01_JavaScript개요

# 챕터번호	1

1. JavaScript 개요

1-1. JavaScript의 역사

JavaScript는 웹 페이지의 보조적인 기능을 수행하기 위해 브라우저에서 동작하는 경량 프로그래밍 언어로서 1996년 넷스케이프 커뮤니케이션즈에서 처음 개발되었다. 이어 마이크로소프트에서도 JavaScript의 파생 버전인 JScript를 만들었는데 두 회사는 자사 브라우저의 시장 점유율을 높이기 위해 자사 브라우저에서만 동작하는 기능을 경쟁적으로 추가했다. 이는 **크로스 브라우징 이슈**-브라우저에 따라 웹 페이지가 정상적으로 동작하지 않는 현상-을 야기했고 표준화 된 자바스크립트의 필요성이 대두되었다.

컴퓨터 시스템의 표준을 관리하는 비영리 표준화 기구인 ECMA 인터네셔널에서 1997년 ECMA-262라 불리는 표준화 된 자바스크립트 초판(ECMAScript 1) 사양이 완성되었고, 상표권 문제로 자바스크립트는 ECMAScript로 명명되었다. 이후 1999년 ECMAScript 3(ES3)이 공개 되고, 10년만인 2009년에 출시된 **ECMAScript 5(ES5)는 HTML5와 함께 출현한 표준 사양**이다. 2015년에 공개된 **ECMAScript 6(ECMAScript 2015, ES6)는 let/const 키워드, 화살표 함수, 클래스 등과 같이 범용 프로그램이 언어로서 갖춰야 할 기능들을 대거 도입하는 큰 변화**가 있었다.

ECMAScript는 자바스크립트 표준 사양으로 프로그래밍 언어의 값, 타입, 객체와 프로퍼티, 함수, 표준 빌트인 객체 등 핵심 문법을 규정한다. 각 브라우저 제조사는 ECMAScript 사양을 준수해서 브라우저에 내장되는 자바스크립트 엔진을 구현한다. 자바스크립트는 일반적으로 프로그래밍 언어로서 기본 뼈대를 이루는 ECMAScript와 브라우저가 별도 지원하는 클라이언트 사이드 Web API(DOM, BOM, XMLHttpRequest 등등)를 아우르는 개념이다.

1-2. JavaScript의 성장

- Ajax(1999년)
  - 자바스크립트를 이용해 서버와 브라우저가 비동기 방식으로 데이터를 교환할 수 있는 통신 기능
  - 전체 웹 페이지 렌더링-HTML, CSS, JavaScript로 작성 된 문서를 해석해서 브라우저에 시각적으로 출력하는 것-이 아닌 필요한 데이터만 한정적으로 렌더링하여 빠른 성능과 부드러운 화면 전환이 가능해졌다.
- jQuery(2006년)
  - 번거롭던 DOM을 쉽게 제어할 수 있게 되었고 크로스 브라우징 이슈에도 많은 도움을 주었다.
- V8 자바스크립트 엔진(2008년)

    - 더욱 빠르게 동작하는 구글 V8 자바스크립트 엔진의 개발로 데스크톱 애플리케이션과 유사한 사용자 경험을 제공할 수 있는 웹 어플리케이션 프로그래밍 언어로 정착하게 되었다.
    - 웹 서버에서 수행되던 로직이 클라이언트(브라우저)로 이동하는 등 프론트엔드 영역이 주목받는 계기가 되었다.
- Node.js(2009년)
    - 브라우저의 자바스크립트 엔진에서만 동작하던 자바스크립트를 브라우저 이외의 환경에서도 동작할 수 있도록 자바스크립트 엔진을 브라우저에서 독립시킨 자바스크립트 런타임 환경이다.
    - 서버 사이드 애플리케이션 개발에 주로 사용되며 이에 필요한 내장 API를 제공한다.
    - 이제 자바스크립트는 프론트엔드는 물론 백엔드 영역까지 아우르는 웹 프로그래밍 언어의 표준으로 자리잡고 있다.
- SPA 프레임워크
    - 모던 웹 어플리케이션은 개발 규모와 복잡도가 날로 상승했으며 복잡해진 개발 과정 수행을 위해 많은 프레임워크가 등장했다.
    - CDB(Component based development) 방법론을 기반으로 하는 SPA(Single Page Application)가 대중화 되면서 Angular, React, Vue.js 등 다양한 SPA 프레임워크/라이브러리가 많은 사용층을 확보하고 있다.

