자바스크립트 환경

브라우저 환경

DOM(Document Object Model)에 접근하여 HTML과 CSS를 조작하며 주로 웹 페이지의 동적 요소를 조작하고, 사용자 인터페이스와 상호작용을 관리하는 데 사용됩니다. 브라우저 특유의 API와 DOM에 집중하여, 웹 페이지 내에서 클라이언트 측 스크립트를 실행합니다.

런타임 환경

서버 측 어플리케이션 개발에 초점을 맞춘 자바스크립트 런타임 환경에서의 JavaScript는 파일 시스템의 접근, HTTP 요청 처리 및 데이터베이스와의 상호작용과 같은 백엔드 기능을 수행합니다. 브라우저에 구애받지 않는, 더 넓은 범위의 서버 사이드 작업을 지원합니다.

브라우저 환경과 런타임 환경의 차이점

분류	브라우저 환경	JS 런타임 환경
용도	*사용자 인터페이스와 관련된 작업*	*서버 측 애플리케이션 개발*
실행 환경	*DOM 조작 (Document Object Model)* *이벤트 처리*	*파일 시스템 접근* *네트워크 요청 처리*
모듈 시스템	*ES6 모듈 지원*	*CommonJS 모듈 시스템 사용*
API와 라이브러리	*WebAPI (`window`, `document`)*	*NodeJS 모듈 (`fs`, `http`)*

▶ 기본 분류 설명

> - 용도
> 브라우저 환경은 웹 페이지의 사용자 인터페이스와 상호작용을 만들고 관리하는 데 집중합니다.
> JS 런타임 환경은 주로 서버 측에서 데이터 처리, 애플리케이션 로직 실행, 서버 관리와 같은 백엔드 작업에 사용됩니다.

> - 실행 환경 및 모듈 시스템
> 브라우저 환경에서의 개발은 주로 DOM의 조작과 이벤트 처리에 초점을 맞춥니다.
> 런타임 환경(예: Node.js)에서는 파일 시스템에 접근하거나 네트워크 요청을 처리하는 일이 중요합니다.
> 자바스크립트의 런타임 종류는 NodeJS, Bun, Deno 등이 있습니다.

> - 모듈 시스템과 API 브라우저 환경에서의 모듈 시스템은 ES6 모듈(import/export)을 주로 사용합니다.
> 런타임 환경에서는 CommonJS 스타일(`require` 및 `module.exports`)의 모듈 시스템이 일반적입니다.
> 브라우저는 WebAPI(예: `window`, `document`)를 제공하여 웹 페이지와 상호작용 합니다.
> 런타임 환경은 Node.js의 기본 모듈(예: `fs` 파일 시스템, `http` 네트워크 요청)을 사용하여 서버 측 기능을 수행합니다.

세부 기능 비교

기능 / API	브라우저 환경	JS 런타임 환경
전역 객체	`globalThis` (`window`)	`globalThis` (`global`)
스크립트 로드	`<script src="">`	`require`
HTTP 요청	`fetch`	라이브러리 (예: `node-fetch`)
클라이언트 저장소	Web Storage API (`localStorage`, `sessionStorage`, `cookie`, `IndexedDB`)	직접적인 Web Storage API는 없음 (외부 DB나 파일 시스템을 통한 데이터 관리 가능)
이벤트 처리	DOM Event	없음
DOM 접근	`document`	없음
파일 시스템 접근	`<input type="file">`를 사용한 사용자의 로컬 파일 접근	`fs` 모듈을 사용한 파일 I/O 처리

▶ 세부 기능과 API 설명

> 전역 객체
>
> - 전역 객체는 모든 전역 변수와 함수, JavaScript 내장 객체를 포함하는 최상위 객체입니다.
> - 브라우저 환경에서는 globalThis(window)가 전역 객체 역할을 하며, Node.js 같은 서버 측 런타임 환경에서는 globalThis(global)이 그 역할을 합니다.
> - 이 객체는 스크립트가 실행되는 환경에 상관없이 어디서든 접근 가능한 범위와 속성을 제공합니다.

> - 브라우저 환경에서 사용되는 globalThis와 window는 JavaScript에서 전역 객체에 접근하는 두 가지 다른 방법이지만, 주요 차이점이 있습니다.

	`window`	`globalThis`
1	웹 브라우저 환경에서 사용되며, JavaScript를 실행할 때 전역 객체로 사용됩니다.	ES2020에서 도입된 표준으로, 모든 JavaScript 환경(브라우저, Node.js, Web Workers 등)에서 일관된 전역 객체에 접근할 수 있습니다.
2	웹 브라우저에 특화된 전역 객체로, 브라우저 창과 관련된 속성과 메서드(예: DOM 조작, 타이머 함수, 콘솔 접근 등)를 제공합니다.	JavaScript가 실행되는 모든 환경에서 동일하게 작동하는 전역 객체에 대한 참조를 제공하며, 코드의 호환성과 이식성을 높이는 데 유용합니다.
3	브라우저 환경에서 사용하면 브라우저 창에 관련된 기능에 접근할 수 있으며, 특정 브라우저 API와 상호 작용하는 데 사용됩니다.	브라우저, Node.js, Web Workers 등 다양한 환경에서 같은 코드로 전역 변수나 객체에 접근할 수 있습니다.

