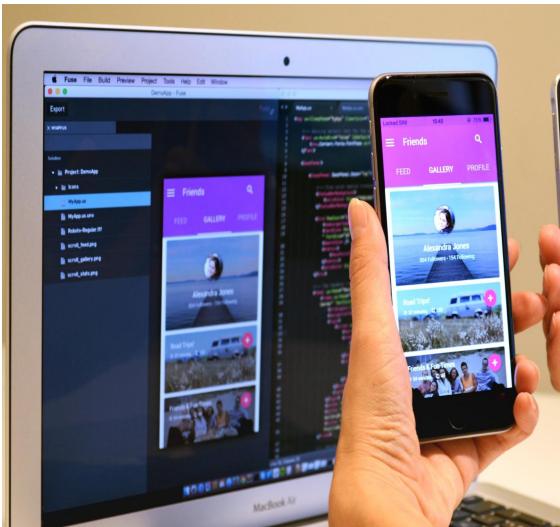


Fuse your potential



[호남대 특강]
Fuse를 활용한 빠른
모바일 앱 프로토타이핑 및 개발

fuse /

[Part 1]

- Fuse 소개
- Fuse 설치하기
- Fuse Studio 사용하기
- UX markup에 대한 기본적인 이해
- Layout 이해하기 (Panel, StackPanel, DockPanel, Grid, ...)
- Layout 활용 (Margin, Padding, Alignment, Units, ...)
- 스타일 적용하기
- 실전1) 앱 개별 화면 설계 – (Splash, 로그인, 메인, 서브 화면)
- Animation 이해 & 구현
- Navigation (PageControl, Router, Navigator, ...)
- 실전2) 앱 개별 화면 Navigation 연결
- UX markup 클래스 정의 및 재사용

[Part 2]

- 자바스크립트에 대한 이해 & 데이터 바인딩
- 실전3) 자바스크립트를 활용한 로그인 및 버튼 기능 연동
- 반복 (Each) 사용하기
- Rest API에 대한 이해 및 활용하기
- Fuse 프로젝트 파일 관리
- 앱 패키징을 위한 추가 설정
- Fuse에서 지원하는 다른 기능 살펴보기
- Q&A



[Part 1]

Fuse 소개

Fusetools 회사 소개

- Smart screen의 개척자로 출발
- 2006년 노르웨이 기업 Falanx가 ARM에 인수됨
- Mali 그래픽 프로세서 제작
- 삼성전자 최상위 모델에 100억 개 이상의 기기 사용
- 2012년 엔지니어와 제품관리 핵심 Fuse 설립
- Norway Oslo본사, 미국지사, 한국지사
- 디지털 세계의 새로운 혁명





Fuse: Fusetools 핵심 Product

- 2016년 포브스에
“10 Hottest Design Tools”
중 하나로 선정
 - 번역: Fuse는 App 디자이너 및 개발자를 위한 UX 툴 모음입니다. 여러 장치에서 동시에 실시간으로 네이티브 앱 룩앤파일 (look and feel)을 만들고 업데이트할 수 있습니다. 현재 OS X 및 Windows에서 사용 가능합니다.

Fuse



Fuse is the UX tool suite for app designers and developers. Image credit: Fuse.

Fuse is the UX tool suite for app designers and developer. It allows you to create and update the look and feel for native apps in real-time on multiple devices simultaneously. Currently available for OS X and Windows users.



화성에서 온 개발자 금성에서 온 디자이너

- 정확한 프로젝트의 Process 구축
- 가장 빠른 Rendering 작업
- UX에 강한 App 개발
- 시간과 비용의 단축



Fuse 의미: 앱 개발 전 과정에서의 소통

Designer

With 기획자
UI / UX 퍼블리셔
QA / 테스터
고객
...

Developer

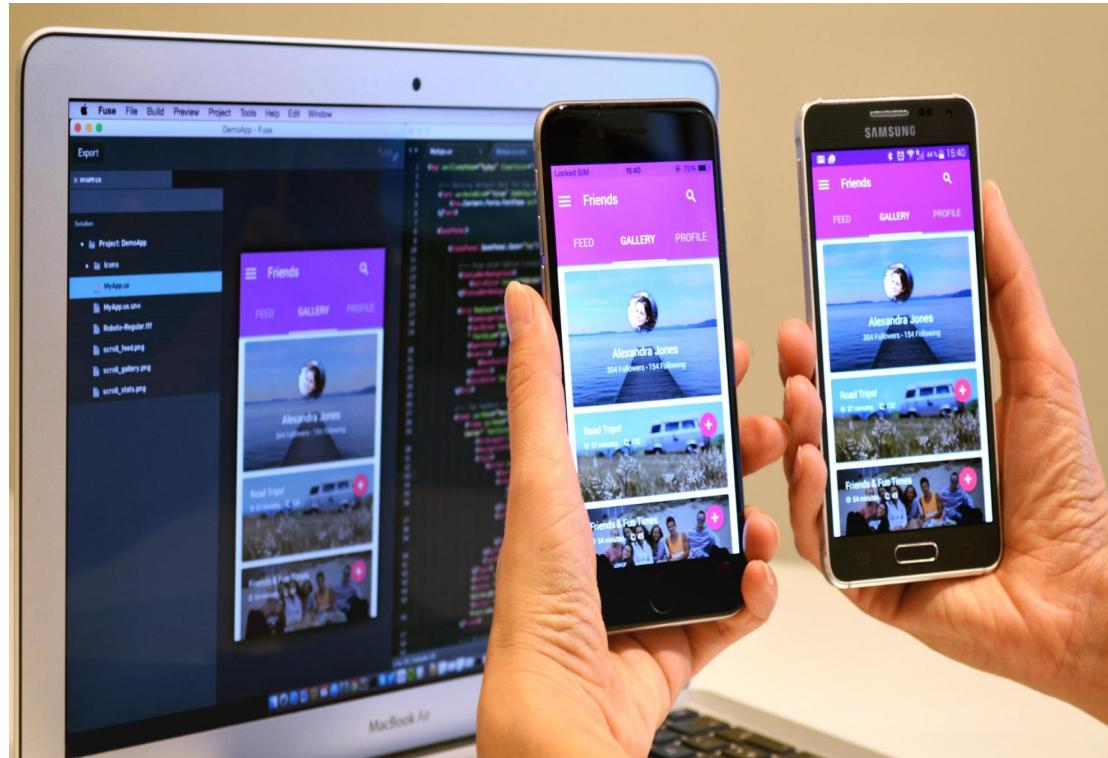
융합하다, 조화를 이루다

도화선, 새로움을 창출하다

fuse  creates a new business



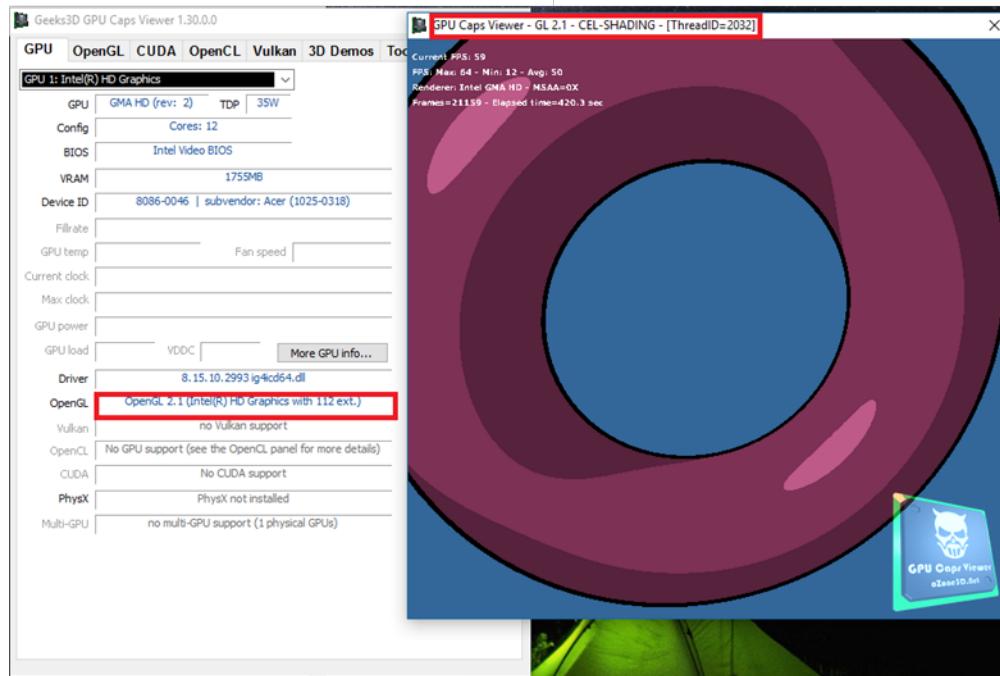
- Fuse 플랫폼은 개발자 업무를 단축해 주는 도구입니다.





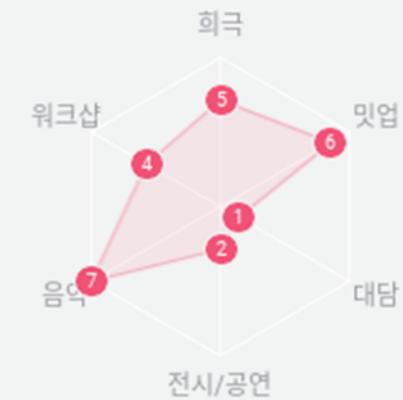
• GPU를 통한 렌더링

- 성능과 애니메이션에 기반을 둔 크로스플랫폼 App 개발 도구
- OpenGL 가속



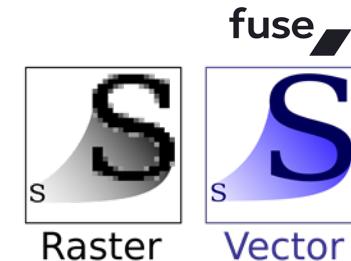
Fuse your potential

```
Build completed in 0.28 seconds  
---  
DEBUG: GL_VERSION: 2.1 INTEL-10.25.13  
DEBUG: GL_VENDOR: Intel Inc.  
DEBUG: GL_RENDERER: Intel(R) Iris(TM) Graphics 550
```



1월

2월



- XML기반의 마크업 언어로 UX 구현

```
<Each Items="{cities}">
  <Panel Margin="0" Height="400" Width="293">
    <Rotation Degrees="{degrees}" />

    <DockPanel Margin="0" Height="100%" Width="100%">
      <Grid ColumnData="auto,1*,auto" Margin="10" >
        <Image ux:Name="visited" File="Assets/VisitedIcon.png" />
        <StackPanel Alignment="VerticalCenter" >
          <Text Value="{name}" FontSize="18" />
          <Text Value="{country}" FontSize="12" />
        </StackPanel>
        <Image ux:Name="notVisited" File="Assets/NotVisitedIcon.png" />
      </Grid>

      <Image ux:Name="visitedOverlay" Stretch="Uniform" />
      <Image ux:Name="notVisitedOverlay" Stretch="Uniform" />
      <Image Source="{DataToResource imageKey}" />

      <Grid ColumnCount="2" Margin="0,0,0,0" >
        <StackPanel Padding="0,5,1,5" Background="White" >
          <Text FontSize="22" Alignment="Left" >{name}</Text>
          <Text FontSize="12" Alignment="Left" >{country}</Text>
        </StackPanel>
      </Grid>
    </DockPanel>
  </Panel>
</Each>
```

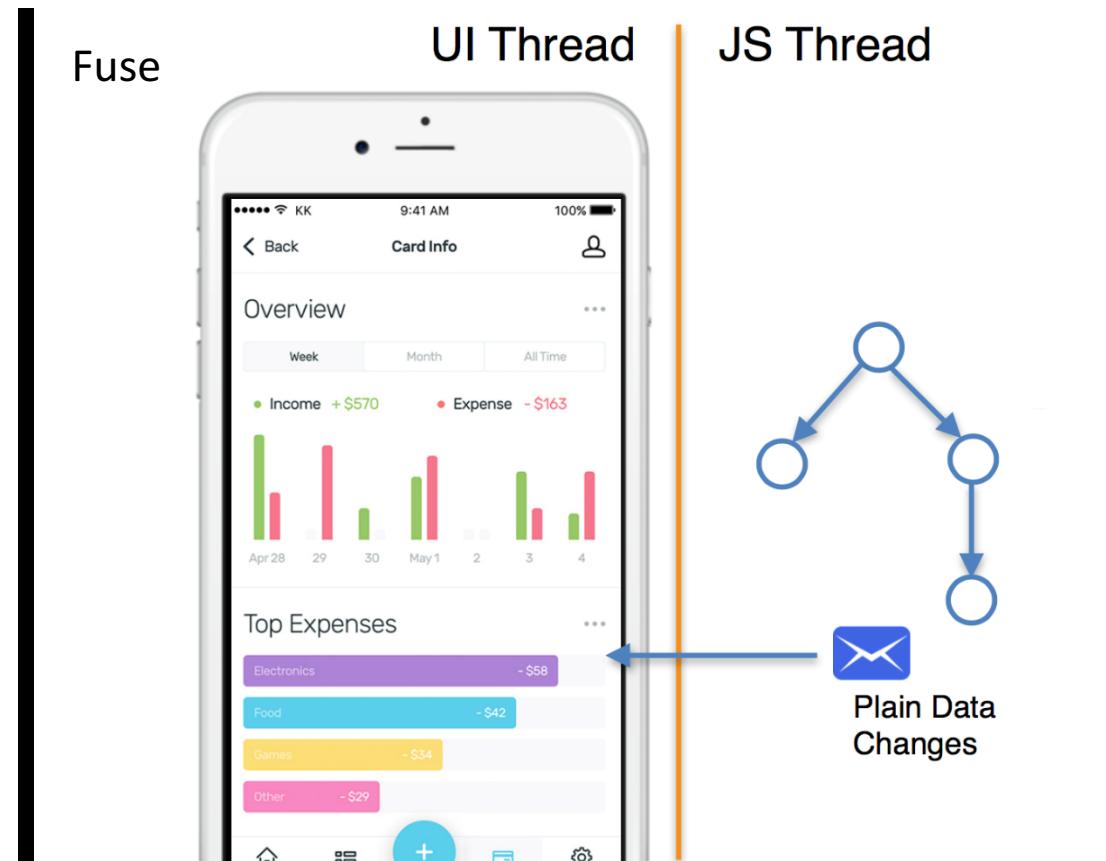
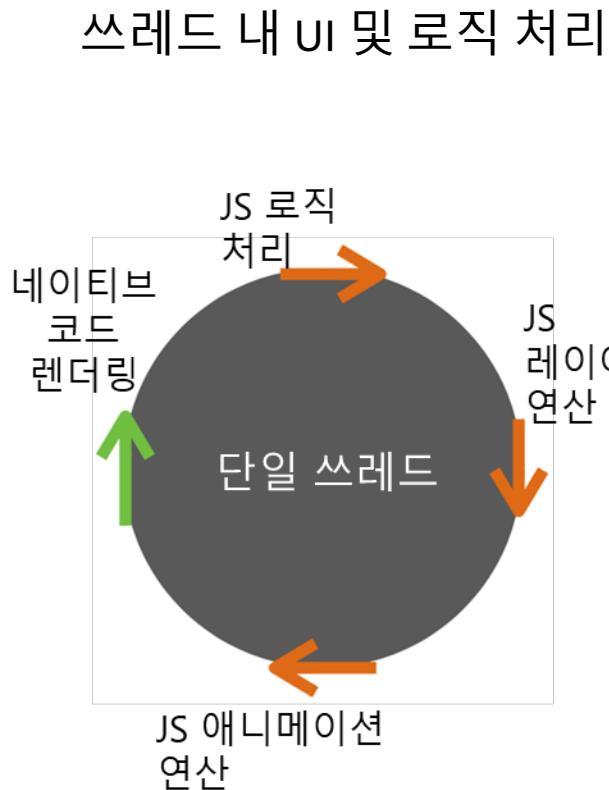
UX Markup

- UI elements and layout
- Animation and motion
- Gesture interpretation & response
- Navigation
- Data binding
- Event response

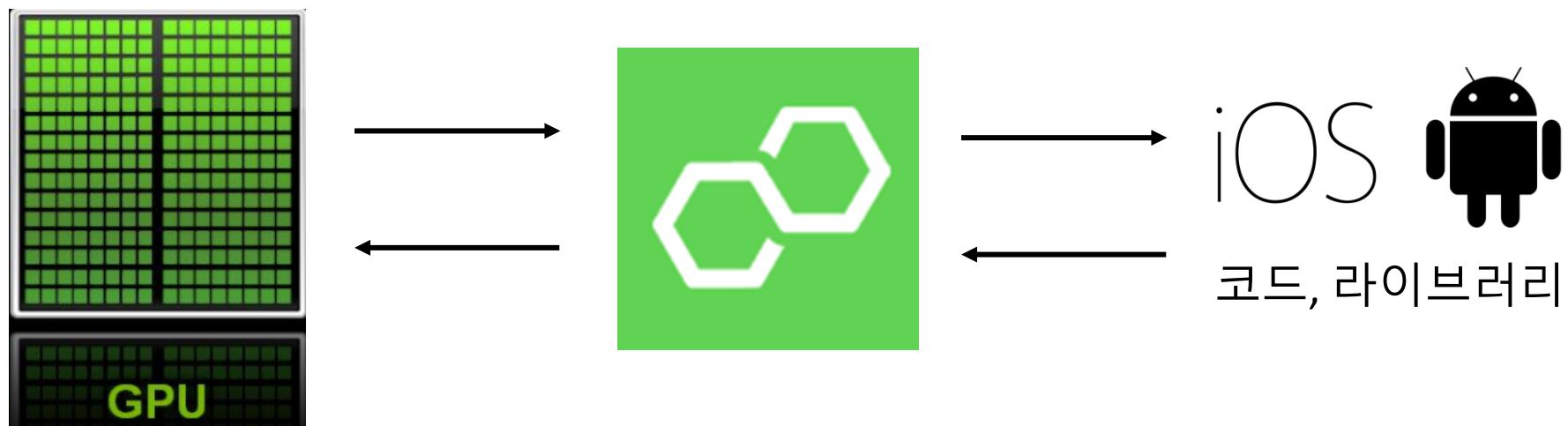


Fuse 플랫폼

- UI와 구별되는 별도의 쓰레드에서 동작하는 JavaScript를 통해 비즈니스 로직 구현

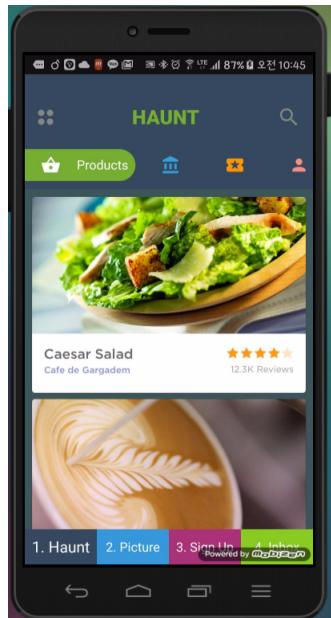


- Uno 플랫폼을 통해 C++, Objective-C, Java로 컴파일
 - C#과 매우 유사한 문법으로 최적화된 코드로 변환
 - Uno를 통해 기존 iOS / Android 코드 및 라이브러리를 Fuse에서 호출 가능

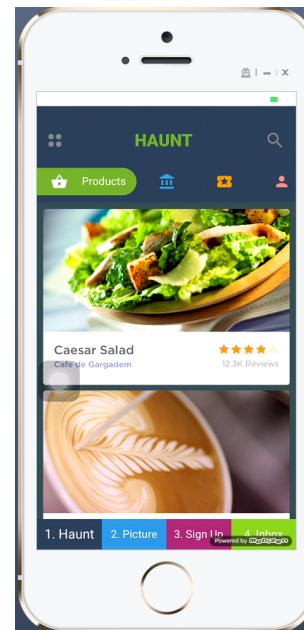




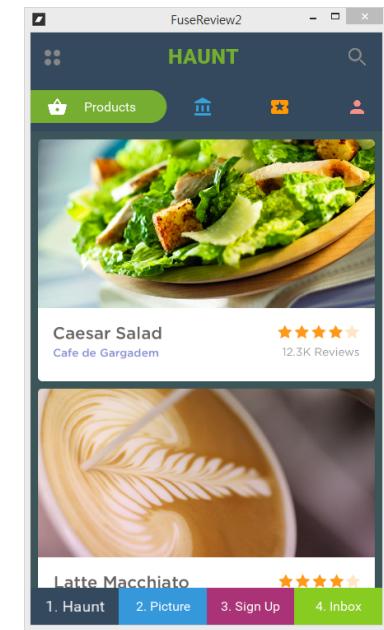
- Android, iOS 동시 지원
 - 코딩을 한 번에
 - 쉬운 마크업, 자바스크립트로 동시 구현



Android



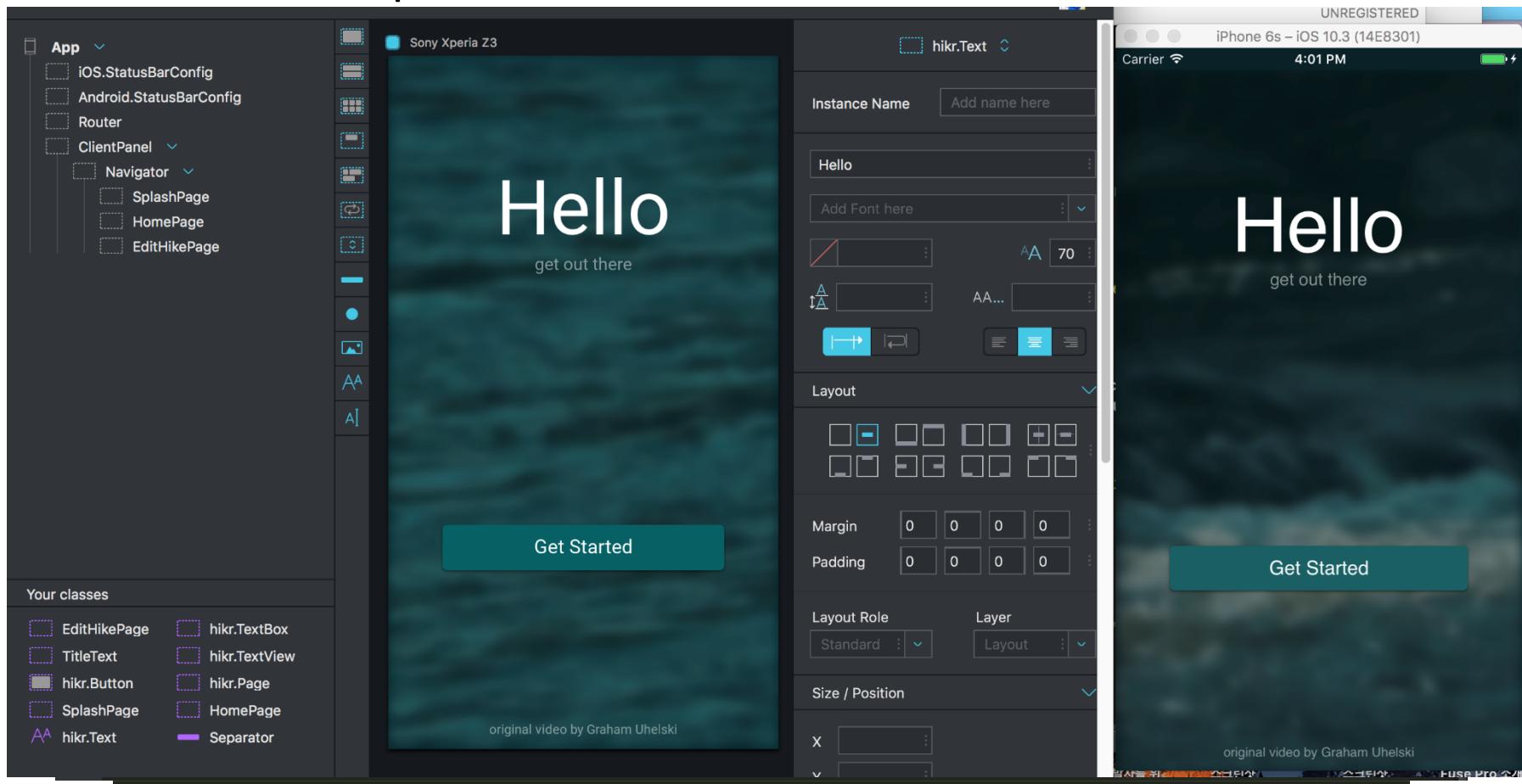
iOS



PC Application
(.NET- preview)

Fuse 플랫폼

- Viewer 및 Live Reloading으로 실시간 디자인 확인
 - One Compile로 개발시간 단축

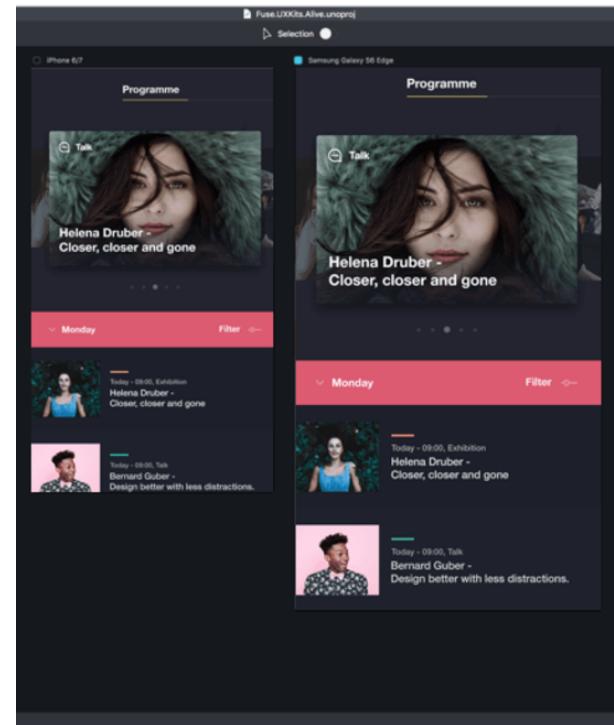
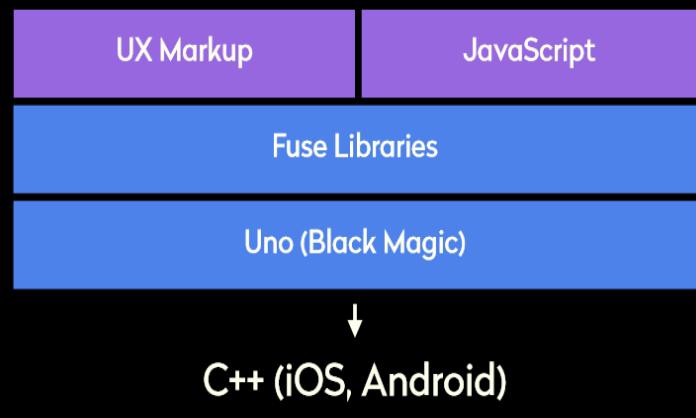


Fuse your potential

- 장점

- Animation & Navigation 개발 신의 한수
- 스마트해진 기술로 80% 코드가 줄어 듭니다.
- 쉽습니다: JavaScript와 XML로 구현합니다.

