Vue Mastery

VUE 3 + TYPESCRIPT CHEAT SHEET

Activate TypeScript

```
<script setup lang="ts">
import { reactive } from 'vue'
import Counter from './components/Counter.vue'
```

Create custom type

```
interface AppInfo {
  name: string,
  slogan: string
}
```

Use custom type

```
const appInfo: AppInfo = reactive({
  name: 'Counter',
  slogan: 'an app you can count on'
})

</script>

<template>
  <div>
    <h1>{{ appInfo.name }}</h1>
    <h2>{{ appInfo.slogan }}</h2>
  </div>
  <Counter :limit="10" ></Counter>
</template>
```

TYPE-SAFE

```
<script setup lang="ts">
```

Use custom type

```
const emit = defineEmits<{
  (event: 'add-count', num: number): void
  (event: 'reset-count'): void
}>()

</script>

<template>
  <p>
    <button @click=" emit('add-count', 1) ">
      Add
    </button>
    <button @click=" emit('reset-count') ">
      Reset
    </button>
  </p>
</template>
```

TYPE-SAFE

```
<script setup lang="ts">
import { ref, onMounted } from 'vue'
import fetchCount from '../services/fetchCount'
import ControlBar from './ControlBar.vue'
```

Type your props

```
interface Props {
  limit: number,
  alertMessageOnLimit?: string
}

const props = withDefaults(defineProps<Props>(), {
  alertMessageOnLimit: 'can not go any higher'
})
```

Type your nullable ref

```
const count = ref<number | null>(null)

onMounted(() => {
  fetchCount((initialCount) => {
    count.value = initialCount
  })
})
```

Type your function parameter

```
function addCount(num: number) {
  if (count.value !== null) {
    if (count.value >= props.limit) {
      alert(props.alertMessageOnLimit)
    }
    else {
      count.value += num
    }
  }
}
</script>


<template>
  <p>{{ count }}</p>
  <ControlBar
    @add-count="addCount"
    @reset-count="count = 0"
  ></ControlBar>
</template>
```

TYPE-SAFE