为什么要在Vue3中多使用Hooks？好处是啥？

拥抱 Hooks!

最近几年，“拥抱 Hooks” 的口号呼声非常高，一开始是 `React`，自动 `Vue3 setup script` 语法的推出之后，现在写 `Vue3` 时也越来越离不开 `Hooks` 了

但是还是有很多人不解，没 Hooks 之前我们也能完成代码需求啊？所以 `Hooks` 到底好在哪呢？对我们的代码开发到底有什么好处呢？

Hooks 的概念？

其实很多人对于 `Hooks` 的概念很模糊，包括我自己，在查阅了一些资料后，我说说我自己的浅见，`Hooks` 就是 `钩子` 的意思，所以 `Hook functions` 也叫 `钩子函数`，我理解的 `钩子函数` 的意思是：

在特定的时机会执行的函数

比如我们在开发中遇到的：

- **点击函数：** 点击时才会执行的函数
- **定时器函数：** 时间到了就会执行的函数
- **生命周期函数：** 在组件各个时间点执行的函数
- **拦截器事件：** 请求和响应时执行的函数
- **某个值改变而执行的函数：** 例如 React Hooks/Vue Hooks
- **github钩子：** 例如 husky 的代码提交前代码检验，github ci 的监听代码提交进行构建

Vue Hooks

先来聊聊 mixins

在 Vue2 时代，`mixins` 是一个为了提高代码复用性而推出的功能，但是官方不推荐使用，这是为啥呢？我们来看一个例子，你就知道使用 `mixins` 有多难受

```
1 // mixin1
2 export default {
3   created() {
4       console.log('我是ikun一号')
5   },
6   method: {
7       sayKunkun() {
```

```
            console.log('kunkun好帅~')
        }
    }
}

// mixin2
export default {
    method: {
        say() {
            this.sayKunkun();
        }
    }
}

// index.vue
export default {
    mixins: [mixin1, mixin2],
    created() {
        this.say()
        this.love()
    },
    method: {
        say() {
            console.log('index.vue ikun')
        },
        love() {
            this.sayKunkun()
        }
    }
}
```

上面有两个 `mixins` 混入了 index.vue ，我来看看最终的输出结果是怎么样的~

```
我是ikun一号
index.vue ikun
kunkun好帅~
```

通过这三个输出，我们可以发现三个现象：

- mixin的 craeted 和 index.vue 的 created 合并执行了
- index.vue 的 say 函数顶掉了 mixin 的 say 函数
- mixin2 居然能访问到 mixin1 的 sayKunkun 函数

上面三个现象都是 mixins 的正常现象，但是这样有很多隐患，当你使用 mixins 去提取公用代码时，若是一个 mixins 文件，那还好说，怎样都行；当 mixins 文件达到多个，去维护修改时就会不知道这个方法、属性来自那个mixins 文件；更不用说，若是每个 mixins 文件功能不独立，mixins 之间相互调用，那就真的是一团乱麻了，就算自己写的，过两天来看，也是一脸懵逼，那时就是 开发一时爽，维护火葬场了

Hooks 取代 Mixins

再来一个例子，我想要维护一套显隐变量，如果使用 mixins 我需要这么做

```
// mixin
export default {
    data() {
        return {
            loading: false
        }
    },
    method: {
        show() {
            this.loading = true
        },
        hiden() {
            this.loading = false
        }
    }
}

// index.vue
<table loading="loading"></table>

export default {
    mixins: [mixin],
    method: {
        handleHiden() {
            this.hiden()
        },
        handleShow() {
            this.show()
        }
    }
}
```

而我们使用 Hooks 来做的话，需要封装一个以 `use` 开头的函数，自定义 Hooks 有一个潜规则，就是要 `use` 开头

```ts
// useLoading.ts
import { ref } from 'vue'
export useLoading = () => {
    const loading = ref(false)
    const show = () => {
        loading.value = true
    }
    const hiden = () => {
        loading.value = false
    }

    return {
        loading,
        hiden,
        show
    }
}

// index.vue
<table loading="loading"></table>

<script setup lang="ts">
import { useLoading } from './hooks/useLoading.ts'

const {
    loading,
    hiden,
    show
} = useLoading()
</script>

```

以上就是一简单的 `自定义 Hooks` 的实践，其实 `自定义 Hooks` 本质还是为了提高代码的可复用性~

但是这个时候可能就会有朋友说了，这个 `useLoading` 其实不就相当于一个函数吗？这就涉及到了 `utils` 和 `Vue 自定义Hooks` 的区别：

