React

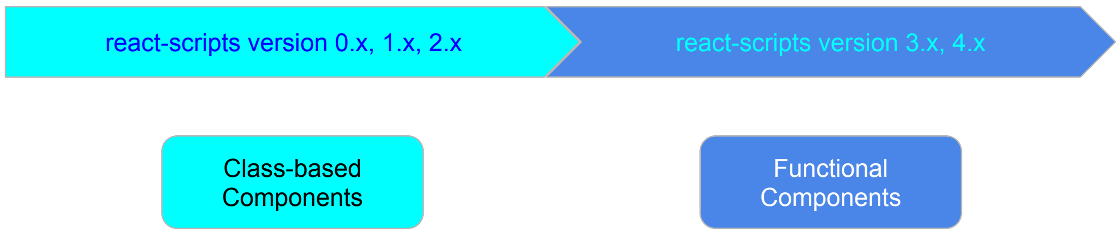
Functional Components in
TypeScript

State of JS surveys:
2020 2021 2022

React Release History

- Maintained by Facebook
- 2013 -- 2016 : versions 0.3.0 - 0.14.7
- *Major version jumped*
- Versions 15.0.0 (Apr 2016) - 15.6.1 (Aug 2017)
- Versions 16.0.1 (Aug 2018) - 16.14.x (Oct 2020)
- **Version 17.0.0 (Oct 2020) - 17.0.2 (Mar 2021) [My GitHub Sample]**
- Version 18.2 (2022)

React Components


react-scripts version 0.x, 1.x, 2.x
react-scripts version 3.x, 4.x
Class-based Components
Functional Components

```
class MyComponent extends React.Component {

  render(): JSX.Element {
     return <!-- HTML contents --->;
  }
}

export default MyComponent
```

```
function MyComponent (): JSX.Element {

  return <!-- HTML contents --->;
}

export default MyComponent
```

Installation & Project Setup (Option #1)

```
# For npm 6.x
npm create vite@latest my-first-react --template react-ts

# npm 7.x or later
npm create vite@latest my-first-react -- --template react-ts
```

```
# Add vite to your project
npm i -save  vite
# yarn add vite
```

```
# Launch the local server
cd my-first-react
npm install
npm run dev
# yarn dev
```

Installation & Project Setup (Option #2)

```
# One time installation
yarn global add create-react-app
create-react-app --version          # 4.0.3 or newer
```

```
# Setup a project with functional components
create-react-app my-first-react --template typescript
```

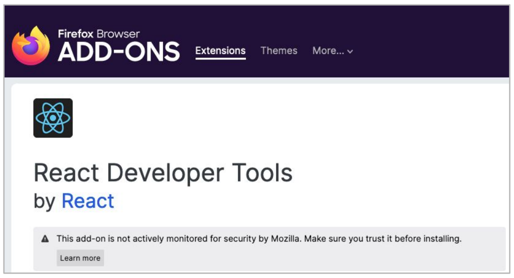
```
# Launch the local server
cd my-first-react
yarn start
```

Project Structure

```
TOP-LEVEL-DIR-OF-PROJECT
├── public
│   └── index.html
│
├── src
│   ├── App.css
│   ├── App.tsx
│   ├── index.css
│   ├── index.tsx
│   └── logo.svg
│
├── package.json
├── tsconfig.json
├── tsconfig.node.json
└── vite.config.ts
```

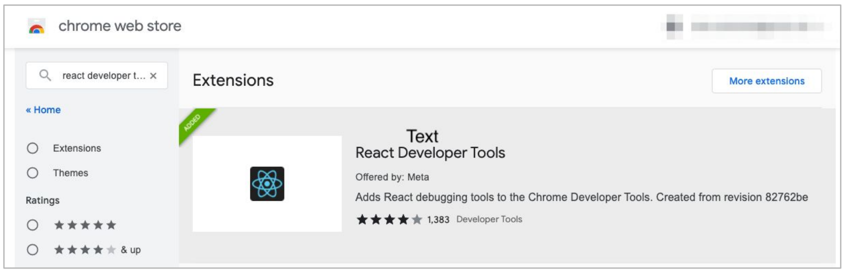
You will spend 99% of your time here

React DevTools (FireFox & Chrome Extensions)



React Developer Tools
by React

This add-on is not actively monitored for security by Mozilla. Make sure you trust it before installing.



