Missing Semester 실습 스크립트

소개

시간이 다 된 것 같으니 개발환경 라이브실습을 시작해보도록 하겠습니다.
안녕하세요 저는 현재 멋쟁이 사자처럼 10기 운영진 소속 박경준이라고 합니다.

금일 진행되는 라이브 실습은 기본적으로 구름 온라인 강의로 업로드 되어 있는 내용 중
실습이 필요할 것 같은 부분을 함께 진행해보는 식으로 할 예정입니다.

이번 실습에 MacOS 시스템 설정부터 터미널 설정까지 개발환경 세팅에 필요한 부분을 함께 세팅해보고 많은 개발자들이 애용하는 에디터인 비주얼 스튜디오 코드를 활용하는 방법과 유용한 익스텐션들을 직접 사용해보도록 하겠습니다.

질문사항은 선들고

개발환경 세팅

라이브실습 첫 번째 내용은 맥 OS 개발환경 세팅입니다.

기본적인 맥 키보드 설정, 미션 컨트롤 등의 설정은 강의를 보고 직접 따라해보실 수 있는 수준이라고 판단되어 라이브 실습 내용으로는 제외하였습니다. 각자의 취향에 맞게 잘 설정해주시면 좋을 것 같습니다.

파인더 설정

맥 개발환경 세팅의 첫 번째 순서로는 시스템 설정입니다.
온라인 강의에도 설명 되어있듯 디렉토리 구조가 기본적으로 맥 유저 이름으로 된 디렉토리 기준으로 다양한 파일들이 가지치기 되는 형태이기 때문에
새로운 파인더 윈도우를 열었을 때 루트 폴더를 유저네임 폴더로 설정해두어야 로드가 더 빠릅니다.

파인더 기본 폴더를 설정하는 방법은 다음과 같습니다.

1. 파인더를 엽니다.
2. 파인더의 환경설정에 들어갑니다.
3. 파인더 환경설정의 일반 탭 ➜ 새로운 Finder 윈도우에서 보기를 유저 네임 디렉토리로 설정합니다.

백쿼트 입력

다음 설정은 맥에서 원화 입력을 백쿼트 입력으로 바꾸는 설정입니다.

맥 OS에서는 백쿼트 키가 한글일 때에는 원화로, 영문일 때에는 백쿼트가 입력됩니다. 자바스크립트 코딩 시에 백틱 키를 자주 사용하게 되는데 이때 원화가 입력되는 것은 상당히 귀찮은 일이 됩니다.

하지만 저희는 기본 설정으로 원화 입력을 백쿼트 입력으로 바꿀 수 있습니다.

1. Finder를 열고 자신의 맥 폴더 ➜ 라이브러리 폴더로 이동합니다.
2. KeyBindings 폴더를 생성하고 폴더 내에 DefaultKeyBindings.dict 파일을 생성합니다.
3. 키 바인딩스 폴더가 라이브러리 내에 존재하지 않으면 새로 생성하면 됩니다.
4. 참고로 맥에서는 우클릭으로 빈 파일을 생성할 수 있는 메뉴를 제공하지 않기 때문에 텍스트 에디터로 생성하면 문제없이 진행할 수 있습니다.

재부팅을 하게 되면 한글 또는 영문에 상관 없이 백쿼트만 입력되는 것을 확인하실 수 있습니다.

```
{
    "₩" = ("insertText:", "`");
}
```

위와 같은 과정을 직접 진행해도 잘 적용되지만
깃헙 gist를 이용한 자동화 쉘 스크립트를 활용하면 코드 한줄로 모든 작업이 처리됩니다.

```
curl -sSL https://gist.githubusercontent.com/redism/
43bc51cab62269fa97a220a7bb5e1103/raw/0d55b37b60e0e0bd3d0d7f53995de0a722f9820c/
kr_won_to_backquote.sh | sh
```

다음 코드를 터미널 상에 입력해보세요.

터미널 상에 Done이라고 뜨면 정상처리된 것입니다. 저는 이미 존재해서 Already exists라는 문구가 뜨네요.

Homebrew 설치

다음은 홈브루를 설치해보겠습니다.
홈브루 설치에 앞서 홈브루가 뭔지 아직 잘 모르는 분들을 위해 다시 한 번 설명 드리겠습니다.

