Frontend Study

> 0 week

Contents

1. 스터디 개요

스터디 팀장

- 윤상석

스터디 인원

- 9명

스터디 기간

- 3~4개월

스터디 학습 목표 1

- **최신 자바스크립트 (ES6~) 이해**
    i. 공식 문서 기반으로 가장 최신 문법 정리
    ii. 기념일 디데이 웹 애플리케이션 개발, 숫자 야구 웹 게임 만들기 (문법 응용)
    iii. 할일 목록 웹 앱 만들기 (DB 핸들링하기)
    iv. 지도 웹 애플리케이션 만들기 (API 핸들링하기)
    v. p5.js 맛보기 (JS로 인터렉티브 디자인 구현)

> JS를 사용하여 크롤링, 블록체인 구현, 데이터 분석, 알고리즘 등은 과제로 수행합니다.

스터디 학습 목표 2

- **React로 웹 앱 만들기**
    i. 공식 문서 기반으로 React의 기초 정리
    ii. React로 타이머 만들기 (React의 생명 주기 함수 이해)
    iii. React로 영화 정보 웹 SPA 개발 (API와 통신하기)
    iv. React로 JWT Token 관리하기 (백엔드와 통신하기)
    v. React로 블록체인(이더리움) 구현 또는 머신러닝 앱 구현

스터디 학습 목표 3

- **React Native로 모바일 앱 만들기**
    i. expo 기반의 크로스 플랫폼 모바일 앱 만들기
    ii. 몬드리안 만들기 (React-Native 레이아웃)
- **React와 여러 기술 융합 논의**
    i. 블록체인
    ii. 백엔드
    iii. 머신러닝
    iv. 모바일 앱
    v. VR

스터디 궁극적 목표

- 프런트엔드 기술과 스터디 구성원들의 기술을 합쳐서 융합형 **사이드** 프로젝트 시행.

  > 사이드 프로젝트(Side Project)란??
  > 주가 되는 업무가 아닌, 주가 되는 업무를 방해하지 않는 선에서 진행하는 다른 활동
  > 즐기면서 특정 목표를 가지고 노력하여 취미 이상을 꿈꾸는 활동

- 괜찮은 결과물이 나오면 실제 서비스 배포 및 공모전, 해커톤, 대회 참가 계획.

스터디 사전 지식

- 객체 지향 언어중 하나라도 쓸 수 있으면 됩니다.
  > Python, C++, Go, Java, JS, ...
- HTML에 대한 이해
- 간단한 CSS 개념

스터디 방식

리액트나 자바스크립트 가볍게 이론 위주로 학습하고자 하시는 분들도 계실 것이고
이론 학습을 넘어서 실제 개발 자체를 하시고 싶으신 분들이 계실 것 같습니다.

그래서 **오프라인 인원 3~4인**은 [개발 + 프런트엔드 이론], **온라인 인원**은 [프런트엔드 이론] 위주로 온프라인과 온라인을 동시에 진행할 생각입니다.

오프라인 3~4인 사람들끼리 모여 수업을 진행하고 **동시에** 화면 공유로 줌을 켜서 진행하려고 합니다.

- 2시간은 이론 강의이고(온+오프), 1시간은 개발 스터디(오프)로 진행.
- 주 1회 2시간(온라인 + 오프) + 1시간(오프) 동안 일요일 4시에 진행.
- 장소는 학교 근처 스터디 룸에서 진행.
- 중간고사, 기말고사 기간 2주는 쉽니다.

> 온라인 / 오프라인 참여 여부는 OT가 끝난 후에 설문지에서 체크

스터디 과제

- 매주 과제가 있습니다.
- 과제에 투자하는 시간은 적어도 8시간은 투자해야 합니다.

실습 및 개발 환경

- 코드 편집기 (IDE) : **Webstorm** (VSCode 사용하셔도 됩니다.)
- 브라우저 : FireFox, Chrome
- 소스 보관 : Github

스터디 회비

- 30000원 (수업준비 및 장소 대여비로 사용할 예정입니다.)

2. 기본적인 용어 정리

컴퓨터의 구성 요소

- CPU + 기억장치[RAM(메모리) / HDD, SSD] + 입출력 장치 + 시스템 버스

프로그램

- 어떤 문제를 해결하기 위해 컴퓨터에게 주어지는 처리 방법과 순서를 기술한 일련의 명령문의 집합체

OS

- 어떤 프로그램을 사용했을 때 그것을 작동시키는 환경
- 데스크탑 : Mac OS, Window OS, Ubuntu OS
- 모바일 : IOS, Android

Process (프로세스)

- OS가 CPU와 메모리 자원을 활용해서 실행시킨 프로그램

Thread (스레드)

- Process 내부에서 작동하게 되는 실행 단위.

