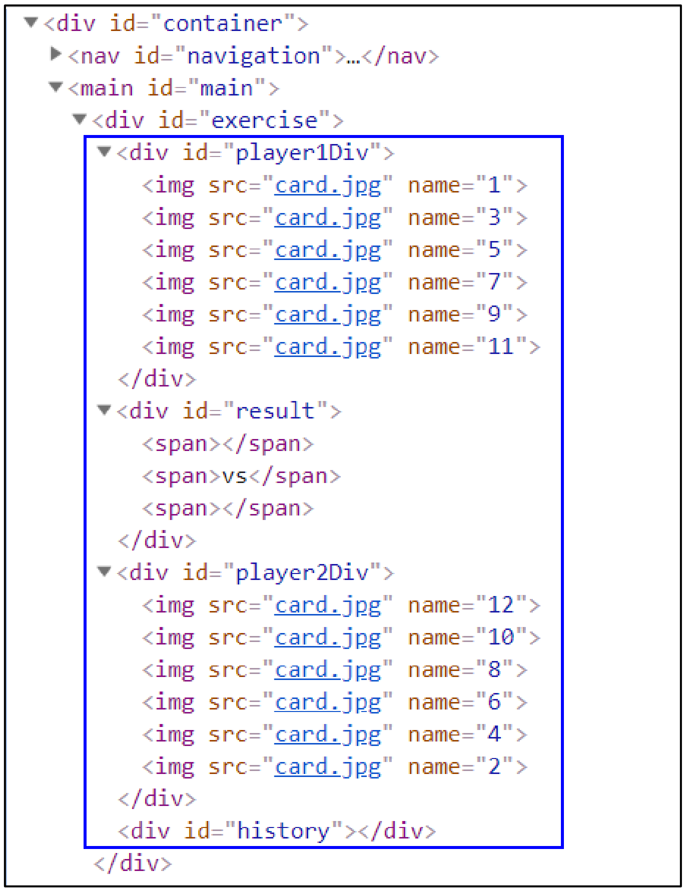
Exercise: Intro to DOM

Problems for exercises and homework for the "JavaScript Fundamentals" course @ SoftUni
Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1426

1. Cards

In this problem, you should **create a JS functonality** which **checks all cards**, and shows which one **is greater** and **keeps history of all hands**.

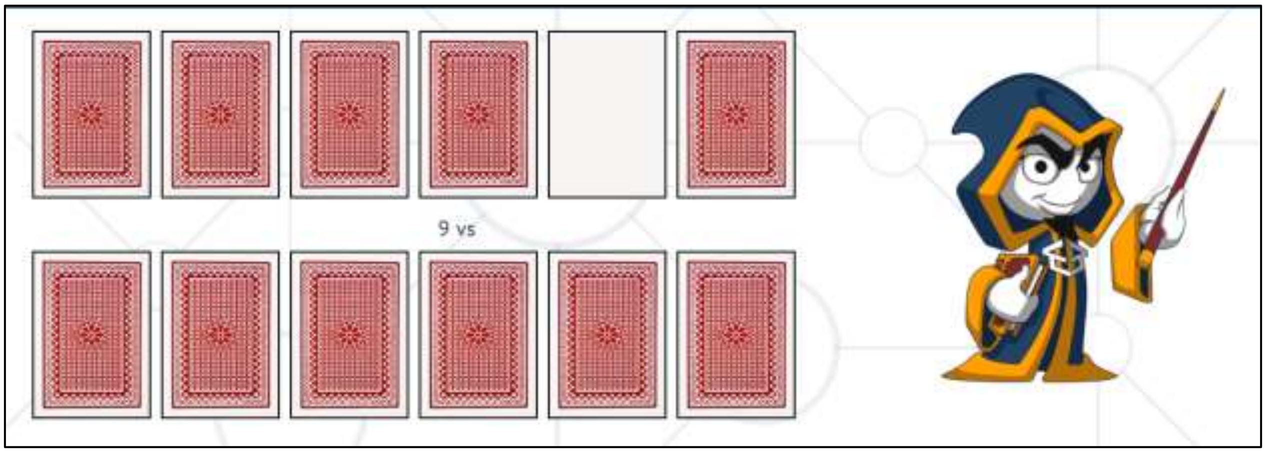


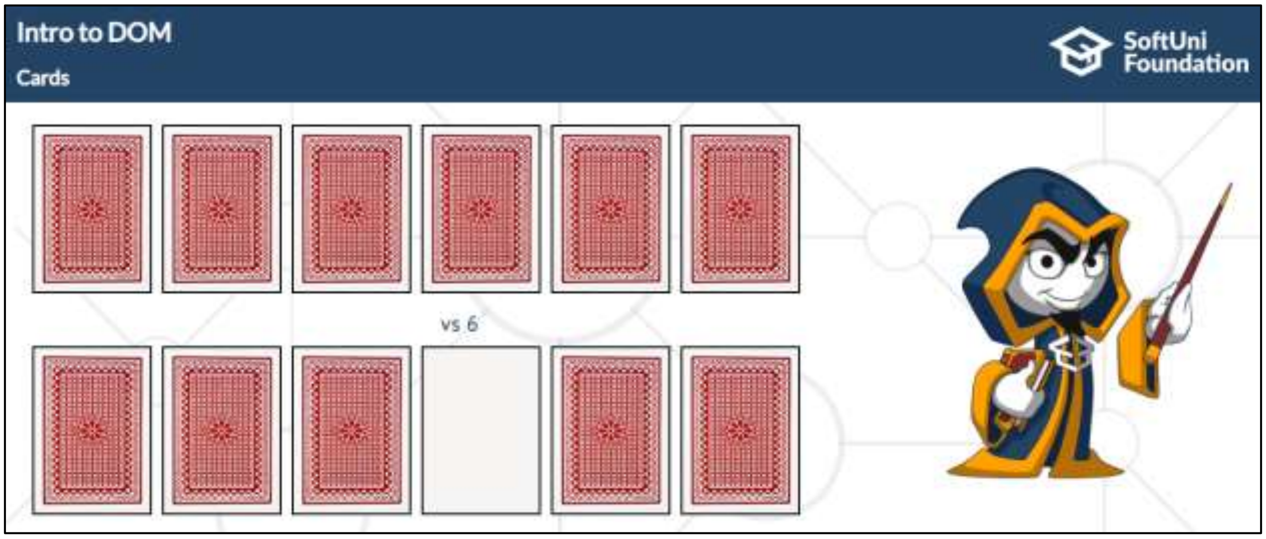


```
▼<div id="container">
  ▶<nav id="navigation">…</nav>
  ▼<main id="main">
    ▼<div id="exercise">
      ▼<div id="player1Div">
          <img src="card.jpg" name="1">
          <img src="card.jpg" name="3">
          <img src="card.jpg" name="5">
          <img src="card.jpg" name="7">
          <img src="card.jpg" name="9">
          <img src="card.jpg" name="11">
        </div>
      ▼<div id="result">
          <span></span>
          <span>vs</span>
          <span></span>
        </div>
      ▼<div id="player2Div">
          <img src="card.jpg" name="12">
          <img src="card.jpg" name="10">
          <img src="card.jpg" name="8">
          <img src="card.jpg" name="6">
          <img src="card.jpg" name="4">
          <img src="card.jpg" name="2">
        </div>
        <div id="history"></div>
      </div>
```

First of all, you need to **add click events** to **all cards**. After that, when some of the cards are clicked, the current background card must be changed with "**whiteCard.jpg**" picture (Which is given to us with the skeleton) and **append** the **card name** to the one of the **span elements into the div with id result.**

If a card from the top side is clicked, you should **append the card name** to **the left span** (first empty span), otherwise you should **append the card name** to **the right span** (second/last span).



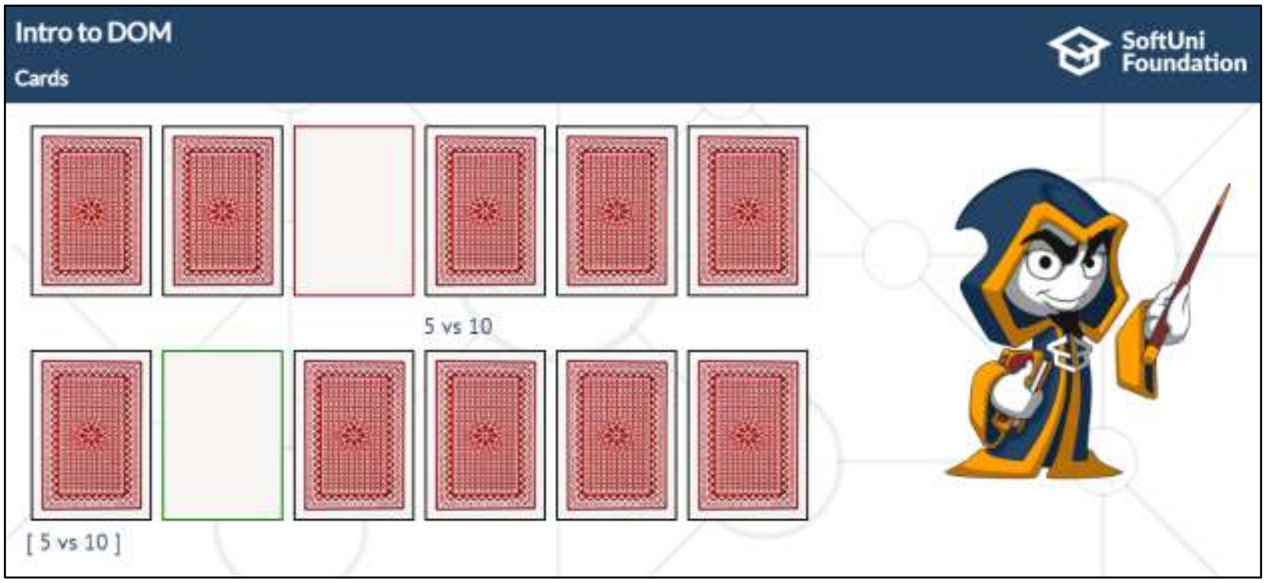

Intro to DOM

Cards

SoftUni Foundation

vs 6

When **one card** from **both sides is selected,** you should **check** which card name is **greater**. The greater card must be bordered with "**2px solid green**" and lower card with "**2px solid darkred**".