홈브루(Homebrew)는 맥 OS의 패키지 매니저입니다. 윈도우에서 깃, 파이썬, mySQL등 여러 프로그래밍 언어나 어플리케이션을 설치할 때 수동으로 설치하는 경우가 대부분이지만 맥에서 홈브루 패키지 매니저를 통해 터미널 상에서 편리하게 설치할 수가 있게 됩니다. 오픈소스에 대한 접근까지도 쉽게 이루어집니다.

홈브루를 설치하는 명령어는 다음과 같습니다.
…

(전체 설치 7분 소요)

설치 중에 잠깐 설명을 드리자면,
패키지 매니저란 컴퓨터 OS를 위해 일정한 방식으로 컴퓨터 프로그램의 설치, 업그레이드, 구성, 제거 과정을 자동화하는 소프트웨어 도구들의 모임입니다. 소프트웨어에 대한 수동 설치 및 수동 업데이트 필요성을 근절하기 위해 설계되었습니다.

일반적으로 저희가 맥으로 어플리케이션을 다운로드 받을 때 겪게되는 과정들을 (웹사이트로 가서 dmg파일을 받고 어플리케이션 위치를 앱으로 직접 이동시켜주는 등) 자동화시켜준다고 생각하시면 됩니다.
이후 여러 소프트웨어 설치 시 홈브루 패키지매니저를 이용하기 위해서는 brew 키워드를 붙여 사용하면 됩니다.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

설치가 아직 되지 않으신 분이 계실까요?

넵 1분만 더 기다리다가 우선 진행하도록 하겠습니다.

설치 완료 후에 brew doctor를 입력했을때 Your System is ready to brew라는 메세지가 출력되면 정상적으로 브류 설치가 완료된 것입니다.

git 설치

다음은 버전관리 시스템인 깃을 홈브루를 이용하여 설치합니다.
git-lfs는 깃헙에서 다루지 못하는 대용량 파일을 다룰 수 있게 해주는 오픈소스 소프트웨어입니다. 자세한 사용법은 다음 링크를 참조해주세요.

깃과 관련된 강의는 추후 다른 강의에서 자세히 다룰 예정이니 자세한 설명은 생략하고
기본적인 설정만 하고 넘어가도록 하겠습니다.
(설치 약 20초 소요)

설치가 완료 되었으면 깃 컨피그 명령어를 통해 사용자명과 이메일을 등록합니다.
설정파일 확인을 위해 `git config --list` 를 입력합니다.

Cask 설치

다음 캐스크 설치입니다.

cask는 홈브루 익스텐션입니다. 비주얼 스튜디오 코드 익스텐션과 비슷한 개념이라고 생각하시면 될 것 같습니다.
기존 홈브루에서는 없는 기능인 Mac전용 GUI 프로그램 설치를 지원합니다.

cask 설치가 완료 되었으면 cask를 활용해 파이어폭스 브라우저를 설치해보겠습니다.
홈브루 공식 사이트의 Browse All Casks 메뉴에 들어가 파이어폭스를 검색한 뒤
나타나는 명령어를 복사하여 터미널에 입력하면 됩니다.

2분 소요

설치가 완료되면 Moving App firefox.app~~ 과 같은 문구가 출력됩니다.
출력된 문구에서 볼 수 있듯이 캐스크를 통해 설치한 앱은 별도의 GUI 내 작업 필요 없이
모든 설치 과정이 자동화 됩니다. 기존 설치 과정에서는 설치 후 앱을 앱 디렉토리로 드래그하여 옮기는 작업도 있었지만 터미널 내에서 자동으로 모든 작업이 처리되었었죠.

터미널 설정 - ITerm2

다음으로는 터미널 설정을 해보겠습니다. 맥을 쓰는 이유가 아름다운 개발환경 구축에 있다고 해도 과언이 아니죠.
이를 위해 필수적이라고 할 수 있는 ITerm2를 설치합니다.

다음의 명령어를 입력해주세요.

설치된 아이템2를 실행하시면 UI자체는 더 깔끔하지만 투박한 색 조합을 더 보기좋게 꾸밀 수 있는데요, 다양한 테마가 존재하지만 저는 snazzy 테마를 적용해보겠습니다.
(GitHub - sindresorhus/iterm2-snazzy: Elegant iTerm2 theme with bright colors)

스내찌 테마를 배포하는 깃허브 레포지토리에 들어갑니다. 짚파일을 다운로드 한 뒤 압축을 풀면 내부에 itemcolors라는 확장자를 갖는 파일이 있습니다. 해당 파일을 더블클릭하게 되면 iterm2가 열리면서 컬러 스키마가 임포트되었다는 알림이 뜹니다.

