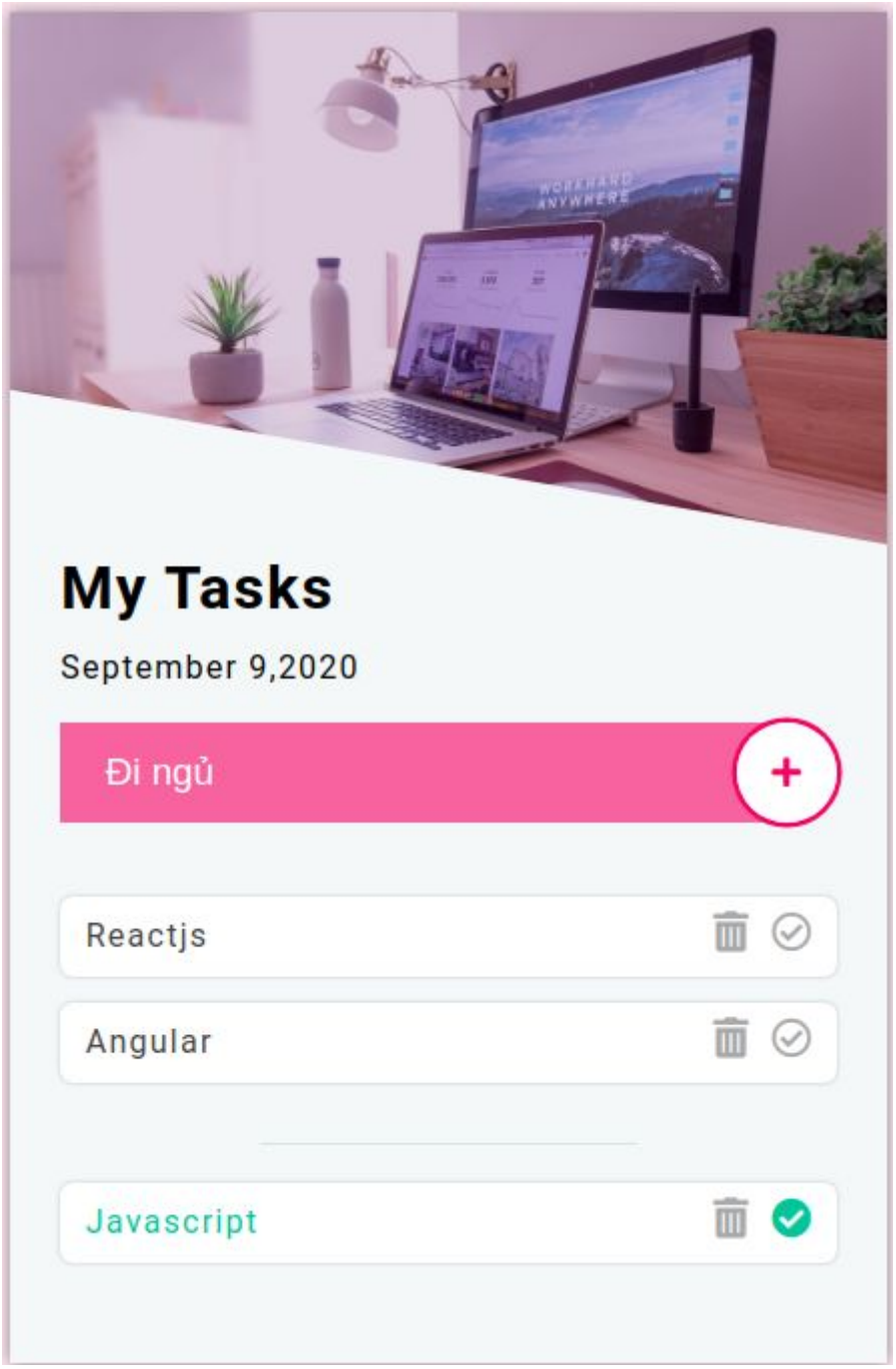
Bài tập TodoList

- Tạo file main.js
- Tạo lớp đối tượng Task
- Tạo lớp đối tượng TaskService: sử dụng axios handle các phương thức (`getListTask, addTask, deleteTask, getTaskById, updateTask`).



- Tại ô input thêm task vào.
- Task thêm thành công sẽ có trạng thái là “todo” (đang làm)

- Khi action vào button check, sẽ chuyển trạng thái task “todo” thành “completed” (đã hoàn thành) và ngược lại.
- Task “todo” & “completed” sẽ được liệt kê ở 2 thẻ ul tại file index.html
- Xóa task

❖ Lưu ý: Bài tập này hơi thách thức tại chức năng check task. Vì khi check phải biết được id task để lấy thông tin chi tiết. Có thông tin chi tiết rồi sau đó mới update task được. (Xử lý Asyns).

❖ Sử dụng mockapi.io tự tạo API theo như sau (đừng có làm trên link của mình nhé :D) `https://5a6451dcf2bae00012ca1a6a.mockapi.io/api/todo (GET, POST, PUT, DELETE)`

Link demo mẫu: https://mediumstock.000webhostapp.com/todolist/index.html

* Chức năng Bonus (nâng trình): Không làm không sao, không bị trừ điểm.

Trong quá trình `getListTask, addTask, deleteTask, getTaskById, updateTask` sẽ mất 1 khoảng thời gian Pending chờ kết quả. Làm thêm vòng quay “Loading” để User biết được là trong quá trình chờ.
link demo loading: https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_loader

Gợi ý: tạo 1 biến làm cờ để check trạng thái. (Có nhiều cách để làm, đây là 1 trong những cách).
B1: Tạo isLoading = false (global)
B2: Tạo 1 hàm checkLoading()
    Nếu isLoading đúng => append HTML Loading vào tag <body>
    Ngược lại => remove HTML Loading
B3: Trước khi gọi service, bật isLoading = true. Gọi hàm checkLoading()
B4: Sau khi thành công. bật isLoading = false. Gọi hàm checkLoading()
B5: Trường hợp thất bại. làm giống B4

* css loading cho đẹp xíu.