DEVIEW 2020

어서와, SSR은 처음이지?

(네이버 블로그 Node.js 기반의 Server-Side Rendering 적용기)

손찬욱 Apollo CIC / Apollo FE Dev

네이버 블로그 SSR 적용기

왜?
적용했는가?

무엇을
준비했는가?

어떻게
해결했는가?

적용
효과

1. 네이버 블로그는 왜?

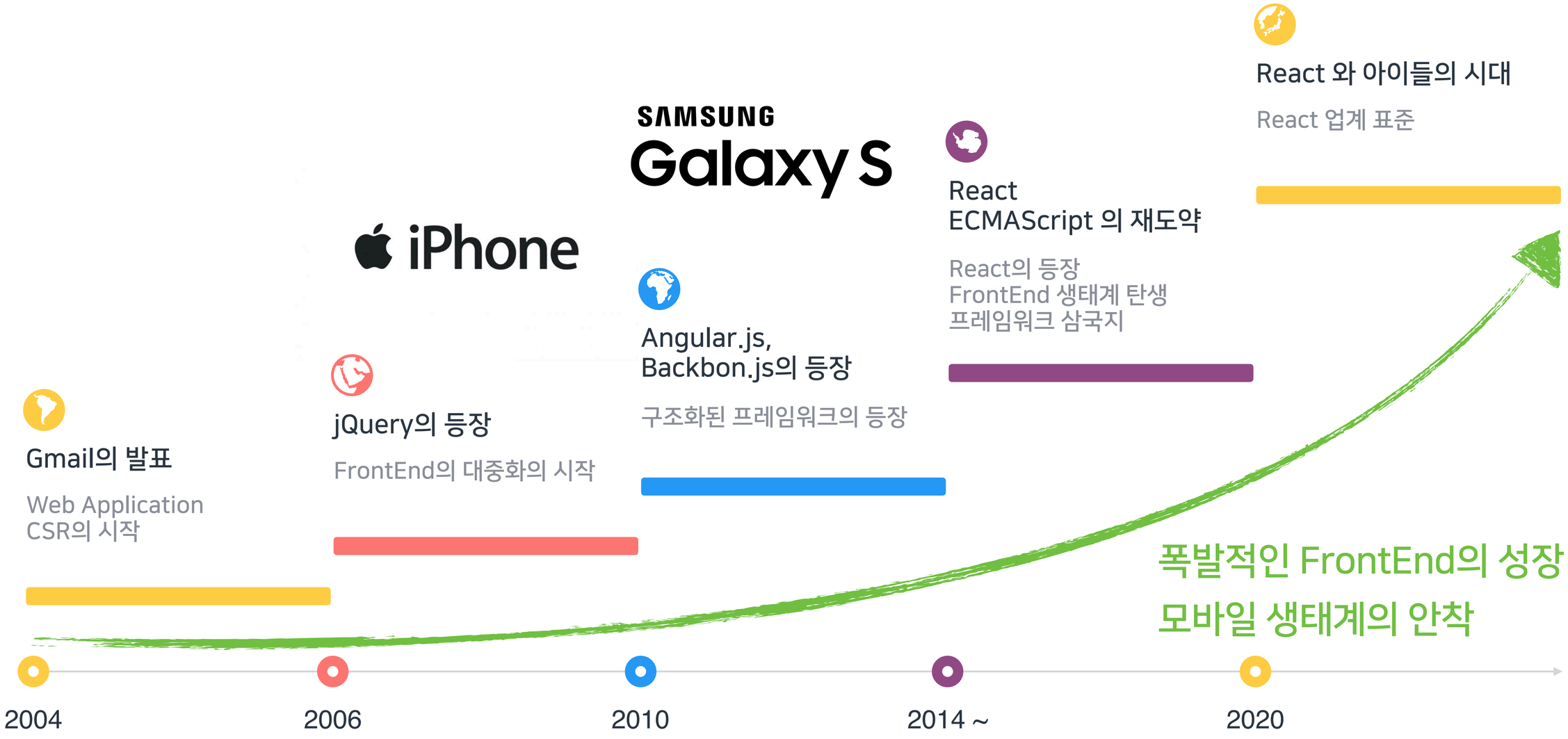
Node.js 기반의 SSR을 도입하였나?

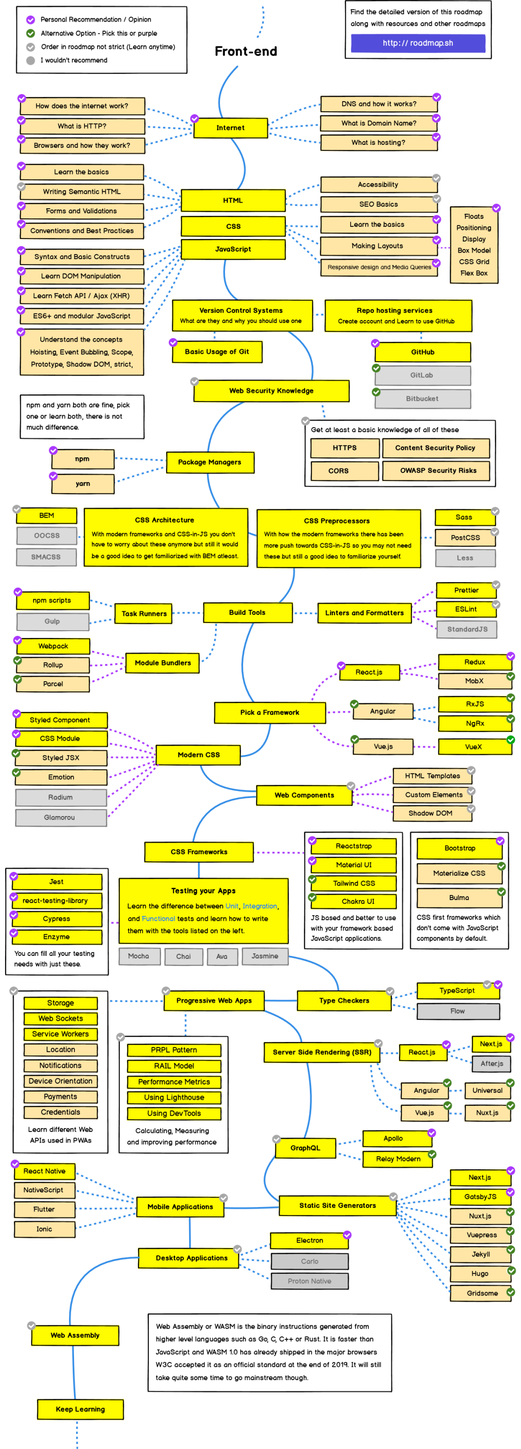

COSTCO

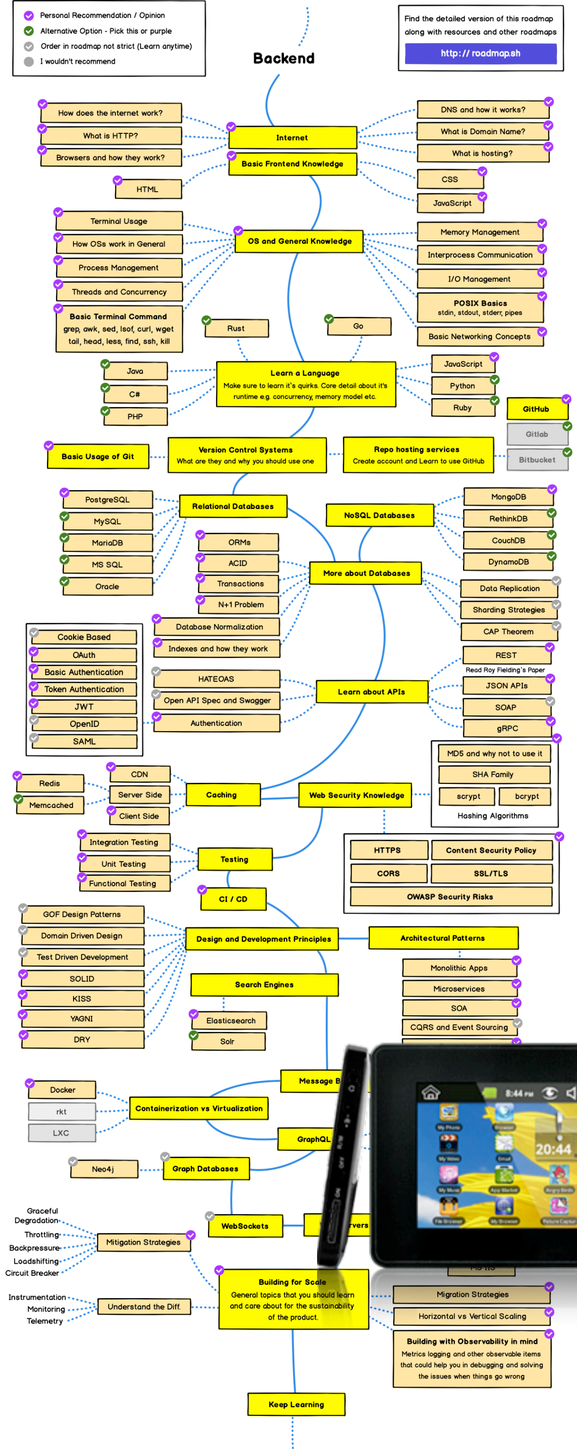
1.1 블로그의 오늘

2003년부터 시작된
네이버와 함께 성장한 서비스

1.2 17년간 웹 생태계는 어떻게 바뀌었나?







DEVIEW 2020

FrontEnd, BackEnd 알아야 할 것은 많고…

대응해야 할 것도 많고…

블로그도
한 사람의 개발자가 해야할 것이 많음

서비스 개발도 진행 해야해서…

개발자 전문성 하락

사용자의 FrontEnd 요구사항을
신속하게 처리하기가 어려워짐

1.3 FrontEnd 생산성을 높여보자

전 FrontEnd
개발자 입니다.

개발 프로세스

1.4 FrontEnd 개발 프로세스

FrontEnd 개발자

API는 어떻게 할까요?

HTML 구조는 이렇게 하면 좋을 것 같아요.

SEO에 노출되려면 HTML에 관련 정보를 넣어야해요.
접근성 대응을 위해서는 HTML 속성 추가가 필요해요

FE 성능 개선을 해야하는데요.
HTML 내려줄 때 데이터를 이미지 속성정보가 필요해요.

상품정보는 서버와 클라이언트 둘다 표현이 필요해요

...

