KALAA ART GALLERY

FINAL PROJECT APPLICATION

SHAKILA SHRESTHA

PURDUE OF INDIANAPOLIS

CIT 21500: WEB PROGRAMMING

PROF. TJ KANE

11/26/2024

Abstract

Showcase a gallery management application where artists and users can track, manage artwork, artist, exhibitions, gallery details. This app will use CRUD (Create, Read, Update, Delete) operations to handle gallery data.

Kalaa Art Gallery

Kalaa Art Gallery is a web-based gallery management application designed for gallery managers to track, manage, and display artworks, artists, and exhibition details. The application uses CRUD (Create, Read, Update, Delete) operations to handle data related to artists and artworks. The core functionality of the gallery includes dynamically adding and displaying artists, managing artworks, and providing an interactive gallery view for each artist. This project uses jQuery for DOM manipulation and a clean, responsive design to offer an intuitive user experience.

Method and Steps

Purpose of the Application

The **Kalaa Gallery** project is designed to help gallery managers efficiently manage their collection of artists and artworks. The application implements various JavaScript functions to facilitate the following operations:

Objective

The goal of the system is to:

- Load Artists Dynamically: Display a list of artists and their details.
- Manage Artworks: Allow users to view, add, update, or delete artworks associated with each artist.
- User Interaction: Provide a user-friendly interface for managing the artist gallery.
- Validation: Ensure that all fields are updated properly when making changes to an artwork.

Detailed Functionality

1. **Loading Artists**:
    - The loadArtists() function dynamically loads the artists and their details onto the page. It iterates over the artists array and creates a visual representation of each artist.
2. **Viewing Artist Gallery**:
    - The openGallery(id) function displays the artworks of the selected artist. It uses the artist's ID to fetch and display their artwork details.
3. **Add New Artist**:
    - The addNewArtist() function generates a new artist object with default properties and appends it to the artists array.

4. **Delete Artist**:
    - The deleteArtist(id) function removes an artist from the artists array, triggering a reload of the artist list.
5. **Add New Artwork**:
    - The addNewArtwork(artistId) function allows the user to add a new artwork to the selected artist's gallery by pushing a default artwork object into the artworks array of the corresponding artist.
6. **Update Artwork**:
    - The updateArtwork(artistId, artworkIndex) function allows users to update the title, medium, and size of an artwork. If any field is left blank, an error message is displayed using an alert.
    - The function validates the input and updates the artwork details in the artists array.
7. **Delete Artwork**:
    - The deleteArtwork(artistId, artworkIndex) function deletes an artwork from an artist's collection by removing it from the artworks array.

User Interface (UI)

- The UI consists of artist cards that display each artist's image and name.
- Each artist card has buttons to open their gallery and delete the artist.
- The gallery for each artist displays their artworks with buttons to delete or update the artworks.
- The interface is designed to be interactive, with smooth transitions between loading and displaying content.

Challenges and Solutions

1. **Challenge**: **Default Values During Update**
    - Initially, when updating artwork details, the system required all fields (title, medium, size) to be entered, which was cumbersome if only one field needed a change. The prompt-based input method also caused issues when left blank.
    - **Solution**: To handle this, a console message was used to log updates, and the system was modified to allow selective updates, leaving fields blank if users choose not to update them. This provides more flexibility for users while maintaining data consistency.
2. **Challenge**: **Dynamic Updates Without Refresh**
    - Each time an artwork was added, deleted, or updated, the gallery needed to be reloaded to reflect changes.
    - **Solution**: The `openGallery()` function was used to dynamically refresh the content, ensuring smooth updates without page reloads.

Task	Fulfillment
Define an array of objects and store it in an external file. Create a function to return the array.	The array of objects (`artists`) is stored in an external file (`data.js`). A function `getArtists()` returns the array, which is used in the main script (`script.js`).
Reflect the objects in the display area using jQuery.	The `loadArtists()` function dynamically generates and displays the objects.
Create a mechanism for creating a new object and refreshing the display.	The "Add Artist" form allows users to input new artist details. When submitted, the form adds the object to the array and updates the display.
Indicate when an item has been created.	Visually show the card.
Read: Mechanism for listing specific object details.	The `loadArtists()` function reflects all details dynamically in the display area. Each artist's card contains all relevant information.
Update: Change a part of the object without a prompt.	The "Update" button generates text fields within the object's card, allowing inline editing. After submitting, the data updates in the array, and the display refreshes.
Delete: Mechanism to remove an object and refresh.	The "Delete" button removes the artist from the array using `splice()` and triggers a refresh with `loadArtists()`. A visual notification confirms the deletion.
Indicate when an item has been deleted.	Remove the card
UI Design: Unique and professional.	$("body").css, $(".artist-card img, .artwork-card img").css({ Is used for the card images

Additional Considerations

- **Code Structure**: The code follows modern JavaScript best practices, using `let` and `const` for variable declarations, and `function` declarations to ensure better scoping and maintainability.
- **Testing**: Before final submission, the app was tested on different browsers and devices to ensure cross-browser compatibility and responsiveness.

Conclusion

The **Kalaa Gallery** application successfully meets its objective of managing gallery data through intuitive CRUD operations. It provides an interactive and user-friendly platform for gallery managers to track and display artists and artworks.