Workshop 02 script

What is the idea?

- write the interface entirely in javascript, React will "compile" them into actual html
- structure the code into smaller cohesive parts that describe reusable components
  -> ideally, you'll never have to directly touch the DOM

1: Create React-App Toolchain

- apart from the core library, we'll use:
  - a package manager to easily manage third party stuff
  - a bundler to structure the code into several parts
  - a compiler for JSX
- Open console in application folder
- enter `npx create-react-app appname`

Structure of the app

- index.js is the entry point of the app
- it doas only one thing: it calls ReactDOM.render(). The First Argument is the element it calls it on. The second argument is the parent element

JSX

- JSX is not HTML, it is Javascript that looks like html (the compiler will translate it as such)
- JSX is a valid JS expression: it can be returned from any function. Under the hood, it comples to a React.createElement() call that creates a react element which is than rendered as an actual html element
- Attributes: as a string
- empty Tags may be closed immediately
- there has to be one container (components won't appear in the Dom, only the contents!)

Components

- two types of components: function components and stateful components
- function components are functions that return the JSX and take props as an argument
- class components beave like objects that have one method that returns the JSX and a constructor that takes props as an argument

- the "display" part (the JSX) of a function component is the return value
- class components hav a method calles render(), which has the JSX as return value

Component lifecycle

- Components hav a lifecycle: they initialze, they update, they may get destroyed
- any setup of a component's content can only be done after the component itself is initialized.
- for those purposes, React.Component has serveral lifecycle-methods

props

- function components
- props must not be changed by the component itself!

state

- the state is a "private" property of a component
- the state is only controlled by te property itself
- state can ony be used by class components (because function components are really just functions)

->

Paasing data between components

-> React.createRef + ref-Attribute