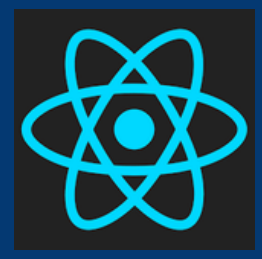
USECALLBACK

INTERVIEW QUESTIONS-86



Follow on in

@DUVVURU KISHORE

First we will learn usecallback hook and then we will compare with usememo hook

USECALLBACK

- The useCallback hook is used to memoize functions in React.
- It returns a memoized version of the callback function that only changes if one of the dependencies has changed.
- This is useful for optimizing performance by preventing unnecessary re-renders of components

WITHOUT USECALLBACK

- The code example beside..It maintains two state variables using the useState hook: number and dark.
- The getItems function generates an array of items based on the current value of number.
- The component renders an input field for number,
- a button to toggle the theme
- and the List component passing getItems as a prop.

APP COMPONENT

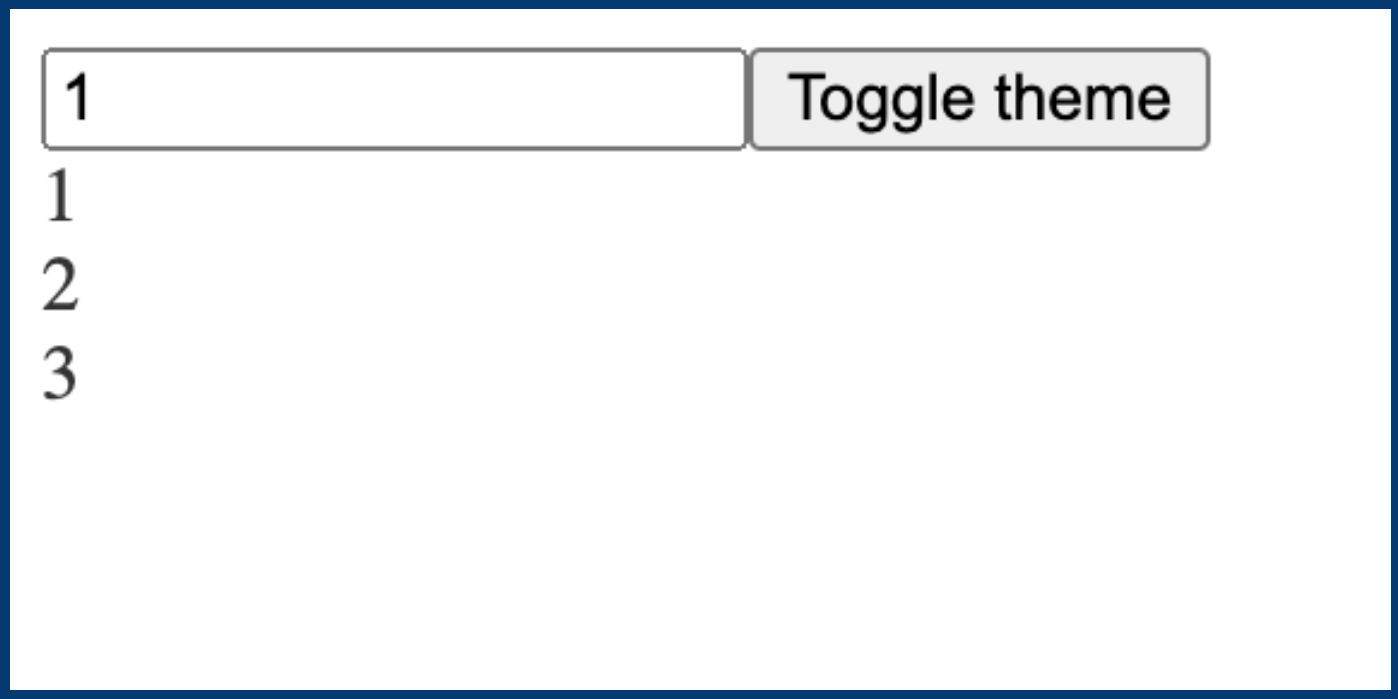
```jsx
import React, { useState } from 'react';
import List from './List.js';

export default function App() {
  const [number, setNumber] = useState(1);

  const [dark, setDark] = useState(false);

  const getItems = () => {
    return [number, number + 1, number + 2];
  };

  const theme = {
    backgroundColor: dark ? '#333' : '#FFF',
    color: dark ? '#FFF' : '#333',
  };

  return (
    <div style={theme}>
      <input
        type="number"
        value={number}
        onChange={(e) => setNumber(parseInt(e.target.value))}
      />
      <button onClick={() => setDark((prevDark) => !prevDark)}>
        Toggle theme
      </button>
      <List getItems={getItems} />
    </div>
  );
}
```

