

# Vue.js 세미나

[ 시스템응용팀 이하영 ]

# Contents

**1 Vue.js 개요**

**2 개발 환경 설정**

**3 Vue.js 구성요소**

**4 Vue CLI**

**5 애플리케이션 만들기**

# Content 1

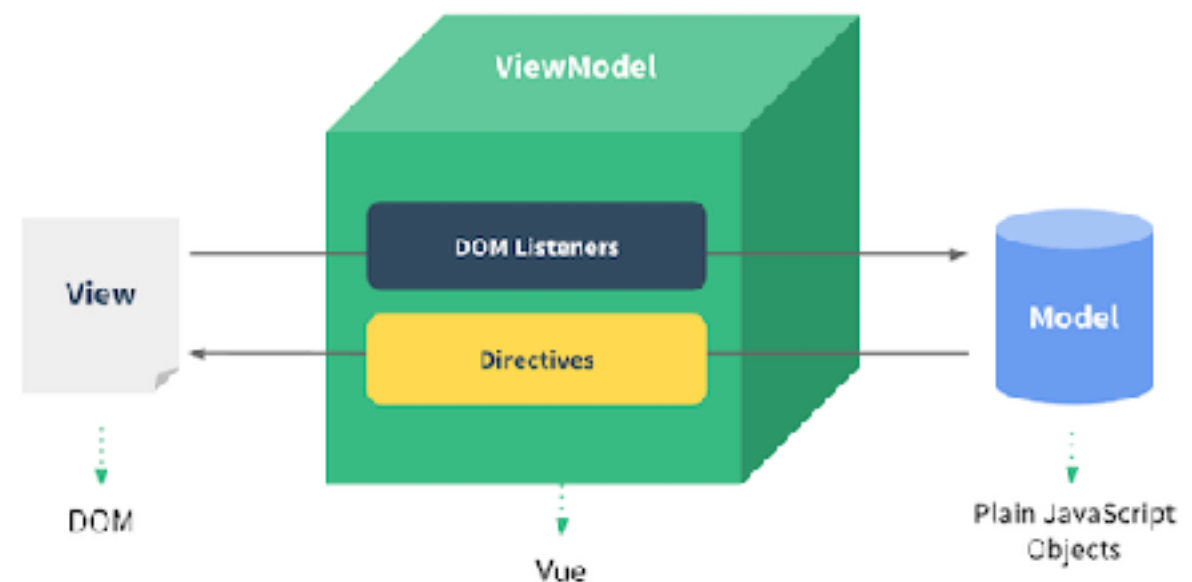
## Vue.js 개요

# Vue.js란?

2013년 12월, Google Creative Lab에서 일하던 Evan you가 UI를 빠르게 개발하기 위해서 만들었습니다. UI 개발에 최적화된 프레임워크 입니다.

## 특징

- ✓ MVVM 패턴의 라이브러리
- ✓ 컴포넌트 기반 프레임워크
- ✓ 양방향 데이터 바인딩
- ✓ 단방향 데이터 통신
- ✓ 가상 돔 렌더링 방식 적용



# Content 2

## 개발 환경 설정

# 개발 환경 설정

## ✓ Node.js 설치

뷰 CLI를 이용하여 쉽게 뷰 프로젝트를 구성하기 위해서 필요합니다. 또한, 뷰 CLI로 생성한 프로젝트에서 개발을 할 때도 node.js 서버를 사용합니다.

## ✓ 크롬 브라우저 + 뷰 개발자 도구 설치

뷰로 만든 웹 앱의 구조를 간편하게 디버깅, 분석할 수 있습니다.

## ✓ Vue-CLI 설치

Vue.js 앱을 개발할 때 프로젝트의 기본적인 인터페이스와 틀을 제공하는 커맨드라인 인터페이스 기반의 도구입니다.

# Content 3

## Vue.js 구성요소

[ 인스턴스, 컴포넌트, 라우터, 템플릿 ]

# 뷰 인스턴스

뷰로 화면을 개발하기 위해 필수적으로 생성해야 하는 기본 단위로, 뷰 생성자 함수인 **new Vue()**로 인스턴스를 생성합니다.

```
<div id="app">
  {{ message }}
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue.js'
  }
})
```





# 뷰 인스턴스

Vue 객체를 생성할 때 아래와 같이 data, template, el, methods, life cycle callback 등의 옵션들을 포함할 수 있습니다.

```
var vm = new Vue({  
  template: ...,  
  el: ...,  
  methods: {  
  },  
  created: {  
  }  
  //....  
});
```

