Desktop Webapp 개발 시작하기

김희준
Microsoft VC++ MVP

Microsoft

서문 – 개인적 경험 2015

- Desktop app 개발
- 조건
  - Cross-platform, Mobile
  - 빠른 결과물, 예쁜 UI
  - 3rd party 라이브러리가 많고, 사용 용이할 것
- 결과
  - CEF 기반으로 개발 – Adobe Bracket-Shell 코드를 많이 참고
  - 많은 삽질(어?! 안되는 거였어?)
  - 느낀 점은 발표를 정리할 때

질문

- Desktop Application 개발 임무가 주어졌다면?
- 고려해야 할 것들…
  - Platform (OS)
    - Windows, OS X, Linux?
  - 사용할 수 있는 언어나 기술
    - .NET Framework 4, or later?
    - 언어 – C#, VB, WPF?

Cross-platform development

- Cross Platform lib based
  - QT
  - Xamarin
- Script language
  - Python - PyQT
- Desktop Web app?

Chromium based Web Apps

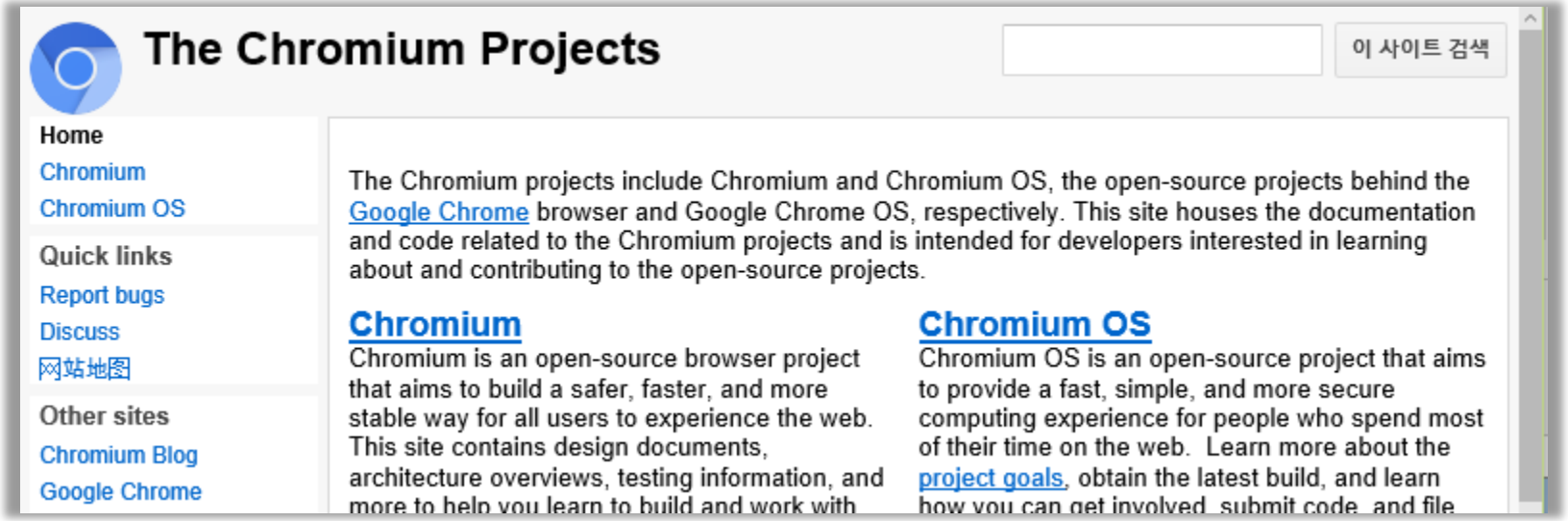
- Examples
  - Visual Studio Code
  - Adobe Bracket…
  - Slack, Wunderlist…
  - 밴드, 비트윈
- Technologies behind the web apps
  - Node webkit (NW.js)
  - CEF(Chromium Embedded Framework)
  - Electron

장점과 단점

- 장점
  - Cross platform ready
  - UI 개발 및 표현 편리함
- 단점
  - 새로운 기술에 대한 learning curve
  - 소스 코드에 대한 보안
  - 배포 사이즈
  - …

Chromium, chrome

- Chromium is the open-source project behind Google Chrome.
- Chromium is similar to Chrome, but lacks built-in automatic updates an d built-in Flash player, as well as Google branding and has a blue-colore d logo instead of the multicolored Google logo.



