React – přehled komponent a funkcí pro procvičování 14. 2. 24

Ve většině hotových komponent se objevují některé stejné properties. Zde je obecný přehled, který, bude-li třeba, u jednotlivých komponent bude specifikován:

- label – vždy použito jako text labelu/popisku ke komponentě
- id – vždy použito jako id komponenty
- dataIn – vstupní data komponenty
- handleData – property, která dále směřuje na funkci zpracovávající výstupní data z komponenty – předávají se jak data, tak id komponenty pro identifikaci
- handleEvent – property, která dále směřuje na funkci zpracovávající eventy – předává se id

Button.jsx

Komponenta tlačítka, reagující na kliknutí.

Properties:

- handleEvent předává id tlačítka, na které uživatel kliknul, podle id se pak spouštějí příslušné funkce

```
function Button({ label, handleEvent, id }) {
  const handleClick = () => {
    handleEvent(id);
  };
  return (
    <button
      type="button"
      id={id}
      onClick={handleClick}
      className="btn btn-outline-secondary"
    >
      {label}
    </button>
  );
}
```

ChbGroup.jsx

Komponenta skupiny checkboxů, reagujících na změnu stavu zaškrtnutí.

Properties:

- dataIn – pole objektů { label: string, value: string } – každý objekt reprezentuje jeden checkbox, mající svůj label a hodnotu
- selectedValue – pole hodnot označených checkboxů – provázání s proměnnou (pole) v nadřízené komponentě
- handleChange – pokud se změní stav checked konkrétního checkboxu – property předává pole hodnot aktuálně zaškrtnutých checkboxů do nadřízené komponenty

```
function ChbGroup({ label, id, dataIn, handleData, selectedValue }) {
  const handleChange = (e) => {
    const value = e.target.value;
    const isChecked = e.target.checked;
    if (isChecked) {
      handleData([...selectedValue, value], id);
    } else {
      handleData(
        selectedValue.filter((item) => item !== value),
        id
      );
    }
  };

  return (
    <div id={id}>
      <div className="mb-1">{label}</div>
      {dataIn.map((item, index) => (
        <div key={index} className="form-check form-check-inline">
          <input
            className="form-check-input"
            type="checkbox"
            id={item.label}
            value={item.value}
            onChange={handleChange}
            checked={selectedValue.includes(item.value)}
          />
          <label className="form-check-label" htmlFor={item.label}>
            {item.label}
          </label>
        </div>
      ))}
    </div>
  );
}
```

Clock.jsx

Komponenta zobrazující span element s aktuálním časem.

```
function Clock() {
  const [currentTime, setCurrentTime] = useState(new
Date().toLocaleTimeString());

  useEffect(() => {
    const timerId = setInterval(() => {
      setCurrentTime(new Date().toLocaleTimeString());
    }, 1000);

    return () => {
      clearInterval(timerId);
```

```
    };
  }, []);

  return <span>{currentTime}</span>;
}
```

File.jsx

Komponenta, která načte text z .txt souboru. Text musí být ve formátu UTF-8.

Properties:

- handleData – načtený text a id se předávají do nadřízené komponenty

```
function File({ label, handleData, id }) {
  const handleChange = (ev) => {
    const file = ev.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.readAsText(file);
      reader.onload = (e) => {
        const text = e.target.result;
        handleData(text, id);
      };
    }
  };
  return (
    <>
      <input
        className="form-control"
        type="file"
        id={id}
        onChange={handleChange}
      />
      <label htmlFor={id} className="form-label">
        {label}
      </label>
    </>
  );
}
```

Image.jsx

Komponenta obrázku.

Properties:

- source – naimportovaný obr.
- width – preferovaná šířka, např. ‘100%’
- enabled – boolean prop, aby bylo jasné, zda má být obr. vykreslen

```
function Image({ source, width, enabled, id }) {
  if (!enabled) {
    return null;
  }

  return (
    <img src={source} width={width} alt={id} id={id} />
  );
}
```

NumImp.jsx

Komponenta číselného inputu, který reaguje na změnu hodnoty.

properties:

- dataIn – provázání s proměnnou v nadřízené komponentě
- handleData – předává novou hodnotu do nadřízené komponenty

```
function NumImp({ dataIn, label, handleData, id }) {
  const handleChange = (e) => {
    handleData(e.target.value, id);
  };

  return (
    <>
      <label className="form-label" htmlFor={id}>
        {label}
      </label>
      <input
        type="number"
        className="form-control"
        value={dataIn}
        id={id}
        onChange={handleChange}
      />
    </>
  );
}
```

ProgressBar.jsx

Komponenta reprezentující nějaký inkrement nebo dekrement.

Properties:

- dataIn – hodnota předaná z nadřízené komponenty 0-100, reprezentující procentuální vyplnění komponenty

```
function ProgressBar({ dataIn, id }) {
  return (
    <div
```

```
      id={id}
      className="progress"
      role="progressbar"
      aria-label="Animated striped example"
      aria-valuenow={dataIn}
      aria-valuemin="0"
      aria-valuemax="100"
    >
      <div
        className="progress-bar progress-bar-striped progress-bar-animated"
        style={{ width: `${dataIn}%` }}
      ></div>
    </div>
  );
}
```

Range.jsx

Komponenta s daným rozsahem – posuvníkem se mění hodnota.

