ReactJS 1

1. **React란?** UI를 만드는 도구.

- 공식 API에서 "A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES"라고 설명하듯이 UI를 위한 JS 라이브러리로 표현할 수 있다. 하지만 이 설명으로는 조금 부족하다고 느껴진다.

- React는 아래 수식에서 View 함수에 해당한다.

```
UI = View(State)
```

- 중요한 점은 **View를 State가 같다면 항상 같은 UI를 결과로 갖는 함수로 본다**는 것이다.

- 위와 같이 React를 **View 함수 개발에 도움을 주는 라이브러리**로 본다면 다음과 같은 React의 특이자 장점을 자연스럽게 이해할 수 있다.

  - 함수의 정의가 그러하듯 **단방향 사고**를 강제한다.
  - 함수가 그러하듯 **특정 state, props에 따른 render 결과가 바뀌지 않는다.**
  - 함수 내용을 정의하듯 **JSX**를 통해 어떻게 화면을 그릴지 정의한다.
  - 함수 간 합성(Composition)이 가능하듯이 **컴포넌트 간 합성**을 할 수 있다.

2. **Component란?** 재사용 가능한 조각 UI 조각.

- 컴포넌트는 개념적으로 `props`를 input으로 하고 UI가 어떻게 보여야 하는지 정의하는 React Element를 output으로 하는 함수다.

- 따라서 합성을 이용하여 "UI를 재사용할 수 있고 독립적인 단위로 쪼개어 생각"할 수 있게 한다. 그래서 컴포넌트는 `React.Component`를 상속받아 정의하지만 컴포넌트 간에는 상속보다는 합성을 사용하길 권장한다.

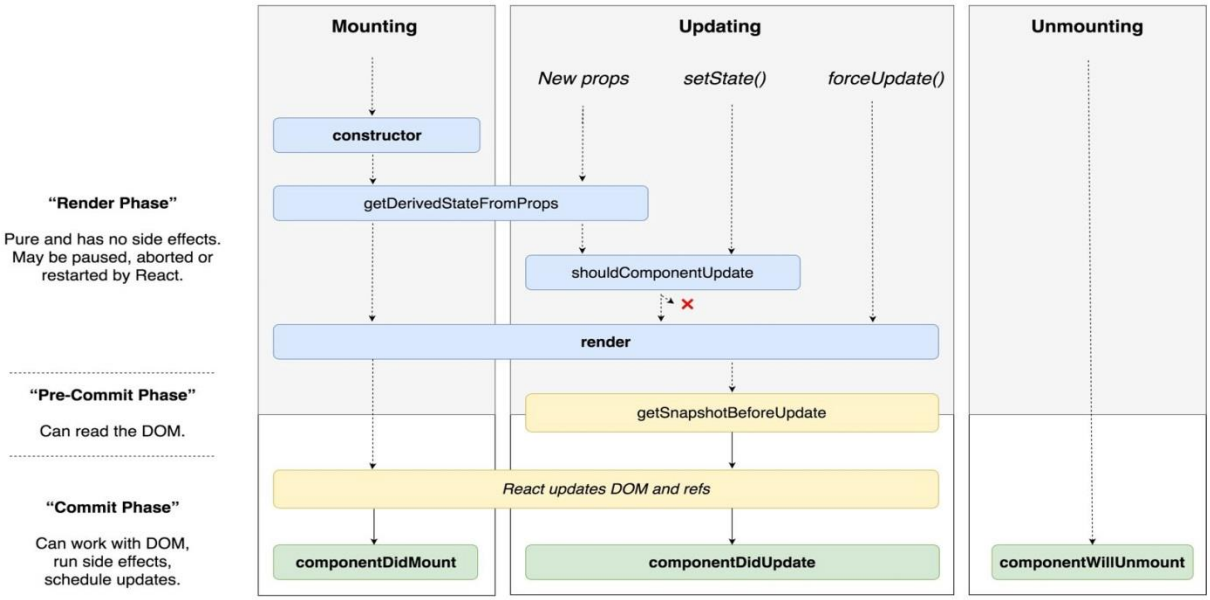
- 컴포넌트는 각 프로세스가 진행될 때에 따라 Lifecycle 함수로 불리는 특별한 함수가 실행된다. UI를 구성하기 위해서는 화면에 컴포넌트를 그리고(Mounting), 갱신하고(Updating), 지워야(Unmounting) 한다.

- 개발자는 이를 재정의하여 컴포넌트를 제어한다. 그러므로 Lifecycle 함수들을 완전하게

이해해야 한다. 프로세스와 세부 프로세스, 그리고 각 프로세스에 대응하는 Lifecycle 함수들은 아래 다이어그램을 통해 쉽게 파악할 수 있다.