Chromium Content Module

- https://www.chromium.org/developers/content-module
- … Core code needed to render a page using a multi-process sandboxed browser.
- It includes all the web platform features (i.e. HTML5) and GPU acceleration.
- It does not include Chrome features, i.e. extensions/autofill/spelling etc.
- http://www.chromium.org/blink

NW.js

- https://github.com/nwjs/nw.js/
- Roger Wang - https://twitter.com/wwr
- Sponsored by Intel
- Usages
  - https://github.com/nwjs/nw.js/wiki/List-of-apps-and-companies-using-nw.js

Electron

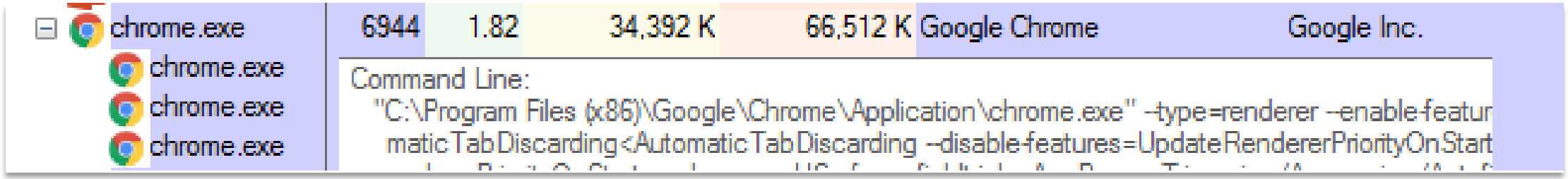
- Atom-Shell
- Visual Studio Code & Atom Editor
- Sponsored by GitHub
- Chromium + Node

CEF

- https://bitbucket.org/chromiumembedded/cef
- CEF1, CEF3
- Marshall Greenblatt
- Adobe – Bracket (Bracket-shell)
- No Node.js!
- Supported Languages: .NET, Python, etc…
- Dynamic linking (libcef.dll)

Browser vs Renderer

- Main Process(Browser window)
- Render Process(Blink rendering + JavaScript execution)
- IPC 통신 필요



Electron – detail & demo

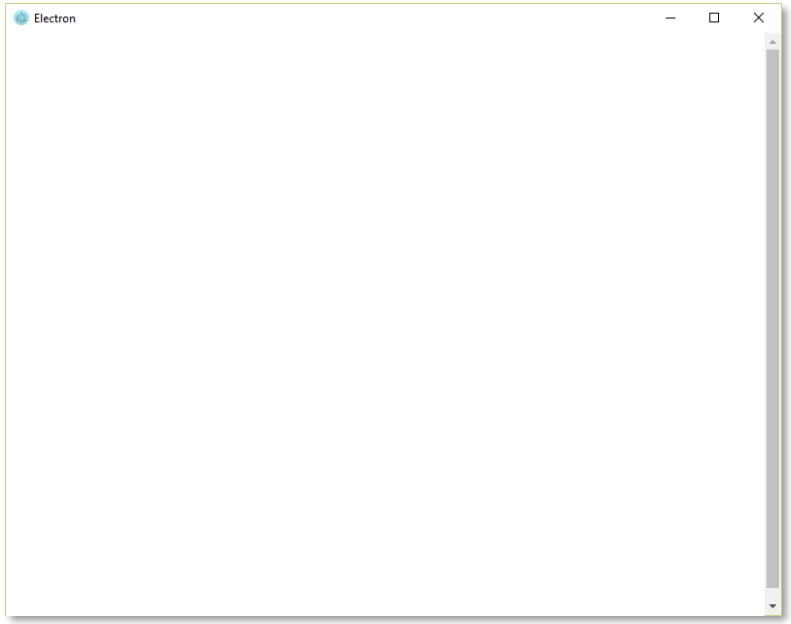
- 준비물
  - node, npm
- 진행 과정
  - electron-prebuilt 설치
  - package.json 작성
    - Html, js, css…
- ~~패키징과 배포~~
  - ~~아이콘 변경~~
  - ~~asar, electron-packager~~

```
{
  "name"    : "your-app",
  "version" : "0.1.0",
  "main"    : "main.js"
}
```

