



# 자바스크립트 개발 환경과 실행 방법

모든 브라우저는 자바스크립트를 해석하고 실행할 수 있는 자바스크립트 엔진을 내장하고 있다. 브라우저뿐만 아니라 Node.js도 자바스크립트 엔진을 내장하고 있다. 따라서 자바스크립트는 브라우저와 Node.js 환경에서 실행할 수 있다. 기본적으로 브라우저에서 동작하는 코드는 Node.js 환경에서도 동작한다.

그런데 브라우저와 Node.js는 존재 목적이 다르다. 브라우저는 HTML, CSS, 자바스크립트를 실행하여 웹 페이지를 화면에 렌더링하는 것이 주된 목적이지만, Node.js는 서버 개발 환경을 제공하는 것이 주된 목적이다. 따라서 브라우저와 Node.js 모두 자바스크립트의 코어인 ECMAScript를 실행할 수 있지만 브라우저와 Node.js에서 ECMAScript 이외에 추가적으로 제공하는 기능은 호환되지 않는다.

예를 들어 브라우저는 HTML 요소를 선택하거나 조작하는 기능들의 집합인 DOM API를 기본적으로 제공한다. 하지만 서버 개발 환경을 제공하는 것이 주 목적인 Node.js는 클라이언트 사이드 Web API인 DOM API를 제공하지 않는다. 서버에서는 HTML 요소를 다룰 일이 없기 때문이다. 반대로 Node.js에서는 파일을 생성하고 수정할 수 있는 File 시스템을 기본 제공하지만 브라우저는 이를 지원하지 않는다. (Web API인 File API FileReader 객체를 사용해 사용자가 지정한 파일을 읽어 들이는 것은 가능하다.) 브라우저는 사용자 컴퓨터에서 동작한다. 만약 브라우저를 통해 사용자 컴퓨터에 파일을 생성하거나 기존 로컬 파일을 수정할 수 있다면 사용자 컴퓨터는 악성 코드에 노출되기 쉽기 때문에 보안 상 이유로 이를 금지하고 있다.

이처럼 브라우저는 ECMAScript와 DOM, BOM, Canvas, XMLHttpRequest, Fetch, requestAnimationFrame, SVG, Web Storage, Web Component, Web worker와 같은 클라이언트 사이드 Web API를 지원한다. Node.js는 클라이언트 사이드 Web API는 지원하지 않고 ECMAScript와 Node.js 고유의 API를 지원한다.

이를 염두에 두고 자바스크립트 개발 환경을 구축하고 자바스크립트를 실행하는 방법에 대해 살펴보자. 웹 브라우저에서 실행하는 방법과 Node.js 환경에서 실행하는 방법 그리고 코

드 에디터인 비주얼 스튜디오 코드(Visual Studio Code)에서 실행하는 방법에 대해 살펴볼 것이다.