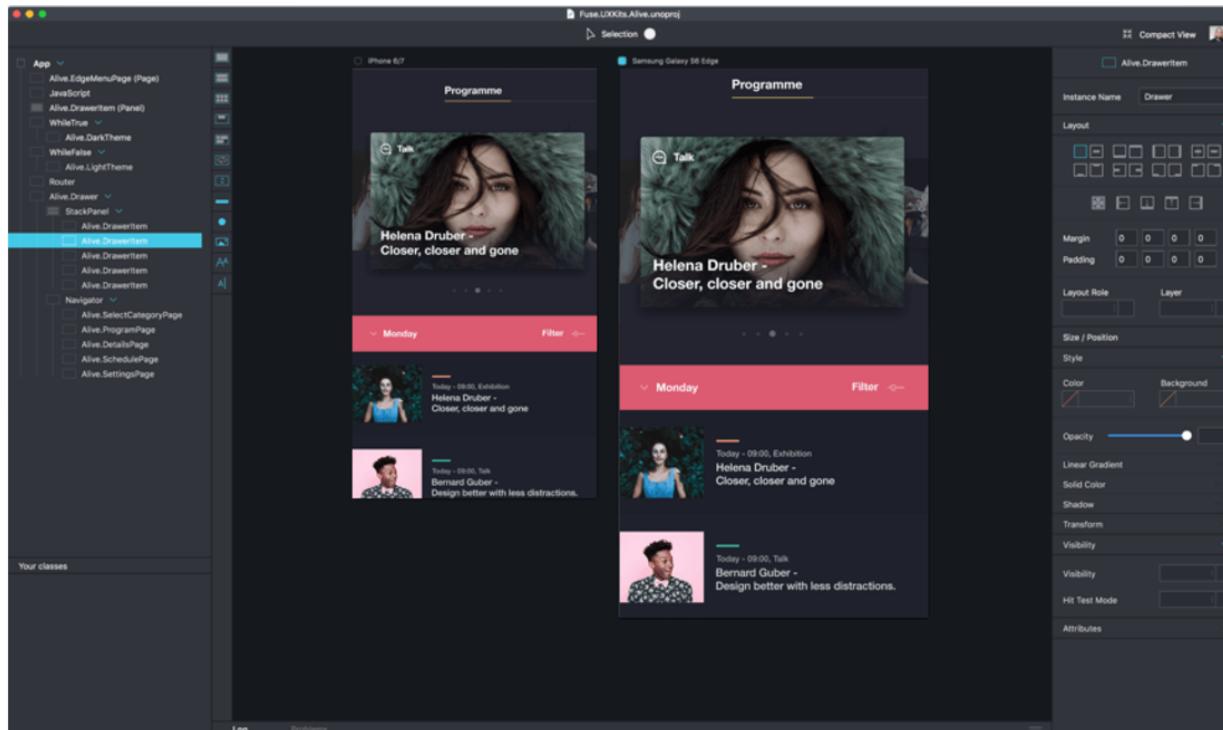
Technology Stack





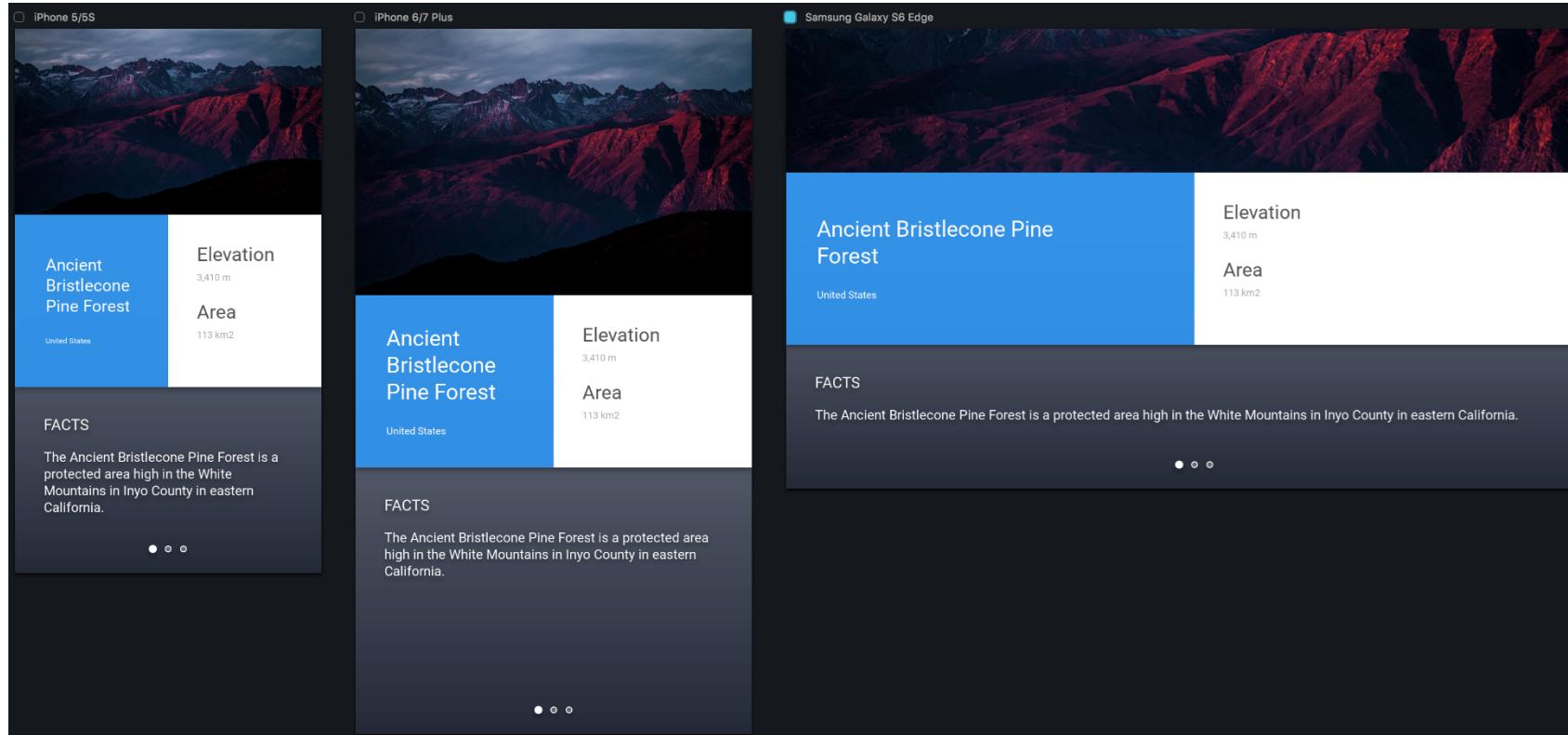
• Fuse Studio

- 시각적 도구를 사용하여 앱에서 요소를 쉽게 추가 및 수정
- 레이아웃과 프로젝트 구조를 더욱 직관적으로 표시
- 디자이너와 개발자 협업이 가능한 공간으로 작업 효율 향상





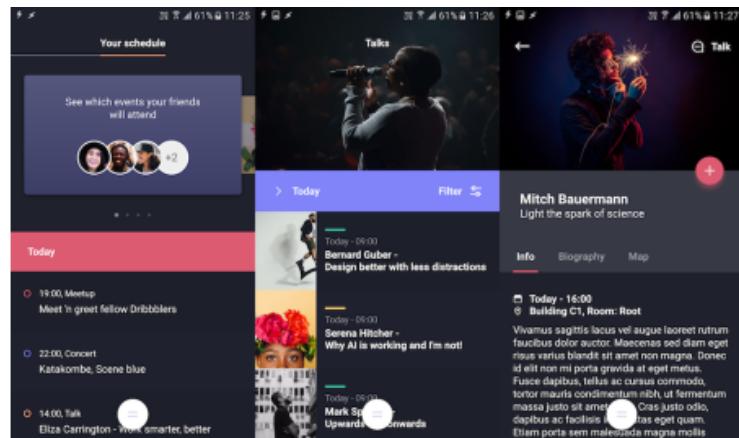
- Fuse Studio – 디자이너 및 기획자를 위한 Fuse
 - 앱 개발 결과를 WYSIWYG 기능 및 라이브 프리뷰 모드로 바로 확인
 - 멀티 뷰 포트 기능을 통해 장비 변경에 따른 해상도 대응



Fuse your potential

- Premium Component

- Xcode & Android Studio Integration
⇒ Native Code와 완벽하게 통합 될 수 있는
XCode & Android Studio 라이브러리 생성
- Chart
⇒ 앱에서 멋진 **그래프와 차트**를 쉽게 만들 수 있는 강력한 도구를 이용해 차별화된 앱, 살아있는 앱 구현
- CameraView
⇒ 앱에서 실시간 카메라 입력을 통합하고 맞춤 설정 및 Native 기능의 지속적인 추가지원
- Alive UX Kit
⇒ 광범위한 사용 사례를 위해 사용자 정의 가능한 UI 구성 요소를 종합적으로 제공

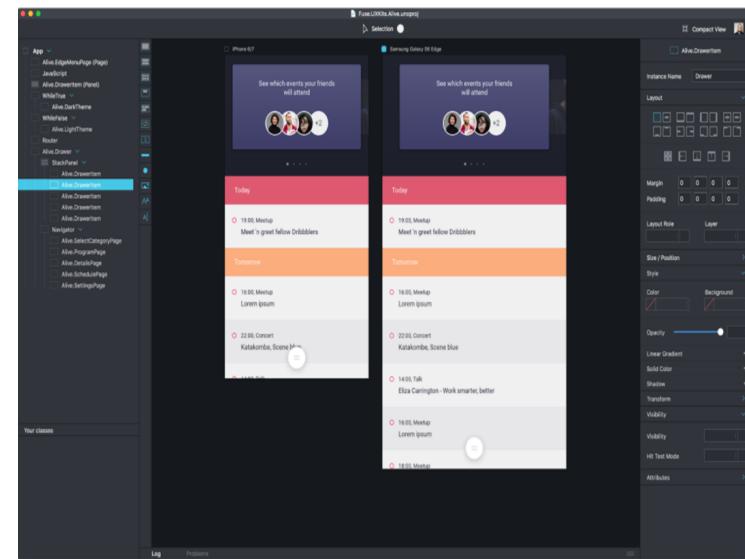


Fuse Pro를 활용한 디자인 & 앱 개발 워크플로우 개선

- 디자이너: 여러 해상도 & 변경 요소 즉시 확인
- 개발자: 비즈니스 로직에 집중



디자이너



```
<MyRectangle>
  <Tapped>
    <Scale Factor="0.8" Duration="0.3" Easing="QuadraticInOut"/>
    <Move Y="-0.3" Duration="0.3" Easing="QuadraticInOut" RelativeTo="Size"/>
  </Tapped>
</MyRectangle>
```

with



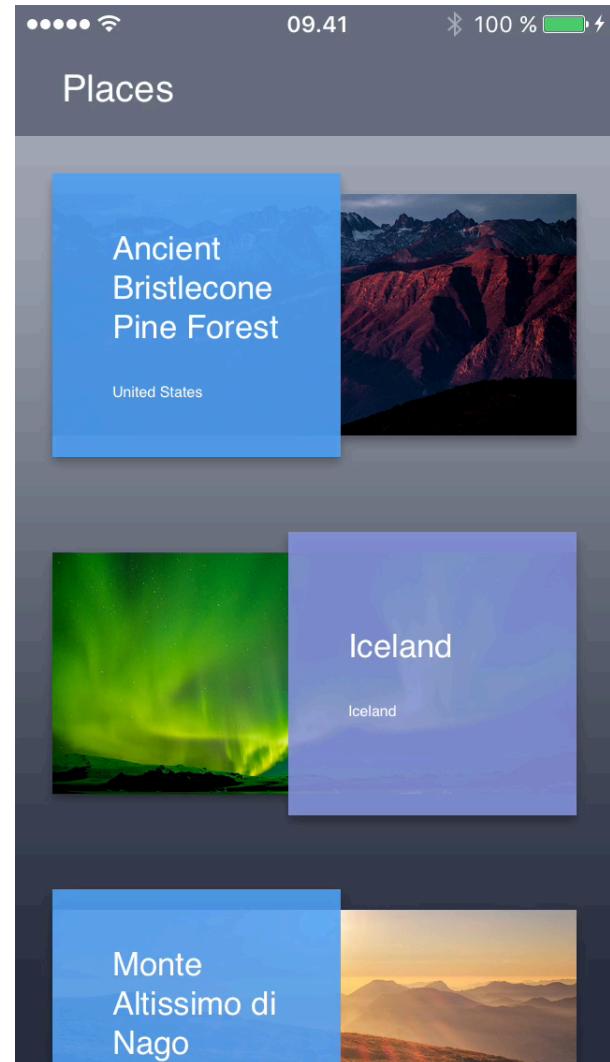
개발자

UI 코딩없이 직접 XML 코드를 작성할 수 있어 디자이너와 개발자 간 소통이 더 쉽고 간단해지며, 상세 설계서 작업 부담이 현저히 줄어들고, 개발자는 상세 설계서를 더 이상 자세히 들여다보지 않아도 됩니다.

Fuse 샘플

- 예제 중심으로 Fuse를 학습하고자 하는 경우
 - Fuse 홈페이지에서 예제 살펴보기
 - 관심있는 예제 다운로드
 - Fuse에서 정상 동작 확인
 - 원하는 색상, 텍스트 수정
 - ...

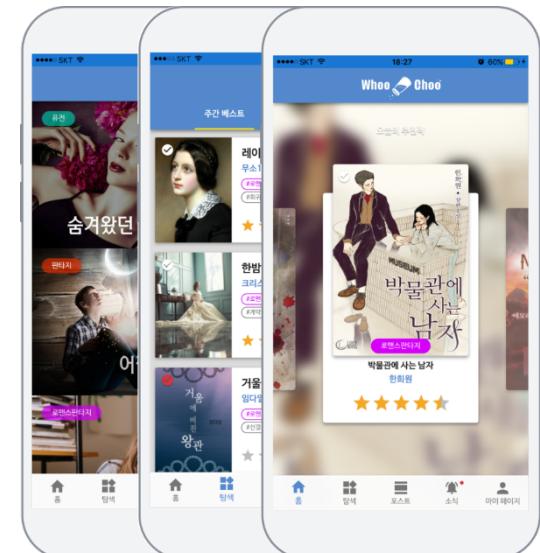
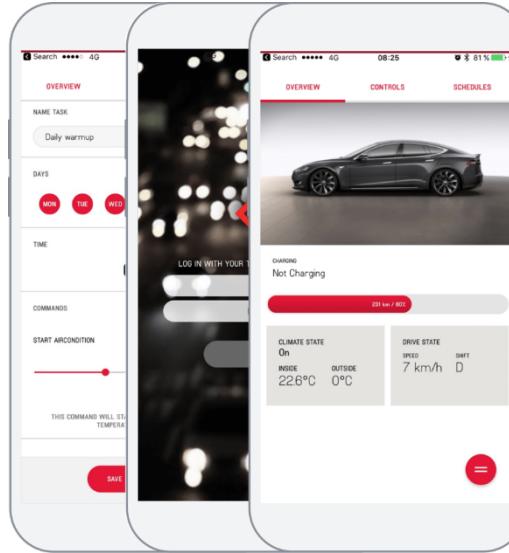
<https://www.fusetools.com/examples>



Fuse로 만든 앱

주요 앱

- Criminal Mind (Book)
- 후추 (웹 소설 추천)
- Tesla CTRL
- DEEM
- ...



<https://www.fusetools.com/showcases>

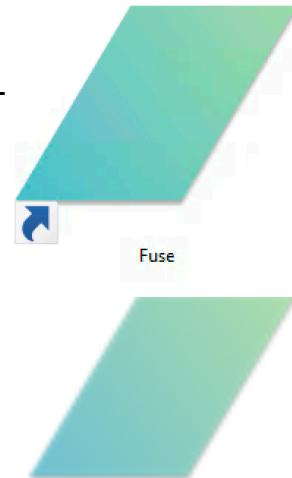
Fuse your potential

Fuse 설치하기

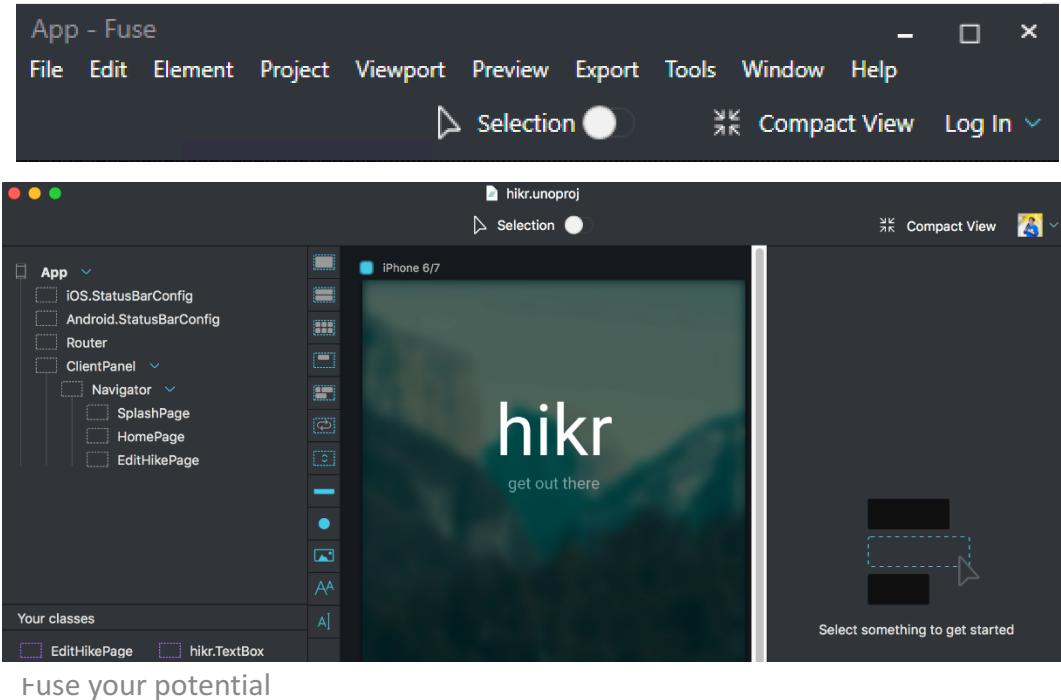
Fuse 설치하기

- 지원 플랫폼
 - macOS: 10.10 (Yosemite) 이상, 10.12 (Sierra) 권장
 - Windows: 7 이상
 - iOS 앱 패키징을 위해서는 macOS 환경 사용 필요
 - 베타 버전 OS는 미지원

Windows 아이콘



macOS 아이콘



Fuse 설치하기

- 하드웨어 요구 사항
 - 그래픽 카드: OpenGL 2.1을 지원하는 GPU
 - 예: Intel GMA – OpenGL 2.1 미지원
- 소프트웨어 요구 사항
 - Android SDK
 - Android 4.1 (Jelly Bean, API level 16) 이상 호환
 - (Fuse에서 쉽게 설치 가능)
 - 64비트 환경에서 사용 가능
 - Xcode
 - iOS 8.0 이상을 지원하는 Xcode 개발 환경 필요
 - Apple App Store에서 직접 설치

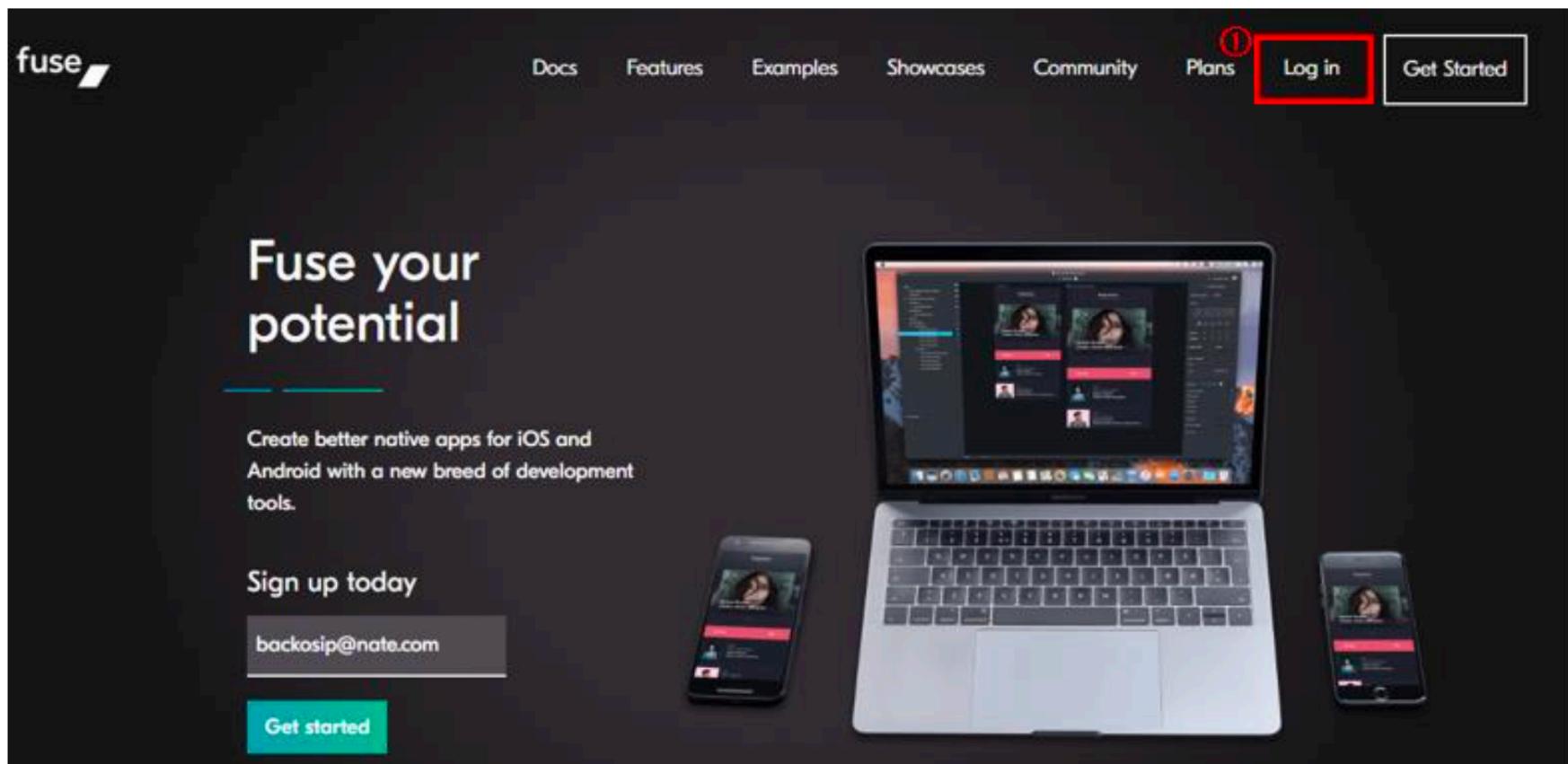
Fuse 설치하기

- 설치 과정
 - Fuse 다운로드
 - Fuse 설치
 - Fuse 실행 및 로그인
 - Fuse 추가 패키지, 텍스트 편집기 설치
 - 옵션1: Android 개발 환경 설치 (64비트 환경)
 - 옵션2: iOS 개발 환경 설치 (macOS 전용)

Fuse 설치하기

1. Fuse 다운로드

- www.fusetools.com 사이트 접속 후 “Log in” 클릭



Fuse your potential

Fuse 설치하기

1. Fuse 다운로드

- 등록 계정 있는 경우: 1에 정보 입력
- 등록 계정 없는 경우: 2 ("here") 클릭하여 계정 생성
 - (이메일 인증 필요 – 다음 페이지 참고)

① E-mail address:

Your e-mail address

Password:

Your password

Remember me on this computer

Log in

[Forgot your password?](#)

② New user? Register [here.](#)

Fuse 설치하기

- [참고] Fuse 회원 가입하기
 1. 이메일 입력 후 “Sign up” 클릭

Docs Features Examples Showcases Community

Register new account

Get a free Fuse account. You need an account to download Fuse Free, to start a trial or to buy Fuse Professional. We never share your email with anyone.

①

e.com

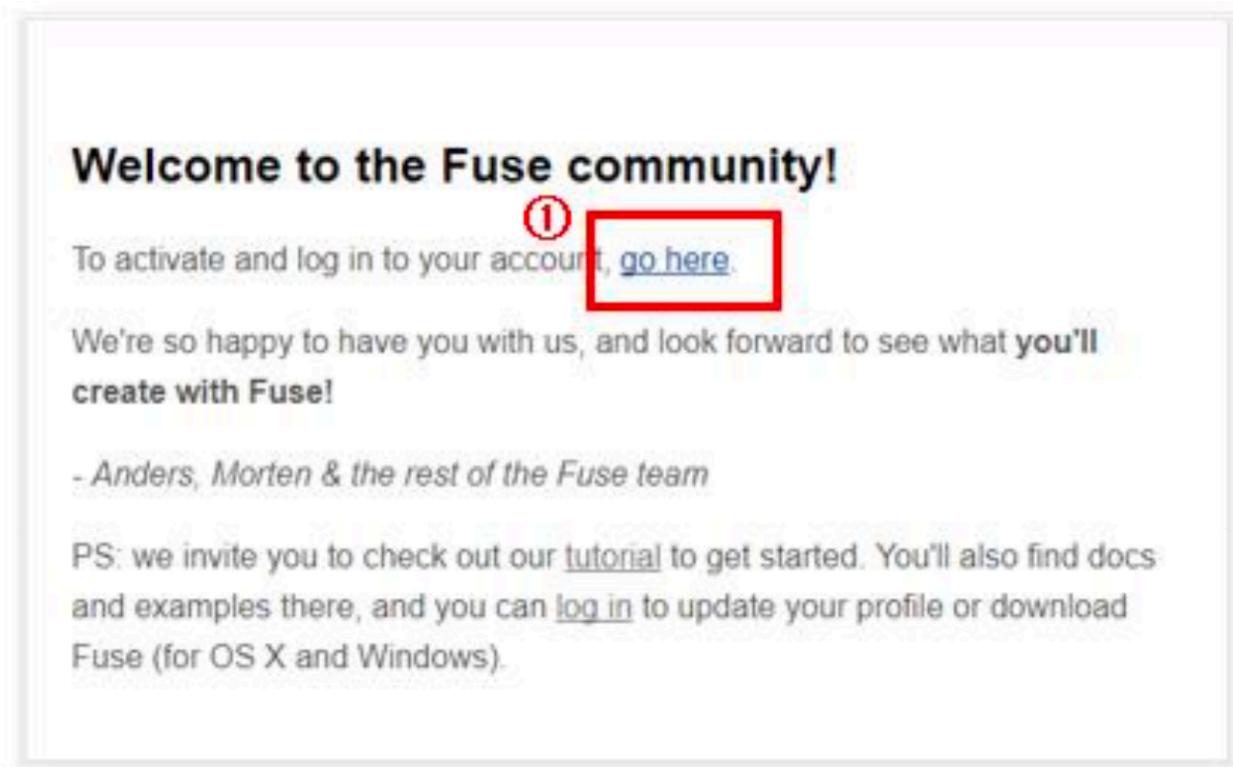
Already have an account? [Log in](#).

②

[Sign up](#)

Fuse 설치하기

- [참고] Fuse 회원 가입하기
 2. 인증 이메일 확인 후, “go here” 클릭하여 계정 인증 페이지로 이동



Fuse 설치하기

- [참고] Fuse 회원 가입하기
 - 이름, 패스워드, 패스워드 확인, Fuse 약관 동의 선택한 후 “Confirm and submit”을 클릭하면 가입 완료

Activate account

Complete the form below to finish creating your account and download Fuse.

Step 1: Verify e-mail



Email address:

A text input field with a blue placeholder bar on the left side.

Step 2: Complete account details

①

Name:

Select a password:

Confirm your new password:

I accept the [license agreement](#) for Fuse

②

Confirm and submit

Fuse your potential

Fuse 설치하기

- [참고] Fuse 회원 가입하기
 1. 이메일 입력 후 “Sign up” 클릭

Docs Features Examples Showcases Community

Register new account

Get a free Fuse account. You need an account to download Fuse Free, to start a trial or to buy Fuse Professional. We never share your email with anyone.

①

e.com

Already have an account? [Log in](#).

