

Lesson

Monday

Intermediate JavaScript (/intermediate-javascript)

/ Object-Oriented JavaScript

(/intermediate-javascript/object-oriented-javascript)

/ Address Book: Delete Functionality and Polish

Text

Cheat sheet

In this lesson, we'll add a user interface for our Address Book's delete functionality. We'll need to add a delete button to our HTML, an event listener for the new button, as well as a user interface function to handle actually calling

`AddressBook.prototype.deleteContact()` to delete the contact.

At the end of this lesson, we'll also add some bootstrap classes that will make our Address Book application look more polished.

Delete Functionality

First, let's add a button to our HTML. We'll add this button to our contact details section. This will allow us to connect delete functionality to a specific `Contact` object.

index.html

```
<div id="contact-details" class="hidden">
  <h2>Contact Details:</h2>
  <p>First Name: <span class="first-name"></span></p>
  <p>Last Name: <span class="last-name"></span></p>
  <p>Phone Number: <span class="phone-number"></span></p>
  <button class="delete" type="button">Delete</button>
</div>
```

Next, let's turn to our scripts. First, we'll create a new event listener attached to our delete button that listens for a 'click' event. Then we'll create the beginning of a handler function called `handleDelete()`.

js/scripts.js

```
...

function handleDelete(event) {

}

...

window.addEventListener("load", function (){
  document.querySelector("form#new-contact").addEventListener("submit", handleFormSubmission);
  document.querySelector("div#contacts").addEventListener("click", displayContactDetails);
  // The line below this one is new!
  document.querySelector("button.delete").addEventListener("click", handleDelete);
});
```

Next, we'll need to create the actual delete functionality. Our functionality relies on the `AddressBook.prototype.deleteContact()` method, which takes a `Contact` object `id` as an argument. This

means that our button will need access to the id of the current contact that's being displayed.

The `displayContactDetails()` function already has access to the `Contact` object's details, so let's update that function to also update our new delete button with an `id` attribute set to the id of the contact.

js/scripts.js

```
function displayContactDetails(event) {  
    const contact = addressBook.findContact(event.target.id);  
    document.querySelector(".first-name").innerText = contact.firstName;  
    document.querySelector(".last-name").innerText = contact.lastName;  
    document.querySelector(".phone-number").innerText = contact.phoneNumber;  
    // The line below this one is new!  
    document.querySelector("button.delete").setAttribute("id", contact.id);  
    document.querySelector("div#contact-details").removeAttribute("class");  
}
```

With `document.querySelector("button.delete").setAttribute("id", contact.id);`, we are accessing our delete button and then giving it an `id` attribute set to the `Contact` object's id. We'll do this every time we display the details of a `Contact` object, so our button will always have the value of the currently displayed `Contact`.

Now it's time to add code to the `handleDelete()` function:

js/scripts.js

```
function handleDelete(event) {  
  addressBook.deleteContact(event.target.id);  
  document.querySelector("button.delete").removeAttribute  
("id");  
  document.querySelector("div#contact-details").setAttribut  
e("class", "hidden");  
  listContacts(addressBook);  
}
```

Let's break down this new code line by line:

- First we call `AddressBook.prototype.deleteContact()` to delete the `Contact` object. Since `addressBook` is a global variable, we can call on it from within the `handleDelete()` function. Remember that we're using the global `addressBook` variable as a mock database and that we typically should avoid global variables.
- Then, we remove the `id` attribute from the delete button, thereby resetting it. This line of code is actually optional, because any time we use the `Element.setAttribute()` method to create an `id` for an element, it will overwrite the existing `id` attribute, if there is one. This means that when we call `Element.setAttribute()` in the `displayContactDetails()` function, it will overwrite the existing `id` attribute. Whether or not we include this line of code depends on the needs of our application. It may turn out to be better to ensure that our `handleDelete()` function clears the `id` for the delete button so it can't be incorrectly used elsewhere.
- In the next line of code, we are hiding the contact details div once more.
- Finally, we call the `listContacts` function again to refresh the list of contacts, this time without the contact that we've deleted.

More Improvements

Our address book is now fully functional. Let's add a few more user experience improvements before wrapping up.

Bootstrap Styles

First, let's utilize Bootstrap classes to add styling and a more organized layout to our page.

```
index.html
```

```
<!DOCTYPE html>
<html>
  <head>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dis
t/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1R
WuH61DGLwZJEdK2Kadq2F9CUG65"
      crossorigin="anonymous">
    <link href="css/styles.css" rel="stylesheet" type="tex
t/css">
    <script src="js/scripts.js"></script>
    <title>Address Book</title>
  </head>
  <body>
    <div class="container">
      <h1>Address Book</h1>
      <div class="row">
        <div class="col-md-6">
          <hr>
          <h2>Add a Contact:</h2>
          <form id="new-contact">
            <div class="form-group">
              <label for="new-first-name">First Name</label>
              <input type="text" class="form-control" id
="new-first-name" name="new-first-name">
            </div>
            <div class="form-group">
              <label for="new-last-name">Last Name</label>
              <input type="text" class="form-control" id
="new-last-name" name="new-last-name">
            </div>
            <div class="form-group">
              <label for="new-phone-number">Phone Number</l
abel>
              <input type="text" class="form-control" id
="new-phone-number" name="new-phone-number">
            </div>
            <button type="submit" class="btn-primary">Add</
```

```
button>
    </form>
    <hr>
    <h2>Contacts:</h2>
    <div id="contacts">
        </div>
    </div>
    <div class="col-md-6">
        <div id="contact-details" class="hidden">
            <h2>Contact Details:</h2>
            <p>First Name: <span class="first-name"></span>
        </p>
            <p>Last Name: <span class="last-name"></span></
        p>
            <p>Phone Number: <span class="phone-number"></s
        pan></p>
            <button class="btn-primary delete" type="butto
        n">Delete</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

Now our webpage will look a little nicer!

Empty Form Fields

Let's also make sure to empty out our form fields after submission:


```
js/scripts.js
```

```
...

function handleFormSubmission(event) {
  event.preventDefault();
  const inputtedFirstName = document.querySelector("input#new-first-name").value;
  const inputtedLastName = document.querySelector("input#new-last-name").value;
  const inputtedPhoneNumber = document.querySelector("input#new-phone-number").value;
  let newContact = new Contact(inputtedFirstName, inputtedLastName, inputtedPhoneNumber);
  addressBook.addContact(newContact);
  listContacts(addressBook);
  document.querySelector("input#new-first-name").value = null;
  document.querySelector("input#new-last-name").value = null;
  document.querySelector("input#new-phone-number").value = null;
}

...
```

If you want to see all the updated code, check the branch in the repository below.

 **Example GitHub Repo for the Address Book**
(https://github.com/epicodus-lessons/oop-address-book-v2/tree/8_adding_delete_functionality_and_polish)

Previous (/intermediate-javascript/object-oriented-javascript/address-book-event-bubbling-event-delegation-and-the-event-object)
Next (/intermediate-javascript/object-oriented-javascript/imposter-syndrome)

Lesson 21 of 33
Last updated March 23, 2023

disable dark mode



© 2023 Epicodus (<http://www.epicodus.com/>), Inc.