

JavaScript Seminar

Chap 1. 프로그래밍

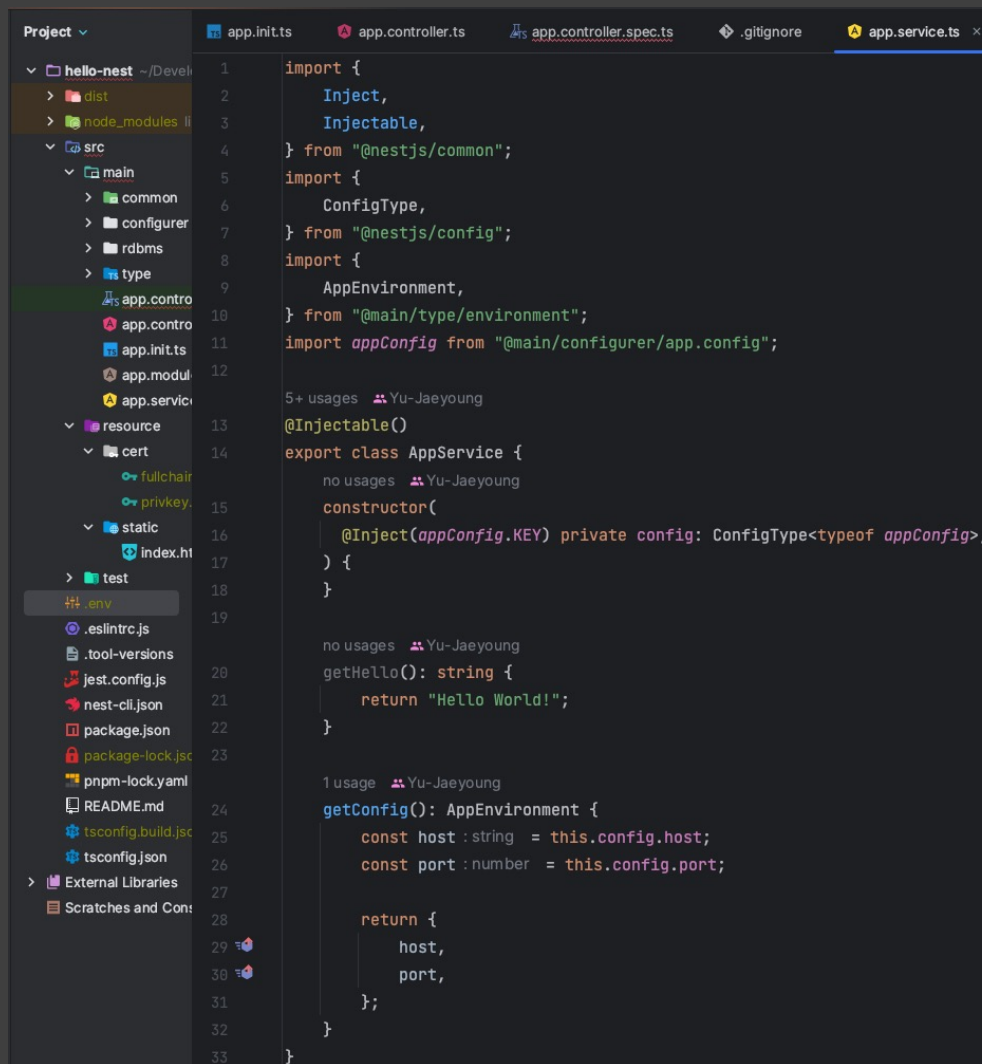
목차

- **Chap 1. 프로그래밍**
 - 프로그래밍이란?
 - 프로그래밍 언어
 - 구문과 의미
- **Chap 2. 자바스크립트란?**
 - 자바스크립트의 탄생
 - 자바스크립트의 표준화
 - 자바스크립트의 성장의 역사
 - Ajax
 - jQuery
 - V8 자바스크립트 엔진
 - Node.js
 - SPA 프레임워크
 - 자바스크립트와 ECMAScript
 - 자바스크립트의 특징
 - ES6 브라우저 지원 현황
- **Chap 3. 자바스크립트 개발 환경과 실행 방법**
 - 자바스크립트 실행 환경
 - 웹 브라우저
 - 개발자 도구
 - 콘솔
 - 브라우저에서 자바스크립트 실행
 - 디버깅
 - Node.js
- **번외. npm, pnpm**

프로그래밍이란?