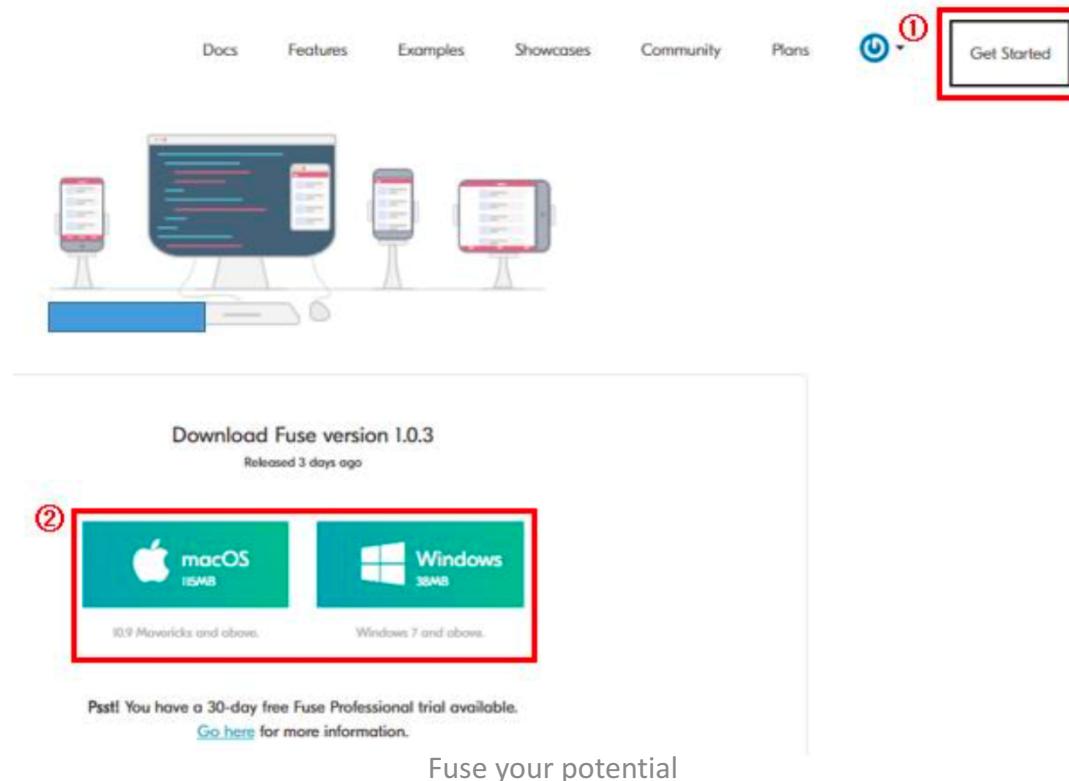
②

[Sign up](#)

Fuse 설치하기

1. Fuse 다운로드

- “Get Started” 클릭하여 다운로드 페이지 이동 후 운영체제에 맞는 Fuse 설치파일 다운로드
 - 참고: 최신 버전 (2017년 8월 16일 기준): 1.2.0



Fuse 설치하기

2. Fuse 설치

- 다운로드 한 설치 파일 실행

Windows 환경



[fuse_win_1_1_0_13808.exe](#)

Type: Application

macOS 환경

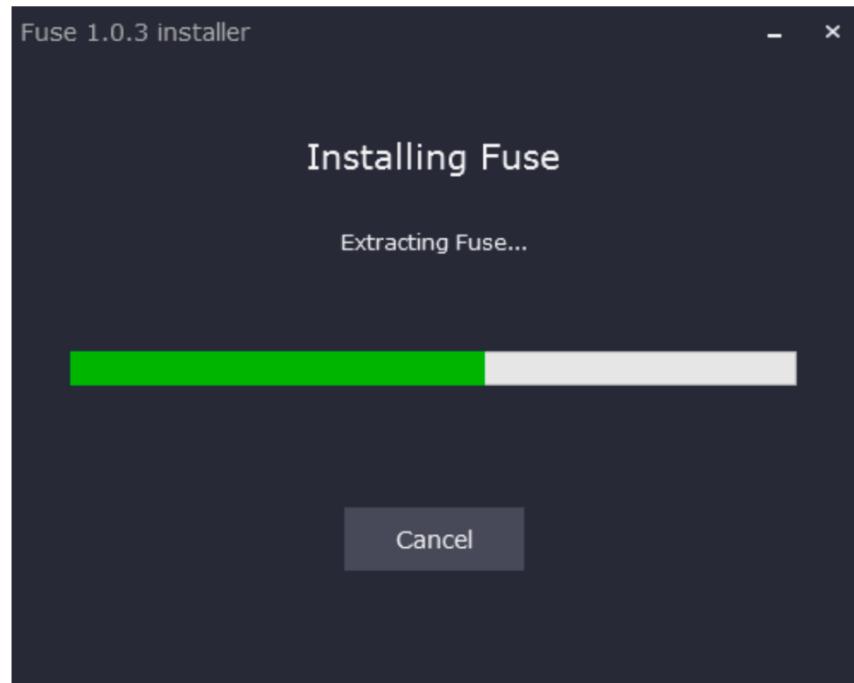
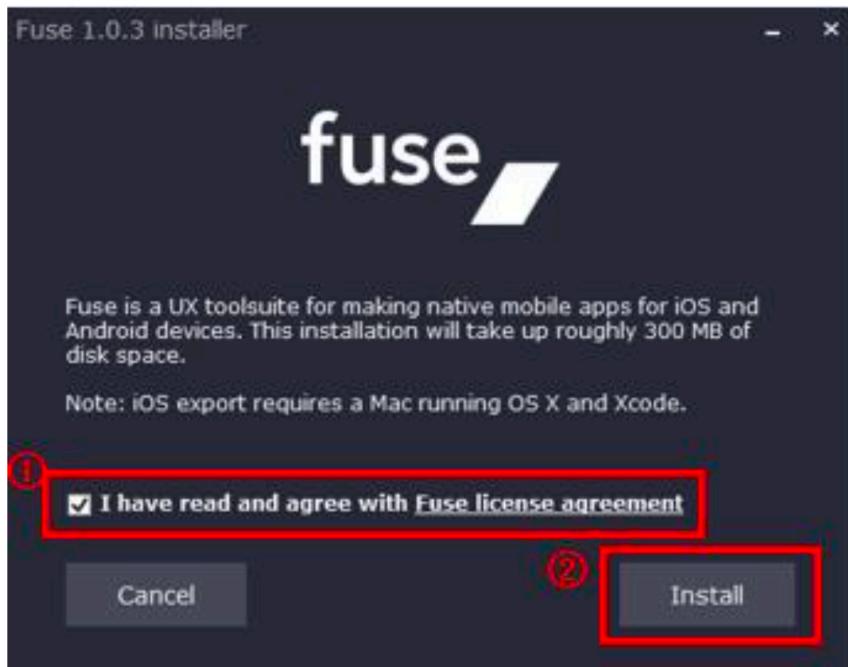


[fuse_osx_1_1_0_13808.pkg](#)

Fuse 설치하기

2. Fuse 설치

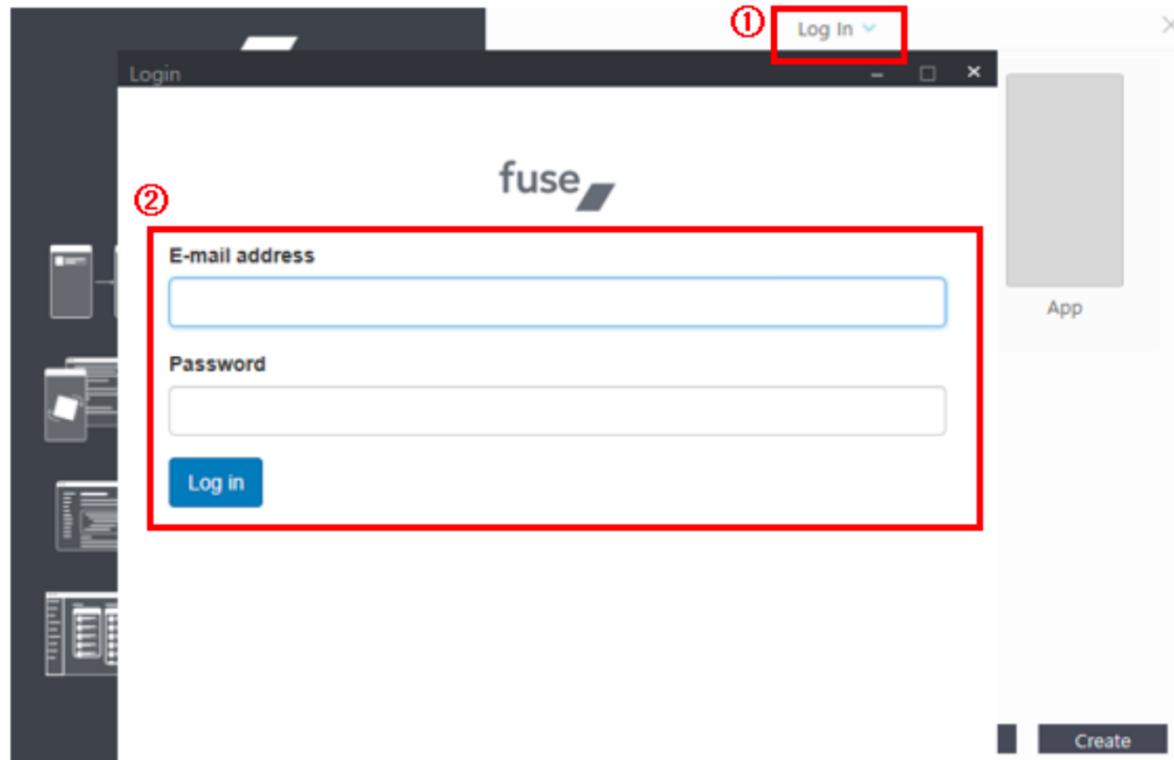
- 라이선스 동의 선택 후 “Install” 버튼 클릭



Fuse 설치하기

3. Fuse 실행 및 로그인

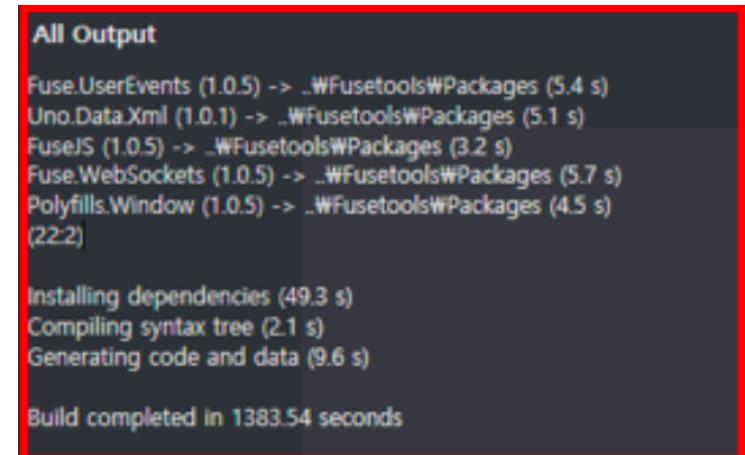
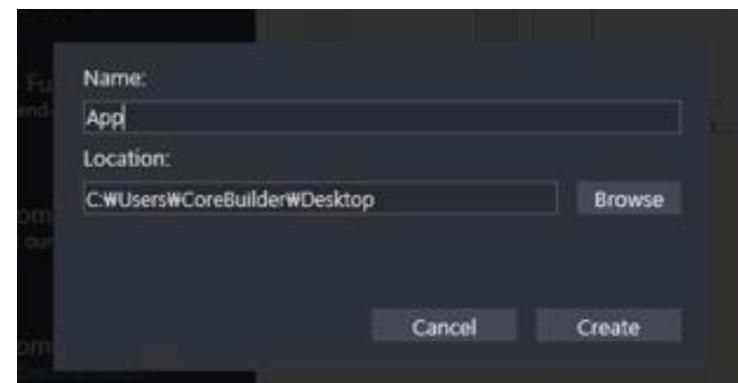
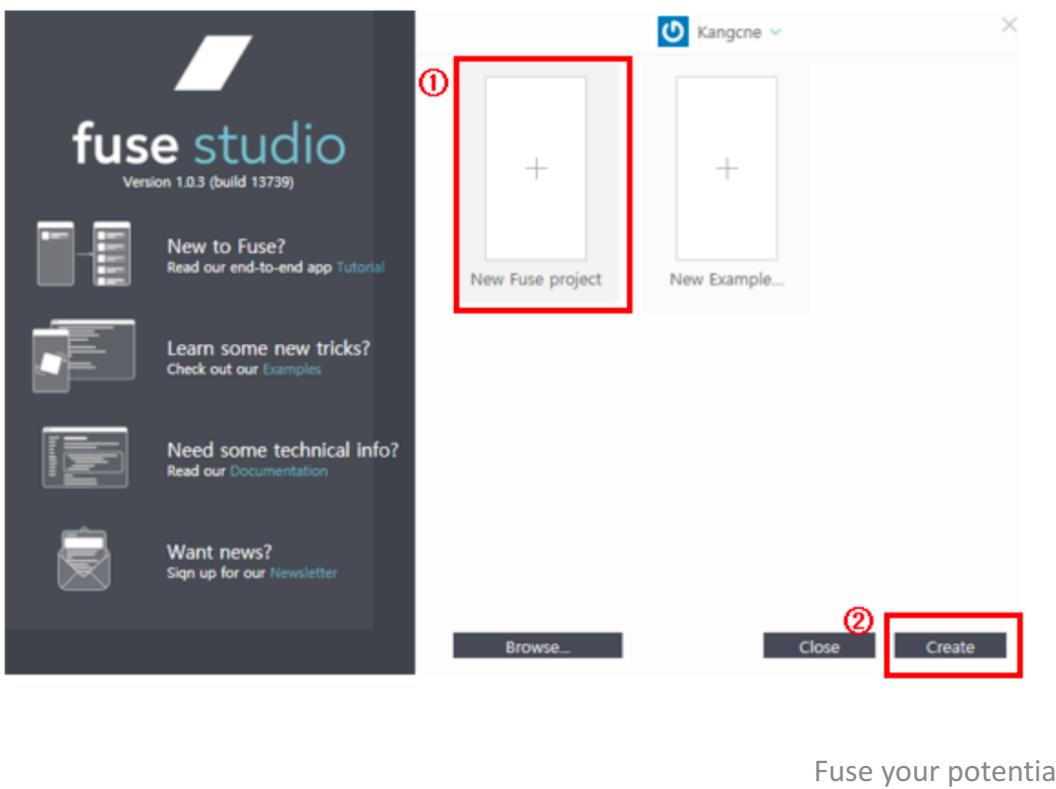
- (설치 완료 후 Fuse Studio 자동으로 실행)
- "Log In" 클릭 후 등록한 이메일 & 패스워드 입력



Fuse 설치하기

4. Fuse 추가 패키지, 텍스트 편집기 설치

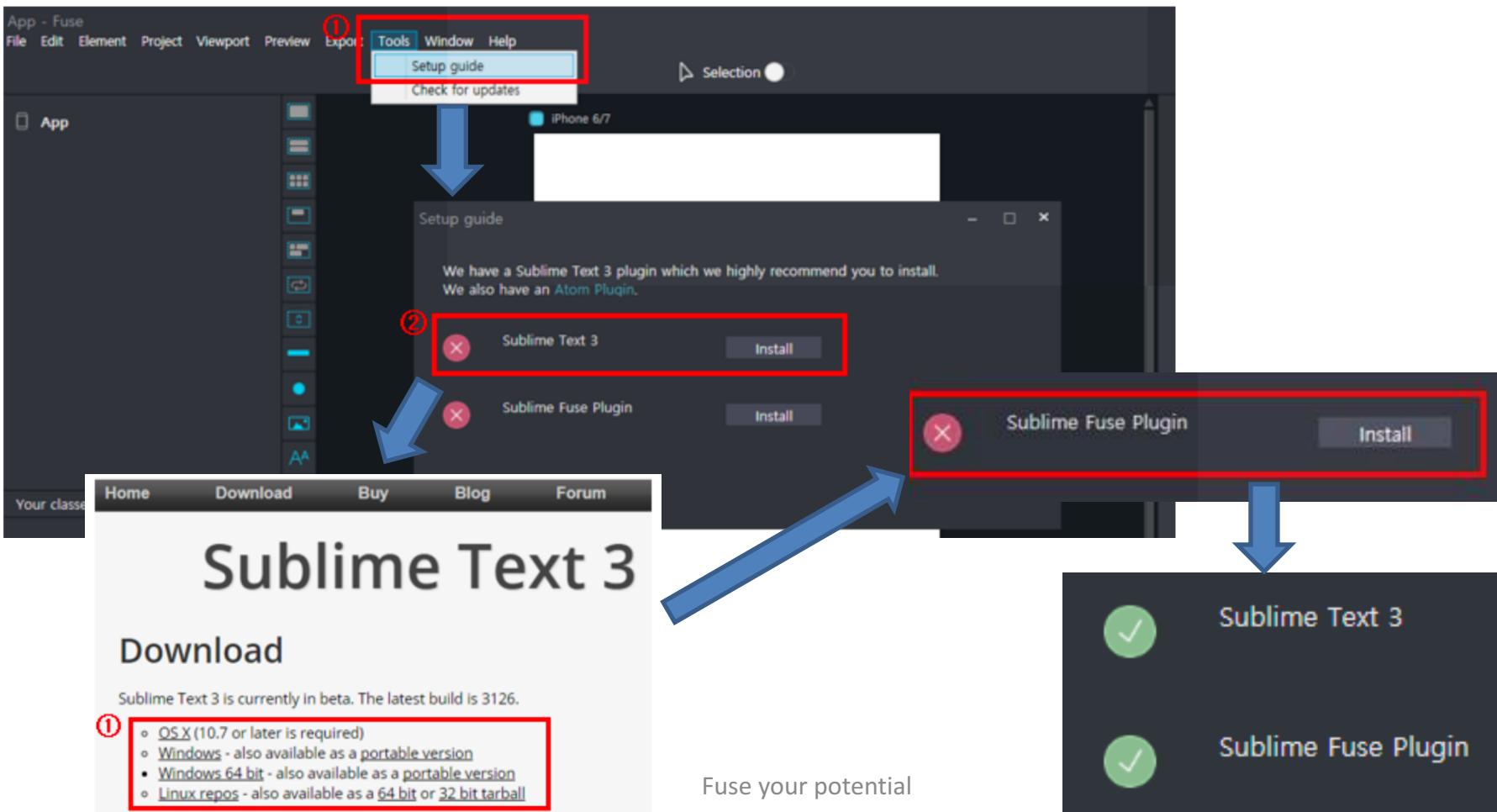
- 추가 패키지 설치: 새 프로젝트 생성하면 자동 설치
 - PC 사양 및 인터넷 속도에 따라 오래 걸릴 수 있음
 - “Build completed” 메시지 확인



Fuse 설치하기

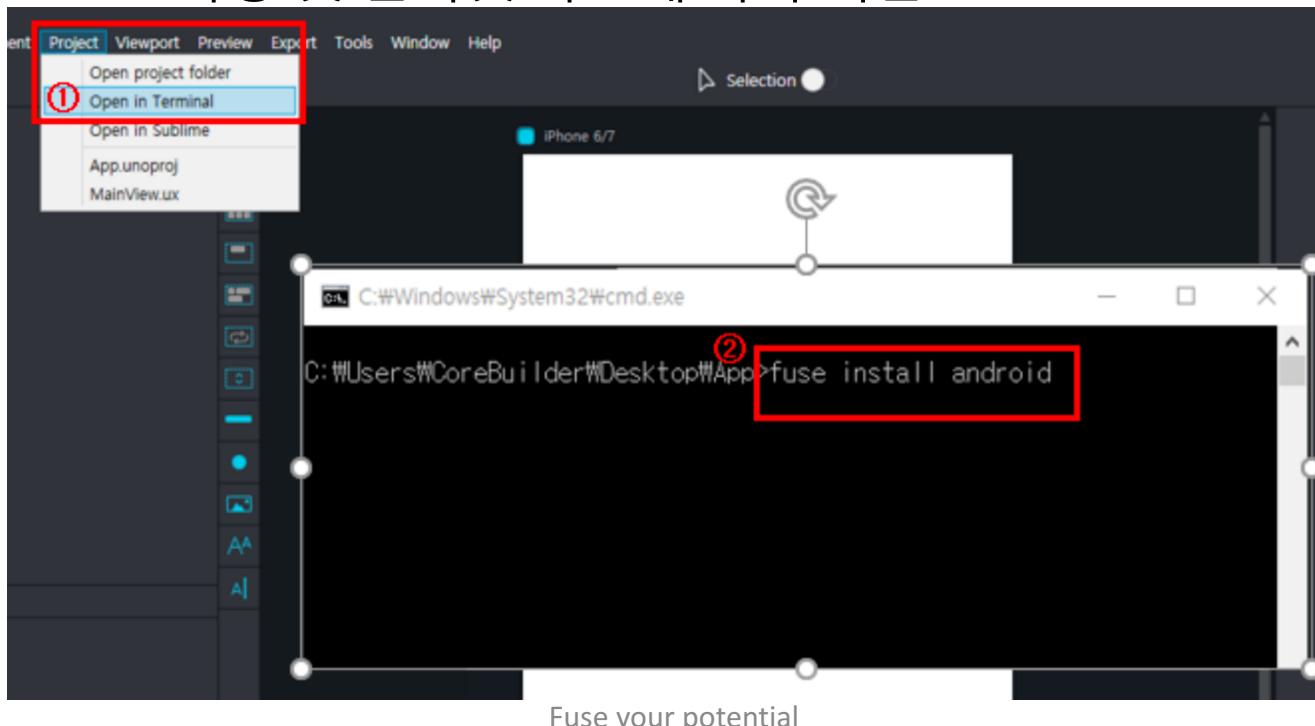
4. Fuse 추가 패키지, 텍스트 편집기 설치

- 텍스트 편집기 설치: Setup Guide에 따른 설치 진행



5. 옵션1: Android 개발 환경 설치

- “Fuse Studio > Project > Open in Terminal” 클릭
 - (Windows: 명령 프롬프트 실행, macOS: 터미널 실행)
- “fuse install android” 를 입력 후 Enter키 입력
 - ❖ PC 사양 및 인터넷 속도에 따라 시간 소요



Fuse 설치하기

5. 옵션1: Android 개발 환경 설치

- Android 개발 환경 설치 중 묻는 옵션: Enter/"y" 입력

```
fuse install android
# Starting android installer
Java Development Kit wasn't found or an incompatible version of Java Development
Kit found.
If you have an existing version of Java Development Kit, please specify its path
here. If not, just press Enter to continue: Enter 입력

Ant wasn't found or an incompatible version of Ant found.
If you have an existing version of Ant, please specify its path here. If not, jus
t press Enter to continue: Enter 입력

Android SDK wasn't found or an incompatible version of Android SDK found.
If you have an existing version of Android SDK, please specify its path here. If
not, just press Enter to continue: Enter 입력

Last updated 02 April 2013
Do you accept the license? (y/n): y 입력 후 Enter
```

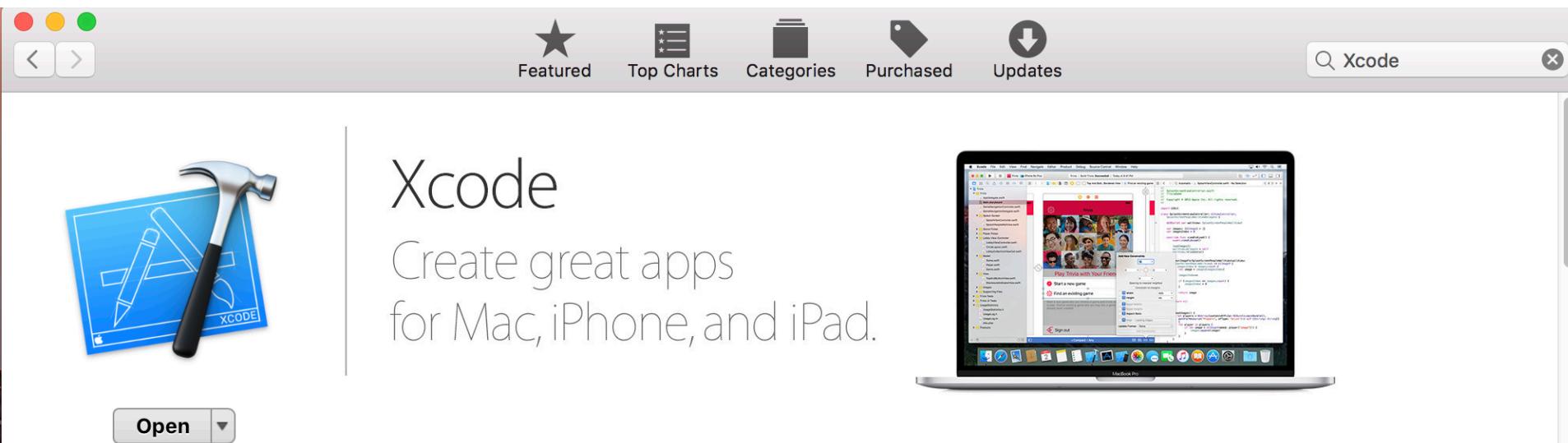
December 9, 2016
Do you accept the license? (y/n):

y 입력 후 Enter

Fuse 설치하기

6. 옵션2: iOS 개발 환경 설치 (macOS 전용)

- “App Store”에서 “Xcode”를 검색하여 설치 진행



Xcode 4+

Essentials

Xcode includes everything developers need to create great applications for Mac, iPhone, iPad, Apple TV, and Apple Watch. Xcode provides developers a unified workflow for user interface design, coding, testing, and debugging. The Xcode IDE combined with the Swift programming language make developing apps easier and more fun than ever before.

...

