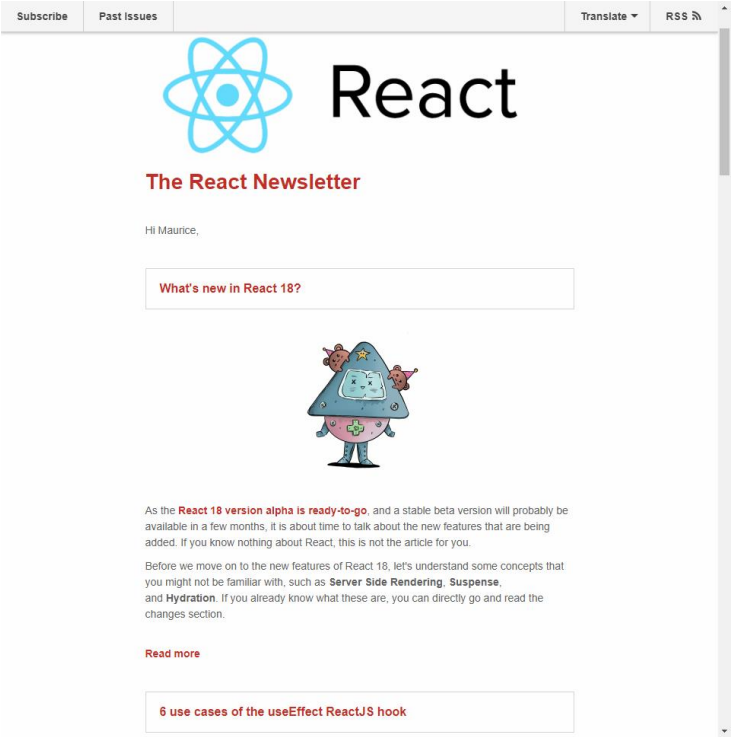
Concurrent Rendering Adventures in React 18

- Maurice de Beijer
- The Problem Solver
- Microsoft MVP
- Freelance developer/instructor
- Twitter: @mauricedb
- Web: http://www.TheProblemSolver.nl
- E-mail: maurice.de.beijer@gmail.com

The React Newsletter



Subscribe | Past Issues | Translate | RSS

React

The React Newsletter

Hi Maurice,

What's new in React 18?

As the **React 18 version alpha is ready-to-go**, and a stable beta version will probably be available in a few months, it is about time to talk about the new features that are being added. If you know nothing about React, this is not the article for you.

Before we move on to the new features of React 18, let's understand some concepts that you might not be familiar with, such as **Server Side Rendering**, **Suspense**, and **Hydration**. If you already know what these are, you can directly go and read the changes section.

Read more

6 use cases of the useEffect ReactJS hook

Course goal

- Learn about using **<Suspense />** today
  - Lazy loading of data
  - Nest and/or parallelize <Suspense /> components as needed
  - Error handling while suspended with an <ErrorBoundary />
- Learn about using **concurrent mode** tomorrow
  - Using createRoot() to render a React 18 application
  - The what and how of React concurrent mode
  - Orchestrating <Suspense /> boundaries using <SuspenseList />
  - Using startTransition() and/or useTransition() to defer work

Type it out by hand?

"Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now!"

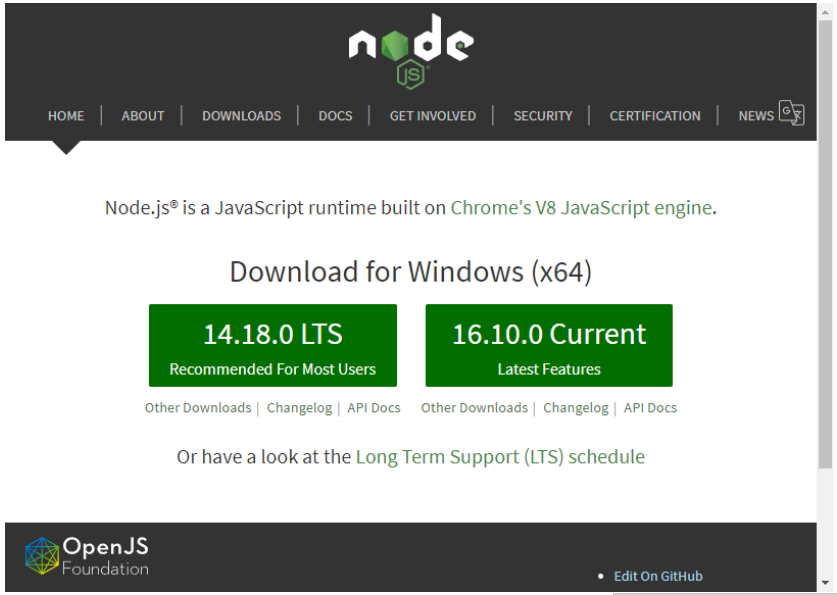
See you in the next video

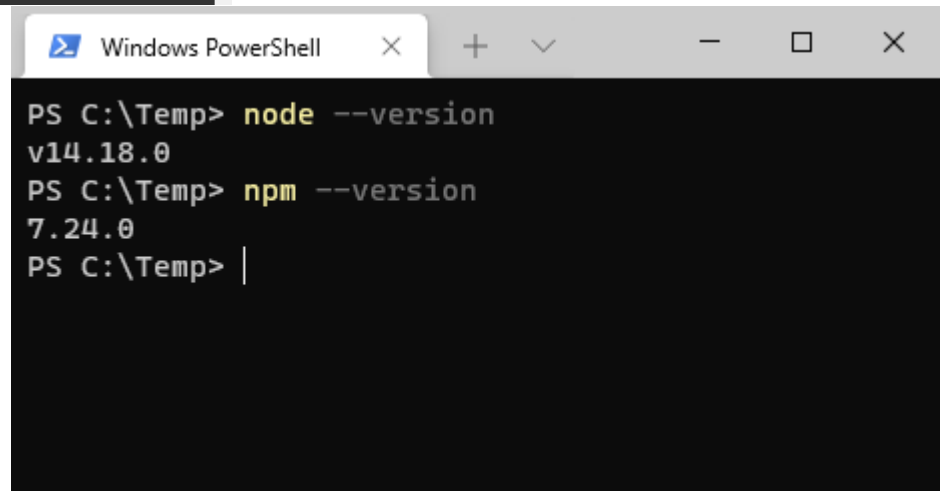
Prerequisites

Install Node & NPM

Install the GitHub repository

Install Node.js & NPM





Clone the GitHub Repository

Windows PowerShell

```
PS C:\Temp> git clone git@github.com:mauricedb/concurrent-rendering-adventures-in-react-18.git
Cloning into 'concurrent-rendering-adventures-in-react-18'...
remote: Enumerating objects: 228, done.
remote: Counting objects: 100% (228/228), done.
remote: Compressing objects: 100% (120/120), done.
Receiving objects: 100% (228/228)used 189 (delta 103), pack-reused 0 eceiving objects:  95% (217/228)
Receiving objects: 100% (228/228), 558.64 KiB | 2.10 MiB/s, done.
Resolving deltas: 100% (142/142), done.
PS C:\Temp>
```

Install NPM Packages

Windows PowerShell

```
PS C:\Temp\concurrent-rendering-adventures-in-react-18> npm ci
npm WARN deprecated flatten@1.0.3: flatten is deprecated in favor of utility frameworks such as lodash.
npm WARN deprecated @hapi/topo@3.1.6: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/bourne@1.3.2: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will break on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated sane@4.1.0: some dependency vulnerabilities fixed, support for node < 10 dropped, and newer ECMAScript syntax/features added
npm WARN deprecated @hapi/address@2.1.4: Moved to 'npm install @sideway/address'
npm WARN deprecated rollup-plugin-babel@4.4.0: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-babel.
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated babel-eslint@10.1.0: babel-eslint is now @babel/eslint-parser. This package will no longer receive updates.
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is
r details.
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is
r details.
npm WARN deprecated @hapi/hoek@8.5.1: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/joi@15.1.1: Switch to 'npm install joi'
npm WARN deprecated core-js@2.6.12: core-js@<3.3 is no longer maintained and not recommended for usage due to the number of issues. Because of th
ld cause a slowdown up to 100x even if nothing is polyfilled. Please, upgrade your dependencies to the actual version of core-js.

added 1988 packages, and audited 1989 packages in 12s

153 packages are looking for funding
  run `npm fund` for details

58 vulnerabilities (16 moderate, 42 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Temp\concurrent-rendering-adventures-in-react-18>
```

Following Along

TS App.tsx M TS UserDetails.tsx M ×

src > components > users > TS UserDetails.tsx > ...

```
11 export function UserDetails({ userId, movieId }: Props) {
12   return (
13     <div>
14       <h4 className="text-center mt-5">User details</h4>
15       <Suspense fallback={<Loading />}>
16         <AccountDetails userId={userId} />
17         <h4 className="text-center mt-5">Favorite movie</h4>
18         <MovieDetails movieId={movieId} />
19       </Suspense>
20     </div>
21   );
22 }
```

- Repository: https://github.com/mauricedb/concurrent-rendering-adventures-in-react-18
- Slides: http://www.theproblemsolver.nl/concurrent-rendering-adventures-in-react-18.pdf

See you in the next video

React 17

<Suspense />

<Suspense />

- Allows React to **"suspend"** rendering a component subtree
  - Used when a (grand) child component is not ready to be rendered
    - ECMAScript bundle containing the component isn't loaded yet
    - The data needed for a component to render isn't available yet
- The **"fallback" component** will be rendered instead
  - Replaces the complete children component tree
- Rendering is suspended when **a promise is thrown**
  - And resumed when the promise resolves

