Ver.2021 노베이스 모던 웹개발

작성자: 경상국립대학교 컴퓨터과학과(cs) 실습조교 이자룡

목차

* 개발자(software developer) 는 프로그램을 만드는 사람, 곧 프로그래머를 말한다. 개발자는 개발(programming)을 하고, 프로그램을 만든다.

0. Orientation

ver.2021 노베이스 모던 웹개발 코스의 첫번째 시간에 온 걸 환영한다. 이 코스에선, 다음 기술로 웹앱(브라우저로 작동하는 프로그램)을 만들어볼 것이다.

1. TypeScript
2. GraphQL
3. React.js Hook
4. React Router
5. Apollo
6. MongoDB
7. Tailwind CSS

언급한 기술들은 2021년 기준으로 모던 웹, 즉 최신 웹 기술들이다. 먼저 다음 질문에 답해보겠다.

Q. 진짜 노베이스 수업 맞나?
A. 저 리스트를 처음 보는 순간 이게 대체 뭔가 싶을 것이다. 잘못 찾아왔다는 생각에 벌써 도망갈 생각 하고 있는지도 모른다. 그러나, 이 코스는 정말로, 프로그래밍의 기초인 변수, 자료형, 배열, 함수부터 차근차근히 배워가는 노베이스 코스가 맞다.

Q. 왜 하필 저 기술들인가?
A. 글을 쓰는 현재 시점에서 가장 쉽게, 눈에 보이는 프로그램을 만들 수 있는 기술이기 때문이다. 컴퓨터공학을 전공한 학생들조차도, 졸업할때까지 자신의 힘으로 프로그램 하나 만들 줄 모르는 학생들이 생각보다 많다는 건 안타까운 현실이다. 전공자든 비전공자든 일단 뭔가 만들어봐야하지 않겠는가?

Q. 난이도는 둘째치고 배워야 할 과목이 무려 7개나 되지 않나?
A. 7개 각각이 하나의 과목이라 생각하면 당연히 막막하다. 저 7개가 각각 의미하는 것은 '과목'이 아니라 '기술'이다. 해당 기술을 깊게 들어가면 한도 끝도 없이 어렵고 그것 하나만으로도 책이 되지만, 우린 개발할 때 쓰는 것들만 간단히 배워볼 것이다.

Q. 웹 시장이 너무나 빨리 변하기 때문에, 1년후에 구식이 될지도 모르는 기술들이다. 배우는 게 의미가 있을까?
A. 그런 생각이라면 아무것도 만들 수 없다. 신기술은 끊임없이 나오기때문에, 우린 새로운 기술을 끊임없이 기다리는게 아니라, 이미 나와있는 기술들을 사용해봐야된다. 당연히 저 기술들은 언젠가 구식이 될 것이다. 우린 그런 거 따질 게 아니라, 어느 정도 배웠을 때 잠시 멈춰 스스로 자신만의 프로그램을 만들어봐야한다.

이 수업을 듣기 위해 필요한 지식은 아무것도 없다. 컴퓨터를 켜서 구글 크롬에 접속할줄만 알면 바로 시작해도 된다. 만약, 수업을 듣다가 모르는 단어나 기술이 나온다면, 설명을 빼먹은 게 아니라 지금은 설명할 단계가 아니라는 필자의 판단 때문에 설명을 하지 않은 것이다. 그냥 책 읽는다 생각하고 쭉쭉 앞으로 나가자. 다시 말하지만 이 수업의 대상은 '노베이스', 컴퓨터로 할 줄 아는 건 게임밖에 없는 학생들을 위한 것이다.

지금부터, 모던 웹의 기본이라고 할 수 있는 TypeScript 수업을 시작한다.

cf. 수업에서, 필자는 학생들이 영어로 꼭 알아뒀으면 하는 단어는 영어로 썼다. 프로그래머는 좀 힘들더라도, 영어에 익숙해져야한다. 우리가 마주칠 기술들의 공식문서는 영어로 쓰여져 있고, 버그(= 에러)를 잡기 위해 구글링을 할 때도 영어로 찾는 게 훨씬 좋다. 일부러 영어로 쓴 단어는 개발을 위해 꼭 알아뒀으면 하는 단어이기에, 꼭 알아두도록 하자!