Text
React Developer Tools
Offered by: Meta
Adds React debugging tools to the Chrome Developer Tools. Created from revision 82762be
★★★★★ 1,383 Developer Tools

Use VueJS Concepts for learning React

Vue vs. React

- A React Point of Vue by Divya Sashidaran (Oct 2018)
  - VueJS sample code
  - React sample code
- React for Vue Developers (May 2019)
- VueJS Developors Guide to React (Oct 2020)

Prerequisites

- Understanding TSX Syntax
- Array.map() function & lambda expressions

(JS|TS)X = (Java|Type)Script + XML

TSX = TypeXMLScript Quick Tour

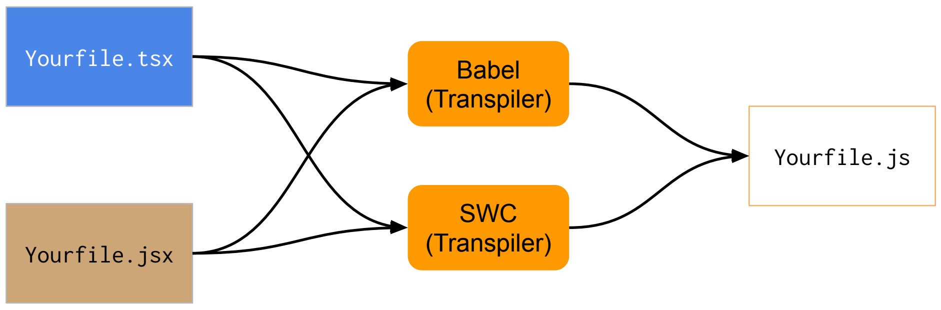
```
const msg = `Your total is ${ euroExchangeRate * amountUSD }EURO`
console.log(typeof msg)        // Output: string
```

```
const msgx = <p>Your total is { euroExchangeRate * amountUSD } EURO</p>
console.log(typeof msgx)       // Output: object
```

```
if (payInEURO)
  return <p>Your total is { euroExchangeRate * amountUSD } EURO</p>
else
  return <p>Your total is { amountUSD } USD</p>
```

BabelJS Playground SWC Playground (Written in Rust)

TSX|JSX Transpiler



SWC: Speedy Web Compiler

Using TSX you can:

- Use TS expressions inside XML
- Use XML expressions inside TS

TS in HTML in TS (in ...)

TSX General rules:

1. *Place TS expressions within { and }*
2. *Place HTML expressions within <someTag> and </someTag>*

HTML in TS and TS in HTML

```
const mailCount = 5;
const minYear = 1900; maxYear = 2025;
```

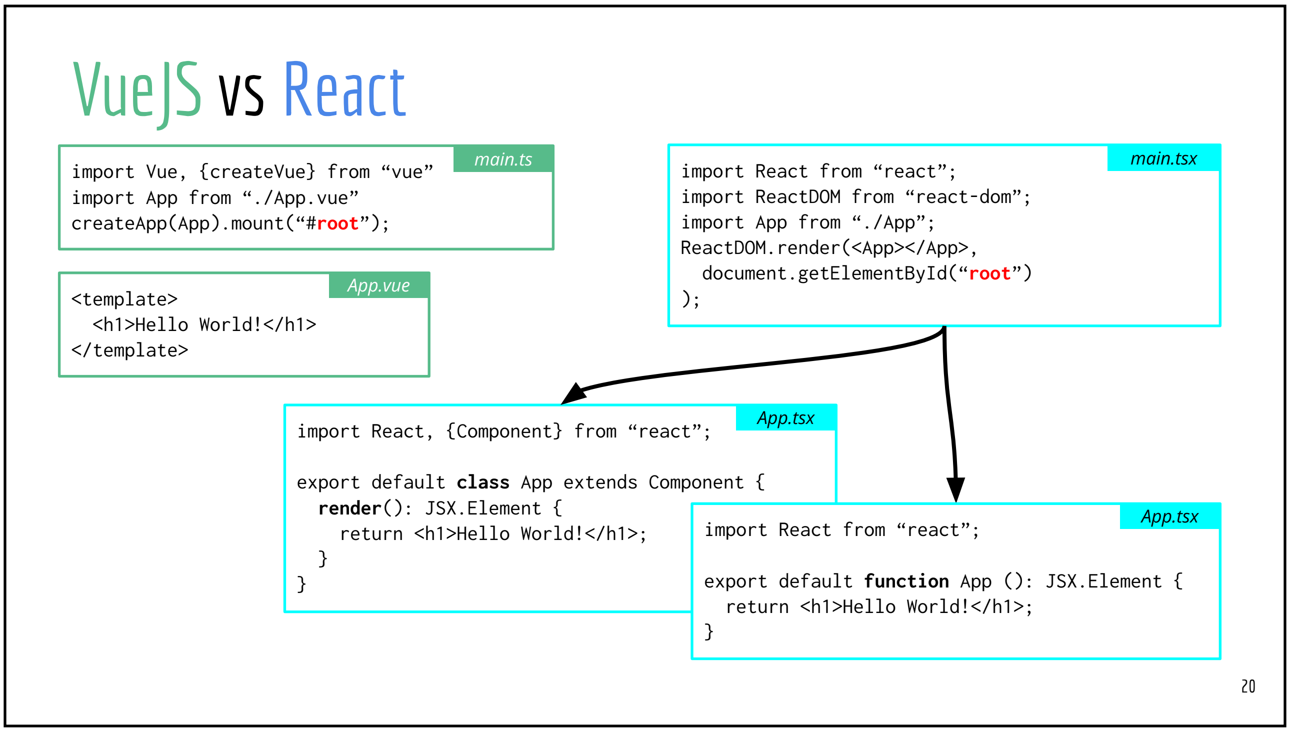
$$\texttt{const yearInput = }\underbrace{\texttt{<input type="slider" min=}\underbrace{\texttt{\{minYear\}}}_{TS}\texttt{ max=}\underbrace{\texttt{\{maxYear\}}}_{TS}\texttt{ />}}_{HTML}\texttt{;}$$

