Day2 Vue directives

자료	Vue
구분	Vue
과목	

저번 시간에 사용자의 입력 값을 실시간으로 저장하기 위해서 `<input>` 태그를 사용한 `v-model` 을 학습하고 `v-on` 을 가지고 HTML에서 이벤트도 받아 보았다.
(기억이 안나시는 분은 반드시 어제 PDF를 확인해 주세요)

오늘은 `v-bind` `v-if` 그리고 `v-for` 에 대해서 학습해보자.

5. `:` (`v-bind`)

v-bind는 태그의 속성을 state(상태)로 지정 할 때 사용한다.

예를 살펴보자.

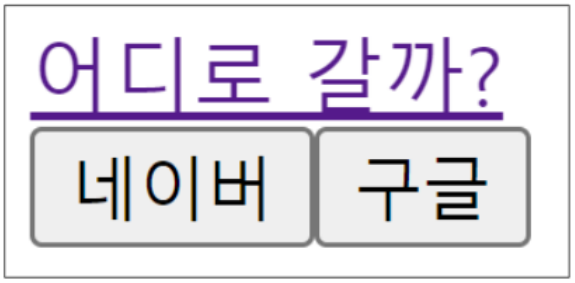
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id='asdf'>
    <a v-bind:href="url" target="_blank">어디로 갈까?</a>
    <div>
      <button v-on:click="setURLtoNaver">네이버</button>
      <button v-on:click="setURLtoGoogle">구글</button>
    </div>
  </div>

  <script>
    const { createApp, ref } = Vue;

    const app = createApp({
      setup() {
        const url = ref("");

        function setURLtoNaver() {
          url.value = "https://www.naver.com";
        }

        function setURLtoGoogle() {
          url.value = "https://www.google.com";
        }
```

```
      return {
        url,
        setURLtoNaver,
        setURLtoGoogle
      };
    }
  });

  app.mount('#asdf');
 </script>
</body>

</html>
```

라이브 서버를 킨 다음 브라우저를 통해서 확인해보자.
구글 또는 네이버 버튼을 클릭 후, “어디로 갈까?” 를 클릭 해보자.

소스코드를 살펴보면, 현재 `<a>` 태그를 보면, `v-bind:href="url"` 이 달려 있는 것을 볼 수 있다.
원래라면 `href="https://naver.com"` 처럼 고정된 URL 이 들어가지만, 우리는 상황에 따라서
`href` 속성 값을 다르게 만들고 싶었다.

이 때 `v-bind` 를 사용해 `href` 는 `url` 이라고 정의한 state 에 묶이도록(bind) 하고,
현재 `url` state 는 빈 문자열 `""` 로 지정되어있다.
버튼을 누르면, 해당 `url` 의 값은 “네이버”가 될 수도, “구글”이 될 수도 있다.



이처럼, `v-bind` 는 태그의 속성을 state 와 묶을 때 사용한다.

- `v-bind` 와 `v-model` 은 처음 배울 때 헷갈리는 개념이다.
  - `v-bind` : 태그의 속성을 state 와 묶을 때 사용한다.
  - `v-model` : `<input>` 태그에 사용한다. 이렇게 구분이 가능하다.

  `v-bind` 를 `<input>` 에 사용하는 실수가 흔하게 발생하니 항상 유의하자.

`v-bind:` 의 축약은 `:` 이다. 즉, 다음과 같이 변경 가능하다.

```
<a v-bind:href="url">어디로 갈까?</a>

// 위의 코드를 아래와 같이 변경 가능하다.
```

```
<a :href="url">어디로 갈까?</a>
```

`v-on` 과 마찬가지로, 프로젝트 전체에서 `v-on` 이든 `v-bind` 든, 축약을 쓰려면 축약만 사용하고,
만약에 풀네임을 사용 하거든, 풀네임만 사용하자. (축약어랑 풀네임이랑 섞어 사용하지 말자)
그리고 위에서 작성한 소스코드를 조금 더 살펴보자.

`target="_blank"` 속성이 있다. 이 속성은 링크를 클릭했을 때 새로운 탭이나 창에서 해당 URL을 열도록 지정하는 속성이다.
만약에 새 탭에서 구글 또는 네이버 창이 열리는 것이 싫다면 `target="_blank"` 속성을 지운 후,
클릭해보자. 그러면 현재 탭에서 구글 또는 네이버로 이동을 할 것이다.

[<script setup> 옵션을 사용해서 작성해 놓은 아래 코드는 참고용으로 봐주세요]

