2025.12.31

생성일	@2025년 12월 31일 오전 9:27
태그	HTML OT 웹 프로그래밍 기초

목표

- 클라우드
  - 외산: AWS
  - 국산: NCA, NCP
- 데이터분석
  - ADsP, ADP
  - 빅데이터분석기사
  - AICE-Associate
- 프로젝트
  - 토이 프로젝트 3개
  - 팀 프로젝트

목차

▼ 1) 웹 브라우저

1. 역할
   - 웹 페이지 탐색
     - URL 검색: Local Hosts(캐싱)에서 먼저 찾고 없다면 DNS
     - 단점: 올리는건 쉽지만 전세계 캐싱 데이터를 내리는건 쉽지 않다.
     - 해결책: Local Hosts에 저장될 데이터로 IP 주소 저장
   - 탭 브라우징
   - 북마크
   - 개인 정보 보호
     - 쿠키: 서버가 브라우저가 함께 사용하는 정보(ex.로그인 정보, 사용자 설정,...) 저장 기간 설정이 가능하지만 자동으로 데이터를 서버에 전송하기에 민간한 정보에 주의해야 한다.
     - 캐시: 로딩 속도를 위해 정적 파일(HTML, CSS, JS)을 브라우저에 저장
     - 세션: 실제 데이터는 서버에 저장하고 브라우저는 세션 ID만 가지고 있다. 보안에 민감한 데이터를 다룬다.
   - 확장 프로그램 및 플러그인
   - 웹 페이지 로딩 속도 최적화

2. 구성 요소
    - 사용자 인터페이스
    - 브라우저 엔진
        - 사용자 인터페이스와 렌더링 엔진 사이 중재자 역할
    - 렌더링 엔진
        - HTML 파싱: DOM 트리 생성
        - CSS 파싱
        - 레이아웃 계산: DOM과 CSSOM을 기반으로 요소의 크기와 위치 계산
        - 렌더링: 그리기
            - 렌더링 엔진(Blink, Gecko, WebKit)
    - 자바스크립트 엔진
        - 서버용JS, 클라이언트용JS
        - 역할
            - 파싱 및 컴파일
            - 실행 및 메모리 관리
            - 이벤트 처리
        - 엔진
            - V8
            - SpiderMonkey
            - JavaScriptCore
    - 네트워킹
        - HTTP(80)/HTTPS(443) 요청 - SSL/TLS 암호화 유무 차이
        - 데이터 전송
        - 캐시 관리
    - 데이터 스토리지
        - 쿠키: 서버에 전송되는 사용자, 세션 유지용 작은 텍스트 파일
        - 로컬 스토리지: 로컬에만 저장되는 영구 데이터 저장소
        - 세션 스토리지: 탭을 닫으면 삭제되는 임시 저장소
        - 엔덱스드 DB: 대용량 비관계형(NoSQL) 데이터 베이스
        - 캐시: 정적 리소스를 저장하여 속도 향상

특성	Cookie	Local Storage	Session Storage	IndexedDB	Cache Storage
용량	~4KB	~5-10MB	~5-10MB	수백MB~GB	제한 없음
지속성	만료 시간 설정	영구	탭 닫으면 삭제	영구	영구
서버 전송	✅자동 전송	❌	❌	❌	❌
동기/비동기	동기	동기	동기	비동기	비동기
데이터 타입	문자열	문자열	문자열	객체, Blob 등	Request/Response
사용 난이도	쉬움	쉬움	쉬움	어려움	중간
주요 용도	인증, 세션	설정, 상태 저장	임시 데이터	대용량 데이터	정적 리소스

3. 동작 순서
    a. URL 입력
    b. DNS 조회: Domain Name → Server IP
    c. 서버 연결
    d. 서버의 응답

- CDN (Contents Delivery Network): 분산된 캐싱 서버 데이터 활용

e. HTML 파싱
f. CSS 파싱
g. 자바스크립트 실행
h. 렌더링
i. 사용자와 상호작용

▼ 2) 프론트엔드와 백엔드

1. 프론트의 역할
   - UI/UX 관리
   - 서버와 통신
   - 반응형 웹 디자인
2. 프론트의 확장성
   - 웹 성능 최적화
   - 웹 접근성
3. 백엔드의 역할
   - 서버, DB, API, 애플리케이션 로직
     - SQL: 엄격하고 정교한 데이터 관리 (정합성, 무결성)
     - NoSQL: 대용량 json 데이터를 빠르고 간결하게 처리 (대용량 트랙잭션)
     - 하이브리드: NoSQL에서 처리된 데이터를 SQL로 저장
   - 프론트엔드의 요청을 처리하고 데이터를 처리 및 관리
   - 보안 관리
   - 서버와의 통신
4. 백엔드의 확장성
   - 로드 밸런싱: 트래픽을 분산하여 과부하 방지 및 가용성 확보
   - 캐싱: 자주 요청되는 데이터를 미리 저장하여 응답 속도 향상