LIST COMPONENT

- Receives a list of items as a prop.
- Logs a message whenever the list of items changes.
- Renders the list of items as a series of elements.

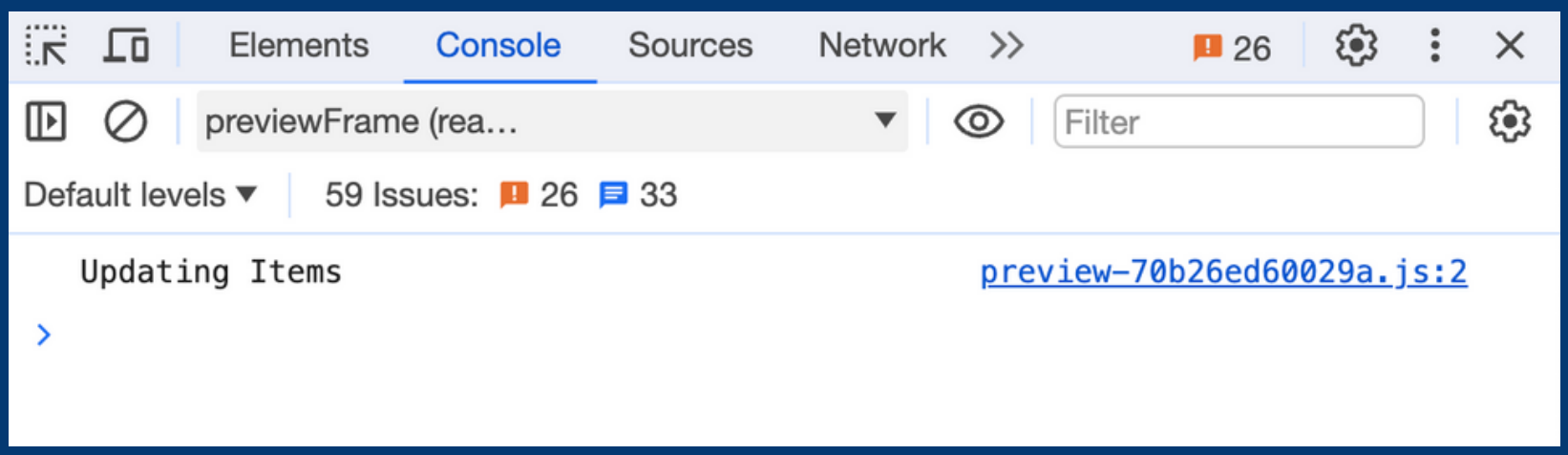
```jsx
import React, { useEffect, useState } from 'react';

export default function List({ getItems }) {
  const [items, setItems] = useState([]);

  useEffect(() => {
    setItems(getItems());
    console.log('Updating Items');
  }, [getItems]);

  return items.map((item, index) => <div key={index}>{item}</div>);
}
```