...More

[Apple Web Site](#)

[Xcode Support](#)

[App License Agreement](#)

Fuse your potential

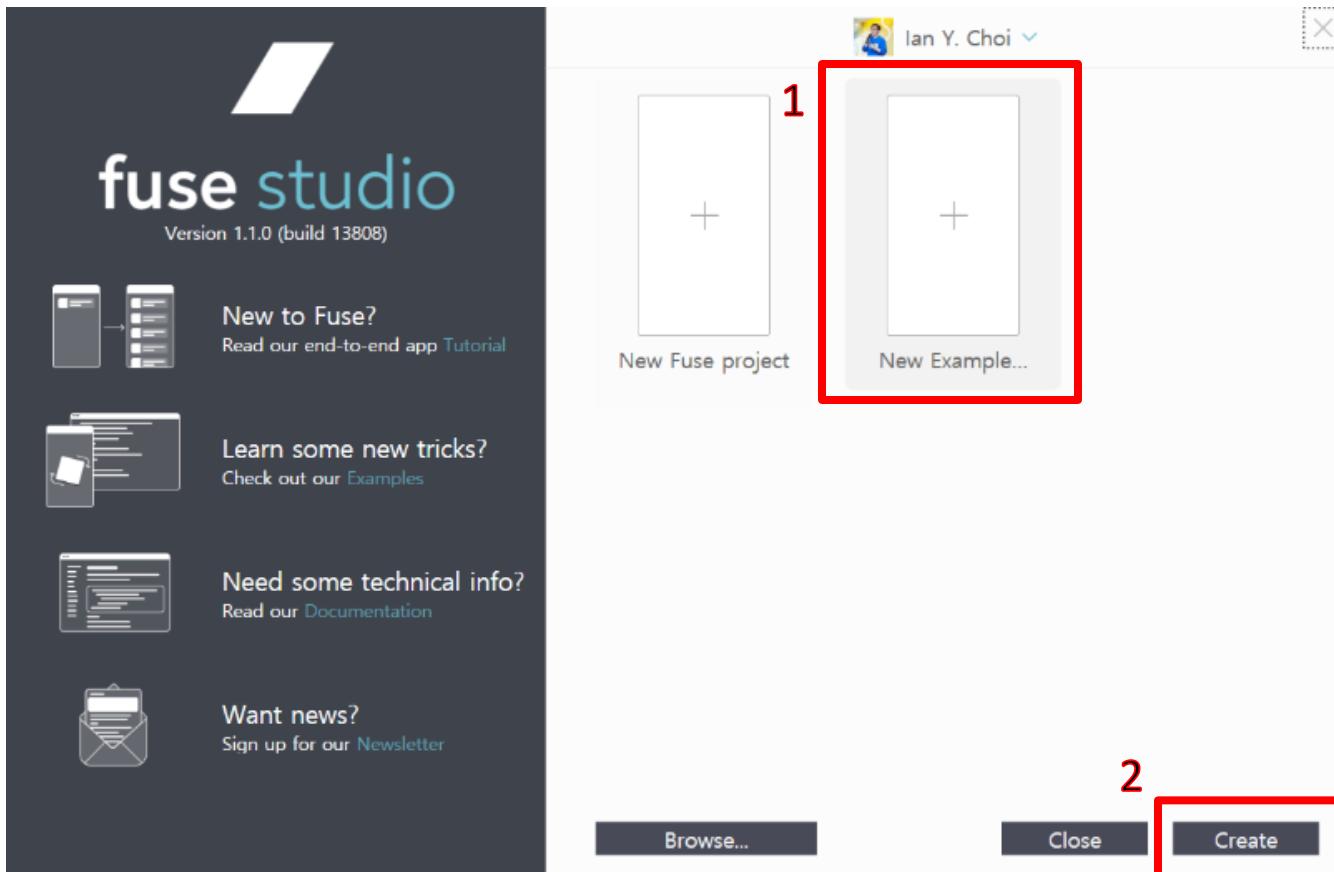
Fuse Studio

사용하기



Fuse Studio 사용하기

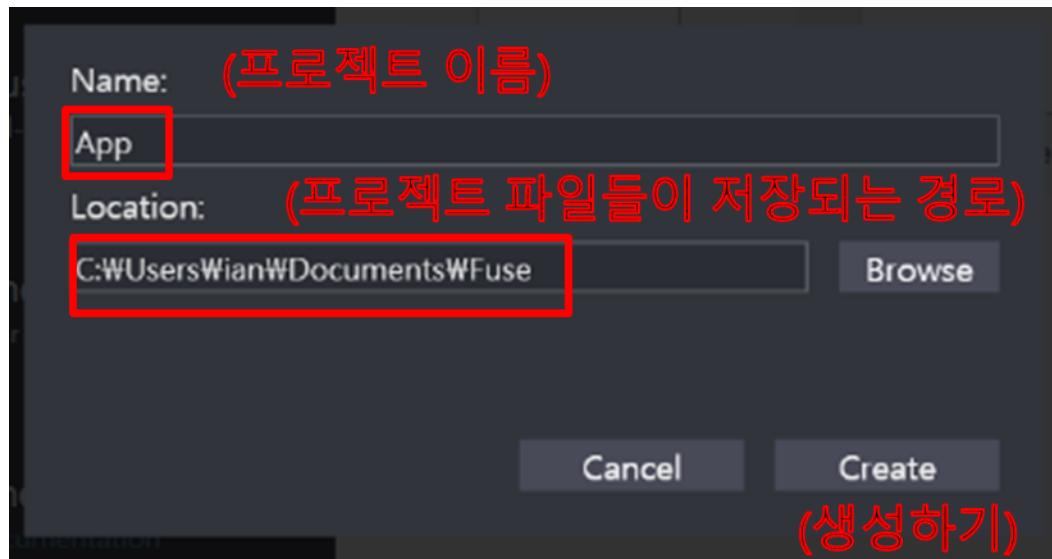
- 기본 예제 프로젝트를 클릭해봅시다



Fuse your potential

Fuse Studio 사용하기

- 프로젝트 이름 및 생성 경로 확인

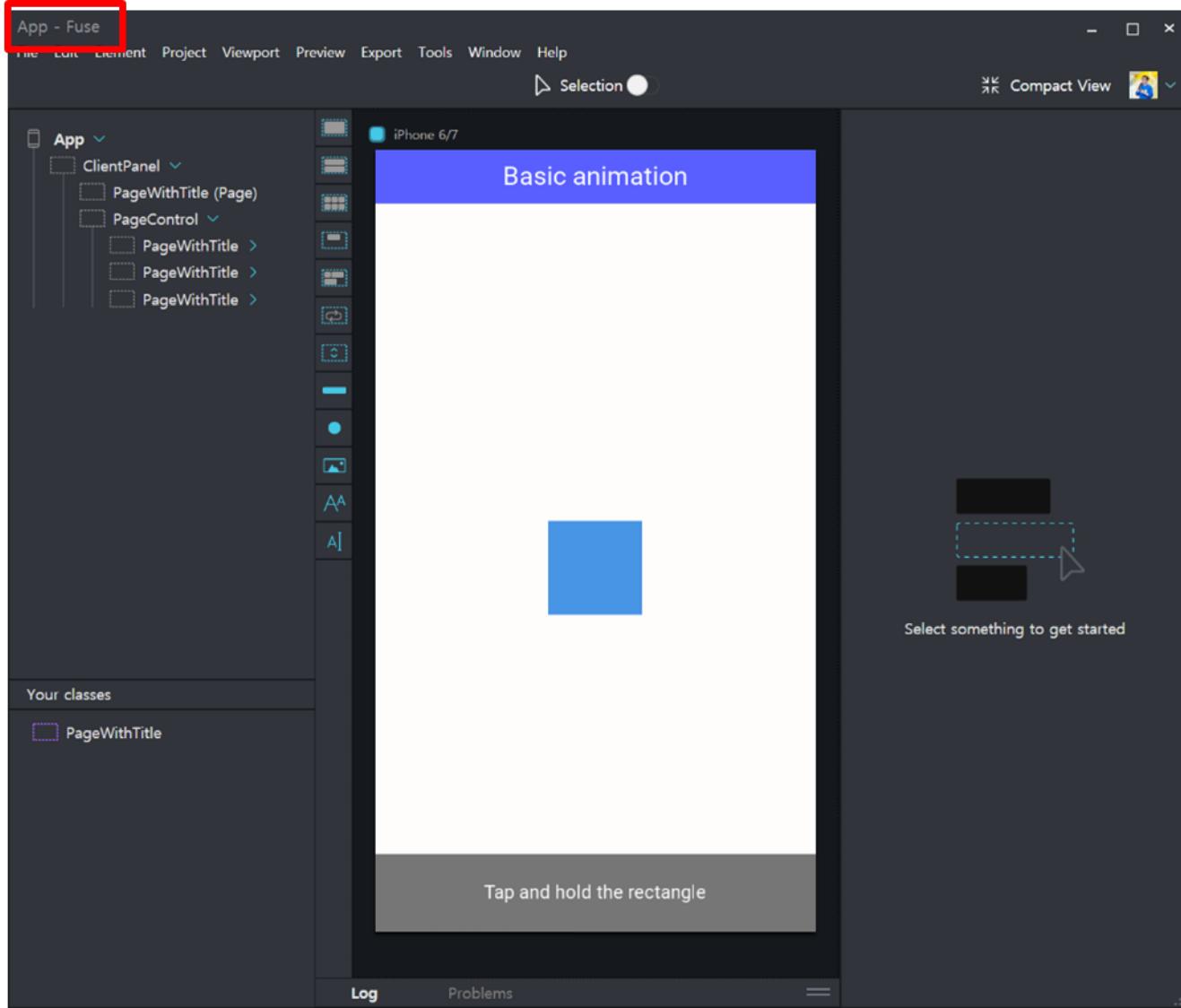


(폴더 위치
변경하고 싶을 때
클릭)

Fuse Studio 사용하기

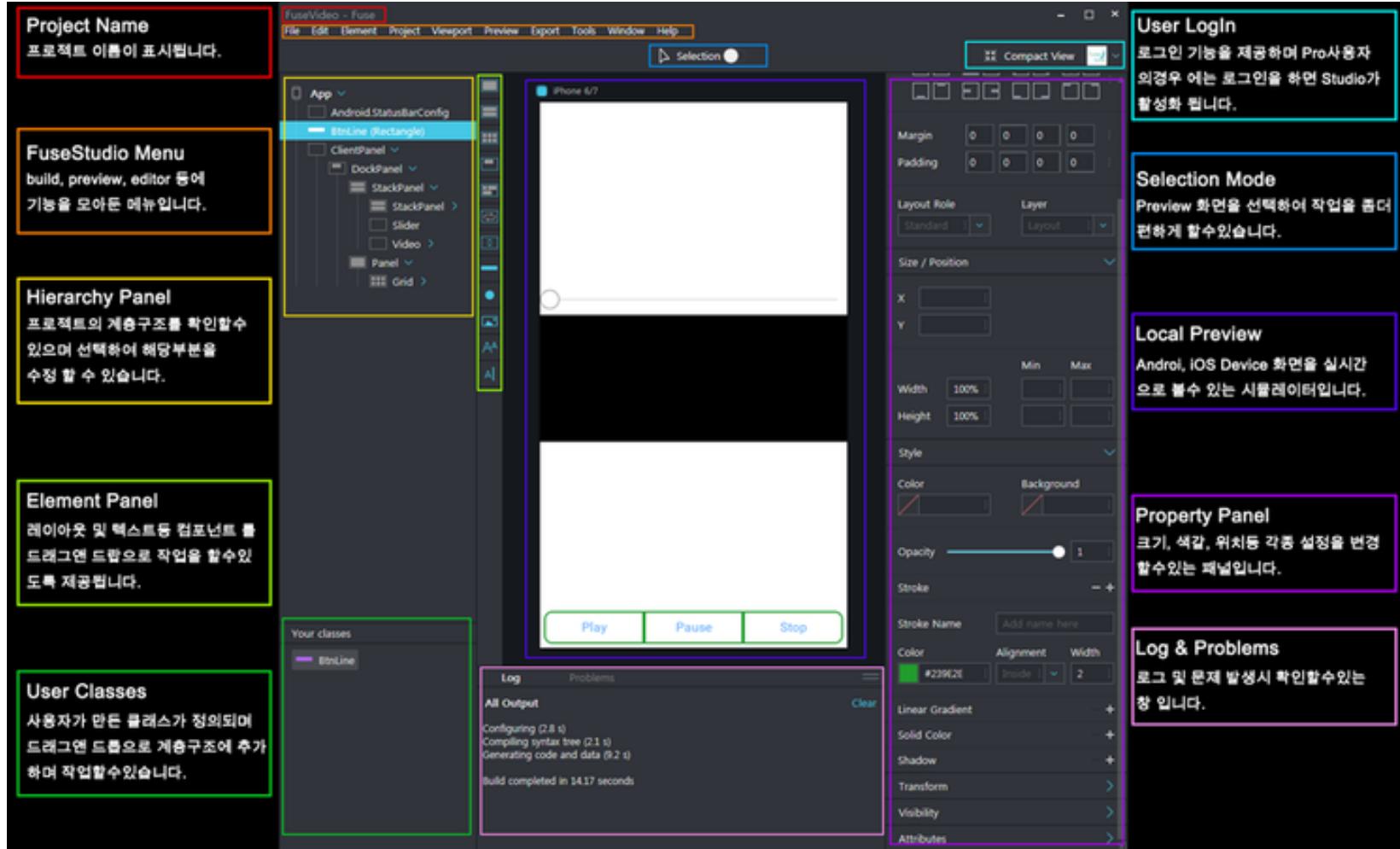
- 프로젝트 이름 및 생성 경로 확인

(프로젝트
이름과 동일)



Fuse Studio 사용하기

• Fuse Studio 구성요소

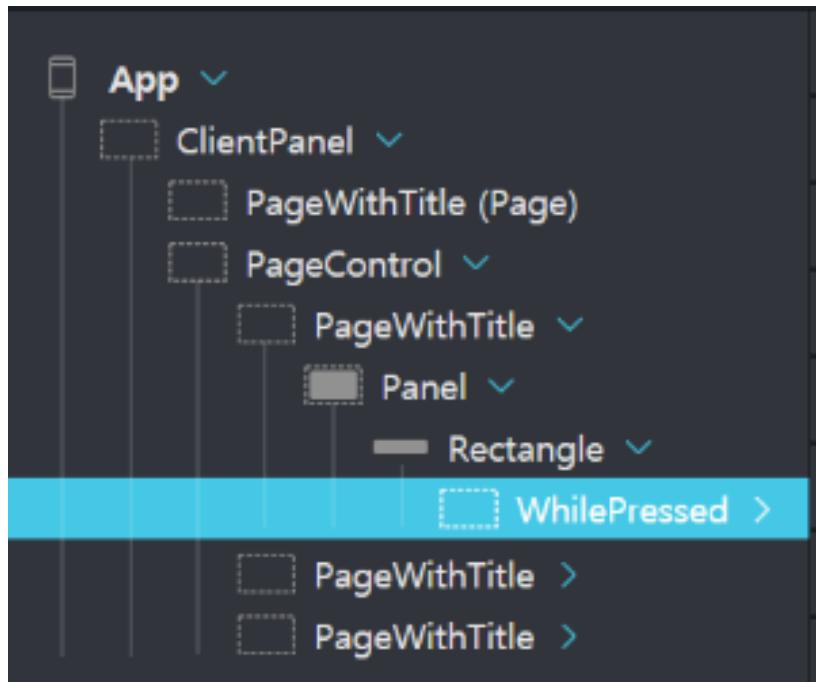


Fuse your potential

Fuse Studio 사용하기

1. 계층 구조 뷰

- App 구조를 보여줌
- 요소 클릭시 오른쪽 속성 편집기가 나타남



- App은 ClientPanel을 가짐
- PageControl에 3개의 페이지가 있음
- 첫 번째 페이지에 있는 Panel 안에서 사각형 (Rectangle)이 있음
- 사각형이 눌린 상태일 때 무엇인가를 처리하도록 되어 있음

Fuse Studio 사용하기

2. 사용자 정의 클래스 뷰

- UI/UX 마크업 요소를 재사용하기 위한 목적
- 속성을 지정하여 사용 가능

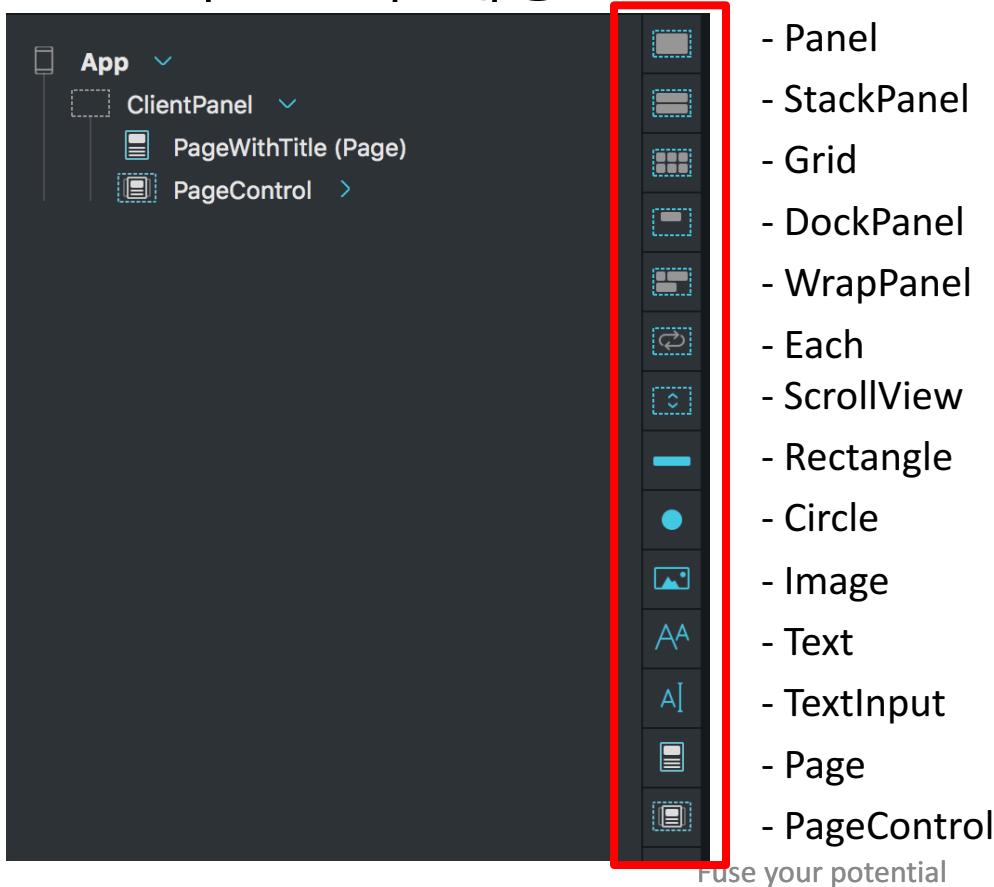
The screenshot shows the Fuse Studio interface with two main panes. On the left is the 'Object Explorer' pane, which displays the class hierarchy:

- App
- ClientPanel
 - PageWithTitle (Page) **(Selected)**
 - PageControl
 - PageWithTitle
 - Panel
 - Rectangle
 - WhilePressed
 - Rotate
 - Scale
 - Change
 - Change
 - PageWithTitle
 - PageWithTitle

Fuse Studio 사용하기

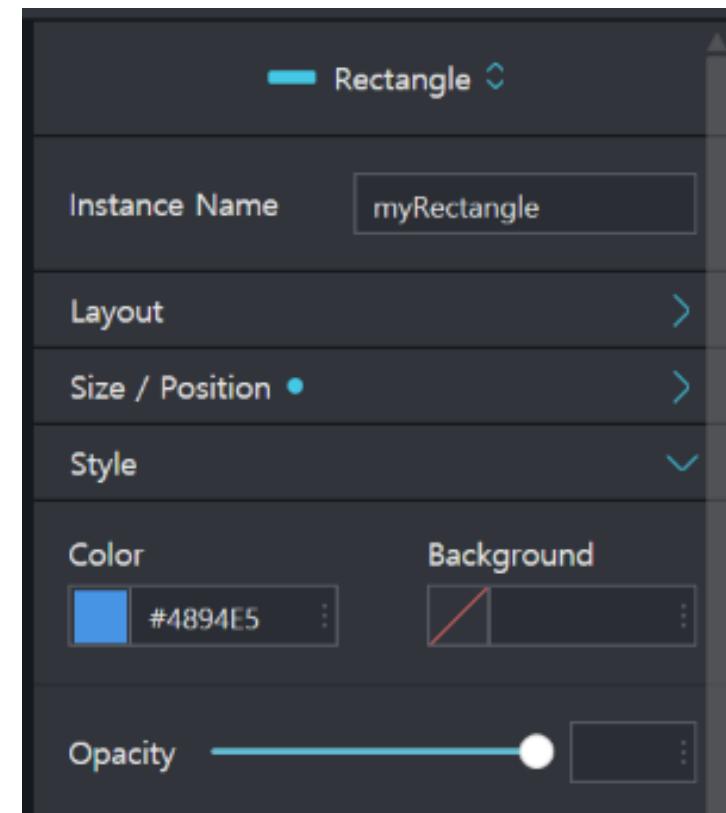
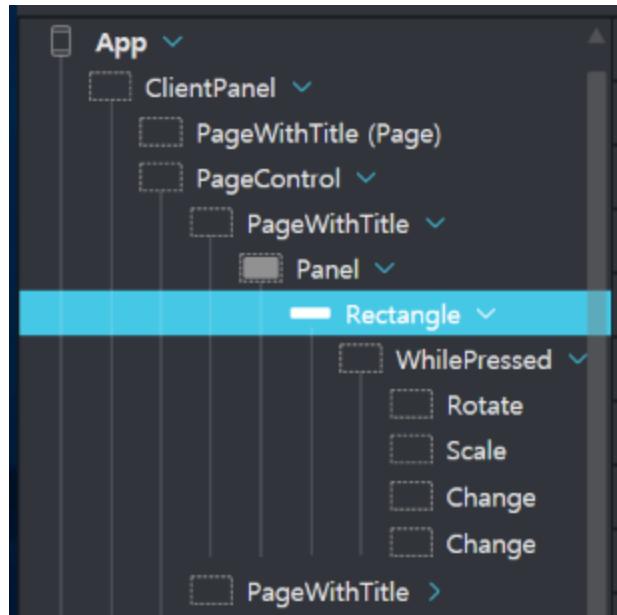
3. 도구 상자

- 드래그 앤 드롭으로 계층 구조 뷰에 추가
- 기본 도구 제공



4. 속성 편집기

- 계층 구조 뷰를 선택함에 따라 달라짐
- 디자인 요소를 직접 변경하여 결과 바로 확인 가능

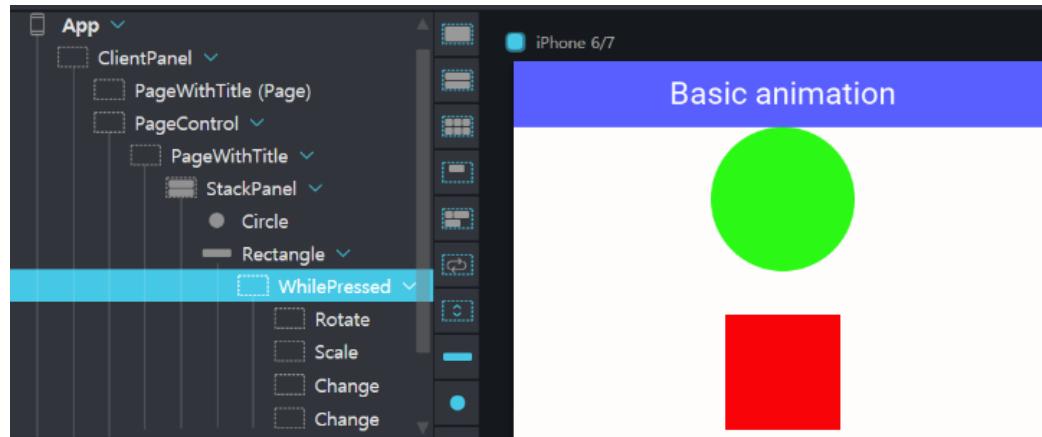




Fuse Studio 사용하기

[실습]

1. 계층 구조 뷰에서 첫 번째 페이지 사각형 요소를 선택하기
2. 사각형 색상을 파랑→빨강으로 변경하기
3. 첫 페이지 Panel을 StackPanel로 변경하기
4. 100x100 크기 녹색 원을 추가하기



Fuse Studio 사용하기

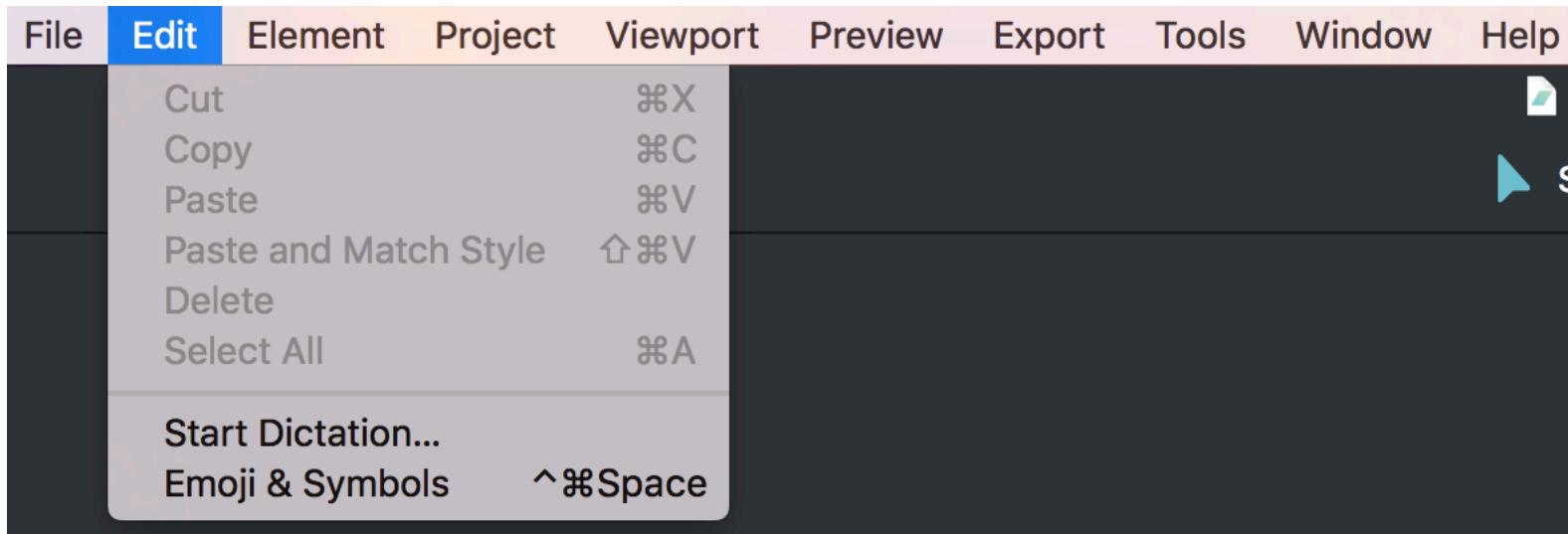
- 메뉴 살펴보기
 - File



- New : 새로운 Fuse 프로젝트 생성
- Open : 기존에 만들어진 Fuse 프로젝트 열기

Fuse Studio 사용하기

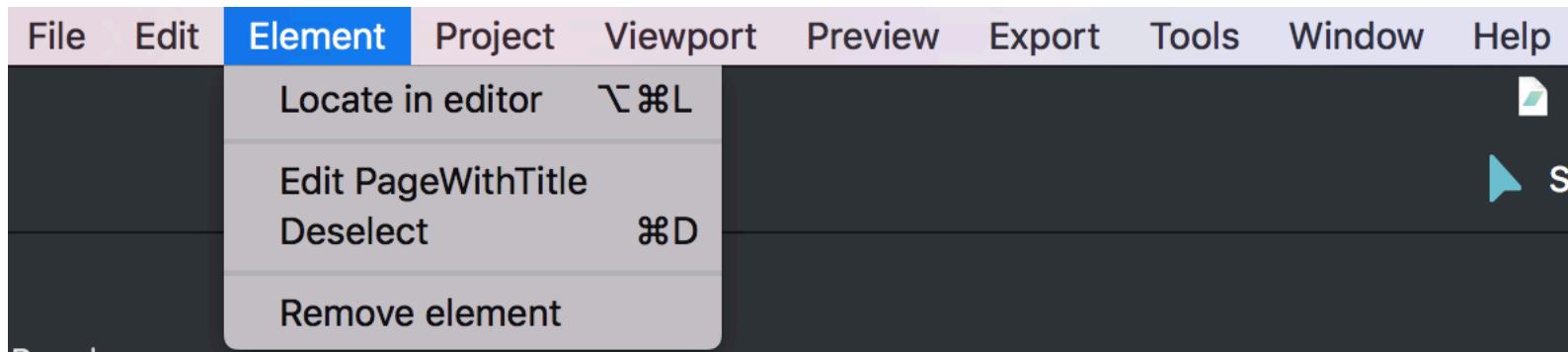
- 메뉴 살펴보기
 - Edit



- Cut : 잘라내기 / Copy : 복사
- Paste : 붙여넣기 / Paste and Match Style : 붙여놓고 스타일 일치
- Delete : 삭제
- Select All : 전체선택

