**<이화이언 5차 수습과제>**

#9강

Conditional Rendering 조건부 렌더링

어떠한 조건에 따라서 렌더링이 달라지는 것

js - truthy(true는 아니지만 true로 여겨지는 값) ... {}, [], 0아닌 수, “0”, “false”

- falsy(flase는 아니지만 flase로 여겨지는 값)... flase, 0, -0, 0n, ‘’, ””, null, undefined, NaN(숫자x)

Element Varibles - 엘리먼트를 변수처럼 다룸

-> 간결 ->

Inline Conditions - 조건문을 코드 안에 집어 넣는것

if문의 경우 - &&연산자 사용

true && expression -> expression

false && expression -> false

Inline If-Else

? 연산자를 사용

condition ? true:false

Ex) {props.isLoggedIn ? ‘로그인’ : ‘로그인하지 않은’}

Component 렌더링 막기

null을 리턴

\*class coponent 함수에서 널을 리턴 컴포넌트의 생명주기 함수 영향x

#10강

List 목록

Array 배열 js의 변수나 객체들을 하나의 변수로 묶어 놓은 것

열쇠 - 각 개체나 아이템을 구분할 수 있는 고유한 값

여러 개의 Component 렌더링

같은 코드 반복 -> 불필요

화면 바뀜 -> 까다로움

=> map()

in react // key: 아이템을 구분하기 위한 고유한 문자열

key의 값 -> 같은 List에 있는 Elements 사이에서만 고유한 값이기만 하면 됨!

Ex] key로 값을 사용하는 경우

key={number.toString()}

key로 id를 사용하는 경우

key={todo.id}

key로 index를 사용하는 경우

key={index}

map() 함수 안에 있는 Elements는 꼭 key가 필요!

#11강

From: 양식

-> 사용자로부터 입력을 받기 위해서 사용함

Controlled Components

:값이 리액트의 통제를 받는 Input Form Element

모든 데이터를 state에서 관리(vs HTML Form : 자체적으로 state 관리)

사용자의 입력을 직접적으로 제어 가능

Textarea 태그(controlled components 방식)

:여러 줄에 걸쳐 긴 텍스트를 입력받기 위한 HTML 태그

Select 태그

:Drop-down 목록을 보여주기 위한 HTML 태그

option태그가 감쌈

여러 개의 옵션 선택 가능!

File input 태그( u.c )

