컴포넌트이론.txt

```
컴포넌트는 전역 또는 로컬에 등록 한 후, 사용자 정의 태그로 사용 가능

Vue.component 메소드를 사용해서 전역에 등록하면 자동으로 모든 컴포넌트에서 사용 가능
메소드의 매개변수로 전달되는 정보
1) 사용자 정의 태그로 사용할 이름
2) 컴포넌트의 옵션 객체

컴포넌트의 정의
Vue.component( 'my-component', {
  template: '<p>MyComponent</p>'
})

컴포넌트 사용
<div id="app">
  <my-component></my-component>
</div>

실제 렌더링 결과
<div id="app">
  <p>MyComponent</p>
</div>


로컬에 등록
컴포넌트 정의를 특정한 컴포넌트의 components 옵션에 등록
-> 로컬로 등록된 해당 컴포넌트 범위(스코프;scope) 내부에서만 해당 컴포넌트를 사용할 수 있도록 제한 가능

// 컴포넌트 정의
var myComponent = {
  template = '<p>MyComponent</p>'
}

new Vue( {
  el: '#app',
  components: {
    // <my-component>가 루트에서만 사용 가능
    'my-component': myComponent
  }
} );


컴포넌트 옵션
루트 생성자 new Vue() 의 옵션과 같이 컴포넌트 전용 템플릿 외에도 데이터와 메소드 정의 가능

Vue.component( 'my-component', {
  //템플릿
  template: '<p> {{ msg }} </p>,
  // 데이터는 객체를 리턴하는 함수 지정
  // data는 함수여야 하며, 객체를 리턴하는 함수로 정의해야 함!
  data: function () {
```

```
    return {
      msg: 'Hello! Vue~~'
    }
  },
  methods: {
    // 메소드, 산출 속성, 와처의 정의 방법
    // -> 루트 생성자의 객체와 동일
  }
 } )

## 템플릿의 루트 요소는 반드시 하나여야만 한다!
// p 요소가 하나를 초과하기 때문에 사용 불가
template: '<p>내용1</p><p>내용2</p>'

// 여러 개의 요소가 필요한 경우 전체를 다른 요소로 감싸주면 결과적으로 1개의 요소이므로 조건 만족
template: '<div id="app"><p>내용1</p><p>내용2</p></div>'


컴포넌트 인스턴스(instance)
<div id="app">
  <my-component></my-component>
  <my-component></my-component>
</div>

컴포넌트를 여러 번 사용한 경우 my-component를 기반으로 만들어진 완전히 다른 인스턴스로 취급됨
별도의 고유 속성을 가지고 있지 않으면 두 가지는 완전히 같은 동작을 함

```

01_컴포넌트통신.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>컴포넌트 간의 통신</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <h3>프로퍼티로 문자열 전달</h3>
    <comp-child val="자식A"></comp-child>
    <comp-child val="자식B"></comp-child>
    <hr>
    <h3>프로퍼티로 데이터 전달</h3>
    <comp-child :val="valueA"></comp-child>
    <comp-child :val="valueB"></comp-child>
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 컴포넌트에서 프로퍼티를 전달하기 위한 props 정의
    Vue.component( 'comp-child', {
      // 템플릿에서 val 사용하기
      template: '<p>{{ val }}</p>',
      // 받을 속성 이름 지정하기
      props: ['val']
    } );

    new Vue( {
      el: '#app',
      data: {
        valueA: '자식A',
        valueB: '자식B'
      }
    } );
  </script>
</body>
</html>
```

02_리스트렌더링.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>리스트 렌더링</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <ul>
      <comp-child v-for="item in list"
        v-bind:key="item.id"
        v-bind:name="item.name"
        v-bind:hp="item.hp"></comp-child>
    </ul>
  </div>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 자식 컴포넌트
    Vue.component( 'comp-child', {
      template: '<li>{{ name }} HP.{{ hp }}</li>',
      props: ['name', 'hp']
    } );

    // 부모 컴포넌트
    new Vue( {
      el: '#app',
      data: {
        list: [
          { id: 1, name: '슬라임', hp: 100 },
          { id: 2, name: '고블린', hp: 200 },
          { id: 3, name: '드래곤', hp: 500 }
        ]
      }
    } );
  </script>
</body>
</html>
```

03_props로받을자료형지정.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>props로 받을 자료형 지정</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <example v-bind:value="value"></example>
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 1~4 까지 하나씩 확인!
    // 1. 자료형 확인을 생략하는 경우
    // Vue.component( 'example', {
    //   props: ['value']
    // } );

    // 2. 자료형만 확인하는 경우
    // Vue.component( 'example', {
    //   props: {
    //     value: 자료형
    //   }
    // } );

    // 3. 인스턴스를 확인하는 경우
    // function Cat(name) {
    //   this.name = name
    // }
    // Vue.component( 'example', {
    //   props: {
    //     value: Cat // Cat 데이터만 허가
    //   }
    // } );
    // new Vue( {
    //   data: {
    //     value: new Cat('구름') // value는 Cat 데이터
    //   }
    // } );