SWR and Suspense

- SWR is used in the application to **load data**
  - A convenient hook to fetch data
- SWR makes it easy to **start using suspense**
  - Add `suspense: true` to the `<SWRConfig />`

index.tsx

TS index.tsx M × | TS UserList.tsx M | TS AccountDetails.tsx M | TS MovieDetails.tsx M

src > TS index.tsx > ...

```
24  ReactDOM.render(
25    <ErrorBoundary FallbackComponent={ErrorFallback}>
      <Suspense fallback={<Loading />}>
27        <React.StrictMode>
28          <SWRConfig value={{ fetcher, suspense: true }}>
29            <App />
30          </SWRConfig>
31        </React.StrictMode>
32      </Suspense>
33    </ErrorBoundary>,
34    document.getElementById('root')
35  );
```

UserList.tsx

TS index.tsx M | TS UserList.tsx M × | TS AccountDetails.tsx M | TS MovieDetails.tsx M

src > components > users > TS UserList.tsx > ...

```
6   export function UserList() {
7     const [selectedUser, setSelectedUser] = useState<Account | null>(null);
8     const { data } = useSWR<Account[], Error>(
9       `${process.env.REACT_APP_API_BASE}/accounts`
10    );
11
12    return (
13      <div className="row">
14        <h2 className="text-center mt-5">Users</h2>
15
16        <div className="col-3 g-2">
17          <ul className="list-group">
18            {data?.map((user) => (
19              <li
```

AccountDetails.tsx

TS index.tsx M | TS UserList.tsx M | TS AccountDetails.tsx M × | TS MovieDetails.tsx M

src > components > users > TS AccountDetails.tsx > ...

```
9   export function AccountDetails({ userId }: Props) {
10    const { data } = useSWR<Account, Error>(
11      `${process.env.REACT_APP_API_BASE}/accounts/${userId}?sleep=1000`
12    );
13
14    const account = data!;
15
16    return (
17      <div className="row">
18        <LabelInput label="Firstname" value={account.firstname} readOnly />
19        <LabelInput label="Surname" value={account.surname} readOnly />
20        <LabelInput label="Email address" value={account.email} readOnly />
21      </div>
22    );
23  }
```

MovieDetails.tsx

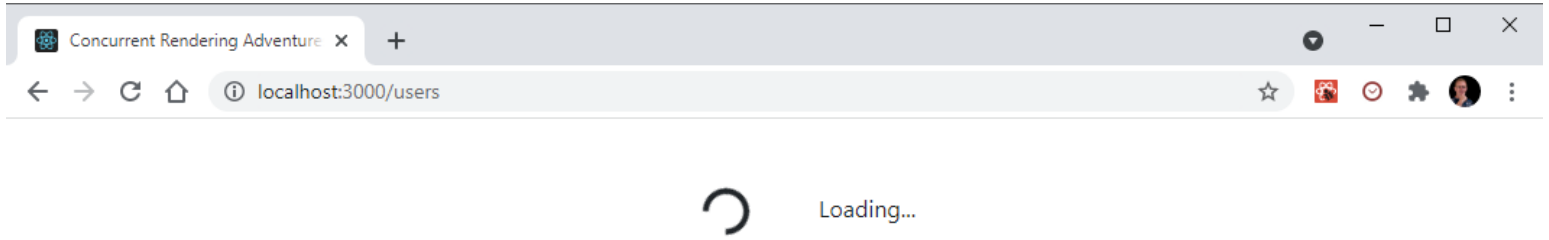
TS index.tsx M | TS UserList.tsx M | TS AccountDetails.tsx M | TS MovieDetails.tsx M ×

src > components > users > TS MovieDetails.tsx > ...

```
10  export function MovieDetails({ movieId }: Props) {
11    const { data } = useSWR<Movie, Error>(
12      `${process.env.REACT_APP_API_BASE}/top-rated-movies/${movieId}?sleep=500`
13    );
14    const movie = data!;
15
16    return (
17      <div className="row">
18        <LabelInput label="Title" value={movie.title} readOnly />
19        <LabelInput label="Release date" value={movie.release_date} readOnly />
20        <LabelTextarea
21          label="Overview"
22          value={movie.overview}
23          readOnly
24          rows={5}
25        />
26      </div>
27    );
28  }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Loading...

See you in the next video

<Suspense /> & Errors

<Suspense /> & Errors

- If a suspense resource **fails** to load an **error is thrown**
  - When requesting it during the render()
- Catch the error using an **ErrorBoundary**
  - Just like other runtime errors in React lifecycle functions
- Error boundaries **can be nested**
  - Just like suspense boundaries

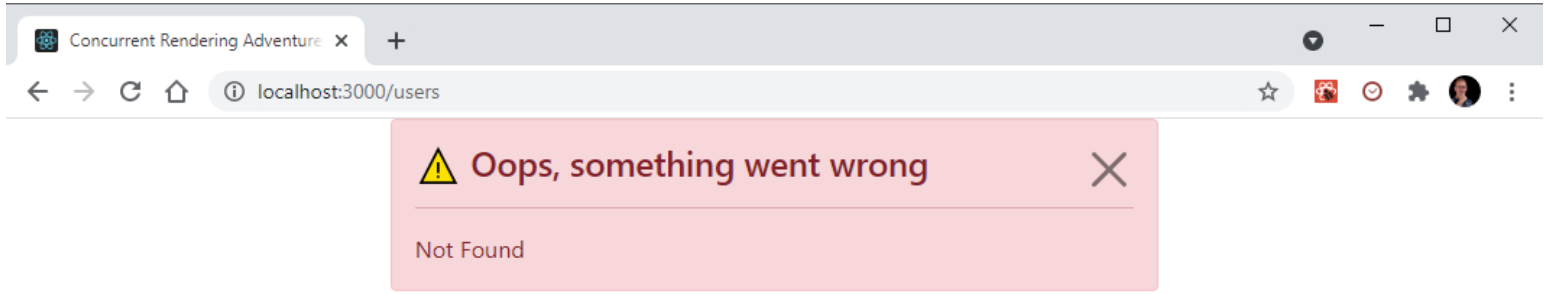
index.tsx

TS index.tsx ×

src > TS index.tsx > ...

```
24  ReactDOM.render(
      <ErrorBoundary FallbackComponent={ErrorFallback}>
26      <Suspense fallback={<Loading />}>
27        <React.StrictMode>
28          <SWRConfig value={{ fetcher, suspense: true }}>
29            <App />
30          </SWRConfig>
31        </React.StrictMode>
32      </Suspense>
33    </ErrorBoundary>,
34    document.getElementById('root')
35  );
```

The Result


Concurrent Rendering Adventure
localhost:3000/users
Oops, something went wrong
Not Found

UserDetails.tsx

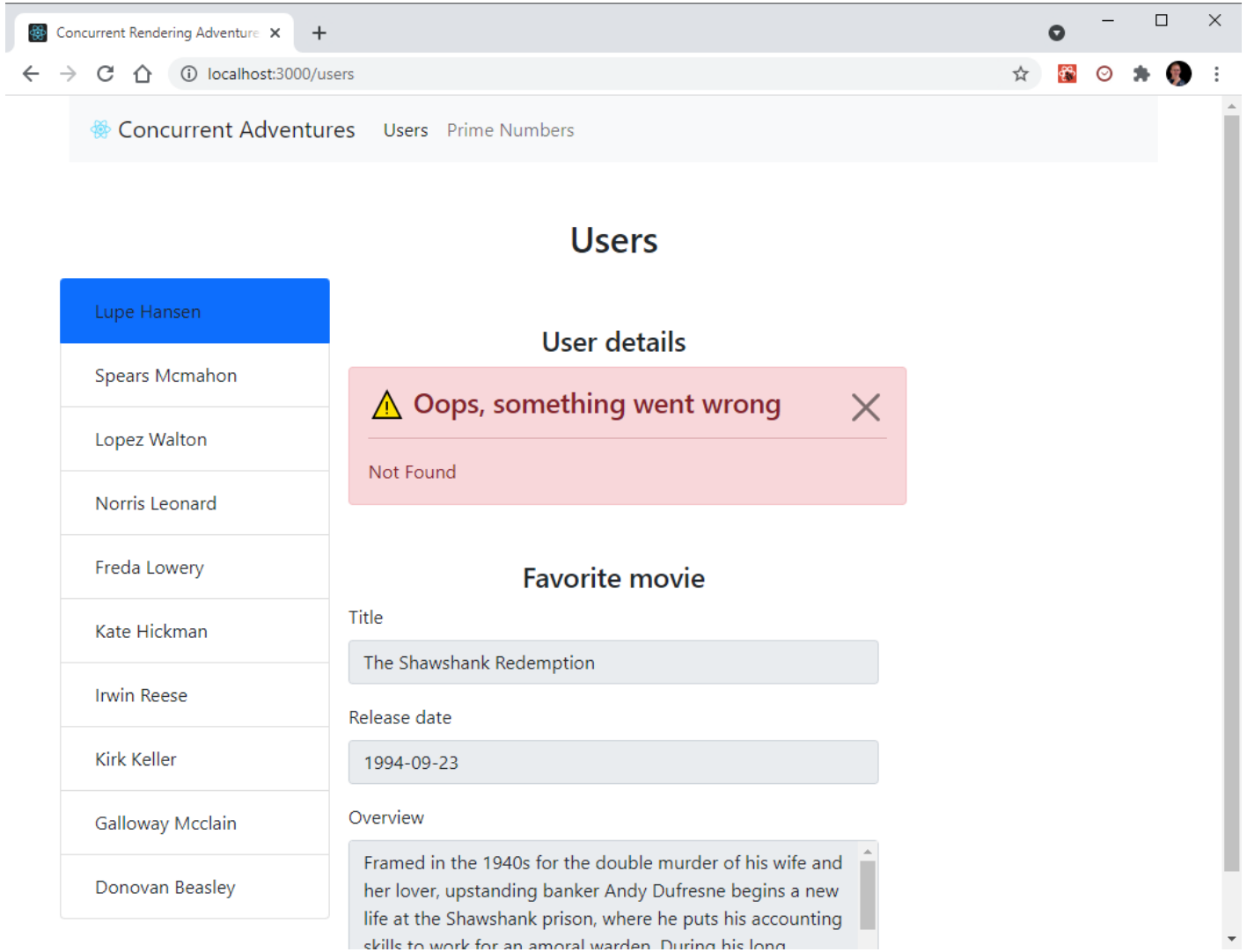
TS UserDetails.tsx M ×

src > components > users > TS UserDetails.tsx > ...

