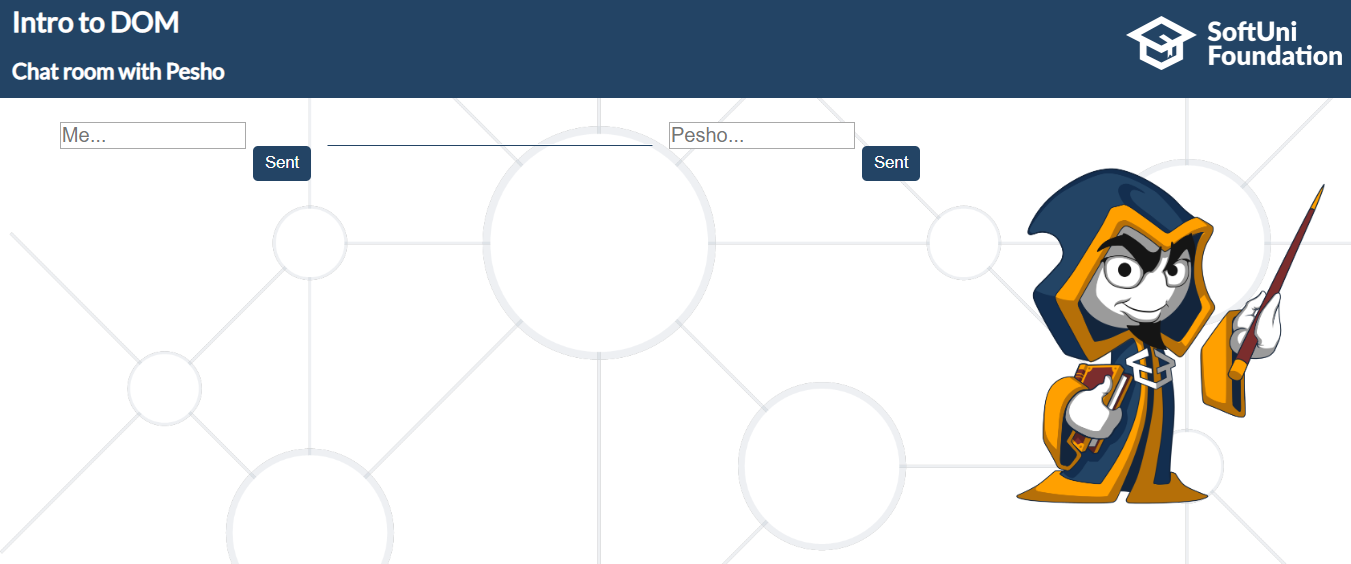
# Exercise: Intro to DOM

Problems for exercises and homework for the ["JavaScript Fundamentals" course @ SoftUni](https://softuni.bg/trainings/2247/js-fundamentals-january-2019)   
 Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1426>

## 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**).

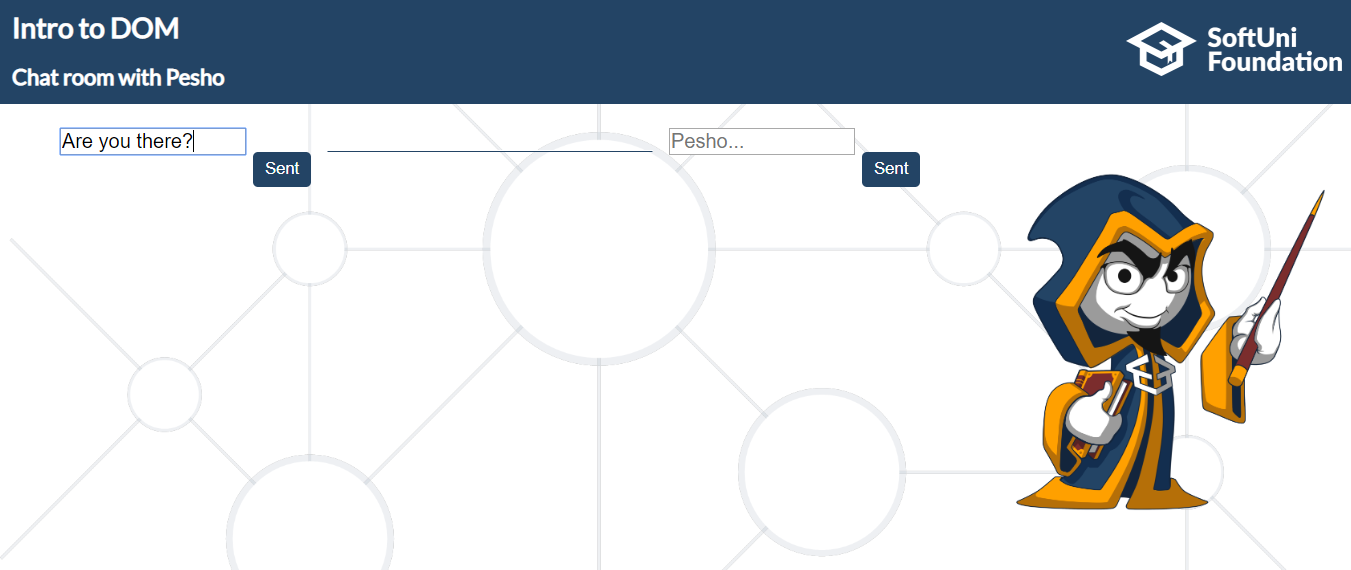
**The span element** should contain only the sender name (Me or Pesho).

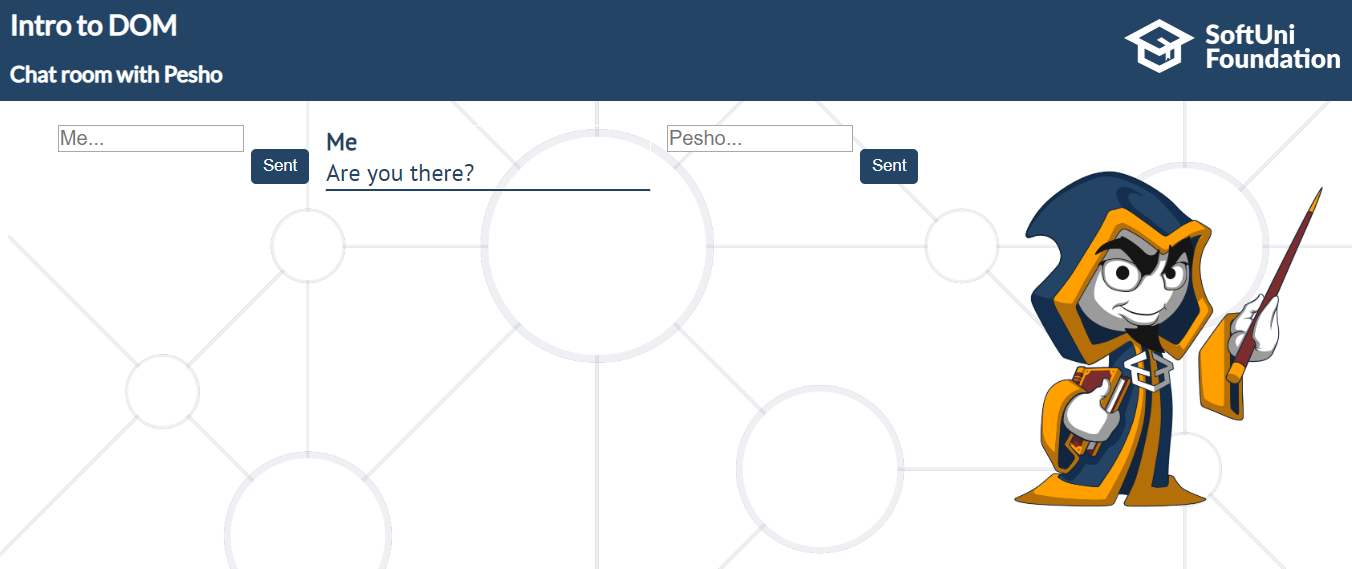
**The p element** should contain the current message.