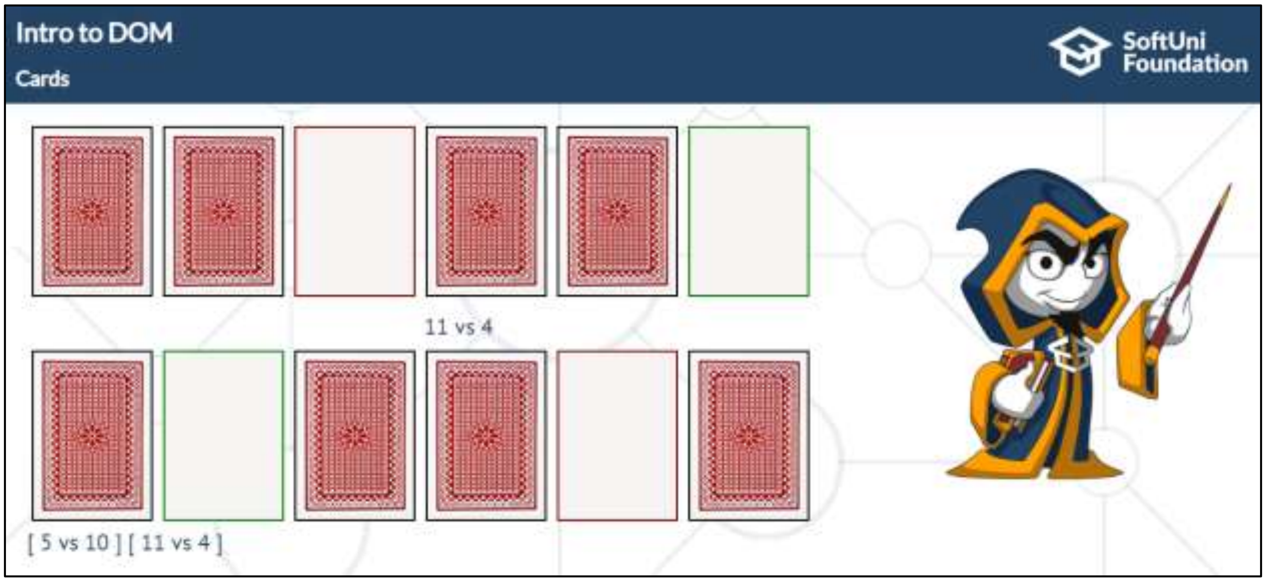

Intro to DOM

Cards

SoftUni Foundation

5 vs 10

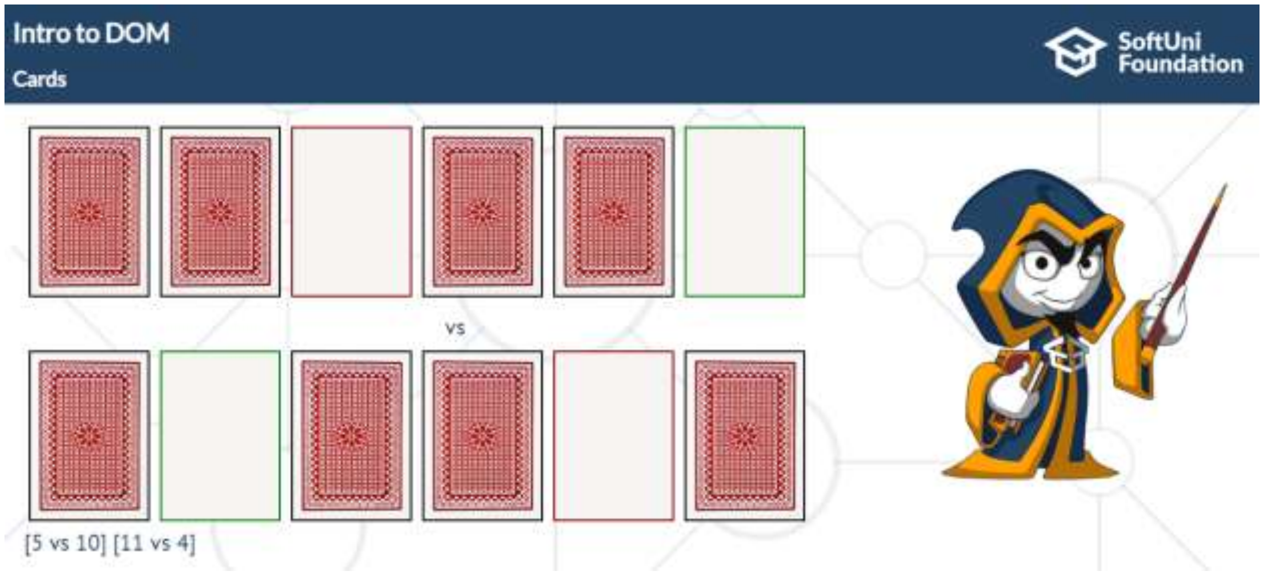
[5 vs 10]



After that you should use **`setTimeOut` function** to **clear** the **span elements** which **hold current cards names. (**Use **2000 ms** to solve it**).** Also, **after every hand** you should push the current cards names to the history div in format:

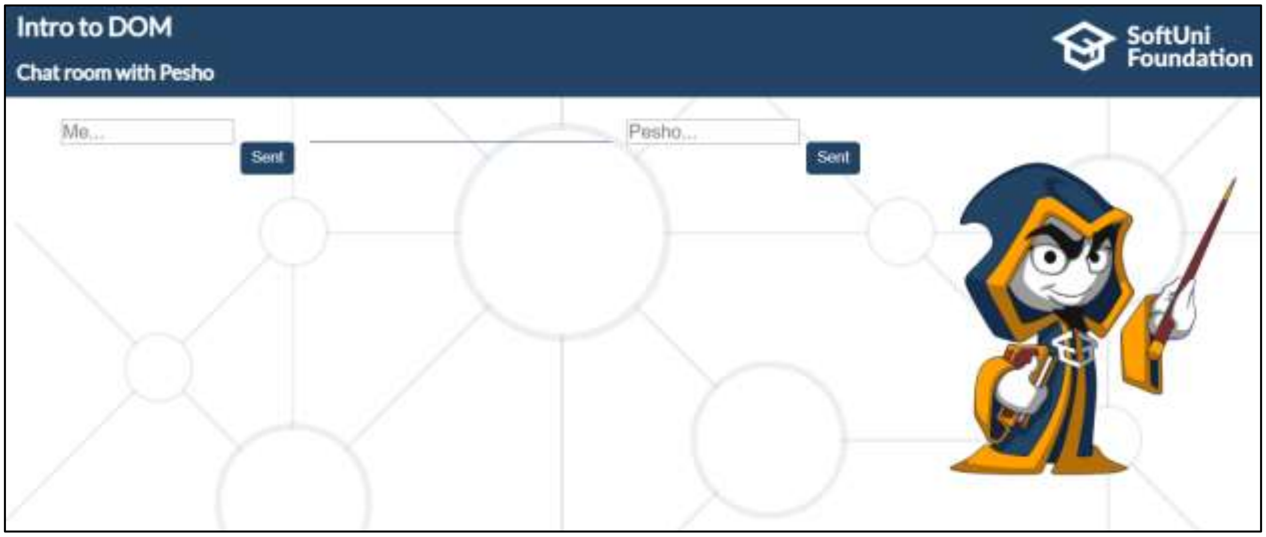
[{top side card name} **vs** {bottom side card name}**]**

If we take for example, the examples so far, after 2 or more seconds the expected result should be… (**look the next example below**).



2. Chat room with Pesho

In this problem, you should **create a JS functonality** which **creates a chat room where we chat with our good friend Pesho.**



First of all, **don't forget** to **add event listeners** to **both of the buttons**!

Any sent message, regardless of the sender, is saved into a **div element**. This div contains **two elements** (**span** and **paragraph**).

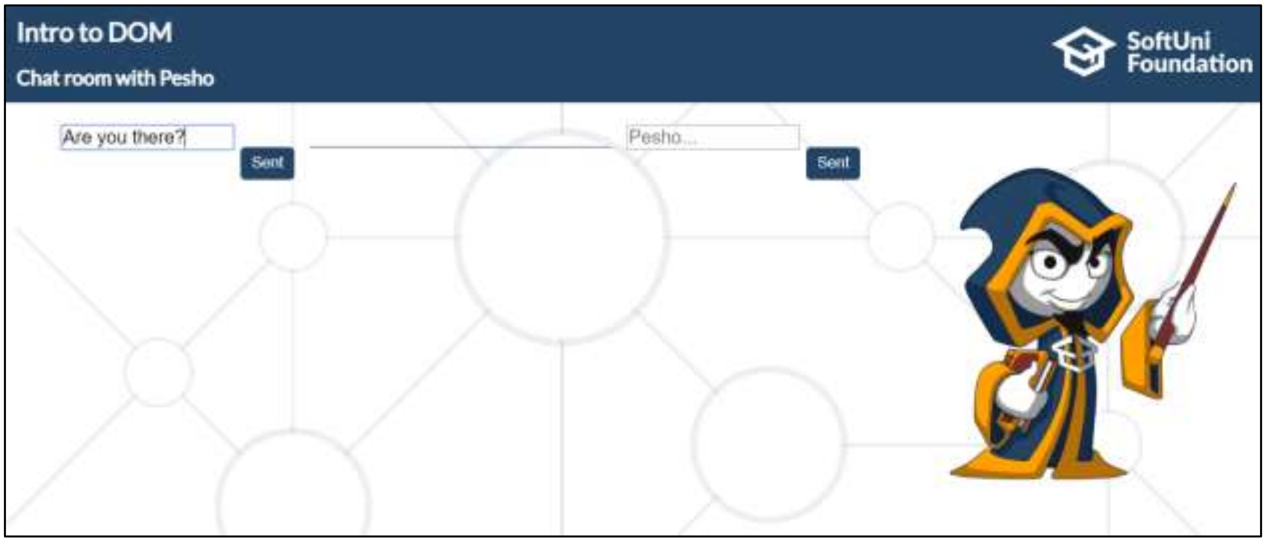
The span element should contain only the sender name (Me or Pesho).

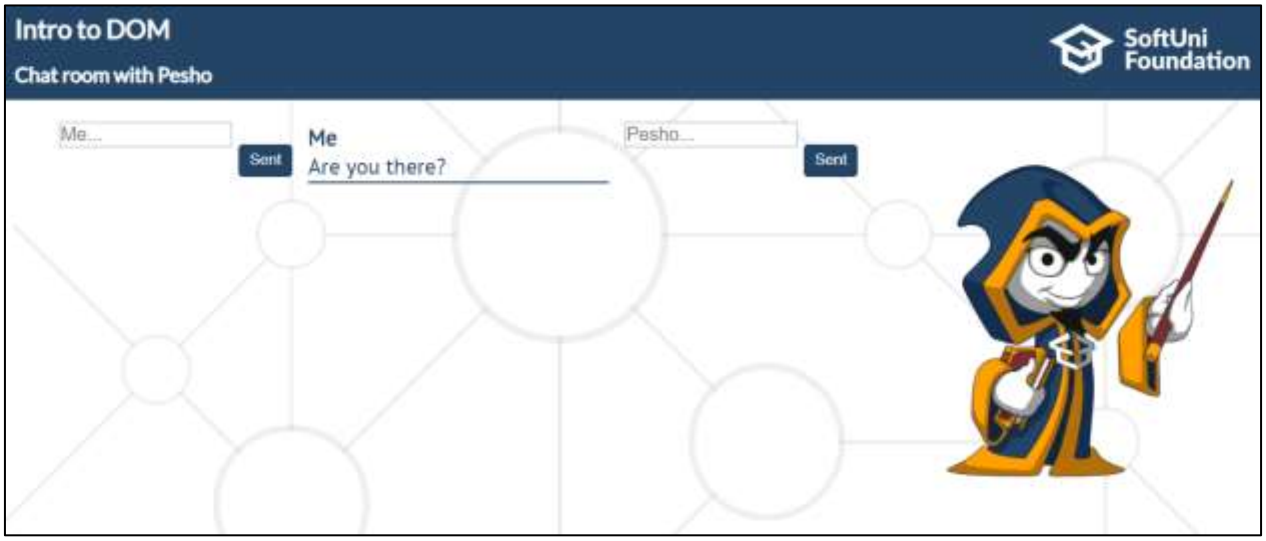
The p element should contain the current message.

The final step is to append the **current div** to the div which has an **id chatChronology.**

Keep in mind that if the sender is "Me", the **text-align** inside the current div should be **left**, otherwise when Pesho sends some message the **text-align** needs to be **right**.

<u>After each click on the buttons, **the current message should be cleared**.</u>

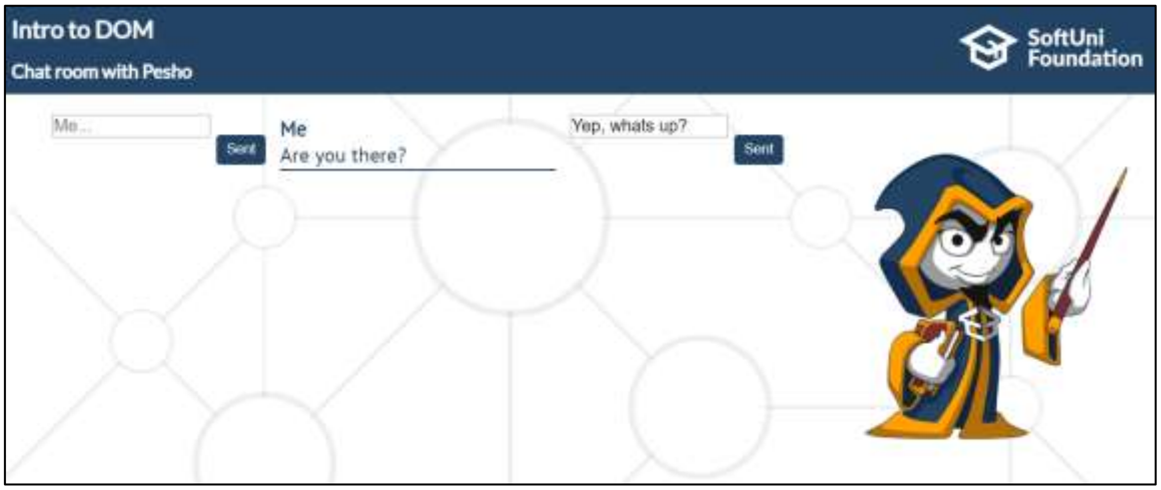



Intro to DOM

Chat room with Pesho

Me

Are you there?

