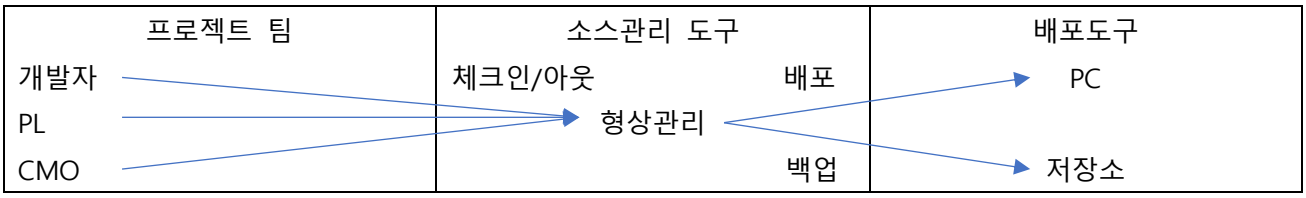
버전 관리 시스템(VCS : Version Control System) : 형상관리



버전 관리 시스템의 종류

크게 클라이언트-서버 모델과 분산 모델로 나눈다.

<클라이언트-서버 모델>

하나의 중앙 저장소를 공유한 후 각각의 클라이언트(개발자)는 저장소의 일부만 갖는 형태이다. 다시 말하면 개발자가 자신이 작업하는 부분만 로컬에 임시로 저장한 후 작업하는 형태이다. (CVS)

이 모델은 중앙 저장소에서 프로젝트 관리의 모든 것을 처리한다. 따라서 클라이언트(로컬)에서 할 수 있는 것이 파일 수정과 서버로의 커밋(commit)등 많지 안다. 만약 서버가 고장이 난다면 불안전한 로컬의 파일 사본들만 남게 된다.

<분산 모델>

프로젝트에 참여하는 모든 클라이언트(개발자)가 전체 저장소에 대한 자료를 개별적으로 로컬 저장소에 저장하고 작업하는 형태이다. (git, subversion, Mercurial)

서버 뿐만 아니라 프로젝트에 참여하는 모든 컴퓨터 각각에 로컬 저장소가 있으므로 모든 사용자의 로컬 저장소에도 반영될 수 있다.

<Git의 특징>

Git은 master 저장소 서버와 master 저장소의 완전한 사본을 가지는 클라이언트 저장소로 구성되어 있다.

여기서 저장소는 하나의 프로젝트라고 이해하자.

- 로컬 및 원격 저장소 생성
- 로컬 저장소에 파일 생성 및 추가
- 수정 내역을 로컬 저장소에 제출
- 파일 수정 내용 추적
- 원격 저장소에 제출된 수정 내역을 로컬 저장소에 적용
- master에 영향을 끼치지 않는 브랜치 생성

- 브랜치 사이의 병합(Merge)

- 브랜치를 병합하는 도중의 충돌 방지

2명 이상의 팀 프로젝트 개발에서 협업하는데 있어 가장 중요한 팀원 사이의 버전 맞춤, 할당된 작업의 결과물 관리, 특정 결과물이 누구의 것인지 추적하는 것이 Git을 이용함으로써 가능하다.

<원격 저장소와 GitHub>

Git 기반의 원격 저장소를 제공하는 대표적인 서비스가 GitHub이다.