*0과1밖에 알지 못하는 기계가 실행할 수 있을 정도로
정확하고 상세하게 요구사항을 설명하는 작업*

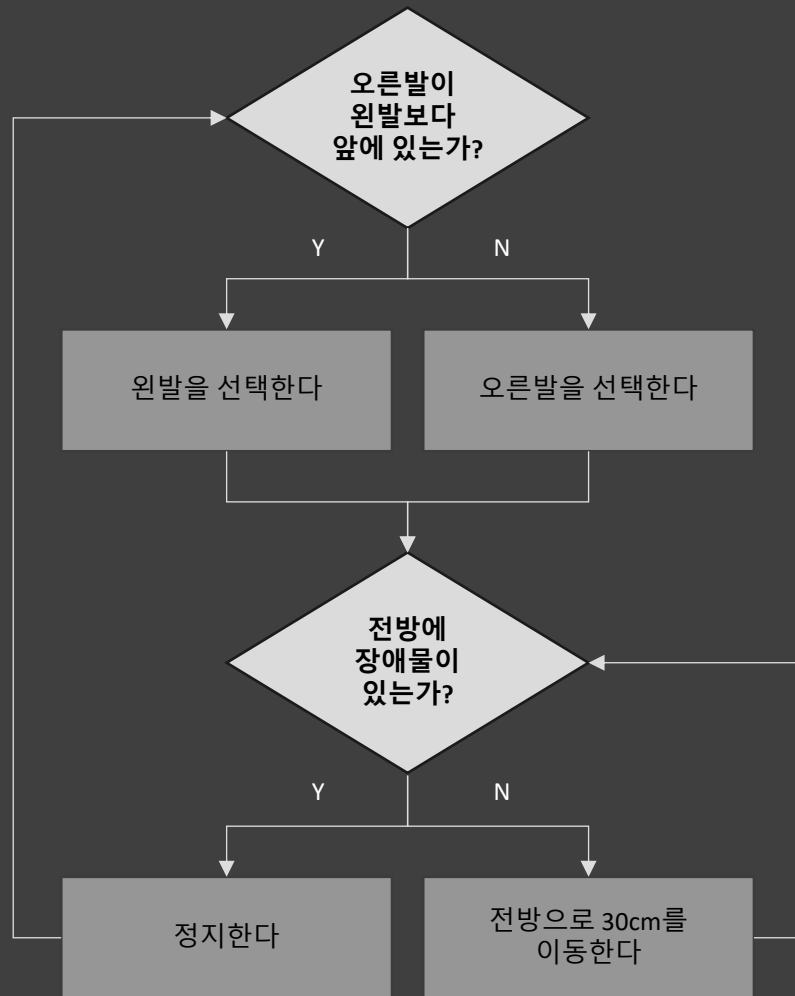
프로그래밍이란?



The screenshot shows a code editor with a project explorer on the left and a code editor on the right. The project explorer shows a directory structure for a project named 'hello-nest'. The code editor shows the implementation of the 'AppService' class in 'app.service.ts'.

```
1 import {
2   Inject,
3   Injectable,
4 } from "@nestjs/common";
5 import {
6   ConfigType,
7 } from "@nestjs/config";
8 import {
9   AppEnvironment,
10 } from "@main/type/environment";
11 import appConfig from "@main/configurer/app.config";
12
13 5+ usages Yu-Jaeyoung
14 @Injectable()
15 export class AppService {
16   no usages Yu-Jaeyoung
17   constructor(
18     @Inject(appConfig.KEY) private config: ConfigType<typeof appConfig>,
19   ) {
20
21   no usages Yu-Jaeyoung
22   getHello(): string {
23     return "Hello World!";
24   }
25
26   1 usage Yu-Jaeyoung
27   getConfig(): AppEnvironment {
28     const host: string = this.config.host;
29     const port: number = this.config.port;
30
31     return {
32       host,
33       port,
34     };
35   }
36 }
```

프로그래밍이란?



