

블로그로 “진짜 개발자”처럼 보이는 법

내가 블로그를 쓰는 방식과 노하우



김성현 SungHyun Kim

witch.work

soakdma37@gmail.com

발표자 소개



김성현

- “마녀”, 블로그 witch.work 운영중
- solved.ac [Diamond IV](#)
- 서강대 기계/컴퓨터 복수전공 졸업
- 일단은 티맥스핀테크 프론트엔드 연구원
- 구직중. 연락주세요

오늘 이야기할 것

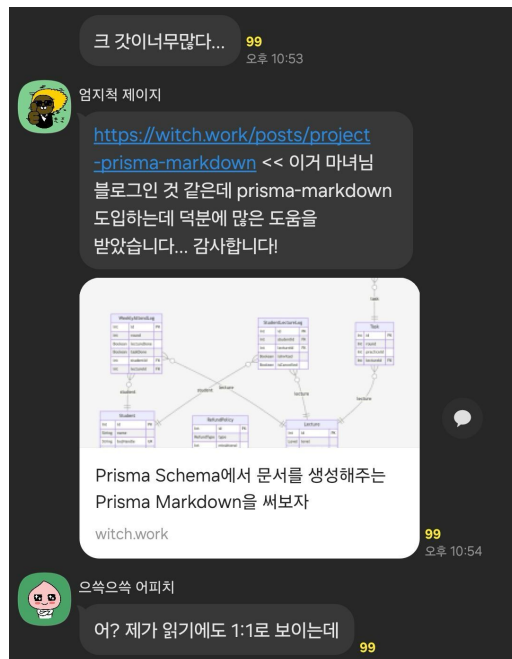
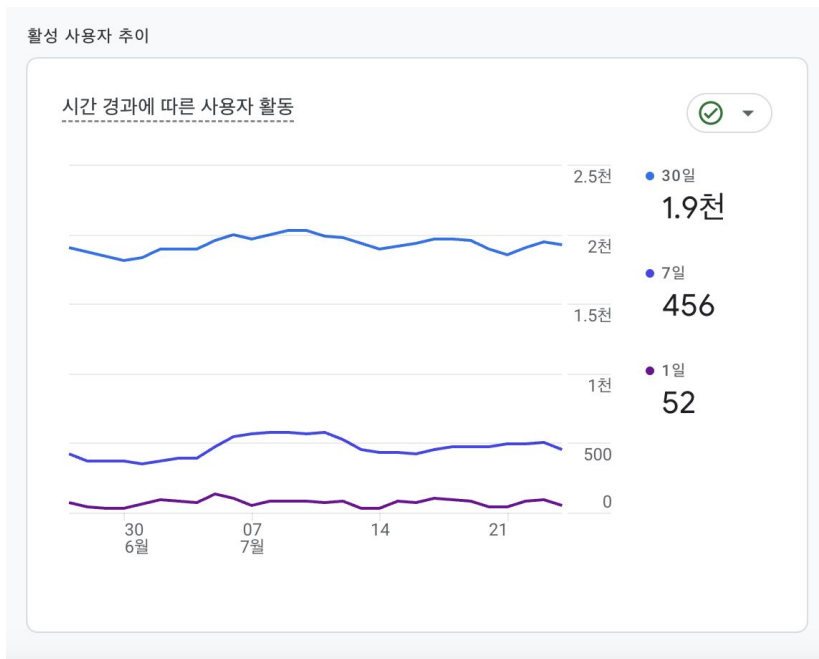
- 개발자에게 블로그는 어떤 의미인가?
- 블로그 글쓰기의 길
 - 탐구하는 법
- 더 할 수 있는 건 없을까?
 - 더 하고 기록하는 법
- 중간중간 예시(나)

넌 뭐했는데?

- 일단 글 많이 썼음
 - 2021년부터 써서 2024-07-25 현재 블로그 글 207개
- 내 입으로 말하긴 그렇지만 잘 썼음
 - 네이버 FE News(스타 5.5K) 2024-02 읽을거리 선정(JS의 주석은 //과 /* */뿐만이 아니다)
 - 개발자들이 개발 블로그 쓰는 모임 '글또' 9기(약 450명 규모)에서 주간 큐레이션 10번 중 4번 선정됨

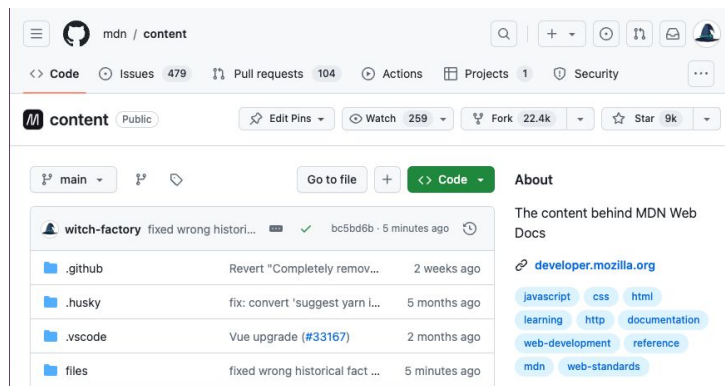
넌 뭐했는데?

- 블로그 조회수도 월 1500~2000은 나오고 언급도 조금은 되는 듯 함



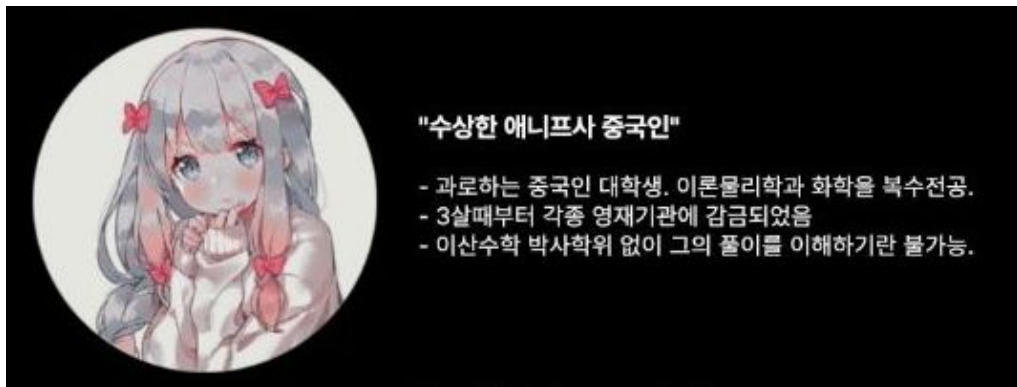
넌 뭐했는데?

- 그러면서 이것저것 했음
 - NextJS로 블로그 직접 제작(+성능 최적화, 홈서버 배포 etc.)
 - Javascript의 역사를 다룬 문서(약 140쪽) 번역 + 관련 발표
 - 사파리의 Javascript 엔진 코드 기여(주석이지만)
 - MDN 문서 번역 기여
 - MDN 원본 문서 기여
 - 이외 자잘한 오픈소스 기여들 몇 개

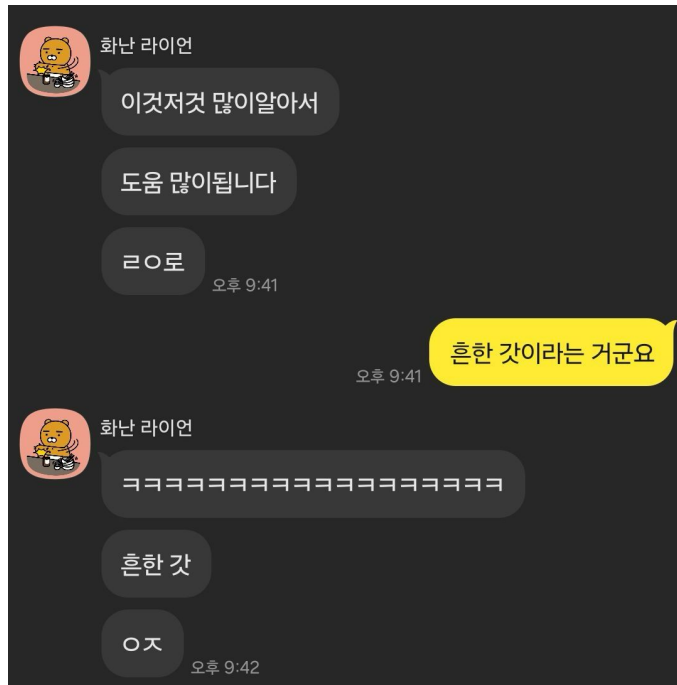


개발자에게 블로그란

날고 기는 개발자들이 너무 많다



고수의 상징
“수상한 애니프사 중국인”
유명한 오픈소스 기여자
목록에 반드시 있다



갓이 너무 많아서 그냥 흔해짐

그럼 우린?



우리도 먹고는 살아야지

- 여러 전략이 있을 수 있다
 - 개발 실력 강조
 - 도메인 지식이나 자신만의 배경
 - 소프트 스킬
 - 혹은 어떤 성격적인 강점일 수도
- 어쨌든 “좋은 개발자”임을 어필하기 위한 노력

좋은 개발자란?



이런 모습?



소프트 스킬과
관련된 무언가일지도

좋은 개발자가 뭔지는 몰라도

- 개발 실력을 보여주는 무언가가 있다고 많은 사람들이 믿음
 - vim을 쓴다거나

마우스 쓰지 말란 말이야!

- 요즘 핫한 기술이나 언어를 공부한다거나
- 알고리즘을 좋아한다거나 유명한 오픈소스에 기여했거나
- 이 장식 없는 PPT도 그렇게 보일지도

좋은 개발자가 뭔지는 몰라도

- 개발 실력을 보여주는 무언가가 있다고 많은 사람들이 믿음

- vim을 쓴다거나

마우스 쓰지 말란 말이야!

- 요즘 핫한 기술이나 언어를 공부한다거나
- 알고리즘을 좋아한다거나 유명한 오픈소스에 기여했거나
- 이 장식 없는 PPT도 그렇게 보일지도
- 참고로 블로그도 그렇게 보이는 좋은 수단 중 하나

당신은 무엇을 가지고 있는가?

- 자신을 보여줄 수 있는 포인트는 계속 고민해야
- 블로그는 하나의 수단이다
- 그리고 블로그로 뭘 보여줄지도 자기가 만들어 나가기 나름

블로그로 나를 보여주기

- 블로그는 자신을 압축적으로 보여줄 수 있는 수단
 - 블로그를 직접 만든다면 그 자체로 프로젝트
 - 블로그 글 제목만 봐도 공부 방식이 보임

블로그는 나의 모든 하드 스킬을 담을 수 있다

- 코드만으로는 드러나기 쉽지 않은 부분도 블로그에서는 가능
 - “그 코드까지 가면서 거친 길과 생각들”
 - 내가 버그 고치면서 한 고생들을 어떻게 보여줄 것인가?