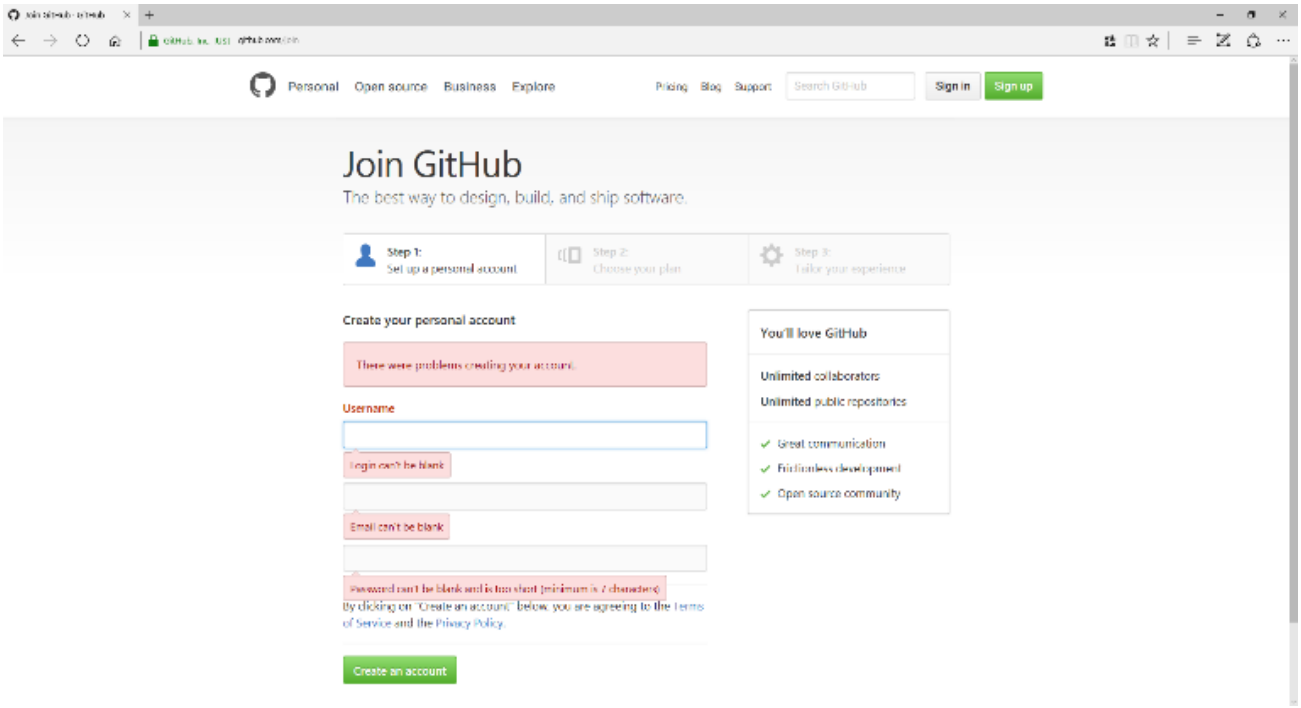
- 전 세계에서 진행되는 오픈 소스 프로젝트가 많이 모여 있어 이에 참여하고 오픈 소스에 기여할 수 있는 기회가 있다.

- 개발자는 GitHub를 이용해 자신이 작성했던 코드를 곧바로 제공할 수 있다.

- 개발자, 기획자 및 디자이너들이 포트폴리오나 기획 문서를 공개할 수 있다.

<GitHub 가입>

https://github.com/



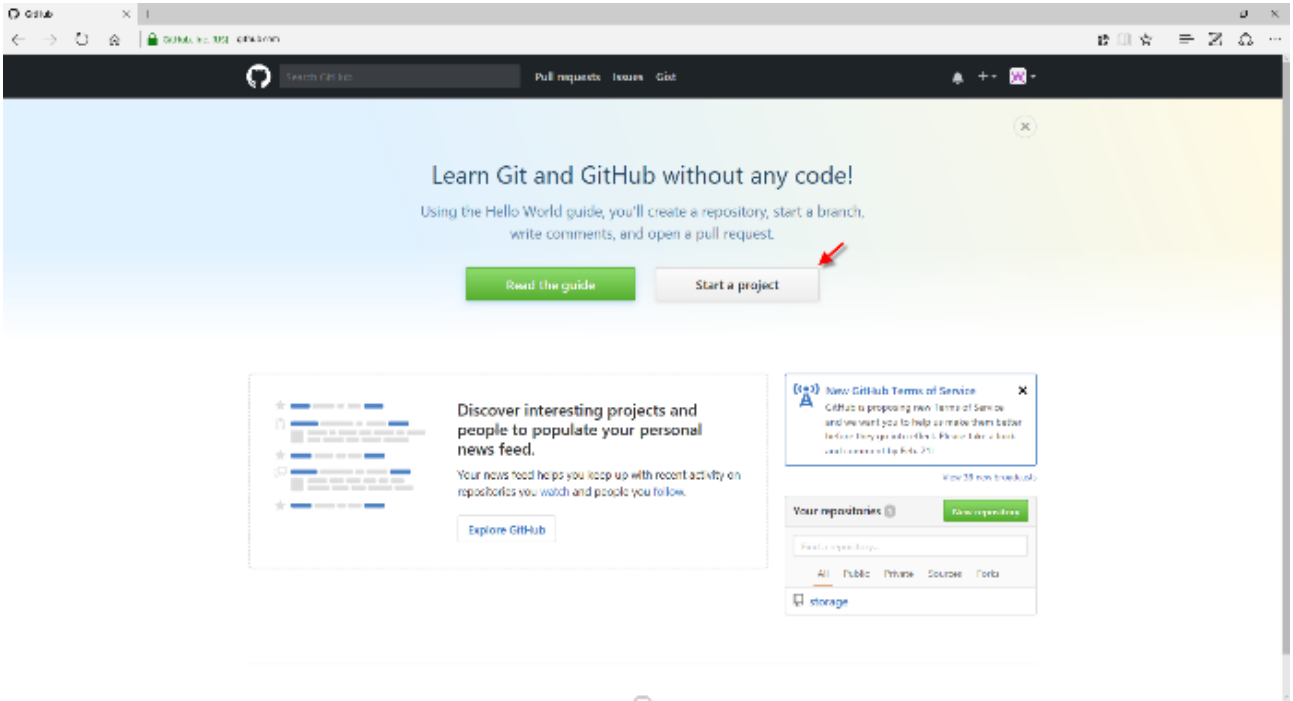
가입 후에는 GitHub의 계정 유형(플랜)을 설정한다. 무료로 이용하려면 GitHub에서 생성하는 모든 저장소는 공개 저장소가 된다.