Kernel (커널)

- OS의 핵심 부분으로 자원 관리와 메모리 제어 등을 담당

레거시 코드

- 실제로 에러가 나는 코드는 아니지만 트렌드를 따라가지 못하는 코드

리펙토링

- 레거시 코드를 현재 트렌드와 조직의 바뀐 개발 스타일에 맞게 수정하는 것

디버깅

- 코드에서 오류를 찾는 과정

라이브러리

- 특정 목적을 위해 개발자들이 미리 정리해 놓은 코드 뭉치.
- 예를 들어 파이썬의 numpy, 텐서플로우, JS의 React 등이 있다.

프레임워크

- 특정 프로그램을 개발하기 위한 여러 요소들과 메뉴얼을 제공하는 코드 뭉치.
- 예를 들어 파이썬의 웹 개발을 위한 프레임워크인 Flask, Django 등이 있다.

애플리케이션 (Application = APP)

- 기능이 있는 프로그램 (응용 프로그램)
- 흔히 말하는 앱 스토어에서 다운 받은 어플, 앱은 정확하게 모바일 애플리케이션이다. 넷플릭스, 인스타그램과 같은 프로그램을 웹 애플리케이션이라고 한다.

API (Application Programming Interface)

- 애플리케이션에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻한다.

인터넷

> 인터넷이란 전 세계 컴퓨터들을 하나로 연결하는 **거대한 컴퓨터 통신망**을 의미한다.

웹

> 인터넷을 기반으로 한, 정보를 공유, 검색할 수 있게 하는 서비스
> 웹 = World-Wide-Web = WWW = W3
> URL(주소), HTML(내용), HTTP(규칙)

브라우저

> 인터넷 상에서 웹에 연결시켜 주는 소프트웨어
> 사파이, 크롬, 파이어 폭스, ...

호스팅

- 서버 컴퓨터를 임대해주는 서비스.

IP 주소

- 인터넷상에 있는 컴퓨터의 고유한 주소. ex) 122.13.193.617 (https://ip.pe.kr)
- IP 주소를 통해 인터넷상의 한 컴퓨터에서 다른 컴퓨터로 데이터를 주고 받을 수 있다.

도메인

- 인터넷 사이트 주소.
- 숫자로만 구성된 아이피(IP) 주소의 단점을 보완하기 위해 사용한다.

DNS

- 인터넷 도메인 이름들의 위치를 알아내기 위한 IP 주소로 바꾸어주는 시스템.

URL

- 인터넷에서 어느 사이트에 접속하기 위해서 입력해야 하는 주소를 포함한 일련의 문자.
- 맨 앞에 `http://'를 입력하고 다음에 해당 사이트의 주소를 표시한다.

프로토콜

- 복수의 컴퓨터 사이에서 데이터 통신을 원활하게 하기 위해 필요한 통신 규약.

HTTP (HyperText Transfer Protocol)

- 웹 상에서 정보를 주고받을 수 있는 프로토콜. 주로 HTML 문서를 주고받는 데에 쓰인다.
  HTTPS는 HTTP에서 보안이 강화된 것이다.

