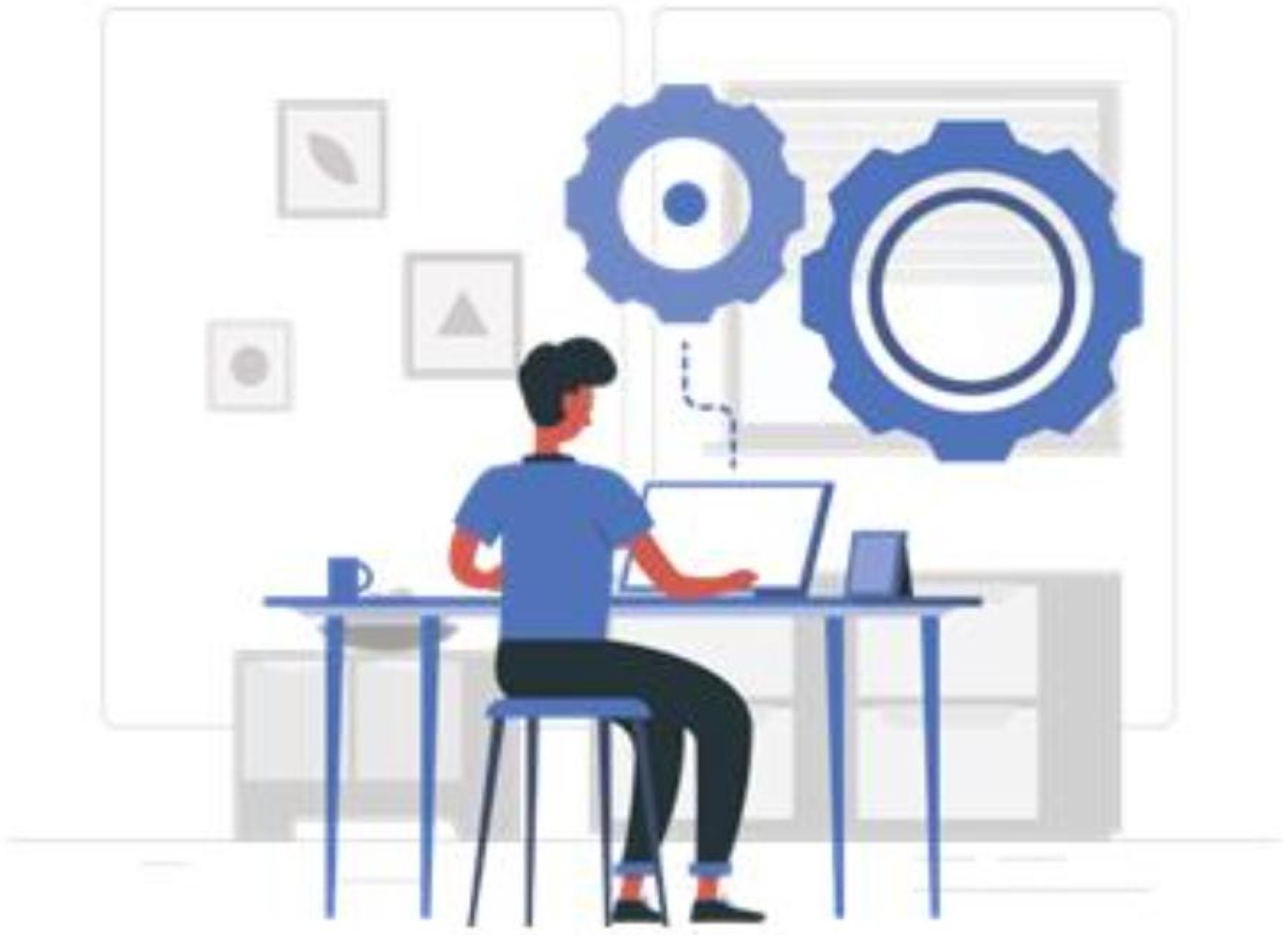


Challenge

Build Interactive Web Pages With TypeScript

Challenge

- Displays fruits in Fruit Fantasy app

Menu

00:02 00:00:14 02/ 14

Points to Remember

- Install TypeScript on your system using NPM.
- Use appropriate data types for all the data used in the solution.
- Avoid using the data type "*any*"**,** unless you are unsure about the data type.
- Indent the TypeScript code appropriately.
- Include comments, wherever required, to make the code readable.