```
12  export function UserDetails({ userId, movieId }: Props) {
13    return (
14      <div>
15        <SuspenseList revealOrder="together">
16          <h4 className="text-center mt-5">User details</h4>
17          <Suspense fallback={<Loading />}>
18            <ErrorBoundary FallbackComponent={ErrorFallback}>
19              <AccountDetails userId={userId} />
20            </ErrorBoundary>
21          </Suspense>
22          <h4 className="text-center mt-5">Favorite movie</h4>
23          <Suspense fallback={<Loading />}>
24            <ErrorBoundary FallbackComponent={ErrorFallback}>
25              <MovieDetails movieId={movieId} />
26            </ErrorBoundary>
27          </Suspense>
28        </SuspenseList>
29      </div>
30    );
31  }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley
User details
Oops, something went wrong
Not Found
Favorite movie
Title
The Shawshank Redemption
Release date
1994-09-23
Overview
Framed in the 1940s for the double murder of his wife and her lover, upstanding banker Andy Dufresne begins a new life at the Shawshank prison, where he puts his accounting

See you in the next video

Nesting <Suspense />

Nesting <Suspense />

- Multiple suspense components **can be nested**
- React will use **the closest parent** <Suspense /> component
  - Very useful to control what part of the UI is replaced by a fallback

- 👉There is a behavior change in React 18 with null fallback 👈

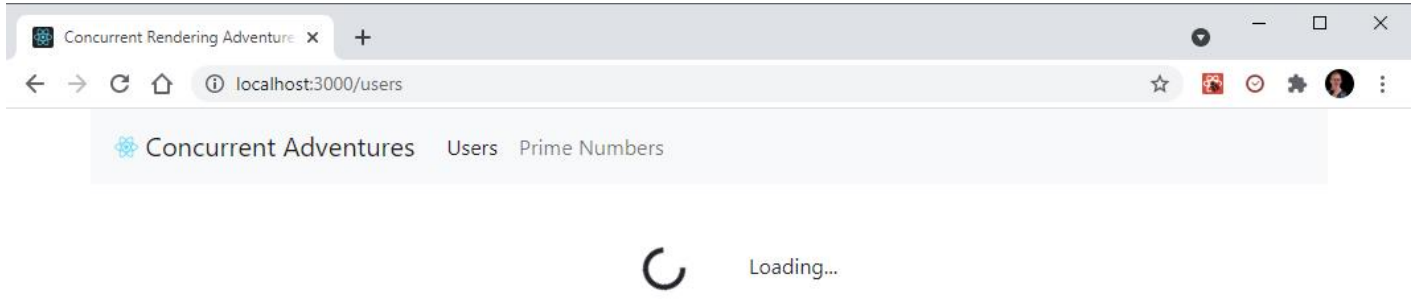
App.tsx

TS App.tsx M ×

src > TS App.tsx > ...

```
8   function App() {
9     return (
10      <div className="container">
11        <BrowserRouter>
12          <NavBar />
13          <Suspense fallback={<Loading />}>
14            <AppRoutes />
15          </Suspense>
16        </BrowserRouter>
17      </div>
18    );
19  }
```

The Result


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Loading...