컬러 스키마가 등록만 된 것이고 실제 적용하기 위해서는 아이템2의 설정에 들어가야합니다.

터미널 설정 - oh-my-zsh 설치

오마이제트쉘 설치에 앞서 제트쉘 설치를 진행합니다.

맥은 기본 터미널 세팅이 bash, 배시 쉘로 설정되어 있습니다. 기본 설정된 쉘 스크립트가 무엇인지 확인하기 위해서는 echo $0 명령어를 입력하면 됩니다.

배시 쉘을 zsh로 변경하기 위해서는 zsh 명령어만 입력하면 됩니다. 만약 zsh 설치가 되어 있지 않았다면 No such file or directory 경고가 나타납니다. 홈브루를 통해 설치할 수 있습니다. (설치 화면으로 넘기기)

그렇다면 제트쉘과 배시 쉘의 차이점이 무엇일까요?
두 쉘의 차이점은 크게 두 가지로, 하나는 디렉토리 이동 방식이고 또 하나는 숨김폴더 표기 여부입니다.

절대경로를 통해 홈 디렉토리로 이동하는 방법에 차이가 존재합니다. 배시 쉘은 직접 입력하여 이동해야 하는 반면, zsh는 홈 디렉토리 경로가 /home/ubuntu라고 가정했을 때 cd /h/u를 입력한 뒤 탭 키를 입력하면 절대경로를 자동완성 해줍니다.

```
pwd로 홈 디렉토리 위치 확인


cd /u/i (users/imac) + Tab
```

방금 소개드렸던 자동완성 기능이 홈 디렉토리 뿐만 아니라 전체 디렉토리에 적용됩니다. 다만 디렉토리 첫 글자와 중복되는 디렉토리들이 몇 가지 존재하게 될텐데, 이러한 경우 탭 키를 여러번 눌러서 다른 디렉토리를 선택할 수 있습니다.

첫 글자 중복되는 폴더 골라서 탭키 입력…… 실습하기

zsh상에서 디렉토리 위치 변경이 편리한 점은 현재 내가 속한 디렉토리의 위치와 동등한 위치의 디렉토리로의 위치변경이 쉽습니다. 잠시 디렉토리 구조를 보면,
다음과 같은데, Users 디렉토리 하위에는 A, B 디렉토리가 존재합니다. cd /u/p/A 자동완성 기능을 통해 A 디렉토리로 이동한 뒤 배시 상에서 B 로 이동하기 위해서는
cd .. , cd B 명령어를 입력해야하지만 zsh에서는

cd (현재 디렉토리명) (이동할 디렉토리명)# 을 입력하면 됩니다.

이와 같이 제트쉘이 편리한 기능을 다수 보유하고 있는데요, 이제 설치하게 될 오 마이 제트쉘은 이 제트쉘 환경설정을 다루는 프레임워크이며 다양한 테마와 플러그인들을 가지고 있습니다.

오 마이 제트쉘 설치 명령어는 다음과 같습니다.

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/
```

```
install.sh)"
```

3분정도 시간 부여

설치 후 온라인 강의에서 언급된 powerlevel10k 테마를 적용해보겠습니다.

```
git clone https://github.com/romkatv/powerlevel10k.git ~/powerlevel10k

source ~/powerlevel10k/powerlevel10k.zsh-theme
```

다음 두 명령어를 입력하게 되면 테마 옵션을 선택하게 됩니다. 명령에 따라 진행하게 되면 최종적으로 oh-my-zsh 파워레벨10k 테마가 적용됩니다.

파워레벨10k가 아닌 다른 테마를 적용하고 싶으시다면 zshrc 파일을 수정하면 됩니다.
vim 편집기를 이용하여 zshrc 편집 모드에 진입하여
ZSH_THEME이라고 되어있는 부분을 바꾸고싶은 테마 이름으로 변경하면 됩니다.

현재 powerlevel 10k 테마가 적용되어 있으므로 위와 같이 테마가 저장되어 있는데,
이를 agnoster로 변경한 뒤 source명령어로 적용해보겠습니다.

`p10k configure`

제트쉘 플러그인

다음으로 제트쉘에 플러그인을 설치해보겠습니다.
git을 통해 레포지토리를 다운로드 받습니다. 제트쉘 플러그인을 관리하는 디렉토리에 다운받는 명령어입니다.

파일 다운로드 후 zshrc 파일에 들어가 플러그인 목록을 찾습니다. 슬래시 plugins 입력 후 엔터를 누르면 찾을 수 있습니다.

oh-my-zsh 테마 및 유용한 플러그인 정리

