1. 초기 웹(1990년대 초반)

- **1990년**: 팀 버너스리(Tim Berners-Lee)가 월드 와이드 웹(WWW)을 발명하면서 시작되었습니다. 그는 HTML(HyperText Markup Language), HTTP(HyperText Transfer Protocol), 그리고 최초의 웹 브라우저(WorldWideWeb)를 개발했습니다.
- 이 시기의 웹 페이지는 단순한 텍스트와 하이퍼링크만 포함되었으며, 정적인 정보 제공이 주된 목적이었습니다.
- **HTML 1.0**: 웹의 표준으로 사용된 첫 번째 언어로, 텍스트 중심의 웹사이트가 주를 이루었습니다.

2. 웹 1.0 시대(1990년대 중반-후반)

- 웹이 상업적으로 도입되기 시작했으며, 대중적으로 브라우저가 보급되었습니다. 넷스케이프(Netscape Navigator)와 마이크로소프트의 인터넷 익스플로러가 주된 브라우저였습니다.
- **HTML, CSS, JavaScript의 등장**: HTML로 구조를, CSS로 스타일을, JavaScript로 간단한 동적 기능을 구현하기 시작했습니다.
- 이 시기의 웹사이트는 주로 정적 콘텐츠로 구성되었으며, 사용자와의 상호작용이 제한적이었습니다.

3. 웹 2.0 시대(2000년대 초반)

- 웹 2.0은 사용자 참여와 상호작용이 강조된 시기로, 소셜 미디어와 웹 애플리케이션의 부상이 특징입니다.
- **AJAX**(Asynchronous JavaScript and XML): 사용자가 페이지를 새로 고침하지 않고도 서버와 데이터를 주고받을 수 있게 되어, 동적인 웹 애플리케이션이 가능해졌습니다.
- 이 시기의 주요 특징은 사용자 생성 콘텐츠(블로그, 위키, 소셜 미디어 등)와 웹 애플리케이션의 급격한 발전입니다. 페이스북, 유튜브, 위키피디아 등이 이 시기의 대표적인 예입니다.
- **페이지 기반 웹 애플리케이션**
- 웹 2.0 시대에는 대부분의 웹 애플리케이션이 서버 렌더링을 기반으로 했습니다. 서버에서 HTML 페이지를 완전히 생성해 클라이언트(브라우저)로 보내주는 방식입니다. 사용자가 어떤 요청(예: 새로운 페이지 이동, 폼 제출)을 하면 서버는 해당 요청을 처리한 후 새 HTML 페이지를 반환하고, 브라우저는 이 페이지를 다시 로드하는 방식이었습니다. 이는 정적 웹 페이지와 동적 웹 페이지 모두에서 일반적인 방식이었습니다.
- **AJAX의 등장**
    - AJAX(Asynchronous JavaScript and XML)의 등장은 웹 2.0 개발 방식에서 큰 전환점을 마련했습니다. 이전에는 사용자가 요청을 보낼 때마다 전체 페이지가 새로 고침되었지만, AJAX는 특정 데이터만 서버와 비동기적으로 교환할 수 있게 했습니다. 이를 통해 페이지 전체를 다시 로드하지 않고도 일부분만 업데이트할 수 있었으며, 웹 애플리케이션이 더욱 동적이고 반응성을 갖추

게 되었습니다. 예를 들어, 구글 맵스는 AJAX 기술을 적극 활용한 대표적인 웹 애플리케이션으로, 사용자가 지도를 이동할 때마다 페이지를 새로 고침하지 않고도 실시간으로 지도가 변하는 경험을 제공했습니다.

- **서버와 클라이언트 간 통신**
  - AJAX가 나오기 전에는 서버와 클라이언트 간의 통신이 HTTP 요청/응답 방식에 의존했습니다. 이는 주로 HTML 폼을 제출하거나 페이지 링크를 클릭했을 때 발생하며, 클라이언트는 GET 또는 POST 요청을 보내고 서버는 그에 대한 HTML 응답을 반환했습니다. AJAX가 도입된 이후에는 클라이언트가 XML 또는 JSON 형식의 데이터를 서버로 요청하고, 서버는 데이터를 처리한 후 해당 데이터를 클라이언트로 반환하는 비동기 방식이 가능해졌습니다. 이로 인해 웹 애플리케이션이 더 복잡한 상호작용을 처리할 수 있었습니다.
- **백엔드와의 통합**
  - API가 대중화되기 전에는 백엔드 로직과 프론트엔드 로직이 강하게 결합된 경우가 많았습니다. 서버 측에서는 JSP, ASP, PHP와 같은 기술을 사용해 서버에서 직접 HTML을 생성하고, 백엔드와 프론트엔드 코드가 혼합되어 있었습니다. 이러한 방식은 유지보수가 어려울 수 있었지만, 당시에는 주로 이 방식을 사용했습니다. 이때 서버 측 스크립트가 데이터베이스와 통신하고, 그 결과를 HTML 페이지로 렌더링한 후 클라이언트에 전달하는 방식이 주된 개발 패턴이었습니다.
- **초기 API 형태**
  - 비록 오늘날처럼 RESTful API나 GraphQL 같은 명확한 API 패러다임이 확립되지 않았지만, 웹 2.0 시기에도 웹 서비스 간에 데이터를 교환하는 방법은 있었습니다.
    - **SOAP**(Simple Object Access Protocol): XML 기반의 웹 서비스 프로토콜로, 원격 시스템 간의 상호작용을 가능하게 했습니다. SOAP는 HTTP, SMTP 등을 통해 데이터를 전송했으며, 당시 웹 서비스에서 널리 사용되었습니다.
    - **XML-RPC**: SOAP와 유사한 방식으로, 원격 프로시저 호출을 XML로 인코딩하여 데이터를 교환하는 방법입니다. 간단한 구조였지만 확장성은 제한적이었습니다.
- **클라이언트 측 자바스크립트와의 연동**
  - 웹 2.0 시기에 JavaScript와 DOM 조작 기술이 점차 발전하면서, 클라이언트 측에서도 더욱 동적인 상호작용이 가능해졌습니다. AJAX와 함께 JavaScript 라이브러리들이 등장하면서 클라이언트에서 서버로의 요청을 비동기적으로 보내고, 그 결과를 처리해 UI를 동적으로 업데이트하는 방식이 보편화되었습니다. 대표적으로 **jQuery**가 많은 인기를 끌며 DOM 조작 및 AJAX 통신을 쉽게 처리할 수 있게 해주었습니다.

