

웹 서비스 플랫폼

Hybrid Web 2

모바일 웹 환경

<https://open.kakao.com/o/gKmQ3Z2d>



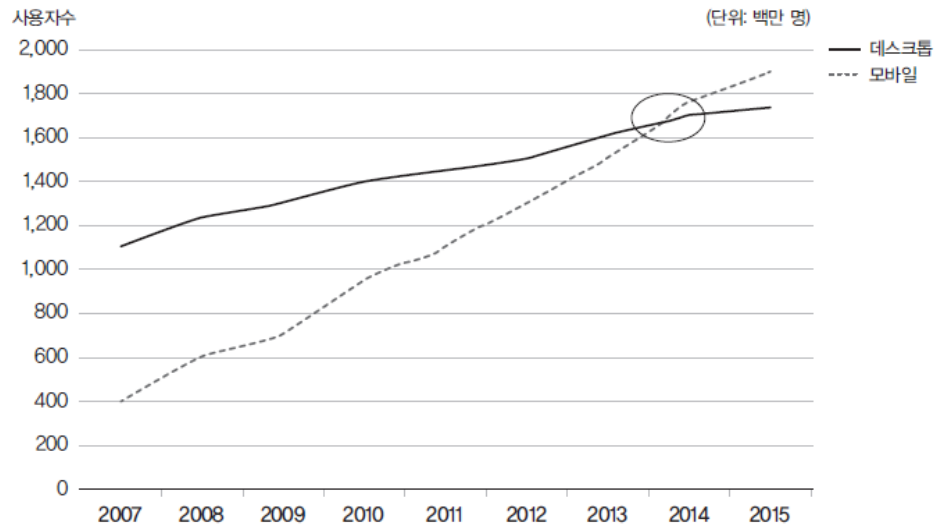
1.1 웹 환경의 변화

- 웹(web)
 - 월드 와이드 웹(WWW 또는 W3; [World Wide Web](#))의 줄임말
 - 전 세계를 연결하는 [인터넷](#) 망을 통해 서로 정보를 공유하는 정보 서비스 공간
- 웹 정보의 작성과 표현
 - 웹의 모든 정보는 HTML(Hyper Text Markup Language) 언어로 표현되는 웹 문서로 작성
 - HTTP(Hyper Text Transfer Protocol) 통신 규약을 통해 전달
 - 전달된 웹 문서는 웹 브라우저에 의해 해석되어 사용자에게 제공
- 팀 버너스 리(Tim Berners-Lee, 스위스 CERN 입자물리학연구소)
 - 웹의 선구자(웹 개념의 제안자, HTML의 최초 개발자)
 - 웹 관련 모든 코드를 공개, W3C를 통해 HTTP, HTML 표준 등을 제안
 - 최초의 웹 브라우저 'World Wide Web'도 제공



'모바일'이라는 새로운 웹 접근 환경의 등장

- 데스크톱 환경에서 모바일 환경으로의 이동과 변화가 진행중
 - [그림 1-1] 데스크톱 & 모바일 사용자 수의 변화



- 새로운 모바일 웹 환경이 PC를 기반으로 한 데스크톱 웹 환경을 빠르게 대체
- 주요 변화
 - 사용자 : 이동 중에도 웹을 접근할 수 있게 됨으로써 자주, 더 빠르게, 더 다양한 웹 정보를 요구
 - 개발자 : 다양한 모바일 장치와 모바일 OS, 모바일 웹 브라우저를 지원해야 하는 부담 증가



- 웹 서비스 제공 방식의 변화
 - 초기 '데이터 중심의 웹' 방식
 - 일방적으로 정보를 제공하던 정적인 형태
 - 점차 '서비스 중심의 웹' 방식으로 발전
 - 상호 교류하며 양방향의 정보 이동이 가능한 동적인 형태
- 웹 패러다임의 진화([웹 1.0에서 웹 3.0으로](#))
 - 웹 1.0
 - 포털(portal) 중심의 웹의 시대
 - 일방적으로 정보를 제공
 - 웹 2.0
 - 플랫폼 중심의 웹 시대
 - 정보의 생성, 공유, 참여가 가능
 - 예) 유튜브나 위키피디아(사용자끼리 뭉쳐 새로운 콘텐츠를 개발할 수 있는 공간)
 - 웹 3.0
 - 웹이 모든 환경의 플랫폼이 되는 시대
 - 원하는 정보를 찾아 개인별 맞춤 서비스가 가능



