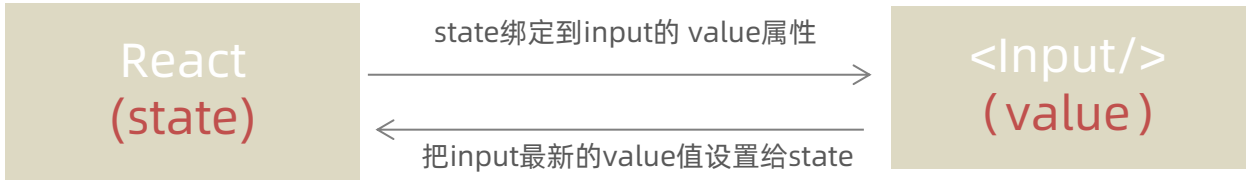
受控表单绑定

受控表单绑定

概念：使用React组件的状态（useState）控制表单的状态



1. 准备一个React状态值

```
const [value, setValue] = useState('')
```

2. 通过value属性绑定状态，通过onChange属性绑定状态同步的函数

```
<input
  type="text"
  value={value}
  onChange={(e) => setValue(e.target.value)}
/>
```

React中获取DOM

React中获取DOM

在 React 组件中获取/操作 DOM，需要使用 useRef React Hook钩子函数，分为两步：

1. 使用useRef创建 ref 对象，并与 JSX 绑定

```
const inputRef = useRef(null)
```

```
<input type="text" ref={inputRef} />
```

2. 在DOM可用时，通过 inputRef.current 拿到 DOM 对象

```
console.log(inputRef.current)
```

案例：B站评论 — 发表评论

B站评论案例 —— 核心功能实现

1. 获取评论内容

2. 点击发布按钮发布评论

B站评论案例 — id处理和时间处理

```
{
  rpid: 100,
  user: {
    uid: '30009257',
    avatar,
    uname: '黑马前端',
  },
  content: content,
  ctime: '10-19 09:00',
  like: 66,
}
```