:디바이스의 저장 장치로부터 하나 또는 여러 개의 파일을

선택할 수 있게 해주는 HTML 태그

Uncontrolled Component

: 값이 리액트의 동작을 받지 않음

Multiple Inputs

: 여러 개의 state를 선언하여 각각의 입력에 대해 사용

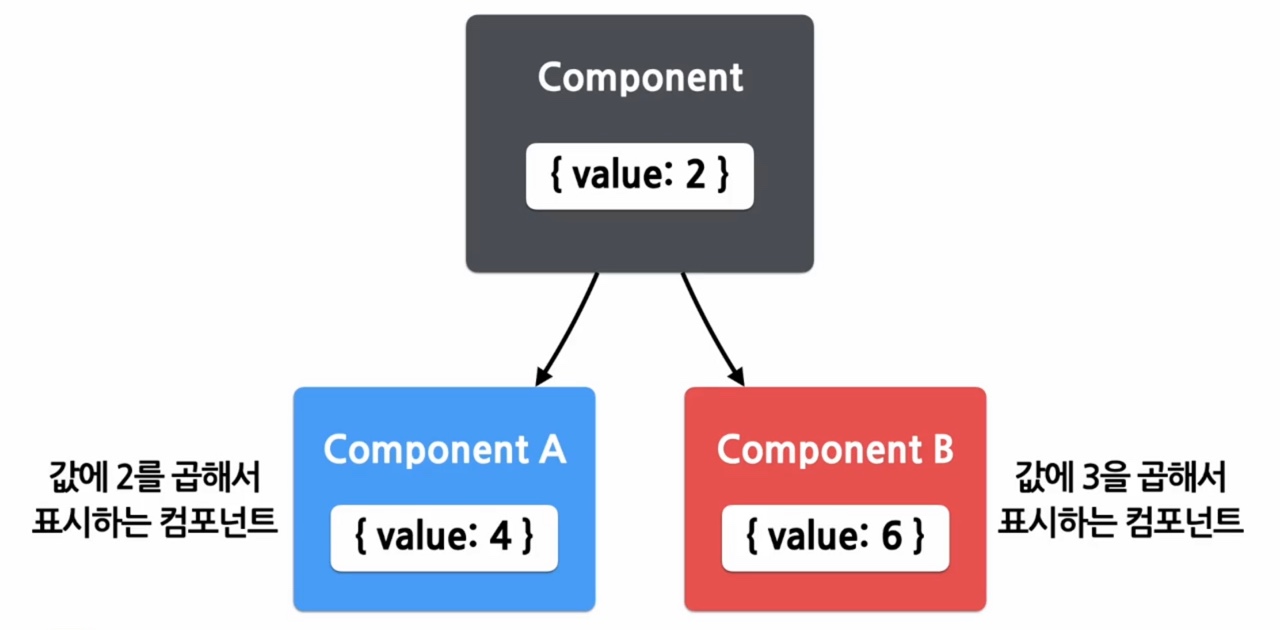
Input Null value

: 자유롭게 입력받고 싶을때 사용

#12강

shared state

: state에 있는 데이터를 여러 개의 하위 컴포넌트에서 공통적으로 사용하는 경우



하위 컴포넌트가 공통된 부모 컴포넌트의 state를 공유하여 사용함

#13강

Composition(합성)

:여러 개의 컴포넌트를 합쳐서 새로운 컴포넌트를 만드는 것, 많이 사용됨

어떻게 조합할것인가?

1. containment(담다, 포함하다)  
 :하위 컴포넌트를 포함하는 형태의 합성 방법

Sidebar나 Dialong 같은 Box형태의 컴포넌트는 자신의 하위 컴포넌트 미리 알 수 없음 -> 이때 사용

children이라는 prop을 사용해서 조합함

if 여러개의 children? -> 별도의 props 조합해서 사용

2. specialization(전문화, 특수화)

:범용적인 개념을 구별이 되게 구체화 하는 것

리액트에서는 합성을 사용하여 specializaion 구현

(vs 기존의 객체지향 언어: 상속 사용하여 구현)

Inheritance(상속)

:다른 컴포넌트로부터 상속을 받아서 새로운 컴포넌트를 만드는 것

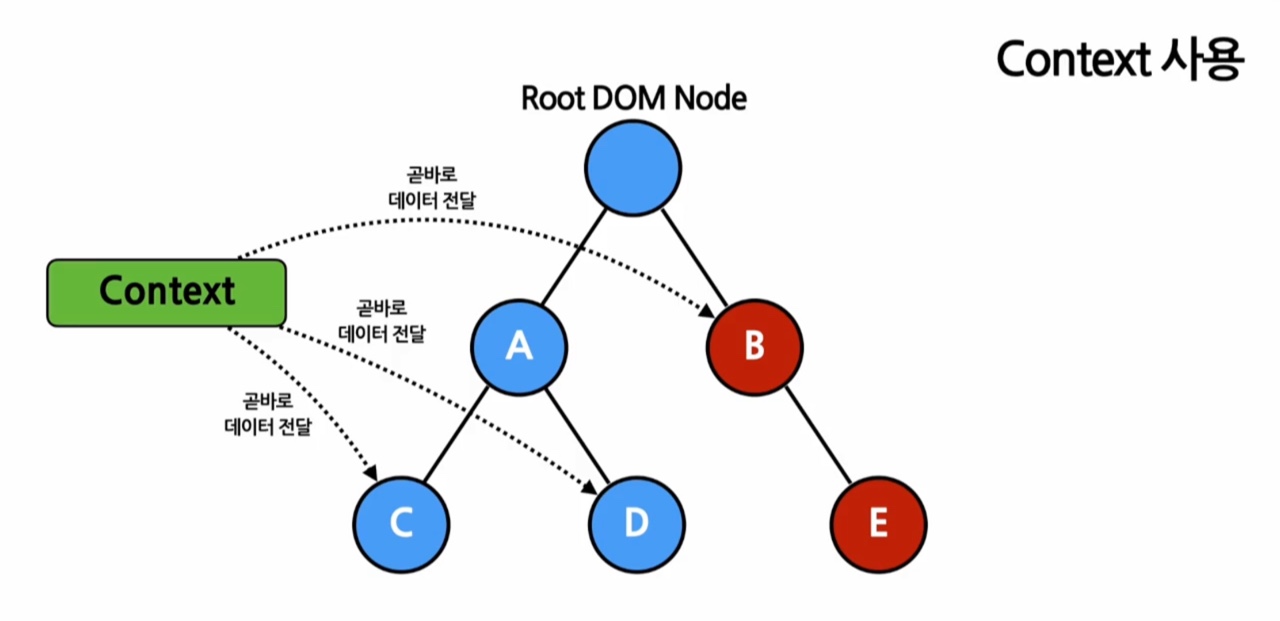
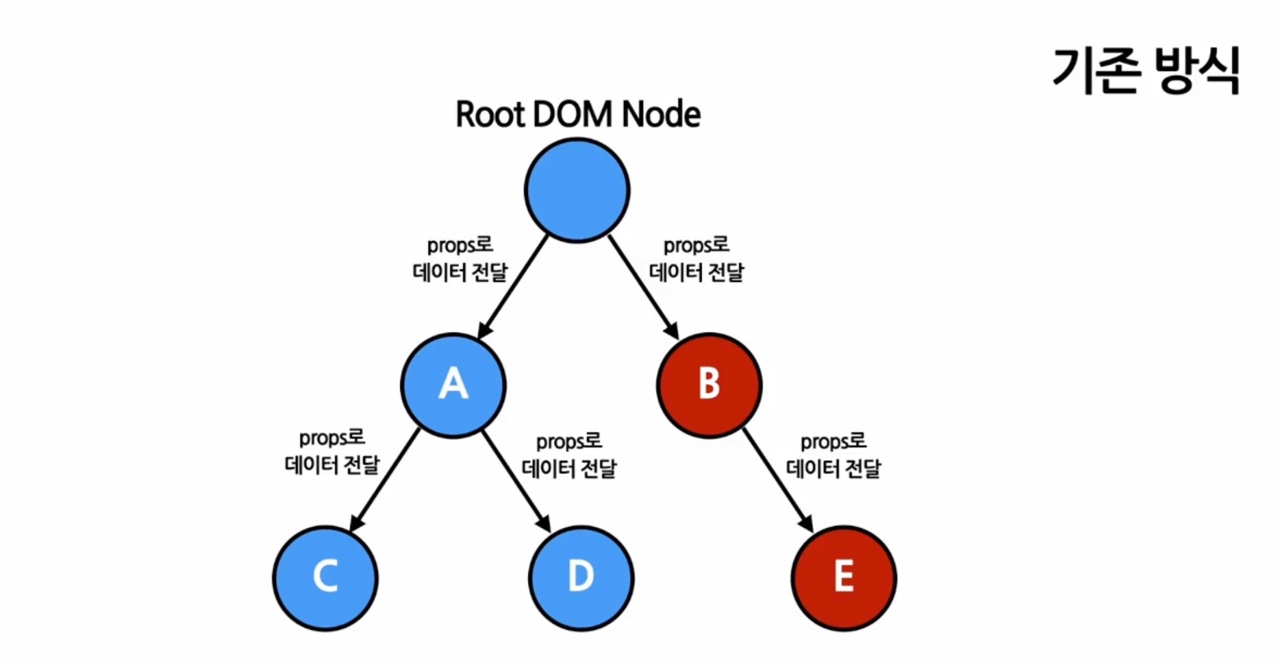
-> 리액트에서 비추천

