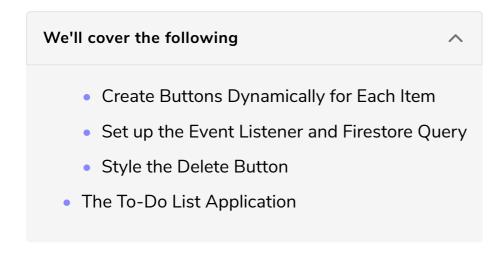
Deleting Completed Items

In this lesson, we will add the ability to delete items from the list in the application once the "to do" tasks have been completed.



Create Buttons Dynamically for Each Item

Remember when we looped through our collection of items and placed each *to-do* inside a paragraph tag, then appended that to a container element? We are going to create buttons inside that loop with a similar method. We use JavaScript to dynamically create them and add the attribute of data to each one, which will hold the id of that item. We will use a click event to read that attribute and delete the correct item.

JavaScript

Set up the Event Listener and Firestore Query

We make an event listener for the delete buttons. When a button is clicked, we

make a delete query to Firestore to delete that item based on the item's ID.

```
// Delete a to do list item
document.body.addEventListener('click', event => {
    if (event.target.matches('.delete-button')) {
        key = event.target.getAttribute('data');
        database.collection('to-do-lists').doc(uid).collection('my-list').doc(key).delete(
    };
});
JavaScript
```

Style the Delete Button

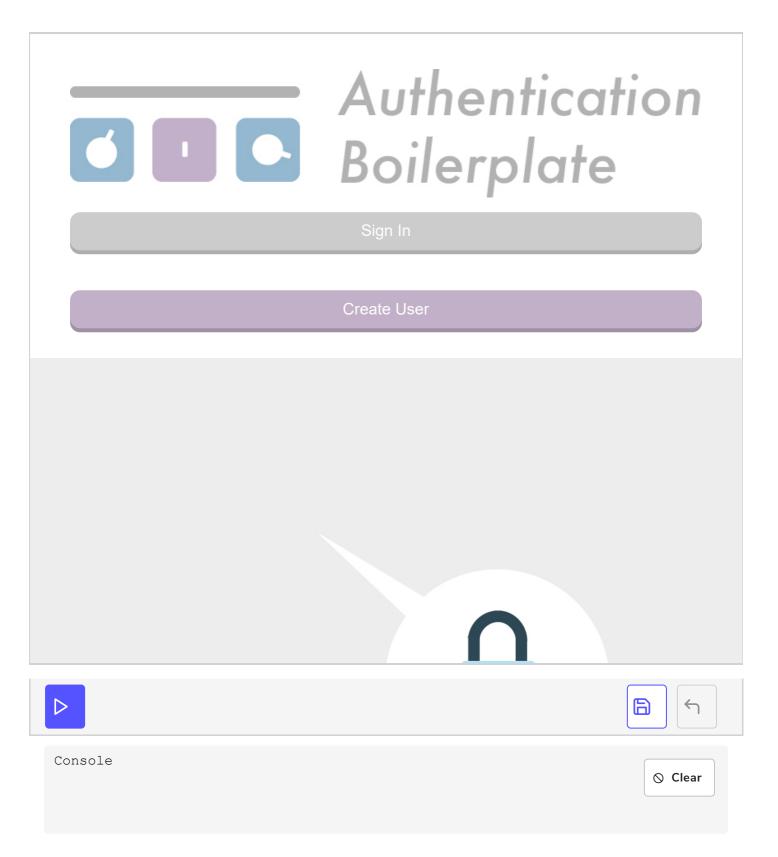
All we need for the delete button is a little margin on the left side of it to separate it from our to-do text.

```
.delete-button{
    margin-left: 20px;
}
```

The To-Do List Application

When you have completed a to-do item you can remove it from your list by deleting it.

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



If user that is not you tries to read, write or delete your to-do list items we need our application to block those attempts. We cover that in the next lesson which is all about securing your data.