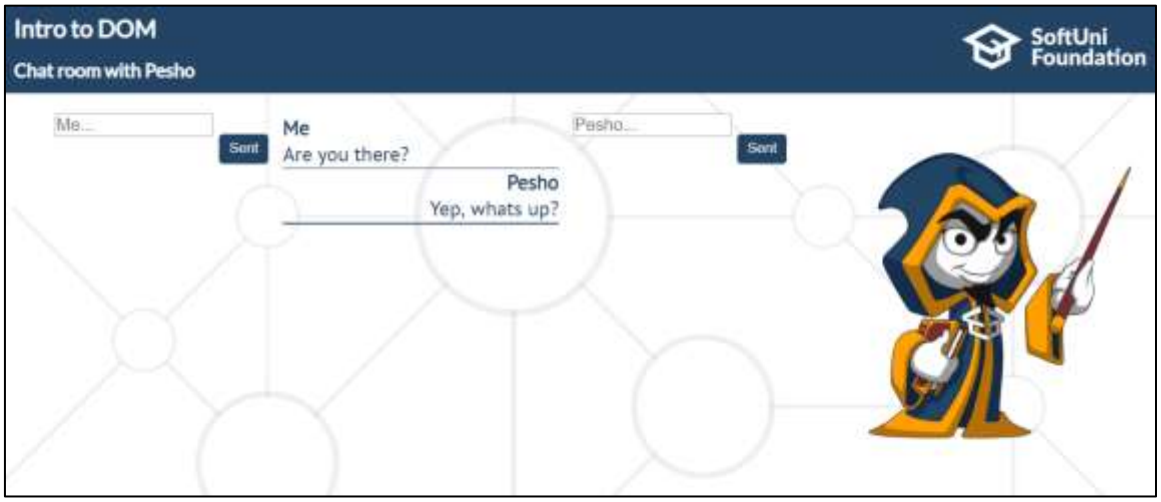

Intro to DOM

Chat room with Pesho

Yep, whats up?

Me

Are you there?


Intro to DOM

Chat room with Pesho

Me

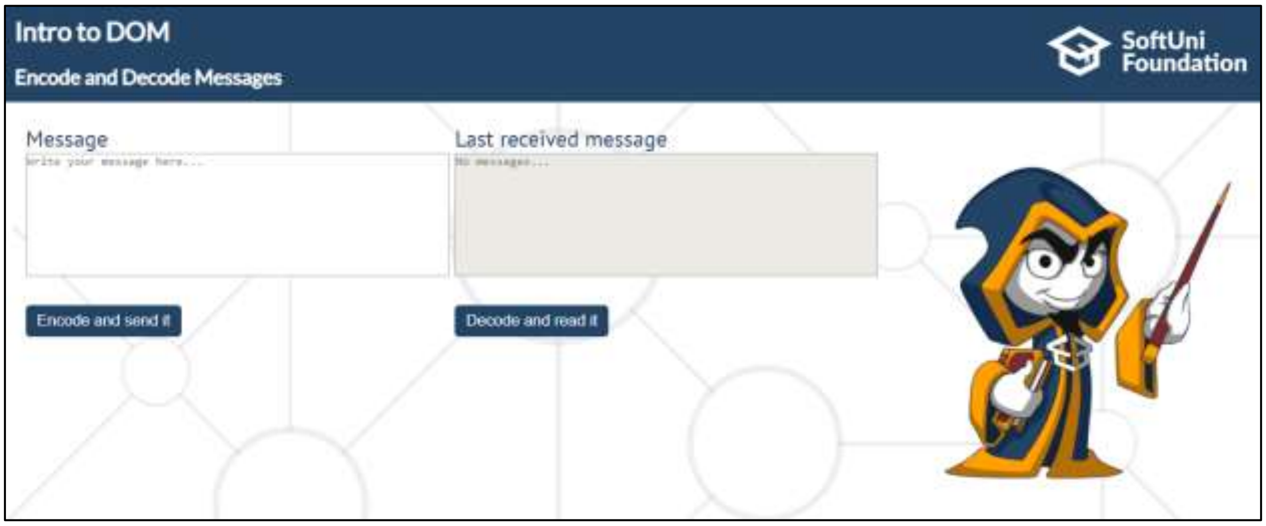
Are you there?

Pesho

Yep, whats up?

3. Encode and Decode Messages

In this problem, you should **create a JS functonality** which **encodes and decodes some messages which travel to the network.**



This program should contain **two functionalities**.

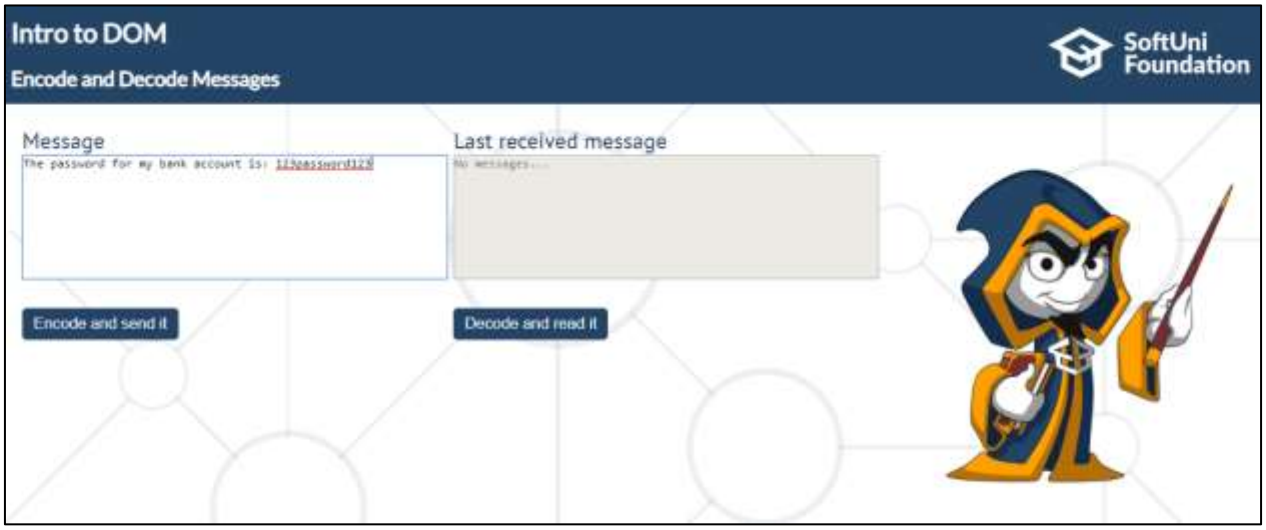
The first one is to **encode the given message** and **send it** to the **receiver**.

The second one is to **decode the received message** and **read it (display it)**.

When the "**Encode and send it**" button is clicked, you should get the given message from the first textarea. When you get the current message, you should encode it as follows:

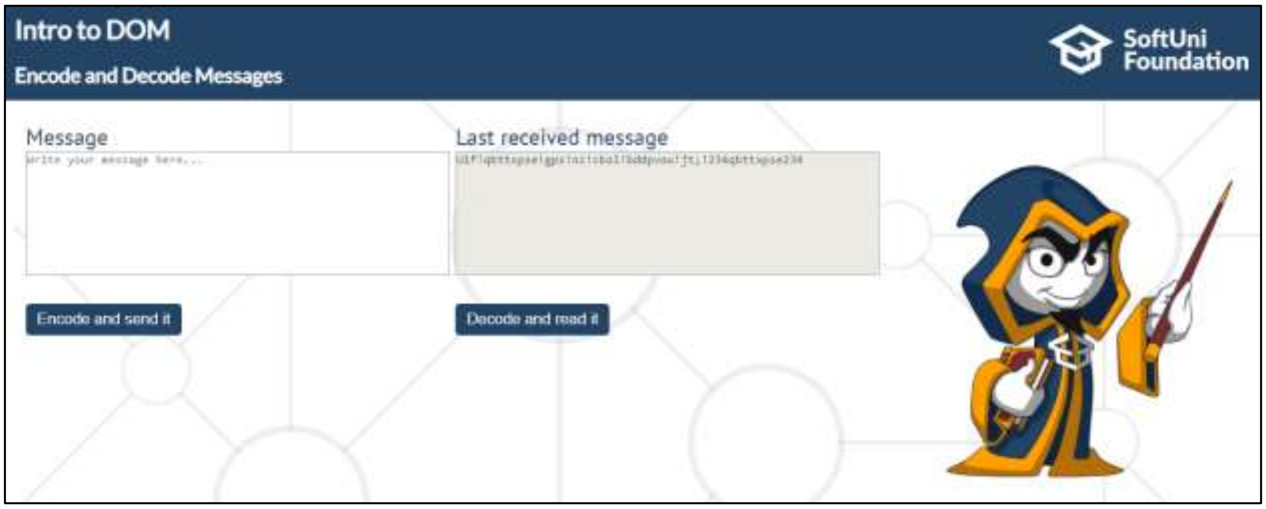
You **should change** the **ASCII CODE** on **every single character** in that message, when you **add 1** to the current **ASCII NUMBER**, that represent the current character in that message.

When you do that just **clear** the **sender textarea,** and **append the encoded message** to the **receiver textarea.**



```
The password for my bank account is: 123password123
```

```
No messages...
```

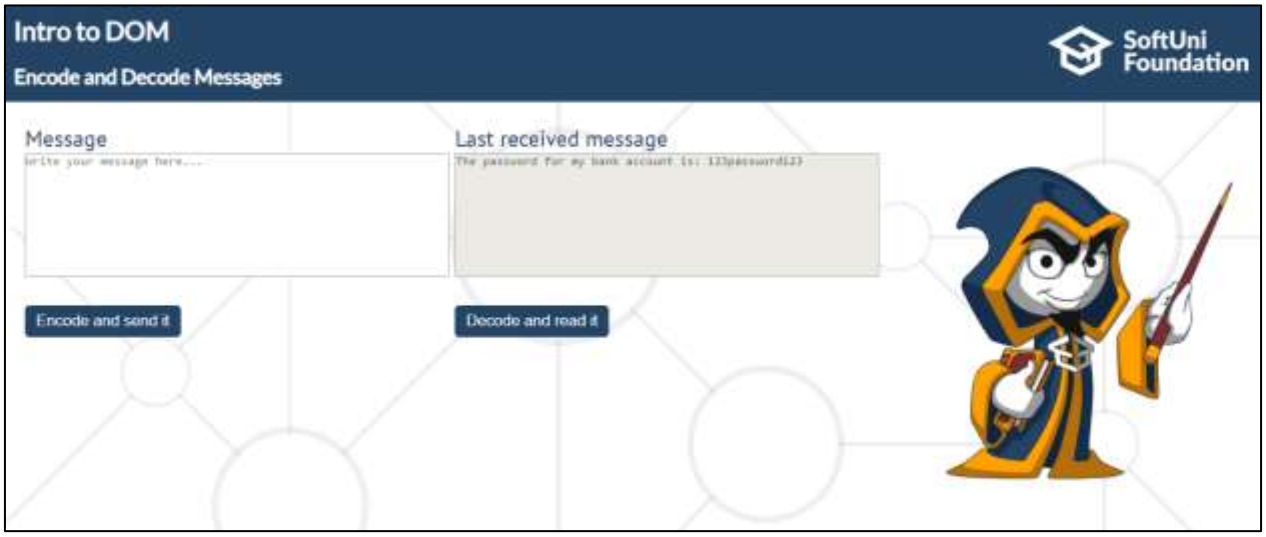


```
Write your message here...
```

```
Uif!qbttxpse!gps!nz!cbol!bddpvou!jt;!234qbttxpse234
```

After that, when the "**Decode and read it**" button is clicked. You need to get the **encoded message** from **the receiver textarea** and do the **opposite logic** from encoding, **subtract 1** from the current **ASCII NUMBER**, that represents the current character in that message.

When you do that, just replace the **encoded message** with the already **decoded message** in the receiver textarea, to make it readable.