$$\texttt{const msg = }\underbrace{\texttt{mailCount > 20 ? }\underbrace{\texttt{<p>You have to many emails</p>}}_{HTML}\texttt{ : }\underbrace{\texttt{<p>You have }\underbrace{\texttt{\{mailCount\}}}_{TS}\texttt{ emails</p>}}_{HTML}}_{TS}\texttt{;}$$

```
const msgSpan = <span>
   { mailCount > 20 ? <p>You have too many emails</p> : <p>You have {mailCount} emails</p> }
  </span>;
```

Using Array.map() with lambda expressions [A different slide]

React Part A: Basic Concepts



VueJS vs React

main.ts
```
import Vue, {createVue} from “vue”
import App from “./App.vue”
createApp(App).mount(“#root”);
```

App.vue
```
<template>
  <h1>Hello World!</h1>
</template>
```

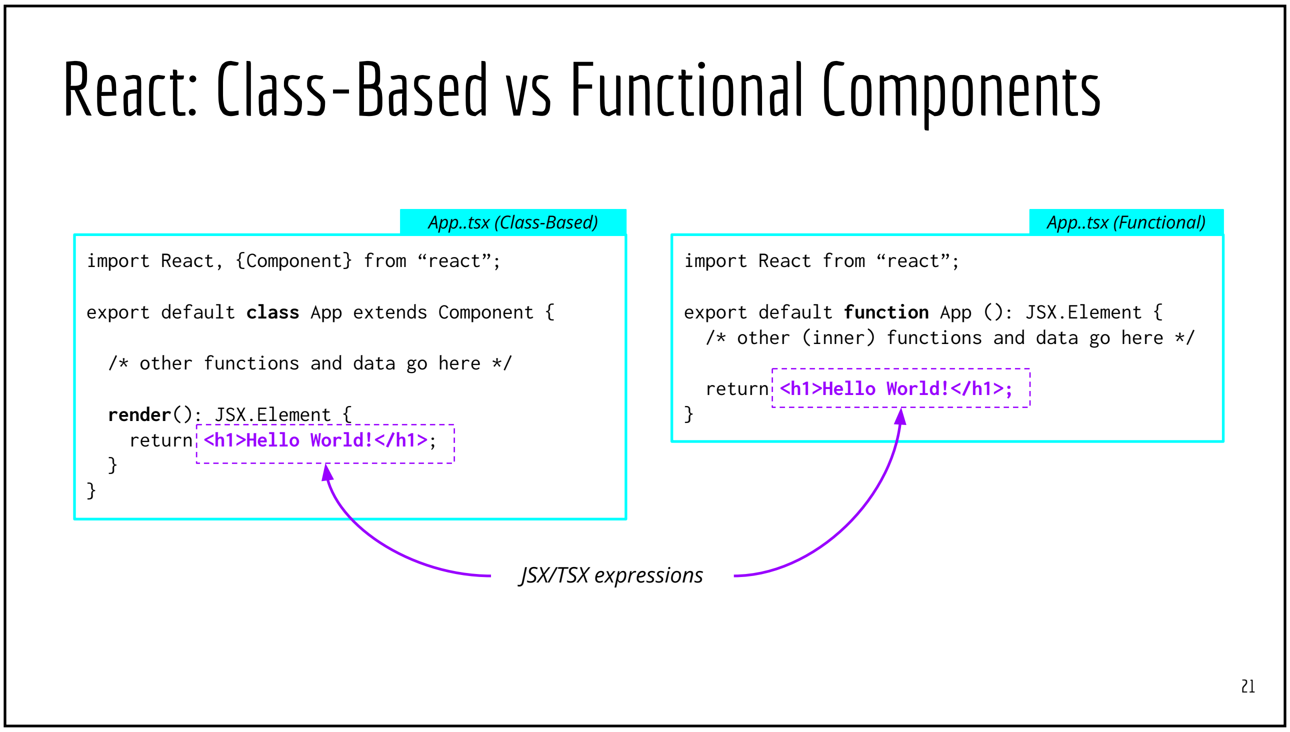
main.tsx
```
import React from “react”;
import ReactDOM from “react-dom”;
import App from “./App”;
ReactDOM.render(<App></App>,
  document.getElementById(“root”)
);
```