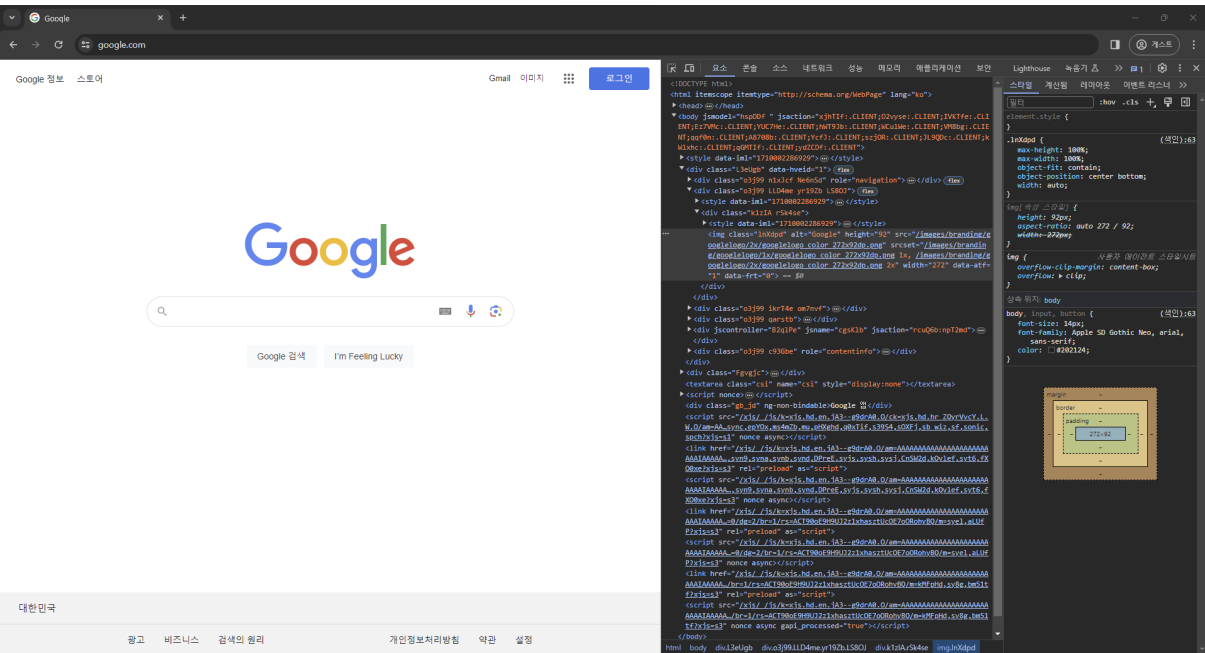
## 웹 브라우저

다양한 웹 브라우저가 있지만 구글 크롬(Chrome) 브라우저를 사용하기로 하자. 크롬 브라우저의 V8 자바스크립트 엔진은 Node.js에서도 사용하고 있다. 만약 크롬 브라우저가 설치되어 있지 않다면 아래의 웹사이트를 방문하여 먼저 최신 버전을 설치하기 바란다.

## 개발자 도구

크롬 브라우저가 제공하는 개발자 도구(DevTools)은 자바스크립트 개발에 필수적인 강력한 도구이다. 개발자 도구는 브라우저에 기본 내장되어 있으므로 별도의 설치가 필요없다. 개발자 도구는 아래의 단축키로 오픈할 수 있다.

운영 체제	단축키
Windows	F12 또는 Ctrl + Shift + I
macOS	command ⌘ + option ⌥ + I



개발자 도구는 웹 개발에 유용한 다양한 기능을 제공한다. 자주 사용하는 개발자 도구 기능은 아래와 같다.

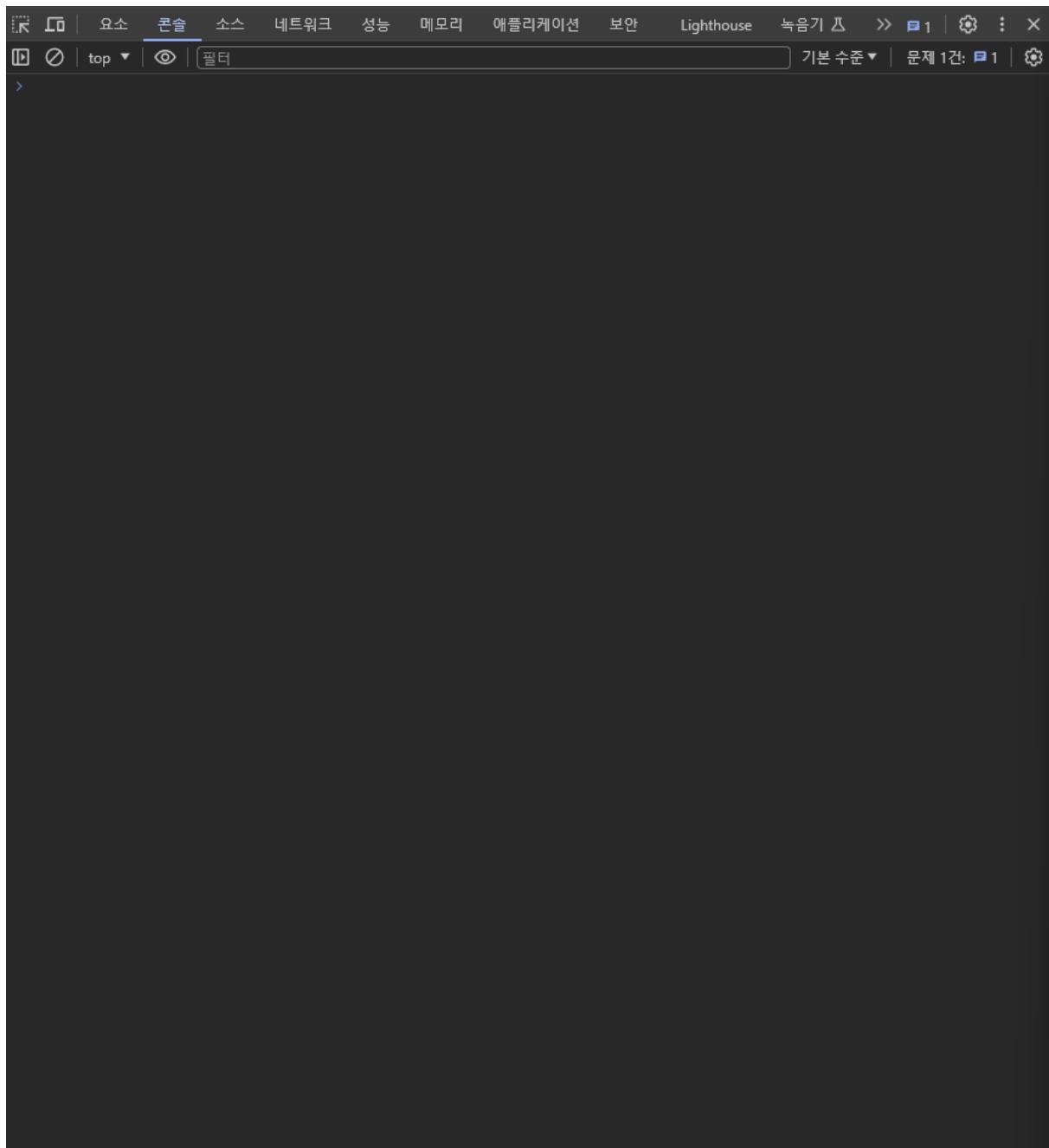
패널	설명
Elements	로딩된 웹 페이지의 DOM과 CSS를 편집하여 렌더링된 뷰를 확인해 볼 수 있다. 단, 편집한 내용이 저장되지는 않는다. 웹 페이지가 의도된 대로 렌더링되지 않았다면 이 패널을 확인하여 유용한 힌트를 얻을 수 있다.
Console	로딩된 웹 페이지의 에러를 확인하거나 자바스크립트 소스코드에 포함시킨 <code>console.log</code> 메소드의 결과를 확인해 볼 수 있다.
Sources	로딩된 웹 페이지의 자바스크립트 코드를 디버깅할 수 있다.
Network	로딩된 웹 페이지에 관련한 네트워크 요청(request) 정보와 퍼포먼스를 확인할 수 있다.
Application	웹 스토리지, 세션, 쿠키를 확인하고 관리할 수 있다.

