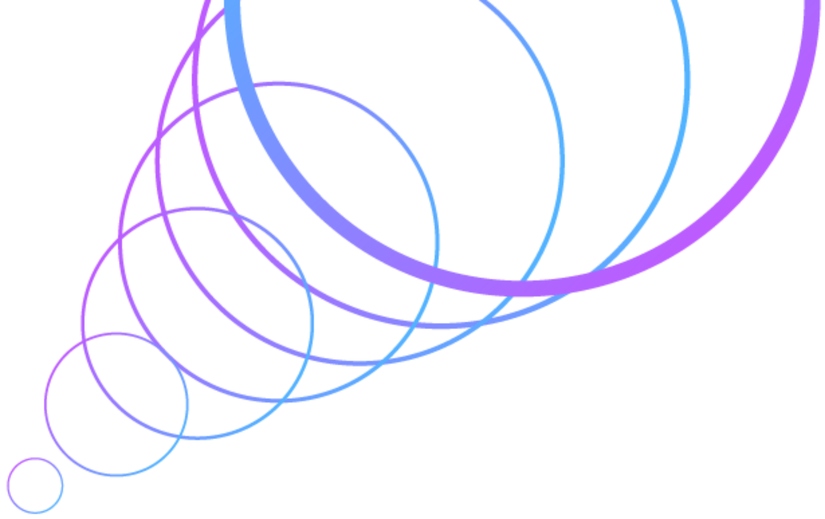
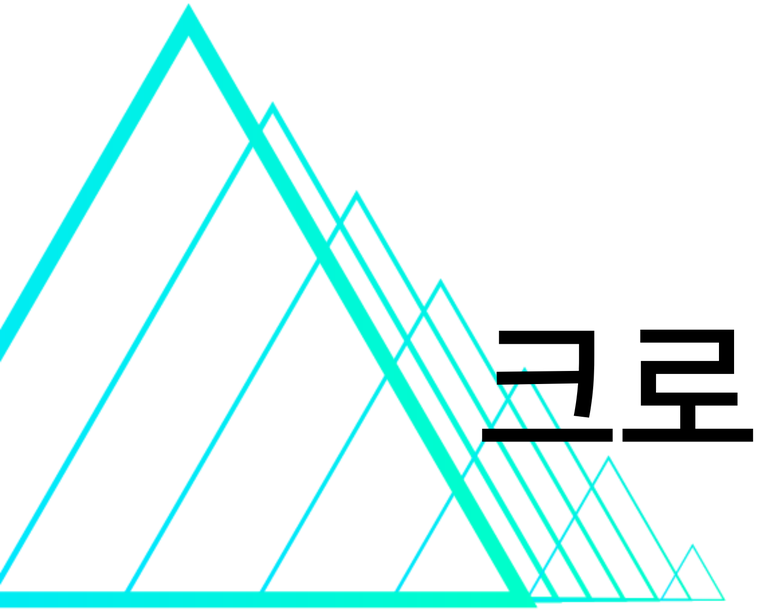




DEVIEW 2021

크로미움으로 크로스플랫폼 앱 개발하기

이원규 NAVER Whale

CONTENTS

1. 플랫폼 독립적인 C++ 앱 개발

1.1 C++ 표준의 발전

최근 C++의 발전으로 다양한 플랫폼에서 동일한 코드로 개발 가능

C++ 11

- thread, async, lambda, etc.

C++ 17

- filesystem, parallel algorithms, etc.

하지만 부족한 기능은 POSIX, Boost 라이브러리 등을 이용

크로미움이 대안이 될 수 있음!

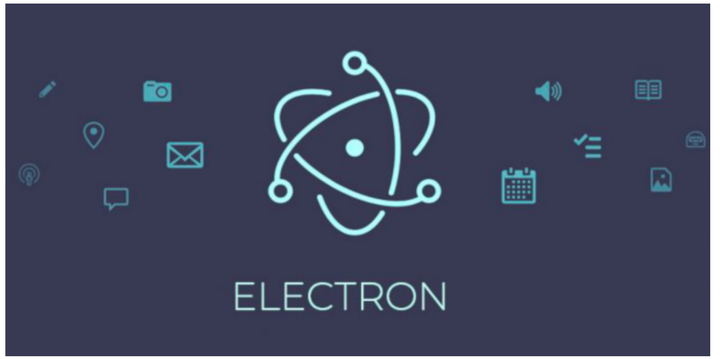
1.2 C++ 크로스플랫폼 UI 라이브러리

UI는 플랫폼에 독립적으로 만들기 힘듦

C++ 기반 UI 프레임워크들

- CEF, EFL, JUCE, Qt, etc.

Electron, CEF(Cromium Embeded Framework)는 크로미움 기반



크로미움이 대안이 될 수 있음!

https://en.wikipedia.org/wiki/List_of_platform-independent_GUI_libraries

1.3 크로미움 기반 브라우저

DEVIEW 2021

다양한 플랫폼 지원

- Desktop: Windows, Linux, MacOS
- Mobile: Android, iOS
- ChromiumOS, WhaleOS

크로스플랫폼 개발을 위한 코드 베이스

- base, ui, media, network, content, etc.

1.4 크로미움 모듈 소개

DEVIEW 2021

Base

- strings
- files
- i18n
- task
- process/threading
- synchronization
- time
- etc.

1.4 크로미움 모듈 소개

UI

- color
- display
- surface
- views/widets
- webui
- window managing
- etc.

1.4 크로미움 모듈 소개

DEVIEW 2021

Network

- ftp
- http
- socket
- ssl
- cert
- etc.

1.4 크로미움 모듈 소개

DEVIEW 2021

Media

- audio
- video
- capture
- cast
- ffmpeg
- filters
- midi
- muxers
- webrtc
- etc.

1.4 크로미움 모듈 소개

Content

- browser, renderer, etc.

그 외

- device, IPC, v8, pdf, skia, sql, etc.

2.개발환경 세팅

2.1 크로미움 버전 선택

DEVIEW 2021

OmahaProxy

- OS: linux, mac, win, cros, android, webview, etc.
- Channel: dev / beta /stable

os	channel	current_version	previous_version	current_reldate	previous_reldate	branch_base_commit	branch_base_position	branch_commit	true_branch	v8_version	changelog
linux	dev	96.0.4664.9	96.0.4662.6	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
linux	beta	95.0.4638.54	95.0.4638.49	10/18/21	10/13/21	159257cab5585bc8421abf34…	920003	91ad1c4f1b187a13b6189ea8…	4638	9.5.172.21	[cr]
linux	stable	94.0.4606.81	94.0.4606.71	10/07/21	09/30/21	35b0d5a9dc8362adfd44e261…	911515	471d3c96c601926db975ee03…	4606	9.4.146.21	[cr]
mac	beta	95.0.4638.54	95.0.4638.49	10/18/21	10/13/21	159257cab5585bc8421abf34…	920003	91ad1c4f1b187a13b6189ea8…	4638	9.5.172.21	[cr]
mac	dev	96.0.4664.9	96.0.4662.6	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
mac	stable	94.0.4606.81	94.0.4606.71	10/07/21	09/30/21	35b0d5a9dc8362adfd44e261…	911515	471d3c96c601926db975ee03…	4606	9.4.146.21	[cr]
win	stable	94.0.4606.81	94.0.4606.71	10/07/21	09/30/21	35b0d5a9dc8362adfd44e261…	911515	471d3c96c601926db975ee03…	4606	9.4.146.21	[cr]
win	canary_asan	97.0.4673.1	97.0.4673.0	10/18/21	10/18/21	e1d11e634f6d6e8618e5e0d6…	932383	5dd1aebadc7f5874425193e5…	4673	9.7.28	[cr]
win	canary	97.0.4673.3	97.0.4673.2	10/18/21	10/18/21	e1d11e634f6d6e8618e5e0d6…	932383	d97788ae817b4fe92d81ca49…	4673	9.7.28	[cr]
win	dev	96.0.4664.9	96.0.4662.6	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
win	beta	95.0.4638.54	95.0.4638.49	10/18/21	10/13/21	159257cab5585bc8421abf34…	920003	91ad1c4f1b187a13b6189ea8…	4638	9.5.172.21	[cr]
win64	canary	97.0.4673.3	97.0.4673.2	10/18/21	10/18/21	e1d11e634f6d6e8618e5e0d6…	932383	d97788ae817b4fe92d81ca49…	4673	9.7.28	[cr]
win64	dev	96.0.4664.9	96.0.4662.6	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
win64	beta	95.0.4638.54	95.0.4638.49	10/18/21	10/13/21	159257cab5585bc8421abf34…	920003	91ad1c4f1b187a13b6189ea8…	4638	9.5.172.21	[cr]
win64	stable	94.0.4606.81	94.0.4606.71	10/07/21	09/30/21	35b0d5a9dc8362adfd44e261…	911515	471d3c96c601926db975ee03…	4606	9.4.146.21	[cr]
mac_arm64	canary	97.0.4673.3	97.0.4673.2	10/18/21	10/18/21	e1d11e634f6d6e8618e5e0d6…	932383	d97788ae817b4fe92d81ca49…	4673	9.7.28	[cr]
mac_arm64	dev	96.0.4664.9	96.0.4662.6	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
mac_arm64	beta	95.0.4638.54	95.0.4638.49	10/18/21	10/13/21	159257cab5585bc8421abf34…	920003	91ad1c4f1b187a13b6189ea8…	4638	9.5.172.21	[cr]
mac_arm64	stable	94.0.4606.81	94.0.4606.71	10/07/21	09/30/21	35b0d5a9dc8362adfd44e261…	911515	471d3c96c601926db975ee03…	4606	9.4.146.21	[cr]
cros	stable	93.0.4577.107	94.0.4606.97	10/19/21	10/14/21	761ddde228655e313424edec…	902210	161da0529bc555e41267a2f7…	4577	9.3.345.19	[cr]
android	canary	97.0.4673.2	97.0.4673.0	10/18/21	10/18/21	e1d11e634f6d6e8618e5e0d6…	932383	5dd1aebadc7f5874425193e5…	4673	9.7.28	[cr]
android	dev	96.0.4664.9	96.0.4663.2	10/14/21	10/08/21	24dc4ee75e01a29d390d43c9…	929512	c07acff9c15ced2d3b90f3fe…	4664	9.6.180.5	[cr]
android	beta	95.0.4638.50	95.0.4638.40	10/13/21	10/06/21	159257cab5585bc8421abf34…	920003	c9bd7ca99e29d6e194e0b020…	4638	9.5.172.21	[cr]
android	stable	94.0.4606.85	94.0.4606.80	10/12/21	10/07/21	e9f990ba820b124b2cdc3049…	1319	c6a46dc7a8496c0360b684d7…	4606_80	9.4.146.21	[cr]
webview	beta	95.0.4638.50	95.0.4638.40	10/13/21	10/06/21	159257cab5585bc8421abf34…	920003	c9bd7ca99e29d6e194e0b020…	4638	9.5.172.21	[cr]

	mac	stable	94.0.4606.81
	win	canary_asan	97.0.4673.1
	win	canary	97.0.4673.3
	win	dev	96.0.4664.9
	win	beta	95.0.4638.54

https://omahaproxy.appspot.com/

2.2 Get the Code

크로미움 가이드를 따라 코드 다운로드 & 빌드

- https://www.chromium.org/developers/how-tos/get-the-code

GN(Generates Ninja build files)