먼저, TypeScript는 기본적으론 JavaScript (JS)이기 때문에 JavaScript에 대해 간략하게만 알아보자.

JavaScript: 웹에 쓰이는 단 하나의 프로그래밍 언어이다. 하나밖에 없다는 건, 불편할 순 있어도 어쩔 수 없이 써야되는 것이다.

* 프로그래밍 언어: 컴퓨터와 소통하기 위한 언어이다. 요즘은 google assistant, siri, alexa 처럼, 뭐 좀 해달라하면 알아서 척척 해주는 똑똑한 것들이 생겼지만, 그것들조차 조금만 복잡하게 말하면 무슨 말인지 못알아듣는다. 컴퓨터와 대화하기 위해선, 대충 말해도 알아듣는 사람의 언어와는 다르게, 정해진 프로그래밍 언어의 문법대로 똑바로 말해줘야한다. 전 세계에는 한국어, 영어, 일본어, 중국어 등등 약 6,500개의 언어가 있듯이, 프로그래밍 언어는 지금까지 약 700개가 만들어졌다.

만약 브라우저를 떠나서, 서버 개발자의 세계만 가도 수많은 기술들이 있어 선택의 자유가 있지만, 브라우저는 오로지 JavaScript만 프로그래밍 언어로 인정한다. 왜 그랬을까? 그렇게 하도록 정해놓았고, 안 바뀌었기 때문이다. 모든 브라우저는 오로지 JavaScript만 이해하며, 빠르게 변해가는 웹 시장의 수많은 제품들은 JavaScript로 작성되었다.

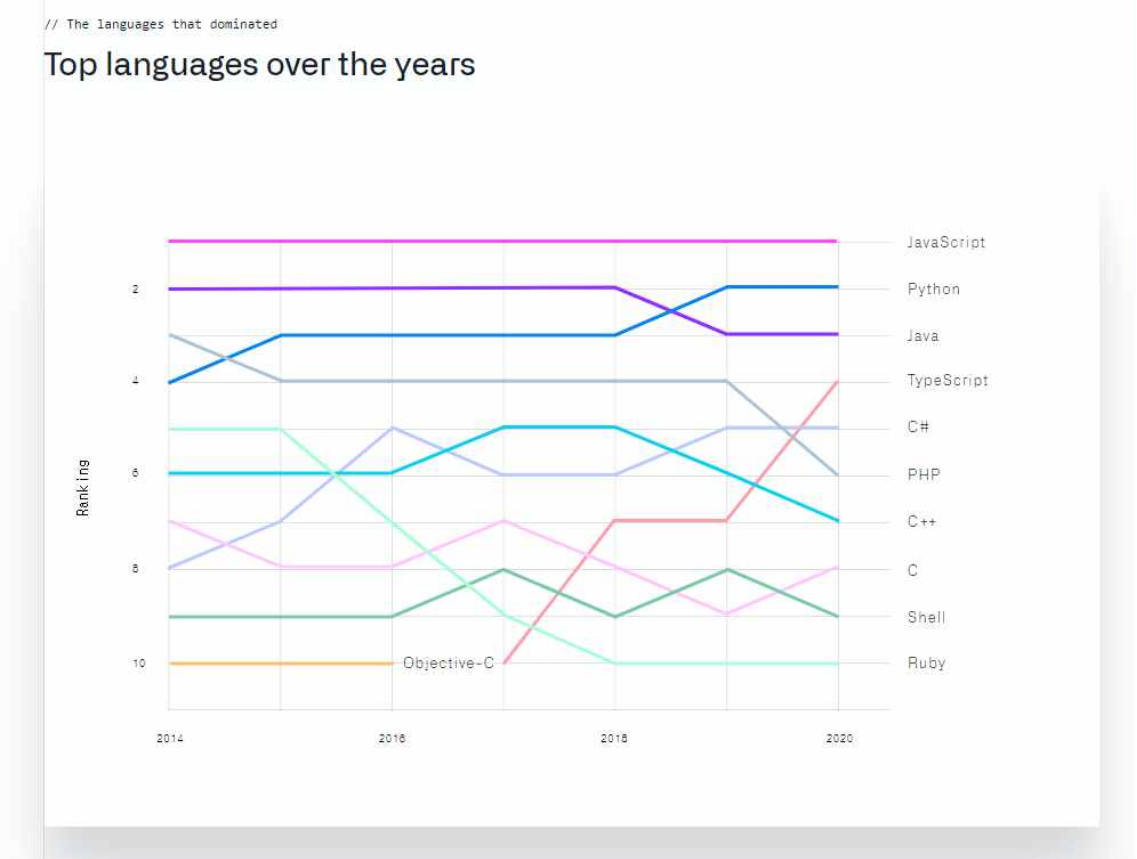
그러나 보통의 JavaScript개발자들은 개발 과정에서 버그 잡는데만 엄청난 시간을 쓴다. 특히, JavaScript는 Type이 없기 때문에 Type Check를 하지 않고, 데이터가 오면 그냥 받아들여 각종 에러를 발생시켰다. 조금 어려운 문장이라 무슨 말인지 모를수도 있겠으니, 간단하게만 설명해보겠다. 이건 마치 중국집에 초밥 주문이 들어오는것과 같다. 주방장은 적어도 메뉴판에 있는 메뉴가 주문 올 것이라고 확신할텐데, JavaScript라는 카운터 직원은 중국집 직원인데도 초밥, 피자, 김치찌개 주문 거절을 못해 주방을 올스톱시킨다. 즉, 프로그램이 멈춰버린다. 이처럼 무슨 데이터가 들어올지 예상할 수 없는 프로그램은 당연히 위험할 수밖에 없다.