사람처럼 두 발로 걷는 로봇을 위해
“걷다”라는 기능을 디자인

프로그래밍 언어



*사람이 이해할 수 있는 자연어가 아닌,
컴퓨터가 이해할 수 있는 언어 "기계어"로 명령을 전달해야함*

프로그래밍 언어



약속된 구문(문법)으로 구성된 "프로그래밍 언어"를 사용해 프로그램을 작성한 후,
그것을 컴퓨터가 이해할 수 있는 기계어로 변환하는 번역기를 이용하는 것

컴파일러 / Compiler or 인터프리터 / Interpreter

구문과 의미

```
const number = 'string';  
console.log(number * number); // NaN
```

프로그래밍은 요구사항의 집합을 분석, 적절한 자료구조와 함수의 집합으로 변환 후, 흐름을 제어하는 것

자바스크립트의 탄생



변화 과정 : *Mocha* → *LiveScript* → *JavaScript*

자바스크립트의 표준화



크로스 브라우징 이슈 발생!!

자바스크립트의 표준화

비영리 표준화 기구 ECMA International

버전	출시 연도	특징
ES1	1997	초판
ES2	1998	ISO/IEC 16262 국제 표준과 동일한 규격을 적용
ES3	1999	정규 표현식, try ... catch
ES5	2009	HTML5와 함께 출현한 표준안 JSON, strict mode, 접근자 프로퍼티, 프로퍼티 어트리뷰트 제어, 향상된 배열 조작 가능(forEach, map, filter, reduce, some, every)
ES6	2015	let/const, 클래스, 화살표 함수, 템플릿 리터럴, 디스트럭처링 할당, 스프레드 문법, rest 파라미터, 심벌, 프로미스, Map/Set, 이터러블, for ... of, 제너레이터, Proxy, 모듈 import/export
ES7	2016	지수(**) 연산자, Array.prototype.includes, String.prototype.includes
ES8	2017	async/await, Object 정적 메서드 (Object.values, Object.entries, Object.getOwnPropertyDescriptors)
ES9	2018	Object rest/spread 프로퍼티, Promise.prototype.finally, async generator, for await ... of
ES10	2019	Object.fromEntries, Array.prototype.flat, Array.prototype.flatMap, optional catch binding
ES11	2020	String.prototype.matchAll, BigInt, globalThis, Promise.allSettled, null 병합 연산자, 옵셔널 체이닝 연산자, for ... in enumeration order
ES12	2021	String.prototype.replaceAll, promise.any(), AggregateError, 새로운 논리 할당 연산자(??=, =, &&=, ??=), 약한 참조(WeakRef), FinalizationRegistry, 가독성 있는 숫자 리터럴 작성(Numeric literal separators)

자바스크립트 성장의 역사



랜더링 / rendering

Ajax



Asynchronous JavaScript and XML

데스크톱 애플리케이션과 유사한
빠른 성능과 부드러운 화면 전환



XML ?

