provide and inject in Vue3

Imagine a grandparent wants to give a gift (like money or advice) to their grandchild, but not through the parent.

In Vue, normally data is passed like this:

```
Grandparent → Parent → Child → Grandchild
```

But with provide and inject, Vue lets you do this:

```
Grandparent (provide) → Grandchild (inject)
```

1. provide()

The provide() function is used inside a component to make a value available to all its descendant components, no matter how deep they are in the component tree.

```
provide(key, value)
```

key is a string or symbol to identify the value

value is actual data you want to share (string, number, object, function, ref, etc.)

2. inject() The inject() function is used inside a descendant component to access a value that was provided() by an ancestor.

```
const value = inject(key, defaultValue)
```

key should be same key used in provide()

defaultValue (optional), fallback if no value was provided

Example : To undertans syntax

```
// In Parent.vue
provide('color', 'blue') // "I’m giving the color 'blue' to my family"

// In Child.vue
const color = inject('color') // "Hey, is there a color someone provided for me?"
```

Example : Basic app

```
<!-- Parent.vue : provide() is added here-->
<script setup>
import { provide } from 'vue'

provide('message', 'Hello from Parent!')
</script>

<template>
  <Child />
</template>

<!-- Child.vue : inject() in the descendant -->
<script setup>
import { inject } from 'vue'

const message = inject('message')
</script>

<template>
  <p>{{ message }}</p> <!-- Output: Hello from Parent! -->
</template>
```

Example : Reactive value

count is reactive here, if you update it in Provider, Consumer reflects it

```
<!-- Provider.vue -->
<script setup>
import { provide, ref } from 'vue'

const count = ref(0)
provide('count', count)
</script>

<template>
  <Child />
</template>

<!-- Consumer.vue -->
<script setup>
import { inject } from 'vue'

const count = inject('count')
</script>

<template>
  <div>Count: {{ count }}</div>
</template>
```