# 뷰 인스턴스의 옵션 속성

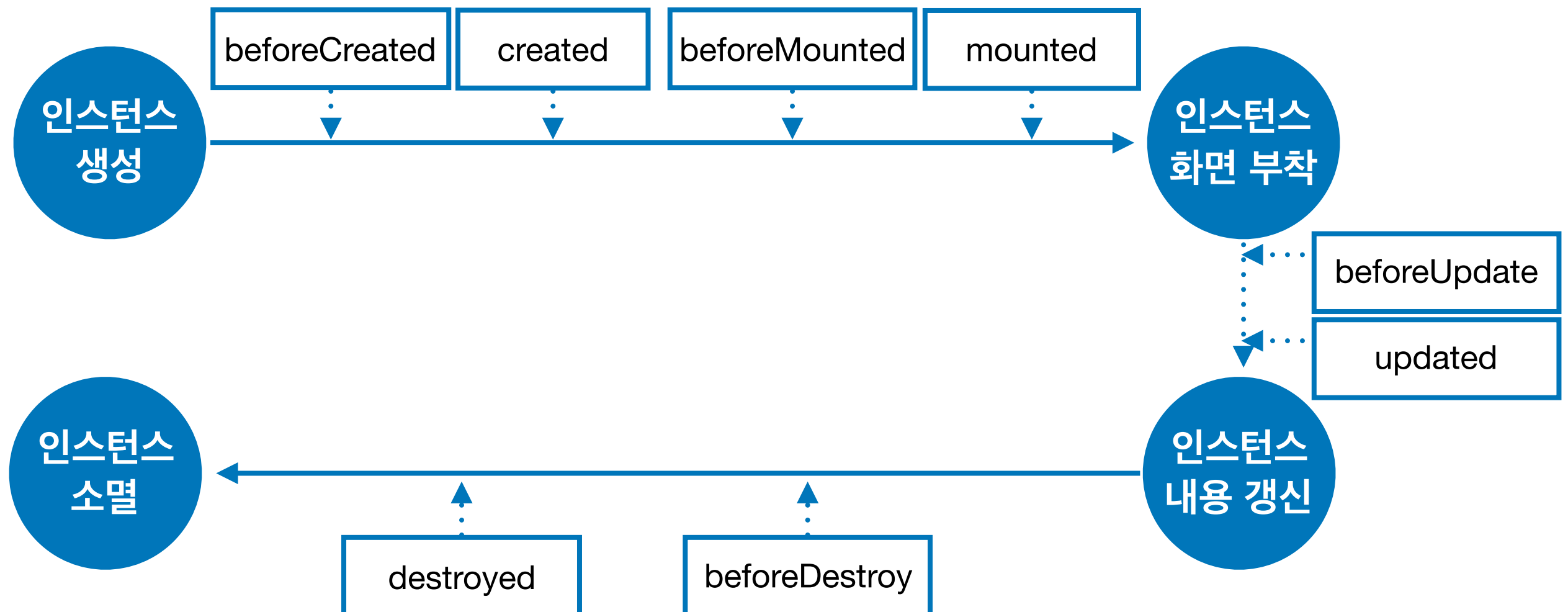
인스턴스를 생성할 때 정의할 속성입니다.

- ✓ el : 뷰 인스턴스를 연결할 HTML DOM 요소를 지정합니다.
- ✓ data : 인스턴스의 상태를 저장하는 속성으로, 템플릿에 바인딩 할 데이터 객체를 지정합니다.
- ✓ template : 화면에 표시할 HTML, CSS 등의 마크업 요소를 정의
- ✓ methods : 화면 로직 제어와 관련된 메서드를 정의
- ✓ created : 뷰 인스턴스의 라이프 사이클 속성으로, 인스턴스가 생성되자마자 실행할 로직을 정의할 수 있는 속성

# 뷰 인스턴스 라이프 사이클

라이프 사이클 속성은 인스턴스의 상태에 따라 호출할 수 있는 속성들입니다.