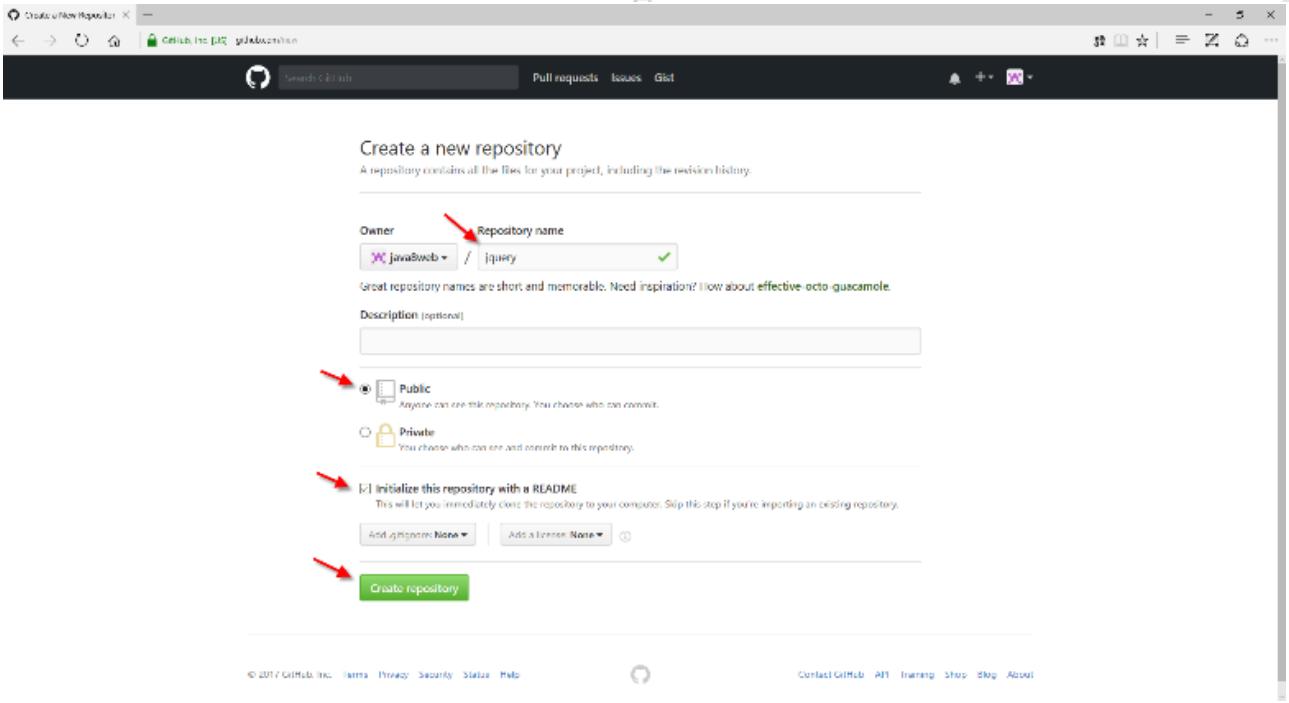
"Free" 항목을 그대로 두고 "Finish sign up"를 클릭해 다음단계를 진행한다.

"Help me set up an organization next" 체크 박스는 협업할 때 팀을 만들어서 활동하도록 설정하는 것이다.

다음 단계로 "Email" 항목이 나타난다. 가입한 계정이 맞으며 발송된 메일에 첨부된 링크는 GitHub에 가입했다는 것을 인증하는 하이퍼링크를 클릭하여 계정을 인증한다. 가입한 계정을 인증하는 웹 브라우저가 실행되며 가입한 이메일 수조가 맞는지 확인하고 "Confirm"을 클릭한다.

sign in 후 저장소 생성





- Owner : 사용자의 아이디, 협업 환경에서는 다른 사용자의 아이디를 지정할 수 있다.

- Repository name : 새로 생성할 원격 저장소 이름을 입력한다. 가능하면 로컬 환경에서 작업할 Git 프로젝트 디렉토리 이름과 같게 하는 것이 좋다.