App.tsx
```
import React, {Component} from “react”;

export default class App extends Component {
  render(): JSX.Element {
    return <h1>Hello World!</h1>;
  }
}
```

App.tsx
```
import React from “react”;

export default function App (): JSX.Element {
  return <h1>Hello World!</h1>;
}
```



React: Class-Based vs Functional Components

App..tsx (Class-Based)
```
import React, {Component} from “react”;

export default class App extends Component {

  /* other functions and data go here */

  render(): JSX.Element {
    return <h1>Hello World!</h1>;
  }
}
```

App..tsx (Functional)
```
import React from “react”;

export default function App (): JSX.Element {
  /* other (inner) functions and data go here */

  return <h1>Hello World!</h1>;
}
```

JSX/TSX expressions

Live Demo: 01-minimal.tsx

Online Playgrounds

https://playcode.io

https://stackblitz.com (longer code)

VueJS vs. React

VueJS	React

`const myVar = ref(____)`
`{{ myVar }}`
	`const [myVar] = useState(____);`
	`{ myVar }`

`v-bind:width=”myVar”`	`width={myVar}`
`v-if, v-else`	*Use Typescript if-statement*
`v-show`	*N/A*
`v-for`	*Use TypeScript Array.map*
`v-on:click=”myFunction”`	`onClick={myFunction}`
`v-model`	*Use change listener & state setter*

React State ⇔ Variables bound to UI

Functional React Hook #1: useState()

Purpose: to define variables that will be ***bound to the UI***

```
// Typical usage
const [stateName, setterFunc] = useState(initial_value_of_the_state_var);
```

- **`useState()`** `defines a new state variable in a functional component`
- `stateName` *is a read-only variable for getting the current value of the state variable*
- *Updates to the state variable (in a functional component)* ***must be*** *carried out by invoking the* `setterFunc` *function*
- Similar to VueJS 3.x ref()

VueJS ref(), Ref vs. React useState()

VueJS	React
`const city= ref("Allendale")`	`const [city] = useState("Allendale")`
`const cities = ref(["Ada", "Boston", "Chicago"])`	`const [cities] = useState(["Ada", "Boston", "Chicago"])`
`const primes: Ref<number[]> = ref([])`	`vondy [primes] = useState<number[]>([])`

Variable Binding to UI

App.vue

```
<script setup lang="ts">
import {ref} from "vue"


const rand = Math.round(Math.random() * 57)
const who = ref("World")
const visitorNum = ref(1037 + rand)
</script>


<template>
  <div>
    <h1>Hello {{who}}!</h1>
    <p>You're visitor #{{visitorNum}}</p>
  </div>
<template>
```

Functional App.tsx

```
import React, { useState } from "react";

export default function App(): JSX.Element {
  const rand = Math.round(Math.random() * 57);
  const [who] = useState("World");
  const [visitorNum] = useState(1037 + rand);

  return <div>
      <h1>Hello {who}!</h1>
      <p>You're visitor #{visitorNum}</p>
    </div>;
}
```

Online Playground: 10-state-binding

Variable Binding to HTML attributes

VueJS

```
<script setup lang="ts">
const imgLoc = ref("http://imgur.com/abc.png")
const imgKlaz = ref("thumbnail")
</script>

<template>
   <img :src="imgLoc" :class="imgKlaz">
</template>
```

React

```
export default function Sample() {
  const [imgLoc] = useState("http://____");
  const  [imgKlaz] = useState("thumbnail");


  return <>
    <img src={imgLoc} className={imgKlaz}>
  </>
}
```

Online Playground: 12-attr-binding

VueJS v-for in array ⇔ React Array.map()

```
<p>Chemical Elements</p>
<ul>
  <li>Argon</li>
  <li>Barium</li>
  <li>Carbon</li>
  <li>Fluor</li>
</ul>
```

