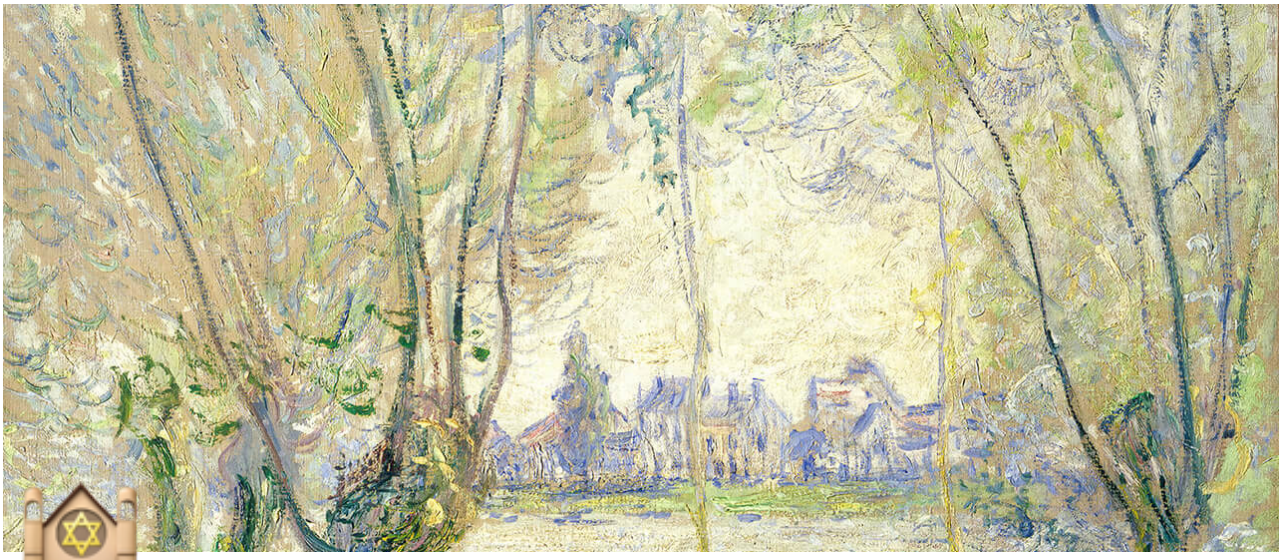
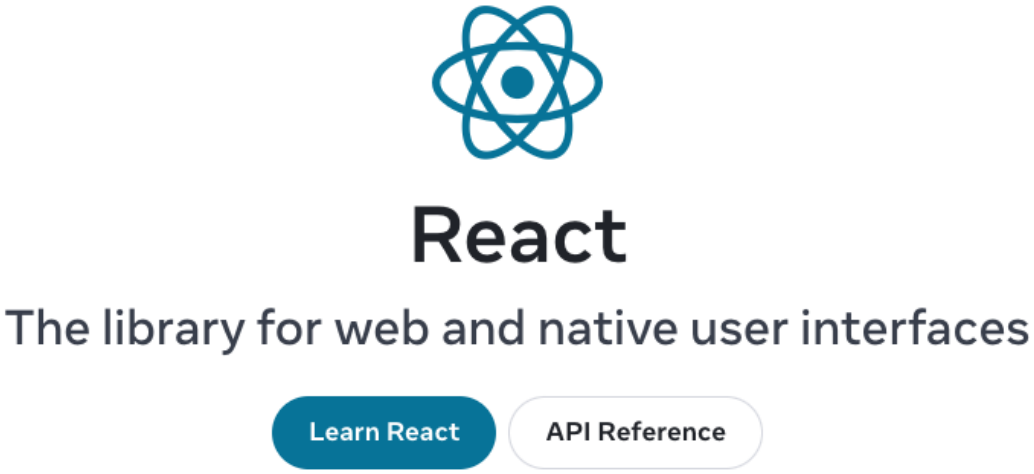


2-3 Thinking in React/用 React 的方式写 React



资源

1. React 文档-英文
2. React 文档-中文
3. React github

React

React，用于构建 Web 和原生交互界面的库。

组件化

React，用组件创建用户界面。

因此在我们使用 React 构建用户界面时，首先要把它分解成一个个的组件。然后，把这些组件组合在一起，使用数据流来进行通信。

状态 state

React UI = fn(state)

fn(x) = x+1

state 是变量，一般情况下，state 改变之后，组件会更新。

当你有一个值，

- 这个值变化之后，页面需要随之更新，那么这个变量就可以定义成 **state，比如 useState、useReducer、第三方状态管理库**。官网链接：state：组件的记忆
- 如果你只是需要记忆这个值，但是这个值的改变之后，页面不需要随之更新，那么可以使用 **useRef** 定义。官网链接：useRef
- 如果你不需要记忆，那么就是个普通变量。如果你需要缓存这个值，那么可以使用 useMemo 或者 useCallback。官网链接：useMemo、useCallback

> *注意：使用 useState 的时候，如果 state 不发生变化，组件不会更新。但是 useReducer 反之。*

关于状态，更多参看下个章节《React 中的状态管理与状态管理库》

单向数据流

单向数据流，即对不可变数据状态使用单向变换。

React 的基础模型就是单向数据流，我们在修改 state 的时候，不会像 Vue 那样直接修改，而是通过 setState 函数进行修改。

React 的传统状态管理库 Redux 亦是遵循如此模式，Redux 使用 reducer 作为数据修改规则，而 reducer 是纯函数，不能直接修改 state，只能返回新的 state。

如下一个 React input 的例子：

JavaScript

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

export default function Input() {
  const [input, setInput] = useState("");

  return <input value={input} onChange={(e) => setInput(e.target.value
}
```

如下是一个 Vue input 的例子：

Vue

```
<script setup>

import { ref } from 'vue'
// 文本插值
const msg = ref('Hello')

</script>


<template>
    <input v-model="msg" />
</template>
```

但是，这个**数据不可变性**亦是被很多初学者吐槽，觉得~~学习成本比较高~~ ~

因此现在一些周边库也支持在 React 中实现直接修改数据，比如 Immer。