REACT – The complete guide - Maximilian

React is a library for creating highly reactive and superfast js based web applications.

JS runs in the browser.

React is a JS library for building user interfaces

React is writing custom HTML elements

React is all about writing components

Codepen.io – to create a workspace for writing html/css/js code

ReactDOM is for rendering the components to real DOM

In its basic form, a React component is just a function.

The syntax React uses is called jsx.

Html --

<div id="p1"> </div>

<div class="person">

<h1> Hello </h1>

<p> age: 29 </o>

</div>

<div id='p2'> <h3> Romeo </h3> </div>

--

Css –

.person {

color: red

}

---

Js –

function Person(){

return (

<div class=”person”>

<h1> Max </h1>

<p> Your age: 28 </p>

</div>

);

}

ReactDOM.render(<Person />, document.querySelector('#p1'));

Output:

# Max

Your age: 28

# Hello

age: 29

### Romeo

Html –

<div id="p1"> </div>

<div class="person">

<h1> Hello </h1>

<p> age: 29 </o>

</div>

<div id='p2'> <h3> Romeo </h3> </div>

--

Css --

.person {

color: red

}

.person1 {

color: green

}

--

function Person(props){

return (

<div class="person1" >

<h1> {props.name} </h1>

<p> {props.age} </p>

</div>

);

}

ReactDOM.render(<Person name="Jaison" age="45" />, document.querySelector('#p1'));

---

# Jaison

45

# Hello

age: 29

### Romeo

Html –

<div id='app'</div>

--

Css –

.person {

color: red

}

.person1 {

color: orange

}

---

function Person(props){

return (

<div class="person1" >

<h1> {props.name} </h1>

<p> {props.age} </p>

</div>

);

}

var app = (<div> <Person name="Jaison" age="45" /><Person name="Sonia" age="39" /> </div>)

ReactDOM.render(app, document.querySelector('#app'));

---

Output:

# Jaison

45

# Sonia

39

In 3 ways, the component is written. 1) by function and render. 2) by function with parameter and render with argument. 3) create function with parameter and define the property, declare variable with function property definition, render with variable. With this approach, can create single page applications.

Angular an Vue are alternatives to React. JQuery is more about traversing the DOM and targeting the DOM elements.

Single page and multi page applications.

In a single page application, the root component manages the entire page.

Redux is a third party library

Jsbin.com is a web editor

When ‘this’ keyword is used with arrow functions, it always keeps its context.

Classes are blueprints for js objects. Properties are like variables attached to objects, methods are like functions attached to objects.

‘spread’ operator is used to split up array elements or object properties.

‘rest’ is used to merge a list of function arguments into an array

‘destructuring’ is extract array elements or object properties and store them in variables.

‘jsx’ is html in js syntax

Dependency management tool: npm or yarn. Bundler is Webpack. Native js compiler is Babel

Npm install –g create-react-app

Create-react-app my-app –scripts-version 1.1.5

Cd my-app // npm start

The nodes-module holds all the dependencies and sub-dependences of the project

The ‘public’ folder is surfed by the web server

Manifest.json is the metadata about the application

React.createElement(‘h1’,{ object} / null,’Hello’)

--

import React, { Component } from 'react';

import './App.css';

class App extends Component {

render() {

return (

// <div className="App">

// <h1> Hi, I am a React App!!</h1>

// </div>

React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

In JSX, use className. Class is reserved in JS

JSX elements are not HTML. React library provides it.

A component is just a function that returns a jsx (html)

Components & JSX Cheat Sheet

Components are the **core building block of React apps**. Actually, React really is just a library for creating components in its core.

A typical React app therefore could be depicted as a **component tree** - having one root component ("App") and then an potentially infinite amount of nested child components.

Each component needs to return/ render some **JSX** code - it defines which HTML code React should render to the real DOM in the end.

**JSX is NOT HTML** but it looks a lot like it. Differences can be seen when looking closely though (for example className in JSX vs class in "normal HTML"). JSX is just syntactic sugar for JavaScript, allowing you to write HTMLish code instead of nested React.createElement(...) calls.

