SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Vue.js

SAMSUNG SW ACADEMY FOR YOUTH

FrontEnd Framework

Vue Component

- JavaScript module
- Vue Component
- 컴포넌트 간 통신

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

JavaScript Module

JavaScript Module

Module

- 프로그램을 기능별로 여러 개의 파일로 나누는 형태
- 브라우저의 지원 여부 확인 https://developer.mozilla.org/ko/docs/Web/JavaScript/Guide/Modules

Module 시스템

- CommonJS (NodeJS) - 웹 브라우저 밖에서도 동작될 수 있는 모듈 규칙 설립
- AMD (Asynchronous Module Definition) - 비동기적으로 모듈을 로딩
- ESM (ECMAScript Module, ECMA215, es6) - 자바스크립트 자체 모듈

Module 정의 및 사용

- 가져오기 : import
- import 모듈 명 from 모듈위치

- 내보내기 : export
- exprot
- exprot default

- 내보내기 : export

```
export const title = '계산기 모듈';
export function add(i, j) {
    return i + j;
}
export function sub(i, j) {
    return i - j;
}
```

- 가져오기 : import

```
import {title, add, sub} from '파일명';
```

Module 정의 및 사용

- 가져오기 : import
- import 모듈 명 from 모듈위치

- 내보내기 : export
- exprot
- exprot default

- 내보내기 : export

```
const title = '계산기 모듈';
function add(i, j) {
    return i + j;
}
function sub(i, j) {
    return i - j;
}
export { title, add, sub }
```

- 가져오기 : import

```
import {title, add, sub} from '파일명';
```

Module 정의 및 사용

- 가져오기 : import
- import 모듈 명 from 모듈위치

- 내보내기 : export
- exprot
- exprot default

- 내보내기 : export default

```
export default {
    title: '계산기 모듈',
    add(i, j) { return i + j; },
    sub(i, j) { return i – j;  }
}
```

- 가져오기 : import

```
import cal from '파일명';
cal.title
cal.add(20, 10)
cal.sub(20, 10)
```

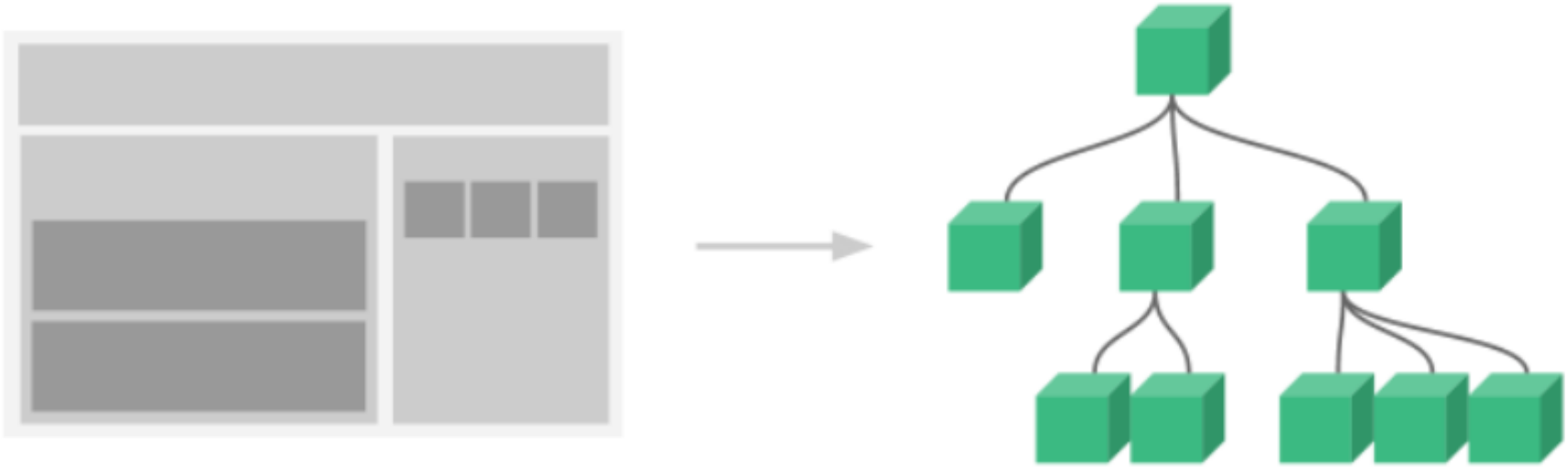
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Vue Component