MovieDetails.tsx

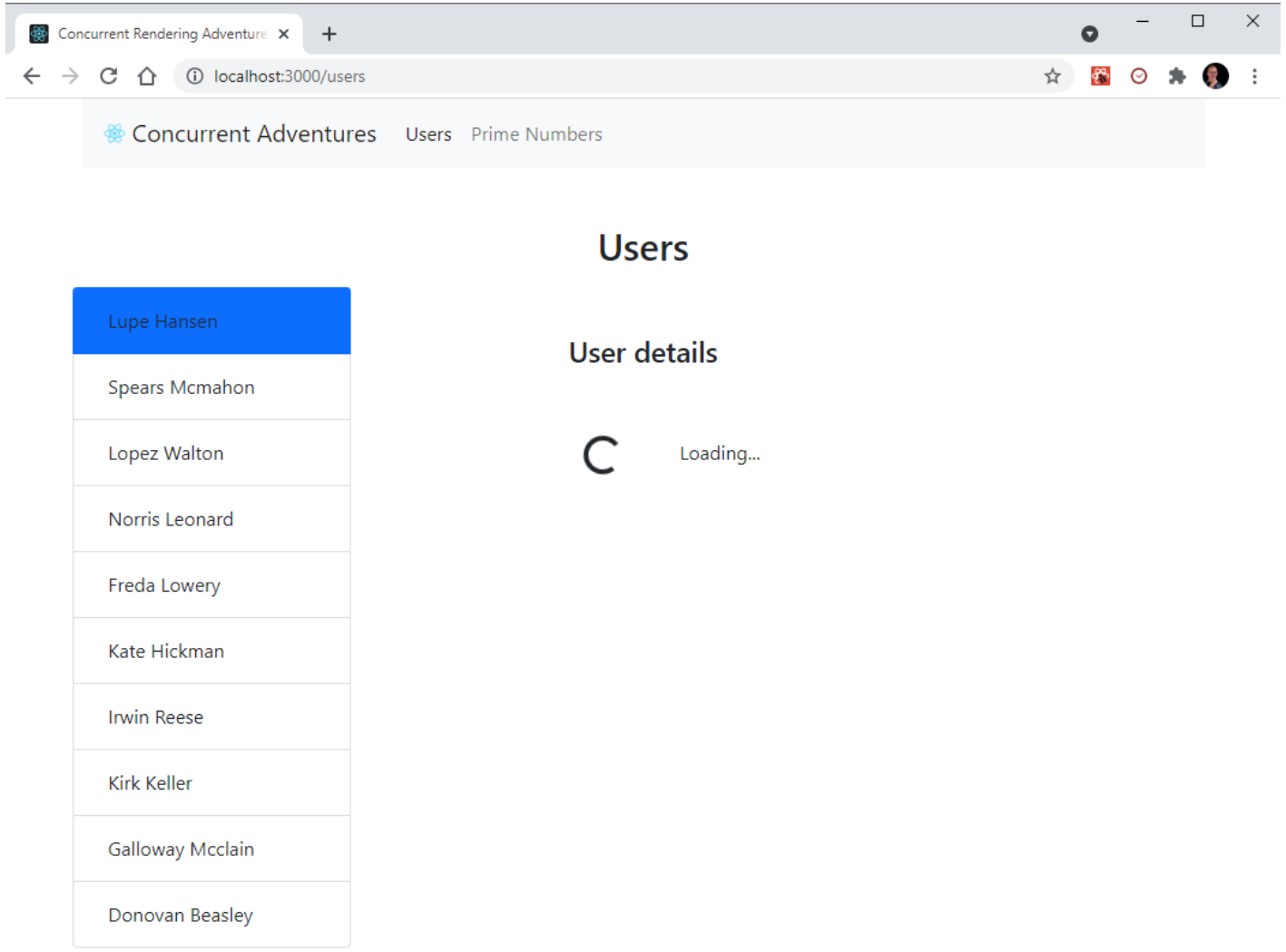
TS App.tsx M TS UserDetails.tsx M ×

src > components > users > TS UserDetails.tsx > ...

```
11  export function UserDetails({ userId, movieId }: Props) {
12    return (
13      <div>
14        <h4 className="text-center mt-5">User details</h4>
          <Suspense fallback={<Loading />}>
16          <AccountDetails userId={userId} />
17          <h4 className="text-center mt-5">Favorite movie</h4>
18          <MovieDetails movieId={movieId} />
19        </Suspense>
20      </div>
21    );
22  }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley
User details
Loading...

See you in the next video

Parallel <Suspense />

Parallel <Suspense />

- Multiple suspense boundaries can **suspend in parallel**
  - React will suspend them all and show multiple fallback components
- If you want to **render a component while others are still loading**
- **Multiple suspending components** in a **single <Suspense/>** is also fine
  - Will resume when all resource promises are resolved

MovieDetails.tsx

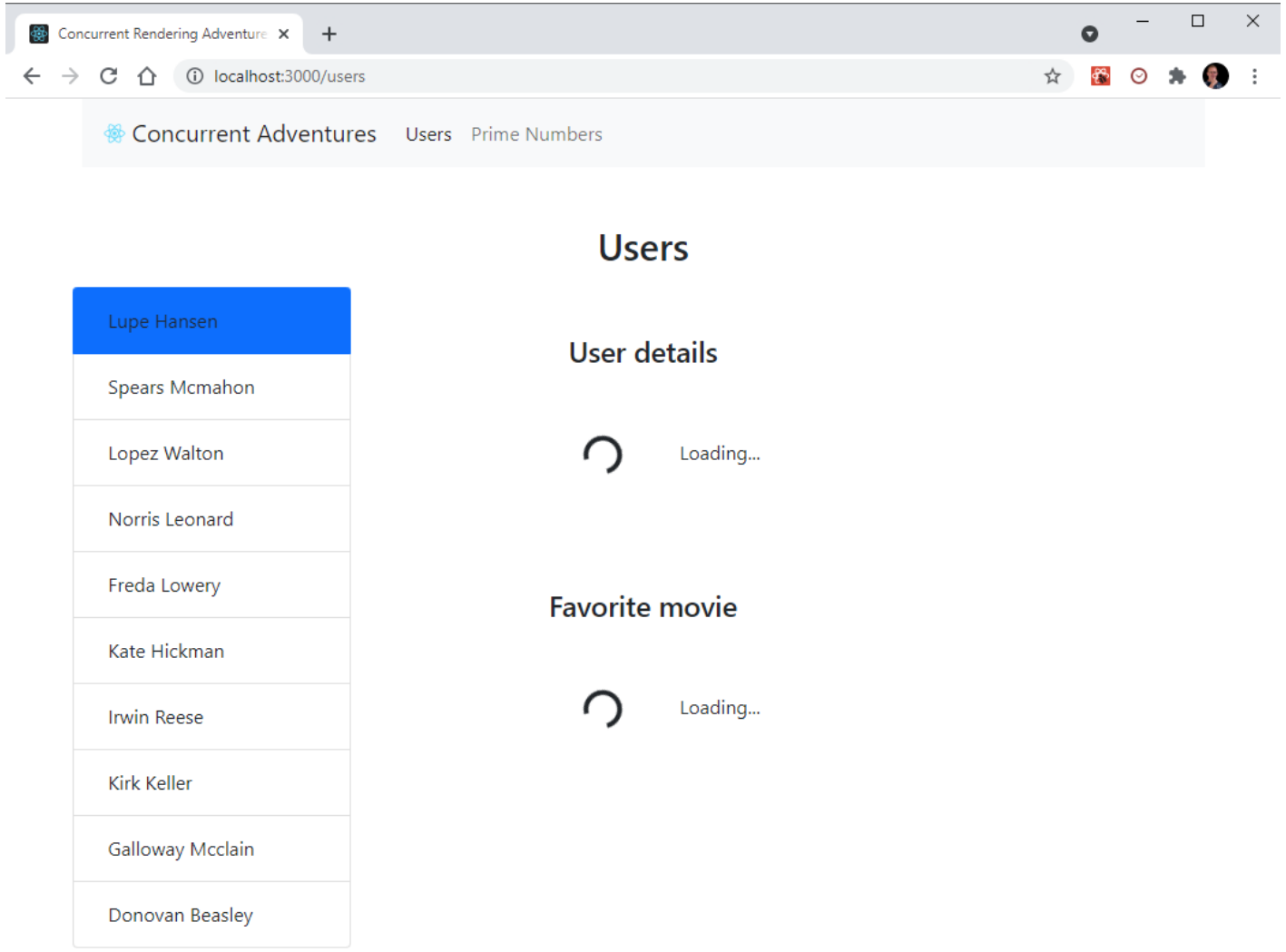
TS UserDetails.tsx M ×

src > components > users > TS UserDetails.tsx > ...

```
11  export function UserDetails({ userId, movieId }: Props) {
12    return (
13      <div>
14        <h4 className="text-center mt-5">User details</h4>
          <Suspense fallback={<Loading />}>
16          <AccountDetails userId={userId} />
17        </Suspense>
18        <h4 className="text-center mt-5">Favorite movie</h4>
          <Suspense fallback={<Loading />}>
20          <MovieDetails movieId={movieId} />
21        </Suspense>
22      </div>
23    );
24  }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley
User details
Loading...
Favorite movie
Loading...

See you in the next video

React 18

React 18

- React 18 is still in alpha/preview version right now
  - Daily publish to NPM using the **@next** and the **@alpha** tags
- `npm install react@next react-dom@next --force`

package.json

package.json

package.json > ...

```
5   "dependencies": {
6     "bootstrap": "^5.1.1",
7     "react": "^18.0.0-alpha-bdd6d5064-20211001",
8     "react-dom": "^18.0.0-alpha-bdd6d5064-20211001",
9     "react-error-boundary": "^3.1.3",
10    "react-router-dom": "^6.0.0-beta.5",
11    "swr": "^1.0.1",
12    "web-vitals": "^1.1.2"
13  },
```

index.tsx

{} package.json M TS index.tsx M ×

src > TS index.tsx > ...

```
ReactDOM.createRoot(document.getElementById('root')!).render(
  <ErrorBoundary FallbackComponent={ErrorFallback}>
    <Suspense fallback={<Loading />}>
      <React.StrictMode>
        <SWRConfig value={{ fetcher, suspense: true }}>
          <App />
        </SWRConfig>
      </React.StrictMode>
    </Suspense>
  </ErrorBoundary>
);
```

See you in the next video

New hooks

New hooks

- useDeferredValue()
  - Returns a deferred version of the value that may lag behind
- useTransition()
  - Avoid undesirable states when waiting for content
- useMutableSource()
  - Enables React components to safely and efficiently read from a mutable external source in Concurrent Mode
  - Avoids tearing
- useOpaqueIdentifier()
  - Can be used to generate unique ID's in an SSR-safe way

useOpaqueIdentifier()

useOpaqueIdentifier()

- Can be used to generate unique ID's in a SSR-safe way
- 👉 Still prefixed with `unstable_` 👈

LabelInput.tsx

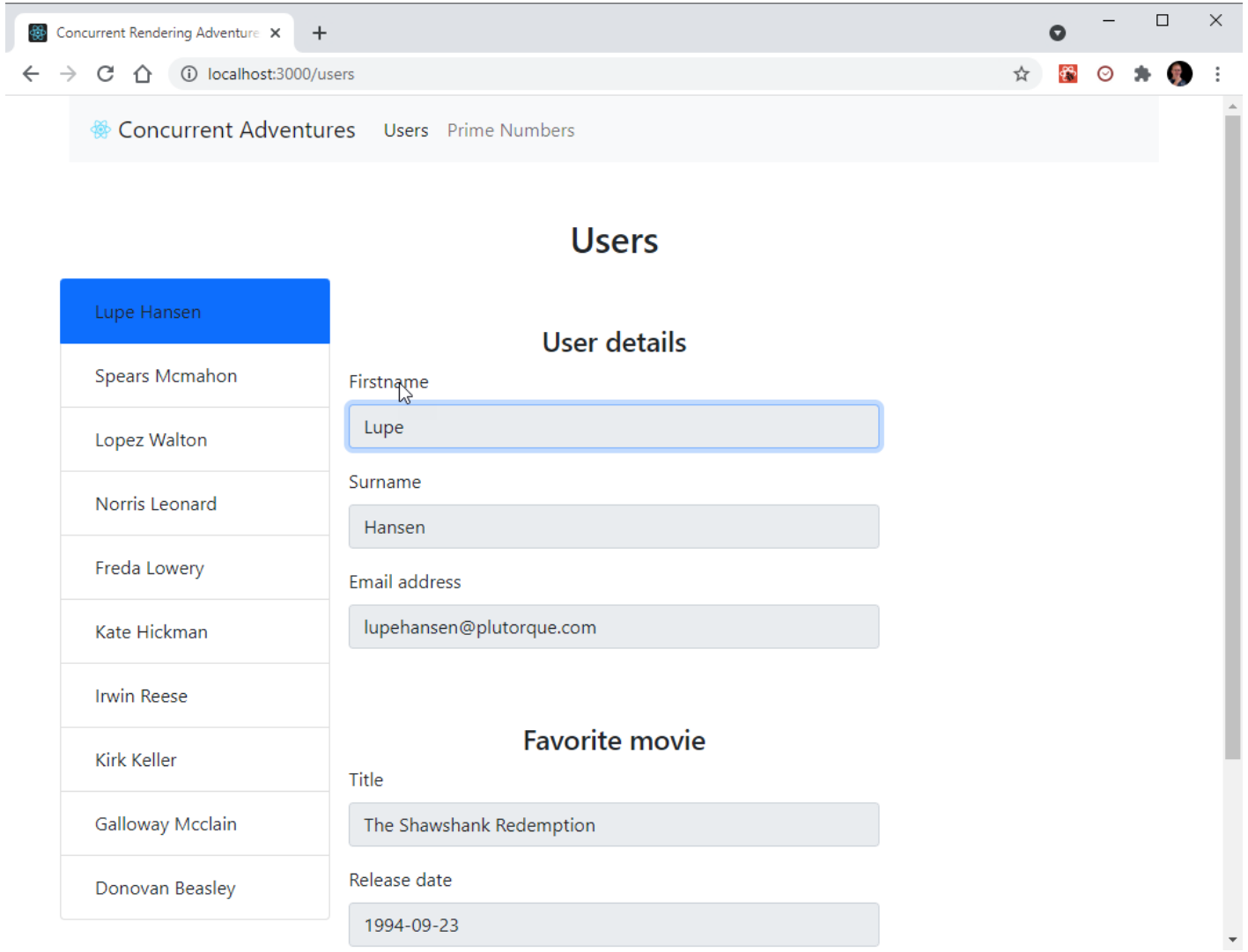
TS LabelInput.tsx M × TS LabelTextarea.tsx M