When creating components, you have the choice between**two different ways:**

1. **Functional components** (also referred to as "presentational", "dumb" or "stateless" components - more about this later in the course) => const cmp = () => { return <div>some JSX</div> } (using ES6 arrow functions as shown here is recommended but optional)
2. **class-based components** (also referred to as "containers", "smart" or "stateful" components) => class Cmp extends Component { render () { return <div>some JSX</div> } }

We'll of course dive into the difference throughout this course, you can already note that you should use 1) as often as possible though. It's the best-practice.

Children refers to any element between the opening and closing tags of the component

‘porps’ are object giving access to all the attributes given in the component.

‘state’ is a property defined in a class which extends Component. ‘state’ is managed from inside a component. ‘state’ is a js object.

Props & State

props  and state  are **CORE concepts** of React. Actually, only changes in props  and/ or state  trigger React to re-render your components and potentially update the DOM in the browser (a detailed look at how React checks whether to really touch the real DOM is provided in section 6).

**Props**

props  allow you to pass data from a parent (wrapping) component to a child (embedded) component.

**Example:**

**AllPosts Component:**

1. const posts = () => {
2. return (
3. <div>
4. <Post title="My first Post" />
5. </div>
6. );
7. }

Here, title  is the custom property (prop ) set up on the custom Post  component. We basically replicate the default HTML attribute behavior we already know (e.g. <input type="text">  informs the browser about how to handle that input).

**Post Component:**

1. const post = (props) => {
2. return (
3. <div>
4. <h1>{props.title}</h1>
5. </div>
6. );
7. }

The Post  component receives the props  argument. You can of course name this argument whatever you want - it's your function definition, React doesn't care! But React will pass one argument to your component function => An object, which contains all properties you set up on <Post ... /> .

{props.title}  then dynamically outputs the title  property of the props  object - which is available since we set the title  property inside AllPosts  component (see above).

**State**

Whilst props allow you to pass data down the component tree (and hence trigger an UI update), state is used to change the component, well, state from within. Changes to state also trigger an UI update.

**Example:**

**NewPost Component:**

1. class NewPost extends Component { // state can only be accessed in class-based components!
2. state = {
3. counter: 1
4. };
6. render () { // Needs to be implemented in class-based components! Needs to return some JSX!
7. return (
8. <div>{this.state.counter}</div>
9. );
10. }
11. }

Here, the NewPost  component contains state . Only class-based components can define and use state . You can of course pass the state  down to functional components, but these then can't directly edit it.

state  simply is a property of the component class, you have to call it state  though - the name is not optional. You can then access it via this.state  in your class JSX code (which you return in the required render()  method).

Whenever state  changes (taught over the next lectures), the component will re-render and reflect the new state. The difference to props  is, that this happens within one and the same component - you don't receive new data (props ) from outside!

To Which Events Can You Listen?

In the last lecture, we saw that you can react to the onClick event - but to which other events can you listen? You can find a list of supported events here: <https://reactjs.org/docs/events.html#supported-events>

#### **Clipboard Events**

Event names:

1. onCopy onCut onPaste

Properties:

1. DOMDataTransfer clipboardData

Composition Events

Event names:

1. onCompositionEnd onCompositionStart onCompositionUpdate

Properties:

1. string data

Keyboard Events

Event names:

1. onKeyDown onKeyPress onKeyUp

Properties:

1. boolean altKey
2. number charCode
3. boolean ctrlKey
4. boolean getModifierState(key)
5. string key
6. number keyCode
7. string locale
8. number location
9. boolean metaKey
10. boolean repeat
11. boolean shiftKey
12. number which

Focus Events

Event names:

1. onFocus onBlur

These focus events work on all elements in the React DOM, not just form elements.

Properties:

1. DOMEventTarget relatedTarget

Form Events

Event names:

1. onChange onInput onInvalid onSubmit

For more information about the onChange event, see [Forms](https://reactjs.org/docs/forms.html).

Mouse Events

Event names:

1. onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit
2. onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave
3. onMouseMove onMouseOut onMouseOver onMouseUp

The onMouseEnter and onMouseLeave events propagate from the element being left to the one being entered instead of ordinary bubbling and do not have a capture phase.