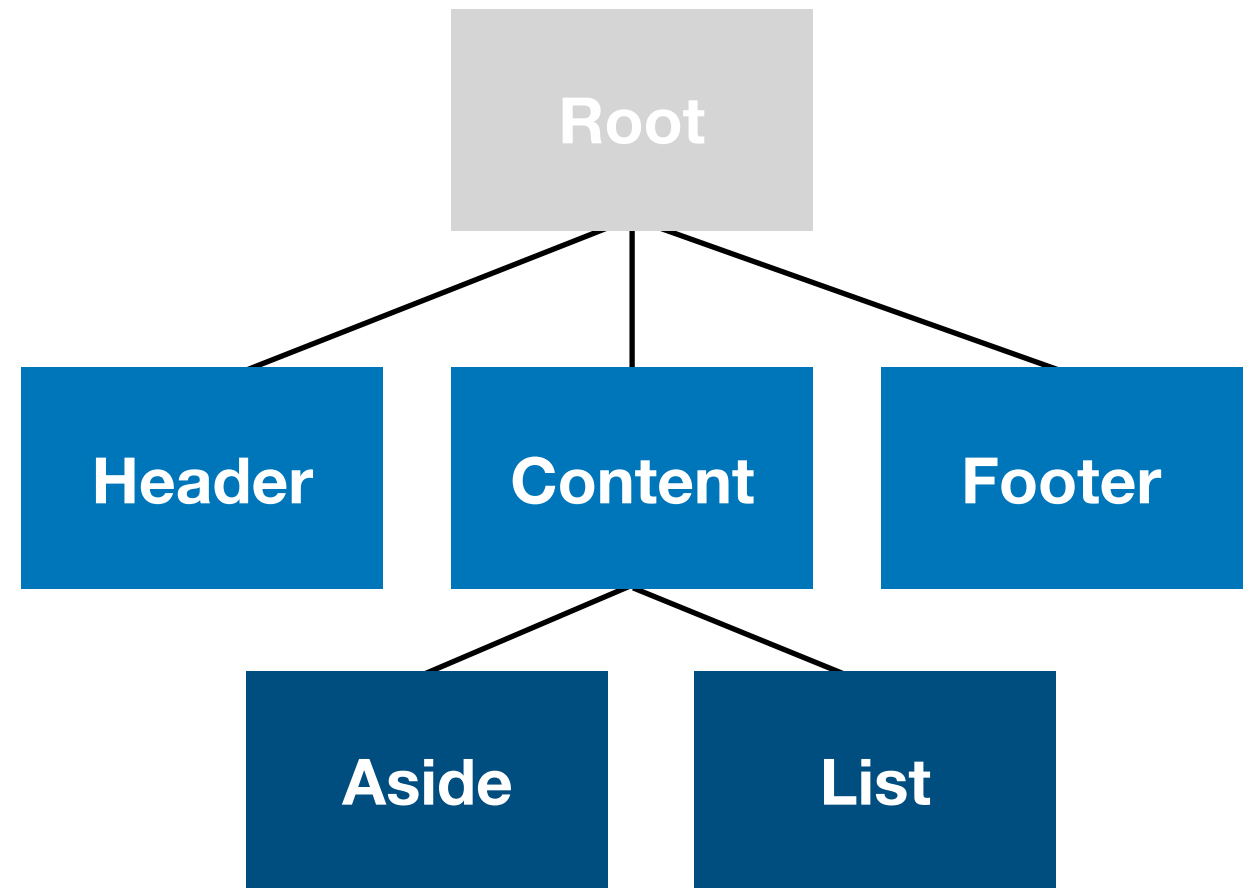
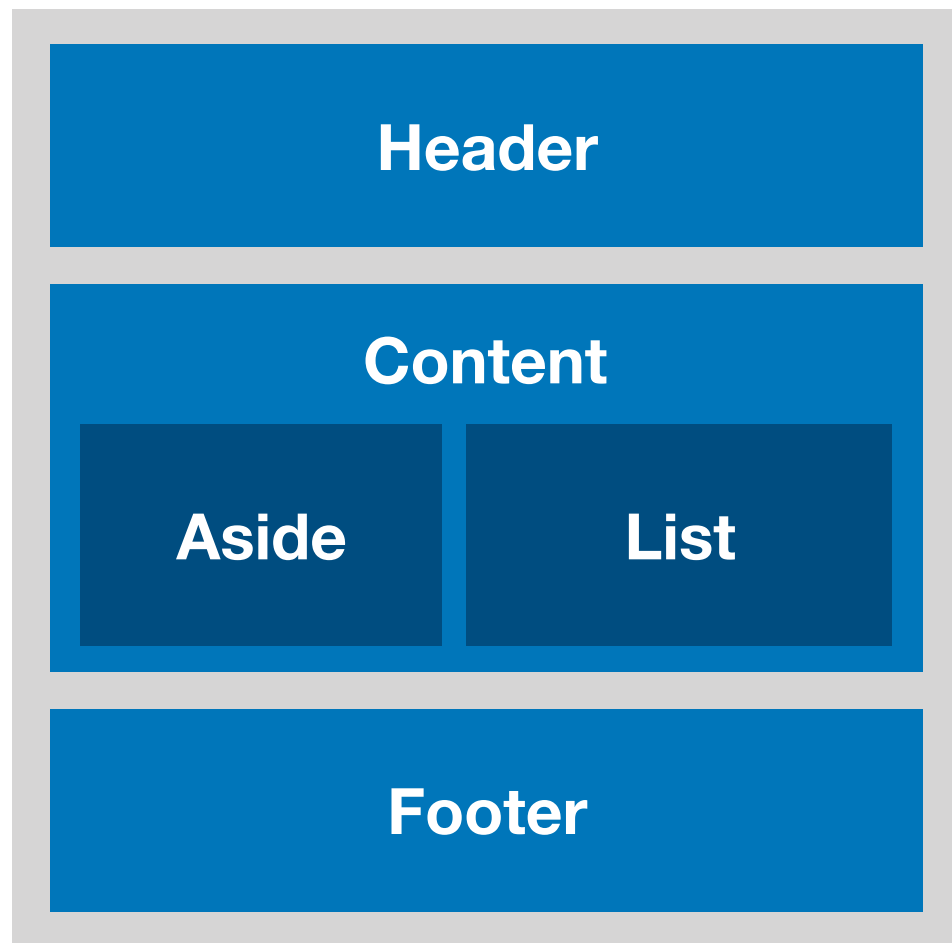
뷰의 라이프 사이클 단계는 크게 인스턴스의 생성, 부착, 갱신, 소멸의 4단계로 나눌 수 있으며, 이와 관련된 라이프 사이클 속성은 총 8개가 있습니다.