● [표 1-1] 웹 패러다임의 변화

비교	웹 1.0	웹 2.0	웹 3.0
시기	2000년 이전	2000년 ~ 2010년	2010년 이후
개념	포탈(portal)로서의 웹 웹 사이트의 집합체	플랫폼(platform)으로서의 웹 웹 애플리케이션 제공	시맨틱 웹
중심	기술	인간	지능화된 개인
콘텐츠 특성	폐쇄성	개방성(참여와 공유)	맞춤형
실행 환경	종속성	독립성	융합·개방형
정보 교류	단방향	양방향	자유로운 소통
기술	HTML, ActiveX	Ajax, HTML5, CSS3, RSS, XML	RDF, 시맨틱웹, RFID, USN
사용자 역할	정보소비자	정보생산자/소비자	참여자/의사결정자



- 웹 마크업 언어의 발전

- 마크업(markup)

- 문서 내용 자체가 아닌 내용에 관한 크기와 모양, 표시 위치와 같은 부가적인 정보

- [표 1-2] 웹 마크업 언어의 종류

마크업 언어	특징
SGML	Standard Generalized Markup Language HTML과 모든 마크업 언어의 기반이 된 마크업 언어 다양한 전자 문서들의 구조와 내용을 명세하기 위한 표준
HTML	Hyper Text Markup Language 쉽고 편리한 웹 페이지 작성용 마크업 언어 웹 기술의 출발점이 된 웹 문서 표준
XML	eXtensible Markup Language 문서 형식과 의미를 표현하는 마크업 언어 문서나 정보의 교환을 위한 웹 문서 표준
XHTML	eXtensible Hyper Text Markup Language HTML에 XML의 장점을 접목한 마크업 언어 현재 진행이 중단된 웹 문서 표준
HTML5	HTML 4.01과 XHTML 1.1을 계승한 차세대 웹 기술 웹 페이지에서 웹 애플리케이션까지 지원하는 마크업 언어이자 플랫폼 기술 현재 진행 중인 차세대 웹 문서 표준



● 웹 표준

- 나이나 장애 유무에 상관없이 누구나 어떤 환경에서도 자유롭게 웹을 접근하고 서비스를 받기 위해서는 반드시 필요함
- W3C가 정한 표준 기술만을 사용하여 웹 문서의 구조와 표현, 기능을 제공하는 것
- 사용자뿐만 아니라 개발자에게도 많은 이점을 제공

● 웹 표준을 따라야 하는 이유

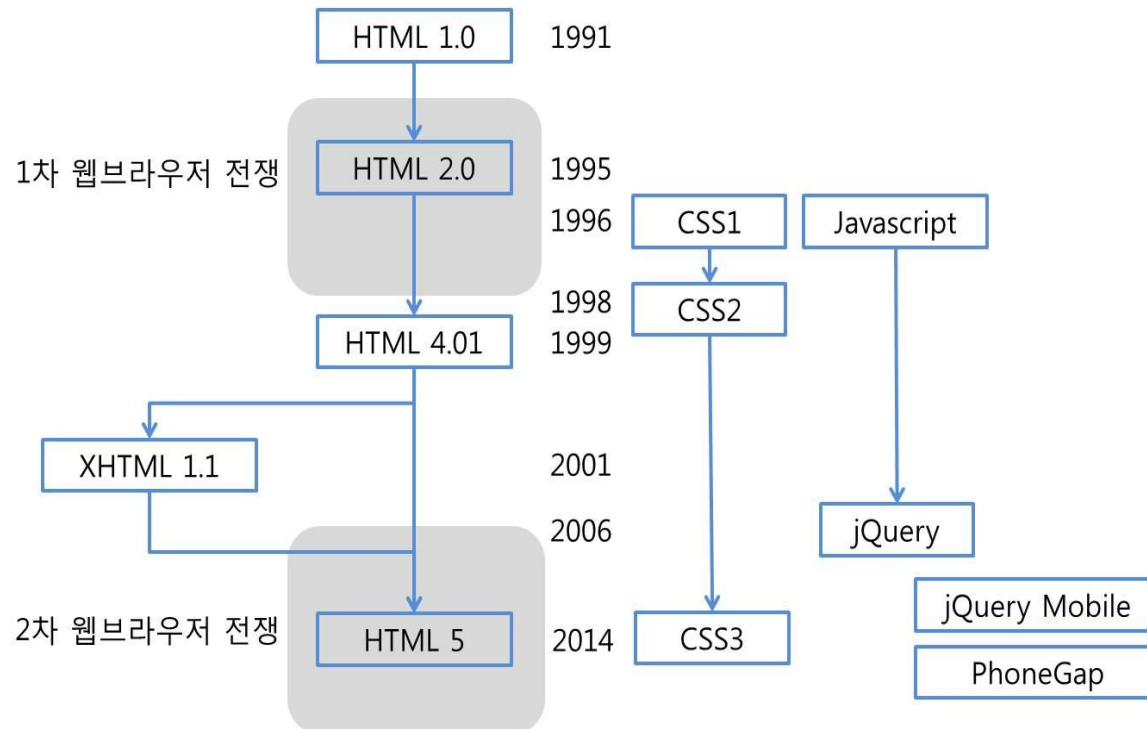
- 다양한 장치의 접근성을 지원
- 구조와 표현이 분리됨으로써 디자인, 기획, 개발 업무가 단순화
- 개발을 위해 배워야 하는 내용이 제한되어 개발 부담이 감소
- 웹 브라우저 상·하위 버전간의 호환성이 유지
- 다양한 플랫폼에서 실행 가능한 단일 프로그램 개발로 개발 및 유지 비용이 감소
- 장애인, 노약자 등에 대한 보편적인 웹 접근성이 지원
- 유효성 검사 서비스(W3C 제공)를 통해 웹 표준 준수 여부를 쉽고 빠르게 검증