Properties:

1. boolean altKey
2. number button
3. number buttons
4. number clientX
5. number clientY
6. boolean ctrlKey
7. boolean getModifierState(key)
8. boolean metaKey
9. number pageX
10. number pageY
11. DOMEventTarget relatedTarget
12. number screenX
13. number screenY
14. boolean shiftKey

Selection Events

Event names:

1. onSelect

Touch Events

Event names:

1. onTouchCancel onTouchEnd onTouchMove onTouchStart

Properties:

1. boolean altKey
2. DOMTouchList changedTouches
3. boolean ctrlKey
4. boolean getModifierState(key)
5. boolean metaKey
6. boolean shiftKey
7. DOMTouchList targetTouches
8. DOMTouchList touches

UI Events

Event names:

1. onScroll

Properties:

1. number detail
2. DOMAbstractView view

Wheel Events

Event names:

1. onWheel

Properties:

1. number deltaMode
2. number deltaX
3. number deltaY
4. number deltaZ

Media Events

Event names:

1. onAbort onCanPlay onCanPlayThrough onDurationChange onEmptied onEncrypted
2. onEnded onError onLoadedData onLoadedMetadata onLoadStart onPause onPlay
3. onPlaying onProgress onRateChange onSeeked onSeeking onStalled onSuspend
4. onTimeUpdate onVolumeChange onWaiting

Image Events

Event names:

1. onLoad onError

Animation Events

Event names:

1. onAnimationStart onAnimationEnd onAnimationIteration

Properties:

1. string animationName
2. string pseudoElement
3. float elapsedTime

Transition Events

Event names:

1. onTransitionEnd

Properties:

1. string propertyName
2. string pseudoElement
3. float elapsedTime

Other Events

Event names:

1. onToggle

----

This.getState({}) – applicable only in class –state

From React 16.8, react ‘hooks’ can be used to manage ‘state’ from functions

React hooks are a collection of functions which can be used inside functional components.

‘useState’ is the most important React hook.

‘stateful’ components are called smart or container components. Others are stateless presentational or dumb components.

2 ways of styling: .css file, inline styling.

Ek.js

import React from 'react'

const ek = () => {

return <p> Hello EK</p>

}

export default ek

--

App.js

Import Ek from ‘./Person/Ek’

----

Person.js

//props

import React from 'react'

const person = (props) =>{

return <p> Hello I am {props.name} {props.age} years old {props.children} </p>

}

export default person

---

App.js

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name= "Jaison" age = "45"/>

<Person name="Sonia" age= "39"> Facebook </Person>

<Ek />

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

//state App.js

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{name:'Max',age:28},

{name:'Manu', age:29},

{name:'Stephanie',age:27}

]

}

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name= {this.state.persons[0].name} age = {this.state.persons[0].age}/>

<Person name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

App.js

//eventHandler

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{name:'Max',age:28},

{name:'Manu', age:29},

{name:'Stephanie',age:27}

]

}

clickHandler = () => console.log("I am clicked")

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name= {this.state.persons[0].name} age = {this.state.persons[0].age}/>

<Person name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={this.clickHandler}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

App.js //setState

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{name:'Max',age:28},

{name:'Manu', age:29},

{name:'Stephanie',age:27}

]

}

clickHandler = () => {

this.setState({persons: [

{ name: 'Maxmilian', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 22 }

]})

}

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name= {this.state.persons[0].name} age = {this.state.persons[0].age}/>

<Person name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={this.clickHandler}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

‘useState’ always returns 2 elements, exactly 2 elements!. The first element which returns always is the ‘current state’. 2nd return element is the function that updates the current state.

App.js

//useState

import React, { useState } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

const app = () => {

const [currentState, updateState] = useState({

persons: [

{name:'Max',age:28},

{name:'Manu', age:29},

{name:'Stephanie',age:27}

]})

const clickHandler = () => {

updateState({persons: [

{ name: 'Maxmilian', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie Max', age: 22 }

]})

}

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name={currentState.persons[0].name} age={currentState.persons[0].age}/>

<Person name={currentState.persons[1].name} age={currentState.persons[1].age}> Facebook </Person>

<Person name={currentState.persons[2].name} age={currentState.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={clickHandler}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

export default app;

---

eventHandler.bind(this,’Name’) – to pass arguments.