1. rpid要求一个唯一的随机数id - uuid

2. ctime要求以当前时间为标准，生成固定格式 - dayjs

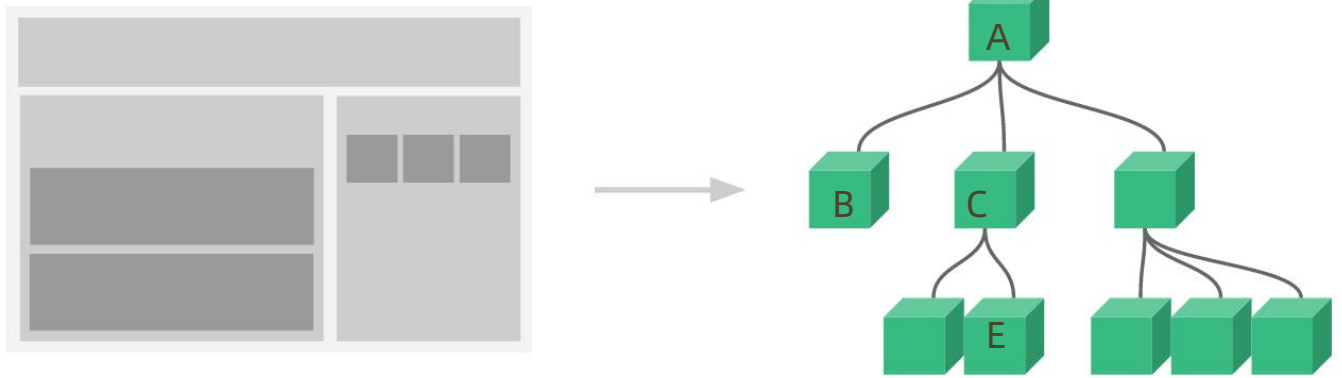
B站评论案例 — 清空内容并重新聚焦

1. 清空内容 - 把控制input框的value状态设置为空串
2. 重新聚焦 - 拿到input的dom元素，调用focus方法

组件通信

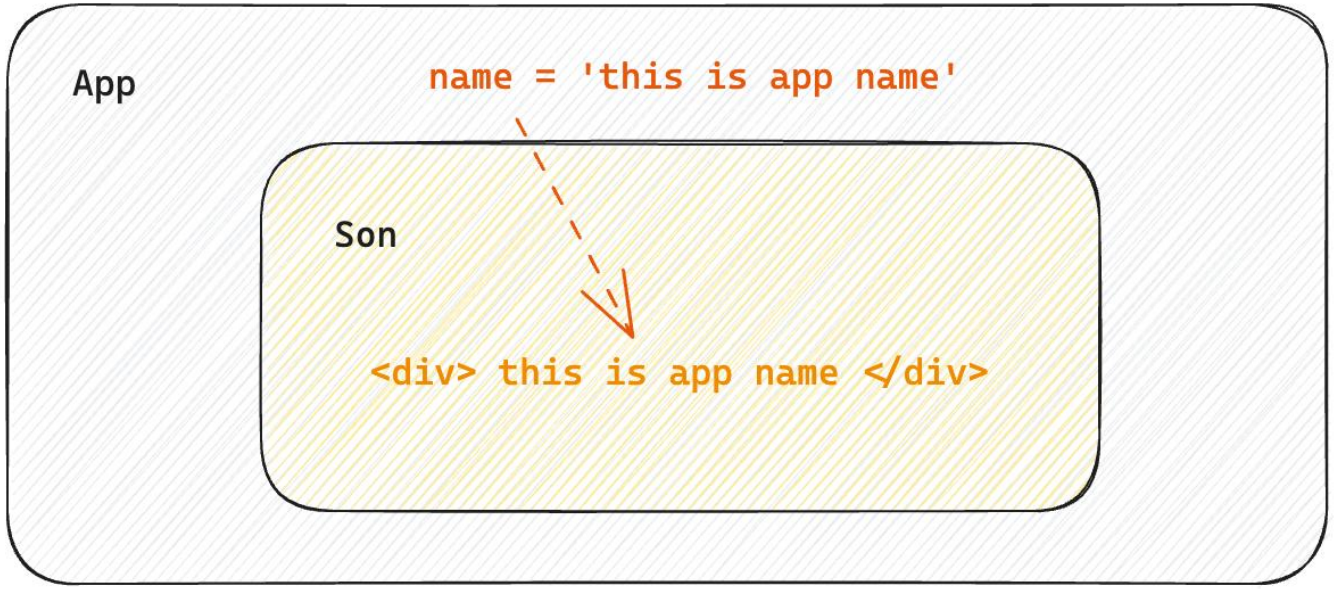
理解组件通信

概念：组件通信就是组件之间的数据传递，根据组件嵌套关系的不同，有不同的通信方法



A-B 父子通信　　B-C 兄弟通信　　A-E 跨层通信

父传子-基础实现



实现步骤

1. 父组件传递数据 - 在子组件标签上绑定属性
2. 子组件接收数据 - 子组件通过props参数接收数据

父传子-props说明

1. props可传递任意的数据

数字、字符串、布尔值、数组、对象、函数、JSX

```jsx
<Son
  name={appName}
  age={20}
  isTrue={false}
  list={['Vue', 'React']}
  obj={{ name: 'jack' }}
  cb={() => console.log(123)}
  child={<span>this is span child</span>}
/>
```

```
props
  age: 20
  cb: ƒ cb() {}
  child: <span />
  isTrue: false
▸ list: ["Vue", "React"]
  name: "this is app name"
▸ obj: {name: "jack"}
```

