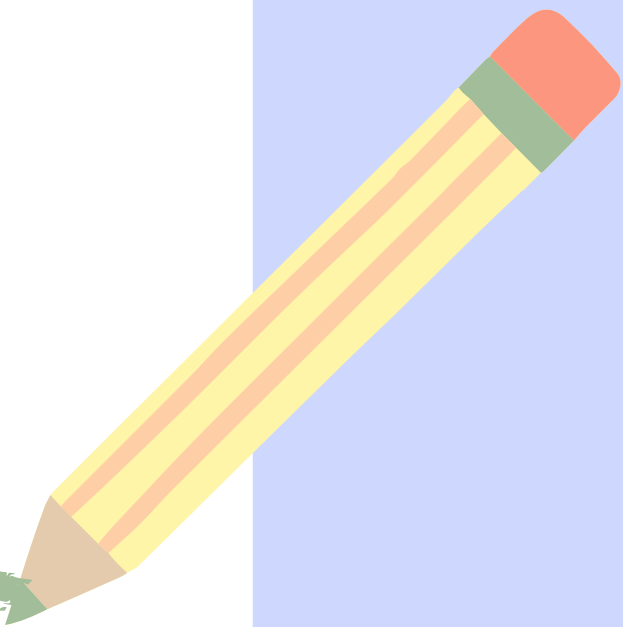




# END OF REACT STUDY

2024.4.28 두선아



# LIST OF CONTENT



- ① 리액트 애플리케이션 고려사항
- ② 오픈소스 고려사항
- ③ 웹개발 고려사항
- ④ RECAP



# REACT PROJECT



# 1. 유지보수 & 레거시

## 버전업

최소 16.8.6에서 가능하다면 17 버전

## 함수 컴포넌트로의 전환

레거시 클래스 컴포넌트는?

- 함수 컴포넌트와 함께 동작 가능
- hooks를 점진적 채택

대규모  
재작성을  
피하자

리액트 클래스 컴포넌트는 근미래에 사라지지 않을 것

<https://legacy.reactjs.org/docs/hooks-intro.html#gradual-adoption-strategy>

**Crucially, Hooks work side-by-side with existing code so you can adopt them gradually.**

There is no rush to migrate to Hooks. We recommend avoiding any "big rewrites", especially for existing, complex class components. It takes a bit of a mind shift to start "thinking in Hooks". In our experience, it's best to practice using Hooks in new and non-critical components first, and ensure that everybody on your team feels comfortable with them. After you give Hooks a try, please feel free to [send us feedback](#), positive or negative.

# REACT 17로 업그레이드가 필요한 이유

- 스터디 6주차

가장 많이 사용되는 버전인 16  
따라서 16.8 이후 고착화된 서비스가 많을 것

## 버전 업그레이드를 왜 해야 할까?

- 모든 라이브러리가 그렇듯, 리액트도 더 나은 기능과 성능을 위해 버전업
- 버전 16.8의 Hook처럼 획기적인 업그레이드가 있을 수도 있음
- 리액트에 의존하는 라이브러리를 사용할 시, peerDependencies를 고려해야 함

### Gradual Upgrades



- semantic version을 기반으로 업데이트  
=> 기존 버전과 호환되지 않기 때문에, major 버전을 올리기 힘들다.
- React 17 버전부터 점진적 업그레이드를 지원한다.  
=> 일부 트리, 컴포넌트에 대해서만 버전을 업그레이드
- modern(17)과 legacy(16, lazy loading)로 구성된 예제 repository  
=> 렌더링 과정에서 버전 불일치로 인한 에러 발생  
=> 두 개의 react root: 컴포넌트, 혹은 Context를 서로 불러와서 사용할 수 있음

For most apps, upgrading all at once is still the best solution

Gradual Upgrades: <https://legacy.reactjs.org/blog/2020/10/20/react-v17.html#gradual-upgrades>  
demo source code: <https://github.com/reactjs/react-gradual-upgrade-demo/>