2.1 HTML5의 등장

● HTML5의 주요 발전 과정

- 초기 HTML – 정보 전달을 위한 간단한 웹 문서를 만들기 위한 언어, 기술로 한정됨
- HTML5 – 웹 애플리케이션을 만들기 위한 웹 플랫폼 기술로 확장되고 있음
- HTML 표준화 그룹
 - [W3C](#)(World Wide Web Consortium)
 - [WHATWG](#)(Web Hypertext Application Technology Working Group)



HTML5 등장의 의미

- HTML의 최신 버전이며 웹 변화의 중심
 - 2011년 초안을 발표한 뒤, 2014년에 정식 최종 권고안을 발표
 - 웹 브라우저마다 빠르게 지원을 확대하고 있는 중
- 단순 마크업 수준을 넘어서는 애플리케이션으로서의 웹을 실현하는 강력한 기능을 제공
 - 'Web on Everything' 개념에 기반한 표준 기술
 - 모든 스마트 기기(스마트 카, 스마트 홈)에 적용되는 차세대 웹 플랫폼 형태로 진화
 - HTML5를 웹 운영체제로 사용하여 웹과 앱 서비스를 제공하는 형태로 발전할 예정



● HTML5 사양(spec)

- 다양한 웹 브라우저들이 구현해야 할 공통된 기능
- 모든 브라우저는 HTML 페이지가 동일하게 실행되도록 호환성을 보장해야 하며 이를 위해 공통된 사양을 지원해야 함
- 주요 HTML5 API(HTML5의 주요 기능)

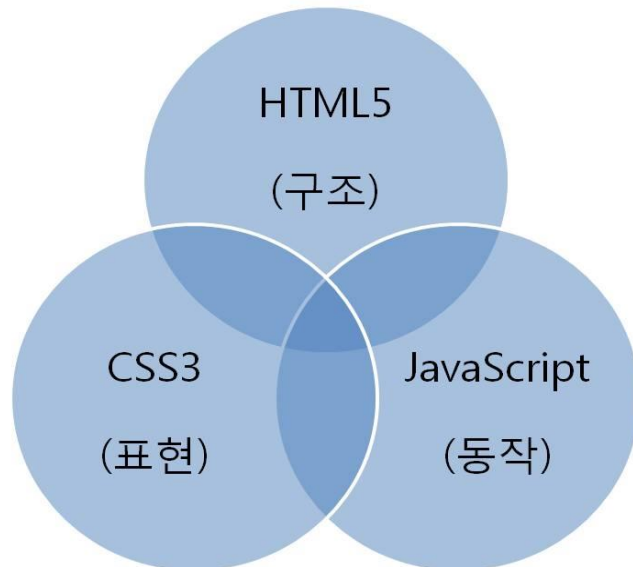
기능	내용
캔버스	2차원 그래픽을 그리기 위한 캔버스 API 제공
멀티미디어	별도의 미디어 플레이어 없이도 실행 가능한 비디오와 오디오 API 제공
위치정보	지리적 위치 정보를 제공하는 위치 정보 API 제공
오프라인 웹	인터넷 연결 없이도 브라우저 보관 HTML과 자원을 통해 정상적인 실행을 지원하는 어플리케이션 캐시 API 제공
웹 데이터베이스	브라우저를 통해 데이터 관리를 지원하는 데이터베이스 API 제공
로컬저장소	브라우저를 통해 데이터를 저장하기 위한 API 제공
웹 소켓	서버와의 실시간 양방향 데이터 교환을 위한 API 제공
웹 워커	웹 애플리케이션을 위한 스레드 관련 API 제공



