8_26_2022_Web_vue.js

프레임 워크 -

- jQuery - 구형 웹브라우저에서 호환성을 고려하기 위해 많이 썼음(js기초가 되어있다면 쉽다)
- react js, angular js, vue js 등 - js 가 발전해서 사용하게 됨.

js로 했던 일

- 화면의 동작(ui의 동작)
- 이벤트 처리
- 입력받은 데이터 전송(ajax)
- css조작

사실 이런 동작들은 프레임 워크의 도움을 받지 않고 할 수도 있었다. 하지만 프로젝트를 할 때 js가 너무 커질 수도 있고 여러 사람이 한 프로젝트를 같이 할 수도 있다.

vue.js

- 클라이언트 웹브러우저에 외부 js를 로딩시켜서 로딩된 js를 활용해서 어떠한 동작을 하는 프레임 워크 html파일을 사용한다.
- “v-” 형식을 따른다.

```
<input type="text" v-model="msg"/>
<p>{{msg}}</p>
```

input 요소에 입력한 값을 msg 라는 모델명으로 관리하겠다
모델이 수정이 되면 모델 반영 된 곳에 자동 업데이트

같은 기능을 바닐라js로만 사용한다면

```
<div>
  <input type="text" id="inputMsg"/>
  <p id="result"></p>
</div>
<script>
  document.querySelector("#inputMsg").addEventListener("input", function(){
    document.querySelector("#result").innerText=this.value;
  });
</script>
```

vue js 로딩

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

new Vue({ });

- vue 객체를 생성하면서 object(구조 {key:value , key2:value})를 전달

```
new Vue({
  el:"#app",
  data:{
    fortune:"동쪽으로 가면 귀인을 만나요!",
    myName:"김구라",
    msg:"",
    nums:[10, 20, 30]
  },
  methods:{
    clicked:function(){
      //alert("오잉?");
      this.fortune="오후에도 vue 를 배우게 될거에요";
    },
    clicked2:function(){
```

```
      //this.myName="에이콘";
      this.nums=["김구라", "해골", "원숭이"];
    }
  }
});
```

el이라는 key값에는 String type이 있다.

data라는 key값에는 object type이 있다.

model들의 초기값이 들어있다

method라는 key값에는 object type이 있다.

특정 시점에 실행할 함수를 모아 놓을 수 있다.

```
<button v-on:click="clicked">눌러보셈</button>
<button v-on:click="clicked2">이름 바꾸기</button>
```

```
<div id="app">
new Vue({el:"#app",});
```

v-xxx=”value” 형식

v-for=”” for 반복문 형식

▼ /Vue/Step01_event.html

v-on:이벤트명=”method object에 있는 호출될 함수명

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step01_event.html</title>
</head>
<body>
    <h1>이벤트 테스트</h1>
    <div id="app">
        <!--
            v-on:이벤트명="methods object 에 있는 호출될 함수명"
        -->
        <button v-on:click="clicked">눌러보셈</button>
        <p>{{msg}}</p>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                msg:""
            },
            methods:{
                clicked:function(){
                    console.log("버튼을 눌렀네요?");
                    //data object 안에있는 msg 참조
                    this.msg="버튼을 눌렀네요?";
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step01_event.html

on:click 이벤트로 count 하기

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

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step01_event2.html</title>
</head>
<body>
    <h1>이벤트 테스트2</h1>
    <div id="app">
        <button v-on:click="increase">+</button>
        <button v-on:click="decrease">-</button>
        <p>{{count}}</p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
               count:0
            },
            methods:{
                increase:function(){
                    this.count++;
                },
                decrease:function(){
                    this.count--;
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step01_event.html

on:mouse 이벤트

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step01_event3.html</title>
    <style>
        .box{
            width: 500px;
            height: 500px;
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h1>이벤트 테스트3</h1>
    <div id="app">
        <div class="box" v-on:mousemove="moved">
            x : <strong>{{x}}</strong>
            <br>
            y : <strong>{{y}}</strong>
        </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>

        let app=new Vue({
            el:"#app",
            data:{
               x:0,
               y:0
            },
            methods:{
                //함수에는 이벤트 객체가 전달된다.
                moved:function(e){
                    console.log(e);
                    //이벤트가 일어난 곳의 좌표를 x, y 에 넣어준다.
                    this.x=e.offsetX;
                    this.y=e.offsetY;
                }
            }
        });
    </script>
</body>
</html>
```