블로그는 나의 모든 하드 스킬을 담을 수 있다

- 코드만으로는 드러나기 쉽지 않은 부분도 블로그에서는 가능
 - “그 코드까지 가면서 거친 길과 생각들”
 - 내가 버그 고치면서 한 고생들을 어떻게 보여줄 것인가?
 - 당연히 글로 쓰면 된다

NextJS metadata 오류 해결

2023. 09. 13

[front](#) [blog](#)

목차

1. 시작
 2. 문제 해결 시도
 - 2.1. 클라이언트 컴포넌트 분리
 - 2.2. 라우트 변경해보기
 - 2.3. import의 문제
 3. 후속조치
- [참고](#)

프로젝트 트러블 슈팅 - react-router-dom child route 만들기

2022. 08. 19

[web](#) [study](#) [front](#) [HTML](#) [react](#)

목차

1. 문제의 발생
 - 2-1. 가능한 이유
 3. 좀더 복잡한 경우의 문제 - parent route도 다룰 경우
 - 3.1 해결1 - 비 URL, 자식 컴포넌트를 만들기
 - 3.2 해결2 - index 사용하기
- [참고](#)

블로그는 나의 모든 하드 스킬을 담을 수 있다

- 면접관들은 내 코드까지 봐줄 시간 없음
 - 근데 블로그에 버그 수정했다는 제목만 100개 있으면?
- 혼자 공부한 것도 얼마든지 적을 수 있음



일단 해보라구요? UX를 읽으며 메모

2023. 12. 02

study

목차

감상

메모

21~22쪽

87쪽

148쪽

156쪽

166~167쪽

179쪽

184쪽

210쪽

227쪽

내가 책 읽고
썼던 글

이 책 읽은 걸 이력서에
이력으로 쓸 순 없잖아
근데 블로그엔 가능

기술 말고 다른 게 강점이라면 그걸 써라

- 기술적이지 않은 내용도 많이 적음
 - 본인의 인간성이나 소프트 스킬을 녹여낼 수도
 - 모범적이었던 갈등 해결이 있다면 쓸 수도
 - 자신이 어떤 걸 중시하는 사람인지도 쓸 수 있다
 - ~~“우당탕탕 n년차 주니어 개발자 회고”~~

우당탕탕 개발자 회고

전체 이미지 뉴스 동영상 쇼핑 도서 웹 더보기

 velog
<https://velog.io> > 한-달차-주니어-개발자의-우당탕탕-회고


한 달차 주니어 개발자의 우당탕탕 회고

2024. 5. 16. — 한 달차 주니어 개발자의 우당탕탕 회고 · 드디어 개발을 시작! 그 제대로 찍는 법을 배웠다. · 하지만 아직 길길 ...

 브런치스토리
<https://brunch.co.kr> > ...

우당탕탕 2022년, PM 회고하기 (1)

PM은 서비스 기획팀, 그리고 개발자는 개발 1팀과 개발 2팀으로 묶여 서로 다른 하는 형태였다. 장점은, 각 개발 팀에 PM이 두 명 혹은 ...

 velog
<https://velog.io> > 취업-후-2주가-된-주니어-개발자의-일기

2주차 주니어 개발자의 우당탕탕 회고

2024. 4. 26. — 취뽀한지 2주가 지났다. 걱정했던 것과 다르게, 나는 아직 큰 / 중이다. 물론 신입인 지금. 애초에 큰 사고를 칠 만한 ...

 주식회사 윙잇
<https://company.wingeat.com> > story

서비스기획 우당탕탕 2022년, PM 회고하기 : 스토리

2023. 3. 21. — 처음 입사했을 당시에는 기능 조직으로만 구성이 되어있었다. 개발자는 개발 1팀과 개발 2팀으로 묶여 서로 다른 조직의 팀원 ...

 LinkedIn
<https://kr.linkedin.com> > pulse > 신입개발자의-우당탕탕...

신입개발자의 우당탕탕회고

2023. 4. 23. — 신입 프론트엔드개발자가 어떤 감정을 느끼고 생각했는지를 우 경험에 대한 회고글을 작성했습니다

뭘 보여줄지는 당신이 정해야 한다

- 직접 짠 블로그에서 기술적인 글을 적을 수도 있다
- 벨로그나 티스토리도 후딱 만들어도 컨텐츠만 있다면야
 - TIL을 성실하게 적을 수도
 - 자기가 짠 코드나 버그픽스만 짧게 요약해 올릴 수도
 - 회고나 자신이 삶에서 느낀 것들을 보여줄 수도

뭘 보여줄지는 당신이 정해야 한다

- 직접 짠 블로그에서 기술적인 글을 적을 수도 있다
- 벨로그나 티스토리도 후딱 만들어도 컨텐츠만 있다면야
 - TIL을 성실하게 적을 수도
 - 자기가 짠 코드나 버그픽스만 짧게 요약해 올릴 수도
 - 회고나 자신이 삶에서 느낀 것들을 보여줄 수도
- 심지어 “글로 남긴다”는 그 자체도 하나의 강점이고 보여줄 수 있는 것

단, 기회비용을 생각하라

- 나는 블로그를 택해서 나름 멀리 왔음
- 그래서 이렇게 발표를 하고 있다
- 하지만 이것도 분명 시간과 노력이 든다
- 당신에게 개발이 오로지 실용과 문제 해결과 커뮤니케이션이라면 블로그를 쓰는 건 오히려 시간 지연일 수도 있음

블로그에 얼마나 시간을 부을지 선택

- 나는 솔직히 개발자에게 세속적인 판단을 요구하는 기류를 좋아하지 않음
- 개발은 개발로서 아름답다ㅋㅋ
- 그래서 나 혼자 혹은 소수의 “진짜”들과 함께하는 이런 방식이 좋았음
- 그럼 당신은 개발에서 뭘 얻고 싶은가?
- 당신은 어떤 모습으로 보이고 싶은가?

이 발표는 그냥 나의 방식

- 나는 이걸로 갈 수 있는 데까지 가기로 했다
- 그래서 블로그를 쓸 생각이 있다면 조금 앞서서 가본 경험을 전해줄 수 있다
- 여기서는 내가 택한 방식을 보여드릴 뿐
- 다음은 직접 선택해야

**좋은 글을 위해
뭐라도 쓰기**

일단, 좀 시작해라

- 일단 뭐든 좀 쓰라고

C - 1.3. 두번째 예제의 개선

C언어 1.3 두번째 예제의 개선

language

2021. 07. 31

C - 1.2. 두번째 예제

C언어 1.2 두번째 예제

language

2021. 07. 02

C - 1.1. 시작

C언어 1.1 시작

language

2021. 06. 25

프론트 지식 익히기 tailwind - 3

프로젝트 tailwind 사용기 - 태그 정리 2

web study front

2022. 07. 29

프론트 지식 익히기 tailwind - 2

프로젝트 tailwind 사용기 - 태그 정리

web study front

2022. 07. 28

프론트 지식 익히기 tailwind - 1

프로젝트에서 tailwind를 사용하는 기록

web study front

2022. 07. 27

- 내가 2021, 2022년에 쓴 글들
- 책이나 문서를 베껴 적다시피 한 글
- 차마 좋은 글이라고는 할 수 없고 부끄럽다
- 그럼 안 부끄러운 글은 언제 쓰는데?

그런 글쓰기는 의미 없다고?

- TIL을 쓰거나 회고를 하거나 1일 1커밋을 하거나...
 - 의미 없다는 말도 많지만, 쓰레기라도 쌓아야 그 위에 선다

그런 글쓰기는 의미 없다고?

- TIL을 쓰거나 회고를 하거나 1일 1커밋을 하거나...
 - 의미 없다는 말도 많지만, 쓰레기라도 쌓아야 그 위에 선다
 - **그래도 복붙글은 쓰지 말고 직접 타이핑이라도 해라**

**깊은 글을 위해
질문 던지기**

자 다시, 사람들이 뭘 본다고?

- 개발 실력을 보여주는 무언가가 있다고 많은 사람들이 믿음
 - vim을 쓴다거나

마우스 쓰지 말란 말이야!

- 요즘 핫한 기술이나 언어를 공부한다거나
- 알고리즘을 좋아한다거나 유명한 오픈소스에 기여했거나
- 이 장식 없는 PPT도 그렇게 보일지도

그럼 어떤 글이 “진짜” 같을까?

- 일단 누가 봐도 틀린 정보를 쓰면 안 됨
 - 예시) “에라토스테네스의 체”를 검색해 보면 틀린 정보들이 많다

2) i 가 소수인지 검증해야 한다. 이전에 내가 작성했던 코드는 1부터 i 까지의 값을 모두 비교하여 소수인지 찾아내었다. 하지만 '에라토스테네스의 체' 알고리즘으로 풀어보면 2부터 i 의 제곱근까지만 소수인지 검증하면 된다.

```
# 2부터 i의 제곱근까지만 소수인지 검증 → 빠르게 검증 가능
for j in range(2, int(i**0.5)+1):
    if i % j == 0:
        break
else:
    print(i)
```


그럼 어떤 글이 “진짜” 같을까?