@https://hackernoon.com/reactjs-component-lifecycle-methods-a-deep-dive-38275d9d13c0

@ https://www.zerocho.com/category/React/post/579b5ec26958781500ed9955

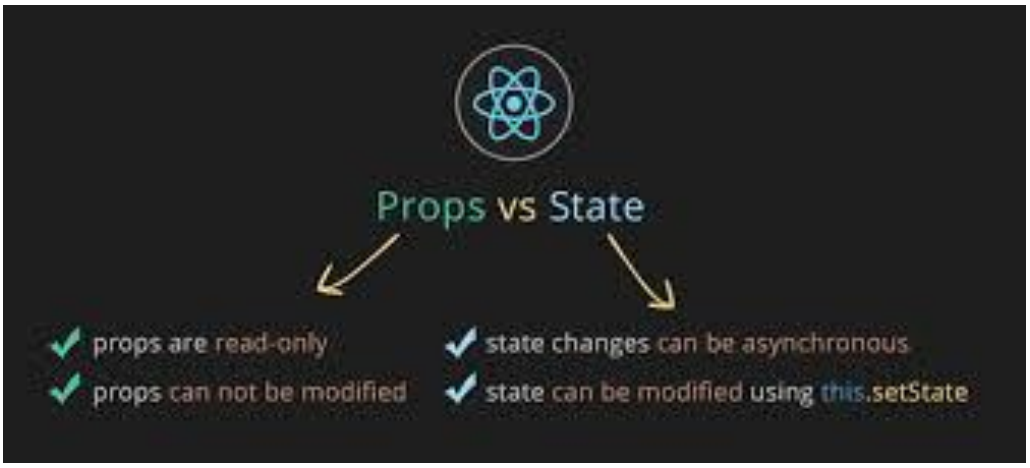


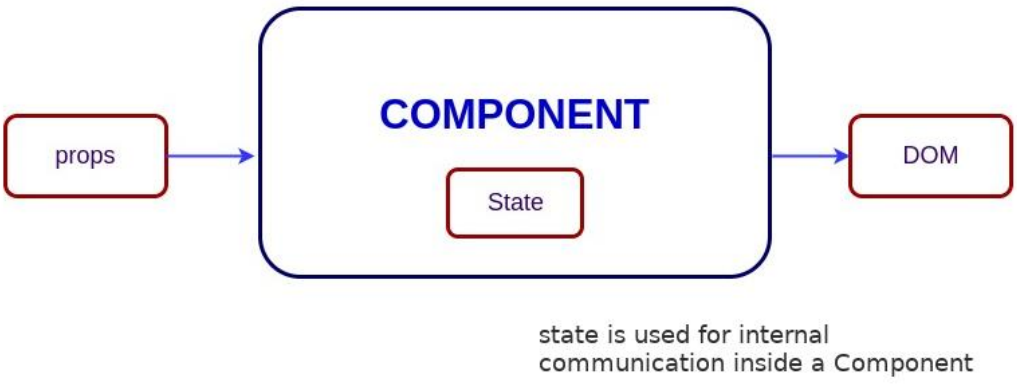
3. State & Props (Component의 두 가지 속성)

- 컴포넌트는 두 가지 인스턴스 속성(property) **props**와 **state**를 가지고 있다.
- `props`는 컴포넌트의 mounting, updating 프로세스 시점에 값이 할당될 뿐 컴포넌트 내부에서 값을 변경할 수 없다.
- 상황에 따라 변경되어야 하는 값들은 `state`를 이용해야 한다.
- **왜 `props`와 `state`로 나누어 사용하도록 설계했을까요? 무슨 이점이 있을까요?**
- 먼저 개발자들에게 **명확한 관념 모델**(static mental model)을 제공한다. 관념 모델은 무엇이 어떻게 동작하는지 이해할 때 진행되는 일련의 사고 프로세스를 의미한다. 즉, **논리적으로 이치에 맞는 사고 모델을 제공한다**는 것이다.
- 만약 input으로 들어오는 `props`를 컴포넌트 내부에서 변경할 수 있다면 어떻게 되어야