    // 4. 옵션도 사용하는 경우
    // Vue.component( 'example', {
    //   props: {
    //         value: {
    //         type: [String, Number], // 자료형, 배열
    //         default: 100,           // 데이터, 함수
    //         required: true,         // Boolean, 필수 여부
    //         validator: function (value) { // 사용자 정의 유효성 검사 함수,
```

03_props로받을자료형지정.html

```
      Boolean 값 리턴
53    //           return value > 10
54    //         }
55    //       }
56    //     }
57    //   } );
58   </script>
59 </body>
60 </html>
```

04_자식이벤트를부모에서잡기.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>자식 이벤트를 부모에서 잡기</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <!-- 부모 템플릿 -->
    <comp-child v-on:childs-event="parentsMethod">□</comp-child>
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 자식 컴포넌트
    Vue.component( 'comp-child', {
      template: '<button v-on:click="handleClick">이벤트</button>',
      methods: {
        // 버튼 클릭 이벤트 핸들러로 childs-event 실행
        handleClick: function () {
          this.$emit('childs-event')
        }
      }
    } );
    // 부모 컴포넌트
    new Vue( {
      el: '#app',
      methods: {
        // childs-event가 실행되었을 경우 실행
        parentsMethod: function () {
          alert('자식에서 전달받은 이벤트!')
        }
      }
    } );

  </script>
</body>
</html>
```

05_부모데이터조작.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>부모 데이터 조작</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <ul>
      <comp-child v-for="item in list"
        v-bind:key="item.id"
        v-bind="item"
        v-on:attack="handleAttack"></comp-child>
    </ul>
  </div>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 자식 컴포넌트
    Vue.component( 'comp-child', {
      template: '<li>{{ name }} HP.{{ hp }}\
      <button v-on:click="doAttack">공격하기</button></li>',
      props: {
        id: Number,
        name: String,
        hp: Number
      },
      methods: {
        // 버튼 클릭 이벤트 핸들러에서 $emit을 호출해서 attack 이벤트 실행
        doAttack: function () {
          // 매개 변수로 자신의 ID 전달
          this.$emit('attack', this.id)
          // $emit : 컴포넌트에 연결되어 있는 이벤트를 명시적으로 실행시키는 메소드
          // jQuery의 trigger()와 유사함
        }
      }
    } );

    new Vue( {
      el: '#app',
      data: {
        list: [
          { id: 1, name: '슬라임', hp: 100 },
          { id: 2, name: '고블린', hp: 200 },
          { id: 3, name: '드래곤', hp: 500 }
        ]
      },
      methods: {
        // attack가 실행된 경우
```

05_부모데이터조작.html

```
53          handleAttack: function (id) {
54            // 매개 변수의 ID를 기반으로 요소 검색
55            var item = this.list.find(function (el) {
56              return el.id === id
57            })
58            // HP가 0보다 크면 지정한 수치만큼 줄이기
59            if (item !== undefined && item.hp > 0) item.hp -= 70
60          }
61        }
62      } );
63    </script>
64  </body>
65  </html>
```

06_컴포넌트간통신.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>컴포넌트 간 통신</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    부모 자식 관계가 아닌 경우 컴포넌트 사이의 통신
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 이벤트 bus 전용 인스턴스 생성
    var bus = new Vue( {
      data: {
        count: 0
      }
    } );

    Vue.component( 'component-b', {
      template: '<p> bus: {{ bus.count }} </p>',
      computed: {
        // bus 데이터를 산출 속성으로 사용
        bus: function () {
          return bus.$data
        }
      },
      created: function () {
        bus.$on( 'bus-event', function () {
          // bus-event 이벤트가 발생할 때 처리
          // 익명 함수 내부의 this는 bus의 인스턴스
          this.count++
        } )
      }
    } );
  </script>
</body>
</html>
```

07_자식컴포넌트참조.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>자식 컴포넌트를 참조하는 $refs</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <comp-child ref="child"></comp-child>
  </div>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    // 자식 컴포넌트
    Vue.component( 'comp-child', {
      template: '<div>...</div>',
      created: function () {
        // 자신의 처리
        this.$on('open', function () {
          console.log('무언가 처리하기')
        })
      }
    } );

    // 부모 컴포넌트
    new Vue({
      el: '#app',
      methods: {
        handleClick: function () {
          // 자식 컴포넌트의 이벤트 실행
          this.$refs.child.$emit('open')
        }
      }
    });