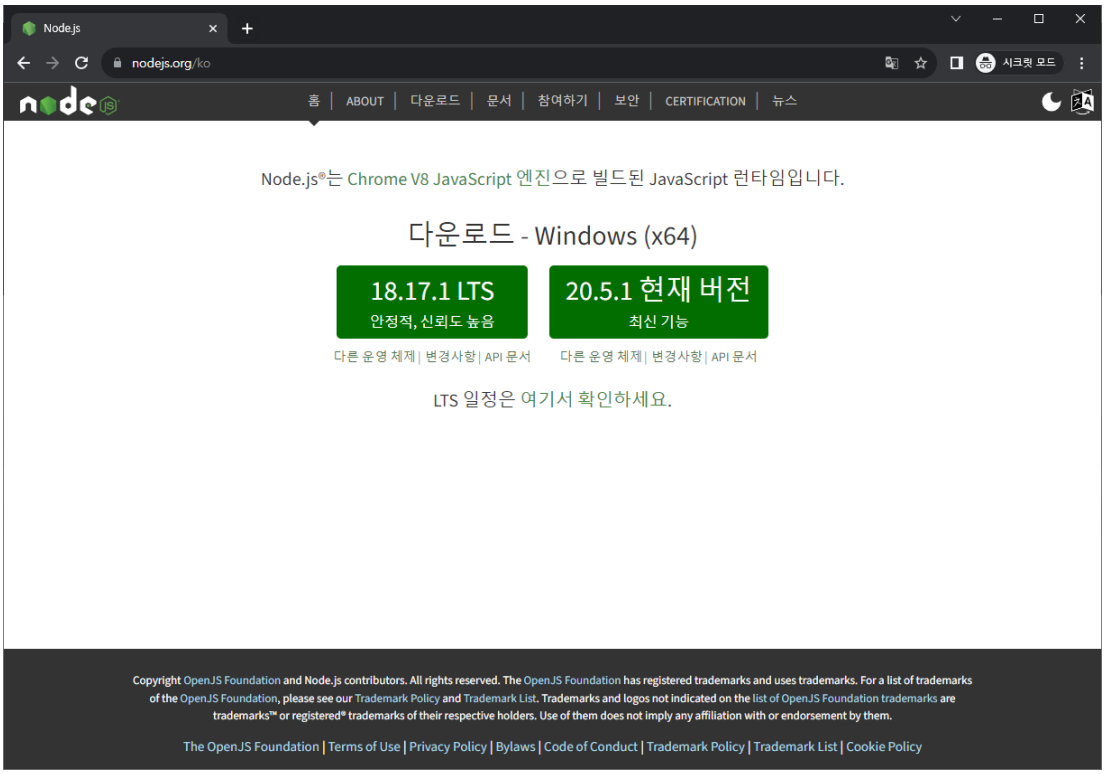
1-3. JavaScript의 특징

- **웹 브라우저에서 동작하는 유일한 프로그래밍 언어**이다.
    - 기본 문법은 C, Java와 유사하지만 프로토타입 기반, 일급 함수의 개념 등은 다른 언어에서 차용했다.
- 개발자가 별도의 컴파일 작업을 수행하지 않는 **인터프리터 언어**이다.
    - 인터프리터 언어 : 코드가 실행되는 단계인 런타임에 문 단위로 한 줄씩 중간 코드인 바이트코드로 변환한 후 실행하는 언어
    - 대부분의 모던 자바스크립트 엔진은 인터프리터와 컴파일러의 장점을 결합해 비교적 처리 속도가 느린 인터프리터의 단점을 해결했다.
- 클래스 기반 객체지향 언어보다 효율적이면서 강력한 **프로토타입 기반의 객체지향 언어**이다.