TCP

- 데이터 전송을 제어하는 프로토콜

웹의 동작 원리

링크

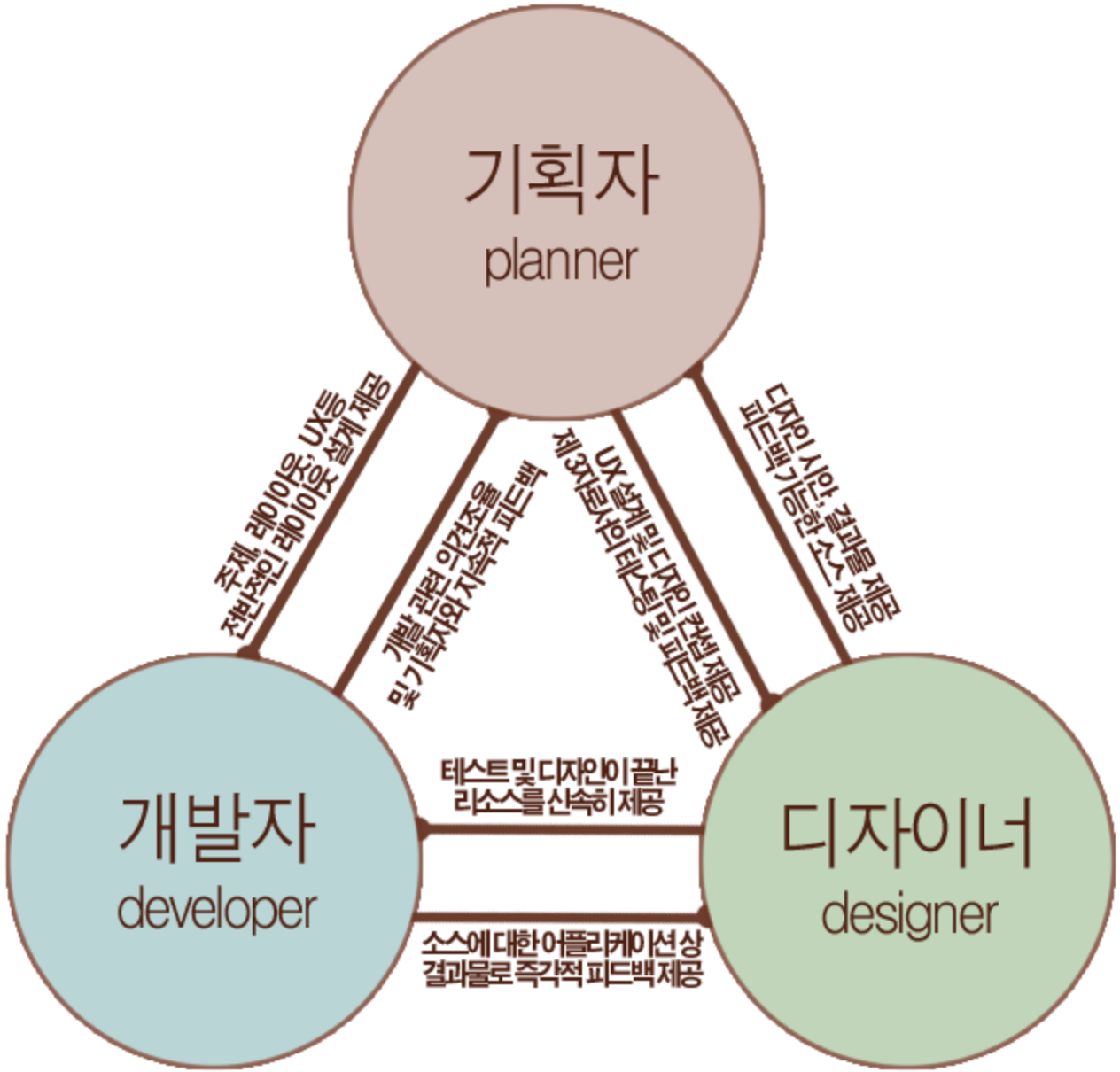
3. 소프트웨어 개발의 큰 그림

소프트웨어란??

컴퓨터 = 하드웨어 + 소프트웨어

- 하드웨어란?? -> 딱딱한 것
- 하드웨어의 종류
  - 데트크탑의 키보드, 마우스, CPU, ...
  - 노트북의 키보드, 트랙패드, CPU, 스피커, ...
  - 패드의 터치 스크린, ...
  - 스마트폰의 터치 스크린, 지문 인식 장치, ...
  - 스마트 워치의 터치스크린, 생체 인식 장치, ...
- 하드웨어만 있는 컴퓨터는 고철 덩어리에 불과합니다.

소프트웨어란?? -> 하드웨어에게 명령을 하는 역할을 합니다.


기획자
planner
개발자
developer
디자이너
designer
주제, 레이아웃, UX등
전반적인 레이아웃 설계 제공
개발 관련 의견조율
및 기획자와 지속적 피드백
UX 설계 및 디자인 컨셉 제공
제3자로서의 테스팅 및 피드백 제공
디자인 시안, 결과물 제공
피드백 가능한 소스 제공
테스트 및 디자인이 끝난
리소스를 신속히 제공
소스에 대한 어플리케이션 상
결과물로 즉각적 피드백 제공

소프트웨어공학의 과정

정의 과정			개발 과정			유지보수 과정		
무엇(What)			어떻게(How)			변경		
시스템 분석	프로젝트 계획 수립	요구사항 분석	설계	구현	테스트	수정	적응	기능 향상
-개별 요소의 역할 정의 -소프트웨어가 수행해야 할 역할 할당	-위험 분석 -자원 할당 -비용 추정 -작업내용과 일정 결정	- 개발방향 제시	-소프트웨어에 대한 요구 사항을 자료구조,알고리즘적 절차, 인터페이스의 특성을 묘사하는 일련의 표현으로 변환	-설계된 내용을 프로그래밍 언어로 변환	-소프트웨어의 기능적,논리적 구현에서의 결함을 발견하기 위한 테스트	-기존의 결함이 수정 되도록 유지보수	-외부적인 환경변화를 수용하도록 변경	-본래의 요구된 기능을 능가하도록 확장