Demo - skeleton

- 최소한의 코드
  - package.json
  - main.js (브라우저)
  - Index.html (렌더러)

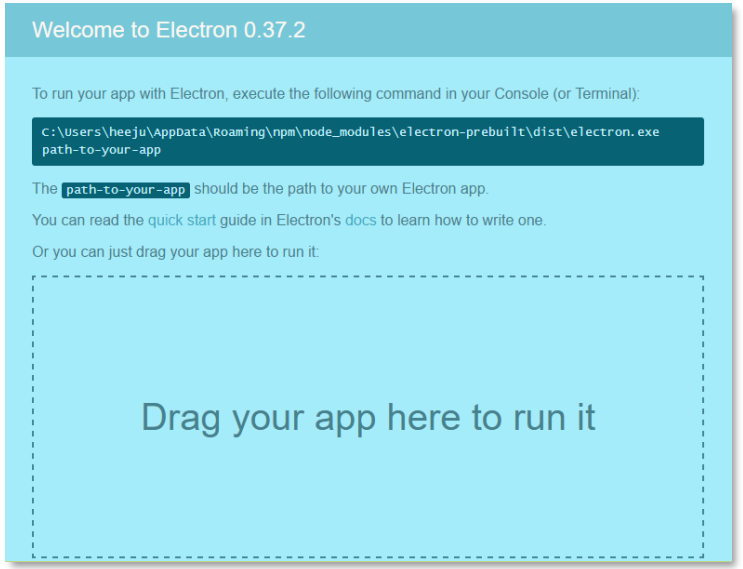
Browser – Renderer (skeleton demo)



Browser process(main.js)