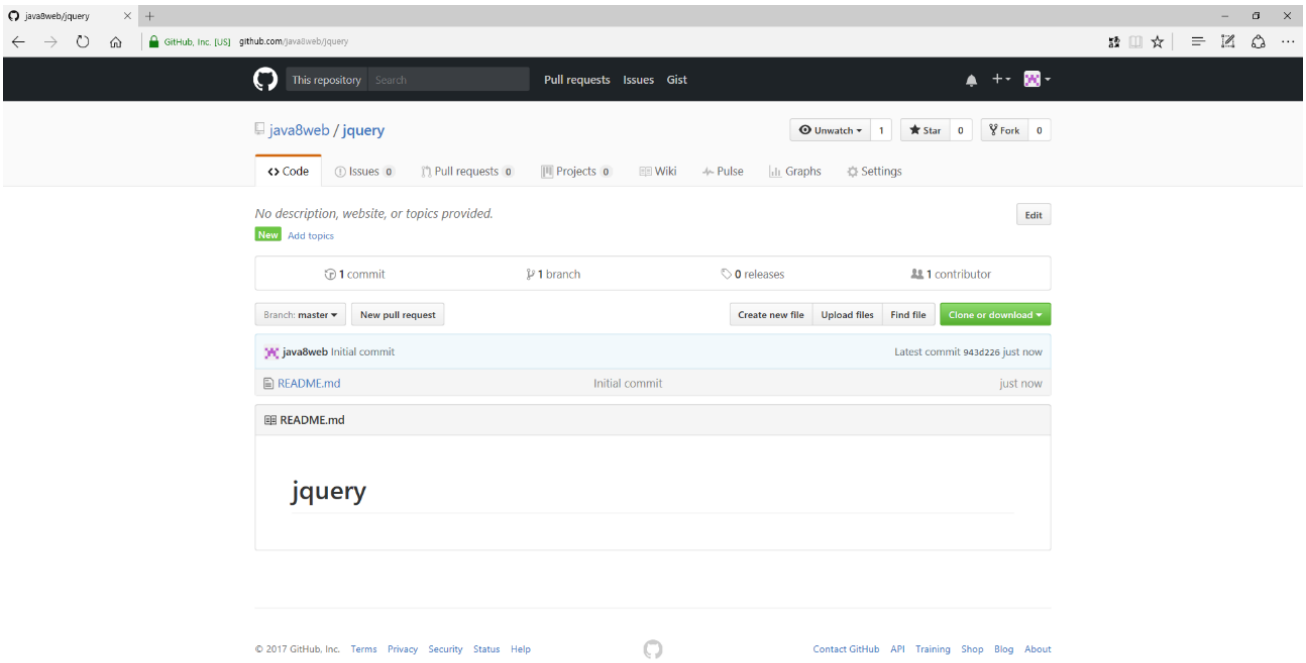
- Description : 생성한 원격 저장소의 역할에 대한 설명을 적는다. 생략 가능

- Public/Private : 원격 저장소의 공개 여부를 선택한다. 무료 사용자는 Public만 선택할 수 있다.

- Initiallze this repository with a README : 기본적으로 체크 표시를 해준다. GitHub에서 생성한 원격 저장소를 바로 로컬 저장소에 복사해서 가져올 수 있다. 또한, 저장소 이름과 Description 항목의 내용을 담은 README.md 파일을 생성한다.

- Add .gitgnore : 원격 저장소에 포함되지 않을 파일들의 목록을 만들 때 사용한다. 지금은 'none'으로 설정한다.

- Add a license : 원격 저장소에 저장할 프로젝트가 어떤 라이선스에 속할지를 선택한다. 지금은 'none'으로 설정한다.



<GitHub의 기능>

- 포크(Fork) : 다른 사람의 저장소를 복사하는 기능.