BackEnd 개발자

동적인 요소
(JavaScript)

Browser

API 인터페이스,페이지에 대한
커뮤니케이션

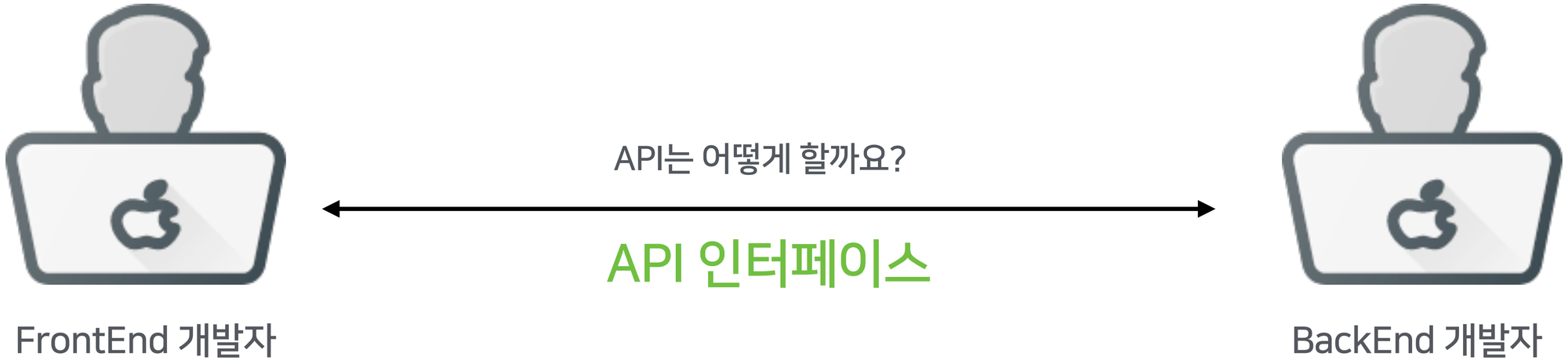
동일 모듈에 대한 개발

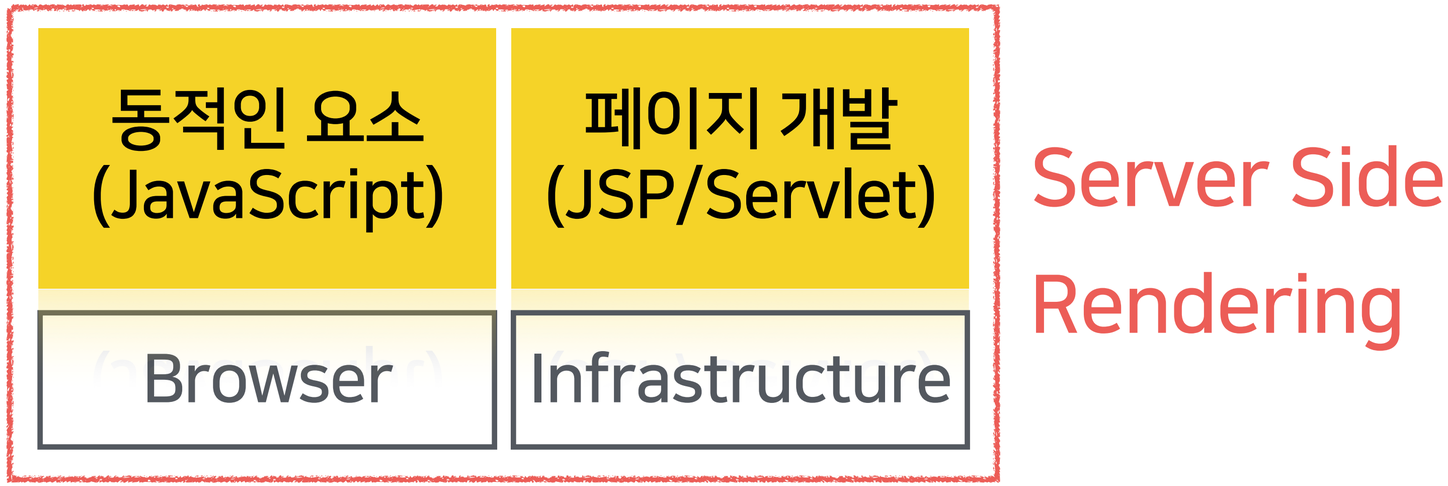
페이지 개발
(JSP/Servlet)

API
(Java)

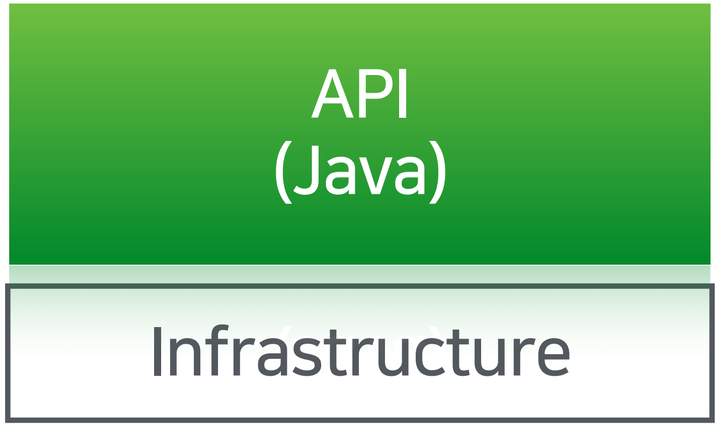
Infrastructure

1.4 FrontEnd 개발 프로세스





Server Side Rendering



SSR의 이점

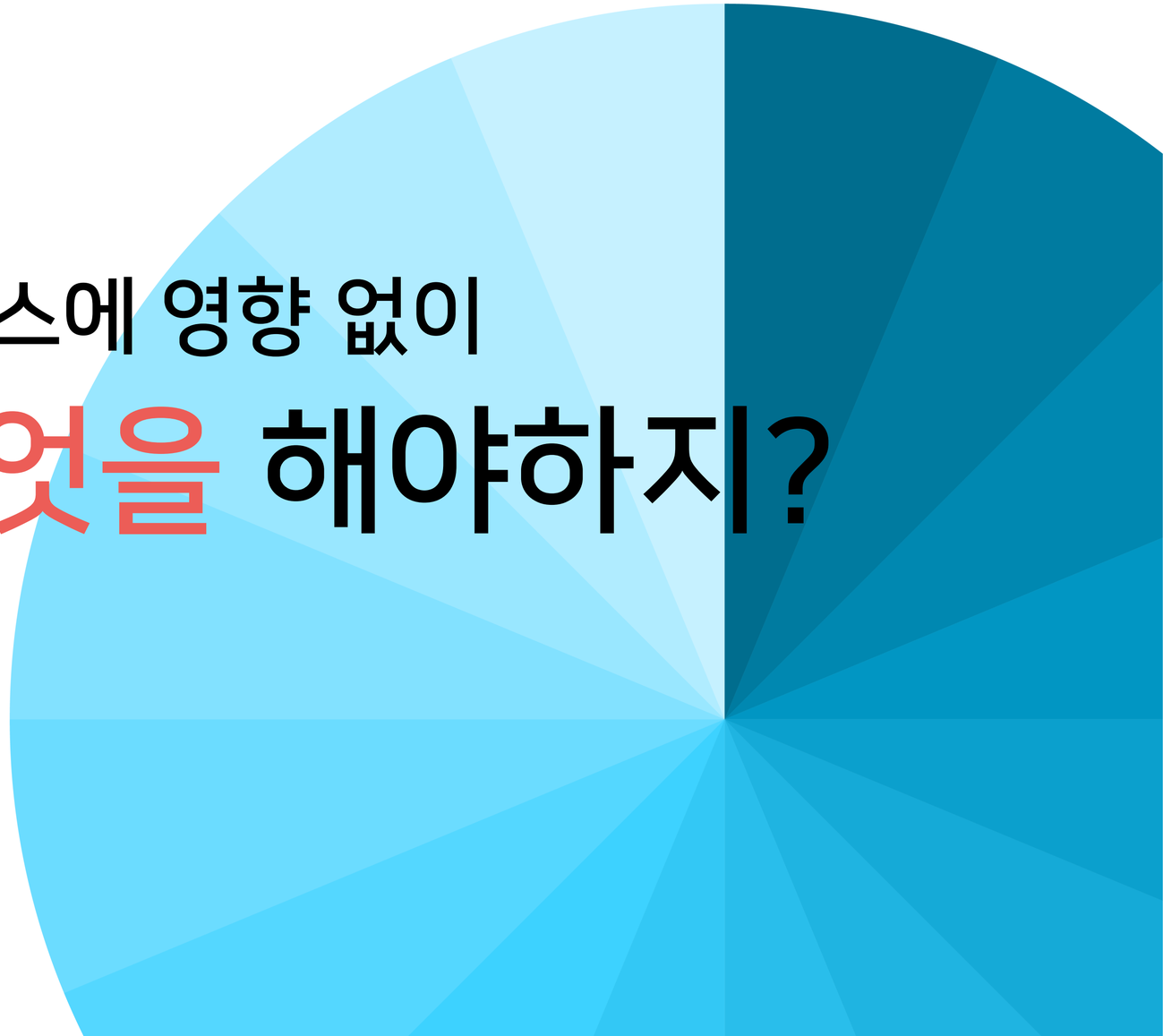
DEVIEW 2020

FE 생산성 향상

- 커뮤니케이션 비용이 감소
- React 기반의 SSR을 선택함으로써 universal language의 장점
- 성장한 React 생태계를 이용

부가적으로 성능

2.기존 서비스에 영향 없이 적용하려면 무엇을 해야하지?




어떻게 기존 시스템을
유지하면서 적용할 수 있을까?

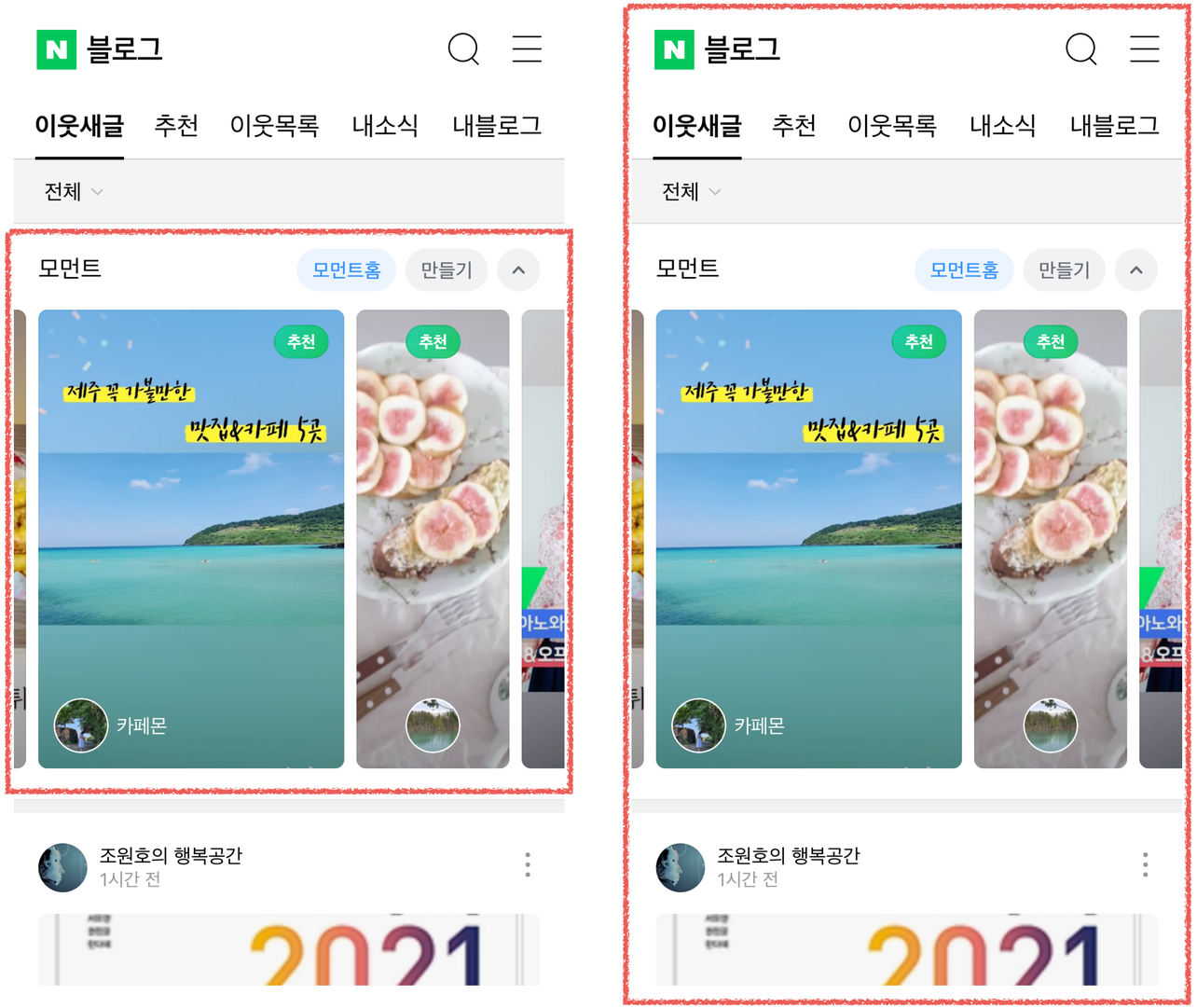
2.1 어떻게 기존 시스템을 유지하면서 적용할 것인가?