Context

Fruit Fantasy is a local fruit vendor in San Francisco that launched an app that lets its customers view the fresh fruits offered daily. They wanted to serve their customers better and offer the best in terms of quality, benefits, flavor, and taste.

As a first step, design a web page using TypeScript to display the fresh fruits from the database using fetch API. The fruits must be displayed as cards, ensuring the content is easy to scan.

Instructions for Accessing the Boilerplate

- Click here for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the practice exercise.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open command terminal run the command **npm install** to install Mocha and Chai as dependencies.
- Open the folder containing the boilerplate code in VS Code.
- Provide the solution code in the files specified with the task details.

Instructions for the Challenge

- Write code inside **solution/index.html** to fulfil the requirements stated with the tasks.
- Write TypeScript code in the **app.ts** located inside **solution/src** folder of the boilerplate as per the requirements stated in the upcoming slides.
- Open the terminal in the root directory and run the command **`npm install`** to install the dependencies.
- Give the command **`npm run build`** to compile **`.ts`** files and verify the converted files are located inside **`solution/public/js`** folder.
- Open another terminal window and start the json-server by giving the command **`json-server solution/json/fruits.json`**.
- Open the **index.html** file using Live Server and test the expected output.
- Submit the files for evaluation.

Display Fruits in Fruit Fantasy App

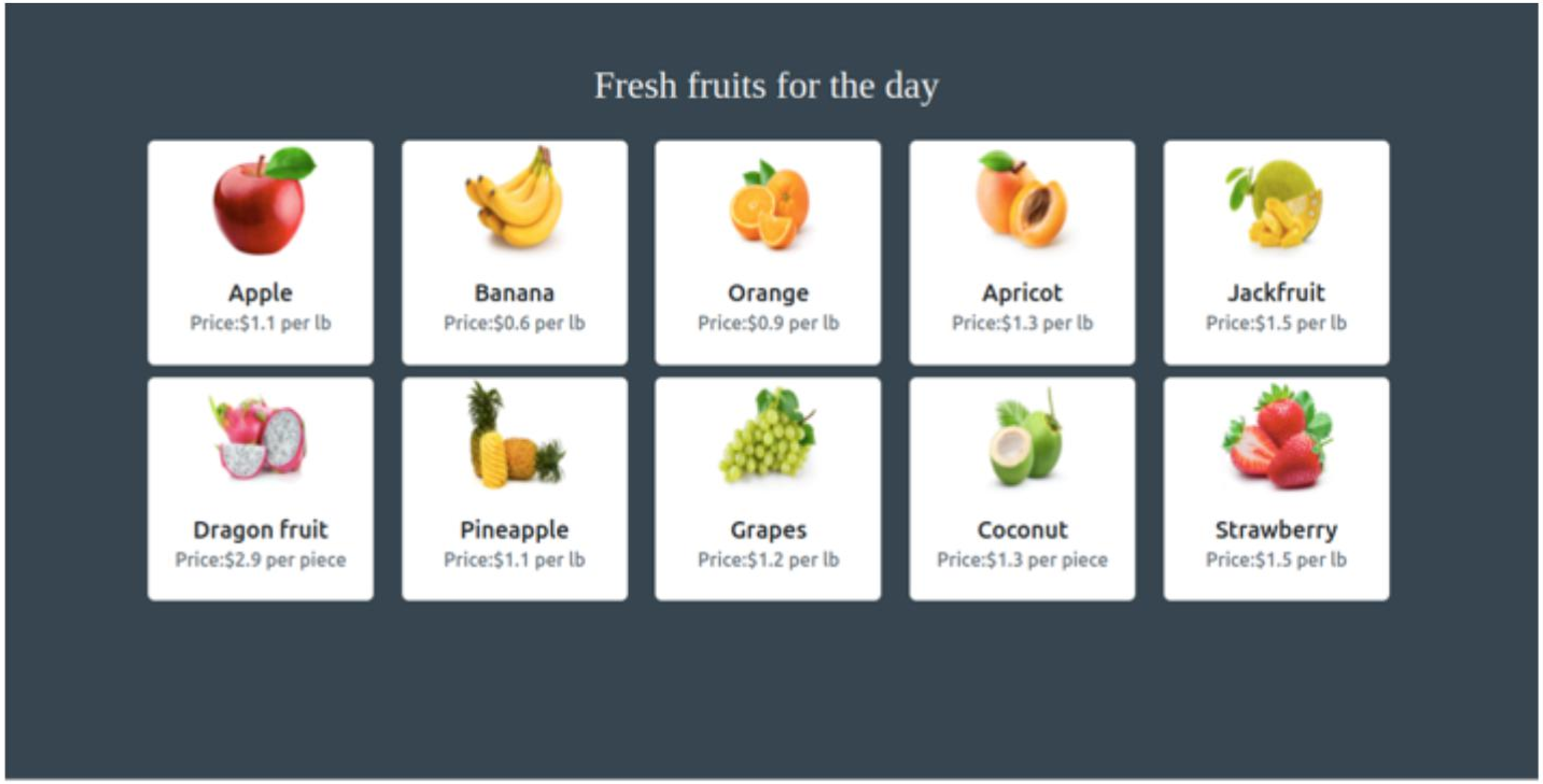
Fruit Fantasy is a local vendor that has launched an app which lets its customers view the fresh fruits offered on a daily basis.