1-4. 개발환경구축

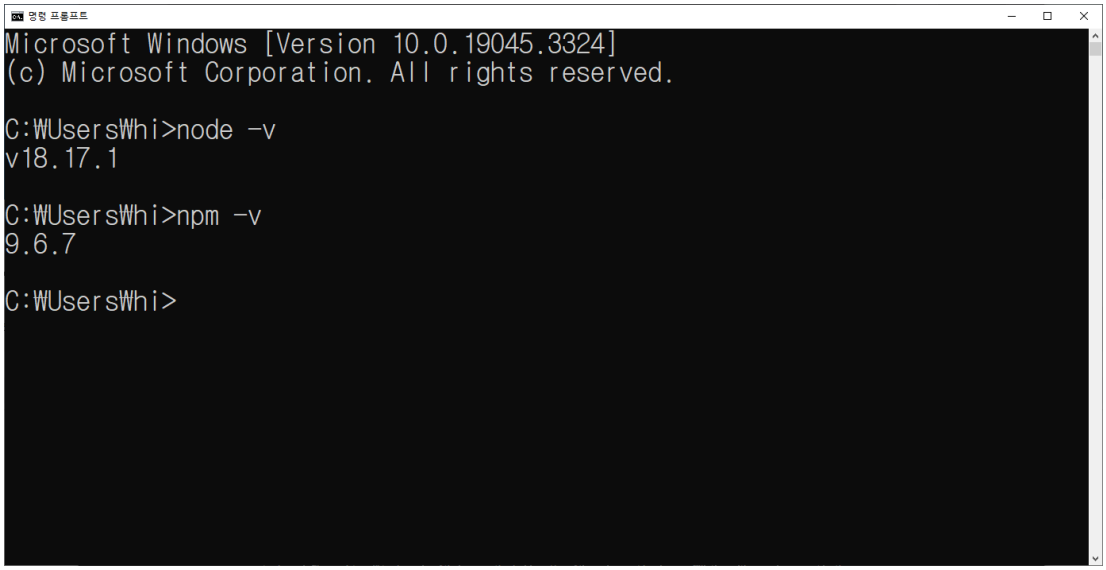
- 자바스크립트는 브라우저 환경 또는 Node.js 환경에서 실행할 수 있다.
    - 두 환경 모두 자바스크립트의 코어인 ECMAScript는 실행할 수 있다.
    - 브라우저는 파싱 된 HTML 요소를 선택하거나 조작하는 기능의 집합인 DOM API를 기본적으로 제공하지만 브라우저 외부에서 자바스크립트 개발 환경을 제공하는 것이 주 목적인 Node.js는 DOM API를 제공하지 않는다.

    - 반대로 Node.js에서는 파일 시스템을 기본 제공하지만 브라우저는 보안상의 이유로 이를 제공하지 않는다.
- 다양한 웹 브라우저 환경 중 ECMAScript 사양을 준수하며 시장 점유율도 높은 구글 Chrome 브라우저를 사용한다.
    - 개발자 도구, 콘솔, 디버깅 기능 등을 활용한다.
- 간단한 개발은 브라우저만으로도 가능하지만 프로젝트 규모가 커 다양한 프레임워크/라이브러리 도입이나 여러 도구를 사용이 필요하다면 Node.js 환경을 사용한다.
    - Node.js(https://nodejs.org/) 접속
    - LTS(Long Term Support) 버전 다운로드 후 기본 설정으로 설치