```
이벤트 객체를 e로 지정해주고 전달해줌
```

▼ /Vue/Step01_event.html

on:click이벤트로 css 연동

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step01_event4.html</title>
    <style>
        .box{
            width: 100px;
            height: 100px;
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h1>이벤트 테스트4</h1>
    <div id="app">
        <div class="box" v-on:click="divClicked">div1</div>
        <div class="box" v-on:click="divClicked">div2</div>
        <div class="box" v-on:click="divClicked">div3</div>
        <div class="box" v-on:click="divClicked">div4</div>
        <div class="box" v-on:click="divClicked">div5</div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{

            },
            methods:{
                divClicked:function(e){
                    //클릭한 바로 그 div 의 innerText 를 "clicked!" 로 바꾸고 싶다면?
                    console.log(e);
                    // e.target 은 이벤트가 발생한 문서객체의 참조값이다.
                    e.target.innerText="clicked!";
                    e.target.style.backgroundColor="yellow";
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step01_event.html

on:submit 전송 기능

prevent 사용해 이벤트 막기

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step01_event5.html</title>
</head>
<body>
    <h1>이벤트 테스트5</h1>
    <div id="app">
        <form action="test.jsp" v-on:submit="onSubmit">
            <input type="text">
            <button type="submit">전송</button>
        </form>
        <br>
        <!-- .prevent 수식어를 이용해서 기본 이벤트 막기 -->
        <form action="test.jsp" v-on:submit.prevent="onSubmit2">
            <input type="text">
            <button type="submit">전송</button>
        </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

```
    <script>
        let app=new Vue({
            el:"#app",
            data:{

            },
            methods:{
               onSubmit:function(e){
                   //기본 이벤트를 막아서 폼이 제출되지 않도록 한다.
                   e.preventDefault();
               },
               onSubmit2:function(){

               }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step02_model.html

model 기본 사용법

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step02_model.html</title>
</head>
<body>
    <h1>v-model 사용하기</h1>
    <div id="app">
        <!-- input 요소의 value 를 msg 라는 모델명으로 사용하겠다 -->
        <input type="text" v-model="msg" >
        <p>{{msg}}</p>
        <!-- p 요소의 innerText 에 msg 라는 모델 안에 들어 있는 값을 넣기 -->
        <p v-text="msg"></p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
              msg:""
            },
            methods:{

            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step02_model2.html

checkbox에 medel을 지정하면 boolean 값으로 관리가 된다.
select에 model을 지정하면 선택한 option의 value가 model의 값으로 관리가 된다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step02_model2.html</title>
</head>
<body>
    <h1>v-model 사용하기2</h1>
    <div id="app">
        <input type="text" v-model="msg">
        <br>
        <input type="checkbox" v-model="isChecked">
        <br>
        <select v-model="lunch">
            <option value="">선택</option>
            <option value="ramen">라면</option>
            <option value="kimbap">김밥</option>
            <option value="dduk">떡복기</option>
```

```
        </select>
        <br>
        <textarea v-model="comment"></textarea>
        <br>
        <!--
            boolean 값으로 관리 하지 않고
            원하는 문자열 값으로 관리하기
        -->
        <input type="checkbox"
            v-model="isRun"
            true-value="yes"
            false-value="no">
        <p>
            msg : <strong>{{msg}}</strong>
            <br>
            isChecked : <strong>{{isChecked}}</strong>
            <br>
            lunch : <strong>{{lunch}}</strong>
            <br>
            comment : <strong>{{comment}}</strong>
            <br>
            isRun : <strong>{{isRun}}</strong>
        </p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                msg:"",
                isChecked:true,
                lunch:"",
                comment:"",
                isRun:"no"
            },
            methods:{

            }
        });
    </script>
</body>
</html>
```

Vue에서 class 관리하는 방법

▼ /Vue/Step03_class.html

참조값을 model로 지정해 상황에 따라 바꾸겠다(버튼)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step03_class.html</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN
</head>
<body>
    <h1>class 제어 하기</h1>
    <div id="app">
        <button class="btn btn-primary btn-lg">버튼</button>
        <br>
        <button class="btn"
            v-bind:class="{'btn-primary':isPrimary, 'btn-lg':isLg}">Vue 버튼</button>
        <br>
        <label>
            파란색 버튼 <input type="checkbox" v-model="isPrimary">
        </label>
        <label>
            큰 버튼 <input type="checkbox" v-model="isLg">
        </label>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                isPrimary:false,
                isLg:false
            },
            methods:{
```

```
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step03_class2.html

classObject 사용

종속된 모델의 값이 바뀌면 다시 호출되어서 연산된 값을 리턴

isPrimary, isLg둘이 false로 되어있는데 true로 바뀌게되면 다시 호출되어 연산된 값을 리턴

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step03_class2.html</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN
</head>
<body>
    <h1>class 제어 하기</h1>
    <div id="app">
        <button class="btn btn-primary btn-lg">버튼</button>
        <br>
        <button class="btn"
            v-bind:class="classObject">Vue 버튼</button>
        <br>
        <label>
            파란색 버튼 <input type="checkbox" v-model="isPrimary">
        </label>
        <label>
            큰 버튼 <input type="checkbox" v-model="isLg">
        </label>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                isPrimary:false,
                isLg:false
            },
            methods:{

            },
            //종속된 모델이 바뀌면 다시 호출되어서 연산된(computed) 값을 리턴하는 함수
            computed:{
                classObject:function(){
                    return {'btn-primary':this.isPrimary, 'btn-lg':this.isLg};
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step03_class3.html

배열에 추가할 class를 item으로 가짐

삼항연산자로 관리할 수 있다.

