Předávání dat komponentám

Jonáš Václavek, Tereza Vaňková, 23. 4. 2025

React girls

Poděkování

MEWS

Obsah druhé lekce

- Opakování a
- Array metody a *key* atribut
- Podmíněné renderování (conditional rendering)
- Předávání dat komponentám (props)

01

Opakování

Tvorba komponenty

5 pravidel

Tvorba komponenty

1. Vytvořím funkci (kterou za použití PascalCase pojmenuju)
2. Funkce musí něco vracet – return
3. Funkce mi bude vrací JSX (velmi podobné html, jen malé rozdíly)
4. Vše co vracím musí být obalené v jednom elementu
5. Funkci musím exportovat, abych ji mohl použít jinde v aplikaci

Procvičování

- Vytvoř novou react aplikaci pomocí VITE
  - `npm create vite@latest`
  - projdi průvodce
- Spusť aplikaci pomocí
  - `cd nazev`
  - `npm install`
  - `npm run dev`

Procvičování

- V aplikaci vytvoř a zobraz novou komponentu Menu, která bude vracet nadpis (h1), list (ul) libovolných stringů (li) a tlačítko (button)
- Text nadpisu ulož do proměnné a použij v JSX
- Po kliknutí na tlačítko se do konzole vypíše nadpis, který je uložený v proměnné

02

Array metody a key atríbut

Array (pole) metody

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- .map(), .filter(), .flat() a další
- Pro úpravu dat, které získáme (ze serveru) například jako pole objektů

Atribut key

- Pokud tvoříme pole komponent (např. pomocí .map()), React potřebuje pro každý prvek pole unikátní klíč (viz chyba v konzoli)
- Je to z toho důvodu, že když data aktualizujeme, (přidáváme, mažeme apod.) unikátní klíče umožňují správně identifikovat, o který prvek jde
- Nějaké *id* je ideální jako unikátní klíč
  - pokud ho máme v datech
- Ukázka na příkladu –>

Atribut key

```
const Menu = () => {
const menu = [“Úvod”, “O nás”, “Článek”]
return <ul>{
  menu.map(item=>{
   return <li key={item}>{item}</li>
  })
 }</ul>
}
```

Procvičování .map()

- Vytvoř si komponentu Colors
- Do ní vytvoř pole *colors* a přidej libovolné barvy (minimálně 5)
- Vypiš barvy do seznamu pomocí .map() metody
- Nezapomeň přidat *key*

Procvičování .map(), .filter()

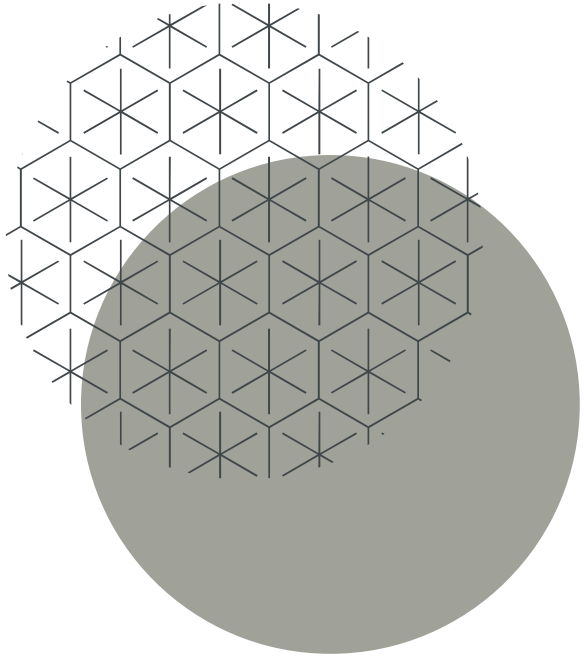
- Vytvoř si komponentu Numbers
- Do ní vytvoř pole numbers a přidej čísla od 1 do 10
- Vyfiltruj si čísla menší než 6 (pomocí .filter()) a vypiš je do seznamu (.map())
- Můžeme metody skládat za sebou, pro někoho může být přehlednější to dělat postupně
- Nezapomeň přidat *key*