# 컴포넌트란?

화면을 구성할 수 있는 블록을 의미합니다.

화면의 영역을 컴포넌트 형태로 관리하면 코드의 재사용이 편리하며, 다른 사람이 작성한 코드를 직관적으로 이해할 수 있습니다.



# 전역 컴포넌트 등록하기

전역 컴포넌트는 여러 인스턴스에서 공통으로 사용할 수 있습니다. 즉, 뷰로 접근 가능한 모든 범위에서 사용할 수 있습니다.

```
<div id="app">
  <h1>전역 컴포넌트 등록</h1>
  <!-- 전역 컴포넌트 표시 -->
  <my-component></my-component>
</div>

<script>
// 전역 컴포넌트 등록
Vue.component('my-component', {
  template: '<div>전역 컴포넌트</div>'
});

new Vue({
  el: '#app'
})
</script>
```

## 전역 컴포넌트 등록 형식

```
Vue.component('컴포넌트 이름', {
  // 컴포넌트 내용
});
```

## 전역 컴포넌트가 등록된 HTML 실제 코드

```
<div id="app">
  <h1>전역 컴포넌트 등록</h1>
  <div>전역 컴포넌트</div>
</div>
```

# 지역 컴포넌트 등록하기

지역 컴포넌트는 특정 인스턴스에서만 유효 범위를 가지므로, 특정 범위 내에서만 사용할 수 있습니다.

```
<div id="app">
  <h1>지역 컴포넌트 등록</h1>
  <local-component></local-component>
</div>

<script>
var cmp = {
  // 컴포넌트 내용
  template: '<div>지역 컴포넌트</div>'
};

new Vue({
  el: '#app',
  // 컴포넌트 속성에 컴포넌트 이름과 내용 정의
  components: {
    'local-component': cmp
  }
})
</script>
```

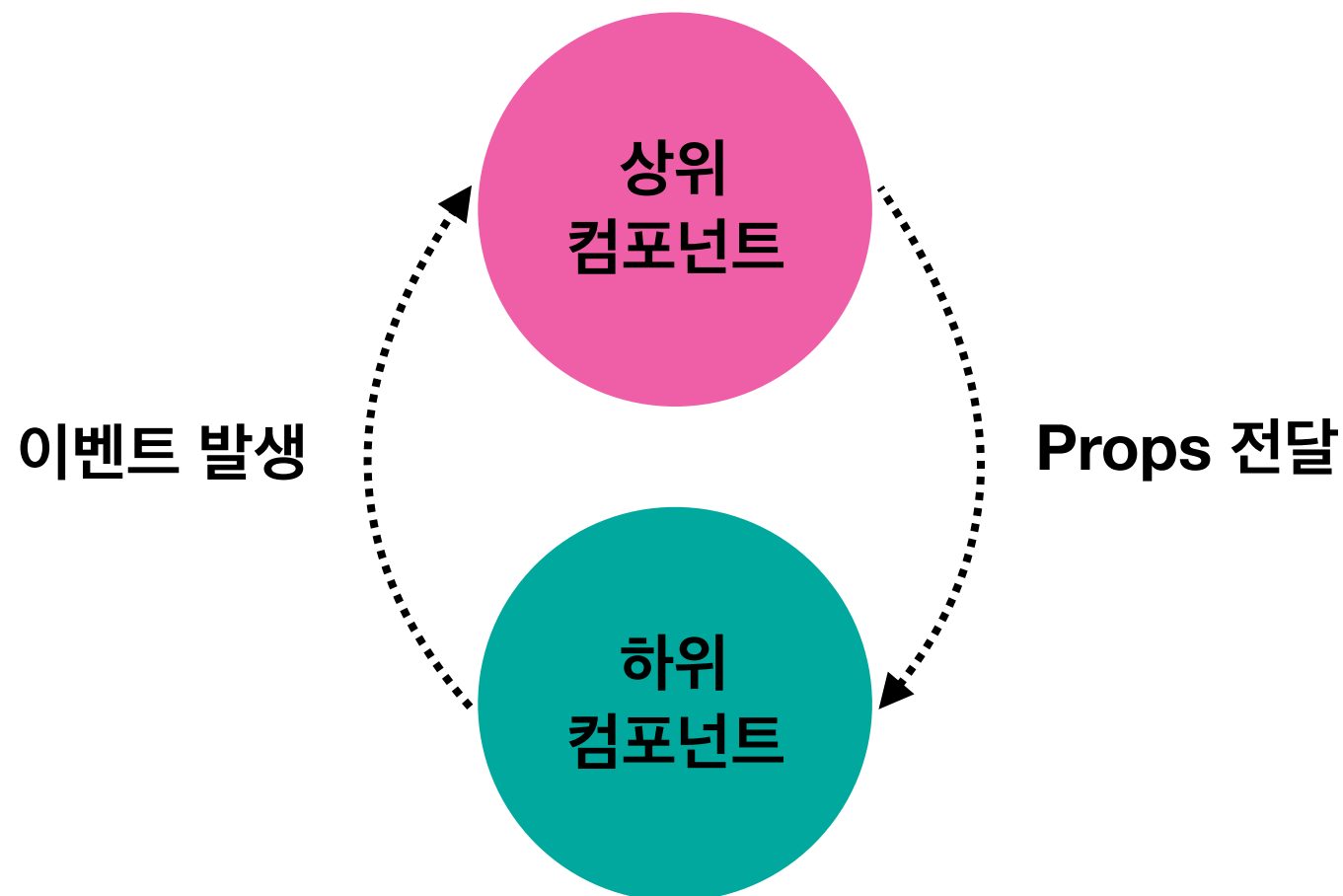
## 지역 컴포넌트 등록 형식

```
new Vue({
  components: {
    '컴포넌트 이름' : 컴포넌트 내용
  }
});
```

