学习

TS

https://typescript.bootcss.com/tutorials/react.html TS文档

```ts
// TS高级类型

// 1.Intersection Types(交叉类型)

type LeftType = {
  id: number;
  left: string;
};
type RightType = {
  id: number;
  right: string;
};
type IntersectionType = LeftType & RightType;

function showType(args: IntersectionType) {
  console.log(args);
}
showType({ id: 1, left: 'test', right: 'test' });


// 2.Union Types(联合类型)

type UnionType = string | number;

function showType(arg: UnionType) {
    console.log(arg);
}

showType('test');
showType(7);

// 3.Generic Types(泛型)

function showType<T>(args: T) {
  console.log(args);
}
```

```ts

showType('test');
showType(1);

// 泛型接口

interface GenericType<T> {
  id: number;
  name: T;
}

function showType(args: GenericType<string>) {
  console.log(args);
}

showType({ id: 1, name: 'test' });

// Partial<T>: 将T类型的所有属性设为可选

interface PartialType {
  id: number;
  firstName: string;
  lastName: string;
}

function showType(args: Partial<PartialType>) {
  console.log(args);
}

showType({ id: 1 });

// Required<T>: 将某个类型里的属性全部变为必选项

interface RequiredType {
  id: number;
  firstName?: string;
  lastName?: string;
}

function showType(args: Required<RequiredType>) {
  console.log(args);
}

showType({ id: 1, firstName: 'John', lastName: 'Doe' });

// Readonly<T>: 将属性变为只读

```

```ts
interface ReadonlyType {
  id: number;
  name: string;
}

function showType(args: Readonly<ReadonlyType>) {
  args.id = 4; // 报错
  console.log(args);
}

showType({ id: 1, name: 'Doe' });

// Pick<T, K>：从一个已存在的类型 T中选择一些属性作为K，从而创建一个新类型


interface PickType {
  id: number;
  firstName: string;
  lastName: string;
}

function showType(args: Pick<PickType, 'firstName' | 'lastName'>) {
  console.log(args);
}

showType({ firstName: 'John', lastName: 'Doe' });

// Omit<T, K>：从类型T中删除K个属性。

interface PickType {
  id: number;
  firstName: string;
  lastName: string;
}

function showType(args: Omit<PickType, 'firstName' | 'lastName'>) {
  console.log(args);
}

showType({ id: 7 });

// Extract<T, U>：从T中提取所有可分配给U的属性。

interface FirstType {
  id: number;
  firstName: string;
  lastName: string;
```

```ts
}

interface SecondType {
  id: number;
  address: string;
  city: string;
}

type ExtractType = Extract<keyof FirstType, keyof SecondType>;
// Output: "id"

// Exclude<T, U>: 从 T 中剔除可以赋值给 U 的类型。

interface FirstType {
  id: number;
  firstName: string;
  lastName: string;
}

interface SecondType {
  id: number;
  address: string;
  city: string;
}

type ExcludeType = Exclude<keyof FirstType, keyof SecondType>;
// Output; "firstName" | "lastName"

// Record<K,T>: 构造具有给定类型T的一组属性K的类型

interface EmployeeType {
  id: number;
  fullname: string;
  role: string;
}

let employees: Record<number, EmployeeType> = {
  0: { id: 1, fullname: 'John Doe', role: 'Designer' },
  1: { id: 2, fullname: 'Ibrahima Fall', role: 'Developer' },
  2: { id: 3, fullname: 'Sara Duckson', role: 'Developer' },
};

// NonNullable<T>: 从 T 中剔除 null 和 undefined

// Mapped Types( 映射类型)

// Type Guards(类型保护)
```

使用in检查参数对象上是否存在属性x。

// Conditional Types(条件类型)

