Performance on iOS and watchOS

Strategies and tools

Session 230

Ben Englert iOS Performance

Introduction

Why should I think about performance?

Introduction

Why should I think about performance?

How should I think about performance?

Introduction

Why should I think about performance?

How should I think about performance?

Specific strategies

Introduction

Why should I think about performance?

How should I think about performance?

Specific strategies

New platform: watchOS 2

왜 > 퍼포먼스를 고려

어떻게

접근방법

watch OS

Performance Is a Feature

퍼포먼스를 가장 간단히 표현

Performance Is a Feature

Performance Is a Feature

Responsiveness delights and engages users

Performance Is a Feature

Responsiveness delights and engages users

Be a good neighbor, especially in Multitasking on iPad

반응성

멀티태스킹

Performance Is a Feature

Responsiveness delights and engages users

Be a good neighbor, especially in Multitasking on iPad

Efficient apps extend battery life

배터리

Performance Is a Feature

Responsiveness delights and engages users

Be a good neighbor, especially in Multitasking on iPad

Efficient apps extend battery life

Supports the whole range of iOS 9 hardware

iOS는 많은 종류의 디바이스를 지원해야 함

Thinking About Performance

Thinking About Performance

Choosing technologies

Thinking About Performance

Choosing technologies

Taking measurements

Thinking About Performance

Choosing technologies

Taking measurements

Setting goals

Thinking About Performance

Choosing technologies

Taking measurements

Setting goals

Performance workflow

기술 선택

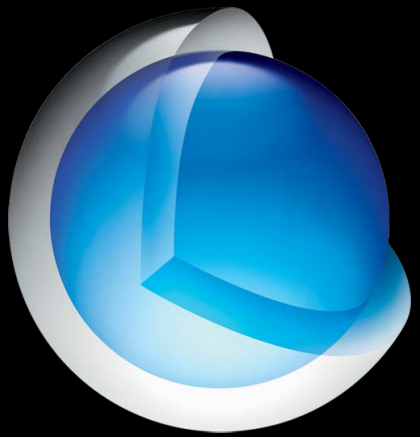
측정 → 목표 세우기

Use the Right Tool for the Job

Proactively architect your app for great performance

성능이 잘 나오게 하기 위해 사전에 설계

기능 선택이 중요하다.

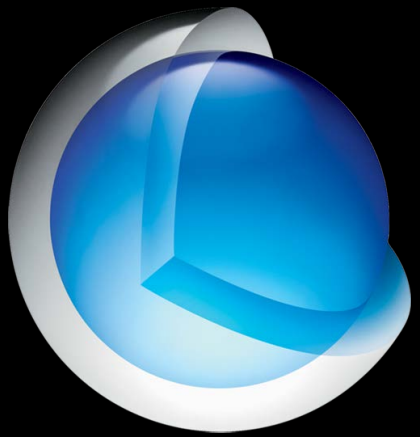


Use the Right Tool for the Job

Proactively architect your app for great performance

Know the technologies

기술들을 알아내서



Use the Right Tool for the Job

Proactively architect your app for great performance

Know the technologies

Pick the best ones for your app

나에게 베스트를 선택
3개의 몰라도 괜찮아라면
plist or User default

3000개의 몰라도 괜찮다면
Core data

Use the Right Tool for the Job

Proactively architect your app for great performance

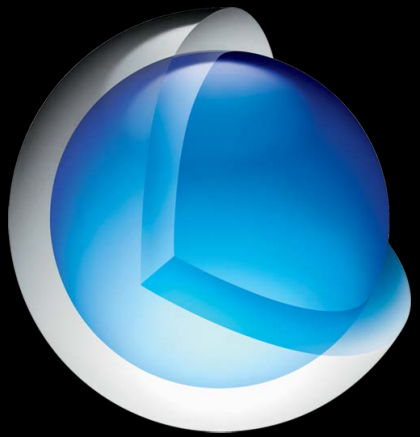
Know the technologies

Pick the best ones for your app

Apple technologies are optimized
(we use them)

애플 제공 기술들은 최적화 고려되어있다

API, framework



Use the Right Tool for the Job

Proactively architect your app for great performance

Know the technologies

Pick the best ones for your app

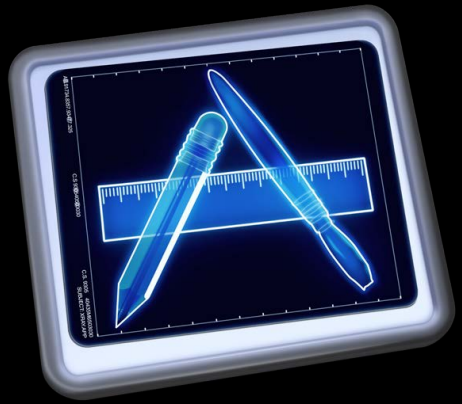
Apple technologies are optimized (we use them)