DEVIEW 2020

빅뱅 VS 점진적 배포

블럭별, 컴포넌트별 변경 작업

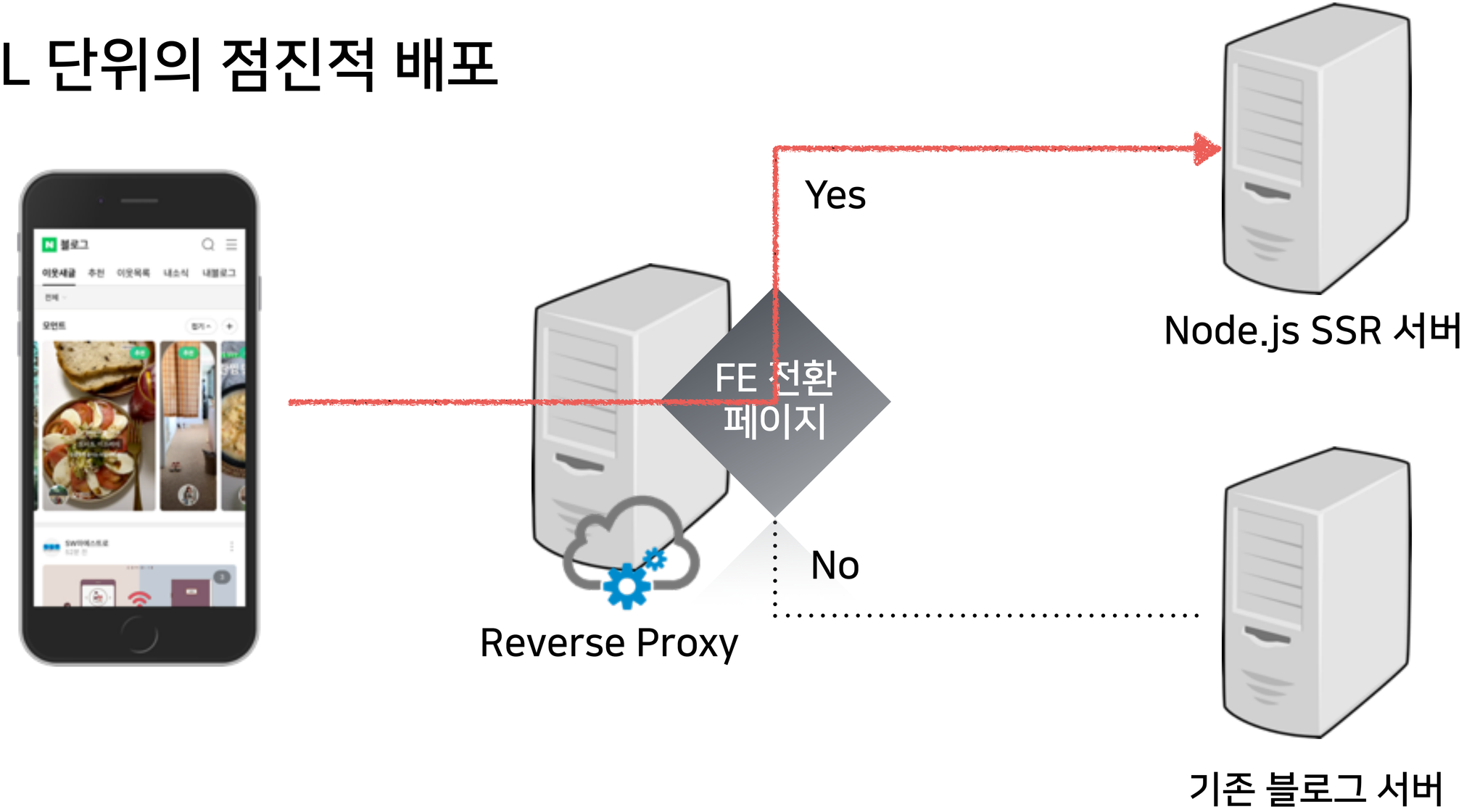
페이지별 변경 작업



2.1 어떻게 기존 시스템을 유지하면서 적용할 것인가?

DEVIEW 2020

URL 단위의 점진적 배포



2.1 어떻게 기존 시스템을 유지하면서 적용할 것인가?

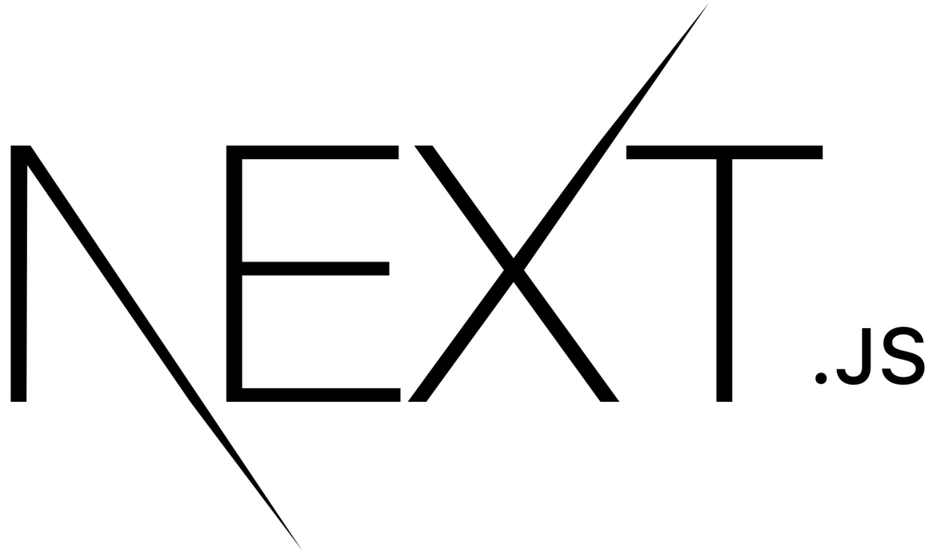
DEVIEW 2020

URL 단위의 점진적 배포

Node.js SSR 서버

기존 블로그 서버

2.2 개발 환경은 어떻게 선정했나?



React Recommended Toolchains

If you're building a server-rendered website with Node.js, try Next.js

https://reactjs.org/docs/create-a-new-react-app.html#recommended-toolchains

2.2 개발 환경은 어떻게 선정했나?

자체 구축

구분	대상
언어	ES.Next VS TypeScript
상태 관리	Redux vs Mobx
비동기 미들웨어	Redux-Thunk vs Redux-Saga
Node.js 웹 프레임워크	Express vs Koa

개발 생산성

투명성

안정성

2.3 Node.js는 Java를 대처할 수 있을까?

DEVIEW 2020

세계적으로 검증된 Node.js

NETFLIX

LinkedIn

PayPal

Walmart
Save money. Live better.

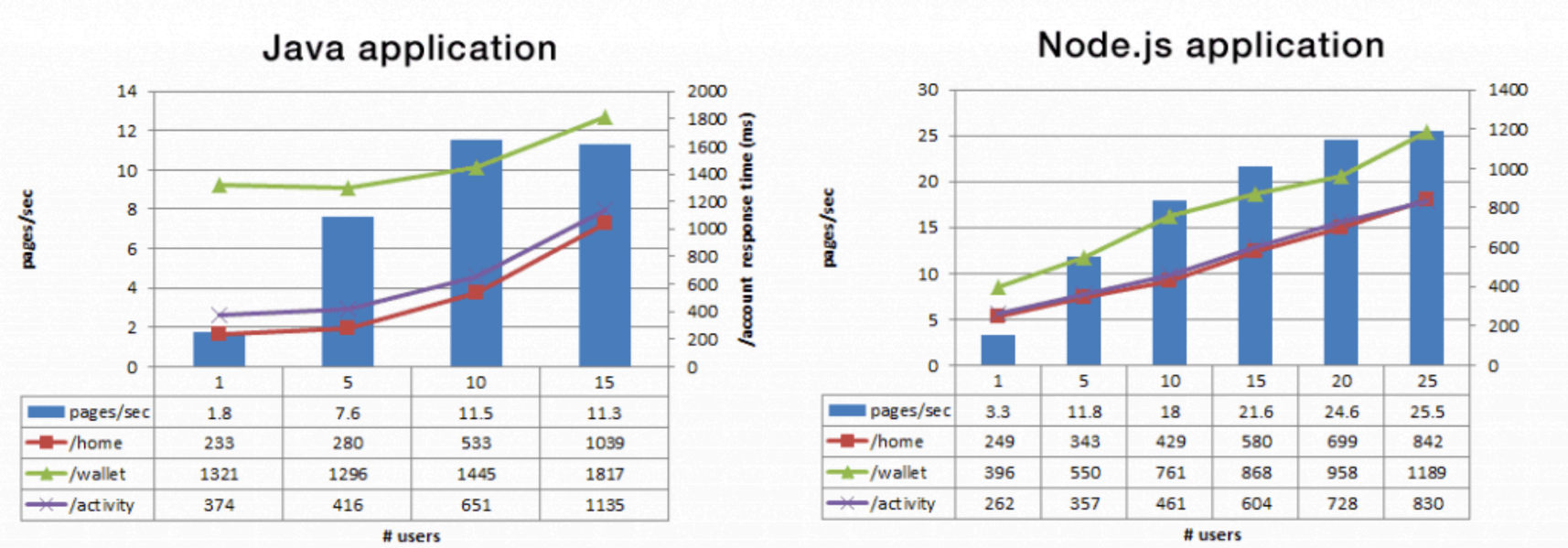
UBER

2.3 Node.js는 Java를 대처할 수 있을까?

DEVIEW 2020

PayPal™

35% decrease in the average response time for the same page.
This resulted in the pages being served 200ms faster



Java application	1	5	10	15
pages/sec	1.8	7.6	11.5	11.3
/home	233	280	533	1039
/wallet	1321	1296	1445	1817
/activity	374	416	651	1135

Node.js application	1	5	10	15	20	25
pages/sec	3.3	11.8	18	21.6	24.6	25.5
/home	249	343	429	580	699	842
/wallet	396	550	761	868	958	1189
/activity	262	357	461	604	728	830

출처: Node.js at PayPal (https://medium.com/paypal-engineering/node-js-at-paypal-4e2d1d08ce4f)


(의심병 발동)
거짓말이야...!

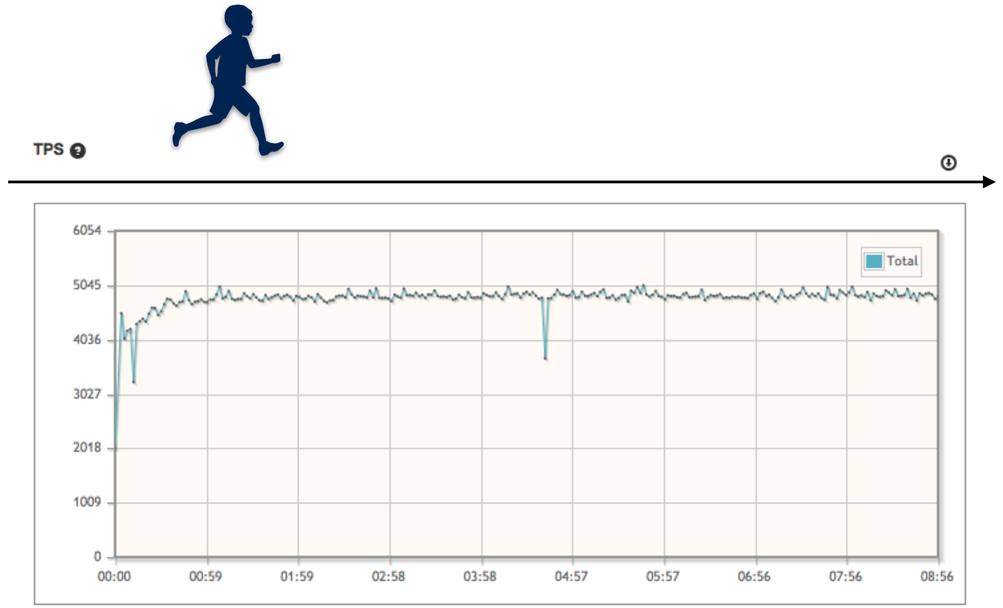
2.4 부하 테스트로 무엇을 얻어야 하는가?

