

현대 웹 애플리케이션 개발

인터넷이란?

- 사용자가 원하는 정보를 얻기 위해 직접 먼 곳을 가지 않고도 언제 어디서든지 접근할 수 있는 기술
 - 과거 : 서울에서 제주도의 맛집을 알기 위해서는 직접 여기저기 돌아다녀야함
 - 현재 : 네이버와 같은 검색포털을 이용하여 맛집 블로그를 둘러보고 간접체험을 통해 지식을 습득.

웹이란?

- 인터넷을 이용하여 원거리에 있는 문서들을 열람하는 공간
 - 해당 문서들은 웹 브라우저가 해석 가능한 특정 형식으로 작성되어 있음

웹 브라우저

- 웹에 있는 문서들을 사용자가 보기 편하게 표시해주는 소프트웨어
- URL (Uniform Resource Locator) 을 통해 해당 문서에 접근이 가능하도록 구성되어있음

URI (Uniform Resource Identifier) : <https://www.naver.com/?id=1&pw=2>

URL (Uniform Resource Locator) : <https://www.naver.com>

웹 페이지 & 웹 애플리케이션

- 초창기 웹에서 브라우저로 접근 가능한 문서들을 웹 페이지라고 일컬음
- 페이지의 작은 인터렉션을 추가하던 js 의 역할이 범용적으로 넓어지면서, 웹 애플리케이션이라는 표현이 생기기 시작
- 현재도 static sites 들은 웹 페이지, interacting sites 는 웹 애플리케이션 으로 분류
- 그 외에도 규모와 복잡도로 규정 짓는 경향

웹 페이지가 화면에 표시되는 과정

1. 웹 브라우저에 사이트 주소(URL) 입력
2. 입력한 주소의 서버에서 해당 정보를 찾음
3. 해당 정보를 브라우저 화면에 표시
4. 사용자가 해당 사이트 내용을 확인

웹 애플리케이션 구동 과정 (자세히)

1. URL entered : 사용자가 웹 브라우저에서 사이트 주소를 입력한다.
2. DNS Lookup : DNS 를 이용하여 사이트 주소에 해당되는 Server IP 를 접근한다.
3. Socket Connection : Client (브라우저) 와 Server 간 접속을 위한 TCP 소켓 연결.
4. HTTP Request : Client 에서 HTTP Header 와 데이터가 서버로 전송.
5. Content Download : 해당 요청이 Server 에 도달하면 사용자가 원하는 문서를 다시 웹 브라우저에 전송한다.

DNS - Domain Name System

TCP - Transmission Control Protocol

6. Browser Rendering : 웹 브라우저의 렌더링엔진에서 해당 문서를 다음과 같은 순서로 파싱

- HTML 를 DOM (Document Object Model) 으로 변환
- CSS 를 DOM 에 추가 (CSSOM 생성)
- DOM 으로 렌더트리 생성
- 렌더트리 배치
- 렌더트리 그리기

7. Display Content : 렌더트리를 브라우저에 표시 후 사용자에게 웹 페이지로 보여준다.

웹 개발 기술

- HTML5 : 화면에 나타나는 요소 (텍스트 또는 이미지 등)
- CSS3 : 화면에 나타나는 요소를 이쁘게 꾸미는 기술
- Javascript : 화면에 나타나는 요소의 동작을 제어

