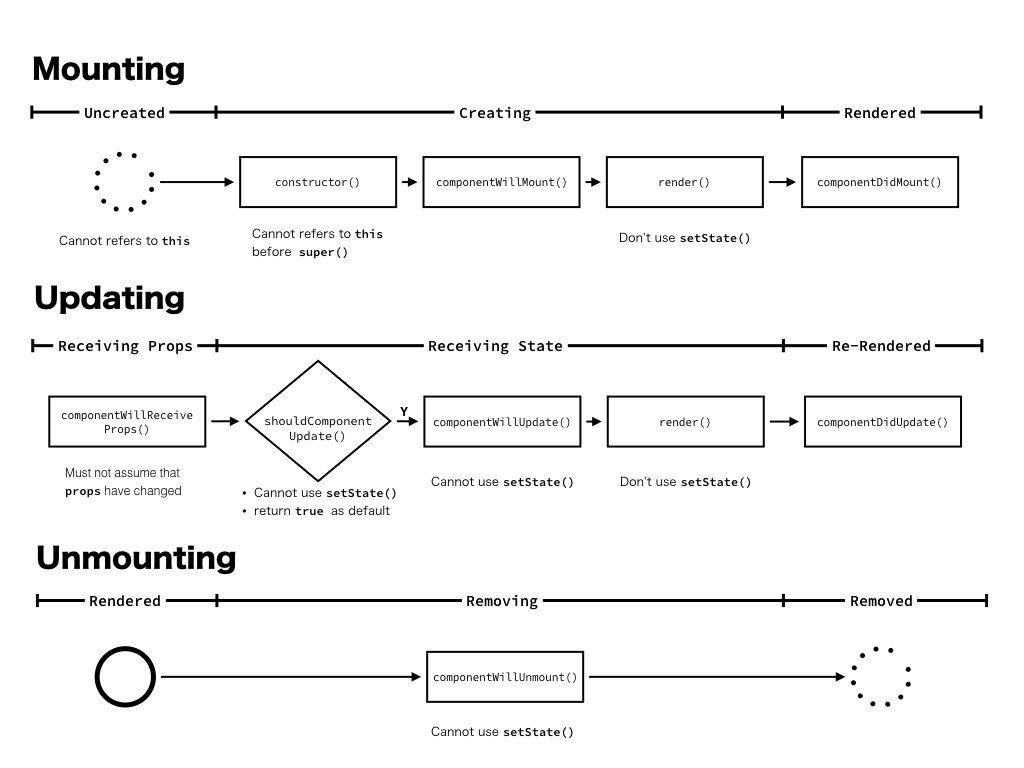
**1. 컴포넌트 설계가 갖는 장점은 무엇인가. 컴포넌트란 무엇인가. 컴포넌트 설계는 어떻게 해야 하는가. 컴포넌트의 리랜더링은 언제 이루어지는가.**

**컴포넌트(Component)**

리액트 프로젝트에서 특정 부분이 어떻게 생길지 정하는 선언체가 있는데, 이를 컴포넌트(Component)라고 한다. 컴포넌트는 개념적으로 props를 input으로 하고 UI가 어떻게 보여야 하는지 정의하는 [React Element](https://facebook.github.io/react/blog/2015/12/18/react-components-elements-and-instances.html)를 output으로 하는 함수입니다. 따라서 합성을 이용하여 “UI를 재사용할 수 있고 독립적인 단위로 쪼개어 생각”할 수 있게 합니다.

UI를 구성하기 위해서는 화면에 컴포넌트를 그리고(Mounting), 갱신하고(Updating), 지워야(Unmounting) 합니다. 컴포넌트는 각 프로세스가 진행될 때에 따라 Lifecycle 함수로 불리는 특별한 함수가 실행됩니다. 개발자는 이를 재정의하여 컴포넌트를 제어하게 됩니다. 그러므로 Lifecycle 함수들을 완전하게 이해해야 합니다. 프로세스와 세부 프로세스, 그리고 각 프로세스에 대응하는 Lifecycle 함수들은 아래 다이어그램을 통해 쉽게 파악할 수 있습니다.



위 개념을 이해하면 쉽게 저지르는 실수를 줄일 수 있는데, 예를 들면 다음과 같습니다.

