Stav komponent

Jonáš Václavek, Tereza Vaňková, 28. 4. 2025

React girls

Poděkování

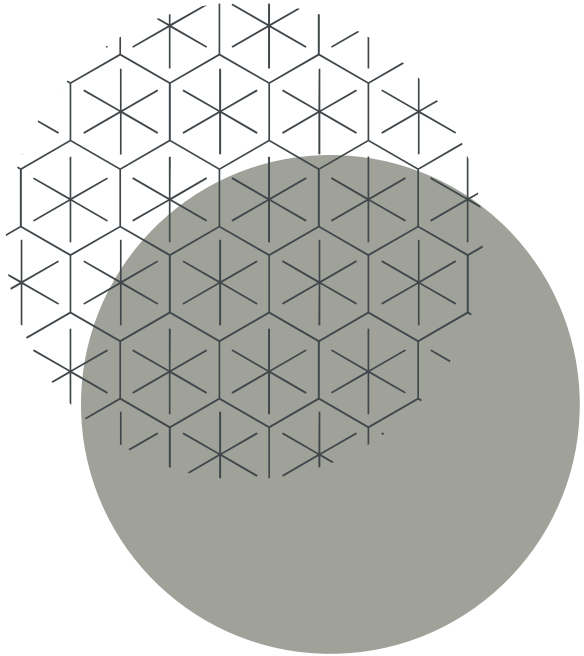
MEWS

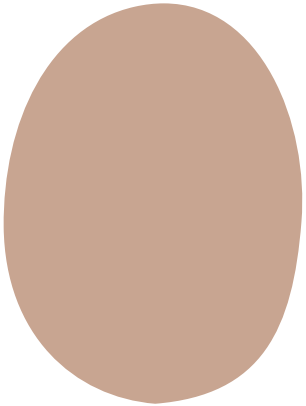
Obsah třetí lekce

- Opakování z minulé lekce
- Funkce a children jako *props*
- Stav komponenty
- *useState* hook