2.2 HTML5의 특징

● HTML5의 발전 방향

- 웹 브라우저라는 플랫폼 위에서 동작하는 애플리케이션(웹앱)을 HTML과 CSS, 자바스크립트 언어로 만들 수 있도록 하는 것
- 애플리케이션으로 동작하기 위해 필요한 기능들을 API 형태로 제공하도록 확장됨
- CSS3도 단순 화면 구성이 아닌 애플리케이션에 맞는 세련된 화면과 화면 움직임을 제공하도록 확장됨
- 자바스크립트는 단순한 HTML 문서의 조작 수준을 넘어 다양한 HTML5 API를 활용하고 제어하며 통신하는 역할로 확대됨
- [그림 1-4] 웹 페이지의 구성 요소



- 웹 표준으로서의 위상 강화
 - 다양한 장치뿐만 아니라 운영체제, 웹 브라우저의 종류에 상관없이 어떤 환경에서도 정상적으로 동작하는 웹 페이지를 지원(비 호환성 문제를 해결)
- 웹 문서의 의미 구조화 지원
 - 시맨틱 태그를 추가하고 기존 태그들도 의미를 강화함으로써 '[시맨틱 웹](#)' 개념을 지원
- 웹 언어에서 웹 플랫폼으로의 기능 확장
 - 플러그인이 필요 없도록 다양한 기능의 API(HTML을 단순한 웹 표현 언어에서 웹 어플리케이션 개발 플랫폼으로 확장)를 추가로 제공
 - 웹 페이지 자체가 하나의 애플리케이션이 될 수 있는 환경이 됨
- 웹 폼 기능 지원
 - 다양한 유형의 입력 웹 폼을 통해 편리한 입력 인터페이스와 데이터 유효성 검증 제공
- 풍부한 웹 페이지(미디어) 기능 지원
 - HTML5는 캔버스, SVG, 오디오, 비디오 등의 각종 미디어 관련 태그를 제공
- 모바일 웹 어플리케이션 개발 지원
 - 다양한 지원 API를 통해 네이티브 애플리케이션을 어느 수준까지는 대체할 수 있는 모바일 웹 애플리케이션을 개발 가능하도록 함



3.1 웹 브라우저

- 플랫폼이 된 웹 브라우저

- 단순한 뷰어(viewer)가 아니라 프로그램을 실행할 수 있는 플랫폼(platform)으로 간주됨
- 'HTML 페이지'라는 프로그램이 '웹 브라우저'라는 운영체제에서 실행되는 것으로 이해

- 웹의 목표

- 장비, 환경 등의 차이에도 불구하고 모든 사용자가 제약 없이 접근할 수 있는 정보 공간이 되는 것 (Web for All)
- 1990년대 중반 웹 브라우저들이 각자 고유 기술을 경쟁적으로 도입함으로써 비호환성 문제 발생
=> 인터넷 익스플로러가 시장을 점유하는 동안 비호환성 문제는 더욱 심화됨