Fuse Studio 사용하기

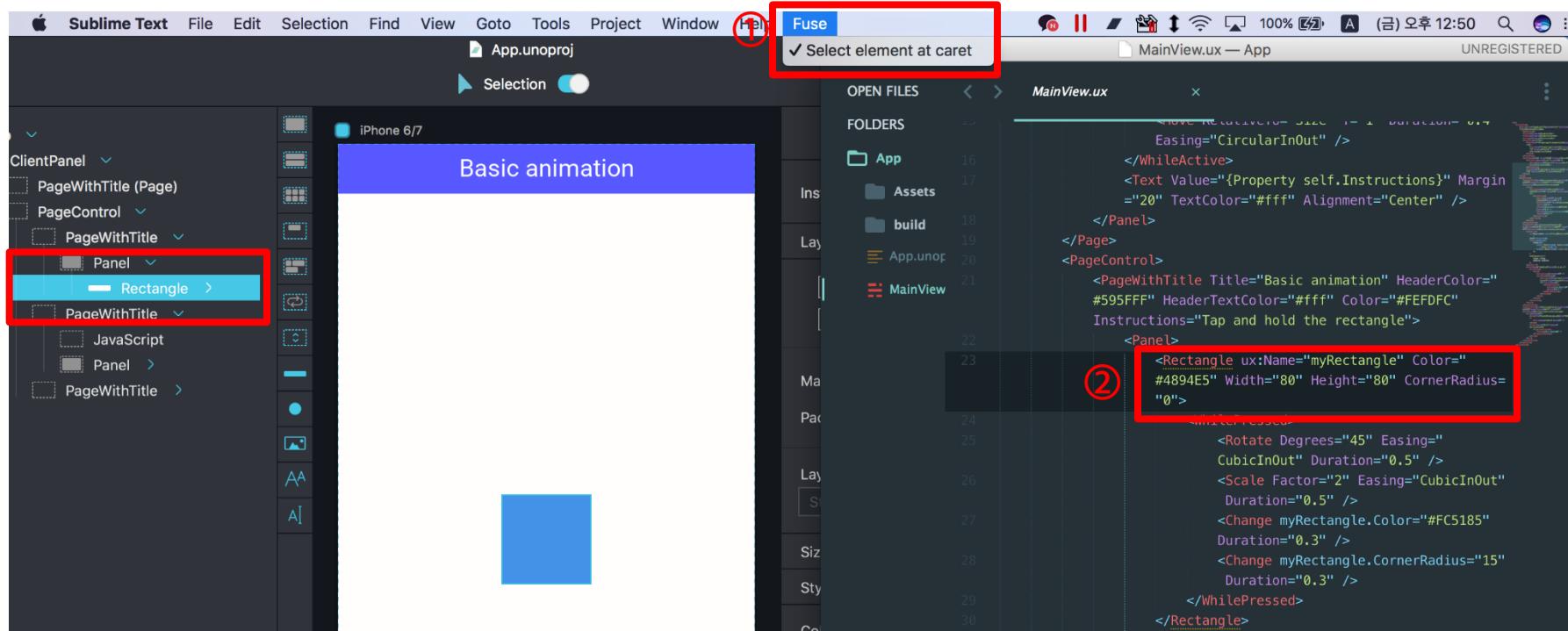
- 메뉴 살펴보기
 - Element



- Location in editor : 코드 편집기의 선택한 요소의 위치로 이동 (Sublime Text가 실행되어 있어야 함)
- Edit [Page Element] : 해당 [Page Element] 작업 모드로 변경
- Deselect : 선택 해제
- Remove element : 선택한 요소 삭제

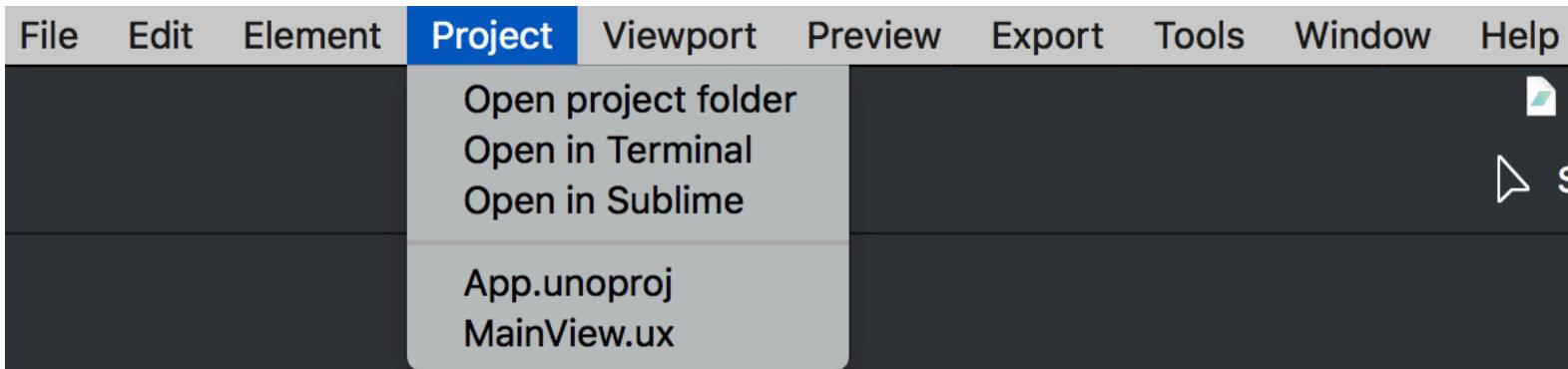
Fuse Studio 사용하기

- 메뉴 살펴보기 (Sublime Text)
 - Sublime Text → Fuse → Select element at caret
 - 선택 시 커서가 위치한 코드의 요소가 Fuse Studio에 자동 선택



Fuse Studio 사용하기

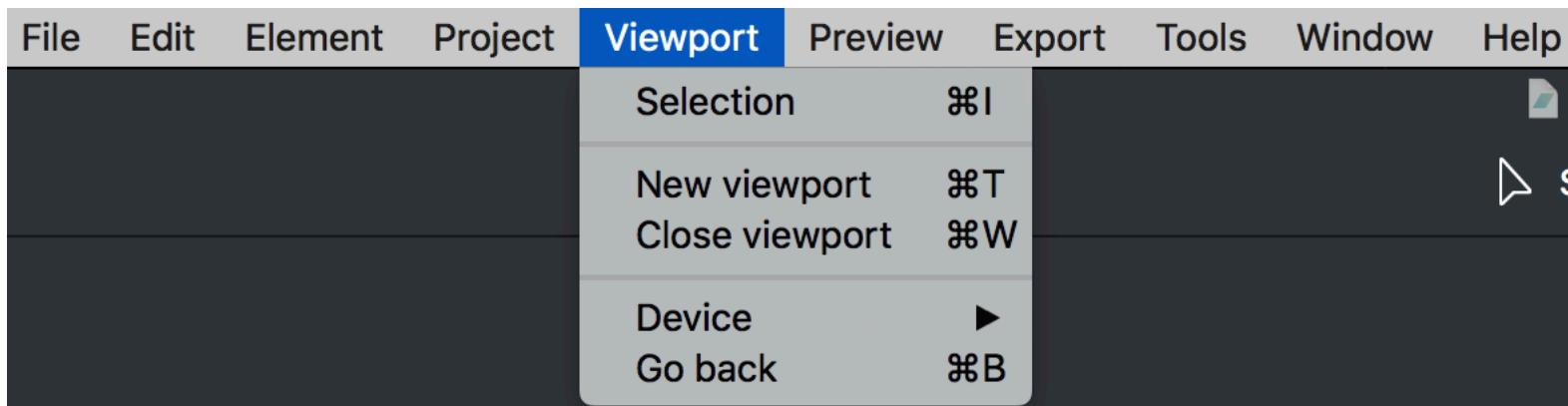
- 메뉴 살펴보기
 - Project



- Open project folder : 현재 프로젝트의 디렉토리 탐색
(Windows : Explorer / Mac : Finder)
- Open in Terminal : 현재 프로젝트 위치에서 Terminal 실행
- Open in Sublime : 현재 프로젝트를 Sublime Text 편집기에서 열기

Fuse Studio 사용하기

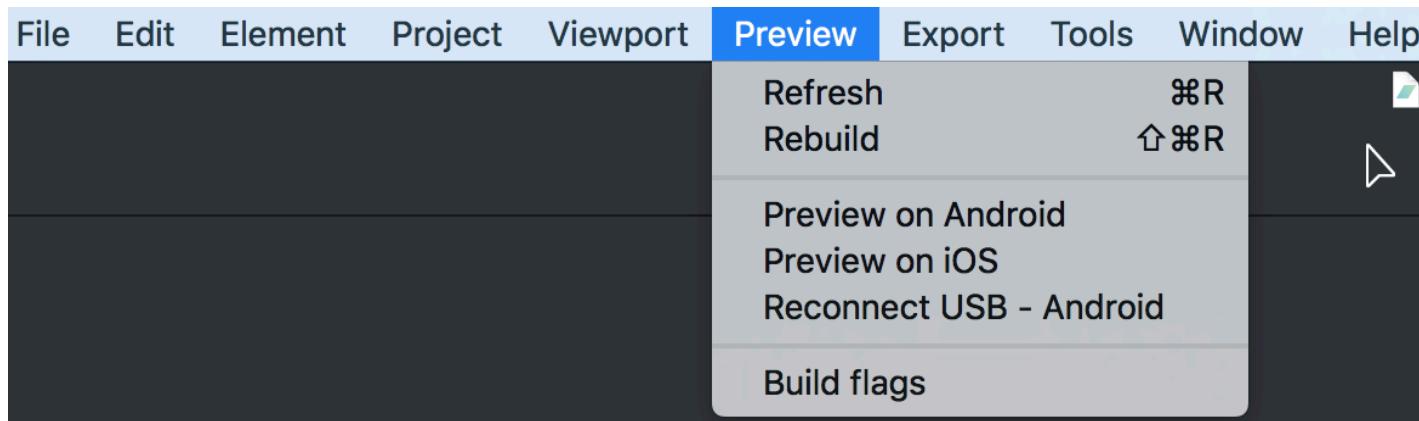
- 메뉴 살펴보기
 - Viewport



- Selection : Viewport Selection 기능 Toggle
- New viewport : 새로운 Viewport 생성
- Close viewport : 선택 된 Viewport 닫기
- Device : 가로세로 전환 및 디바이스 해상도 설정
- Go back : 하드웨어 뒤로가기 버튼 (Android)

Fuse Studio 사용하기

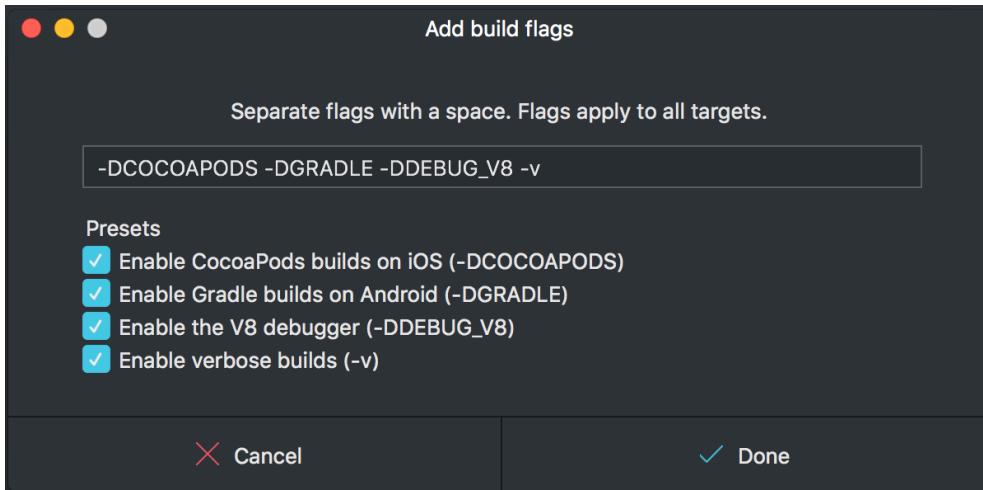
- 메뉴 살펴보기
 - Preview



- Refresh : Preview 화면 새로고침
- Rebuild : 프로젝트 다시 빌드
- Preview on Android : 안드로이드 기기로 Preview App 설치
- Preview on iOS : iOS 기기로 Preview App 설치 (Only Mac, Xcode)
- Reconnect USB – Android : 안드로이드 기기 재 연결
- Build flags : Build 옵션 설정 (다음장 참조)

Fuse Studio 사용하기

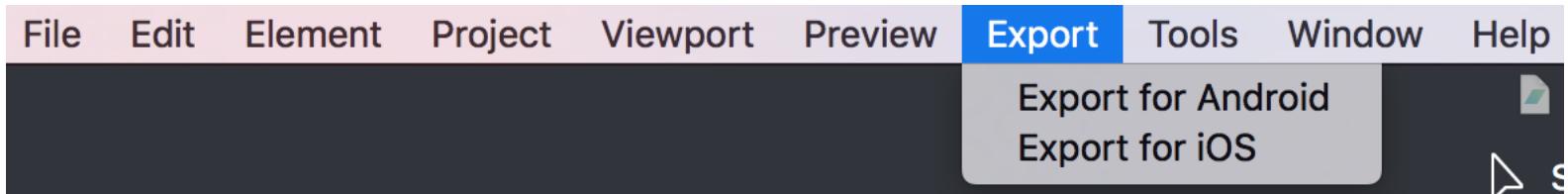
- 메뉴 살펴보기
 - Preview – Build flags



- -DCOCOAPODS : iOS 빌드 시 CocoaPods 사용 설정
- -DGRADLE : 안드로이드 빌드 시 Gradle 사용 설정
- -DDEBUG_V8 : Javascript 디버깅 사용 설정
- -v : 컴파일러에서 더 자세한 정보 출력

Fuse Studio 사용하기

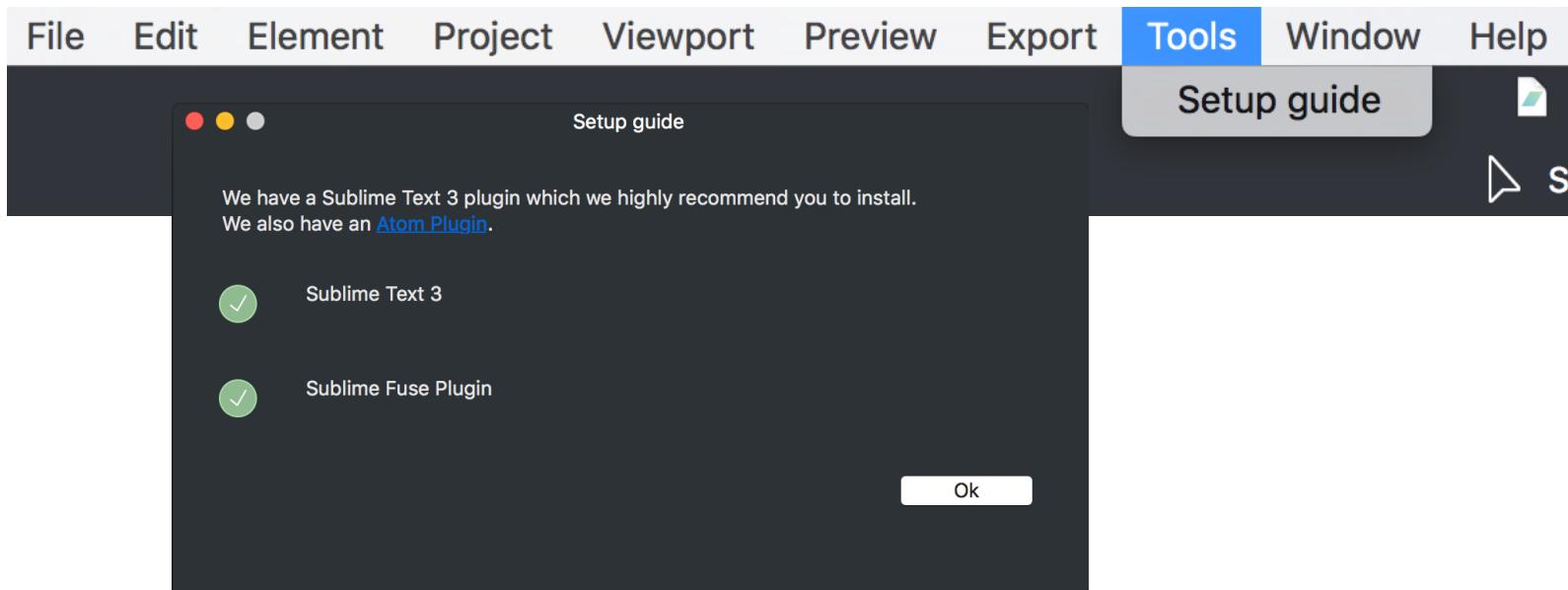
- 메뉴 살펴보기
 - Export



- Export for Android : 안드로이드 기기로 App 설치
- Export for iOS : iOS 기기로 App 설치

Fuse Studio 사용하기

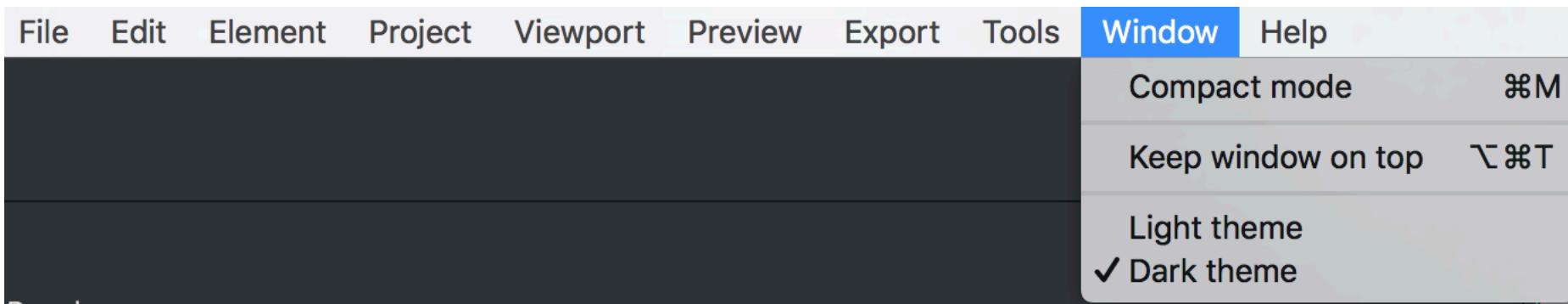
- 메뉴 살펴보기
 - Tools



- Setup Guide : Sublime Text 및 Sublime Fuse Plugin 설치 및 확인

Fuse Studio 사용하기

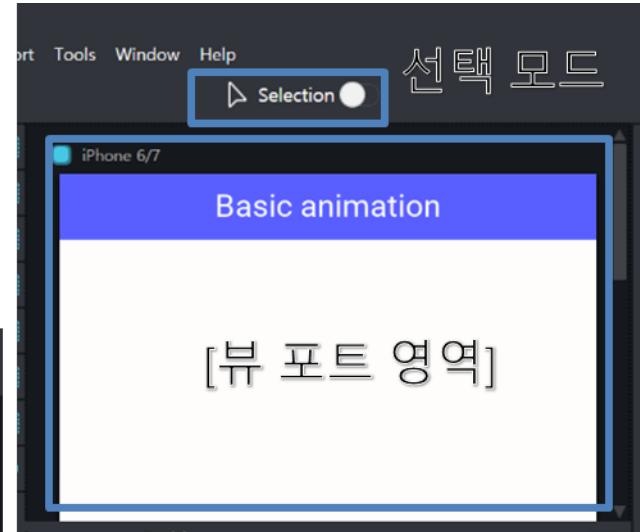
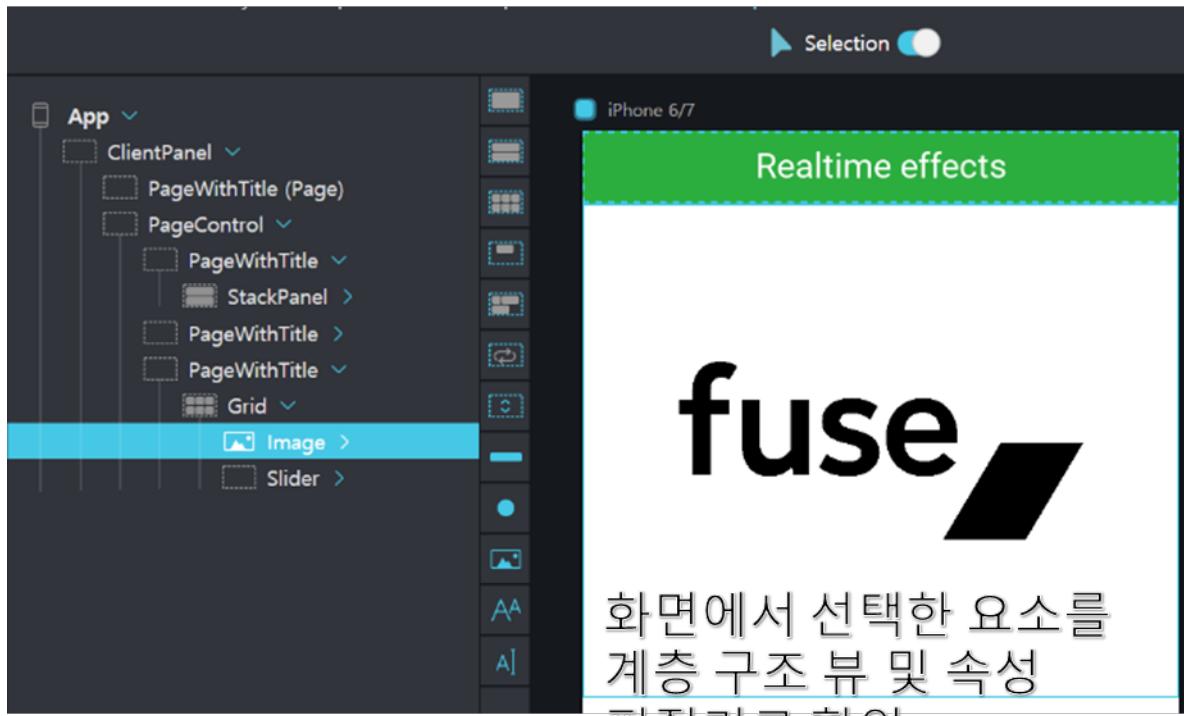
- 메뉴 살펴보기
 - Window



- Compact mode : Fuse Studio 창을 Compact mode로 변경
- Keep window on top : Fuse Studio를 항상 위에 유지
- Light theme : Fuse Studio Light theme 적용
- Dark theme : Fuse Studio Dark theme 적용

Fuse Studio 사용하기

- 선택 모드
 - 실행 중인 앱 요소를 클릭하여 속성 편집기에서 실시간 수정



Fuse Studio 사용하기

- Multi Viewport
 - 다양한 디바이스 해상도의 Viewport 및 멀티 디바이스지원



Fuse your potential



[Part 1]

UX markup 이해

UX markup에 대한 기본적인 이해

- UX markup
 - XML 기반의 마크업 언어 (Html 과 유사)
 - Fuse의 UI Layout, Animation, Gesture, Navigation, Data Binding, Event Response 등 앱의 시각적인 측면 구현
 - 모든 UX 파일 이름은 .ux 파일 확장자로 끝남
 - <App> 시작해서 </App> 종료
 - 모든 Fuse 프로젝트는 한쌍의 <App></App>태그만 존재
 - 처음 하나의 ux파일로 시작하여 원하는 만큼 ux파일 분리 가능

UX markup에 대한 기본적인 이해

- UX markup 작성 방법

```
<!-- 프로젝트 생성 시 <App> 태그만 존재 -->
<!-- Tag는 < 시작해서 > 끝남 -->
<App>
</App>
```

- 아래의 코드를 입력해 보세요.

```
<!-- 태그 시작-->
<App>
    <!-- 태그 시작 & 종료를 한 줄에서 함께 처리 -->
    <Text Value="Hello, Fuse world!" />
<!-- 태그 종료 -->
</App>
```

UX markup에 대한 기본적인 이해

- UX markup 작성 방법

```
<StackPanel Orientation="Vertical">
    <Text Value="Hello Fuse" />
    <Text Value="World!!" />
</StackPanel>
```

- UX markup 해석

```
<!-- (태그명: StackPanel) 태그 열기. 2개 자식 태그를 사용. -->
<!-- Orientation 속성을 가짐 -->
<StackPanel Orientation="Vertical">
    <!-- 자식 객체 / self closing tag / Value 속성을 가짐 -->
    <Text Value="Hello Fuse" />
    <!-- 자식 객체 / self closing tag / Value 속성을 가짐 -->
    <Text Value="World!!" />
<!-- 태그 종료 (태그명: StackPanel) -->
</StackPanel>
```

Layout 이해하기

Layout 이해하기

- Layout 이란?
 - 화면에서 요소의 크기를 조정하고 배치하는 행위
 - 다양한 해상도의 Layout를 작성 하려면 올바르게 구조화 하는 방법을 알아야 함
 - Fuse의 여러가지 Panel을 통해 객체를 그룹화하여 각기 다른 Layout을 구현
 - Panel은 사전 정의 된 Layout 규칙에 따라 객체의 크기와 위치를 부여하는 역할을 담당
 - Panel은 자식을 배치해야하는 특정 사용가능 공간이 있으며 이 크기는 자신의 하위에 크기를 할당하는 방식

Layout 이해하기

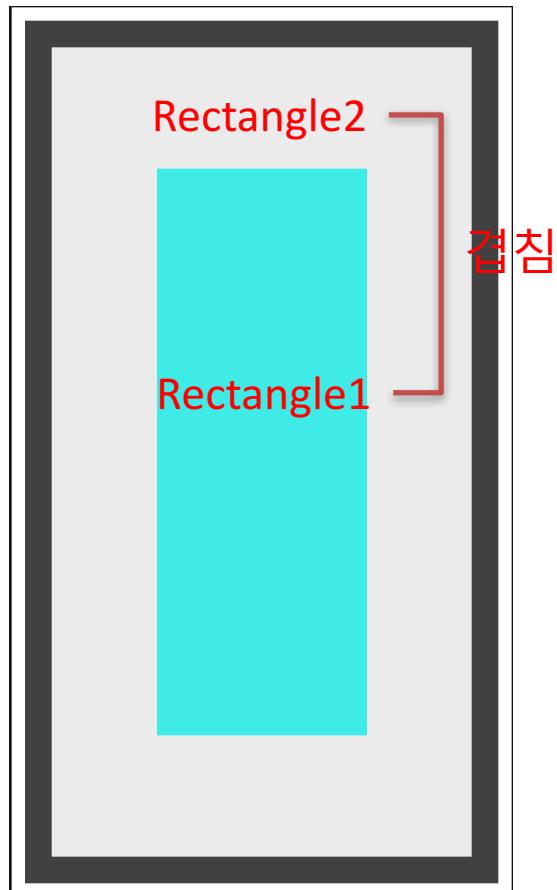
- Layout 유형
 - Panel
 - 기본 Layout으로 자식 요소는 사용 가능한 공간을 모두 차지함
 - Panel 안에 여러개의 자식 요소가 있다면 서로 겹쳐짐
 - Alignment, Margin, Padding 속성을 이용하여 자식 요소 배치

```
<Panel Margin="10" Padding="20" Color="#444">
    <Rectangle Color="#4bedea" Height="70%" Width="50%" />
    <Rectangle Color="#eee" />
</Panel>
```



