목차

웹(Web)

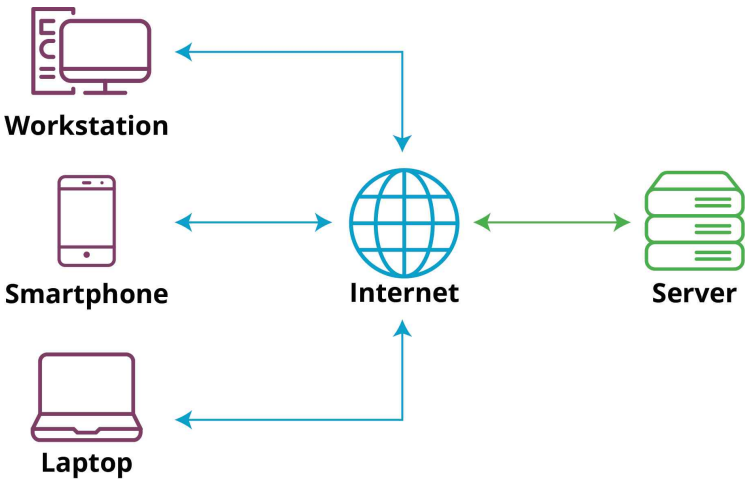
World Wide Web(세계적인 정보 네트워크)의 약어로, 인터넷을 통해 전 세계적으로 연결된 문서, 이미지, 동영상, 애플리케이션 및 다양한 멀티미디어 콘텐츠를 제공하는 정보 시스템입니다. 웹은 주로 HTML(HyperText Markup Language)이라는 마크업 언어를 사용하여 문서와 다른 리소스를 설명하는데, 이러한 문서들은 하이퍼링크(Hyperlink)를 통해 서로 연결되어 있습니다. 사용자는 웹 브라우저를 사용하여 인터넷을 통해 웹페이지에 접속하고, 하이퍼링크를 클릭하면 다른 웹페이지로 이동하거나 다양한 콘텐츠를 보거나 다운로드할 수 있습니다.

웹 브라우저(Web browser)

사용자가 웹에서 정보를 검색하고 보는데 사용되는 소프트웨어 애플리케이션으로, HTML(HyperText Markup Language)과 같은 웹 페이지를 해석하고 표시하여 사용자가 웹사이트를 볼 수 있도록 합니다. 웹 브라우저는 하이퍼링크를 따라 이동하고, 웹사이트의 텍스트, 이미지, 비디오 및 기타 멀티미디어 콘텐츠를 표시하는 동작을 수행합니다. 대표적인 웹 브라우저에는 Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera 등이 있습니다.

※ 참고로 https://caniuse.com/ 라는 사이트에서 내가 작업한 결과물이 해당 브라우저에서 올바르게 동작되는지 미리 확인할 수 있습니다.

웹의 동작 방식



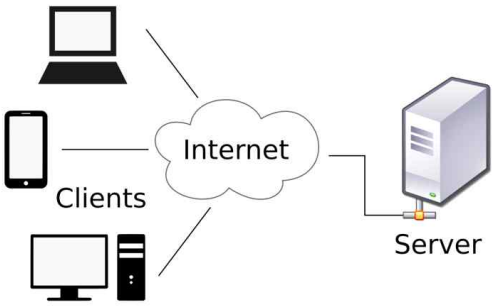
1. 클라이언트의 요청(request)
: 사용자가 웹 브라우저를 통해 특정 웹페이지에 접속하거나
이미지, 동영상 등의 자원을 요청합니다.

2. 서버로 클라이언트의 요청 전송
: 웹 브라우저가 사용자의 요청을 HTTP(Hypertext Transfer Protocol) 요청 메시지로 변환하여 해당 웹 서버로 전송합니다.

3. 서버의 응답(response)
: 웹 서버는 클라이언트의 요청을 받고, 요청된 웹페이지나 리소스를 찾아서 HTTP 응답 메시지(HTTP response message)로 변환합니다.

4. 응답의 처리
: 웹 브라우저는 받은 HTTP 응답 메시지를 해석하여 화면에 표시합니다. 이때, HTML, CSS, JavaScript 등의 웹 기술을 사용하여 웹 페이지를 렌더링합니다.