DEVIEW 2020

- 서버 환경의 **최적의 환경**을 찾는 것.
- **시스템의 가용량**을 확인하는 것

Node.js SSR 서버

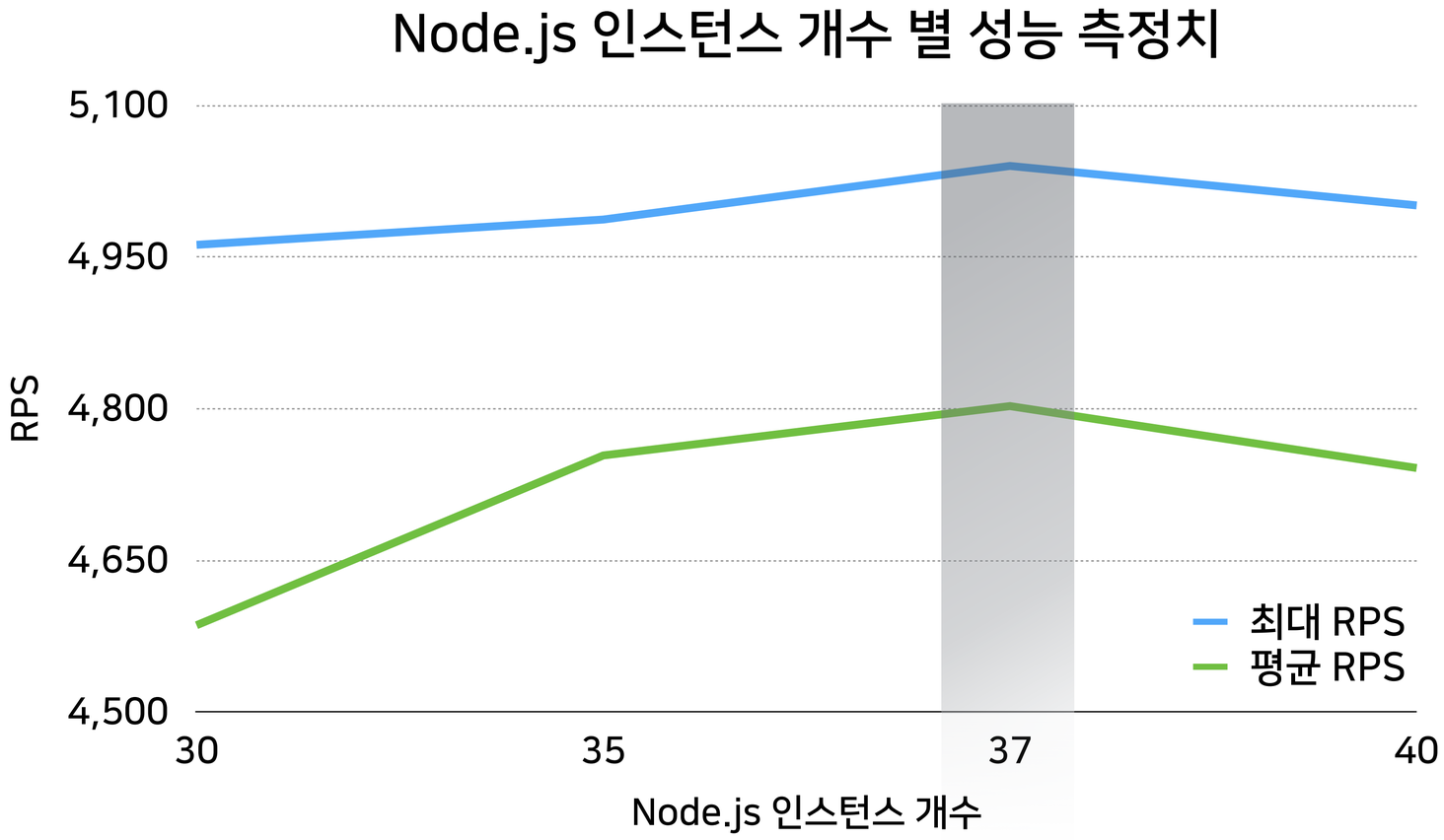
nGrinder



2.4 부하 테스트로 무엇을 얻어야 하는가?

DEVIEW 2020

서버의 최적의 환경 찾기



CPU Core : 40 개

Node.js 인스턴스 / CPU
0.92 비율에서 최고치

2.4 부하 테스트로 무엇을 얻어야 하는가?

시스템 가용량 찾기

최대 수용할 수 있는 서비스의 RPS

= 1개의 서버(pod)가 수용할수 있는 최대 RPS * n

2.5 프로라면 준비해야할 운영 준비


계획은 무슨
계획대로 되지 않는 게
인생이라는 거야!

3. SSR 적용시 마주치게 되는 이슈

어떻게 해결할 것인가?

3.1 Node.js 운영 전략

DEVIEW 2020

비동기 상황에서의 에러

```
try {
  setTimeout(function() {
    throw new Error("Error");
  }, 1000)
} catch(e) {
  console.log(“잡았다 요놈");
}
```

UncatchedException 이 발생하면서 node.js 인스턴스가 죽음

3.1 Node.js 운영 전략

DEVIEW 2020

```
process.on('uncaughtException', (err, origin) => {})
```

The event should not be used as an equivalent to On Error Resume Next. Unhandled exceptions inherently mean that an application is in an undefined state. Attempting to resume application code without properly recovering from the exception can cause additional unforeseen and unpredictable issues.

https://nodejs.org/api/process.html#process_warning_using_uncaughtexception_correctly

UncatchedException 이 발생하면 node.js 인스턴스 재시작

3.1 Node.js 운영 전략

Transtion이 꼭! 보장되어야하는 서비스에서는 사용하기 어렵다.

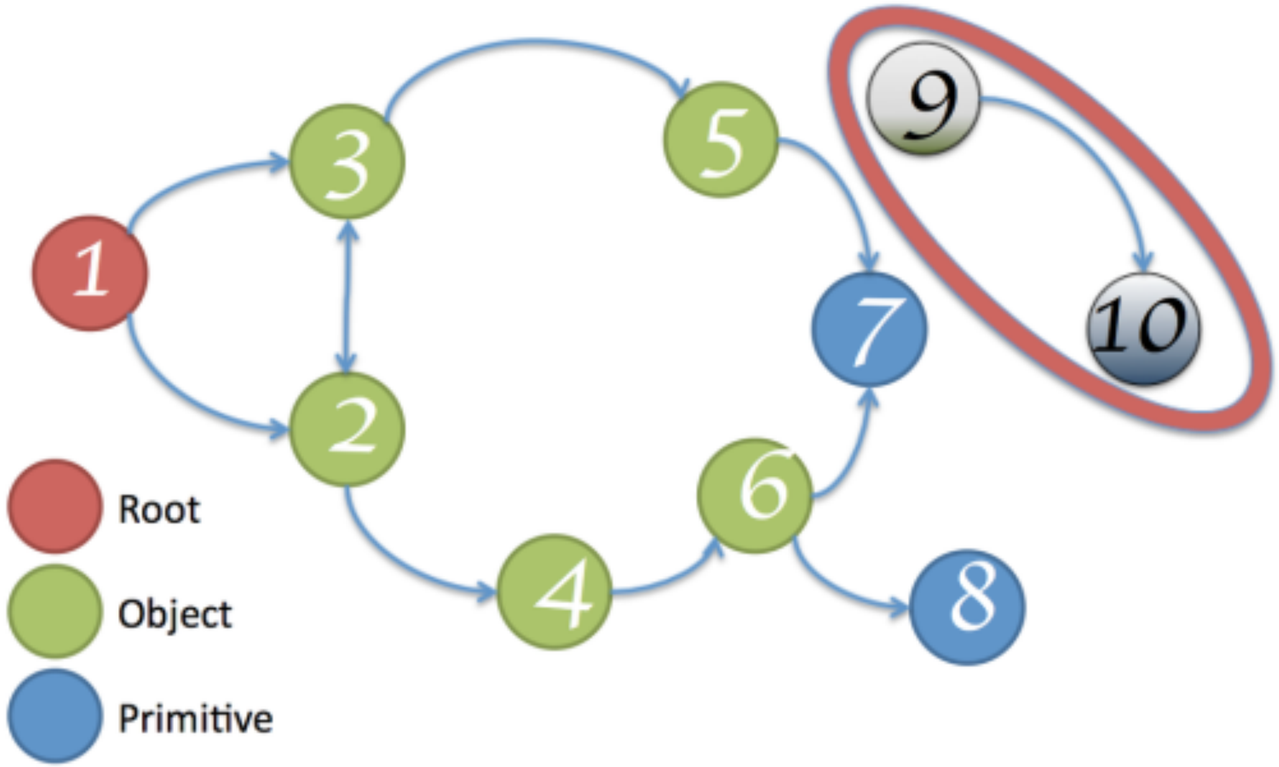
조회성, I/O가 빈번한 도메인에 유리하다.

