1-1~2

const app = new Vue({

        el:'#root',

        data:{

            liked:false,

        },

        methods:{

            onClickButton(){

                //method에서 this는 data를 가르킨다.

                this.liked = true;

            },

        },

    });

Vue, react를 할 때는 Data만 관리한다.

Data 중심으로 생각하는 방식을 가져야 한다.

Data만 관리하면 화면은 Vue가 알아서 바꿔주는 것이 Vue의 강점이다.

<div v-if="liked">

V가 붙은 속성의 값(따옴표 안)에는 자바스크립트 코드를 쓸 수가 있다.

1-3~4

<span>{{first \* 100}} 이렇게 적는것도 가능하다.</span>

<input type="number" v-model="value">

v-model으로 input값과 data를 연결할 수 있다.

항상 어느 부분이 바뀌는지 생각해야 한다. (바뀌는 data를 항상 생각해야 한다.)

<input type="number" v-model="value" ref="answer">

this.$refs.answer.focus();

input 태그에 focus를 주고 싶은대 focus 같은 경우는 data라 보기 애매하다.

그래서 vue가 식별할 수 있게 ref 속성에 값을 주고, javascript에서 $refs.이름.focus();

이렇게 할 수 있다.

주의!! : $refs 으로 data를 바꾸면 vue가 자동으로 값을 바꿔주지 않아서 화면의 불일치가 일어나니 위와 같은 행동은 하지 않는 것이 좋다.

1-5

항상 화면은 처음에 만들어놓고 데이터 중심으로 생각!!

1-6

<div v-if="route === 'main'">메인화면</div>

    <div v-else-if="route === 'sub'">서브화면</div>

    <div v-else-if="route === 'third'">세번째화면</div>

data:{

            word:'제로초',

            result:'',

            value:'',

            route:'main',

와 같은 형식으로 화면 전환하며 보여주기 가능하다.

2-1

반복되는 부분은 컴포넌트로 만든다.

<script>

Vue.component('word-relay', {

data:{

},

methods:{

}

})

</script>

컴포넌트를 생성할 때는 data 부분에서 다른점이 존재한다.

2-2

데이터를 Vue 인스턴스에서는 객체형식으로 표현해왔는데, 컴포넌트에서는 함수 형식으로 만들어야 한다.

template:`

<div>{{word}} </div>

<form v-on:submit="onSubmitForm">

<input type="text" v-model="value" ref="answer">

<button type="submit">입력!</button>

</form>

<div>{{result}}</div>

`,

template에서는 위 처럼 동등한 형제 테그가 있으면 안된다.

template:`

<div>

<div>{{word}} </div>

<form v-on:submit="onSubmitForm">

<input type="text" v-model="value" ref="answer">

<button type="submit">입력!</button>

</form>

<div>{{result}}</div>

</div>

위처럼 template은 항상 하나의 부모 테그로 묶어줘야한다는 제약조건이 있다.

컴포넌트는 같은 것을 랜더링하지만 중복을 제거하는 것. 컴포넌트로 생성된 서로 다른 것들은 서로 독립적인 각각의 data를 가진다.

컴포넌트를 만드는 스크립트를 어떤 스크립트보다 상단에 두어야 한다.