01

Opakování



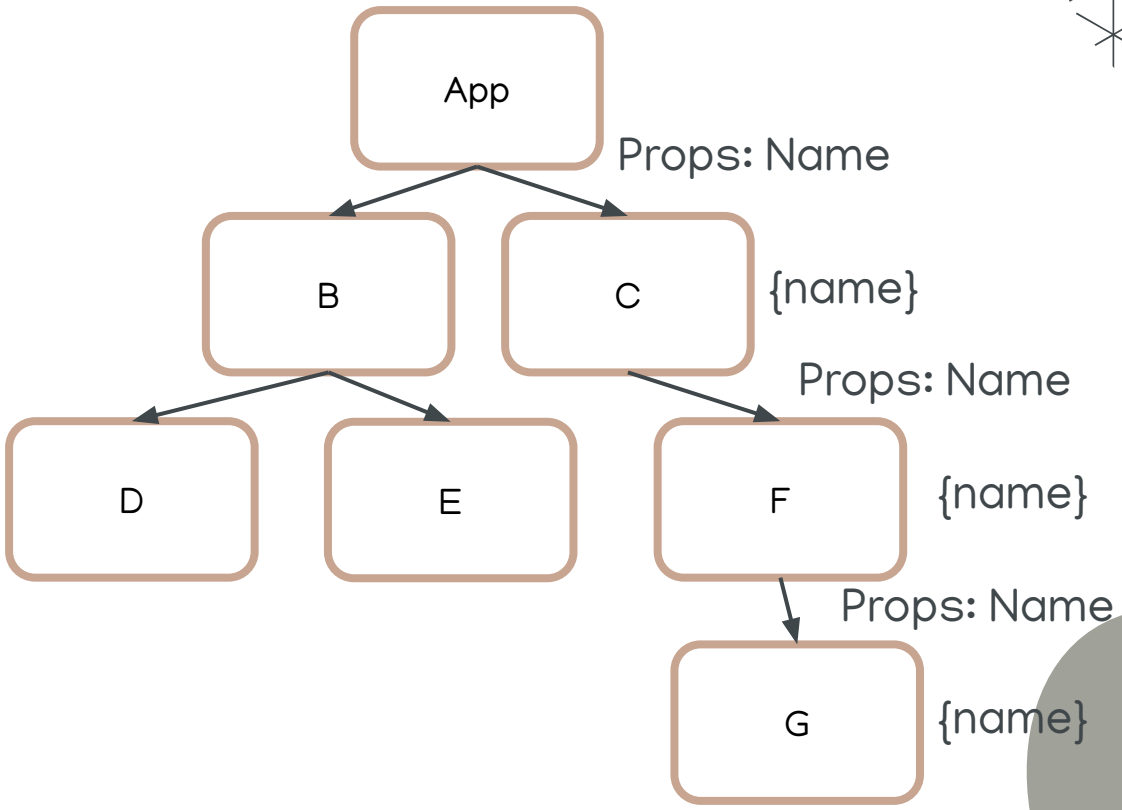


Co jsou props?

Props

- Posílání dat mezi komponentami
- *props* je jediný argument v komponentě, jedná se o *props* objekt

Props



Props

```
export const App = () => {
    const name = "Ondra"
    const age = 22
    return <Person name={name} age={age}>
}
export const Person = ({name, age}) => {
    return <p>You are {name}, {age}<p>
}
```

Procvičování

- Ve složce `react-akademie` si vytvoř novou složku `lecture_3` a do ní si vytvoř novou react aplikaci, ve které dnes budeme pracovat
  - `npm create vite@latest`
- Vytvoř novou komponentu User a použij ji v parent komponentě App
- Vyzkoušej si do User vložit *props* bez toho, aniž by sis do nich cokoliv posílal/a z parent komponenty a vypiš *props* v konzoli -> co se vypíše?

Procvičování

```
export const User = (props) => {
    console.log(props);
    return <div>User< /div>;
};
```

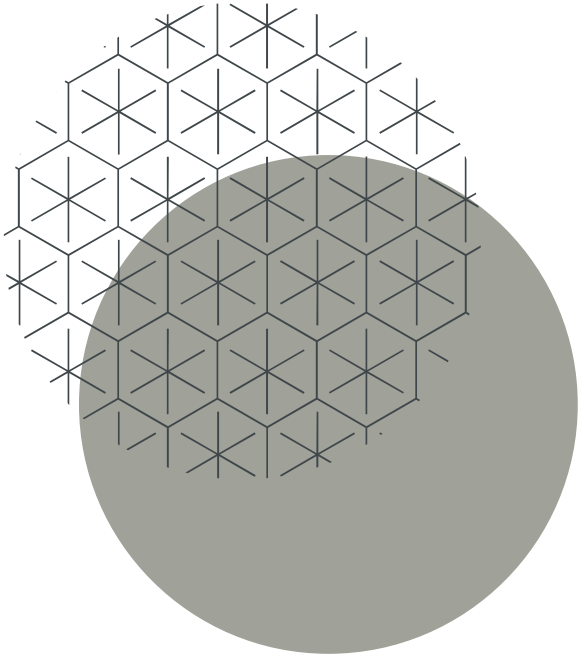
- V konzoli je prázdný objekt, *props* jsou vždy objekt, ať už do něj posílám cokoliv z parent komponenty
- Proto můžeme použít destructuring syntax