점진적 업그레이드

- 17.0.0 더 살펴보기
  - <https://github.com/facebook/react/releases/tag/v17.0.0>
- 리액트 16 애플리케이션이 있다면 17로 업데이트 하는 것이 좋다
  - 점진적 업그레이드 대비
  - 리액트팀 피셜, 16 -> 17 업데이트의 공수가 크기 않음
    - 10만 개의 컴포넌트 중 호환성이 깨지는 변경 사항은 20개 미만이라 한다





## 2. 인터넷 익스플로러 11 지원 시

인터넷 익스플로러 11을 지원하지 않는 대표적 라이브러리

- 리액트 18 이후
- Next.js 13 이후
- query-string 6.x 버전 이후

인터넷 익스플로러 11을 지원하는 애플리케이션은 라이브러리 설치에 주의

- 문제가 없는지 판단 후 라이브러리 설치 필요



### 3. 서버 사이드 랜더링 애플리케이션 고려하기



자바스크립트 코드의 실행 속도에 의존적일수록,  
평균적으로 우수한 성능을 지닌 웹사이트를 제공하기 어렵다.

- 사용자별 모바일 기기의 성능은 천차만별

대부분의 싱글 페이지 어플리케이션은  
Lighthouse, WebPageTest, Chrome Devtools에서 좋은 결과를 얻기 어렵다.

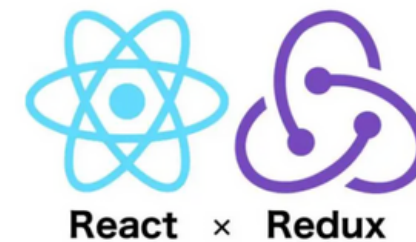
관리자 페이지, 이벤트 페이지, 혹은 서버 준비가 여의치 않은 상황을 제외하면  
(여유가 된다면) 시작부터 서버 사이드 랜더링을 고려하는 것이 좋다.

- Next.js, Remix, Hydrogen
- 

## 4. 상태 관리 라이브러리는 필요할 때만

추세

- 옛날에 리액트+리덕스를 세트로 쓸 때가 있었음
- 현재 다양한 상태 관리 라이브러리가 있고, Context API로 상태 주입도 가능



상태 관리 라이브러리가 필요한가?

- 애플리케이션에서 관리해야 할 상태가 많은가?
- ex) 문서 편집기, 다양한 상태 합성



## 5. 리액트 의존성 라이브러리 설치 조심

react-\*\*

- peerDependencies의 리액트 버전을 확인해야 한다.
- 리액트 16.8버전 이상(혹)
- 리액트 18(외부 상태관리 방법 변경)

```
{  
  "peerDependencies": {  
    "react": "^16.8.6 || ^17.0.0",  
    "react-dom": "^16.8.6 || ^17.0.0"  
  }  
}
```



## 6. 언젠가 사라질 수 있는 React

리엑트는 “지금 가장 널리 쓰이는” FE 라이브러리

완벽하지 않음

- (짬뽕 문서) 클래스 컴포넌트 -> 함수 컴포넌트
- Svelte, Vue 공식문서가 깔끔함

리엑트 규칙과 개념들

- useEffect 콜백에 async 안됨
  - 변수와 useState, 의존성 배열, 클린업
- > 상대적으로 어렵게 느껴질 수 있다

Svelte 코드 예시: 더 직관적

```
<script>
  import {getFruits} from './service';
  import {onMount, onDestroy} from 'svelte';

  let fruits = []
  let interval

  onMount(async () => {
    fruits = await getFruits()

    interval = setInterval(() => {
      fruits = [...fruits, 'banana']
    })
  })

  onDestroy(() => clearInterval(interval))
</script>

<ul>
  {#each fruits as fruit}
  <li>{fruit}</li>
  {/each}
</ul>
```