# 상위 - 하위 컴포넌트 관계

컴포넌트는 각각 독립적인 유효 범위를 가지기 때문에 직접 다른 컴포넌트의 값을 참조할 수 없습니다.

데이터 전달은 상위에서 하위 컴포넌트로만 가능합니다(단방향 데이터 통신).  
하위에서 상위 컴포넌트로의 통신은 이벤트를 이용합니다.



# 상위에서 하위 컴포넌트로 데이터 전달하기-1

상위 컴포넌트에서 하위 컴포넌트로 데이터를 전달하기 위해 props 속성을 사용합니다.

## props 속성 사용하기

1. 먼저 하위 컴포넌트의 속성에 정의합니다.

```
Vue.component('child-component', {  
  props: ['props 속성 이름']  
})
```

2. 상위 컴포넌트의 HTML 코드에 등록된 child-component 컴포넌트 태그에 v-bind 속성을 추가합니다.

```
<child-component v-bind:props속성이름="상위 컴포넌트의 data 속성">  
</child-component>
```



# 상위에서 하위 컴포넌트로 데이터 전달하기-2

```
<div id="app">  
  <child-component v-bind:propsdata="message"></child-component>  
</div>
```

props속성 이름

상위 컴포넌트의 데이터 속성

```
<script>  
Vue.component('child-component', {  
  props: ['propsdata'],  
  template: '<p>{{ propsdata }}</p>'  
});
```

```
new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello, Vue.js!'  
  }  
});  
</script>
```

# 하위에서 상위 컴포넌트로 이벤트 전달하기-1

하위에서 상위 컴포넌트로는 이벤트를 발생시켜 상위 컴포넌트에 전달하는 방법으로 통신합니다.