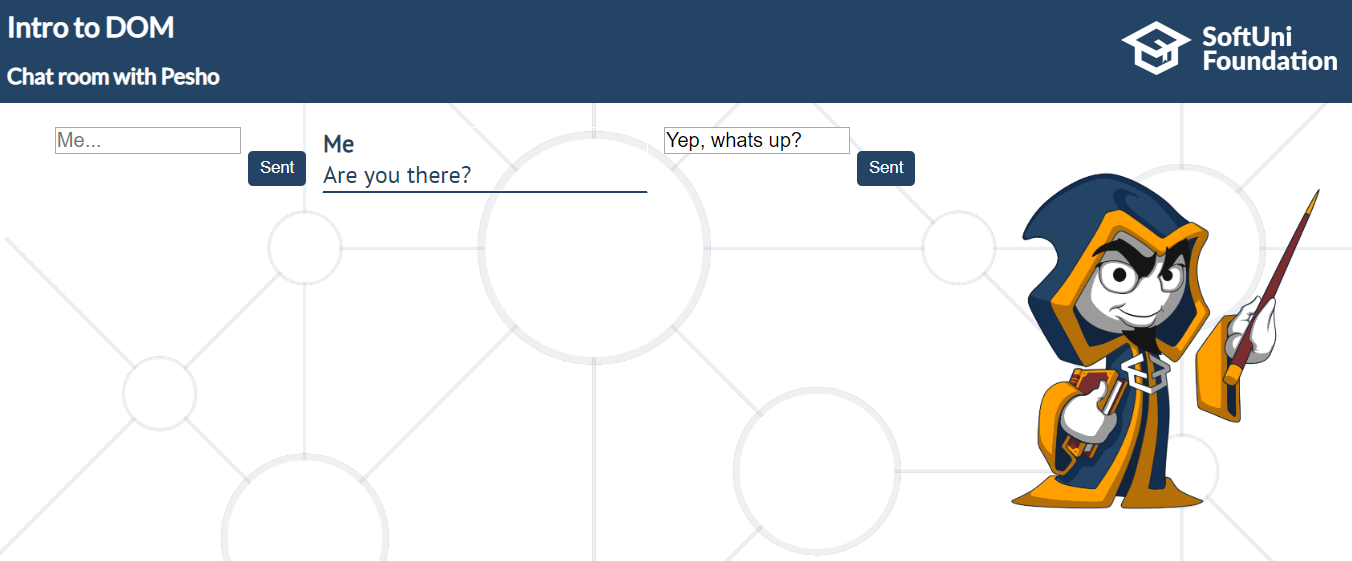
The final step is to append the **current div** to the div which has an **id** **chatChronology.**

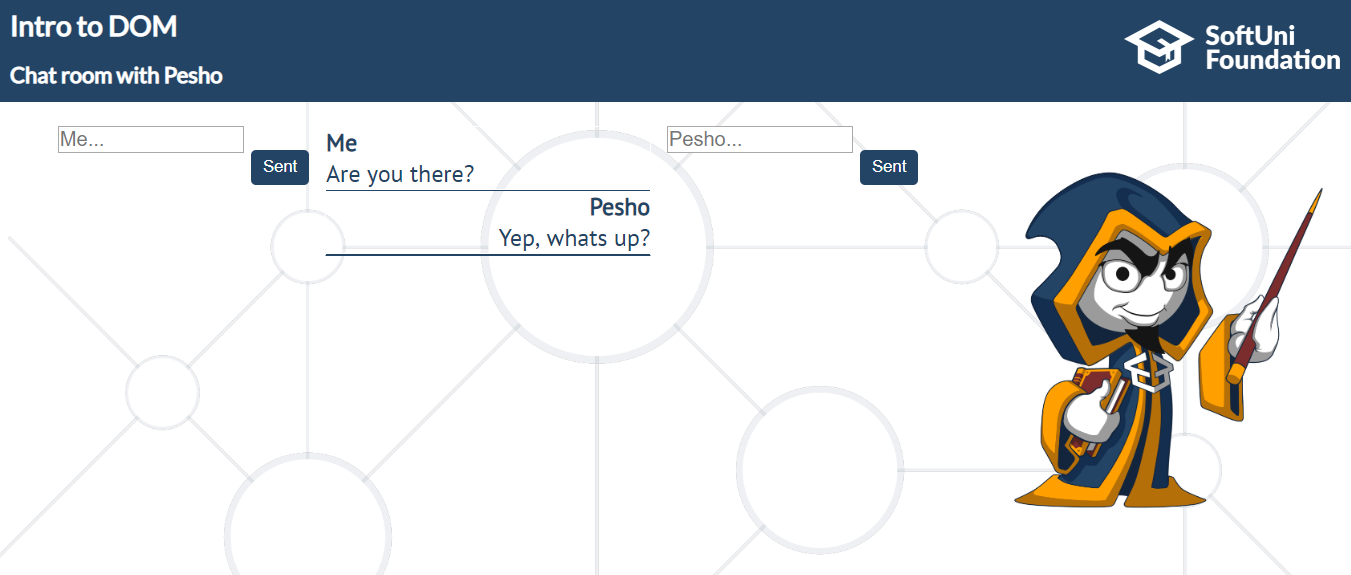
**Keep in mind that** if the sender is "Me", the **text-align** inside the current div should be **left**, otherwise when Pesho sends some message the **text-align** needs to be **right**.

Аfter each click on the buttons, **the current message should be cleared**.





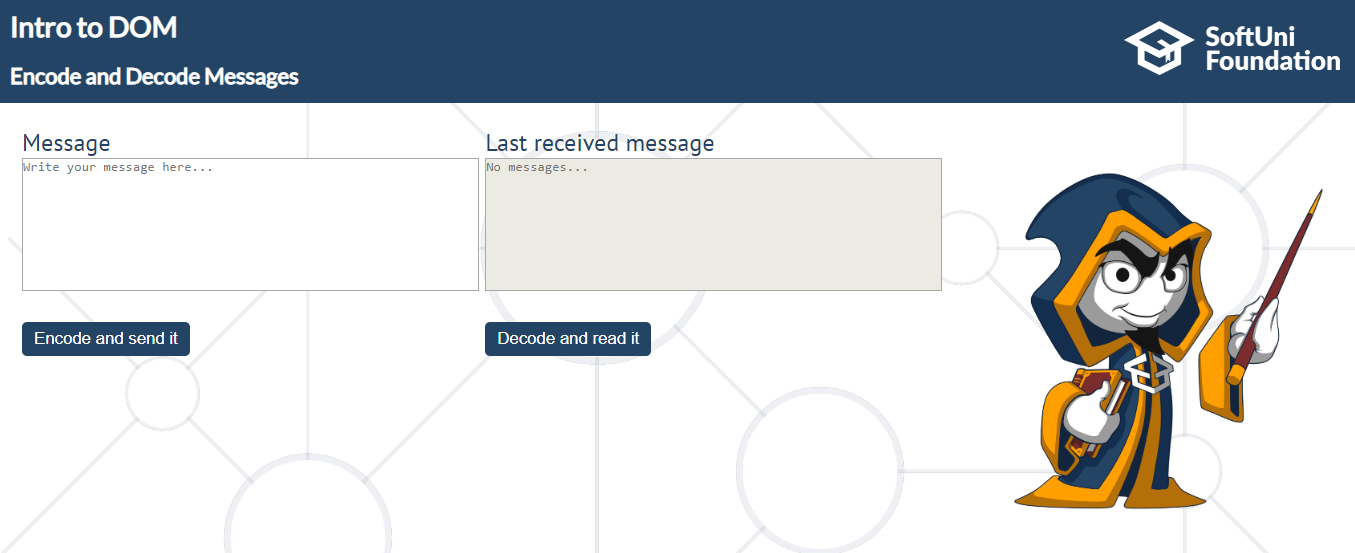




## Encode and Decode Messages

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**.