TypeScript(TS)는 이런 문제점을 보완하기 위해 등장했다. TypeScript는 너무나 자유로운 JavaScript에 '규칙'을 부여한다. 해당 규칙은 프로젝트 진행 과정에서 발생할 수 있는 버그들을 최소화할 수 있다. 이것은 우리가 작성하는 코드가 예측 가능하고, 읽기 쉬워진다는것을 의미한다. 메뉴판에 있는 주문만 주문받고, 만약 다른 걸 주문하면 주방에 오기 전에 전화로 그런 메뉴 없다고 안내하는 똑똑한 카운터 직원과 같다.

TypeScript를 써야 할 다른 이유가 있다. 마이크로소프트가 직접 개발했고, 관리하는 언어이기 때문이다. 웹 시장에서 대기업이 직접 해당 기술을 관리한다는건 그 언어가 매우 안정적이라는 증거이며, 기술을 선택할 때 매우 중요한 요소이다.

Q. TypeScript를 배우기 전, JavaScript를 꼭 알아야 하는가?
A. TypeScript는 다른 언어가 아니다. JavaScript에 '규칙'들이 더해진 것일 뿐이다. 쉽게 말하면 '업그레이드 버전' 이며, 개선된 JavaScript이다. 따라서 JavaScript가 할 수 있는