## 이벤트 발생과 수신 형식

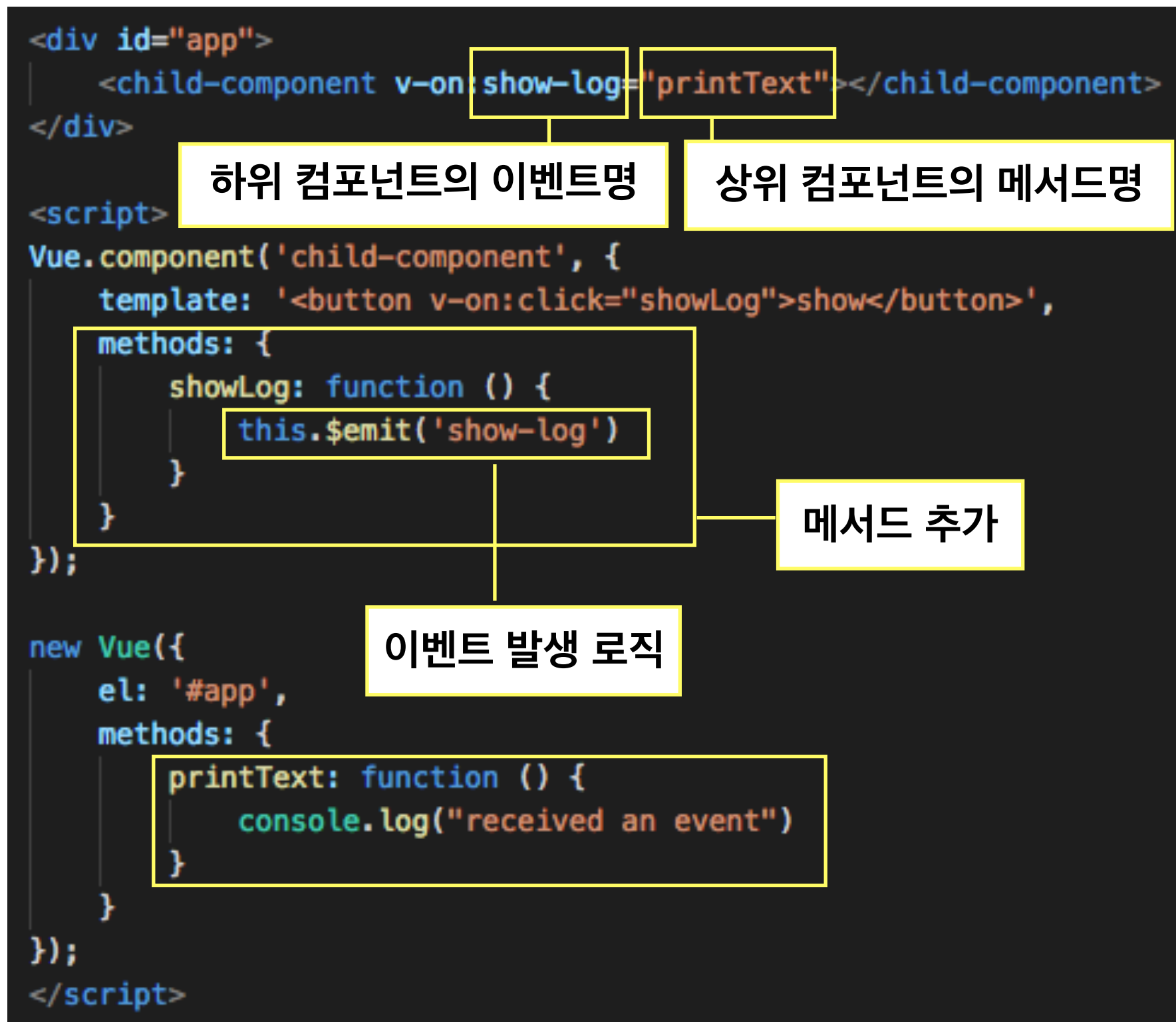
- \$emit()을 이용한 이벤트 발생

```
this.$emit('이벤트명');
```

- v-on: 속성을 이용한 이벤트 수신

```
<child-component v-on:이벤트명="상위 컴포넌트 이벤트명">  
</child-componet>
```

# 하위에서 상위 컴포넌트로 이벤트 전달하기-2



# 뷰 라우터-1

## 라우팅

웹 페이지 간의 이동 방법으로, 싱글 페이지 애플리케이션(SPA)에서 주로 사용됩니다.

## 뷰 라우터

뷰에서 라우팅 기능을 구현할 수 있도록 지원하는 공식 라이브러리입니다.  
뷰 라우터를 이용하여 뷰로 만든 페이지 간에 자유롭게 이동할 수 있습니다.

## 뷰 라우터 구현을 위해 필요한 태그

- `<router-link to="URL값">` : 페이지 이동 태그
- `<router-view>` : 페이지 표시 태그. 변경되는 URL에 따라 해당 컴포넌트를 뿌려주는 영역

# 뷰 라우터-2

```
<div id="app">
  <h1>뷰 라우터 예제</h1>
  <p>
    <router-link to="/main">Main 컴포넌트로 이동</router-link>
    <router-link to="/login">Login 컴포넌트로 이동</router-link>
  </p>
  <!-- URL 값에 따라 갱신되는 영역 -->
  <router-view></router-view>
</div>
```

URL : '/'

## 뷰 라우터 예제

[Main 컴포넌트로 이동](#) [Login 컴포넌트로 이동](#)

URL : '/main'

## 뷰 라우터 예제

[Main 컴포넌트로 이동](#) [Login 컴포넌트로 이동](#)

Main 컴포넌트 입니다.

URL : '/login'

## 뷰 라우터 예제

[Main 컴포넌트로 이동](#) [Login 컴포넌트로 이동](#)

Login 컴포넌트 입니다.

```
<script>
  // Main, Login 컴포넌트 정의
  var Main = { template: '<div>Main 컴포넌트 입니다.</div>' };
  var Login = { template: '<div>Login 컴포넌트 입니다.</div>' };

  // 각 URL에 맞춰 표시할 컴포넌트 지정
  var routes = [
    { path: '/main', component: Main },
    { path: '/login', component: Login }
  ];

  // 뷰 라우터를 생성하고, routes를 삽입하여 URL에 따라 화면이 전환될 수 있도록 정의
  var router = new VueRouter({
    routes
  });

  var app = new Vue({
    router
  }).$mount('#app');
</script>
```

# 뷰 템플릿

HTML, CSS 등의 마크업 속성과 뷰 인스턴스에서 정의한 데이터 및 로직들을 연결하여 사용자가 브라우저에서 볼 수 있는 형태의 HTML로 변환해 주는 속성

```
<div id="app"></div>
<script>
new Vue({
  el: '#app',
  data: {
    message: 'Vue.js!'
  },
  template: '<h1>Hello {{ message }}</h1>'
})
</script>
```

..... ES5에서 뷰 인스턴스의 template 속성 활용

```
<template>
  <h1>Hello {{ message }}</h1>
</template>

<script>
export default {
  data() {
    return {
      message: 'Vue.js!'
    }
  }
}
</script>
```

싱글 파일 컴포넌트 체계에서 template 속성 사용 .....