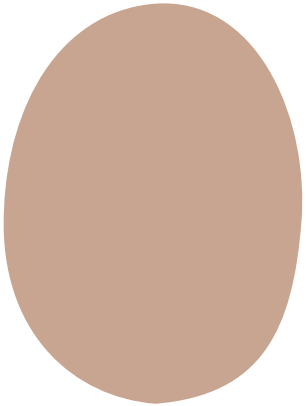
Příklad

- Vytvoř dvě komponenty, jedna se bude jmenovat *Students* a druhá *StudentDetail*
- *StudentDetail* bude child komponenta od komponenty *Students*
- *Students* bude child komponenta od komponenty *App*
- Komponenta *StudentDetail* bude vracet v paragrafu <p> *firstName*, *lastName* a *age*, které dostane z komponenty *Students* jako props
- Komponenta *Students* dostane z parent (*App*) komponenty prostřednictvím props *lide* data (z *lide.js*)
- *Students* komponenta bude mít podmínku, že pokud bude array se studenty prázdné (použíj .length vlastnost), vypíše text “No students”
- Pokud bude array students mít data, použíj .map() metodu a vykresli detail studenta za použití komponenty *StudentDetail*

02

Funkce a children prop





Cvičení - funkce jako props

```
export const Button = ({name, onClickFunction}) =>{
    return (
      <button onClick={onClickFunction}>
        {name}
      </button>;
    )};
```

- Příklad na dalším slidu

Příklad (pokračování)

- Komponentu *StudentDetail* uprav tak, aby používala Button komponentu
- Přidej funkci *onStudentClick* jako props do komponenty *StudentDetail*
- Ve *Students* komponentě vytvoř funkci, která bude vypisovat do konzole jméno studenta a pošli jí jako *props* komponentě *StudentDetail*
- Pošli *onStudentClick* do komponenty Button jako *onClickFunction* props

Předávání JSX jako children

- Někdy chceme zanořit jednu komponentu do druhé (použití například u Modalu nebo Layoutu)
- Pokud chceme do jedné komponenty zanořit JSX využijeme tzv. children prop
- Tuto komponentu obsahující children prop pak použijeme jako wrapper dalšího JSX nebo jiné komponenty

Cvičení - children jako props

```
export const Card = ({ children }) => {
  return (
    <div className="card">
      {children}
   < /div>
  );
}
```

Cvičení - children jako props

```
export const App() {
  return (
    <Card>
      <p>Some content or some component< /p>
    < /Card>
  );
}
```

Cvičení - children jako props

- Využití například jako Modal komponenta
- Obecný layout stránky
- Stylování
- apod

Procvičování

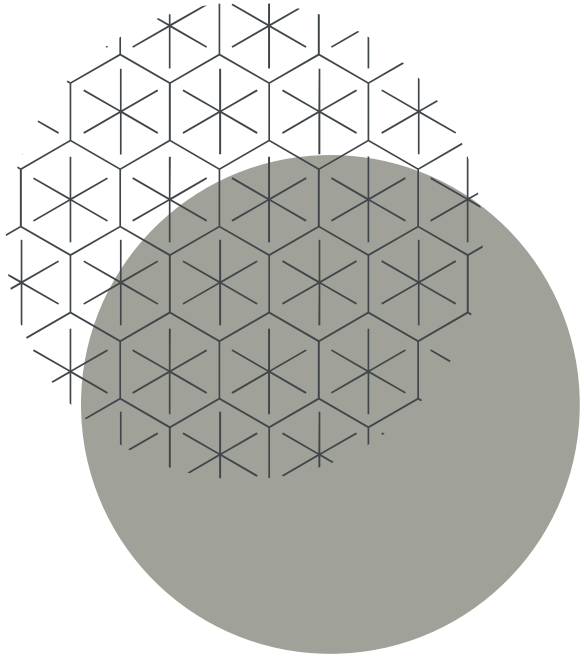
- Vytvoř si komponentu Wrapper
- Komponenta bude mít children prop
- Komponenta bude vracet následující JSX:

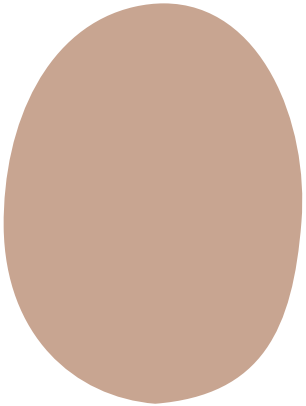
```
<div>
        <p>Header< /p>
        {children}
        <p>Footer< /p>
< /div>
```