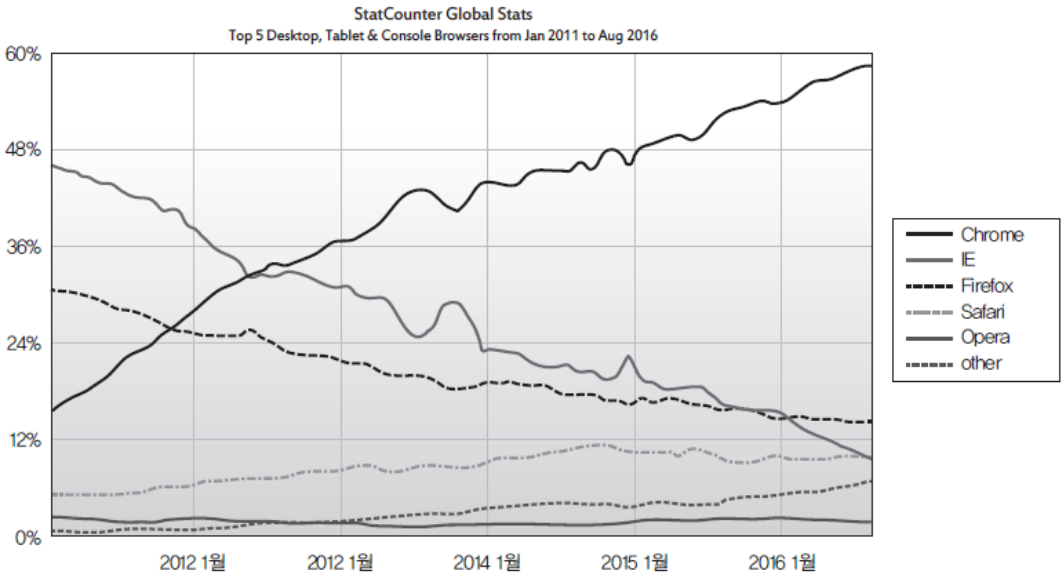
- 웹 브라우저 전쟁

- 1차 웹 브라우저 전쟁 : 네스케이프 <-> 익스플로러
- 현재는 2차 웹 브라우저 전쟁 중
 - 구글사의 크롬(Chrome), 애플사의 (Safari), 모질라 재단의 파이어폭스(Firefox), 오페라(Opera)
<-> 익스플로러
 - 모바일 환경의 변화를 틈타 반격을 시도