## 7. 광범위한 자유



### 파편화된 React 기술 스택

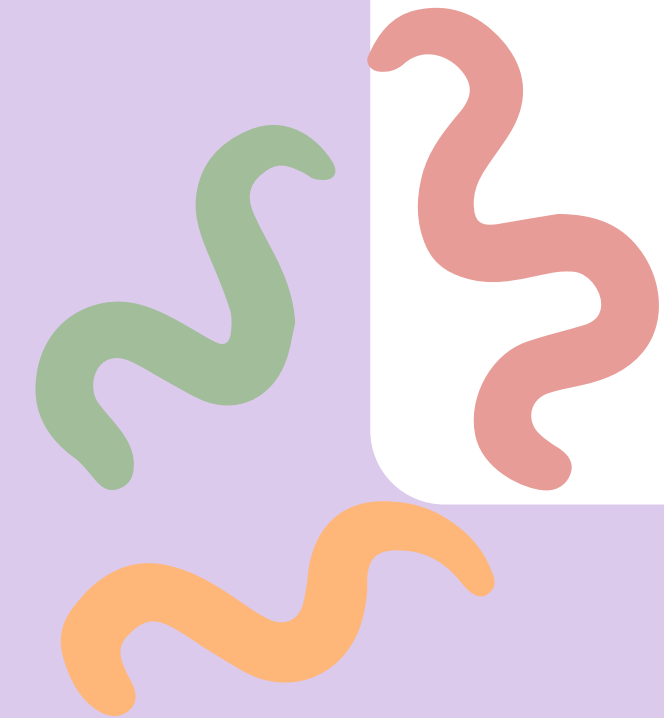
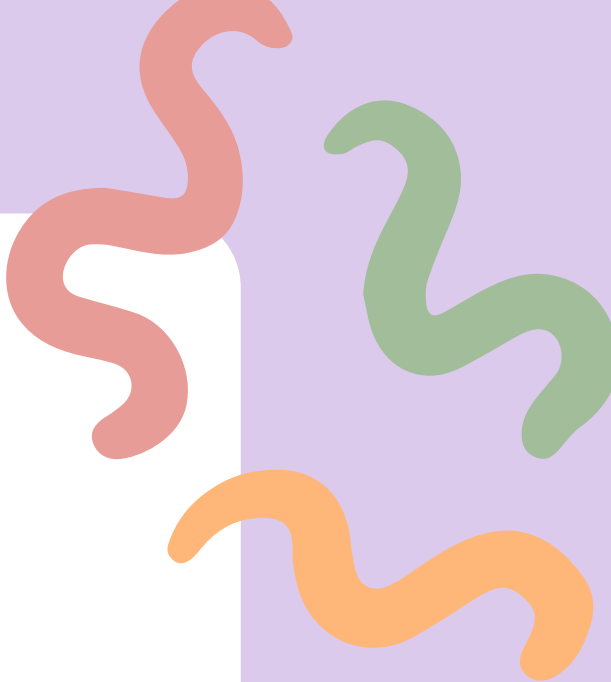

- 스타일: 외부 CSS Import, inline, CSS Module, styled-components, emotion
- 상태관리: Redux, MobX, Jotai, Zustand, React Tracked, Valtio
- 데이터 fetch 방법 등

### 단점

- 비직관적인 JSX
- 급진적으로 변화하는 API

리엑트를 De facto standard(사실 상의 표준)로 여긴다면  
급진적으로 변화하는 프론트엔드 생태계의  
새로운 변화를 받아들이고 유연하게 적응하기 어려워질 것