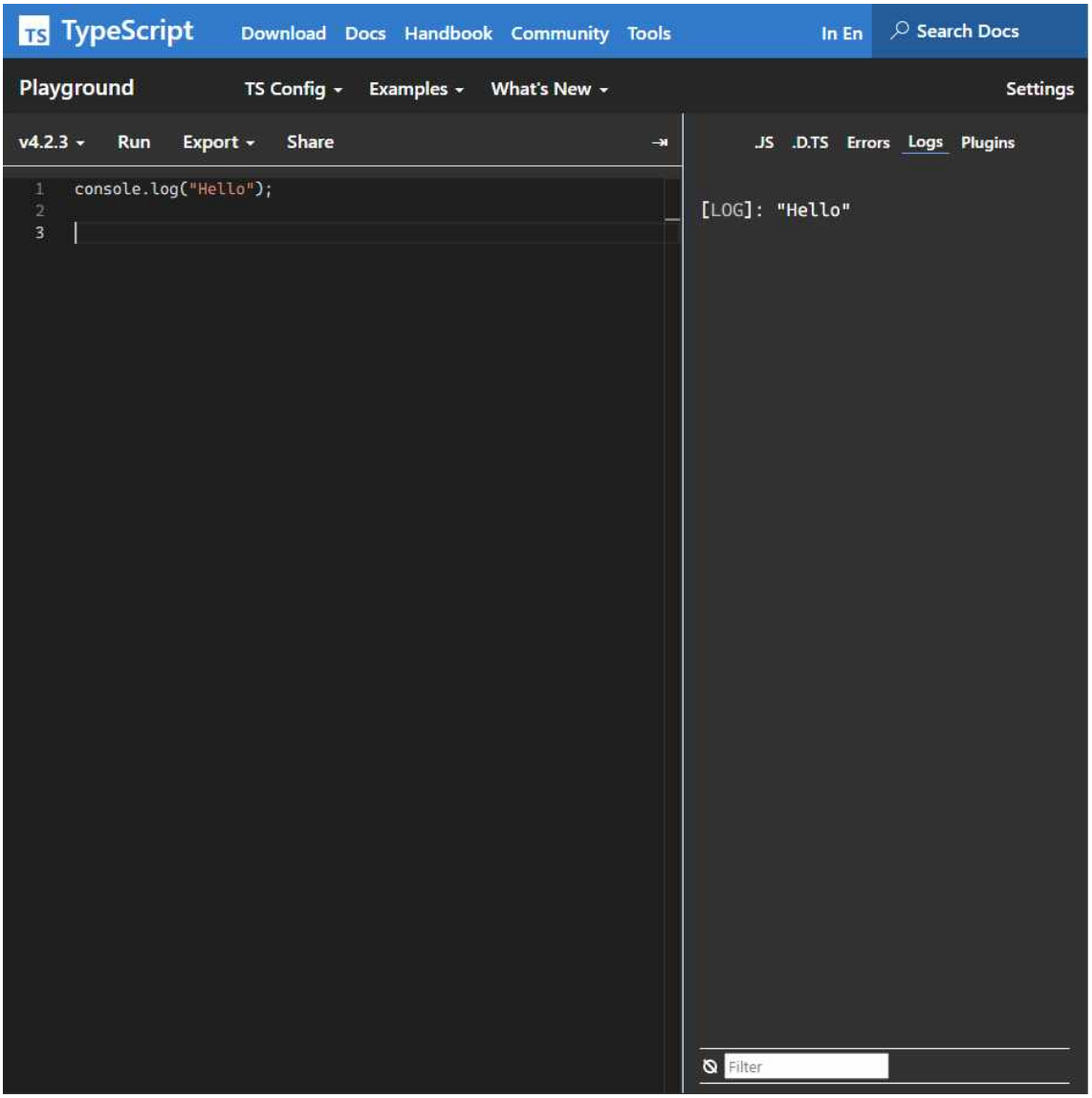
거의 모든 것을 TypeScript로도 당연히 할 수 있다. 말하자면, JavaScript를 먼저 제대로 공부하고 TypeScript를 배우는것보다, 시작부터 TypeScript로 배우는 게 오히려 낫다.



Github 연례보고서 2020 The State of the Octoverse.
TypeScript는 폭발적인 성장을 하고 있고, 이는 현재 진행형이다.

실습을 위해 별다른 개발환경세팅을 하진 않는다. 개발환경세팅을 경험해보는 건 매우 좋지만, 보통은 TypeScript만으로 개발하지 않고, 이 챕터의 목적은 단순히 TypeScript문법을 연습하는 것이기 때문에 개발환경세팅은 다음 챕터인 GraphQL Backend에서 해볼 것이다. 대신, Microsoft에서 제공하는 TypeScript Playground를 통해 TypeScript를 연습해보겠다.