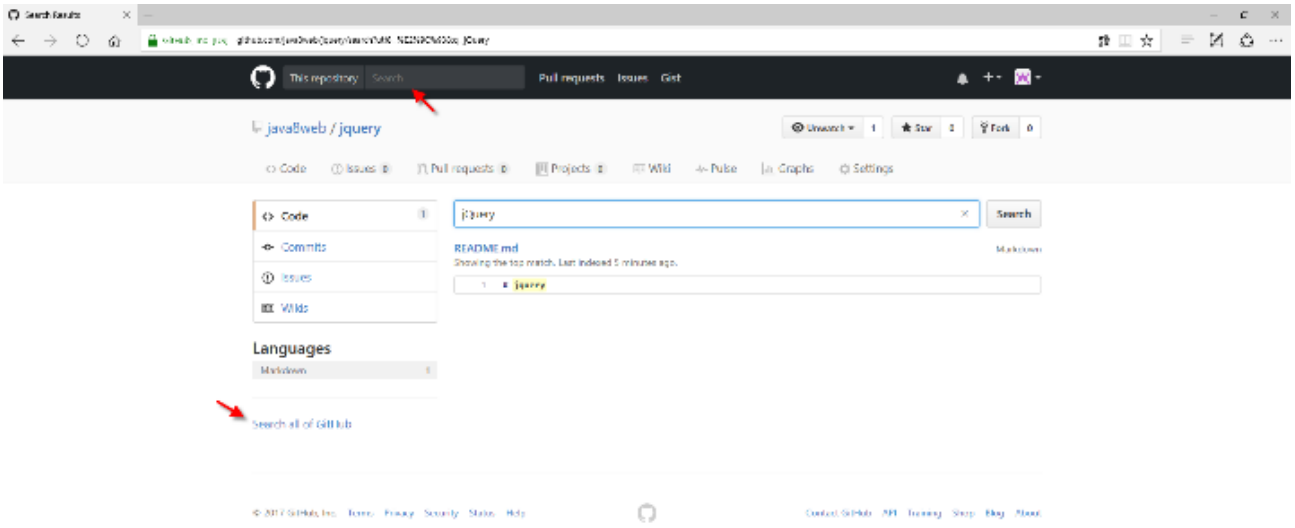
- 풀 리퀘스트(Pull Reguest) : 포크한 저장소를 수정해 다시 원본 저장소에 병합하라는 요청을 보내 사용자 사이의 상호 작용을 하는 기능.

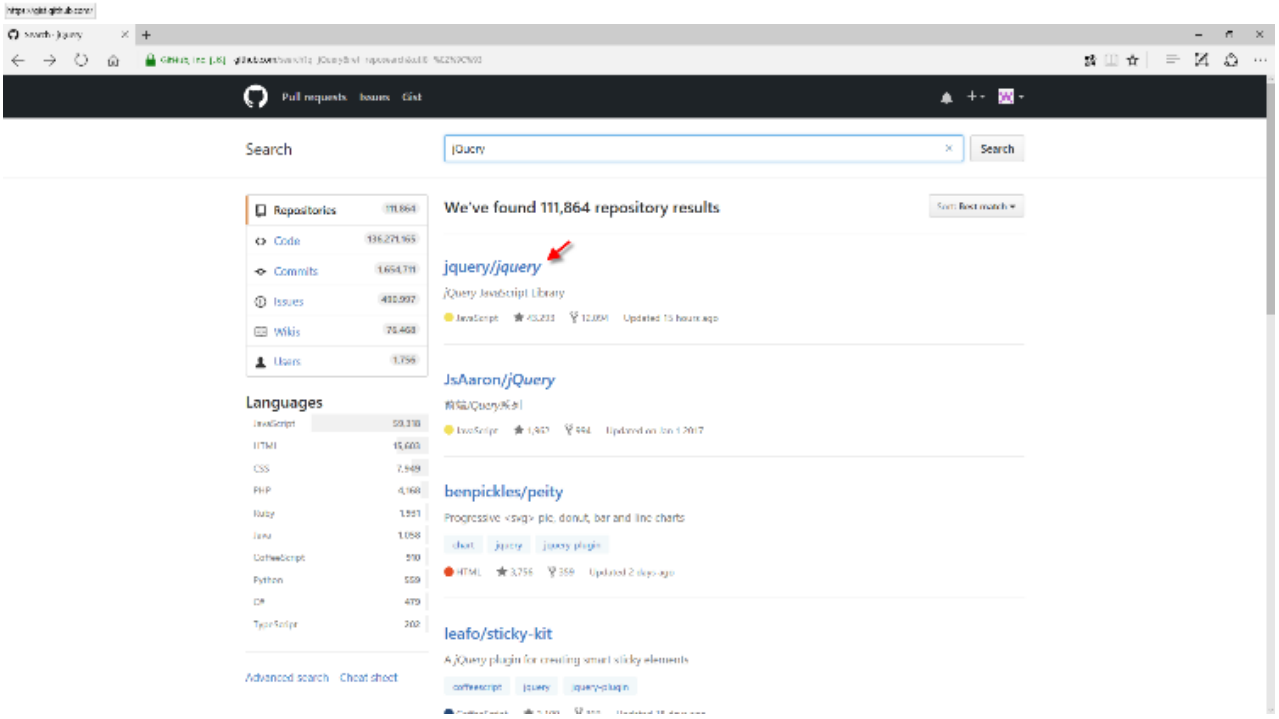
- 이슈(Issues) : 저장소 안에서 사용자들 사이의 문제를 논의하는 기능.

- 위키(Wiki) : 저장소와 관련된 체계적인 기록을 남기는 기능.