Chemical Elements

- Argon
- Barium
- Carbon
- Fluor

VueJS

```
<script setup lang="ts">
const elements = ref(["Argon", "Barium", "Carbon", ...])
</script>
<template>
  <p>Chemical Elements</p>
  <ul>
    <li v-for="(e,pos) in elements" :key="pos">{{e}}</li>
  </ul>
</template>
```

React

```
export default function Sample(): JSX:Element {
  const elements = ["Argon", "Barium", "Carbon", ...]
  return <>
    <p>Chemical Elements</p>
    <ul>
      {
        elements.map((e:string, pos:number) =>
          <li key={pos}>{e}</li>)
      }
    </ul>
  </>;
```

VueJS v-for in array ⇔ React Array.map()

```
<p>Chemical Elements</p>
<ul>
  <li>Argon (weight 39.948)</li>
  <li>Barium(weight 127.33)</li>
  <li>Carbon (weight 12.011)</li>
</ul>
```

VueJS

```
<script setup lang="ts">
const  elements = ref([{name: "Ar_", weight: 39.948},
              {name: "Ba_", weight: 127.33}, /* more data */]);
}
</script>
<template>
  <div>
    <p>Chemical Elements</p>
    <ul>
      <li v-for="(e,pos) in elements" :key="pos">
        {{e.name}} (weight {{e.weight}})</li>
    </ul>
  </div>
</template>
```

React

```
export default function Sample(): JSX:Element {
  const elements = useState([
   {name: "Ar_", weight: 39.948},
   {name: "Ba_", weight: 127.33}, /* more */]);

  return <>
    <p>Chemical Elements</p>
    <ul>
      {
         elements.map((e:any, pos:number) =>
           <li key={pos}>
             {e.name} (weight {e.weight})
           </li>
         )
      }
    </ul>
  </>;
```

Live Demo: 20-forloop.tsx

useState() with proper typing

```
// POOR example
const [myGetter, mySetter] = useState([])      // myGetter is an array of unknown!!!
                                          // mySetter is a function that takes any array
```

```
// Better!!!
type AtomType = {
  name: string,
  symbol: string,
  weight: number
}

const [atoms, setAtoms] = useState<Array<AtomType>>([])   // atoms is an array of AtomType

setAtoms([20, 15, 100])   // ERROR: incompatible AtomType vs. number
```

Conditional Rendering (v-if, v-else)

VueJS

```
<template>
  <p v-if="currentHour < 12">Good morning</p>
  <p v-else>Welcome</p>
</template>
```

React

```
function Sample() {
  return  <>
    { currentHour < 12 ?
        <p>Good morning</p> :
        <p>Welcome</p>
    }
  </>
```

condition ? trueExpression : falseExpression

Online Playground: 30-conditional

Event Handling

VueJS

```
<script setup lang="ts">
function doOne(ev: MouseEvent): void {
  console.log("Inside doOne");
}
const  doTwo = (ev: MouseEvent): void => {
    console.log("Inside doOne");
}
</script>

<template>
  <div>
    <button v-on:click="doOne">One</button>
    <img @:click="doTwo">Two</img>
  </div>
</template>
```

React

```
import React, {Component, MouseEvent} from 'react'

export default function Sample(): JSX.Element {
  // Named function
  function doOne(ev: MouseEvent<HTMLButtonElement>): void {
    console.log("Inside doOne", this.location);
  }
  // FAT Arrow function
  const doTwo = (ev: MouseEvent<HTMLImageElement>): void => {
    console.log("Inside doTwo", this.location);
  }
  return <>
    <button onClick={ doOne }>One</h2> <!-- FAT call not required -->
    <img onClick={ doTwo }>Two</img>
  </>;
}
```

In functional components, event handling functions can be declared as either a named and a fat arrow function. They both exhibit the same behavior

Generic Event Types

TSX declaration	Event Handling Function Signature
`<input onChange={foo} />`	`function foo (ev: ChangeEvent<HTMLInputElement>): void { /* code */ }`
`<button onClick={foo}>xxx</button>`	`function foo (ev: MouseEvent<HTMLButtonElement>): void { /* code */ }`
`<div onMouseEnter={foo}></div>`	`function foo (ev: MouseEvent<HTMLDivElement>): void { /* code */ }`

Event Class	Event types
MouseEvent	mouseenter,mouseleave, click, mousemove
KeyboardEvent	keyup, keydown, keypress
Go to Mozilla Dev Network online docs for Event details	