# OPEN SOURCE

# 페이스북 라이선스 이슈

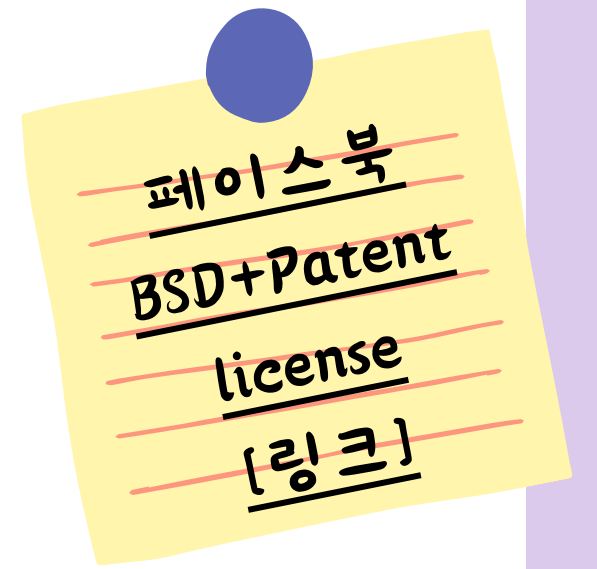


오픈소스 라이선스 중 가장 널리는 라이선스 MIT

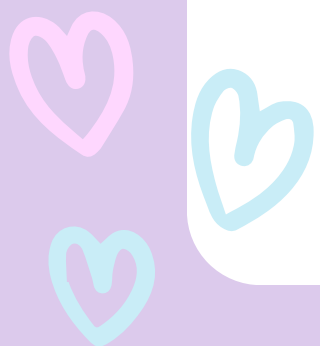
- 상업적 이용, 배포, 개인적 이용, 제약 없이 소프트웨어를 취급할 수 있는 자유로운 라이선스

이슈

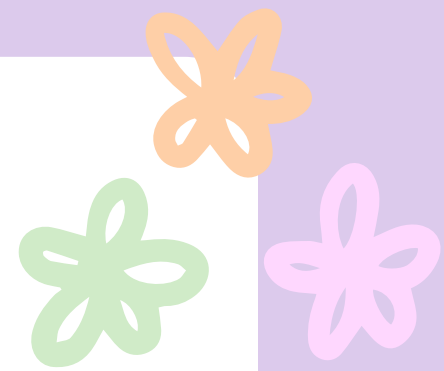
1. 페이스북의 React, Immutable, Jest 등에 BSD+Patents 사용
2. 2017년 7월 아파치 재단에서 BSD+Patents 금지
3. 페이스북의 BSD+Patents 라이선스 유지
4. 찬반 논란
5. 워드프레스 진영에서 3의 이유로 자사 소프트웨어에 React를 사용하지 않을 것으로 밝힘
6. 페이스북이 MIT 라이선스로 바꿈



📌 오픈소스로 만든 소프트웨어의 권리를 잃어버릴 수 있다는 경각심을 갖게 하는 사례



# 오픈소스는 무료로 계속 제공될까?



## 바벨

- 오픈소스로 이루어져 있고, 풀타임 개발자를 고용하고 있지만, 재정난을 겪고 있다
  - 연간 330,000달러가 필요하며 이를 위한 모금 진행

## 악용 사례

- colors.js => 1.4.1 고의적 무한 루프 삽입
- faker.js => 6.6.6 빈 소스코드
  - Pay Me or Fork This

