Home

2022-07-22

> 원문: How Microsoft Wants To Change JavaScript and TypeScript

Microsoft는 JavaScript와 TypeScript를 어떻게 바꾸고 싶어하는가

> 이 제안은 JavaScript와 TypeScript를 바꿀 수 있습니다.

지난 2022년 3월 9일 수요일, Microsoft는 앞으로의 JavaScript와 TypeScript에 개발을 지원하는데 있어 놀랄만한 제안을 발표했습니다.

지금까지 이 내용은 Stage 0 제안이라 불리는 제안에 그쳤지만 Microsoft는 이미 이 제안을 TC39에 제출했음을 알렸습니다. 만약 이 제안이 받아들여지고 구현될 경우, JavaScript에 지금까지 일어난 것들 중 가장 큰 대격변이 일어날 것입니다.

JavaScript의 역사

20년 전을 돌이켜 지금의 웹 개발과 비교해보면 JavaScript가 프로그래밍 언어로서 상당히 발전했지만, JavaScript 주변의 생태계가 훨씬 더 큰 진전이 있었다는 것을 분명하게 알 수 있습니다.

한편으로는 JavaScript 커뮤니티가 지난 20년 동안 훨씬 더 전문적이 되었고, 다른 한편으로는 인터넷의 근본적인 문제가 점점 더 대두되고 있기 때문에 이 두 가지 요점은 상호간에 요구되는 점입니다. 개발자는 사용자가 어떤 브라우저를 사용할 지를 제어할 수 없습니다.

즉, 사용자가 브라우저를 정기적으로 업데이트할 경우에만 JavaScript의 최신 기능을 사용할 수 있습니다. 개인 사용자의 경우에는 요즘 많은 브라우저가 스스로 업데이트를 하거나 묻지 않고 업데이트를 표시하기 때문입니다. 그러나 기업의 경우에는 그렇지 않습니다.

기업에서는 소프트웨어 및 소프트웨어 업데이트에 대한 엄격한 규정이 있습니다. 많은 회사가 업데이트되지 않은 소프트웨어나 브라우저로 일합니다. 이것은 HTML과 CSS에도 영향을 미치는 근본적인 문제이며, 각 브라우저에 의해 해석되어야 하는 프로그래밍 언어에도 영향을 미칩니다.