---

App.js

// passing method reference to other objects

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{ name: 'Max', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 27 }

]

}

clickHandler = () => {

this.setState({

persons: [

{ name: 'Maxmilian', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 22 }

]

})

}

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name={this.state.persons[0].name} age={this.state.persons[0].age} />

<Person name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person click={this.clickHandler} name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={this.clickHandler}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

Person.js

//props

import React from 'react'

const person = props =>{

return <p onClick={props.click}> Hello I am {props.name} {props.age} years old {props.children} </p>

}

export default person

---

App.js

// passing as argument using bind and method call

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{ name: 'Max', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 27 }

]

}

clickHandler = (newName) => {

this.setState({

persons: [

{ name: newName, age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 22 }

]

})

}

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name={this.state.persons[0].name} age={this.state.persons[0].age} />

<Person name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person click= {() => this.clickHandler('Sonia')} name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={this.clickHandler.bind(this,'Jaison')}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

App.js

// onChange with an event object passed

import React, { Component } from 'react';

import './App.css';

import Person from './Person/Person';

import Ek from './Person/Ek'

class App extends Component {

state = {

persons: [

{ name: 'Max', age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 27 }

]

}

clickHandler = (newName) => {

this.setState({

persons: [

{ name: newName, age: 28 },

{ name: 'Manu', age: 29 },

{ name: 'Stephanie', age: 22 }

]

})

}

changeName = (event) => {

this.setState({

persons: [

{ name: 'Maximilan', age: 28 },

{ name: event.target.value, age: 29 },

{ name: 'Stephanie', age: 22 }

]

})

}

render() {

return (

<div className="App">

<h1> Hi, I am a React App!!</h1>

<Person name={this.state.persons[0].name} age={this.state.persons[0].age} />

<Person changed={this.changeName} name={this.state.persons[1].name} age={this.state.persons[1].age}> Facebook </Person>

<Person click= {() => this.clickHandler('Sonia')} name={this.state.persons[2].name} age={this.state.persons[2].age}> Whatsapp </Person>

<Ek />

<button onClick={this.clickHandler.bind(this,'Jaison')}>Click</button>

</div>

// React.createElement('div',{className:'App'},React.createElement('h1',null,"From React"))

);

}

}

export default App;

---

Person.js

import React from 'react'

const person = props =>{

return (

<div>

<p onClick={props.click}> Hello I am {props.name} {props.age} years old {props.children} </p>

<input type="text" onChange={props.changed} value={props.name} />

</div>

)

}

export default person

---

‘inline’ style is defined inside the ‘render()’ method in the class, which can be used in the element declared under ‘return’

Single curly braces {} used to input dynamic content.

Useful Resources & Links

* create-react-app: <https://github.com/facebookincubator/create-react-app>
* Introducing JSX: <https://reactjs.org/docs/introducing-jsx.html>
* Rendering Elements: <https://reactjs.org/docs/rendering-elements.html>
* Components & Props: <https://reactjs.org/docs/components-and-props.html>
* Listenable Events: <https://reactjs.org/docs/events.html>

Javascript expressions can be used inside jsx using single curly braces { expressions }.

Jsx code start with the return statement in the ‘render()’ method in the component. Inside jsx, if condition cannot be used. Instead, ternary operator can be used.

App.js

// Using map function to return array elements in jsx

render() {

let p = null

if(this.state.personOnoff) {

p = (

<div>

{

this.state.persons.map(person => {

return (

<Person name={person.name}

age={person.age} />

)

})

---

App.js

//removing an element from array by click on the text

removeHandler = (personIndex) => {

const list = this.state.persons

list.splice(personIndex,1)

this.setState({persons:list})

}

render() {

let p = null

if(this.state.personOnoff) {

p = (

<div>

{

this.state.persons.map((person,index) => {

return (

<Person click={() => this.removeHandler(index)} name={person.name}

age={person.age} />

)

})

---

newArray = Array.slice() – without arguments simply copies the array and returns a new array.

newArray = […oldArray] – also can be used to copy an array (using spread operator)

using a ‘key’ as a property inside the component is sometimes advisable

Useful Resources & Links

* Conditional Rendering: <https://reactjs.org/docs/conditional-rendering.html>
* Lists & Keys: <https://reactjs.org/docs/lists-and-keys.html>

Jiji