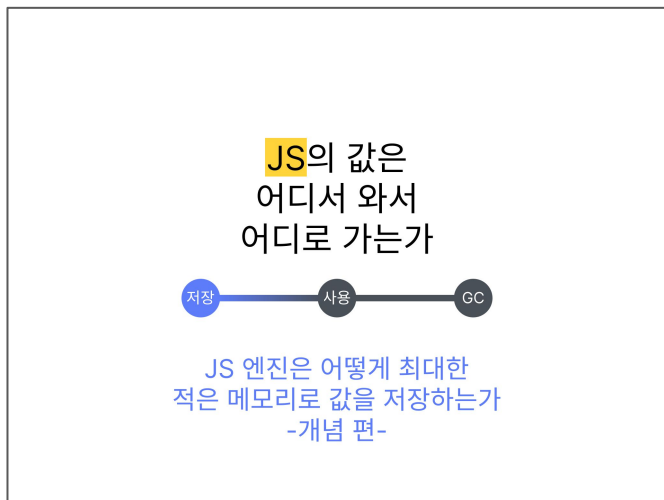
- 원인을 파헤쳐야 함
 - “이렇게 딸깍 하니까 오류 고쳐짐 띠용” <-절대 안됨
 - 뭐가 잘못됐었고 지금은 왜 되는지 샅샅이 알아내서 써야 함
- 남들이 안 하는 거 해야 함
 - Python<<<< Rust<<<< CoQ, Zig
 - 아무도 안할수록 진짜같음
 - 컴파일러 최적화 해킹 이런거 하면 누가봐도 “진짜”임

물론 그게 아니라고 나쁜 건 아님

- 사람들이 더 도움을 받는 건 깊고 근본적인 지식을 담은 게 아닐 수 있음
- 나도 빨리 작업해야 할 땐 탐구 없이 필요한 내용만 볼 때도 많음
- 실용의 관점에서 “진짜”같은 글은 지적 유희일 뿐일 수도

뭐든 장단이 있다

- 가르쳐주는 거 받아쓴 홈서버 세팅글 조회수 >>>>> JS 엔진 내부까지 싹 들여다보면서 온 정성을 다해 적은 글 조회수
- 대신 “알고리즘 1일차도 이해하는 카카오 코테 풀이” 쓰면 조회수 폭발할 것



아무도 안 보는
내 탐구 글 중 하나

어쨌든 그거 어떻게 하냐고



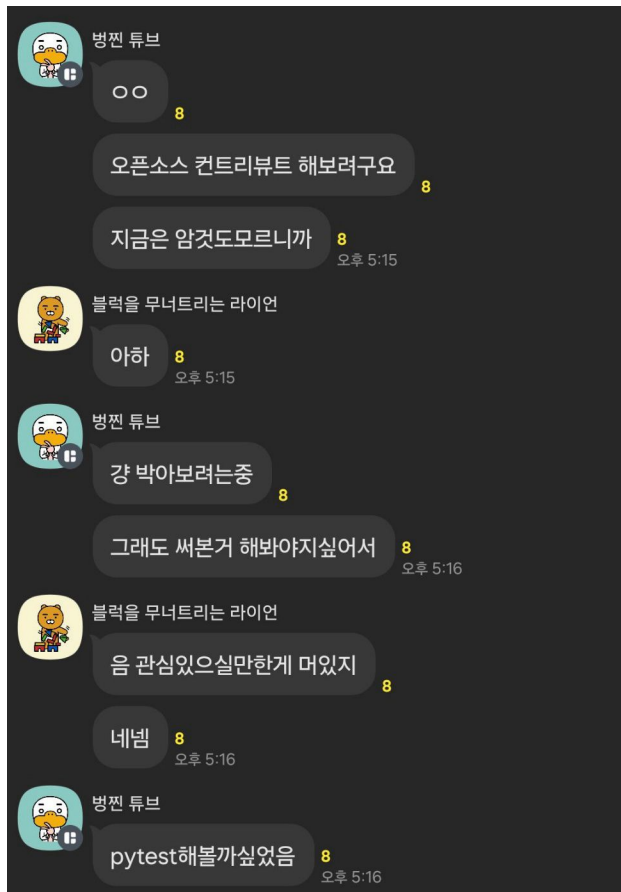
CTO가 커리어를 걷고 비트 레벨까지 내려가서 DB를 해킹했던 이야기

이런 근본 넘치는 글은
몇몇 개발자들의 로망

당장 비트 레벨 해킹을 할 순 없잖아

- 물론 그런 거 바로 하는 인간들도 있긴 함

오픈소스 가져다 쓰기가 아니라
기여부터 하신다구요?



그럼 우린?



어려운 걸 하는 게 핵심이 아니다

- 남들보다 딱 한 걸음만 더 가고
- 질문 하나만 더 던져서 깊이 가보는 것
- 그러다 보면 어려워질걸?

2022년 어느 날 React 관련 영상을 보다가

- 2022년, useState를 몇 번 썼는지 이미 셀 수도 없을 때
- 이런 코드짬은 수십수백 번 째던 상태

```
import { useState } from 'react';

export default function Counter() {
  const [count, setCount] = useState(0);

  function handleClick() {
    setCount(count + 1);
  }

  return (
    <button onClick={handleClick}>
      You pressed me {count} times
    </button>
  );
}
```


그런데 유튜버가 질문을 던진다

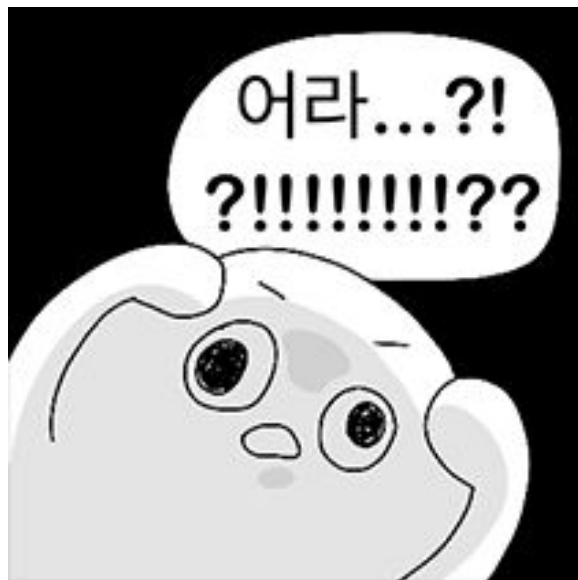
- “useState의 리턴값은 const인데 왜 state 값은 변경될까요?”

```
import { useState } from 'react';

export default function Counter() {
  const [count, setCount] = useState(0);

  function handleClick() {
    setCount(count + 1);
  }

  return (
    <button onClick={handleClick}>
      You pressed me {count} times
    </button>
  );
}
```



어려운 걸 하는 게 핵심이 아니다

- React 써본 프론트치고 useState 안 써본 사람 없다
- 근데 조금만 더 알아보면 수많은 깊은 지식이 튀어나옴
 - JS의 배열 동작 방식
 - useState 내부 구현(이해하려면 의존성 주입도 알아야)
 - 클로저(실제 내부 로직에 쓰임)
 - 일급 객체 함수
- 딱 한 걸음만 갔더니 벌써 꽤 어려워졌음

그러니 간단한 거라도 일단 질문부터



- 내가 던졌던 질문이라면...
- React useReducer는 어디 쓰죠?
- 이분 탐색 구현 방식별 차이는 뭐죠?
- JS에 Symbol 자료형은 왜 있죠?
- JS 배열에는 어떻게 push같은 게 있죠?
- 이외 여럿

간단한 것들도 파고들면 깊어진다

- React에는 useReducer라는 훅이 있다.
 - 그런데 이거 많이 쓰는 거 본 사람? 난 없다
 - 그럼 이런 거 왜 있을까? 어디 쓸까?
 - 궁금해지면 바로 검색해서 글 쓰기 시작해라

간단한 것들도 파고들면 깊어진다

- JS의 배열은 push 메서드가 있다
 - 배열은 고정된 크기 아니었나?
 - 심지어 일반적으로 배열이 메서드 가지는 것도 말이 안됨. 뭐지?
 - 궁금해지면 바로 검색해서 글 쓰기 시작해라

파고드는 질문을 던지는 법

- 지식들을 연결하는 과정에서 일치하지 않는 부분을 찾는다
- 그러면 의문이 생길 때가 많음
- 다른 사람이 알아서 유튜브 같은 데서 질문을 던져 주기도 함
- 바로 그걸 알아보고 조사해서 블로그에 쓰면 됨
- 질문을 하는 능력과 질문의 깊이도 하면 할수록 점점 늘어남

아주 간단한 예시

- C를 처음 배우면 가장 먼저 나오는 printf

```
#include <stdio.h>

int main(void) {
    printf("Hello, World!\n");

    return 0;
}
```

질문의 시작

- 어? 근데 왜 “print”가 아니고 “printf”지?
- 내가 아는 “출력하다”는 print인데 뒤에 왜 f가 붙었을까?
 - 형식에 맞게 출력할 수 있다는 의미로 붙은 formatted의 약자
 - C의 모태가 된 B 언어에는 숫자를 출력하기 위한 printn등이 따로 있었고 구분을 위해 printf가 있었는데 여기서 온 것
 - 이외의 다른 이야기가 있을 수도 있음

C를 계속 배워 나감

- C를 좀 배우다가 이런 코드를 만났다고 하자

```
#include <stdio.h>

int main() {
    int a = 1, b = 2, c = 3;
    printf("%d %d %d\n", a, b, c);
    return 0;
}
```

printf의 귀환

- 어? 근데 왜 보통 함수랑 달리 printf는 인자 갯수가 안 정해져 있지?
- 보니까 scanf도 그렇다
 - printf, scanf는 C의 대표적인 가변 인자 함수
 - stdarg.h를 이용하면 직접 가변 인자 함수를 정의할 수도 있다

printf의 귀환

- 어? 근데 왜 보통 함수랑 달리 printf는 인자 갯수가 안 정해져 있지?
- 보니까 scanf도 그렇다
 - printf, scanf는 C의 대표적인 가변 인자 함수
 - stdarg.h를 이용하면 직접 가변 인자 함수를 정의할 수도 있다
 - printf, scanf, 함수 선언은 컴공 1학년 때 배우는 기초적인 내용
 - 그런데 기존에 알던 것들끼리 비교해 보는 것만으로 많이 깊어졌다!

끝인가?



그럴 리가



그럼 왜 print일까?

- 출력한다는 뜻으로 쓸 수 있는 단어가 print밖에 없나?
 - put도 더 원시적인 출력함수인 putchar()에 있음
 - write도 시스템 콜 함수로 존재함
 - 이외에도 display, output등 쓸 수 있는 단어는 많음
- 그럼 왜 print를 골랐고 나머지와와의 차이는 무엇인가?

그럼 왜 print일까?

- 출력한다는 뜻으로 쓸 수 있는 단어가 print밖에 없나?
 - put도 더 원시적인 출력함수인 putchar()에 있음
 - write도 시스템 콜 함수로 존재함
 - 이외에도 display, output등 쓸 수 있는 단어는 많음
- 그럼 왜 print를 골랐고 나머지와와의 차이는 무엇인가?
- 나도 모른다. 궁금해졌으면 검색해서 글 쓰기 시작해라

다른 간단한 예시

- 어느 날 보았던 three.js의 튜토리얼 코드
- 어? 생각해 보니 배열인데...왜 push가 있지? 게다가 const인데 내용이 변해!

```
const points = [];  
points.push( new THREE.Vector3( - 10, 0, 0 ) );  
points.push( new THREE.Vector3( 0, 10, 0 ) );  
points.push( new THREE.Vector3( 10, 0, 0 ) );  
  
const geometry = new THREE.BufferGeometry().setFromPoints( points );
```


JS 배열은 배열이 아니라

- 구글링하면서 찾은 답. JS 배열은 사실 객체란다

이처럼 자바스크립트의 배열은 엄밀히 말해 일반적 의미의 배열이 아니다. 자바스크립트의 배열은 일반적인 배열의 동작을 흉내낸 특수한 객체이다. 아래 예제를 살펴보자.