Benefit from software updates

OS 업데이트 같은것으로 성능 향상된다

Measuring Performance

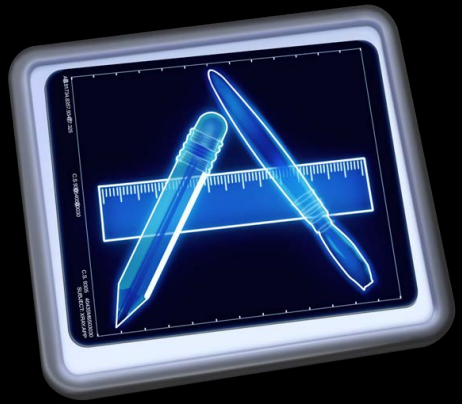
측정 단계로 가보자



Measuring Performance

Animations

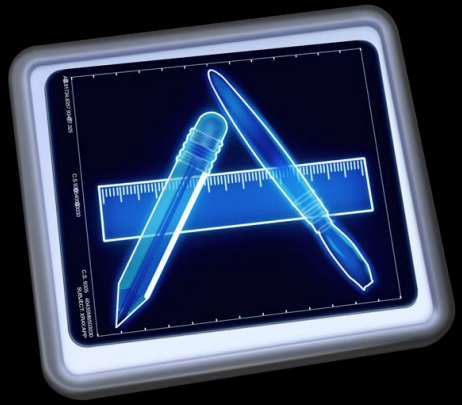
애니메이션은 측정하는건 간단하다.