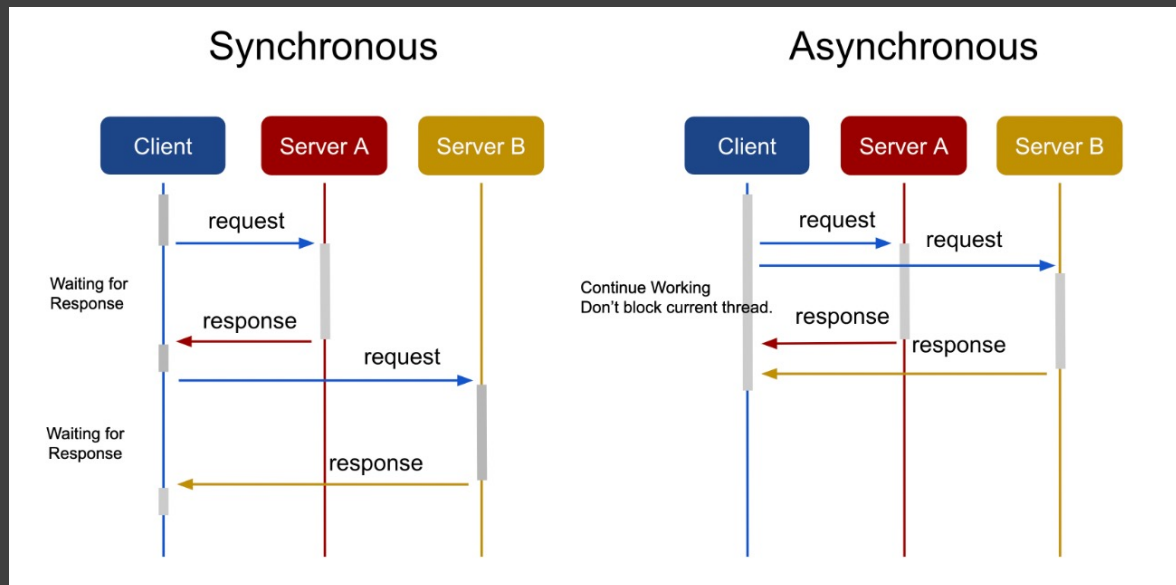


eXtensible Markup Language

주로 다른 종류의 시스템 중 인터넷에 연결된 시스템끼리
데이터를 쉽게 주고 받을 수 있게 함.

→ HTML의 한계를 극복할 목적으로 만들어짐

Asynchronous ?



jQuery



```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>jQuery Hide&Show</title>
6   <script src="./jquery-3.7.0.js"></script>
7   <script>
8     $(document).ready(function () {
9       $("#hide").click(function () {
10         $("p").hide();
11       });
12       $("#show").click(function () {
13         $("p").show();
14       });
15     });
16   </script>
17 </head>
18 <body>
19
20 <p>If you click on the "Hide" button, I will disappear.</p>
21
22 <button id="hide">Hide</button>
23 <button id="show">Show</button>
24
25 </body>
26 </html>
```

```
jquery-3.7.0.js x
1  /*!
2   * jQuery JavaScript Library v3.7.0
3   * https://jquery.com/
4   *
5   * Copyright OpenJS Foundation and other contributors
6   * Released under the MIT license
7   * https://jquery.org/license
8   *
9   * Date: 2023-05-11T18:29Z
10  */
11  Yu-Jaeyoung
12  ( function( global : Window | any , factory ) : void {
13
14    "use strict";
15
16    if ( typeof module === "object" && typeof module.exports === "object" ) {
17
18      // For CommonJS and CommonJS-like environments where a proper `window`
19      // is present, execute the factory and get jQuery.
20      // For environments that do not have a `window` with a `document`
21      // (such as Node.js), expose a factory as module.exports.
22      // This accentuates the need for the creation of a real `window`.
23      // e.g. var jQuery = require("jquery")(window);
24      // See ticket trac-14549 for more info.
25      module.exports = global.document ?
26        factory( global, noGlobal: true ) :
27        function( w ) {
```