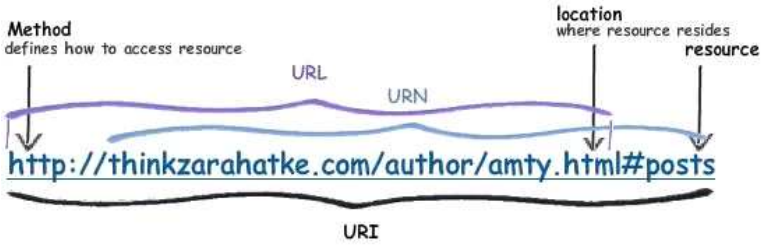
클라이언트-서버 모델



네트워크 기반 응용 프로그램에서 사용되는 기본적인 아키텍처입니다. 클라이언트와 서버 간의 상호작용을 기반으로 하여 클라이언트(사용자)가 웹 브라우저나 애플리케이션을 사용하여 서버에 요청을 보내고, 서버가 그 요청에 대해 응답을 제공하는 방식으로 동작합니다. 여기에서 클라이언트는 사용자가 사용하는 컴퓨터, 스마트폰 등의 디바이스와 웹 브라우저, 애플리케이션 등을 의미하고, 서버는 웹 서버, 데이터베이스 서버와 같이 정보를 제공하거나 처리하는 시스템을 의미합니다. 웹 브라우징, 이메일, 파일 공유, 데이터베이스 관리 등 다양한 네트워크 응용 프로그램에서 사용되며, 전 세계적으로 널리 사용되고 있는 아키텍처입니다.

! 알아두면 쓸데있는 전공 지식 !

URL, URI, URN의 차이



• **URL(Uniform Resource Locator)**

: 특정 리소스(파일, 문서, 데이터베이스 등)의 자원이 어디에 있는지를 지정하는 주소입니다. 일반적으로 URL은 HTTP, FTP 등의 프로토콜, 호스트, 포트 번호, 경로 및 파일 이름으로 구성됩니다.

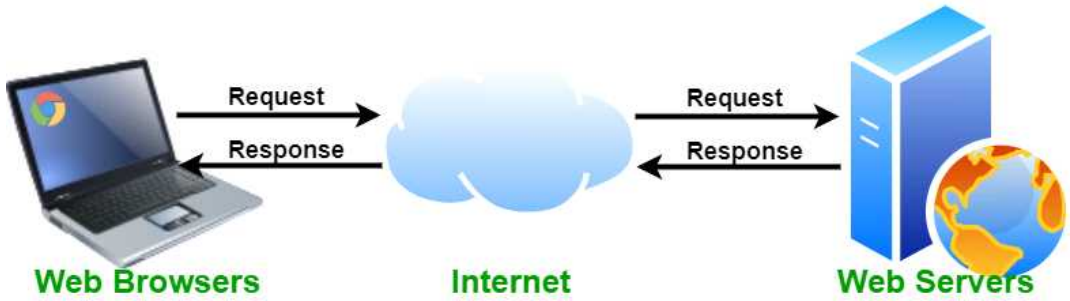
• **URN(Uniform Resource Name)**

: 자원을 식별하기 위한 이름으로, 자원의 위치나 현재 상태에 고유한 식별자를 제공하여 어떤 자원인지를 식별하는 데 사용됩니다.

• **URI (Uniform Resource Identifier)**

: 자원을 식별하기 위한 일반적인 개념으로, URL과 URN을 모두 포함하는 상위 개념입니다. URL과 URN을 포함하며, URL은 자원의 위치를, URN은 자원의 이름을 식별합니다. URI는 이 둘을 포함한 모든 자원 식별자를 나타낼 수 있습니다.

웹 서버(Web Server)



웹의 기본적인 구성 요소 중 하나로, 전 세계적으로 인터넷을 통해 웹 사이트를 공개하는데 사용됩니다. 인터넷 상에서 웹 페이지나 웹 애플리케이션을 저장하고 관리하는 컴퓨터 프로그램으로, 클라이언트로부터 HTTP 요청을 받아들여 해당 요청에 대한 HTML 문서, 이미지 파일 등의 정적인 컨텐츠나 웹 애플리케이션의 실행 결과와 같은 동적인 컨텐츠를 생성하여 클라이언트에게 응답하는 역할을 수행합니다. 이러한 과정에서 웹 브라우저나 웹 애플리케이션과 같은 클라이언트 프로그램은 사용자가 웹 사이트를 방문하고 상호작용할 수 있게 됩니다.