```
<script setup>
import { ref } from "vue";

const url = ref("");

function setURLtoNaver() {
 url.value = "https://naver.com";
}
function setURLtoGoogle() {
 url.value = "https://google.com";
}
</script>

<template>
 <a v-bind:href="url">어디로 갈까?</a>
 <div>
  <button v-on:click="setURLtoNaver">네이버</button>
  <button v-on:click="setURLtoGoogle">구글</button>
 </div>
</template>
```

자 이번에는 클래스를 동적 바인딩 해보자.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Document</title>
 <style>
  .active-title {
   color: green
  }
  .inactive-title {
   color: gray
  }
 </style>
```

```
</head>
<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id="app">
    <h2 :class="isActive ? 'active-title' : 'inactive-title'">타이틀</h2>
    <button @click="isActive = !isActive">토글</button>
  </div>

  <script>
    const { createApp, ref } = Vue

    createApp({
      setup() {
        const isActive = ref(false)
        return { isActive }
      }
    }).mount('#app')
  </script>
</body>
</html>
```

- active가 true면 초록색(.active-title), false면 회색(.inactive-title) 클래스를 적용한다.
- 버튼을 눌러 active 값을 반전시키면 클래스도 바뀐다.

Toggle을 구현하려면 당연히 함수를 연결 시켜서 구현해야 할 것 같지만, 간단하게 state 만 바꿀 경우에는 위의 예제와 같이 함수 없는 Toggle을 구현할 수도 있다.

- 페이지가 로드될 때 `isActive`의 초기값은 `false`이다.
- 버튼을 클릭하면 `isActive`의 값이 토글되어 `true` 또는 `false`로 변경된다.
  - isActive값이 true라면 false값으로 바뀔 것이며
  - isActive 값이 false라면 true로 값이 바뀔 것이다.
- 결과적으로 `isActive`의 값에 따라 class 속성의 값이 'active-title' 또는 'inactive-title' 가 되어 초록색 또는 회색이 나오는 것이다.

위 코드를 다르게 표현할 수도 있다.

삼항 연산자 대신 객체를 바인딩 하는 방법도 있다.

```
<!-- <h2 :class="isActive ? 'active-title' : 'inactive-title'">타이틀</h2> -->

위 코드 대신
아래 코드를 적용

<h2 :class="{ 'active-title': isActive, 'inactive-title': !isActive }">타이틀</h2>
<button @click="isActive = !isActive">토글</button>
```

`v-bind:class="{ '클래스 이름': 조건 }"` 을 사용함으로써

조건이 참이면 해당 클래스를 적용하고 거짓이면 적용 시키지 않는 방식이다.

이번에는 스타일을 바인딩 해보자.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    .active-title {
      color: green
    }
    .inactive-title {
      color: gray
    }
  </style>
</head>
<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id="app">
    <h2 :class="isActive ? 'active-title' : 'inactive-title'">타이틀</h2>
    <button @click="isActive = !isActive">토글</button>

    <p :style="{fontSize:(largeText?'24px':'14px'),color:'blue'}">
      스타일바인딩 예시임
    </p>
    <button @click="largeText=!largeText">크기변경</button>