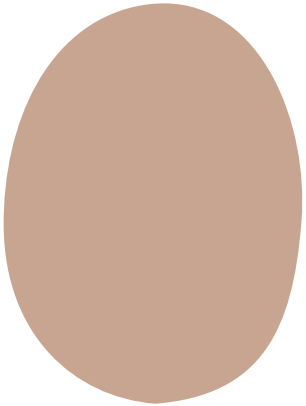
Procvičování .map(), .filter()

- Ulož soubor lide.js do složky src
- Vytvoř si komponentu People
- Ulož si data do proměnné people
- Vytvoř seznam jmen lidí, kteří jsou starší než 30 let
- Použij metody .filter() a .map()
- Do komponenty People vypiš jméno, příjmení i věk

03

Podmíněné renderování





Podmíněné renderování

- Podmínky
- Komponenty často renderují jiné věci v závislosti na nějaké podmínce
  - (ne)platná data
  - (ne)přihlášený uživatel
  - (ne)existující data
- Používáme proto JS syntax a operátory

`if, &&, ? :`

- Příklady použití v kódu ->

Procvičování (Podmíněné renderování)

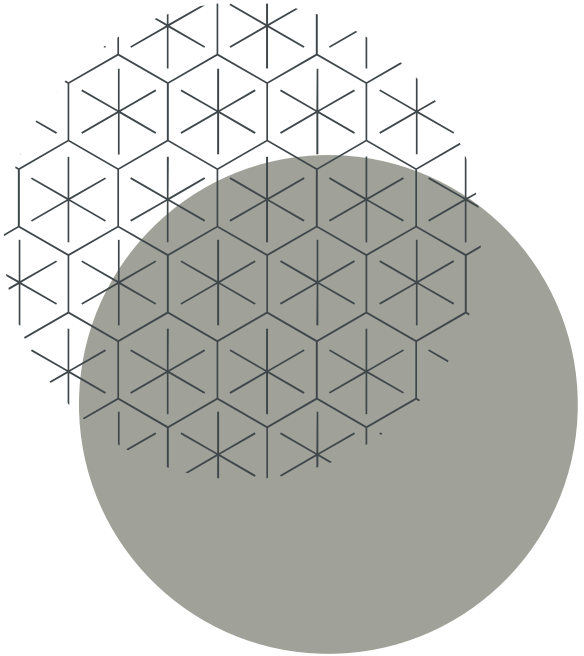
- Vytvoř komponentu List
- Do ní vytvoř prázdné pole, které si ulož do proměnné
- Napiš podmínku za použití if, kdy komponenta bude vracet null, pokud bude pole prázdné (využij .length vlastnost)
- Pokud v poli budou položky, bude komponenta vracet seznam (využij .map() metodu)

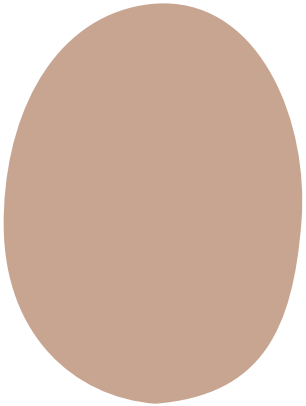
Procvičování (Podmíněné renderování)

- V komponentě List přepiš podmínku za použití `&&`
- Vidíš nějaký rozdíl v použití `if/&&` ?
- Vyzkoušej si podmínku napsat i za pomoci `? :`
- A přidej paragraf s textem “List je prázdný” v případě prázdného pole

