

demo master start ⌂ FCBased.tsx

```
jsx.tsx FCBased.tsx
import {FunctionComponent} from "react";
type MyProps = {
  text: string;
};
export const FCBased: FunctionComponent<MyProps> = ({ text })=> {
  return <h1>{text}</h1>;
}
```

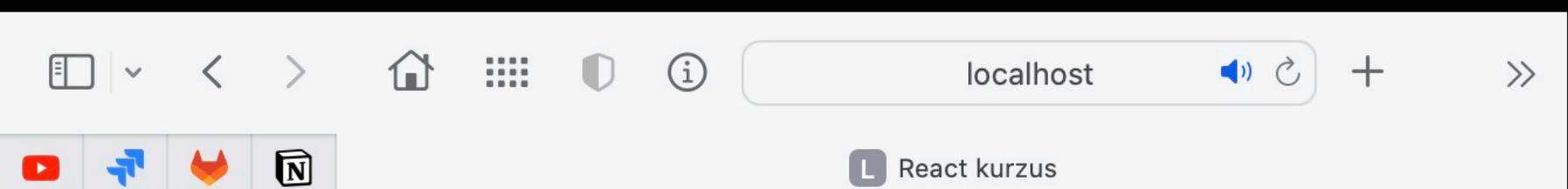
demo > src > examples > components > FCBased.tsx

10:1 LF UTF-8

The screenshot shows a dark-themed code editor interface. On the left, there's a sidebar with icons for file operations like copy, paste, and search. The main area displays two files: 'jsx.tsx' and 'FCBased.tsx'. The 'FCBased.tsx' file is currently active and contains the following TypeScript code:

```
import {FunctionComponent} from "react";
type MyProps = {
  text: string;
};
export const FCBased: FunctionComponent<MyProps> = ({ text })=> {
  return <h1>{text}</h1>;
}
```

The code defines a function component 'FCBased' that takes a prop 'text' of type 'string' and returns an `<h1>` element containing the value of 'text'. The code editor has a status bar at the bottom showing the path 'demo > src > examples > components > FCBased.tsx' and the current time '10:1'.



demo master start ⌂ FuncBased.tsx

```
jsx.tsx FCBased.tsx FuncBased.tsx
1 export function FuncBased() {
2   return <h1>Hello</h1>;
3 }
4
```

File Explorer

Terminal

Issues

Search

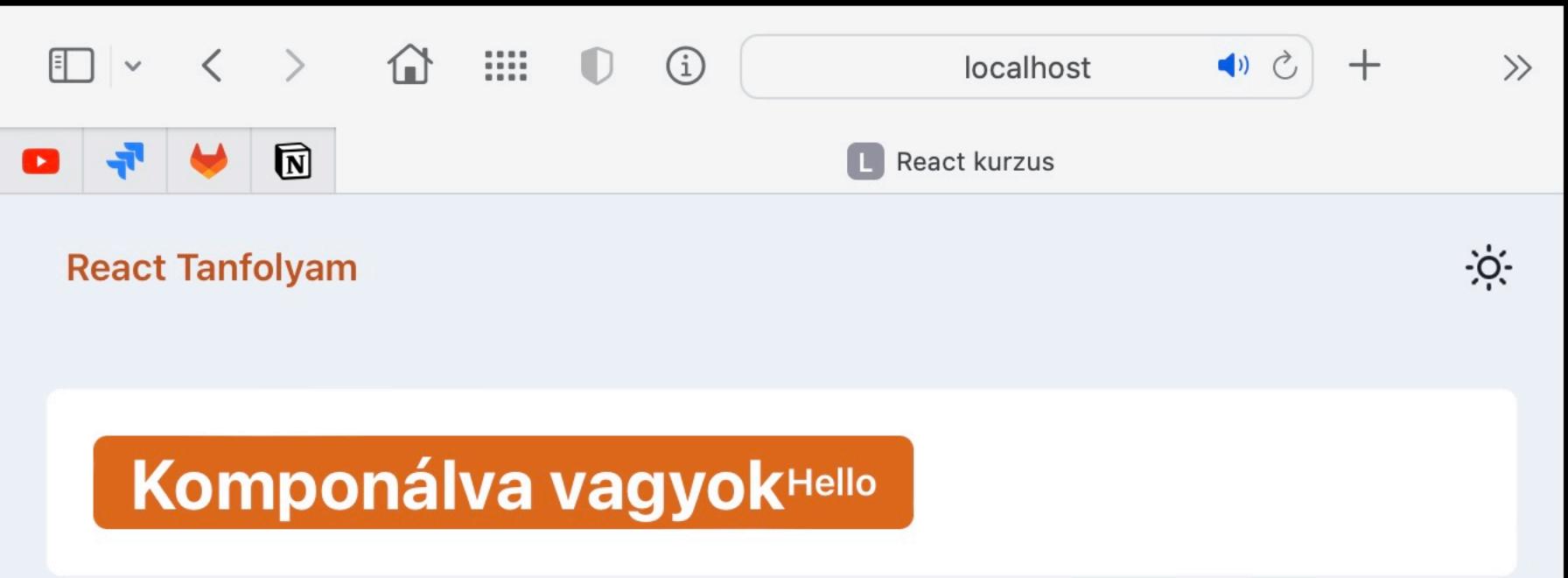
File demo src examples components FuncBased.tsx

4:1 LF UTF-8

This screenshot shows a split-screen development environment. The left side displays a web browser window for 'localhost' showing a React application with the title 'React kurzus' and the text 'Hello'. The right side is a code editor in dark mode, showing three files: 'jsx.tsx', 'FCBased.tsx', and 'FuncBased.tsx'. The 'FuncBased.tsx' file is active and contains the following TypeScript code:

```
export function FuncBased() {
  return <h1>Hello</h1>;
}
```