```
<button class="btn"
  v-bind:class="[ isPrimary ? 'btn-primary' : '', isLg ? 'btn-lg' : '']">
  Vue 버튼
</button>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Step03_class3.html</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN
</head>
<body>
    <h1>class 제어 하기</h1>
    <div id="app">
        <button class="btn btn-primary btn-lg">버튼</button>
        <br>
        <button class="btn"
            v-bind:class="[ isPrimary ? 'btn-primary' : '', isLg ? 'btn-lg' : '']">Vue 버튼</button>
        <br>
        <label>
            파란색 버튼 <input type="checkbox" v-model="isPrimary">
        </label>
        <label>
            큰 버튼 <input type="checkbox" v-model="isLg">
        </label>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                isPrimary:false,
                isLg:false
            },
            methods:{

            },
            //종속된 모델이 바뀌면 다시 호출되어서 연산된(computed) 값을 리턴하는 함수
            computed:{

            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step03_class4.html

원하는 클래스 입력해 추가할 수도 있음

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step03_class4.html</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN
</head>
<body>
    <h1>클래스 테스트</h1>
    <div id="app">
        <h2>{{greet}}</h2>
        <button class="btn btn-primary">버튼1</button>
        <br>
        <button v-bind:class="{'btn':true, 'btn-primary':true}">버튼2</button>
        <br>
        <button v-bind:class="['btn', 'btn-primary']">버튼3</button>
        <br>
        <button v-bind:class="classObject">버튼4</button>
        <br>
        <button v-bind:class="classArray">버튼5</button>
        <br>
        <button v-bind:class="{'btn':isBtn, 'btn-primary':isPrimary}">버튼6</button>
        <label>
            btn <input type="checkbox" v-model="isBtn">
        </label>
        <label>
            btn-primary <input type="checkbox" v-model="isPrimary">
        </label>
        <br>
        <button v-bind:class="arr">버튼7</button>
        <input type="text" v-model="inputClass"/>
        <button v-on:click="add2">추가</button>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
```

```
                greet:"좋은 아침 입니다.",
                classObject:{'btn':true, 'btn-primary':true},
                classArray:['btn', 'btn-primary'],
                isBtn:true,
                isPrimary:true,
                arr:[],
                inputClass:""
            },
            methods:{
                add:function(){
                    // arr 배열에 입력한 클래스명을 추가하기
                    this.arr.push(this.inputClass);
                },
                add2(){
                    this.arr.push(this.inputClass);
                }
            }
        });
    </script>
</body>
</html>
```

css조작하기

▼ /Vue/Step04_style.html

```
<p v-bind:style="{color:'red', fontSize:'20px'}">
object 속성은  camel case로 설정 fontSize      font-size 아님
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step04_style.html</title>
</head>
<body>
   <div class="container" id="app">
        <h1>vue js 에서 css 조작하기</h1>
        <p v-bind:style="{color:'red', fontSize:'20px'}">Lorem ipsum dolor sit amet consectetur adipisicing elit. Id, aperiam mini
        <p v-bind:style="{color:fontColor, fontSize: fontSize+'px'}">Lorem ipsum dolor sit amet consectetur adipisicing elit. Haru
        <p v-bind:style="pStyle">Lorem ipsum dolor, sit amet consectetur adipisicing elit. Labore unde voluptatem non ea ipsam ani
        <input type="text" placeholder="글자색 입력..." v-model="fontColor">
        <input type="text" placeholder="글자의 크기 입력..." v-model="fontSize">
    </div>
   <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
   <script>
       let app=new Vue({
           el:"#app",
           data:{
              fontColor:"black",
              fontSize:16
           },
           computed:{
                pStyle:function(){
                    return {color:this.fontColor, fontSize: this.fontSize+'px'};
                }
           }
       });
   </script>
</body>
</html>
```

if문

▼ /Vue/Step05_if.html

v-if는 문서 객체를 만들지 말지를 결정한다(추가,제거)

v-show는 문서 객체를 보일지 숨길지를 결정한다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step05_if.html</title>
    <style>
        .box{
            width: 100px;
            height: 100px;
            background-color: yellow;
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h1>v-if and v-show 사용하기</h1>
    <div id="app">
    <!--  if는 문서 객체를 만들지 말지를 결정한다(추가,제거) -->
        <div class="box" v-if="true">box1</div>
        <div class="box" v-if="false">box2</div>
        <div class="box" v-if="isMake">box3</div>
        isMake <input type="checkbox" v-model="isMake">
        <!-- v-show는 문서 객체를 보일지 숨길지를 결정한다. -->
        <div class="box" v-show="true">box4</div>
        <div class="box" v-show="false">box5</div>
        <div class="box" v-show="isShow">box6</div>
        isShow <input type="checkbox" v-model="isShow">
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                isMake:true,
                isShow:false
            }
        });
    </script>
</body>
</html>
```

