Dave Manchenko

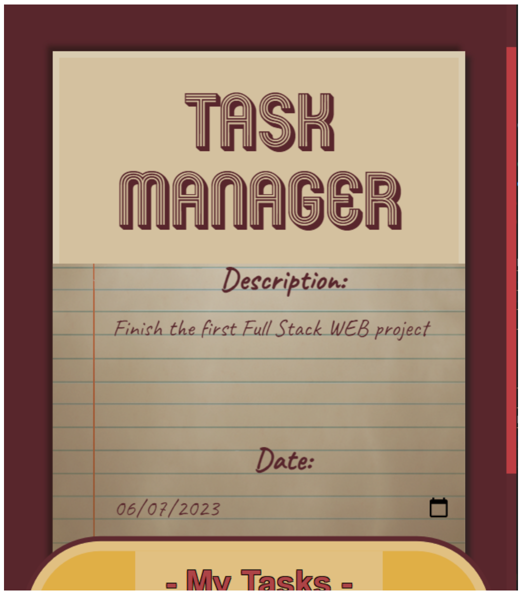
davemanchenko@gmail.com
https://www.linkedin.com/in/dave-manchenko/
(054)767-7190

Task Manager

June 07, 2023

Project Overview

The aim of this school project is to develop a task management website that allows users to fill a form with task details such as task description, task due-by date, and time. The submitted data will be stored and added to a list of tasks that users can view on the website. The project will involve designing a user-friendly interface, implementing form submission functionality, and displaying the task list dynamically.



Key features

Task Form: The website will include a task form where users can enter task details, including a task description and the due-by date and time.

Task Submission: Upon completing the task form, users will be able to submit the data, which will be processed and stored in a database or appropriate storage system.

Task List: A task list page will be provided to display the tasks submitted by the users. The list will show the task description, due date, and time, allowing users to have a clear overview of their tasks.

Responsive Design: The website will be designed to be responsive, ensuring it functions well and provides a seamless experience across various devices, including desktops, laptops, tablets, and mobile phones.

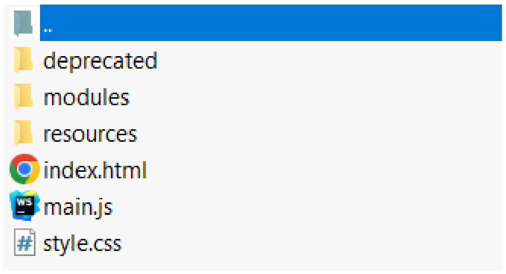
Error Handling: Proper error handling mechanisms will be implemented to validate user input, handle any unexpected errors, and provide appropriate feedback to users when necessary.

Data Persistence: The submitted task data will be stored persistently to ensure that the task list remains available even after the user logs out or the website restarts.

User Experience (UX): The overall user experience will be a key consideration throughout the development process, with attention given to intuitive navigation, visually appealing design, and smooth interactions.

Installation and Setup

1. Download the .zip file from the github repo.
2. Unpack into a folder of your choice, make sure to unpack all included files and folders.
3. Double click on index.html to open the website in your browser

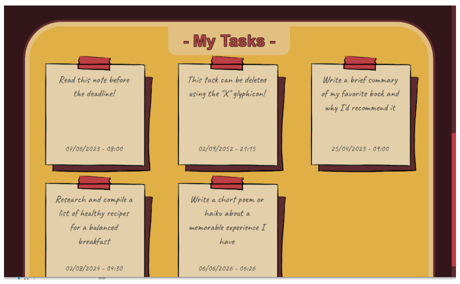


```
..
deprecated
modules
resources
index.html
main.js
style.css
```

That's it! The project should now be running in your web browser. You can interact with the form, fill in task details, and submit them. The submitted tasks will be displayed in the task list on the same page.

Note: Make sure you have a compatible web browser installed on your system to run the project successfully.

Usage

1. **Task Submission:**
   Upon accessing the website, users will be presented with a task form.
   To add a new task, users need to fill in the required task details in the form.
   The task details typically include a task description, due date, and time.
   Once all the required information is entered, users can click the "Submit" button to submit the task.
2. **Task List:**
   
   - My Tasks -

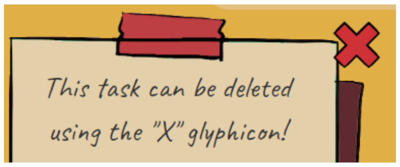
   Read this note before the deadline!
   07/06/2023 - 08:00

   This task can be deleted using the "X" glyphicon!
   02/09/2052 - 21:15

   Write a brief summary of my favorite book and why I'd recommend it
   25/04/2023 - 09:00

   Research and compile a list of healthy recipes for a balanced breakfast
   02/08/2024 - 04:30

   Write a short poem or haiku about a memorable experience I have
   06/06/2026 - 06:26

   After submitting a task, it will be added to the task list, which is displayed on the same page. Users can view their submitted tasks in the task list, which shows the task description, due date, and time.
   The task list provides an overview of all the tasks that have been added.
   Users can scroll through the list to see all the tasks.
3. **Task Deletion:**
   Each task in the task list is accompanied by a "Delete" button.
   