src > components > users > TS LabelInput.tsx > ...

```
9   export function LabelInput({ label, value, ...rest }: Props) {
      const id = unstable_useOpaqueIdentifier();
11
12    return (
13      <div className="mb-3">
14        <label htmlFor={id} className="form-label">
15          {label}
16        </label>
17        <input id={id} className="form-control" value={value} {...rest} />
18      </div>
19    );
20  }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley
User details
Firstname
Lupe
Surname
Hansen
Email address
lupehansen@plutorque.com
Favorite movie
Title
The Shawshank Redemption
Release date
1994-09-23

See you in the next video

Using <SuspenseList />

Orchestrating <Suspense /> boundaries

Using <SuspenseList />

- **<SuspenseList />** will let you control how multiple **<Suspense />** components **render their fallback**
  - The order in which child components show when ready
  - If multiple child fallbacks components are displayed

UserDetails.tsx

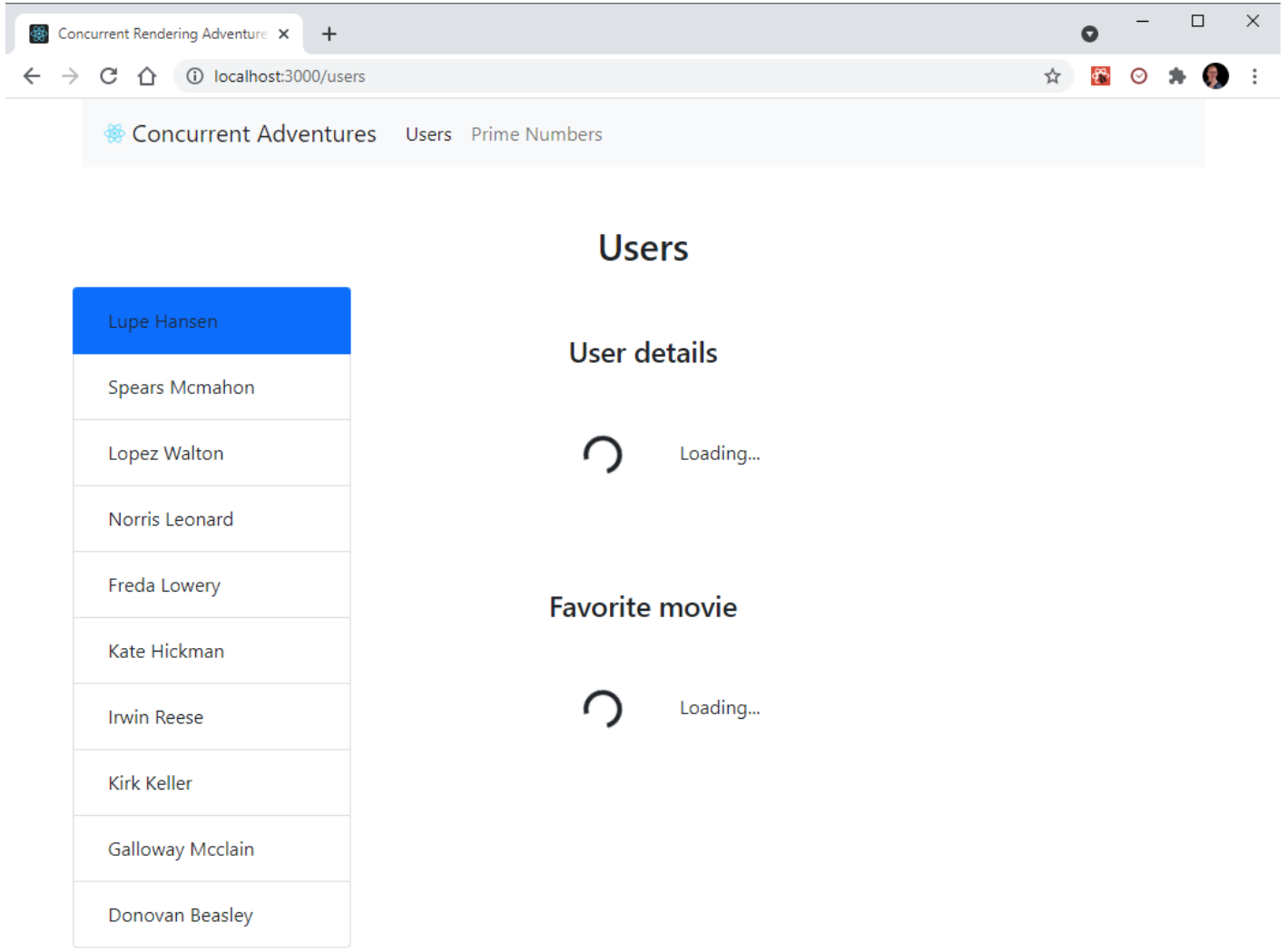
TS UserDetails.tsx M × TS UserList.tsx M

src > components > users > TS UserDetails.tsx > ...