Live Demo: 40-event-handling.tsx

Mimicking v-model

```
<script setup lang=”ts”>
const  name = ref(“Anonymous”)
</script>




<template>
   <h3>You enter: {name}</h3>
   <input type="text" v-model=”name” />
</template>
```

```
export default function Sample (): JSX.Element {
  const [name, setName] = useState(“Anonymous”);

  updateName(ev: ChangeEvent<HTMLInputElement>): void {
    setName(ev.target.value);
  }

  render() {
    return <>
        <h3>You enter: {name}</h3>
        <input type="text"
          onChange={ updateName } />
      </>;
  }
}
```

Online Playground: 43-vmodel

Live Demo: 45-state-update.tsx

VueJS	React (Functional Component)

`<span>{{varName}}</span>`	`const [varName] = useState(___)`
	`return <span>{varName}</span>;`

`<img ref="abc"/>`	`const abc = useRef();`
	`return <img ref={abc} />`

`<li v-for="(x,pos) in arr" :key="pos"> {{x}} </li>`	`{ arr.map((x,pos:number) => <li key={pos}> {x} </li>}`

`<img v-bind:src={{ imageUrl }} />`	`const [imageUrl] = useState(___)`
	`return <img src={ imageUrl } />`

`<p v-if="len > 0">{{len}} items</p>`
`<p v-else>Select items below</p>`
	`const len = useState(51);`
	`return len > 0 ?`
	`  <p>{len} items</p> : <p>Select items below</p>;`

`// Just method name if no args needed`
`<button v-on:click="doIt"></button>`
`<button @click="doIt($event)"></button>`
	`<button onClick={doIt}></button>`

`<input type="text" v-model="uName">`	**Not Supported Natively**

Passing Data to (Child) Components

Passing Data to (Child) Components

3:37am

VueJS
```
<RoundClock :hour="3"
            :minute="37"
            label="Chicago">
```

React
```
<RoundClock hour={3}
            minute={37}
            label={"Chicago"}>
```

VueJS
```
<RoundClock :hour="3"
            :minute="37"
            label="Chicago" pm>
```

React
```
<RoundClock hour={3}
            minute={37}
            label={"Chicago"} pm>
```

3:37pm

- ***Hour***, ***minute***, *and* ***label*** *are mandatory props*
- ***pm*** *is an optional prop*

Receiving Props

VueJS

```
<script setup lang="ts">
import {defineProps} from "vue"
type MyProp = {
  hour: number,
  minute: number,
  label?: string,
  pm?: boolean
}
const props = defineProps<MyProp>()
</script>

<template>
  <p>Time is {{props.hour}}:{{props.minute}}</p>
</template>
```

React

```
type MyProp = {
  hour: number,
  minute: number,
  label?: string, // Use ? for optional prop
  pm?: boolean
}

function RoundClock(props:MyProp): JSX.Element {
  return <>
    <p>Time is {props.hour}:{props.minute}
       {props.pm ? "PM":"AM"}
    </p>
  </>;
}
```

Object Destructuring

```
const marsInfo = {
  name: "Mars",
  speed: 14.5, /* miles per second */
  diameter: 4_220, /* miles */
  tiltAxis: 25, /* degrees *.
  lengthOfYear: 1.88, /* times of Earth year */
}
```

```
const {speed, name} = marsInfo
console.log(speed)        // 14.5
console.log(name)         // Marse
```

```
const {name, tiltAngle} = marsInfo
console.log(name)          // Mars
console.log(tiltAngle)     // undefined
```

```
const {name = "Earth", gravity = 9.8} = marsInfo
console.log(name)          // Mars    (override "Earth")
console.log(gravity)       // 9.8     (use default value)
```

Props Default Value

React

```
type MyProp = {
  hour: number,
  minute: number,
  label?: string, // Use ? for optional prop
  pm?: boolean
}

function RoundClock(props:MyProp): JSX.Element {
  return <>
    <p>{props.label} time is
         {props.hour}:{props.minute}
       {props.pm ? “PM”:”AM”}
    </p>
  </>;
}
```

React

```
type MyProp = {
  hour: number,
  minute: number,
  label?: string, // Use ? for optional prop
  pm?: boolean
}

function RoundClock(props:MyProp): JSX.Element {
  // Default values label=”Your location”, pm=false
  const {label = “Your location”, pm = false} = props
  return <>
    <p>{label} time is {props.hour}:{props.minute}
       {pm ? “PM”:”AM”}
    </p>
  </>;
}
```

State ⇒ UI Variables (ReadWrite)
Props ⇒ Input Parameters (ReadOnly)

Live Demo: 50-props-clock.tsx
LiveDemo: Timer with default value

“Lifecycle” Functions

Functional React Hook #2: useEffect()