4. 현대 웹 개발(2010년대 이후)

- **웹 프레임워크의 등장**: 다양한 서버 및 클라이언트 프레임워크의 출현은 개발 프로세스를 체계화하고 효율성을 높였습니다. 이러한 프레임워크들은 반복적인 작업을 자동화하고, 코드의 재사용성을 높이며, 개발자가 비즈니스 로직에 집중할 수 있도록 지원합니다. 서버 측에서 Ruby on Rails, Django, Spring과 같은 프레임워크는 MVC 아키텍처를 통해 코드의 구조를 명확히 하고, 클라이언트 측에서는 Angular, React, Vue.js와 같은 JavaScript 프레임워크가 동적인 사용자 인터페이스를 구현하는 데 기여했습니다.

    - 서버 측: **Ruby on Rails**, **Django**, **Spring** 등이 대표적입니다.
    - 클라이언트 측: **Angular**, **React**, **Vue.js**와 같은 JavaScript 프레임워크들이 인기를 끌었습니다.
- **모바일 퍼스트 디자인**: 스마트폰과 모바일 기기의 보급이 가속화되면서, 웹 디자인은 모바일 퍼스트 원칙을 채택하게 되었습니다. 이는 개발자가 먼저 모바일 환경을 고려하여 웹 애플리케이션을 설계하고 개발하도록 유도합니다. 반응형 웹 디자인은 다양한 화면 크기에 자동으로 적응하여 사용자 경험을 최적화합니다. 이를 통해 사용자는 다양한 디바이스에서 일관된 경험을 누릴 수 있으며, 기업은 더 넓은 고객층에 접근할 수 있게 되었습니다.
- **API와 마이크로서비스**
    - **REST API와 GraphQL**: REST API의 등장으로 서버와 클라이언트 간의 데이터 통신이 효율적으로 이루어졌습니다. RESTful 서비스는 자원에 대한 CRUD 작업을 직관적으로 처리할 수 있도록 해 주며, GraphQL은 클라이언트가 필요한 데이터를 명시적으로 요청할 수 있게 하여 오버페칭과 언더페칭 문제를 해결했습니다.
    - **마이크로서비스 아키텍처**: 대규모 서비스를 작은 독립적인 서비스로 나누어 개발하고 배포하는 접근 방식입니다. 각 서비스는 특정 비즈니스 기능을 수행하며, API를 통해 상호작용합니다. 이를 통해 개발 효율성과 확장성을 높이고, 각 팀이 독립적으로 작업할 수 있게 됩니다.
- **클라우드 컴퓨팅과 DevOps**: AWS, GCP와 같은 클라우드 서비스의 도입은 서버 인프라 관리의 간편함을 제공하며, DevOps 문화가 확산됨에 따라 CI/CD(지속적 통합 및 지속적 배포) 파이프라인이 널리 사용되었습니다. 이는 개발자와 운영 팀 간의 협업을 증진시키고, 소프트웨어 배포 주기를 단축시켜 더 높은 품질의 소프트웨어를 빠르게 제공할 수 있도록 합니다.

5. 미래 웹 (현재 진행 중)

- **웹 3.0**: 분산화와 블록체인 기술을 활용한 탈중앙화 웹의 개념이 떠오르고 있습니다. Web3는 주로 블록체인과 암호화폐, 스마트 계약 등을 기반으로 한 새로운 인터넷 환경을 지향합니다.
- **프로그레시브 웹 앱(PWA)**: 웹과 모바일 앱의 경계를 허물고, 웹 애플리케이션이 네이티브 앱과 유사한 경험을 제공할 수 있도록 하는 기술이 발전하고 있습니다.
- **AI와 머신러닝의 접목**: 인공지능(AI) 기술이 웹 애플리케이션에 접목되면서, 보다 지능적이고 개인화된 경험을 제공하는 웹 서비스가 증가하고 있습니다.