복잡한 컴포넌트를 쪼개서 여러 개의 컴포넌트로 만들고, 만든 컴포넌트들을 조합해서 새로운 컴포넌트들을 조합해서 새로운 컴포넌트를 만들자!

#14강

context?

:리액트에서는 컴포넌트의 props를 통한 데이터 전달, 데이터 접근 용이



-언제?

로그인 여부, 로그인 정보, UI테마, 현재 언어 등..

-고려해야할 점?  
컴포넌트와 컨텍스트가 연동되면 재사용성 떨어짐

->다른 레벨의 많은 컴포넌트가 필요한 것이 아니라면 기존방식 사용!

-context 생성

-> React.createContext()

if 상위 레벨에 매칭되는 Provider X ? -> 기본값 사용됨

(기본값으로 undefined를 넣으면 기본값 사용X)

-context.Provider

Provider 사용

컨슈밍 컴포넌트: 컨텍스값의 변화를 지켜보다가 값이 변경되면 재렌더링됨

-Provider Value에서 주의해야할 것?

Provider 컴포넌트가 재렌더링될 때마다 모든 하위 consumer 컴포넌트가 재렌더링 됨

=> state를 사용항 재렌더링을 막음

-Class.contextType

프로바이더 하위에 있는 클래스 컴포넌트에서 컨텍스트 데이터에 접근하기 위해 사용, 요즘 --

-Context.Consumer

:컨텍스트의 데이터를 구독하는 컴포넌트

컨텍스트의 값에 따라서 컴포넌트들을 렌더링함

-Context.displayName

디스플레이 네임이라는 문자열 속성을 가짐

-여러 개의 context?

