

Übungsbeispiel

Aufbauend auf die letzte Übung „Zellen mit Context“ folgende Anforderungen für einen Rechentrainer mit Context.

Anforderungen

Eingabe

- Nutze die 9 Zellenkomponente um Rechnungsergebnisse einzugeben.
- Lege eine Rechenvorschrift mit den Zellen und Spaltenziffern fest
z.B. Spalte * Zeile (2 * 3 = 6)
- Lass die Ergebnisse eingeben

Auswerten

- Werte auf ButtonKlick aus
- Färbe Ergebnisse rot (falsch) und grün (richtig) ein und gib pro richtigen Ergebnis einen Punkt. Zeige die erreichten Punkte an
- (optional) Gib Ergebnisse in den LocalStorage. Ermögliche mit Button up und down Historieergebnisse durch zu gehen

SpielstandHighscore

- Richte einen Button Score ein der
 - Lege erreichten Punkte mit Namen versehen ab
 - Nutze einen eigenen Context für die Ablage
 - Zeige in einer eigenen Komponente die Highscoretabelle an

Achtung Context: Übe mit richtigen Datentypen, indem die Attribute und Methoden des verwendeten Context richtig angegeben werden.

...

```
import {createContext, useState, ReactNode, useContext} from 'react';
```

```
type Language = 'de' | 'en';
```

```
interface LanguageContextType {  
  language: Language;  
  texts: typeof texts['de'];  
  setLanguage: (lang: Language) => void;  
}
```

```
const texts = {  
  de: { greeting: 'Hallo', selectColor: 'Wähle eine Farbe', green: 'Grün', red: 'Rot', clickCell: 'Klicke auf eine Zelle', occupied: 'ist besetzt. Wähle eine andere.' },  
  en: { greeting: 'Hello', selectColor: 'Select a color', green: 'Green', red: 'Red', clickCell: 'Click on a cell', occupied: 'is occupied. Choose another.' },  
};
```

```
export const LanguageContext = createContext<LanguageContextType | undefined>(undefined);
```

```

export const LanguageProvider = ({ children }: { children: ReactNode }) => {
  const [language, setLanguage] = useState<Language>('de');

  return <LanguageContext.Provider value={{language, texts: texts[language], setLanguage}}>
    {children}
    <button onClick={() => setLanguage('de')}>Deutsch</button>
    <button onClick={() => setLanguage('en')}>English</button>
  </LanguageContext.Provider>;
};

```

Beispiel Nutzen von Local Storage in einer Komponente

```

import React, { useState, useEffect } from 'react';

const MyComponent = () => {
  // Initialzustand aus localStorage laden
  const [name, setName] = useState(() => {
    return localStorage.getItem('name') || ''; // Leerer String, falls kein Wert vorhanden
  });

  return (
    <div>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
    </div>
  );
};

```

Denke an das CheatSheet mit den MUI Komponenten

Aus dem DCCS Projekt

1) Einfügen der Material Komponenten

Dialogkomponente

<https://mui.com/material-ui/react-dialog/#introduction>

Bei Abbrechen „no“ bleibt er in der Details und bei „ja“ geht er über Lifting nach oben und stellt wieder die Overview dar.

a) Dialogkomponente (Richtig kopieren braune Zeilen sind Hilfe)

```

return (
  <>
    <Dialog open={isCancelDialogOpen}>
      <DialogTitle>Confirmation dialog</DialogTitle>
      <DialogContent>
        Are you sure you want to cancel?
      </DialogContent>
    </Dialog>
  </>
);

```

```

<DialogActions>
  <Button onClick={() => setIsCancelDialogOpen(false)}>No</Button>
  <Button onClick={() => { onCancel(); setIsCancelDialogOpen(false)}} autoFocus>
    Yes
  </Button>
</DialogActions>
</Dialog>

<Grid2 className={classes.root} container direction="column">
  <Grid2 className={classes.header}>

```

b) useState vorher noch rein setzen

```
const [isCancelDialogOpen, setIsCancelDialogOpen] = useState(false)
```

c) Im Grid des Abbrechen Buttons die setIsCancelDialogOpen reinsetzen

```

<Button className={classes.cancelButton} onClick={() => setIsCancelDialogOpen(true)} variant={"contained"}
color={"error"}>Cancel</Button>

```

GridCol Komponente

<https://mui.com/x/api/data-grid/grid-col-def/>

Wird eingefügt, um die Spalten in der Overview einfach zu sortieren.

Wichtig sind die renderCell Werte

```

const columns: GridColDef<Movie>[] = [
  {
    sortable: false,
    resizable: false,
    disableColumnMenu: true,
    width: 40,
    headerName: "",
    field: "isFavorite",
    renderCell: (value: GridRenderCellParams<Movie>) => {
      return (
        <span className={classes.icon} onClick={e => {
          e.stopPropagation()
          value.row?.id && toggleFavorite(value.row?.id)
        }}>
          {value.row?.isFavorite ? <Star/> : <StarOutline/>}
        </span>
      )
    },
  },
  {
    sortable: false,
    resizable: false,
    disableColumnMenu: true,
    width: 40,
    headerName: "",

