setup에서 라이프사이클 훅 & Template refs

Mounting

DOM에 컴포넌트를 삽입하는 단계이다. `onBeforeMount`와 `onMounted`가 있다.

- 서버렌더링에서 지원되지 않는다
- 초기 렌더링 직전에 돔을 변경하고자 한다면 이 단계에서 활용할 수 있다

onBeforeMount

컴포넌트가 마운트되기 직전에 호출됩니다.

- 대부분의 경우 사용을 권장하지 않는다

onMounted

컴포넌트가 마운트된 후에 호출됩니다. DOM에 접근할 수 있습니다.

- 모든 자식 컴포넌트가 마운트되었음을 의미합니다.
- 자체 DOM 트리가 생성되어 상위 컴포넌트에 삽입되었음을 의미합니다.

Updating

반응형 상태 변경으로 컴포넌트의 DOM 트리가 업데이트된 후 호출될 콜백을 등록합니다.

- 디버깅이나 프로파일링 등을 위해 컴포넌트 재 렌더링 시점을 알고 싶을 때 사용하면 된다.

onBeforeUpdate

반응형 상태 변경으로 컴포넌트의 DOM 트리를 업데이트하기 직전에 호출될 콜백을 등록합니다.

컴포넌트에서 사용되는 반응형 상태 값이 변해서, DOM에도 그 변화를 적용시켜야 할 때가 있습니다. 이 때, 변화 직전에 호출되는 것이 바로 onBeforeUpdate 훅입니다.

onUpdated

반응 상태 변경으로 인해 컴포넌트가 DOM 트리를 업데이트한 후에 호출됩니다.

상위 컴포넌트의 `onUpdated` 훅은 하위 컴포넌트의 훅 이후에 호출됩니다. (`Child` → `Parent`)

이 훅은 다른 상태 변경으로 인해 발생할 수 있는 컴포넌트의 DOM 업데이트 후에 호출됩니다. 특정 상태 변경 후에 업데이트된 DOM에 액세스해야 하는 경우 대신 `nextTick()` 을 사용하십시오.

> **WARNING**
>
> `onUpdated` 훅에서 컴포넌트 상태를 변경하지 마십시오. 그러면 무한 업데이트 루프가 발생할 수 있습니다!

Destruction

해체(소멸)단계 이며 `onBeforeUnmount` 와 `onUnmounted` 가 있습니다.

onBeforeUnmount

컴포넌트가 마운트 해제되기 직전에 호출됩니다.

onUnmounted

컴포넌트가 마운트 해제된 후 호출됩니다.

Template refs

`ref` 특수 속성을 통해 마운트된 DOM 요소 또는 자식 컴포넌트에 대한 참조를 얻을 수 있습니다.

Refs 접근하기

> 💡 Composition API로 참조를 얻으려면 동일한 이름의 참조를 선언해야 합니다.

```
<template>
  <input ref="input" type="text" />
    <div>{{ input }}</div>
    <!--<template> 안에서 input으로 Refs참조에 접근하려는 경우 렌더링되기 전에는 참조가 null일 수 있습니다.-->
    <!--<template> 안에서 $refs 내장 객체로 Refs 참조에 접근할 수 있습니다.-->
  <div>{{ $refs.input }}</div>
    <div>{{ input === $refs.input }}</div>
    <!--
        // ref속성에 문자열 키 대신 함수를 바인딩할 수도 있습니다.
        <input :ref="(el) => { /* assign el to a property or ref */ }">
    -->
</template>

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

export default {
  components: {},
  setup() {
    const input = ref(null);

        //컴포넌트가 마운트된 후에 접근할 수 있습니다.
    onMounted(() => {
      input.value.value = 'Hello World!';
      input.value.focus();
    });
    return {
      input,
    };
  },
};
</script>
```

v-for 내부 참조

> v3.2.25 이상에서 동작합니다.

`v-for` 내부에서 `ref`가 사용될 때 `ref`는 마운트 후 요소 배열로 채워집니다.

```
<template>
  <ul>
    <li v-for="item in list" ref="itemRefs">
      {{ item }}
    </li>
  </ul>
</template>

<script>
import { ref, onMounted } from 'vue'

export default {
  setup() {
    const list = ref([1, 2, 3])

    const itemRefs = ref([])

    onMounted(() => console.log(itemRefs.value))

    return {
      list,
      itemRefs
    }
  }
}
</script>
```

컴포넌트 Refs

`ref`로 자식 컴포넌트의 모든 속성과 메서드로 접근할 수 있습니다.

Child.vue

```vue
<template>
  <div>Child Component</div>
</template>

<script>
import { ref } from 'vue';

export default {
  setup() {
    const message = ref('Hello Child!');
    const sayHello = () => {
      alert(message.value);
    };
    return {
      message,
      sayHello,
    };
  },
};
</script>
```

Parent.vue

```vue
<template>
  <button @click="child.sayHello()">child.sayHello()</button>
  <Child ref="child"></Child>
</template>

<script>
import { onMounted, ref } from 'vue';
import Child from './Child.vue';

export default {
  components: { Child },
  setup() {
    const child = ref(null);
    onMounted(() => {
```

```
    console.log(child.value.message);
  });
  return { child };
 },
};
</script>
```

$parent = 상위 컴포넌트 참조

Parent.vue

```
...

export default {
 components: { Child },
 setup() {
  const child = ref(null);
  const message = ref('메시지')
  onMounted(() => {
   console.log(child.value.message);
  });
  return { child, message };
 },
};
```

Child.vue

```
<template>
 <div>Child Component</div>
 <!--자식 컴포넌트에서 상위 컴포넌트 참조하기 위해서는 $parent 내장객체를 사용할 수 있습니다.-->
 <div>{{ $parent.message }}</div>
</template>
...
```

- 컴포넌트 Refs를 사용하면 부모/자식 컴포넌트간 의존도가 생기기 때문에 반드시 필요한 경우에만 사용해야 합니다
- 일반적으로 ref보다 표준 props를 사용합니다