대표적인 웹 서버(Typical Web Server)

- Oracle Web Server
- Google Web Server
- Microsoft Azure Web Server
- Amazon Web Service - EC2(Elastic Compute Cloud)

WS(Web Server) & WAS(Web Application Server)

웹 서버(Web Server)

- 기본적으로 웹 서비스를 제공하는 역할
- 정적인 콘텐츠 (HTML 파일, 이미지, CSS 파일 등)를 제공하는데 주로 사용
- 클라이언트로부터 HTTP 요청을 받아 해당 요청에 대한 정적인 파일을 반환

웹 서버의 종류

1) Apache HTTP Server

: 가장 널리 사용되는 오픈 소스 웹 서버로, 확장성이 높고 다양한 모듈과 기능을 제공합니다. 다양한 운영 체제에서 동작하며, 안정성과 신뢰성을 중시하는 경우에 많이 사용됩니다.

ex) 개인 웹사이트, 기업 웹 애플리케이션, 블로그 등

2) Nginx

: Nginx는 가벼우면서도 높은 성능을 제공하는 웹 서버로, 비동기 이벤트 기반 아키텍처를 기반으로 다수의 동시 연결을 처리할 수 있습니다. 또한 리버스 프록시, 로드 밸런싱, 캐싱 등 다양한 기능을 지원합니다.

ex) 대규모 트래픽을 처리하는 웹 애플리케이션, 로드 밸런서로 사용되는 경우

웹 어플리케이션 서버(Web Application Server)

- 동적인 콘텐츠 생성 및 처리, 데이터베이스 연동, 비즈니스 로직 실행
- 웹 서버와 함께 동작하여 동적인 페이지 생성 및 웹 어플리케이션의 로직 처리
- Java EE, ASP.NET, PHP와 같은 다양한 웹 어플리케이션 프레임워크를 지원

웹 어플리케이션 서버의 종류

1) Apache Tomcat

: Apache 소프트웨어 재단에서 개발한 오픈 소스 웹 애플리케이션 서버로, Java Servlet, JSP(JavaServer Pages), Java Server-side 기술을 실행하는 데 사용됩니다. Tomcat은 경량이면서도 강력한 웹 애플리케이션 서버이기에 많은 개발자들과 기업에서 사용합니다.

ex) 웹 애플리케이션 호스팅, 프로토타이핑 및 개발

2) IBM WebSphere Application Server:

: IBM이 개발한 상용 웹 애플리케이션 서버로 Java EE(Enterprise Edition) 플랫폼을 기반으로 한 고성능 웹 애플리케이션 및 웹 서비스를 지원합니다. 큰 기업과 기관에서 안정성과 확장성을 요구하는 대규모 웹 애플리케이션을 구축하는 데 많이 사용됩니다.

ex) 대규모 기업 애플리케이션, 분산 컴퓨팅

<u>차이점</u>

1. 콘텐츠 처리

• 웹 서버

: 정적인 콘텐츠를 처리합니다.

• 웹 어플리케이션 서버

: 동적인 콘텐츠와 비즈니스 로직을 처리합니다.

2. 언어 및 프로토콜 지원:

• 웹 서버

: 주로 HTTP 프로토콜을 처리, 정적 파일 제공에 중점

• 웹 어플리케이션 서버

: 다양한 프로그래밍 언어와 프로토콜을 지원하여
동적 웹 애플리케이션 실행

<u>관계성</u>

웹 서버(WS)와 웹 어플리케이션 서버(WAS)는 대부분 두 가지를 함께 구축하여 하나의 웹 애플리케이션을 구축합니다. WS는 **정적인 콘텐츠**를 처리하고, 동적인 요청이나 비즈니스 로직이 필요한 경우 해당 요청을 WAS로 전달하여 **동적인 콘텐츠 요청을 처리**하고, 필요한 데이터베이스 연동이나 비즈니스 로직을 수행한 후 결과를 다시 웹 서버에게 반환하여 클라이언트에게 응답하는 방식으로 구축함으로써 정적 콘텐츠와 동적 웹 애플리케이션을 효율적으로 제공할 수 있습니다.

웹 동작을 위해 사용되는 프로토콜

1. HTTP (Hypertext Transfer Protocol)
: HTTP는 웹 서버와 웹 클라이언트 간에 텍스트 기반의 데이터를 주고받기 위한 프로토콜입니다. HTTP는 보통 80번 포트를 사용하며, 웹 브라우저가 서버로부터 웹페이지나 리소스를 요청하고 받을 때 사용됩니다.