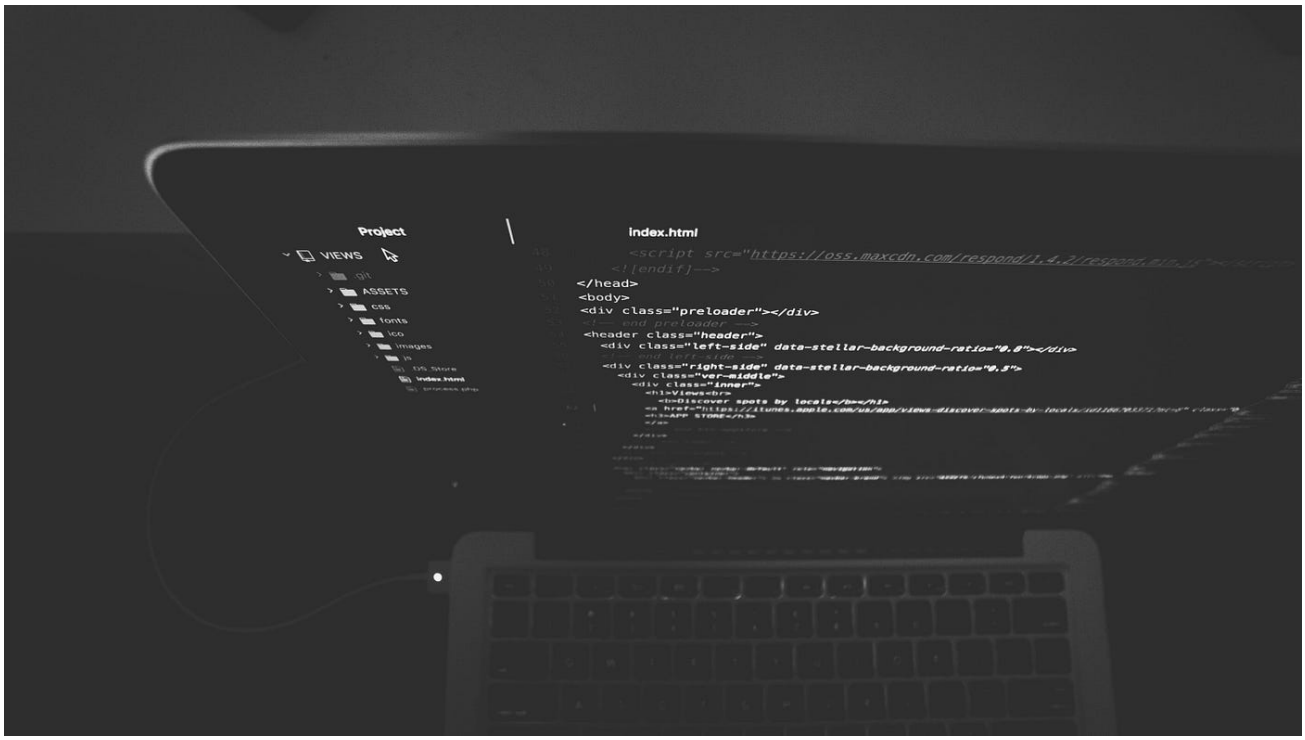
트랜스파일러와 번들러

웹 개발자로서, 여러분은 프로그래밍을 단순하게 하는 모던 JavaScript, CSS, 또는 사용성 등에서 더 나은 결과를 이끌어내는 HTML 기능에 의존하는 것 또는 구식 브라우저 때문에 모든 사람이 이용할 수 없기 때문에 이러한 현대적 기능들을 사용하지 않는 두 가지 이데올로기 사이에서 결정을 내려야 하는 문제에 직면했습니다. 업그레이드하면 특정 사용자들에게 버그가 발생할 수 있습니다.

또한 그것과는 별개로 수십 년 동안 JavaScript를 위한 적절한 모듈 시스템이 없었습니다. Node.js는 CommonJS로 표준 세웠지만, 이는 단지 서버에 국한되었습니다.

브라우저에는 오랫동안 아무 일도 일어나지 않았고, 그래서 번들러가 트랜스파일러와 함께 온 것입니다. 그리고 JIT(Just-in-time) 컴파일된 프로그래밍 언어로 정상적으로 실행될 수 있지만, 항상 복잡한 빌드 프로세스를 처리해야 했습니다. 소스 코드를 실제 코드로 변환한 다음 브라우저에서 실행되고 해석됩니다.

이것이 약 10년 전의 상황입니다.



TypeScript의 등장

정확히 10년 전 이 시점에서 Microsoft는 TypeScript를 발명했습니다. Microsoft는 배포 전에 JavaScript 코드를 변환하기 위해 트랜스파일러가 필요하다면 이 빌드 프로세스의 추가 단계는 큰 문제가 되지 않을 것이라고 생각했습니다.

그 대신 모던 JavaScript를 바닐라 JavaScript로 변환하는 좋은 트랜스파일러를 얻게 되었습니다. 또한 TypeScript는 정적 타입 시스템으로 JavaScript를 훨씬 더 확장 가능하게 하고, 새로운 개념을 전달했으며, 팀 내에서 JavaScript를 효율적으로 개발할 수 있도록 하는 데 상당한 기여를 했습니다.

TypeScript가 매우 빠르게 자리를 잡았고 오늘날 산업에서 JavaScript 개발의 표준이 된 것은 놀라운 일이 아닙니다.

지난 10년 동안, 세상 또한 변화했습니다. 최신 버전이 아니며 자체적으로 업데이트되지 않는 브라우저는 여전히 남아있지만, 오늘날에는 예전보다 훨씬 더 작은 역할을 수행합니다.

만약 Evergreen Browser들만 고려하고 ESM(ECMAScript Module)을 사용하면 트랜스파일러 없이도 꽤 작업이 가능하다는 것을 의미합니다. 이렇게 함으로써 서버 측과 클라이언트 측 모두에서 작동하는 네이티브 JavaScript 기반 모듈 시스템도 사용할 수 있습니다.

> (역주) Evergreen Browser: 오래된 브라우저가 그랬던 것처럼 제조업체로부터 새로운 버전을 배포함으로써 업데이트되는 것이 아니라 미래 버전으로 자동 업그레이드되는 브라우저를 가리킨다. MS 인터넷 익스플로러 초기처럼 전반적으로 기술이 발전하고 다양한 새로운 제품들이 MS의 독주를 위협하면서 지난 몇 년간 브라우저의 설계와 그 전달이 얼마나 빠르게 변화했는지를 반영한 용어이다.