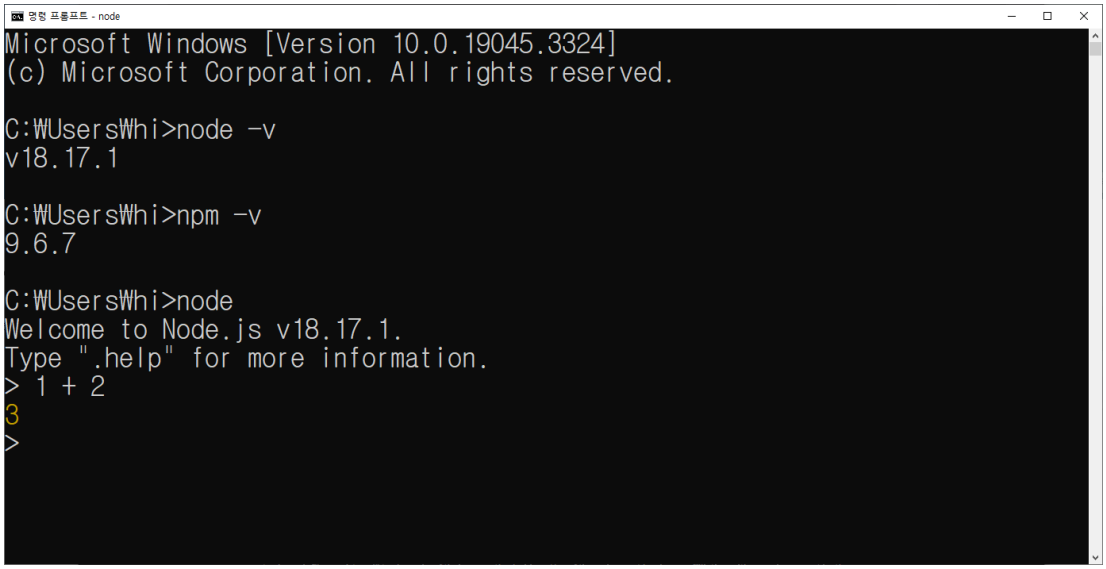


Node.js homepage

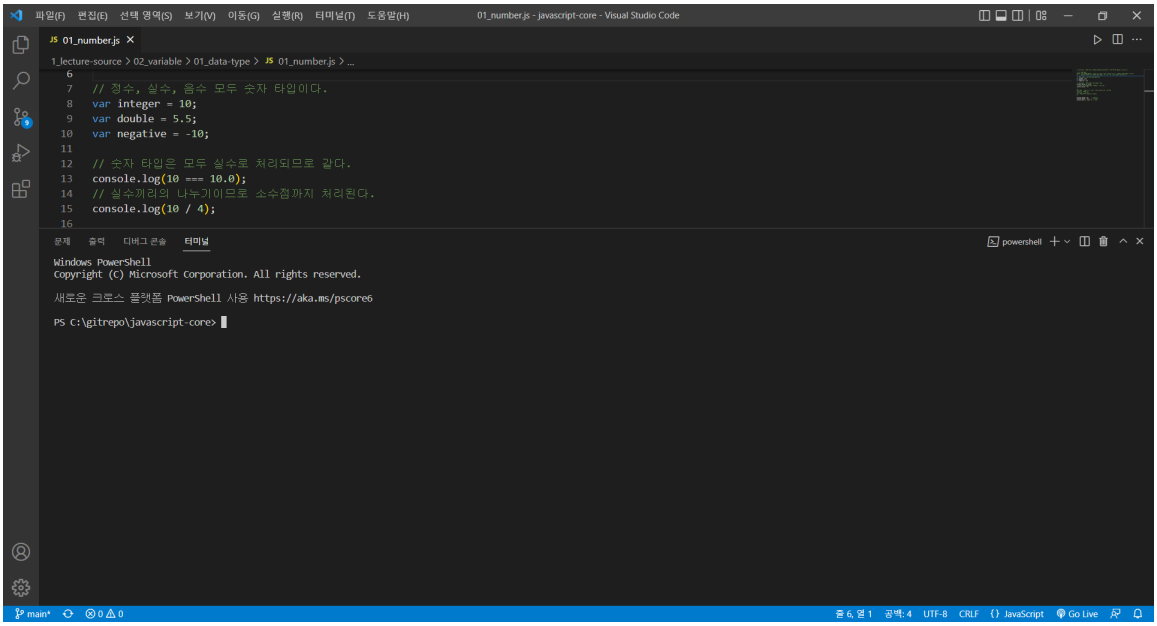
    - 설치 완료 후 cmd 창에서 `node -v`, `npm -v` 입력하여 Node.js와 함께 설치 된 자바스크립트 패키지 매니저인 npm(node pacakge manager)의 버전을 출력해 정상적으로 설치되었는지 확인



- cmd에서 `node`라는 명령어를 실행하면 프롬프트가 `>`로 변경되고 자바스크립트 코드를 실행해볼 수 있다.