## 콘솔

개발자 도구의 Console(콘솔) 패널은 자바스크립트 코드에서 에러가 발생하여 애플리케이션이 정상적으로 동작하지 않을 때 가장 우선적으로 살펴보아야 할 곳이다. 구현 단계에서는 에러가 빈번하게 발생하므로 항상 콘솔을 열어둔 상태에서 개발을 진행하는 것이 좋다. 콘솔을 열어두지 않으면 에러가 발생했는지조차 알 수 없는 경우가 있다.

에러가 발생한 경우가 아니더라도 콘솔은 매우 유용하다. 구현 단계에서 디버깅을 실행하는 것보다 간편하게 값을 확인하며 개발을 진행하기 위해 `console.log` 함수를 사용하는 경우가 많다. `console.log(...)`는 소괄호 안에 있는 코드의 실행 결과를 콘솔에 출력하는 함수이다. 이 함수를 사용해 확인하고 싶은 값을 콘솔에 출력해 확인할 수 있다.

콘솔은 자바스크립트 코드를 직접 입력하여 그 결과를 확인할 수 있는 REPL(Read Eval Print Loop: 입력 수행 출력 반복) 환경으로 사용할 수도 있다. 개발자 도구의 Console 패널을 클릭하면 아래와 같이 프롬프트(>)가 깜박이는 것을 확인할 수 있다.

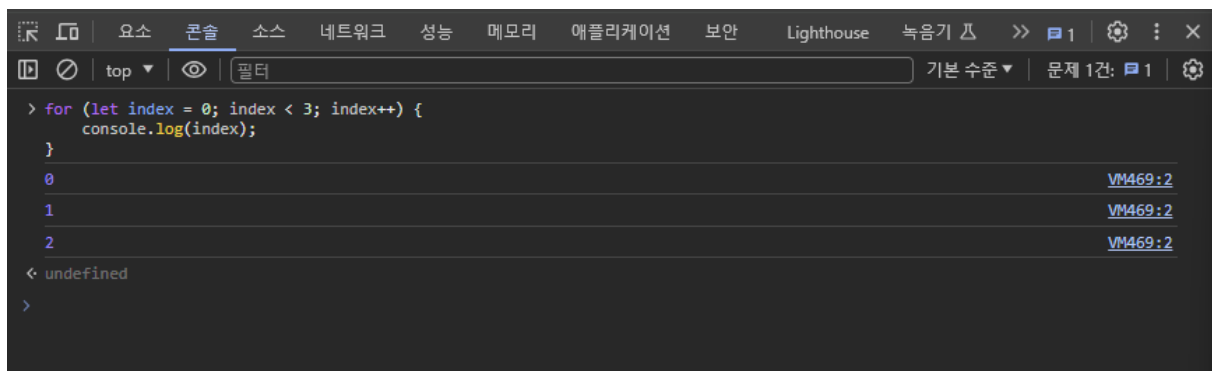


프롬프트에 자바스크립트 코드를 입력하면 다음 줄에 실행 결과가 표시된다. 엔터 키를 입력하면 다음 프롬프트로 이동한다.