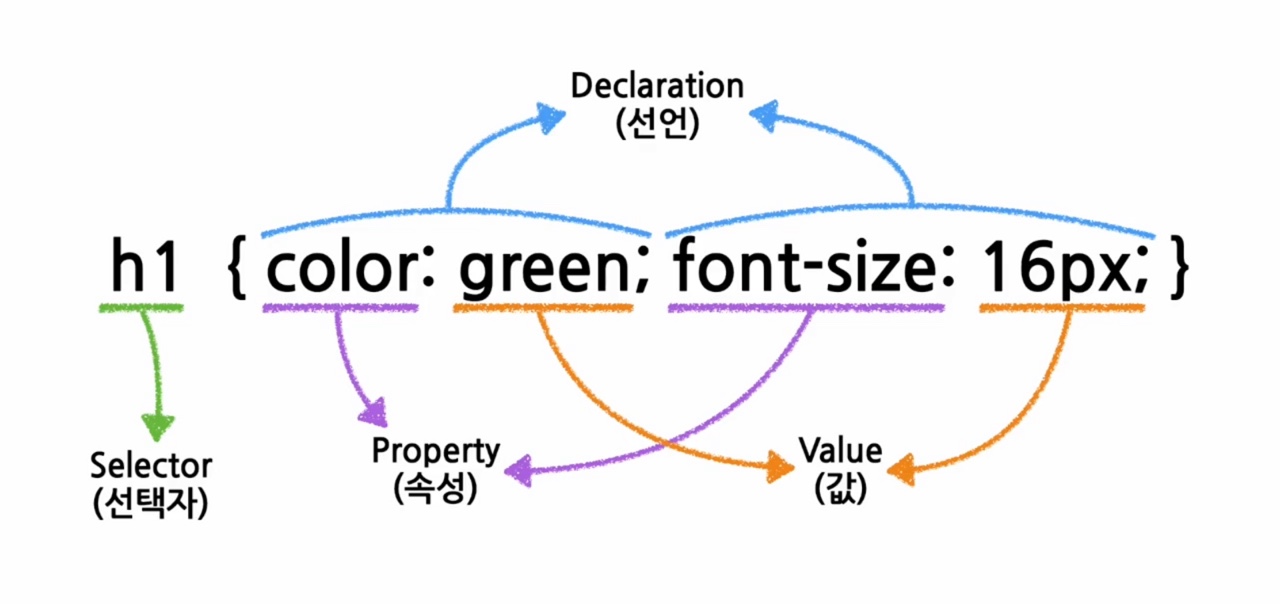
중첩해서 사용

-useContext() 훅 -> 편리

#15강

-CSS (cascading style sheets) 우선권을 가지고 적용

Styling을 위한 언어



-Selector의 유형

1. Elemnet Selcector

2. ID selector

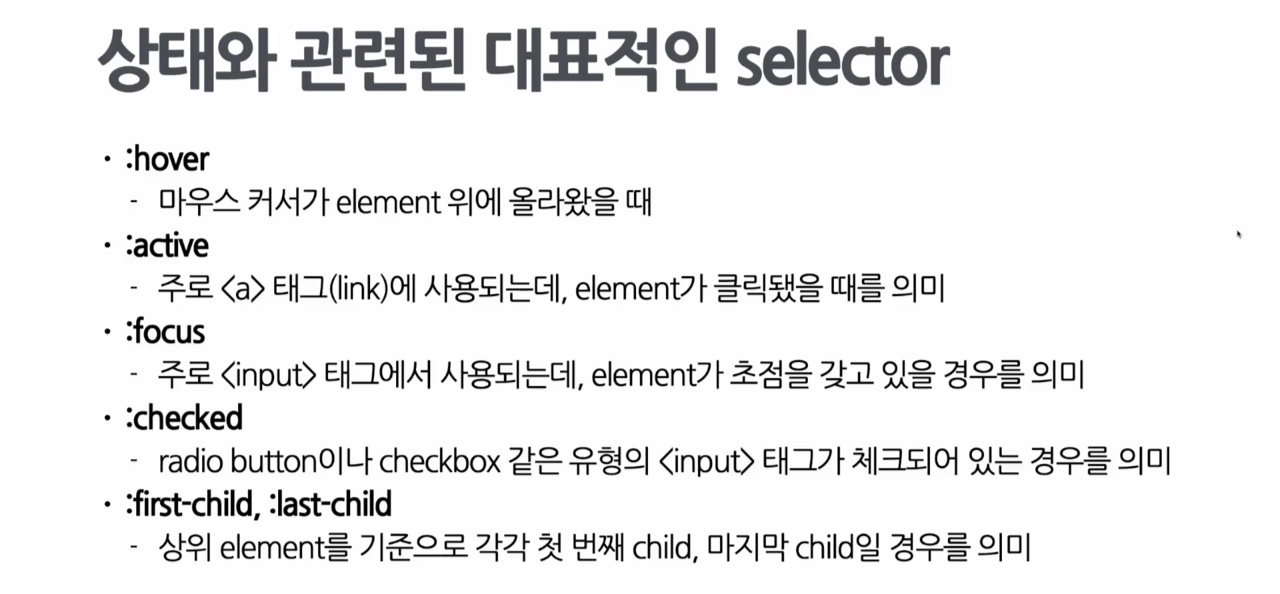