```
11  export function UserDetails({ userId, movieId }: Props) {
12    return (
13      <div>
        <SuspenseList revealOrder="together">
15        <h4 className="text-center mt-5">User details</h4>
16        <Suspense fallback={<Loading />}>
17          <AccountDetails userId={userId} />
18        </Suspense>
19        <h4 className="text-center mt-5">Favorite movie</h4>
20        <Suspense fallback={<Loading />}>
21          <MovieDetails movieId={movieId} />
22        </Suspense>
23      </SuspenseList>
24      </div>
25    );
26  }
```

The Result

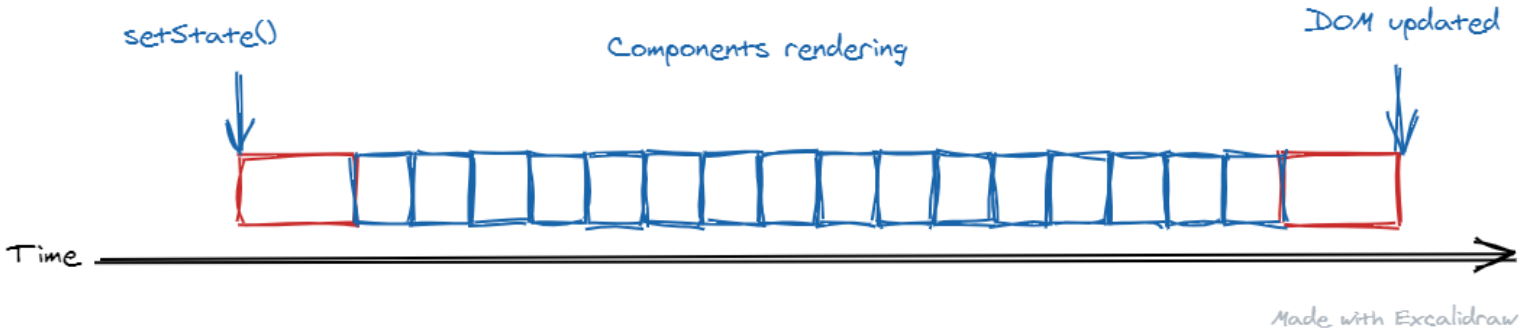

MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley
User details
Loading...
Favorite movie
Loading...

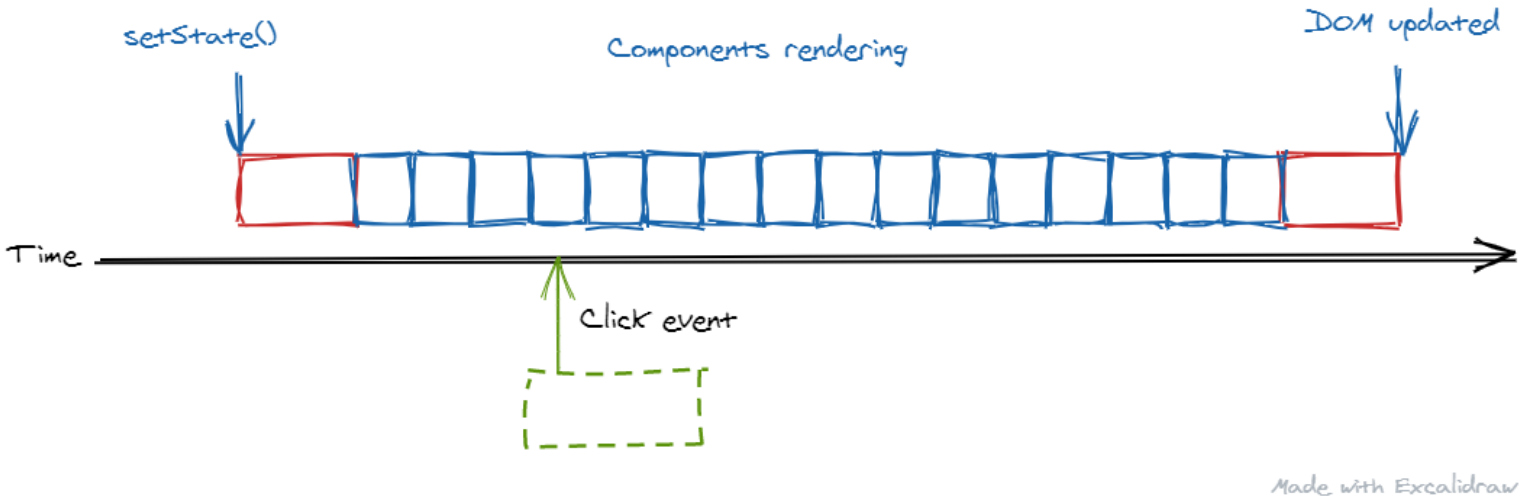
See you in the next video

Concurrent Mode

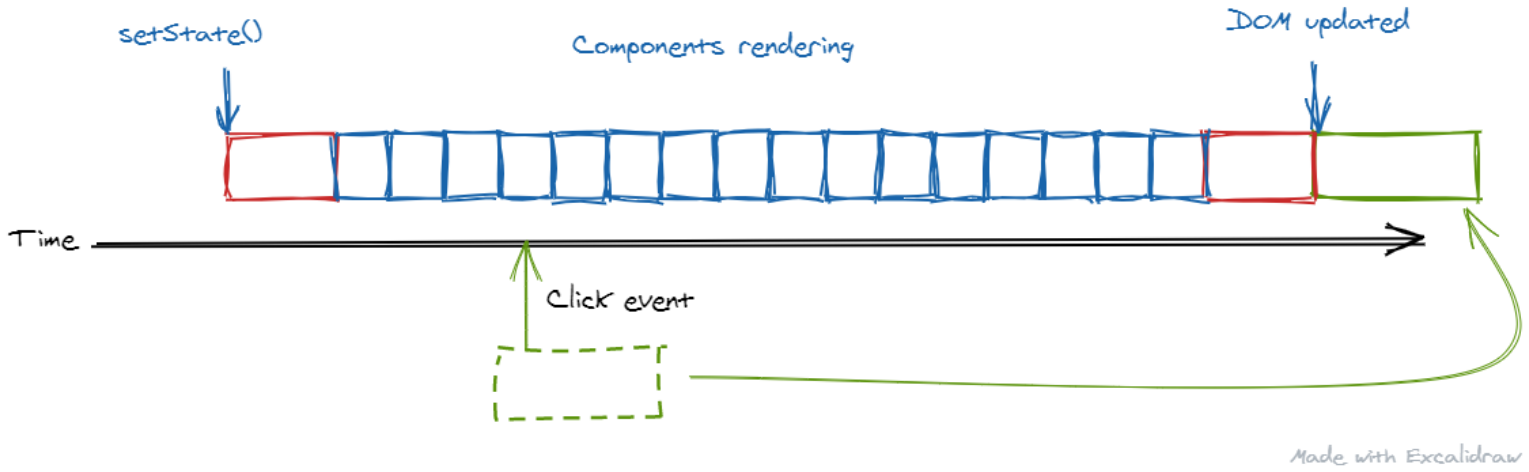
React 17 rendering components



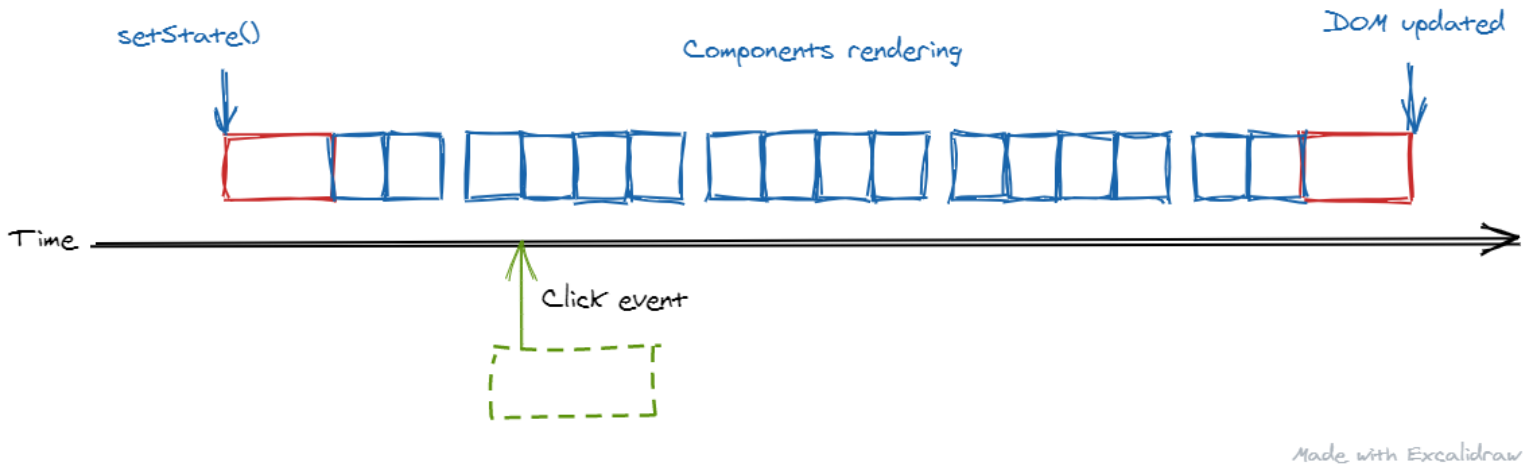
User click event



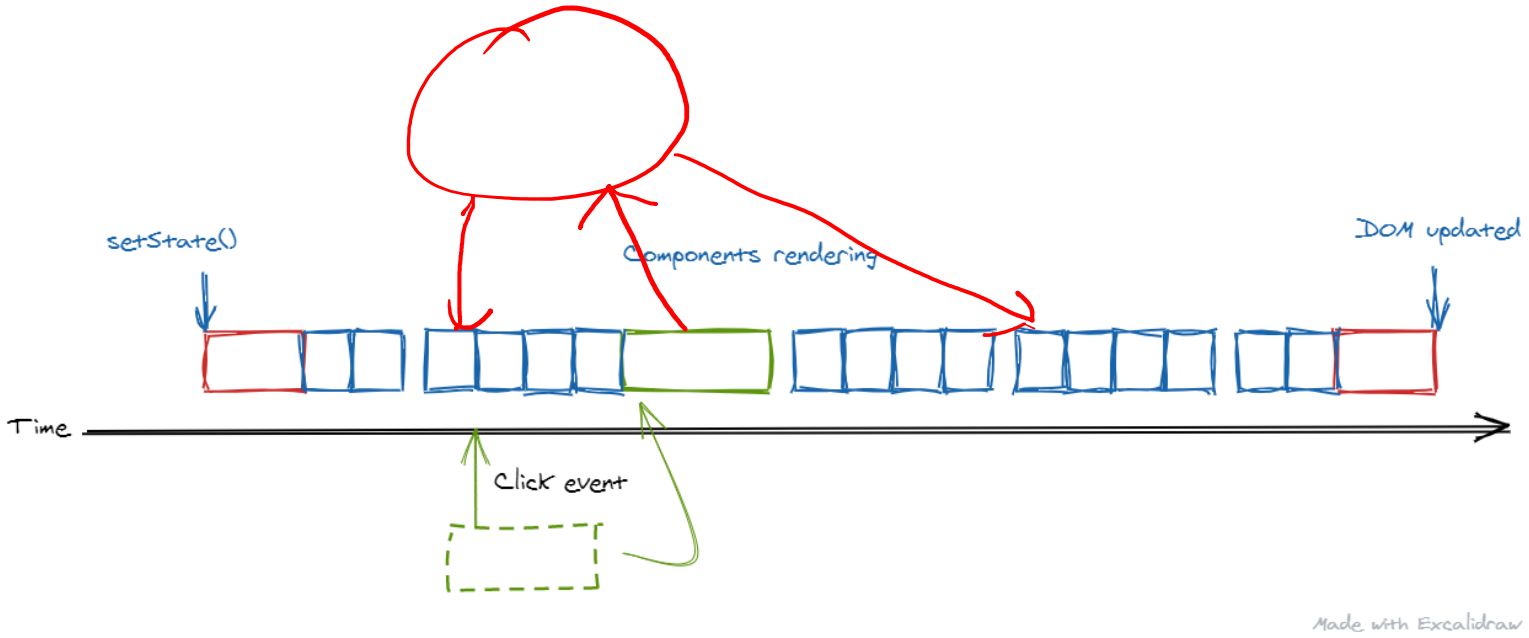
Event running



React 18 Concurrent mode



Event running with concurrent mode



See you in the next video

startTransition()

To defer lower priority work

PrimeNumbers.tsx

TS PrimeNumbers.tsx M ×

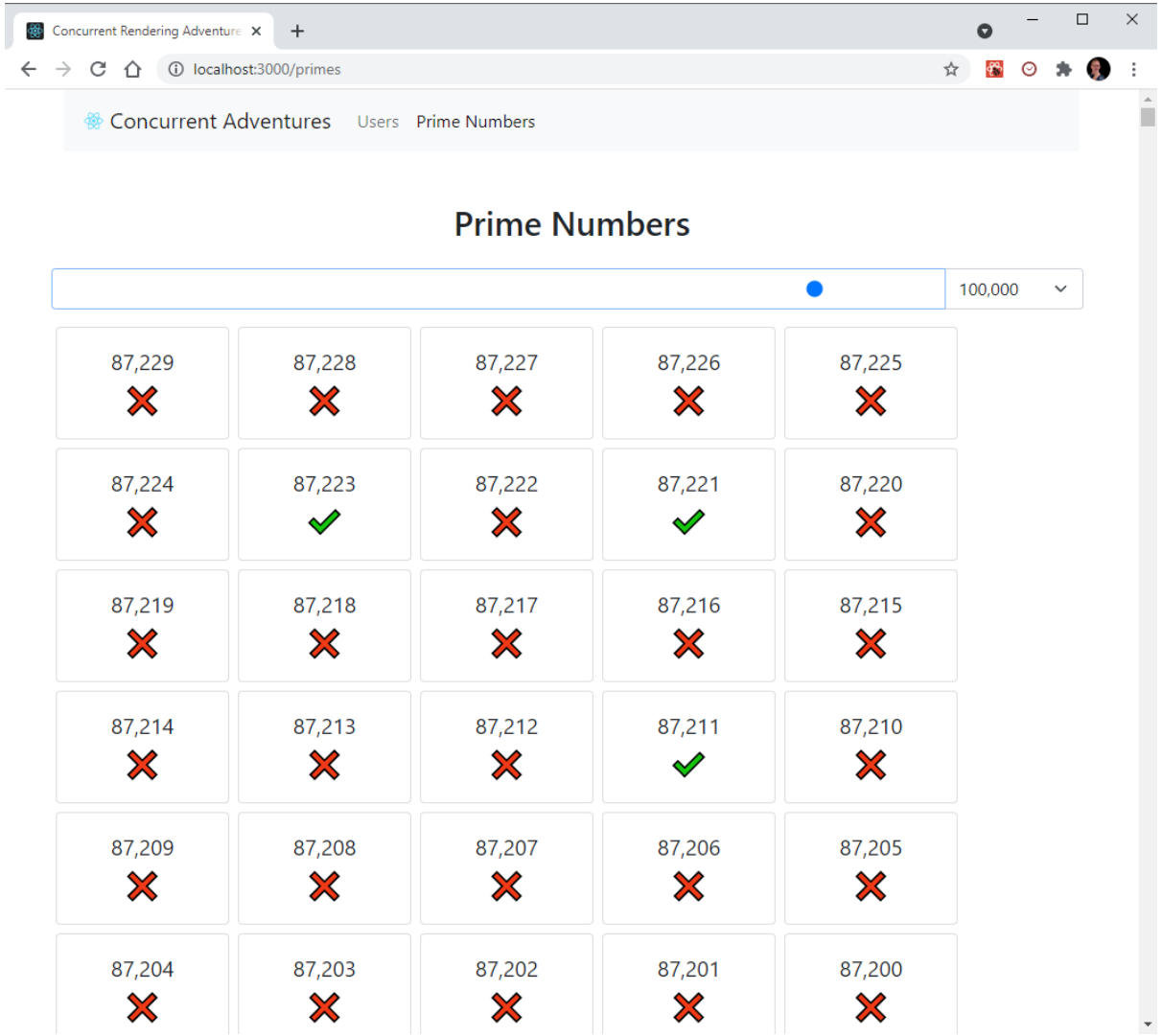
src > components > primes > TS PrimeNumbers.tsx > ...

You, 2 minutes ago | 1 author (You)