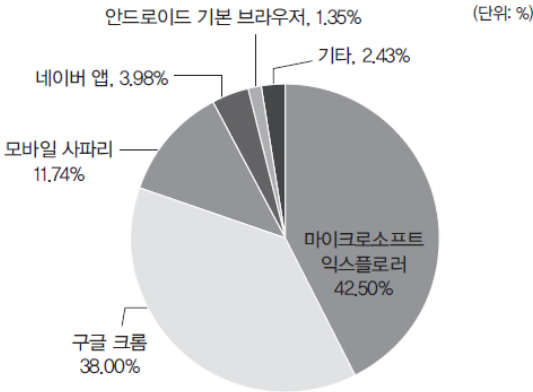


웹 브라우저 시장의 변화






● 최근 1년간의 세계 시장 점유율 변화



● 최근 1년간 국내 주요 웹사이트 접속 브라우저의 현황



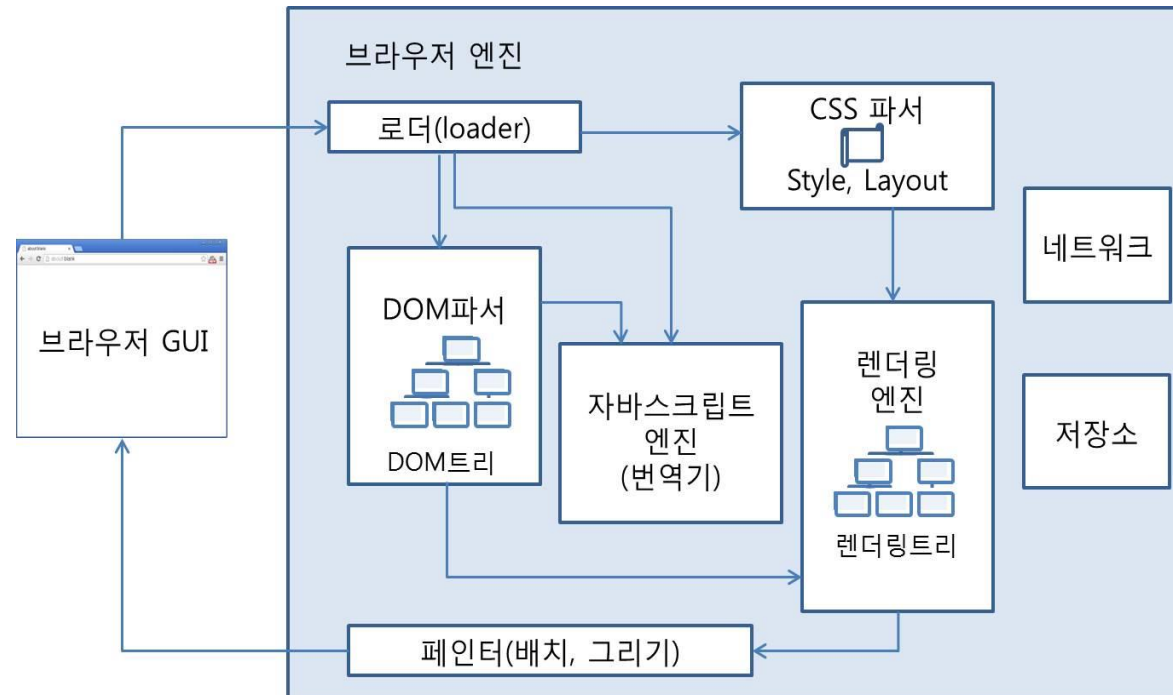
주요 웹 브라우저의 종류

<u>브라우저</u>	제조사	특징	최신버전	로고
익스플로러	마 이 크 로 소프트	가장 범용적인 브라우저 액티브-X 기술 지원 보안이 취약하고 웹 표준 준수율 낮음	14	
크롬	구글	가볍고 빠른 브라우저 안드로이드 호환성으로 최근 점유율 향상됨	54	
파이어폭스	모질라	공개 코드로 개발중인 점유율 높은 브라우저 빠른 속도와 보안, 프라이버시에 강점	50	
사파리	애플	간결한 디자인과 독특한 기능 모바일용으로 많이 사용됨	10	
오페라	오 페 라 소 프트웨어	작은 용량과 빠른 속도 오랜 역사, 모바일용으로 사용됨	39	



3.2 웹 브라우저 엔진

- 웹 브라우저
 - 단순한 번역기(interpreter) 이상의 역할을 수행
 - HTML, CSS, 자바스크립트 코드를 받아 해석하고 처리하는 기능을 수행
 - 내부에 웹 브라우저 엔진 가짐
- 웹 브라우저 엔진 기본 요소



- DOM 트리 구성 : DOM 파서
 - HTML 문서를 파서(parser)를 통해 파싱해서 메모리에 'DOM(Document Object Model) 트리'라는 내부 트리를 구성
- 렌더 트리 구성 : 렌더 엔진
 - 렌더 트리(DOM 트리 중 화면에 보여지는 엘리먼트만을 선별해서 만든 내부 저장 구조)
 - CSS 스타일 내용을 분석하여 렌더 트리가 완성됨
- 레이아웃 구성
 - 구성된 렌더 트리를 웹 브라우저 화면의 지정된 영역에 매핑하여 배치함으로써 레이아웃(layout)을 구성
 - CSS 스타일이 분석되어 적용됨
- 페인팅 : 페인터
 - 구성된 레이아웃을 실제 화면의 브라우저 창에 표현



웹 브라우저 엔진의 종류

- 웹 브라우저 엔진

- 웹 브라우저의 핵심
- 개발 난이도가 높은 다양한 코드와 기술 요소들이 결합된 모듈 소프트웨어
- 웹 브라우저는 내부적으로는 다음 핵심 엔진 중 하나에 각기 다른 부속 기능과 스킨 등을 추가함

- 웹 브라우저 엔진의 종류

- 게코(Gecko) – 모질라 파이어폭스(Firefox), 모질라 파이어폭스 모바일(Fennac)
- 웹킷(Webkit) – 애플 사파리(Safari), 애플 사파리 모바일(Safari Mobile), 구글 크롬(Chrome)
- 트라이던트(Trident) – 마이크로소프트의 인터넷 익스플로러(IE), IE Mobile
- 프레스토(Presto) – 오페라소프트웨어의 오페라