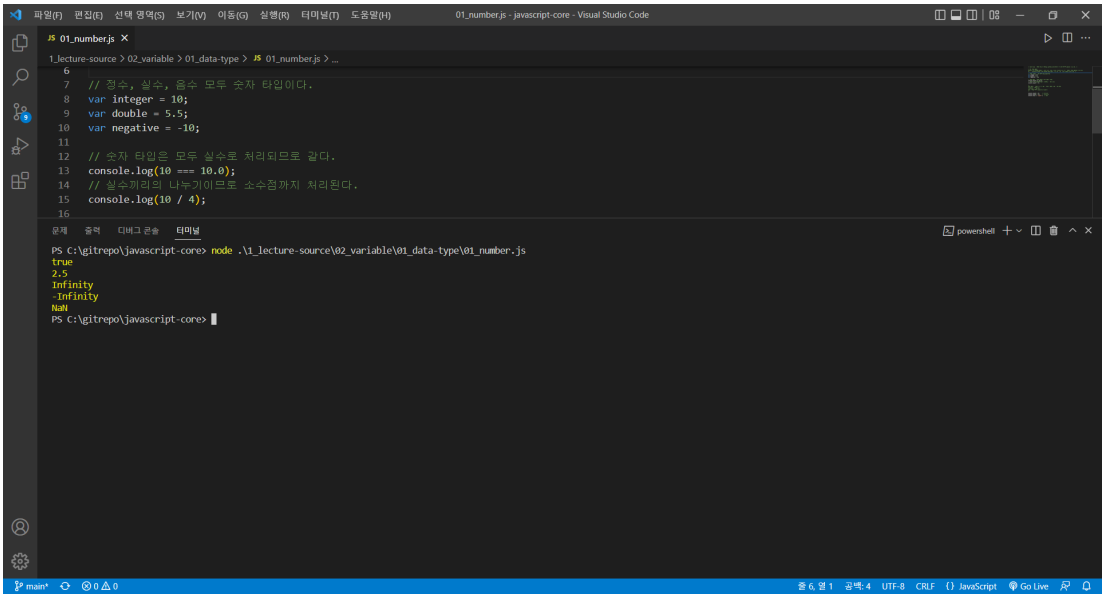


- VS Code
  - 브라우저의 콘솔, Node.js의 REPL(Read Eval Print Loop)에서 자바스크립트 코드를 실행할 수 있지만 애플리케이션 개발을 위해서는 코드 에디터의 기능이 필요하므로 VS Code를 통해 작업한다.
  - VS Code에서 제공하는 내장 터미널은 Ctrl + ` 를 누르면 열린다.

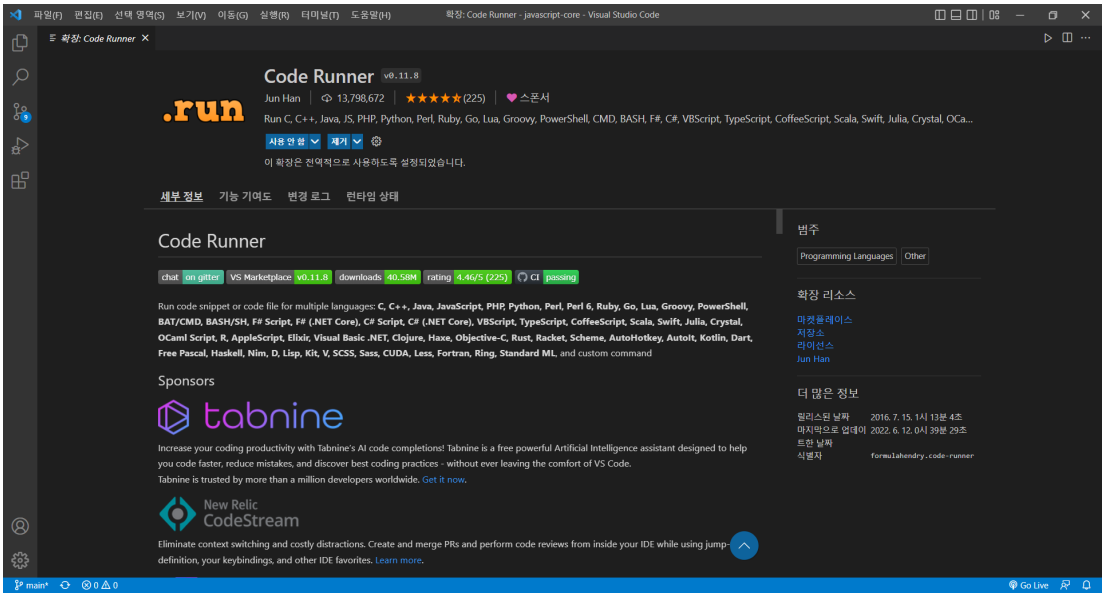


Node.js cmd

- 프롬프트에 `node 파일명`을 입력하여 자바스크립트 파일을 실행할 수 있다.



- 확장(Extension)에서 Code Runner를 Install



- Ctrl + Alt + N 을 누르면 현재 표시 중인 자바스크립트 파일을 실행