웹 개발 관련 용어

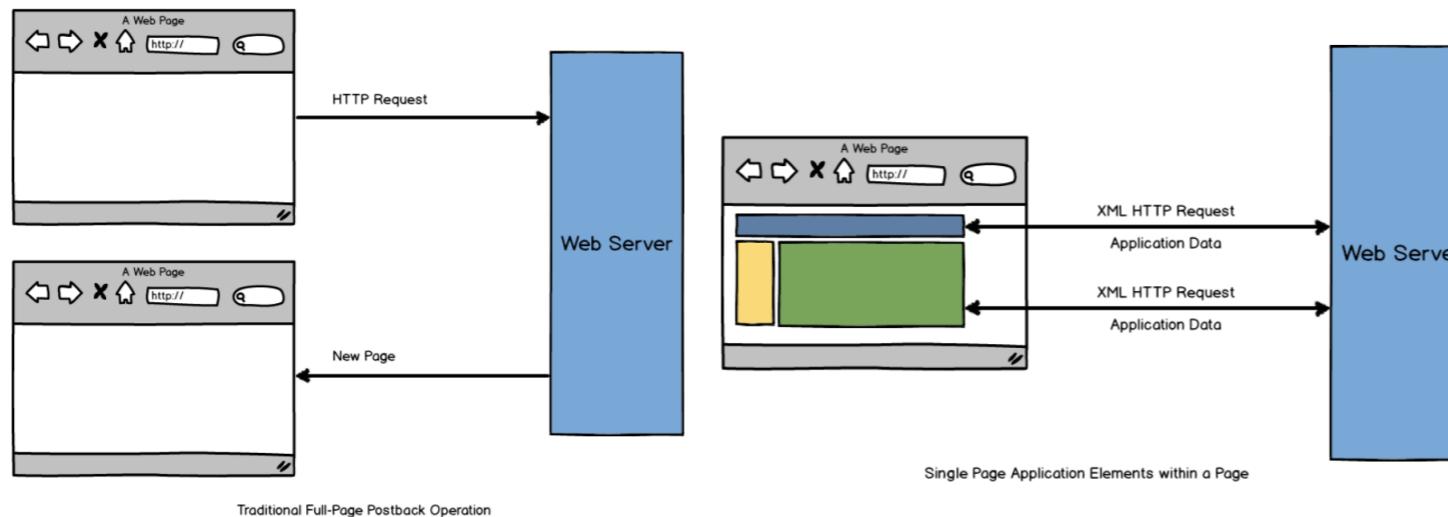
- DOM (Document Object Model) : HTML 의 요소 접근을 용이하게 하기 위해 파싱하여 트리 형태로 구성한 XML / HTML 문서
- Client : 서버에 요청을 보내는 기기, 컴퓨터. 리소스에 접근하려는 사용자
- Server : 서비스 제공자, 원격 컴퓨터, 클라이언트 요청을 받는 컴퓨터
- IP 주소 (Internet Protocol): 인터넷에 연결된 컴퓨터나 기기의 주소를 가리킴
- DNS (Domain Name System) : 도메인 주소에 해당하는 컴퓨터의 IP 주소를 찾는데 사용
- Web Protocols : 웹이 동작하는데 사용되는 프로토콜. *HTTP, HTTPS, FTP, SMTP*
- HTTP : 브라우저가 웹 서버와 통신하는데 사용되는 주 프로토콜. *GET, POST, PUT, DELETE*

Front-End Developer & Back-End Developer

- 프론트엔드 : 화면에 비춰지는 부분. 화면을 기준으로 위치를 생각
 - 마크업 + 스크립트 (UI 화면 개발자)
- 백엔드 : 화면에 뒷단에서 일어나는 데이터 처리에 관한 작업
 - REST + DB + Server (화면 뒷단 처리)

Single Page Application (SPA)

- 웹 개발 초기의 화면 전환 방식은 Full Request 방식
- 근래에는 사용자 경험 (UX) 을 중시하여 화면의 번쩍거림과 불필요한 화면 전환을 자제
- jQuery 의 ajax 통신을 시작으로 여러 js MVC 프레임워크가 등장하여 널리 사용중



Client-Side Rendering vs Server-Side Rendering

- 초기 웹 사이트 개발방법은 URL 요청을 받아 해당 문서를 넘겨주는 서버 사이드 렌더링이 주로 쓰임
- 사용자 경험을 중시하는 웹 개발 트렌드에 따라 Angular, Vue, React 와 같은 자바스크립트 라이브러리를 중심으로 클라이언트 사이드 렌더링이 지배적으로 활용
- 각 렌더링 기법에 따라 장 / 단점이 확실하여 목적에 맞게 사용하는 것이 중요

	장점	단점
Client Side Rendering	높은 사용자 경험 빠른 웹 페이지 전환 JS 라이브러리 결합 용이	SEO 최적화 필요 긴 초기 렌더링 시간 외부 라이브러리 사용 불가피
Server Side Rendering	SEO 우위 첫 페이지 로딩 우수 정적인 사이트에 적합	잦은 서버 요청 페이지 전환 느림 사용자 경험 낮음

