Extra: SMS Code Verification System

Objective:

Create an interactive code verification system that uses a 6-character code, supposedly received via SMS, and allows users to input it for validation.

Here is an example.

Instructions:

Initial Setup:

1. Initial Variable Declarations:

- **form:** Grabs the form element using its `id`.
- **numList:** Gets a NodeList of all text input fields inside the form.
- **hidCode:** Grabs the hidden input element which is used to store the entered code for submission.
- **visCode:** Grabs the visible input which is used to show the actual SMS code to the user.
- **sms:** Generates a random 6-character alphanumeric string that simulates the SMS code (use Google and stackOverflow for this).

2. Initial Setup:

- Display the generated `sms` code in `visCode` input's value for the user to see.
- Set the focus on the first input field, ready for the user to start entering the code.

3. `fillCode(el)`:

Create a function `fillCode(el)`. This function validates and processes the input for each individual input field:

- If the value entered in the input is an alphanumeric character (`/^[a-zA-Z0-9]$/.test(el.value)`), it processes the input.

- Combine the values of all individual input fields into one string and assigns it to the hidden input `hidCode.value`.

  Instructions for Combining Input Field Values:

  1. **NodeList to Array Transformation**: Convert the NodeList `numList` to an array. This transformation allows you to use a broader range of array methods.
  2. **Use Reduce for Combination**: Apply the `reduce` method on the newly formed array. This method will iterate through each input field in the array. During each iteration:
     - Combine the current input field's value with the accumulated value.
     - The process starts with an empty string and builds upon it by appending each input field's value.
  3. **Standardize the Case**: Once you have the combined string, convert the entire string to lowercase to ensure consistency.

- Move the focus to the next input field **if one exists**.
- If the value entered is not an alphanumeric character, it clears the input.

4. `fillFromClipboard(event)`:

Create a function `fillFromClipboard(event)`. This function Allows the user to paste a code:

- Start with a prevent default on the event to stop any standard pasting functionality.
- Create a `paste` variable that will be assigned to the `clipboardData` from the `event` object or the `window` object.
- Check the pasted content:
  - If it's shorter than 6 characters or contains non-alphanumeric characters, the function returns `false`, without doing anything.
  - Otherwise, assign each character of the pasted content to the corresponding input field valus (`numList[index].value`).

5. `checkAndSubmit()`:

Create a function `checkAndSubmit()` that checks whether the `visCode.value` matches the generated `sms` code (`hidCode.value`):

- If they don't match, return `false`.

- If the two codes match, submit the the form.

6. `checkValue(el)`:

Create a function `checkValue(el)` that updates the UI based on input validity:

- If the input value is an alphanumeric character, the "error" class is removed, indicating the input is correct.
- Otherwise, the "error" class is added, indicating a mistake in the input.

7. Event Listeners:

"JavaScript addEventListener() with Examples"

Instructions for Setting Up Input Field:

1. For each input (`el`) field in `numList` :
    - Attach an event listener for the `keyup` event:
        - As soon as a key is released after being pressed, pass the `fillCode(el)` function to validate and process the typed input.
    - Attach another event listener for the `keyup` event:
        - Immediately after, pass the `checkValue(el)` function. This checks the entered value's correctness and adjusts the input field's appearance accordingly.
    - Attach a separate event listener for the `paste` event:
        - When content is pasted into the input field, pass the `fillFromClipboard(event)` function. This allows for direct pasting of a code and ensures it populates the input fields appropriately.
    - Attach another event listener for the `keyup` event once more:
        - Following any key release, pass the `checkAndSubmit()` function. This verifies the entirety of the entered code. If all input fields contain the correct sequence, the form will automatically submit.