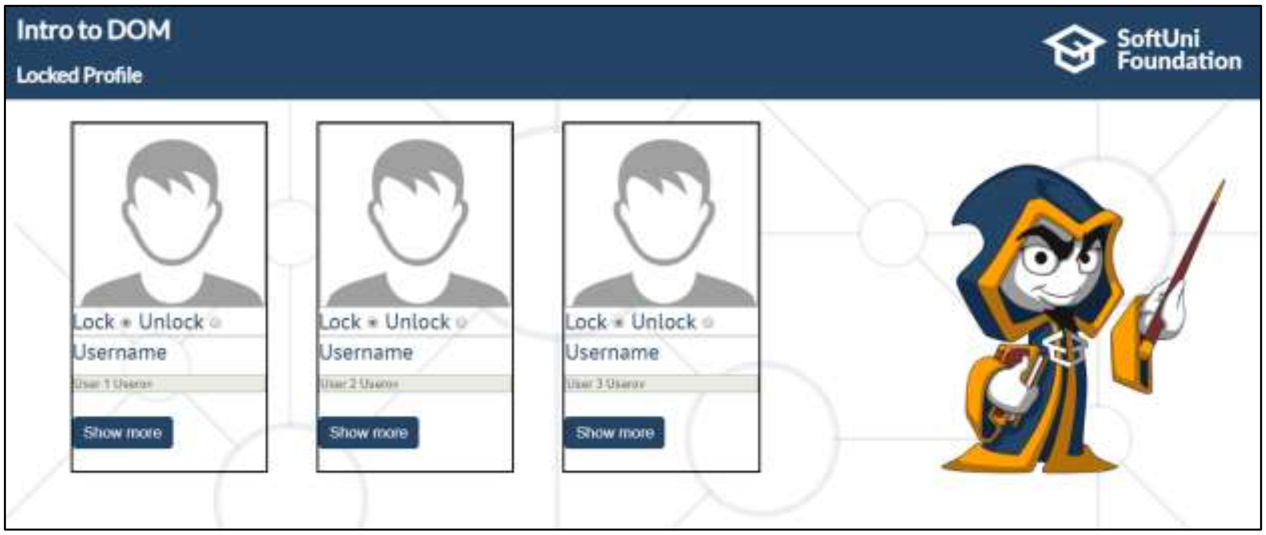


write your message here...

The password for my bank account is: 123password123

4. Locked Profile

In this problem, you should **create a JS functonality** which **shows** and **hides** the additional information about users.

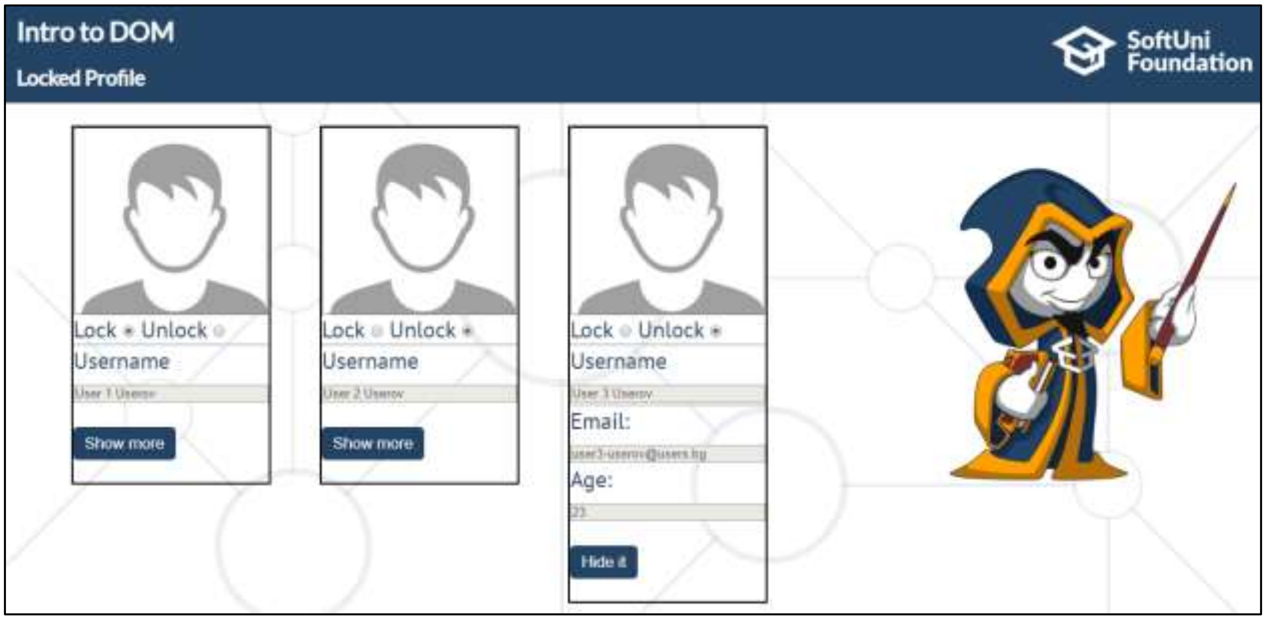


User 1 Usernm

User 2 Usernm

User 3 Usernm

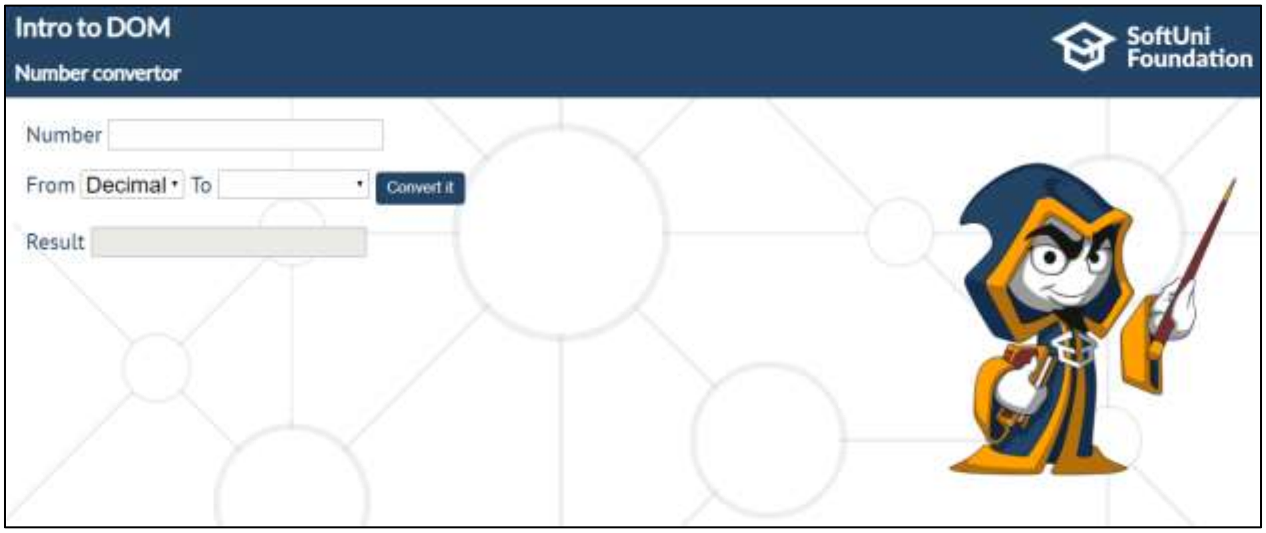
When one of the "**Show more**" **buttons** is clicked, the **hiden information** inside the div should be shown, only if **the profile is not locked**! If the current profile is **locked,** nothing should happen.



If the **hidden information is displayed** and we **lock the profile again**, the "**Hide it**" button should **not be working**! Otherwise, when the profile is **unlocked** and we click on the "**Hide it"** button, the new fields must hide again.

5. Number convertor

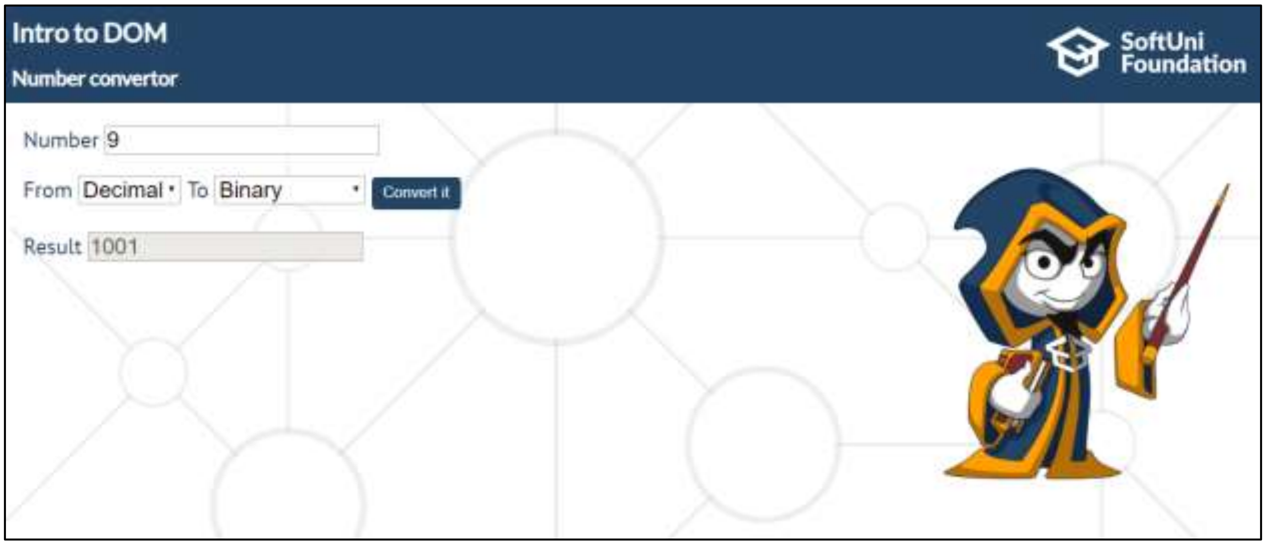
In this problem, you should **create a JS functonality** that **converts** a **decimal number** to **binary** and **hexadecimal**.



The given number will always be in **decimal format.** The "**from"** select menu will only have a **Decimal** option, but the "**To"** select menu will have **two options**: **Binary** and **Hexadeicmal**.

This means that our program should have the functionality to **convert decimal** to **binary** and **decimal** to **hexadecimal**.

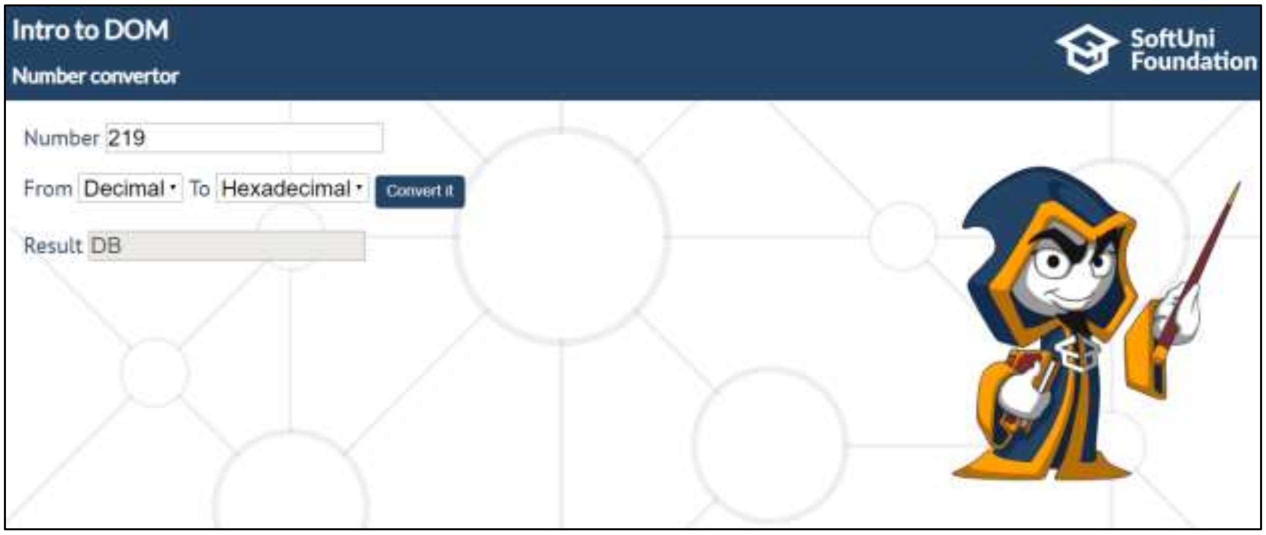
When the "**Convert it** "button is **clicked**, the expected result should appear in the "**Result"** input field.