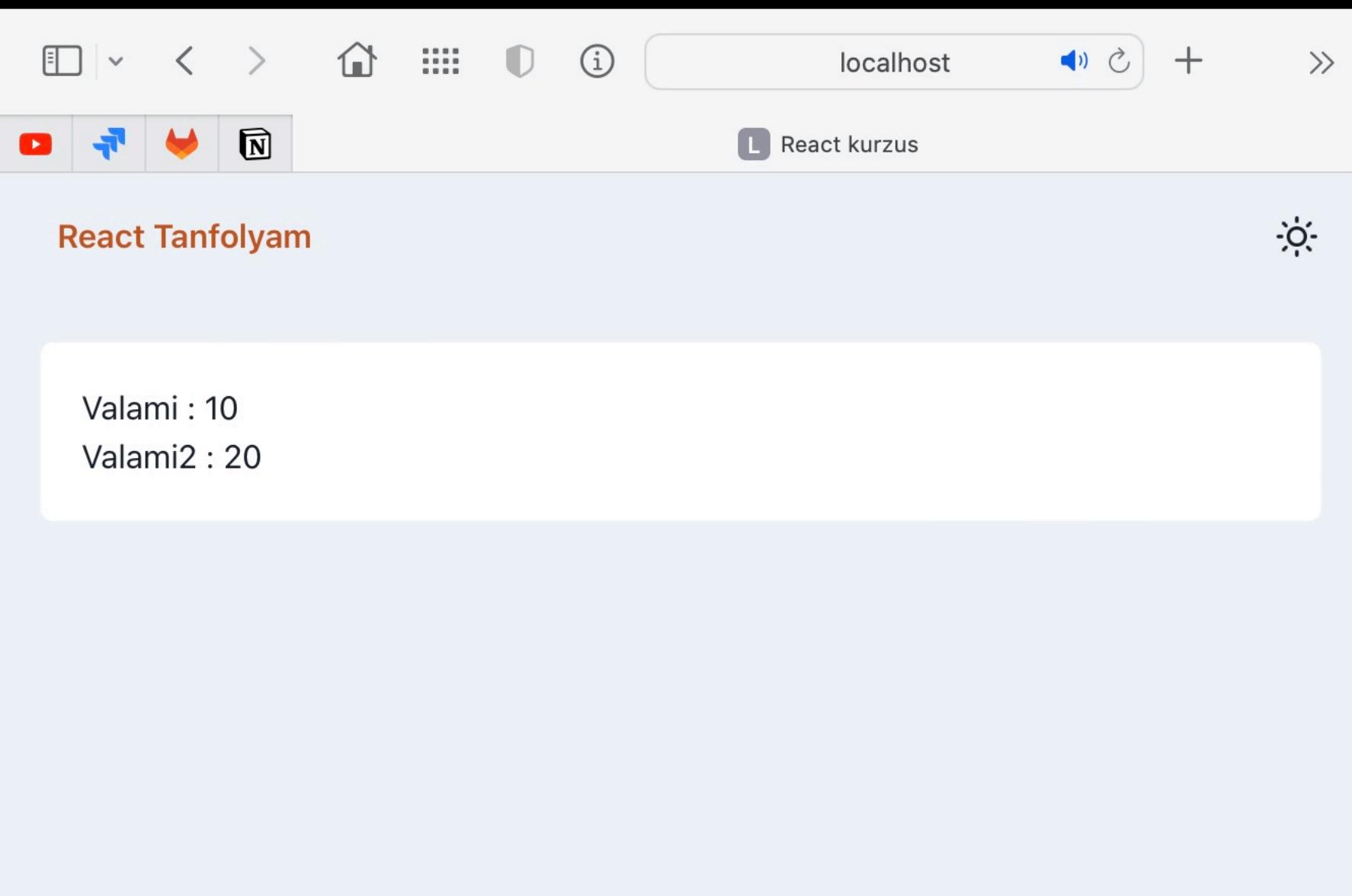
The code editor includes standard VS Code features like file navigation, a sidebar for file exploration, a terminal for running commands, and an issues panel for tracking errors.



demo master start ⌂ Composition.tsx

```
1 import { Button } from '@chakra-ui/react';
2 import { PropsWithChildren } from 'react';
3
4 export function Composition(props: PropsWithChildren) {
5   return <Button colorScheme='orange'>{props.children}</Button>;
6 }
7
```

demo > src > examples > components > Composition.tsx 1:1 LF UTF-8

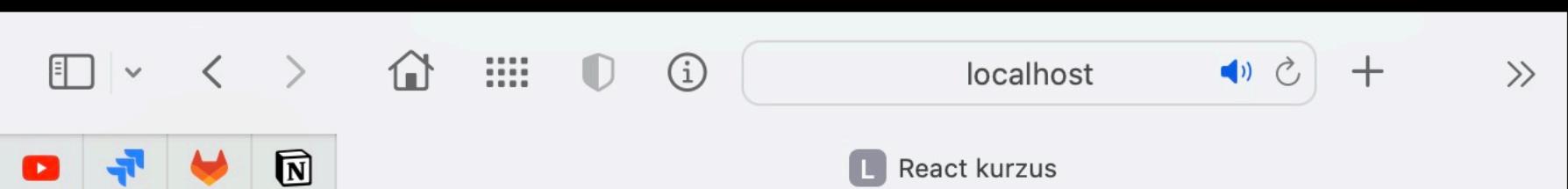


A screenshot of a web browser window titled "React kurzus". The page content is a slide from a presentation with the title "React Tanfolyam" and the number "8" in the top left corner. The main content area contains a large, semi-transparent white box with rounded corners, which obscures most of the slide's text. The browser's address bar shows "localhost" and the tab bar has a "React kurzus" tab.

A screenshot of a code editor interface. The title bar shows "demo" and "master". The current file is "Lifecycle.tsx", indicated by the tab icon. The code editor displays the following TypeScript code:

```
1 import { useEffect, useState } from 'react';
2
3 export function Lifecycle() {
4     const [count, setCount] = useState(initialState: 0);
5
6     useEffect( effect: () => {
7         console.log('MOUNT');
8
9         const id = setInterval(callback: () => {
10             console.log('Tick');
11             setCount(value: (count) => count + 1);
12         }, ms: 1000);
13
14         return () => {
15             console.log('UNMOUNT');
16             clearInterval(id);
17         };
18     }, deps: []);
19
20     return <p>{count}</p>;
21 }
22
```

The code uses the `useEffect` hook to log "MOUNT" when the component mounts and a tick counter every second. It also logs "UNMOUNT" when it unmounts and clears the interval.



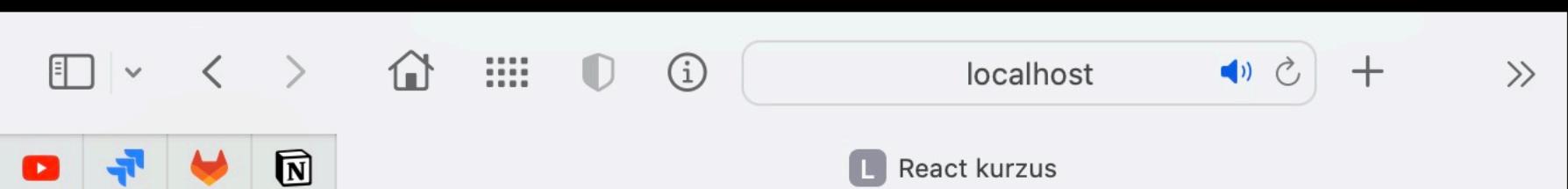
localhost

demo master start ⌂ States.tsx

```
1 import { Button } from '@chakra-ui/react';
2 import { useState } from 'react';
3
4 export function States() {
5   const [count, setCount] = useState<number>({ initialState: 0 });
6   console.log('hello');
7   return (
8     <Button
9       colorScheme='orange'
10      onClick={() => {
11        setCount({ value: count + 1 });
12      }}
13    >
14     Kattintva ennyiszer: {count}
15   </Button>
16 );
17 }
18
```