- Ninja로 응용프로그램 프로젝트를 구축 할 수 있도록
  Ninja 빌드 파일을 생성하는 메타 빌드 시스템

GN tutorial

- https://gn.googlesource.com/gn/+/HEAD/docs/quick_start.md

2.3 IDE - VS Code

https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/vscode.md

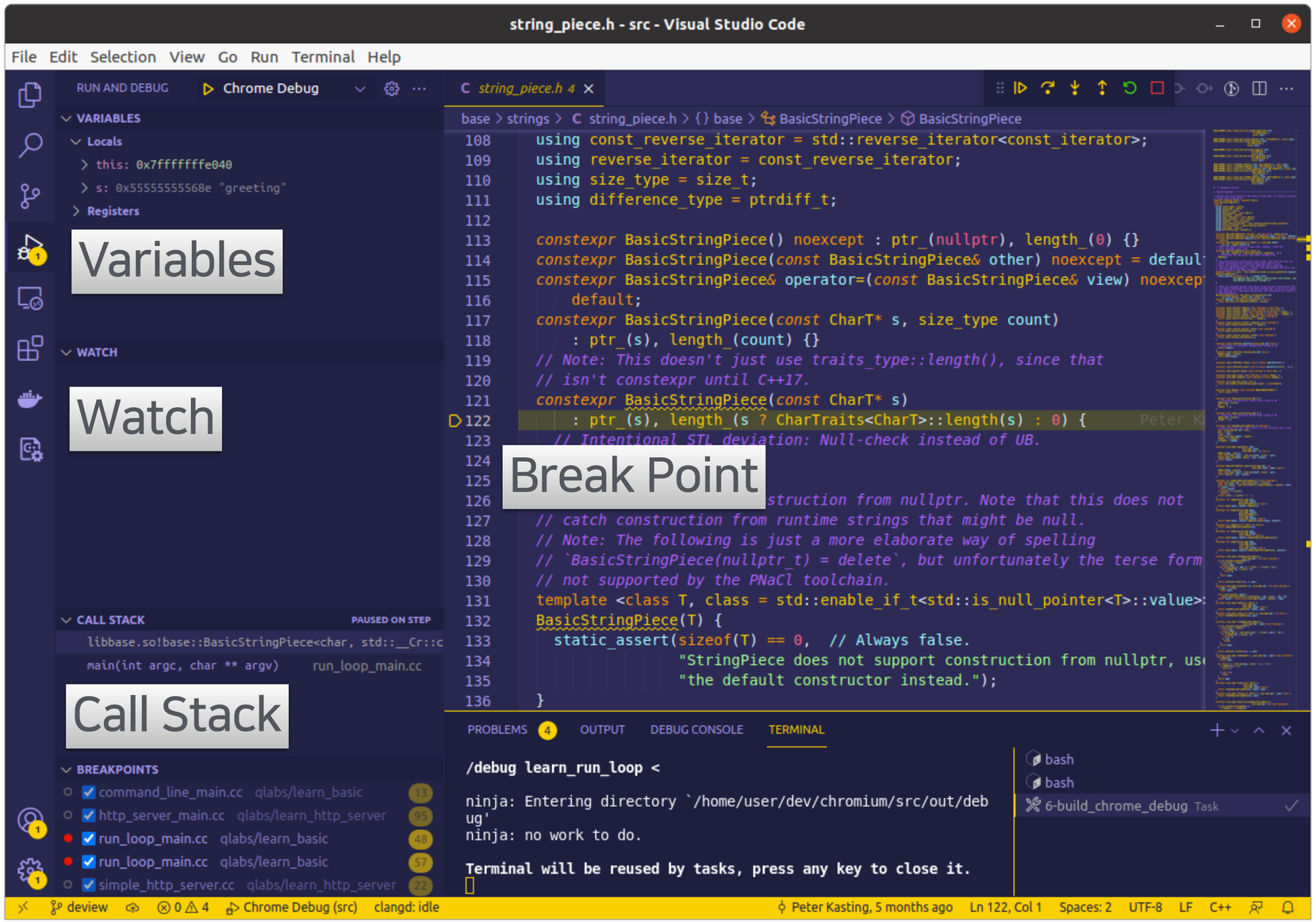
clangd를 사용해서 IDE에 기능 제공

- Auto complete, go to definition, find all references, etc.

GN 명령으로 ninja 파일 갱신하고,
generate_compdb.py로 컴파일 DB 생성

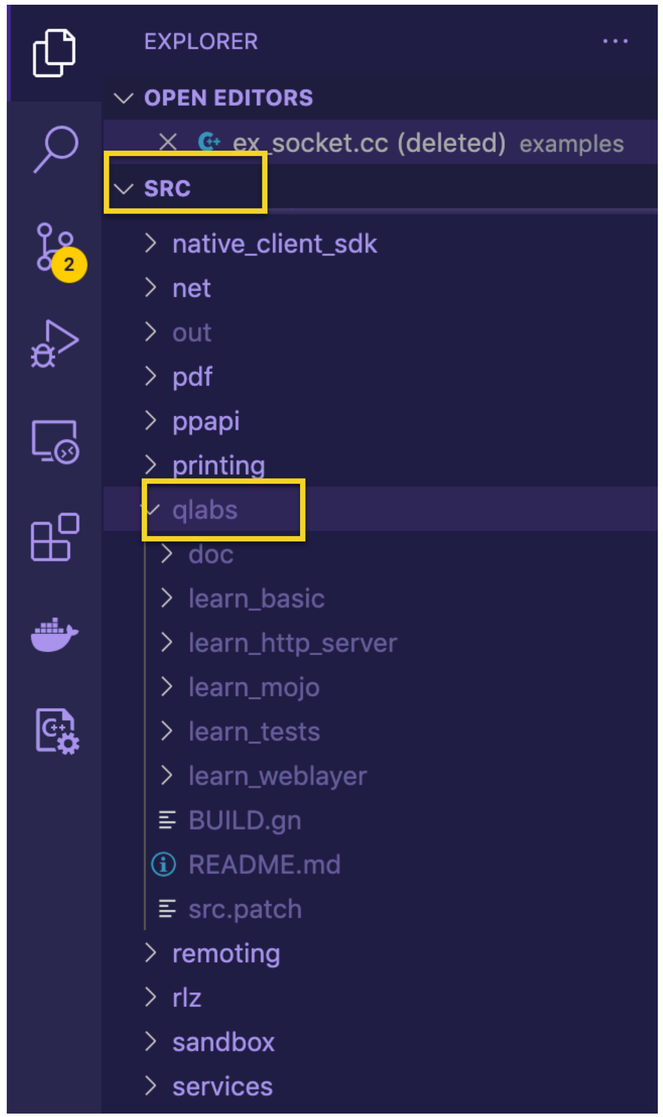
```
src$ gn gen out/debug

src$ tools/clang/scripts/generate_compdb.py \
-p out/debug \
> compile_commands.json
```

2.3 IDE - VS Code



3. 프로젝트 구성
저장소 관리
테스트

3.1 프로젝트 구성



크로미움을 설치하면 root인 src/가 생김

src/ 안에 프로젝트 폴더를 둠

- GN은 src/ 내 코드들을 대상으로 프로젝트 관리 / 빌드
- Ex. src/qlabs: qlabs 프로젝트

Demo GitHub Repo

- https://github.com/Wonkyu-Lee/qlabs

3.2 GN

DEVIEW 2021

Build Dependency Tree

```
qlabs_all
└── qlabs:qlabs
    ├── learn_basic
    ├── learn_http_server
    ├── learn_mojo
    └── learn_web_layer

qlabs_test_all
└── qlabs:tests
    └── learn_tests
```

src/BUILD.gn

```
...

group("qlabs_all") {
  deps = [ "//qlabs" ]
}

group("qlabs_test_all") {
  testonly = true
  deps = [ "//qlabs:tests" ]
}
```

src/qlabs/BUILD.gn

```
group("qlabs") {
  deps = [
    "learn_basic",
    "learn_http_server",
    "learn_mojo",
    "learn_weblayer",
  ]
}

group("tests") {
  testonly = true
  deps = [ "learn_tests" ]
}
```

3.2 GN

DEVIEW 2021

Build Dependency

```
qlabs_all
└── qlabs:qlabs
    ├── learn_basic
    ├── learn_http_server
    ├── learn_mojo
    └── learn_web_layer

qlabs_test_all
└── qlabs:tests
    └── learn_tests
```

src/BUILD.gn

```
...

group("qlabs_all") {
  deps = [ "//qlabs" ]
}

group("qlabs_test_all") {
  testonly = true
  deps = [ "//qlabs:tests" ]
}
```

src/qlabs/BUILD.gn

```
group("qlabs") {
  deps = [
    "learn_basic",
    "learn_http_server",
    "learn_mojo",
    "learn_weblayer",
  ]
}

group("tests") {
  testonly = true
  deps = [ "learn_tests" ]
}
```

3.2 GN

DEVIEW 2021

Build Dependency

```
qlabs_all
└── qlabs:qlabs
    ├── learn_basic
    ├── learn_http_server
    ├── learn_mojo
    └── learn_web_layer

qlabs_test_all
└── qlabs:tests
    └── learn_tests
```

Terminal

```
# Generate Ninja Files
src$ gn gen out/debug
```

3.2 GN

DEVIEW 2021

Build Dependency

```
qlabs_all
└── qlabs:qlabs
    ├── learn_basic
    ├── learn_http_server
    ├── learn_mojo
    └── learn_web_layer

qlabs_test_all
└── qlabs:tests
    └── learn_tests
```

Terminal

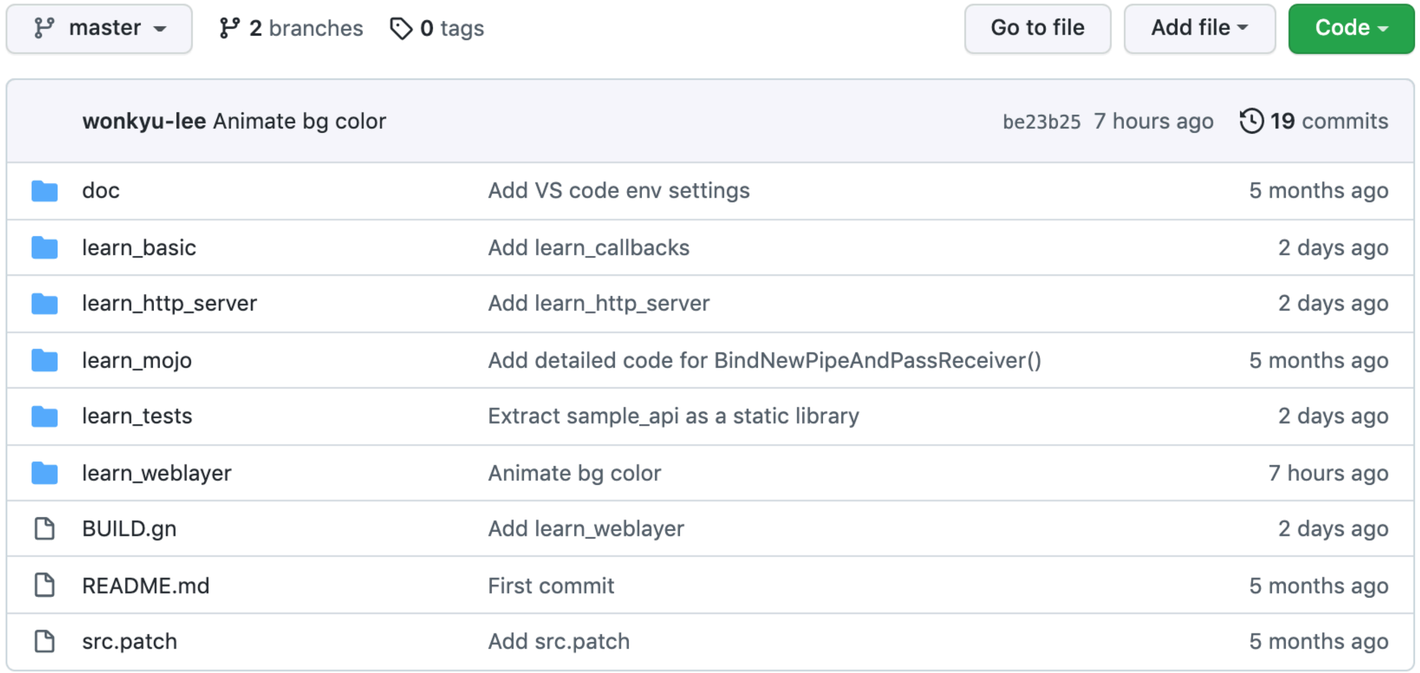
```
# Ninja Build
src$ autoninja -C out/debug qlabs_all
```

3.3 프로젝트 저장소 관리

DEVIEW 2021

크로미움 변경 없이
앱만 개발하고자 한다면,

- 크로미움 변경 사항은 patch로 관리
- 앱 디렉토리만 별도 저장소로 관리



[장점]
- 저장소 용량이 적게 듦
- Chromium.git과 독립적임

[단점]
- 크로미움 변경이 많을수록 이력 관리가 어려움
- 크로미움 버전 업데이트 시, 수작업으로 patch를 만들어야 함

3.3 프로젝트 저장소 관리

DEVIEW 2021

크로미움 전체를 저장소에 관리한다면,
- Fork Chromium.git

[장점]
- 크로미움과 앱의 변경 이력을 함께 관리

[단점]
- 저장공간이 많이 필요함
- 크로미움 업데이트 시, 변경 사항 충돌 관리 (=리베이스)

3.4 Hello, Chromium SDK!

DEVIEW 2021

GN: src/qlabs/learn_basic/BUILD.gn

```
executable("learn_hello") {
  sources = [ "hello_main.cc" ]
}
```

3.4 Hello, Chromium SDK!

DEVIEW 2021

GN: src/qlabs/learn_basic/BUILD.gn

```
executable("learn_hello") {
  sources = [ "hello_main.cc" ]
}
```

Code: src/qlabs/learn_basic/hello_main.cc

```
#include <iostream>

int main(int argc, const char* argv[]) {
  std::cout << "Hello, Chromium SDK!" << std::endl;
  return 0;
}
```