JAVASCRIPT

```
console.log(Object.getOwnPropertyDescriptors([1, 2, 3]));  
/*  
{  
  '0': { value: 1, writable: true, enumerable: true, configurable: true },  
  '1': { value: 2, writable: true, enumerable: true, configurable: true },  
  '2': { value: 3, writable: true, enumerable: true, configurable: true },  
  length: { value: 3, writable: true, enumerable: false, configurable: false }  
}  
*/
```

근데 그러면 느릴 텐데...

- 모던 자바스크립트 엔진에서는 배열을 다루기 위한 최적화를 구현
- 히든클래스랑 원소 타입이랑 최적화 많은데 아무튼 조사(아직도 다 모름)

[V8 Deep Dives] Understanding Array Internals

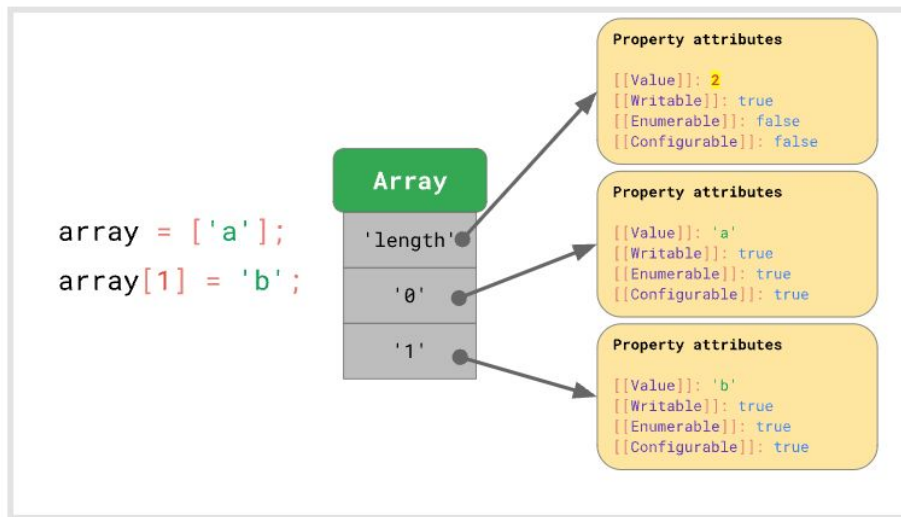


Andrey Pechkurov · Follow
Published in ITNEXT · 8 min read · Mar 16, 2021

👍 418 💬 6



Photo by Antonio Garcia on Unsplash



어디까지 깊어질 것인가

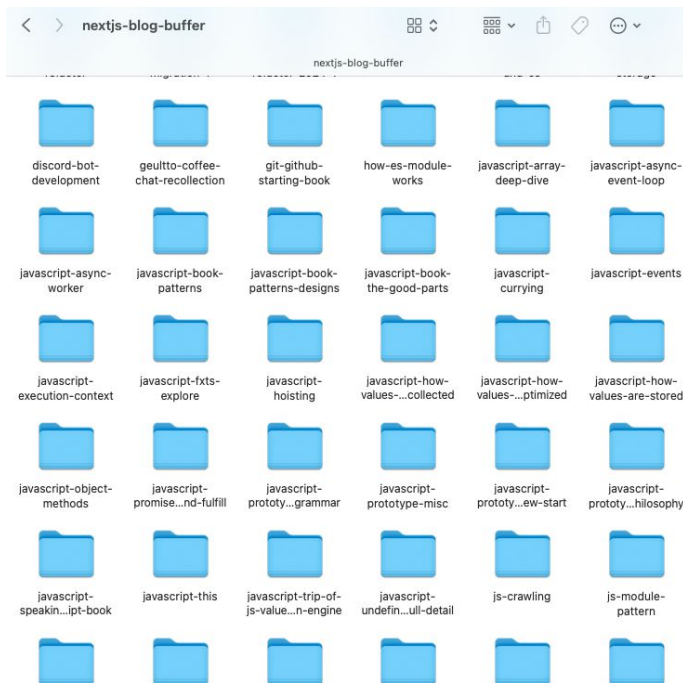
- 모든 걸 무한히 파고들 수는 없잖아

Fast I/O

어디까지 ~~빨라질~~ 것인가
깊어질

어디까지 깊어질 것인가

- 모든 걸 무한히 파고들 수는 없잖아



내가 파고들다가 보류하고 임시 저장해 놓은 글들 중 매우 일부 (“시간과 예산이 조금만 더 있었더라면...”)

스스로 시원해질 때까지라고 합시다

- printf에 왜 f가 붙은지 궁금해해야만 하는 건 아니다
 - 억지로 하나하나 태클을 거는 것도 비효율적이다
- 중요한 건 아는 것끼리 자연스럽게 충돌시키면서 의문을 가지는 “능력”
 - ‘내가 아는 출력은 print인데 왜 여기선 printf라고 할까?’
 - ‘나는 분명 배열이 고정 길이라고 배웠는데 왜 JS 배열은 길이가 달라질까?’
- 이런 의문을 죽어라 파면 “진짜”같은 글 완성

그리고 한 걸음씩만 더 나아가기

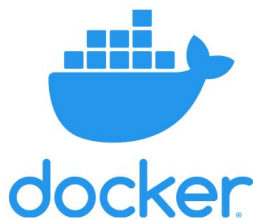
- 남들보다, 이전보다 한 걸음씩만 더 가다 보면 어느새 고여 있을 것



스스로의 기준이 점점 높아진다

- 나는 처음에 내가 보는 책 말고 다른 자료(블로그 등)를 찾는 게 한계였다

개발 인턴 Garden의 
즐거로운 **인턴**생활 ^_^



흔한 튜토리얼 글
(출처: 벨로그 어딘가)

스스로의 기준이 점점 높아진다

- 그러다가 MDN 문서나 기술 블로그를 보게 되었고
- 이내 MDN 번역본이 아니라 원문을 보게 된다



전체 개발 디자인

업무용 툴, 내 동료가 정말 만족했나요?

토스의 데이터 직군이 매일 쓰는 서비스의 만족도를 1점 끌어올리기 위해 10개월간 집요하게 파고든 과정을 소개할게요.

2024년 7월 24일 · 한지유



고성능 GPU 클러스터 도입기 #1: 요리하라고 해서 왔는데 프라이팬이 없어요

오늘은 토스증권은 왜 자체 LLM을 개발하기로 결정했는지, 그리고 왜 고성능 자체 GPU 클러스터가 필요했는지 설명드릴게요. 다음 포스트에서는 더 자세한 도입기를 다뤄보겠습니다.

2024년 7월 19일 · 김진용



mdn web docs
MDN Web Docs

Web과 함께 시작하기 > JavaScript 기본

- 이 페이지는 영어로부터 커뮤니티에 의하여 번역되었습니다. MDN Web Docs에서 한국 커뮤니티에 가입하여 자세히 알아보세요.

JavaScript 기본

이전

Overview: Web과 함께 시작하기

다음

JavaScript는 여러분의 웹사이트에 상호작용성(예를 들면, 게임, 버튼이 눌러거나 폼에 자료가 입력될 때 반응, 동적인 스타일링과 애니메이션)을 더해 주는 프로그래밍 언어입니다. 이 글은 여러분이 이 흥미로운 언어를 시작하는 것을 도와드리고 가능한 것에 대한 아이디어를 제공할 것입니다.

mdn web docs

Getting started with the web > JavaScript basics

JavaScript basics

Previous

Overview: Getting started with the web

Next

JavaScript is a programming language that adds interactivity to your website. This happens in games, in the behavior of responses when buttons are pressed or with data entry on forms; with dynamic styling; with animation, etc. This article helps you get started with JavaScript and furthers your understanding of what is possible.

What is JavaScript?



스스로의 기준이 점점 높아진다

- 이후 어디까지든 갈 수 있다
- 스택오버플로우 답변
- ECMAScript 언어 명세, WHATWG의 HTML 표준
- V8이나 JavascriptCore 등의 엔진 개발자들의 글
- 표준 위원회 회의록, 구루들의 트윗, 메일링 리스트
- 옛날 자료, 오픈소스 이슈 게시판, PL 논문, 엔진 코드...
- 그 한계의 깊이가 당신을 보여줄 것



내용이 아니라 깊이를 보여 주는 것이다

- 진짜로 printf가 왜 printf인지가 중요한 건 아니다
- printf같은 기본적인 부분에서도 질문을 던질 수 있고
- 그걸 알아보기 위해 어디까지 갈 수 있는지를 보여 주는 것