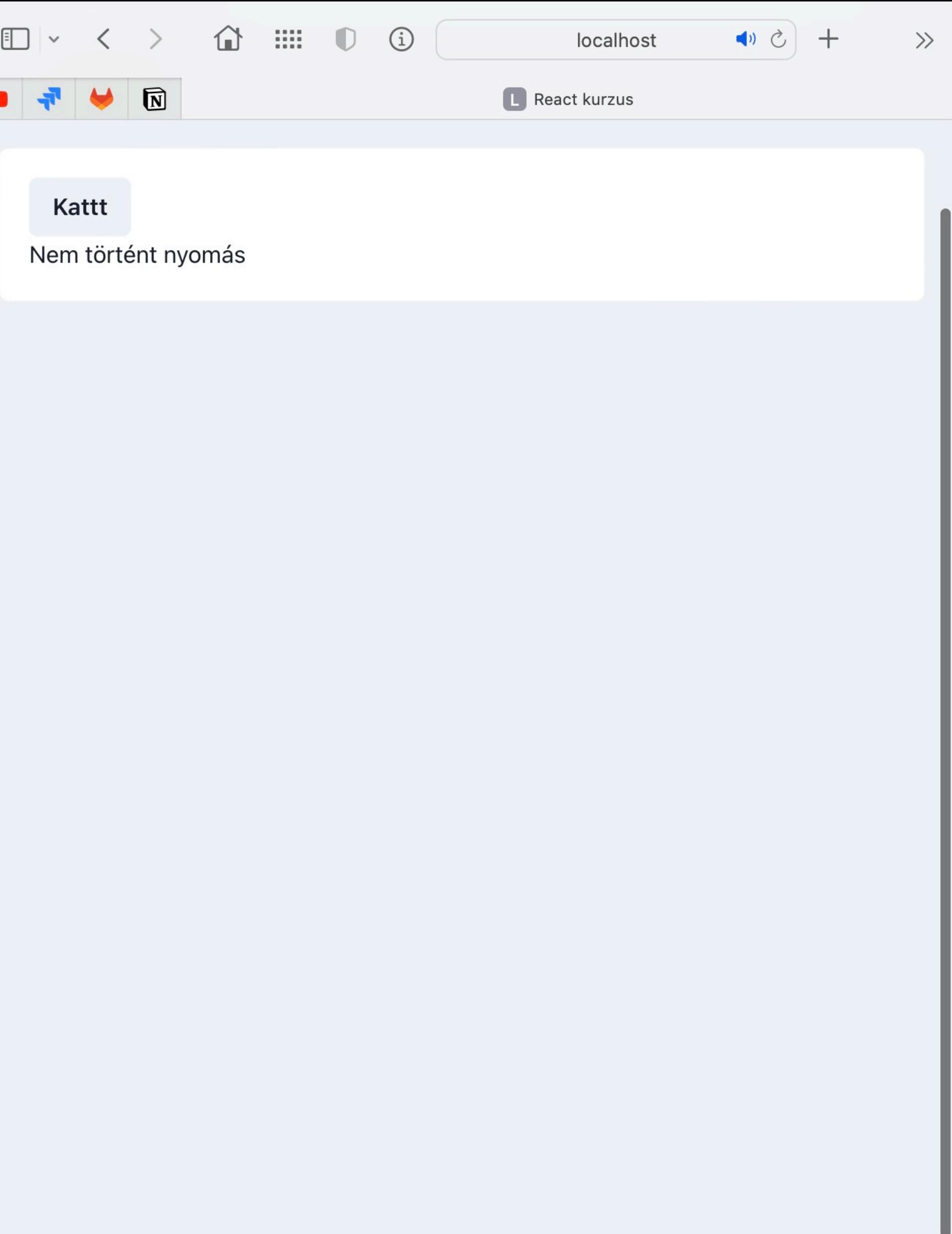
demo > src > examples > components > States.tsx

1:1 LF UTF-8

A screenshot of a code editor interface. The title bar shows 'demo' and 'master'. The top right has icons for 'start', 'refresh', 'stop', and 'search'. The left sidebar has icons for folder, file, and other project navigation. The main editor area is titled 'StatesWithFunction.tsx'. The code is as follows:

```
1 import { Button } from '@chakra-ui/react';
2 import { useState } from 'react';
3
4 export function StatesWithFunction() {
5   const [count, setCount] = useState<number>({ initialState: 0 });
6   const increment = () => {
7     setCount({ value: count + 1 });
8   };
9   return (
10     <Button colorScheme='orange' onClick={increment}>
11       Kattintva ennyiszer: {count}
12     </Button>
13   );
14 }
15
```

The code uses Chakra UI's Button component and the useState hook from React to manage a state variable 'count' and a function 'increment' to update it.



A screenshot of a browser window titled "React kurzus" showing a simple React application. The page contains a button labeled "Kattt" and a heading "Nem történt nyomás".

The browser's address bar shows "localhost" and the tab title is "React kurzus".

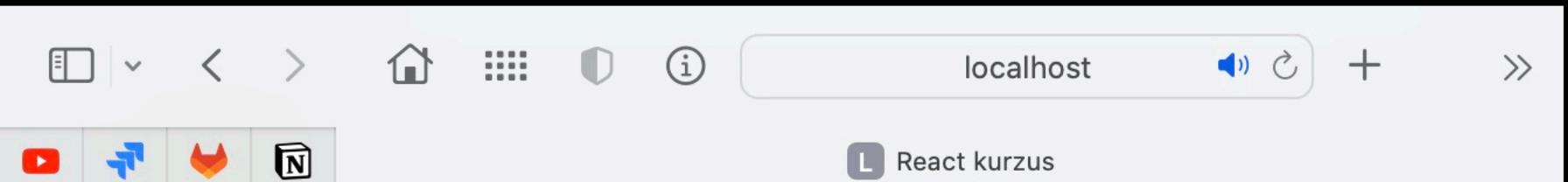
On the right side of the image, a code editor window is open, showing the file "FunctionInteraction.tsx". The code defines a functional component "FunctionInteraction" that returns a component with a button and an inner component. The button's onClick event calls a function "func" with the argument "text: 'hello'". The code editor interface includes tabs for "StatesWithFunction.tsx" and "FunctionInteraction.tsx", a sidebar with navigation icons, and a status bar at the bottom.

```
import { Button } from '@chakra-ui/react';
import React, { useState } from 'react';

export function FunctionInteraction() {
  const [text, setText] = useState<string>('Nem történt nyomás');
  const func = (newText: string) => {
    setText(newText);
  };
  return (
    <>
      <InnerComponent func={func} />
      <h1>{text}</h1>
    </>
  );
}

type MyProps = {
  func: (text: string) => void;
};

const InnerComponent: React.FC<MyProps> = ({ func }) => {
  return (
    <Button
      onClick={() => {
        func('hello');
      }}
    >
      Kattt
    </Button>
  );
};
```



localhost

demo master start ⌂ ContextInteraction.tsx

```
export function ContextInteraction() {
  return (
    <MyContextProvider>
      <div>
        <div>
          <InputComponent />
        </div>
      </div>
      <OutputComponent />
    </MyContextProvider>
  );
}

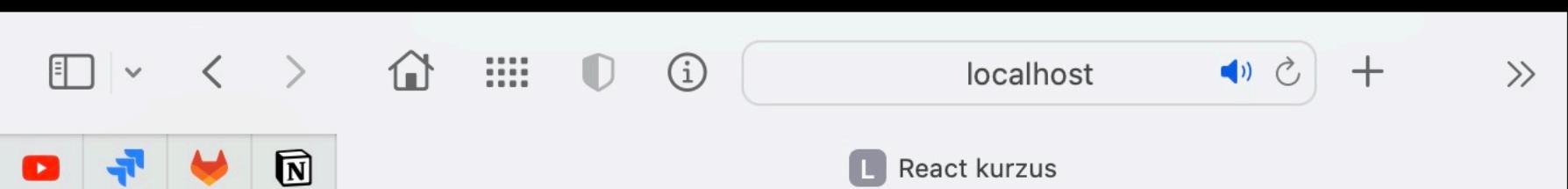
type MyContextType = {
  value: string;
  setValue: (newValue: string) => void;
};

const MyContext = createContext<MyContextType>({ defaultValue: {
  value: '',
  // eslint-disable-next-line @typescript-eslint/no-empty-function
  setValue: () => {},
});

function MyContextProvider({ children }: PropsWithChildren) {
  const [value, setValue] = useState<string>({ initialState: 'Kezdő érték' });

  return <MyContext.Provider value={{ value, setValue }}>{children}</MyContext.Provider>;
}

function InputComponent() {
  const { setValue } = useContext<MyContextType>(MyContext);
}
```



demo master

start

UseCallback.tsx

```
1 import { Button, ButtonProps } from '@chakra-ui/react';
2 import { useCallback, useState } from 'react';
3
4 export function UseCallback() {
5   const [state, setState] = useState(initialState: false);
6   const handleClick = useCallback(callback: () => {
7     setState(!state);
8   }, deps: [state]);
9   return <MyButton onClick={handleClick}>Toggle to {!state ? 'Tr
10 }
11
12 function MyButton(props: ButtonProps) {
13   console.log('Render');
14   return <Button {...props} />;
15 }
16
```