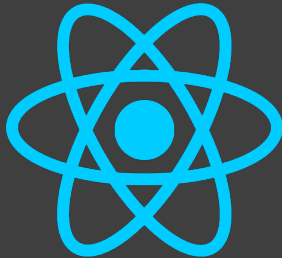

V8 JavaScript 엔진



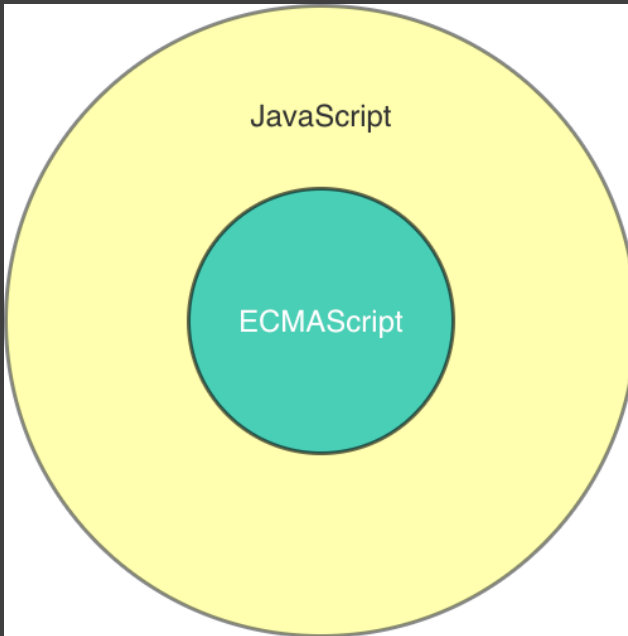
Node.js



SPA 프레임워크



JavaScript, ECMAScript



JavaScript의 특징

인터프리터 언어 vs 컴파일 언어

컴파일러 언어	인터프리터 언어
코드가 실행되기 전 단계인 컴파일 타임에 소스코드 전체를 한번에 머신 코드로 변환한 후 실행	코드가 실행되는 단계인 런타임에 문 단위로 한 줄씩 중간 코드인 바이트코드로 변환한 후 실행
실행 파일을 생성	실행 파일을 생성하지 않음
컴파일 단계와 실행 단계가 분리되어 있음. 명시적인 컴파일 단계를 거치고, 명시적으로 파일을 실행	인터프리터 단계와 실행 단계가 분리되어 있지 않음. 인터프리터는 한 줄씩 바이트코드로 변환하고 즉시 실행
실행에 앞서 컴파일은 단 한번 수행됨	코드가 실행될 때마다 인터프리터 과정이 반복 수행됨
컴파일과 실행 단계가 분리되어 있으므로 코드 실행 속도가 빠름	인터프리터 단계와 실행 단계가 분리되어 있지 않고 반복 수행되므로 코드 실행 속도가 비교적 느림

→ 클래스 기반 객체지향 언어보다 효율적이면서 강력한
프로토타입 기반의 객체지향 언어

ECMAScript 지원 현황

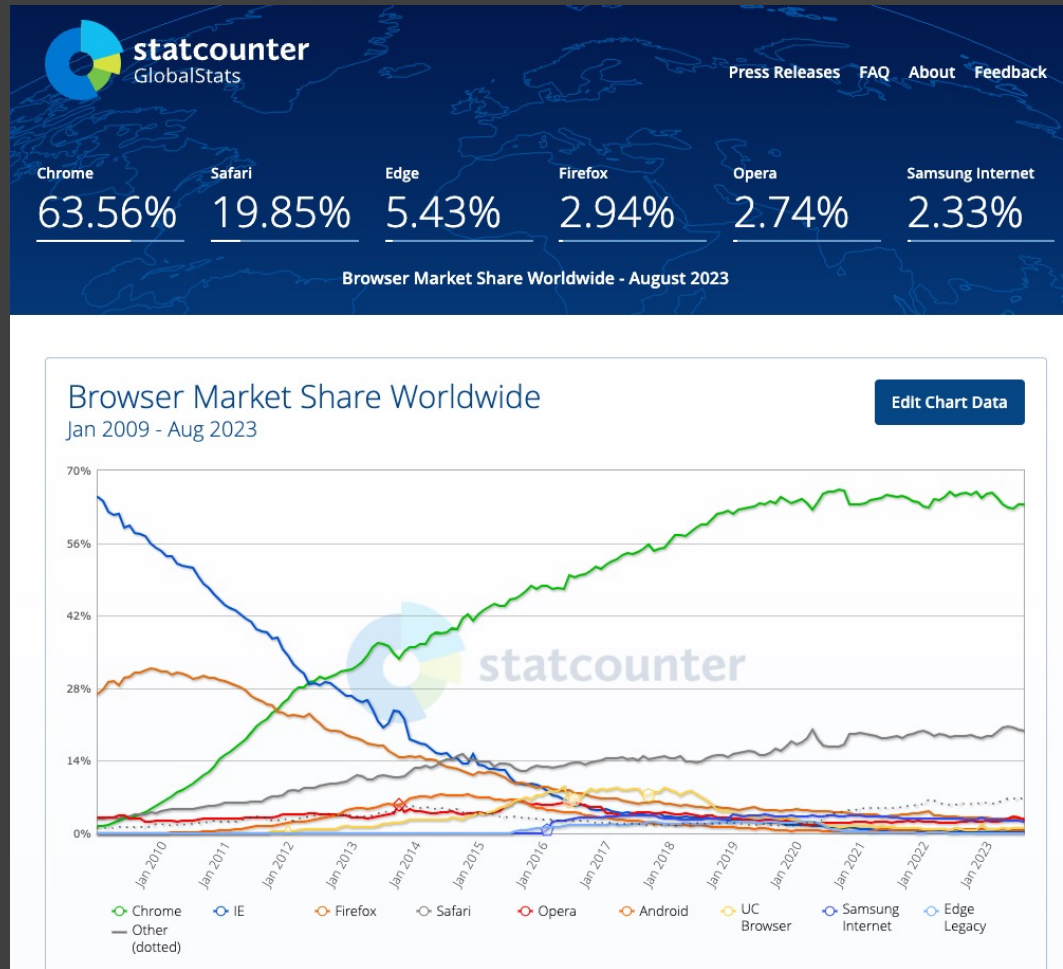
[illegible]