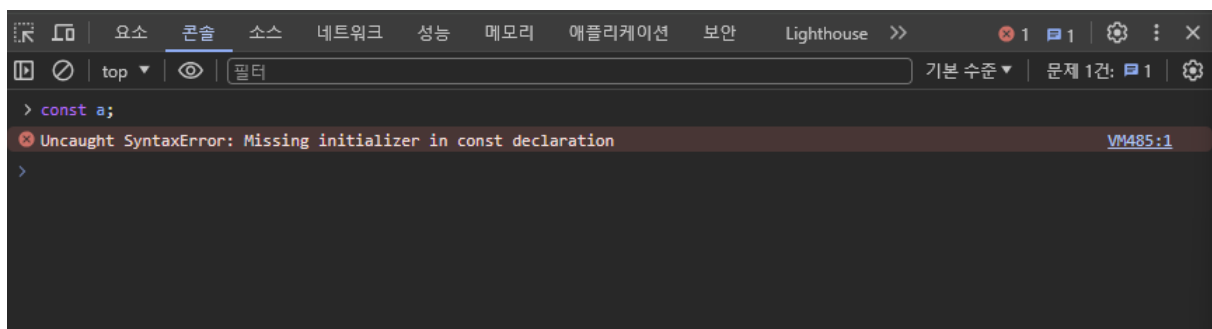
```
> 1 + 2
< 3
> Math.max(1, 2, 3)
< 3
> [1, 2, 3].filter(v => v % 2)
< ▶ (2) [1, 3]
>
```

여러 줄로 이루어진 코드를 실행하는 경우, 줄바꿈이 필요할 때 Shift 키를 누른 상태에서 엔터 키를 누른다.



```
> for (let index = 0; index < 3; index++) {
  console.log(index);
}
0
1
2
< undefined
>
```

자바스크립트 코드 실행 중에 에러가 발생하면 에러의 내용이 콘솔에 출력된다.



```
> const a;
Uncaught SyntaxError: Missing initializer in const declaration
>
```

## HTML에 포함된 자바스크립트를 브라우저에서 실행

브라우저는 HTML 파일을 로드하면 script 태그에 포함된 자바스크립트 코드를 실행한다. 자바스크립트 코드 내에 console.log 함수가 있다면 콘솔에 실행 결과가 출력될 것이다. 아래와 같이 자바스크립트가 포함된 HTML 파일을 생성하고 브라우저로 열어보자.

```
<!DOCTYPE html>
<html>
<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
    const $counter = document.getElementById('counter-x');
    const $increase = document.getElementById('increase');
    const $decrease = document.getElementById('decrease');

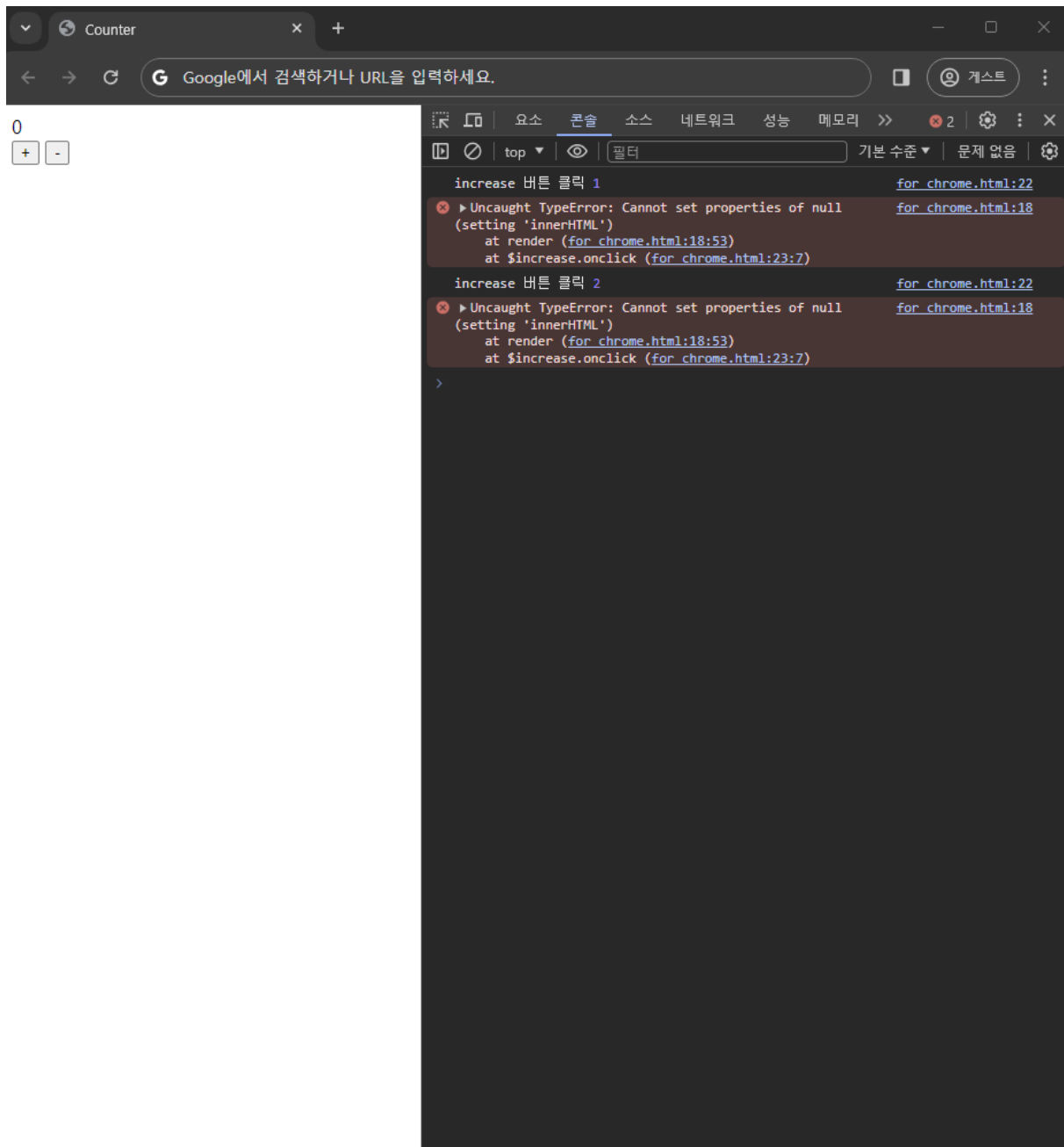
    let num = 0;
    const render = function () { $counter.innerHTML = num; };

    $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>
```