```
git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions
```

플러그인 목록에 zsh-autosuggestions를 추가합니다.

다음 플러그인으로는 syntax highlighting이 있습니다. 이 플러그인은 올바른 커맨드의 색을 변경하여 알려줍니다.

```
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
```

위 명령어로 플러그인을 설치하고
다음 명령어를 추가로 입력하면 됩니다. 직접 파일을 열어서 추가해도 되지만 플러그인 목록에 추가할 필요 없이 다음 코드를 파일 맨 마지막에 추가만 해도 되는 부분이라 다음 명령어를 입력하면 됩니다.

```
echo "source ~/.oh-my-zsh/custom/plugins/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh" >> ${HOME}/.zshrc
```

추가적인 플러그인들은 구글링을 통해 더 추가해보시면 될 것 같습니다.

Vscode

다음은 비주얼 스튜디오 코드와 익스텐션 활용에 대해 실습을 진행해보도록 하겠습니다.
먼저 vscode가 설치되어 있지 않으신 분들이 계시다면 사이트에 들어가 설치를 진행해주시기 바랍니다.

설치를 아직 하지 않으신 분 있으시면 손좀 들어주시면 감사하겠습니다 (1분)

네 설치가 완료된 분들께서는 비주얼 스튜디오 코드를 열어주시구요
코딩 시 유용한 단축키를 직접 활용해 보기 위해 html파일을 하나 생성해보도록 하겠습니다

먼저 첫 번째로 볼 단축키는 명령팔레트 단축키입니다.
명령팔레트 단축키는 브이에스 코드가 다루는 모든 기능에 쉽게 접근할 수 있게 해주는 단축키입니다. 다양한 기능들을 추후 다루시게 될 때에 이 단축키를 활용하시면 좋습니다.

다음은 퀵 오픈 단축키입니다.
퀵 오픈 단축키는 좌측 워크스페이스에서 하나하나 파일을 스크롤하여 찾아볼 필요 없이 빠르게 접근할 수 있는 단축키입니다. 프로젝트 규모가 커질 경우 편리하게 이용하실 수 있습니다.

다음으로는 세팅을 여는 단축키입니다. 커맨드 + 콤마이고 좌측 하단에 톱니 버튼 - preference로 설정을 열어도 되지만 단축키를 통해 쉽게 접근하실 수 있습니다.

다음으로는 사이드바 토글 단축키입니다. 커맨드 + B로 할 수 있고 좌측 워크스페이스를 숨길 수 있습니다. 이 단축키와 위에서 소개드렸던 퀵 오픈 단축키를 활용하면 더 확장된 화면으로 개발을 편리하게 진행하실 수 있습니다.

다음으로는 터미널 토글 단축키입니다. 깃 관련 작업을 하게되면 앞으로 터미널을 자주 사용하시게 될텐데, 이때 터미널 토글 단축키를 사용하면 쉽게 열고 닫을 수 있습니다 .

VScode 단축키 - 코드편집

지금까지의 비주얼 스튜디오 코드 단축키는 에디터 자체를 다루는 단축키 모음이었다면
지금부터 소개드릴 단축키는 코드편집 단계에서의 단축키를 소개드리도록 하겠습니다.

커서 이동의 경우
커맨드 + 위 아래 방향키를 통해 파일 맨 위 아래로 이동할 수 있습니다. 또한
맥 기준 옵션 키 (알트 키) 에 좌/우 입력으로 단어 기준 커서 이동을 할 수 있습니다.

Vscode에서는 코드 편집에 유용한 단축키가 여럿 있습니다.

1. 코드 라인 한 줄을 다른 위치로 이동시키기 위해 복사 붙여넣기 후 지우는 과정 없이 맥 기준 옵션 키 (윈도우 기준 알트) 키 + 상 하 키로 코드라인 한 줄을 이동할 수 있습니다. 이에 대한 응용으로 여러 코드 줄을 묶어서 한 번에 이동할 수도 있습니다.