Layout 이해하기

- Layout 유형
 - Panel - 결과



Layout 이해하기

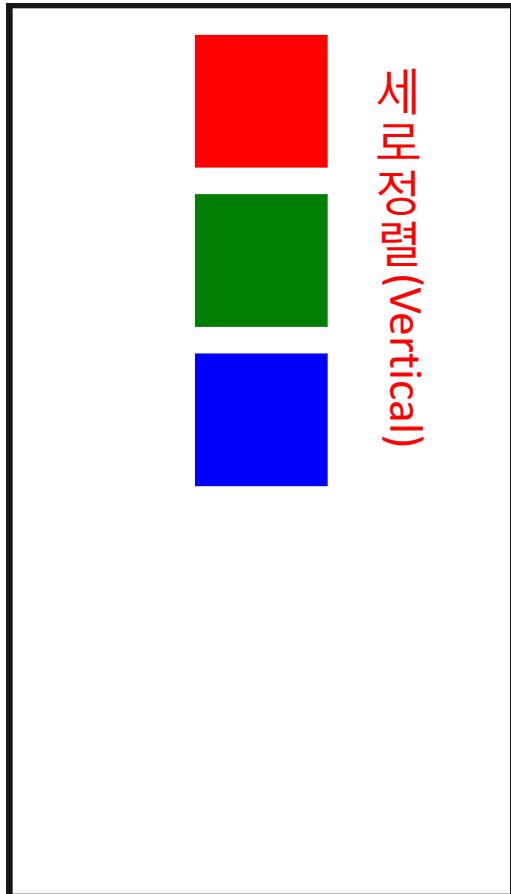
- Layout 유형
 - StackPanel
 - 자식요소를 세로(기본값) 또는 가로로 차곡차곡 정렬
 - Orientation 속성을 사용하여 가로(Horizontal), 세로(Vertical) 형태로 배치 설정
 - ItemSpacing : 자식 요소 사이에 간격 설정

```
<StackPanel ItemSpacing="20" Orientation="Vertical"  
Padding="20">  
    <Panel Width="100" Height="100" Background="Red" />  
    <Panel Width="100" Height="100" Background="Green" />  
    <Panel Width="100" Height="100" Background="Blue" />  
</StackPanel>
```



Layout 이해하기

- Layout 유형
 - StackPanel - 결과



Layout 이해하기

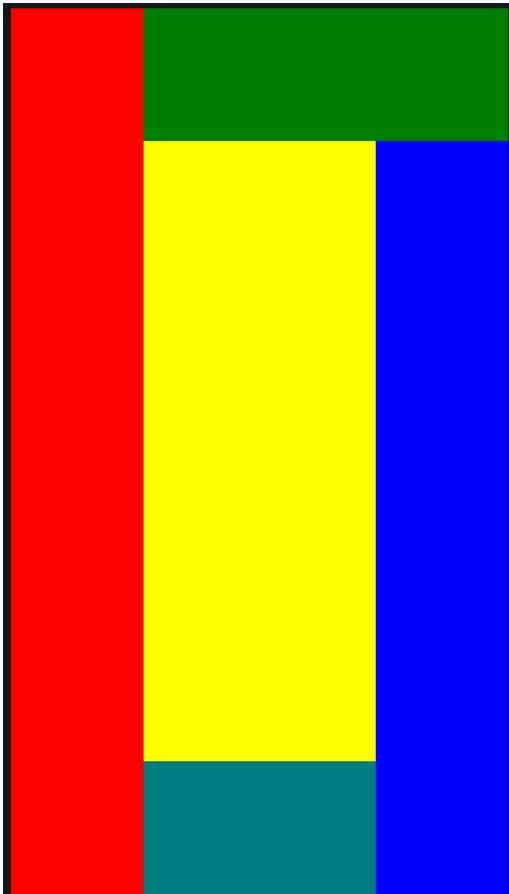
- Layout 유형
 - DockPanel
 - 자식요소를 상하좌우로 도킹하여 정렬
 - 자식요소에서 Dock 속성을 사용하여 Left, Right, Top, Bottom 지정
 - 자식요소에서 Dock 속성을 지정하지 않으면 나머지 공간을 채움 (Panel과 동일)

```
<DockPanel>
    <Rectangle Dock="Left" Width="100" Color="Red" />
    <Rectangle Dock="Top" Height="100" Color="Green" />
    <Rectangle Dock="Right" Width="100" Color="Blue" />
    <Rectangle Dock="Bottom" Height="100" Color="Teal" />
    <Rectangle Color="Yellow" />
</DockPanel>
```

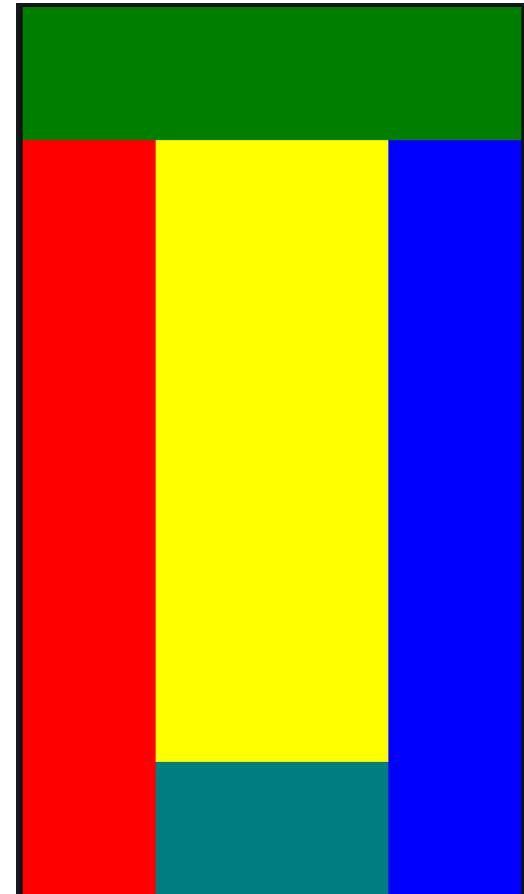


Layout 이해하기

- Layout 유형
 - DockPanel - 결과



자식요소의
순서를 변경하면??



Layout 이해하기

- Layout 유형
 - Grid
 - 자식요소를 행과 열을 사용하여 정렬
 - RowCount / ColumnCount 속성을 이용하여 행 및 열의 수 지정
 - Rows / Columns 속성을 이용하여 행 및 열의 수와 크기를 한번에 지정 할수 있음
 - 기본적으로 자식요소는 왼쪽에서 오른쪽(RowMajor), 위에서 아래로(ColumnMajor) 정렬 (ChildOrder 속성)
 - Row / Column 속성을 사용하여 셀의 위치를 명시적으로 지정 할 수 있음
 - 자식요소가 여러행이나 열을 차지하도록 하려면 RowSpan / ColumnSpan 속성을 사용할 수 있음

Layout 이해하기

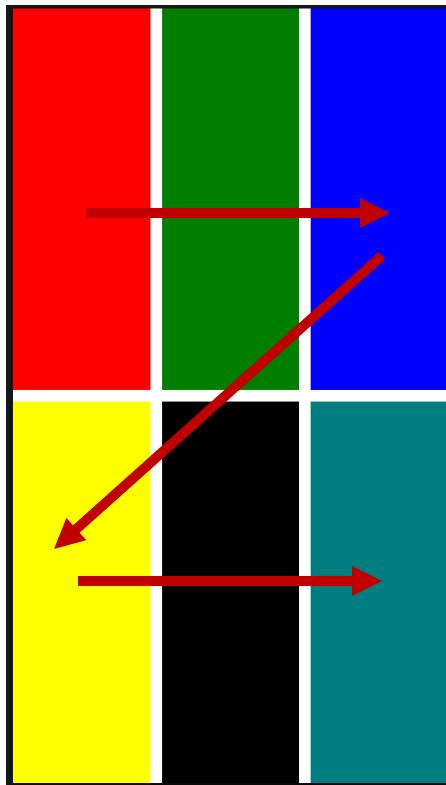
- Layout 유형
 - Grid 속성
 - RowCount
 - ColumnCount
 - CellSpacing
 - ChildOrder : RowMajor(default), ColumnMajor

```
<Grid RowCount="2" ColumnCount="3" CellSpacing="10"  
ChildOrder="RowMajor">  
    <Rectangle Color="Red" />  
    <Rectangle Color="Green" />  
    <Rectangle Color="Blue" />  
    <Rectangle Color="Yellow" />  
    <Rectangle Color="Black" />  
    <Rectangle Color="Teal" />  
</Grid>
```

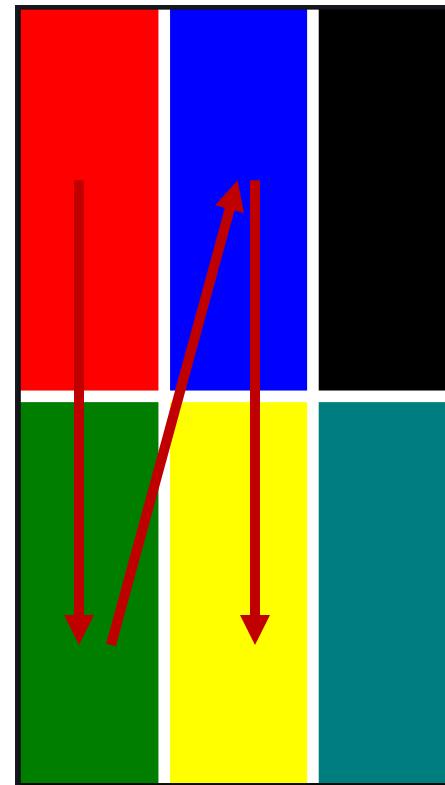


Layout 이해하기

- Layout 유형
 - Grid 속성 결과



ChildOrder=“RowMajor”



ChildOrder=“ColumnMajor”

Layout 이해하기

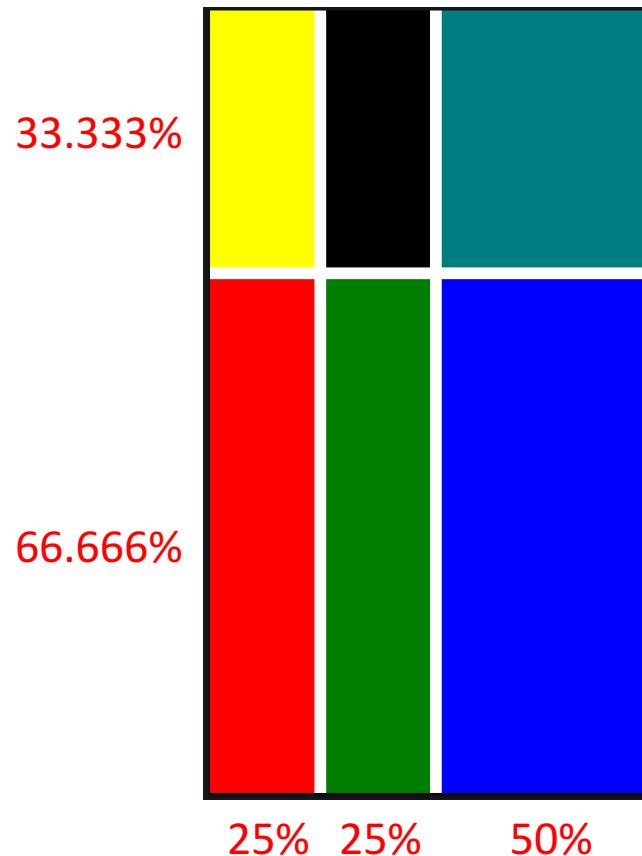
- Layout 유형
 - Grid 속성
 - Rows (예) 1행 $1+2=3$, $1/3=33.333\%$
 - “auto, 1*” 값 설정을 해보세요
 - Columns (예) 1열 $1+1+2=4$, $1/4=25\%$
 - “auto, 1*, 1*” 값 설정을 해보세요
 - Row
 - Column

```
<Grid Rows="1*, 2*" Columns="1*, 1*, 2*" CellSpacing="10">
    <Rectangle Color="Red" Row="1" Column="0" />
    <Rectangle Color="Green" Row="1" Column="1" />
    <Rectangle Color="Blue" Row="1" Column="2" />
    <Rectangle Color="Yellow" Row="0" Column="0" />
    <Rectangle Color="Black" Row="0" Column="1" />
    <Rectangle Color="Teal" Row="0" Column="2" />
</Grid>
```



Layout 이해하기

- Layout 유형
 - Grid 속성 결과



Layout 이해하기

- Layout 유형
 - Grid 속성
 - RowSpan
 - ColumnSpan

```
<Grid Rows="1*, 2*" Columns="1*, 1*, 2*" CellSpacing="10">
    <Rectangle Color="Red" RowSpan="2"/>
    <Rectangle Color="Green" ColumnSpan="2" />
    <Rectangle Color="Black" />
    <Rectangle Color="Teal" />
</Grid>
```

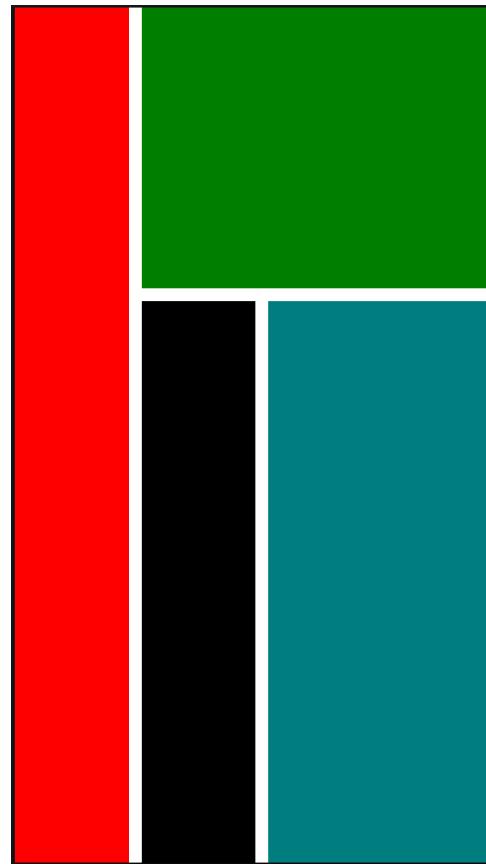


Layout 이해하기

- Layout 유형
 - Grid 속성 결과

ColumnSpan="2"

RowSpan="2"



Layout 이해하기

- Layout 유형
 - WrapPanel
 - 자식요소를 지정된 방향으로 차례대로 정렬하다 영역의 끝에 도달 할 경우 다음 칸부터 다시 정렬
 - FlowDirection 속성을 이용하여 왼쪽 => 오른쪽(LeftToRight), 오른쪽 => 왼쪽(RightToLeft) 정렬 방향을 지정
 - Orientation 속성을 이용하여 가로(Horizontal), 세로(Vertical) 진행방향 설정

Layout 이해하기

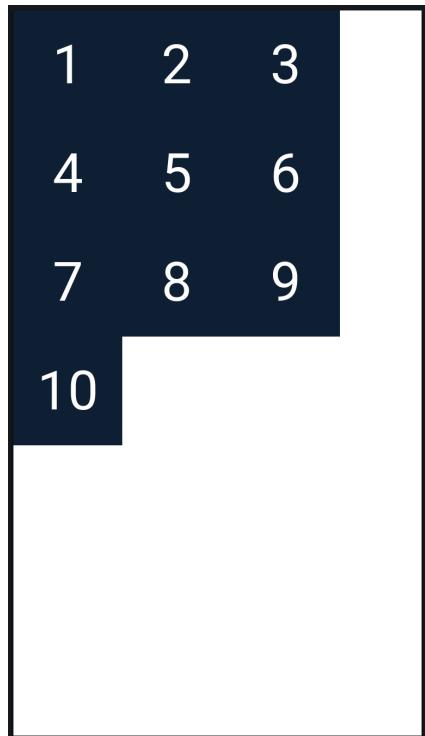
- Layout 유형
 - WrapPanel

```
<Rectangle ux:Class="MyRec" Width="100" Height="100">
    <string ux:Property="TextValue" />
    <Text Alignment="Center" TextColor="White" FontSize="50"
Value="{Property TextValue}" />
</Rectangle>

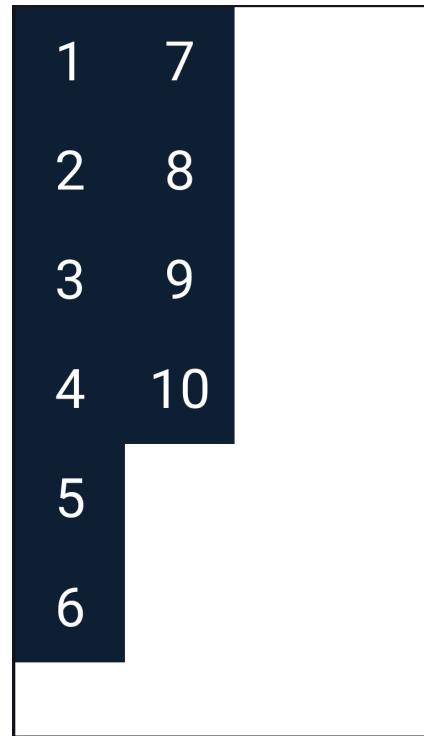
<WrapPanel FlowDirection="LeftToRight">
    <MyRec TextValue="1" Color="#112233" />
    <MyRec TextValue="2" Color="#112233" />
    <MyRec TextValue="3" Color="#112233" />
    <MyRec TextValue="4" Color="#112233" />
    <MyRec TextValue="5" Color="#112233" />
    <MyRec TextValue="6" Color="#112233" />
    <MyRec TextValue="7" Color="#112233" />
    <MyRec TextValue="8" Color="#112233" />
    <MyRec TextValue="9" Color="#112233" />
    <MyRec TextValue="10" Color="#112233" />
</WrapPanel>
```

Layout 이해하기

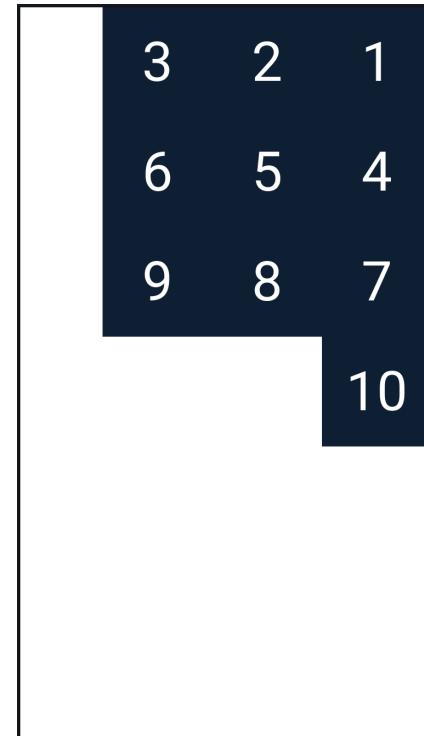
- Layout 유형
 - WrapPanel 결과 (refresh 필요)



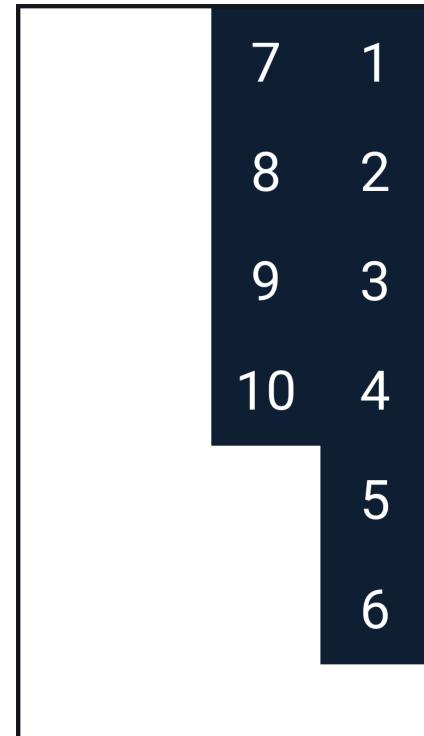
FlowDirection="LeftToRight"
Orientation="Horizontal"



FlowDirection="LeftToRight"
Orientation="Vertical"



FlowDirection="RightToLeft"
Orientation="Horizontal"



FlowDirection="RightToLeft"
Orientation="Vertical"

Layout 이해하기

- Layout 유형
 - ColumnLayout
 - Panel안에 위치하며 자식요소를 가로 또는 세로 형태로 배치
 - 속성
 - Orientation : 가로(Horizontal), 세로(Vertical) 방향 설정
 - ColumnCount : 열의 수 지정
 - ColumnSize : 열의 크기 설정
 - ColumnSpacing : 각 열 사이의 간격
 - ItemSpacing : 열의 각 항목사이의 간격
 - Sizing : 열이 사용가능한 공간을 채울지 여부(Fixed / Fill)

Layout 이해하기

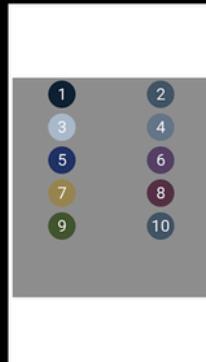
- Layout 유형
 - ColumnLayout 결과

Code >

```
<JavaScript>
  var items = [
    {text:"1", color:"#112233"},  
    {text:"2", color:"#445566"},  
    {text:"3", color:"#AABBCC"},  
    {text:"4", color:"#667788"},  
    {text:"5", color:"#223344"},  
    {text:"6", color:"#554365"},  
    {text:"7", color:"#998754"},  
    {text:"8", color:"#543345"},  
    {text:"9", color:"#445432"},  
    {text:"10", color:"#445565"}
  ];
  module.exports = {
    items
  }
</JavaScript>
<Circle ux:Class="MyCircle" Height="50" Width="50" Margin="5">  
  <string ux:Property="Text" />  
  <Text Alignment="Center" TextColor="White" FontSize="30" Value="{Property Text}" />  
</Circle>

<Panel Alignment="Center" Color="#909090" Height="400">  
  <ColumnLayout Orientation="Vertical" />  
  <Each Items="{items}">  
    <MyCircle Text="{text}" Color="{color}" />  
  </Each>  
</Panel>
```

Result >



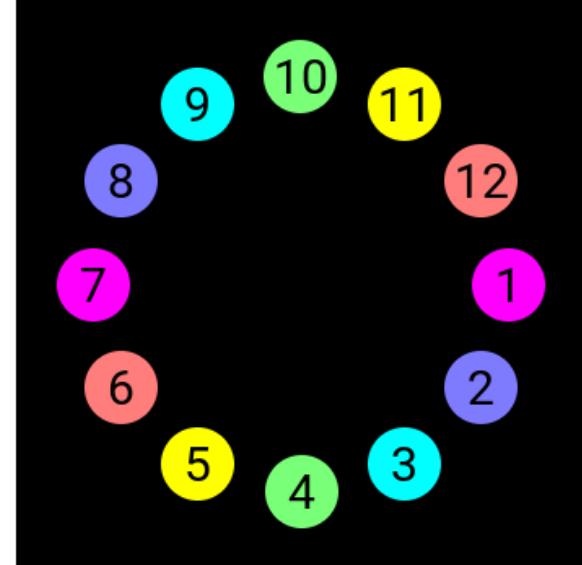
Layout 이해하기

- Layout 유형
 - CircleLayout
 - Panel안에 위치하며 자식요소를 원을 그리며 배치
 - 속성
 - StartAngleDegrees : 원 세그먼트가 시작되는 각도(0도: 3시)
 - EndAngleDegrees: 원 세그먼트가 끝나는 각도
 - ItemSpacingDegrees : 원의 각 요소 사이의 간격(도)
 - Radius : 원의 반지름

Layout 이해하기

- Layout 유형
 - CircleLayout 결과

```
<JavaScript>
var items = [
  {text:"1", color:"#ff00ff"},  
  {text:"2", color:"#7f7fff"},  
  {text:"3", color:"#00ffff"},  
  {text:"4", color:"#7fff7f"},  
  {text:"5", color:"#ffff00"},  
  {text:"6", color:"#ff7f7f"},  
  {text:"7", color:"#ff00ff"},  
  {text:"8", color:"#7f7fff"},  
  {text:"9", color:"#00ffff"},  
  {text:"10", color:"#7fff7f"},  
  {text:"11", color:"#ffff00"},  
  {text:"12", color:"#ff7f7f"}  
];  
module.exports = {  
  items  
}  
</JavaScript>  
  
<Circle ux:Class="MyCircle" >  
  <string ux:Property="Text" />  
  <Text Alignment="Center" TextColor="Black" FontSize="30" Value="{Property Text}" />  
</Circle>  
  
<Panel Width="350" Height="350">  
  <Panel Width="300" Height="300">  
    <CircleLayout ItemSpacingDegrees="10"/>  
    <Each Items="{items}">  
      <MyCircle Text="{text}" Fill="{color}" />  
    </Each>  
  </Panel>  
</Panel>
```





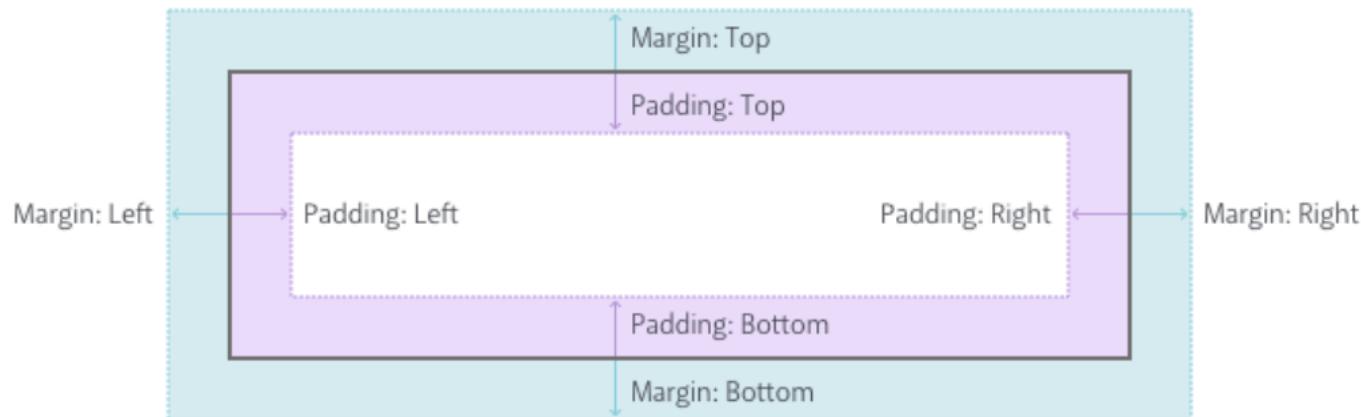
[Part 1]

Layout 활용

Layout 활용

- Margin
 - 요소의 바깥쪽 여백을 설정하는 속성
 - 4개의 값으로 왼쪽, 위, 오른쪽, 아래 순서로 구성

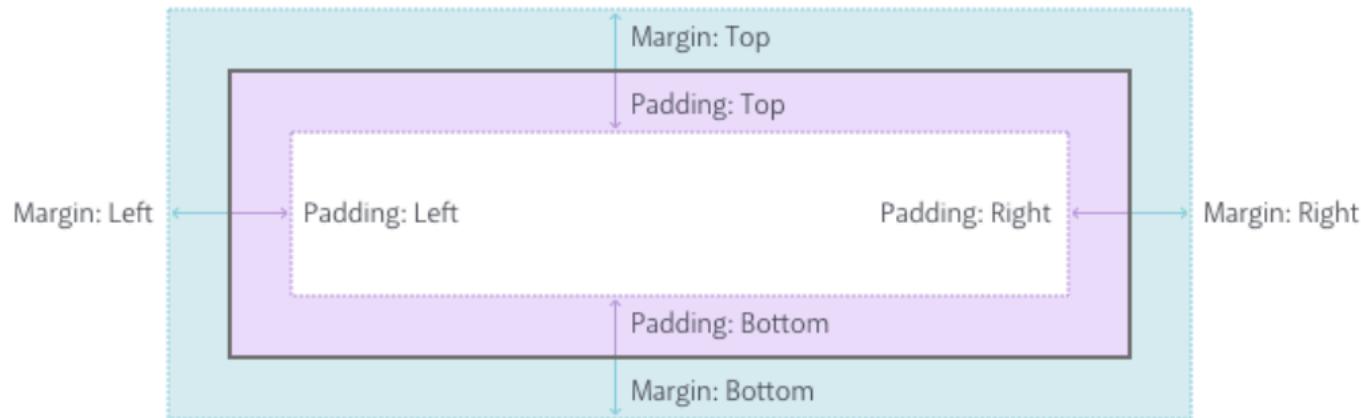
```
<Panel Margin="10,20,30,40">
<Panel Margin="10"> <!--10,10,10,10-->
<Panel Margin="10,20"> <!--10,20,10,20-->
```