T extends U ? X : Y ，即如果类型T可以被赋值给类型U，那么结果类型就是X类型，否则为Y类型。

3.https://ahooks.gitee.io/zh-CN/hooks/use-request/index ahooks文档

4.https://usehooks-ts.com/react-hook/use-countdown usehooks-ts

useCountDown

实现

```ts
import { useBoolean, useCounter, useInterval } from 'usehooks-ts'

interface CountdownOption {
  countStart: number
  intervalMs?: number
  isIncrement?: boolean
  countStop?: number
}
interface CountdownControllers {
  startCountdown: () => void
  stopCountdown: () => void
  resetCountdown: () => void
}

export function useCountdown(
  countdownOption: CountdownOption,
): [number, CountdownControllers] {

  let isDeprecated = false

  let countStart,
    intervalMs,
    isIncrement: boolean | undefined,
    countStop: number | undefined

  ;({ countStart, intervalMs, isIncrement, countStop } = countdownOption)


  intervalMs = intervalMs ?? 1000
  isIncrement = isIncrement ?? false
  countStop = countStop ?? 0
```

```
32
33    // count == countStart, increment +1 ,decrement -1
34    const {
35      count,
36      increment,
37      decrement,
38      reset: resetCounter,
39    } = useCounter(countStart)
40
41    const {
42      value: isCountdownRunning,
43      setTrue: startCountdown,
44      setFalse: stopCountdown,
45    } = useBoolean(false)
46
47    const resetCountdown = () => {
48      stopCountdown()
49      resetCounter()
50    }
51
52    const countdownCallback = useCallback(() => {
53      if (count === countStop) {
54        stopCountdown()
55        return
56      }
57
58      if (isIncrement) {
59        increment()
60      } else {
61        decrement()
62      }
63    }, [count, countStop, decrement, increment, isIncrement, stopCountdown])
64
65    // 定时器，如果true，则执行一次
66    useInterval(countdownCallback, isCountdownRunning ? intervalMs : null)
67
68    return
69     [
70          count,
71          {
72            startCountdown,
73            stopCountdown,
74            resetCountdown,
75          } as CountdownControllers,
76        ]
77  }
```

useDebounce

实现

```ts
import { useEffect, useState } from 'react'

export function useDebounce<T>(value: T, delay?: number): T {
  const [debouncedValue, setDebouncedValue] = useState<T>(value)

  useEffect(() => {
    const timer = setTimeout(() => setDebouncedValue(value), delay || 500)

    return () => {
      clearTimeout(timer)
    }
  }, [value, delay])

  return debouncedValue
}
```

useEventListener

```ts
import { useEventListener } from 'usehooks-ts'

export default function Component() {
  const buttonRef = useRef<HTMLButtonElement>(null)
  const documentRef = useRef<Document>(document)

  const onScroll = (event: Event) => {
    console.log('window scrolled!', event)
  }

  const onClick = (event: Event) => {
    console.log('button clicked!', event)
  }

  const onVisibilityChange = (event: Event) => {
    console.log('doc visibility changed!', {
      isVisible: !document.hidden,
      event,
    })
  }
  useEventListener('scroll', onScroll)
  useEventListener('visibilitychange', onVisibilityChange, documentRef)
  useEventListener('click', onClick, buttonRef)
```

```
24
25    return (
26      <div style={{ minHeight: '200vh' }}><button ref={buttonRef}>Click me</button
27    )
28  }
```

实现

```
1  import { RefObject, useEffect, useRef } from 'react'
2
3  import { useIsomorphicLayoutEffect } from 'usehooks-ts'
4
5  // Window Event
6  function useEventListener<K extends keyof WindowEventMap>(
7    eventName: K,
8    handler: (event: WindowEventMap[K]) => void,
9    element?: undefined,
10   options?: boolean | AddEventListenerOptions,
11 ): void
12
13 // Element Event
14 function useEventListener<
15   K extends keyof HTMLElementEventMap,
16   T extends HTMLElement = HTMLDivElement,
17 >(
18   eventName: K,
19   handler: (event: HTMLElementEventMap[K]) => void,
20   element: RefObject<T>,
21   options?: boolean | AddEventListenerOptions,
22 ): void
23
24 // Document Event
25 function useEventListener<K extends keyof DocumentEventMap>(
26   eventName: K,
27   handler: (event: DocumentEventMap[K]) => void,
28   element: RefObject<Document>,
29   options?: boolean | AddEventListenerOptions,
30 ): void
31
32 function useEventListener<
33   KW extends keyof WindowEventMap,
34   KH extends keyof HTMLElementEventMap,
35   T extends HTMLElement | void = void,
36 >(
37   eventName: KW | KH,
```

```
  handler: (
    event:
      | WindowEventMap[KW]
      | HTMLElementEventMap[KH]
      | Event,
  ) => void,
  element?: RefObject<T>,
  options?: boolean | AddEventListenerOptions,
) {
  const savedHandler = useRef(handler)

  // 判断当前是浏览器环境还是服务器环境
  useIsomorphicLayoutEffect(() => {
    savedHandler.current = handler
  }, [handler])

  useEffect(() => {
    // 获取当前DOM元素
    const targetElement: T | Window = element?.current ?? window
    // 目标元素不存在
    if (!(targetElement && targetElement.addEventListener)) return
    // linstener处理函数：类型是handler，执行handler
    const listener: typeof handler = event => savedHandler.current(event)

    targetElement.addEventListener(eventName, listener, options)

    return () => {
      targetElement.removeEventListener(eventName, listener, options)
    }
  }, [eventName, element, options])
}

export { useEventListener }
```