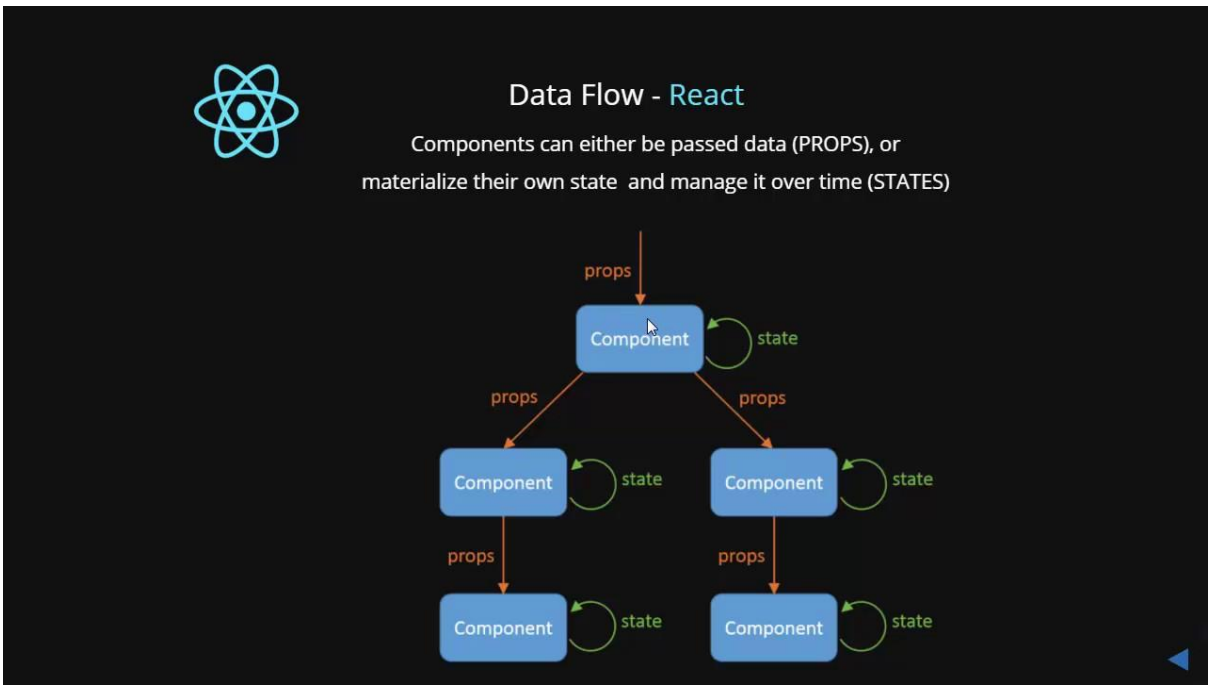
할까? `props`를 내려주는 부모 컴포넌트에도 영향이 가야 할까? `state`가 없다면 유저 이벤트에 맞춰 변경돼야 하는 값은 어떻게 관리할까?

- 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.
- 지금 컴포넌트에서 필요한 값이 `props`인지 `state`인지 판단하고 어느 Lifecycle과 관련이 있는지 이 값을 어떤 컴포넌트에 어떻게 넘겨 줄지만 생각하여 코드를 작성하면 컴포넌트를 완성할 수 있다.
- Props vs State





- 계층 기준으로 부모 자식 관계를 표현한 React 컴포넌트 관계도



- 지금 컴포넌트에서 필요한 값이 `props`인지 `state`인지 판단하고 어느 Lifecycle과 관련이 있는지 이 값을 어떤 컴포넌트에 어떻게 넘겨 줄지만 생각하여 코드를 작성하면 컴포넌트를 완성할 수 있다.

4. create-react-app 개발환경 세팅

- **React설치를 위한 npm 설치**

- https://nodejs.org/ko/ 에서 NodeJS 설치

- NodeJS설치시 npm이 포함된다.

- npm이란? Node Packaged Manager

- package는 모듈이라고도 불리는데 패키지나 모듈은 프로그램보다는 조금 작은 단위의 기능들을 의미한다. npm이라는 것은 NodeJS로 만들어진 pakage(module)을 관리해주는 툴이다. (Java랑 비교를 하자면 Maven과 비슷한 역할을 하는 것 같다.).

- **npm으로 create-react-app 설치**

- cmd 실행

- $ npm install -g create-react-app@2.1.8 (2.1.8버전 전역 설치)

- **react 설치 확인**

- $ create-react-app

- 설치한 React의 정보가 출력되면 설치 정상적으로 완료.

- **React 프로젝트 생성**

- 원하는 경로에 새폴더 생성

- $ cd 해당폴더의경로 (생성한 폴더로 경로 변경)

- $ create-react-app . (. 은 현재 경로를 의미, React 개발환경 설치)

- **react 샘플 웹앱 구동 & 종료**

- $ npm run start

- $ ctrl+c

- **App.js파일의 내용이 Function이라면 Class형식으로 변경**

<code>
import React, { Component} from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render(){
    return (
      ...
    );
  }
}
export default App;
</code>

- **배포하는 법**

- $ npx serve -s build