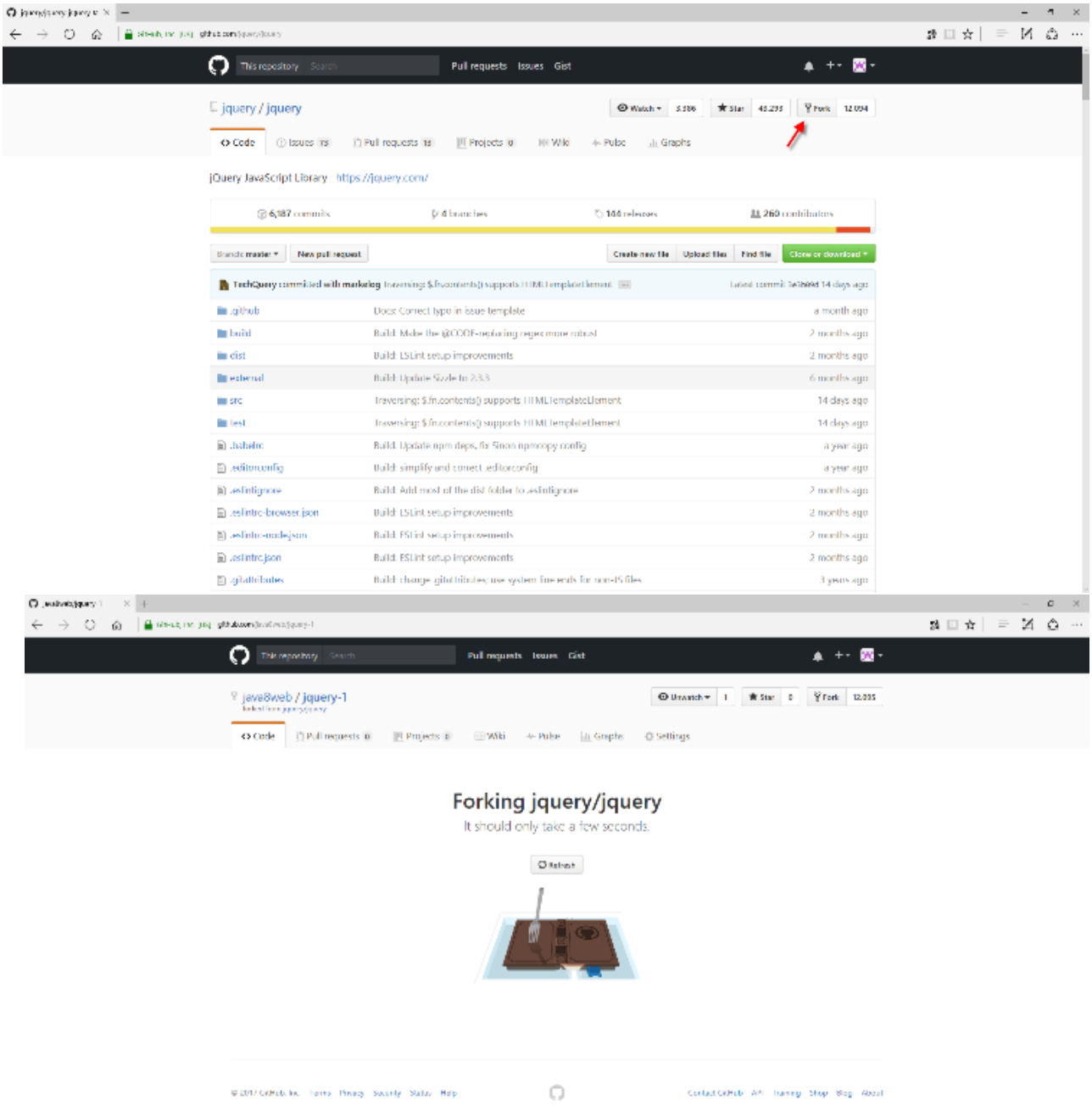
<Fork>

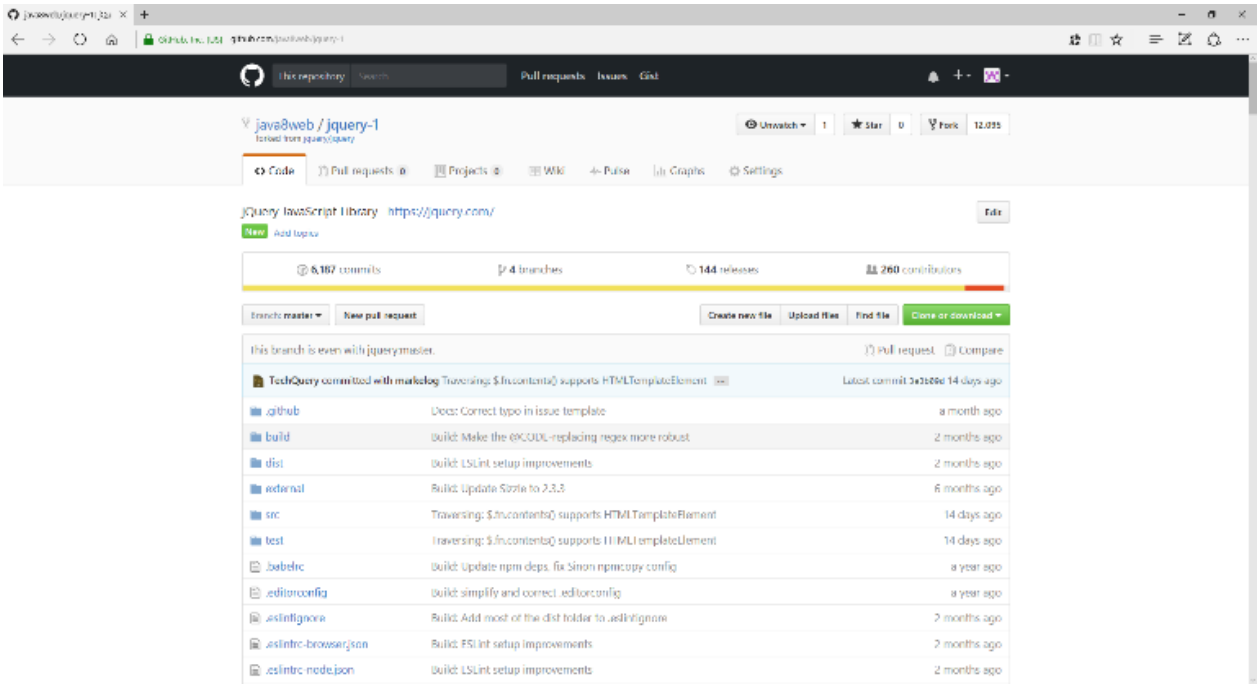
검색창에 jQuery로 검색한 후 왼쪽 하단의 'Search all of GitHub'를 클릭한다.

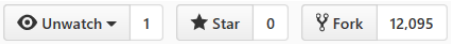




jquery/jquery를 클릭한 후 오른쪽 상단의 Fork를 클릭한다.


This repository
Pull requests
Issues
Gist
jquery / jquery
Code
jQuery JavaScript Library https://jquery.com/
Forking jquery/jquery
It should only take a few seconds.

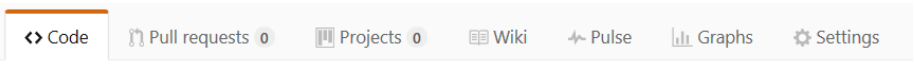




- Watch : 원격 저장소의 활동 내역을 사용자에게 열려준다. 댓글이나 이슈 등에서 언급될 때만 알려주는 No Watching, 모든 활동 내역을 알려주는 Watching, 모든 알림을 무시하는 Ignoring을 선택할 수 있다. 숫자는 현재 활동 내역을 보고 있는 방문자의 수이다.