ipc

ipc & remote

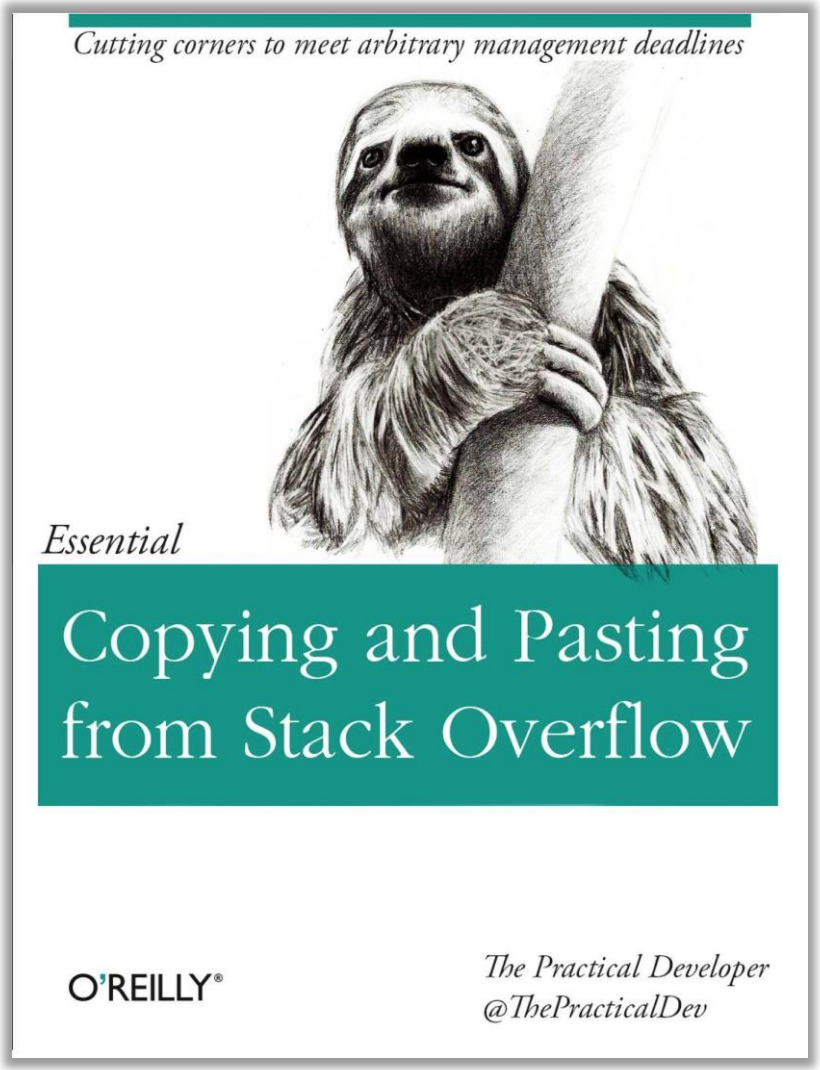


Renderer process (index.html)

ipc, remote: node module

어떤 기능들이 필요할까요?

- app basic features
  - 기본 UI 다루기
    - 버튼, 편집창, 멀티 윈도우...
    - Menu, Tray
  - Local file access
  - RESTful 서비스 사용
- Demo
  - Comm
  - Menutray
  - fileupload



https://twitter.com/ThePracticalDev/status/705825638851149824

Electron IPC

	브라우저	렌더러

Browser -> Renderer	win.webContents.send(str, arg);
	
	// ipc 핸들러에서..
	ipcMain.on(str, function(event, srg) {
	event.sender.send('listDirCallback', files) })
		require('electron').ipcRenderer.on(str, function(event, arg){})

Renderer -> Browser	require('electron').ipcMain.on(str, function(event, arg){});	require('electron').ipcRenderer.send()

Electron (개인적인) 정리

- ‘node.js 데스크탑 애플리케이션 개발’로 요약
- 패키지와 배포 역시 npm module로 솔루션이 제공됨
  - 애플리케이션 아이콘(리소스)
- Browser – node code 디버깅은 의문
- 풀어야 할 숙제 ‘보안 '

CEF 시작하기

- https://bitbucket.org/chromiumembedded/cef
- 이미 빌드된 cef 바이너리 다운로드
  - https://cefbuilds.com/
  - 플랫폼에 맞게 다운로드
  - '빌드된' CEF dynamic modules + 기본 브라우저 app source
  - Source code는 cmake 지원 => Visual Studio 사용하여 개발&빌드
  - **cefclient** sample

CEF 브라우저/렌더러 구현 전략

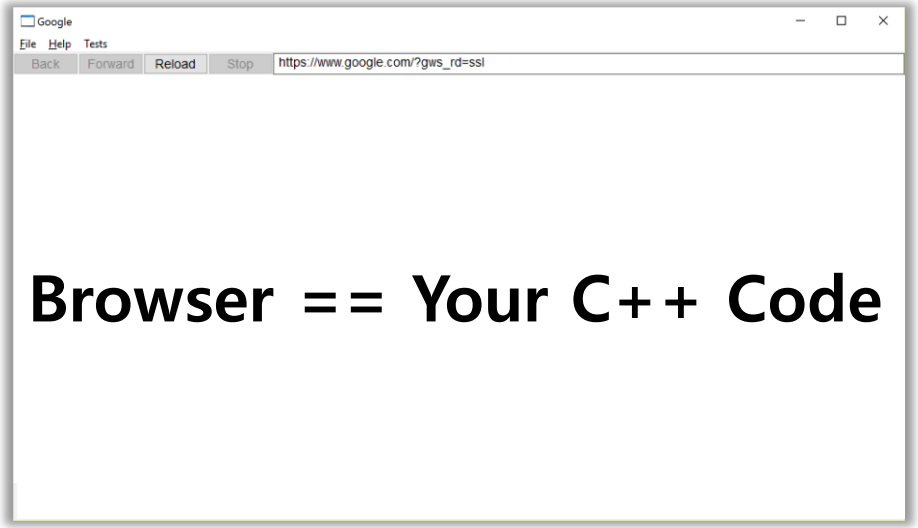
No node.js!