```

```

        field: "imageUrl",
        renderCell: (value: GridRenderCellParams<Movie>) => {
            return <img className={classes.image} src={value.row?.imageUrl} alt={"pic"} />
        }
    },
    {
        sortable: false,
        resizable: false,
        disableColumnMenu: true,
        width: 400,
        headerName: "Title",
        field: "title",
        renderCell: (value: GridRenderCellParams<Movie>) => `${value.id}. ${value.row?.title}`
    },
    {
        sortable: false,
        resizable: false,
        disableColumnMenu: true,
        width: 70,
        headerName: "Year",
        field: "year"
    },
    {
        sortable: false,
        resizable: false,
        disableColumnMenu: true,
        width: 110,
        headerName: "Watched",
        field: "isWatched",
        renderCell: (value: GridRenderCellParams<Movie>) => {
            return (
                <span className={classes.icon} onClick={e => {
                    e.stopPropagation()
                    value.row?.id && toggleWatched(value.row?.id)
                }}>
                    {value.row.isWatched ? <BookmarkAdded /> : <BookmarkBorder />}
                </span>
            )
        }
    },
    {
        sortable: false,
        resizable: false,
        disableColumnMenu: true,
        width: 40,
        headerName: "",
        field: "delete",
        renderCell: (value: GridRenderCellParams<Movie>) => {
            return (
                <span className={classes.iconDelete} onClick={e => {
                    e.stopPropagation()
                    value.row?.id && onDeleteMovie(value.row?.id)
                }}>
                    <Delete />
                </span>
            )
        }
    }
}

```

```
}
]
```

wichtig im Return einzufügen (Richtig kopieren braune Zeilen sind Hilfe)

```
}}>Add movie</Button>
</Grid2>
```

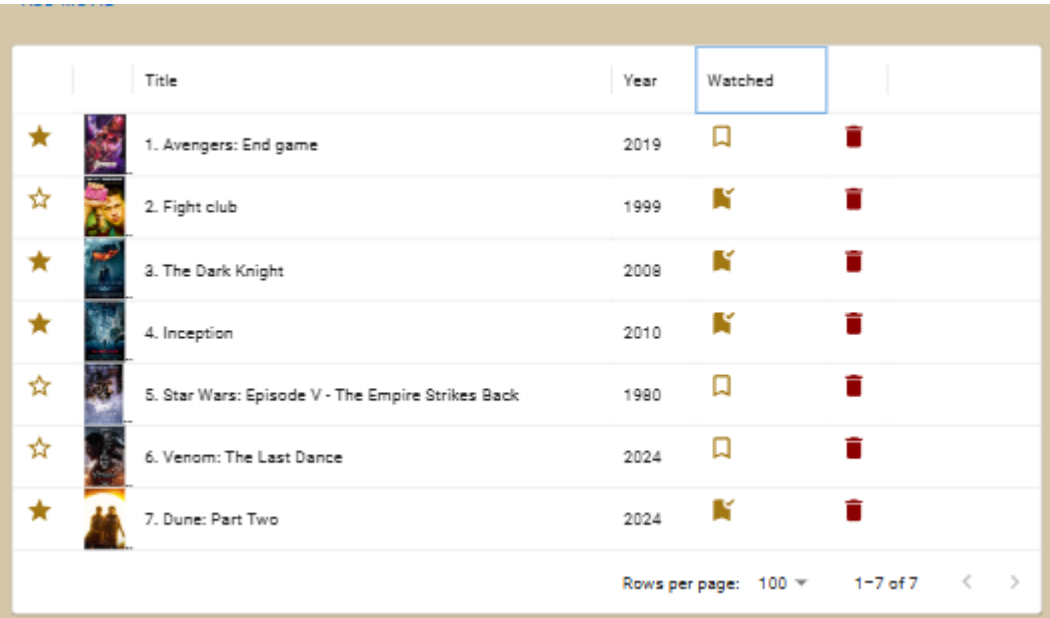
```
<Grid2 container direction={"column"} gap={"10px"} paddingX={"20px"}>
  <Paper style={{width: "100%"}}>
    <DataGrid
      columns={columns}
      rows={movies}
      onClick={({params}) => onOpenDetails(params.row)}
    />
  </Paper>
</Grid2>
```

```
<Grid2 padding={"20px 0"}>
```

Weiters die Komponenten Delete, Star, StarOutlined, BookmarkAdded, BookmarkBorder zu importieren.

Sortiert wird mit indem Sortable = true eingestellt wird bei dem jeweiligen Attribut.

Ergebnis:



	Title	Year	Watched	
★	1. Avengers: End game	2019	🔖	🗑️
☆	2. Fight club	1999	🔖	🗑️
★	3. The Dark Knight	2008	🔖	🗑️
★	4. Inception	2010	🔖	🗑️
☆	5. Star Wars: Episode V - The Empire Strikes Back	1980	🔖	🗑️
☆	6. Venom: The Last Dance	2024	🔖	🗑️
★	7. Dune: Part Two	2024	🔖	🗑️

Rows per page: 100 ▾ 1-7 of 7 < >

Material Komponente Snackbar

<https://mui.com/material-ui/react-snackbar/#introduction>

In der App.tsx (Richtig kopieren braune Zeilen sind Hilfe)

```
<Grid2 className={classes.root} container direction={"column"} alignItems={"center"}>
  <Snackbar
    anchorOrigin={{ vertical: "top", horizontal: "center"}}
    open={isSnackbarOpen}
    autoHideDuration={3000}
    TransitionComponent={Slide}
    onClose={() => setIsSnackbarOpen(false)}
  >
    <Alert
      onClose={() => setIsSnackbarOpen(false)}
      severity="success"
      variant="filled"
      sx={{ width: '100%' }}
    >
      Movie successfully saved
    </Alert>
  </Snackbar>
</AppHeader onClick={handleBackToOverview} />
```

Wichtig: Import von Snackbar, Alert und Slide

Anpassen der Methode handleBackToOverview

```
const handleBackToOverview = (): void => {
  setSelectedMovie(undefined)
  setSelectedView(AppViews.MovieOverview)
  setIsSnackbarOpen(true)
}
```

Einfügen des useState

```
const [isSnackbarOpen, setIsSnackbarOpen] = useState(false)
```