Vue Component

- Vue의 가장 강력한 기능 중 하나.
- 기본 HTML Element를 확장하여 재사용 가능한 코드를 캡슐화 하는데 도움이 됨.
- Vue Component는 Vue Instance 이기도 함.
  (루트에서만 사용하는 옵션을 제외하고 모든 옵션 객체 사용가능)
- Life Cycle Hook 사용가능
- 전역 컴포넌트와 지역 컴포넌트 사용.



https://kr.vuejs.org/v2/guide/index.html#컴포넌트그림

Vue Component (전역)

- 전역 컴포넌트를 등록하려면 Vue.Component(tagName, options)를 사용
- 권장하는 컴포넌트 이름 : 케밥 케이스

```
Vue.component(“my-comp”, {
    // 옵션
});
```

```
<div id="app1">
  <my-comp></my-comp>
</div>

<div id="app2">
  <my-comp></my-comp>
</div>
```

```
Vue.component('my-comp', {
  template: '<div>전역 컴포넌트</div>'
});
new Vue({
  el: '#app1'
});
new Vue({
  el: '#app2'
});
```

Vue Component (지역)

- 모든 컴포넌트를 전역으로 등록할 필요 X
- 컴포넌트를 components 인스턴스 옵션으로 등록함으로써 다른 인스턴스/컴포넌트의 범위에서만 사용할 수 있는 컴포넌트 생성 가능.

```
var Child = {
  template: '<div>사용자 정의 컴포넌트 입니다!</div>'
}

new Vue({
  // ...
  components: {
  // <my-component> 는 상위 템플릿에서만 사용할 수 있습니다.
    'my-component': Child
  }
})
```

Vue Component (지역)

- 모든 컴포넌트를 전역으로 등록할 필요 X
- 컴포넌트를 components 인스턴스 옵션으로 등록함으로써 다른 인스턴스/컴포넌트의 범위에서만 사용할 수 있는 컴포넌트 생성 가능.

```
<div id="app1">
    <my-comp></my-comp>    (O)
</div>

<div id="app2">
    <my-comp></my-comp>    (X)
</div>
```

```
let comp = {
    template: '<div>지역 컴포넌트</div>'
}
new Vue({
    el: "#app1",
    components: {
        'my-comp': comp
    }
});
```

Vue Component data

- data는 반드시 함수여야 한다.
- data 객체를 공유하는 문제를 막고 새로운 데이터 객체를 반환해서 해결한다.

```
<div id="app">
  <count-view></count-view>
  <count-view></count-view>
</div>
<template id="count-view">
<div>
 <span>{{ count }}</span>
 <button @click="count++">..</button>
</div>
</template>
```

```
let data = { count: 0 };
Vue.component('count-view', {
  data() {
    return data
  },
  template: '#count-view'
});
```

Vue Component data

- data는 반드시 함수여야 한다.
- data 객체를 공유하는 문제를 막고 새로운 데이터 객체를 반환해서 해결한다.

```
<div id="app">
  <count-view></count-view>
  <count-view></count-view>
</div>
<template id="count-view">
<div>
 <span>{{ count }}</span>
 <button @click="count++">..</button>
</div>
</template>
```

```
Vue.component('count-view', {
  data() {
    return {
      count: 0
    }
  },
  template: '#count-view'
})
```

Vue Component Template

- DOM을 템플릿으로 사용할 때, Vue는 템플릿 콘텐츠만 가져올 수 있기 때문에 HTML이 작동하는 방식에 고유한 몇 가지 제한 사항이 적용됨.
- 따라서 가능한 경우 항상 문자열 템플릿을 사용하는 것이 좋음

```
<div id="app">
  <my-comp></my-comp>
</div>
```

```
<template id=“my-temp”>
  <div>로컬</div>
</template>
Vue.component('my-comp', {
  template: “#my-temp”
})
new Vue({
  el: '#app'
})
```