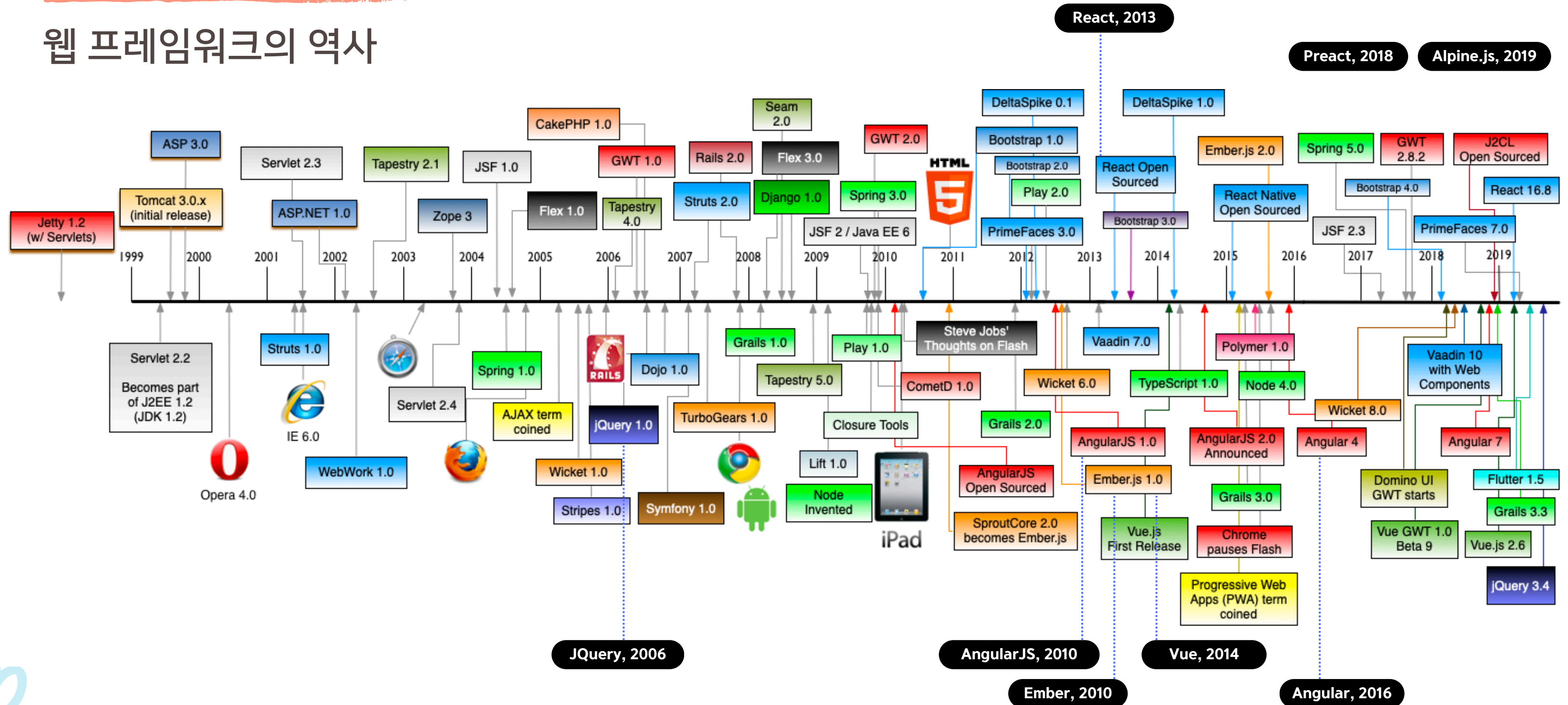
## 오픈소스를 당연히 무료로 쓰는 현상에 대한 비판 진영

- color.js 여파는 아마존 웹서비스 CLI까지 영향을 주었음 - [\[링크\]](#)
- 만약 바벨이 패키지 관리를 중단한다면...?
  - 오픈 소스가 무슨 일을 하고 있는 지 알고 있어야 한다