# 데이터 바인딩

HTML 요소를 뷰 인스턴스의 데이터와 연결하는 것을 의미합니다.  
{{ }} 문법과 v-bind 속성을 사용합니다.

## {{ }} 콧수염 문법

```
<div id="app">
  {{ message }}
</div>

<script>
new Vue({
  el: '#app',
  data: {
    message: 'Vue.js!'
  }
})
</script>
```

## v-bind

```
<div id="app">
  <p v-bind:id="idA">아이디 바인딩</p>
  <p v-bind:class="classA">클래스 바인딩</p>
  <p v-bind:style="styleA">스타일 바인딩</p>
</div>

<script>
new Vue({
  el: '#app',
  data: {
    idA: 10,
    classA: 'container',
    styleA: 'color: blue'
  }
})
</script>
```

# 디렉티브

HTML 태그 안에 v- 접두사를 가지는 모든 속성들을 의미합니다.

## 주요 디렉티브

- ✓ v-if : 뷰 데이터의 참, 거짓 여부에 따라 HTML 태그를 표시하거나 표시하지 않습니다.
- ✓ v-for : 뷰 데이터의 개수만큼 HTML 태그를 반복 출력합니다.
- ✓ v-show : v-if와 유사합니다. v-if는 해당 태그를 완전히 삭제, v-show는 css 속성을 이용하여 화면 상으로만 보이지 않습니다.
- ✓ v-bind : HTML 태그의 기본 속성과 뷰 데이터 속성을 연결합니다.
- ✓ v-on : 화면 요소의 이벤트를 감지하여 처리할 때 사용합니다.
- ✓ v-model : 폼에 입력한 값을 뷰 인스턴스의 데이터와 즉시 동기화 합니다.  
<input>, <select>, <textarea> 태그에만 사용할 수 있습니다.



# Content 4

## **Vue CLI**

# 싱글 파일 컴포넌트 체계

.vue 파일로 프로젝트 구조를 구성하는 방식입니다. 확장자 .vue 파일 1개는 뷰 애플리케이션을 구성하는 1개의 컴포넌트 입니다.

## .vue 파일의 기본 구조

```
<template>
  <!-- HTML 태그 내용 -->
  <header>
    <h1>싱글 파일 컴포넌트 구조</h1>
  </header>
</template>

<script>
export default {
  // 뷰 컴포넌트의 내용을 정의하는 영역
}
</script>

<style>
  /* CSS 스타일 내용 */
</style>
```

싱글 파일 컴포넌트 체계를 사용하기 위해서는 .vue 파일을 웹 브라우저가 인식할 수 있는 형태의 파일로 변환해 주는 웹팩과 같은 도구가 필요합니다.

뷰에서는 편하게 프로젝트를 구성할 수 있는 CLI(Command Line Interface) 도구를 제공합니다. CLI는 커맨드 창에서 명령어로 특정 동작을 수행할 수 있는 도구입니다.