2. props是只读对象

子组件只能读取props中的数据，不能直接进行修改，父组件的数据只能由父组件修改

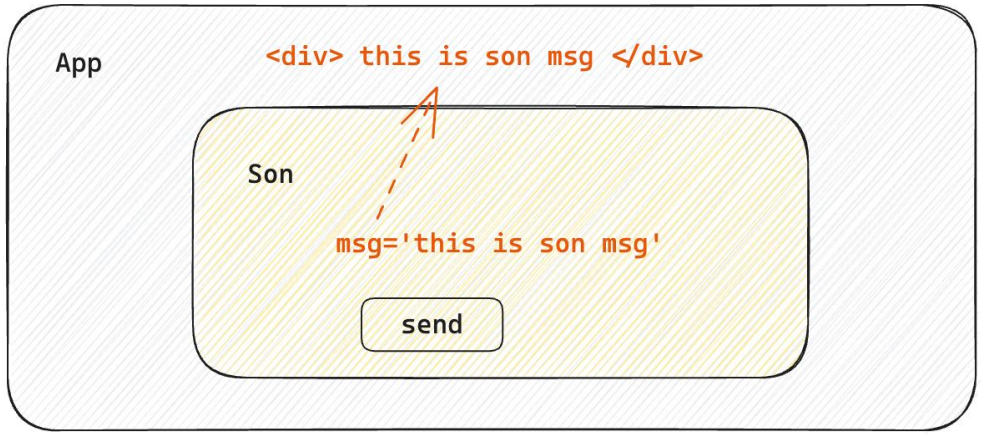
父传子 - 特殊的prop children

场景：当我们把内容嵌套在子组件标签中时，父组件会自动在名为children的prop属性中接收该内容

```
<Son>
  <span>this is span</span>
</Son>
```

```
props
  children: <span />
  new entry: ""
```

父子组件通信-子传父

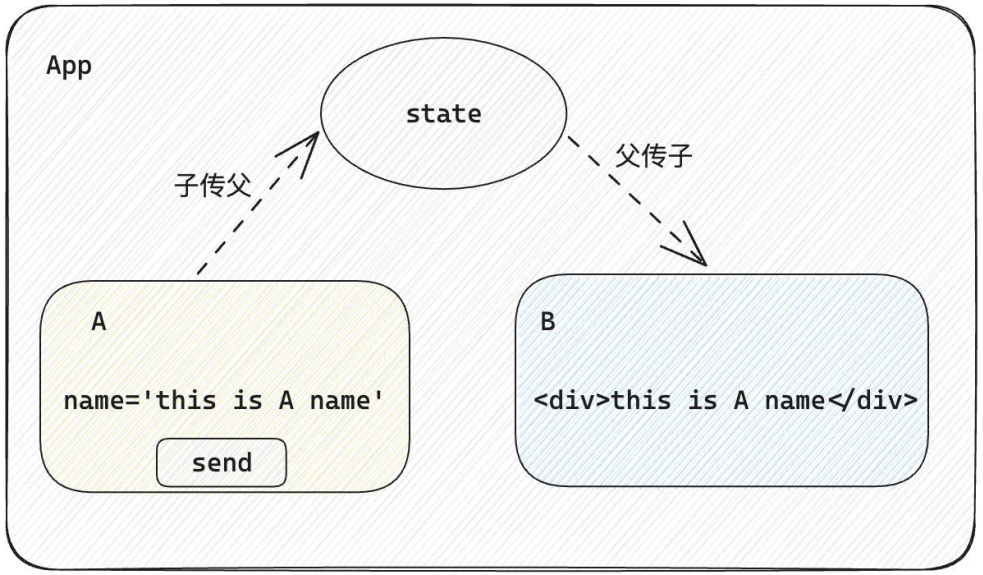


核心思路：在子组件中调用父组件中的函数并传递参数

```jsx
function App () {
  const getMsg = (msg) => console.log(msg)
  return (
    <div>
      <Son onGetMsg={getMsg} />
    </div>
  )
}
```

```jsx
function Son ({ onGetMsg }) {
  const sonMsg = 'this is son msg'
  return (
    <div>
      <button onClick={() => onGetMsg(sonMsg)}>send</button>
    </div >
  )
}
```

