# 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.

## Book Generator

Create a function that accepts a selector, a title, an author and an ISBN and **uses** them to **create** the **HTML code** for a **book** and **inserts it** into the **selector**.

### HTML and JavaScript Code

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

|  |
| --- |
| book-generator.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Book Generator</**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="book-generator.js"**></**script**> <**script**>  window.onload = **function** () {  *createBook*(**"#wrapper"**, **"Alice in Wonderland"**, **"Lewis Carroll"**, 1111);  } </**script**> </**body**> </**html**> |

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

|  |
| --- |
| book-generator.js |
| **function** *createBook*() {  *//* ***TODO*** } |

Your function will receive **4 parameters** - a **string value** representing a selector (for example "#wrapper" or ".root"), a **string value** representing the titleof the book, a **string value** representing the authorof the book and a **number** representing the ISBN of the book. **After** the book is **created** it should be **attached** to the passed in **selector**.

The **number** in the Id of the containing div should be **incremented** by **one** **for each successive book created** (i.e. first book should have id = "book1", second id = "book2" and so on…). The title, author and ISBN should be **paragraphs** with a class equal to their **respective role** - class="title" for the **title paragraph**, class="author" for the **author paragraph** and class="isbn" for the **ISBN paragraph**. A book should also contain **2 buttons** – [Select] and [Deselect], when the [Select] button is pressed the border of the div element should be set to "2px solid blue". When the [Deselect] button is pressed it should be set to "none".

### Screenshots

The HTML code for a book should have the following structure:



Selected book:





### Hints

You can use what is known as an **IIFE** (Immediately Invoked Function Expression) to declare and instantly execute a function that will keep the id variable in its scope. This way you will receive the inner function and for it the variable id will be **shared between all calls**, essentially becoming like a **static variable** for the function:



An **element’s** **css properties** can easily be changed with **jQuery** in the following way:



## DOM Dynamic Form

Write a JS function that **generates a form** for managing a list of items and inserts it in an HTML document by given **selector** (e.g. by div id).

### HTML and JavaScript Code

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

|  |
| --- |
| dom-dynamic-form.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>DOM Dynamic Form</**title**>  <**script src="https://code.jquery.com/jquery-3.1.0.min.js"  integrity="sha256-cCueBR6CsyA4/9szpPfrX3s49M9vUU5BgtiJj06wt/s="  crossorigin="anonymous"**></**script**>  <**style**>  **input** {  **position**: **absolute**;  **left**:6**em**;  }  .**button** {  **background-color**: **darkgrey**;  **color**: **white**;  **font-weight**: **bold**;  **position**: **absolute**;  **left**: 15**em**;  **border**: 1**px solid black**;  **padding**: 0 5**px** 0 5**px**;  }  .**result-controls** .**button** {  **position**: **relative**;  **left**: 0;  **font-size**: 50%;  **margin-right**:1**em**;  **padding**: 0;  **bottom**: 3**px**;  }  **li** {  **list-style-type**: **none**;  }  </**style**> </**head**> <**body**> <**div id="content"**></**div**> <**script src="dom-dynamic-form.js"**></**script**> <**script**>  *domDynamicForm*(**"#content"**); </**script**> </**body**> </**html**> |

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

|  |
| --- |
| dom-dynamic-form.js |
| **function** *domDynamicForm*() {  *//* ***TODO*** } |

Your function will receive **two arguments** – the **first** is a **selector** to an HTML element.

The user must be able to:

* Add a new item with specified string content.
* Delete an existing element.

Each of the controls must be in a separate div. Place the add controls inside a div with a class add-controls, and provide a label with the text “Enter text:”, an **empty input field** and an anchor with the class button applied to it. The anchor acts as an **add button** and when the user presses it, a **new item must be added to the list**, using the string in the **input field** as its **name**.

Place the result controls inside a div with a class result-controls and in a list of class items-list. Each element has the class list-item.

Every element is composed of an anchor with inner text "X" functioning as a **delete button** and its name wrapped in a <strong> tag. The anchor has the class button. When the button is clicked, the element is **deleted** from the list.

### Screenshots

### Hints

Take a look at the following screenshot – it contains the HTML hierarchy for the form you see in the previous images. Note the last three elements in the list are set to style display:none;, which will hide them from view.