> 스크립트 로드
>
> - 웹 페이지와 런타임 환경에서 스크립트를 로드하는 방식은 서로 다릅니다.
> - 브라우저 환경에서는 `<script src="">` 태그를 사용하여 HTML 문서에 외부 자바스크립트 파일을 포함하고 브라우저는 지정된 경로의 스크립트 파일을 로드하고 실행합니다.
> - Node.js와 같은 런타임 환경에서는 require 함수를 사용하여 모듈 또는 라이브러리를 로드합니다. require 함수는 지정된 경로의 모듈을 로드하고 그 내용을 반환하여 서버 측 스크립트에서 활용할 수 있게 해줍니다.

> HTTP 요청
>
> - 웹 애플리케이션은 HTTP 요청을 통해 서버와 데이터를 교환하며, 사용자에게 최신 정보를 제공하거나 사용자의 입력을 서버에 전달합니다.
> - 서버로부터 데이터를 받아와 실시간으로 웹 페이지를 업데이트함으로써, 사용자에게 동적인 콘텐츠를 제공하고 상호작용적인 경험을 제공합니다.
> - 외부 API에 HTTP 요청을 보내 다양한 기능과 정보(날씨 정보, 지도, 외부 데이터베이스 접근 등)를 웹 애플리케이션에 통합합니다.

> 클라이언트 저장소
>
> - 로그인 상태 유지, 언어 설정, 테마 선택 등 사용자의 개인 설정을 브라우저에 저장합니다.
> - 필요한 정보를 서버에 요청하지 않고 로컬에서 빠르게 불러와 웹 페이지 로딩 시간을 단축합니다.
> - 인터넷이 끊겨도 필요한 정보를 로컬에서 불러와 웹 사이트 기능 일부를 계속 사용할 수 있게 합니다.

> DOM(Document Object Model)
>
> - 웹 페이지의 구조를 나타내는 프로그래밍 인터페이스입니다.
> - 웹 페이지의 HTML과 XML 문서를 트리 구조로 표현합니다.
> - 자바스크립트를 통해 문서의 내용, 구조, 스타일을 동적으로 읽고 수정할 수 있게 해줍니다.

> 파일 시스템
>
> - 웹 기반의 이미지 편집기, 텍스트 에디터와 같은 애플리케이션에서 사용자가 로컬 파일을 열고, 편집하며, 저장할 수 있도록 할 때 사용됩니다.
> - 드래그 앤 드롭 인터페이스를 통해 파일을 업로드하고 처리할 때 사용됩니다.
> - 사용자의 로컬 파일 시스템에서 파일을 읽고 쓰는 기능을 필요로 하는 웹 애플리케이션을 개발할 때 사용됩니다.

비동기 처리

비동기 처리는 컴퓨터가 여러 작업을 동시에 처리할 수 있게 하는 방식입니다. 예를 들면, 커피를 내리는 동안에도 다른 일을 할 수 있는 것처럼, 한 작업을 기다리는 동안 다른 작업을 진행할 수 있습니다.

환경	예시
브라우저 환경	AJAX 요청, `setTimeout`, `Promise`, `async/await`를 사용하여 비동기 작업을 수행
JS 런타임 환경	파일 I/O, 네트워크 요청 등을 `Promise`, `async/await`를 사용하여 비동기적으로 처리

▶ 비동기 설명

> 비동기가 필요한 이유
>
> 1. 길게 걸리는 작업을 기다리는 동안 다른 작업을 할 수 있어, 컴퓨터가 시간을 더 절약하여 효율적으로 사용할 수 있습니다.
> 2. 웹 페이지에서 어떤 내용을 기다릴 때 페이지가 멈추지 않고 사용자가 다른 활동을 계속할 수 있어, 사용자 경험이 개선됩니다.

3. 서버에서 여러 요청을 동시에 처리할 수 있어, 자원 사용을 최적화하고 대기 시간을 줄여서 작업 처리 효율을 증가시킬 수 있습니다.

> 비동기 처리 방식
>
> - 브라우저 환경과 JS 런타임 환경에서의 비동기 처리 방식은 해당 환경의 주요 작업과 관련이 있습니다.
> - 브라우저 환경은 주로 사용자와의 상호작용과 관련된 작업에 중점을 두고
> - JS 런타임 환경은 파일이나 서버와의 상호작용에 더 많은 초점을 맞춥니다.

> 브라우저 환경에서 비동기 처리
>
> - 사용자와 상호작용하며 데이터를 처리할 때 AJAX 요청, setTimeout, Promise, async/await 같은 기술을 사용합니다.
> - 웹사이트에서 서버로부터 새로운 데이터를 불러오거나, 일정 시간 후에 특정 작업을 수행하는 것 등이 있습니다.

> JS 런타임 환경 (Node.js 등)에서 비동기 처리
>
> - 파일 읽기/쓰기(File I/O)와 서버로의 네트워크 요청과 같은 백엔드 작업을 처리할 때 Promise나 async/await를 사용합니다.
> - 서버가 파일 시스템에서 데이터를 읽거나, 다른 서버에 데이터를 요청하는 경우 등이 있습니다.