3.4 Hello, Chromium SDK!

GN: src/qlabs/learn_basic/BUILD.gn

```
executable("learn_hello") {
  sources = [ "hello_main.cc" ]
}
```

Code: src/qlabs/learn_basic/hello_main.cc

```
#include <iostream>

int main(int argc, const char* argv[]) {
  std::cout << "Hello, Chromium SDK!" << std::endl;
  return 0;
}
```

Terminal

```
src$ gn gen out/debug
Done. Made 17750 targets from 2920 files in 2788ms
```

3.4 Hello, Chromium SDK!

DEVIEW 2021

GN: src/qlabs/learn_basic/BUILD.gn

```
executable("learn_hello") {
  sources = [ "hello_main.cc" ]
}
```

Code: src/qlabs/learn_basic/hello_main.cc

```
#include <iostream>

int main(int argc, const char* argv[]) {
  std::cout << "Hello, Chromium SDK!" << std::endl;
  return 0;
}
```

Terminal

```
src$ gn gen out/debug
Done. Made 17750 targets from 2920 files in 2788ms

src$ autoninja -C out/debug learn_hello
ninja: Entering directory `out/debug'
[2/2] LINK ./learn_hello
```

3.4 Hello, Chromium SDK!

DEVIEW 2021

GN: src/qlabs/learn_basic/BUILD.gn

```
executable("learn_hello") {
  sources = [ "hello_main.cc" ]
}
```

Code: src/qlabs/learn_basic/hello_main.cc

```
#include <iostream>

int main(int argc, const char* argv[]) {
  std::cout << "Hello, Chromium SDK!" << std::endl;
  return 0;
}
```

Terminal

```
src$ gn gen out/debug
Done. Made 17750 targets from 2920 files in 2788ms

src$ autoninja -C out/debug learn_hello
ninja: Entering directory `out/debug'
[2/2] LINK ./learn_hello

src$ out/debug/learn_hello
Hello, Chromium SDK!
```

3.4 Unit Tests

DEVIEW 2021

Google Test
- 유닛테스트에 Google Test사용

예제
- ‘ex_api’ 라이브러리를 만들고,
- ‘ex_api_unittests’에서 라이브러리 테스트

Build Dependency

```
ex_api_unittests
└── ex_api
```

3.4 Unit Tests

DEVIEW 2021

Google Test
- 유닛테스트에 Google Test사용

예제
- ‘ex_api’ 라이브러리를 만들고,
- ‘ex_api_unittests’에서 라이브러리 테스트

Build Dependency

```
ex_api_unittests
└── ex_api
```

GN: learn_tests/BUILD.gn

```
import("//testing/test.gni")

static_library("ex_api") {
  sources = [
    "ex_api.cc",
    "ex_api.h",
  ]
}

test("ex_api_unittests") {
  testonly = true
  sources = [ "ex_api_unittests.cc" ]
  deps = [
    ":ex_api",
    "//base/test:run_all_unittests",
    "//testing/gtest",
  ]
}
```

3.4 Unit Tests

DEVIEW 2021

LIbrary: learn_tests/ex_api.cc

```
#include "qlabs/learn_tests/ex_api.h"

namespace ex_api {
bool CallApi() {
  return true;
}
}  // namespace ex_api
```

3.4 Unit Tests

DEVIEW 2021

LIbrary: learn_tests/ex_api.cc

```cpp
#include "qlabs/learn_tests/ex_api.h"

namespace ex_api {
bool CallApi() {
  return true;
}
}  // namespace ex_api
```

Test: learn_tests/ex_api_unittests.cc

```cpp
#include "qlabs/learn_tests/ex_api.h"
#include "testing/gtest/include/gtest/gtest.h"

namespace ex_api {
namespace {

TEST(ExApi, ApiTest) {
  EXPECT_TRUE(CallApi());
}

}  // namespace
}  // namespace ex_api
```

3.4 Unit Tests

DEVIEW 2021

LIbrary: learn_tests/ex_api.cc

```cpp
#include "qlabs/learn_tests/ex_api.h"

namespace ex_api {
bool CallApi() {
  return true;
}
}  // namespace ex_api
```

Test: learn_tests/ex_api_unittests.cc

```cpp
#include "qlabs/learn_tests/ex_api.h"
#include "testing/gtest/include/gtest/gtest.h"

namespace ex_api {
namespace {

TEST(ExApi, ApiTest) {
  EXPECT_TRUE(CallApi());
}

}  // namespace
}  // namespace ex_api
```

Terminal

```
src$ out/debug/ex_api_unittests
[1/1] ExApi.ApiTest (0 ms)
SUCCESS: all tests passed.
Tests took 0 seconds.
```

4. Base

4.1 Tasks and Callbacks

DEVIEW 2021

Callback
- OnceCallback: 한 번만 호출 가능
- RepeatCallback: 여러 번 호출 가능

Bind
- 함수나 멤버 메소드, 람다를
  인자들과 묶어서 Callback을 만듦

- BindOnce(method, params…):
  Returns OnceCallback

- BindRepeat(method, params…):
  Returns RepeatCallback

```
base::OnceCallback<int(int)> taskPlus10

= base::BindOnce(

  // 인자 두 개를 받는 람다
  [](int a, int b) {
    return a + b;
  },

  // 첫 번째 인자를 바인딩
  10
);


int result1 = std::move(taskPlus10).Run(3);


// Crash!!!
int result2 = std::move(taskPlus10).Run(3);
```

https://chromium.googlesource.com/chromium/src/+/HEAD/docs/callback.md

4.1 Tasks and Callbacks

DEVIEW 2021

Example

- 덧셈을 특정 스레드에서 수행한 뒤, 메인 스레드에서 결과를 받아서 출력

```
auto task_runner = base::ThreadPool::CreateTaskRunner(
  {base::TaskPriority::BEST_EFFORT});

auto task = base::BindOnce(
    [](int a, int b) {
      LOG(INFO) << "task : " << a << " + " << b;
      return a + b;
    },
    2, 3
);

auto reply = base::BindOnce(
  [](int result) {
    LOG(INFO) << "reply: " << result;
  }
);

task_runner->PostTaskAndReplyWithResult(
  FROM_HERE,
  std::move(task),
  std::move(reply)
);
```

TaskRunner

Posting된 task들을 실행하기 위해
Task Queue를 가짐

Terminal

```
[20611:INFO:callbacks_main.cc(24)] task : 2 + 3
[20609:INFO:callbacks_main.cc(30)] reply: 5
```

4.2 RunLoop

DEVIEW 2021

Example

- ‘—greeting’ 문자열을 ‘—repeat’ 카운트 만큼 1초 간격으로 출력

```
$ out/debug/learn_run_loop --greeting=hi --repeat=3
```

4.2 RunLoop

DEVIEW 2021

1. CommandLine 인자 읽기

```
int main(int argc, char** argv) {
  base::CommandLine::Init(argc, argv);

  const base::CommandLine& cmd_line = *base::CommandLine::ForCurrentProcess();
  std::string greeting = cmd_line.GetSwitchValueASCII("greeting");
  if (!greeting.empty()) {
    g_greeting = greeting;
  }

  int repeat_count = 5;
  auto repeat = cmd_line.GetSwitchValueASCII("repeat");
  if (!repeat.empty()) {
    base::StringToInt(repeat, &repeat_count);
  }
  . . .
```

4.2 RunLoop

DEVIEW 2021

2. Greeting() 함수

- 재귀적으로 Greeting task를 1초씩 delay를 주어 포스팅
- ‘count == 0’이 되면 Run Loop를 종료

```
std::string g_greeting = "hello";
base::RepeatingClosure g_quit_closure;

void Greeting(int count) {
  if (count == 0) {
    if (g_quit_closure) {
      g_quit_closure.Run();
      return;
    }
  }

  LOG(INFO) << g_greeting;

  base::ThreadTaskRunnerHandle::Get()->PostDelayedTask(
      FROM_HERE,
      base::BindOnce(&Greeting, count - 1),
      base::TimeDelta::FromSeconds(1));
}
```

4.2 RunLoop

DEVIEW 2021

3. RunLoop 실행

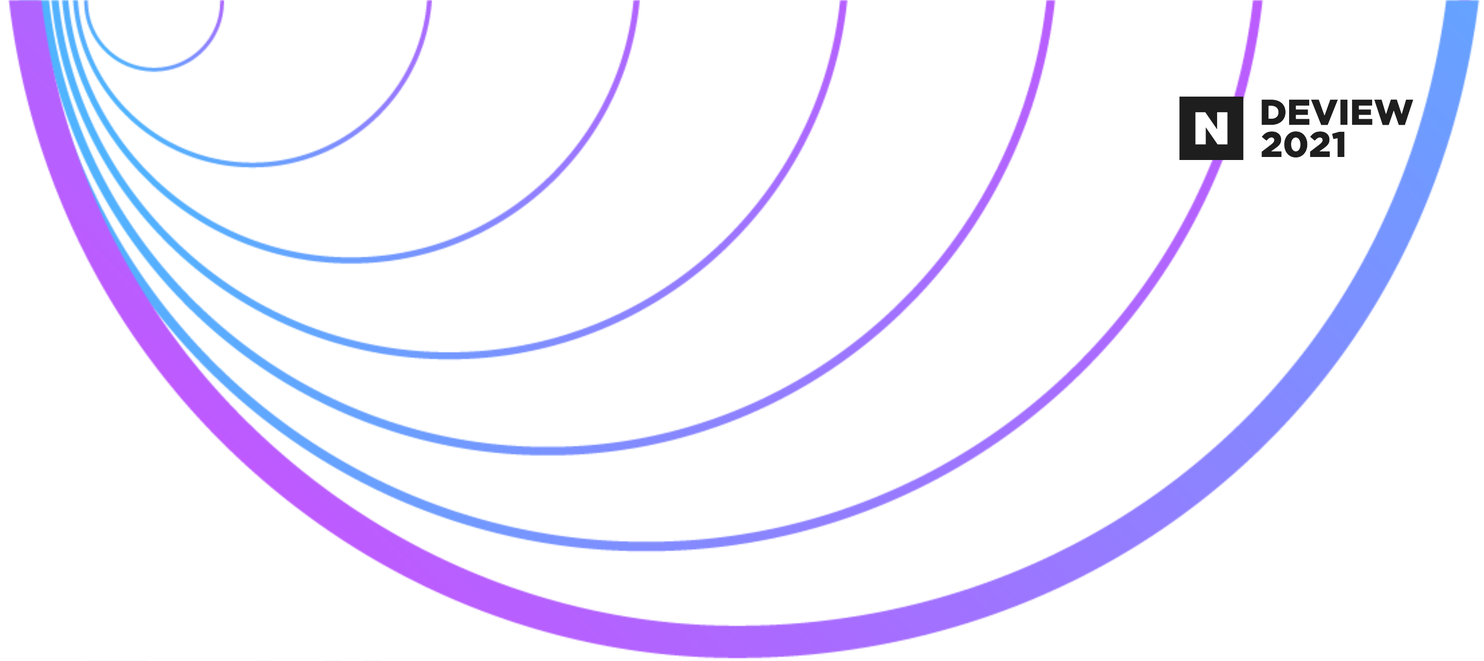
```
int main(int argc, char** argv) {
  . . .

  // UI thread task executor(event polling & running background tasks)
  base::SingleThreadTaskExecutor main_task_executor(base::MessagePumpType::UI);

  base::ThreadPoolInstance::CreateAndStartWithDefaultParams("Run Loop");

  base::RunLoop run_loop;
  g_quit_closure = run_loop.QuitClosure();

  // Post the first event to be polled out
  Greeting(repeat_count);

  run_loop.Run();

  return 0;
}
```

4.2 RunLoop

DEVIEW 2021

Terminal

```
$ out/debug/learn_run_loop --greeting=hi —repeat=3

[1003/124059.190601:INFO:run_loop_main.cc(33)] hi
[1003/124100.194874:INFO:run_loop_main.cc(33)] hi
[1003/124101.199800:INFO:run_loop_main.cc(33)] hi
```