Mouting: Creating 중인 componentWillMount()에서 Ajax 요청을 날리면 응답 시간만큼 컴포넌트를 그리는 것이 늦어짐을 알 수 있습니다. 따라서 일반적으로 componentDidMount()에서 Ajax 요청을 하는 게 낫다는 것을 알 수 있습니다.

Updating: Receiving State 중에 setState() API를 호출하면 프로세스가 끝난 후 또다시 Receiving State 할 것을 알 수 있습니다. 따라서 setState() API를 해당 Lifecycle 함수에서 호출하면 개념적으로 무한 루프에 빠질 수밖에 없다는 것을 알 수 있습니다. (물론 실제로도 무한 루프에 빠지게 됩니다.)

**초기 렌더링 - render() {...}**

리액트의 초기 렌더링 함수로 **컴포넌트의 외형을 정의**하는 역할을 한다.

html 형식의 문자열을 반환하지 않고, 뷰가 어떻게 생겼고 어떻게 작동하는지 정보를 지닌 객체를 반환한다. 렌더링 작업이 끝나면 지니고 있는 정보를 사용하여 HTML 마크업(markup)을 만들고, 이를 우리가 정하는 실제 페이지의 DOM 요소 안에 주입한다.

**리렌더링 - render() {...}**

업데이트의 과정으로 컴포넌트의 **데이터에 변화가 있을 경우** 새로운 요소로 대체한다.

이 과정 또한 render() 함수가 처리를 한다. 초기 렌더링 이후 데이터가 변경될 경우 새로운 데이터를 가지고 render() 함수를 다시 호출한다. 하지만 render 함수가 반환하는 결과를 DOM으로 바로 반영하지 않고, 이전 render 함수가 만들었던 컴포넌트 정보와 현재 render 함수가 만든 컴포넌트 정보를 비교한다. 자바스크립트를 사용하여 두 가지 뷰를 비교한 후 둘의 차이를 알아내 DOM 트리를 업데이트한다. 결국 방식 자체는 루트 노드부터 시작하여 전체 컴포넌트를 처음부터 다시 렌더링하는 것처럼 보이지만 최적의 자원을 사용하여 이를 수행한다.

출처

<https://medium.com/little-big-programming/react%EC%9D%98-%EA%B8%B0%EB%B3%B8-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8%EB%A5%BC-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90-92c923011818>

<http://egloos.zum.com/bengmaru/v/6444977>

**2. 리액트와 뷰의 차이점과 공통점은 무엇인가.**

**차이점**

**Vue**

**단일 파일 컴포넌트(Single File Component)**

Vue는 단일 파일 컴포넌트 기반의 프레임워크입니다. 웹의 뷰(view)를 구성하는 요소인 HTML, CSS, JavaScript 코드를 .vue 확장자를 가진 하나의 파일에 모두 정의하는 방식입니다. 이러한 관리 방식은 적당한 규모의 프로젝트에서 관리의 생산성을 높이고, 협업을 수월하게 한다는 장점이 있습니다.

**HTML 기반 템플릿 구문**

Vue는 개발한 프론트엔드 파일을 사용자가 볼 수 있도록 브라우저 화면에 렌더링하는 과정에 템플릿이란 문법을 사용합니다. 이 템플릿을 구성하는 문법이 HTML 기반으로 이뤄져 있어 배우기 쉽다는 특징이 있습니다.

**React**

**거대한 커뮤니티**

페이스북의 지원을 받는 만큼 지속적인 버전 관리가 이루어지고, React 사용자가 많아 다양한 레퍼런스, 확장 라이브러리가 많습니다. 또한 이미 React를 사용하고 있는 개발 프로젝트가 많기 때문에 개발자 채용 시, React를 다룰 줄 안다는 것이 장점으로 작용할 수 있습니다.

**JSX 기반 컴포넌트 구문**

템플릿 방식으로 관리하는 Vue와는 달리, React는 JSX 코드로 컴포넌트를 작성하고 컴포넌트의 상태(State)를 변화시키지 않고 관리합니다. 변화가 일어나면 실제 브라우저의 DOM에 새로운 것을 적용하는 것이 아니라, 자바스크립트로 이루어진 Virtual DOM에 렌더링을 하고 기존의 DOM과 비교하여 변화가 일어난 곳만 업데이트 합니다.