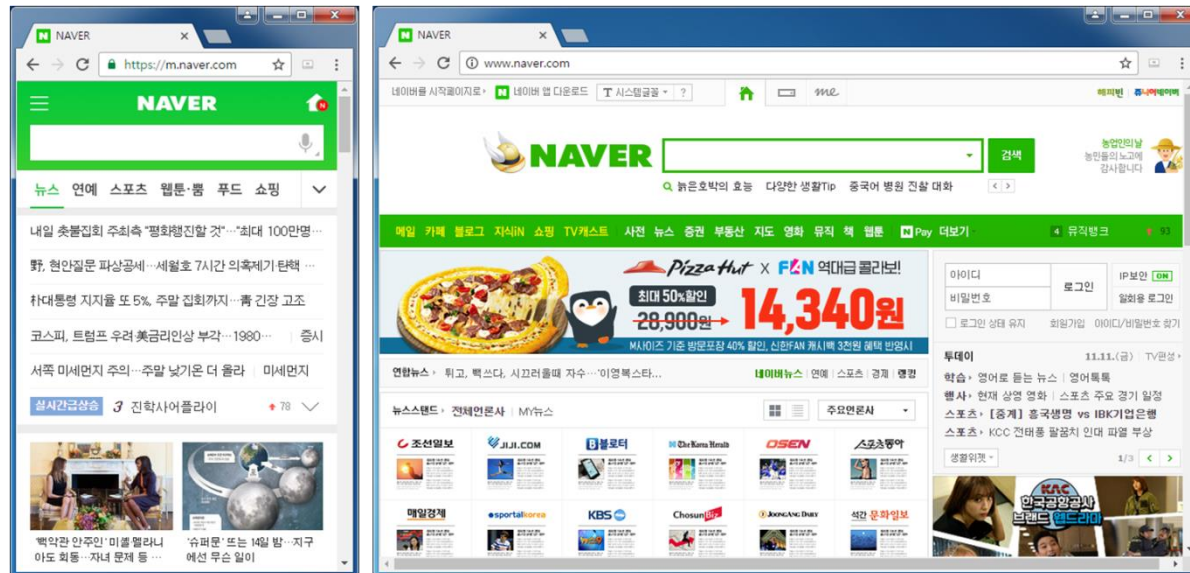
웹 브라우저의 처리 과정

- MVC(Model View Controller) 구조를 그대로 따름
- MVC 개발 방식
 - 프로그램 개발 모듈을 화면(View), 데이터(Model) 그리고 기능(Controller) 영역으로 분할하여 분석, 설계하고, 이들 간의 상호 작용을 통해 전체 프로그램이 실행되는 구조로 개발하는 방식
 - 대표적인 설계/아키텍처 패턴
- [표 1-5] 웹 브라우저의 MVC 구조
 - 하나의 HTML 문서 안에는 MVC 각 요소가 코드로 포함됨

MVC 패턴요소	처리 대상(코드)	기능	브라우저 구성요소
모델(Model)	HTML 코드 (<html> ... </html>)	데이터	DOM 파서
뷰(View)	CSS 코드 (<style> ... </style>)	레이아웃	렌더 엔진
기능 제어기(Controller)	JavaScript 코드 (<script> ... </script>)	동적 처리	자바스크립트 엔진

4.1 데스크톱 웹과 모바일 웹

- 데스크톱 웹(desktop web)
 - 웹 환경을 이야기할 때 말하는 보통의 웹
- 모바일 웹(mobile web)
 - 태블릿, 스마트폰 등의 모바일 브라우저로 접속하도록 만들어진 웹 페이지
 - 일반적인 웹사이트를 모바일 환경으로 그대로 옮긴 형태
- 데스크톱 웹과 모바일 웹은 구분되고 각기 다른 관점에서 개발되어야 함
 - 모바일 장치에서 데스크톱 웹을 실행시키면 화면 해상도가 달라서 제대로 보기가 어려움
 - 웹 페이지를 작성할 때 모바일 장치의 환경에 맞춰 화면을 구성해야 함



4.2 모바일 앱

- 앱(app)

- 애플리케이션(application)의 줄임말
- 모바일 기기에서 실행되는 응용 프로그램
- 데스크톱 환경의 응용 프로그램(애플리케이션)과 구분하기 위한 용어

- 모바일 프로그래밍 방식의 변화

- 초기 모바일 프로그래밍(네이티브앱 방식)
 - 기존 서버 중심의 웹 프로그래밍 -> 새롭게 안드로이드나 아이폰 프로그래밍 기술을 습득해야 했음
 - ASP, JSP, PHP -> Swift나 Java를 새로 익혀 사용
- 최근 '웹의 반격'이 시작됨(웹앱, 하이브리드앱)
 - 기존 웹 기술을 가지고 모바일 장치의 애플리케이션과 유사한 특성을 갖는 프로그램의 개발이 가능해짐
 - 전통적인 웹 기술(HTML5와 CSS3)의 발전 + 새로운 프레임워크 기술(제이쿼리 모바일, 폰갭)의 등장



모바일 앱의 개발 방식

- 네이티브앱 (native app)
 - 네이티브 언어를 사용하여 개발된 모바일 장치에 최적화된 애플리케이션
 - 각 플랫폼 전용의 개발 도구(언어)와 SDK(Software Development Kit)를 이용하여 개발
 - 아이폰(XCode와 Swift), 안드로이드폰(자바), 윈도우폰(C#)
- 웹앱 (web app)
 - 웹 기술로 개발하지만 겉모양은 네이티브앱처럼 보이는 애플리케이션
 - 모바일 브라우저를 통해서 동작하면서도 네이티브앱과 비슷한 화면과 터치 관련 사용자 경험(UX)을 제공
- 하이브리드앱 (hybrid app)
 - 네이티브앱과 웹앱의 장점을 결합한 애플리케이션
 - 대부분의 기능은 웹 기술로 개발하고 최소한의 기능만 네이티브 기술로 구현



