Exercise: jQuery

Problems for exercises and homework for the "JavaScript Advanced" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1548.

1. Increment Counter

You are tasked with creating a piece of **HTML** dynamically using JavaScript and **appending** it to a given element using a passed in **selector**.

HTML and JavaScript Code

You are given the following **HTML** and **CSS**.

incrementCounter.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Increment Counter</title>
    <script src="https://code.jquery.com/jquery-3.1.0.min.js"
            integrity="sha256-cCueBR6CsyA4/9szpPfrX3s49M9vUU5BgtiJj06wt/s="
            crossorigin="anonymous"></script>
</head>
<body>
    <div id="wrapper">
    </div>
    <script src="incrementCounter.js"></script>
</body>
</html>
```

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

incrementCounter.js

```javascript
function increment() {
    // TODO
}
```

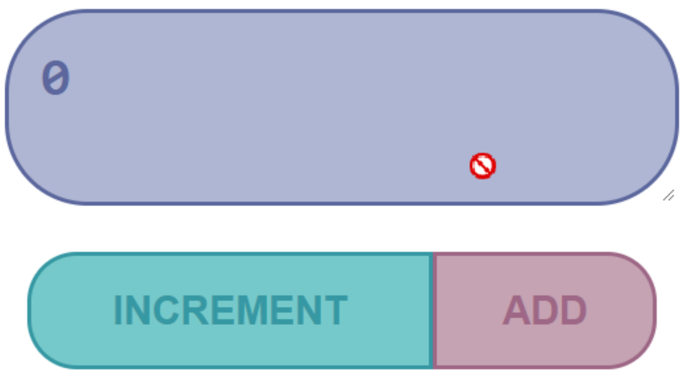
Your function will receive a **string** value representing a **selector** (for example "**#wrapper**" or "**.root**"), all elements created should be appended to the **selector**.

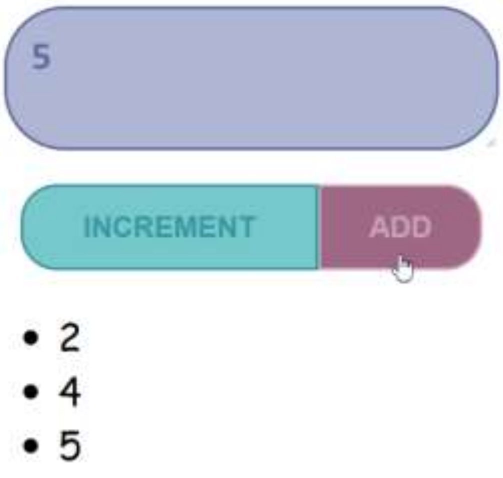
The HTML you create should contain 4 elements:

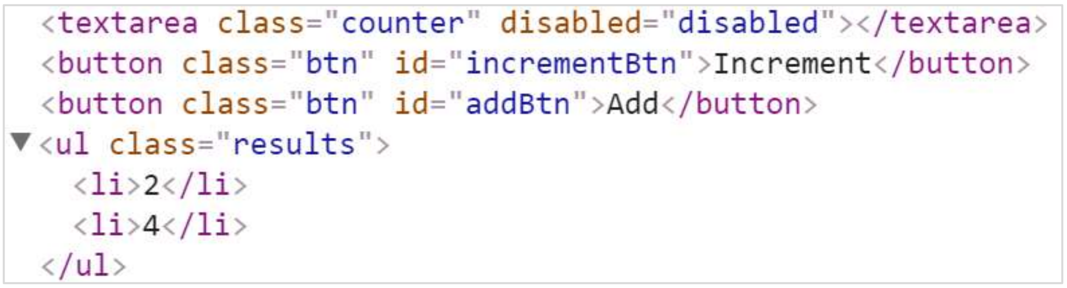
- **<textarea>** with **class="counter"**, **value="0"** and the **disabled** attribute.
- **<button>** with **class="btn"**, **id="incrementBtn"** and text "**Increment**".
- **<button>** with **class="btn"**, **id="addBtn"** and text "**Add**".
- Unordered list **<ul>** with **class="results"**.

When the **[Increment]** is clicked the value of the **textarea** should go up by **one** (if it was 0 it should become 1 e.t.c.). When the **[Add]** is clicked a new list item (**<li>**) with text equal to the current value of the textarea should be added to the unordered list.

Screenshots



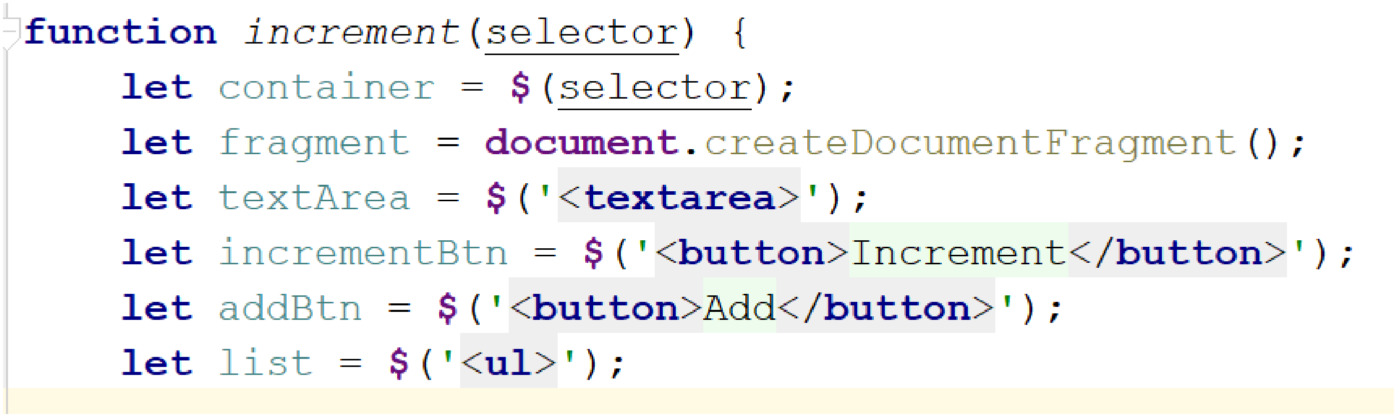