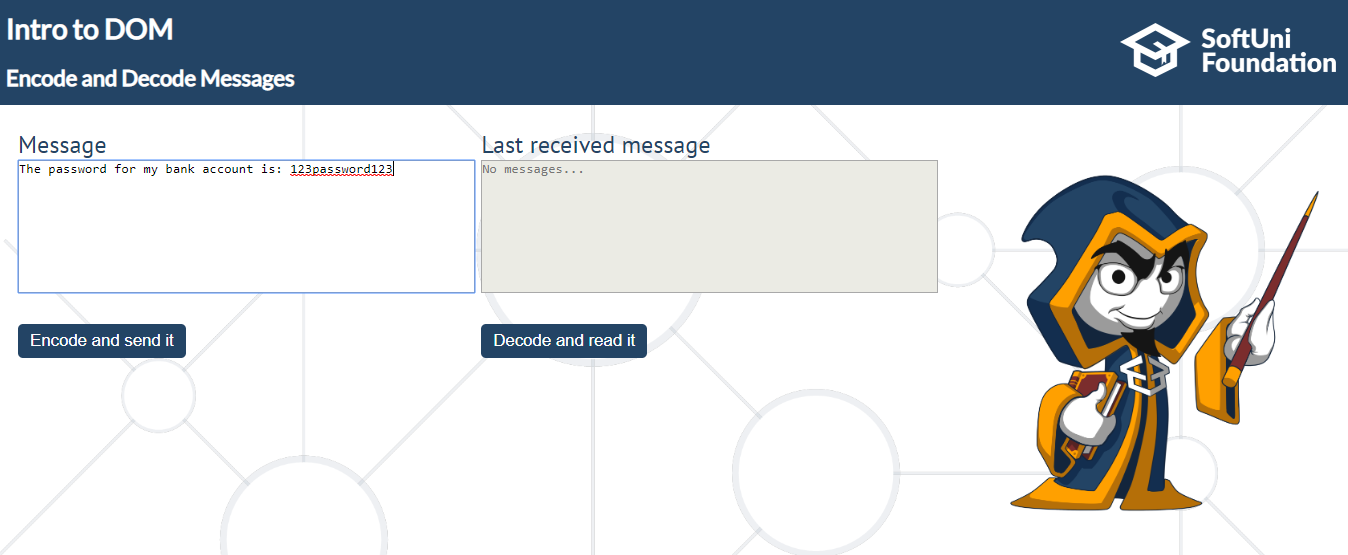
The first one is to **encode the given message** and **send it** to the **receiver**.

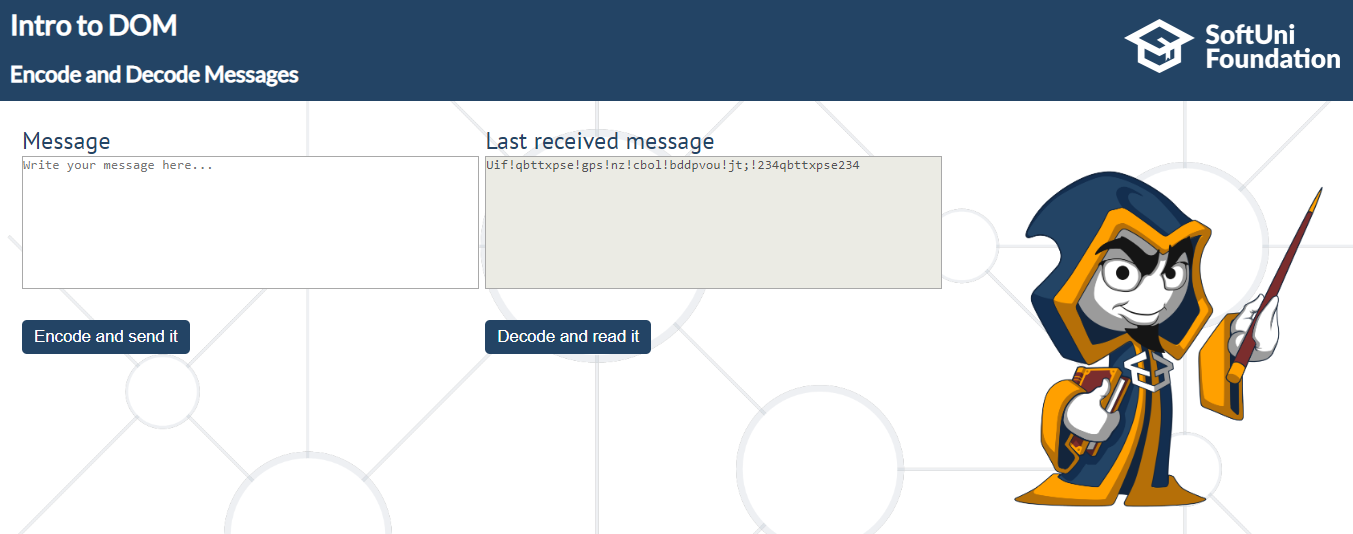
The second one is to **decode the received message** and **read it (display it)**.

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.

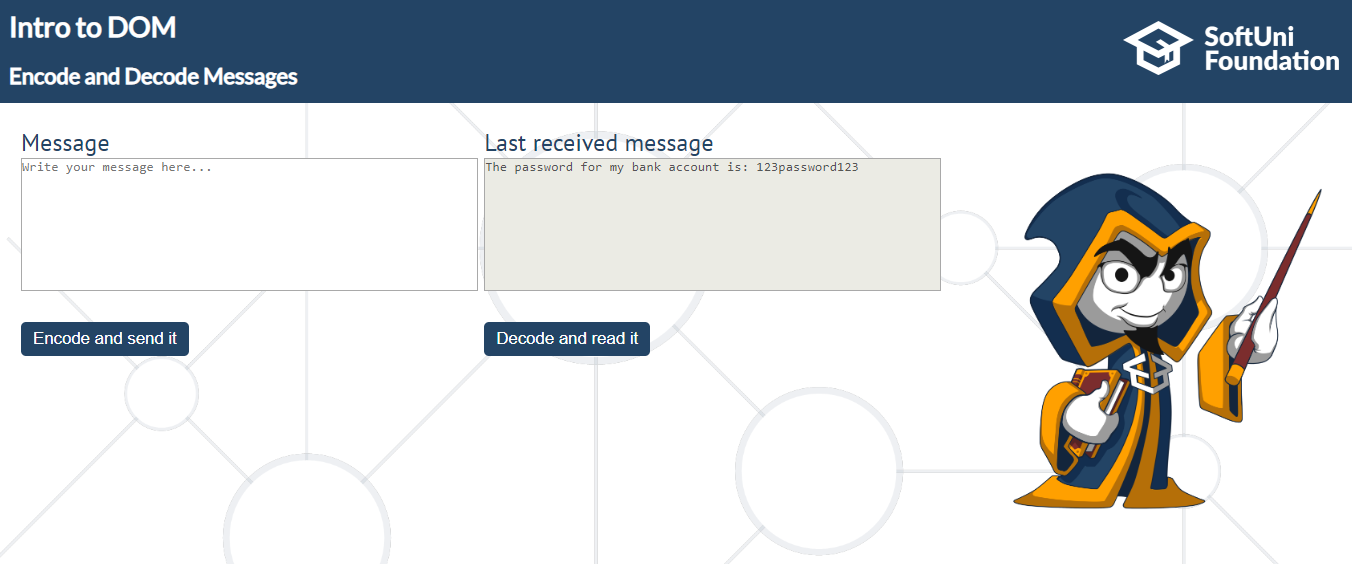
When you do that just **clear** the **sender textarea,** and **append the encoded message** to the **receiver textarea.**





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.