OUTPUT



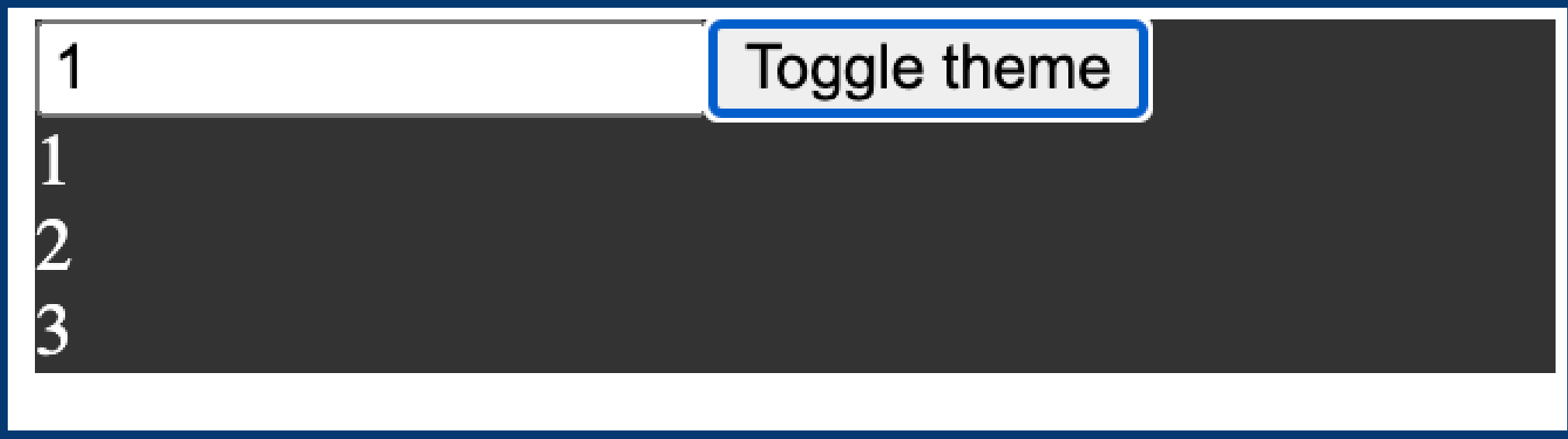
USEEFFECT CALL ON INPUT CHANGE

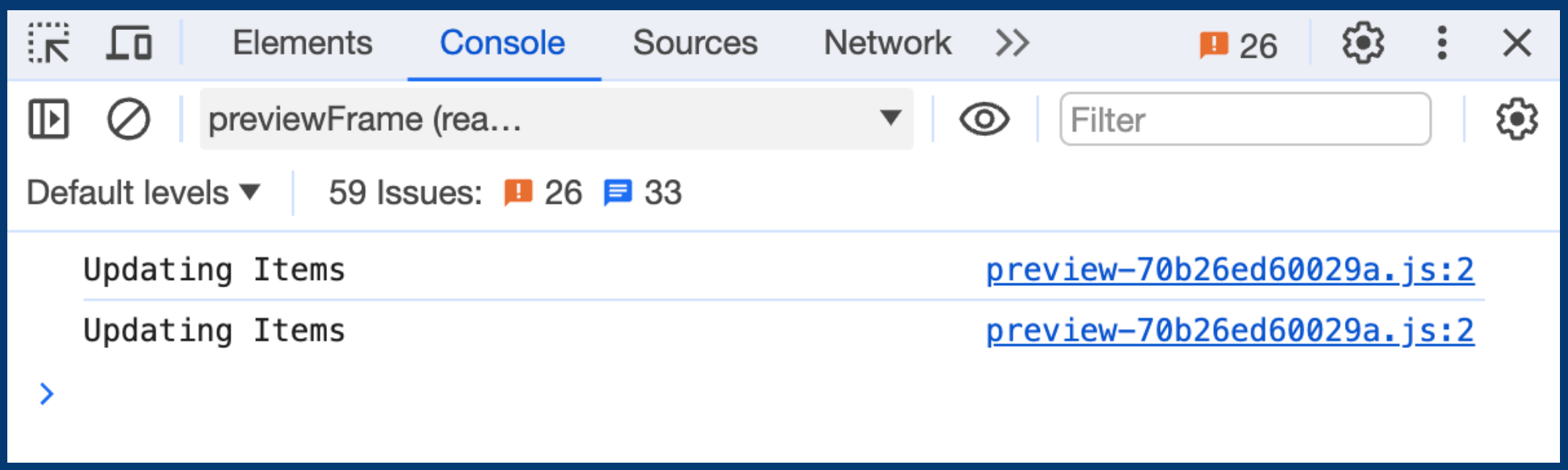
It is called one time on input changes.
this is asusual and working fine



MAIN PROBLEM

When we toggle on the theme button, UseEffect is called again, which means getItems are rendered without the input change.