useScroll

实现

```
import { RefObject, useEffect, useRef, useState } from 'react';

interface State {
  x: number;
  y: number;
}

const useScroll = (ref: RefObject<HTMLElement>): State => {
```

```tsx
  const frame = useRef(0);
  const [state, setState] = useState<State>({
    x: 0,
    y: 0,
  });

  useEffect(() => {
    // 在这个新的动画帧中，它会获取一个元素的滚动位置并更新状态。
    // 取消之前的动画帧请求并创建一个新的动画帧请求。
    const handler = () => {
      // 取消了当前正在等待执行的动画帧
      cancelAnimationFrame(frame.current);
      // 告诉浏览器在下次重绘之前调用指定的回调函数来更新动画
      frame.current = requestAnimationFrame(() => {
        if (ref.current) {
          setState({
            x: ref.current.scrollLeft,
            y: ref.current.scrollTop,
          });
        }
      });
    };

    if (ref.current) {
      ref.current.addEventListener('scroll', handler, {
        // 在冒泡阶段处理
        capture: false,
        // 浏览器可以在没有用户交互的情况下进行滚动
        passive: true,
      });
    }

    return () => {
      if (frame.current) {
        cancelAnimationFrame(frame.current);
      }

      if (ref.current) {
        ref.current.removeEventListener('scroll', handler);
      }
    };
  }, [ref.current]);

  return state;
};

export default useScroll;
```

5.https://github.com/zenghongtu/react-use-chinese/blob/master/README.md react-use

6.vite，rollup插件开发

https://github.com/rollup/plugins/blob/master/packages/dynamic-import-vars/src/index.js rollup-plugin-inline-dynamic-imports插件

https://cn.vitejs.dev/guide/api-plugin.html#vite-specific-hooks vite插件开发

https://cn.rollupjs.org/plugin-development/ rollu插件开发

实现一个插件，将文件中所有console.log移除

https://github.com/xiaoxian521/vite-plugin-remove-console/tree/main 移除console.log插件

- 7.动画
  - 1.Web Animation API https://developer.mozilla.org/zh-CN/docs/Web/API/Web_Animations_API
- 8.框架
  - 1.Svelte https://www.svelte.cn/tutorial/basics
  - 2.Vue
    - 1.https://vue3js.cn/ Vue3生态文档，Vue3源码
  - 3.https://zh-hans.react.dev/ React文档
- 9.Web Components
  - 1.Lit https://lit.dev/docs/
- 10.Form
  - 1.https://react-hook-form.com/get-started 处理Form表单

💡 AI

1.https://github.com/Pythagora-io/gpt-pilot#how-to-start-using-gpt-pilot

Gpt pilot加快代码构建速度

2.https://open.bigmodel.cn/overview

智谱AI 开发AI应用

💡 算法

1.https://github.com/afatcoder/LeetcodeTop

高频笔试算法题

2.https://github.com/geekxh/hello-algorithm

算法学习路线

3.https://github.com/krahets/hello-algo

含jsPDF的算法学习

✍️ 项目部署

1.https://vercel.com/sunnyyees-projects/vercel-vite-project/8gJpQNj6MfGWgfjbgtXFG1NkJdfA

Vercel

🚄 无代码开发

1.https://zion.functorz.com/userCenter/personal

Zion

🍞 自动化

1.http://doc.autoxjs.com/#/?id=%e7%bb%bc%e8%bf%b0

AutoX.js