3. Class Selector

4. Universal selector

\*

5. Grouping selecotr

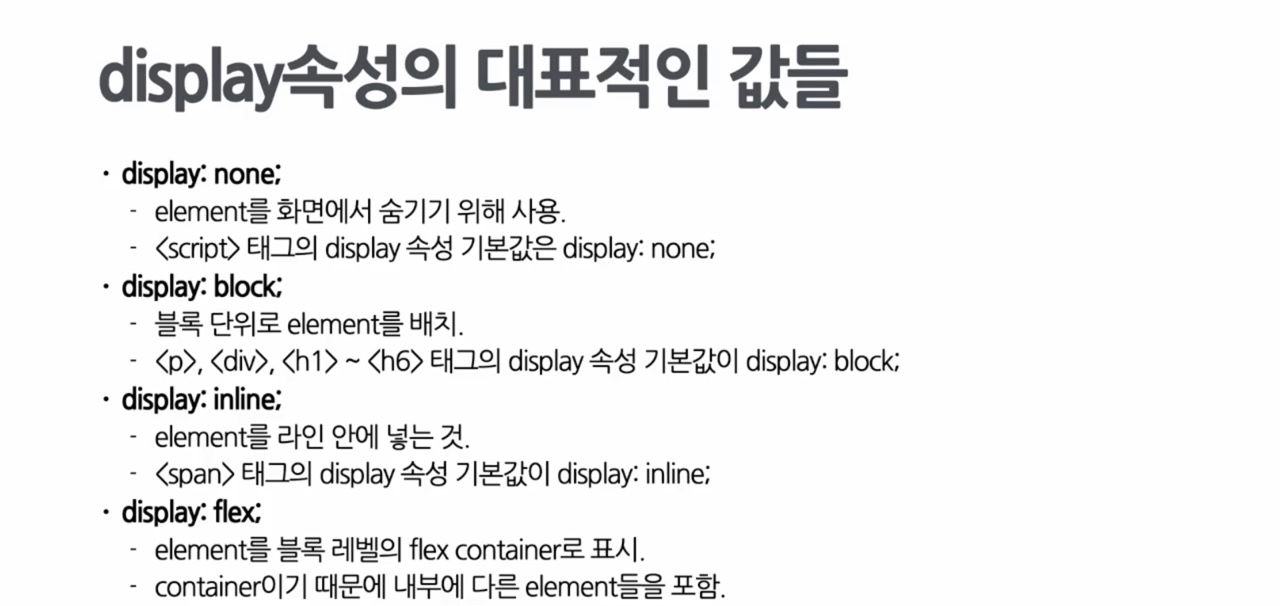
h1, h2, p {}



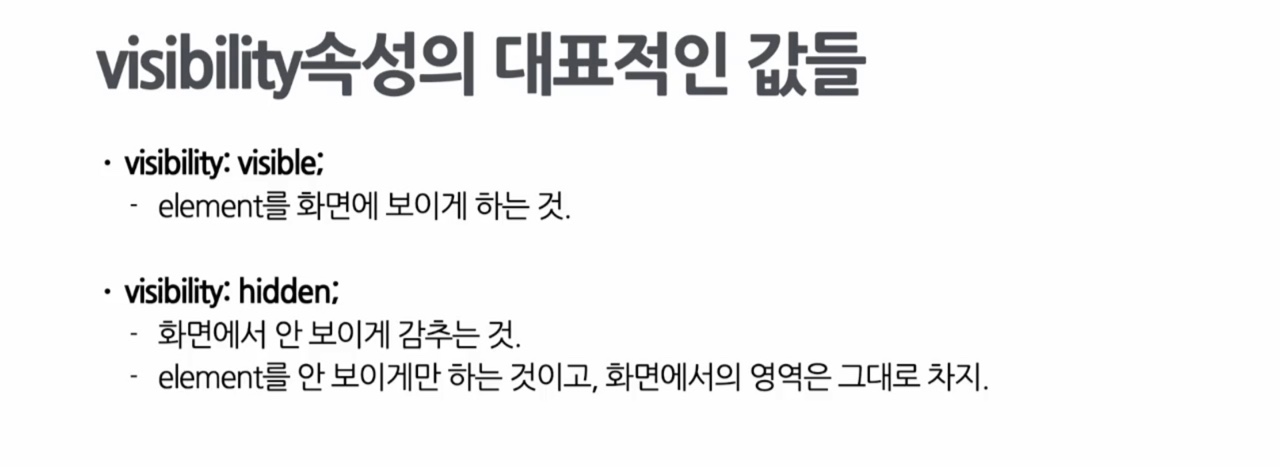
Layout

:화면에 Element들을 어떻게 배치할 것인가?