使用状态提升实现兄弟组件通信

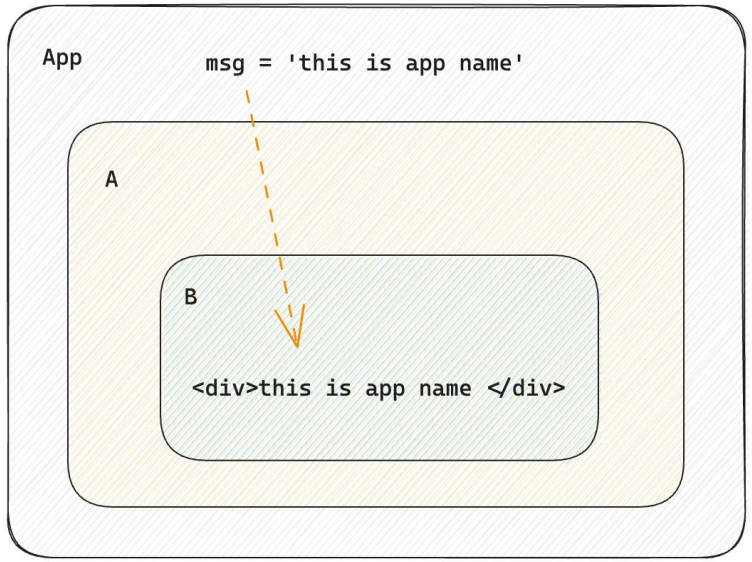


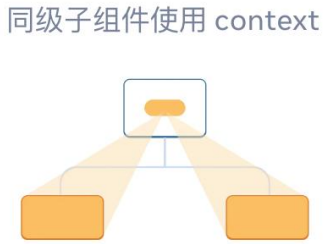
实现思路：借助“状态提升”机制，通过父组件进行兄弟组件之间的数据传递

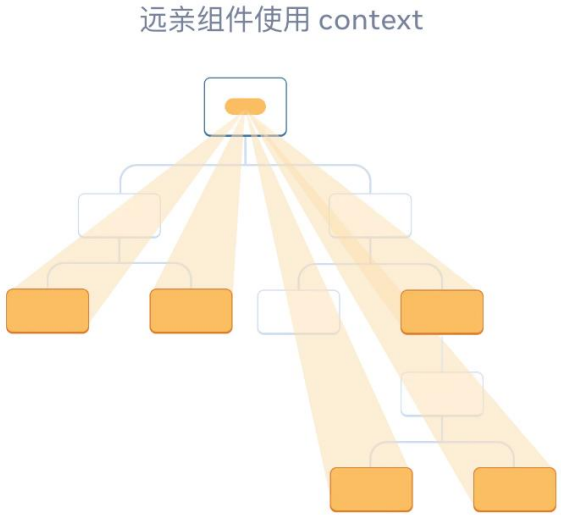
1. A组件先通过子传父的方式把数据传给父组件App

2. App拿到数据后通过父传子的方式再传递给B组件

使用Context机制跨层级组件通信







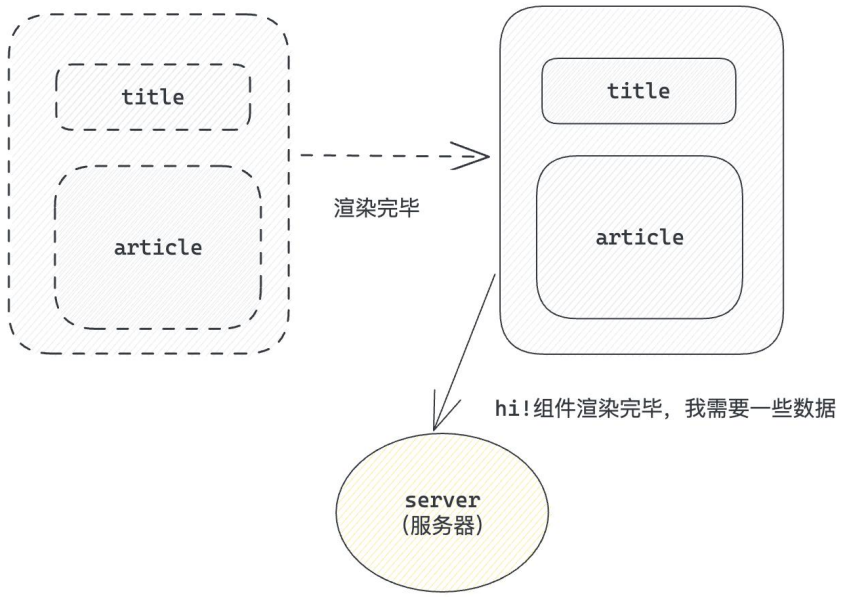
实现步骤：

1. 使用createContext方法创建一个上下文对象Ctx
2. 在顶层组件（App）中通过 Ctx.Provider 组件提供数据
3. 在底层组件（B）中通过 useContext 钩子函数获取消费数据

useEffect 的使用

useEffect 的概念理解

useEffect是一个React Hook函数，用于在React组件中创建不是由事件引起而是由渲染本身引起的操作（副作用），比如发送AJAX请求，更改DOM等等