<https://kangax.github.io/compat-table/es6/>

자바스크립트 실행 환경



웹 브라우저



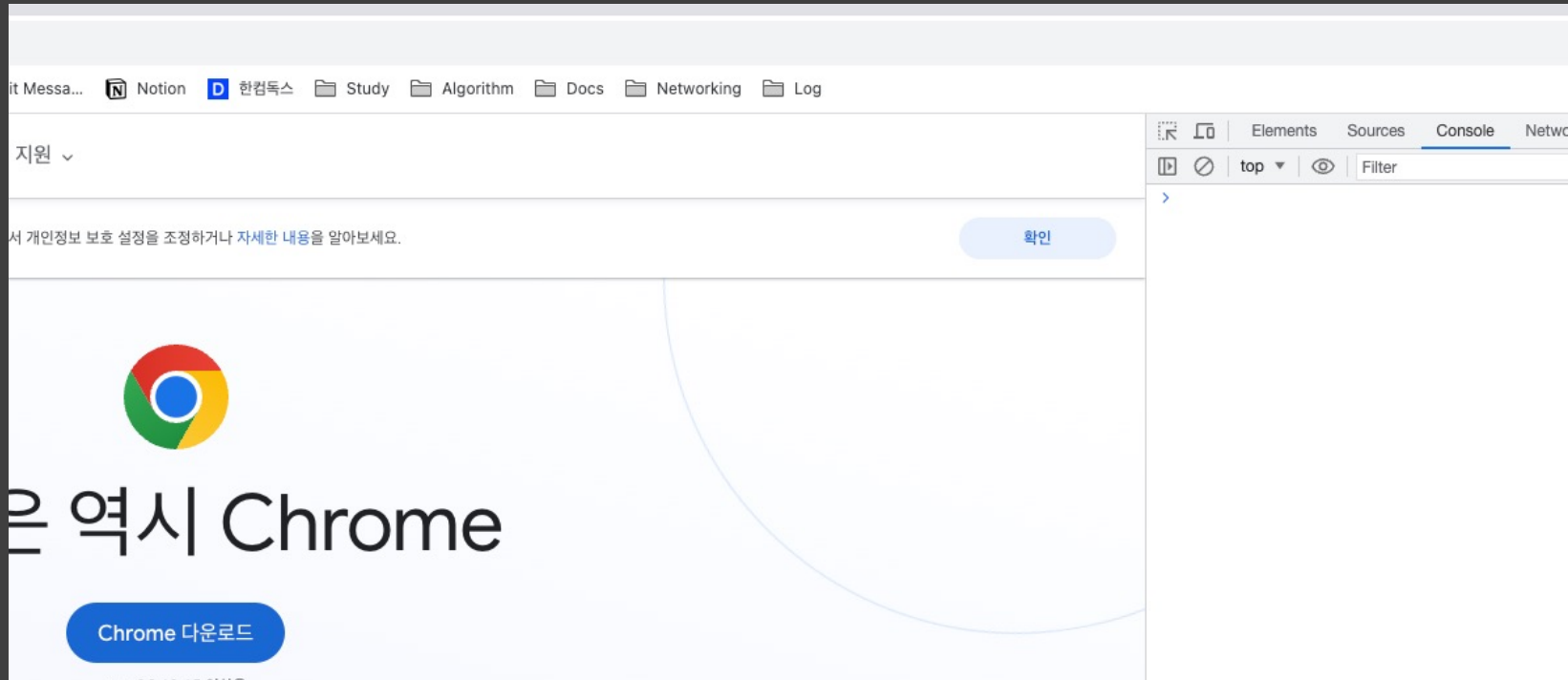
개발자 도구

The image shows the Google Chrome Developer Tools interface. The main window displays the Google homepage. Two callout boxes provide details about the developer tools panels:

운영체제	단축키
윈도우	F12 or Ctrl + Shift + I
macOS	command + option + I

패널	설명
Elements	로딩된 웹사이트의 DOM과 CSS를 편집해서 렌더링된 뷰를 볼 수 있음. 단, 편집한 내용이 저장되지 않음. 웹페이지가 의도된 대로 렌더링되지 않았다면 이 패널을 확인해 유용한 힌트를 얻을 수 있음
Console	로딩된 웹사이트의 에러를 확인하거나 JavaScript 소스코드에 작성한 console.log 메서드의 실행 결과를 확인할 수 있음
Source	로딩된 웹사이트의 JavaScript 코드를 디버깅할 수 있음
Network	로딩된 웹페이지에 관련된 네트워크 요청 정보와 성능을 확인할 수 있음
Application	웹 스토리지, 세션, 쿠키를 확인하고 관리할 수 있음

콘솔



브라우저에서 자바스크립트 실행

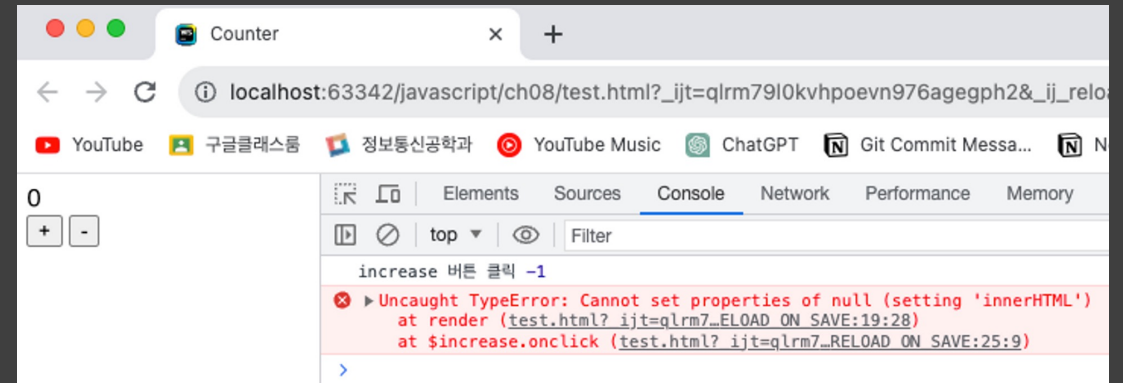
ex03-01.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>Counter</title>
</head>
<body>
  <div id="counter">0</div>
  <button id="increase">+</button>
  <button id="decrease">-</button>
<script>
  // 에러를 발생시키는 코드: 선택지는 'counter-x'가 아니라 'counter'를 지정해야함
  const $counter = document.getElementById("counter-x");
  const $increase = document.getElementById("increase");
  const $decrease = document.getElementById("decrease");