   This task can be deleted using the "X" glyphicon!

   Users can click the "Delete" button associated with a particular task to remove it from the task list.
   This allows users to remove completed or unnecessary tasks from their list.
4. **Task Management:**
   The website facilitates task management by allowing users to add tasks, view the task list, and delete tasks as needed.
   Users can use the form to add new tasks, providing task descriptions and due dates.
   The task list provides an organized view of all the tasks submitted by the user.
   Users can delete tasks from the list, helping them keep the list up to date and relevant.

5. **User Experience:**
   The website aims to provide a simple and intuitive user experience.
   Users can easily submit tasks using the form and view them in the task list.
   The ability to delete tasks offers a way to maintain an updated task list.

Overall, the task management website allows users to submit tasks through a form, view their tasks in a list, and delete tasks as needed. It provides a straightforward way to manage and track tasks, helping users stay organized and focused on their objectives.

Technologies Used

- **HTML:** Used for structuring the web pages and creating the user interface elements.
- **CSS:** Used for styling the HTML elements and applying visual enhancements to the website.
- **JavaScript (ES6+):** Used for implementing dynamic and interactive functionality in the website.
- **Bootstrap:** Utilized the Bootstrap framework for responsive design and pre-built CSS styles, making the website visually appealing and mobile-friendly.
- **LocalStorage API:** Utilized the LocalStorage API to store task data locally in the user's browser, allowing tasks to persist across sessions.

The project primarily focuses on HTML, CSS, and JavaScript, while leveraging frameworks and libraries like Bootstrap and the LocalStorage API to enhance the user interface, improve responsiveness, and simplify development tasks.

Code structure and Architecture

Structure and organization:

- **main.js:** The main module is located in the root directory. It ties all of the other modules together and handles the form's submission.
- **notepad.mjs:** The notepad module is located in the modules folder. It handles creation of task notes visually and storing the associated data in local storage. To do this it depends on storage.mjs.
- **storage.mjs:** The storage module is located in the modules folder. It includes an interface and a class that extends the interface. The extending class is injected into the notepad module and handles storing and removing data from local storage.
- **clipboard.mjs:** The clipboard module is located in the modules folder. It handles the custom scrolling functionality of the clipboard and the appended notes by listening to the scroll event.

Design patterns, architectural choices and programming paradigms:

1. **Modularization:** The code utilizes the ES6 module system to import functionality from separate modules (notepad.mjs, storage.mjs, clipboard.mjs). This promotes modular development, making the code more organized, maintainable, and reusable.
2. **Immediately-Invoked Function Expression (IIFE):** The code wraps the main logic inside an IIFE (function() { ... })();. This pattern is commonly used to create a private scope for variables and functions, preventing them from polluting the global scope.
3. **Dependency Injection:** The code implements the Dependency Injection pattern in the notepad.js module. The Notepad class has a constructor that accepts a StorageInterface object as a parameter. By injecting the dependency, the Notepad class is decoupled from the specific implementation of storage and can work with any object that adheres to the StorageInterface. This promotes code flexibility, testability, and modularity.
4. **Object-Oriented Programming (OOP):** The code demonstrates the usage of objects and classes. The Notepad and StorageManager are instantiated as objects, following the principles of encapsulation and abstraction in OOP.

5. **Event-Driven Programming:** The code sets up an event listener for form submission using addEventListener. It handles the form submission event and executes the corresponding logic. This approach follows an event-driven programming paradigm where the program responds to events triggered by the user.
6. **Separation of Concerns:** The code separates different responsibilities into individual functions. For example, formToObject is responsible for converting form data to an object representation, handleSubmit handles form submission logic, and initialize sets up event listeners and loads stored data. This separation of concerns promotes code organization and maintainability.
7. **Error Handling:** Although not explicitly shown in the provided code, proper error handling and validation are crucial in real-world applications. It is a good practice to validate user inputs and handle potential errors to ensure the code functions correctly and gracefully handles unexpected situations.
8. **Functional Programming:** The code uses functional programming concepts in the formToObject function, utilizing array methods like reduce to transform the form elements into an object representation. Functional programming promotes immutability and the use of pure functions.

Project Status and Future Improvements

The project is a work-in-progress and will continue to receive updates and improvements over time. The first of which will undoubtedly be the style.css file. A major focus of the project has been developing robust front end functionality with an emphasis on proper methodology when it comes to the OOP code. This preference has taken a toll on the organization and structure of the CSS code. While perfectly functional, the CSS code will definitely benefit from some clean up.