In the questions folder we have an app that loads 20000 user names on the page. There is an input field to add a surname to create a full name. How can you prevent unnecessary renders when the user types a value into the input field to create a full name? Do not use **useTransition** hook. For example, we don't want to render every value user enters until he/she stops entering the whole surname, also if user types jennifer for example and then starts deleting the value jennifer, how can you cause a delay so values such as **jennife, jennif** …. are not rendered until all letters are removed ? Remember, every time user deletes a letter from **jennifer**(assuming, user entered jennifer in the input field) by pressing **backspace**, the value will become:

jennife ➔ surname jennife should not be rendered

jennif ➔ surname jennif should not be rendered

- **jenni ➔ surname jenni should not be rendered**
- **jenn ➔ surname jenn should not be rendered**
- **jen ➔ surname jen should not be rendered**
- **je ➔ surname je should not be rendered**
- **j ➔ surname j should not be rendered**

All the values are deleted and now we can render the results

- **Note**: Use browser's performance tab and slowdown the CPU x 4 or x 6 times.

React 18 introduced **concurrent rendering**, a significant improvement in how React manages rendering and updates.

The most important property of Concurrent React is that rendering is interruptible. Pre React 18, we see that once the update starts rendering, nothing can interrupt it until the user can see the result on the screen. But in concurrent render, React may start rendering an update, pause in the middle, then continue later.

So concurrency is possible from React version 18 and we use **useTransition** and **useDeferredValue** hooks to apply concurrency.

What is the difference between useTransition vs useDeferredValue?

useTransition wraps the state updating code, **useDeferredValue** wraps a value that's affected by the state update.

useDeferredValue kicks in only on slow computer but debouncing / throttling using setTimeout is done on every computer in JavaScript and every time. When we wrap the value using useDeferredValue hook, React knows that this value is going to be deferred and it wont do anything until the application is not busy with anything else. And when it is done completing all other tasks it will process the deferred value.

To avoid the rendering of loop on every key press we can use useDefferedValue hook.