죽지않고 튼튼한 서비스 운영이 아닌

빠르게 살리고 빠르게 죽이기 운영 전략

3.2 메모리 누수 확인하고 해결하기

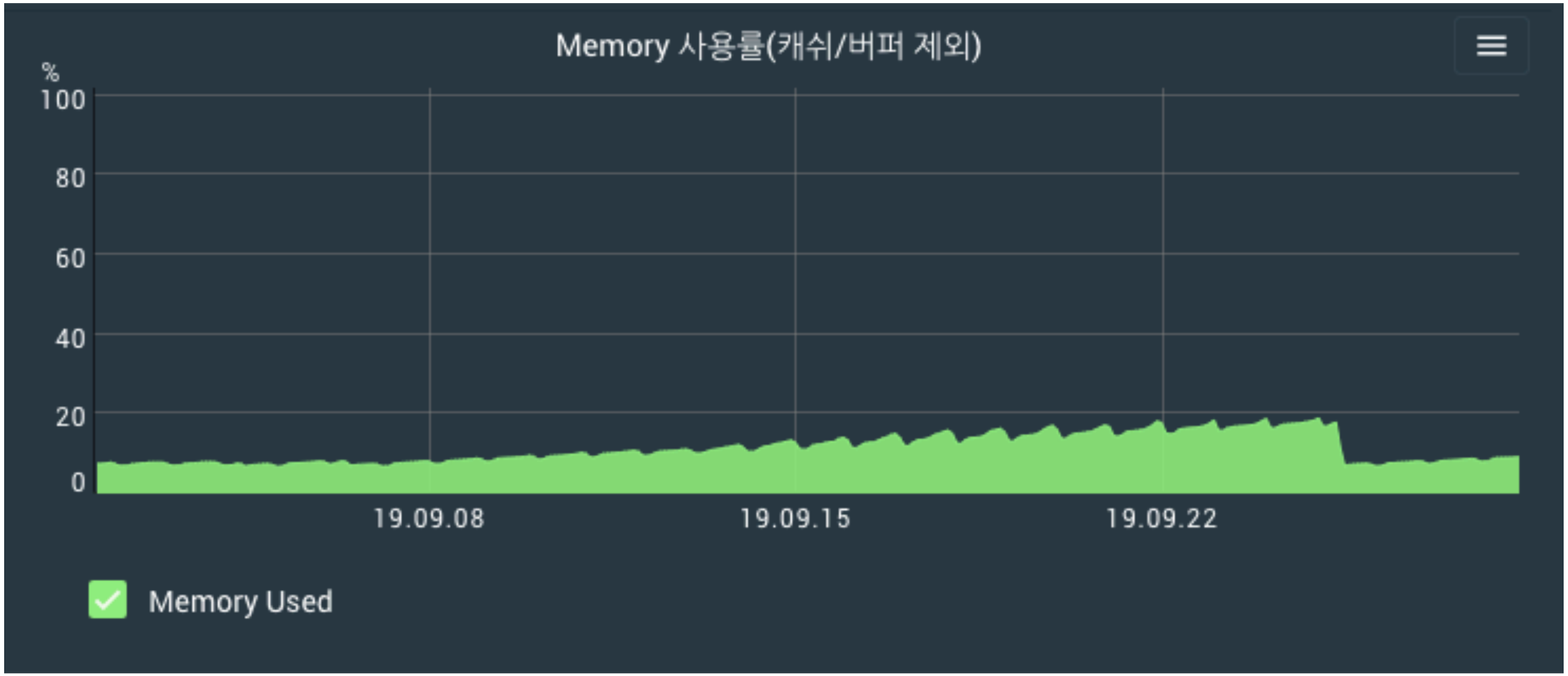
GC를 통해 Root와 끊어진 node 메모리를 해제



3.2 메모리 누수 확인하고 해결하기

DEVIEW 2020

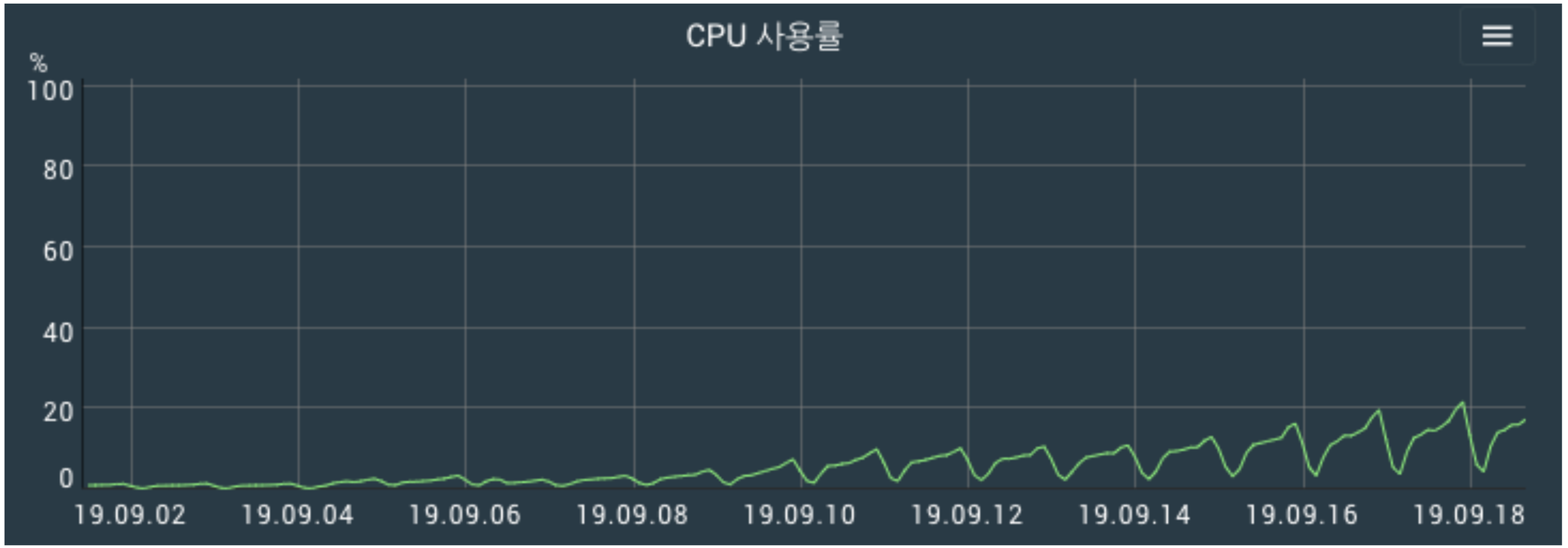
메모리의 계단형 증가



3.2 메모리 누수 확인하고 해결하기

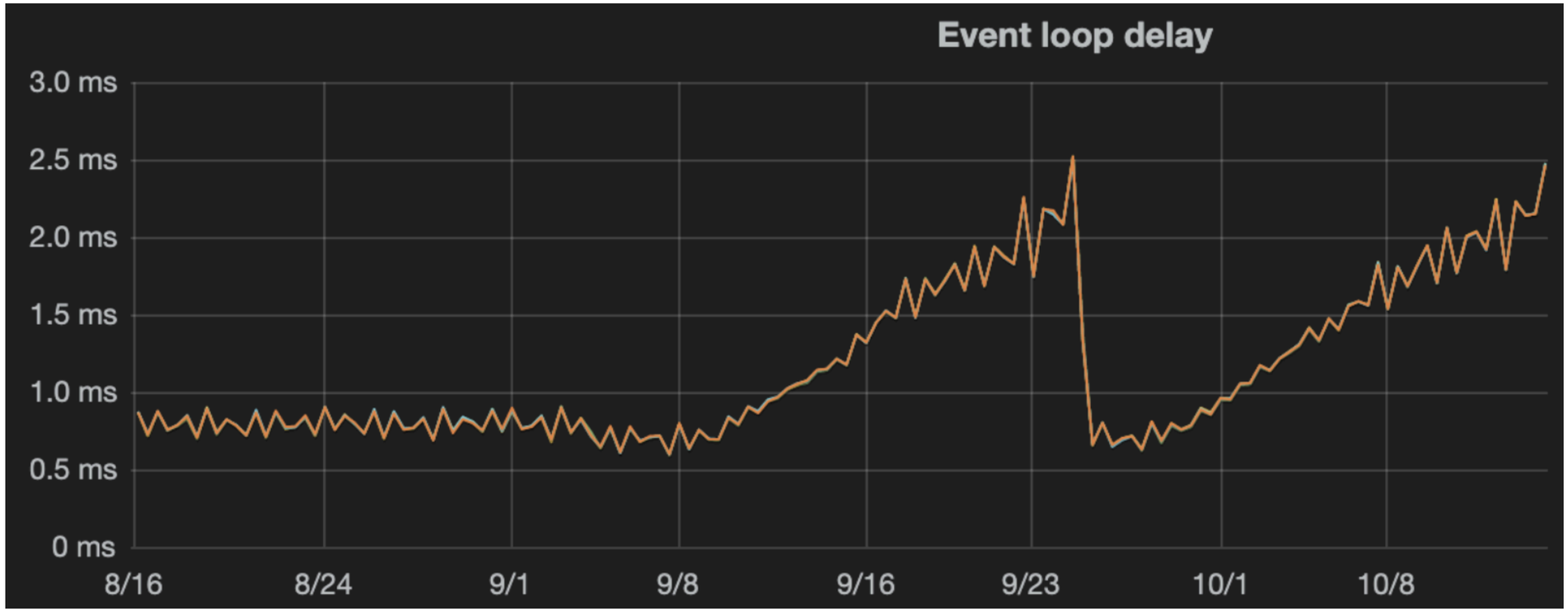
DEVIEW 2020

CPU의 계단형 증가



3.2 메모리 누수 확인하고 해결하기

Event Loop Delay 시간도 증가. 응답속도도 떨어진다.