```
<textarea class="counter" disabled="disabled"></textarea>
<button class="btn" id="incrementBtn">Increment</button>
<button class="btn" id="addBtn">Add</button>
▼<ul class="results">
   <li>2</li>
   <li>4</li>
</ul>
```

Hints

We'll start off by creating the needed elements and parsing the **selector**, we can do it easily with **jQuery** like this:



```
function increment(selector) {
    let container = $(selector);
    let fragment = document.createDocumentFragment();
    let textArea = $('<textarea>');
    let incrementBtn = $('<button>Increment</button>');
    let addBtn = $('<button>Add</button>');
    let list = $('<ul>');
```

Adding multiple elements to the DOM can be expensive, instead of repeatedly adding to the DOM we can create a **`DocumentFragment`** and **add** the elements to it instead. When we have built our hierarchy we can **append the `DocumentFragment`** to the DOM, which will add all of the fragment's elements to the specified selector.

The next step is to **add values**, and **attributes** to the **elements** and **events** to the **buttons**:

```
// Textarea formation
textArea.val(0);
textArea.addClass('counter');
textArea.attr('disabled', true);

// Buttons formation
incrementBtn.addClass('btn');
incrementBtn.attr('id', 'incrementBtn');
addBtn.addClass('btn');
addBtn.attr('id', 'addBtn');

// List formation
list.addClass('results');

// Events
$(incrementBtn).on("click", function () {
    textArea.val(+textArea.val() + 1)
});
$(addBtn).on("click", function () {
   let li = $(`<li>${textArea.val()}</li>`);
   li.appendTo(list);
});
```

The last step is to **add** our elements to the DOM:

```
textArea.appendTo(fragment);
incrementBtn.appendTo(fragment);
addBtn.appendTo(fragment);
list.appendTo(fragment);

container.append(fragment);
```

Our code is now ready.

2. Timer

You will be given an **HTML** file, containing the markup of a **timer** with spans for **seconds**, **minutes** and **hours** and buttons to `[Start]` and `[Pause]` the timer. Your task is to create a JavaScript application that **starts** the timer whenever the `[Start]` button is pressed and **pauses** it when the `[Pause]` button is pressed.

HTML and JavaScript Code

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

timer.html

```
<body>
<div id="timer">
    <span id="hours" class="timer">00</span>:
    <span id="minutes" class="timer">00</span>:
    <span id="seconds" class="timer">00</span>
    <button id="start-timer">Start</button>
    <button id="stop-timer">Stop</button>
</div>
<script src="timer.js"></script>
<script>
    window.onload=function(){
        timer();
    }
</script>
</body>
```

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

timer.js

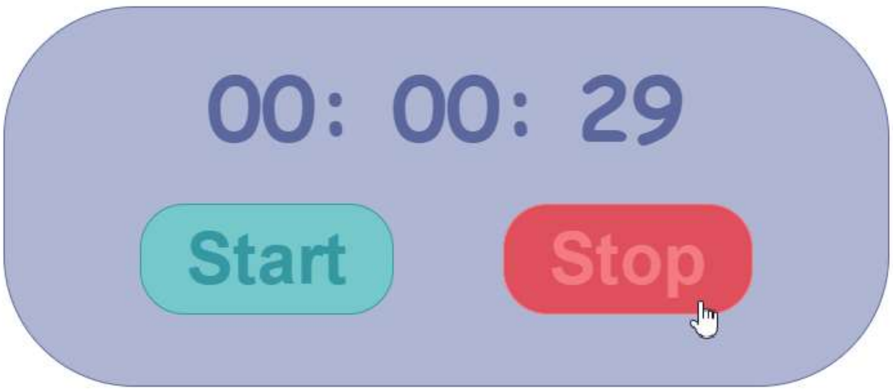
```
function timer() {
    // TODO
}
```

Submit in the judge the **JS** code (implementation) of the above function. It may hold other functions in its body.

Constraints

- The initial value of the timer must always be **00:00:00**

Screenshots



Hints

Note the spans have unique **id** values – we can use these to select and modify the elements with **jQuery**.

```
<div id="timer">
    <span id="hours" class="timer">00</span>:
    <span id="minutes" class="timer">00</span>:
    <span id="seconds" class="timer">00</span>
    <button id="start-timer">Start</button>
    <button id="stop-timer">Stop</button>
</div>
```

JavaScript has a built-in function **`setInterval()`** for executing and repeating an action after a set period of time. It returns an object which can later be used to stop the execution with **`clearInterval()`**.

```
timer = setInterval(step, 1000);

clearInterval(timer);