File Explorer

Search

Terminal

Output

Issues

Problems

Console

Logs

Memory

Performance

Network

File

Edit

View

Insert

Refactor

Format

Tools

Help

1:1 LF UTF-8

The screenshot shows a development environment with two main panes. The left pane is a browser window displaying a React application titled "React kurzus". The page content includes a heading "Itt tart: 0" and two buttons labeled "Count" and "asd". The right pane is a code editor showing the source code for a file named "UseEffect.tsx".

```
demo master
start
localhost
React kurzus
UseEffect.tsx
import { Box, Button } from '@chakra-ui/react';
import { useEffect, useState } from 'react';

export function UseEffect() {
  const [value, setValue] = useState<string>({ initialState: 'Nem indult' });
  const [count, setCount] = useState<number>({ initialState: 0 });
  // Tömbökkel és Objektumokkal óvatosan, referencia check lesz lehetséges
  useEffect(() => {
    console.log(count);
    setValue(`Itt tart: ${count}`);
  }, [count]);
  console.log(value);
  return (
    <Box>
      <h1>{value}</h1>
      <Button
        onClick={() => {
          setCount(count + 1);
        }}
      >
        Count
      </Button>
      <Button
        onClick={() => {
          setValue('asd');
        }}
      >
        asd
      </Button>
    </Box>
  );
}
```

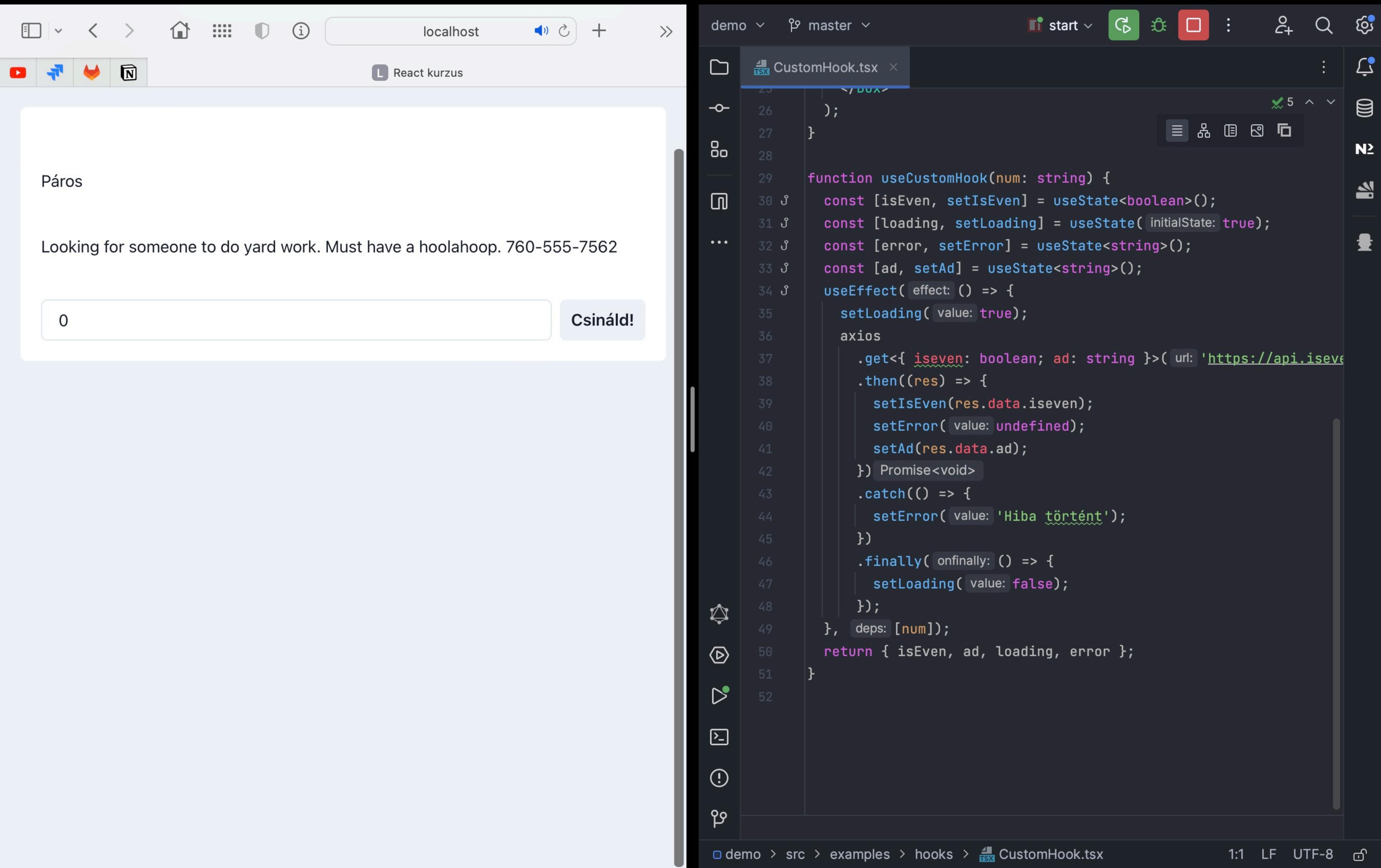
The code defines a functional component "UseEffect" that uses the "useState" hook to manage state for "value" (string) and "count" (number). It logs the current count to the console and updates the value to include the current count. It also contains two buttons: one to increment the count and another to set the value to "asd". The browser preview shows the initial state where "value" is "Nem indult" and "count" is 0. After clicking the "Count" button once, the value is updated to "Itt tart: 1". Clicking the "asd" button sets the value to "asd".

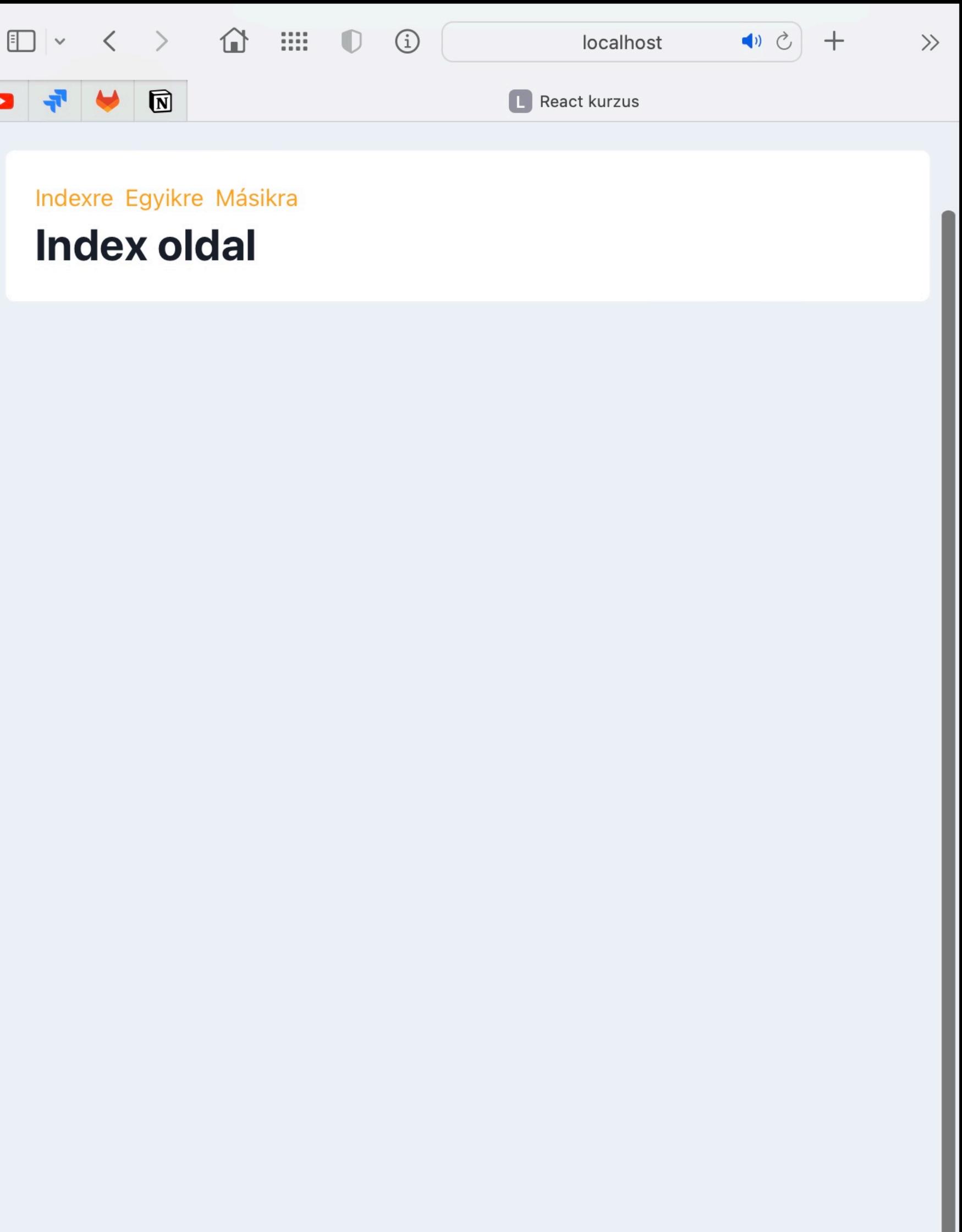
The screenshot shows a browser window with the following elements:

- Top bar: Back, Forward, Home, Grid, Shield, Info icon, Address bar (localhost), Volume, Refresh, Plus, and More icons.
- Toolbar: YouTube, LinkedIn, GitHub, and React logo.
- Page title: React kurzus.
- Main content area:
 - React Components**:
 - React.FC alapú
 - Function alapú
 - Kompozíció
 - Propok és attribútumok
 - Életciklus
 - Állapotok
 - Belső függvények
 - Komponensek interakciója**:
 - Propok
 - Függvények
 - Context
 - Hook-ok**:
 - useState
 - useCallback
 - useEffect
 - useMemo
 - egyedi
 - Routing**:
 - Router DOM
 - Router fügvények
 - Query Paraméterek
 - Útvonal Paraméterek
 - Nagy Példa**:
 - TodoApp

The screenshot shows a dark-themed code editor interface. The top bar includes navigation items like 'demo' and 'master', and various icons for file operations and settings. On the left, a sidebar displays a tree view of project files, with 'UseMemo.tsx' currently selected. The main workspace contains the following code:

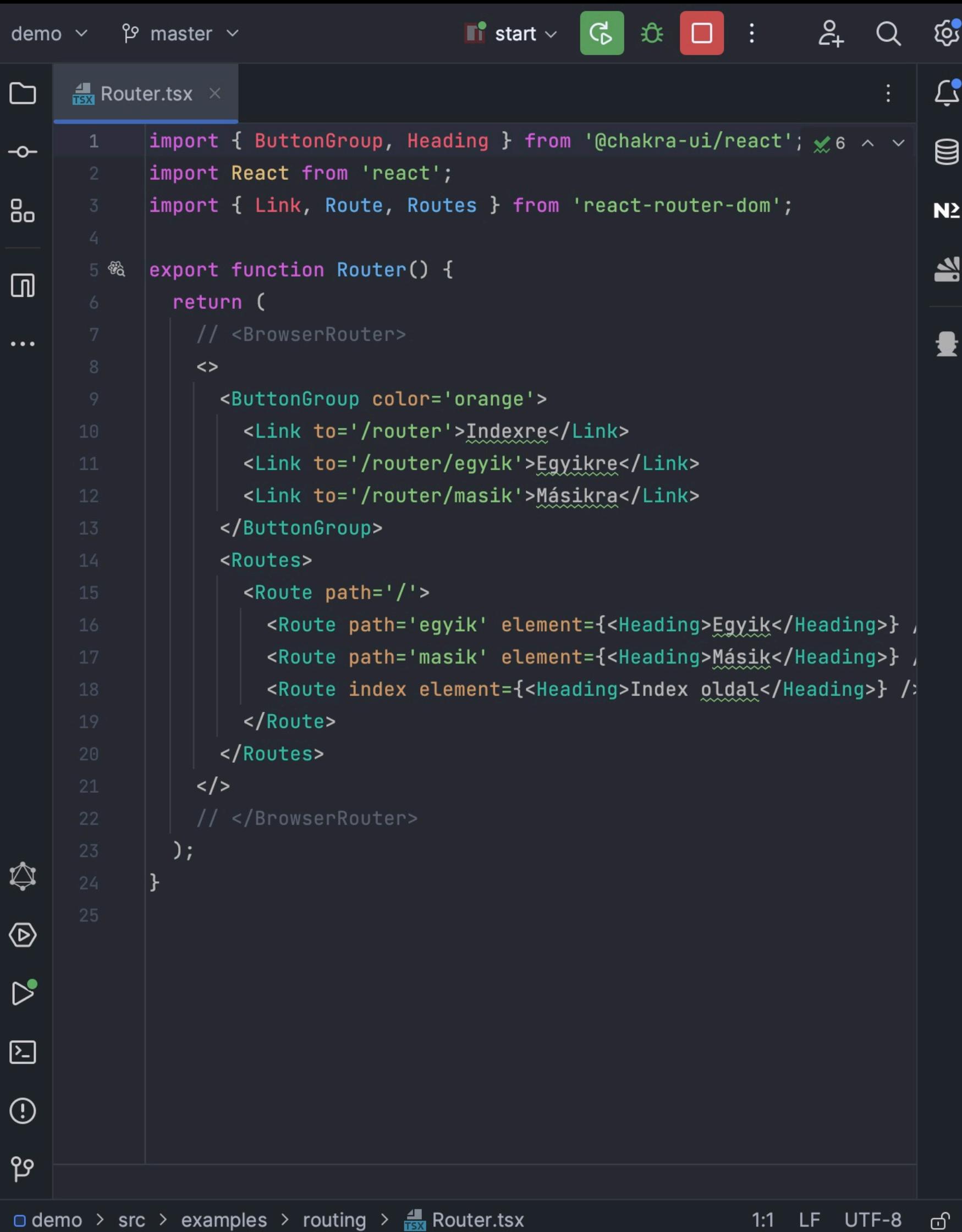
```
4  export function UseMemo() {
5      const [num, setNum] = useState<number>({ initialState: 0 });
6      const iterativeSum = useMemo<number>({ factory: () => {
7          const limit = 1000000000;
8          let sum = 0;
9          for (let i = 0; i < limit; i++) {
10              sum += num;
11          }
12          return sum;
13      }, deps: [num]);
14      const [anotherValue, setAnotherValue] = useState<boolean>({ initialValue: false });
15      return (
16          <Box>
17              <Text my={10}>{iterativeSum}</Text>
18              <Text my={10}>{anotherValue ? 'Igaz' : 'Hamis'}</Text>
19              <ButtonGroup>
20                  <Button
21                      onClick={() => {
22                          setNum({ value: num + 1 });
23                      }}
24                  >
25                      Nőőőő
26                  </Button>
27                  <Button
28                      onClick={() => {
29                          setNum({ value: num - 1 });
30                      }}
31                  >
32                      Csökk
33                  </Button>
34                  <Button
35                      onClick={() => {
```





