Creating Your First "To-Do" Item

This lesson explains how to add to-do list items to the Cloud Firestore database.

WE'LL COVER THE FOLLOWING

- ^
- Enable Cloud Firestore
- To-do List Item Form
 - Styling the form button
- Container For to-do Items
- Real-time Listener for to-do Items
- Generate to-do Items
- Access to-do Item Add Form
- Submit Event and Database Entry
- To-Do List Application

Enable Cloud Firestore

Start by initializing Cloud Firestore in your project. When doing this, please remember that you need to set your security rules to *test mode*. This was covered in the second section of this course if you need a refresher. I will show you how to secure the database in the upcoming lesson **Cloud Firestore security**.

To-do List Item Form

We create a form that will add to-do items.

HTML

Styling the form button

```
#to-do-list-form > button {
    max-width: 150px;
    margin-top: 30px;
}
```

Container For to-do Items

We create an HTML div element that will serve as your injection point for all of your to-do items. We give it an ID of to-do-list-items.

```
<!-- Dashboard -->
                                                                                 6
<div id="dashboard" class="hide-when-signed-out">
        <div>
                <form id="to-do-list-form">
                        <h1>New task</h1>
                        <input type="text" id="item" placeholder="Task" , required au</pre>
                        <button type="submit">Add</button>
                </form>
        </div>
        <div>
                <h1>To do list</h1>
                <div id="to-do-list-items"></div>
        </div>
        <div>
                <div id="account">
                        <h1>Account</h1>
                        <button class="auth" auth="sign-out">sign out
                </div>
        </div>
```

</dlv>

HTML

Real-time Listener for to-do Items

Setting up the real-time listener right now is a good idea because once to-do items are added, we will see them immediately. We are going to place this real-time listener in the authentication state listener because the script will run once a user is authenticated. We need to know who the user is in order to make this work.

Generate to-do Items

As we loop through our collection, my-list, we will create a paragraph and fill it with text from each document.

JavaScript

Access to-do Item Add Form

Before we can add items, we will need to get access to the form HTML element with the id of to-do-list-form.

```
// Get the to do list form for item submissions
const toDoListForm = document.getElementById('to-do-list-form');
```

Submit Event and Database Entry

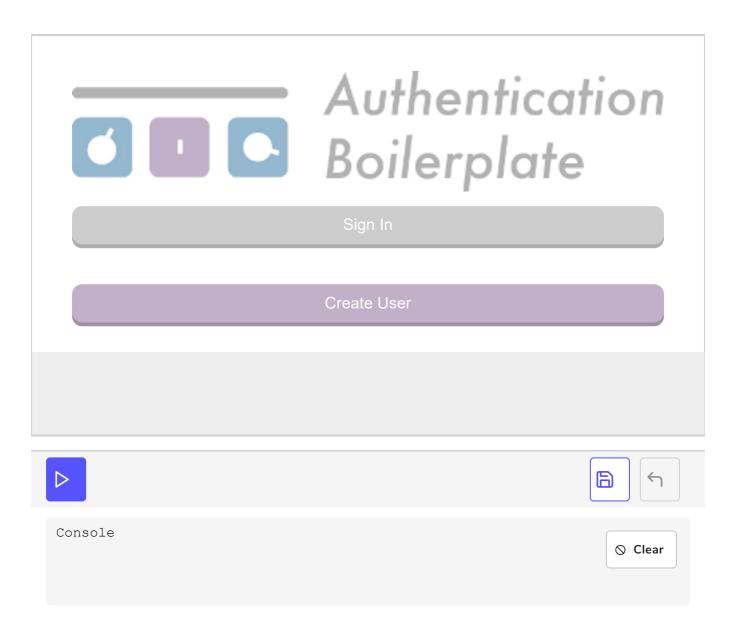
In earlier lessons, we covered submit events and database entries. The only difference with the following code is that it's adding a document and setting the key of the new document as the UID of the user who stores it.

JavaScript

To-Do List Application

Run the code below and add a few to-do items.

This code requires the following keys to execute:	
Key:	Value:
apiKey	Not Specified
authDomain	Not Specified
databaseURL	Not Specified
projectId	Not Specified
storageBucket	Not Specified
messagingSenderld	Not Specified
appld	Not Specified
Output	
JavaScript	
	HTML
	CSS (SCSS)



In the next lesson, I will show you how to delete to-do items. Think of removing them from your list as your way of 'completing' them.