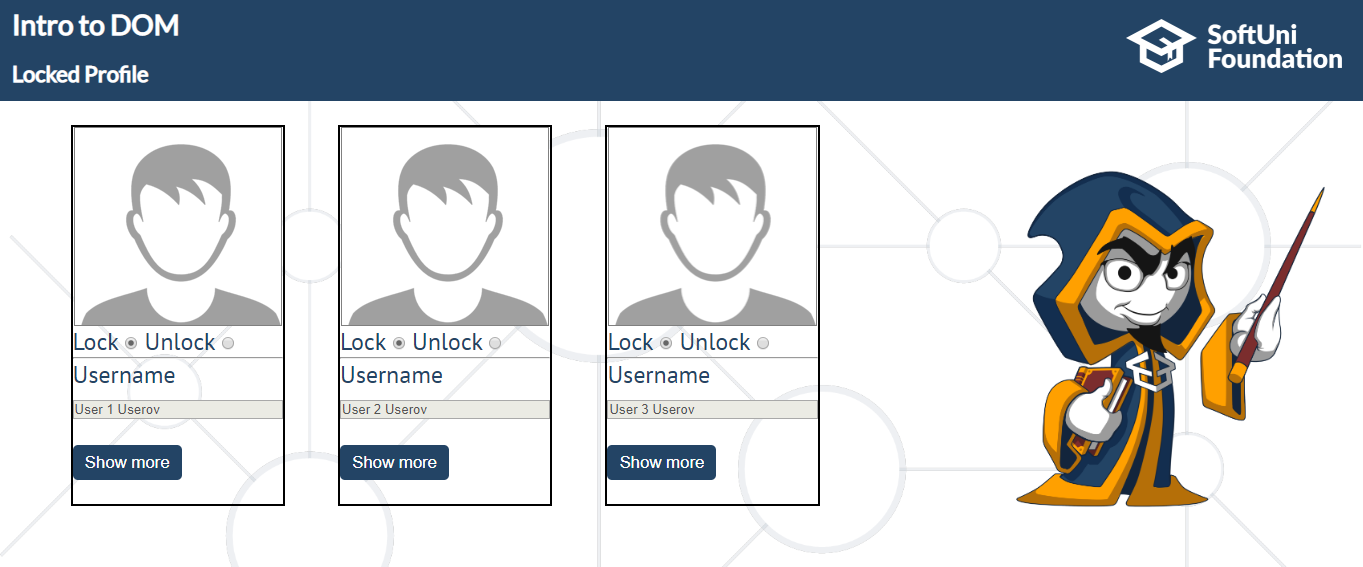
When you do that, just replace the **encoded message** with the already **decoded message** in the receiver textarea, to make it readable.



## Locked Profile

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

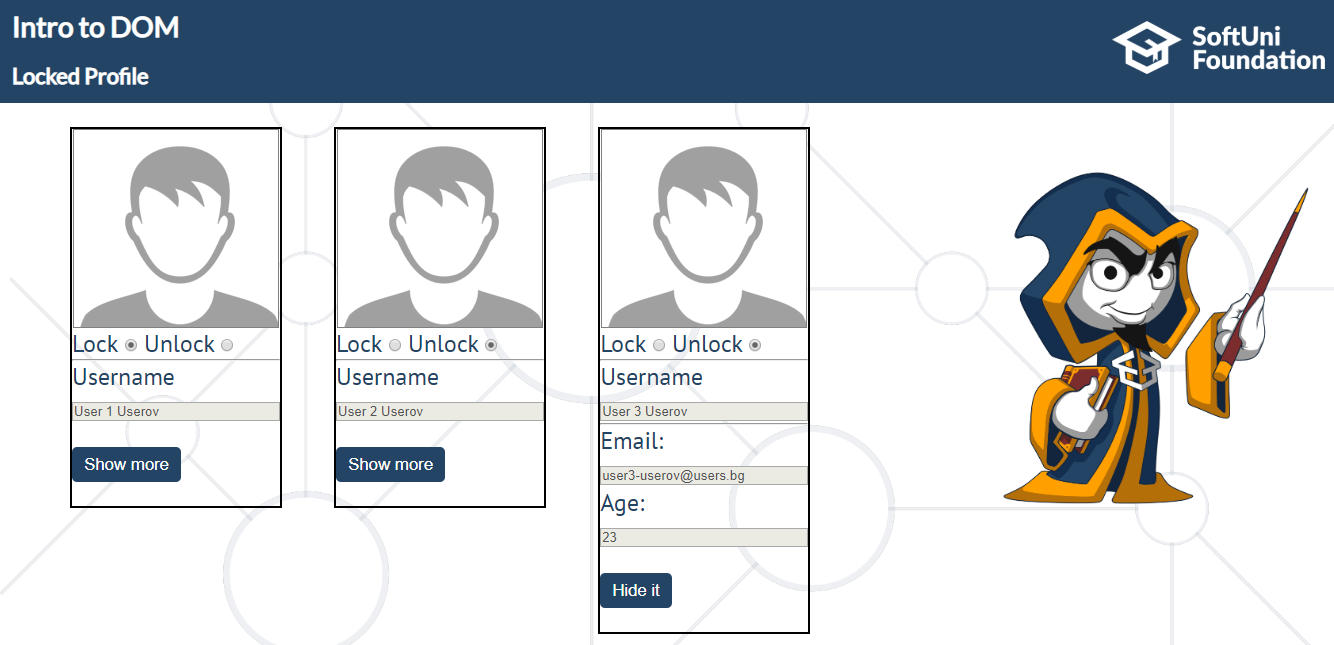
information about users.



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.

## Softuni Quiz

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



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

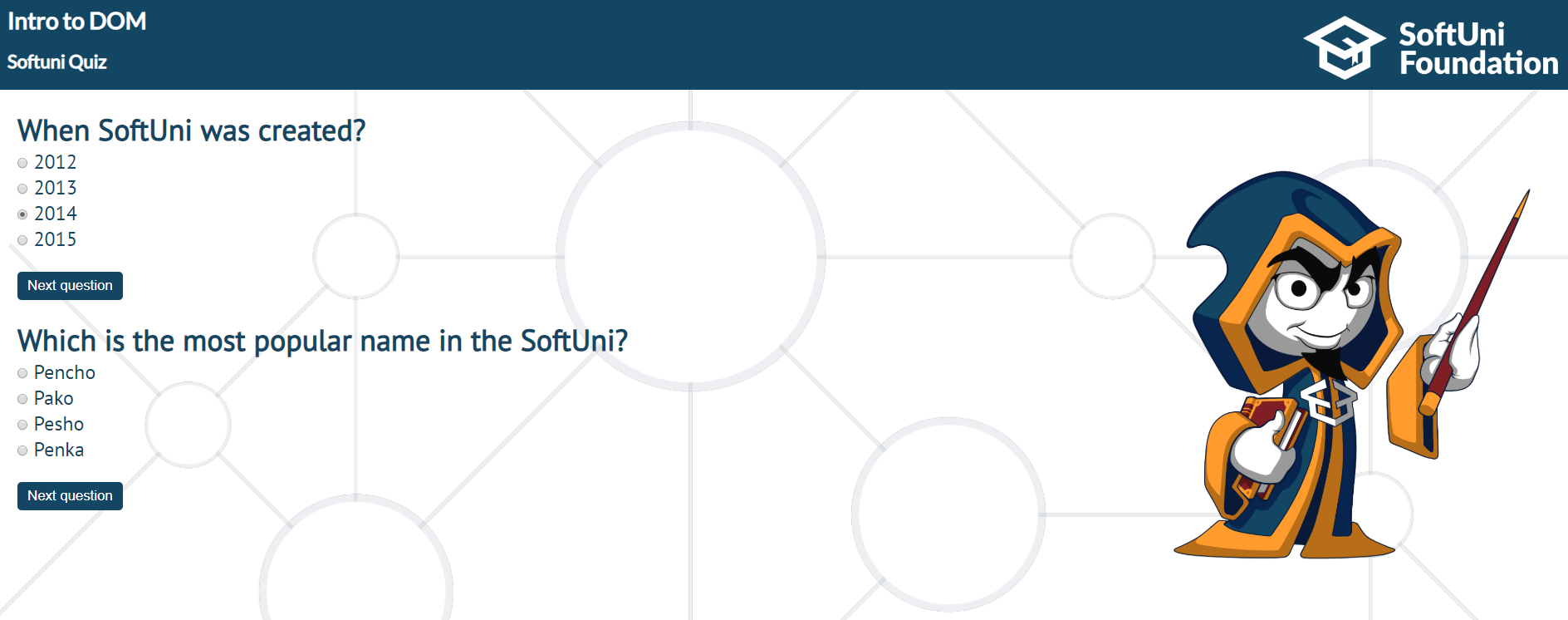
is visible, when you press the button, "**Get the results", the final score** must show into the

