setUp Hook

Setup

`setup()` 함수(hook)는 Composition API 사용을 위한 진입점 역할을 합니다.

그리고 `setup` 함수는 컴포넌트 인스턴스가 생성되기 전에 실행됩니다.

기본 사용

반응형 API(Reactivity API)를 사용하여 반응형 상태를 선언하고 `setup()`에서 객체를 반환하여 `<template>`에 노출할 수 있습니다. 반환된 객체의 속성은 구성 요소 인스턴스에서도 사용할 수 있습니다(다른 옵션이 사용되는 경우):

```
<template>
  <button @click="increment">{{ count }}</button>
</template>

<script>
import { ref } from 'vue'

export default {
  setup() {
    //count 상태 생성
    const count = ref(0)

    //메서드 생성
    const increment = () => {
       count.value++;
    }

    // 템플릿 및 기타 Options API hook에 노출
    return {
      count,
      increment,
    }
  },
  //셋업함수에서 반환된 속성은 컴포넌트 인스턴스에서도 사용할 수 있습니다
  //(mounted와 같은 Options API는 잘 사용하지 않을 것이므로 참고만 하면 됨)
```

```
  mounted() {
    //this키워드로 셋업에서 생성한 count로 접근
   console.log(this.count) // 0
  }
}
</script>
```

Props 접근

`setup` 함수의 첫 번째 매개변수는 `props` 입니다. `props`는 **반응형** 객체입니다.

```
<template>
  <button @click="increment">{{ count }}</button>
</template>

<script>
  ...
  export default {
    props: {
      title: {
        //title: String
        type: String, //데이터의 타입 설정
        default: '페이지 제목' //다른 컴포넌트에서 전달받은 값이 없으므로 default값
설정
      },
    },
    setup(props) {
      console.log(props.title)
        ...
    }
  }
</script>
```

toRef, toRefs

만약 `props`의 반응성을 유지하면서 구조 분해 할당을 해야 한다면(예: 외부 함수에 prop을 전달해야 하는 경우) `toRefs()` 및 `toRef()` 유틸리티 API를 사용하여 이를 수행할 수 있습니다.

```vue
<template>
  <button @click="increment">{{ count }}</button>
</template>

<script>
import { ref, toRefs, toRef } from 'vue';

export default {
  props: {
    title: {
      type: String,
      default: '페이지 제목'
    },
  },
  setup(props) {
    //console.log(props.title)
    // toRefs로 props를 참조 객체로 바꾸고 구조분해할당
    const { title } = toRefs(props)
    console.log(title.value)

    // 또는 toRef메서드로 `props`의 단일 속성을 참조로 전환합니다.
    const title = toRef(props, 'title')
    console.log(title.value)

    ...
</script>
```