2. HTTPS (HTTP Secure)
: HTTPS는 HTTP의 보안 버전으로, 데이터 전송 과정에서 암호화를 사용하여 보안을 강화한 프로토콜입니다. HTTPS는 SSL (Secure Sockets Layer) 또는 TLS (Transport Layer Security) 프로토콜을 기반으로 합니다. HTTPS는 웹사이트에서 민감한 정보를 주고받아야 할 때 주로 사용됩니다.

3. FTP (File Transfer Protocol)
: FTP는 파일을 서버와 클라이언트 간에 전송하기 위한 프로토콜입니다. 웹 개발자들은 FTP를 사용하여 웹 서버에 파일을 업로드하거나 다운로드할 수 있습니다. FTP는 21번 포트를 사용합니다.

4. SMTP (Simple Mail Transfer Protocol)
: SMTP는 이메일을 보내는 데 사용되는 프로토콜입니다. 이메일 클라이언트가 이메일 서버로 이메일을 보내는 데에 SMTP를 사용합니다.

! 알아두면 쓸데있는 전공 지식 !

Nginx & Apache Web Server

- **Apache Web Server**

: Apache는 가장 널리 사용되는 오픈 소스 웹 서버 소프트웨어입니다. 안정성과 다양한 기능, 확장성으로 유명하며, 여러 운영 체제에서 사용됩니다. 다양한 인증 및 권한 부여 메커니즘을 지원하여 웹 애플리케이션의 보안을 강화할 수 있습니다. 대부분의 운영 체제와 호환되며, 사용자 커뮤니티가 크기 때문에 다양한 지원과 확장 기능을 제공합니다.

- **Nginx**

: Nginx는 높은 성능과 안정성을 제공하는 오픈 소스 웹 서버 및 리버스 프록시 소프트웨어로, 초기에는 웹 서버로 시작했지만, 현재는 로드 밸런서, 캐시 서버, 리버스 프록시 서버로 널리 사용됩니다. 비동기 이벤트 기반 아키텍처로 다수의 동시 연결을 처리할 수 있습니다. 가벼우면서도 높은 성능을 제공하여 대규모 트래픽을 효과적으로 처리할 수 있습니다. 캐싱, 로드 밸런싱과 같은 기능을 내장하고 있어 유연한 설정이 가능합니다.

둘 다 강력하고 안정적인 웹 서버 솔루션으로, Nginx는 주로 높은 성능과 대규모 트래픽을 처리할 때 사용되며, Apache는 다양한 모듈과 안정성에 중점을 둔 경우에 사용될 수 있습니다.

웹 페이지(Web Page)

: 웹 페이지(Web page)는 월드 와이드 웹(World Wide Web)에서 볼 수 있는 기본적인 문서 단위로, 웹 브라우저를 통해 인터넷을 통해 접근할 수 있습니다. 일반적으로 텍스트, 이미지, 링크, 멀티미디어 요소 및 다양한 웹 기술을 사용하여 정보를 제공하고 시각적으로 표현합니다. HTML, CSS, JavaScript 등과 같은 웹 기술을 사용하여 디자인되고 구성되는데 HTML은 웹페이지의 기본 구조와 내용을 정의하며, CSS는 웹 페이지의 스타일, 레이아웃 및 디자인을 담당하며, JavaScript는 동적인 요소와 상호작용을 추가할 수 있습니다.

HTML, CSS, Javascript

• **HTML(Hyper Text Markup Language)**

: 오늘날 모든 웹은 HTML로 작성되고, 변환하면 HTML이 나오는 구조로 구성되어 있습니다. 현재 HTML의 표준은 HTML5로, 이 표준을 반드시 지켜서 웹 페이지를 작성하는 것이 원칙입니다. 태그만을 사용하여 구문을 생성하고, 마크업 언어라는 이름에 걸맞게 웹 페이지 마크업(구조화)을 위해 사용됩니다. 웹 페이지 생성 시 핵심적인 역할을 하는 언어입니다. 웹 개발자는 HTML을 사용하여 웹 페이지를 구축하고 디자인하며, 이를 통해 웹의 다양한 기능을 구현할 수 있습니다. 사용자가 HTML 파일을 서버에게 요청하면 웹 브라우저가 해당 요청을 처리한 결과인 HTML 파일을 해석하여 사용자에게 시각적으로 보여줍니다.