Layout 활용

- Padding
 - 요소의 안쪽 여백을 설정하는 속성
 - 4개의 값으로 왼쪽, 위, 오른쪽, 아래 순서로 구성

```
<Panel Padding="10,20,30,40">  
<Panel Padding="10"> <!--10,10,10,10-->  
<Panel Padding="10,20"> <!--10,20,10,20-->
```



Layout 활용

- Alignment

- 모든 객체는 Alignment 속성을 가질 수 있으며 사용가능한 공간 보다 작은 경우 객체가 어떻게 배치되어야 하는지를 결정
- Alignment 속성은 다음 값 중 하나로 설정
 - Default - 사용가능한 모든 영역으로 객체를 늘림
 - Left - 요소를 왼쪽으로 정렬
 - HorizontalCenter - 요소를 수평 가운데 정렬
 - Right - 요소를 오른쪽으로 정렬
 - Top - 요소를 상단으로 정렬
 - VerticalCenter - 요소를 수직 가운데 정렬
 - Bottom - 요소를 하단으로 정렬
 - TopLeft - 요소를 왼쪽 상단으로 정렬

Layout 활용

- Alignment
 - Alignment 속성은 다음 값 중 하나로 설정 (계속)
 - TopCenter - 요소를 상단 가운데로 정렬
 - TopRight - 요소를 오른쪽 상단으로 정렬
 - CenterLeft - 요소를 세로 가운데 정렬하고 왼쪽으로 정렬
 - Center - 요소를 가로 및 세로 모두 가운데 정렬
 - CenterRight - 요소를 세로 가운데 정렬하고 오른쪽으로 정렬
 - BottomLeft - 요소를 왼쪽 하단으로 정렬
 - BottomCenter - 요소를 하단 가운데로 정렬
 - BottomRight - 요소를 오른쪽 상단으로 정렬

Layout 활용

- Units
 - Fuse 기본 단위는 Point
 - Fuse에서 사용하는 단위는
 - Percent(%) : 부모 크기에 상대적인 비율로 표시 됨
 - Point(기본값) : 모든 화면밀도에서 동일한 크기로 표시 됨
 - Android : Device independent pixel(dp) 사용
 - iOS : Point(pt) 사용
 - Pixels : 화면밀도에 따라 다른 크기로 표시 됨

Layout 활용

- Units
 - pt / dp VS px

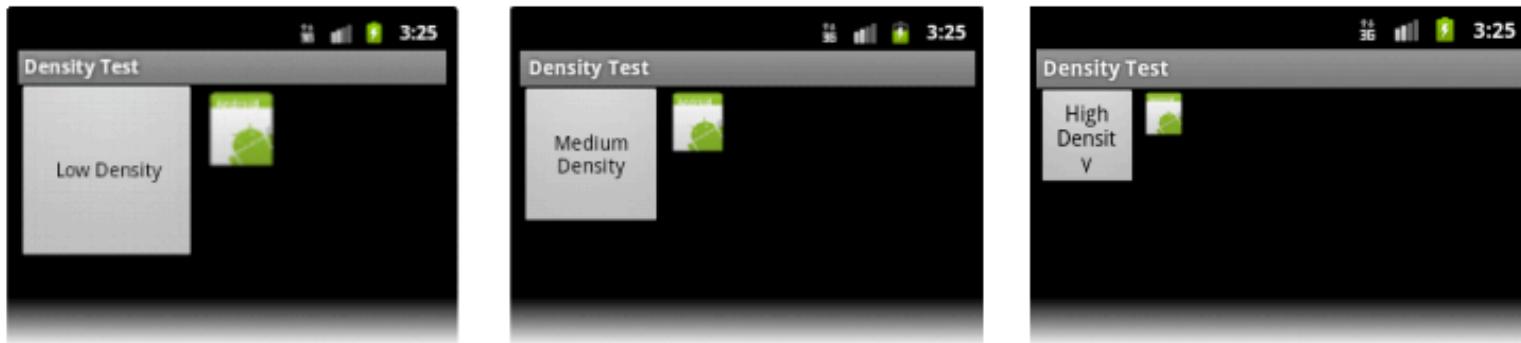


그림 2. 다양한 밀도(저밀도, 중간 밀도 및 고밀도 화면)를 지원하지 않는 예시 애플리케이션.

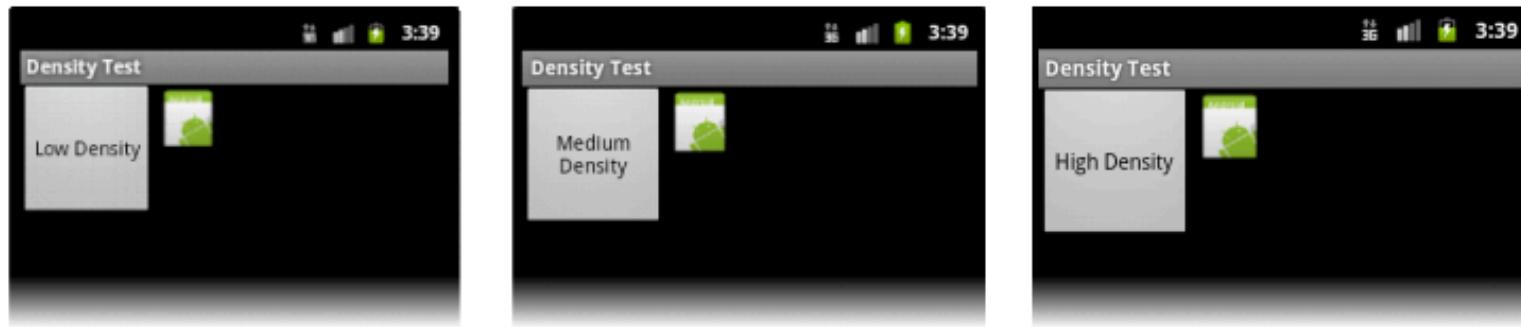


그림 3. 다양한 밀도(저밀도, 중간 밀도 및 고밀도 화면)를 올바로 지원하는 예시 애플리케이션(밀도에 독립적).

스타일 적용하기

스타일 적용하기

- Fuse Styling 은 간단하고 직관적이다.
 - 모든 객체는 속성을 이용하여 스타일을 지정 할 수 있으며 스타일 자체를 객체로 만들수도 있다.
 - 스타일을 객체로 지정하면 애니메이션 및 전환작업 시 부모와 별도로 스타일을 조작 할 수 있는 유연성을 제공 한다.

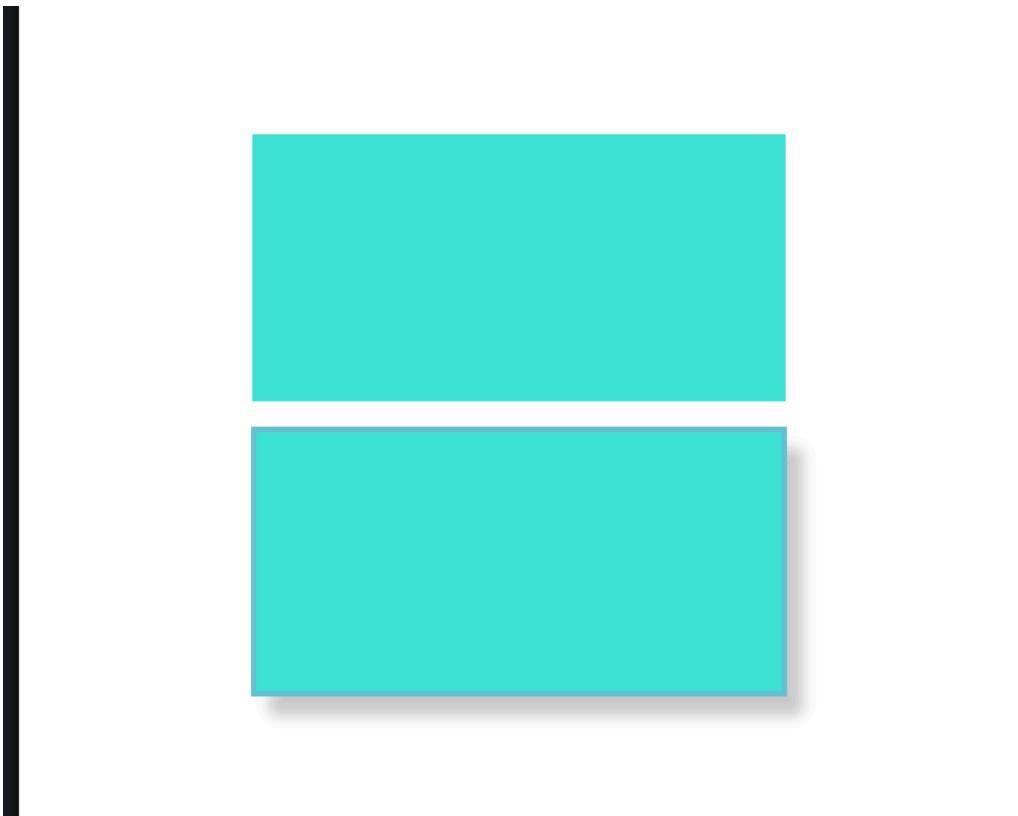
```
<StackPanel ItemSpacing="10" Alignment="Center">
    <Rectangle Width="200" Height="100" Color="#46E3D6" />

    <Rectangle Width="200" Height="100">
        <SolidColor Color="#46E3D6" />
        <Stroke Color="#6DBFD2" Width="2" />
        <Shadow Color="#0003" Distance="10" Angle="130" />
    </Rectangle>
</StackPanel>
```



스타일 적용하기

- Rectangle Styling

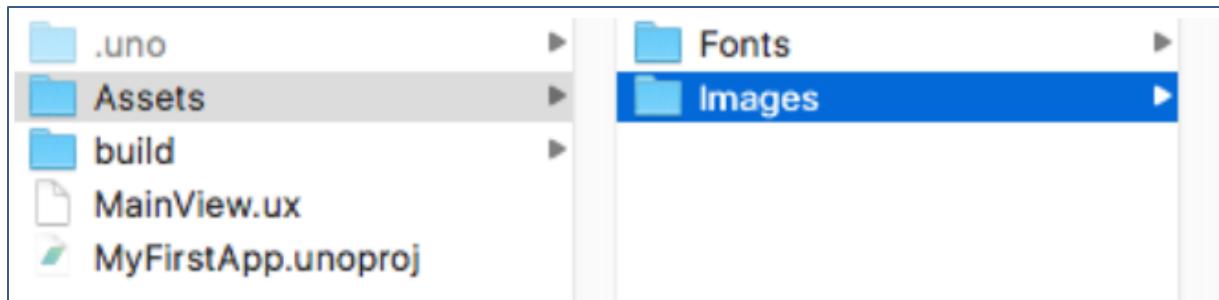


스타일 적용하기

- Assets 사용하기
 - 모든 Assets(FONTs, Images)은 Fuse프로젝트 안에 배치

Project Folder > Assets > Asset Type

- 프로젝트 폴더 구조



```
<Panel Background="Navy">
    <Image File="Assets/Images/fuselogo.png" />
</Panel>
```

스타일 적용하기

- Multi Density Images 사용하기
 - Fuse에서 화면밀도에 맞는 이미지를 자동으로 선택
 - Density속성은 Fuse가 현재 화면밀도를 계산하여 올바른 값을 표시하도록 해줌
 - retina and non-retina 화면에서 동일하게 나타남

```
<Image Background="Green">
    <MultiDensityImageSource>
        <FileImageSource Density="1" File="Assets/Images/fuselogo.png" />
        <FileImageSource Density="2" File="Assets/Images/fuselogo@2x.png" />
        <FileImageSource Density="3" File="Assets/Images/fuselogo@3x.png" />
    </MultiDensityImageSource>
</Image>
```

스타일 적용하기

- **Text Styling**
 - Assets 폴더에 Font 파일을 추가하여 원하는 Font를 사용할 수 있음
 - 앱에서 전역적으로 사용할 수 있는 리소스로 만들기 위해 ux:Global로 지정

```
<Font File="Assets/Fonts/Comic Sans Ms.ttf" ux:Global="MyFont" />
<Text FontSize="30" Value="Default Font" />
<Text Font="MyFont" FontSize="30" Value="Comic Sans Ms Font" />
```

스타일 적용하기

- Color Palettes 만들기
 - 전역적으로 사용할 수 있는 Color 리소스를 만들어 재사용
 - float4 value-type을 사용 함
 - float4 : Fuse에서 Color값을 표시하는 유형 (16진수 : #FFFFFF)

```
<Font File="Assets/Fonts/Comic Sans Ms.ttf" ux:Global="MyFont" />
  <float4 ux:Global="MyFavColor" ux:Value="#8A5182" />
  <float4 ux:Global="MyFavColor2" ux:Value="#207ce5" />

  <Text FontSize="30" Value="Default Font" Color="MyFavColor" />
  <Text Font="MyFont" FontSize="30" Value="Comic Sans Ms Font"
Color="MyFavColor2" />
```

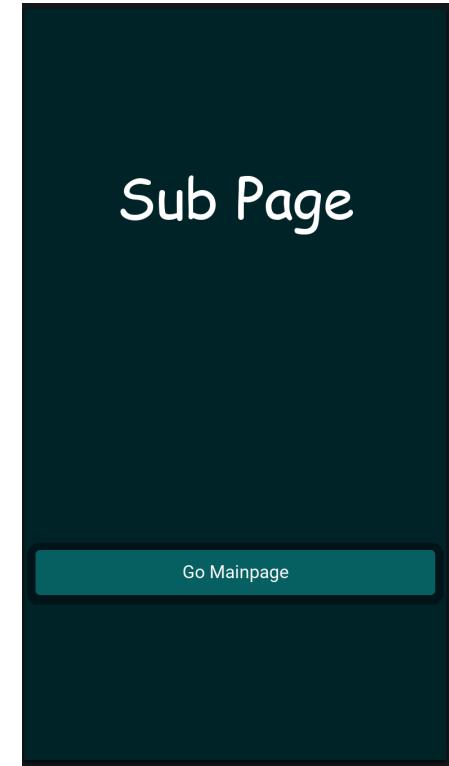
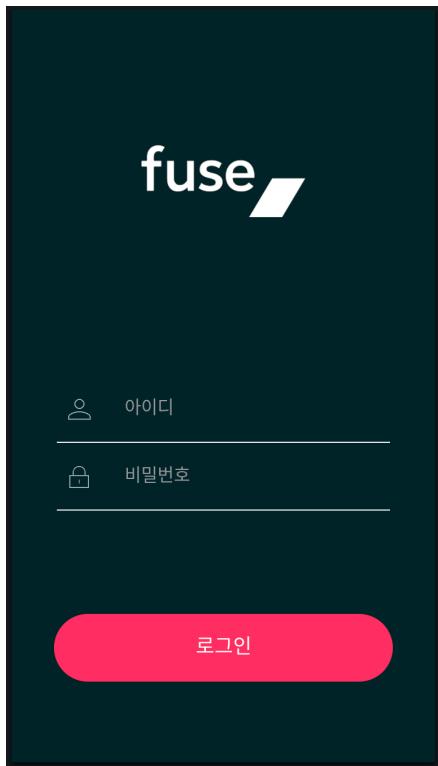
실전 1)

앱 개별 화면 설계

(Splash, 로그인, 메인, 서브 화면)

실전1) 앱 개별 화면 설계

- Splash, 로그인, 메인, 서브 화면 만들기



Fuse your potential

Animation 이해 & 구현

Animation 이해 및 구현

- Fuse Animation
 - Fuse에서 Animation은 가장 핵심이 되는 기술
 - Animation을 만들려면 Trigger 와 Animator 두 가지 핵심 개념이 필요
 - Trigger : 이벤트 또는 사용자 입력에 응답하고 Animator를 활성화 하는 객체
 - Animator : Trigger에 의해 활성화 되며 시간이 지남에 따라 객체 변형 또는 속성 변경을 담당
 - Trigger가 비활성화 될 때마다 객체를 다시 Animate 할 필요가 없다. Fuse가 알아서 처리 함

Animation 이해 및 구현

- Trigger
 - 이벤트 또는 사용자 입력에 응답하고 Animator를 활성화 하는 객체
 - Trigger 종류
 - WhilePressed : 화면을 누르고 있는 동안 활성화
 - Clicked : 화면을 눌렀다 놓으면 활성화
 - WhileTrue : UX에서 Toggle하거나 JavaScript에서 데이터 값을 boolean 값으로 변경 가능
 - AddingAnimation / RemovingAnimation : 화면에 추가 되거나 삭제 된 객체에 Animation 적용
 - LayoutAnimation : 객체가 레이아웃 엔진에 의해 새로운 위치 또는 크기로 지정 될 때 Animation 적용
 - WhileActive : 현재 활성 페이지에 있는 동안 활성화

Animation 이해 및 구현

- Animator

- Trigger에 의해 활성화 되며 시간이 지남에 따라 객체 변형 또는 속성 변경을 담당
- Animator 종류
 - Move : 객체를 이동 시킴
 - Scale : 객체의 크기를 조정
 - Rotate : 객체를 회전 시킴 (2D, 3D 회전 가능)
 - Change : 객체의 모든 속성 변경에 대해 Animation 적용
 - Skew : 객체의 기울기를 변환
 - Spin : 객체를 지속적으로 회전 시킴

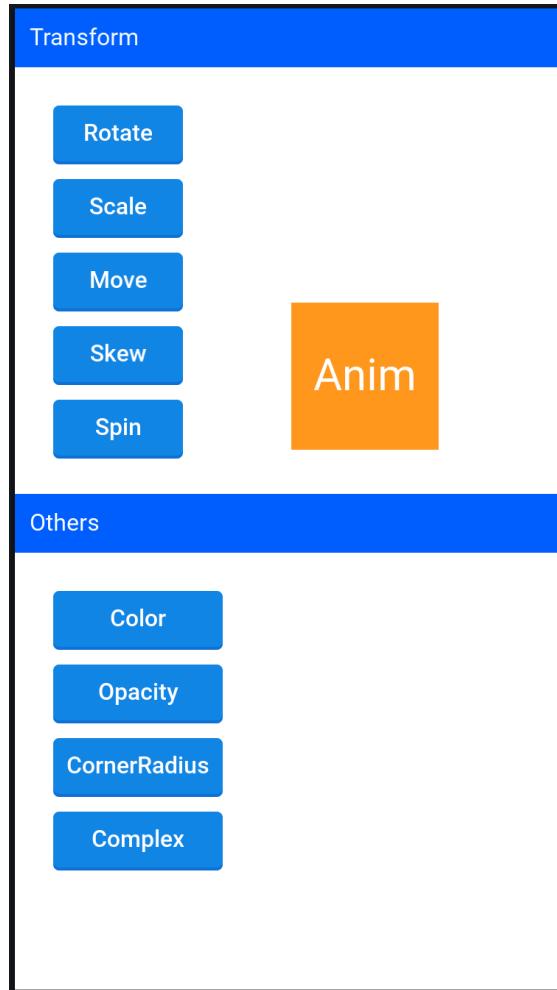
Animation 이해 및 구현

- Animator
 - Animator 속성
 - Duration : Animation 시작부터 끝날 때까지 걸리는 시간 (초)
 - DurationBack : 역방향 Animation 시작부터 끝날 때까지 걸리는 시간 (초)
 - Delay : Animation 시작 전 지연 시간 (초)
 - DelayBack : 역방향 Animation 시작 전 지연 시간 (초)
 - Easing / EasingBack : Fuse에서 미리정의 된 Animation 효과
 - Default : Linear



Animation 이해 & 구현

- Animation 구현



Fuse your potential

Navigation

(PageControl, Router, Navigator)

Navigation

- PageControl
 - 화면을 Swipe 하여 페이지를 탐색
 - Active 속성을 이용하여 기본 페이지 및 현재 페이지 설정

```
<PageControl ux:Name="pagecontrol" Active="Page1">
    <Page Name="Page1" Background="Red">
        <Text Value="Page1" Color="White" Alignment="Center" />
        <Clicked>
            <Set pagecontrol.Active="Page2" />
        </Clicked>
    </Page>
    <Page Name="Page2" Background="Blue">
        <Text Value="Page2" Color="White" Alignment="Center" />
        <Clicked>
            <Set pagecontrol.Active="Page1" />
        </Clicked>
    </Page>
</PageControl>
```

Navigation

- Router
 - 앱에 대한 라우팅 및 네비게이션 기록을 관리
 - PageControl, Navigator와 함께 사용하여 Fuse에서 페이지 탐색의 기초를 형성
 - Fuse 앱은 페이지 이동 시 이동 정보가 Router Instance로 전송 되며 이 정보는 이동 할 페이지 경로와 보낼 데이터로 구성 됨
 - Router Interface
 - push(path, parameter) : 현재 경로 스택에서 새 경로를 푸쉬
 - Router history에 새 경로를 추가
 - goto(path, parameter) : 새 경로로 이동
 - Router history 삭제
 - goBack() : 이전 페이지로 돌아 갑니다. (goto X)

Navigation

- Navigator
 - Router와 함께 사용
 - Page 템플릿을 사용하여 페이지를 인스턴스화하고 재활용 할 수 있음 (ux:Template)
 - DefaultPath 속성을 이용하여 기본 페이지 템플릿을 지정 할 수 있음

Navigation

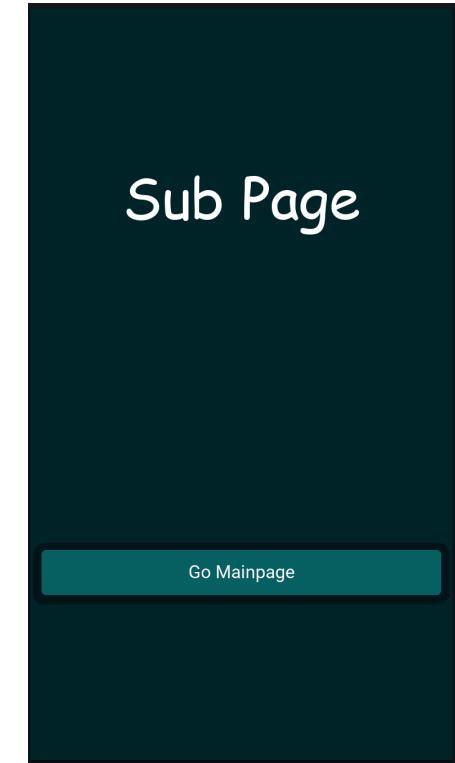
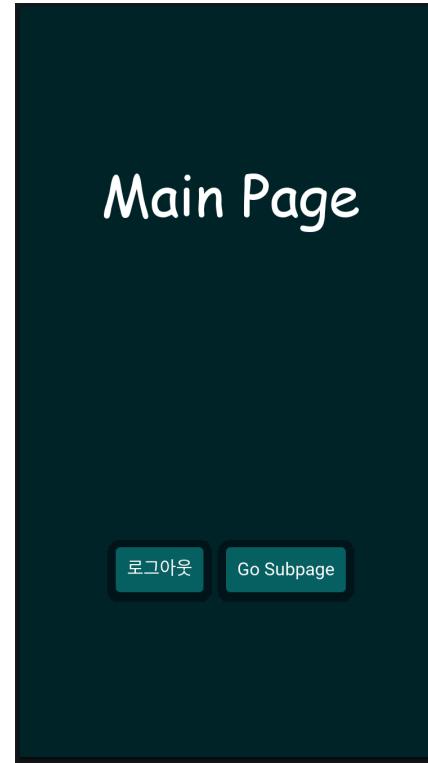
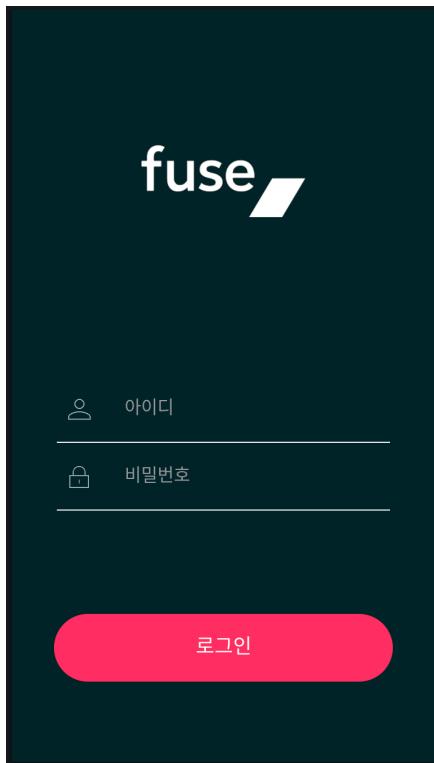
- Navigator

```
<JavaScript>
  module.exports = {
    gotoFirst: function() { router.push("Page1"); }
    gotoSecond: function() { router.push("Page2"); }
  };
</JavaScript>

<Router ux:Name="router" />
<DockPanel>
  <Navigator DefaultPath="Page1">
    <Page ux:Template="Page1" Background="Red">
      <Text Alignment="Center" Color="White">Page1</Text>
    </Page>
    <Page ux:Template="Page2" Background="Blue">
      <Text Alignment="Center" Color="White">Page2</Text>
    </Page>
  </Navigator>
  <Grid Dock="Bottom" Columns="1*,1*">
    <Basic.Button Text="First page" Padding="20" Clicked="{gotoFirst}" />
    <Basic.Button Text="Second page" Padding="20" Clicked="{gotoSecond}" />
  </Grid>
</DockPanel>
```

실전2) 앱 개별 화면 Navigation 연결

- Splash, 로그인, 메인, 서브 화면 Navigation 연결하기



UX markup 클래스 정의 및 재사용

UX 마크업 클래스

- 클래스를 사용하지 않았을 때
 - 반복적인 태그 요소 (그룹)을 그대로 사용할 수 밖에 없음

```
<App>
    <StackPanel ItemSpacing="10">
        <Rectangle Color="#f0a">
            <Text Value="사각형" Margin="10" />
        </Rectangle>

        <Rectangle Color="#f0a">
            <Text Value="사각형" Margin="10" />
        </Rectangle>

        <Rectangle Color="#f0a">
            <Text Value="사각형" Margin="10" />
        </Rectangle>
    </StackPanel>
</App>
```



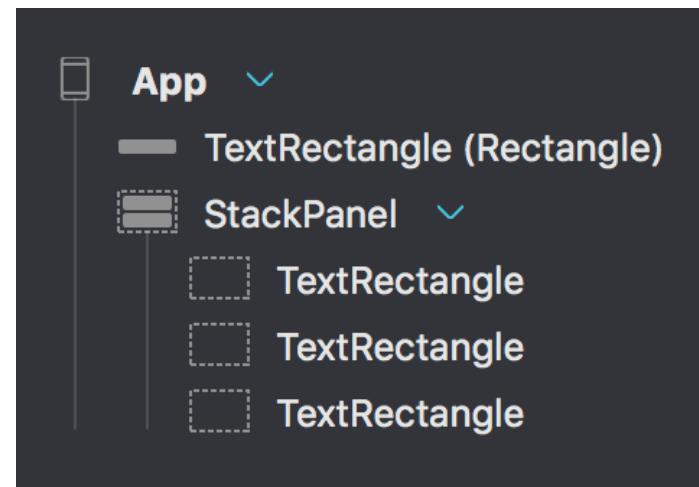
UX 마크업 클래스

- 클래스 정의하기
 - 정의할 태그 속성으로 "ux:Class" 를 지정

```
<Rectangle ux:Class="TextRectangle" Color="#f0a">
    <Text Value="사각형" Margin="10" />
</Rectangle>
```