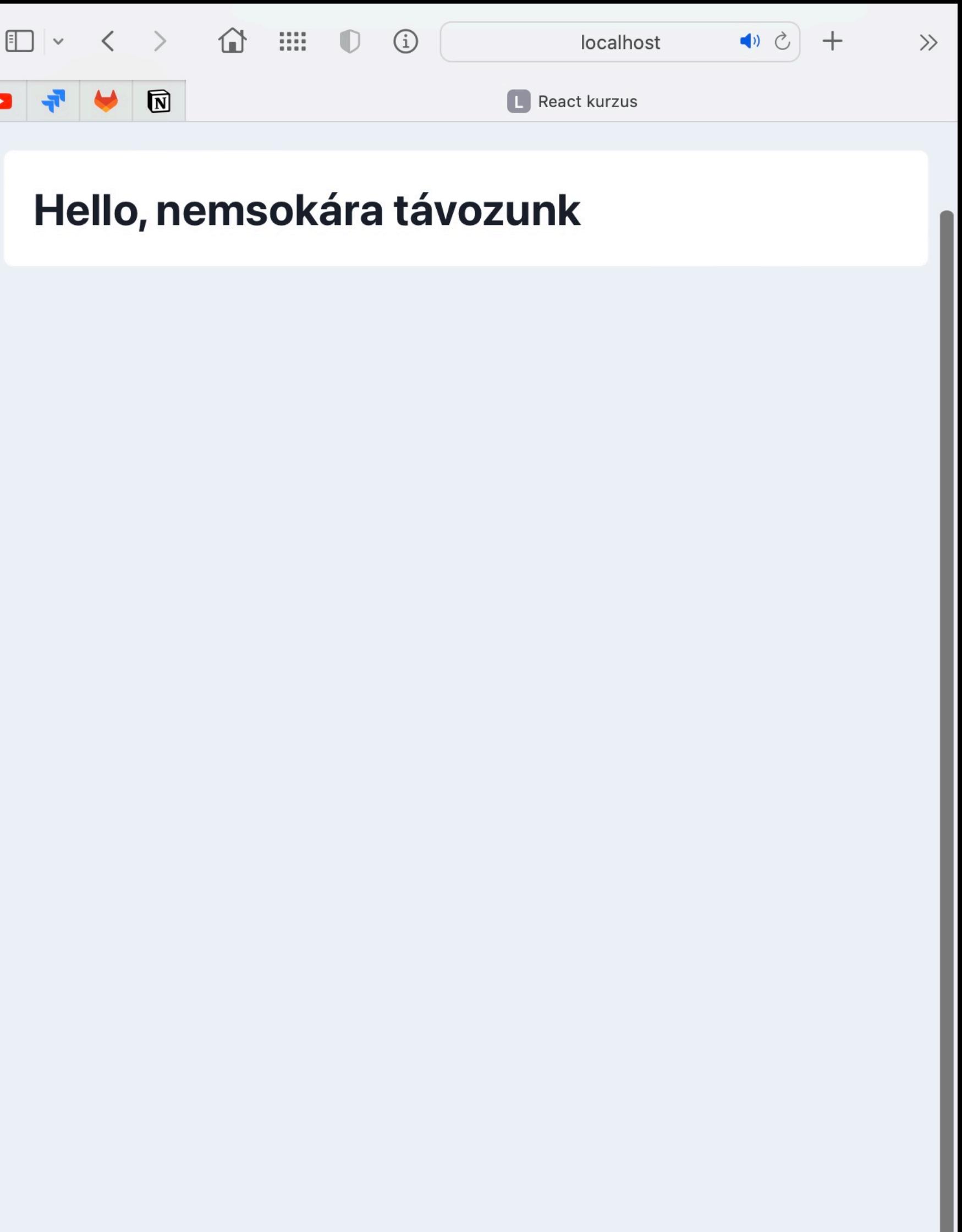
The browser window displays the URL `localhost` in the address bar. The page title is `React kurzus`. The content of the page is:

Indexre Egyikre Másikra
Index oldal



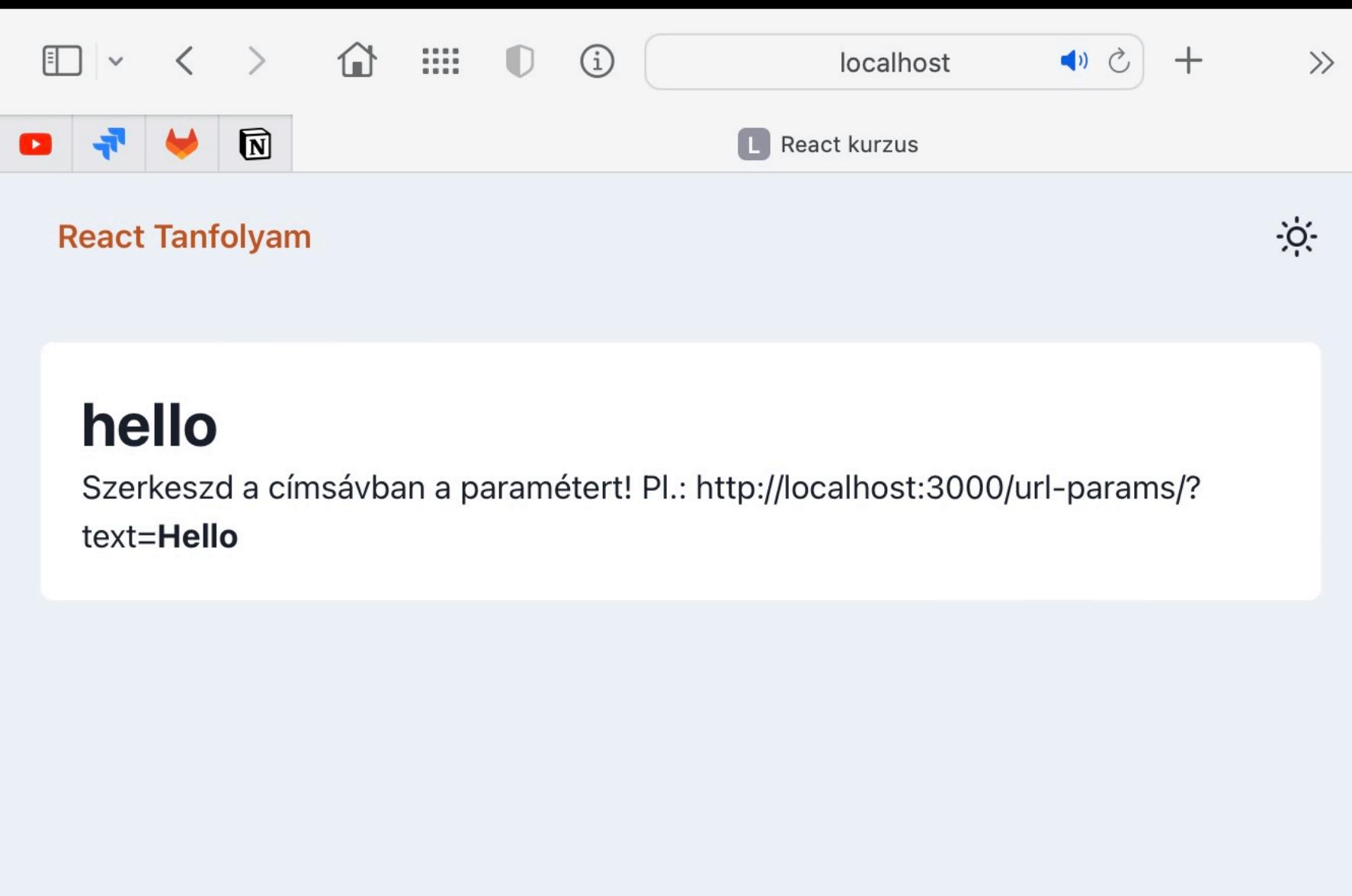
The code editor shows a file named `Router.tsx` with the following content:

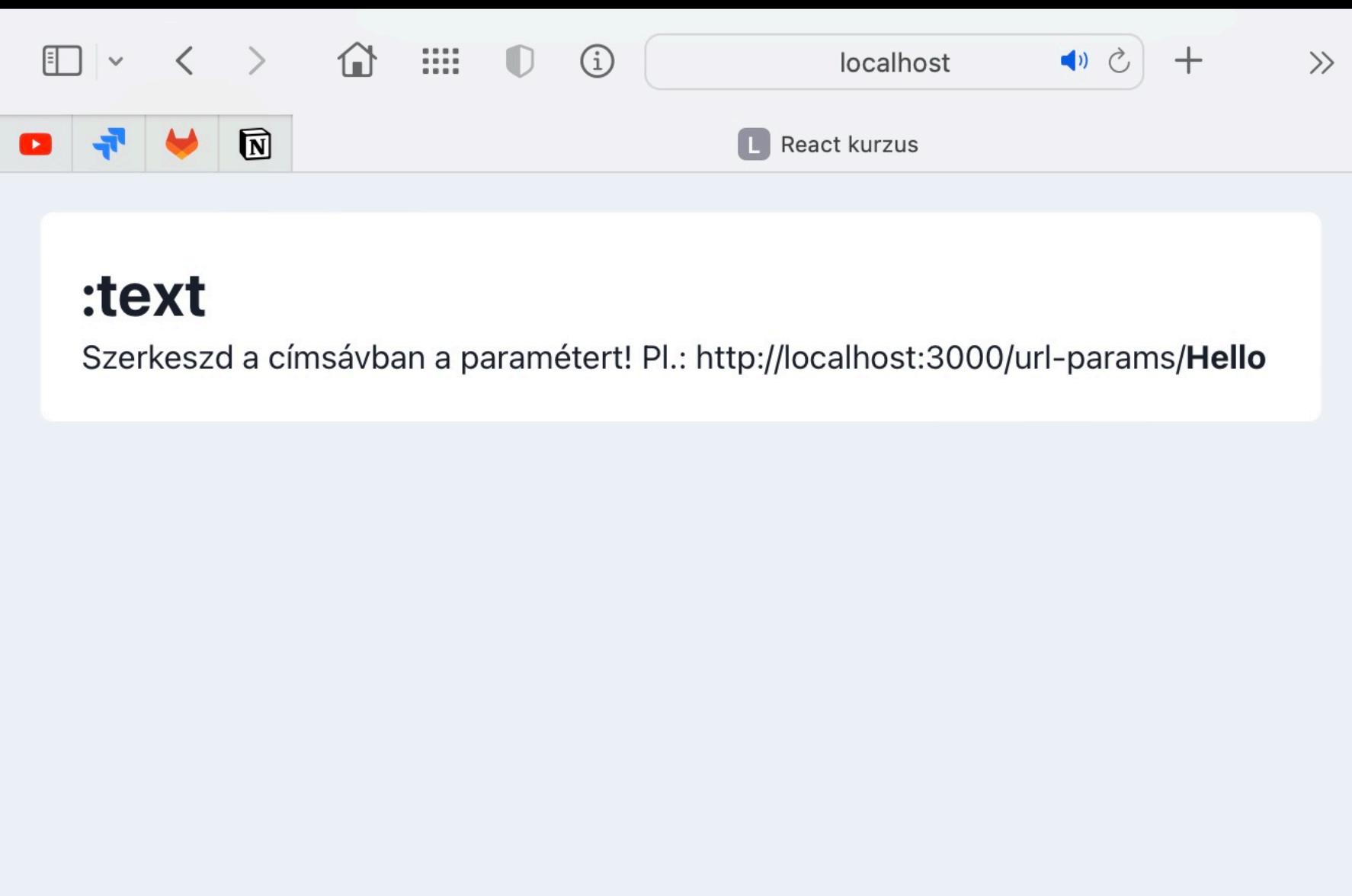
```
1 import { ButtonGroup, Heading } from '@chakra-ui/react';
2 import React from 'react';
3 import { Link, Route, Routes } from 'react-router-dom';
4
5 export function Router() {
6   return (
7     // <BrowserRouter>
8     <>
9       <ButtonGroup color='orange'>
10         <Link to='/router'>Indexre</Link>
11         <Link to='/router/egyik'>Egyikre</Link>
12         <Link to='/router/masik'>Másikra</Link>
13       </ButtonGroup>
14       <Routes>
15         <Route path='/'>
16           <Route path='egyik' element={<Heading>Egyik</Heading>} />
17           <Route path='masik' element={<Heading>Másik</Heading>} />
18           <Route index element={<Heading>Index oldal</Heading>} />
19         </Route>
20       </Routes>
21     </>
22     // </BrowserRouter>
23   );
24 }
```

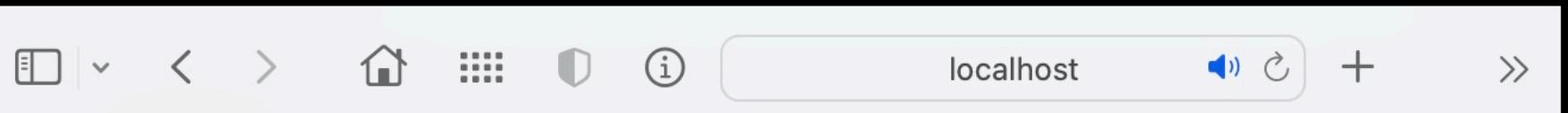


```
demo master start ⌂ RouterFunctions.tsx
1 import { Heading, useTimeout } from '@chakra-ui/react';
2 import React from 'react';
3 import { useNavigate } from 'react-router';
4
5 export function RouterFunctions() {
6   const navigate = useNavigate();
7   useTimeout( callback: () => {
8     navigate( to: '/');
9   }, delay: 5000);
10
11   return <Heading>Hello, nemsokára távozunk</Heading>;
12 }
```

The screenshot shows a code editor in Visual Studio Code. The file 'RouterFunctions.tsx' is open and active. The code uses Chakra UI and React Router to render a heading and perform a navigation after a delay. The code editor interface includes a sidebar with icons for file operations, a status bar at the bottom, and a dark theme.







demo master

start

TodoApp.tsx

```
type FormFields = {
  title: string;
};

const schema = object(spec: {
  title: string().required(msg: 'Nem szabad üresen hagyni!'),
});

export function TodoApp() {
  const [todos, setTodos] = useState<TodoItem[]>([initialState: []]);
  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm<FormFields>({ props: { resolver: yupResolver(schema) } });

  const onSubmit = ({ title }: FormFields) => {
    reset();
    const newTodo: TodoItem = { title, done: false, createdAt: new Date() };
    // Rossz megoldás (nem módosul a referencia, így render nem lehetséges)
    // todos.push(newTodo);
    // setTodos(todos);
    setTodos({ value: [...todos, newTodo] });
  };

  // Function factory!!!! Returns a function for the given number
  const toggleTodo = (createdAt: number) => () => {
    setTodos(
      todos.map((t) => {
        if (t.createdAt === createdAt) t.done = !t.done;
        return t;
      })
    );
  };
}
```

demo > src > examples > todoapp > TodoApp.tsx

1:1 LF UTF-8

A screenshot of a web browser window titled "React kurzus". The page displays a form with two input fields. The first field is labeled "Első mező" and the second "Második mező". Below the inputs are two buttons: "Küldés" (orange background) and "Reset" (white background).