- utils函数：不涉及响应式的函数
- Vue 自定义Hooks：涉及 Vue 的一些响应式api，比如 ref/reactive/computed/watch/onMounted

Vue3 Hooks 应用场景

接下来就介绍一些常用的 Hooks，以及场景，我一般把 Hooks 分为两种类型

- 业务 Hooks：迎合业务封装的，复用性比较低

- 工具 Hooks：方便整个项目的开发，复用性比较高

业务 Hooks

验证码发送完之后的读秒

我们需要封装一个 计时器 Hooks

```
import { ref } from 'vue'

export function useCountDown() {
  const countNum = ref(0)
  const countInterval = ref(null)

  const startCountDown = num => {
    countNum.value = Number(num)
    clearCountDown()
    countInterval.value = setInterval(() => {
      if (countNum.value === 0) {
        clearInterval(countInterval.value)
        countInterval.value = null
        return
      }
      countNum.value--
    }, 1000)
  }

  const clearCountDown = () => {
    if (countInterval.value) {
      clearInterval(countInterval.value)
    }
  }

  return { countNum, startCountDown, clearCountDown }
}
```

表格 Hooks 的封装

我之前在 vben-admin 这个项目中看到了 `useTable` 这个 Hooks，发现封装的很好，只需要传入一些必要参数，就可以获取一些表格所需要的渲染数据，源码比较多，感兴趣的可以去看看这个项目 vue-vben-admin

```
type Props = Partial<DynamicProps<BasicTableProps>>;

```

```ts
type UseTableMethod = TableActionType & {
  getForm: () => FormActionType;
};

export function useTable(tableProps?: Props): [
  (instance: TableActionType, formInstance: UseTableMethod) => void,
  TableActionType & {
    getForm: () => FormActionType;
  },
] {
  const tableRef = ref<Nullable<TableActionType>>(null);
  const loadedRef = ref<Nullable<boolean>>(false);
  const formRef = ref<Nullable<UseTableMethod>>(null);

  let stopWatch: WatchStopHandle;

  function register(instance: TableActionType, formInstance: UseTableMethod) {
    // ...
  }

  function getTableInstance(): TableActionType {
    // ...
  }

  const methods: TableActionType & {
    getForm: () => FormActionType;
  } = {
    // ...
  };

  return [register, methods];
}