Number 9

From Decimal To Binary

Result 1001



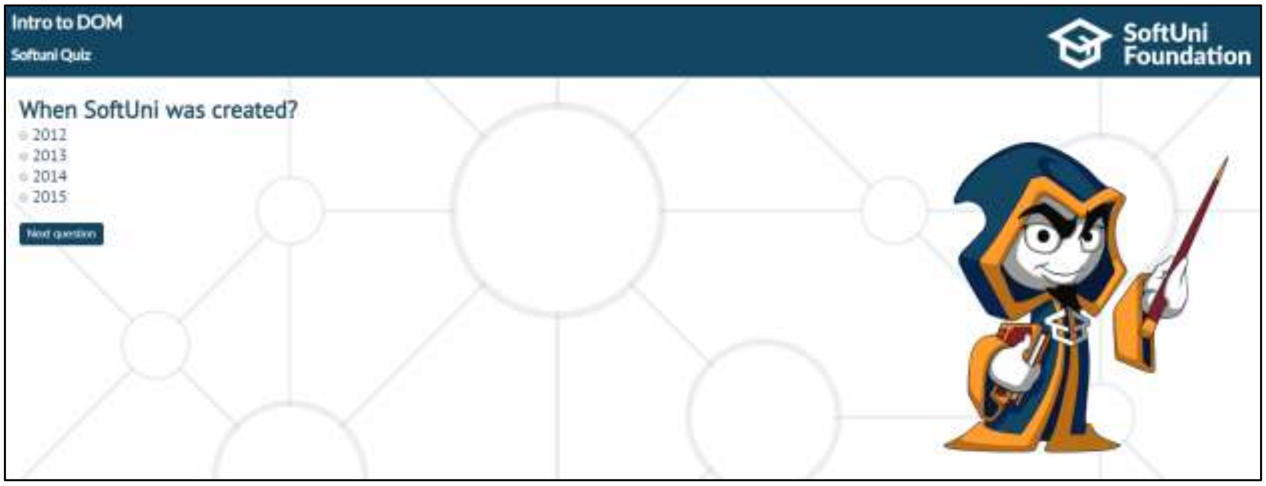
Number 219

From Decimal To Hexadecimal

Result DB

6. Softuni Quiz

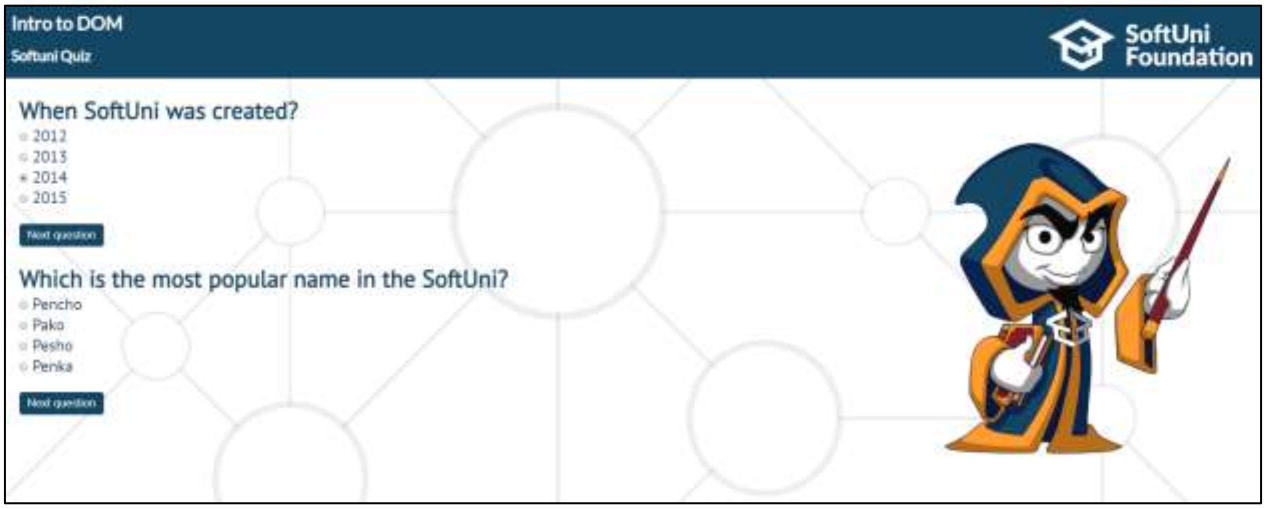
In this problem you should **create a JS functonality** about a quiz.


Intro to DOM
Softuni Quiz
When SoftUni was created?
- 2012
- 2013
- 2014
- 2015

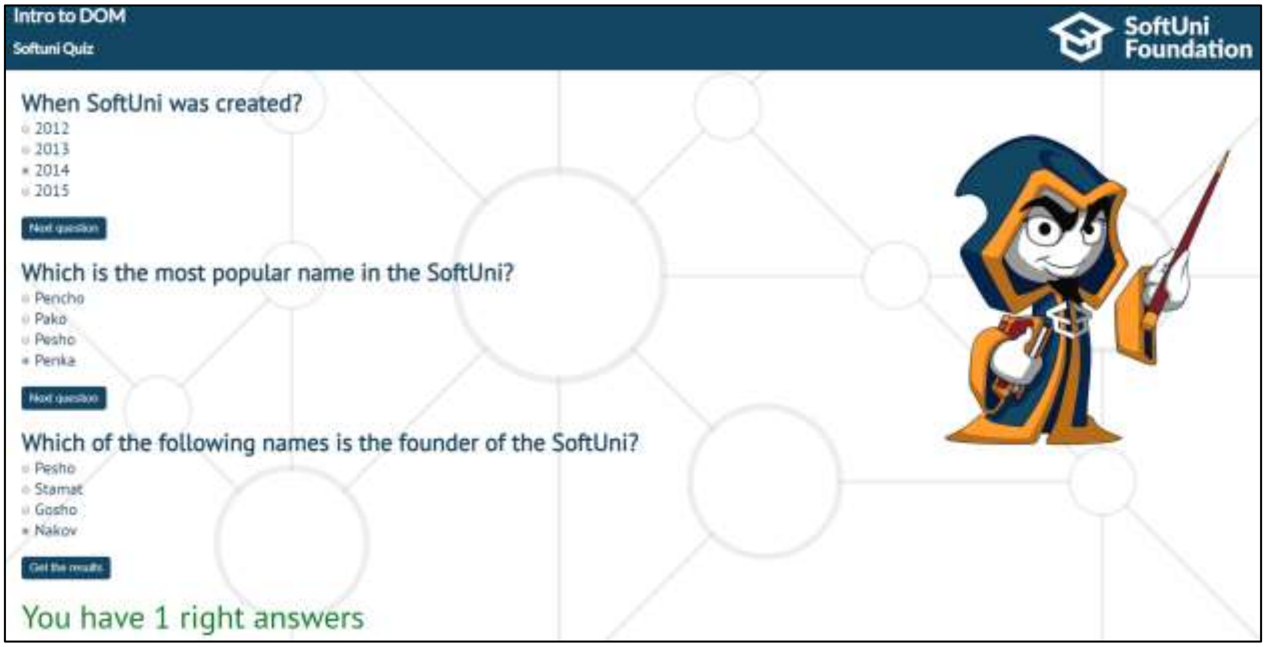
There are three **sections**, that contain **one question and 4 possible answers. The right answer is only one!** The expected functionality is when one of the **radio buttons is selected**, and "**Next question**" button is clicked, the next section **must appear (if any…)** If the third (last one) section is visible, when you press the button, "**Get the results", the final score** must show into the result div.

If all questions are answered right, you need to print: "`**You are recognized as top SoftUni fan!**",
otherwise just print "**You have {rightAnswers} right answers**".

The right answers are (**2013, Pesho and Nakov**).


Intro to DOM
Softuni Quiz
When SoftUni was created?
- 2012
- 2013
- 2014
- 2015

Which is the most popular name in the SoftUni?
- Pencho
- Pako
- Pesho
- Penka



When SoftUni was created?

- 2012
- 2013
- [✓ 2014]
- 2015

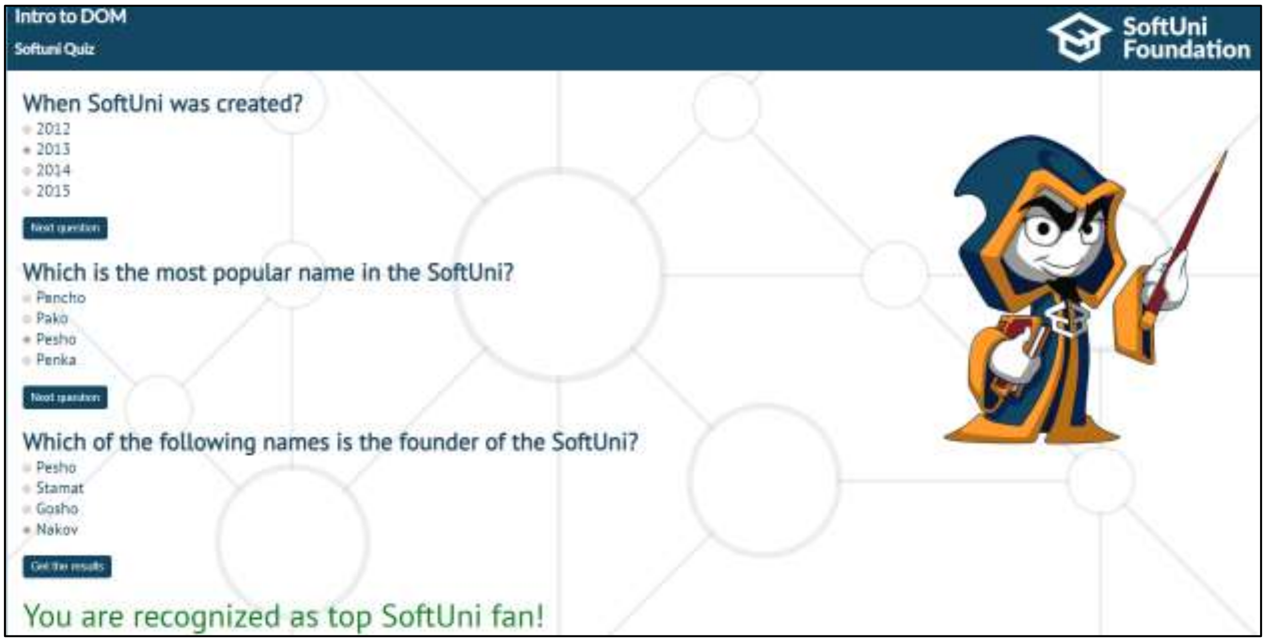
Which is the most popular name in the SoftUni?

- Pencho
- Pako
- Pesho
- [✓ Penka]

Which of the following names is the founder of the SoftUni?

- Pesho
- Stamat
- Gosho
- [✓ Nakov]

You have 1 right answers



When SoftUni was created?

- 2012
- [✓ 2013]
- 2014
- 2015

Which is the most popular name in the SoftUni?

- Pencho
- Pako
- [✓ Pesho]
- Penka

Which of the following names is the founder of the SoftUni?

- Pesho
- Stamat
- Gosho
- [✓ Nakov]

You are recognized as top SoftUni fan!

7. Table – Search Engine

In this problem, you should **create a JS functonality** that **searches** in a **table** by given input.