반복문 for

▼ /Vue/Step06_for.html

배열이 모델일 수도 있음

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step06_for.html</title>
</head>
<body>
    <h1>v-for 사용하기</h1>
    <div id="app">
        <h2>목록출력</h2>
        <ul>
            <li v-for="item in [10,20,30]">{{item}}</li>
        </ul>
        <h2>목록출력2</h2>
        <ul>
            <li v-for="item in nums">{{item}}</li>
        </ul>
        <h2>목록출력3</h2>
        <ul>
            <li v-for="(item, index) in nums">{{item}} {{index}}</li>
        </ul>
        <h2>목록출력4</h2>
        <ul>
            <li v-for="item in members">
                이름 : {{item.name}}
                주소 : {{item.addr}}
            </li>
        </ul>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

```
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                nums:[10,20,30],
                members:[
                    {name:"김구라", addr:"노량진"},
                    {name:"해골", addr:"행신동"},
                    {name:"원숭이", addr:"동물원"}
                ]
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step06_for2.html

```
data:{
    inputMsg:"",
    msgs:[]
}
```

빈배열에 추가해 리스트 만들기

ref=”one”의 의미 레퍼런스를 부여하고 원하는 값을 얻어올 수 있다.

id=”one” 해서 document.querySelector하는 것과 비슷함
하지만 vue에서는 id를 부여해서 참조하는게 드문 일임
reference값을 부여해 참조되는 이름을 통해서 참조값을 익힐 필요가 있음.

💡 $refs →준비된 ref의 이름을 불러오겠다라는 약속된 문자열

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step06_for2.html</title>
</head>
<body>
    <h1>v-for 사용 예제</h1>
    <div id="app">
        <input type="text" v-model="inputMsg" ref="one">
        <button v-on:click="clicked">추가</button>
        <ul>
            <li v-for="(item, index) in msgs">{{item}} <button v-on:click="deleteItem(index)">x</button></li>
        </ul>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                inputMsg:"",
                msgs:[]
            },
            methods:{
                //clicked:function(){}
                clicked(){//추가 버튼을 클릭했을때 호출되는 함수
                    //배열에 아이템 추가하고
                    this.msgs.push(this.inputMsg);
                    //입력창 초기화
                    this.inputMsg="";
                    //입력창에 포커스 주기
                    this.$refs.one.focus();
                },
                deleteItem(i){
                   // i 는 삭제할 배열의 방번호 이다.
                   this.msgs.splice(i, 1);
                }
            }
        });
    </script>
</body>
</html>
```

Component

▼ /Vue/Step07_component.html

재사용 가능한 마크업을 작성해 놓고 component의 이름을 이용해서 원하는 곳에다가 넣을 수 있다. - 복잡한 코드를 먼저 component로 정의 해놓고 사용할 수 있다. jsp는 server에서 연동을 해놓았다고 생각한다면 component는 client단에서 연동을 해놓는 것.

어떤 값을 전달받을 준비도 해 놓을 수 있다.

```
"your-component":{
    template:"<div class='box'>{{greet}}</div>",
    props:["greet"]
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_component.html</title>
    <style>
        .box{
            width: 100px;
            height: 100px;
            background-color: yellow;
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h1>component 정의하고 사용하기</h1>
    <div id="app">
        <my-component></my-component>
        <my-component></my-component>
        <my-component></my-component>
        <input type="text" v-model="msg"/>
        <your-component v-bind:greet="'안녕하세요!'"></your-component>
        <your-component v-bind:greet="msg"></your-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
              msg:""
            },
            components:{
                "my-component":{
                    template:"<div class='box'> box </div>"
                },
                "your-component":{
                    template:"<div class='box'>{{greet}}</div>",
                    props:["greet"]
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step07_component2.html

여러줄의 컴포넌트를 사용하기 위해 백틱 사용

```
template:`
            <ul>
                <li v-for="item in friends">{{item}}</li>
            </ul>
        `,
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_component2.html</title>
</head>
<body>
    <h1>component 사용예제</h1>
    <div id="app">
        <h2>친구 목록 입니다.</h2>
        <friends-component v-bind:friends="friends"></friends-component>