5. UI

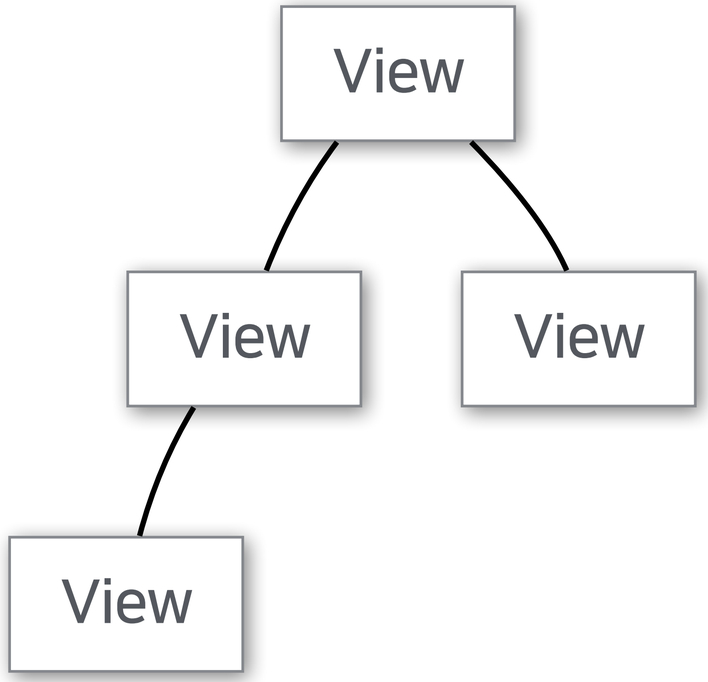
5.1 views::View

DEVIEW 2021

각 View는 다음 프로퍼티를 가짐
- 사각형 바운드
- Boarder
- Background

Layout manager로 자식 위치를 관리

View를 상속해서 새 UI 컴포넌트 만듦



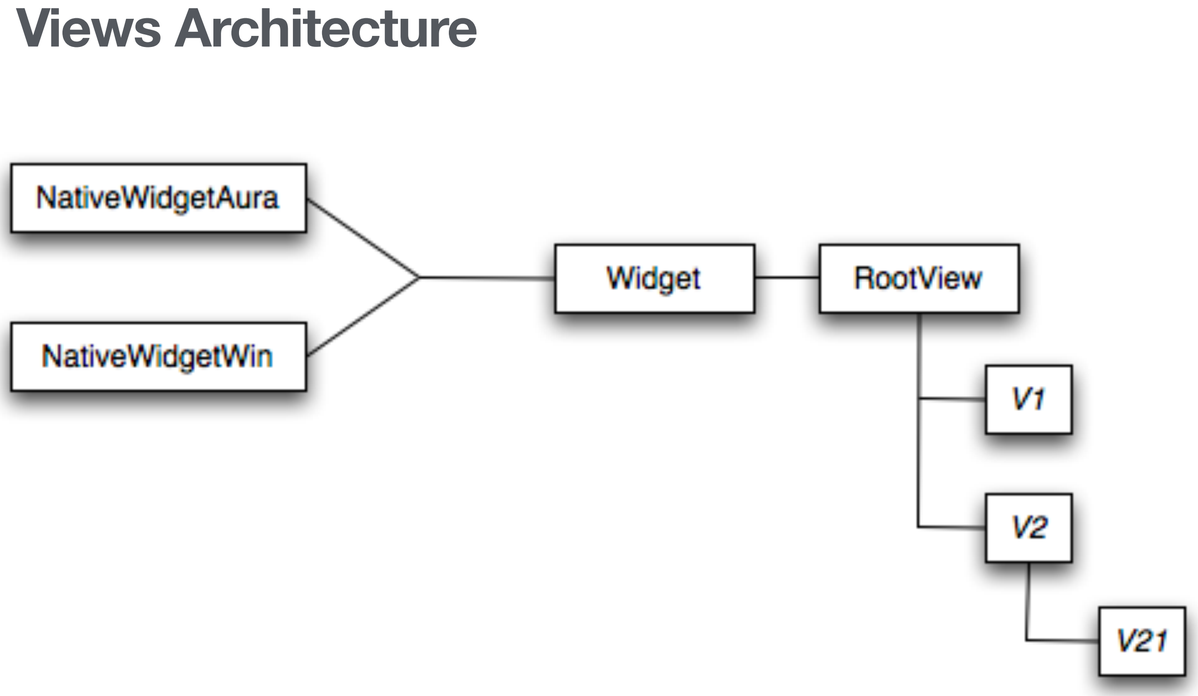
https://chromium.googlesource.com/chromium/src.git/+/refs/heads/main/docs/ui/index.md

5.2 views::Widget

각 OS가 제공하는 네이티브 캔버스로
여기에 View tree를 그림

View 트리와 네이티브 윈도우간 Bridge 역할
- 키보드 포커스 관리
- 윈도우 크기 관리
- 입력 라우팅

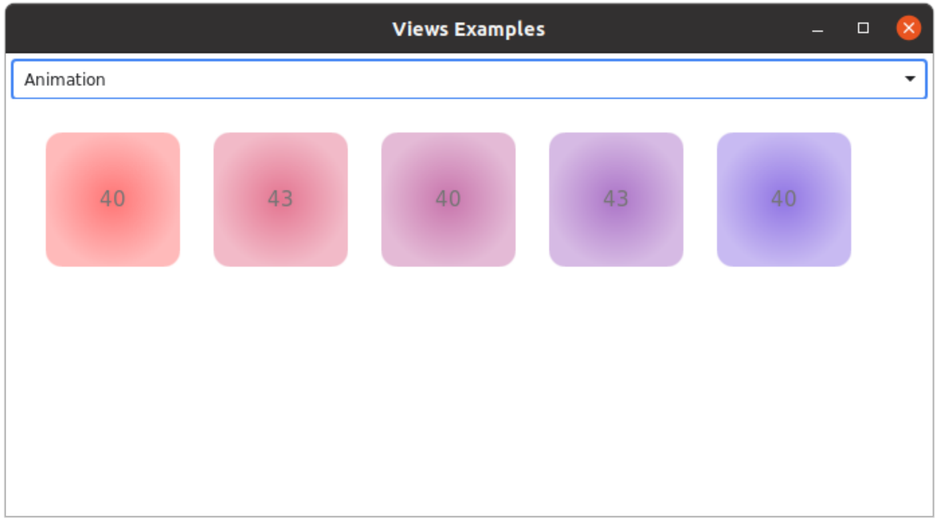
Widget도 계층적으로 구성됨

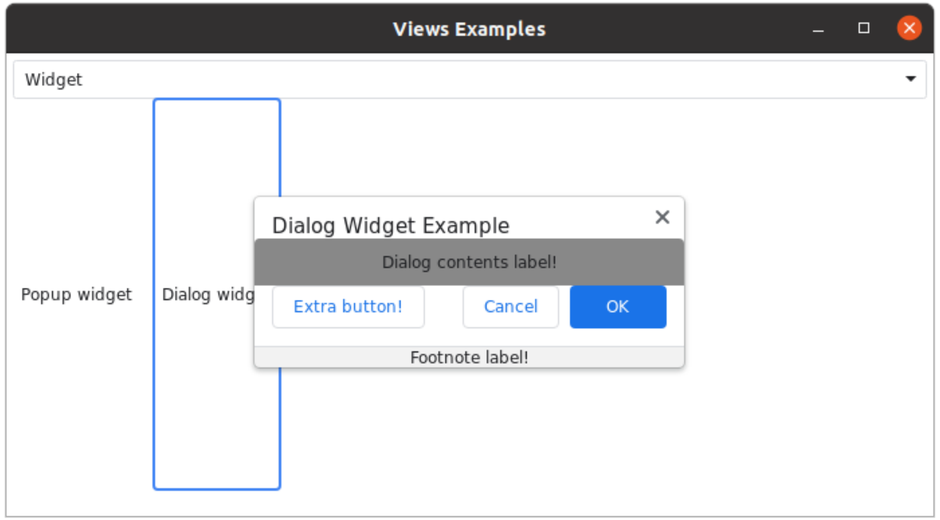


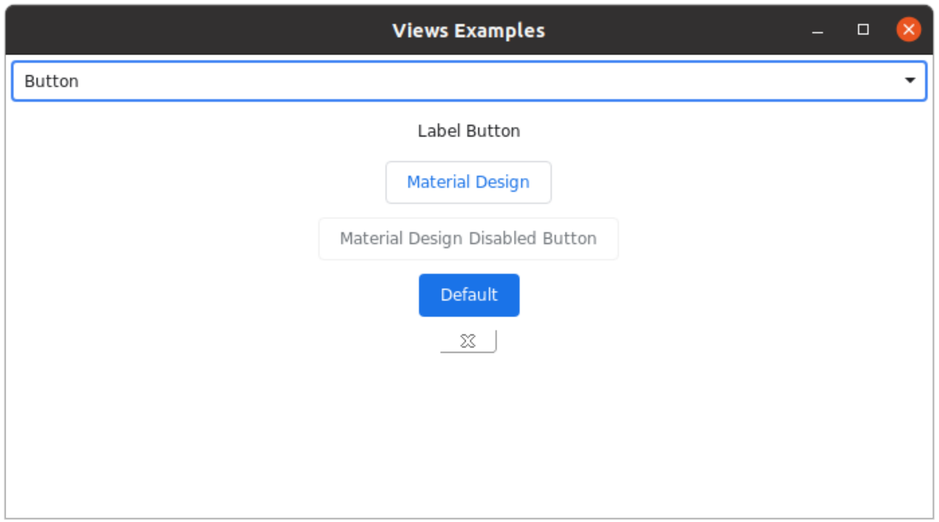
https://chromium.googlesource.com/chromium/src.git/+/refs/heads/main/docs/ui/index.md

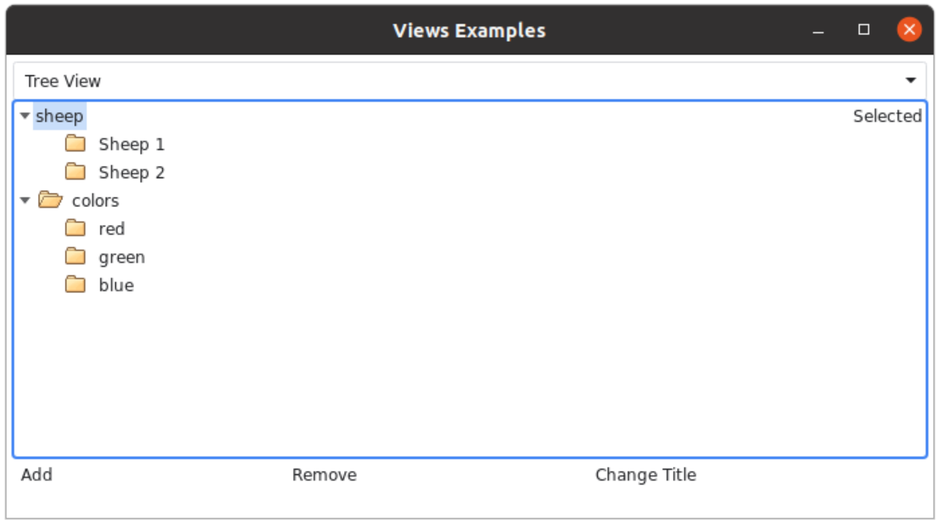
5.3 Views Examples

DEVIEW 2021










```
src$ autoninja -C out/debug views_examples
```

6. WebLayer

6.1 WebLayer

브라우저를 만들기 위한 High-Level Embedding API

- 최신 브라우저 기능 포함
- 크롬의 재사용가능한 버전
  - 사용자 권한을 얻는 UI
  - 자동완성 기능
  - 세이프 브라우징, etc.

크로미움 앱을 만드는 시작 포인트로 추천

Mac을 지원하지 않음

6.2 weblayer:Main()

main()에서 weblayer::Main() 실행
- 메인 메시지 루프 시작

weblayer:MainParams
- delegate: 메시지 루프 시작 전/후 동작
- pak_name: 앱의 리소스 파일

```
weblayer::MainParams CreateMainParams() {
  static const base::NoDestructor<MainDelegateImpl> weblayer_delegate;
  weblayer::MainParams params;
  params.delegate = const_cast<MainDelegateImpl*>(&(*weblayer_delegate));
  params.pak_name = "qlabs_common.pak";
  return params;
}


int main(int argc, const char** argv) {
  base::AtExitManager exit_manager;
  return weblayer::Main(CreateMainParams(), argc, argv);
}
```

6.2 weblayer::MainDelegate

DEVIEW 2021

Note: main()은 여러 번 호출되므로 꼭 MainDelegate를 사용해야 함

weblayer::MainDelegate
- 메시지 루프 시작 전, 초기화
- 메시지 루프 종료 후, 리소스 해제
- 메시지 루프 종료 클로저 처리

```
class MainDelegateImpl : public weblayer::MainDelegate {
 public:
  void PreMainMessageLoopRun() override;

  void PostMainMessageLoopRun() override;

  void SetMainMessageLoopQuitClosure(base::OnceClosure quit_closure) override;

 private:
  ...
};
```

6.2 weblayer::MainDelegate

메시지 루프 시작 전, 초기화

- 윈도우 상태 저장 변수, ViewsDelegate 생성
- Screen 초기화
- 프로필 생성
- UI 셋업

```
void PreMainMessageLoopRun() override {
  wm_state_ = std::make_unique<wm::WMState>();

  views_delegate_ = std::make_unique<views::DesktopTestViewsDelegate>();

  if (!display::Screen::GetScreen()) {
    screen_ = views::CreateDesktopScreen();
    display::Screen::SetScreenInstance(screen_.get());
  }

  profile_ = weblayer::Profile::Create("test_widgets", false);

  root_widget_delegate_ = SetUpWidgets();
}
```