3.2 메모리 누수 확인하고 해결하기

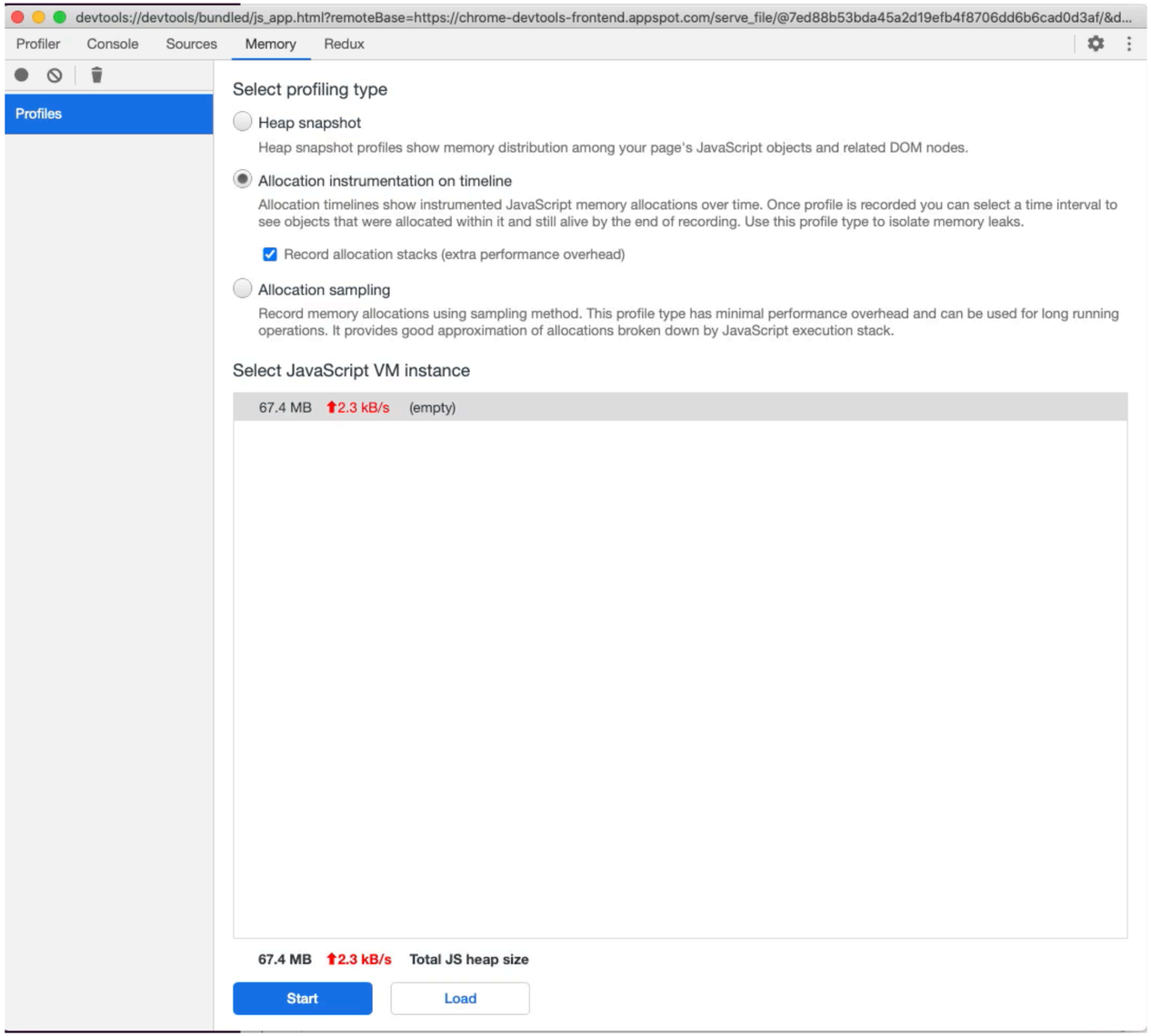
메모리 누수 해결하기

= 계속 증가하면서 잔존하는 객체 찾아 제거하기

참고: https://sculove.github.io/slides/memory/#/

3.2 메모리 누수 확인하고 해결하기

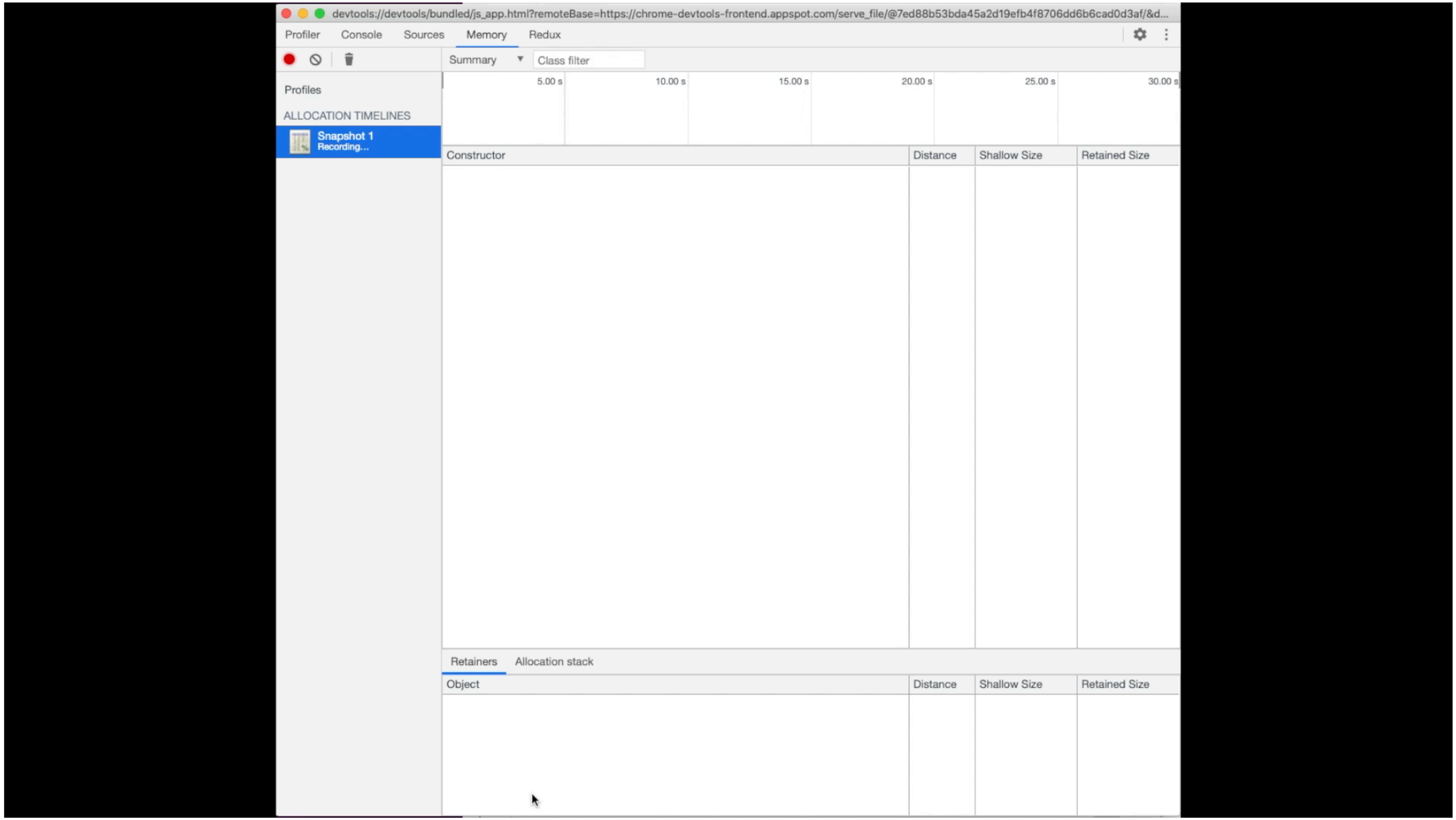
크롬 브라우저의 memory > Allocation instrumentation on timeline 기능으로 찾기



Node 12.0부터 `—heapsnapshot-signal` 옵션을 통해 heap dump 가능

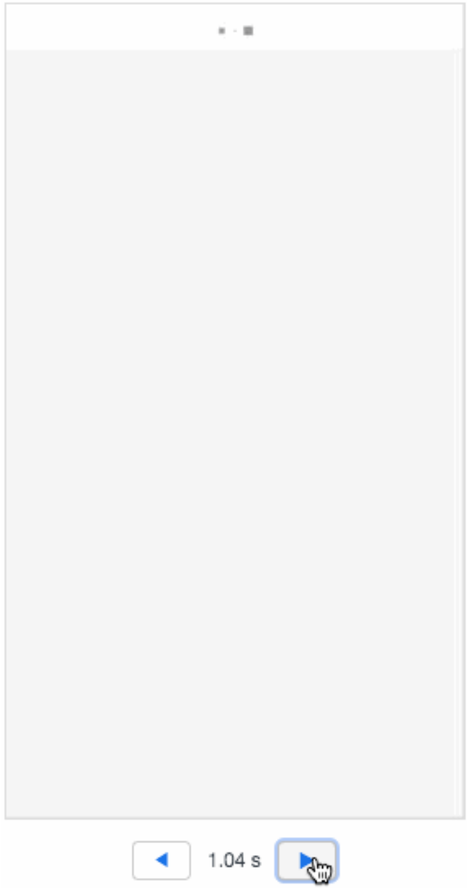
3.2 메모리 누수 확인하고 해결하기

DEVIEW 2020



3.3 SSR환경에서 FE 성능

SSR 방식



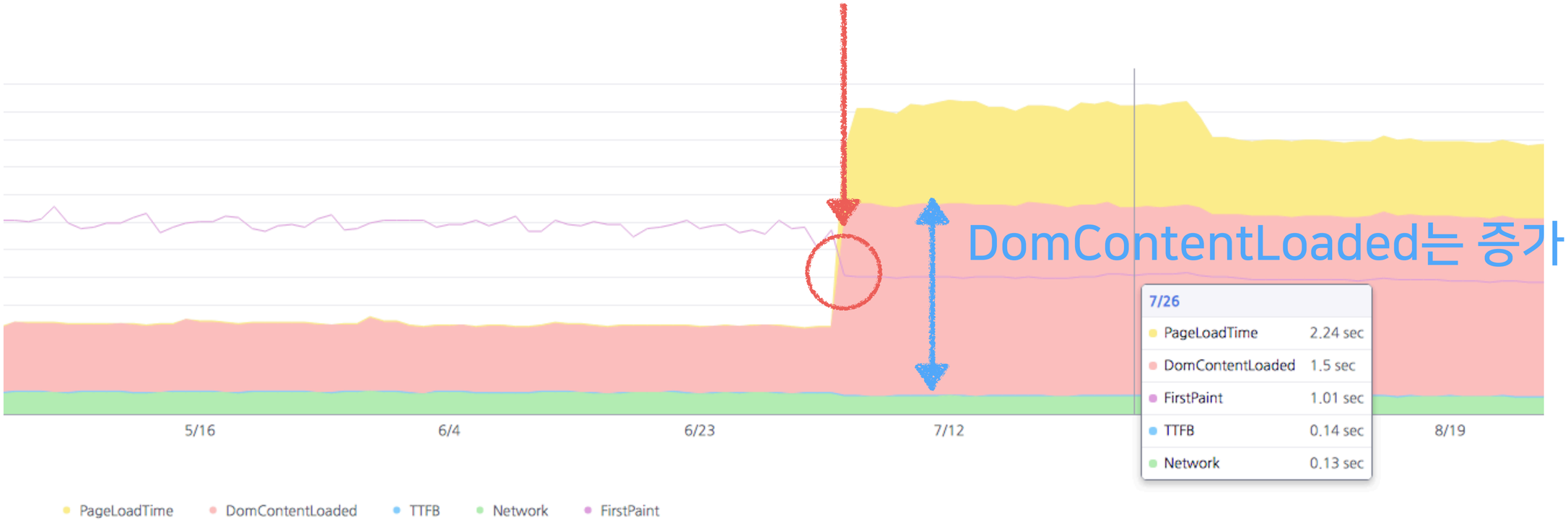
CSR 방식

3.3 SSR환경에서 FE 성능

DEVIEW 2020

전체 로딩 속도 증가

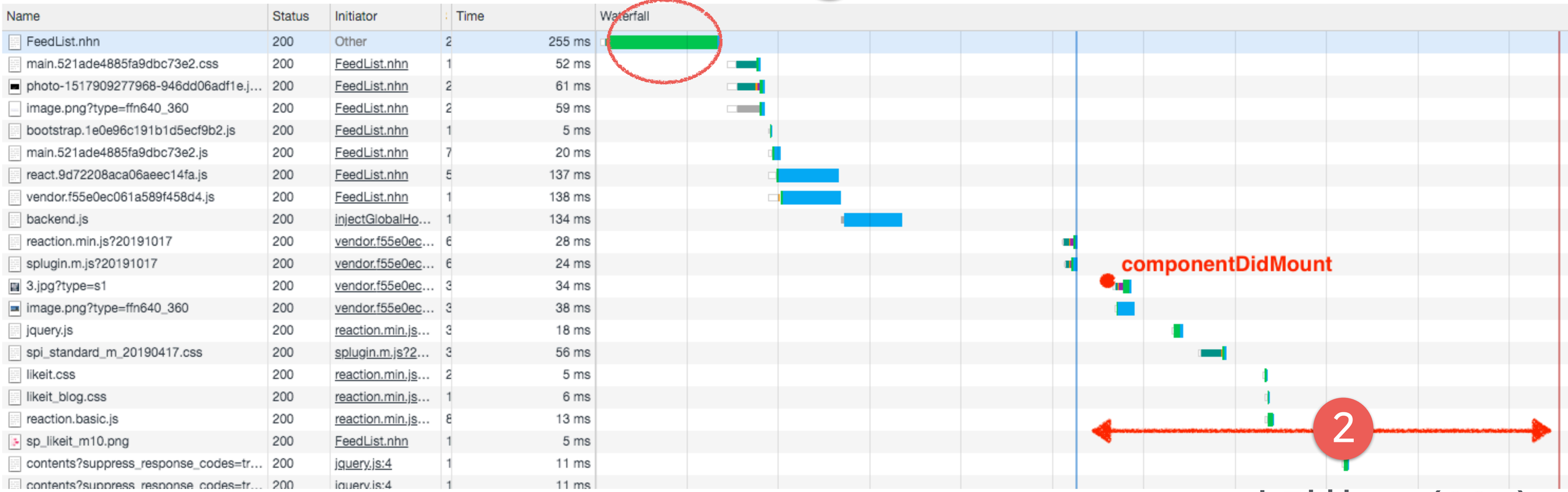
First Paint는 감소



3.3 SSR환경에서 FE 성능

DEVIEW 2020

1 서버에서 HTML을 만드는 시간 (SSR)

