21-2 拓展：哪些 React 未正式发布的功能

资源

1. React 文档
2. https://zh-hans.react.dev/community/versioning-policy#canary-channel
3. https://zh-hans.react.dev/blog/2024/02/15/react-labs-what-we-have-been-working-on-february-2024#react-compiler

从 React 官网上，大家可以看到很多 API，但是这些 API 却无法再正式环境当中使用，比如：

这些其实都是 React 未正式发布的 API。正如以前的 Hook、Concurrent 模式，都是先只可在 experimental 模式下使用，然后正式发布以后，这些相关 API 才可以在正式环境下使用。

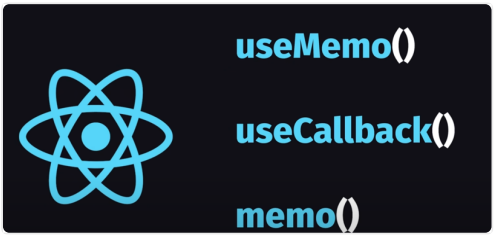
那么 React 中有哪些未正式发布的 API 呢，从 React 的官网提示上我们就可以看到 ~

React Labs: What We've Been Working On – February 2024

1. React Compiler

React forget → React Compiler

现在手动记忆化




```
function App() {
  const [count, setCount] = useState(0)

  const doubled = useMemo(() => {
    return count * 2
  }, [count])
```

App.vue > {} script setup

```vue
const count = ref(0)

const doubled = computed(() => count.value * 2)
```

以后自动记忆化?

waiting...

2. Actions

Action 允许将一个函数传递给诸如 `<form/>` 等 DOM 元素：

React JSX

```jsx
<form action={search}>
  <input name="query" />
  <button type="submit">搜索</button>
</form>
```

`action` 函数可以同步或异步执行。你可以在客户端使用标准 JavaScript 定义它们，也可以在服务器上使用 `'use server'` 指示符。当使用 action 时，React 将帮助管理数据提交的生命周期，提供类似 `useFormStatus` 和 `useFormState` 的 Hook，以访问表单操作的当前 state 与响应。

默认情况下，Action 在 transition 中提交，使当前页面在操作处理过程中保持交互性。由于 Action 支持异步函数，我们还添加了在 transitions 中使用 `async/await` 的功能，这允许在异步请求（如 `fetch`）开始时使用转换的 `isPending` 状态显示待处理 UI，并在应用更新时始终显示待处理 UI。

除了 Action，我们还引入了一个名为 `useOptimistic` 的功能，用于管理**乐观状态更新**。使用此 Hook 可以应用临时更新，一旦最终状态提交，它们就会自动回滚。对于 Action，这将帮助乐观地设置客户端数据的最终状态，假设提交成功，并恢复为从服务器接收到的数据值。它使用常规的 `async` / `await`，因此无论是在客户端上使用 `fetch` 还是在服务器上使用 Server Action，都可以工作。

库作者可以使用 `useTransition` 在自己的组件中实现自定义 `action={fn}` props。我们的目的是，当设计他们的组件 API 时，库应采用 Action 模式，为

React 开发人员提供一致的体验。例如，如果你的库提供了一个 `<Calendar onSelect={eventHandler}>` 组件，则还可以考虑暴露一个 `<Calendar selectAction={action}>` API。

尽管我们最初专注于 Server Action 用于客户端/服务器数据传输，但我们对 React 的理念是在所有平台和环境中提供相同的编程模型。在可能的情况下，如果我们在客户端引入一个功能，我们也会使它在服务器上起作用，反之亦然。这一理念使我们能够创建一组 API，无论您的应用在何处运行，都可以工作，从而使以后更容易升级到不同的环境。

Action 现在在 Canary 通道中可用，并将在下一个 React 发布版本中发布。

3. React Canary 版本中的新特性

React 服务器组件、资源加载、文档元数据与 Action 都已经加入了 React Canary，并且我们已经在 react.dev 上为这些功能添加了文档：

- **指示符**：`"use client"` 与 `"use server"` 是设计用于全栈 React 框架的打包功能。它们标记了两个环境之间的“分割点”：use client 指示符指示打包工具生成一个 `<script>` 标签（类似于 Astro Islands），而 use server 告诉打包工具生成一个 POST 端点（类似于 tRPC Mutations）。它们让你可以编写将客户端交互性与相关的服务器端逻辑组合在一起的可重用组件。
- **文档元数据**：我们内置支持在组件树中的任何位置渲染 `<title>`、`<meta>` 和元数据 `<link>` 标签。这些在所有环境中都以相同的方式工作，包括完全客户端代码、SSR 和 RSC。这为像 React Helmet 这样的库开创的功能提供了内置支持。
- **资源加载**：我们将 Suspense 与样式表、字体和脚本等资源的加载生命周期集成在一起，以便 React 考虑它们来确定像 `<style>`、`<link>` 和 `<script>` 这样的元素中的内容是否已准备就绪。我们还添加了新的 资源加载 API，如 `preload` 和 `preinit`，以提供更大的控制权，指示何时应加载和初始化资源。
- **Action**：如上所述，我们已将 Action 添加到管理从客户端发送数据到服务器的功能中。现在可以将 `action` 添加到像 `<form/>` 这样的元素中，使用 `useFormStatus` 访问状态，使用 `useFormState` 处理结果，并使用 `useOptimistic` 乐观地更新 UI。

4. React 的下一个主要版本

经过几年的迭代，`react@canary` 现在已经准备好发布到 `react@latest`。上面提到的新功能与应用程序运行的任何环境兼容，提供了生产使用所需的一切。由于资源加载与文档元数据可能对一些应用程序造成破坏性变化，因此下一个 React 版本将是一个主要版本：**React 19**。

我们仍然需要做更多准备工作才能发布。在 React 19 中，我们还将添加一些长期请求的改进，这些改进需要进行破坏性更改，如支持 Web Components。我们现在的重点是完成这些改进、为新功能制定最终文档，并发布关于包含哪些内容的公告。

5. Offscreen（已重命名为 Activity）

“Offscreen”意味着它仅适用于不可见的应用程序部分，但在研究该功能时，我们意识到应用程序的某些部分可能是可见但不活动的，例如模态框后面的内容。新名称更贴近于标记应用程序的某些部分为“active”或“inactive”的行为。

Activity 仍处于研究阶段。