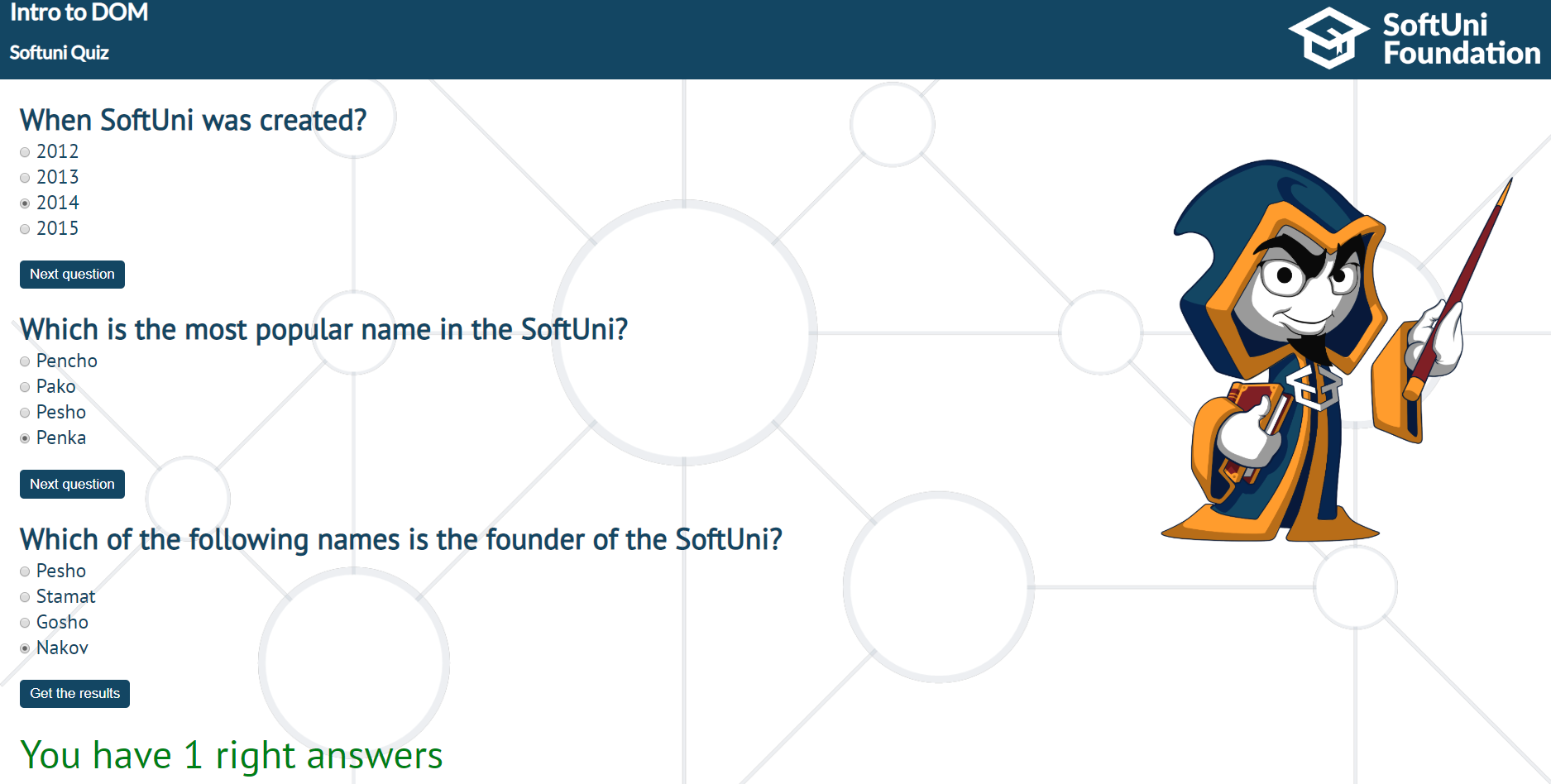
result div.

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**).