```
1  { startTransition, useState } from 'react';
2  import { CheckNumber } from './CheckNumber';
3  import { PrimeRange } from './PrimeRange';
4
5  const defaultValue = 250;
6
7  export function PrimeNumbers() {
8    const [maxPrime, setMaxPrime] = useState(defaultValue);
9    const values = new Array(maxPrime).fill(null);
10
11   return (
12     <div className="row">
13       <h2 className="text-center mt-5">Prime Numbers</h2>
14       <PrimeRange
15         defaultValue={defaultValue}
16         onChange={(value) => startTransition(() => setMaxPrime(value))}
17       />
18
19       <div className="row row-cols-auto g-2">
20         {values
21           .filter((_, index) => index < 10_000)
22           .map((_, index) => {
23             return <CheckNumber key={index} value={maxPrime - index} />;
24           })}
25       </div>
26     </div>
27   );
28 }
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/primes
Concurrent Adventures
Users
Prime Numbers
Prime Numbers
100,000
87,229
87,228
87,227
87,226
87,225
87,224
87,223
87,222
87,221
87,220
87,219
87,218
87,217
87,216
87,215
87,214
87,213
87,212
87,211
87,210
87,209
87,208
87,207
87,206
87,205
87,204
87,203
87,202
87,201
87,200

See you in the next video

Using useTransition()

To defer lower priority work and know about pending updates

Using useTransition()

- The **useTransition() hook** can be used to control how React renders when components suspend
  - Prevent the fallback component being rendered immediately
- The **new components will be rendered** when:
  - Their resources are ready
  - The timeout is expired
- The "old" UI can use the **isPending** state when rendering

PrimeNumbers.tsx

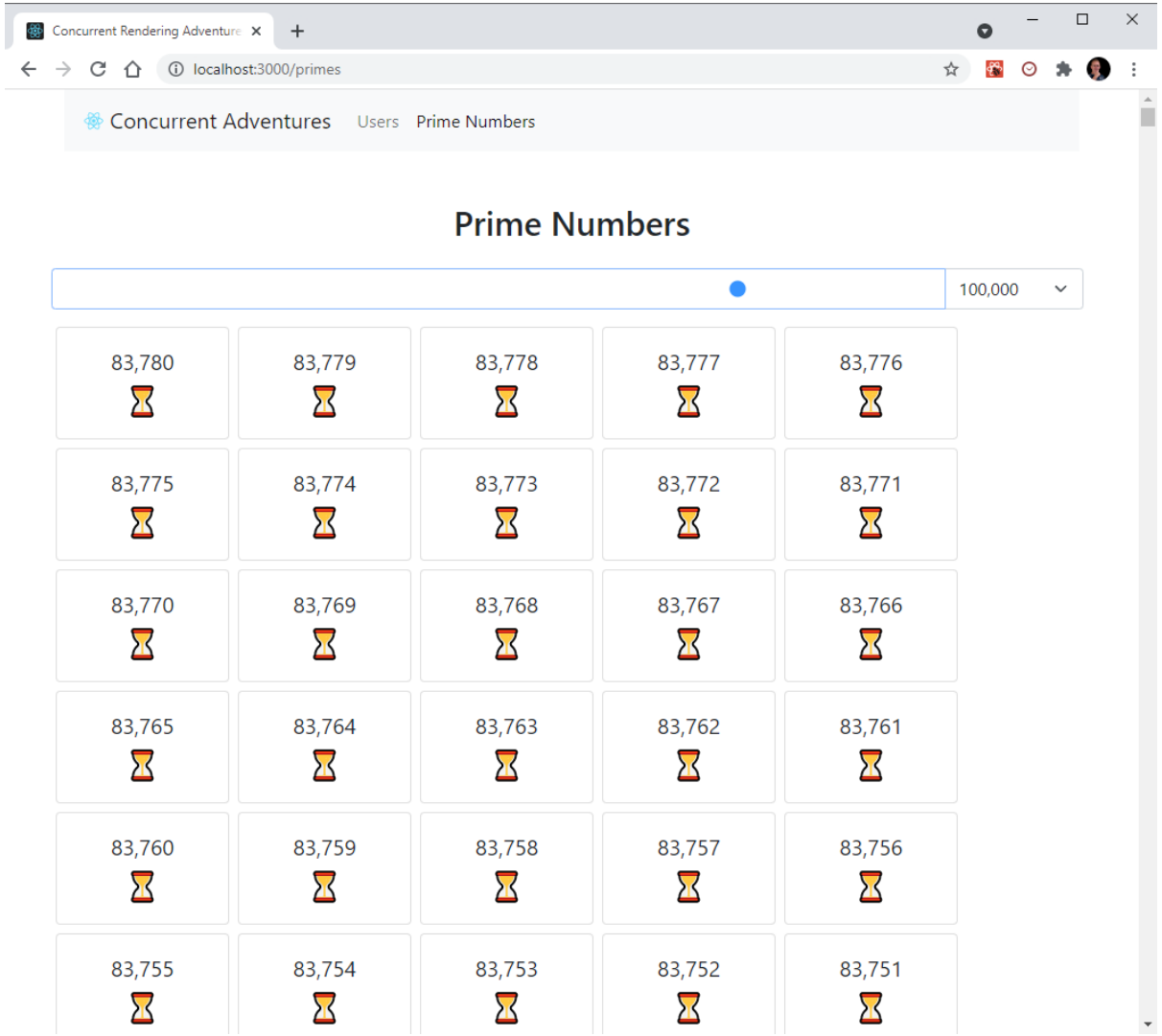
TS PrimeNumbers.tsx M × TS CheckNumber.tsx M