Build a web page using TypeScript which displays the fruits from the database using fetch API.

CHALLENGE

Preview: Display Fruits in Fruit Fantasy App

- The final output of the web page should resemble as below:


Fresh fruits for the day

Apple Price:$1.1 per lb	Banana Price:$0.6 per lb	Orange Price:$0.9 per lb	Apricot Price:$1.3 per lb	Jackfruit Price:$1.5 per lb
Dragon fruit Price:$2.9 per piece	Pineapple Price:$1.1 per lb	Grapes Price:$1.2 per lb	Coconut Price:$1.3 per piece	Strawberry Price:$1.5 per lb

Tasks

The solution for this challenge can be done in 5 steps.

- Step 1: Design the Fruit Fantasy Web Page
- Step 2: Create Fruit Object
- Step 3: Retrieve Fruits using Fetch API
- Step 4: Transform Each Fruit Data
- Step 5: Display the Fruits as Cards

Note: Details about these steps are given in the upcoming slide.

Steps Details

- **Step1: Design the Fruit Fantasy Web Page**
  - Add header in `` `index.html` `` file to specify the title "Fresh fruits for the day" .
  - Styles can be added inside `` `public/style.css` `` file.
  - Use **`onload`** event within body tag to call **`getFruits()`** function to populate the fruits.
  - Add a **`<div>`** container whose content should be dynamically loaded with the fruits.
  - Give the path as `` `public/js/app.js` `` inside the script tag of the `` `index.html` ``.
- **Step 2: Create Fruit Object**
  - Create an object type called 'Fruit' with the attributes like id, name, image, unit and price to define the fruit.

Steps Details (Contd.)

- **Step 3: Retrieve Fruits using Fetch API**
  - Define **`getFruits()`** to retrieve all the fruits from the JSON server using fetch API.
  - Use Promise chaining to process the response.
    - Successful promise callback function should convert the response body text to json object and return promise as result.
    - Returned successful promise callback should call **`transform()`** function by passing the json object as an argument.
- **Step 4: Transform Each Fruit Data**
  - Define the **`transform()`** function, which loops through each fruit data retrieved as a response to create a new object called **`transformedFruit`** that mirrors the object 'Fruit'.
  - Within the **`for`** loop, call **`showFruits()`** function and pass **`transformedFruit`** as an argument to it.

Steps Details (Contd.)

- **Step 5: Display the Fruits as Cards**
  - Declare a constant of HTMLElement type to represent the `<div>` container element.
  - Create a string object which contains the HTML code to display each fruit as a card inside **`showFruits()`** function.
    - Use bootstrap class to style the card.
    - Additional styling can be provided inside **`public/css/style.css`**.
  - Append this string to the constant using **`innerHTML`** property.