템플릿은 재구조화가 어렵고, 에러에 취약하지만, React는 상태를 고유 속성대로 유지하기 때문에 대규모 프로젝트에서도 관리가 수월하고 테스트에도 적합합니다.

출처 :<http://library.gabia.com/contents/infrahosting/8284>

**공통점**

**1. 컴포넌트 기반 SPA 라이브러리**React와 Vue 모두, 웹 UI를 작은 컴포넌트 단위로 구성한다. 컴포넌트는 다른 프로젝트에서도 재사용할 수 있고, 컴포넌트 캡슐화와 확장이 가능해 개발이 유연해지는 장점이 있다. 둘다 virtual DOM을 기반으로 하며, tree알고리즘을 통해 변화사항을 찾아낸다.  
때문에 몇가지 제한사항 같은것들도 서로 닮은 점이 있다. 가령 각 컴포넌트의 HTML에 해당하는 코드는 단일 노드로 감싸져있어야 한다던지, 목록을 출력할때 각각의 항목이 구분될 수 있는 고유한 문자열을 할당해주어야 한다던지 (key props) 등.

**2. props, state, lifecycle**둘 다 컴포넌트 단위로 코드를 작성하는 친구들이다보니, 컴포넌트의 기본적인 구성요소도 겹치는 것들이 있다. 대표적인게 props와 state, 그리고 lifecycle 이다.

**3. 라우터, 전역 상태관리 도구**SPA를 통해 여러 페이지를 보여주기 위한 라우터 역시 리액트와 뷰 둘다 존재한다. 차이점이라면 리액트는 [3rd party library](https://reacttraining.com/react-router/web/guides/quick-start)에서 그걸 지원해주고 있고, 그 상황을 지켜보면서 만들어진 뷰는 [공식 라이브러리](https://router.vuejs.org/kr/)도 함께 만들어서 내놨다는 점 정도이다.  
전역 상태관리 도구 역시 비슷하다. 리액트는 제 3자에 의해 [redux](https://redux.js.org/) 같은게 만들어졌고, 후발주자인 vue는 시작할 때 부터 [redux](https://redux.js.org/)와 [redux-saga](https://redux-saga.js.org/)가 합쳐진 형태의 전역 상태관리 도구인 [vuex](https://vuex.vuejs.org/kr/)를 만들어서 공식으로 내놨다.

**4. cli, webpack, ...**둘다 기본 프로젝트 구조를 만들어주는 툴을 cli로 제공하고 있다. webpack 기반으로 번들링 해주는 기능 역시 마찬가지로 존재한다.

**5. mobile application을 위한 Native 지원**둘다 모바일 앱 개발용 도구를 지원한다. 이 부분은 신기하게도 앞서 말했던부분과 정 반대의 관계인게 있다.  
리액트 네이티브는 페이스북에서 공식적으로 지원하는 라이브러리이며, 뷰 네이티브는 서드파티 라이브러리이다. 아무래도 페이스북은 리액트로 모바일 앱을 직접적으로 개발하고 있다보니, 다른건 서드파티한테 맡기더라도 이런건 직접 하는것으로 보인다. **6. SSR Framework**

둘다 서버사이드 렌더링용 웹서버 프레임워크가 있다. webpack 기반 자바스크립트이다보니 둘다 nodeJS 기반이다. 리액트는 nextjs, 뷰는 nuxtjs.

출처 : <https://velog.io/@vraimentres/react-vs-vue-1>

**3. STATE와 PROPS를 설명하시오.**

props와 state는 기본적으로 다른 속성을 지니고 있습니다. 컴포넌트에 대해 영향을 미치는 객체인 것은 동일하나, 그 쓰임새가 전혀 다른 쓰임새를 지닙니다. React의 특징에서 살펴보았듯이, React는 데이터가 부모 컴포넌트에서 자식컴포넌트로 데이터흐름이 이루어지는 단방향 데이터흐름이 특징입니다. 부모 객체는 자식객체에 props 값을 전달하며, props 값을 받은 자식객체는 이에 관한 부분들을 렌더링 하며, state라는 자체 값을 포함하여 데이터를 변경해 주고, 다시 렌더링 해줄 수 있습니다.