```

i18n 语言切换 Hooks

i18n也是现在很多项目都必不可少的功能，所以封装一个Hooks很有必要

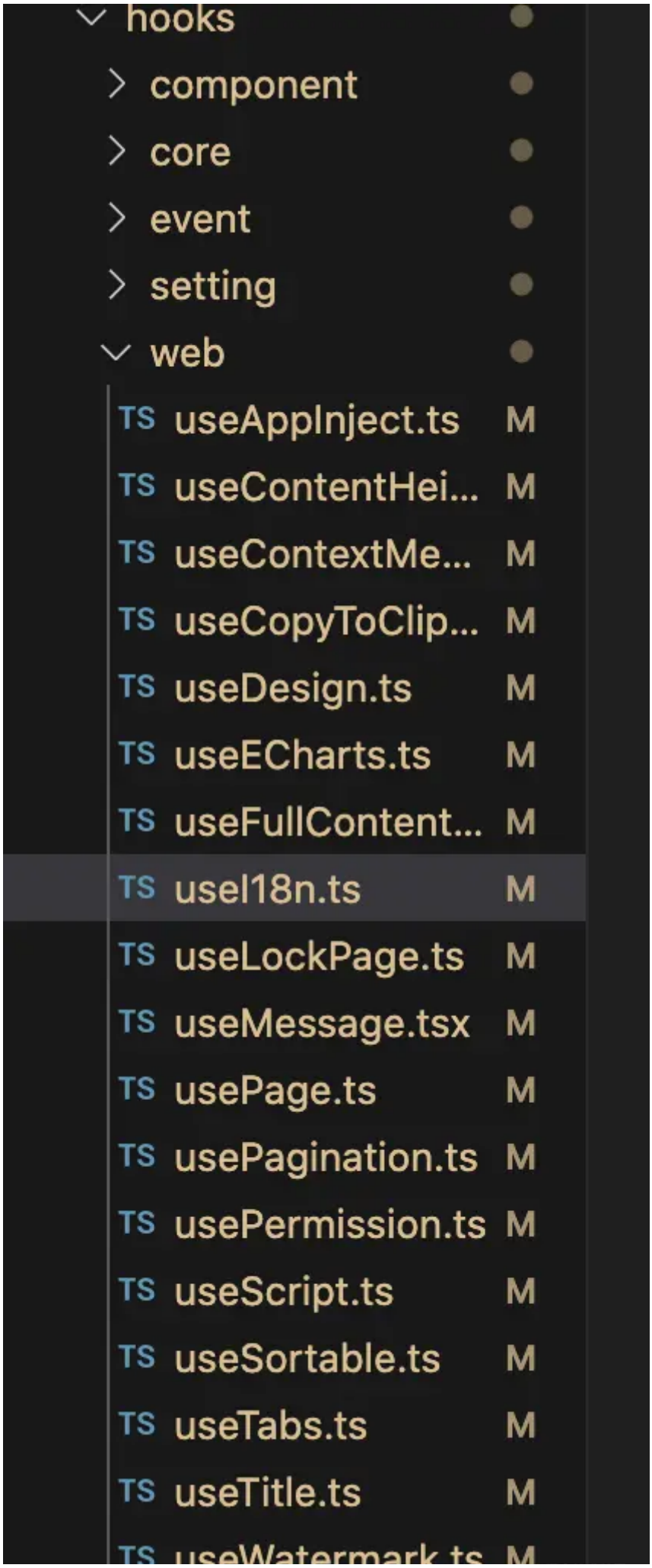
```ts
import { i18n } from '/@/locales/setupI18n';

type I18nGlobalTranslation = {
  (key: string): string;
  (key: string, locale: string): string;
  (key: string, locale: string, list: unknown[]): string;
  (key: string, locale: string, named: Record<string, unknown>): string;
  (key: string, list: unknown[]): string;
```

```ts
  (key: string, named: Record<string, unknown>): string;
};

type I18nTranslationRestParameters = [string, any];

function getKey(namespace: string | undefined, key: string) {
  if (!namespace) {
    return key;
  }
  if (key.startsWith(namespace)) {
    return key;
  }
  return `${namespace}.${key}`;
}

export function useI18n(namespace?: string): {
  t: I18nGlobalTranslation;
} {
  const normalFn = {
    t: (key: string) => {
      return getKey(namespace, key);
    },
  };

  if (!i18n) {
    return normalFn;
  }

  const { t, ...methods } = i18n.global;

  const tFn: I18nGlobalTranslation = (key: string, ...arg: any[]) => {
    if (!key) return '';
    if (!key.includes('.') && !namespace) return key;
    return t(getKey(namespace, key), ...(arg as I18nTranslationRestParameters));
  };
  return {
    ...methods,
    t: tFn,
  };
}

// Why write this function?
// Mainly to configure the vscode i18nn ally plugin. This function is only used for routing and menus. Please use useI18n for other places

// 为什么要编写此函数?
```

```
54 // 主要用于配合vscode i18nn ally插件。此功能仅用于路由和菜单。请在其他地方使用useI18n
55 export const t = (key: string) => key;
56
```

还有很多业务 Hooks

vben-admin 中还有很多封装的很好的业务 Hooks ，大家有兴趣可以去看看代码，学习学习


hooks
component
core
event
setting
web
TS useAppInject.ts M
TS useContentHei... M
TS useContextMe... M
TS useCopyToClip... M
TS useDesign.ts M
TS useECharts.ts M
TS useFullContent... M
TS useI18n.ts M
TS useLockPage.ts M
TS useMessage.tsx M
TS usePage.ts M
TS usePagination.ts M
TS usePermission.ts M
TS useScript.ts M
TS useSortable.ts M
TS useTabs.ts M
TS useTitle.ts M

工具 Hooks

工具 Hooks ，是为了让项目整体的开发代码质量更加高，开发功能更加快捷，其实现在市面上已经有很多很多的 Hooks库 了， `Vueuse` 就是最牛的那个(在vue中)，文档，他提供了很多 Hooks，比如：

1. `useLocalStorage`：提供在本地存储中保存和获取数据的功能。
2. `useMouse`：提供跟踪鼠标位置和鼠标按下状态的功能。
3. `useClipboard`：提供复制文本到剪贴板的功能。
4. `useDebounce`：提供防抖功能，用于延迟执行一个函数，直到一段时间内没有新的触发。
5. `useThrottle`：提供节流功能，用于在一段时间内限制函数的执行频率。
6. `useEventListener`：提供绑定和解绑事件监听器的功能。
7. `useFetch`：提供方便的处理基于 Fetch API 的网络请求的功能。
8. `useIntersectionObserver`：提供对元素是否可见进行观察的功能，可用于实现懒加载等效果。
9. `useRoute`：提供在 Vue Router 中获取当前路由信息的功能。