- Perform "side effect" work AFTER render()
- Three variations of invocation

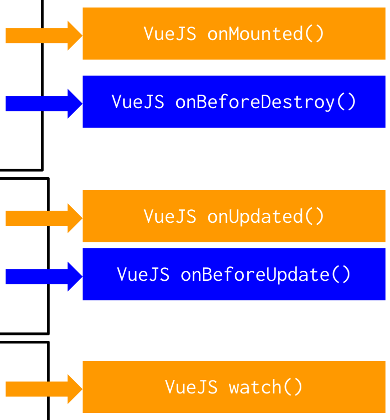
`useEffect(my_effect_func);`	*my_effect_func* runs after EVERY `render()`
`useEffect(my_effect_func, []);`	*my_effect_func* runs after the FIRST `render()`. Similar to VueJS `mounted()`
`useEffect(my_effect_func, [var,list,here]);`	*my_effect_func* runs after `render()` when ONLY WHEN **particular variables** change. Similar to VueJS @Watch function

useEffect() "do" & "undo" functions

```
useEffect(() => {
  // Code that runs AFTER the first render
  // (i.e component mounted)
 return () => {
    // Code that runs BEFORE the last render
    // (i.e. component is unmounted)
  }
}, [])
```

```
useEffect(() => {
  // Code that runs AFTER every render
  return () => {
    // Code that runs BEFORE the next render
  }
})
```

```
useEffect(() => {
  // Code that runs AFTER updates to
  // selected vars
  return () => {
    // Code that runs BEFORE the next render
    //
  }
}, [list,of,vars,here])
```



useEffect(*your-code*, [])

```
<!-- in VueJS -->
<script setup lang=”ts”>
  import {onMounted,
          onBeforeDestroy} from “vue”
  onMounted(() => {
   // Code runs AFTER FIRST render
  })
  onBeforeDestroy(() => {
   // Code runs BEFORE LAST render
  })
</script>
<template>
  <p>Hello</p>
</template>
```

```
// in ReactJS
export default function(): JSX.Element {
  function dying() {
    // Code runs BEFORE LAST render
  }
  useEffect(() => {
    // Code runs AFTER FIRST render
    return dying    // NO parentheses!!!
  }, []);  // ⇐ EMPTY ARRAY!!!

  return <p>Hello</p>
}
```

useEffect(*your-code*)

```
<!-- in VueJS -->
<script setup lang=”ts”>
  import {onMounted,
          onBeforeDestroy} from “vue”
  onUpdated(() => {
   // Code runs AFTER EVERY render
  })
  onBeforeUpdate(() => {
   // Code runs BEFORE NEXT render
  })
</script>
<template>
  <p>Hello</p>
</template>
```

```
// in ReactJS
export default function(): JSX.Element {
  function two() {
    // Code runs BEFORE NEXT render
  }
  useEffect(() => {
    // Code runs AFTER EVERY render
    return two    // NO parentheses!!!
  })  // ⇐ No Second Arg!!!

  return <p>Hello</p>
}
```

useEffect(*your-code*, [a,b,c])

```
<!-- in VueJS -->
<script setup lang="ts">
  import {watch} from "vue"

  watch([a,b,c], () => {
   // Code runs when EITHER a, b, or c
   // changes
  })

</script>
<template>
  <p>Hello</p>
</template>
```

```
// in ReactJS
export default function(): JSX.Element {
  useEffect(() => {
    // Code runs when EITHER a, b, or C
    // changes
  }, [a,b,c])  // ⇐ Array of args!!!

  return <p>Hello</p>
}
```

Do and undo examples

```
useEffect(() => {
 // start a timer
 myTimer = setInterval(
   () => { /* code here */ },
   1000);
 return () => {
    // stop the timer
    clearInterval(myTimer);
  }
}, [/* empty array */])
```

```
useEffect(() => {
 // code to setup Firestore onSnapshot listener

 return () => {
    // code to terminate the listener
  }
}, [/* empty array */)
```

```
useEffect(() => {
 // code to START audio player in background
 return () => {
    // Code STOP audio player
  }
}, [/* empty array */])
```