- Star : 해당 원격 저장소에 관심이 있을 때 클릭한다. 숫자는 관심이 있는 사람의 수이다.

- Fork : 원격 저장소를 포크합니다. 숫자는 포크한 사람의 수이다.



- Code : 해당 원격 저장소의 루트 디렉토리이다.

- Pull Requests : 풀 리퀘스트 전체 목록을 보여준다. 목록마다 댓글 형태로 토론할 수 있다. 숫자는 현재 요청이 온 풀 리퀘스트를 받아들일 것인지에 대한 논의가 몇 개인지 알려준다.

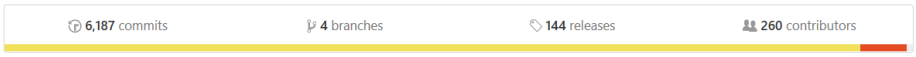
- Projects : 해당 원격 저장소에 있는 프로젝트 수이다.

- Wiki : 공유할 정보나 개발 문서, 참고 자료 등을 작성하기 위한 기능이다.

- Pulse : 해당 원격 저장소의 최근 변경 내역을 확인할 수 있다. 최대 한 달까지의 변경 내역을 확인할 수 있으며, 풀 리퀘스트 몇 개중에 몇 개가 받아들여졌나, 이슈는 몇 개고 몇 개가 해결되었나, 해당 풀 리퀘스트와 이슈에 관련된 활동을 볼 수 있다.

- Graphs : 공헌자의 공헌 내역, 커밋 수 등 해당 저장소의 활동 내역을 그래프로 보여준다.

- Settings : 해당 원격 저장소의 각종 설정을 변경할 수 있다.



- commits : 원격 저장소의 총 커밋 수.