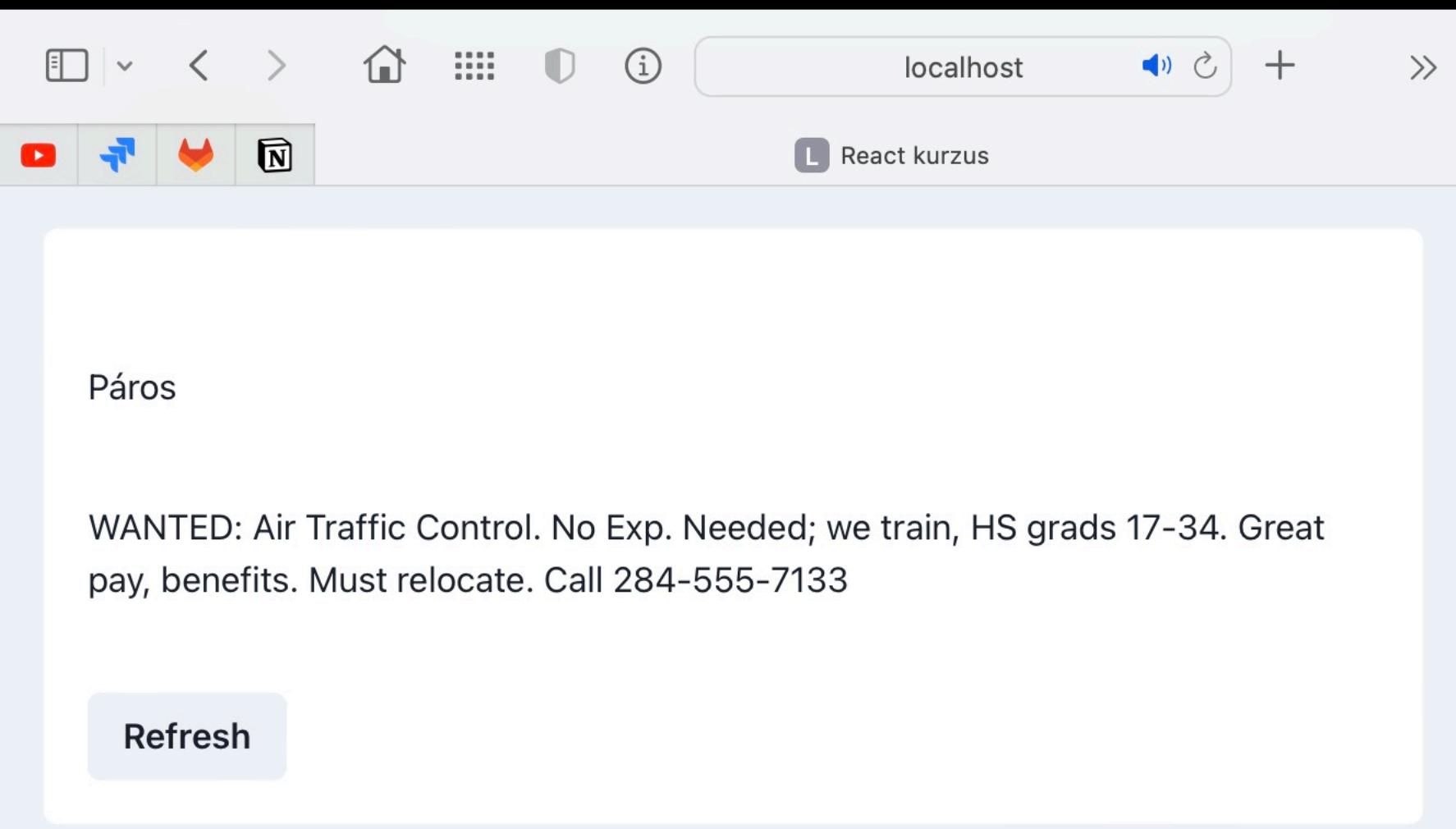
A screenshot of a code editor showing a file named "Form.tsx". The code defines a schema object and a FormFields type, then uses the Yup library to create a form component. The component includes FormControl and FormLabel components for each field, with error messages displayed if validation fails.

```
const schema = object(spec: {
  first: string().required(msg: 'Kötelező ō');
  second: number().required(msg: 'Kötelező ō').positive(msg: 'Minimális érték');
});

type FormFields = {
  first: string;
  second: number;
};

export function Form() {
  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm<FormFields>({ props: { resolver: yupResolver(schema) } });
  const onSubmit = (values: FormFields) => {
    alert(`${values.first} ${values.second}`);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <FormControl isInvalid={!errors.first}>
        <FormLabel>Első mező</FormLabel>
        <Input {...register(name: 'first')} />
        <FormErrorMessage>{errors.first?.message}</FormErrorMessage>
      </FormControl>
      <FormControl isInvalid={!errors.second}>
        <FormLabel>Második mező</FormLabel>
        <Input {...register(name: 'second')} />
        <FormErrorMessage>{errors.second?.message}</FormErrorMessage>
      </FormControl>
    </form>
  );
}
```



The screenshot shows a dark-themed code editor interface with a sidebar on the left containing various icons for file operations like copy, paste, delete, and search. The main area displays a file named `ReactQuery.tsx`. The code implements a `QueryClientProvider` and a `QueryComponent` using the `useQueryClient` hook. It also uses the `useIsEvenQuery` hook to determine if a number is even or odd and to handle the response from an axios query.

```
const client = new QueryClient();

export function ReactQuery() {
    return (
        <QueryClientProvider client={client}>
            <QueryComponent />
        </QueryClientProvider>
    );
}

function QueryComponent() {
    const queryClient = useQueryClient();
    const { data, isLoading, isError } = useIsEvenQuery();
    if (isLoading) return <Spinner />;
    if (isError) return <Text>Hiba</Text>;
    if (!data) return null;
    return (
        <>
            <Text my={10}>{data.iseven ? 'Páros' : 'Páratlan'}</Text>
            <Text my={10}>{data.ad}</Text>
            <Button onClick={() => queryClient.invalidateQueries(queryKey)}>Töröl</Button>
        </>
    );
}

const useIsEvenQuery = () =>
    useQuery({ queryKey: 'isEven', queryFn: async () => {
        const response = await axios.get<{ iseven: boolean; ad: string }>('https://api.example.com/is-even');
        return response.data;
    }});
}
```



A screenshot of a web browser window titled "React kurzus" showing a red circular button with the text "Pöddyök" in white.

The browser address bar shows "localhost".

The code editor on the right has tabs for "ReactQuery.tsx" and "StyledComponents.tsx", with "StyledComponents.tsx" currently selected.

The code in "StyledComponents.tsx" is:

```
import styled from 'styled-components';

export function StyledComponents() {
  return (
    <StyledButton>
      <p>Pöddyök</p>
    </StyledButton>
  );
}

const StyledButton = styled.div`
  border-radius: 600px;
  height: 200px;
  width: 200px;
  background-color: red;
  display: flex;
  align-items: center;
  justify-content: center;
  color: white;
  font-style: italic;
  font-weight: bold;
  font-size: 3rem;
  transition: transform 0.5s ease;
  :hover {
    transform: rotate(720deg) scale(2);
  }
`;
```

Egyéb

[React Hook Forms](#)

[React Query](#)

[Styled Components](#)

[React Docs](#)

[create-react-app](#)

[Vite](#)

[Redux](#)

[ChakraUI](#)

[React Icons](#)

[NextJS](#)