04

Předávání dat komponentám

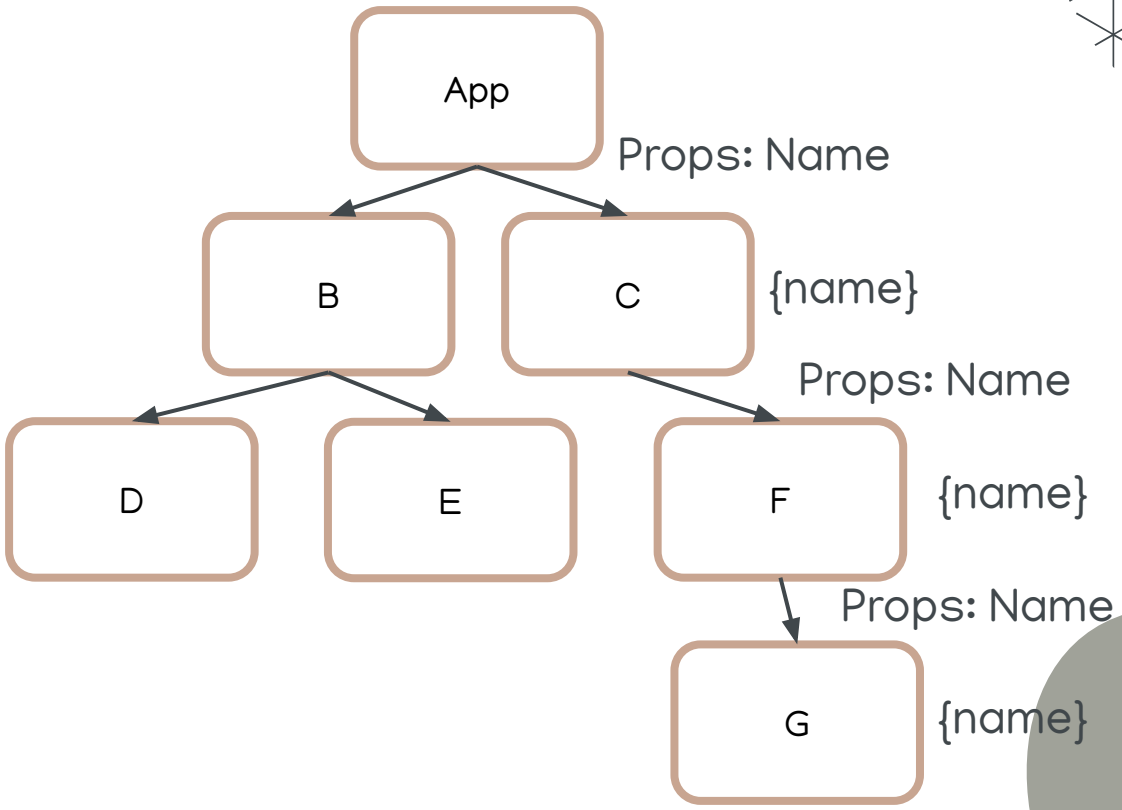




Předávání dat komponentám

- Komponenty využívají *props* na komunikaci mezi sebou
- Každá parent komponenta může předat informace/data svým child komponentám pomocí *props*
- Hodnota *props* může být jakákoliv JS hodnota (objekty, array, funkce apod.)
- Zvyšuje znovuvyužitelnost komponent

Props



Props

- Předání *props* z parent komponenty do child komponenty vypadá podobně jako v html přidání atributu
  - například
- V child komponentě jsou *props* jediný argument funkce
- *props* jsou vždy objekt
- Příklad na následujícím slidu –>

Props

```
export const App = () => {
    const name = "Ondra"
    return <Person name={name}>
}
export const Person = (props) => {
    return <p>You are {props.name}< /p>
}
```

Destructuring

Destructuring

```
const props = {
  name: 'Ondra',
  age: 22
  mood: “happy”
};
const {name, age} = props
console.log(name); // => 'Ondra',
console.log(age); // => 22
```

Props

```
export const Person = (props) => {
    return <p>You are {props.name}<p>
}
```

```
export const Person = ({name}) => {
    return <p>You are {name}<p>
}
```

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>
}
```

Props

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

```
export const Person = (props) => {
    return <p>You are {props.name},{props.age}<p>
}
```

Cvičení - string jako props

- Vytvoř komponentu User, která bude child od komponenty App
- User komponenta bude přijímat od parent komponenty *props* – firstName, lastName, age
- Použij destructuring syntax v User komponentě pro props
- Vrať všechny props v <p> tagu
- V App komponentě vytvoř údaje o uživateli tak, abys je mohl/a poslat pomocí props do User komponenty

Cvičení - funkce jako props

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

- Příklad použití v kódu -> použij Button v Users a vypiš data, která přišla v props

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

05

Příklady

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*

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

06

Shrnutí

Shrnutí

- Seznámili jsme se s *key* atributem potřebným při práci s poli
- Seznámili jsme se s renderováním komponenty s podmínkami
- Seznámili jsme se s *props* – způsobem jak si komponenty předávají data

Díky za pozornost!