6.2 weblayer::MainDelegate

메시지 루프 종료 후, 리소스 해제

```
void PostMainMessageLoopRun() override {
  screen_.reset();
  profile_.reset();
  views_delegate_.reset();
  wm_state_.reset();
}
```

6.2 weblayer::MainDelegate

DEVIEW 2021

메시지 루프 종료 클로저 처리

- quit_closure를 생성한 widget이 닫힐 때 호출되도록 세팅함

```
void SetMainMessageLoopQuitClosure(base::OnceClosure quit_closure) override {
  DCHECK(root_widget_delegate_);
  root_widget_delegate_->SetOnCloseCallback(std::move(quit_closure));
}
```

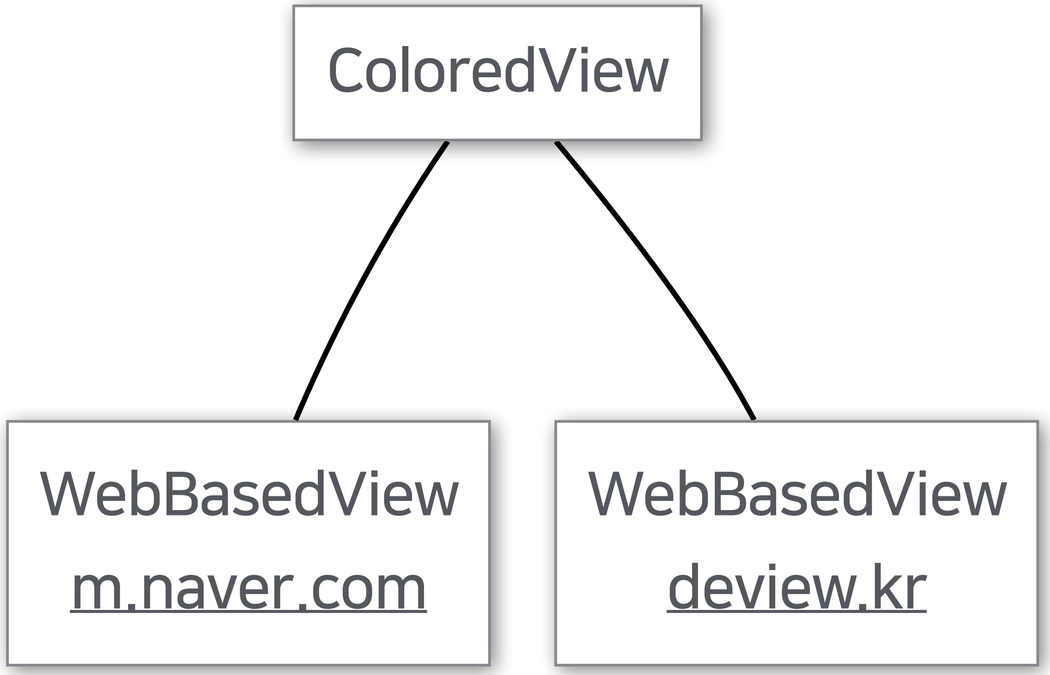
6.3 Set up Widgets

DEVIEW 2021

views::WidgetDelegateView 상속 받은 클래스로 UI 구성

- ColoredView
- WebBasedView
  - views::WebView, weblayer:Tab

```
ColoredView* SetUpWidgets() {
  ColoredView* delegate1 = new ColoredView();
  auto rect1 = gfx::Rect(100, 100, 500, 500);
  views::Widget* widget1 = CreateWidget(delegate1, nullptr, rect1);
  widget1->Show();

  WebBasedView* delegate2 = new WebBasedView(profile_.get());
  auto rect2 = gfx::Rect(50, 50, 300, 300);
  views::Widget* widget2 = CreateWidget(delegate2, widget1, rect2);
  delegate2->LoadUrl(GURL("https://m.naver.com"));
  WebviewPainter::ApplyTo(delegate2->GetWebView());
  widget2->Show();

  WebBasedView* delegate3 = new WebBasedView(profile_.get());
  auto rect3 = gfx::Rect(150, 150, 300, 300);
  views::Widget* widget3 = CreateWidget(delegate3, widget1, rect3);
  delegate3->LoadUrl(GURL("https://deview.kr/2021"));
  WebviewPainter::ApplyTo(delegate3->GetWebView());
  widget3->Show();

  return delegate1;
}
```



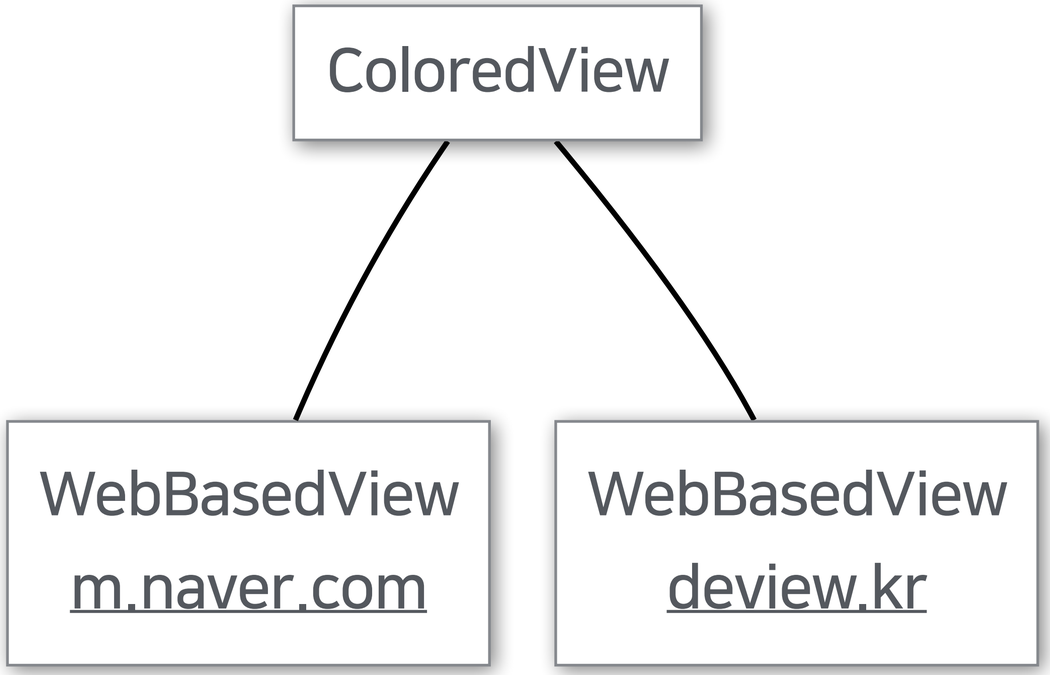
6.3 Set up Widgets

DEVIEW 2021

views::WidgetDelegateView 상속 받은 클래스로 UI 구성

- ColoredView
- WebBasedView
  - views::WebView, weblayer:Tab

```
ColoredView* SetUpWidgets() {
  ColoredView* delegate1 = new ColoredView();
  auto rect1 = gfx::Rect(100, 100, 500, 500);
  views::Widget* widget1 = CreateWidget(delegate1, nullptr, rect1);
  widget1->Show();

  WebBasedView* delegate2 = new WebBasedView(profile_.get());
  auto rect2 = gfx::Rect(50, 50, 300, 300);
  views::Widget* widget2 = CreateWidget(delegate2, widget1, rect2);
  delegate2->LoadUrl(GURL("https://m.naver.com"));
  WebviewPainter::ApplyTo(delegate2->GetWebView());
  widget2->Show();

  WebBasedView* delegate3 = new WebBasedView(profile_.get());
  auto rect3 = gfx::Rect(150, 150, 300, 300);
  views::Widget* widget3 = CreateWidget(delegate3, widget1, rect3);
  delegate3->LoadUrl(GURL("https://deview.kr/2021"));
  WebviewPainter::ApplyTo(delegate3->GetWebView());
  widget3->Show();

  return delegate1;
}
```



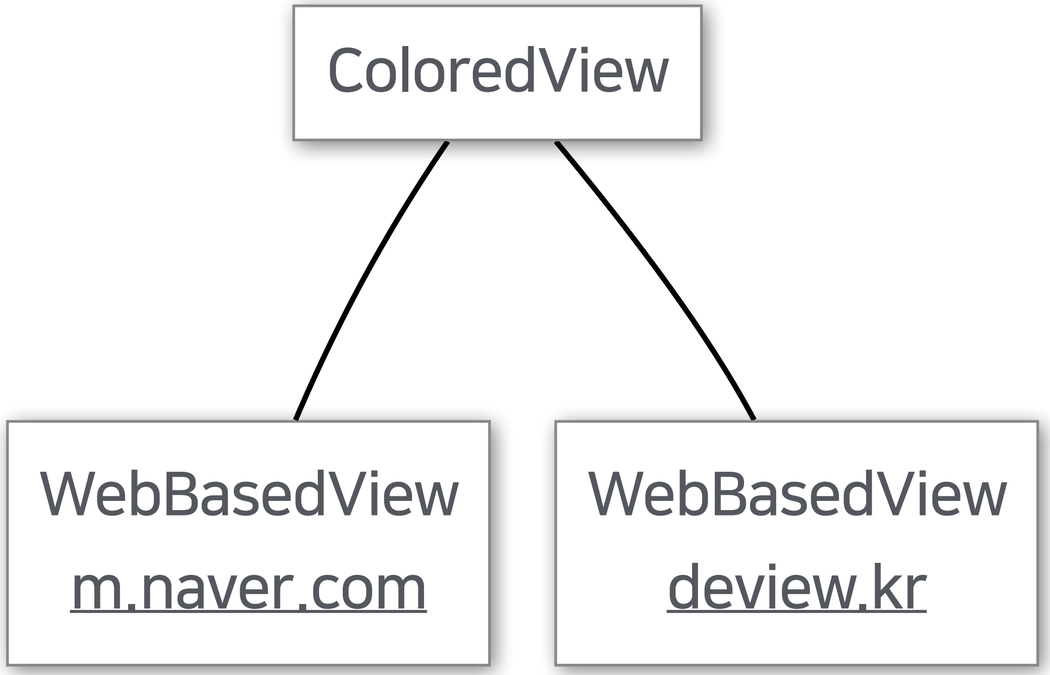
6.3 Set up Widgets

DEVIEW 2021

views::WidgetDelegateView 상속 받은 클래스로 UI 구성

- ColoredView
- WebBasedView
  - views::WebView, weblayer:Tab

```
ColoredView* SetUpWidgets() {
  ColoredView* delegate1 = new ColoredView();
  auto rect1 = gfx::Rect(100, 100, 500, 500);
  views::Widget* widget1 = CreateWidget(delegate1, nullptr, rect1);
  widget1->Show();

  WebBasedView* delegate2 = new WebBasedView(profile_.get());
  auto rect2 = gfx::Rect(50, 50, 300, 300);
  views::Widget* widget2 = CreateWidget(delegate2, widget1, rect2);
  delegate2->LoadUrl(GURL("https://m.naver.com"));
  WebViewPainter::ApplyTo(delegate2->GetWebView());
  widget2->Show();

  WebBasedView* delegate3 = new WebBasedView(profile_.get());
  auto rect3 = gfx::Rect(150, 150, 300, 300);
  views::Widget* widget3 = CreateWidget(delegate3, widget1, rect3);
  delegate3->LoadUrl(GURL("https://deview.kr/2021"));
  WebViewPainter::ApplyTo(delegate3->GetWebView());
  widget3->Show();

  return delegate1;
}
```