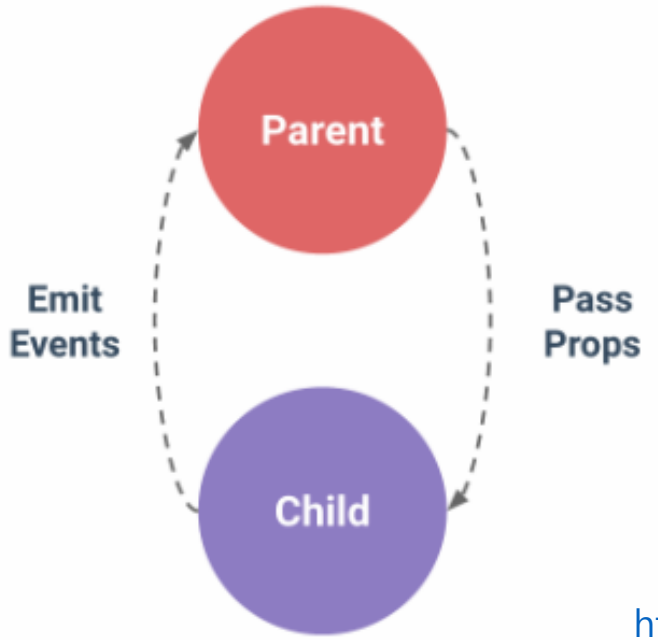
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Vue Component 통신

Vue Component 통신

- 컴포넌트는 부모-자식 관계에서 가장 일반적으로 함께 사용하기 위한 것. (트리구조)
- 부모는 자식에게 데이터를 전달 (Pass Props)
- 자식은 부모에게 일어난 일을 알림 (Emit Event)
- 부모와 자식이 명확하게 정의된 인터페이스를 통해 격리된 상태 유지
- props는 아래로, events는 위로



https://kr.vuejs.org/v2/guide/components.html#컴포넌트작성

Vue Component 통신

부모 컴포넌트 → 자식 컴포넌트

- 상위 컴포넌트에서 하위 컴포넌트로 데이터 전달
- 하위 컴포넌트는 상위 컴포넌트의 값을 직접 참조 불가능
- data와 마찬가지로 props 속성의 값을 template에서 사용 가능

```
Vue.component('child', {
  // props 정의
  props: ['message'],
  // 데이터와 마찬가지로 prop은 템플릿 내부에서 사용할 수 있으며
  // vm의 this.message로 사용할 수 있습니다.
  template: '<span>{{ message }}</span>'
})
```

```
<child message="안녕하세요!"></child>
```

Dynamic Props

- v-bind를 사용하여 부모의 데이터에 props를 동적으로 바인딩 할 수 잇음.
- 데이터가 상위에서 업데이트 될 때마다 하위 데이터로도 전달.

```
<div>
  <input v-model="parentMsg">
  <br>
  <child v-bind:my-message="parentMsg"></child>
</div>
```

- v-vind 단축 구문 사용 가능

```
<child :my-message="parentMsg"></child>
```

Dynamic Props 객체의 속성 전달

- 객체의 모든 속성을 props로 전달할 경우 인자없이 v-bind를 사용

```
todo: {
  text: 'Learn Vue',
  isComplete: false
}
```

```
<todo-item v-bind="todo"></todo-item>
```

- 위의 코드는 아래와 같이 동작

```
<todo-item
  v-bind:text="todo.text"
  v-bind:is-complete="todo.isComplete"></todo-item>
```

Vue Component 통신

단방향 데이터 흐름

- 모든 props는 하위 속성과 상위 속성 사이의 단방향 바인딩을 형성
- 상위 속성이 업데이트 되면 하위로 흐르게 되지만 반대로는 안됨.
- 하위 컴포넌트가 실수로 부모의 상태를 변경하여 앱의 데이터 흐름을 이해하기 어렵게 만드는 일 방지
- 상위 컴포넌트가 업데이트 될 때마다 하위 컴포넌트의 모든 prop들이 최신 값으로 갱신됨.