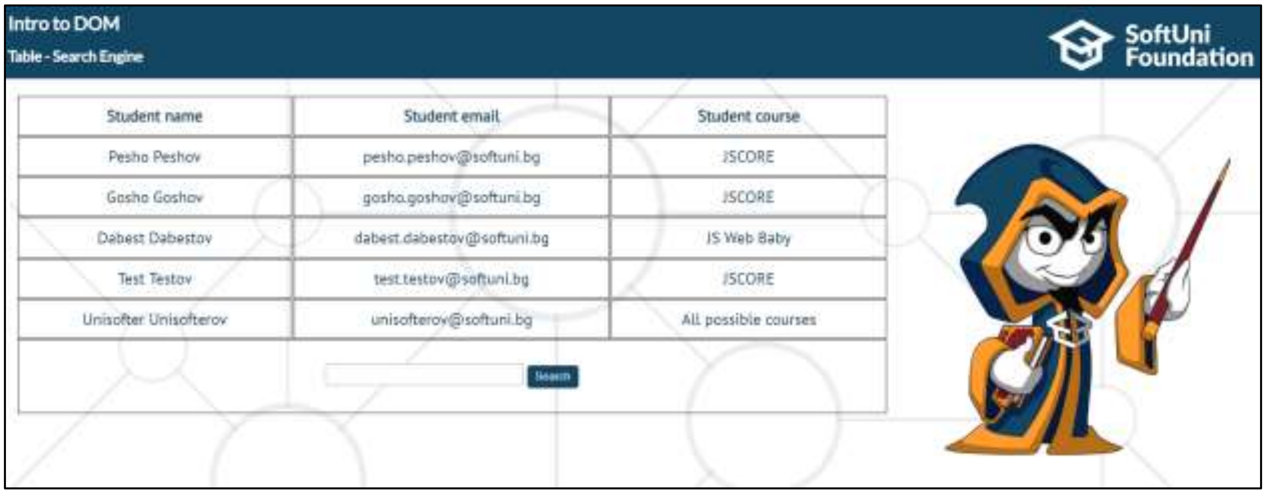

Intro to DOM

Table - Search Engine

Student name	Student email	Student course
Pesho Peshov	pesho.peshov@softuni.bg	JSCORE
Gosho Goshov	gosho.goshov@softuni.bg	JSCORE
Dabest Dabestov	dabest.dabestov@softuni.bg	JS Web Baby
Test Testov	test.testov@softuni.bg	JSCORE
Unisofter Unisofterov	unisofterov@softuni.bg	All possible courses

When the "**Search**" **button** is **clicked**, you should go through all cells in that table except for the first row (Student name, Student email and Student course) and you also need to check if the given input matches somewhere. (whether it's a **full word** or a **single letter**).

If any of the rows contain the string we have submitted, you need to add a **select class** to that row. If more than one row contains that string, add the **select class** to all of them.

Otherwise, if we **cannot** find anything, **nothing should happen**.

Note: After every search ("Search" button is clicked), you should **clear the input field** and **remove all already selected classes** (if available) from the old search, in order for the **new search** to come out only the **new result**.

For instance, if we try to find **dabest:**

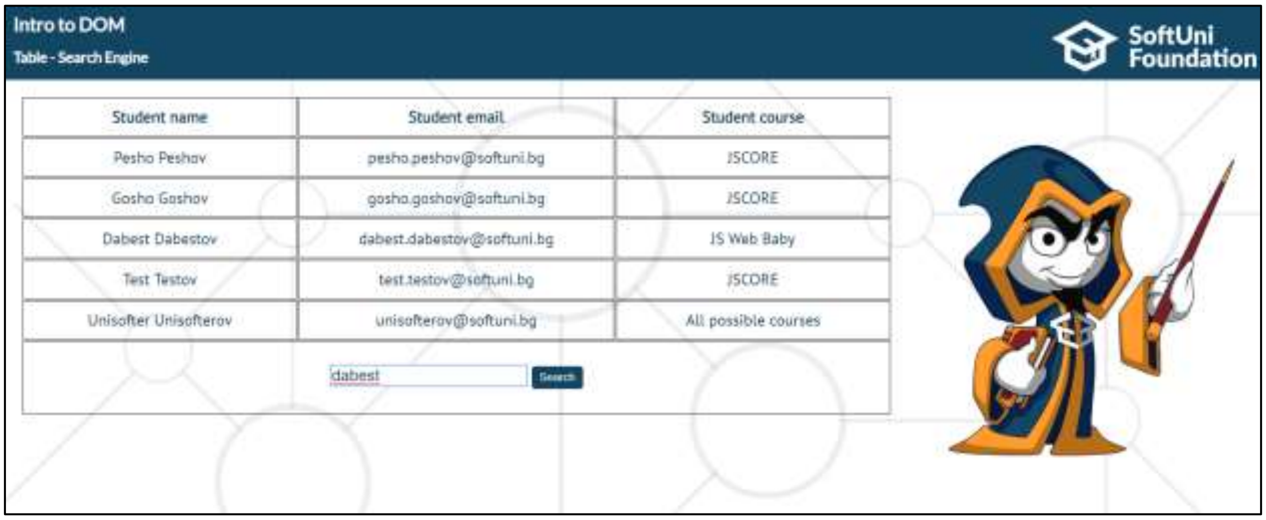
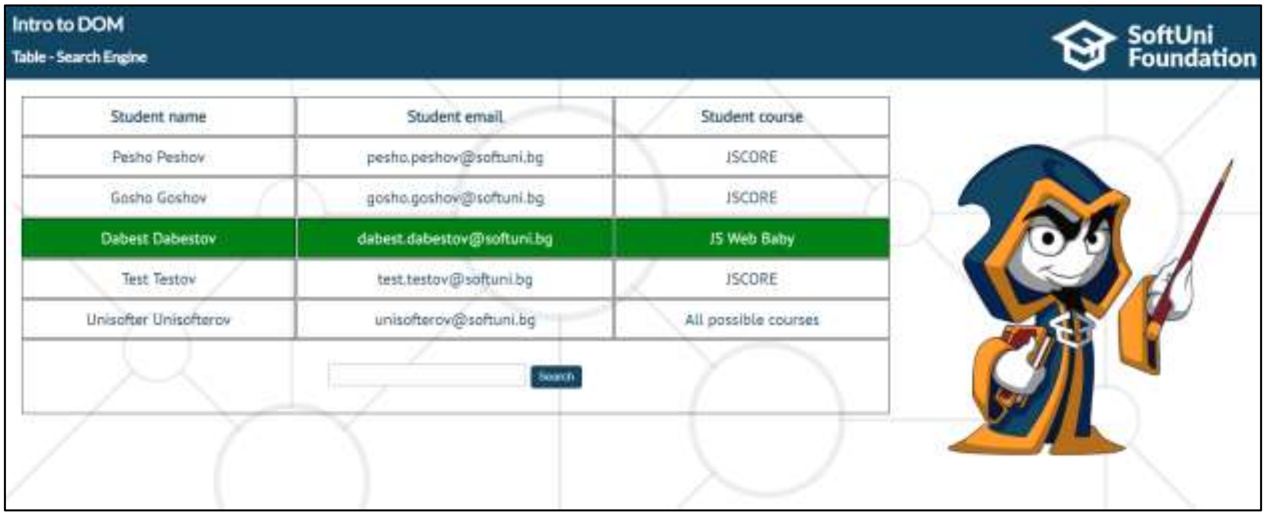

Intro to DOM

Table - Search Engine

Student name	Student email	Student course
Pesho Peshov	pesho.peshov@softuni.bg	JSCORE
Gosho Goshov	gosho.goshov@softuni.bg	JSCORE
Dabest Dabestov	dabest.dabestov@softuni.bg	JS Web Baby
Test Testov	test.testov@softuni.bg	JSCORE
Unisofter Unisofterov	unisofterov@softuni.bg	All possible courses

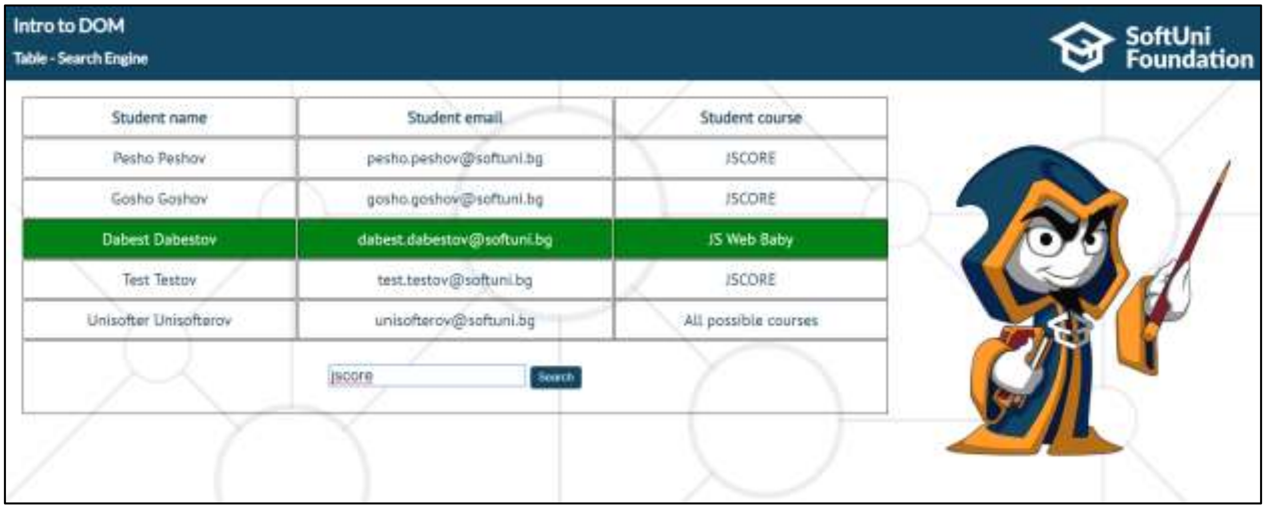
dabest

The result should be:


Intro to DOM
Table - Search Engine

Student name	Student email	Student course
Pesho Peshov	pesho.peshov@softuni.bg	JSCORE
Gosho Goshov	gosho.goshov@softuni.bg	JSCORE
Dabest Dabestov	dabest.dabestov@softuni.bg	JS Web Baby
Test Testov	test.testov@softuni.bg	JSCORE
Unisofter Unisofterov	unisofterov@softuni.bg	All possible courses

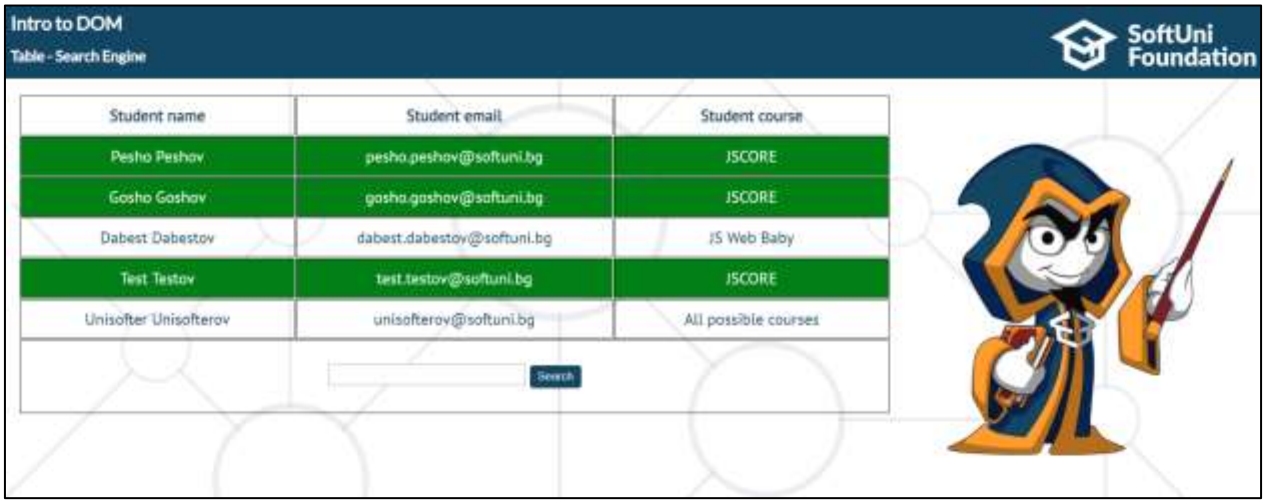
And if we try to find all students who have enrolled for the **jscore** course:


Intro to DOM
Table - Search Engine

Student name	Student email	Student course
Pesho Peshov	pesho.peshov@softuni.bg	JSCORE
Gosho Goshov	gosho.goshov@softuni.bg	JSCORE
Dabest Dabestov	dabest.dabestov@softuni.bg	JS Web Baby
Test Testov	test.testov@softuni.bg	JSCORE
Unisofter Unisofterov	unisofterov@softuni.bg	All possible courses

jscore

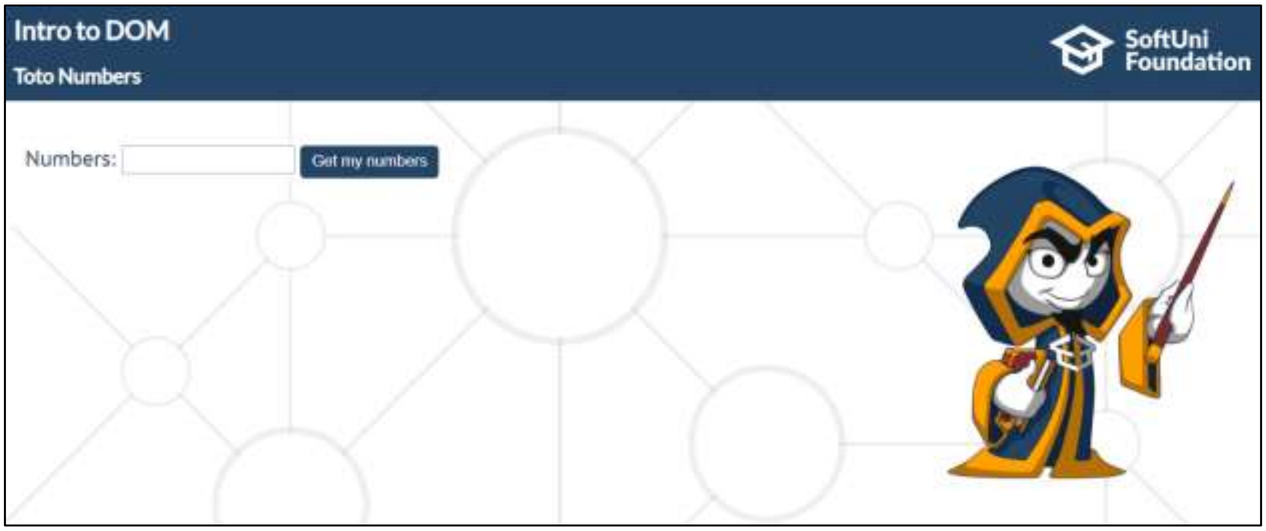
The expected result, should be:


Intro to DOM
Table - Search Engine

Student name	Student email	Student course
Pesho Peshov	pesho.peshov@softuni.bg	JSCORE
Gosho Goshov	gosho.goshov@softuni.bg	JSCORE
Dabest Dabestov	dabest.dabestov@softuni.bg	JS Web Baby
Test Testov	test.testov@softuni.bg	JSCORE
Unisofter Unisofterov	unisofterov@softuni.bg	All possible courses

8. Toto numbers

In this problem, you should **create a JS functonality** which **visualizes toto numbers.**



The Numbers input field should receive exactly 6 numbers in the range of [1 – 49].

If numbers input contains **more or less** numbers **than 6** or **one of all** the given numbers **does not cover the necessary range**, **nothing should happen**.

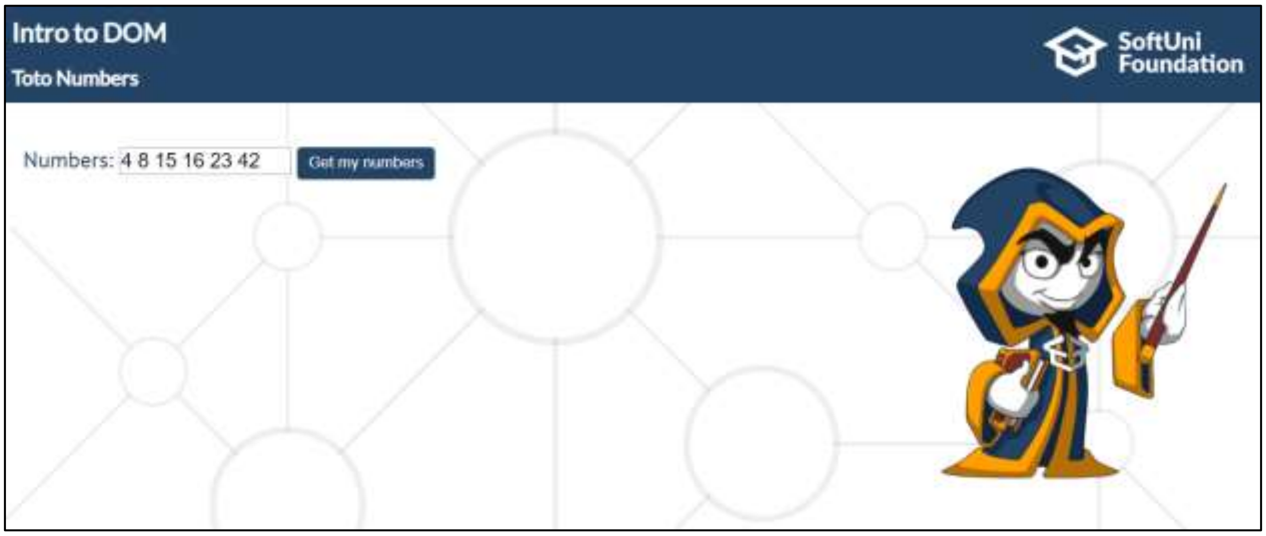
The expected functionality is as follows:

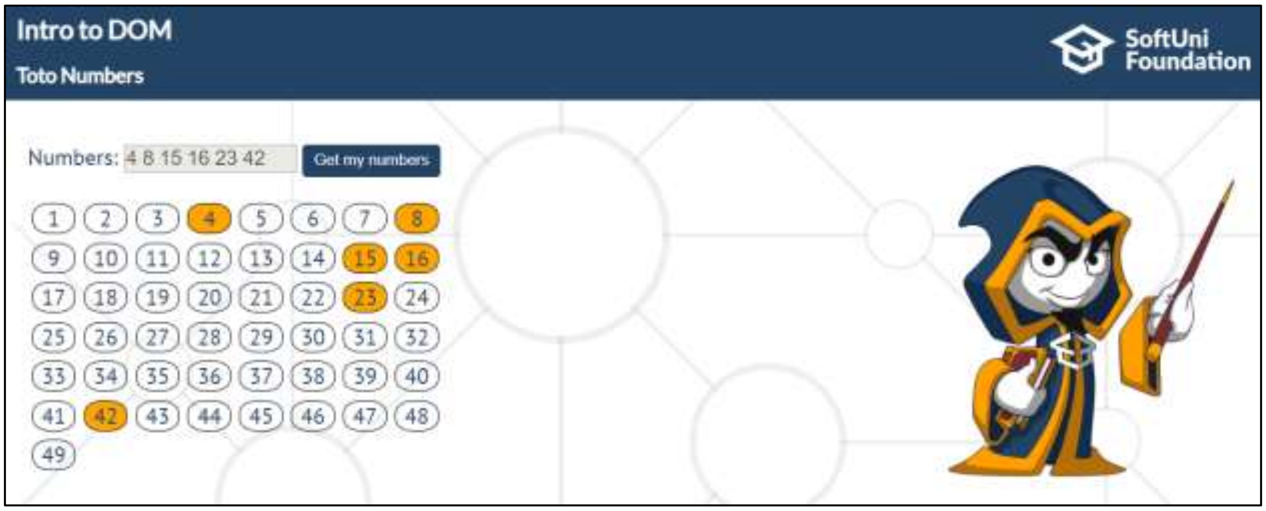
If the above conditions are met, when the "**Get my numbers" button** is **clicked,** you need to **create 49 div elements** and **append** them to the **div** which has an **id** "**allNumbers**".

Each of these 49 div elements **must** have the current number (1-49) as text inside them and **class** "**numbers"**.

If the **current number** is one of the Numbers from the **numbers field,** the **current div** should have an **orange background**.

When all numbers (div elements) **are visualized**, you should **disable the numbers input field** and **the "Get my numbers" button**.



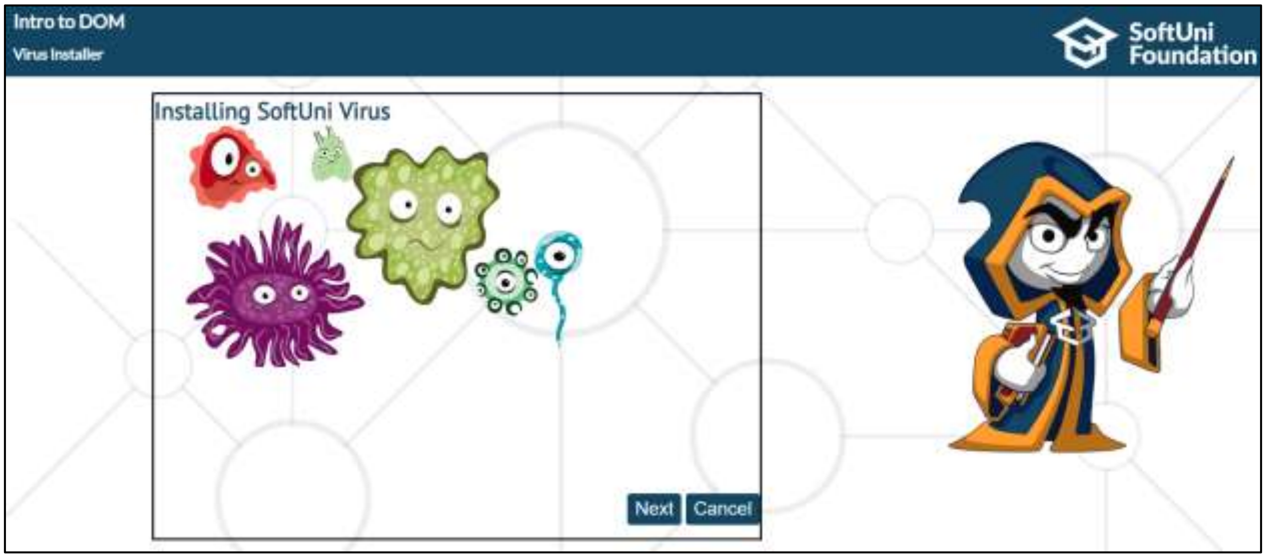


9. Virus Installer

In this problem, you should **create a JS functonality** that represents a "Wizard" which installs a "virus" on your computer.

To have the virus installed successfully, you need to go through **three steps.**

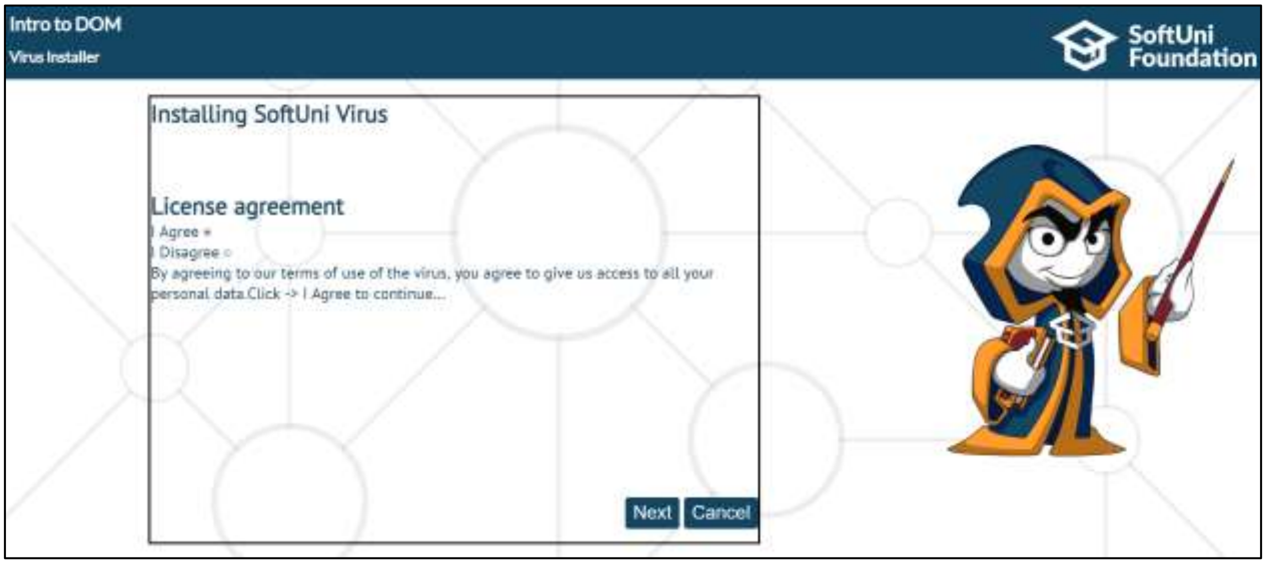
At the beginning (when you open the html file) you will see:



If the "**Next button**" is clicked, you should switch to the next step. (First Step)

You need to hide the background image into the **id content** element, and show the **firstStep div.**