2. 옵션 + 쉬프트 + 상/하키 조합으로 코드라인을 복사하여 바로 아래에 붙여넣을 수 있습니다. 위쪽 방향키 단축키는 커서를 유지시키고 아래쪽 방향키 단축키는 커서를 한 단계 내립니다. 이 또한 코드라인 여러 줄을 복사할 수 있습니다.
3. 커맨드 + 엔터로 다음 라인에 빈 줄을 삽입할 수 있습니다. 보통 코드 맨 끝에 커서를 위치시킨 뒤 엔터 클릭으로 빈 줄을 삽입하지만 그럴 필요 없이 코드 작성 중간에 서도 빈 라인을 삽입할 수 있습니다.
4. 커맨드 + D로 중복되는 코드를 동시에 선택할 수 있고 편집 또한 동시에 처리할 수 있습니다.
5. 커맨드 + U로 바로 전에 작업하던 커서로 이동할 수 있습니다.
6. 옵션 키 (알트 키)를 누른 채로 마우스 왼쪽 클릭을 통해 동시 작업 커서를 추가할 수 있습니다. 한 번에 여러 코드를 편집할 때 편리합니다.
7. 알트 + 쉬프트 + 마우스 드래그로 멀티 커서를 한번에 추가할 수도 있습니다.
8. 알트 + 쉬프트 + I 입력으로 선택한 코드 박스 끝에 커서를 추가합니다.

위의 단축키만 잘 숙지하셔도 비주얼 스튜디오 코드를 활용한 개발은 편리하게 진행하실 수 있으실 겁니다.

Vscode 익스텐션

다음으로 비주얼 스튜디오 코드의 각종 익스텐션들을 설치하고 활용해보도록 하겠습니다.
비주얼 스튜디오 코드의 경우 오픈소스에 단순한 텍스트 에디터이지만 기존 IDE와 맞먹을 정도의 플러그인과 익스텐션 생태계로 인해 많은 사랑을 받고 있습니다.

개발 시 눈이 침침한 저희들을 위해 브이에스 코드는 다양한 테마를 제공합니다.
메터리얼 띰과 나이트 오울 테마 두개 모두 사용해보았지만 개인적으로 다크모드 중심으로 이루어진 나이트 오울 테마가 개발 시에 눈이 편안했습니다. 직접 골라서 맘에 드는 테마를 사용해보시면 되겠습니다.

다음으로는 메터리얼 아이콘 테마 입니다. 이 익스텐션이 없을 때에는 스튜디오 코드 좌측 워크스페이스에 파일 아이콘이 투박한 형태로 나타나지만 익스텐션 설치 후에는 어떤 언어로 작업하고 있는지 알기 쉽게 아이콘이 등록되어 나타납니다.

다음은 프리티어 입니다.
프리티어 포맷터는 코드 작업 완료 후 커맨드 + S 로 파일 저장을 했을 때
미리 지정해둔 양식대로 자동 포맷팅을 해주는 익스텐션입니다.

프리티어 익스텐션 적용을 위해서는 몇 가지 작업이 필요합니다.

1. 우선 프리티어 익스텐션을 설치합니다. (대기)
2. 커맨드 + 콤마로 세팅으로 이동합니다.
3. 검색창에 format on save를 검색하여 체크박스를 클릭합니다.
4. prettier를 검색하여 탭 width가 4로 되어있는 분들은 2로 지정합니다.
5. 비주얼 스튜디오 코드를 껐다 켜보신 뒤에 적용이 되지 않으신 분들은 Default Formatter가 프리티어로 설정되어 있는지 확인해보시기 바랍니다. 다시 한번 껐다 켜보세요.

프리티어는 혼자 작업 시에도 편리하지만 실제 협업 과정에서 각자의 코딩 컨벤션 스타일이 다르기 때문에 이를 통일하기 위한 작업이라고 보시면 좋습니다. 프로젝트 이전에 .prettierrc라는 파일을 생성하여 깃으로 서로 주고받으며 협업하게 됩니다.

마무리

지금까지 맥 개발환경 세팅과 비주얼 스튜디오 코드 활용, 익스텐션 설치까지 실습을 진행해보았는데요
늦은 시간까지 라이브 실습에 참여해주셔서 감사드리고
질문이 있으신 분들은 남아서 질문해주시면 감사하겠습니다.