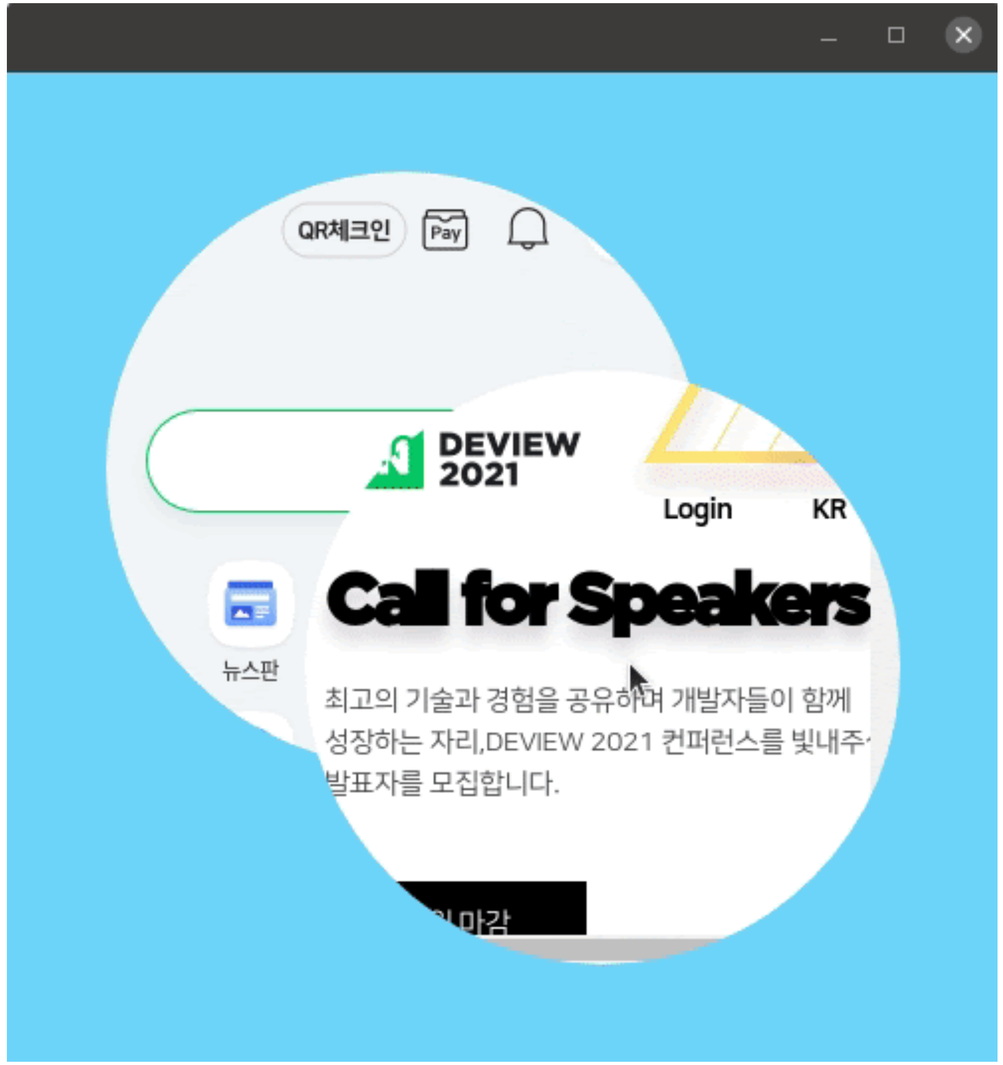
6.3 Set up Widgets

DEVIEW 2021

views::WidgetDelegateView 상속 받은 클래스로 UI 구성

- ColoredView
- WebBasedView
  - views::WebView, weblayer:Tab

```
ColoredView* SetUpWidgets() {
  ColoredView* delegate1 = new ColoredView();
  auto rect1 = gfx::Rect(100, 100, 500, 500);
  views::Widget* widget1 = CreateWidget(delegate1, nullptr, rect1);
  widget1->Show();

  WebBasedView* delegate2 = new WebBasedView(profile_.get());
  auto rect2 = gfx::Rect(50, 50, 300, 300);
  views::Widget* widget2 = CreateWidget(delegate2, widget1, rect2);
  delegate2->LoadUrl(GURL("https://m.naver.com"));
  WebviewPainter::ApplyTo(delegate2->GetWebView());
  widget2->Show();

  WebBasedView* delegate3 = new WebBasedView(profile_.get());
  auto rect3 = gfx::Rect(150, 150, 300, 300);
  views::Widget* widget3 = CreateWidget(delegate3, widget1, rect3);
  delegate3->LoadUrl(GURL("https://deview.kr/2021"));
  WebviewPainter::ApplyTo(delegate3->GetWebView());
  widget3->Show();

  return delegate1;
}
```



6.4 JavaScript와 C++간 통신

DEVIEW 2021

C++ -> JavaScript

- JavaScript string을 Tab::ExecuteScript()로 실행
- base::Value로 결과 콜백을 받음
- Note: Page loading이 완료될 때까지 기다려야 함

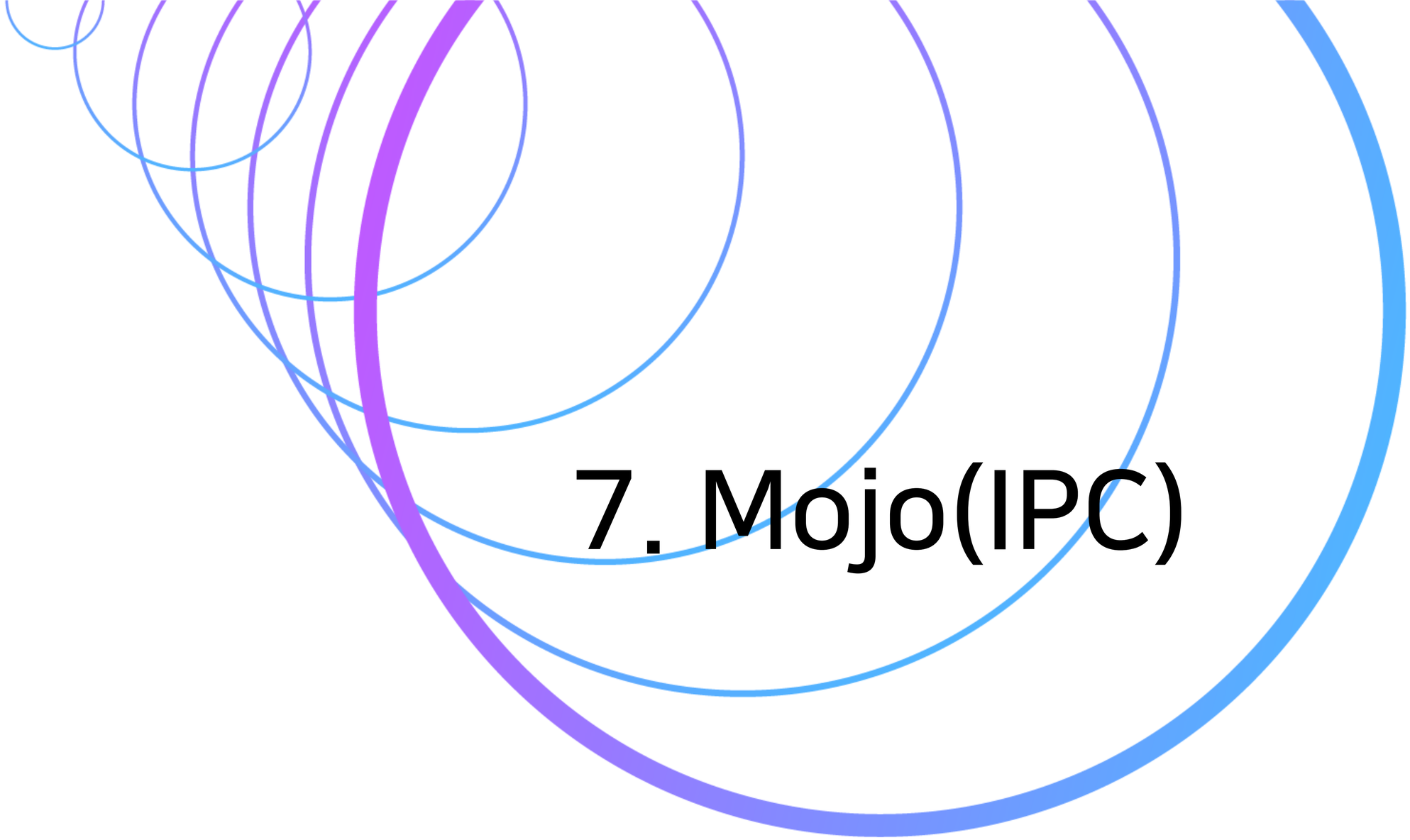
```
using JavaScriptResultCallback = base::OnceCallback<void(base::Value)>;

void Tab::ExecuteScript(const base::string16& script,
                        bool use_separate_isolate,
                        JavaScriptResultCallback callback);
```

6.4 JavaScript와 C++간 통신

DEVIEW 2021

JavaScript -> C++

- Blink 엔진에 새 JavaScript API를 추가
- Mojo를 이용해서 Blink 엔진 수정 없이 JavaScript API 추가



7. Mojo(IPC)

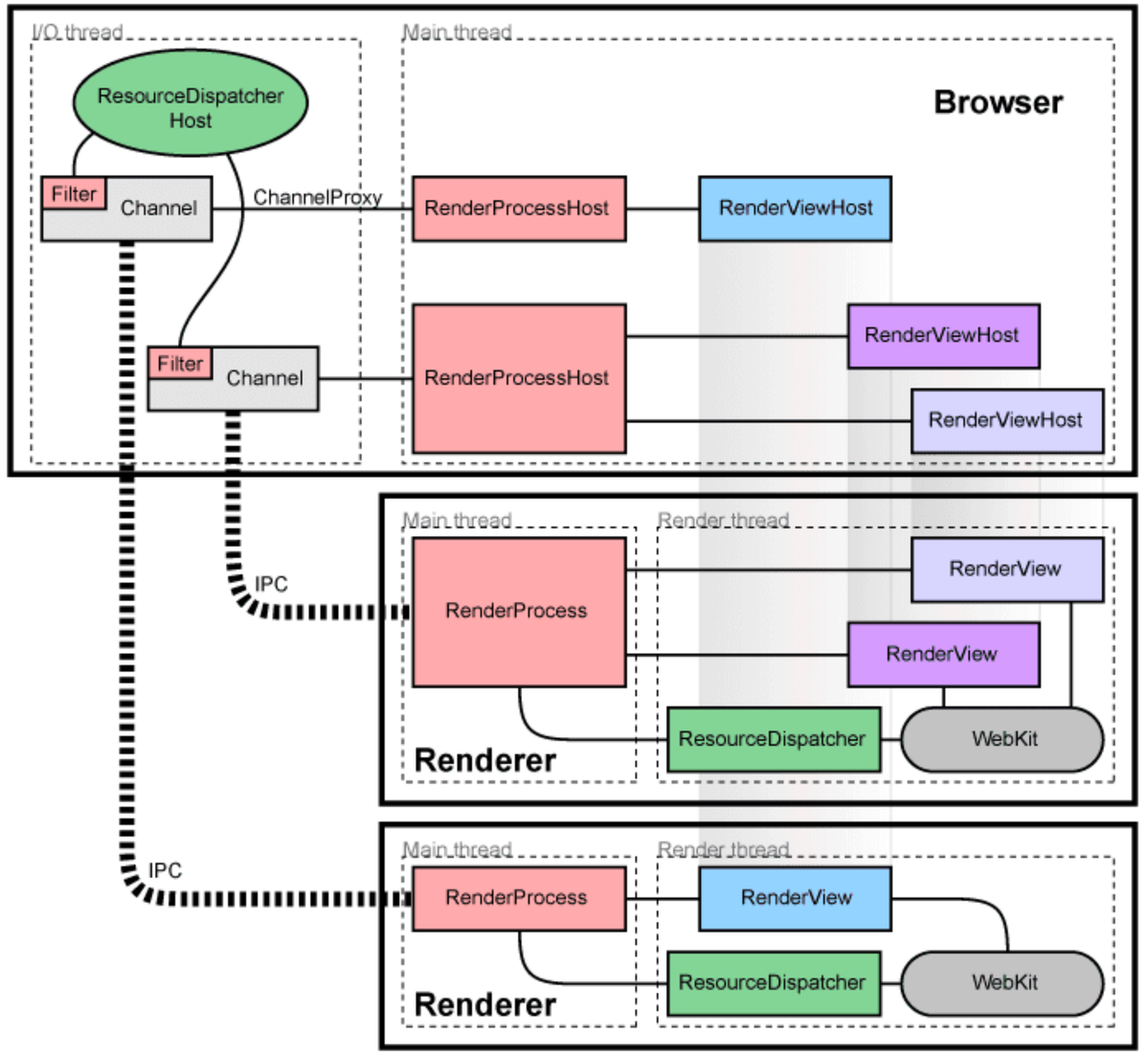
7.1 크로미움 브라우저 프로세스 구조

멀티 프로세스 구조

- Browser Process
  - UI 실행, 탭 관리
  - 렌더러의 시스템 리소스 접근 제어 (파일, 네트워크 등)
- Renderer Processes
  - HTML 해석, 레이아웃

