# Exercises: jQuery, DOM and Events

Problems for exercises and homework for the [“JavaScript for Front-End” course @ SoftUni](https://softuni.bg/trainings/1795/javascript-for-front-end-october-2017).

## 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** code:

|  |
| --- |
| incrementCounter.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 |
| **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





### Hints

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



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**:



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



Our code is now ready.

## 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 |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Timer</**title**>  <**script src="https://code.jquery.com/jquery-3.1.0.min.js"  integrity="sha256-cCueBR6CsyA4/9szpPfrX3s49M9vUU5BgtiJj06wt/s="  crossorigin="anonymous"**></**script**>  <**style**>  **#timer** {  **font-size**: 5**em**;  }  </**style**> </**head**> <**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**> </**html**> |

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**

### Hints

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



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().



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.