- app basic features
  - 기본 UI 다루기
    - 버튼, 편집창, 멀티 윈도우...
    - Menu, Tray
  - Local file access
  - RESTful 서비스 사용

- CEF 제공하는 API (Menu) 사용하여 브라우저에서 구현
- HTML, CSS, JS 사용하여 UI 구성 (렌더러)
- Tray는 Platform dependen하게 처리
  (OS API 호출, 브라우저)

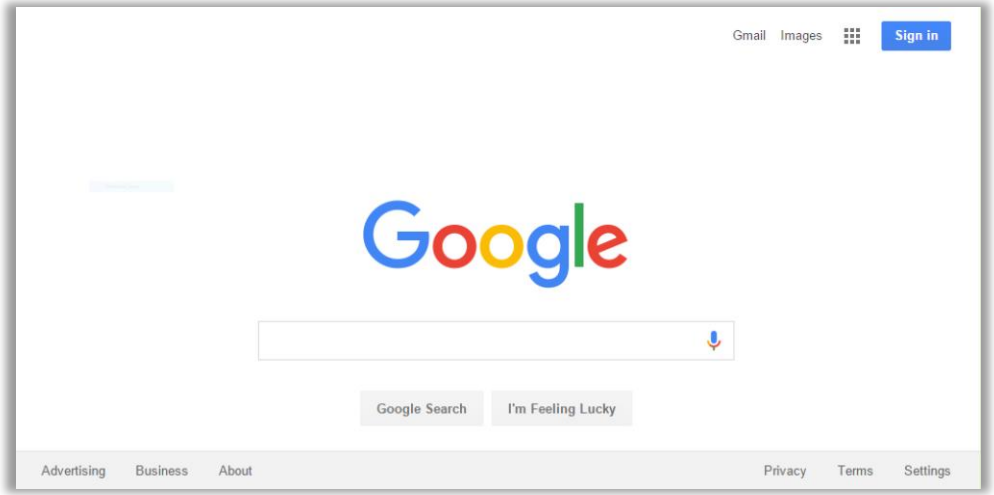
- 브라우저: CEF에서 API를 제공.
- 렌더러: XMLHttpRequest()

Browser – Renderer (CEF)



Browser == Your C++ Code

Message Router



Renderer == Your C++ Code & HTML/JS

- CEF Framework
- Platform dependent code

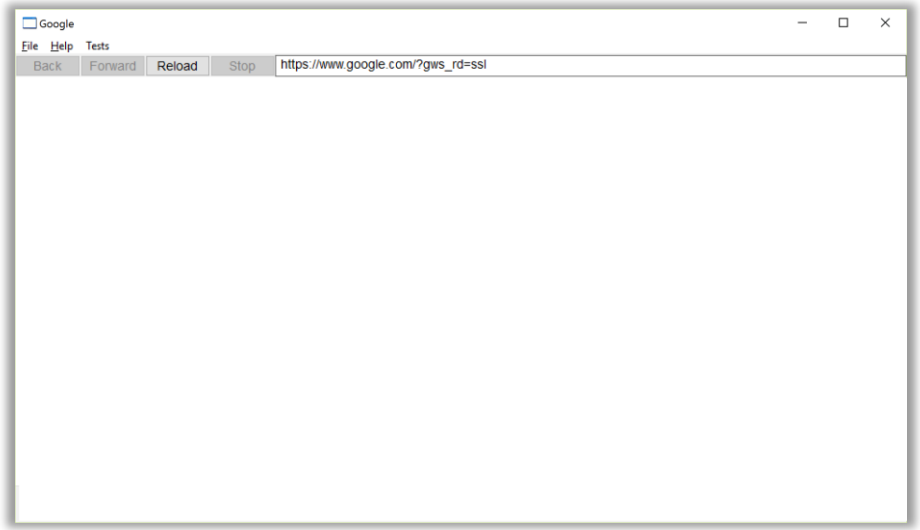
- Renderer ⇔ Browser 간의 통신은 C++ code에서 Message Router가 담당

