1. Fundamental of React

# 1.1 Hello React

-**SPA** (Single-page application) made it easier to build web app that advanced beyond vanilla JS and JQueryy. **React** was released by Facebook in **2013**.

-In the past, websites were **rendered** from **server**: user visits URL in browser and requests one HTML file and associated files. After network delay, users see the rendered HTML in browser (client).

-Modern JS shifted the focus from **server** to **client**. A user visits URL and request one small HTML file and one larger JS file. After network delay, user see the by JS rendered HTML in browser. Additional page transition wouldn’t request more files from web server, but would use initially requested JS to render new page. Every additional interaction by user is handled on client. In this modern version, the server delivers mainly JavaScript across the wire with one minimal HTML file. HTML file then executes all linked JS from files on client-side to render entire app.

-SPA is one bulk of JS, which is organized in folders and files, to create app whereas SPA framework (React) gives all tools to architect it. React takes over for rendering everything in browser as HTML and for dealing with user interactions with JS.

-How we moved from websites to web app: https://www.robinwieruch.de/web-applications/

# 1.2 Requirements

-**Editor** and **Terminal**: Visual Studio Code

-**VCS**: Github

<https://www.robinwieruch.de/git-essential-commands/>

-**Node** and **NPM**

+Both are used to manage libraries (node packages) you will need. These node packages can be libraries or whole frameworks. We install external node packages via npm

+node –version and npm –version.

<https://www.robinwieruch.de/npm-crash-course/>

-**Yarn** and **pnpm**: <https://yarnpkg.com/> https://pnpm.io/

# 1.3 Setting up a React Project

-We use **Vite** to **set up React application**: vite.dev

+Vite is a modern build tool for status quo web frameworks which comes with sensible defaults while staying highly extensible for specific use cases (SVG support, Lint support, TypeScript)

+The essential core of Vite is development server ( start React app on local machine) and a bundler (outputs optimized files for production ready deployment)

-There are 2 ways to create project with Vite: online template (vite.dev/guide/#trying-vite-online) and creating a React project with Viton on local machine for working on it in IDE/editor.

-Crash course for navigating on command line:

A white background with black text

AI-generated content may be incorrect.

**-Create React project** hacker-stories



-**Install** all 3rd party **dependencies** of project and run it locally:

A white sign with black text

AI-generated content may be incorrect.

# 1.4 Project Structure

-code .: see project structure

A close up of a document

AI-generated content may be incorrect.

A text on a white background

AI-generated content may be incorrect.

-src/App.jsx used to implement React components. Later you want to split up React components into multiple files, each file maintains one or more components on its own.

-src/main.jsx is entry point to React world.

-src/index.css and src/App.css style overall app and components.

# 1.5 npm Scripts

-All project-specific commands can be found in package.json under scripts:

A screenshot of a computer code

AI-generated content may be incorrect.

+these script are executed with **npm run <script>:**

A black text on a white background

AI-generated content may be incorrect.

-preview: run production-ready build on local machine for testing purposes

+npm run build -> npm run preview

# 1.6 Meet the React Project

-Every React app is built on the foundation of React components. The 1st React component which is located in src/App.jsx.

-This file will be our focus throughout this book.

-Reduce the component to a more lightweight version

A screenshot of a computer program

AI-generated content may be incorrect.

-Optionally you can make src/index.css and src/App.css file bank for starting from a clean state style-wise.

-App component is just a JS function, it’s defined in PascalCase. Component has to start with a capital letter. App component is a function component: the modern way of using component in React.

-A function component can have implementation details between function signature and return statement:

A screenshot of a computer code

AI-generated content may be incorrect.

-Variable defined in function’s body will re-defined each time this function runs.

A screenshot of a computer code

AI-generated content may be incorrect.

-The function of a component runs every time a component is displayed in browser. This happens for the initial rendering of component, but also whenever the component updates because it has to display sth in different due to changes (re-rendering)

-Since we don’t want to re-define a variable within a function every time the function runs, we could define this variable outside of components as well.

A screenshot of a computer code

AI-generated content may be incorrect.

# 1.7 React JSX

-JSX (JavaScript XML): combine HTML and JS

A screenshot of a computer code

AI-generated content may be incorrect.

-The bridge between React and development server is React Fast Refresh (prior to that it was React Hot Loader) on React’s side and Hot Module Replacement on development server’s side.

-HTML input field and HTML label:

A screenshot of a computer program

AI-generated content may be incorrect.

+htmlFor reflects the for attribute in HTML. JPX replaces a handful of internal HTML attributes caused by internal implementation details of React.

-Find all supported HTML attributes in React documentation: [DOM Elements – React](https://legacy.reactjs.org/docs/dom-elements.html#all-supported-html-attributes)

-React use camelCase naming convention: [Camel case - Wikipedia](https://en.wikipedia.org/wiki/Camel_case)

-When using HTML in JSX, React translate all HTML attributes to JS where certain words such as class or for are reserved during rendering process. Therefore React came up with replacement like className and htmlFor. Once the actual HTML is rendered for browser, the attributes get translated back to their native variant.