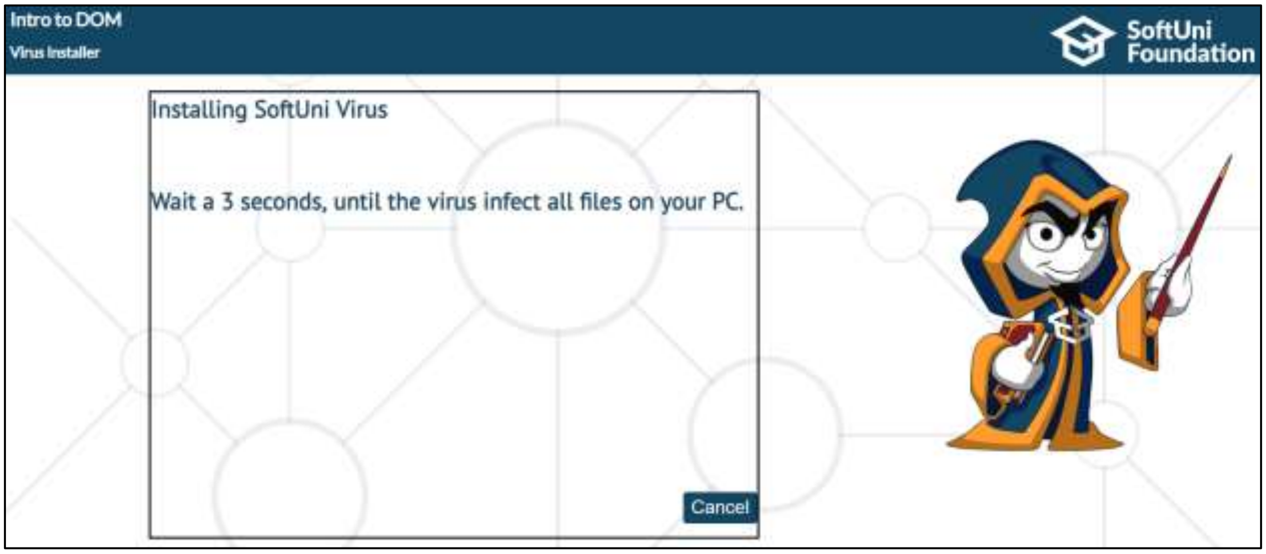
Expected result is:



License agreement

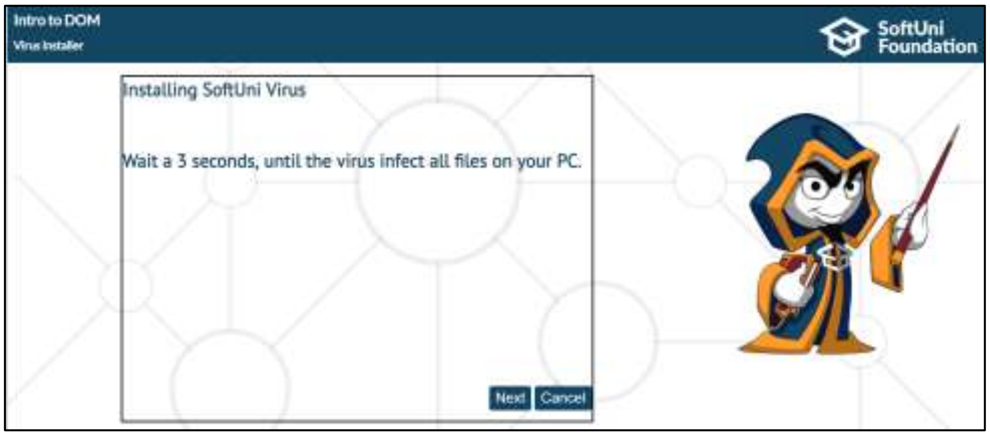
I Agree ●
I Disagree ○
By agreeing to our terms of use of the virus, you agree to give us access to all your personal data.Click -> I Agree to continue...

If we **try** to **click** on the "**Next button**", when "**I Disagree**" radio button is selected, **nothing should happen!** Because we **do not aggree** with the license agreement. Otherwise, when we "Agree" and we click on the "Next" button. **First step** should **hide** and next one should show (**secondStep**). On the second step, you need to use the **setTimeOut function** to continue. As you can see, the text inside it is "**Wait a 3 seconds….**" **Three seconds** after the **second step appears**, **the button should appear near the "Cancel button".**



Installing SoftUni Virus

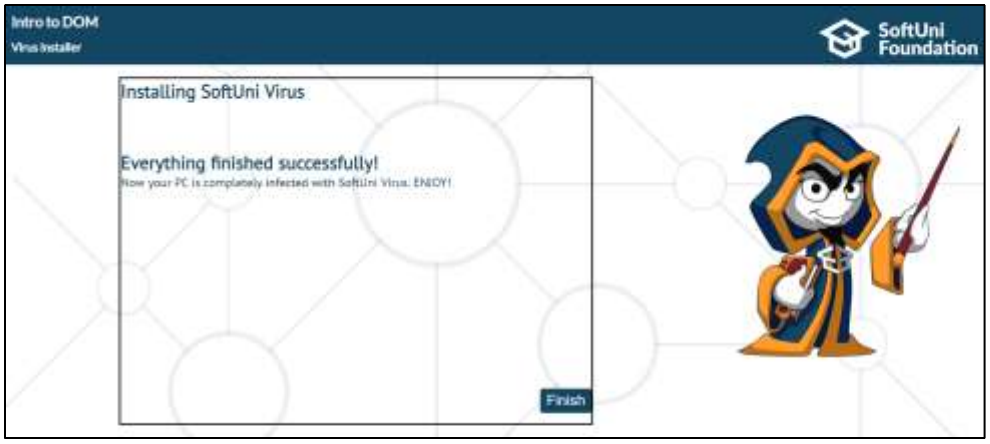
Wait a 3 seconds, until the virus infect all files on your PC.



Installing SoftUni Virus

Wait a 3 seconds, until the virus infect all files on your PC.

When we see the "**Next button**" and we click it, the last (thirdStep) stage in this "**installer**" needs to be visualized.



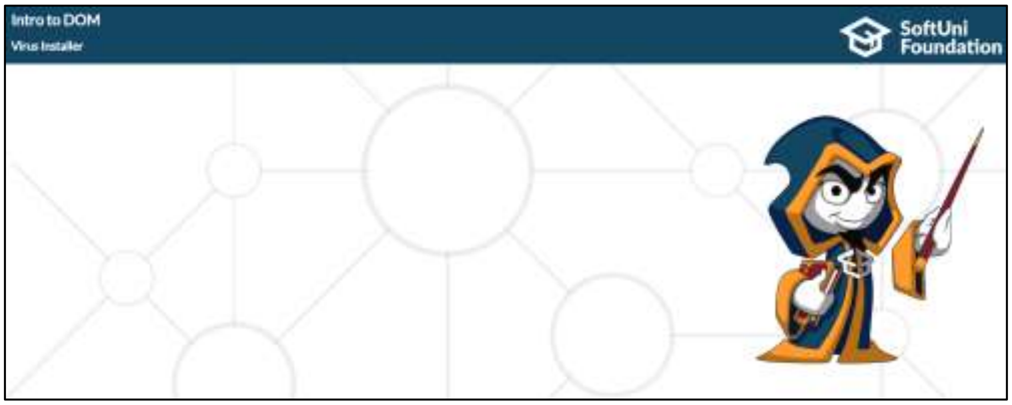
Installing SoftUni Virus

Everything finished successfully!

Now your PC is completely infected with SoftUni Virus. ENJOY!

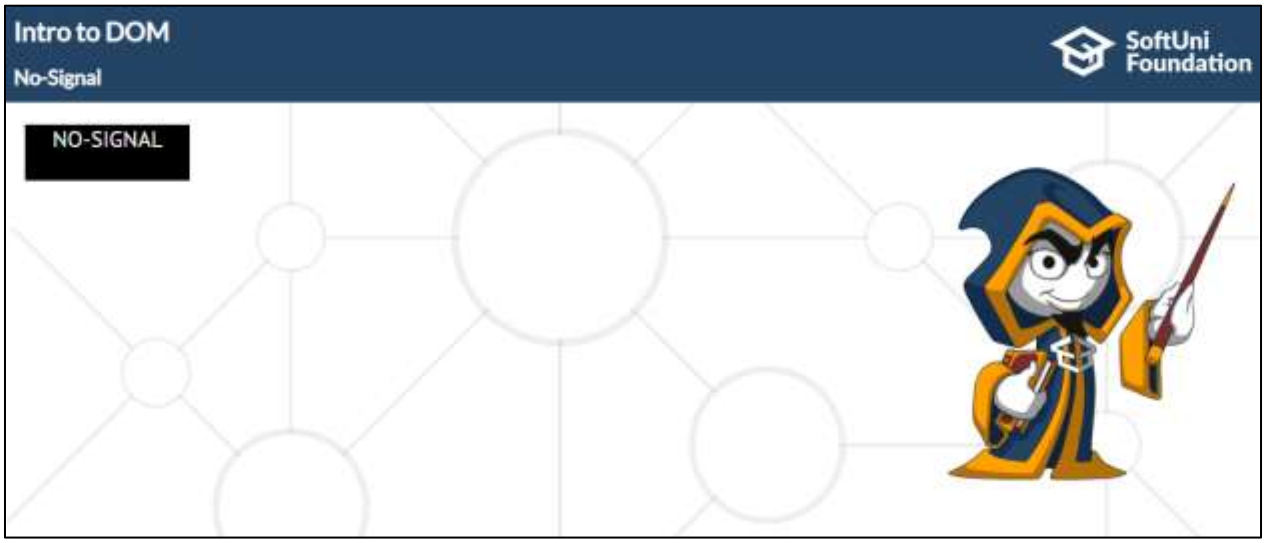
This is the **closing** stage of the "program"…

When we click on "**Finish**" or on **each** of the "**Cancel**" buttons, regardless of the step (firstStep, secondStep..), we should hide the whole **section** that contains (**content**, **h1**, **buttons**…) or simply hide **everything** apart. The expected result is:



10. No Signal

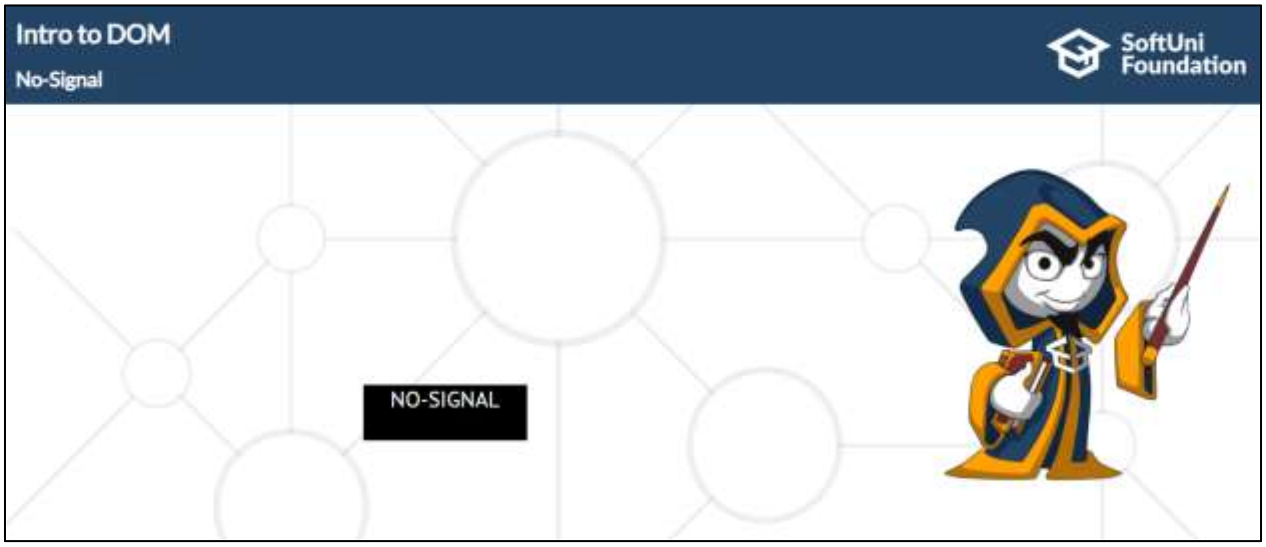
In this problem, you should **create a JS functonality** which puts a **div** on a **random position**.



You should work with the **position styles** on that div element. (**margin**)

The **horizontal range** must be **1-81** in **percent** (%).
The **vertical range** must be **1-45 vh** (viewport height).

Use **setTimeOut** and **Math.Round** to generate **random position** and **change it** every **2000 ms.**



Intro to DOM

No-Signal

NO-SIGNAL