        <h2>동물 친구 목록 입니다.</h2>
        <friends-component v-bind:friends="friends2"></friends-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            data:{
                friends:["김구라","해골","원숭이"],
                friends2:["강아지","고양이","두더지"]
            },
            components:{
                "friends-component":{
                    template:`
                        <ul>
                            <li v-for="item in friends">{{item}}</li>
                        </ul>
                    `,
                    props:["friends"]
                }
            }
        });
    </script>
</body>
</html>
```

자식 컴포넌트에서 사용할 것을 부모 컴포넌트에서 불러오기
부모가 가지고 있는 model을 자식에 전달 해줌

▼ /Vue/Step07_component3.html

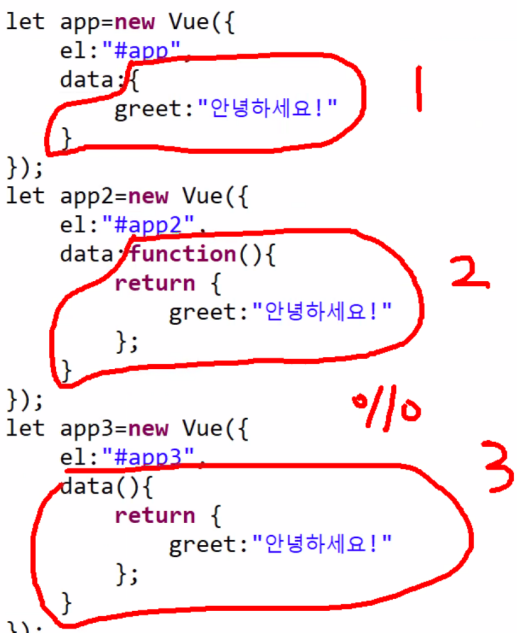
재사용 가능한 컴퓨넌트를 분리해 전역으로 정의하기
Vue.component("컴포넌트 이름", {컴포넌트 데이터});
결국 object type

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

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_component3.html</title>
</head>
<body>
    <h1>component 테스트</h1>
    <div id="app">
        <p> 오늘의 인사 : <strong>{{greet}}</strong></p>
        <my-component></my-component>
    </div>
    <div id="app2">
        <p> 오늘의 인사 : <strong>{{greet}}</strong></p>
        <my-component></my-component>
    </div>
    <div id="app3">
        <p> 오늘의 인사 : <strong>{{greet}}</strong></p>
        <my-component></my-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        /*
            재사용 가능한 컴포넌트를 전역으로 정의하기
            Vue.component("컴포넌트 이름", {컴포넌트 데이터});
        */
        Vue.component("my-component", {
            template:"<div>{{msg}}</div>",
            data(){
                return {
                    msg:"div 입니다."
                };
            }
        });

        let app=new Vue({
            el:"#app",
            data:{
                greet:"안녕하세요!"
            }
        });
        let app2=new Vue({
            el:"#app2",
            data:function(){
                return {
                    greet:"안녕하세요!"
                };
            }
        });
        let app3=new Vue({
            el:"#app3",
            data(){
                return {
                    greet:"안녕하세요!"
                };
            }
        });
    </script>
</body>
</html>
```

신문물

```
{data:function(){}}
//위와 아래는 같다.
{data(){}}
```



부모 component 에서는 세가지 방법 다 같은 결과
단 자식 component 에서는 1번모양이 불가능하다

▼ /Vue/Step07_component4.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_component4.html</title>
</head>
<body>
    <h1>component 테스트</h1>
    <div id="app">
        <fortune-component v-bind:fortune="'동쪽으로 가면 귀인을 만나요'"></fortune-component>
        <fortune-component v-bind:fortune="fortune"></fortune-component>
        <input type="text" v-model="fortune" >
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        /*
            1. props:["fortune"]

            fortune 이라는 이름의 props 를 받을 준비가 되어 있는 컴포넌트

            2. props 를 전달하는 방법

            <자식컴포넌트명 v-bind:프로퍼티명="값">
            <fortune-component v-bind:fortune="fortune">

        */
        Vue.component("fortune-component",{
            template:`
                <div>
                    <h2>오늘의 운세</h2>
                    <p>{{fortune}}</p>
                </div>
            `,
            props:["fortune"]
        });

        let app=new Vue({
            el:"#app",
            data(){
                return {
                    fortune:"북쪽으로 가면 규환이를 만나게 될꺼에요!"
                };
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step07_component5.html

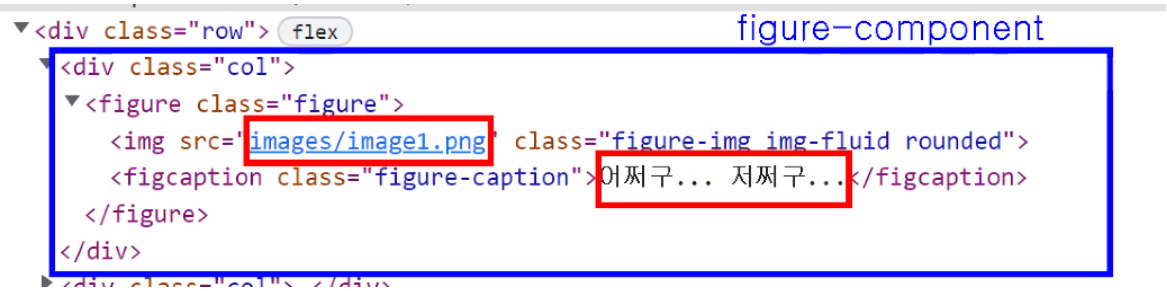
프로퍼티에서 카멜케이스로 만들어 놓은걸 케밥케이스로 연결해라

```
 <fortune-component v-bind:fortune-today="fortune"></fortune-component>
props:["fortuneToday"]
```