A diagram of a programming language

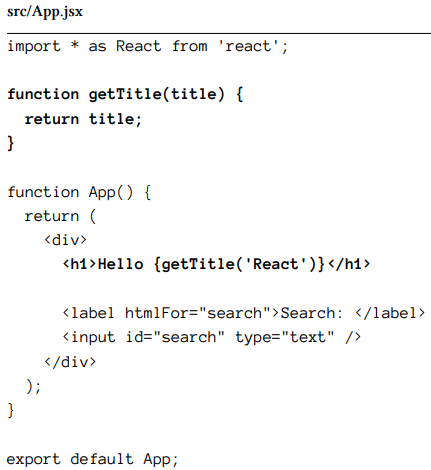
AI-generated content may be incorrect.

-Access properties within JSX:

A screenshot of a computer code

AI-generated content may be incorrect.

-While HTML can be used almost (except for the attributes) in its native way in JSX, everything in curly braces can be used to interpolate JS in it.



-The underlying build tools can be configured to acknowledge JSX in .js file. If they are configured this way, they will transpile JSX to JS. Tools like Vite embrace the .jsx extension though.

A screen shot of a computer code

AI-generated content may be incorrect.

-JSX is the favorite things when being asked about React. Without any extra templating syntax (except curly brace), we can use JS in HTML. Every JS data structure can be used within HTML with JSX.

# 1.8 Linting with ESlint (Optional)

-Linting: process in programming where code is analyzed for potential errors, bugs, and style issues.

-ESLint is popular linting tool for JS.

-Install the respective plugin:

A close-up of a plugin

AI-generated content may be incorrect.

-vite.config.js: allow us to customize the development and build process of a Vite-based project. It gives us options such as setting public path, configure plugins, and modify the build ouput.

A screenshot of a computer code

AI-generated content may be incorrect.

-Install the ESlint dependency: 

-Install one of ESLint’s many standardized linting configurations for React project:

A close up of a sign

AI-generated content may be incorrect.

-If you start project on cmd again, you will see error:



-Therefore we will create ESLint configuration file to define our linting rules: 

A screenshot of a computer code

AI-generated content may be incorrect.

-When starting app on cmd, you will see warning:

A black text on a white background

AI-generated content may be incorrect.

-In VSCode, you can install ESLint Extension.

# 1.9. Lists in React

-When working with JS, most often data is array of objects. **map()**: iterate over each item of a list in order to return a new version of each item

A screenshot of a computer code

AI-generated content may be incorrect.

-Render each object with its title property in React by map() in JSX:

A screen shot of a computer code

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

-Without any made up templating syntax, it’s possible to use JS to map from a list of items to a list of HTML elements. That’s what JSX is for developer in the end: just JS mixed with HTML

A diagram of a diagram

AI-generated content may be incorrect.

-key is an HTML attribute and should be a stable identifier:

A computer screen shot of a code

AI-generated content may be incorrect.

+The **key attribute** used for one specific reason: whenever React has to re-render a list, it checks whether an item has changed. When using keys, React can exchange the changed item.

A close-up of some words

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

+Using index should be avoided though, because it comes with the same rendering performance issues, it can cause actual bugs in UI when order of items got changed.

A computer screen shot of a program

AI-generated content may be incorrect.

# 1.10 Meet another React Component

-Instead of making one component larger and more complex, we’ll split one component into multiple components. We will start with a new List component which extracts functionalities from App component:

A screen shot of a computer program

AI-generated content may be incorrect.

-List component can be used in App:

A screenshot of a computer code

AI-generated content may be incorrect.

-Create Search component:

A screenshot of a computer code

AI-generated content may be incorrect.A black text on a white background

AI-generated content may be incorrect.

-React app consists of many hierarchical components

A diagram of a child development

AI-generated content may be incorrect.

# 1.11 React Component Instantiation

-A class is most often used in OOP languages. JS as a multi-paradigm programming language allows functional programming and OOP to co-exist side-by-side.

A computer screen shot of a code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.A close-up of a computer code

AI-generated content may be incorrect.

-The concept of JS class with declaration and instantiation is similar to React component, which also has only one component declaration, but can have multiple component instances:

A screenshot of a computer program

AI-generated content may be incorrect.

-One we defined component, we can use it as an element in JSX. The element produces an instance of component. You can create as many instances of a component as you want as long as you have a component declaration. It’s not much different from JS class declare + instantiate. But technically JS class and React component are not the same.

# 1.12 Handler Function in JSX

# 1.13 React Props

# 1.14 React Stage

# 1.15 Callback Handlers in JSX

# 1.16 Lifting State in React

# 1.17 React Controlled Components

# 1.18 Props Handling (Advanced)

# 1.19 React Side-Effects

# 1.20 React Custom Hooks (Advanced)

# 1.21 React Fragments

# 1.22 Reusable React Component

# 1.23 React Component Composition

# 1.24 Imperative React

# 1.25 Inline Handler in JSX

# 1.26 React Asynchronous Data

# 1.27 React Conditional Rendering

# 1.28 React Advanced State

# 1.29 React Impossible States

# 1.30 Data Fetching with React

# 1.31 Data Re-Fetching in React

# 1.32 Memoized Functions in React (Advanced)

# 1.33 Explicit Data Fetching with React

# 1.34 Third-Party Libraries in React

# 1.35 Async/Await in React

# 1.36 Form in React