function step() {
// TODO
}
```

The **first argument** can be an inline declaration or a **named function**. The **second argument** is the **time interval**, specified in **milliseconds**. We can easily attach these two functions to the click event of a button.

To get and set the text of a markup element you can either use its **`textContent`** property, or jQuery's **`text()`** function.

Keep in mind that that you should only have one **`setInterval()`** function active when the the timer is working, multiple presses of the **`[Start]`** button should not attach more **`setInterval()`** functions as that would break the correct operation of the timer.

3. Form Validation

You are given the task to write **validation** for the fields of a simple form.

HTML and JavaScript Code

You are provided a **skeleton** containing the necessary files for your program.

The validations should be as follows:

- The **`username`** needs to be between **3** and **20** symbols **inclusively** and only **letters** and **numbers** are allowed.
- The **`password`** and **`confirm-password`** must be between **5** and **15 inclusively** symbols
- The **inputs** of the **`password`** and **`confirm-password`** field **must match**.
- The **`email`** field must contain the "**@**" symbol and **at least one** "**.**"(**dot**) after it.

If the "**Is company?**" checkbox is **checked**, the **CompanyInfo** fieldset should become **visible** and the **Company Number** field must also be **validated**, if it isn't checked the **Company** fieldset should have the style **"display: none;"** and the **value** of the **Company Number** field shouldn't matter.

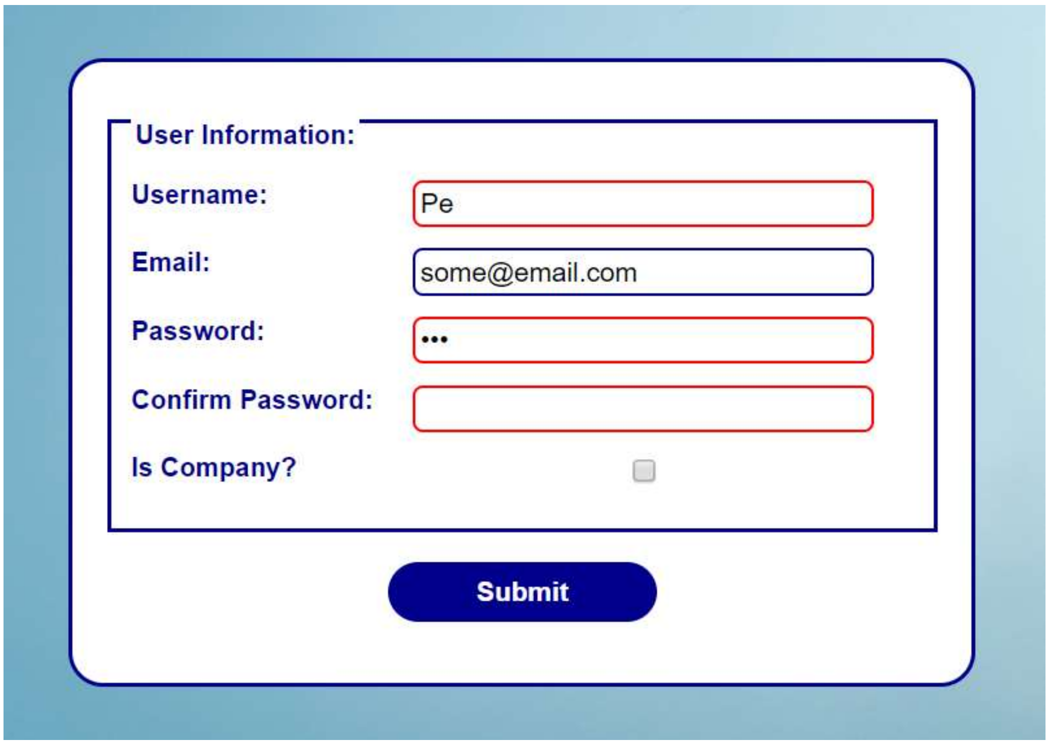