  </div>

  <script>
    const { createApp, ref } = Vue

    createApp({
      setup() {

        const isActive = ref(false)
        const largeText=ref(true)

        return { isActive,largeText }

      }
    }).mount('#app')
  </script>
</body>
</html>
```

- largeText가 true면 큰 글자(24px), false면 작은 글자(14px)를 적용한다.
- 객체 형태로 여러 스타일 속성을 바인딩할 수 있다.
- 즉, 지금 한 것들을 객체 바인딩이라고 하는데 이는

  `v-bind:class="{ '클래스 이름': 조건 }"` 또는 `v-bind:style="{ '스타일 이름': 조건 }"` 형태로, 조건에 맞춰 클래스를 추가하거나 제거할 수 있다는 의미다.

6. `v-if` , `v-else-if` , `v-else`

`v-if` 는 해당 state 의 Boolean 값을 받은 후, 태그를 보여줄지 말지 결정할 때 사용한다.

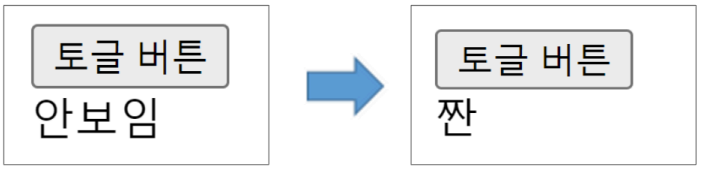
`v-if` 단독으로 사용 가능하며, `v-else-if` , `v-else` 와 혼용해서 사용하는 것도 가능하다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id='asdf'>
    <button @click="isActive = !isActive">토글 버튼</button>
    <div v-if="isActive">짠</div>
    <div v-else>안보임</div>
  </div>
  <script>

    const { createApp, ref } = Vue

    app = createApp({
      setup() {

        const isActive = ref(false)

        return {
          isActive
        }
      }
    })

    app.mount('#asdf')
  </script>
</body>

</html>
```

[<script setup> 옵션은 참고용으로만 봐주세요]

```
<script setup>
import { ref } from "vue";

const isActive = ref(false);
</script>

<template>
```

```
  <button @click="isActive = !isActive">토글 버튼</button>
  <div v-if="isActive">짠</div>
  <div v-else>안보임</div>
</template>
```



여기서, `@click` 구문을 주의해서 보자.

Toggle을 구현하려면 당연히 함수를 연결 시켜서 구현해야 할 것 같지만,

간단하게 state 만 바꿀 경우에는

위의 예제와 같이 함수 없는 Toggle을 구현할 수도 있다.

- 페이지가 로드될 때 `isActive`의 초기값은 `false`이다.
- 버튼을 클릭하면 `isActive`의 값이 토글되어 `true` 또는 `false`로 변경된다.
  - isActive값이 true라면 false값으로 바뀔 것이며
  - isActive 값이 false라면 true로 값이 바뀔 것이다.
- 결과적으로 `isActive`의 값에 따라 "짠" 또는 "안보임"이라는 텍스트가 화면에 표시가 되는 것이다.

7. v-for

태그의 반복문이다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id='asdf'>
    <ul>
      <li v-for="menu in menus" :key="menu.id">{{ menu.name }}</li>
    </ul>
  </div>

  <script>

    const { createApp, ref } = Vue

    app = createApp({
      setup() {
```

```
      const menus = ref([
        {
          id: 1,
          name: "짜장면",
        },
        {
          id: 2,
          name: "짬뽕",
        },
        {
          id: 3,
          name: "탕수육",
        },
      ]);

      return {
        menus
      }
    }
  })

  app.mount('#asdf')
 </script>
</body>

</html>
```

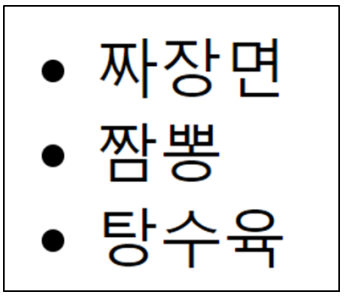
[<script setup> 옵션은 참고용으로만 봐주세요]

```
<script setup>
import { ref } from "vue";

const menus = ref([
 {
  id: 1,
  name: "짜장면",
 },
 {
  id: 2,
  name: "짬뽕",
 },
 {
  id: 3,
  name: "탕수육",
 },
]);
</script>

<template>
 <ul>
  <li v-for="menu in menus" :key="menu.id">{{ menu.name }}</li>
```