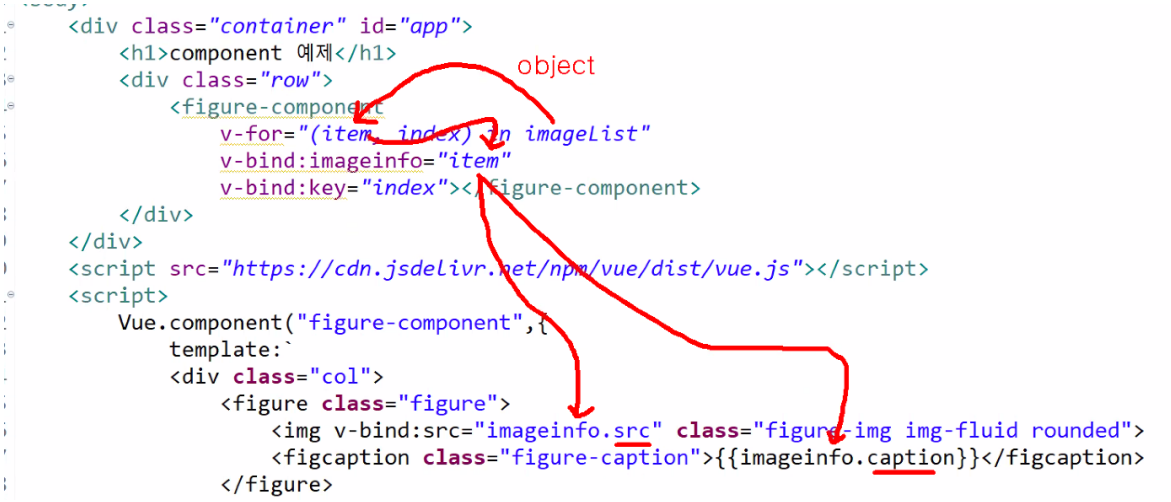
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_component5.html</title>
</head>
<body>
    <h1>component 테스트</h1>
    <div id="app">
        <!--
            props 이름이 camel case 로 작성되어 있으면 kebab case 로 props 를
            전달해야 한다.
        -->
        <fortune-component v-bind:fortune-today="fortune"></fortune-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>

        Vue.component("fortune-component",{
            template:`
                <div>
                    <h2>오늘의 운세</h2>
                    <p>{{fortuneToday}}</p>
                </div>
            `,
            props:["fortuneToday"]
        });

        let app=new Vue({
            el:"#app",
            data(){
                return {
                    fortune:"북쪽으로 가면 귀환이를 만나게 될꺼에요!"
                };
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step07_example.html

figure를 component로 만들어 놓는다




```
<figure class="figure">
    <img v-bind:src="imageinfo.src" class="figure-img img-fluid rounded">
    <figcaption class="figure-caption">{{imageinfo.caption}}</figcaption>
</figure>
```




```
v-bind:src="" //가 필요한 이유
  <p>{{path}}</p>
  <p v-text="path"></p> //path가 출력된다.
  <img src="{{path}}"/> //에러
  <img v-bind:src="path"/> //이렇게 사용
//<img src="모델명"/> 형태로는 사용이 불가하고 v-bind:src"모델명" 형식을 사용해야 한다.
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step07_example.html</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN
</head>
<body>
    <div class="container" id="app">
        <h1>component 예제</h1>
        <div class="row">
            <figure-component
                v-for="(item, index) in imageList"
                v-bind:imageinfo="item"
                v-bind:key="index"></figure-component>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        Vue.component("figure-component",{
            template:`
            <div class="col">
                <figure class="figure">
                    <img v-bind:src="imageinfo.src" class="figure-img img-fluid rounded">
                    <figcaption class="figure-caption">{{imageinfo.caption}}</figcaption>
                </figure>
            </div>
            `,
            props:["imageinfo"]
        });

        let app=new Vue({
            el:"#app",
            data(){
                return {
                    imageList:[
                        {src:"images/image1.png", caption:"어쩌구... 저쩌구..."},
                        {src:"images/image2.png", caption:"어쩌구... 저쩌구..."},
                        {src:"images/image3.png", caption:"어쩌구... 저쩌구..."},
                        {src:"images/image4.png", caption:"어쩌구... 저쩌구..."}
                    ]
                };
            }
        });
    </script>
</body>
</html>
```

v-bind:key 는 뭐에 사용되는 것인가?