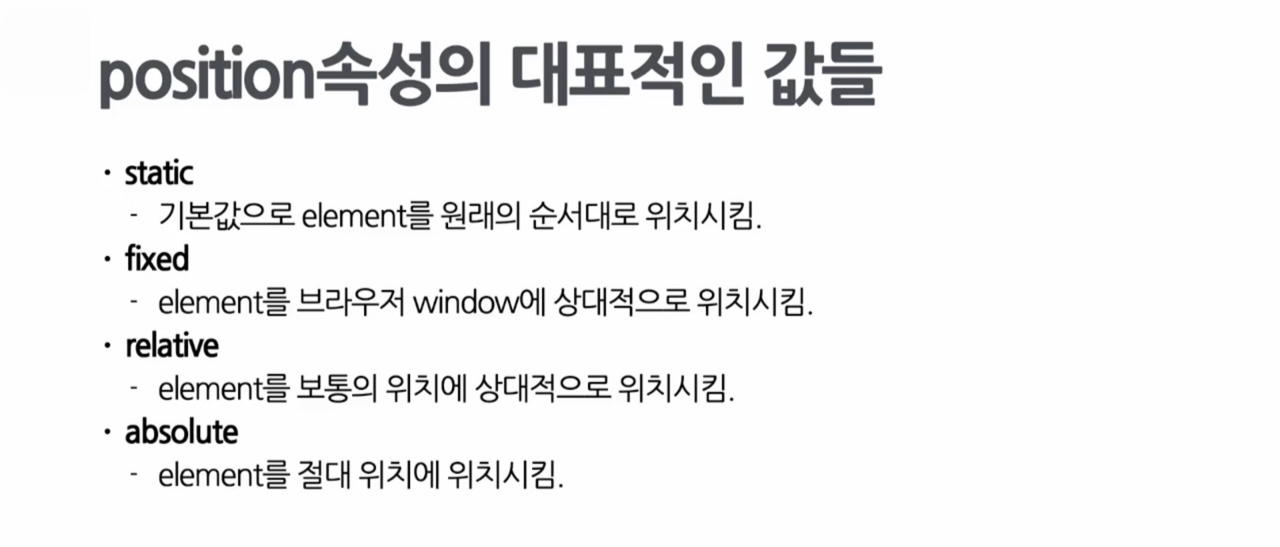
-display 어떻게 표시할지



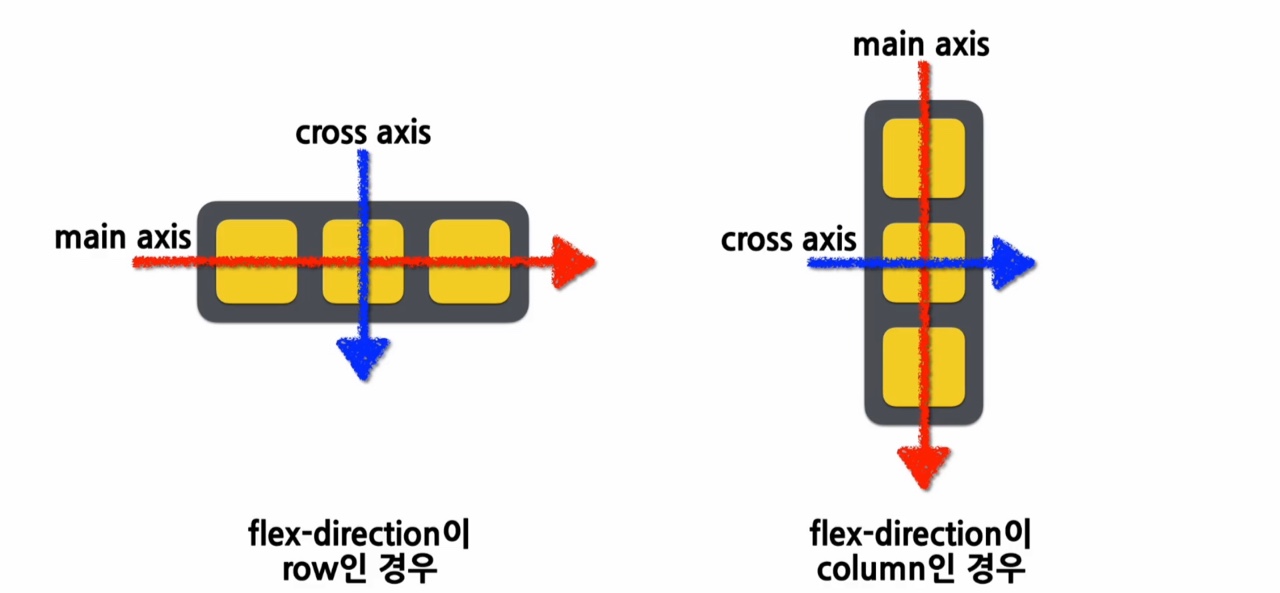
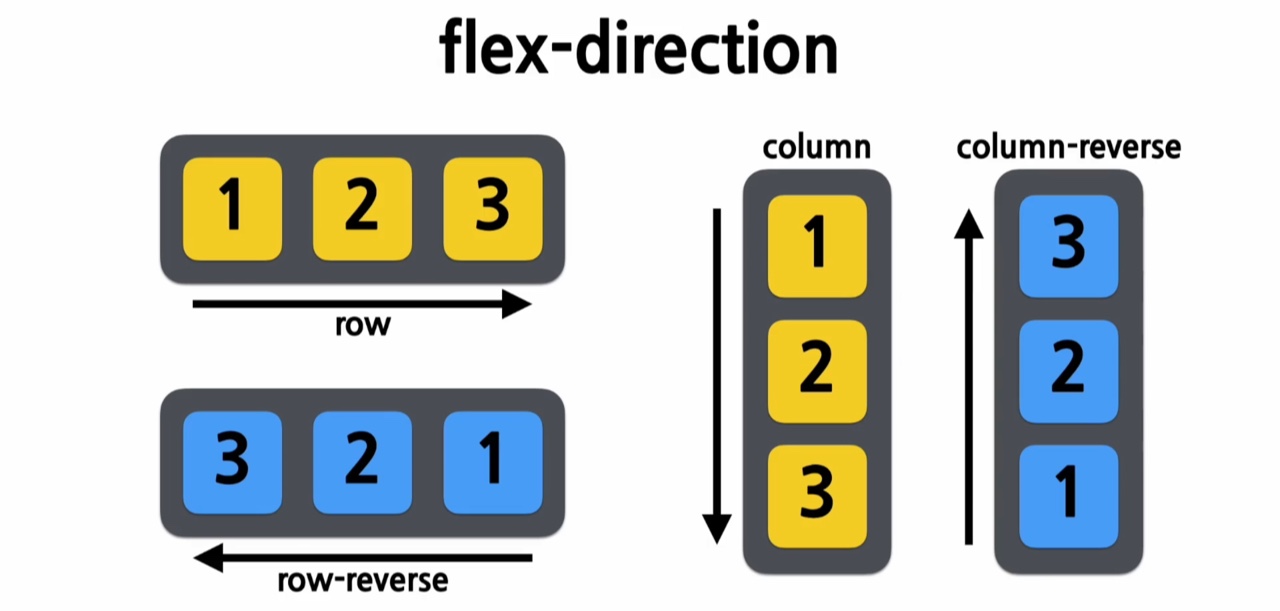