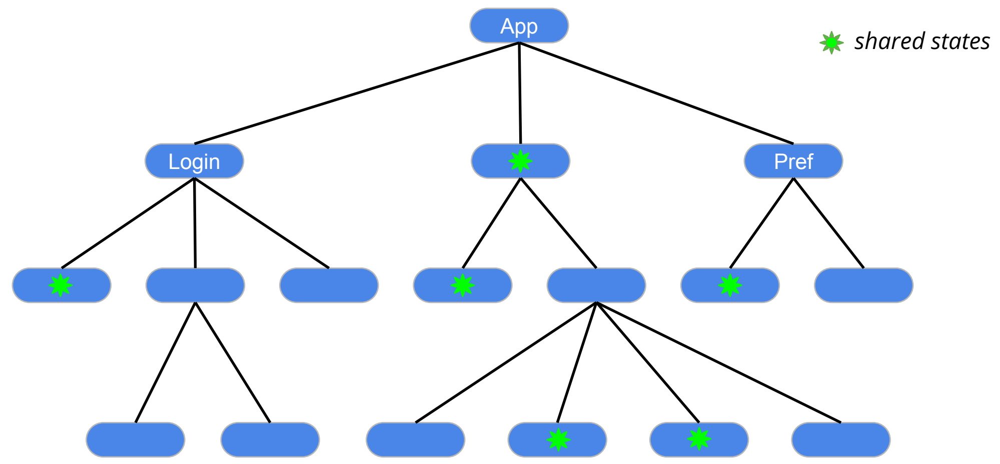
```
useEffect(() => {
 // code to save data into Browser local
 // storage

 return () => {
    // remove those data from Browser
    // local storage
  }
}, [/* empty array */])
```

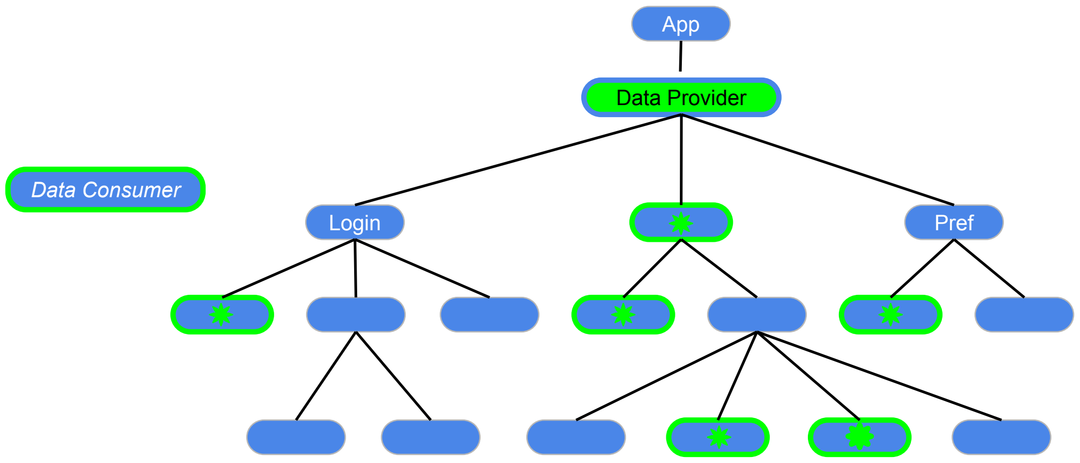
Live Demo: 60-effect.tsx
(Can't use the online playground)

Part B: Advanced Topics: State Sharing across Multiple Components

Hierarchy of Components



Data Sharing in React



React Hook #3: useContext()

Step A: Declare Data Type & Create Context

global-data.tsx

```
import {createContext, useState} from “react”

export type TGlobalData = {
  favTeam: string;
  changeTeam: (t:string) => void;
  /* more state and mutators here */
}

export const MyAppData = createContext<TGlobalData>(undefined!);
```

Step B: Wrap Top-Level & Provide Data

App.tsx

```
import {MyAppData} from "./global-data";
import TopLevel from "./______.tsx";

export default function App() {
  const [myTeam, setTeam] = useState("49ers");        // #1: Declare state(s)

  const initialData = {                                // #2: Initialize global data
    favTeam: myTeam,                                   //     from the state variables
    changeTeam: setTeam
  }

  return (
    <MyAppData.Provider value={initialData}>           // #3: Wrap the top-level components
      <TopLevel/>
    </MyAppData.Provider>);
}
```

Step C: Use Global Data in Components

CompOne.tsx

```
import {useContext} from "react"
import {MyAppData} from "./global-data"

export default function CompOne(): JSX.Element {
  const globData = useContext(MyAppData);

  return <>
      <p>Your team is {globData.favTeam}</p>
  </>;
}
```

CompTwo.tsx

```
import {useContext} from "react"
import {MyAppData} from "./global-data"

export default function CompTwo(): JSX.Element {
  const gd = useContext(MyAppData);

  function switchTeam(t: string) {
    gd.changeTeam(t);
  }

  return <>
      <button onClick={() => switchTeam("Lions")}</button>
  </>;
}
```

Live Demo: 70-context.tsx