- 클래스를 태그처럼 사용

```
<StackPanel ItemSpacing="10">
    <TextRectangle />
    <TextRectangle />
    <TextRectangle />
</StackPanel>
```



자바스크립트 & 데이터 바인딩

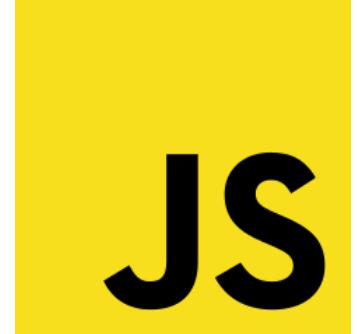
Fuse와 자바스크립트

- Fuse에서의 비즈니스 로직 구현
 - UX 마크업은 화면 UI 및 UX를 정의
 - 비즈니스 로직은 자바스크립트로 구현
- 자바스크립트 사용하기
 - <JavaScript> UX 마크업을 사용
 - 직접 자바스크립트를 작성하기

```
<JavaScript>
    console.log("Hello, Fuse!");
</JavaScript>
```

- 별도 자바스크립트 파일 내용을 가져오기

```
<JavaScript File="SomeCode.js" />
```



자바스크립트 변수 데이터 사용하기

- “module.exports”
 - 자바스크립트 변수값을 UX 마크업에 사용 가능하도록 선언
 - 변수값을 처리하는 비즈니스 로직을 자바스크립트로 구현 후 사용 가능

```
<App>
  <JavaScript>
    module.exports.foo = "bar";
  </JavaScript>
  <Panel>
    <Text Value="{foo}" />
  </Panel>
</App>
```

//(자바스크립트로 열심히 짜 보았는데.. 문제가?)

```
<App>
  <JavaScript>
    var hello = "world";

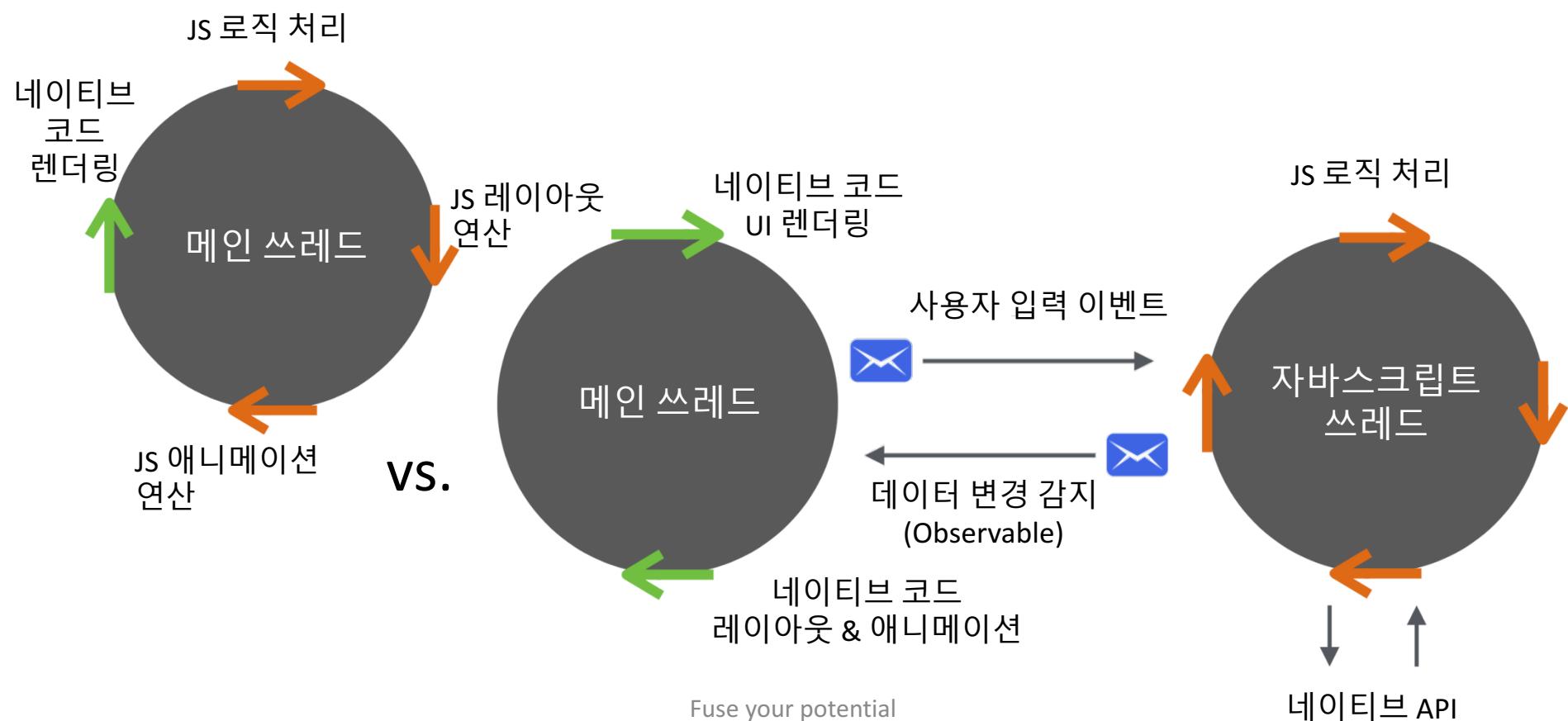
    function writeHello(){
      console.log("hello " + hello);
    }

    module.exports = {
      hello : hello,
      writeHello : writeHello
    };
  </JavaScript>
  <StackPanel>
    <StackPanel Orientation="Horizontal" ItemSpacing="10">
      <Text Value="Input your text:" Color="Blue" />
      <TextInput Value="{hello}" />
    </StackPanel>

    <Button Text="Click!" Clicked="{writeHello}" />
  </StackPanel>
</App>
```

자바스크립트: Observable

- Observable
 - 데이터 통신이 필요한 부분만을 사용
 - 쓰레드 간 통신을 최소화하고자 하는 목적



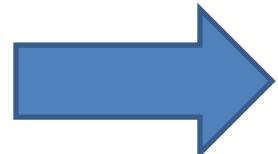
자바스크립트 & 데이터 바인딩

- 필요한 부분만을 Observable로 사용
 - Observable을 통해 해당 부분을 객체로 사용

```
<JavaScript>
    var hello = "world";

    function writeHello(){
        console.log("hello " + hello);
    }

    module.exports = {
        hello : hello,
        writeHello : writeHello
    };
</JavaScript>
```



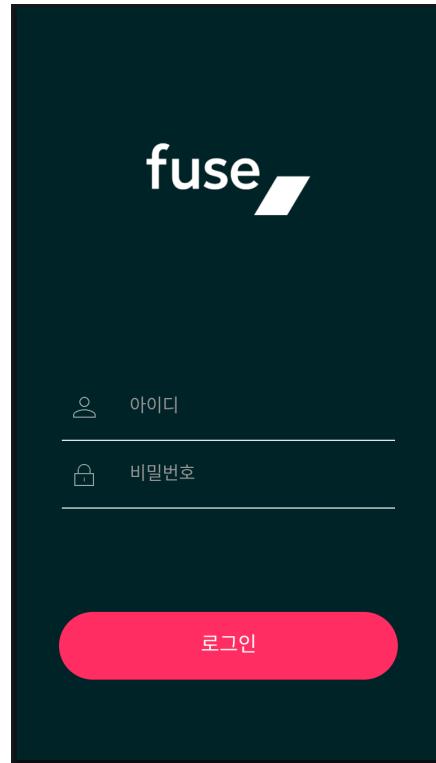
```
<JavaScript>
    var Observable = require("FuseJS/Observable");
    var hello = Observable("world");

    function writeHello(){
        console.log("hello " + hello.value);
    }

    module.exports = {
        hello : hello,
        writeHello : writeHello
    };
</JavaScript>
```

[실전3] 로그인 및 버튼 기능 연동 (자바스크립트)

- 로그인 성공 아이디 / 비밀번호 확인
- 로그인 실패시 메시지 출력



Fuse your potential

반복 (Each) 사용하기

반복이 필요한 이유

- 클래스로 반복되는 내용을 줄일 수 있으나,

```
<Rectangle ux:Class="TextRectangle" Color="#f0a">
    <Text Value="사각형" Margin="10" />
</Rectangle>
```

```
<StackPanel ItemSpacing="10">
    <TextRectangle />
    <TextRectangle />
    <TextRectangle />
</StackPanel>
```

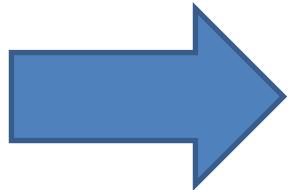
- 이 경우, <TextRectangle />이 더 많으면?

```
<StackPanel ItemSpacing="10">
    <TextRectangle />
    <TextRectangle />
    <!-- ... -->
    <TextRectangle />
</StackPanel>
```

Each 태그

- 반복할 부분을 태그 안에 넣음
- 반복 횟수 지정 가능

```
<StackPanel ItemSpacing="10">
    <TextRectangle />
    <TextRectangle />
    <!-- ... -->
    <TextRectangle />
</StackPanel>
```



```
<StackPanel ItemSpacing="10">
    <Each Count="10">
        <TextRectangle />
    </Each>
</StackPanel>
```

Each 태그 적용 예시

<App>

```
<Rectangle ux:Class="TextRectangle" Color="#f0a">
    <Text Value="사각형" Margin="10" />
</Rectangle>

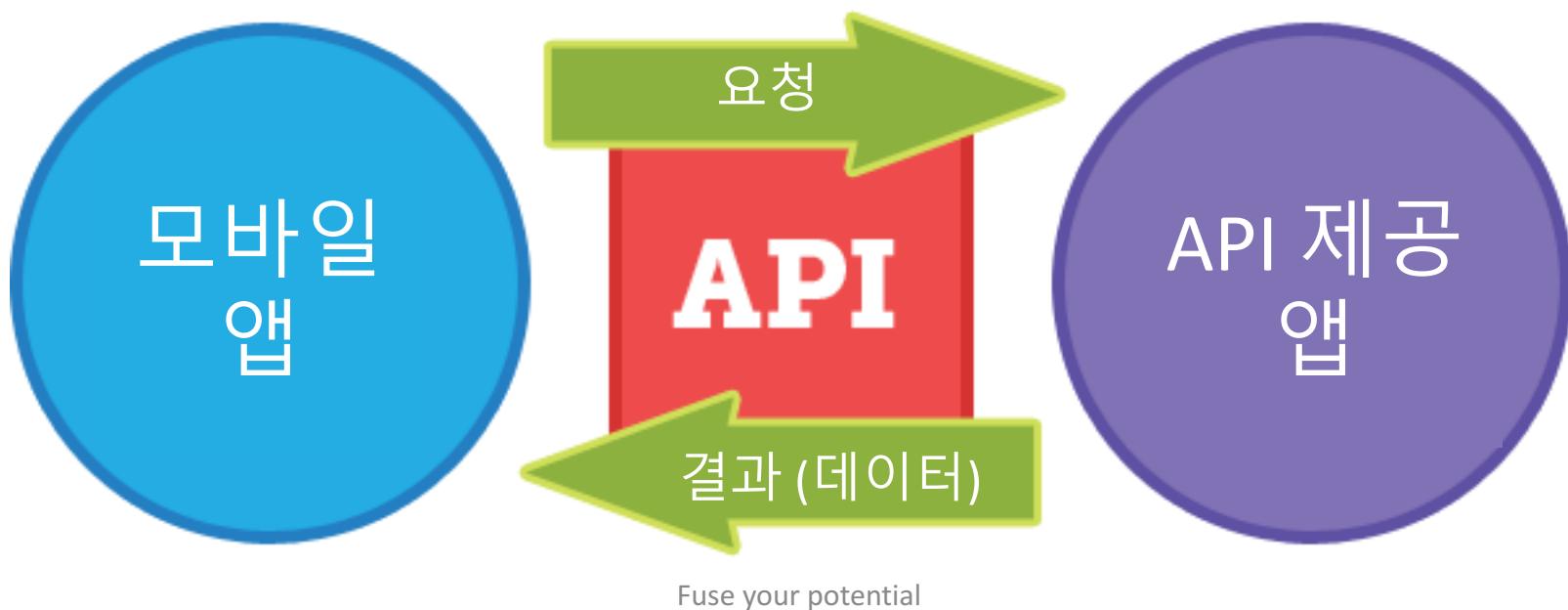
<StackPanel ItemSpacing="10">
    <Each Count="10">
        <TextRectangle />
    </Each>
</StackPanel>
</App>
```



Rest API

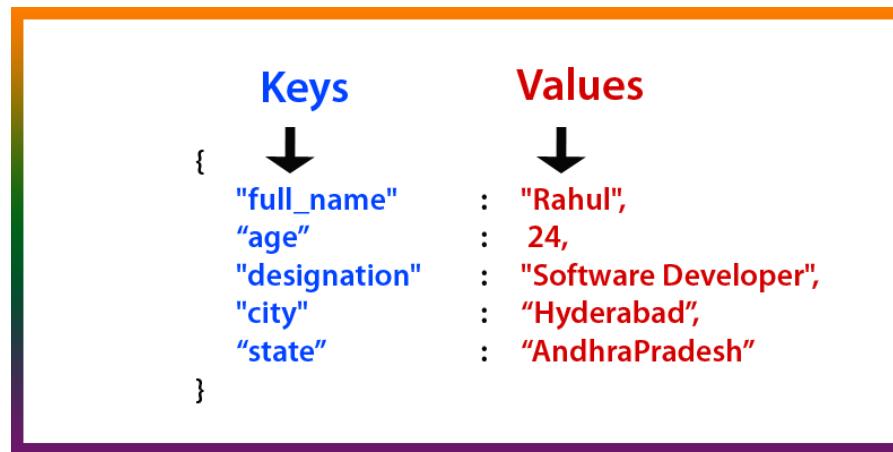
이해 및 활용

- API (Application Programming Interface)
 - 응용프로그램 프로그래밍 인터페이스
 - 어떤 응용프로그램과 연동하여 사용하고자 할 때 어떻게 접근하는지에 대한 방식 (인터페이스)
 - 보통 요청을 하여 데이터 결과를 가져옴
 - 예: 날씨 데이터, 최신 뉴스, ...



REST API란

- Rest API (Representational State Transfer API)
 - 오늘날 인터넷에서 제공하는 많은 API들은 REST API라는 규격으로 서비스를 제공함
 - 동작 방식
 - 앱에서는 GET / POST / PUT / DELETE 라는 규격화된 방식으로 요청
GET: 조회 / POST: 등록 / PUT: 수정 / DELETE: 삭제
 - 데이터 교환은 일반적으로 JSON 형태를 사용



모바일 앱 개발에서의 Rest API 사용

- 유용한 데이터를 가져와 보여주는 경우
 - 예: 날씨 정보, 뉴스 정보
- 로그인 연동
 - 기본 로그인 (ID, 비밀번호 확인을 따로 거침)
 - 페이스북 / 네이버 등 로그인과 연동
- ...
- (Fuse에서도 당연히 Rest API 연동 지원)

네이버 오픈 API 목록

현재 제공하고 있는 네이버 오픈 API 목록입니다. 네이버 오픈 API를 사용하기 전에 아래 사항들이 되어 있는지 확인 바랍니다.

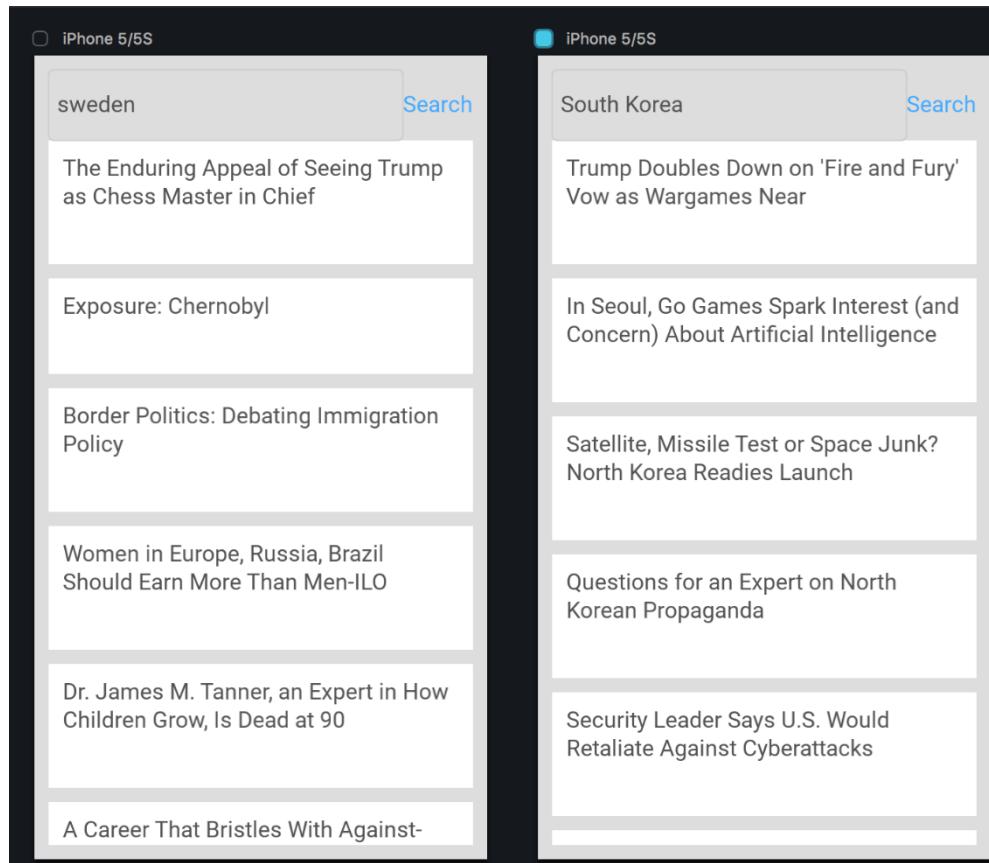
- 애플리케이션 등록 및 클라이언트 아이디와 시크릿 값 확인: [애플리케이션 등록 가이드](#) >
- API 권한 설정 확인: [API 권한 설정 및 호출 방법 가이드](#) >



REST API 연동 확인

- [분석]

- <https://github.com/kristianhasselknippe/FuseNYTSearchExample>



Fuse 프로젝트 파일 관리

권장하는 폴더 & 파일명

경로 / 파일명	설명
/[프로젝트 이름].unoproj	프로젝트 정의 파일
/MainView.ux	앱이 처음 시작될 때 참고하는 내용을 담은 파일로, <App> </App> 태그를 포함하는 것을 권장
/MainView.js	위 파일에서 메인으로 사용하는 자바스크립트를 담기
/Components	반복적으로 사용하는 클래스를 컴포넌트 폴더에 모아놓기
/Pages	각 페이지에 대한 내용을 모아놓기
/Assets	이미지, 사운드, 비디오, 폰트, 데이터 등을 보관

프로젝트 파일 관리

- [분석]

- <https://github.com/fusetools/hikr>

 Assets	Initial commit
 Components	master: Change Property->ReadProperty
 Modules	Initial commit
 Pages	Simplified EditHikePage layout a bit
 .gitignore	Initial commit
 LICENSE	Initial commit
 MainView.ux	changing DefaultTemplate to DefaultPath
 README.md	Updated readme to link to tutorial in docs
 hikr.gif	Initial commit
 hikr.unoproj	Initial commit

앱 패키징을 위한 추가설정

■ 앱 패키징을 위한 추가 설정

- **.unoproj 파일**

- .unoproj파일은 프로젝트가 참조하는 구성요소 및 iOS, Android 앱의 패키지 하는 방법을 지정 하는 곳
- .unoproj파일은 다음과 같은 JSON 텍스트 파일 형식

```
{  
    "Packages": [  
        "Fuse",  
        "FuseJS"  
    ],  
    "Includes": [  
        "*"  
    ]  
}
```

■ 앱 패키징을 위한 추가 설정

- Android 설정

```
{  
  "Mobile": {  
    "Orientations": "Portrait"      -- 화면 가로/세로 설정  
  },  
  
  "Android": {  
    "ApplicationLabel": "My App",    -- 앱아이콘에 표시 되는 이름  
    "VersionCode": 412,              -- 앱 버전 코드  
    "VersionName": "0.5.2",          -- 앱 버전 네임  
    "Package": "com.mycompany.myapp" -- 앱 패키지명  
    "Icons": {  
      "LDPI": "Icon-ldpi-36x36.png",  
      "MDPI": "Icon-mdpi-48x48.png",  
      "HDPI": "Icon-hdpi-72x72.png",  
      "XHDPI": "Icon-xhdpi-96x96.png",  
      "XXHDPI": "Icon-xxhdpi-144x144.png",  
      "XXXHDPI": "Icon-xxxhdpi-192x192.png"  
    },  
  }  
}
```

apk 패키징을 위한 추가 설정

- Android 설정 (계속)

```
"Android": {  
    "Key": {  
        "Alias": "application",  
        "AliasPassword": "<alias password>",  
        "Store": "release.keystore",  
        "StorePassword": "<store password>"  
    },  
    "GooglePlay": {  
        "SenderID": "Push Sender Id"  
    },  
    "SDK": {  
        "BuildToolsVersion": "23.0.0",  
        "CompileVersion": 19,  
        "MinVersion": 10,  
        "TargetVersion": 19  
    }  
}
```

apk 패키징을 위한 추가 설정

- iOS 설정

```
"iOS": {  
    "BundleIdentifier": "com.mycompany.myapp", -- 앱 번들ID  
    "BundleName": "My App", -- 앱아이콘에 표시 되는 이름  
    "BundleVersion": "0.5.2", -- 앱 버전  
    "DeploymentTarget": "8.0", -- 앱실행 최소 iOS버전  
    "Icons": {  
        "iPhone_29_2x": "Icon-iPhone-29@2x.png",  
        "iPhone_29_3x": "Icon-iPhone-29@3x.png",  
        "iPhone_40_2x": "Icon-iPhone-40@2x.png",  
        "iPhone_40_3x": "Icon-iPhone-40@3x.png",  
        "iPhone_60_2x": "Icon-iPhone-60@2x.png",  
        "iPhone_60_3x": "Icon-iPhone-60@3x.png",  
        "iPad_29_1x": "Icon-iPad-29@1x.png",  
        "iPad_29_2x": "Icon-iPad-29@2x.png",  
        "iPad_40_1x": "Icon-iPad-40@1x.png",  
        "iPad_40_2x": "Icon-iPad-40@2x.png",  
        "iPad_76_1x": "Icon-iPad-76@1x.png",  
        "iPad_76_2x": "Icon-iPad-76@2x.png"  
    },  
}
```

apk 패키징을 위한 추가 설정

- iOS 설정

```
"iOS": {  
    "LaunchImages": { -- 화면밀도에 따른 LaunchImage  
        "iPhone_Portrait_2x": "...", // 640x960  
        "iPhone_Portrait_R4": "...", // 640x1136  
        "iPhone_Portrait_R47": "...", // 750x1334  
        "iPhone_Portrait_R55": "...", // 1242x2208  
        "iPhone_Landscape_R55": "...", // 2208x1242  
        "iPad_Portrait_1x": "...", // 768x1024  
        "iPad_Portrait_2x": "...", // 1536x2048  
        "iPad_Landscape_1x": "...", // 1024x768  
        "iPad_Landscape_2x": "..." // 2048x1536  
    }  
}
```

■ 앱 패키징을 위한 추가 설정

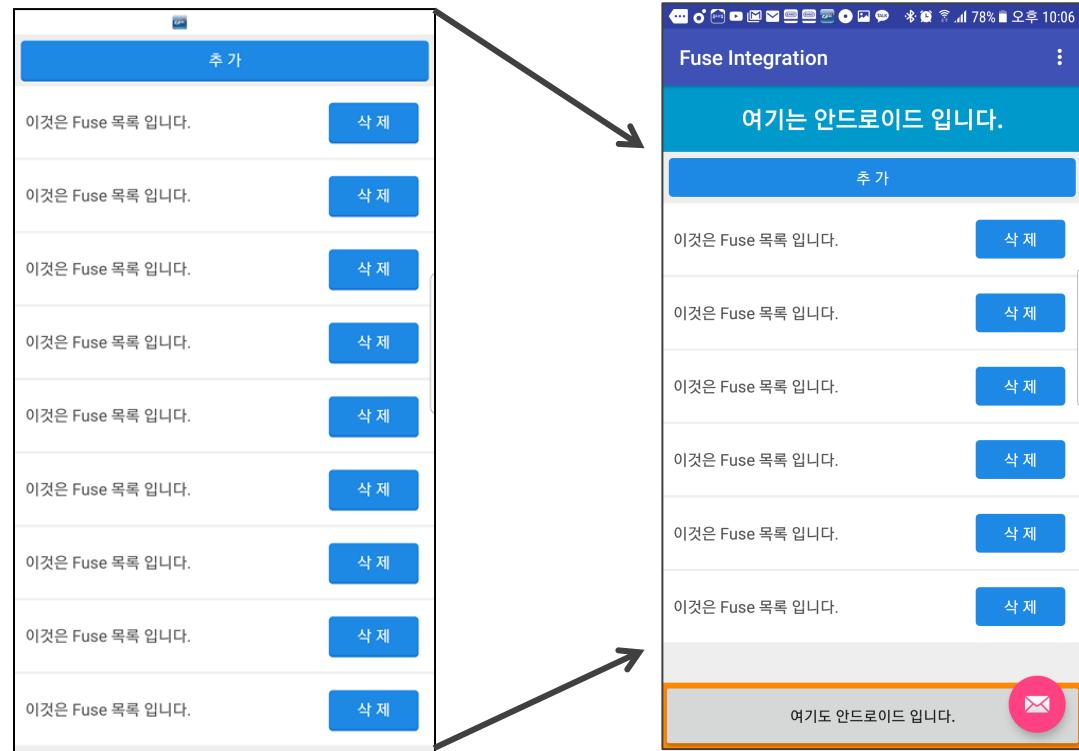
- iOS.PList 설정

```
"iOS": {  
    "PList": {  
        -- 앱이 작동 해야 할 디바이스 관련 기능  
        "UIRequiredDeviceCapabilities": [  
            "camera-flash",  
            "gps"  
        ],  
        "UIApplicationExitsOnSuspend": false, -- 홈버튼 눌렀을 때 앱 종료여부  
        "UIFileSharingEnabled": true, -- PC연결 시 iTunes 통해 파일공유 가능 여부  
        "UIViewEdgeAntialiasing": true -- Antialiasing 사용여부  
    }  
}
```

Fuse에서 지원하는 다른기능 살펴보기

Fuse Pro 기능 활용

- Xcode & Android Studio 인티그레이션
 - 강력한 UI/UX 구현은 이제 Fuse에서!
 - 기존 앱 UI/UX를 유지하면서 바꾸고 싶은 부분만 Fuse로 대체
 - 기존 사용 앱을 업그레이드 가능



경청해 주셔서 감사합니다

- Fuse를 더 공부하고 싶다면?
 - Fusetools - <https://www.fusetools.com>
 - 한글문서 : <https://wonthada.com/?docs=fusetools-docs>
 - 한국사용자 그룹 : <https://www.facebook.com/groups/fusetools/>
 - Youtube(해외) - https://www.youtube.com/channel/UCPizp_2dBkLIXRFnbieG3Qw
 - Youtube(국내) -
https://www.youtube.com/playlist?list=PLaOLyVLvZXGcnauh_LTb5oJM2W6inHmFj
 - 카페 : <http://cafe.naver.com/fusefactory>
 - Github : <https://github.com/fusetools/fuse-samples>
- Fusetools Korea 플래티넘 파트너