**PROPS**

props는 개발에서 많이들 사용되는 용어인 프로퍼티(properties)의 줄임말입니다. props는 React에서는 사용자가 컴포넌트에 전달해서 보관하길 원하는 데이터입니다. 즉, 컴포넌트 내에서 데이터가 보관되면, 이 데이터는 수정되지 않고 보존되어야 하는 법칙이 성립됩니다. 만약 props의 값을 변경하고자 할때에는 컴포넌트 내부가 아닌, 부모 컴포넌트에서 이에 대한 부분이 변경되어야 합니다. props는 컴포넌트의 mounting, updating 프로세스 시점에 값이 할당될 뿐 컴포넌트 내부에서 값을 변경할 수 없습니다

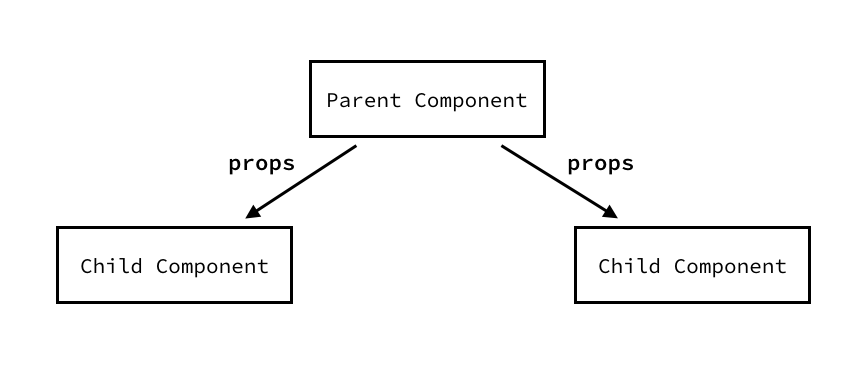
**STATE**

React 컴포넌트는 컴포넌트의 상태를 저장할 수 있습니다. props와의 차이점이라면, state는 컴포넌트 내부에 존재하고 있기 때문에, 상태값 변경이 가능하다는 것입니다.

**PROPS 와 STATE로 구분해서 사용하는 이유**

먼저 개발자들에게 명확한 관념 모델(static mental model)을 제공합니다. 관념 모델은 무엇이 어떻게 동작하는지 이해할 때 진행되는 일련의 사고 프로세스를 의미합니다. 즉, 논리적으로 이치에 맞는 사고 모델을 제공한다는 것이죠.

만약 input으로 들어오는 props를 컴포넌트 내부에서 변경할 수 있다면 어떻게 되어야할까요? props를 내려주는 부모 컴포넌트에도 영향이 가야 할까요? state가 없다면 유저 이벤트에 맞춰 변경돼야 하는 값은 어떻게 관리할까요? 개발자는 이러한 질문에 고민할 필요가 없습니다. 컴포넌트 간에는 무조건 props를 통해서만 데이터를 주고받고 props는 컴포넌트 내부에서 변경되지 않습니다. 따라서 위/아래 양쪽에 대해 동시에 고민할 필요가 없고 아래 한쪽 방향(uni-directional) 그리고 자기 자신에 대해서만 고민하면 됩니다.



계층 기준으로 부모 자식 관계를 표현한 React 컴포넌트 관계도

지금 컴포넌트에서 필요한 값이 props인지 state인지 판단하고 어느 Lifecycle과 관련이 있는지 이 값을 어떤 컴포넌트에 어떻게 넘겨줄지만 생각하여 코드를 작성하면 컴포넌트를 완성할 수 있습니다.

퍼포먼스 측면의 이점도 있습니다. 만약 props와 state가 하나의 객체로 관리된다면, Updating을 할지 결정하는 shouldComponentUpdate() 함수에서 O(keys(props+state))만큼 값이 변경되었는지를 비교해야 하지만, 애초에 props와 state로 분리되어 있으므로 O(keys(state))만큼만 비교하면 Updating을 결정할 수 있습니다.

출처 : <https://medium.com/little-big-programming/react%EC%9D%98-%EA%B8%B0%EB%B3%B8-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8%EB%A5%BC-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90-92c923011818>