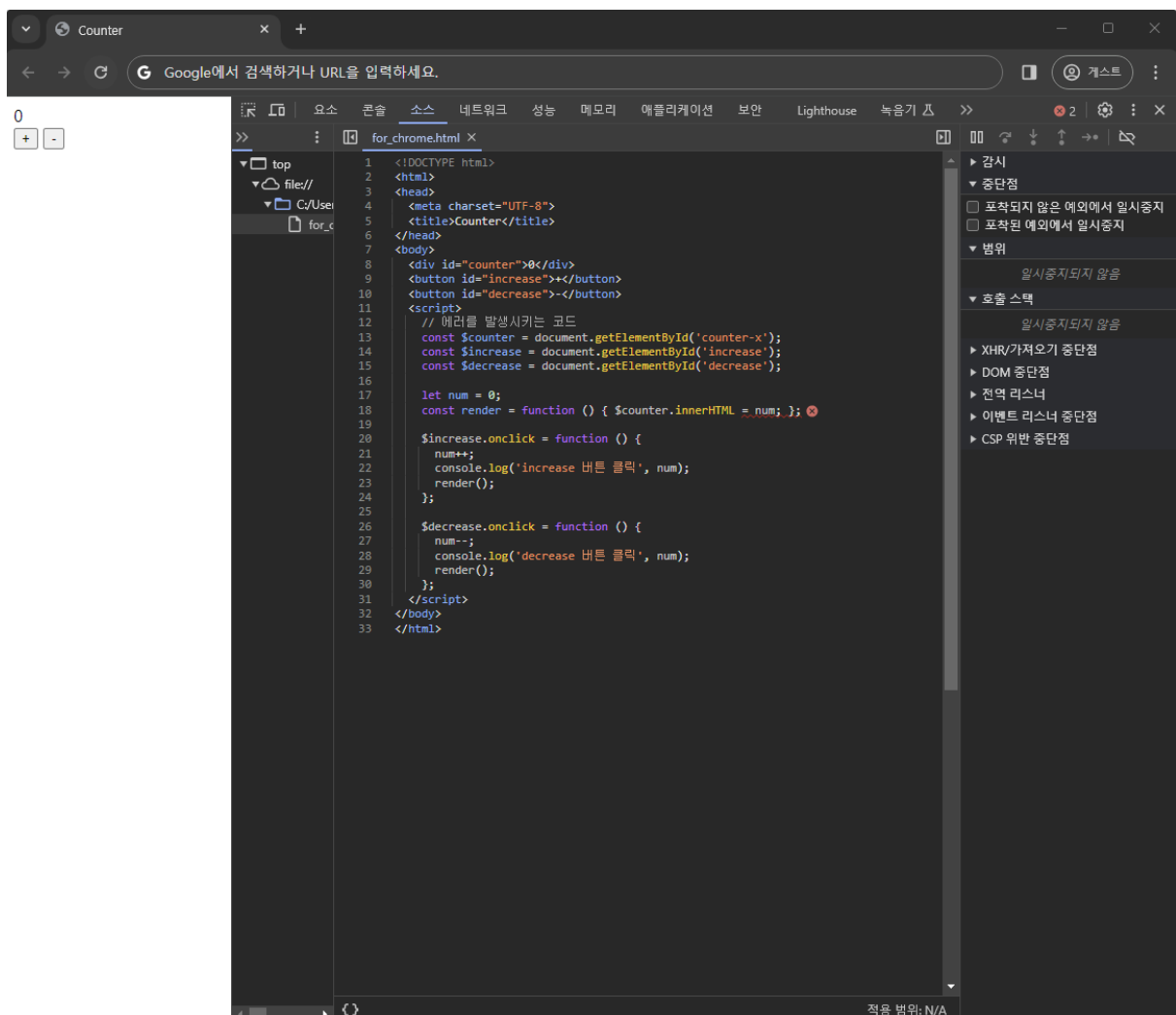
'+' 또는 '-' 버튼을 클릭하면 에러가 발생한다. 만약 개발자도구의 콘솔을 열어둔 상태가 아니라면 에러가 발생한 것을 알아차리기 어렵다. 에러를 확인하기 위해 개발자 도구의 콘솔을 열어보자.