Front-End UI Framework

- [Bootstrap](#)
- [Foundation](#)
- [Materialize Design Lite \(MDL\)](#)
- [MaterialzeCSS](#)

Responsive Web Design

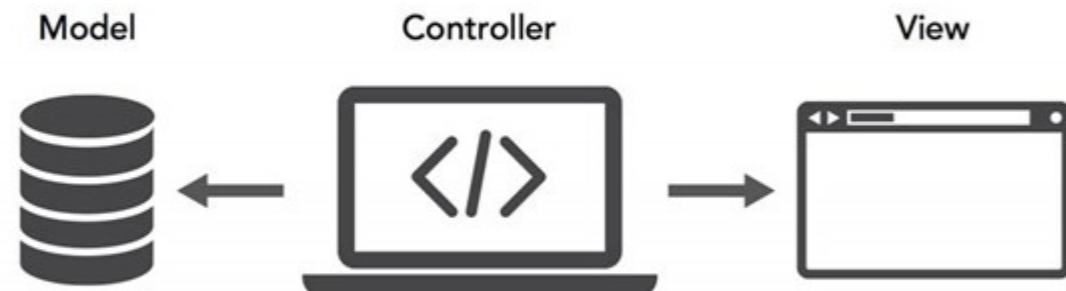
- 모든 기기 (PC, 모바일, 태블릿) 에서 웹 페이지가 잘 보이도록 설계하는 디자인 기법

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- CSS pre-processor : [Sass](#), Less, Stylus

Javascript MVC Framework

- MVC : Model * View * Controller 의 전형적인 웹 애플리케이션 개발 방식
- 대표 프레임워크 : [Angular](#), [Vue](#), [React](#)



Javascript 유ти리티성 라이브러리

- 앱 구현 시 자주 사용되는 기능들을 API 형태로 제공해주는 라이브러리
- 종류 :
 - [jQuery](#)
 - [Underscore.js](#)
 - [Lodash](#)

Web Task Manager

- 프론트엔드 개발에 있어서 반복적인 작업 (새로고침, 프리컴파일, 압축 ...) 등을 자동화 해주는 도구
- 종류 :
 - [Gulp](#)
 - [Grunt](#)

Web Module Loader & Bundler

- 앱의 복잡도가 증가하면 코드를 모듈 별로 관리할 필요성이 증가
- [Require.js \(AMD\)](#) : 모듈 로더
- [Webpack 1 & 2](#) : 모듈 번들러

모듈화, 의존성 관리, 최적화 등의 더 나은 웹 앱 개발을 위한 도구들의 등장

Unit Testing

- 모듈의 기능 확장 및 코드 리팩토링 시에 유용하게 사용
- [Jest](#)
- [Mocha](#)
- [Jasmine](#)

Hybrid Application & React Native

- HTML, CSS, JS 웹 기술로 모바일 앱을 개발
- 기존의 웹 자원 및 웹 기술 스택을 활용하여 개발 생산성 증대가 가능
- 각 Native 개발환경에서 Web 자원을 Native 로 빌드하여 앱으로 뽑아내는 형태
- 하이브리드 - [Web View](#) 기반, [Cordova](#) 의존
- React Native - Native 위젯 컴파일, React 의존

앱 라이프 사이클 : 개발 -> 빌드 -> 배포 -> 설치

Progressive Web Apps

- 모바일 앱의 기능과 웹의 접근성을 결합한 최신 웹 애플리케이션
- 브라우저에 종속적이지만 웹으로 앱을 커버할 수 있다는 장점
- 개발 - 빌드 - 배포 - 설치에서 개발 - 설치로 단축
- Offline, Push 알람, GPS, Install Banner 등의 강력한 기능

참고

- [Internet 101 - Khan](#)
- [How DNS works](#)
- [How Internet works - Django](#)
- [CS-rendering vs SS-rendering](#)
- [RequireJS - Naver D2](#)
- [Webpack - Naver D2](#)

끝