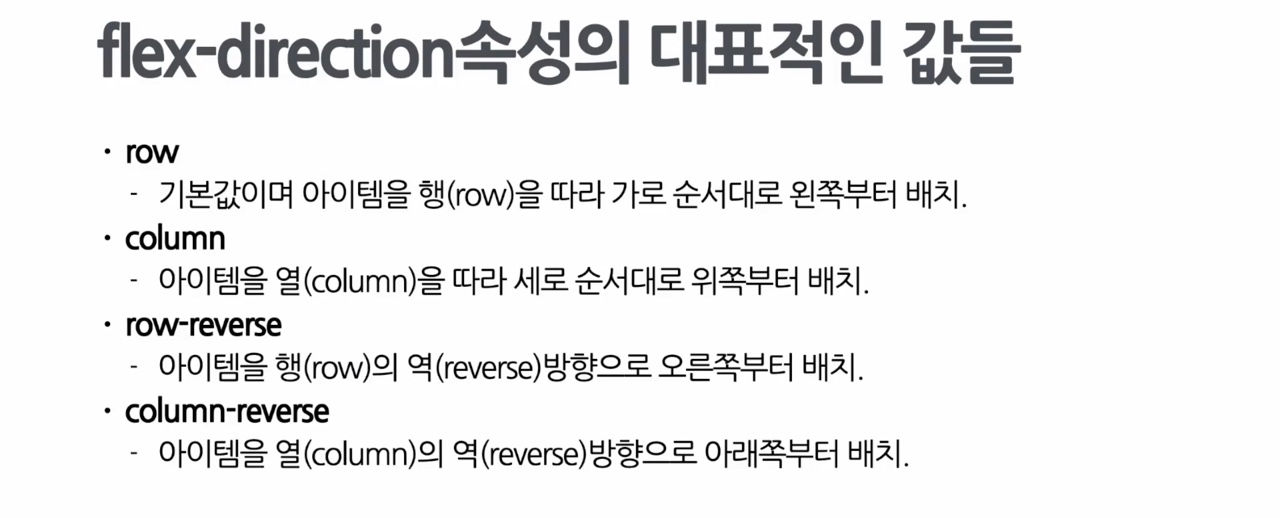
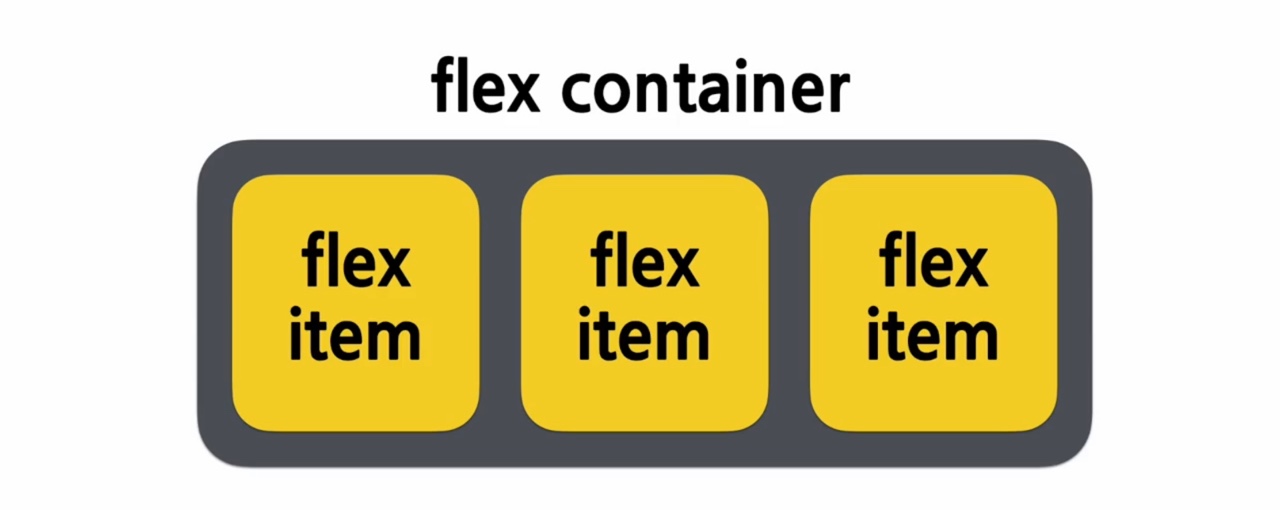
-visibility 표현하거나 감출때 사용