**블로그 너머로
가는 길
- 기록의 길 -**

끝일 리가

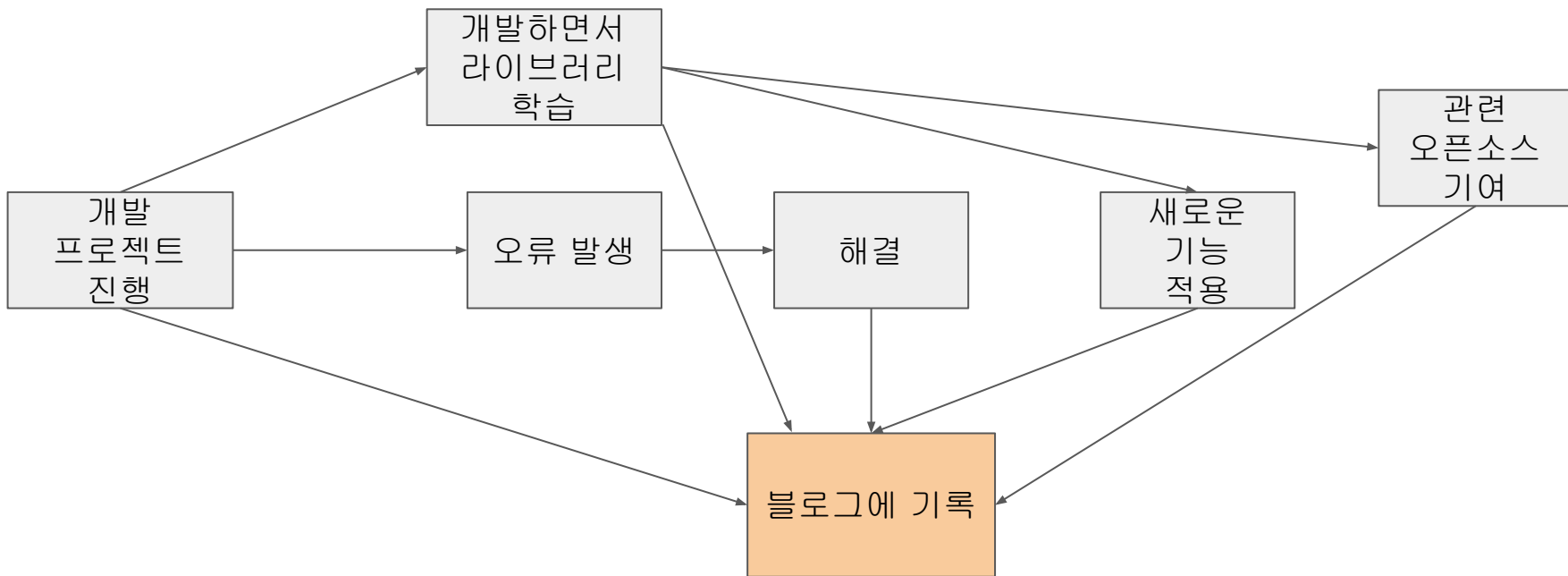


블로그만 쓸 수는 없다

- 테크니컬 라이터가 직업인 건 아니잖아
 - 개발 공부도 또 진짜 개발도 해야 하고
 - 오픈소스 기여도 해야 할 거 같고
 - 대외 활동도 해야 할 거 같고
 - 아무튼 글쓰기 말고도 개발자로서 해야 할 게 많다

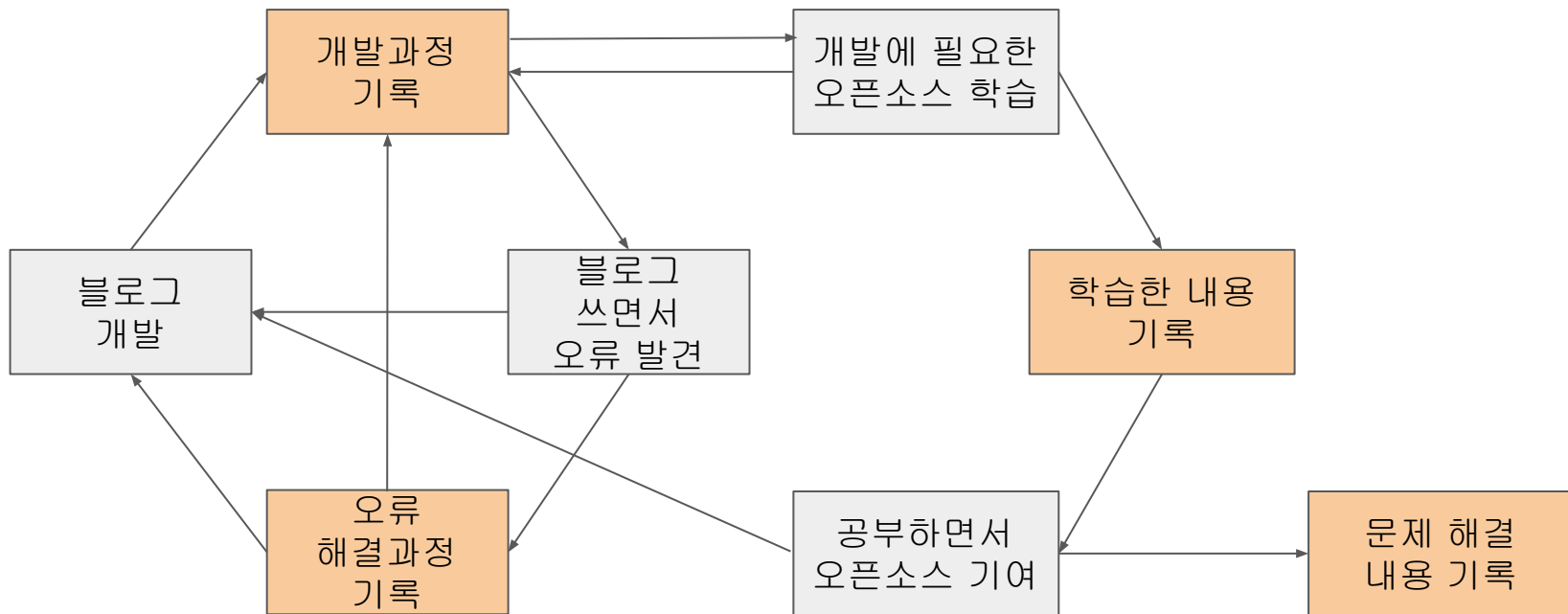
그럼 어떡하죠?

- 블로그와 다른 것들 간에 연결을 만들자
- 예시



사이클 만들기

- 만약 블로그를 직접 만든다면



블로그 만들기

- 어느 날 이력서에 넣을 마땅한 프로젝트가 없다는 걸 깨달았다
- 그래서 블로그를 만들기로 함
- NextJS도 배울 겸 해서...
- 근데 블로그 만들어도 넣을 글이 없으면 의욕이 꺾이니까 블로그를 만드는 과정을 쓰기로 함

블로그를 직접 개발하면서 기록

● 내가 블로그 만들면서 쓴 글

 **Witch-Work**


블로그 만들기 - 5. 디자인 이전에 남은 작업들
메인 페이지의 글 프리뷰, 헤더, About 페이지, 전체 글 목록 페이지 작업
[blog](#) [web](#)
2023. 05. 24

블로그 만들기 - 4. 이미지 경로 문제 해결
마크다운에서 이미지를 상대 경로로 불러올 수 있게 하자.
[blog](#) [web](#)
2023. 05. 24

블로그 만들기 - 3. 글 상세 페이지 구조
Contentlayer를 이용해서 블로그의 글 상세 페이지를 만들어 보자.
[blog](#) [web](#)
2023. 05. 22

블로그 만들기 - 2. 메인 페이지 HTML 구조
블로그의 HTML 구조를 만들어 보자.
[blog](#) [web](#)
2023. 05. 21

블로그 만들기 - 1. 기본 세팅
블로그를 새로 만드는 일의 시작
[blog](#) [web](#)
2023. 05. 19

 **Witch-Work**

블로그 만들기 - 10. 폰트, 프로젝트 소개, 태그
폰트 변경과 프로젝트 소개 카드 세부디자인
[blog](#) [web](#)
2023. 06. 01

블로그 만들기 - 9. 글 썸네일 만들기
canvas로 글의 썸네일 사진을 자동으로 생성하자.
[blog](#) [web](#)
2023. 05. 30

블로그 만들기 - 8. 글 목록/상세보기 페이지
글 목록 페이지와 글 상세보기 페이지 CSS를 손봐주자. 디자인, TOC, SEO 작업도
[blog](#) [web](#)
2023. 05. 30

블로그 만들기 - 7. 메인 페이지 컴포넌트 디자인
메인 페이지 CSS를 작성해 보자
[blog](#) [web](#)
2023. 05. 26

블로그 만들기 - 6. 기본적인 페이지 레이아웃
너무 못생긴 페이지, 가독성부터 높여보자.
[blog](#) [web](#)
2023. 05. 25

 **Witch-Work**

블로그 만들기 - 12. 페이지 테마, 댓글, 검색
댓글 기능과 다크 테마를 달고 검색 기능을 구현하자
[blog](#) [web](#)
2023. 06. 09

블로그 만들기 - 11. 글 조회수 달기
글의 조회수를 카운팅하기 위한 장렬한 전투 기록
[blog](#) [web](#)
2023. 06. 04

블로그 만들기 - 10. 폰트, 프로젝트 소개, 태그
폰트 변경과 프로젝트 소개 카드 세부디자인
[blog](#) [web](#)
2023. 06. 01

블로그를 개선하면서 또 기록

● 블로그를 최적화하는 과정



블로그 최적화 - 4. 검색 페이지 무한 스크롤, 아이콘 소스 최적화

겁나 빠른 마녀 : 블로그 최적화 그 네번째

[blog](#) [web](#)

2023. 06. 11

블로그 최적화 - 3. 이미지 최적화

겁나 빠른 마녀 : 블로그 최적화 그 세번째

[blog](#) [web](#)

2023. 06. 11

블로그 최적화 - 2. 글 목록 페이지 최적화

겁나 빠른 마녀 : 블로그 최적화 그 두번째

[blog](#) [web](#)

2023. 06. 11

블로그 최적화 - 1. 메인 페이지 최적화

겁나 빠른 마녀 : 블로그 최적화 그 첫번째

[blog](#) [web](#)

2023. 06. 10

● 서버 속도를 위해 직접 배포하고 기록



홈 서버로 블로그 배포하기 - 더 좋은 배포를 향하여

방화벽 설정, 자동 배포

[blog](#)

2023. 10. 03

홈 서버로 블로그 배포하기 - 블로그 올리기

블로그를 홈 서버에 올려보자

[blog](#)

2023. 10. 03

블로그 테마 변경 버튼 리뉴얼하기

블로그의 테마 변경 버튼을 다시 만들어보자

[blog](#)

2023. 09. 19

홈 서버로 블로그 배포하기 - proxmox, pfsense 초기세팅

홈 서버를 세팅해보자

[blog](#)

2023. 09. 18

고치는 거 있으면 또 기록

- 고치고 기록하고



Witch-Work



블로그 고치기 - Next.js 페이지에 사이트맵 추가하기

Next.js로 만들어진 페이지의 사이트맵을 생성하는 법

[blog](#) [web](#)

2024. 04. 07



블로그 고치기 - 블로그에 조회수 달기

없어졌던 블로그의 조회수 카운터를 다시 달아주자

[front](#) [blog](#)

2024. 02. 18

NextJS metadata 오류 해결

2023. 09. 13

[front](#) [blog](#)

목차

1. 시작
 2. 문제 해결 시도
 - 2.1. 클라이언트 컴포넌트 분리
 - 2.2. 라우트 변경해보기
 - 2.3. import의 문제
 3. 후속조치
- [참고](#)

1. 시작

지금 이 블로그는 NextJS 13의 app router로 마이그레이션 중이다. 이 과정은 추후 블로그에 올릴 예정이다. 여기서는 마이그레이션 과정에서 발생한 사소한, 그런데 시간을 꽤 잡아먹었던 문제를 하나 정리한다.