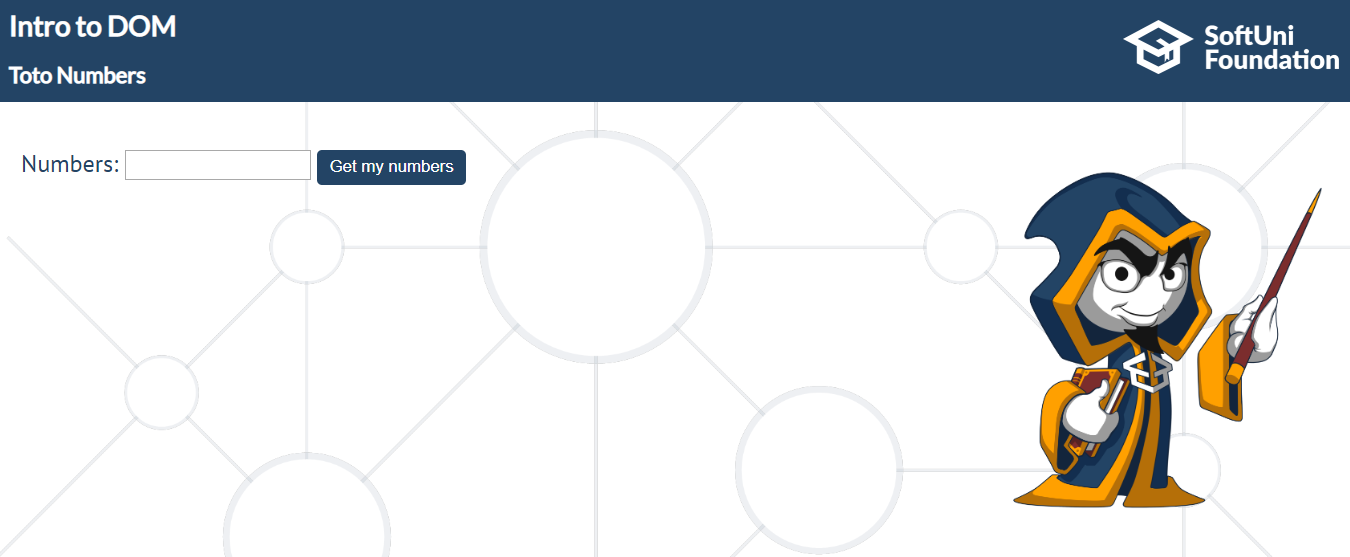






## Toto numbers

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].**

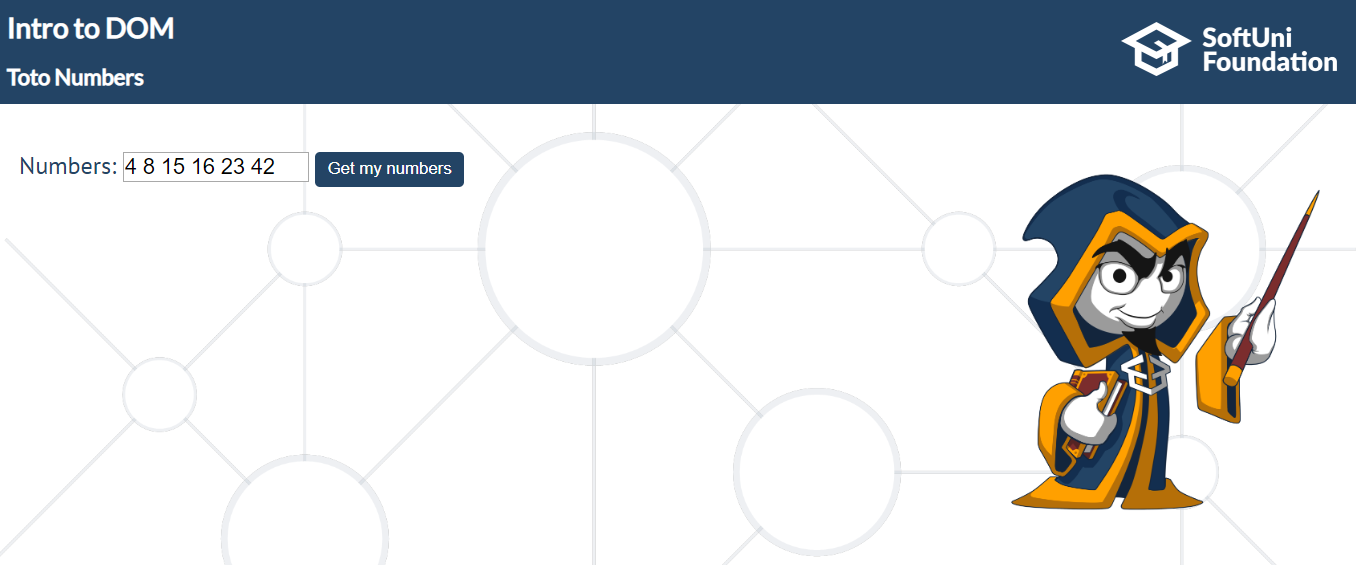
**If** numbers input contains **more or less** numbers **than 6** or **one of all** the given numbers **does not cover** **the necessary range**, **nothing should happen**.

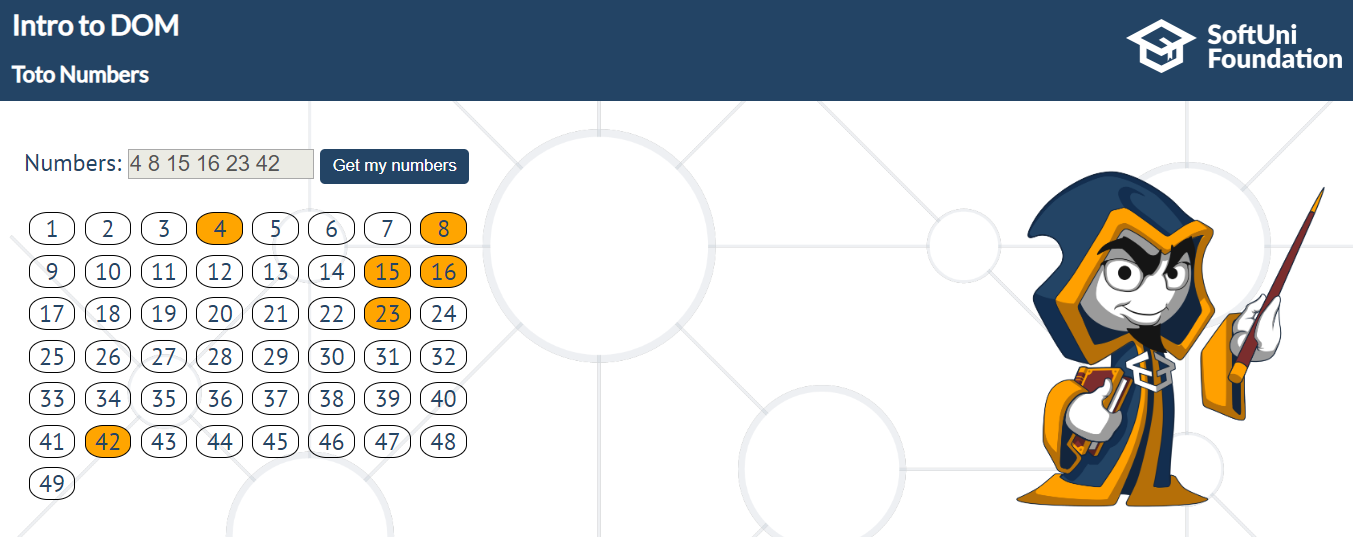
The expected functionality is as follows:

**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**".

**Еach** 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**.

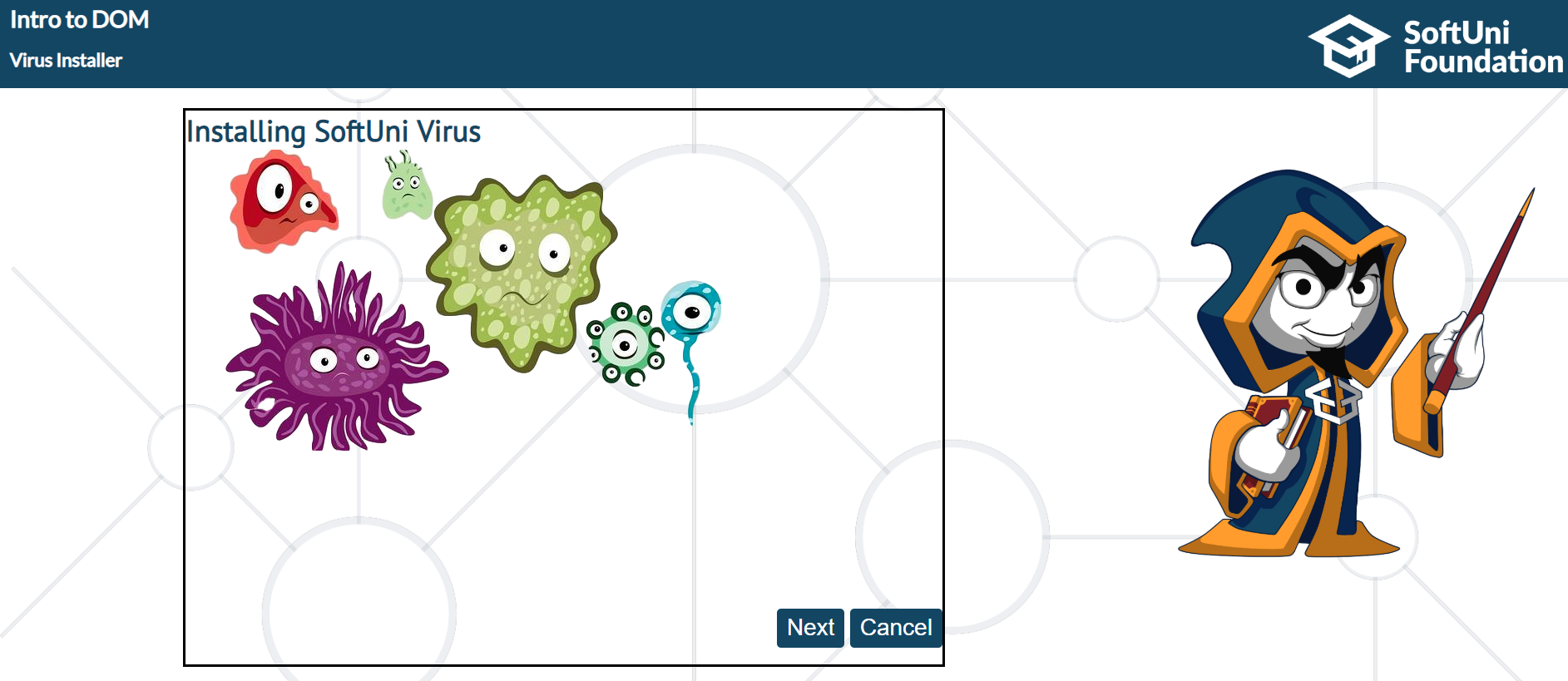


## Virus Installer

In this problem, you should **create a JS functonality** that represents a "Wizard" which installs a

"virus" on your computer.