- branches : 원격 저장소의 브랜치 수.

- releases : 원격 저장소의 태그(tag) 수를 나타낸다. 주로 특정 버전에 표시를 주고 싶을 때 사용한다. 이 표식을 통해서 특정 버전을 다운로드 할 수 있다.

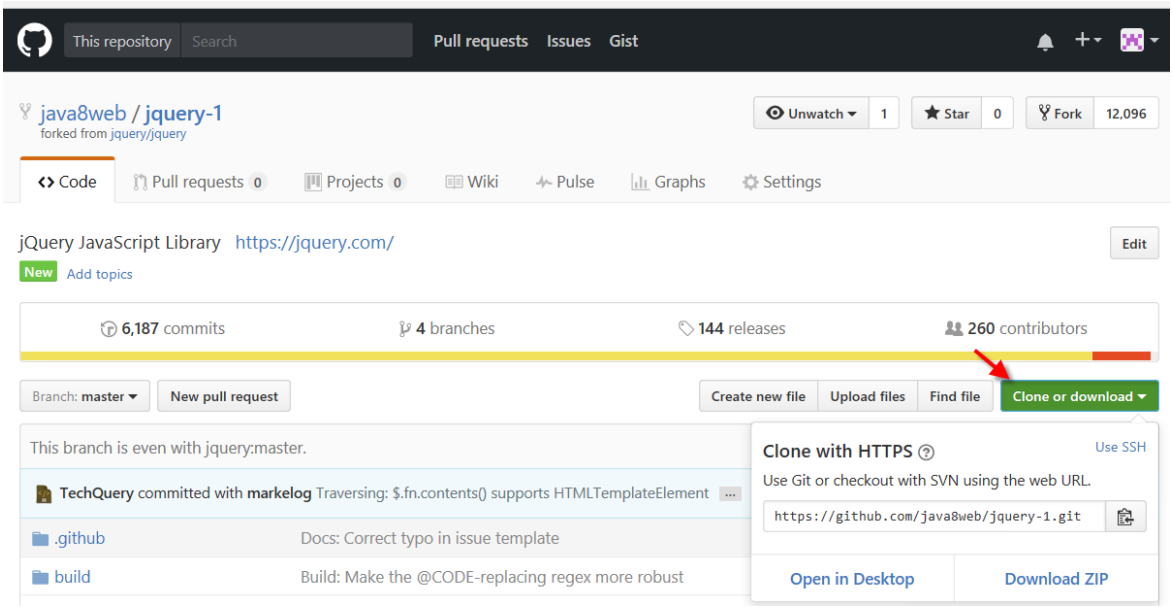
- contributors : 원격 저장소의 커밋 또는 풀 리퀘스트가 받아들여진 사용자의 수이다. 이 저장소가 오픈 소스라면 이 오픈 소스에 공헌한 사람의 수라 생각해도 된다.

소유권을 이전하게 되면 소유권을 이전한 원래 원격 저장소 관리자는 공헌자(Contribution)가 된다.

원격 저장소	특징
공개 원격 저장소	저장소 관리자, 협업자(Collaborators) 이외에는 쓰기 권한이 없다. GitHub 사용자라면 누구나 읽기 권한과 포크 권한이 있다. GitHub 사용자 누구에게든 소유권을 이전할 수 있다.
비공개 원격 저장소	관리자가 지정한 협업자만 접근해서 다룰 수 있다. 지정한 협업자에게만 포그 기능이 열려 있다. 유료 사용자에게만 소유권을 이전할 수 있다.

<git clone> : 원격 저장소의 내용을 로컬 저장소로 가져오기

GitHub의 원격 저장소와 내 컴퓨터를 연결해 데이터를 복사하는 작업을 클론(Clone)이라 한다.



<git remote> : 로컬 자장소와 원격 저장소 연결하기

로컬 저장소를 빈 원격 저장소와 연결한다.

<git push> : 로컬 작업 내역을 원격 저장소에 올리기

원격 저장소에 자신이 작업한 결과물을 업로드한다. 백지 상태인 원격 저장소에 로컬 저장소에서 작업한 것을 푸시해야 하므로 README.md 파일을 생성해서는 안 된다.

<git fetch 와 git pull> : 원격 저장소와 로컬 저장소의 간격.

로컬 저장소에서 작업하는 도중에 다른 협업자가 원격 저장소를 먼저 변경 할 수 있는데 이런 경우 push를 허용하지 않는다. 로컬 저장소의 커밋들을 원격 저장소와 맞추어야 하는데 이럴때 하는 것이 fetch이다. fetch는 원격 저장소의 커밋들을 로컬 저장소로 가져온다.

pull은 원격 저장소의 정보를 가져오면 자동으로 로컬 브랜치에 병합까지 수행한다.