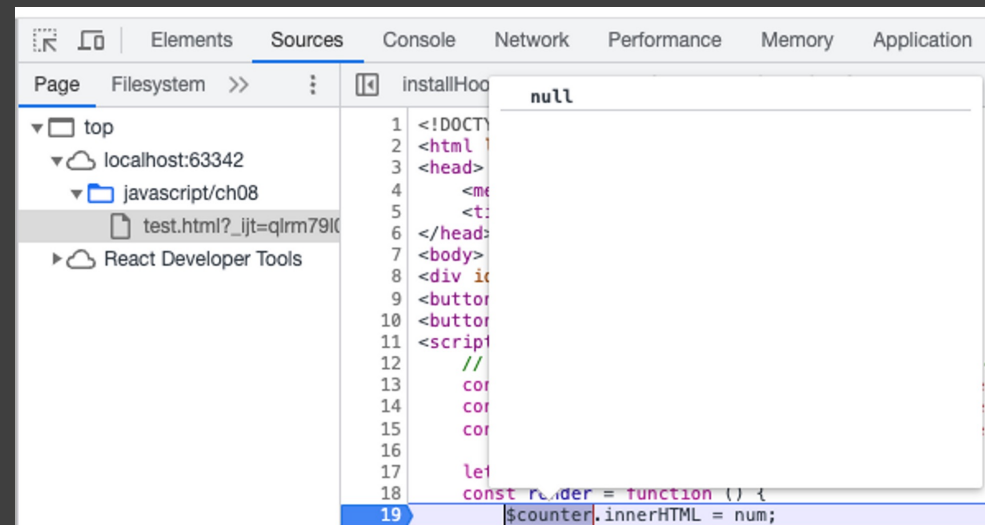
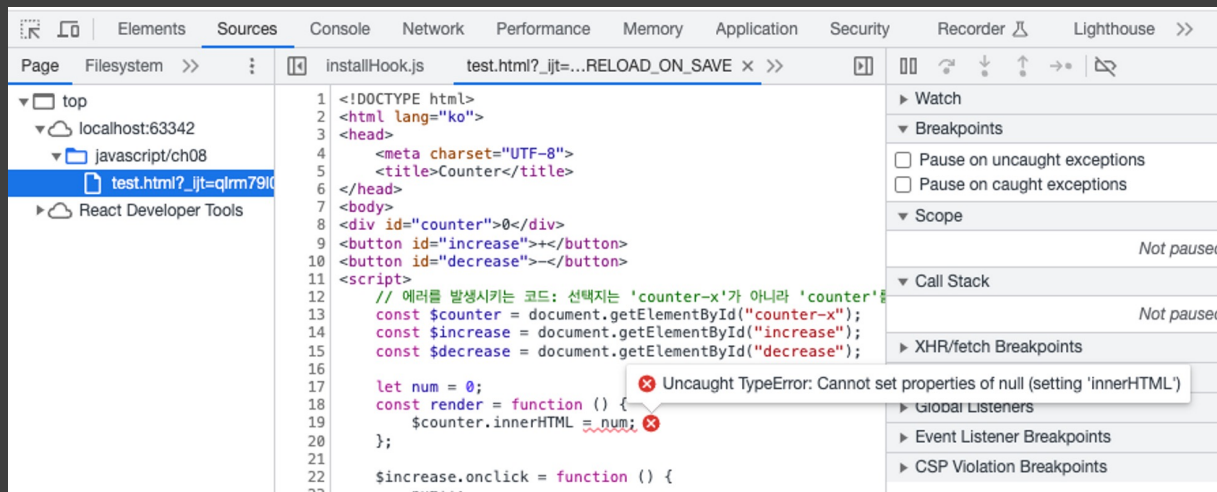
  let num = 0;
  const render = function () {
    $counter.innerHTML = num;
  };

  $increase.onclick = function () {
    num++;
    console.log("increase 버튼 클릭", num);
    render();
  };

  $decrease.onclick = function () {
    num--;
    console.log("decrease 버튼 클릭", num);
    render();
  };
</script>
</body>
</html>
```



디버깅



Node.js

Node.js®는 **Chrome V8 JavaScript 엔진**으로 빌드된 JavaScript 런타임입니다.

다운로드 - macOS

18.17.1 LTS

안정적, 신뢰도 높음

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

20.6.1 현재 버전

최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

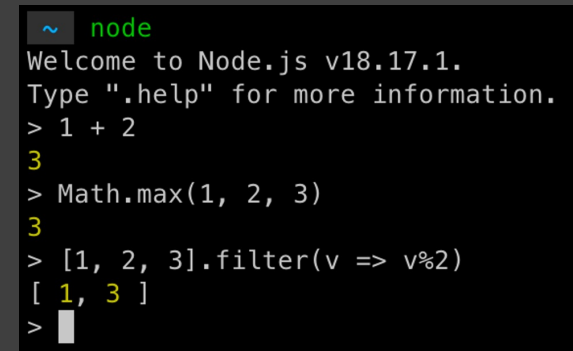
LTS 일정은 [여기서](#) 확인하세요.

Node.js



A terminal window titled "Yu-Jaeyoung — /Users/yujaeyeong" showing the results of version checks for Node.js and npm. The window includes system status bars at the top (CPU 100%, battery 11%, memory 5.1 GB) and a list of commands with their outputs and timestamps.

```
~ node -v ok | 21:27:40
v18.17.1
~ npm -v ok | 21:27:42
9.6.7
~ ok | 21:27:45
```



A terminal window showing the Node.js CLI interface. It displays the welcome message and several example commands and their outputs.

```
~ node
Welcome to Node.js v18.17.1.
Type ".help" for more information.
> 1 + 2
3
> Math.max(1, 2, 3)
3
> [1, 2, 3].filter(v => v%2)
[ 1, 3 ]
>
```

번외. Pnpm ?