```
  </ul>
</template>
```

`menus` 의 각 요소를 `menu` 로 받고, `key` 를 `menu.id` 로 지정했다.

그리고 `menu.name` 을 출력한다.



총 세 개의 태그가 찍힌 것을 확인할 수 있다.

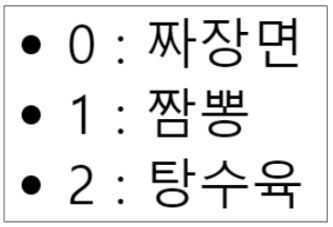
- `v-for` 를 사용할 때에는 `key` 를 반드시 지정을 해줘야 하며, `key` 는 중복되지 않는 값이어야만 한다.
- 만약 `key` 를 `menu.name` 로 지정한다면 매우 부적절 하다고 할 수도 있다. 그 이유는 메뉴 이름이 중복될 수 있기 때문이다.
- `:key` 속성은 Vue가 각 항목을 고유하게 식별하는 데 사용된다.

  여기서는 `menu.id` 를 사용하여 각 메뉴 항목에 고유한 키를 부여한 것이다.
- Vue는 리스트의 항목을 업데이트할 때, 어떤 항목이 변경 되었는지 추적하기 위해

  `key` 를 사용한다. 이는 통해 불필요한 DOM 요소를 재구성하지 않고, 최적화된 방식으로 변경된 항목만 업데이트를 하기 위함이다.

만약에 배열의 인덱스도 같이 출력하려면 아래와 같이 for문을 수정하면 된다.

```
<template>
  <ul>
    <li v-for="(menu, idx) in menus" :key="menu.id">{{ idx }} : {{ menu.name }}</li>
  </ul>
</template>
```



아래 코드를 살펴보자.

만약에 `id` 역할을 할 만한 요소가 없으면 `key` 를 지정하기 어려울 수 있다.

이 경우에는 배열의 인덱스를 `key` 로 활용할 수 있을 것이다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id='asdf'>
    <ul>
      <li v-for="(menu, idx) in menus" :key="idx">{{ idx }} : {{ menu }}</li>
    </ul>
  </div>
  <script>

    const { createApp, ref } = Vue

    app = createApp({
      setup() {

        const menus = ref(["짜장면","짬뽕","탕수육"]);

        return {
          menus
        }
      }
    })

    app.mount('#asdf')
  </script>
</body>

</html>
```

하지만 Vue 공식문서에 따르면 이 방법은 권장되지 않는다. 그 이유는

리스트의 항목이 추가 되거나 삭제가 될 경우, 인덱스가 바뀌게 되기 때문이다.

배열의 인덱스가 바꾸면 불필요한 DOM 재랜더링이 되거나 상태가 꼬이게 되는 경우가 많기 때문이다.

8. v-htlm

`v-html`은 HTML 문자열을 출력하는 데 사용된다.

HTML 태그가 포함된 문자열을 렌더링할 때 사용한다고 보면 된다. 아래 코드를 살펴보자.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

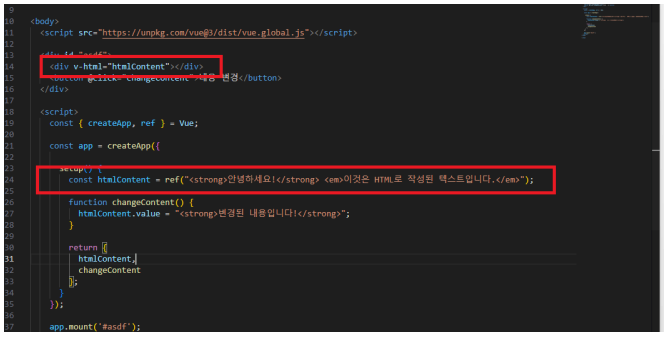
```
<div id="asdf">
  <div v-html="htmlContent"></div>
  <button @click="changeContent">내용 변경</button>
</div>

<script>
  const { createApp, ref } = Vue;

  const app = createApp({

    setup() {
      const htmlContent = ref("<strong>안녕하세요!</strong> <em>이것은 HTML로 작성된 텍스트입니다.</em>");

      function changeContent() {
        htmlContent.value = "<strong>변경된 내용입니다!</strong>";
      }

      return {
        htmlContent,
        changeContent
      };
    }
  });

  app.mount('#asdf');
 </script>
</body>

</html>
```

