React with Hooks

- Presentation
- Workshop

React - Props

```
function Hi(props) {
  return <h1>Hi, {props.name}</h1>;
}

function Hello({ name }) {
  return <h1>Hello, {name}</h1>;
}

function App() {
  return (
    <div>
      <Hi name="Sara" />
      <Hello name="Cahal" />
    </div>
  );
}
```

React - Hook

- function

```
function fn() {
  const [count, setCount] = useS...(...);

  useS...(...);

  return <>React component</>;
}
```

Hook - Motivation

- Reuse logic between components
- Reducing the size of components
- Functions are simpler than classes

React Hooks

- Basic Hooks
  - `useState`
  - `useEffect`
  - `useContext`
- Additional Hooks
  - `useReducer`
  - `useCallback`
  - `useMemo`
  - `useRef`
  - `useImperativeHandle`
  - `useLayoutEffect`

React Hooks 2

- Additional Hooks
  - `useDebugValue`
  - `useDeferredValue`
  - `useTransition`
  - `useId`
- Library Hooks
  - `useSyncExternalStore`
  - `useInsertionEffect`

SOFA repo Hooks

- Basic Hooks
  - `useState` 188
  - `useEffect` 122
  - `useContext` 16
- Additional Hooks
  - `useCallback` 76
  - `useMemo` 19
  - `useRef` 40
- Cutom hooks
  - ...

Hook - useState

```
import React, { useState } from "react";

export default function State() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

Hook - useEffect *

```
import React, { useState, useEffect } from "react";

function Example() {
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(
    () => {
      document.title = `You clicked ${count} times`;
    },
    // undefined
    // []
    [count]
  );

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

Hook - useContext

```
const themes = {
  light: {
    foreground: "#000000",
    background: "#eeeeee",
  },
  dark: {
    foreground: "#ffffff",
    background: "#222222",
  },
};

const ThemeContext = React.createContext(themes.light);

function App() {
  return (
    <ThemeContext.Provider value={themes.dark}>
      <Toolbar />
    </ThemeContext.Provider>
  );
}

function Toolbar(props) {
  return (
    <div>
      <ThemedButton />
    </div>
  );
}

function ThemedButton() {
  const theme = useContext(ThemeContext);
  return (
    <button style={{ background: theme.background, color: theme.foreground }}>
      I am styled by theme context!
    </button>
  );
}
```

Hook - useRef

```
function TextInputWithFocusButton() {
  const inputEl = useRef(null);
  const onButtonClick = () => {
    // `current` points to the mounted text input element
    inputEl.current.focus();
  };
  return (
    <>
      <input ref={inputEl} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
}
```

Building Your Own Hooks

usehooks.com

sofa-fenix/hooks

tailwind-ui-shop/hooks

useToggle

```
import { useCallback, useState } from "react";

function App() {
  const [isTextChanged, setIsTextChanged] = useToggle();

  return (
    <button onClick={setIsTextChanged}>
      {isTextChanged ? "Toggled" : "Click to Toggle"}
    </button>
  );
}

const useToggle = (initialState = false) => {
  const [state, setState] = useState(initialState);

  // Returns a memoized callback.
  const toggle = useCallback(() => setState((state) => !state), []);

  return [state, toggle];
};
```

Create React App

Create React App is a comfortable environment for learning React, and is the best way to start building a new single-page application in React.

Node >= 14.0.0

npm >= 5.6

```
npx create-react-app react-hooks-demo
cd react-hooks-demo
npm start
```

Zdroje:

https://reactjs.org/docs/hooks-intro.html

https://reactjs.org/docs/hooks-intro.html#motivation

https://usehooks.com/