목적

- 페이지가 죽어도 브라우저가 죽으면 안됨
- OS가 제공하는 메모리 보호와 액세스 제어의 이점을 가져옴



7.2 Mojo

Chromium이 제공하는 IPC 메커니즘

기능

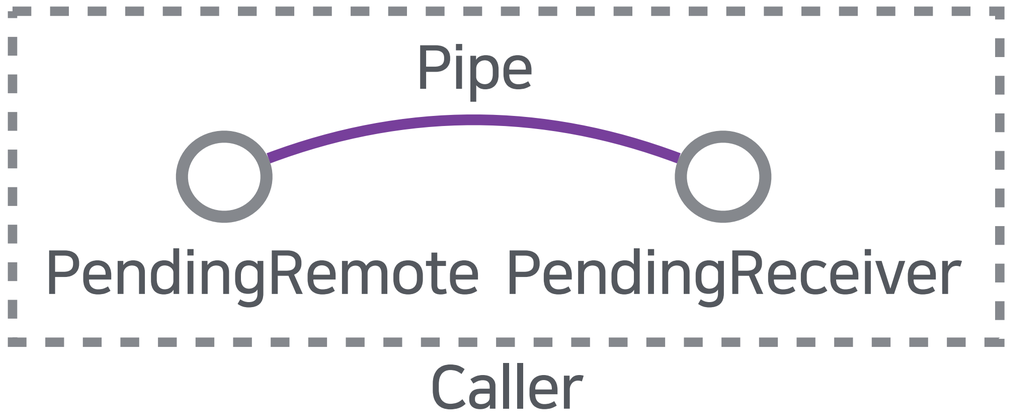
- Message Pipe
- Data Pipe
- Shared Buffer

언어 바인딩

- C++
- Java
- JavaScript

7.3 Mojo Message Pipe

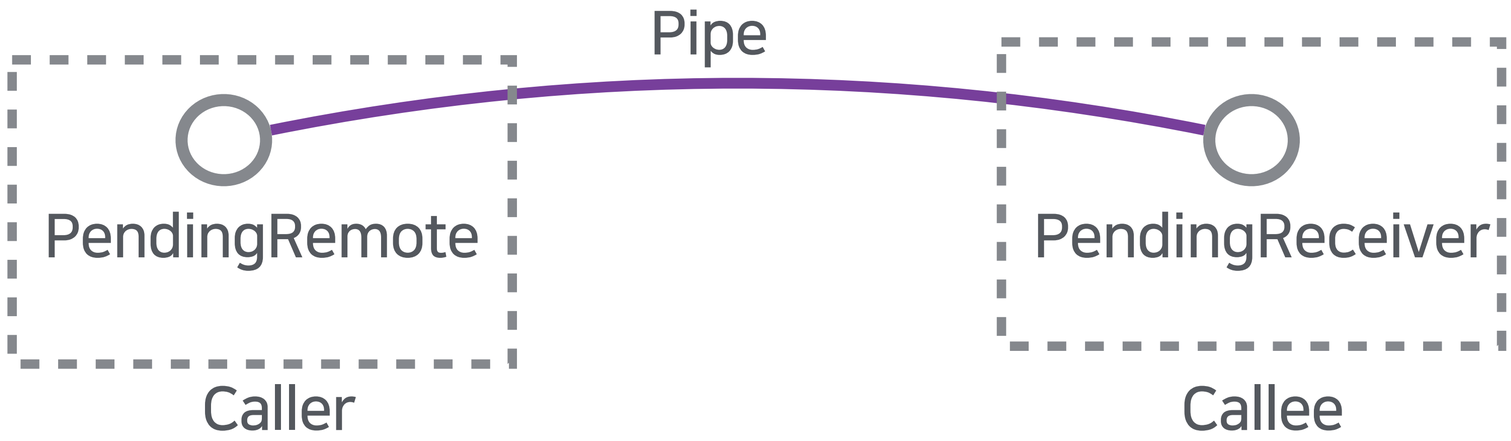
1. Pipe 생성



파이프 하나를 만들면 엔드 포인트 두 개가 생김

7.3 Mojo Message Pipe

2. Pipe 종단 하나를 통신할 대상에 전달



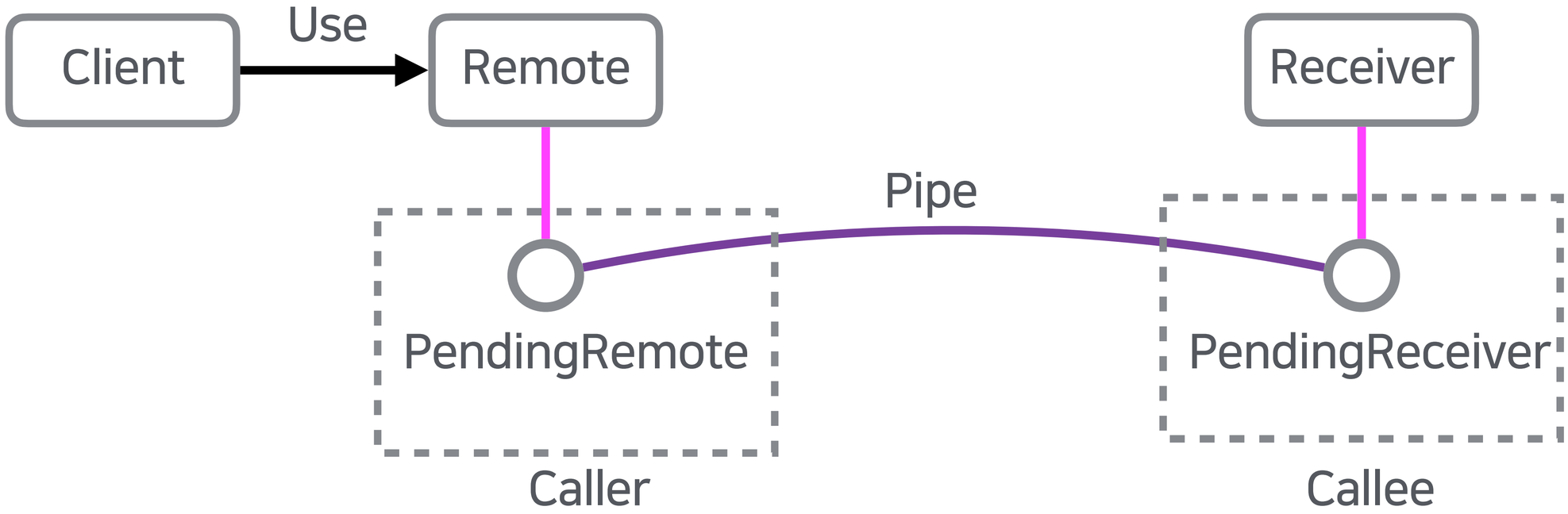
엔드 포인트 하나를 다른쪽으로 넘겨줘야 함

처음 브라우저와 차일드 프로세스 간에 메시지 파이프를 미리 만들어둠

이 메시지 파이프를 통해 엔드 포인트를 전달함

7.3 Mojo Message Pipe

3. Pipe Binding



7.4 Echo Example

Example: JavaScript에서 Echo 인터페이스를 통해 C++의 EchoImpl 서비스를 사용

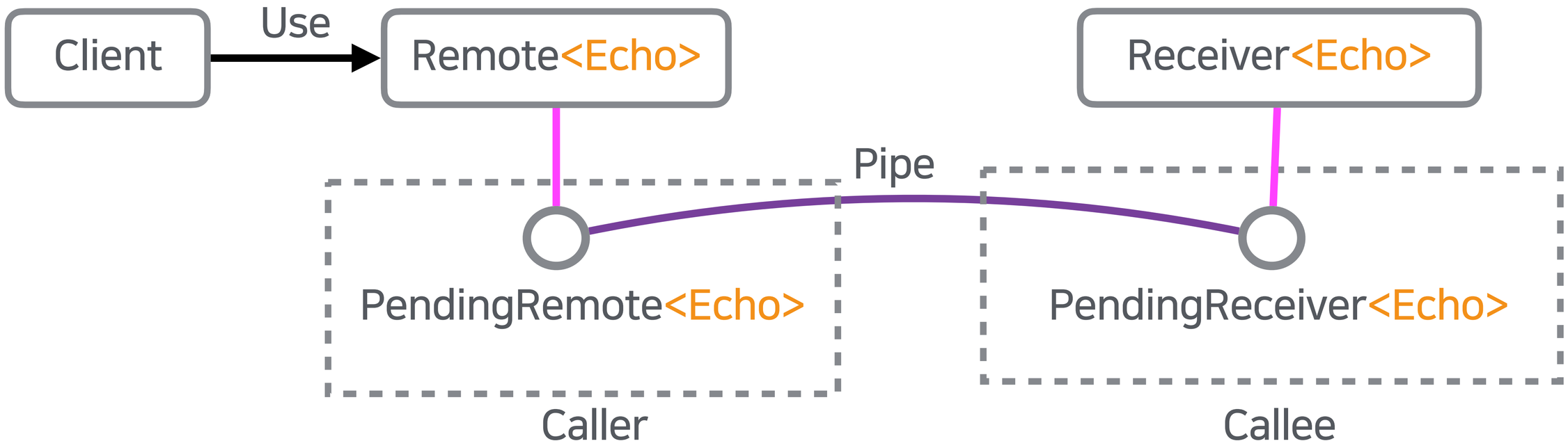
echo.mojom.Echo 정의

Mojom qlabs/learn_weblayer/echo/echo.mojom

```
module echo.mojom;

interface Echo {
  Execute(string request) => (string result);
};
```

7.4 Echo Example

Example: JavaScript에서 Echo 인터페이스를 통해 C++의 EchoImpl 서비스를 사용



7.4 Mojo 인터페이스 정의

Echo 인터페이스를 정의하고 빌드하면
스텁 코드들이 생성된다.

Mojom qlabs/learn_weblayer/echo/echo.mojom

```
module echo.mojom;

interface Echo {
  Execute(string request) => (string result);
};
```

GN qlabs/learn_weblayer/echo/BUILD.gn

```
import("//mojo/public/tools/bindings/mojom.gni")

mojom("mojo_bindings") {
  sources = [ "echo.mojom" ]
}
```

7.4 Mojo 인터페이스 정의

Echo 인터페이스를 정의하고 빌드하면
스텁 코드들이 생성된다.

Mojom qlabs/learn_weblayer/echo/echo.mojom

```
module echo.mojom;

interface Echo {
  Execute(string request) => (string result);
};
```

GN qlabs/learn_weblayer/echo/BUILD.gn

```
import("//mojo/public/tools/bindings/mojom.gni")

mojom("mojo_bindings") {
  sources = [ "echo.mojom" ]
}
```

Generated {out_dir}/gen/qlabs/learn_weblayer/echo

```
echo.mojom.cc
echo.mojom-forward.h
echo.mojom.h
echo.mojom.html
echo.mojom-import-headers.h

echo.mojom.js
echo.mojom-lite-for-compile.js
echo.mojom-lite.js
echo.mojom.m.js

echo.mojom-module
echo.mojom-params-data.h

echo.mojom-shared.cc
echo.mojom-shared.h
echo.mojom-shared-internal.h
echo.mojom-shared-message-ids.h

echo.mojom-test-utils.cc
echo.mojom-test-utils.h

mojo_bindings_blink.typemap_config
mojo_bindings.build_metadata
```

7.4 Mojo 인터페이스 정의

Echo 인터페이스를 정의하고 빌드하면
스텁 코드들이 생성된다.

Mojom qlabs/learn_weblayer/echo/echo.mojom

```
module echo.mojom;

interface Echo {
  Execute(string request) => (string result);
};
```

GN qlabs/learn_weblayer/echo/BUILD.gn

```
import("//mojo/public/tools/bindings/mojom.gni")

mojom("mojo_bindings") {
  sources = [ "echo.mojom" ]
}
```

Build

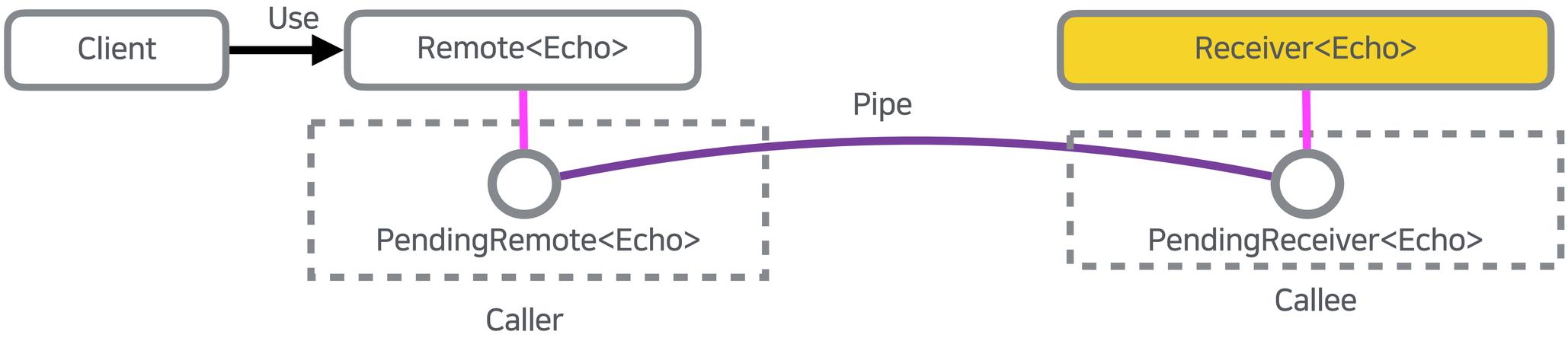
Generated {out_dir}/gen/qlabs/learn_weblayer/echo

```
echo.mojom.cc
echo.mojom-forward.h
echo.mojom.h
echo.mojom.html
echo.mojom-import-headers.h

echo.mojom.js
echo.mojom-lite-for-compile.js
echo.mojom-lite.js
echo.mojom.m.js

echo.mojom-module
echo.mojom-params-data.h

echo.mojom-shared.cc
echo.mojom-shared.h
echo.mojom-shared-internal.h
echo.mojom-shared-message-ids.h

echo.mojom-test-utils.cc
echo.mojom-test-utils.h

mojo_bindings_blink.typemap_config
mojo_bindings.build_metadata
```