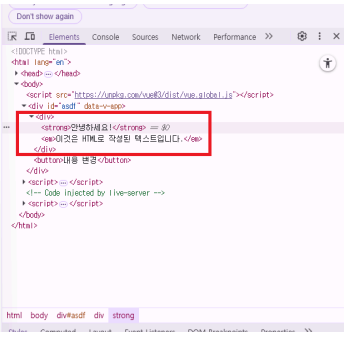
안녕하세요! *이것은 HTML로 작성된 텍스트입니다.*

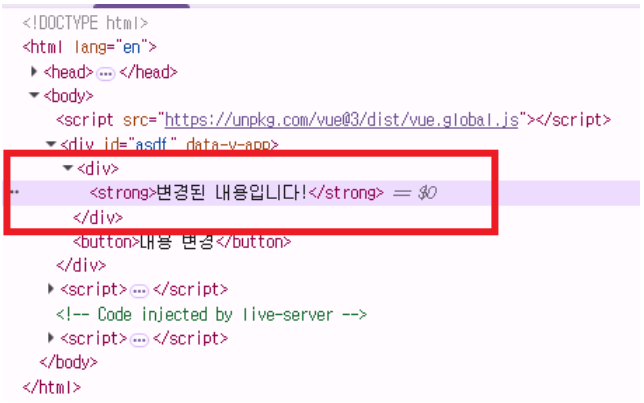
내용 변경

변경된 내용입니다!

내용 변경







- `v-html="htmlContent"`를 보면 `htmlContent` 변수의 값은 HTML 태그가 포함된 문자열이다.

  이 문자열이 HTML 형식으로 렌더링이 되는 것이다.

  즉, HTML이 실재 dom으로 삽입을 할 때 v-html을 사용한다.

- 하지만 `v-html`을 사용할 때는 보안상 매우 조심해야 한다. HTML 콘텐츠를 외부에서 동적으로 삽입할 때 XSS(교차 사이트 스크립팅, Cross-Site Scripting) 공격에 노출될 수 있기 때문이다.
  - [참고] XSS 공격이란?
    - XSS 공격은 악의적인 사용자가 입력한 JavaScript 코드가 페이지에 삽입되어 실행되는 보안 취약점을 말한다.

9. v-text

`v-text`는 HTML 태그를 포함하지 않고, 텍스트만 보여줄 때 사용한다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id="asdf">
    <div v-text="textContent"></div>
    <button @click="changeText">텍스트 변경</button>
  </div>

  <script>
    const { createApp, ref } = Vue;

    const app = createApp({
      setup() {
        const textContent = ref("안녕하세요. 변경 전 텍스트 입니다.!");
```

```
    function changeText() {
      textContent.value = "변경된 텍스트입니다.";
    }

    return {
      textContent,
      changeText
    };
  }
});