  </script>
</body>
</html>
```

08_사용자정의컴포넌트-슬롯.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <!-- 부모 컴포넌트 / 슬롯 컨텐츠 정의 -->
  <comp-child>
    <header slot="header">
      Hello Vue.js!
    </header>
    Vue.js는 JavaScript 프레임워크입니다.
  </comp-child>

  <!-- 자식 컴포넌트 / 슬롯 사용 -->
  <section class="comp-child">
    <slot name="header">
      <header>
        디폴트 타이틀
      </header>
    </slot>
    <div class="content">
      <slot>디폴트 콘텐츠</slot>
    </div>
    <slot name="footer">
      <!-- 없다면 아무 것도 출력하지 않습니다. -->
    </slot>
  </section>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    var app = new Vue( {
      el: '#app',

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

09_v-model사용자정의.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>v-model 사용자 정의</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">

  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    Vue.component( 'my-calendar', {
      model: {
        // 현재 값을 value가 아니라 current로 할당하고 싶은 경우
        prop: 'current',
        // 이벤트를 change로 사용하고 싶은 경우
        event: 'change'
      },
      // props에서 설정하기
      props: {
        current: String
      },
      created: function () {
        this.$emit('change', '2021-01-01')
      }
    } );
  </script>
</body>
</html>
```

10_양방향데이터바인딩-vsync.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>양방향 데이터 바인딩 - .vsync</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <my-component
      v-bind:name.sync="name"
      v-bind:hp.sync="hp">
    </my-component>
  </div>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    /* 자식 컴포넌트 */
    Vue.component('my-component', {
      template: '<div class="my-component">\
      <p>이름.{{ name }} HP.{{ hp }}</p>\
      <p>이름 <input v-model="localName"></p>\
      <p>HP <input size="5" v-model.number="localHp"></p>\
      </div>',
      props: {
        name: String,
        hp: Number
      },
      computed: {
        // 산출 속성의 세터와 게터를 통해 v-model 사용
        localName: {
          get: function () {
            return this.name
          },
          set: function (val) {
            this.$emit('update:name', val)
          }
        },
        localHp: {
          get: function () {
            return this.hp
          },
          set: function (val) {
            this.$emit('update:hp', val)
          }
        }
      }
    } );

    /* 부모 컴포넌트 */
```

10_양방향데이터바인딩-vsync.html

```
53      new Vue( {
54        el: '#app',
55        data: {
56          name: '슬라임',
57          hp: 100
58        }
59      } );
60    </script>
61  </body>
62  </html>
```

11_함수형컴포넌트.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>함수형 컴포넌트</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">

  </div>

  <!-- Vue.js -->
  <script
  src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    Vue.component( 'functional-component', {
      functional: true,
      render: function (createElement, context) {
        return createElement('div', context.props.message)
      },
      props: {
        message: String
      }
    } );
  </script>
</body>
</html>
```

12_동적컴포넌트.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>동적 컴포넌트</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <button v-on:click="current^=1">컴포넌트변경</button>

    <div v-bind:is="component"></div>
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    /* 자식 컴포넌트 */
    // 컴포넌트A
    Vue.component( 'my-component-a', {
      template: '<div class="my-component-a">component A</div>'
    } );
    // 컴포넌트B
    Vue.component( 'my-component-b', {
      template: '<div class="my-component-b">component B</div>'
    } );

    /* 부모 컴포넌트 */
    new Vue( {
      el: '#app',
      data: {
        // 컴포넌트 리스트
        componentTypes: ['my-component-a', 'my-component-b'],
        // 렌더링할 컴포넌트를 선택하는 index
        current: 1
      },
      computed: {
        component: function () {
          // current와 일치하는 index 컴포넌트 사용
          return this.componentTypes[this.current]
          // `return current ? 'my-component-b' : 'my-component-a'` 사용 가능
        }
      }
    } );
  </script>
</body>
</html>
```

13_공통처리-Mixin.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>공통 처리 - Mixin</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>

  <div id="app">

  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    /* 믹스인 정의 */
    var mixin = {
      created: function () {
        this.hello()
      },
      methods: {
        hello: function () {
          console.log('hello from mixin!')
        }
      }
    }

    /* 믹스인 사용 */
    Vue.component( 'my-component-a', {
      mixins: [mixin], // 믹스인 등록
      template: '<p>MyComponentA</p>'
    } );
    Vue.component( 'my-component-b', {
      mixins: [mixin], // 믹스인 등록
      template: '<p>MyComponentB</p>'
    } );
  </script>
</body>
</html>
```

14_상태유지-keep-alive.html

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>상태 유지 - keep-alive</title>
  <link rel="stylesheet" href="../css/main.css">
</head>
<body>
  <div id="app">
    <button v-on:click="current='comp-board'">메시지 목록</button>
    <button v-on:click="current='comp-form'">입력 양식</button>

    <keep-alive>
      <div v-bind:is="current"></div>
    </keep-alive>
  </div>

  <!-- Vue.js -->
  <script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.9/vue.min.js'></script>
  <script>
    /* 자식 컴포넌트 */
    // 메시지 목록 전용 컴포넌트
    Vue.component( 'comp-board', {
      template: '<div>Message Board</div>',
    } );
    // 입력 양식 전용 컴포넌트
    Vue.component( 'comp-form', {
      template: '<div>Form<textarea v-model="message"></textarea></div>',
      data: function () {
        return {
          message: ''
        }
      }
    } );

    /* 부모 컴포넌트 */
    var app = new Vue( {
      el: '#app',
      data: {
        current: 'comp-board' // 동적으로 변경
      }
    } );
  </script>
</body>
</html>
```