에러가 발생하기는 했으나 HTML 파일에 포함된 자바스크립트가 실행된 것은 확인하였다. 다음은 디버깅에 대해 살펴보자.

## 디버깅

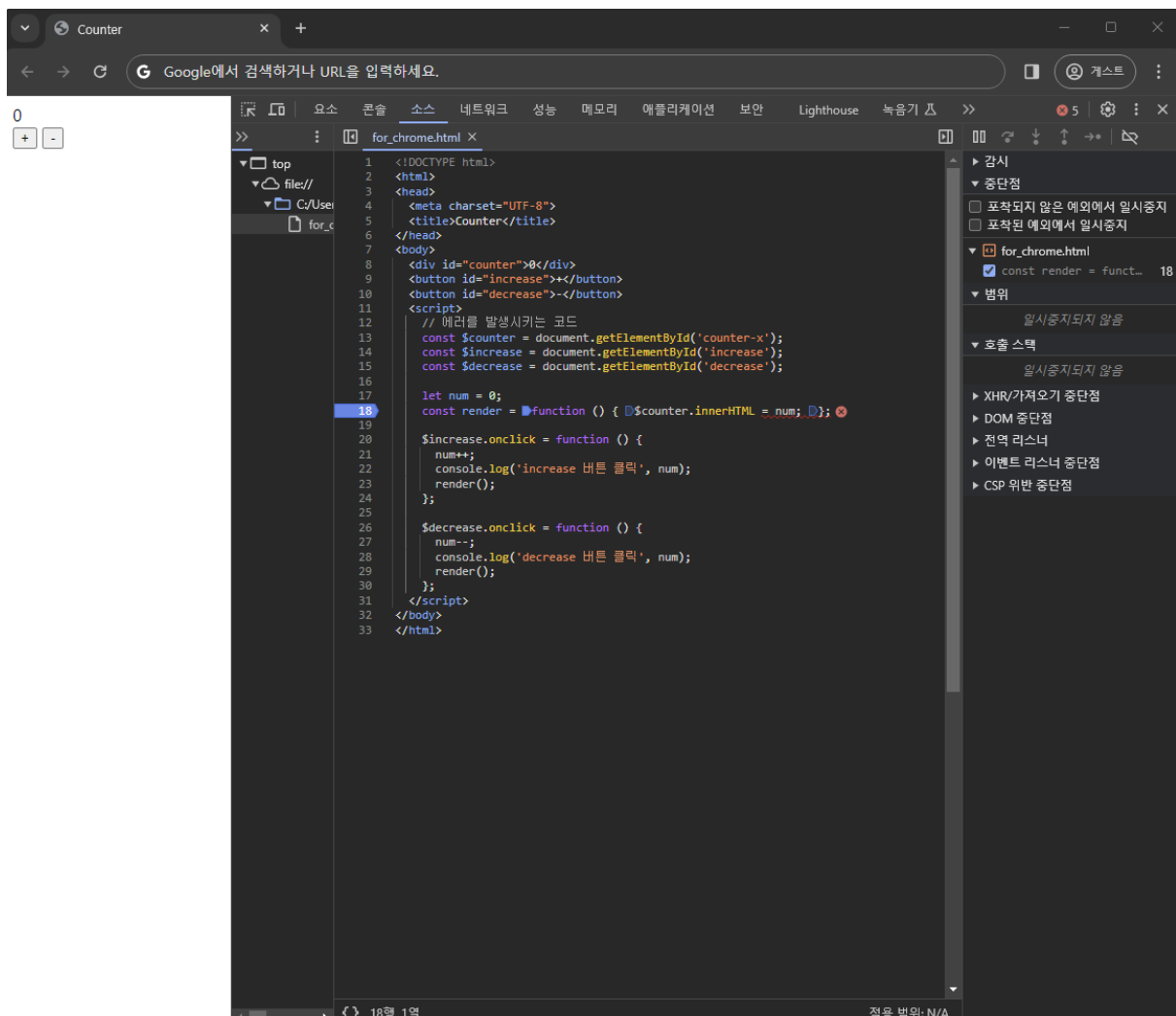
에러 정보의 오른쪽에 에러 발생 위치를 나타내는 링크를 클릭해보자. 자바스크립트 코드를 디버깅을 할 수 있는 Sources 패널로 이동할 것이다.