먼저 앱 라우터로 구성된 페이지 구조는 다음과 같이 짜여 있다.

블로그 다듬으면서 공부하고 또 기록하고

- 공부하고 기록하고
- 기여할 수 있으면 하고

정적 콘텐츠를 쉽게 다룰 수 있게 해주는 **velite**를 알아보자

2024. 04. 13

[blog](#) [front](#) [react](#)

목차

velite의 발견

1. 소개
2. 기본적인 사용
 - 2.1. collection 정의
 - 2.2. 설정 정의
 - 2.3. 설정 객체 타입 정의
 - 2.4. 컨텐츠 변환과 사용
3. transform을 이용한 속성 정의
 - 3.1. 변환된 데이터를 이용한 추가 속성 정의
 - 3.2. 메타데이터를 이용한 추가 속성 정의
 - 3.2.1. 커스텀 스키마 만들기
 - 3.2.2. 추가 속성 만들기
4. 컨텐츠 변환 후 추가 작업
 - 4.1. prepare
 - 4.2. complete
5. velite의 전반적인 평가
6. 비교
 - 6.1. contentlayer
 - 6.2. @next/mdx
 - 6.3. marked

[참고](#)

fix: class File to VeliteFile #117

 Merged zce merged 3 commits into [zce:main](#) from [witch-factory:file-to-velitefile](#)  on Apr 15

 Conversation 1  Commits 3  Checks 1  Files changed 5



witch-factory commented on Apr 14

[Contributor](#) ...

While reviewing the `VeliteFile` section at <https://velite.js.org/reference/types>, I noticed that the class is named `File`, which is already defined in Node.js. Additionally, the document title is `VeliteFile`, but the class name is `File`, which seems inconsistent. Therefore, I have renamed the `File` class to `VeliteFile` and updated the associated files accordingly. I have also made these changes in the documentation.

In documents like [/guide/custom-schema](#), the links that should point to the `VeliteFile` reference were not functioning correctly. This was due to the links pointing to `#VeliteFile` instead of `#velitefile`. I have corrected these links in the documentation.

 1

무한 자가발전 가능

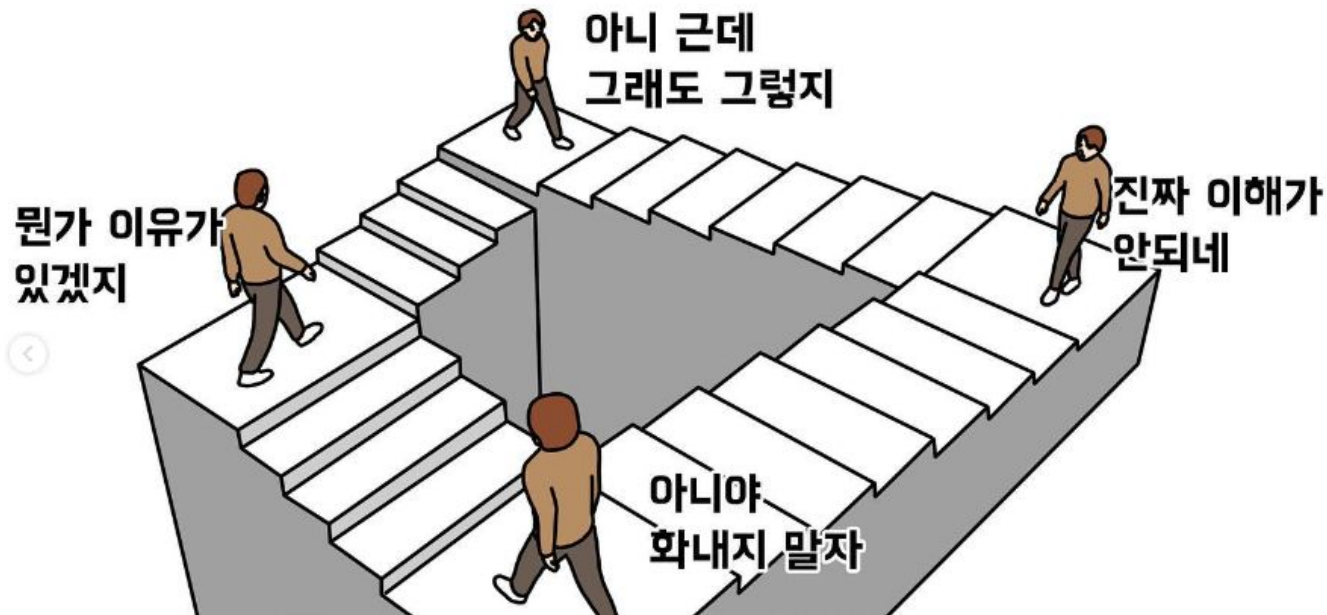
★무한동력 배터리★

무한동력 배터리 이다

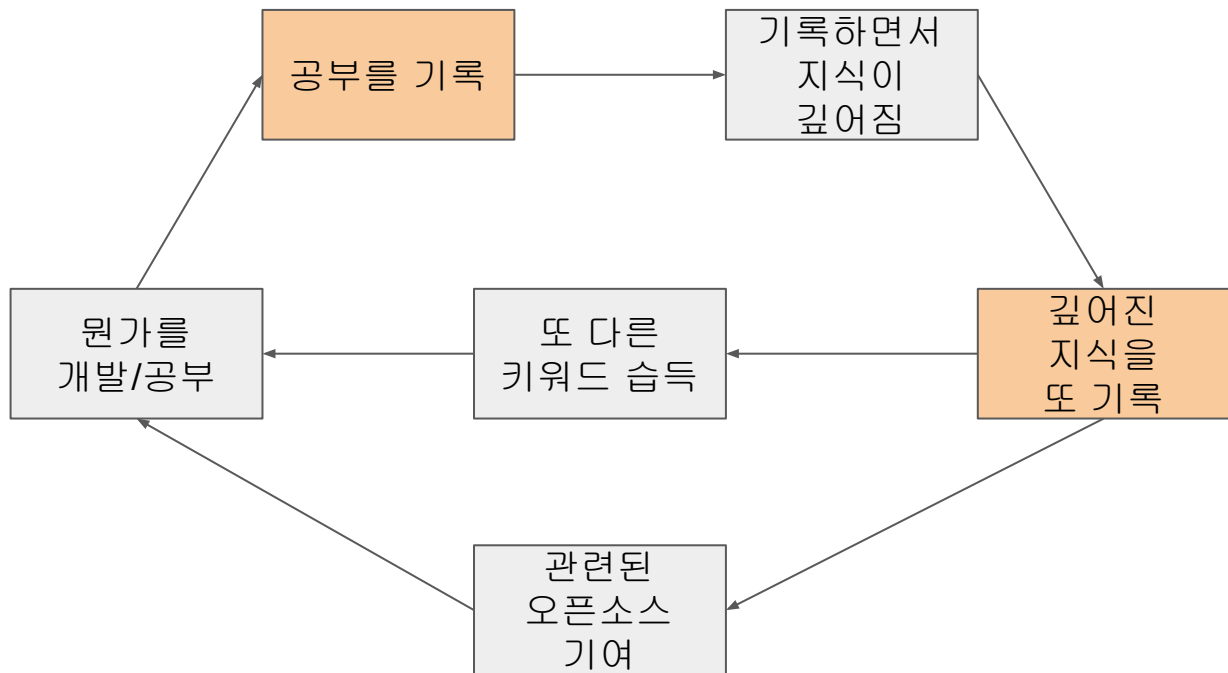


사이클 만들기 2

- 공부와 질문과 탐구의 무한 사이클



사이클 만들기 2



어느 날 글 하나를 보았다

- 임성묵 님의 “자바스크립트는 왜 프로토타입을 선택했을까”
- 프론트엔드에서는 꽤 유명한 글

자바스크립트는 왜 프로토타입을 선택했을까



임성묵 (Sungmook Lim) · Follow

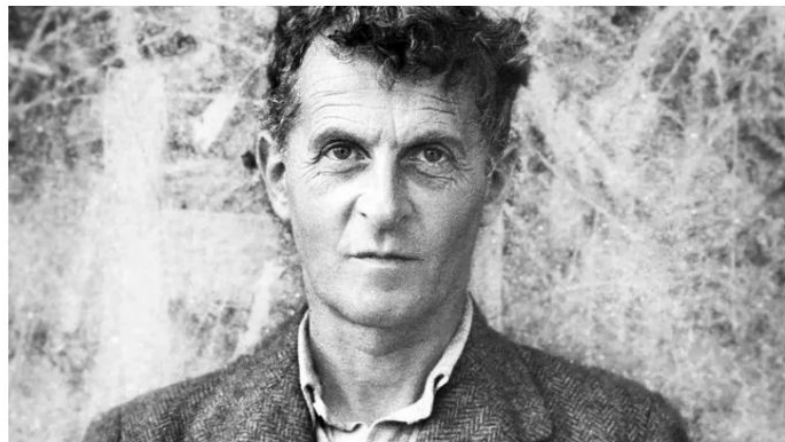
27 min read · Dec 9, 2021



2.5K



51



호이스팅

- JS에서 변수 선언이 스코프 최상위로 끌어올려지는 ‘호이스팅’
- 해당 글에서는 이를 철학적 맥락에서 설명
- 하지만 사실 var 호이스팅은 급하게 만들면서 생긴 실수라는 JS 제작자의 트윗을 봄



호이스팅은 급하게 만들면서 실수한 거라고?

- 하긴 JS의 이상한 부분들은 워낙 유명하다
- 그래도 JS를 급하게 만들었다니?
- 왜 그런 말이 나왔을까 궁금해짐
- 조사 시작

프로그래밍인사이드



자바스크립트는 왜 그 모양일까?

더글러스 크락포드가 알려주는

더글러스 크락포드 지음 | 박수현 옮김

위험한 자바스크립트를 안전하게 사용하는 법

인사이드



JavaScript: the first 20 years

- JS의 20년 역사를 다룬 문서 발견
- 바로 5달에 걸쳐 번역
- 그러면서 JS의 주석 관련해서 알게 된 사실을 글로 작성 - 네이버 FE News 선정

JS 탐구생활 - JS의 주석은 //과 /* */뿐만이 아니다

2024. 01. 06

javascript

썸네일



Javascript: The First 20 Years 번역

Javascript의 20년 이야기

이 페이지는 Allen Wirfs-Brock, Brandon Eich가 2020년 쓴 "[JavaScript: The First 20 Years](#)"의 번역입니다. 원본 저작권은 원 저자에게 있으며 번역 저작권은 번역자에게 있습니다.