▼ 3) JSON

- 특징
  - 가볍고 간단한 구조
  - 텍스트 기반: API 표준
  - 언어 독립적: 모든 언어 지원
  - 키-값 구조: (Key: Value)
  - 읽기/쓰기 용이: 가독성은 떨어지나 간단한 문법 → 보안한 문법: YAML
  - Object(객체): { }
  - Array(배열): [ ]
  - 단점: 스키마 부재, 주석 미지원

▼ 4) DOM

- 개념
  - 웹 페이지의 구조화된 표현, 객체
  - 계층적 트리 구조를 가지며 각 요소는 노드로 표현
- QuerySelector: DOM 요소 접근
- js로 요소 추가/삭제 및 이벤트 처리: (createElement(), appendChild(), addEventListener("click", function() {})...

▼ 5) 프레임워크

- 코드 흐름의 제어권을 프레임워크가 가짐 (제어 역전: Inversion of Control)
- 재사용 가능 코드 구조를 반복(구조화된 개발 패턴)하여 코드의 일관성, 효율성 증대
- 플러그인, 모듈로 확장 가능
- 현업에선 개발용, QA용, 테스트용 프레임워크를 분리해서 소스를 관리함
- 단점: 오버헤드(불필요한 기능 추가)

```
// React 컴포넌트
function UserProfile() {
  return <div>사용자 정보</div>;
}

// 내가 이 함수를 직접 호출하지 않음
// React가 필요할 때 자동으로 호출함
```

▼ 6) 라이브러리

- 코드 흐름 제어권을 사용자가 가짐
- 재사용 가능 코드 모음
- 모듈화
- 단점: 의존성(특정 프레임워크 요구)

```
// 내가 작성한 코드
function getUserData() {
  const result = axios.get('/api/user');  // 내가 필요할 때 라이브러리 호출
  return result;
}

getUserData();  // 내가 실행 시점 결정
```

▼ 7) API

- 응용 프로그램들이 서로 상호작용하기 위한 규칙이나 프로토콜
- 내부 로직을 몰라도 인터페이스를 통해 기능 사용 가능(추상화)
- 통신의 표준화, 재사용성
- REST API (Representational State Transfer)
  - 엔드포인트(URL): 어떤 리소스에 접근할지 결정
  - HTTP 메서드: 무슨 행위를 할지 결정
    - GET: 조회
    - POST: 생성
    - PUT/PATCH: 수정
    - DELETE: 삭제
  - 헤더: 메타데이터 (데이터 타입, 인증 정보)
    - Content-Type
    - Authorization
  - 바디: 실제 데이터 (주로 JSON)
  - 응답 상태 코드
    - 200번대는 성공
    - 400번대는 클라이언트 오류

        - 500번대는 서버 오류
    - 인증: 반드시 HTTPS 사용
        - API Key: 헤더나 URL에 고유한 키 포함
        - OAuth: 제 3자 인증에 사용
        - JWT: 서버가 발급한 토큰을 클라이언트가 보관하다가 요청마다 헤더에 포함
- GraphQL: 클라이언트가 필요한 데이터만 쿼리하는 방식
- gRPC: 구글에서 만든 고성능 API 프레임워크
- 종류
    - OS API
        - Windows API: 윈도우 앱의 시스템 제어
        - POSIX API: 유닉스 계열 시스템 제어
    - 기타
        - 라이브러리 API: 라이브러리 호출
        - 하드웨어 API: 프린터, 센서
- 단점
    - 네트워크 의존성 (속도 문제)
    - 버전 관리의 어려움
    - 보안 취약점 (설계 미흡 시)

▼ 8) IDE

- 주요 기능
    - 코드 편집
    - 디버깅
    - 통합 터미널
    - 버전 관리
    - 프로젝트 관리

▼ 9) HTML

- HyperText Markup Language
- 문서의 구조를 잡고 다양한 컨텐츠를 정의하는 역할
- Selectoring
    - 논리적 구조화, SEO