PC의 크롬 브라우저로 다음 사이트에 접속하자.
https://www.typescriptlang.org/play



TypeScript Playground 화면. 기존에 쓰여져 있는 모든 걸 삭제하고,

```
console.log("Hello");
```

라고 입력해보자.

오른쪽에 Logs 탭을 클릭하면 결과를 볼 수 있다. 코드를 입력하고 ctrl+enter 를 누르면 오른쪽 콘솔에 결과가 나온다. 이것을, 개발자들은 '출력'되었다라고 표현한다. 오른쪽 아래 동그라미에 대각선 선이 그어져있는 버튼을 클릭하면 콘솔창이 깨끗하게 지워진다.

Hello가 정상적으로 나왔는가? 축하한다. 여러분의 첫 TypeScript 코드를 실행시켰다.

* 콘솔(console): 윈도우에서 '명령 프롬프트(cmd)', 'PowerShell', 리눅스에서 'Terminal'이라고 부르는 것이다. 까만 배경에 명령어를 치면 실행한 결과를 보여준다.

* console.log() 는 TypeScript에서 콘솔 화면에 데이터를 찍어볼 때 쓰는, console 객체 안에 있는 함수다. 함수가 뭔지, 객체가 뭔지 몰라도, 지금은 일단 뭔가를 테스트할 때 쓰는 것이라고 알아두자.

배워볼 내용은 다음과 같다.

1. 변수(Variable)
2. 자료형(Data Type)
3. 배열(Array)
4. 객체(Object)
    - TypeScript에서 말하는 객체란?
5. 인터페이스(Interface)
6. 함수(Function)

이제 모든 프로그래밍 언어 교과서의 첫 장인, 변수부터 시작해보자.

1. 변수(Variable)

1-1. let

* 지금부터, 읽다보면 '코딩 습관 들이기'라는 매우 지루할 수 있는 규칙 설명란이 나올 것이다. 사실 '코딩 습관 들이기' 부분은, 무시해도 프로그램은 잘 실행되는 것들만 써놨다. 그래서 어려운 게 싫은 사람들은, 나올때마다 무시해도 좋으나, 좋은 습관을 미리 익혀놓고 싶은 사람들은 처음 배울때부터 습관을 들여놓도록 하자. 대형 앱(프로그램)을 만드는 고수 수준이 될수록 들여놓은 습관은 큰 도움이 된다.

컴퓨터는 기본적으로 계산기다. 페이스북같은 대형 웹앱도, 근본적으로는 끝없는 계산식으로 이루어져 있다.

정말 간단한 계산 하나 해보자.

```
1   console.log(2 + 3);
```

결과:

```
[LOG]: 5
```

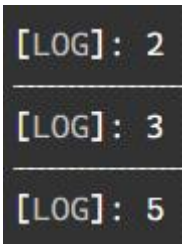
코딩 습관 들이기:
  연산자(+) 와 피연산자(2 그리고 3) 사이엔 칸을 한 칸 띄어준다.
  어떤 하나의 문장을 끝마칠때는, 세미콜론(;)을 찍어준다.

실전상황에서 이 코드는 매우 쓸모없는 것이다. 2 + 3이 정해져있으면 그게 무슨 계산기인가? 적어도, a + b 처럼, a에 어떤 게 들어오든지, b에 어떤게 들어오든지 둘을 더한 결과(+)를 낼 수 있어야 한다.

변수(variable): 변경되거나, 변할 수 있는 수. 우리는 쓰고싶은 변수에 이름을 지어줘야 한다. dog, cat, jajangmyeon으로 지어도 되지만, 나는 내가 쓰고자 하는 변수를 a, b로 이름지어주겠다. 실전에선 이렇게, 뭔지 한번에 알아챌 수 없는 이름으로 지으면 안되지만, 간단한 예시를 보여주기 위해 a, b를 쓰겠다.
참고로, "짜장면" 이라고 한글로 지으면 안된다. 뭐가 되었든지, 영어로 써야한다.

```
1   let a
2   let b;
3   a = 1;
4   a = 2;
5   b = 3;
6   console.log(a);
7   console.log(b);
8   console.log(a + b);
```

결과:



코딩 습관 들이기:
‘=’ 이 가운데에 있을 때는 한 칸 띄워준다.