목차

- 초록
- 목차
- 1. 소개

1부: Javascript의 기원

- 2. Javascript 이전 시대
- 3. Javascript 1.0과 1.1
- 4. Microsoft JScript
- 5. Mocha에서 SpiderMonkey까지
- 6. 막간: Javascript에 관한 평가

2부: 표준 만들기

- 7. 표준을 어디서 만들 것인가
- 8. 첫 TC39 회의
- 9. 명세를 향한 여정
- 10. 표준의 이름 정하기
- 11. ISO 패스트트랙
- 12. ECMAScript 3 정의하기
- 13. 막간: Javascript는 더 이상 Java가 필요하지 않다

JavaScript: the first 20 years

- 그렇게 알게 된 JS 역사 일부를 글로 작성
- 글 쓰는 개발자들 모임 “글또” 큐레이션 선정

4 회차

- [Python으로 모노레포 구성하기](#)
- [동영상 안끊기고 보는법 \(feat. LL-HLS\)](#)
- [쿠버네티스의 Eviction Threshold 조정하기](#)
- [리액트와 SCSS로 구현한 눈송이 애니메이션 코드: 사이트에 겨울 불어넣기](#)
- [LangCursor :: 한영키 오타를 방지하는 IntelliJ 플러그인 개발기](#)
- [토스페이스를 클론코딩 해보자](#)
- [JS의 소리를 찾아서 - Javascript의 초기 실수와 선택들](#)
- [표준 스트림을 이용한 프로세스 간 통신으로 재현성\(Reproducibility\) 지키기](#)
- [LocalStorage 대신 IndexedDB 사용하기](#)
- [Type-Driven Development](#)
- [AI가 실세계의 글자를 읽는 방법, STR\(Scene Text Recogniton\)](#)
- [Trino Gateway를 한번만 알아보자](#)
- [꼬리에 꼬리를 무는 시계열 개념 정리, 정상성부터 공적분까지](#)
- [Zero Copy 에 대하여](#)
- [새로운 기능 개발 전, 물음표 살인마 되기](#)
- [\[Python\] 데코레이터\(Decorator\)란 무엇일까?](#)
- [숫자 1은 올바른 JSON 형식인가?](#)
- [\[iOS\] Stretchable Image \(a.k.a iOS 9-Patch\)](#)
- [ChatGPT 개인화 및 Store 탐방하기](#)

JavaScript: the first 20 years

- 작은 세미나지만 발표도 함

2023 Winter	
축사	머신러닝 어린이
Algorithm SCUPC 출제 후기	최유빈
Knowledge Code Sandboxing	김기동
Algorithm LDH의 논물리에 - SEIVE와 S3-FIFO 캐시 자료구조 소개	ldh
Life 질문 잘하는 법	패트
Knowledge 블로그에서 검색 유입을 판별하는 원리	Yun
Knowledge JS는 왜 이 모양 이 꼴일까?	마녀

with문과 함께 춤을

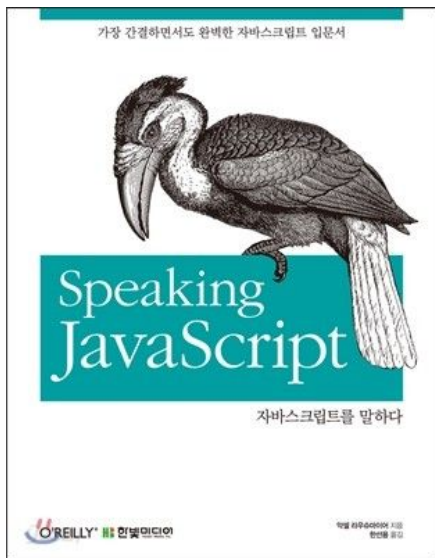
- 해당 역사 문서에서 with문이라는 게 나왔다
- ES5의 엄격 모드 이후에서는 금지된 문법
- 나는 문서 번역하면서 처음 들었다

```
let a, x, y;
const r = 10;

with (Math) {
  a = PI * r * r;
  x = r * cos(PI);
  y = r * sin(PI / 2);
}
```

with문과 함께 춤을

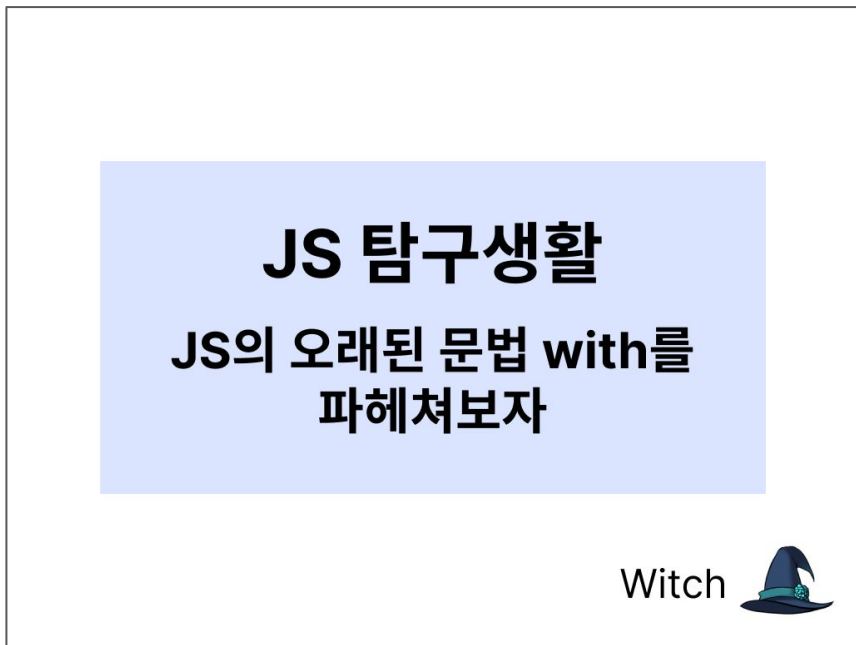
- 이후 다른 책을 보다가 다시 with문의 언급을 발견
- with문이 실제로 Firefox에서 버그를 일으켰다고 함
- 어? 이거 뭐지? 조사 시작



내가 본 책

with문과 함께 춤을

- 일단 with문이 뭔지 조사해서 글부터 씬



이건 블로그 썸네일

지나가다 MDN 번역 기여

- 그 과정에서 MDN 문서 2개 번역함

[ko] with 문서 신규 번역 #22055

 Merged lilsang merged 2 commits into `mdn:main` from `witch-factory:ko-with-statement`  3 weeks ago

 Conversation 17  Commits 2  Checks 7  Files changed 1



witch-factory commented last month

Description

deprecated된 명령문이라는 하지만 기존에 한국어로 존재하지 않던 문서 [with](#)을 신규 번역하였습니다.

Motivation

Additional details

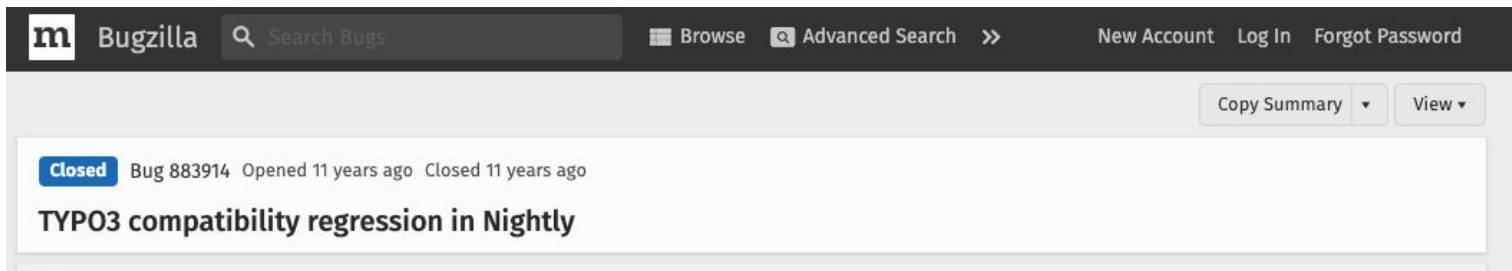
Related issues and pull requests



2

조사, 조사...

- 책에 나왔던 버그 리포트 원문 읽기



The screenshot shows the Bugzilla interface for a specific bug report. At the top, there is a navigation bar with the Bugzilla logo, a search box labeled "Search Bugs", and links for "Browse", "Advanced Search", "New Account", "Log In", and "Forgot Password". Below the navigation bar, there are two buttons: "Copy Summary" and "View". The main content area displays the bug's status as "Closed" in a blue box, followed by the bug ID "Bug 883914" and the dates "Opened 11 years ago" and "Closed 11 years ago". The title of the bug report is "TYPO3 compatibility regression in Nightly".

조사, 조사...

- 관련 es-discuss 토픽, JS 표준 위원회인 TC39 회의록 조사...

Array.prototype.values() compatibility hazard



Jason Orendorff (11 years ago)

Reply

Firefox added `Array.prototype.values()` and immediately ran into compatibility issues.

bugzilla.mozilla.org/show_bug.cgi?id=883914, bugzilla.mozilla.org/show_bug.cgi?id=881782

Both bug reports have to do with Sencha Ext JS. I haven't looked closely yet. It seems `with(values)` appears in Ext.

4.3 Array.prototype.values

(Allen Wirfs-Brock, Rick Waldron)

AWB: Ext.js uses a `with(...)` {}

```
function f(values) {  
  with(values) {  
    ...  
  }  
}
```

YK: Means that we can't add common names for common objects?

RW: ...Explained that Ext.js fixed the issues, but face a commercial customer update challenge. In the meantime, it continues to break several large scale sites.

AWB: Brendan's workaround (from post on the list)

```
values() -> @@values();  
keys() -> @@keys();  
entries() -> @@entries();
```

Importing from a module...