에러가 발생한 위치에 빨간 밑줄이 표시되고 그 위에 마우스를 올려 보면 Uncaught  
TypeError: Cannot set property 'innerHTML' of null이라는 에러 정보가 표시된다. 이  
에러는 innerHTML 프로퍼티에 값을 할당하기 위해 객체 \$conter를 참조했으나 그 객체가



null이기 때문에 발생한 에러다. \$conter의 값이 null인지 확인해보고 null이라면 그 이유를 알아내어 에러 발생 원인을 제거해 보자. 에러가 발생한 코드 왼쪽의 라인 번호를 클릭하여 브레이크 포인트(중단점)를 걸고 다시 버튼을 클릭하면 아래와 같이 디버깅 모드로 들어가게 된다.



\$conter의 값이 null인 것을 확인했다. 그 원인은 13 라인에서 \$conter에 값을 할당할 때, HTML 요소의 아이디를 'counter-x'로 잘못 지정한 탓이다. 다시 소스코드로 돌아가 HTML 요소의 아이디를 'counter'로 정확히 지정하면 에러가 제거될 것이다.

콘솔과 디버깅에 대한 보다 자세한 내용은 구글의 [Tools for Web Developers: 콘솔 사용](#)과 [Tools for Web Developers: Chrome DevTools에서 자바스크립트 디버깅 시작하기](#)를 참고하기 바란다.

## Node.js

클라이언트 사이드, 즉 웹 브라우저에서 동작하는 간단한 웹 애플리케이션은 브라우저만으로도 개발을 할 수 있다. 하지만 프로젝트의 규모가 커짐에 따라 React, jQuery와 같은 외부 라이브러리를 도입하거나 Babel, Webpack, ESLint 등 여러 가지 도구를 사용해야 할 필요가 있다. 이때 Node.js와 npm이 필요하다.

### Node.js와 npm 소개

2009년 라이언 달(Ryan Dahl)이 발표한 Node.js는 Chrome V8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임 환경(Runtime Environment)이다. 간단히 말해 브라우저에서만 동작하던 자바스크립트를 브라우저 이외의 환경에서 동작시킬 수 있는 자바스크립트 실행 환경이 Node.js이다.

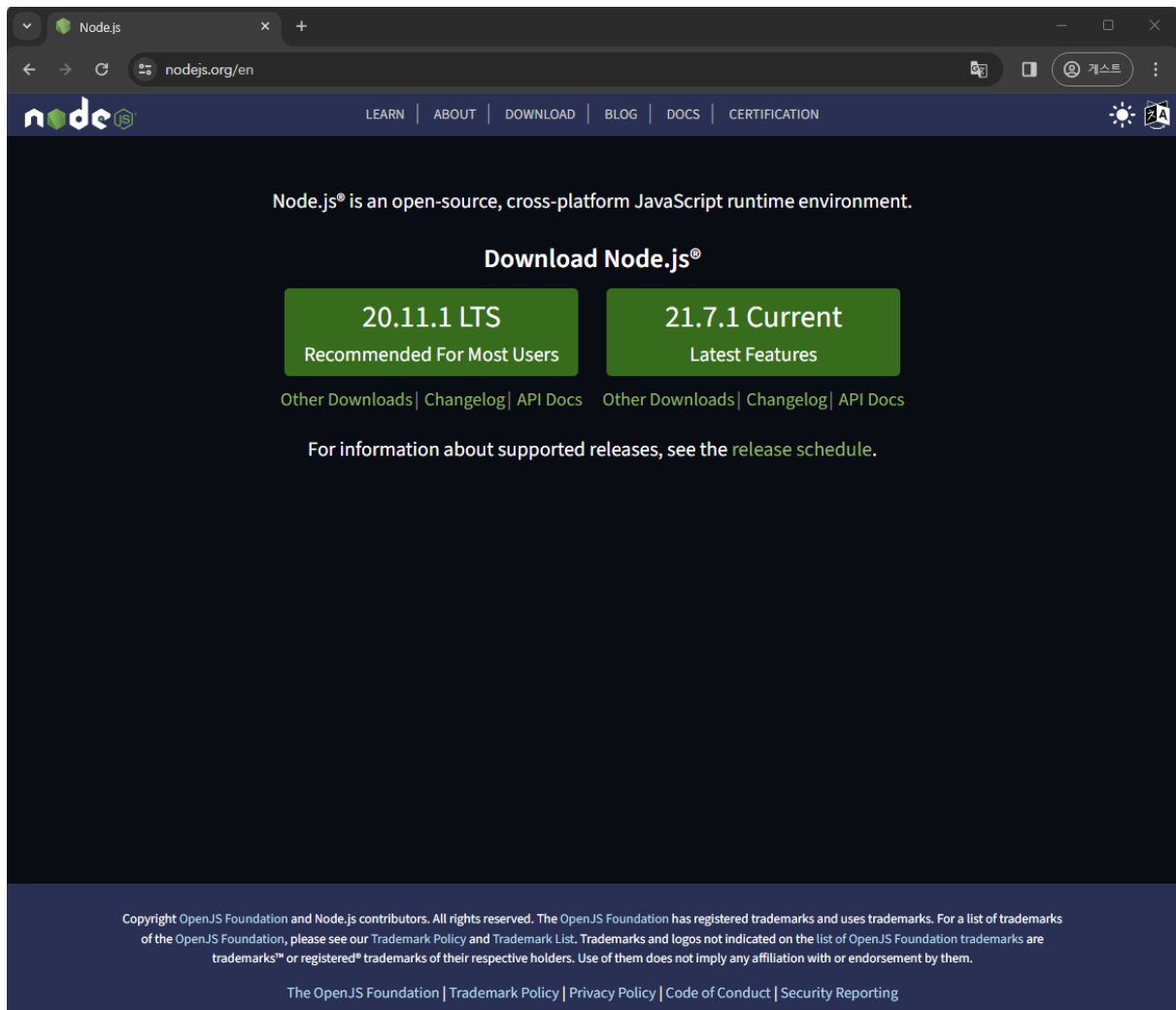
Node.js는 주로 서버 사이드 애플리케이션 개발에 사용되며 이에 필요한 모듈, 파일 시스템, HTTP 등 빌트인 API를 제공한다. Node.js는 데이터를 실시간 처리하여 빈번한 I/O가 발생하는 SPA(Single Page Application)에 적합하다. 하지만 CPU 사용률이 높은 애플리케이션에는 권장하지 않는다.