- The **Company Number** field must be a number between **1000** and **9999**.

Every field with an **incorrect** value when the **[Submit]** button is **pressed** should have the following style applied **border-color: red;**, alternatively if it's correct it should have style **border: none;**. If there are **required fields** with an incorrect value when the **[Submit]** button is pressed, the **div** with **id="valid"** should become **hidden** ("**display: none;**"), **alternatively** if all fields are correct the **div** should become **visible**.

Constraints

- **You are NOT allowed to change the HTML or CSS files provided.**

Screenshots



User Information:

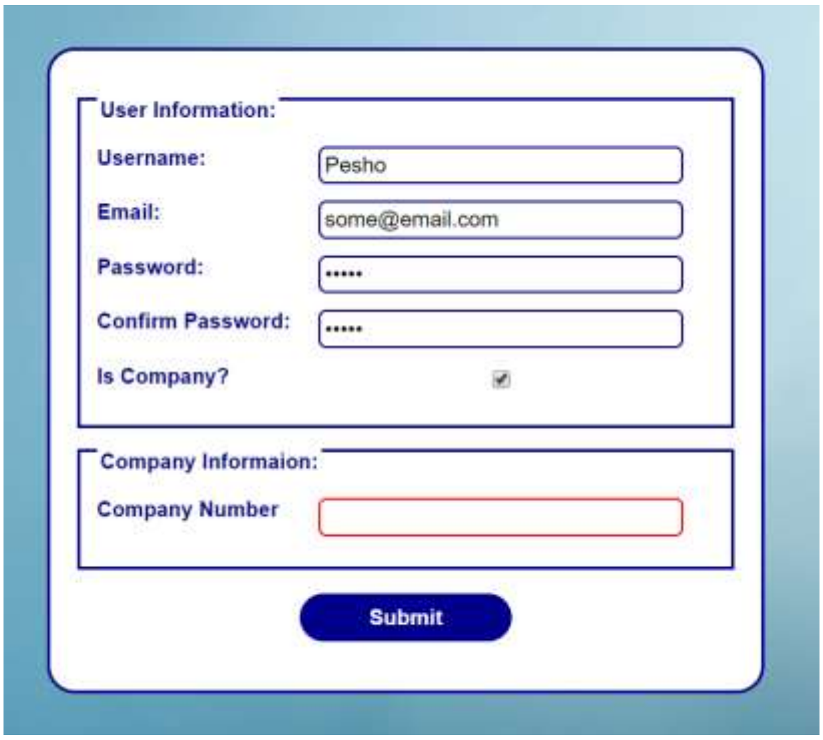
Username: Pe

Email: some@email.com

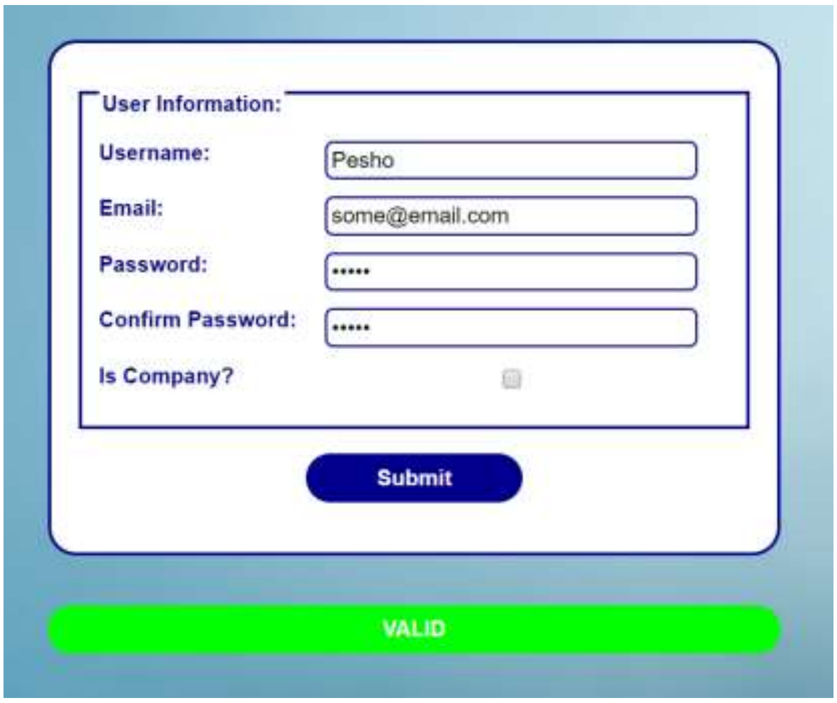
Password: •••

Confirm Password:

Is Company? ☐



User Information:	
Username:	Pesho
Email:	some@email.com
Password:	•••••
Confirm Password:	•••••
Is Company?	✓
Company Informaion:	
Company Number	



User Information:	
Username:	Pesho
Email:	some@email.com
Password:	•••••
Confirm Password:	•••••
Is Company?	☐

Hints

- Use **`addEventListener()`** or jQuery's **`on()`** function to **attach** an **event listener** for the "**`change`**" event to the **checkbox**.
- All buttons within a **`<form>`** automatically work as **submit** buttons, unless their type is **manually assigned** to something else, in order to avoid **reloading the page** upon **clicking** the **`[Submit]`** button you can add the following code in the function that handles the on click event:

```
submit.on('click', function(ev) {
    ev.preventDefault();
```

- The validation for the separate fields can be done using **regex**.