Properties:

- dataIn – provázáno s hodnotou v nadřízené komponentě
- min, max – hranice rozsahu

```
function Range({ min, max, label, handleData, dataIn,id }) {
  const handleChange = (e) => {
    handleData(e.target.value, id);
  };

  return (
    <>
      <label className="form-label" htmlFor={id}>
        {label}
      </label>
      <input
        type="range"
        className="form-range w-100"
        min={min}
        max={max}
        onChange={handleChange}
        id={id}
        value={dataIn}
      />
    </>
  );
}
```

RbGroup.jsx

Komponenta skupiny radiobuttonů, reagujících na změnu stavu zaškrtnutí.

Properties:

- dataIn – pole objektů { label: string, value: string } – každý objekt reprezentuje jeden radiobutton, mající svůj label a hodnotu
- selectedValue – hodnota označeného checkboxů – provázání s proměnnou v nadřízené komponentě
- handleChange – pokud se změní stav checked konkrétního rb – property předává hodnotu aktuálně zaškrtnutého rb do nadřízené komponenty

```
function RbGroup({ label, id, dataIn, handleData, selectedValue }) {
    const handleChange = (e) => {
      handleData(e.target.value, id);
    };

    return (
      <div id={id}>
        <div className="mb-1">{label}</div>
        {dataIn.map((item, index) => (
          <div key={index} className="form-check form-check-inline">
            <input
              className="form-check-input"
              type="radio"
              name={id}
              id={item.label}
              value={item.value}
              onChange={handleChange}
              checked={selectedValue === item.value}
            />
            <label className="form-check-label" htmlFor={item.label}>
              {item.label}
            </label>
          </div>
        ))}
      </div>
    );
  }
```

Select.jsx

Komponenta input selectu.

Properties:

- dataIn – pole hodnot
- selectedValue – hodnota označeného option provázaná s proměnnou v nadřízené komponentě
- handleChange – předání hodnoty aktuálního označeného option

```
function Select({ dataIn, label, handleData, selectedValue, id }) {
  const handleChange = (e) => {
    handleData(e.target.value, id);
  };

  return (
```

```
    <>
      <label className="form-label" htmlFor={id}>
        {label}
      </label>
      <select
        className="form-select"
        id={id}
        onChange={handleChange}
        value={selectedValue}

      >
        {dataIn.map((item, index) => (
          <option key={index} value={item}>
            {item}
          </option>
        ))}
      </select>
    </>
  );
}
```

TextArea.jsx

Text area pro práci s textem.

Properties:

- handleChange – předává aktuální hodnotu textu do nadřízené komponenty do nějaké proměnné
- dataIn – provázání s proměnnou v nadřízené proměnné
- height – výška komponenty v pixelech

```
function TextArea({ label, dataIn, handleData, height, id }) {
  const handleChange = (e) => {
    handleData(e.target.value, id);
  };
  return (
    <>
      <label className="form-label" htmlFor={id}>
        {label}
      </label>
      <textarea
        className="form-control"
        onChange={handleChange}
        value={dataIn}
        id={id}
        style={{ height: `${height}px` }}
      />
    </>
  );
}
```

TextArea.jsx

Komponenta input textu.

Properties:

- handleChange – předává aktuální hodnotu textu do nadřízené komponenty do nějaké proměnné
- dataIn – provázání s proměnnou v nadřízené proměnné

```
function TextBox({ label, dataIn, handleData, id }) {
  const handleChange = (e) => {
    handleData(e.target.value, id);
  };
  return (
    <>
      <label className="form-label" htmlFor={id}>
        {label}
      </label>
      <input
        type="text"
        className="form-control"
        onChange={handleChange}
        value={dataIn}
        id={id}
      />
    </>
  );
}
```

---

saveText.js

Funkce pro stažení souboru s příponou .txt.

Fci předáváme parametr text – string. nejdříve převede tento string na blob objekt typu text/plain. Poté vytvoří URL – cestu k tomuto blob objektu. Vytvoříme si element odkazu tuto URL.

Nastavíme jméno stahovaného souboru. Odkaz si hodíme do dokumentu a nastavíme na něj klik – začne stahování souboru. Poté odstraníme tento element z dokumentu a uvolníme vytvořenou URL.

```
const saveText = (text) => {
  const blob = new Blob([text], { type: "text/plain" });
  const fileDownloadUrl = URL.createObjectURL(blob);
  const link = document.createElement("a");
  link.href = fileDownloadUrl;
  link.download = "downloaded_text.txt";
  document.body.appendChild(link);
  link.click(); // Iniciuje stahování
  document.body.removeChild(link);
  URL.revokeObjectURL(fileDownloadUrl);
}
```

validateFloat.js

Funkce vracející boolean, zda je hodnota validní float number. Kontrolujeme jednak, zda se jedná vůbec o číslo a také zda se parseFloat nezbavil nějakých zadaných nečíselných částí vstupního stringu.

```
const validateFloat = (input) => {
  const num = parseFloat(input);
  return !isNaN(num) && input == num;
}
```

validatePositiveInt.js

Funkce vracející boolean, zda je hodnota validní nezáporné int number. Kontrolujeme jednak, zda se jedná vůbec o číslo a také zda se parseInt nezbavil nějakých zadaných nečíselných částí vstupního stringu. (např.: parseInt(10aaaaa) === 10)

```
const validatePositiveInt = (input) => {
  const num = parseInt(input);
  return !isNaN(num) && input == num && num > 0;
}
```