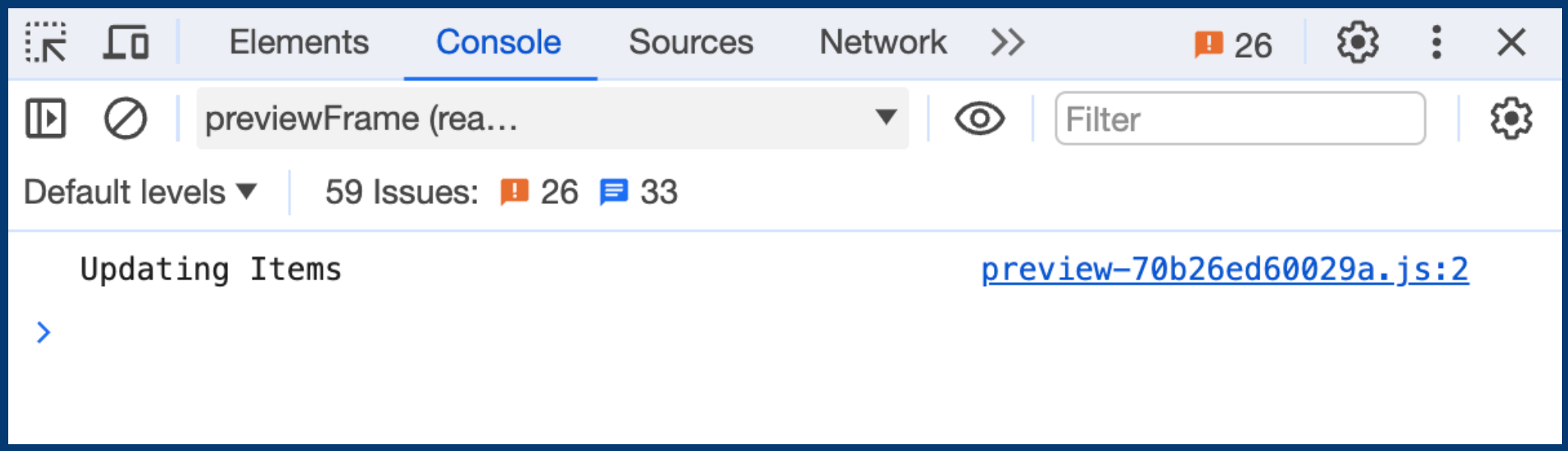


TO AVOID THIS PROBLEM

We are going to use useCallback hook that memoize functions in react that only changes if one of the dependencies has changed.