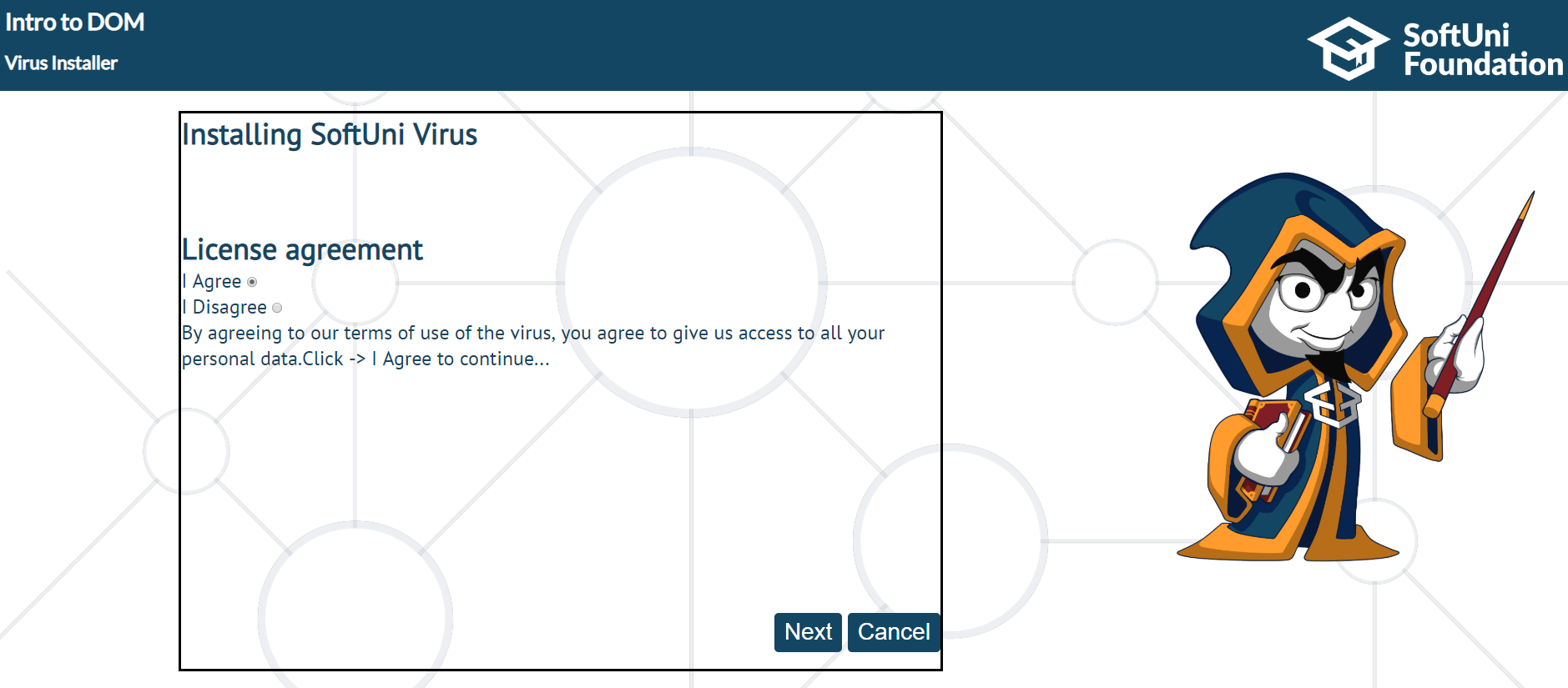
To have the virus installed successfully, you need to go through **three steps.**   
At the beginning (when you open the html file) you will see:



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:



If we **try** to **click** on the "**Next** **button**", when "**I Disagree**" radio button is selected, **nothing**

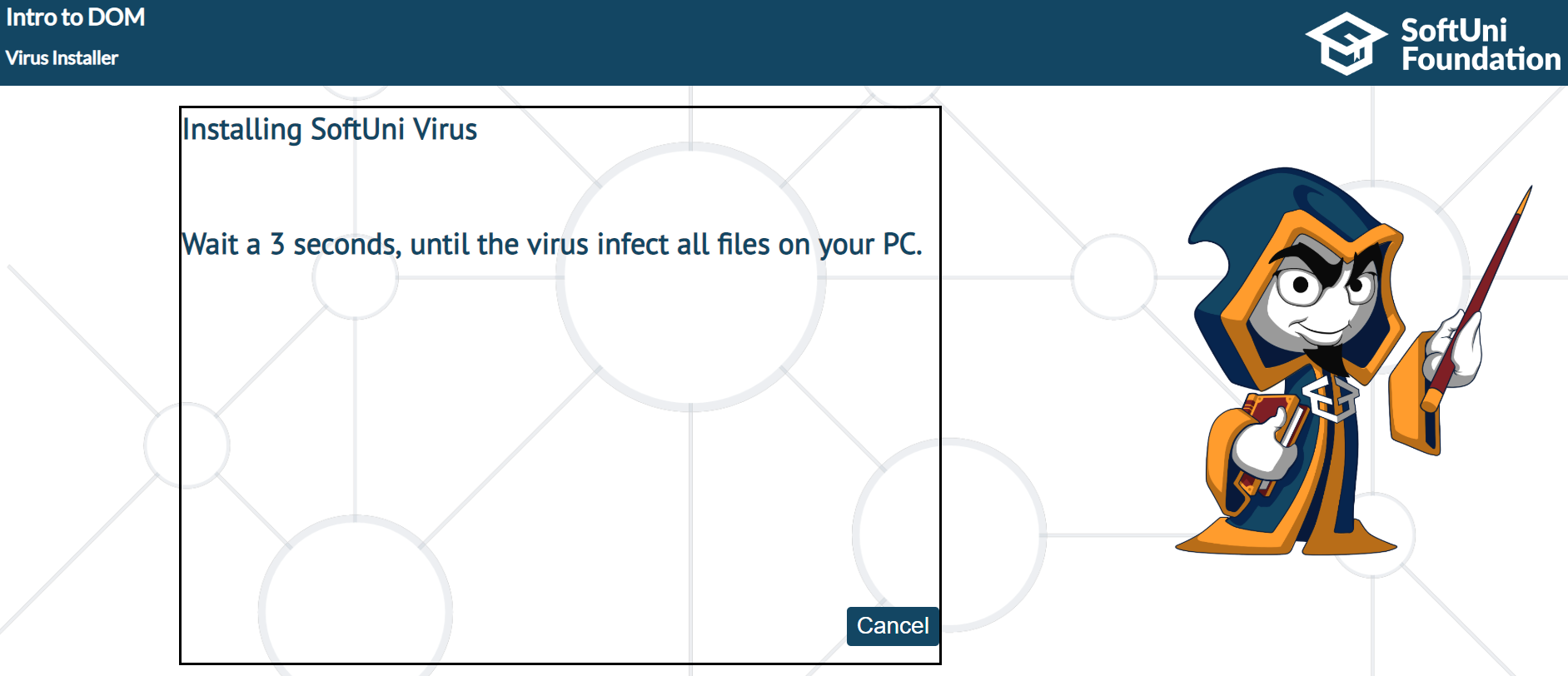
**should happen!** Because we **do not aggree** with the license agreement. Otherwise, when we

