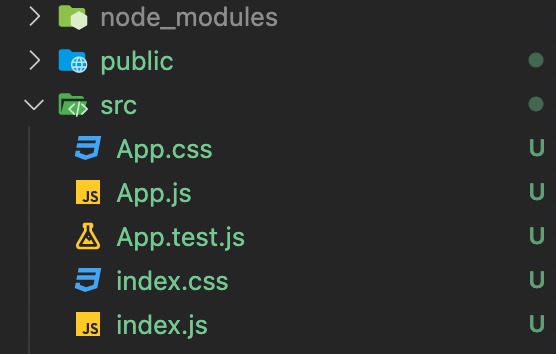
React Cheat sheet

(path/ file, element, code)

* Start up:

1. come into folder in terminal
2. In terminal run: create-react-app folder\_name
3. Get in to folder\_name folder
4. Run: npm run start

* Path:



1. we create react components in App.js and export it
2. In index.js we import App.js react component and use ReactDom to render it

3. In public/index.html we have a div with its id to place the React component

* JSX (use HTML tag/code in .js file)

<Comp /> or <Comp></Comp>

* Components in React( = javascript function return (JSX code) => !return element must in a main element like a <div> )

React components must start with an uppercase name

?can I use arrow function to define a component ? Yes it works! let comp = () => {return()}

Index.js > app.js > components.js

!!! One page = index.js can have multiple app => each app can have multiple components !!!

Create components and export it export default comp=> In App.js import needed components, use JSX tag <Comp />to render then export app => In index.js import needed apps, use ReactDom to render to the right place.

* Style change

Import Css file (import ‘./x.css’)=> use the keyword className instead of class in JSX

<div className = ‘head’>

* Dynamic data

Use {} to hold dynamic data

!!! Props (transfer data between different (normally parent and child components) components

Parent: def data => transfer in JSX element through child components

Child: use props to get data then use it in {props.xxx} xxx from Parent.js

App.js => const obj = {name:’Sam’, age:12} => <Child name={obj.name}></Child>

Child.js=> function child(props) => <div>{props.name}</div> = <div>Sam</div>

* Handle events

Add event listener in JSX <button onClick={function\_name}>

!!! if add () after function\_name, the function will execute when the line render so don’t add ()!!!

Step:

1. add event listener =>
2. def executed function =>
3. in components use useState hook: const [value, setFun] =useState(props.target) value=props.target, setFun = function =>
4. in defined function use the setFun to update variable value setFun(‘updated value’)
5. JSX re-render and update the value

|  |
| --- |
| 1. 2<input onChange={updateTitle}>   3.Const [title, setTitle] = useState(‘’)  4. Const updateTitle = (e)=>{  // update title value by setTitle function  setTitle(e.target.value)  } |

useState(‘’) => return an array [1. value, 2. function used to update value]

When fun called in JSX => go to defined function=> use fun created by useState to update the value call fun(‘new value’) => REACT render the JSX again

|  |
| --- |
| Event listener List  OnClick(fun) => click event  OnChange(fun) => input change = fun(e) => e.target.value => get current updated value  OnSubmit(fun) => when button in form and its type = submit  Event.function  E.preventDefault() => tell the page not reload |

* Two-way bind

One way => when event occur => execute function => change data

Two way => when data change re-render JSX set value = useState()[0]

! Use useState()[1], which is a setFun to reset date like setTitle(‘’) => make title = ‘’

* Pass Data from child to parent

Parent to child => use props

Child to parent => complex

Define in Parent

1. in Child JSX add a props <Child props\_name={function}>
2. Def the function in Parent

2.1 const function = (data we needed in child) => {const x = {...data we needed in child}}

Const function = (data) => {const data = {...data}} //here we get data from child

1. go back to child, props.props\_name(data in child). tranfer data in child to parent and store in x

When event occur => execute function

Function => generate data => pass to Parent component

|  |  |
| --- | --- |
| Child | Parent |
| Obj is the target date need to be passed  Const submit = (e) =>{  Const obj = {.....}  Props.transFromChild(obj)  } | <Child transFromChild={getObj}/>  Const getObj = (target) =>{  Const objFromChild = {...target}  //  Now objFromChild is the data get from Child !!!  } |