이는 적어도 기술적 관점에서 볼 때 번들러가 더 이상 필요하지 않음을 의미합니다. 번들러는 단지 HTTP 요청을 최적화하기 위한 하나의 단계에 불과하며 서버에서 로드해야 하는 작은 개별 파일의 수는 줄이고

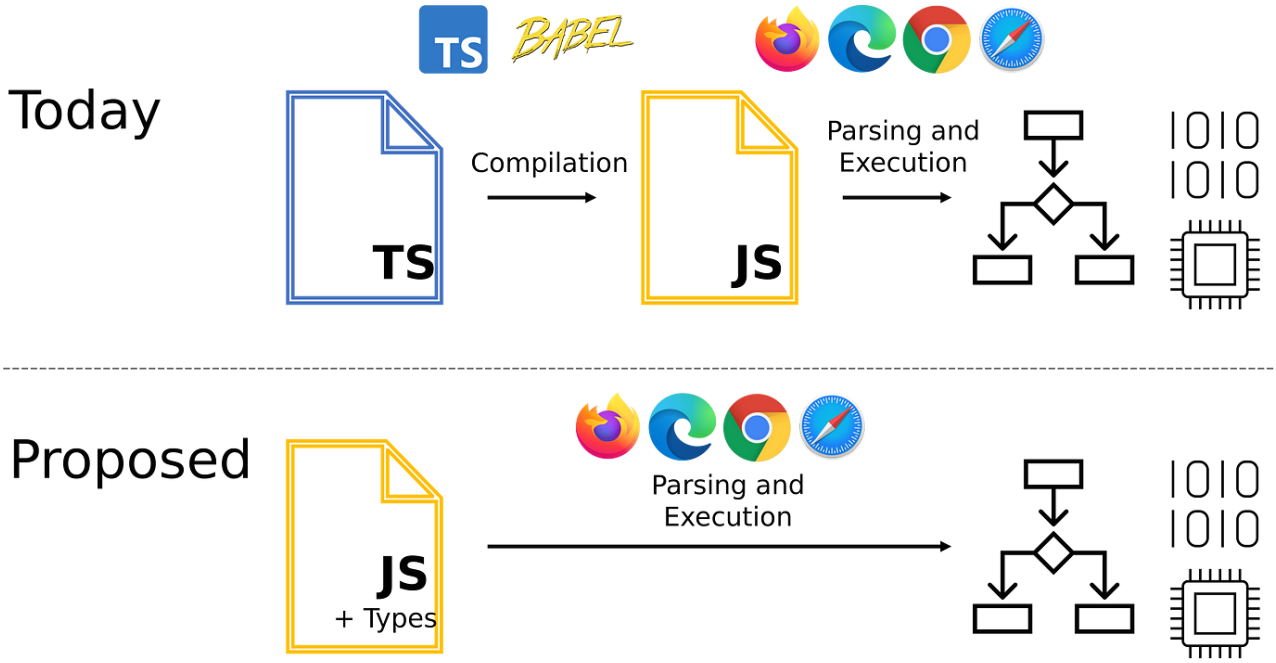
큰 파일 몇 개만 로드할 수 있도록 합니다. 이는 궁극적으로 훨씬 더 효율적으로 간단하게 수행될 수 있습니다.

TypeScript, 장애물인가?

이는 빌드 프로세스가 점차 단순해지거나 상관없어지는 것을 의미합니다. Microsoft는 이제 웹 브라우저와 기타 JavaScript 런타임 환경에서는 TypeScript를 이해하지 못하는 이유로 TypeScript 컴파일러만 남아 있을 것으로 예상하고 있습니다.

즉, Microsoft의 TypeScript가 갑작스레 매우 실용적인 도구에서 다소 성가신 도구로 바뀌었다는 것을 의미합니다. Microsoft는 TypeScript가 장애물이 되는 대신 개발자들에게 영감을 주길 원합니다.

그들이 우려했던 결과는, 머지 않아 JavaScript 개발이 20년 전처럼 빠르고, 직접적이며 효과적이게 되어서 TypeScript를 사용하지 않아 트랜스파일러 등이 더 이상 필요하지 않게 되는 것입니다.



브라우저를 위한 TypeScript

지금 확실한 방법은 웹 브라우저 및 기타 런타임 환경에서 JavaScript에 대한 대체 프로그래밍 언어로서 TypeScript의 통합을 요구하는 것입니다. 이론적으로, 그렇게 억지스럽지는 않습니다. Deno는 이미 확실한 방법으로 시도하고 있습니다.

Node.js에 사용할 수 있는 ts-node라는 npm 패키지가 있으며, 이는 Deno와 유사한 접근 방식을 취합니다. 응용 프로그램을 빌드/로드할 때 메모리에서 컴파일하면 TypeScript가 즉시 실행되는 것처럼 느껴지지만 실제로는 그렇지 않습니다.