src > components > prime > TS PrimeNumbers.tsx > ...

```
1  { useTransition, useState } from 'react';
2  import { CheckNumber } from './CheckNumber';
3  import { PrimeRange } from './PrimeRange';
4
5  const defaultValue = 250;
6
7  export function PrimeNumbers() {
8    const [isPending, startTransition] = useTransition();
9    const [maxPrime, setMaxPrime] = useState(defaultValue);
10   const values = new Array(maxPrime).fill(null);
11
12   return (
13     <div className="row">
14       <h2 className="text-center mt-5">Prime Numbers</h2>
15       <PrimeRange
16         defaultValue={defaultValue}
17         onChange={(value) => startTransition(() => setMaxPrime(value))}
18       />
19
20       <div className="row row-cols-auto g-2">
21         {values
22           .filter((_, index) => index < 10_000)
23           .map((_, index) => {
24             return (
25               <CheckNumber
26                 key={index}
27                 value={maxPrime - index}
28                 isPending={isPending}
29               />
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/primes
Concurrent Adventures
Users
Prime Numbers
Prime Numbers
100,000
83,780
83,779
83,778
83,777
83,776
83,775
83,774
83,773
83,772
83,771
83,770
83,769
83,768
83,767
83,766
83,765
83,764
83,763
83,762
83,761
83,760
83,759
83,758
83,757
83,756
83,755
83,754
83,753
83,752
83,751

startTransition() vs useTransition()

startTransition()

- Can be used anywhere
- No additional renders

useTransition()

- Needs to be used in a functional component
- One additional render with `isPending`

<Suspense /> & Transitions

<Suspense />
&
Transitions

- Suspense can **cooperate** with a `startTransition()`
  - The new UI isn't visible until the transition completes

UserList.tsx

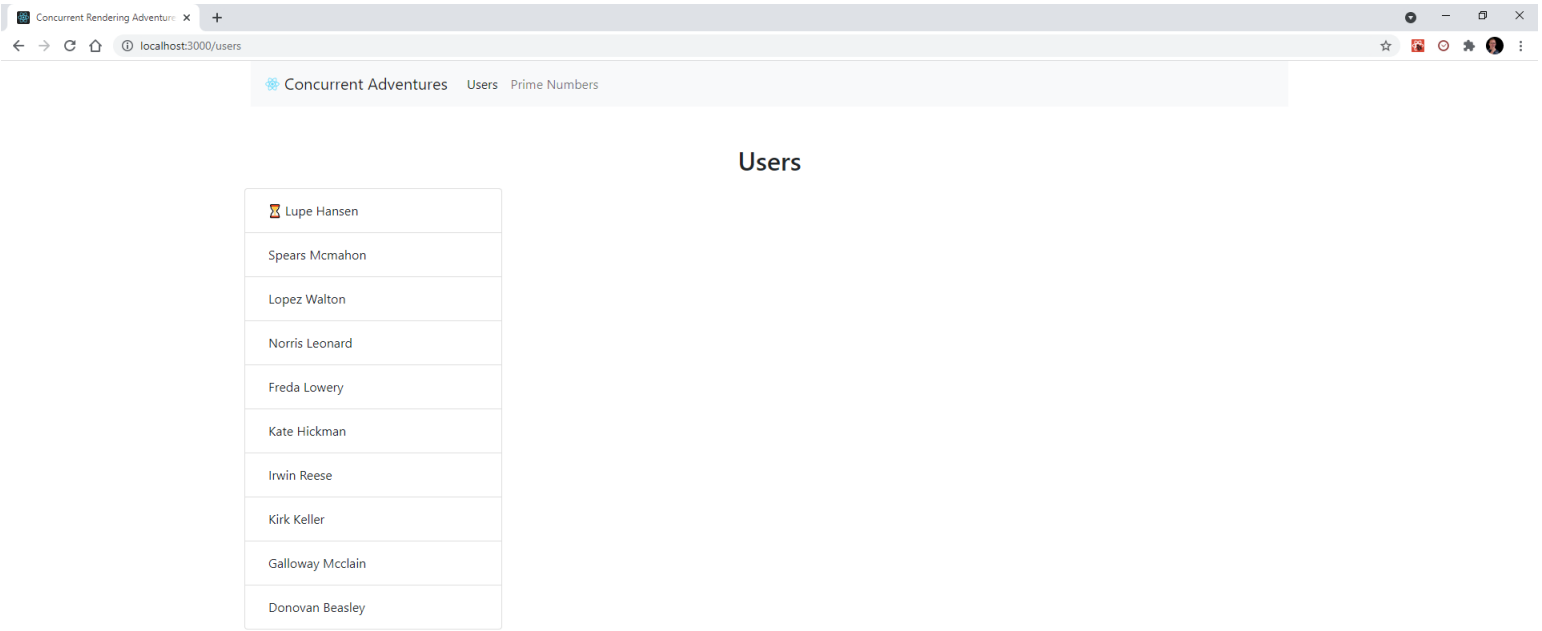
TS UserList.tsx M ×

src > components > users > TS UserList.tsx > ...

```
14    return (
15      <div className="row">
16        <h2 className="text-center mt-5">Users</h2>
17
18        <div className="col-3 g-2">
19          <ul className="list-group">
20            {data?.map((user) => (
21              <li
22                key={user.id}
23                className={
24                  'list-group-item' + (user === selectedUser ? ' active' : '')
25                }
26              >
27                <button
28                  className="btn shadow-none"
29                  onClick={() => {
30                    setSelectedUserId(user.id);
31                    startTransition(() => {
32                      setSelectedUser(user);
33                    });
34                  }}
35                >
36                  {isPending && selectedUserId === user.id && '⏳ '}
37                  {user.firstname}
38                   
39                  {user.surname}
40                </button>
41              </li>
42            ))}
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/users
Concurrent Adventures
Users
Prime Numbers
Users
Lupe Hansen
Spears Mcmahon
Lopez Walton
Norris Leonard
Freda Lowery
Kate Hickman
Irwin Reese
Kirk Keller
Galloway Mcclain
Donovan Beasley

See you in the next video

Using useDeferredValue()

Using useDeferredValue()

- The **useDeferredValue() hook** can be used create a deferred version of the value that may "lag behind"
  - Can prevent extra re-renders of expensive components
- https://reactjs.org/docs/concurrent-mode-reference.html#usedeferredvalue

PrimeRange.tsx

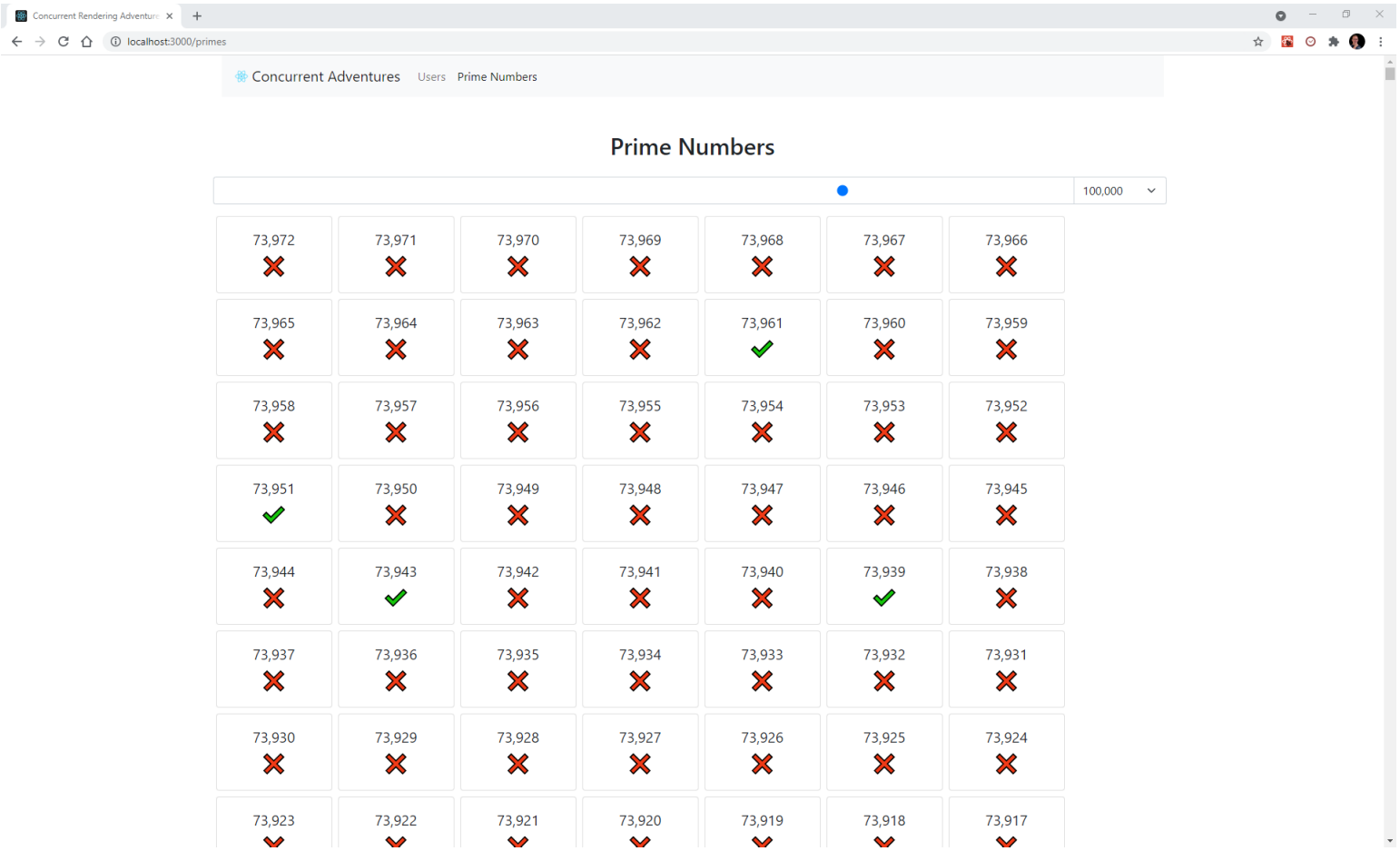
TS PrimeRange.tsx M ×

src > components > primes > TS PrimeRange.tsx > ...

```
12  export function PrimeRange({ defaultValue, onChange }: Props) {
13    const [maxPrimeRange, setMaxPrimeRange] = useState(defaultValue);
14
15    const deferredMaxPrimeRange = useDeferredValue(maxPrimeRange);
16    useEffect(() => {
17      onChange(deferredMaxPrimeRange);
18    }, [deferredMaxPrimeRange, onChange]);
```

The Result


MAKE IT SO
memegenerator.net


Concurrent Rendering Adventure
localhost:3000/primes
Concurrent Adventures
Users
Prime Numbers
Prime Numbers
100,000
73,972 73,971 73,970 73,969 73,968 73,967 73,966
73,965 73,964 73,963 73,962 73,961 73,960 73,959
73,958 73,957 73,956 73,955 73,954 73,953 73,952
73,951 73,950 73,949 73,948 73,947 73,946 73,945
73,944 73,943 73,942 73,941 73,940 73,939 73,938
73,937 73,936 73,935 73,934 73,933 73,932 73,931
73,930 73,929 73,928 73,927 73,926 73,925 73,924
73,923 73,922 73,921 73,920 73,919 73,918 73,917

See you in the next video

Conclusion

- You can use **<Suspense /> today**
  - Suspend when lazily loading components and/or **fetching data**
  - Handle error with an **<ErrorBoundary />**
  - **Nest and/or parallelize** as needed
- **Concurrent mode**
  - **Coming soon** to a React application near you
  - Can make large applications **more responsive**
  - Render a React 18 application using **createRoot()**
  - Use **<SuspenseList />** to orchestrate <Suspense /> components
  - **Defer work** with **startTransition()** and/or **useTransition()**

Maurice de Beijer

@mauricedb

maurice.de.beijer
@gmail.com