Functional React - Basics

https://reactjs.org/

1 Project

- Repository: https://github.com/paula-andra/functional-react.git
- StackBlitz: https://stackblitz.com/edit/node-rxidke?file=src/index.tsx
  - (!) the shortcuts for Visual Studio work in StackBlitz
- yarn scripts:
  - start the application using the port 8081:

```
yarn && yarn start
```

  - run the tests from certain files:

```
yarn [Regex], z.B. yarn list wird alle Teste in den Dateien, die list beinhalten.
```

2 JSX

- User defined component must be capitalized
- Any JavaScript expression can be used as property
- Any JavaScript expression can be used as a child element, as long as it is enclosed in
- boolean, null, and undefined are valid child components, but will be ignored when rendering
- <React.Fragment></React.Fragment> - used to group components together without DOM element; the only allowed attribute is key
- <></> - the short version for React.Fragment, but it is missing the key attribute
- List elements should use keys to uniquely identify one element
- Html elements can be used as child element
  - properties and attributes, including event handlers, should be camelCased;
  - any standard or custom DOM attributes are fully supported
  - The value attribute is useful for building controlled components and is supported by <input>, <select> and <textarea> components.
- Conditional rendering

```
{[condition] && [ReactNode]} // [ReactNode] will be rendered only if [condition] is truthy
```

3 Components examples

```
const SimpleComponent: VFC<{ name: string }> = ({ name }) => <Row>{name}</Row>;

const ComponentWithChildren: FC<{ name: string }> = ({ name, children }) => <Row>{name} {children}</Row>;

const ComponentWithChildrenWithProps: VFC<{ name: string, children?: (props: { name: string }) => ReactNode | undefined }> =
  ({ children, ...props }) => {
    return <Row> {children && children(props)} </Row>;
  };
```

4 useState Hook

```
const [count, setCount] = useState(0);
setCount(10);//sets the variable count to 10
setCount(prevCount => prevCount + 1);//increases count by 1
```

5 useEffect Hook

```
useEffect(() => {
    // code to perform
    // Specify how to clean up after this effect:
    return () => {
      // clean up code
    };
}(, dependencies));
```

- The optional list of dependencies determines which variables need to change s.t. the effect runs; otherwise the effect will be skipped at rendering.
  - If dependencies is an empty array, the effect will be called only once.
- If the dependencies parameter is missing, the effect will be called on every rendering