Day2

자료	Vue
구분	Vue

저번 시간에 사용자의 입력값을 실시간으로 저장하기 위해서 `v-model` 와 `<input>` 태그를 사용 하였고 `v-on` 을 가지고 HTML에서 이벤트도 받아 보았다.

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

오늘은 vue 템플릿 문법 중에 v-bind 부터 살펴 보자.

v-bind는 태그의 속성을 state로 지정 시 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'>
  <a v-bind:href="url">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

    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>
```

[<script setup> 옵션은 참고용으로만 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>
```

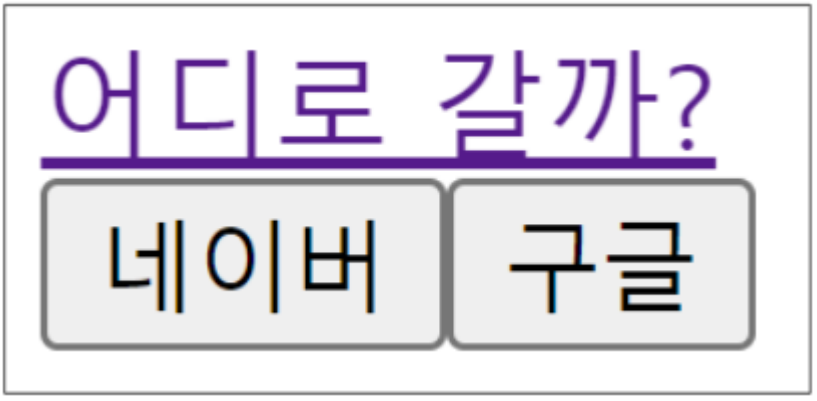
현재 `<a>` 태그를 보면, `v-bind:href="url"` 이 달려있는것을 볼 수 있다.

원래라면 `href="https://naver.com"` 처럼 고정된 URL 이 들어가지만, 우리는 상황에 따라서 `href` 속성 값을 다르게 만들고 싶다.

이 때 `v-bind` 를 사용해 `href` 는 `url` state 에 묶이도록(bind) 하고,

현재 `url` state 는 빈 문자열 `""` 로 지정되어있다.

버튼을 누르면, 해당 `url` 은 “네이버”가 될수도, “구글”이 될수도 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 애트리뷰트를 state 와 묶을 때
  - `v-model` : `<input>` 태그에 사용

  `v-bind` 를 `<input>` 에 사용하는 실수가 흔하니 유의하자.

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

```
<a v-bind:href="url">어디로 갈까?</a>

// 다음과 같이 변경 가능
<a :href="url">어디로 갈까?</a>
```

`v-on` 과 마찬가지로, 프로젝트 전체에서 `v-on` 이든 `v-bind` 든, 축약을 쓰려면 축약만, 풀네임을 쓸거면 풀네임만 쓴다.

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

해당 state 의 Boolean 값을 받은 후, 태그를 보여줄지 말지 결정한다.

`v-if` 단독으로 사용 가능하며, `v-else-if` , `v-else` 와 혼용 해서도 사용 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>
```

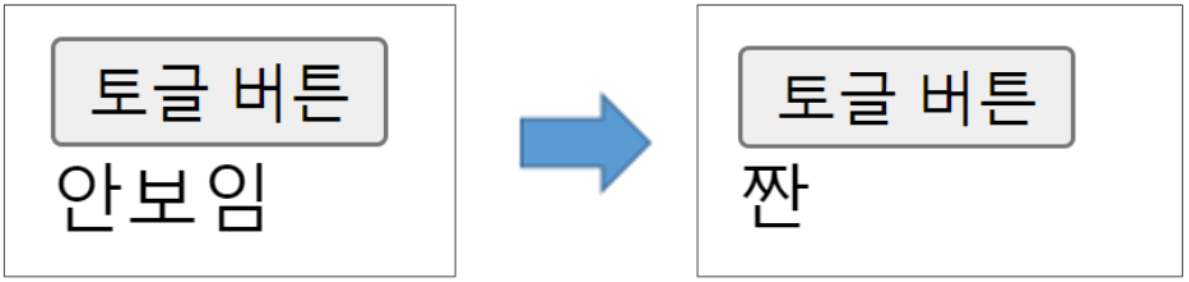
[<script setup> 옵션은 참고용으로만 봐주세요]

```
<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 경우 위와 같이 함수 없는 Toggle을 구현할 수 있다.

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]

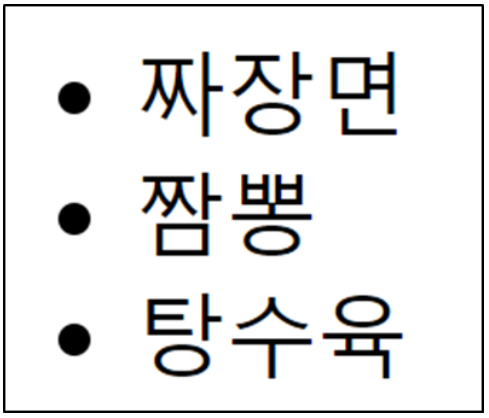
```
<script setup>
import { ref } from "vue";

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.

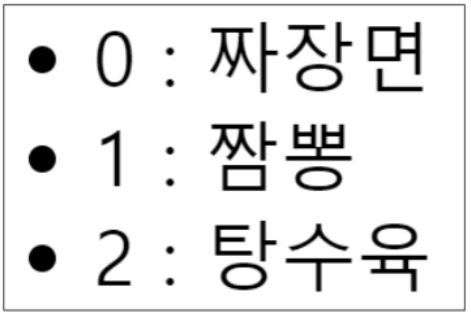


총 세 개의 태그가 찍힌 것을 확인할 수 있다.

- `key` 는 반드시 지정을 해줘야 하며, 중복되지 않는 값이어야만 한다. 만약 `key` 를 `menu.name` 으로 지정한다면 매우 부적절한데, 그 이유는 메뉴 이름이 중복될 수 있기 때문이다.

상황에 따라, 배열의 인덱스가 필요 할 수도 있다.

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



언제 인덱스를 받는 게 좋을까? 어떤 배열은 `id` 역할을 할 만한 요소가 없어 `key` 를 지정하기 어려울 수 있다. 이 경우, 배열의 인덱스를 `key` 로 활용할 수 있을 것이다.