웹에서 하는 역할

1) 콘텐츠 구조화
2) 하이퍼링크 생성
3) 문서 스타일링
4) 시맨틱 마크업

DOM(Document Object Model)

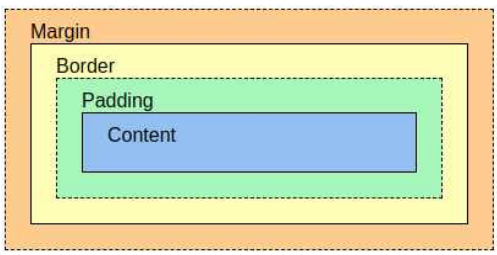
: HTML의 DOM(Document Object Model) 객체는 트리 형태로 표현된 웹 페이지의 계층적인 구조를 JavaScript를 이용하여 이벤트를 처리할 수 있는 기능을 제공하는 인터페이스입니다. DOM은 태그, 속성, 텍스트 등의 웹 페이지의 모든 요소를 객체로 표현하며, 이러한 객체들은 계층 구조를 형성합니다. JavaScript를 사용하여 DOM을 조작하면 웹 페이지의 내용, 구조, 스타일, 이벤트 등을 동적으로 변경할 수 있습니다.
DOM은 JavaScript를 사용하여 DOM 객체에 접근할 수 있습니다. 객체의 요소를 ID, 클래스, 태그 이름 등을 사용하여 선택하거나, 이벤트 핸들러를 통해 동적으로 요소에 접근하여 속성을 읽거나 변경하고, 새로운 요소를 생성하고 추가할 수 있습니다.

- ## CSS(Cascading Style Sheet)

: CSS(포괄적인 스타일 시트, Cascading Style Sheets)는 웹 페이지의 레이아웃, 디자인, 표현을 제어하기 위한 스타일 언어입니다. HTML(하이퍼텍스트 마크업 언어)와 함께 사용되며, HTML은 웹 페이지의 구조를 정의하고, CSS는 해당 구조에 스타일을 적용하여 웹 페이지를 시각적으로 꾸밀 수 있게 해줍니다.

웹에서 하는 역할
1) 콘텐츠 각각의 속성 수정을 통한 스타일링
2) 레이아웃 제어
3) 애니메이션 및 전환 기능

CSS 박스모델



- **Javascript**

: 자바스크립트(JavaScript)는 웹 개발에서 가장 널리 사용되는 프로그래밍 언어 중 하나입니다. 웹 브라우저에서 동적인 웹 페이지를 구현하기 위해 사용되어 왔지만, 현재는 웹 프론트엔드/백엔드뿐만 아니라 다양한 환경에서 사용되고 있습니다.

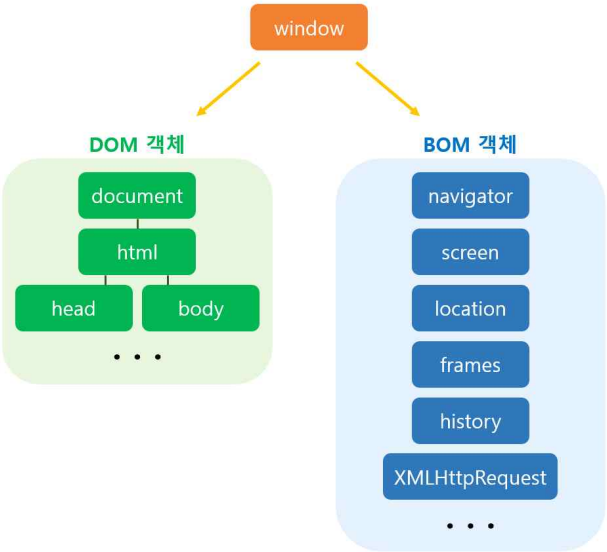
자바스크립트는 객체 기반의 스크립트 언어로, 변수, 함수, 조건문, 반복문과 같은 기본적인 프로그래밍 구조를 제공합니다. 또한, 이벤트 처리, DOM(Document Object Model) 조작, AJAX(Asynchronous JavaScript and XML)를 통한 비동기 통신 등과 같은 기능을 지원하여 웹 페이지를 동적으로 만들고 사용자와 상호작용할 수 있게 합니다.