# 뷰 CLI로 프로젝트 생성하기

- ✓ 뷰 CLI 설치

\$ npm install vue-cli -global

- ✓ 프로젝트 생성

\$ vue init webpack-simple <프로젝트명>

- ✓ 관련 라이브러리 설치

\$ npm install

- ✓ 프로젝트 구동

\$ npm run dev

- ✓ 라우터 모듈 설치

\$ npm install -d vue-router

# webpack-simple 프로젝트 폴더 기본 구조

## node\_modules

npm install로 다운받은 라이브러리가 존재하는 위치

## src

.vue 파일과 애플리케이션 동작에 필요한 로직이 들어갈 위치

## index.html

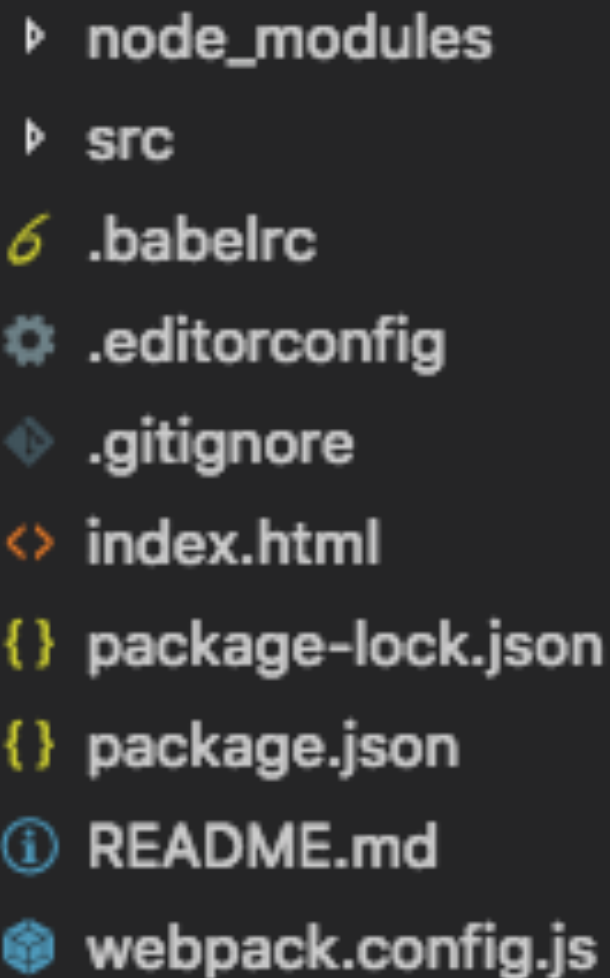
뷰로 만든 웹 앱의 시작점. npm run dev 실행시 로딩되는 파일

## package.json

npm 설정 파일. 뷰 애플리케이션이 동작하는 데 필요한 라이브러리들을 정의하는 파일

## webpack.config.js

웹팩 설정 파일. 웹팩 빌드를 위해 필요한 로직을 정의하는 파일



- ▶ node\_modules
- ▶ src
- .babelrc
- .editorconfig
- .gitignore
- index.html
- package-lock.json
- package.json
- README.md
- webpack.config.js

# Content 5

## **애플리케이션 만들기**

# 애플리케이션 구성-1

연락처 리스트 페이지 ( URL : / )

## 연락처 관리

loglevel:webpack-dev-server 📞 INFO

이하영 📞 010-5555-6666

+

copyright.

연락처 등록 페이지 ( URL : /register )

## 연락처 관리

이름

전화번호

🔄 돌아가기

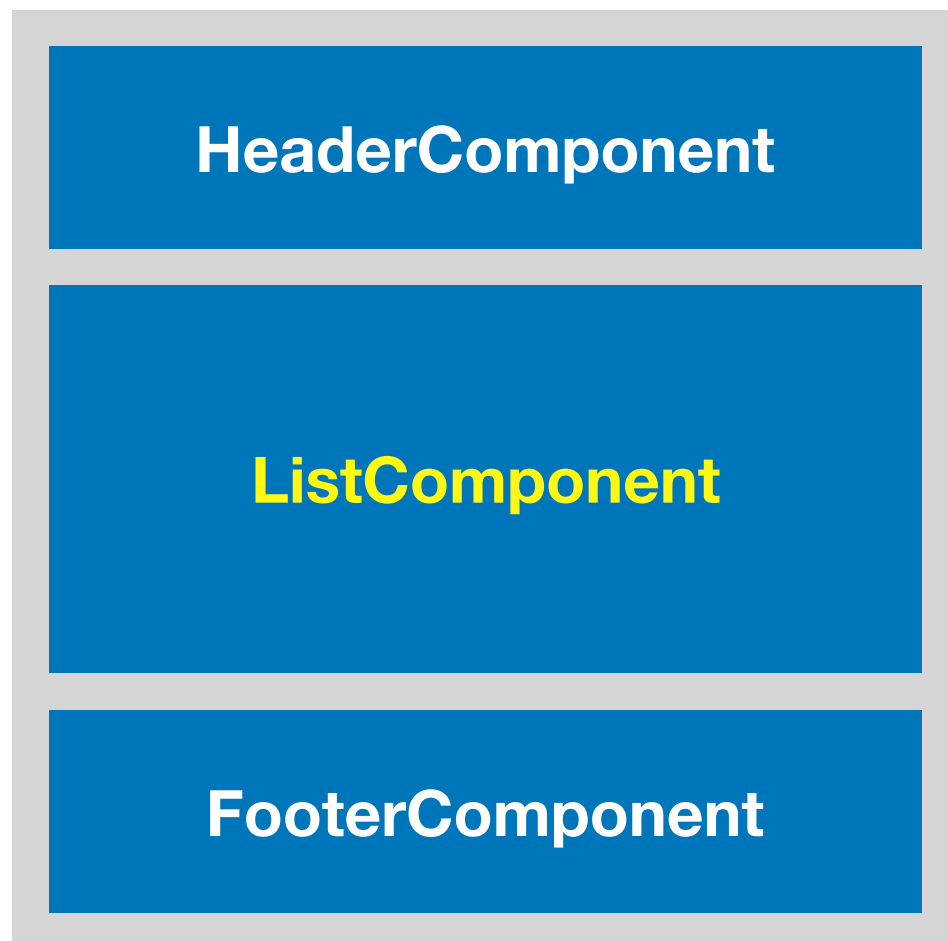
등록하기

copyright.

# 애플리케이션 구성-2

header, footer 컴포넌트를 공통으로 사용하며,  
URL이 변경될 때마다 Section 컴포넌트의 내용을 변경합니다.

**URL : /**



**URL : /register**