이 그림을 설명하기 전에, 프로그래밍의 기초로서 확실히 익히고 넘어가야할 게 있다. 위 코드는 수학식처럼 보이지만 수학하던 방식으로 읽으면 안된다. 수학에선, 글을 읽듯이 왼쪽에서 오른쪽으로 읽는다. line3를 수학하던 방식으로 읽으면, ‘a는 1’이다. 그러나, 프로그래밍에서 ‘=’ 기호는 ‘입력’ 을 의미하므로, 변수를 읽을때는 오른쪽에서 왼쪽으로 읽어야 한다. 즉, ‘1을 a에 집어넣는다’로 이해해야 한다.

이제 변수를 이해해보자. 결과를 보면,
a는 2이고, b는 3이고, a + b는 5이다. a는 line2에서 1이었지만, line3에서 2로 변했다.
그래서 변수다. 변경되거나, 변할 수 있는 수.

let은 대체 뭐하는 놈일까? a라는 변수를 쓰기 위해선 컴퓨터에게 이걸 쓰겠다고 “선언”해야한다.
선언은 컴퓨터에게 ‘나 이걸 쓰겠다’ 고 말하는 행동이다. 쓰고자 하는 변수 앞에 let을 붙여 ‘선언’한다. 그리고 선언 이후 그 변수를 ‘사용’할때는 let을 쓰지 않는다. line2, line3, line4 에서 변수를 사용하지만 let은 쓰지 않았다.

변수 선언하고, 값이 주어지면, 그 변수는 ‘초기화(initialize. 줄여서 init)’되었다 라고 한다. line2 에서 a는 초기화가 완료된 것이다. 어떤 변수를 선언해 쓰고자 할 땐, 반드시 초기화까지 마쳐야한다.

* let 은 영어단어 let 에서 나왔다. ‘~하게 하다’ 라는 의미다.

* 선언은 명사로 declaration, 동사로 declare 라고 쓴다. 이 영어단어는 공개석상이나 연설, 정부기관에서 쓰는, 영어권 사용자들도 언론에서나 보는 어려운 단어다. 한국어로 번역된 ‘선언’이라는 단어도, 일상생활에선 잘 쓰지 않는다. 개발자들은 프로그래밍을 너무 많은 사람들이 배워서 자기들 밥줄이 끊기는 걸 염려해서인지, 가장 기본적인 개념을 설명하면서도 이런 어려운 단어를 쓰곤 한다.

위 코드는 다음과 같이 짧게 줄일 수 있다.

```
1   let a, b;
2   a = 1;
3   a = 2;
4   b = 3;
5   console.log(a, b, a + b);
```

결과:

```
[LOG]: 2,  3,  5
```

코딩 습관 들이기:
  콤마 뒤엔 한 칸 띄워준다.

line1: 여기선 a, b를 동시에 선언하기 위해 콤마로 구분지었다.
line5: console.log()에서 여러개를 출력시키고 싶을 땐 콤마를 쓴다.

선언과 동시에 값을 넣을수도 있다. 즉, 한 줄에 초기화까지 마칠 수 있다.

다음과 같이, a를 선언함과 동시에 0을 집어넣어 초기화시켰다.

```
1   let a = 0;
2   a = 2;
3   console.log(a);
```

결과:

```
[LOG]: 2
```

개발하면서 변수를 쓸 땐 항상 이렇게, 변수 초기화까지 한 줄에 마치는 게 권장된다.

1-2. const

그러나, 지금까지 설명했던것과는 다르게, 우린 let을 사용하지 않고, 대부분의 경우 const를 사용할 것이다. const로 프로그래밍을 하다가 안 돌아갈 때 let을 쓰면 해결되는 경우를 제외하고, const만 쓴다. 그럼 잘 쓰지도 않는 let은 대체 왜 이제껏 설명한 것인가?