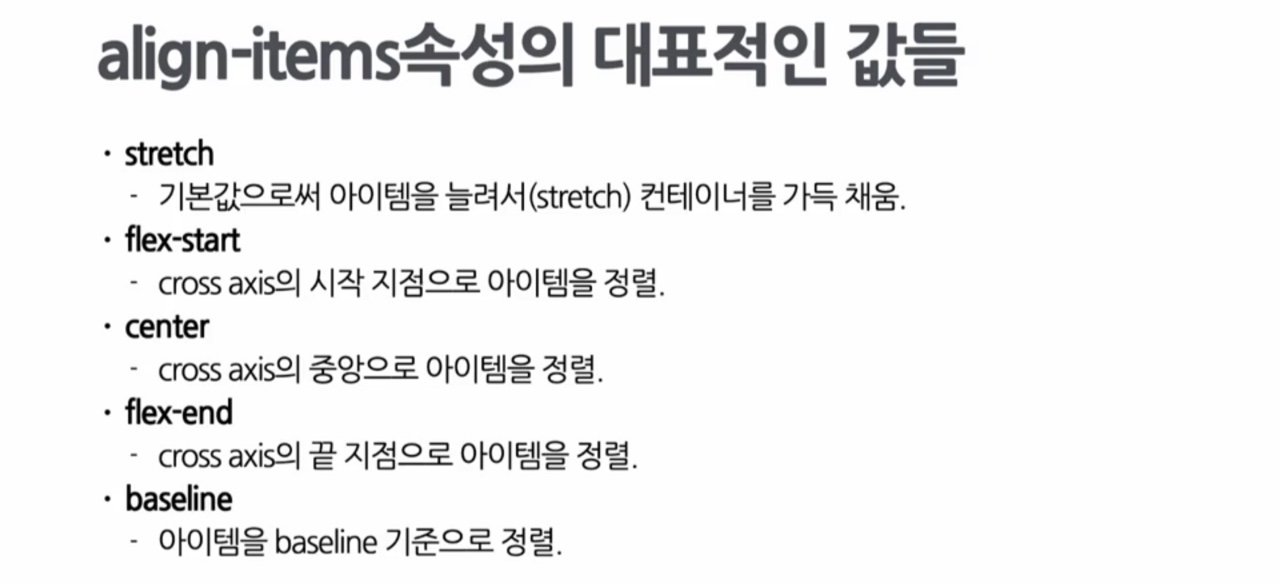
-position 어떻게 위치할지



-Flexbox 기존 불편한 부분 고치기 위해



-align-items 어떻게 정렬



-jutify-content 컨테이너 안에서 아이템들을 어떻게 맞추는지