CEF 세부 구현 방법 예

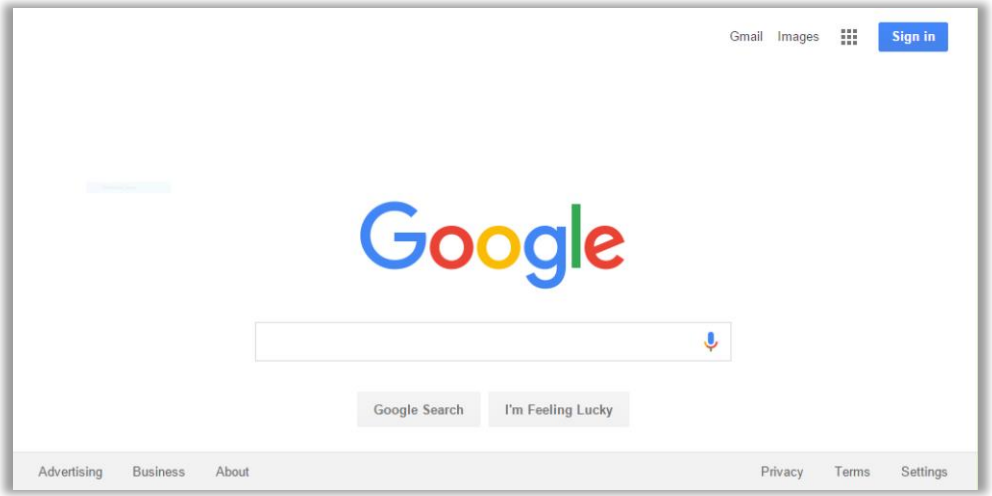
- [Browser] Context Menu - ClientHandler::OnBeforeContextMenu 구현
  - cefclient₩browser₩client_handler.cc(170):void ClientHandler::OnBeforeContextMenu
- [Browser] Tray Icon - Platform dependent 구현 필요
- Local file access
  - [Browser] 방법 1. 브라우저 생성 옵션
    - cefclient₩browser₩root_window_manager.cc(67): settings.file_access_from_file_urls = cef_state_t::STATE_ENABLED;
  - [Browser] 방법 2. 브라우저 요청을 CEF에서 반환
    - ClientHandler::GetResourceHandler에서 로컬 리소스를 읽어 반환 (브라우저)
    - cefclient₩browser₩test_runner.cc(687): resource_manager->AddProvider(CreateBinaryResourceProvider(test_origin),

CEF 세부 구현 방법 예 #2

- REST(GET/POST) 요청
  - 브라우저 프로세스에선,
    - 브라우저 -> CefURLRequest::Create() 메시지 전송
    - Cefclient₩browser₩urlrequest_test.cc(122): CefRefPtr<CefRequest> cef_request = CefRequest::Create();
  - 렌더러(HTML/JS)에선,
    - XMLHttpRequest() 호출 가능, 그러나 **파일 엑세스가 문제**
    - 해결책)
    - 1) [렌더러] XMLHttpRequest()로 브라우저에 파일 로드 요청(responseType blob)
    - 2) [브라우저] cef에서 (1) 요청을 파일 바이너리를 읽어 응답
    - 3) [렌더러] XMLHttpRequest 요청 결과를 new FormData().append(..., blob, ...)