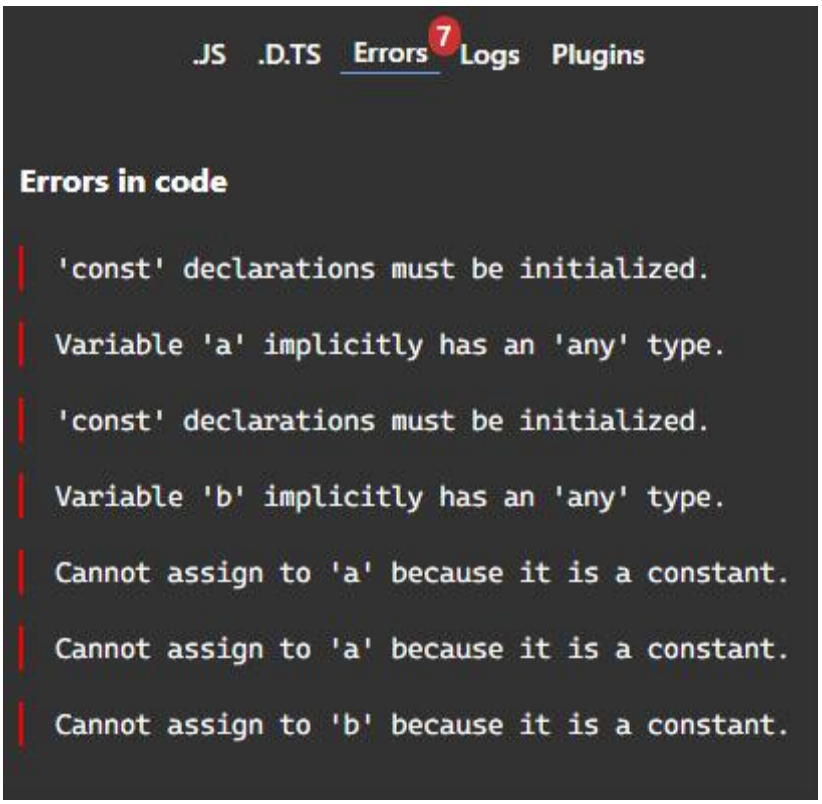
const는 constant 의 줄임말이다. 즉, 특수한 변수인 '상수'인데, 특별한 경우를 제외하곤 보통 변수라고 부른다. 중요한 건, const는 '변하지 않는다.'

변하지 않는데 어떻게 변수가 된단 말인가? 지금부터 하는 얘기는 이해하려 하지 말고 받아들였으면 좋겠다. 개발자들은 const로 선언되었어도, 특수하게 이름지은 케이스(이름 전체를 대문자로 지은 것. ex. NAME)를 제외하곤 '상수'라고 안 부르고 '변수'라고 부른다. 다시 말하면, const로 선언된 경우, 변수처럼 '변경되거나, 변할 수 있는 수'가 아님에도, '변수'라고 부른다.

const에 대해서 설명을 늘어놓는것보다 코드를 보는 게 이해하기 쉬울 것이다. let과의 차이점을 확인해보자. let을 사용했던 코드를 const로 바꿔보았다.

```
1   const a, b;
2   a = 1;
3   a = 2;
4   b = 3;
5   console.log(a, b, a + b);
```

빨간색 물결 밑줄이 있다는 건, 에러가 있다는 뜻이다. 무슨 에러인지 확인하기위해, 오른쪽 상단 Errors 탭을 클릭해보자.



개발자에게 에러는 적이자 친구이다. 단언컨데, 우리의 개발 실력은 에러를 얼마나 많이 해결해 봤는지(debug)에 따라 늘 것이다. 영어라서 겁이 나는가? 미안하지만, 개발자는 영어와 친숙해져야된다. 그렇다고 영어를 매우 잘하라는 뜻이 아니다. 나도 구글번역과 네이버 사전의 도움을 많이 받는다.
여기서 type 에러는 지금 공부할 단계가 아니다. 중요한 에러는 크게 두가지로 볼 수 있다.

