Lecture-18: TO-DO App Enhancements - Part 2
(FINAL LECTURE OF SEMESTER)

- **Teacher:** Sir Arsalan Shah
- **Typist:** Yousuf Naveed

Splice Method and Array Manipulation:

The splice method allows you to modify arrays by adding or removing elements at specific positions. It's a powerful tool for manipulating arrays to customize their content according to your needs.

Array Manipulation involves various actions that you can perform on arrays, such as adding, removing, and modifying elements. The splice method plays a key role in array manipulation by providing the ability to insert and delete elements within an array. This allows you to maintain and control the data within the array dynamically.

Splice Method Syntax:

The syntax for the splice method is as follows:

```
array.splice(index, remove_count, item_list);
```

- **`index`**: The index at which the modification should begin.
- **`remove_count`**: The number of elements to be removed from the array starting at the specified index.
- **`item_list`**: Optional. The elements to be added to the array at the specified index.

By using the splice method, you can easily tailor the content of your arrays to suit your application's requirements. It's a versatile function that empowers you to perform intricate array manipulations with ease.

TO-DO App:

In this part, we will continue enhancing our TO-DO app with new features. We will focus on refining the functionality and user experience.

10. Implement Delete Button Functionality

- Create a delete button for each item in the list.
- Use `**onclick**` event to call the `**deleted**` function and pass the index parameter.

11. Updated `deleted` Function

- Remove the item from the `**itemjsonArray**` using the `**splice**` method.

12. Separate Functions for Table and Adding Items

- Move the table-related functionality to a new function for better organization.
- Similarly, separate the functionality for adding items to the TO-DO list.

13. Update `showdata` Function

- Call `**showdata**` function within the `**deleted**` and `**addtolist**` functions to update the UI after changes.

14. Implement ClearAll Button

- Add a "ClearAll" button to delete all items from the TO-DO list.
- Use `**confirm**` dialog to ask for user confirmation before clearing the list.
- If confirmed, use `**localStorage.clear()**` to clear the storage and then call the `**showdata**` function.

HTML and CSS Positions

Positions in HTML and CSS refer to how elements are placed within a web page. There are four main positions that can be utilized:

1. **Static:** The default position for all HTML tags. Elements in static position are not movable and are displayed based on their order in the HTML structure.
2. **Relative:** This position allows you to move an element from its normal position using the `**top**`, `**right**`, `**bottom**`, or `**left**` properties. It is often used for creating space around elements using margin or padding.

3. **Absolute:** When an element is given an absolute position, it's positioned relative to its closest non-static ancestor. This means that the element's position is determined based on its parent element's positioning. For instance, if you move a child element with absolute positioning, it will reference its parent with non-static positioning.

4. **Fixed:** Elements with a fixed position remain in a fixed location on the screen, even when you scroll. This can be useful for navigation bars or elements you want to "stick" to a specific position.

5. **Sticky:** Similar to fixed, sticky positioning also remains in place while scrolling, but only within the context of its parent container. Once the container's edge is reached, the element will stick to that edge.

z-index:

The `**z-index**` property is used to control the stacking order of positioned elements along the z-axis (depth) on a web page. Elements with a higher `**z-index**` will be displayed above those with a lower value, essentially determining which elements are on top of others.

Difference Between Fixed and Sticky:

The key difference between the `**position: fixed**` and `**position: sticky**` is how they determine their fixed placement. With `**position: fixed**`, the element is fixed in relation to the viewport, regardless of scrolling. In contrast, `**position: sticky**` is based on its parent container. When scrolling, a sticky element will remain within its container until it reaches the edge, where it will "stick."

Understanding these positions and the z-index property is crucial for creating responsive and visually appealing layouts in HTML and CSS. It enables you to control the placement and layering of elements on a web page, ensuring an organized and structured design.