app.mount('#asdf');
</script>
</body>

</html>
```

`v-text="textContent"`를 보면 `textContent`의 값이 텍스트로 렌더링되며, HTML 태그는 무시된다.

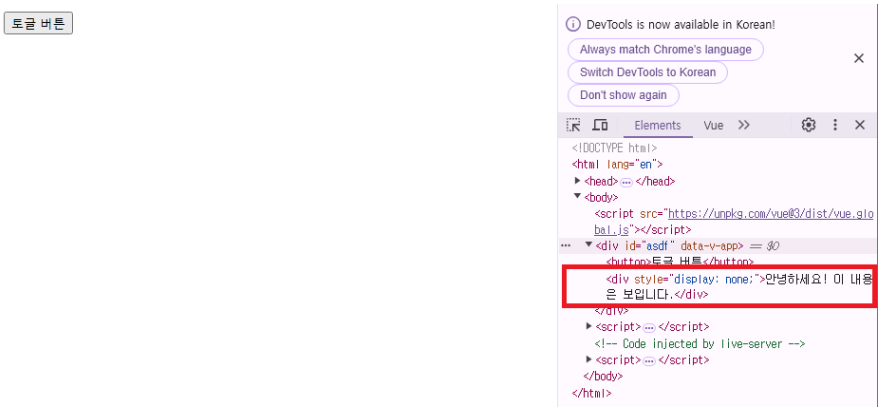
10. v-show

`v-show`는 조건에 따라 요소의 표시 여부를 제어한다.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id="asdf">
    <button @click="isVisible = !isVisible">토글 버튼</button>
    <div v-show="isVisible">안녕하세요. 이 내용이 보이나요?</div>
  </div>

  <script>
    const { createApp, ref } = Vue;

    const app = createApp({
      setup() {
        const isVisible = ref(false);

        return {
```

```
        isVisible
      };
    }
  });

  app.mount('#asdf');
</script>
</body>

</html>
```

토글 버튼
안녕하세요! 이 내용은 보입니다.



토글 버튼을 누르면 `isVisible`의 값이 토글되어 내용이 보이거나 숨겨진다. 이는

`v-show="isVisible"`을 통해서 `isVisible`이 `true`일 때 해당 요소가 보이며,

`false`일 경우에는 숨겨진다.

v-show를 사용 할 경우 요소는 DOM에 남아 있지만, 스타일로 인해 보이지 않는 것이다.

v-show 는 v-if와 상당히 비슷하다. 하지만 분명한 차이점이 있다.

v-if와 달리 v-show는 화면에 랜더링이 되어 DOM에 남아 있지만 화면에 보여지게 할지 말지 정할 때 사용한다. v-show의 내부 원리는 일단 랜더링을 하고 display 속성값을 none으로 할지 말지를 결정하기 때문에 초기 랜더링 비용이 높은 편이다. 하지만 토글비용은 적기 때문에 상황에 따라 특정 게시물이 자주 보였다가 안보였다를 자주 반복 할 때에는 v-if 보다는 v-show를 사용한다.

[결론]

- `v-if`: 조건에 따라 아예 DOM에서 요소를 생성/제거 (무거움, 초기 렌더링 비용 낮음, 토글 비용 높음)
- `v-show`: DOM에는 존재, `display: none` 으로 보였다 안 보였다 처리 (초기 렌더링 비용 높음, 토글 비용 낮음)
- 예시:
  - 자주 보였다가 안 보였다가 반복되면 → `v-show`
  - 드물게 조건이 바뀐다면 → `v-if`

추가로 많이 질문 하는 것들이다.

양방향 바인딩? 단방향 바인딩? 무슨 말이지?

우선, 양방향 바인딩이라는 말은 데이터를 화면(UI)과 JS코드에서의 상태(state)값을 양방향으로 자동 동기화 한다는 뜻이다.

단방향 바인딩 vs 양방향 바인딩

구분	단방향 바인딩 (One-Way)	양방향 바인딩 (Two-Way)
예시 디렉티브	`v-bind` 또는 `:`	`v-model`
데이터 흐름	JavaScript → DOM (화면에 값 표시만)	JavaScript ↔ DOM (입력값과 상태가 동기화)
특징	사용자가 값을 바꿔도 상태에 반영 X	사용자가 값을 바꾸면 상태에도 반영됨
예시 코드	`<input :value="message">`	`<input v-model="message">`

단방향 바인딩 그리고 양방향바인딩 예시

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <div id="app">
    <p>단방향바인딩: {{ message }}</p>

    <input v-model="msg2" />
    <p>양방향바인딩: {{ msg2 }}</p>
  </div>

  <script>
    const { createApp, ref } = Vue;

    createApp({
      setup() {
        const message = ref('단방향 바인딩으로 html에서 ref객체의 값을 수정할 수가 없는경우를 말한다.');
        const msg2 = ref('나를 수정해봐')
        return { message,msg2 };
      }
    }).mount('#app');
  </script>
</body>
</html>
```

<끝>