웹에서 하는 역할
1) 이벤트 처리
2) 동적 콘텐츠 업데이트
3) 웹 브라우저 제어
4) 쿠키 및 로컬 스토리지 관리

이벤트 리스너
JavaScript의 이벤트 리스너(Event Listener)는 특정 이벤트가 발생했을 때 실행할 동작을 정의하는 역할을 합니다. 웹 페이지에서 사용자의 상호작용(클릭, 마우스 오버, 키보드 입력 등) 또는 웹 브라우저의 동작(로딩 완료, 리사이즈 등)에 반응하여 원하는 동작을 수행할 수 있게 해줍니다. 이벤트 리스너는 사용자 경험을 향상시키고 웹 페이지의 동적인 동작을 구현하는 데 사용됩니다.

코어 객체
: Javascript의 코어 객체(Core Objects)는 언어 자체에서 제공하는 기본 객체들로, 웹 개발 및 애플리케이션 개발에서 널리 사용됩니다. 이러한 코어 객체들은 웹 브라우저 환경에서 기본적으로 제공되며, 웹 페이지의 구조를 조작하고 데이터를 다루는 등 다양한 작업을 수행할 수 있게 해줍니다. 대표적인 Javascript의 코어 객체에는 Object, Array, Date 등이 있습니다.

DOM vs BOM, 뭐가 다를까?



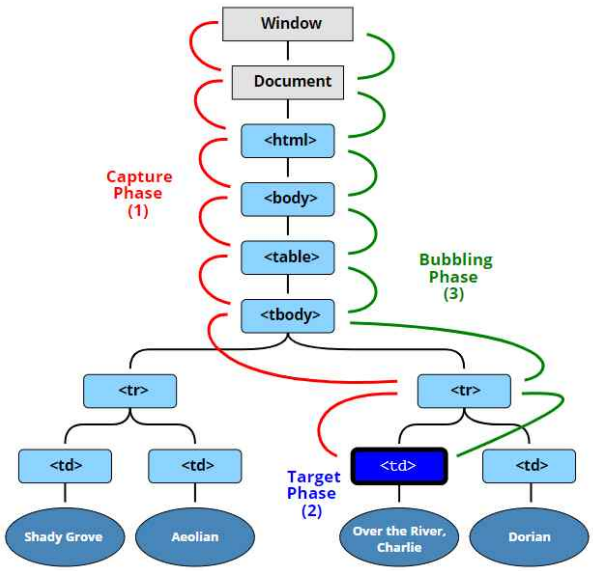
• **DOM(Document Object Model)**

: HTML의 DOM(Document Object Model) 객체는 트리 형태로 표현된 웹 페이지의 계층적인 구조를 JavaScript를 이용하여 이벤트를 처리할 수 있는 기능을 제공하는 인터페이스입니다. DOM은 태그, 속성, 텍스트 등의 웹 페이지의 모든 요소를 객체로 표현하며, 이러한 객체들은 계층 구조를 형성합니다. JavaScript를 사용하여 DOM을 조작하면 웹 페이지의 내용, 구조, 스타일, 이벤트 등을 동적으로 변경할 수 있습니다. DOM은 JavaScript를 사용하여 DOM 객체에 접근할 수 있습니다. 객체의 요소를 ID, 클래스, 태그 이름 등을 사용하여 선택하거나, 이벤트 핸들러를 통해 동적으로 요소에 접근하여 속성을 읽거나 변경하고, 새로운 요소를 생성하고 추가할 수 있습니다.

• **BOM(Browser Object Model)**

: BOM(Browser Object Model)은 웹 브라우저의 창이나 프레임을 제어하기 위한 객체 모델을 의미합니다. DOM이 HTML 문서의 구조를 나타내는 것과는 달리, BOM은 웹 브라우저의 창, 히스토리, 위치 정보 등과 관련된 객체들을 다룰 수 있게 해줍니다. BOM은 브라우저 환경에서 JavaScript를 사용하여 웹 브라우저를 제어하고 상호작용할 때 주로 활용됩니다.

이벤트 리스너



: 이벤트 리스너(Event Listener)는 웹 페이지에서 사용자의 상호작용(클릭, 키보드 입력, 마우스 움직임 등)이나 브라우저의 동작(로딩 완료, 리사이즈 등)과 같은 이벤트에 대한 반응을 정의하는 JavaScript 함수입니다. 이벤트 리스너는 해당 이벤트가 발생했을 때 캡처 단계와 버블 단계를 수행해 특정 이벤트가 올바르게 실행되도록 하는 역할을 수행합니다.