1) 네이티브앱

- 특징

- 카메라, 스피커, GPS 등의 장치들을 직접 제어할 수 있고 효율성이 높음
- 각 플랫폼 별로 다른 프로그래밍 언어를 익혀서 앱을 개발해야 함

- 장점

- 각 플랫폼에 최적화되어 실행 속도가 빠르다.
- 스마트폰 안의 모든 장치(하드웨어, 소프트웨어) 접근이 가능하며 정교한 UI 개발이 가능하다.
- 앱 마켓을 통해 등록함으로써 수익 얻기가 쉽다.

- 단점

- 개발 기간이 길고 많은 비용이 소요된다.
- 각 플랫폼 별로 별도의 버전을 다른 언어로 매번 개발해야 한다.
- 앱 마켓을 통해 배포되므로 갱신이나 유지보수가 복잡하고 어렵다.



2) 웹앱

- 특징

- 기존의 웹 개발 기술을 그대로 활용할 수 있음
- 기기에 상관없이 한 번 개발하면 어떤 환경에도 적용할 수 있음

- 장점

- 기존의 표준 웹 기술을 사용하므로 단기간에 빠르고 쉽게 개발할 수 있다.
- 웹 브라우저만 있으면 다양한 장치와 플랫폼에서 동일하게 실행 가능하다.
- 웹 브라우저를 통해 배포되므로 갱신이나 유지보수가 쉽다.

- 단점

- 웹 페이지 내용이 많을 경우, 성능 문제가 발생할 수 있다.
- 모바일 장치에 대한 제어가 제한적이며 코드 효율성이 낮다.
- 적용할 수 있는 UI에 한계가 있다.



3) 하이브리드앱

- 특징

- 표준 웹 기술로 웹앱을 개발하고 이를 다시 네이티브앱으로 변환한 다음 배포하는 방식
- 크로스 프레임워크(cross framework)를 사용하여 웹 페이지를 애플리케이션으로 변환하거나 애플리케이션처럼 독립적으로 실행할 수 있음

- 장점

- 다양한 플랫폼을 위한 앱 개발 및 유지보수 비용이 낮다.
- 장치 접근도 가능하고 앱 마켓을 통해 등록, 배포할 수 있다.

- 단점

- 네이티브앱에 비해 실행 속도가 느리다.
- 자유로운 UI 구현에 한계가 있다.



모바일 앱 개발 방식의 비교

- 상호 보완 관계 속에 각 방식마다 특성을 갖고 함께 발전할 전망
 - 네이티브앱
 - 각 모바일 장치와 플랫폼의 특성을 최대한 이용하는 방향
 - '기능' 중심의 영역에 강점
 - 웹앱, 하이브리드앱
 - 개발 생산성과 비용 측면을 강화하는 방향
 - 동적인 콘텐츠가 핵심인 '정보' 중심의 영역에 강점

특징	네이티브앱	웹앱	하이브리드앱
실행 속도	빠름	보통	보통
장치 제어	높음	낮음	높음
마켓 등록	가능	불가능	가능
개발 비용	높음	낮음	보통
UI 접근성	높음	낮음	보통
플랫폼	단일	다중	다중
갱신	느림(버전업)	빠름(실시간)	보통(버전업)



4.3 모바일 앱 프레임워크

- 프레임워크(framework)

- 애플리케이션의 기본 골격(예: 건축물)
- 표준화된 방식으로 앱 개발을 쉽게 할 수 있도록 함
- CSS와 자바스크립트 기술의 많은 부분들을 자동화하여 보이지 않게 처리

1) 웹 UI 프레임워크

- 웹 UI를 쉽고 빠르게 만들 수 있도록 웹앱 개발을 지원하는 프레임워크
- 모바일 환경에서 앱은 기존 웹과는 다른 화면 요소와 레이아웃, 화면 전환 효과를 갖기 때문에 이를 효과적으로 지원해주는 웹 UI 프레임워크들이 매우 중요함
- 예) 제이쿼리 모바일(jQuery Mobile)

2) 크로스 플랫폼 프레임워크

- 하이브리드앱 개발을 지원하는 프레임워크
- 하나의 원본 코드를 다양한 플랫폼에서 동작하는 네이티브 애플리케이션들로 변환
- 예) 코르도바(폰갭)



Reference

- ✓ 모바일 웹 웹 앱 하이브리드 앱 입문, 박성진, 생능출판