# JQuery, Angular, React, ...

## 웹 프레임워크의 역사





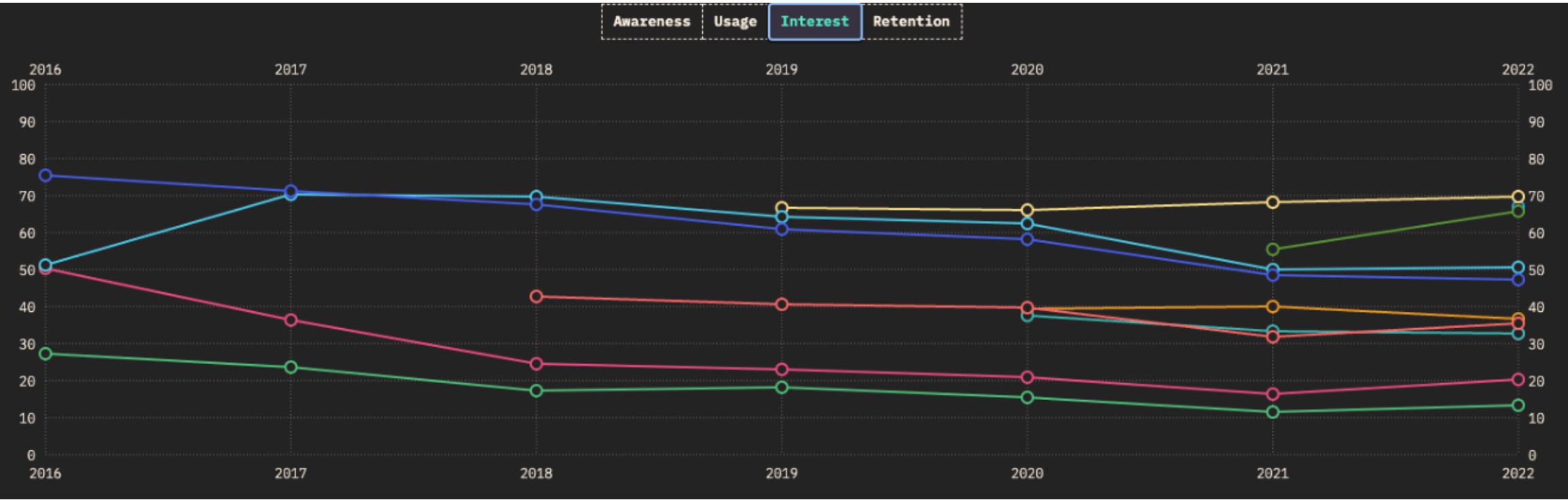
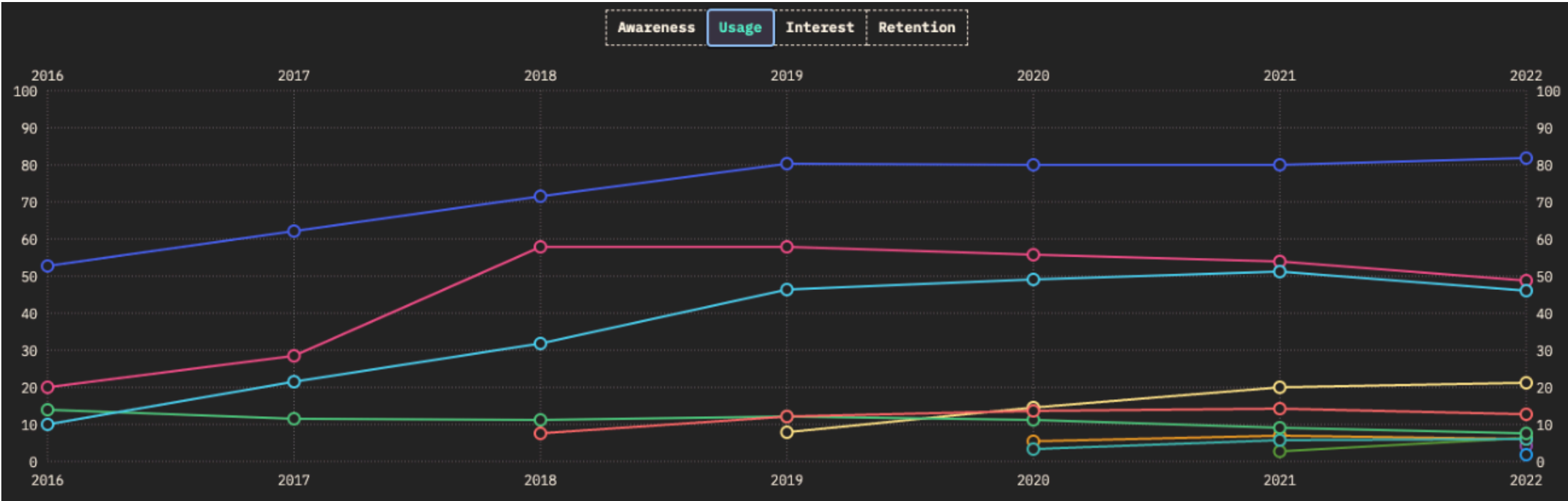
# State of JS 2022