1) 캡처 단계(Capture Phase)

: 이벤트가 DOM 트리의 상위 요소에서 하위 요소로 향하는 단계입니다. 이벤트가 발생한 요소에서부터 DOM 트리의 루트까지 이벤트가 흘러 내려가는 과정입니다. 즉, 이벤트가 최상위 요소에서부터 이벤트가 발생한 요소까지 이동합니다. 이 과정에서 캡처 단계에 등록된 이벤트 리스너들은 순차적으로 실행됩니다. 하위 요소로 내려가면서 상위 요소에 등록된 캡처 단계의 이벤트 리스너가 실행됩니다.

2) 버블 단계(Bubble Phase)

: 이벤트가 발생한 요소에서 시작하여 상위 요소로 향하는 단계입니다. 이벤트가 발생한 요소에서 상위 요소로 이벤트가 올라가는 과정입니다. 이 과정에서 버블 단계에 등록된 이벤트 리스너들은 순차적으로 실행됩니다. 이벤트가 버블 단계를 타고 상위로 올라가면서 각 요소에 등록된 버블 단계의 이벤트 리스너가 실행됩니다.

CMS(Content Management System)

: 콘텐츠 관리 시스템의 약자로, 웹 사이트의 콘텐츠를 생성, 편집, 관리, 배포하는 데 사용되는 소프트웨어나 도구를 의미합니다. CMS는 비전문가들도 웹사이트를 관리하고 업데이트할 수 있게 해주기 때문에 비전문적인 사용자들도 웹사이트 관리를 더 쉽게 만들어줍니다.

제공 기능

1) 콘텐츠 관리 및 편집
2) 템플릿, 테마 제공
3) 사용자 권한 관리
4) 보안 및 안정성

CMS의 예시

1) WordPress

: 가장 널리 사용되는 오픈 소스 CMS로, 블로그부터 기업 웹사이트까지 다양한 종류의 웹사이트를 구축할 수 있습니다.

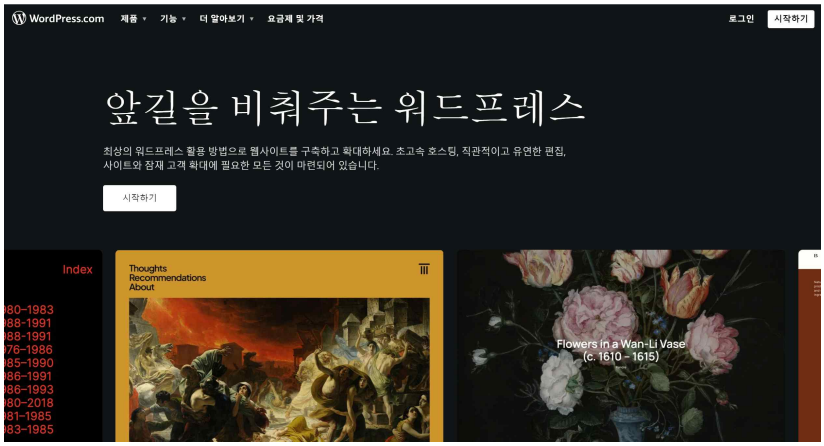
2) Joomla

: Joomla는 유연하고 확장 가능한 오픈 소스 CMS로, 포탈 웹사이트와 기업 웹사이트를 만드는 데 많이 사용됩니다.

3) Drupal

: Drupal은 기업 및 조직용으로 많이 사용되는 고급 오픈 소스 CMS로, 커뮤니티 사이트나 포탈 웹사이트를 구축하는 데 적합합니다.

WordPress



: 가장 널리 사용되는 오픈 소스 콘텐츠 관리 시스템(CMS)으로, 웹 사이트 및 블로그를 만들고 관리하기 위한 플랫폼입니다. WordPress는 사용자 친화적인 인터페이스와 다양한 플러그인, 테마, 위젯 등을 제공할 뿐만 아니라 보안 패치 및 업데이트를 자동으로 제공하여 시스템의 안정성을 유지하기 때문에 비전문가인 이용자도 사이트를 쉽게 구축할 수 있습니다. WordPress를 이용한 웹사이트 구축이 처음이라면 WordPress 사용자 커뮤니티를 통해 다양한 문제에 대한 도움을 받을 수 있습니다.