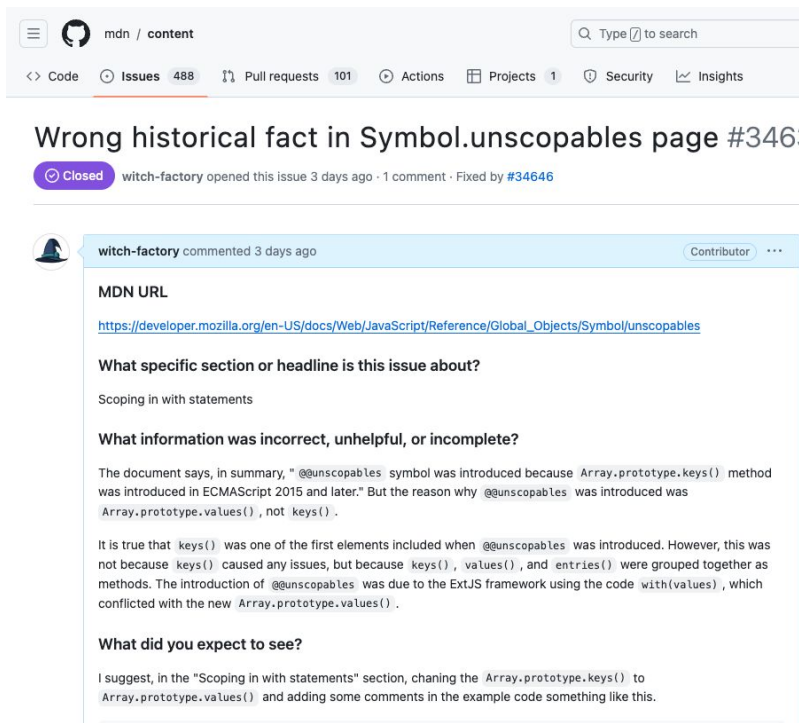
그렇게 조사해서 글++;

- with문이 Firefox에서 만들었던 문제와 해결책으로 나온 심볼 정리

JS 탐구생활
말썹쟁이 with문과
@@unscopables 연대기

조사 과정에서 MDN 원문의 틀린 점 찾기

- 이슈 올림(영어는 ChatGPT가 써줌)



The screenshot shows a GitHub repository page for 'mdn / content'. The issue title is 'Wrong historical fact in Symbol.unscopables page #3466'. The issue is closed and was opened by 'witch-factory' 3 days ago. A comment from 'witch-factory' is visible, providing a detailed explanation of the historical fact error in the MDN article.

mdn / content

Type to search

<> Code Issues 488 Pull requests 101 Actions Projects 1 Security Insights

Wrong historical fact in Symbol.unscopables page #3466

Closed witch-factory opened this issue 3 days ago · 1 comment · Fixed by #34646

witch-factory commented 3 days ago

Contributor

MDN URL

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol/unscopables

What specific section or headline is this issue about?

Scoping in with statements

What information was incorrect, unhelpful, or incomplete?

The document says, in summary, "`@unscopables` symbol was introduced because `Array.prototype.keys()` method was introduced in ECMAScript 2015 and later." But the reason why `@unscopables` was introduced was `Array.prototype.values()`, not `keys()`.

It is true that `keys()` was one of the first elements included when `@unscopables` was introduced. However, this was not because `keys()` caused any issues, but because `keys()`, `values()`, and `entries()` were grouped together as methods. The introduction of `@unscopables` was due to the ExtJS framework using the code `with(values)`, which conflicted with the new `Array.prototype.values()`.

What did you expect to see?

I suggest, in the "Scoping in with statements" section, changing the `Array.prototype.keys()` to `Array.prototype.values()` and adding some comments in the example code something like this.

조사 과정에서 MDN 원문의 틀린 점 찾음

- PR했더니 내가 조사한 거 보충해서 적을 생각 있냐고 하길래 바로 적음

fixed wrong historical fact in Symbol.unscopables page #

Merged Josh-Cena merged 2 commits into [mdn:main](#) from [witch-factory:symbol-unscopables](#) 9 hours ago

Conversation 12 Commits 2 Checks 8 Files changed 2

witch-factory commented 3 days ago Contributor

Description

fixed #34639 .

Motivation

Additional details

Related issues and pull requests

witch-factory requested a review from **a team** as a code owner 3 days ago

witch-factory requested review from **Josh-Cena** and removed request for **a team** 3 days ago

github-actions bot added **Content-JS** **size/js** labels 3 days ago

Josh-Cena commented 3 days ago Member

Do you want to add more content as you mentioned in the issue? That could be interesting.

```
1 00 -31,16 +31,20 00 This protocol is also utilized by DOM APIs, such as |Element.prototype.append|
2
3 31
4 32
5 33
6 34 - The following code works fine in ES5 and below. However, in ECMAScript 2015 and later, the
7   {{{jsref["Array.prototype.keys()"]}}} method was introduced. That means that inside a "with"
8   environment, "keys" would now be the method and not the variable. That's why the "@unscopables"
9   symbol was introduced. A built-in "@unscopables" setting is implemented as {{{jsref["Array@unscopables"]}}} to
10  prevent some of the Array methods being scoped into the "with" statement.
11
12 35
13 36
14 37 - var keys = [];
15
16 38
17 39 - with (Array.prototype) {
18 40   - keys.push("something");
19
20 41
21 42
22 43
23 44
24 45
25
26 46 * The code containing "with (values)" caused some websites to malfunction in Firefox when "Array.prototype.values()"
27   was added (Firefox Bug 883914|https://bugzilla.mozilla.org/show_bug.cgi?id=883914). Furthermore, this implies that any future array method
28   addition may be breaking if it implicitly changes the "with" scope. Therefore, the "@unscopables" symbol was
29   introduced and implemented on "Array" as {{{jsref["Array@unscopables"]}}} "Array.prototype[@unscopables()"] to
30   prevent some of the Array methods being scoped into the "with" statement.
31
32 47 *
33 48
34 49
35 50 You can also set '@unscopables' for your own objects.
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

그래도 2문단은 기여했다...

이 과정을 또 글로 씀

- with문을 탐구한 과정을 기록으로 남김

with와 함께, with 시리즈 탐구기

2024. 07. 08

javascript

web

목차

[이 글은 작성 중입니다.](#)

[시작](#)

[MDN 문서 번역](#)

[MDN 원문에 기여](#)

[트위터](#)

그러다 보면 누가 또 자료를 줌

- 트위터에서 지나가던 사람이 with에 관한 논문을 줬음
- 이러면 또 글의 깊이를 더 늘릴 수 있다(이제 해야 함)

The image shows a screenshot of a Twitter thread. The top tweet is from user 'flow' (@hrmm_flow) dated July 5th, with a link to a paper on dl.acm.org. A reply from user '김성현' (@witch_front) dated July 5th explains that 'with' is a disallowed syntax in JavaScript and that they have organized related papers on witch.work. The bottom tweet is another from 'flow' (@hrmm_flow) dated July 5th, with a link to a paper on person.dibris.unige.it. The thread shows engagement metrics like replies, retweets, and likes.

flow @hrmm_flow · 7월 5일
dl.acm.org/doi/abs/10.114...

김성현 @witch_front · 7월 5일
JS에서 쓰면 안 되는 문법 중 with가 있습니다. 지금은 거의 잊혀져서 걱정할 필요조차 없지만 우연히 이에 대한 자료들을 접해 잊혀진 with문을 정리하였습니다.
witch.work/posts/javascr...
다음 글로는 with가 일으켰던 문제를 하나 정리할 예정입니다.

2 1 4 440

flow @hrmm_flow · 7월 5일
person.dibris.unige.it/zucca-elena/FO...

1 90

바로 이렇게 하기는 쉽지 않다

- 근데 튜토리얼 베껴 쓰기에서 시작해도
조금씩 더 가다 보면 멀리 감
- 나도 처음부터 논문까지 볼 생각 없었음
- 그러니까 일단 떠오르는 거 시작해라

모던 자바스크립트 튜토리얼 part 1.3 코드 품질 - 2

ko.javascript.info part 1-3 두번째

javascript

2022. 12. 31

모던 자바스크립트 튜토리얼 part 1.3 코드 품질 - 1

ko.javascript.info part 1-3 첫번째

javascript

2022. 12. 27

모던 자바스크립트 튜토리얼 part 1.2 자바스크립트 기본 - 3

ko.javascript.info part 1-2 세번째

javascript

2022. 12. 26

모던 자바스크립트 튜토리얼 part 1.2 자바스크립트 기본 - 2

ko.javascript.info part 1-2 두번째

javascript

2022. 12. 17

나도 1년 반쯤 전에는
튜토리얼을 베끼고 있었다

바로 이렇게 하기는 쉽지 않다

- 근데 튜토리얼 베껴 쓰기에서 시작해도
조금씩 더 가다 보면 멀리 감
- 나도 처음부터 논문까지 볼 생각 없었음
- 그러니까 일단 떠오르는 거 시작해라
- 그러다 보면 프로젝트도 생기고 발표도
하고 오픈소스도 기여하고 공부도 하고...

모던 자바스크립트 튜토리얼 part 1.3 코드 품질 - 2

ko.javascript.info part 1-3 두번째

javascript

2022. 12. 31

모던 자바스크립트 튜토리얼 part 1.3 코드 품질 - 1

ko.javascript.info part 1-3 첫번째

javascript

2022. 12. 27

모던 자바스크립트 튜토리얼 part 1.2 자바스크립트 기본 - 3

ko.javascript.info part 1-2 세번째

javascript

2022. 12. 26

모던 자바스크립트 튜토리얼 part 1.2 자바스크립트 기본 - 2

ko.javascript.info part 1-2 두번째

javascript

2022. 12. 17

나도 1년 반쯤 전에는
튜토리얼을 베끼고 있었다

아무렇게나 시작해서 끝까지 가라

- 이것저것 했지만 결국 스스로 속이 시원해질 때까지 딥다이브하는 것
- 시작은 별거 없다
 - 혼자 지식을 비교해 가며 질문을 만들 수도 있고
 - 인터넷 글들 보다가 재밌는 게 있어서 더 조사할 수도 있고
 - 주니어 개발자 면접질문만 가도 키워드는 넘친다
- 그러면서 나오는 것들을 하나씩 쳐내면 됨

아무렇게나 시작해서 끝까지 가라

- 뭐라도 시작해서 남들보다 한 걸음씩만 더 가기



“네가 예전에 믿었던 것보다 한 걸음 더 나아가는 것은 언제나 너의 힘 안에 있다.”

끝까지 가다 보면 뭐라도 되지 않겠습니까

- 솔직히 뭐가 될지는 나도 모르겠다
- 그래도 “진짜”같아 보이지 않았는가?



당신은 어디로 가고 싶은가?

- 나는 깊이 파고들어가고, 이걸 글로 쓰고 나름의 박수를 받는 게 즐거웠다
- 나는 이렇게 왔고, 앞으로도 그럴 것이다
- 같이 갈까요?