소프트웨어 설계 순서

1. 기획 요구사항 정의 (기획자)
2. 요구사항에 따른 개발 견적 완성 (개발자)
3. 요구사항 명세서 & 스토리보드 작성 (디자이너,기획자)
4. 클래스 & 함수 설계 (개발자)
5. 스토리보드 최종 수정 & 디자인 파일 첨부 (기획자,디자이너,개발자)
6. 개발 1차 완성 (개발자)
7. 무한 검토 (개발자,기획자,디자이너)
8. 배포

소프트웨어 개발 분야

1. Frontend Developer
   > 디자인을 화면 프런트에 구현 + API 핸들링
2. Backend Developer
   > API 개발 + DB 핸들링
3. DevOps
   > 소프트웨어의 개발과 운영 : 배포 & 유지보수 & 업데이트
4. Data Scientist

개발 로드맵 링크

Frontend Dev

> HTML + CSS + JavaScript

- 웹 디자인과 UI / UX 디자인을 실제 웹에 구현하는 역할이다.
- **HTML, CSS, JS** 를 기본으로 추가적인 라이브러리 또는 프레임워크를 사용하여 웹을 구현한다.
- CSS 라이브러리 또는 프레임워크 : TailwindCSS, SASS, ...
- JS 라이브러리 또는 프레임워크 : React, Vue, Angular, ...

Backend Dev

- 클라이언트에 필요한 데이터를 핸들링하며 필요한 API를 개발한다.
- 개발 언어 & 프레임워크
  - nodeJS(express), Java(spring), Python(Django, Flask), Go(echo), ...
- API 개발
  - Restfull API, Server API (GraphQL), ...
- 데이터베이스
  - MySQL, MongoDB, Firebase, MariaDB, ...

4. 자바스크립트의 역사

링크

JavaScript = JS

1. JS는 기본적으로 브라우저 위에서 동작

- 초창기 JS는 웹 페이지의 보조적인 기능을 수행하기 위해 한정적인 용도로 사용되었다.
- JS의 각각의 파생 버전은 브라우저마다 다르게 동작

ECMAScript = ES

2. ES 등장

- ES는 모든 브라우저에서 동일하게 동작하는 표준 JS이다.
- ES는 매년 버전업 되면서 관리되고 있다. ES5, ES6, ...
- ES6에서 범용 프로그래밍 언어로서 갖춰야할 기능들이 도입되었다.
- **보통 JS라고 하면 ES를 의미한다.**

3. AJAX 기술 등장

- 부분 렌더링이 가능하게 되었다.
- JS를 이용해서 비동기적(Asynchronous)으로 서버와 브라우저가 데이터를 교환할 수 있는 통신 기능인 AJAX(Asynchronous JavaScript and XML)가 **XMLHttpRequest**이라는 이름으로 등장했다.
- **XMLHttpRequest**는 후에 **Fetch API**가 대체하게 된다.

4. JS의 라이브러리인 JQuery 등장

- DOM을 쉽게 제어할 수 있게 되었다.
- 여기서 DOM은 문서 객체 모델이다.

5. Nodejs의 등장

- 브라우저에서만 동작하던 JS를 **브라우저 이외의 환경에서 동작**시킬 수 있는 JS 실행 환경인 Node.js의 등장으로 JS는 웹 브라우저를 벗어나 서버 사이드 애플리케이션 개발에서도 사용되는 범용 프로그래밍 언어가 되었다.
- JS로 백엔드 개발(Express), 모바일 앱 개발(React-Native), 데스크탑 앱 개발(Electron), 머신러닝(Tensorflow)도 가능하게 되었다.

6. JS의 라이브러리인 React.js의 등장

- JQuery 대체
- SPA(Single Page Application) 개발 가능

7. TypeScript 등장

- 엄격한 문법의 JS
- 컴파일 언어

5. 자바스크립트 언어 특징

인터프리터 언어

동적 타입 언어

> 자바스크립트는 변수 타입이 없다.
> 프로그램을 실행하는 도중에 변수에 저장되는 데이터 타입이 동적으로 바뀔 수 있다.

프로토타입 기반 언어

> 클래스 기반 언어에서는 '상속'을 사용하지만 프로토타입 기반 언어에서는 어떤 객체를 원형(prototype)으로 삼고 이를 복제(참조)함으로써 상속과 비슷한 효과를 얻는다.

함수는 일급 객체

> JS의 함수는 객체이며, 함수에 함수를 인수로 넘길 수 있다.
> 이 특성을 활용해 고차 함수를 구현할 수 있어 함수형 프로그래밍이 가능하다.

과제

1. 온라인 / 오프라인 참석 여부 설문
2. JS의 역사 살펴보기 -> 링크
3. Webstorm IDE 기반의 개발 셋팅

> 개발 환경 구축 : https://youtu.be/Odd2IbMt-e8
> HTML 이론 정리 https://youtu.be/oOC6aZcTjrU