서비스 구현

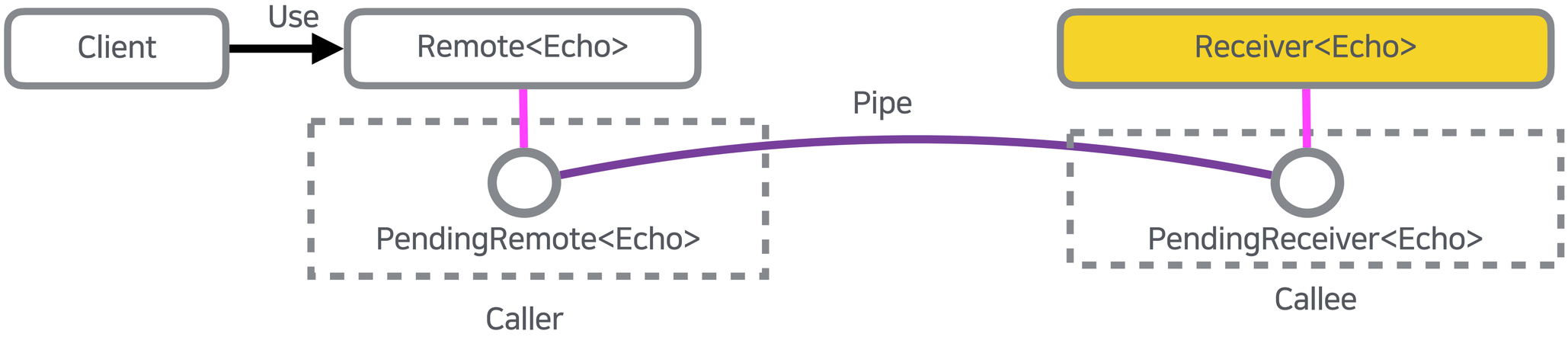
서비스 호출

7.5 Mojo 서비스 구현




```
#include "qlabs/learn_weblayer/echo/echo.mojom.h"


class EchoImpl : public echo::mojom::Echo {
 public:
  void BindInterface(mojo::PendingReceiver<echo::mojom::Echo> pending_receiver) {
    ...
    receiver_.Bind(std::move(pending_receiver));
  }

 private:
  void Execute(const std::string& request, ExecuteCallback callback) override {
    std::move(callback).Run(request);
  }

  mojo::Receiver<echo::mojom::Echo> receiver_{this};
};
```

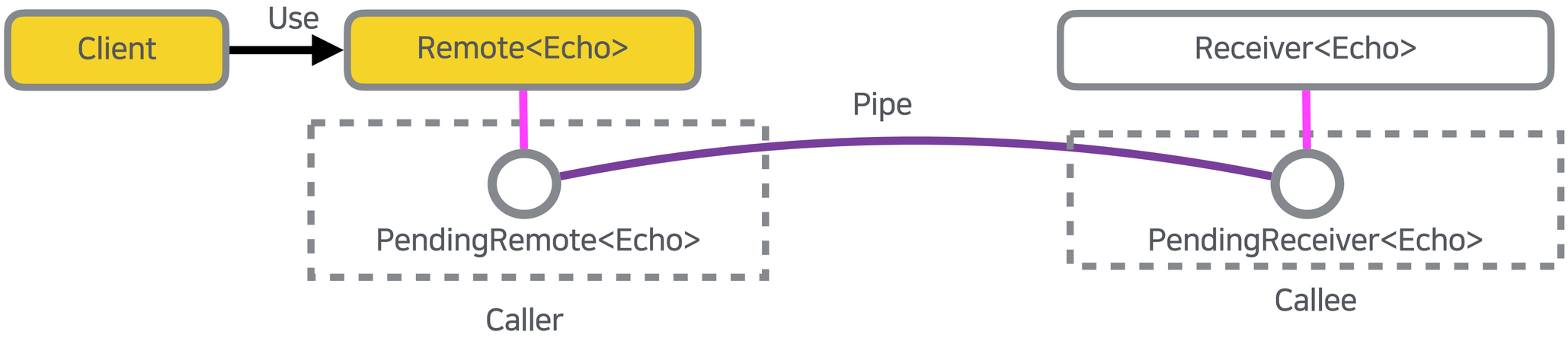
7.6 Mojo 인터페이스-서비스 매핑



weblayer/browser/weblayer_browser_interface_binders.cc

```
void BindEcho(content::RenderFrameHost* host,
              mojo::PendingReceiver<echo::mojom::Echo> receiver) {
  static qlabs::EchoImpl echo_impl;
  ...
  echo_impl.BindInterface(std::move(receiver));
}

void PopulateWebLayerFrameBinders(
    content::RenderFrameHost* render_frame_host,
    service_manager::BinderMapWithContext<content::RenderFrameHost*>* map) {
  map->Add<echo::mojom::Echo>(base::BindRepeating(&BindEcho));
  . . .
}
```

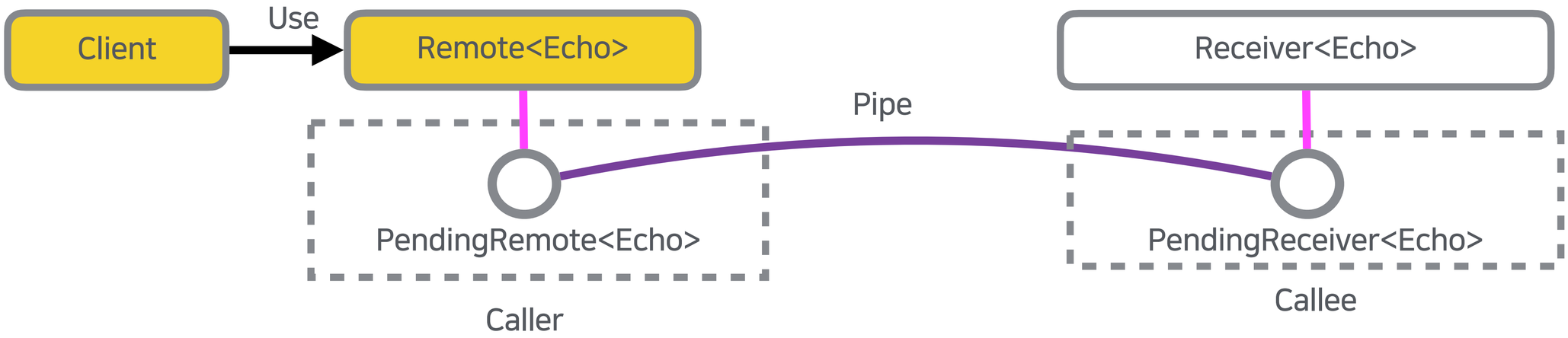
7.7 JavaScript에서 서비스 호출



qlabs/learn_weblayer/echo_server/public/main.js

```
import * as qlabs from './echo.mojom.m.js';

window.document.addEventListener('DOMContentLoaded', async function () {
  try {
    const echo = qlabs.Echo.getRemote();
    const { result } = await echo.execute('Hello, Mojo!');
    document.getElementById("greeting").textContent = result;
  } catch (e) {
    document.getElementById("greeting").textContent = e;
  }
});
```

7.7 JavaScript에서 서비스 호출



Terminal

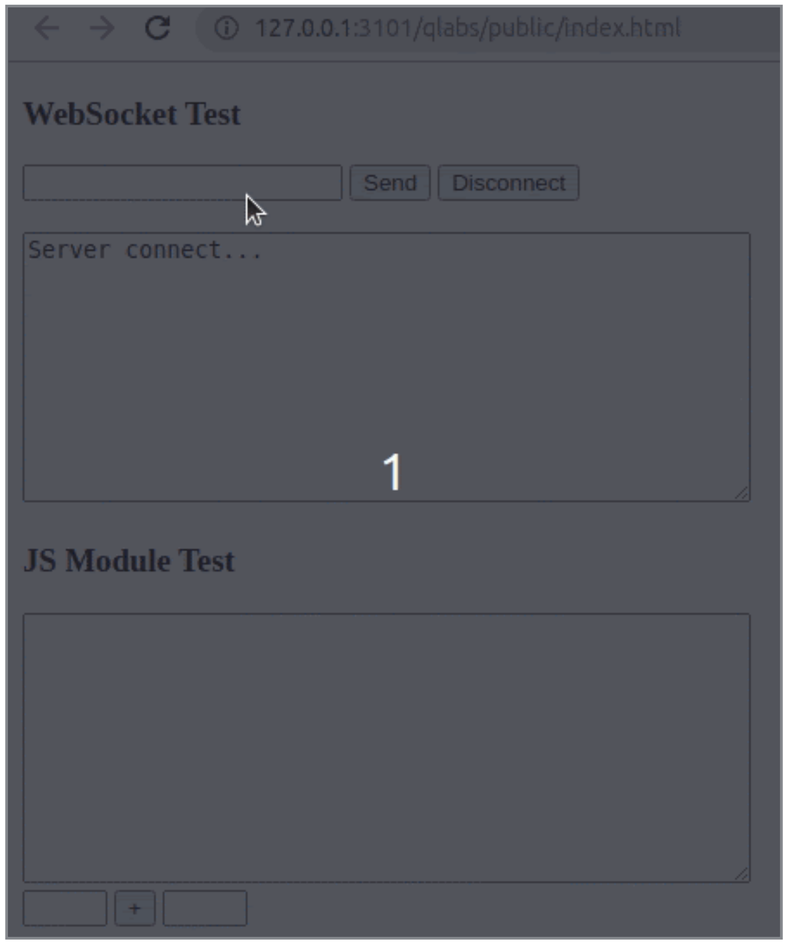
```
src$ out/debug/echo_demo\
—enable-blink-features=Mojo
```

8. Network

8.1 HTTP Server

DEVIEW 2021

HTTP Server 간단히 구현 가능




```
class MySocketFactory : public SimpleSocketFactory {
 public:
  std::unique_ptr<net::ServerSocket> CreateForHttpServer() override {
    return CreateLocalHostServerSocket(3101);
  }

 private:
  std::unique_ptr<net::ServerSocket> CreateLocalHostServerSocket(int port) {
    std::unique_ptr<net::ServerSocket> socket(
        new net::TCPServerSocket(nullptr, net::NetLogSource()));

    if (socket->ListenWithAddressAndPort("127.0.0.1", port, kBackLog) == net::OK)
      return socket;
    if (socket->ListenWithAddressAndPort("::1", port, kBackLog) == net::OK)
      return socket;

    return std::unique_ptr<net::ServerSocket>();
  }
};
```

```
SimpleHttpServer::GetInstance()->Start(
  std::make_unique<MySocketFactory>(),
  socket_dir,
  frontend_dir
);
```

9. 결론

9.1 결론

크로미움의 시스템 API와 UI Framework를 사용하여
플랫폼 독립적인 앱을 개발할 수 있습니다.

Base 모듈에 익숙해지면 비동기 로직을 처리하기 쉽습니다.

앱 개발의 기반으로 WebLayer를 사용할 수 있습니다.

- Multi-Process 기반 구조 이용
- IPC 설비인 Mojo를 사용해서 Process간 통신

9.2 주의할 점

DEVIEW 2021

Public SDK를 제공하지 않음

- 플랫폼별로 빌드된 SDK, 런타임 제공 않음
- Public API가 아님. API Reference 없음

Chromium 소스 트리 안에서 프로젝트를 관리해야 함

- src/ 디렉토리 밖에 프로젝트를 만들 수 없음

9.2 주의할 점

DEVIEW 2021

Android

- Android Studio 개발환경 지원 안 함
  - Android Gradle Plugin 대신 GN을 사용하여 빌드
- GN에서 Kotlin 빌드 지원 안 함

iOS

- 다른 웹엔진 탑재 불가

9.3 WhaleAuto PoC

IVI(In Vehicle Infortainment) System

- 차량 제어 및 상태 표시
- 앱 프레임워크 + 차량용 앱스토어
- 터치 기반 홈스크린
- 멀티스크린 지원

Thank You