usages

React  
Angular  
Vue.js

interest

Svelte  
Ember  
Vue.js  
React



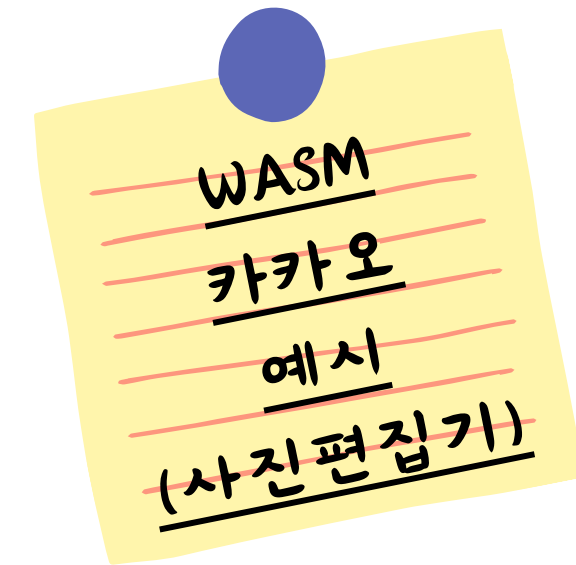




# WEB DEVELOPMENT



# 웹 개발



## WASM

- 느린 자바스크립트를 대신할 하나의 방안
- C, C++, 러스트 같은 시스템 프로그래밍 언어로 작성된 프로그램을 웹에서 사용하게 함
- 자바스크립트와 상호보완
  - 성능이 필요한 작업은 웹어셈블리
  - 일반적인 작업은 자바스크립트

## Rome Biome

<https://biomejs.dev/blog/announcing-biome/>

- ESLint 보다 빠름, 러스트로 작성
- 자바스크립트로 실행되는 바벨과 웹팩을 속도 측면에서 개선하려는 시도



# 웹개발

모든 웹 개발은  
HTML, CSS, Javascript 토대 위에 세워졌음

자바스크립트

- 클로저
- 비동기 작업(마이크로 테스크 큐)
- 클래스

유연한 개발자가 되기





RECAP

# WEEKS

## “리액트 그게 뭔데

리액트 핵심 요소 깊게 살펴보기  
- 모던 리액트 Deep Dive

2024.03.10  
1주차 발표 장표  
두선아

week 1

2장 리액트 핵심 요소 살펴보기

## React & State Management

리액트와 상태 관리 라이브러리

week 2

5장 리액트와 상태 관리 라이브러리

## Debugging React

using React Dev Tools

2024. 3. 24.  
두선아

week 3

6장 리액트 개발 도구로 디버깅하기

# WEEKS

Server-Side Rendering; SSR

## 서버사이드 렌더링

2024.03.31 두선아

week 4

4장 서버 사이드 렌더링

모던 리액트 Deep Dive  
9장 모던 리액트 개발 도구로 개발 및 배포 환경 구축하기

## Modern React Development & Deploy

2024. 4. 7  
Du Sun-A

week 5

9장 모던 리액트 개발 도구로 개발  
및 배포 환경 구축하기

10장 리액트 17과 18의 변경사항 살펴보기

## React 17 & 18

240414 두선아

week 6

10장 리액트 17과 18의 변경 사항 살펴보기

# WEEKS



week 7

12장 모든 웹 개발자가  
관심을 가져야 할 핵심 웹 지표

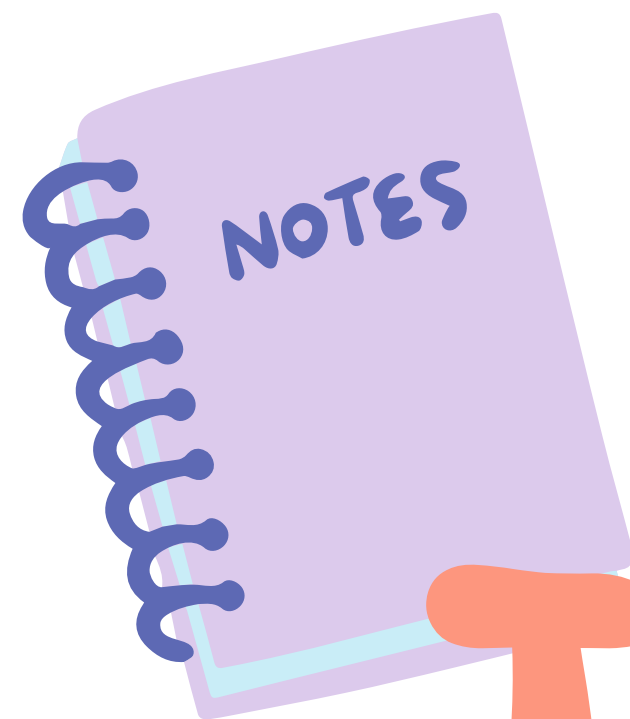


week 8 (here!)

15장 마치며



end()



well  
done!

# THANK YOU

모던 리액트 Deep Dive 스터디  
2024.3.4~2024.4.28