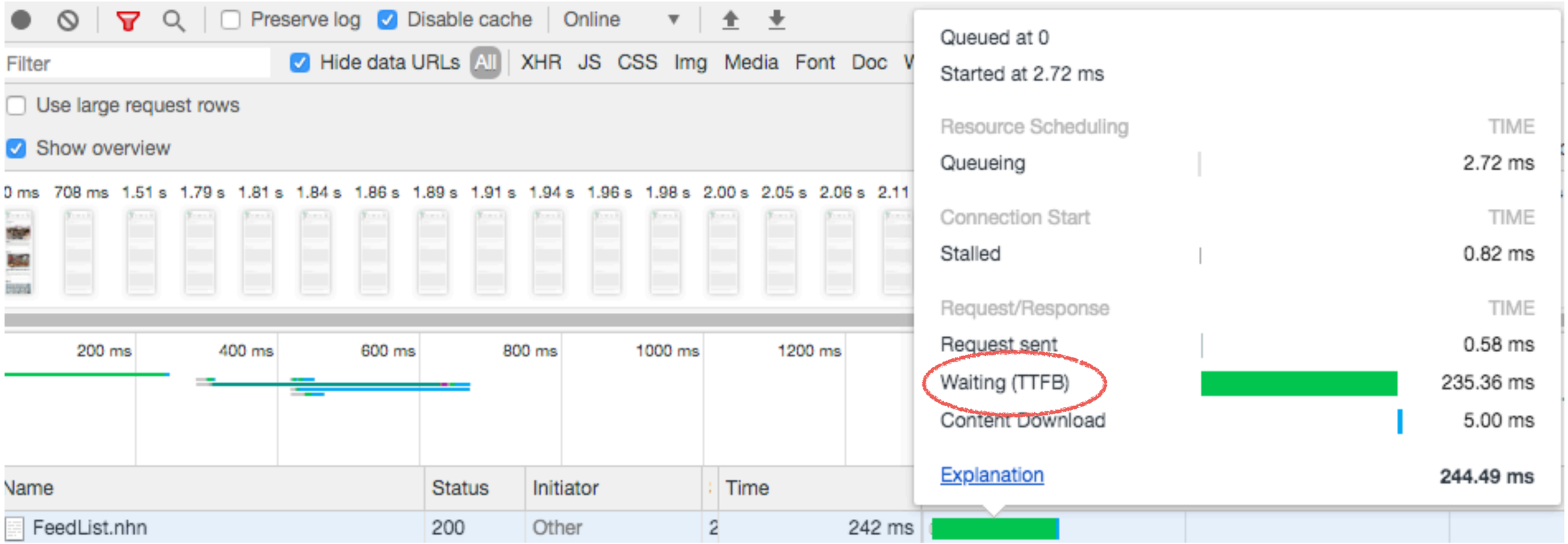


Name	Status	Initiator	Time	Waterfall
FeedList.nhn	200	Other	255 ms	
main.521ade4885fa9dbc73e2.css	200	FeedList.nhn	52 ms	
photo-1517909277968-946dd06adf1e.j...	200	FeedList.nhn	61 ms	
image.png?type=ffn640_360	200	FeedList.nhn	59 ms	
bootstrap.1e0e96c191b1d5ecf9b2.js	200	FeedList.nhn	5 ms	
main.521ade4885fa9dbc73e2.js	200	FeedList.nhn	20 ms	
react.9d72208aca06aeec14fa.js	200	FeedList.nhn	137 ms	
vendor.f55e0ec061a589f458d4.js	200	FeedList.nhn	138 ms	
backend.js	200	injectGlobalHo...	134 ms	
reaction.min.js?20191017	200	vendor.f55e0ec...	28 ms	
splugin.m.js?20191017	200	vendor.f55e0ec...	24 ms	
3.jpg?type=s1	200	vendor.f55e0ec...	34 ms	
image.png?type=ffn640_360	200	vendor.f55e0ec...	38 ms	
jquery.js	200	reaction.min.js...	18 ms	
spi_standard_m_20190417.css	200	splugin.m.js?2...	56 ms	
likeit.css	200	reaction.min.js...	5 ms	
likeit_blog.css	200	reaction.min.js...	6 ms	
reaction.basic.js	200	reaction.min.js...	13 ms	
sp_likeit_m10.png	200	FeedList.nhn	5 ms	
contents?suppress_response_codes=tr...	200	jquery.js:4	11 ms	

React 초기화 + a (CSR)

3.3 SSR환경에서 FE 성능

DEVIEW 2020

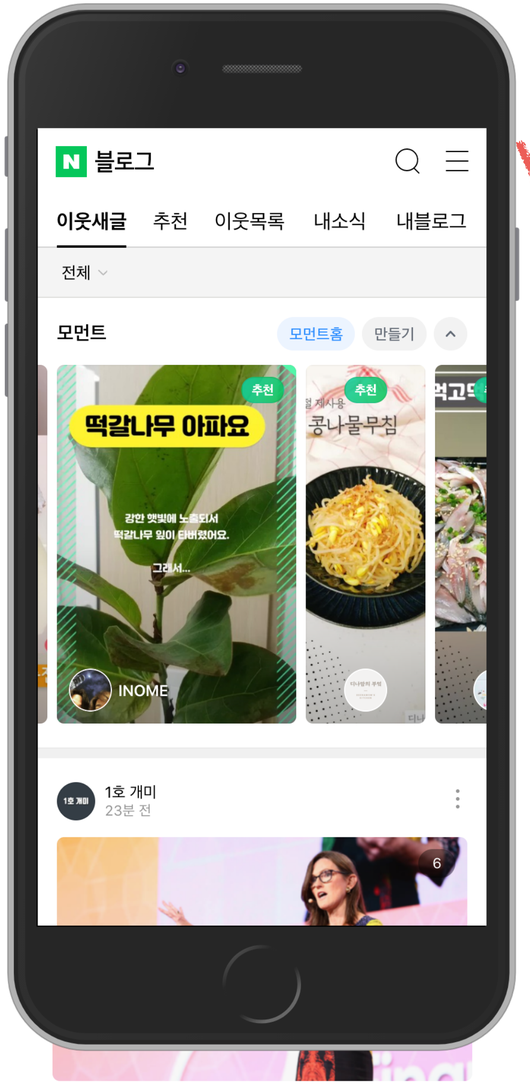


3.3 SSR환경에서 FE 성능

```
async function prefetch(store, qs) {
  const {groupId} = qs;

  await Promise.all([
    store.dispatch(getFeedList({groupId})),
    store.dispatch(getFeedVideoList({groupId})),
    store.dispatch(getHomeNotice()),
    store.dispatch(getSuggestionBlogNotice()),
    store.dispatch(getOfficialBlogNotice()),
    store.dispatch(getBuddyList())
  ]);
}
```

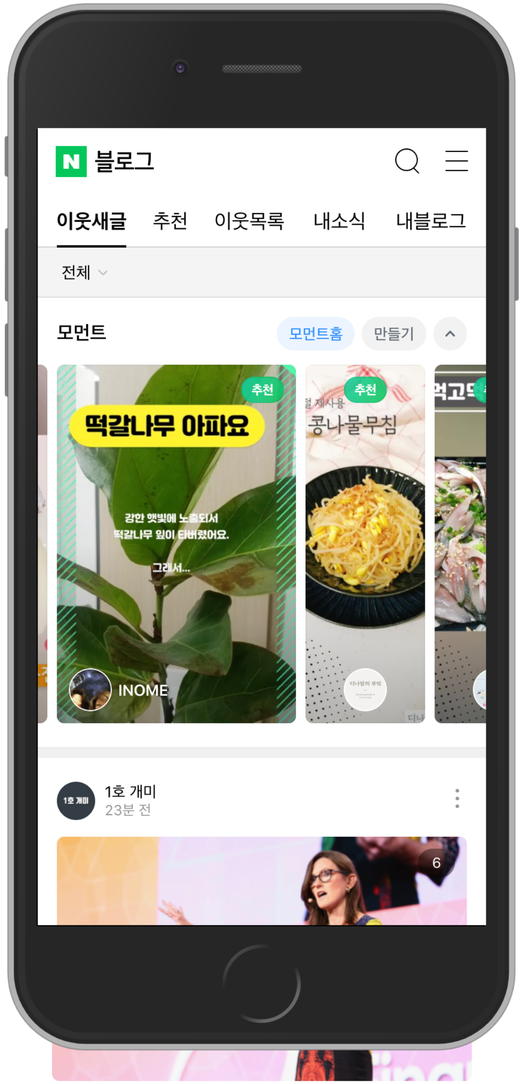
동일 요청이라면 SSR이 효과적



서버에서 처리 (SSR)

브라우저에서 처리 (CSR)

3.3 SSR환경에서 FE 성능



결국 케서린 누님 믿고 ARKK 투자 진행합니다.

스퀘어(SQ) 주식에 관심이 가기 시작했고공부를 하면서 매력에 이끌렸네요.주당 가격이 쌔다보니 단일 주식 매매는조금 ...

SSR! = 만들어진 페이지가 노출된다는 것!

= JavaScript 없이도 잘 동작하는 페이지

= JavaScript 로딩 자체를 load 이후로 보내도 나쁘지 않다…

3.3 SSR환경에서 FE 성능

DEVIEW 2020

Name	Status	Type	Initiator	Size	Time	Waterfall
film1982	200	document	Other	141 kB	171 ms	
20190802_155353_1_1_1_1.gi…	200	gif	film1982:-In…	(memory ca…	0 ms	
main.d4089436d0d4ae31d0f3…	200	stylesheet	film1982:-In…	(disk cache)	1 ms	
styles.e680c438c5386e52c72…	200	stylesheet	film1982	(disk cache)	2 ms	
home.2639868fce21b2b3113…	200	stylesheet	film1982	(disk cache)	34 ms	
grafolio-func.js	200	script	film1982	(memory ca…	0 ms	
SsangmunDongB.0e451c0be…	200	font	main.d4089…	(memory ca…	0 ms	
%EC%95%84%ED%86%B0…	200	jpeg	film1982	(memory ca…	0 ms	
%EB%84%A4%EC%9D%B4…	200	png	film1982	(memory ca…	0 ms	
post_color_icon.a2920522b18…	200	png	home.2639…	(memory ca…	0 ms	
bootstrap.a710e63ce8819440…	200	script	film1982:8	(disk cache)	1 ms	
react.f5cf41dea572d1889923.js	200	script	film1982:8	(disk cache)	12 ms	
main.d4089436d0d4ae31d0f3.js	200	script	film1982:8	(disk cache)	16 ms	
vendor.f9093906e15abfe71aff.js	200	script	film1982:8	(disk cache)	17 ms	
styles.e680c438c5386e52c72…	200	script	film1982:8	(disk cache)	5 ms	
vendors~home.9dc048f95bd1…	200	script	film1982:8	(disk cache)	6 ms	
home.2639868fce21b2b3113…	200	script	film1982:8	(disk cache)	23 ms	
webplayer_4.18.40.all.26b7e2…	200	script	async-scrip…	(memory ca…	0 ms	
splugin.m.js?20201020	200	script	async-scrip…	(disk cache)	2 ms	
ssp.web.sdk.js	200	script	async-scrip…	(memory ca…	0 ms	
m	204	text/plain	influencer.ni…	195 B	9 ms	
m?u=https%3A%2F%2Fin.na…	200	gif	lcslog_v0.8.…	530 B	26 ms	
challenges?ownerId=1182725…	200	xhr	breadcrumb…	11.3 kB	19 ms	

SSR

HTML + CSS
(이미지 + 필수 js lib)

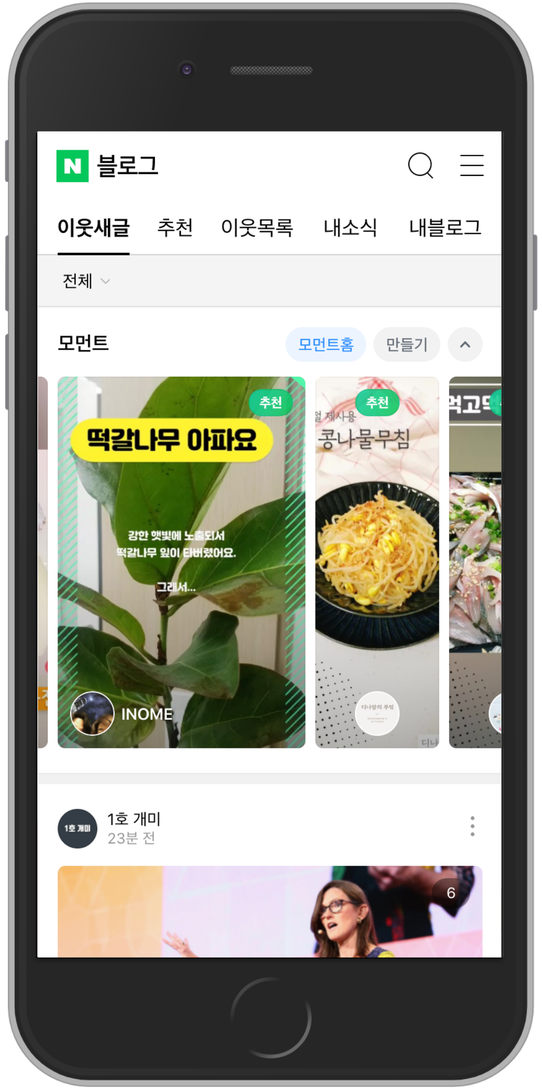
CSR

서비스 JS 로딩/실행

3.4 SSR 을 사용하면 생산성이 높다?

3

브라우저
히스토리 백 버튼



"추천" 클릭

2

Node.js SSR 서버

1

4

3.4 SSR 을 사용하면 생산성이 높다?

DEVIEW 2020

단일 코드로 개발 생산성 향상

- 단일 언어 클라이언트와 서버의 중복 로직 제거

if. 숙련된다면…

4. 적용 후기

4.1 실제로 적용해보니

UI 설계를 함께 FE와 논의해서 좋았다.