또한 TypeScript는 이제 훨씬 더 복잡한 프로그래밍 언어가 되었으며 Microsoft가 TypeScript 컴파일러의 모든 기능을 공통 웹 브라우저에 직접 통합하는 것은 바람직하지 않습니다. 그것은 매우 복잡한 과제이며 새로운 방대한 표준을 통합하기 위해 Apple, Google, Mozilla 등의 협력이 필요합니다.

Microsoft는 이 모든 노력을 회피하고 있습니다. 옳든 그르든, 그것은 아직 미해결 상태이지만, 이 상태는 단순하게도 그들이 원하는 것이 아닙니다.

(이전에) 중도로써의 JSDoc

대신, 이제 다른 접근 방식이 검토되었으며, 이는 마침내 지난 주(2022년 3월 첫째주) Microsoft가 제안한 것으로 이어졌습니다. TypeScript에 익숙한 모든 사람들은 (a) 바닐라 JavaScript를 작성하는 접근 방식과 (b) TypeScript로 완전히 전환하는 접근 방식 사이에 중간이 있다는 것을 알고 있습니다.

TypeScript를 사용하면 JavaScript의 코드를 분석할 수 있으며 적절한 JSDoc 주석을 작성하기만 하면 TypeScript를 작성할 필요 없이 타입을 저장할 수도 있습니다. 일부 회사의 경우 프로젝트를 TypeScript로 완전히 마이그레이션할 필요 없이 TypeScript 컴파일러에서 타입 지원을 받을 수 있는 방법입니다.

```
/**
 * @param {string} p1 - 문자열 매개변수
 * @param {string} p2 - 선택적인 매개변수 (클로저 문법)
 * @param {string} [p3] - 또 다른 선택적인 매개변수 (JSDoc 문법)
 * @param {string} [p4 = "test"] - 기본값에 대한 선택적인 매개변수
 * @return {string} 결과값
 */
function stringsStringStrings(p1, p2, p3, p4 = "test") {
  // TODO
}
```

이 코드의 큰 장점은 여전히 고전적인 JavaScript이기 때문에 컴파일할 필요가 없고 JSDoc 코멘트를 모두 삭제하기만 하면 언제든지 TypeScript에서 벗어날 수 있다는 것입니다. 저도 JSDoc을 써본적이 있지만, TypeScript의 저렴한 대안처럼 느껴집니다.

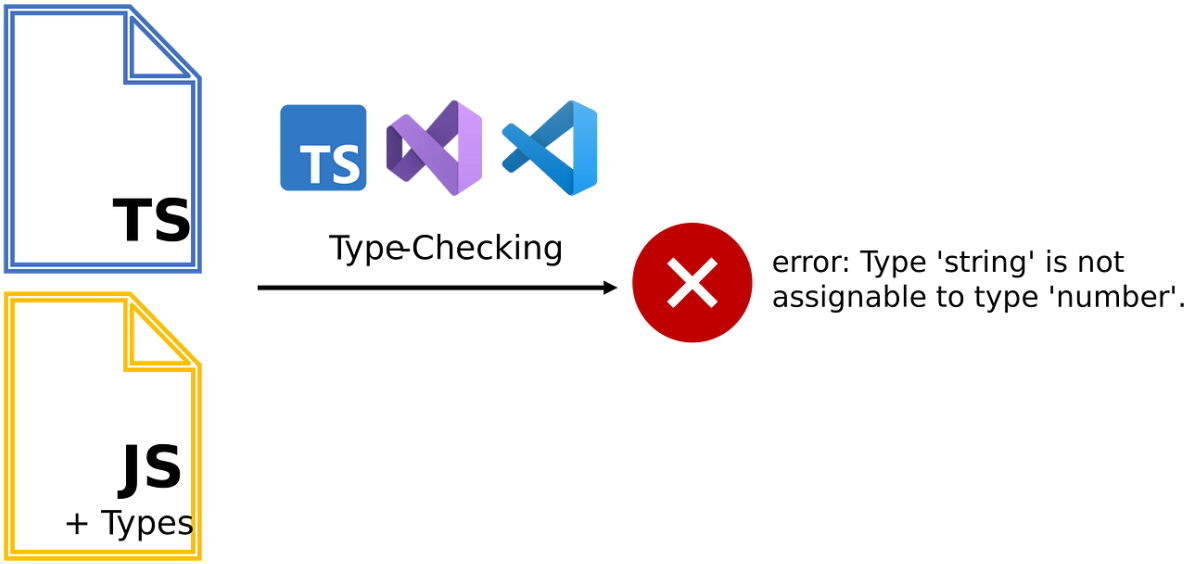
코드를 더 많이 써야 하고, 모든 것이 번거롭고, 복잡한 타이핑은 거의 불가능해지지만, 그래도 전혀 타입이 없는 것보다는 낫습니다.