사용자 정의 이벤트

- 컴포넌트 및 props와는 달리, 이벤트는 자동 대소문자 변환 제공하지 않음.
- 대소문자를 혼용하는 대신에 emit할 정확한 이벤트 이름을 작성하는 것을 권장
- DOM 템플릿의 v-on 이벤트 리스너는 항상 자동으로 소문자 변환 (v-on:myEvent → v-on:myevent)
- 따라서 이벤트 이름에는 kebab-case를 사용하는 것을 권장.

- $on(eventName) : 이벤트 수신
- $emit(eventName) : 이벤트 발생, 추가 인자는 리스너의 콜백 함수로 전달

- 부모 컴포넌트는 자식 컴포넌트가 사용되는 템플릿에서 v-on을 사용하여 자식 컴포넌트가 보낸 이벤트를 청취

Vue Component 통신

자식 컴포넌트 → 부모 컴포넌트

- 이벤트 발생과 수신을 이용
- 자식 컴포넌트에서 부모 컴포넌트가 지정한 이벤트를 발생 ($emit)
- 부모 컴포넌트는 자식 컴포넌트가 발생한 이벤트를 수신(on) 하여 데이터 처리

```
// 이벤트 발생
this.$emit("이벤트명");

// 이벤트 수신
<child v-on:이벤트명="부모 컴포넌트 메서드명"></child>
```

Vue Component 통신

비 부모-자식간 통신

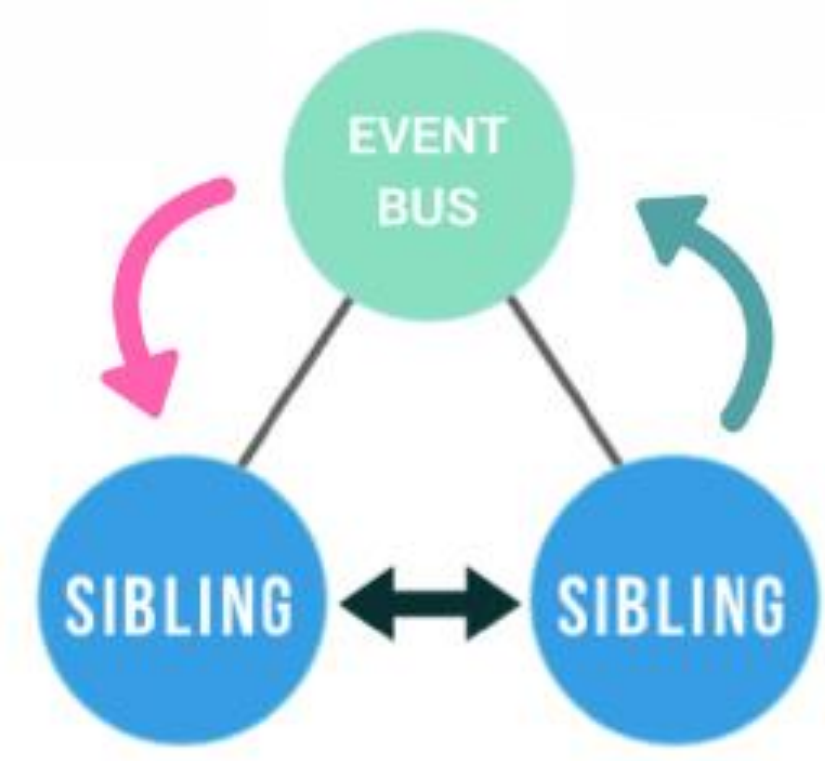
- 두 컴포넌트가 통신할 필요가 있지만 서로 부모/자식이 아닐 수도 있음.
- 비어 있는 Vue Instance 객체를 Event Bus로 사용.

```
var bus = new Vue()


// 컴포넌트 A
bus.$emit('id-selected', 1)

// 컴포넌트 B
bus.$on('id-selected', function (id) { // ... })
```

- 복잡해질 경우 상태관리 라이브러리 Vuex 사용 권장



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미