개발로서 역량을 높일 수 있어서 좋았다.

BE개발자가 2달 걸릴 일을 FE 개발자가

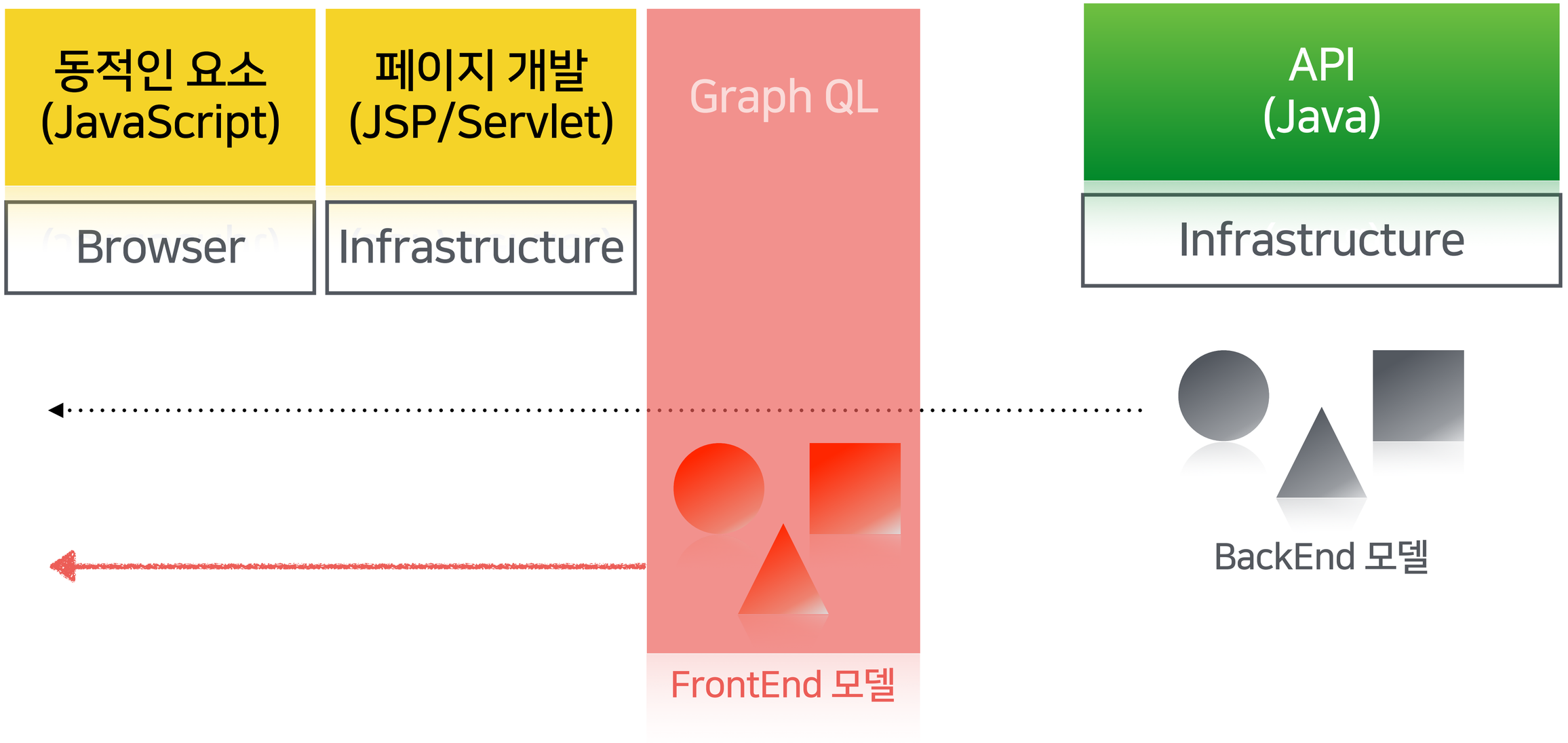
1달만에 처리해줘서 좋았다.

좋았다. 좋았다. 좋았다

...

4.1 실제로 적용해보니

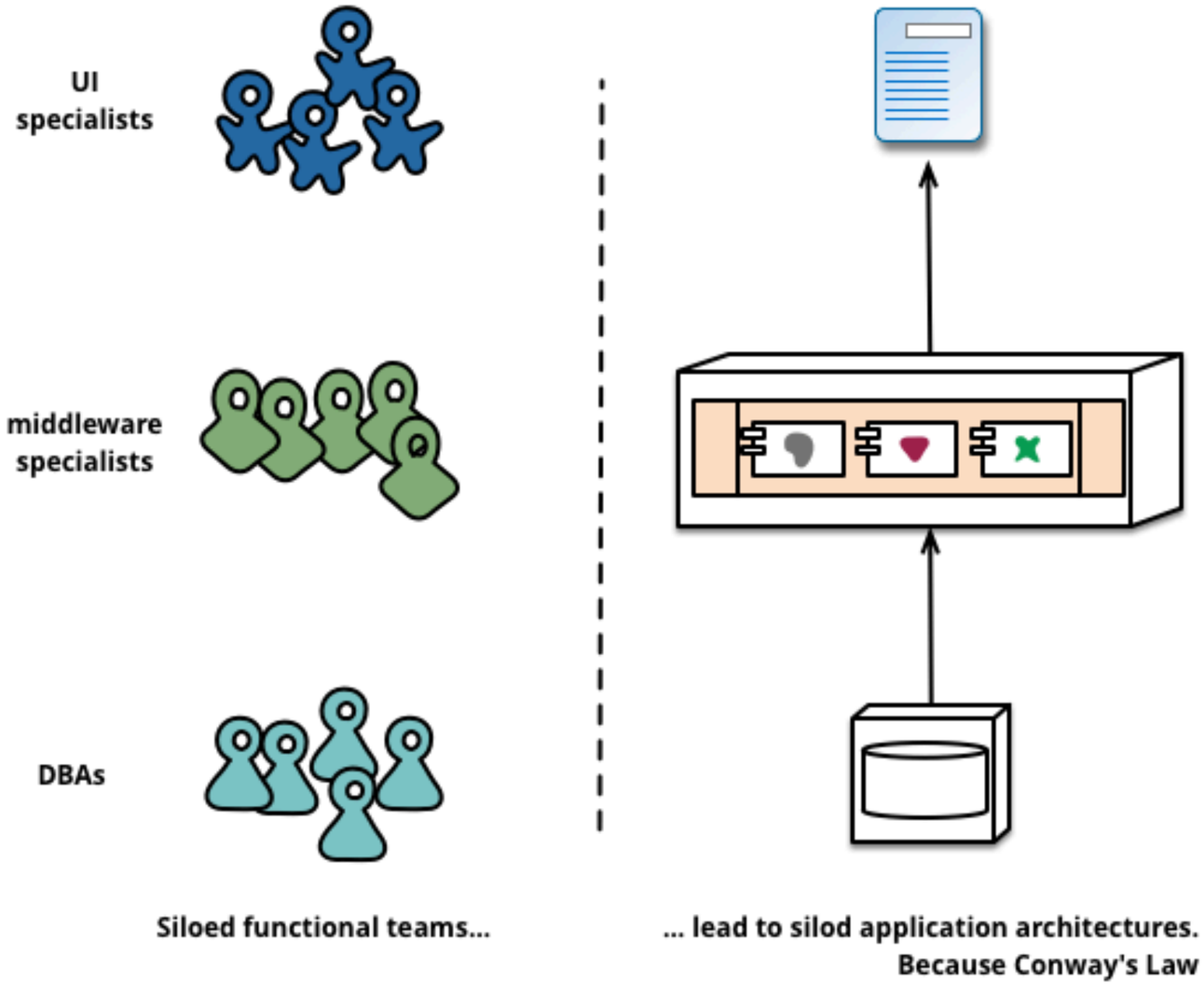
4.2 아쉬운 면도 있었어요.



4.3 저희도 SSR 적용하면 그렇게 될수 있나요?

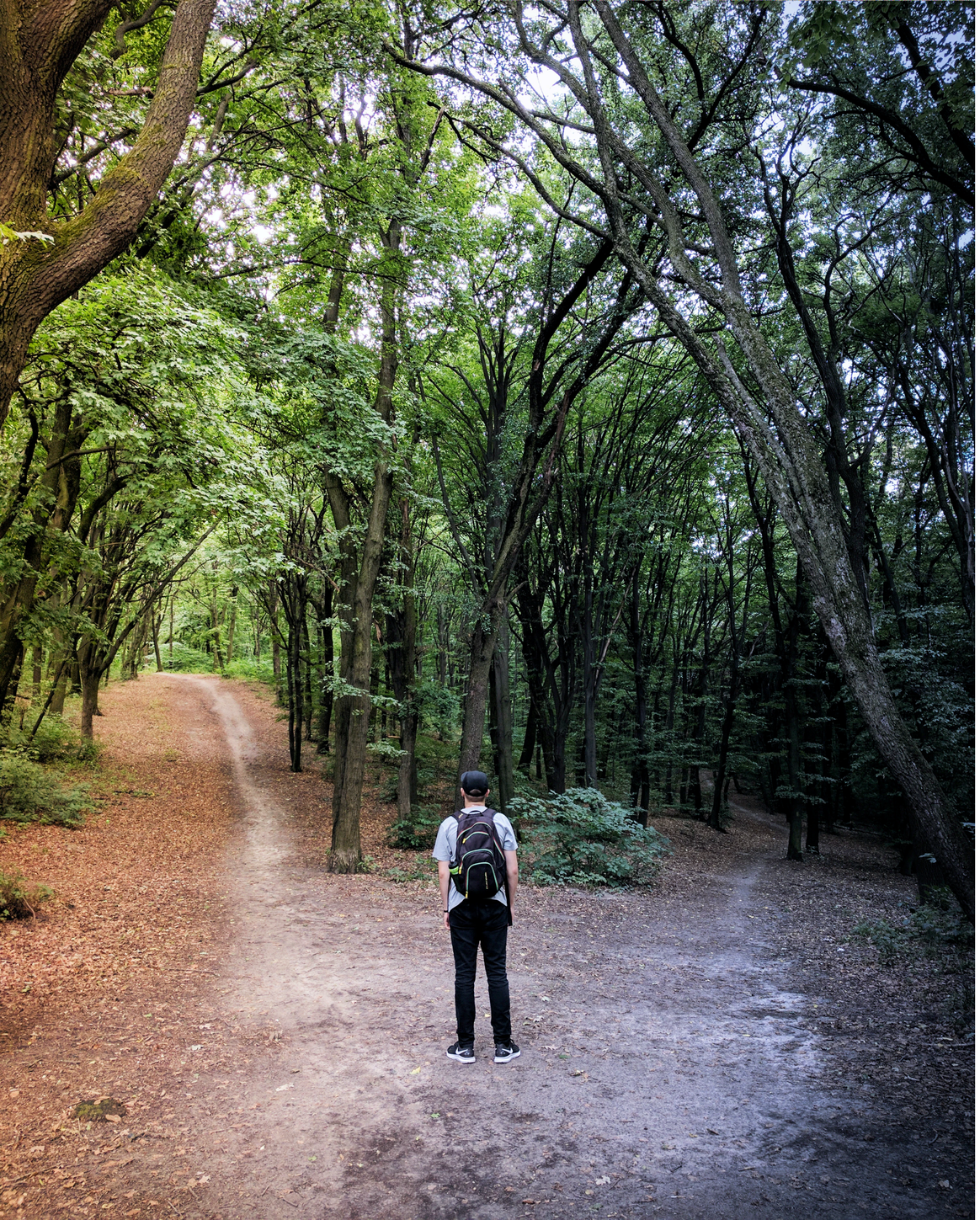
DEVIEW 2020

글쎄요.



Conway's law

시스템을 설계하는 조직은 필연적으로 해당 조직의 커뮤니케이션 구조를 복제한 산출물을 만든다



DEVIEW 2020

기술의 선택은
현재 상황에 적합한 방법을
찾는게 가장 중요합니다.

Q & A

Thank You