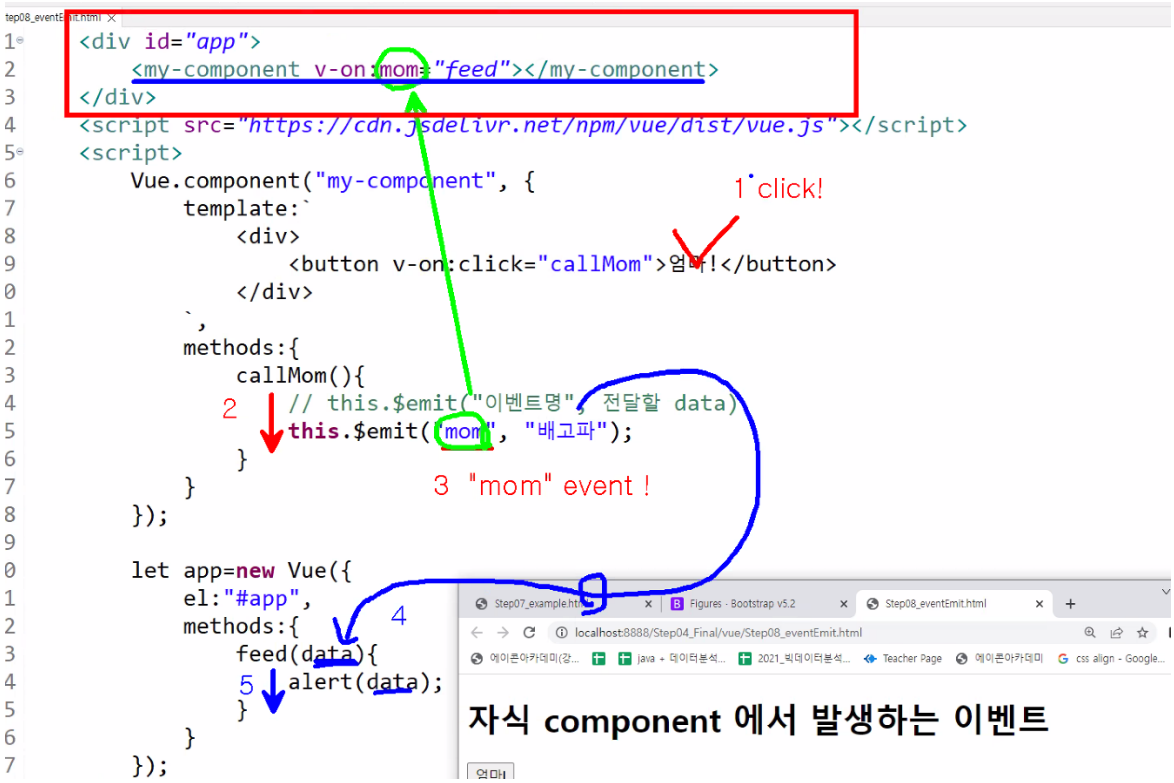
없어도 에러는 아니지만 반복문 사용되면서 여러개의 요소를 출력할 때 해당요소를 유일하게 식별할 수 있는 key 값을 전달해 주는 것이 좋다고 나와있음. 정확한 이유는 모름

자식부모컴퍼넌트 처리

▼ /Vue/Step08_eventEmit.html

```
v-on:event name="함수이름"
```

자식 컴퍼넌트에서 mom이라는 부모 컴퍼넌트가발생하면 feed를 실행
$emit(데이터) 가 함수로 전달.




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step08_eventEmit.html</title>
</head>
<body>
    <h1>자식 component 에서 발생하는 이벤트</h1>
    <div id="app">
        <my-component v-on:mom="feed"></my-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        Vue.component("my-component", {
            template:`
                <div>
                    <button v-on:click="callMom">엄마!</button>
                </div>
            `,
            methods:{
                callMom(){
                    // this.$emit("이벤트명", 전달할 data)
                    this.$emit("mom", "배고파");
                }
            }
```

```
        });

        let app=new Vue({
            el:"#app",
            methods:{
                feed(data){
                    alert(data);
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step08_eventEmit2.html

함수를 호출하면서 전달할 값이 있으면 전달이 가능하다

```
<button v-on:click="deleteItem(index)">삭제</button>
```

삭제기능

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step08_eventEmit2.html</title>
</head>
<body>
    <h1>event emit 예제</h1>
    <div id="app">
        <friend-component
            v-bind:list="members"
            v-on:delete="deleteMember"></friend-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        Vue.component("friend-component",{
            template:`
                <ul>
                    <li v-for="(item, index) in list">
                        {{item}}
                        <button v-on:click="deleteItem(index)">삭제</button>
                    </li>
                </ul>
            `,
            props:["list"],
            methods:{
                deleteItem(i){
                    this.$emit("delete", i);
                }
            }
        });

        let app=new Vue({
            el:"#app",
            data:{
                members:['김구라','해골','원숭이']
            },
            methods:{
                deleteMember(index){
                    this.members.splice(index, 1);
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step08_eventEmit3.html

수정기능 추가

부모가 수정하는 구조

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step08_eventEmit3.html</title>
</head>
<body>
    <h1>event emit 예제</h1>
    <div id="app">
        <friend-component
            v-bind:list="members"
            v-on:delete="deleteMember"
            v-on:update="updateMember"></friend-component>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        Vue.component("friend-component",{
            template:`
                <ul>
                    <li v-for="(item, index) in list">
                        {{item}}
                        <button v-on:click="updateItem(index)">수정</button>
                        <button v-on:click="deleteItem(index)">삭제</button>
                    </li>
                </ul>
            `,
            props:["list"],
            methods:{
                deleteItem(i){
                    this.$emit("delete", i);
                },
                updateItem(i){
                    const newName=prompt("수정할 이름을 입력하세요");