Node.js는 백엔드 영역의 서버 애플리케이션 개발뿐만 아니라 프론트엔드 영역의 다양한 도구나 라이브러리도 Node.js 환경에서 동작한다. 따라서 Node.js는 프론트엔드 모던 자바스크립트 개발에 필수적인 환경이라고 할 수 있다.

npm(node package manager)은 자바스크립트 패키지 매니저이다. Node.js에서 사용할 수 있는 모듈들을 패키지화하여 모아둔 저장소 역할과 패키지 설치 및 관리를 위한 CLI(Command line interface)를 제공한다. 자신이 작성한 패키지를 공개할 수도 있고 필요한 패키지를 검색하여 재사용할 수도 있다.

### Node.js 설치

Node.js를 설치하기 위해 Node.js의 웹사이트(<http://nodejs.org>)에 접속해 보자.



Node.js 웹사이트에 접속하면 두 개의 다운로드 버튼이 보이는데 왼쪽은 LTS 버전, 오른쪽은 Current 버전을 다운로드할 수 있다. LTS(Long Term Supported) 버전은 장기적으로 안정된 지원이 보장된다. Current 버전은 최신 기능을 제공하지만 업데이트가 발생하고 있는 버전으로 안정적이지 않을 수 있다. 따라서 LTS 버전을 다운로드하도록 하자.

이때 `npm`도 동시에 설치된다. Node.js는 아래의 디렉터리에 설치된다. 버전에 따라 설치 장소는 바뀔 수 있다.

- Windows : C:\Program Files\nodejs\node.exe
- Mac : /usr/local/bin/node

설치가 완료되면 터미널(윈도우에서는 명령 프롬프트)에서 Node.js와 npm의 버전을 출력하여 정상적으로 설치되었는지 확인한다.

```
$ node -v
v20.11.1
$ npm -v
10.2.5
```

npm은 Node.js에 포함되어 있어 Node.js 설치 시 자동 설치된다. 따라서 별도의 설치가 필요 없다. 하지만 Node.js보다 자주 업데이트되므로 최신 버전이 아닐 수 있다. 최신 버전으로 npm을 업데이트하도록 하자.

```
$ npm install -g npm@latest
$ npm -v
10.5.0
```

## Node.js REPL

REPL(Read Eval Print Loop)은 Node.js는 물론 대부분의 언어(Java, Python 등)가 제공하는 가상환경으로 간단한 코드를 직접 실행해 결과를 확인해 볼 수 있다. 터미널(윈도우에서는 명령 프롬프트)에 다음과 같은 명령어를 실행해 보자.

```
$ node
```

프롬프트가 >로 변경되면 자바스크립트 코드를 실행해 볼 수 있다.

```
> 1 + 2
3
> Math.max(1, 2, 3)
3
> [1, 2, 3].filter(v => v % 2)
[ 1, 3 ]
```

자바스크립트 파일을 실행하려면 node 명령어 뒤에 파일명을 입력한다. 파일 확장자 .js는 생략할 수 있다.

```
$ node index.js
```

CTRL + C 키를 두번 입력하면 Node.js REPL을 종료시킨다. Node.js REPL에 관한 더 자세한 내용은 Node.js Documentation : REPL을 참조하기 바란다.