It looks like you've shared an HTML document for a Fruit Shop web page along with some associated CSS and JavaScript files. The code includes Bootstrap for styling and form components, and it appears to have a form for adding fruits with their names, prices, and quantities.

Here's a brief overview of your code structure:

1. HTML Structure:

- The document starts with the standard HTML5 doctype declaration.

- The `<head>` section contains the title of the page, links to external stylesheets (including Bootstrap), and possibly other stylesheets (`style.css`).

- The `<body>` section consists of two main containers (`div` elements) and includes a form for adding fruits, buttons to view the fruit basket and total price, and a hidden div for displaying the total.

2. Form Section:

- The form includes input fields for the fruit name, price, and quantity.

- The form has buttons for adding a fruit to the basket, viewing the fruit basket, and viewing the total price.

- The form seems to be using Bootstrap classes for styling.

3. JavaScript:

- JavaScript files (`app.js`) are included at the end of the HTML body.

- It seems like you have some functionality for adding fruits to the basket and viewing the basket and total price.

4.Styling:

- You've used Bootstrap classes for styling, and there's an external stylesheet (`style.css`) that might contain additional styles.

5.Comments:

- There are some commented-out sections in your code (e.g., the total price div and a dummy data example). Uncommenting and using such sections would depend on the specific functionality you want to implement.

6.Dependencies:

- You're using Bootstrap for styling and jQuery, Popper.js, and Bootstrap JS for additional functionalities.

7. Issues/Notes:

- The form has the class `was-validation`, but it seems like it might be intended to be `was-validated` for Bootstrap's form validation to work correctly.

- Some elements have the class `hidden` (e.g., `total-div` and `list`). These might be intended to be displayed dynamically using JavaScript.

Remember to check your JavaScript code (`app.js`) to see how the form submission and other functionalities are handled. Also, ensure that the necessary dependencies (Bootstrap, jQuery, Popper.js) are included and available.

If you have specific questions or need assistance with particular aspects of your code, feel free to ask!

It seems like you've shared JavaScript code for handling the functionality of your Fruit Shop web page. Let me break down the key aspects:

1. Event Listeners:

- The code sets up event listeners for the "Add," "Remove," "View Fruit Basket," and "View Total Price" buttons.

2. State and Data Storage:

- `state` is an object used to store information about each fruit (name, price, quantity).

- `fruitList` is an array storing the names of fruits.

- `total` is a variable keeping track of the total price.

3.Add Button Click Event:

- When the "Add" button is clicked, the function captures the input values (name, price, quantity).

- The input values are used to update the `state` object, `fruitList` array, and `total` variable.

- The `renderList()` and `renderTotal()` functions are called to update the displayed list and total.

4. Remove Button Click Event:

- When a "Remove" button is clicked, the corresponding fruit is removed from the list and state.

- The `renderList()` and `renderTotal()` functions are called to update the displayed list and total.

5.View Total Price Button Click Event:

- Toggling the visibility of the total price div (`totalDiv`) and calling `renderTotal()` to update the displayed total.

6. View Fruit Basket Button Click Event:

- Toggling the visibility of the list of fruits (`list`).

7. Rendering Functions:

- `renderList()` function dynamically generates HTML for each fruit in alphabetical order and updates the `list` element.

- `renderTotal()` function generates HTML for the total price and updates the `totalDiv` element.

8. Utility Function:

- The `capitalize` function capitalizes the first letter of a string.

9. Other Notes:

- The code uses event delegation for the "Remove" button clicks by checking the class of the clicked element.

- The use of `dataset` attributes for storing additional information about each fruit.

10. Comments:

- The code is well-commented, providing explanations for different sections and functions.

11.Suggestions:

- Ensure that the HTML structure and CSS styles complement the JavaScript functionality.
- Test the code thoroughly, especially edge cases and user interactions.
- Consider error handling and validation for user inputs.

Overall, the code looks well-organized and functional for managing a fruit shop application. If you have specific questions or need further assistance with any part of the code, feel free to ask!

It looks like you've provided the CSS styles for your Fruit Shop web page. Let's break down the key aspects of your styles:

1.Global Styles:

- `body, html`: Set to 100% height, ensuring the entire viewport is covered.

2. Flexbox Styles:

- `.flexview`: A class applied to elements using flexbox.
- `.flexview p`: Styling for paragraphs within elements with the `flexview` class.

3. Content Section Styles:

- `.content`: Applied to the content section, set to 85% of the viewport height (`85vh`).

4. Form and Input Styles:

- `.fruit-input`: Styling for fruit input labels.
- `.remove-btn`: Styling for the "Remove" button, setting a background color, and changing color on hover.
- `form`: Styling for the form, including a height of `60vh`, padding at the top, making it 60% of the viewport height, and adjusting for a fixed header.

5. Total Section Styles:

- `.total`: Applied to the total section, set to 25% of the viewport height (`25vh`), with centered content and padding at the top.

6.Visibility Styles:

- `.hidden`: Used to hide elements by setting `display: none`.

7.Other Styles:

- Various margin, padding, and font-size adjustments for better visual presentation.

- The `large` class is applied to increase the font size of certain elements.

8. Button Styles:

- Background and color changes for the "Remove" button and its hover state.

9. Suggestions:

- Consider adding more detailed comments to your CSS to describe the purpose of each section or specific rules.

- Make sure that the styling aligns with your design preferences and is consistent across different browsers.

- Test your styles on various devices and screen sizes to ensure responsiveness.

Your CSS seems well-structured and should contribute to a visually appealing and functional Fruit Shop web page. If you have specific questions or need further guidance on any aspect of your styles, feel free to ask!