1. 'const' declarations must be initialized.
const 선언은 반드시 초기화되어야 합니다.
line2에서 분명 초기화하지 않았는가? 그러나, const는 그런 거 허용하지 않는다. 반드시, 선언부터 초기화까지 한 줄에 마쳐야한다.

2. Cannot assign to 'a' because it is a constant.

a에 assign할 수 없습니다. 왜냐면, constant 이기 때문입니다.
assign은 디버깅하다가 많이 보게 될 단어인데, 한국어로 '배정'이다. 쉽게 말하면 '무언가를 집어넣는다'는 뜻이라고 이해하면 된다. 풀어서 이해해보면,

a에 집어넣을 수 없습니다. 왜냐면, constant이기 때문입니다.
즉, const는 변하지 않는다. 이미 1을 집어넣었기 때문에, 2를 집어넣으려 해도 안 들어가는 것이다.

즉, const는 let과는 다른 두가지 특징이 있다.
1. 선언과 동시에 초기화를 마쳐야한다.
2. 한번 값을 집어넣었으면 바꿀 수 없다.

이것이 대부분의 경우에 let 대신에 const를 써야 될 이유라고 할 수 있는가? 그렇다! 프로그래머는 자신의 변수에 무엇이 들어있는지 확신할 수 있어야한다. 변수가 중간에 변해버리면, 에러 찾기 매우 힘들다. 내가 웹사이트를 운영하는데, 사용자가 입력한 값에 따라서, 변하면 안되는 변수들이 막 바뀐다고 생각해보라. 사이트는 엉망이 될 것이 뻔하다.

해당 에러를 수정해 코드를 다시 써보면 다음과 같다.

코드1)

```
1   const a = 1, b = 3;
2   console.log(a, b, a + b);
```

결과:

```
[LOG]: 1,  3,  4
```

그럼 당연히 이런 질문이 생긴다. 변하지도 않는 놈을 어디다 쓴단 말인가?

코드2)

```
1   console.log(1 + 3);
```

대체 이 한줄의 코드와 뭐가 다른걸까? a에 이미 1이 있고, b에 이미 3이 있으면 대체 무슨 쓸모가 있는가?

그러나, 코드1과 코드2는 큰 차이점이 있다. 변수를 썼다는 것이다. 지금은 단순히 1과 3일 뿐이지만, 나중에 React를 배울때면 다음을 할 수 있을 것이다.

사용자로부터 1과 3을 입력받아, 1을 a에, 3을 b에 집어넣는다. 결과는 4다.
만약 프로그램을 다시 시작하고, '다른 값' 을 넣어보자.
사용자로부터 3과 -2를 입력받아, 3을 a에, -2를 b에 집어넣는다. 결과는 1이다.

즉, a에 어떤 게 들어오든지, b에 어떤게 들어오든지 둘을 더한 결과(+)를 낼 수 있다. 그땐 지금처럼 코드에 1 과 3이라고 써져있는게 아니기 때문이다. 코드에서 볼 땐 안 변하지만, 프로그램이 다시 시작되면 '변한다.'

1-3. var

가끔 다른 개발자들이 쓴 옛날 코드를 볼 때가 있는데, let도, const도 아닌 var를 사용해 변수 선언하는것을 볼 수 있다. var는 2014년까지 쓰이던 구식 변수 선언법이고, 매우 불안정하고 위험하니 쓰지마라. 단, 옛날 코드를 볼 때 var가 나왔다면, 변수를 선언했구나 하고 이해하면 된다.

이제 변수에 숫자말고 다른 것들을 집어넣어보자. 다음 주제인 자료형(Data Type)에서 다룰 것이다.

핵심정리:

const - 대부분의 변수를 선언할 때 사용.
프로그램 실행되는 동안엔 안 변한다.
let - const를 썼을 때 작동 안하면 사용
var - 사용하지 말 것.

개발 실력은 디버그를 많이 할수록 는다.

2. 자료형(Data Type)