1. XMLHttpRequest
responseType = “blob”



2. ifstream open & read
"application/octet-stream” 타입으로 응답

3. XMLHttpRequest(..., blob, ...)

IPC

	브라우저	렌더러
브라우저 -> 렌더러	₩cefclient₩browser₩test_runner.cc(706): frame->ExecuteJavaScript("alert('" + msg + "');", frame->GetURL(), 0);	n/a
	window.cefQuery의 등록된 callback 계속 호출	

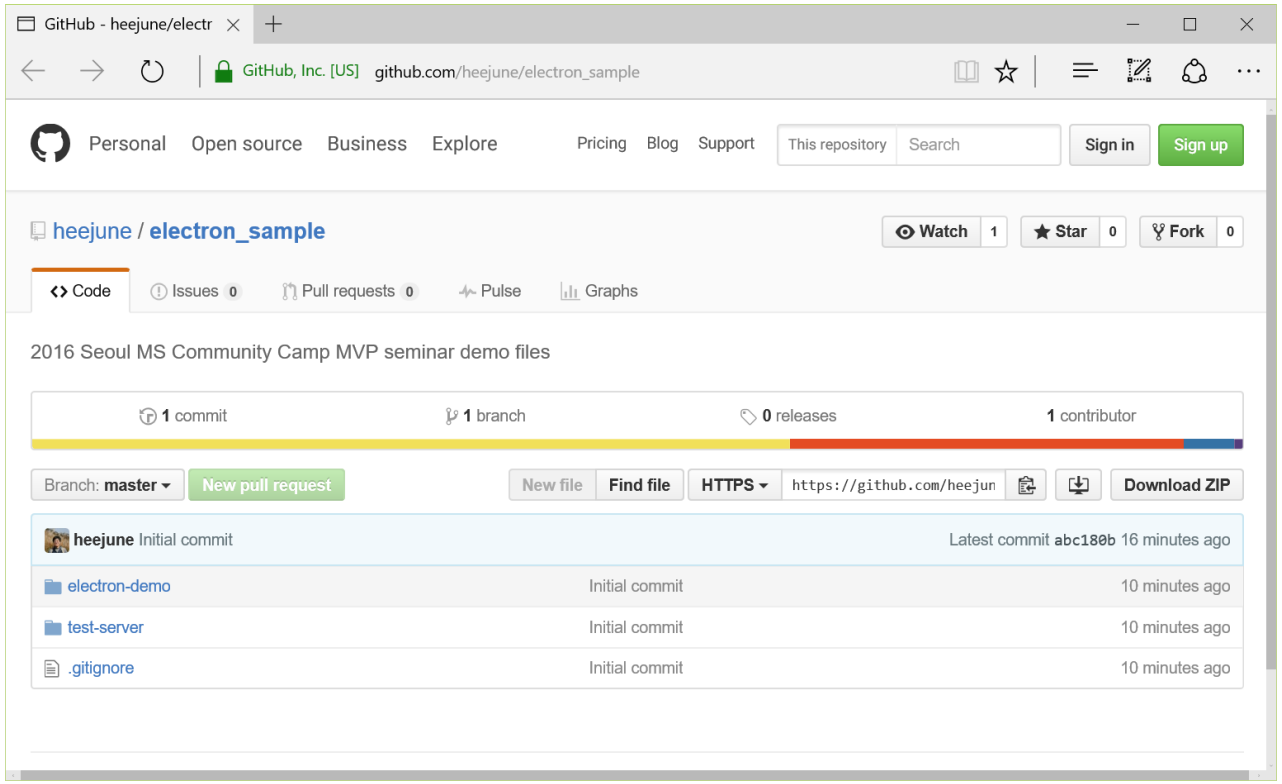
렌더러 -> 브라우저	class Handler : public CefMessageRouterBrowserSide::Handler {
	…
	virtual bool **OnQuery**(CefRefPtr<CefBrowser> browser, … CefRefPtr<Callback> **callback**) OVERRIDE {
	}
		window.cefQuery(request, persistent, onSuccess, onFailure)

느낀 점

- CEF
  - 많은 플랫폼 의존적인 코드 작성이 필요
  - Javascript 코드보다 더 많은 C++ 코드 작성이 필요
    - 장점이자 단점
  - IPC 구현에 약간의 수고가 더 필요함
- Electron
  - Browser code(node) 디버깅?!
  - node 사용은 큰 장점
    - 많은 노드 모듈을 활용한 빠른 개발

데모 & 샘플 파일

- https://github.com/heejune/electron_sample



Microsoft