说明：上面的组件中没有发生任何的用户事件，组件渲染完毕之后就需要和服务器要数据，整个过程属于“只由渲染引起的操作”

useEffect 的基础使用

需求：在组件渲染完毕之后，立刻从服务端获取频道列表数据并显示到页面中

语法：

```
useEffect(() => { }, [])
```

参数1是一个函数，可以把它叫做副作用函数，在函数内部可以放置要执行的操作

参数2是一个数组（可选参），在数组里放置依赖项，不同依赖项会影响第一个参数函数的执行，当是一个空数组的时候，副作用函数只会在组件渲染完毕之后执行一次

接口地址：http://geek.itheima.net/v1_0/channels

useEffect 依赖项参数说明

useEffect副作用函数的执行时机存在多种情况，根据传入依赖项的不同，会有不同的执行表现

依赖项	副作用函数执行时机
没有依赖项	组件初始渲染 + 组件更新时执行
空数组依赖	只在初始渲染时执行一次
添加特定依赖项	组件初始渲染 + 特性依赖项变化时执行

useEffect — 清除副作用

在useEffect中编写的由渲染本身引起的对接组件外部的操作，社区也经常把它叫做副作用操作，比如在useEffect中开启了一个定时器，我们想在组件卸载时把这个定时器再清理掉，这个过程就是清理副作用

```
1  useEffect(() => {
2    // 实现副作用操作逻辑
3    return () => {
4      // 清除副作用逻辑
5    }
6  }, [])
```

说明：清除副作用的函数最常见的执行时机是在组件卸载时自动执行

需求：在Son组件渲染时开启一个定制器，卸载时清除这个定时器

自定义Hook实现

自定义Hook函数

概念：自定义Hook是以 use 打头的函数，通过自定义Hook函数可以用来实现逻辑的封装和复用

不封装直接实现 → 封装自定义Hook实现 → 抽象实现的通用逻辑

React Hooks使用规则

ReactHooks使用规则

使用规则

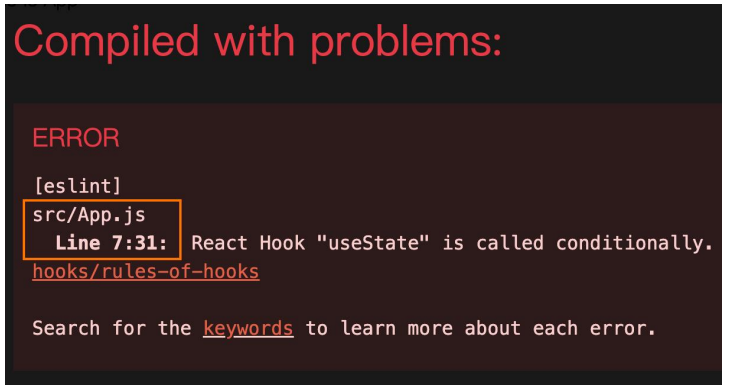
1. 只能在组件中或者其他自定义Hook函数中调用

2. 只能在组件的顶层调用，不能嵌套在 if、for、其他函数中

```
const [value, setValue] = useState('')

function App () {
  return (
    <div>
      this is App
    </div>
  )
}
```

```
function App () {
  if (Math.random() > 0.5) {
    const [value, setValue] = useState('')
  }
  return (
    <div>
      this is App
    </div>
  )
}
```



案例：优化B站评论案例

优化需求

1. 使用请求接口的方式获取评论列表并渲染
2. 使用自定义Hook函数封装数据请求的逻辑
3. 把评论中的每一项抽象成一个独立的组件实现渲染

优化需求-通过接口获取评论列表

1. 使用 json-server 工具模拟接口服务, 通过 axios 发送接口请求

   json-server是一个快速以.json文件作为数据源模拟接口服务的工具

   axios是一个广泛使用的前端请求库

2. 使用 useEffect 调用接口获取数据

```
useEffect(() => {
  // 发送网络请求
}, [])
```

优化需求-封装评论项Item组件

抽象原则：App作为“智能组件”负责数据的获取，Item作为“UI组件”负责数据的渲染

传智教育旗下高端IT教育品牌