- Použij Wrapper komponentu v App a jako children použij JSX: `<p> Content < /p>`

03

State, useState





State (stav) komponenty

- Jedná se o paměť komponenty
- Komponenta si zaznamenává interakci uživatele (například uživatel přidává do košíku)
- Lokální proměnné se neukládají mezi rendery, potřebujeme udržovat stav, který se mezi rendery drží
- Změny lokálních proměnných nezpůsobí re-render komponenty
- Ukázka StudentViewer ->

state, useState

- Pokud chceme aktualizovat komponentu novými daty, musíme :

  1. Uložit data mezi rendery

  2. Donutit komponentu se re-renderovat s novými daty

  Použijeme na to useState hook, který uloží stav a zároveň poskytuje funkci, která stav změní a způsobí re-render komponenty

  Opravená StudentViewer ukázka ->

useState, hooks

- useState je hook
- Hooky jsou speciální funkce https://react.dev/reference/react/hooks
- Jejich pojmenování začíná slovem “use...”
- Existuje několik built-in hooků, ale je možné si vytvořit i tzv. vlastní hooky (my se v rámci akademie seznámíme s několika dalšími, vlastní si zatím tvořit nebudeme)
- Lze je volat vždy pouze na nejvyšší úrovni komponenty, nelze je volat v podmínkách, cyklech a dalších funkcích

useState

- `const [index, setIndex] = useState(0);`

  Importovat useState z react knihovny

- `import { useState } from "react";`

  Zápis: const [something, setSomething] = useState()

  “something” je aktuální stav, “setSomething” je funkce, která nastaví nový stav a re-renderuje komponentu

  Pokud je něco v závorkách useState() je to počáteční hodnota -> ukázka

state, useState

- Komponenta může mít větší počet stavů a různé typy například:
- Komponenta StudentViewer bude mít další stav:

```
const [showMore, setShowMore] =
useState(false);
```

Stav showMore a index jsou na sobě nezávislé

state, useState

- Stav je lokální pro komponentu, když použiju komponentu StudentViewer dvakrát pod sebou, stav bude lokální a nezávislý, pro každou komponentu zvlášť
- Ukázka v kódu –>

Procvičování

- Vytvoř komponentu IncreaseByOne
- Komponenta bude vracet button a paragraf
- Stav komponenty ulož pomocí useState
- V paragrafu zobraz aktuální číslo
- Vytvoř funkci, která po kliknutí na button zvýší číslo o jedna
- Počáteční číslo bude 1

Procvičování

- Přidej do komponenty IncreaseByOne následující logiku
  - Pokud bude číslo sudé zobrazí se text “Even number”, pokud bude liché nezobrazí se nic

Spread operator

- Umožňuje nám “rozbalit” elementy

Příklad:

```
const numbers = [1, 2, 3, 4, 5]

const max = Math.max(numbers) -> co dostaneme?

const max = Math.max(...numbers) -> co dostaneme?
```

- Ukázka v kódu –>

Spread operator

- Umožňuje nám “rozbalit” elementy

Příklad:

```
const numbers = [1, 2, 3, 4, 5]

const new_numbers = [...numbers, 6]
```

- Ukázka v kódu –>

Spread operator

- Umožňuje nám teké “rozbalit” prvky

Příklad:

```
const props ={firstName:”Pavel”,lastName: “Novak”}

const User = (props)=>{

    return <UserDetail {...props} />

}
```

06

Příklady

Animals

- Vytvoř komponentu Animals
- Komponenta bude vracet seznam zvířat, input a tlačítko
- Počáteční stav bude pouze jedno zvíře “dog”
- Uživatel může do inputu napsat zvíře a pomocí tlačítka “Add animal” ho přidá do seznamu

07

Shrnutí

Shrnutí

- Props slouží k předávání dat mezi komponentami
- State je stav komponenty, se kterým manipuluju pomocí useState hooku

Díky za pozornost!