                    //this.$emit("update", {i:i, newName:newName});
                    this.$emit("update", {i, newName});
                }
            }
        });

        let app=new Vue({
            el:"#app",
            data:{
                members:['김구라','해골','원숭이']
            },
            methods:{
                deleteMember(index){
                    this.members.splice(index, 1);
                },
                updateMember(data){
                    //아래처럼 배열을 변경하면 변경이 감지가 안되기 때문에 화면 업데이트가 안된다.
                    //this.members[data.i] = data.newName;

                    //아래의 2가지 방법중 하나로 배열을 변경해야 한다.
                    //Vue.set(this.members, data.i, data.newName);
                    this.$set(this.members, data.i, data.newName);
                }
            }
        });
    </script>
</body>
</html>
```

오브젝트 작성 방법

```
this.$emit("update", {i:i, newName:newName});
this.$emit("update", {i, newName});

//둘은 같은 결과
```

▼ /Vue/Step09_ref.html

ref 사용해서 문서 객체의 참조값 얻어내기

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Step09_ref.html</title>
</head>
<body>
    <h1>ref 를 이용해서 문서 객체의 참조값 얻어내기</h1>
    <div id="app">
        <button v-on:click="clicked">눌러보셈</button>
        <input type="text" ref="one" >
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <script>
        let app=new Vue({
            el:"#app",
            methods:{
                clicked(){
                    // ref="one" 인 문서객체의 참조값
                   console.log(this.$refs.one);
                   this.$refs.one.value="버튼을 눌렀네?";
                }
            }
        });
    </script>
</body>
</html>
```

▼ /Vue/Step10_Ajax.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>/vue/Step10_Ajax.html</title>
</head>
<body>
  <h1>ajax 요청을 통해서 받아온 데이터 사용하기</h1>
  <p> 페이지 전환없이 서버에 요청하는것을 ajax 라고 생각하면 된다.</p>
  <div id="app">
    <button v-on:click="getList">글 목록 받아오기</button>
    <br />
    <table>
      <thead>
        <tr>
          <th>번호</th>
          <th>작성자</th>
          <th>제목</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="tmp in list" v-bind:key="tmp.num">
          <td>{{tmp.num}}</td>
          <td>{{tmp.writer}}</td>
          <td>{{tmp.title}}</td>
        </tr>
      </tbody>
    </table>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    new Vue({
      el:"#app",
      data:{
        list:[]
      },
      methods:{
        getList(){
          //Vue 의 참조값을 self 에 담기
          const self=this;

          fetch("/Step04_Final/cafe/json_list.jsp")
          .then(function(response){
            return response.json();
          })
          .then(function(data){
            console.log(data);
            //서버로 부터 받은 데이터를 list 에 대입하기
            self.list=data;
```

```
        });
      }
    }
  });
</script>
</body>
</html>
```

▼ /Vue/Step10_Ajax2.html

created() - vue가 준비되었을 때 최초 한번 호출됨

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>/vue/Step10_Ajax2.html</title>
</head>
<body>
  <h1>ajax 요청을 통해서 받아온 데이터 사용하기</h1>
  <p> 페이지 전환없이 서버에 요청하는것을 ajax 라고 생각하면 된다.</p>
  <div id="app">
    <table>
      <thead>
        <tr>
          <th>번호</th>
          <th>작성자</th>
          <th>제목</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="tmp in list" v-bind:key="tmp.num">
          <td>{{tmp.num}}</td>
          <td>{{tmp.writer}}</td>
          <td>{{tmp.title}}</td>
        </tr>
      </tbody>
    </table>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script>
    new Vue({
      el:"#app",
      data:{
        list:[]
      },
      created(){
        // Vue 가 준비가 되었을때 (root component 가 준비 되었을때) 최조 한번 호출된다.
        console.log("created!");
        //Vue 의 참조값을 self 에 담기
        const self=this;

        fetch("/Step04_Final/cafe/json_list.jsp")
        .then(function(response){
          return response.json();
        })
        .then(function(data){
          console.log(data);
          //서버로 부터 받은 데이터를 list 에 대입하기
          self.list=data;
        });
      },
      methods:{

      }
    });
  </script>
</body>
</html>
```

▼ /Vue/Step10_example.jsp

서버 사이드 렌더링 비교해보기

```
<%@page import="test.cafe.dao.CafeDao"%>
<%@page import="test.cafe.dto.CafeDto"%>
<%@page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%
```

```
  //글목록
  List<CafeDto> list=CafeDao.getInstance().getList();
  request.setAttribute("list", list);
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>/vue/Step10_example.jsp</title>
</head>
<body>
  <h1>글목록 입니다.(서버 사이드 렌더링)</h1>
  <div id="app">
    <table>
      <thead>
        <tr>
          <th>번호</th>
          <th>작성자</th>
          <th>제목</th>
        </tr>
      </thead>
      <tbody>
        <c:forEach var="tmp" items="${requestScope.list }">
          <tr>
            <td>${tmp.num }</td>
            <td>${tmp.writer }</td>
            <td>${tmp.title }</td>
          </tr>
        </c:forEach>
      </tbody>
    </table>
  </div>
</body>
</html>
```

오늘 사용한 vue js 기능

$refs

$emit

$set