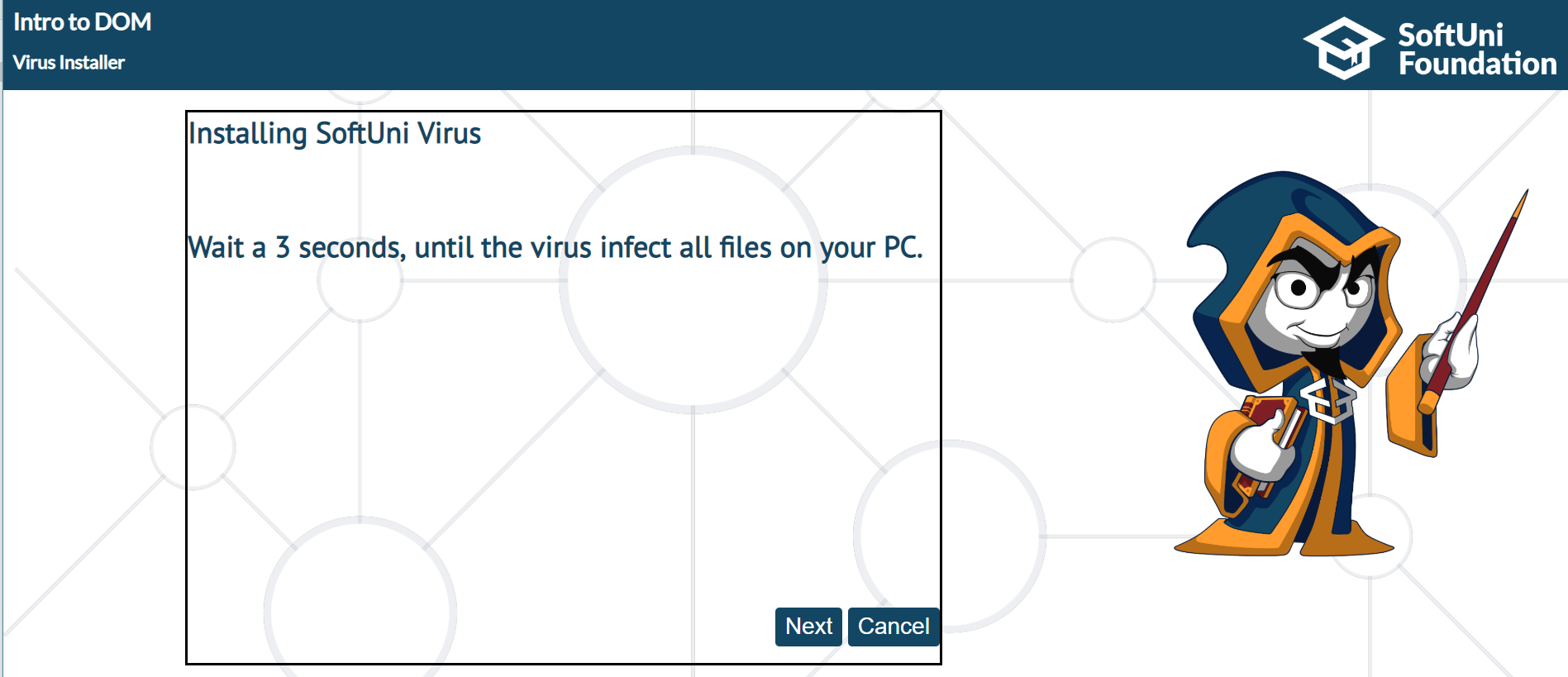
"Agree" and we click on the "Next" button. **First step** should **hide** and next one should show

(**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**,

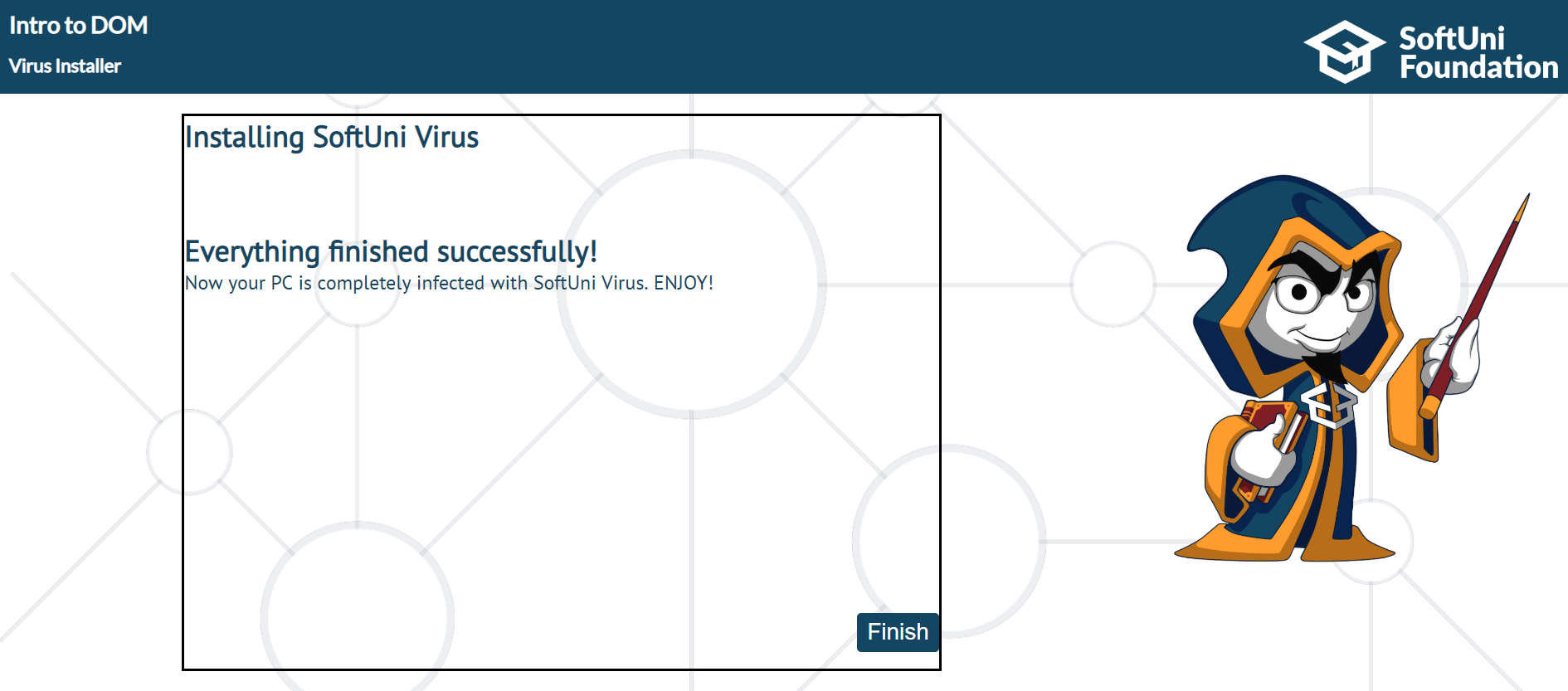
**the button should appear near the "Cancel button".**





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

needs to be visualized.



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:



## 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.**