하나의 제안: 주석으로서 타입

Microsoft가 지지하는 이 제안은 TypeScript 어노테이션을 주석으로 간주하는 것입니다. 그러면 TypeScript를 더 이상 컴파일할 필요가 없습니다.

형식 어노테이션 및 키워드(예: "public" 또는 "private")는 JavaScript 실행 중에 무시되는 주석으로 간주됩니다. 즉, TypeScript를 JavaScript 코드로 컴파일할 필요 없이 TypeScript를 작성할 수 있습니다.

여전히 TypeScript 컴파일러를 사용하여 타입 검사를 수행할 수는 있지만 코드를 실행하기 위해 컴파일할 필요는 없습니다. 타입이 JavaScript 코드에서 사용 가능하기 때문에 소위 "d.ts" 라고 불리는 파일도 더 이상 필요하지 않습니다.



따라서 TypeScript 컴파일러는 ESLint와 같은 린터와 같은 선택적 추가 기능이 되며 모든 타입의 어노테이션들은 실행된 코드에서 보이지 않습니다.

그리하여 이 제안은 "주석으로서의 타입"이라고 불립니다.

주석으로서 타입의 단점

제가 이 글을 처음 읽었을 때, **저는 이 아이디어가 마음에 들었습니다.** 그것이 제가 원하던 상황이었습니다. 하지만 조금 더 생각해보고 제안서를 반복해서 읽었으며 제가 회의적이라는 것을 인정해야 할 것 같습니다. 이론상으로는 훌륭하게 보이지만, 몇 가지 단점이 있습니다.

TypeScript는 원시적인 타입의 함수 매개변수로만 구성되는 것이 아니라 인터페이스, 유니온 타입, 타입 키워드, 매우 복잡하고 중첩된 타입, "as" 키워드, pulbic/private/protected 키워드, generic 타입 등에 대해 이야기하고 있다는 것을 잊어서는 안 됩니다.

그리고 이 모든 것이 미래의 자바스크립트에서는 주석으로 간주되어야 하는 것입니다. 이미 작성하고 있는 주석도 유지하면서 말이죠. 저는 이것이 불가능하다고 말하고 싶은 것은 아닙니다. 다만, 두 가지 유형의 주석(한 줄짜리 주석과 블록 주석)을 갖는 대신, 키워드를 한 번에 표현할 수 있는 여러 가지 새로운 타입이 필요하게 됩니다.

그래서 저는 이것이 좋은 방법인지 자문해야 했습니다. 왜냐하면, 저는 현재의 해결책이 좋기 때문입니다. 비록 완벽하지 않고, 먼저 컴파일되어야 한다고 해도 말이죠.

게다가, 우리는 값과 타입이 섞여 있기 때문에 **enum 없이 코딩**해야 할 것 같습니다. 여기에는 네임스페이스, TypeScript의 JSX 지원, 매개 변수 속성 등도 포함됩니다. 따라서 enum을 사용하려면 컴파일해야 하지만 enum을 사용하지 않으면 컴파일하지 않아도 됩니다.

저는 이 제안으로 JavaScript와 TypeScript의 실제 구분이 보이지 않습니다. 그리고 이것이 미래가 되는 것이 맞는지 의심스럽습니다.

제 생각에, TypeScript 개발자들을 혼란과 새로운 두가지 계층으로 이끌 것입니다.

마치며 든 생각

제 생각에 TypeScript는 컴파일해야 하더라도 완전히 자체 기능을 유지하거나 TypeScript와 같이 JavaScript는 선택적 정적 타입 시스템을 사용해야 합니다.

이것은 TypeScript가 새로운 JavaScript가 되어야 한다는 것을 의미하지만, Microsoft는 이미 그것을 배제했습니다. 따라서 TypeScript는 이러한 방식으로 근본적인 변경을 해서는 안 되며, 특히 Microsoft가 현재 제안하고 지원하는 방식으로는 변경되어서는 안 된다고 생각합니다.

어쨌든, 저는 이것이 앞으로 몇 달과 몇 년 동안 어떻게 발전할지 궁금합니다.

감사합니다!

참고 자료

- A Proposal For Type Syntax in JavaScript (Microsoft DevBlog)
- ECMAScript proposal: Types as Comments (Github Repository)