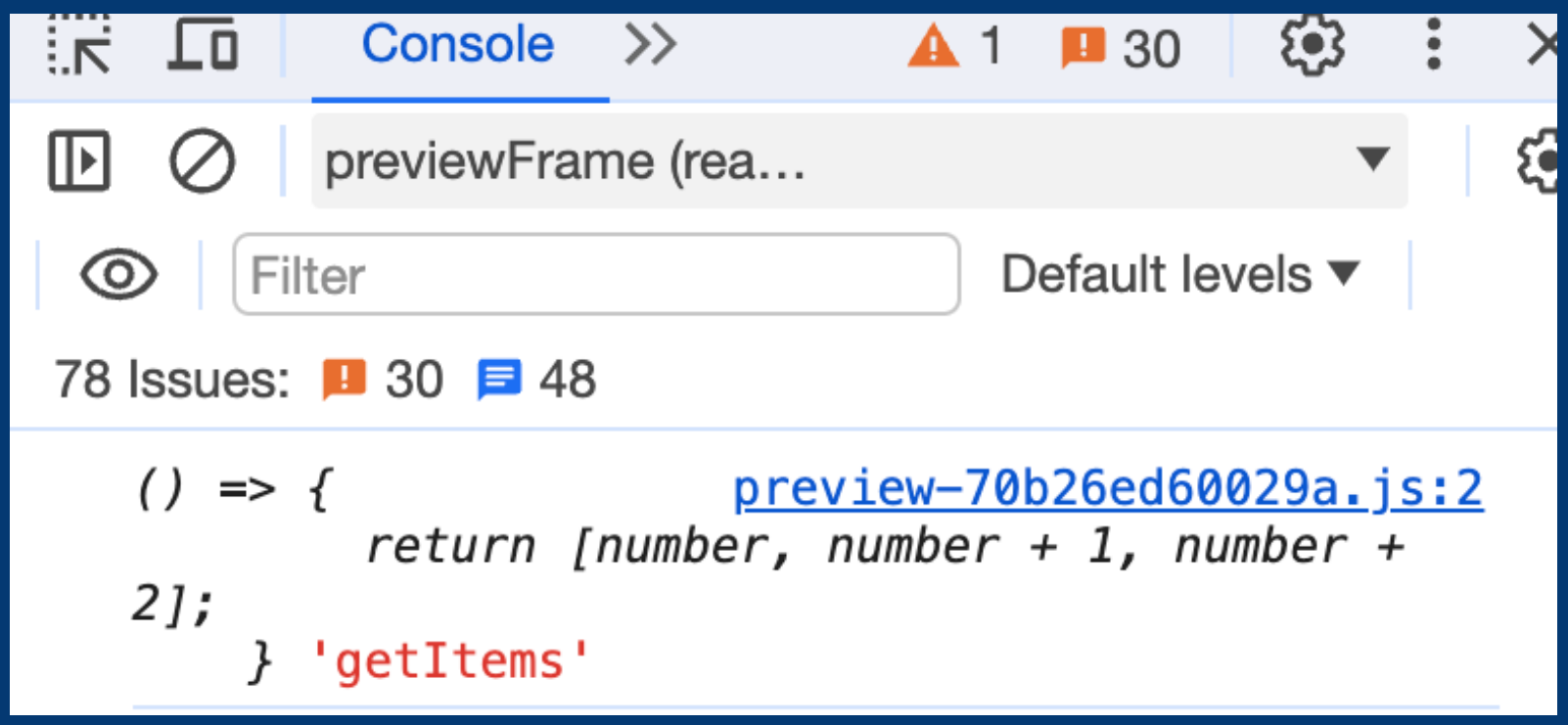
```jsx
import React, { useState ,useCallback} from 'react';
import List from './List.js';

export default function App() {
  const [number, setNumber] = useState(1);

  const [dark, setDark] = useState(false);

  const getItems = useCallback(() => {
    return [number, number + 1, number + 2];
  },[number]);

  const theme = {
    backgroundColor: dark ? '#333' : '#FFF',
    color: dark ? '#FFF' : '#333',
  };

  return (
    <div style={theme}>
      <input
        type="number"
        value={number}
        onChange={(e) => setNumber(parseInt(e.target.value))}
      />
      <button onClick={() => setDark((prevDark) => !prevDark)}>
        Toggle theme
      </button>
      <List getItems={getItems} />
    </div>
  );
}
```

Now useEffect will be triggered only on input change. As we added useCallback hook to trigger the function only when dependencies change



Now you may think the UseMemo also doing same ,NO **useMemo returns the value** ...but **useCallback returns the function itself**

```
const getItems = useCallback(() => {
  return [number, number + 1, number + 2];
}, [number]);
console.log(getItems, 'getItems');
```



In the previous **usememo** post, you can clearly notice it **returns a value**. I will link the useMemo post in the comments

Follow on

@Duvvuru Kishore