Measuring Performance

Animations

- Instruments: Core Animation



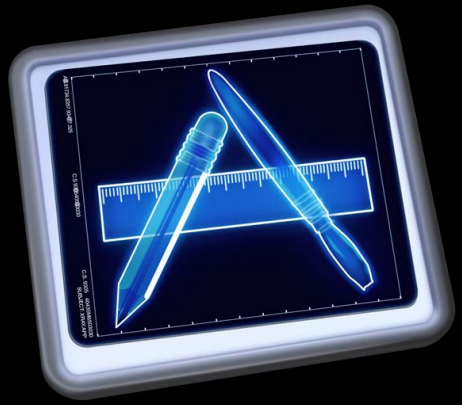
Measuring Performance

Animations

- Instruments: Core Animation

Responsiveness

입력에 대한 반응 측정



Measuring Performance

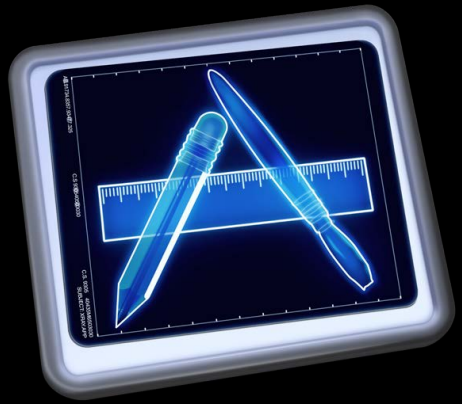
Animations

- Instruments: Core Animation

Responsiveness

- Code instrumentation

약간 떨어지는 기술 같지만
직접 코드로 측정



Measuring Performance

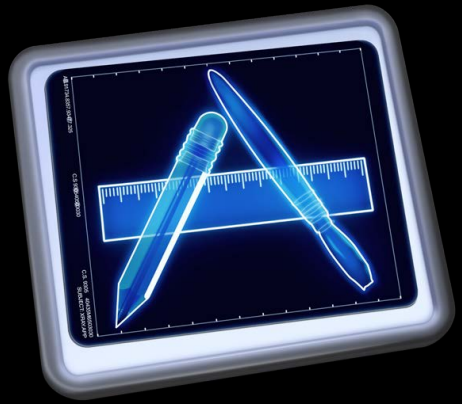
Animations

- Instruments: Core Animation

Responsiveness

- Code instrumentation
- Instruments: System Trace

선생가 만들어 보장은 인터랙션 - System Trace



Measuring Performance

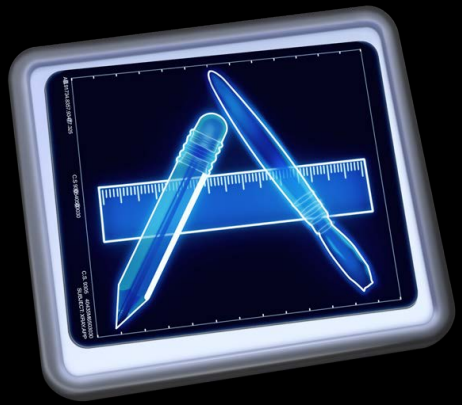
Animations

- Instruments: Core Animation

Responsiveness

- Code instrumentation
- Instruments: System Trace

Memory



Measuring Performance

Animations

- Instruments: Core Animation

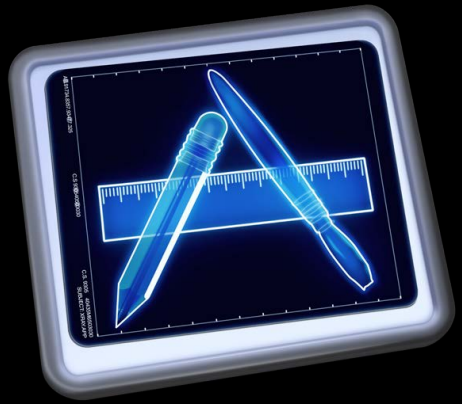
Responsiveness

- Code instrumentation
- Instruments: System Trace

Memory

- Xcode debugger

간단하지만 강력한 앱은 Xcode debugger



Measuring Performance

Animations

- Instruments: Core Animation

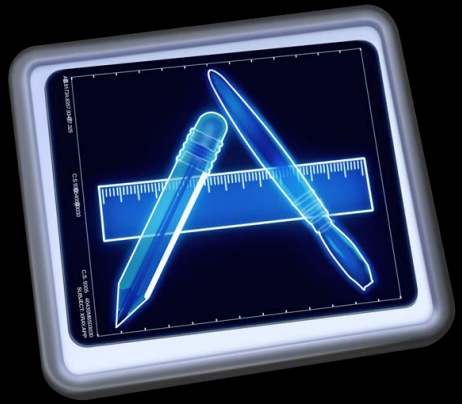
Responsiveness

- Code instrumentation
- Instruments: System Trace

Memory

- Xcode debugger
- Instruments: Allocations

Instrument, 핸드폰, leak 측정



Measuring Performance

Animations

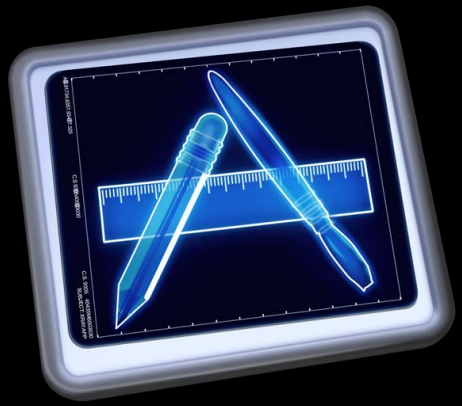
- Instruments: Core Animation

Responsiveness

- Code instrumentation
- Instruments: System Trace

Memory

- Xcode debugger
- Instruments: Allocations
- Instruments: Leaks



Code Instrumentation

Measuring responsiveness

```
@IBAction func showImageTapped(sender: UIButton) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Code 측정 예시

- 얼마나 걸리는지 알고 싶다면?

Code Instrumentation

Collect start and end times

```
@IBAction func showImageTapped(sender: UIButton) {
    let startTime = CFAbsoluteTimeGetCurrent()
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
    let endTime = CFAbsoluteTimeGetCurrent()
}
```

CFAbsoluteTimeGetCurrent()
사용해서 시간 측정

Code Instrumentation

Convert to appropriate units

```
@IBAction func showImageTapped(sender: UIButton) {
    let startTime = CFAbsoluteTimeGetCurrent()
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
    let endTime = CFAbsoluteTimeGetCurrent()
    let totalTime = (endTime - startTime) * 1000
    print("showImageTappedTimed took \(totalTime) milliseconds")
}
```

밀리세컨드 단위로
초 단위로 변환

Code Instrumentation

Don’t ship your instrumentation

```
    @IBAction func showImageTapped(sender: UIButton) {
#if MEASURE_PERFORMANCE
        let startTime = CFAbsoluteTimeGetCurrent()
#endif
        let myData = NSData(contentsOfFile: self.path)!
        let myImage = self.watermarkedImageFromData(myData)
        self.imageView.image = myImage
#if MEASURE_PERFORMANCE
        let endTime = CFAbsoluteTimeGetCurrent()
        let totalTime = (endTime - startTime) * 1000
        print("showImageTappedTimed took \(totalTime) milliseconds”)
#endif
    }
```

측정코드는 스토어에 올리지 말자

Code Instrumentation

Measuring responsiveness

어디에서 측정 해야 할까

Code Instrumentation

Measuring responsiveness

Taps and button presses

Code Instrumentation

Measuring responsiveness

Taps and button presses

- IBAction

Code Instrumentation

Measuring responsiveness

Taps and button presses

- IBAction
- touchesEnded

Code Instrumentation

Measuring responsiveness

Taps and button presses

- IBAction
- touchesEnded
- UIGestureRecognizer target

Code Instrumentation

Measuring responsiveness

Taps and button presses

- IBAction
- touchesEnded
- UIGestureRecognizer target

Tabs and modal views

Code Instrumentation

Measuring responsiveness

Taps and button presses

- IBAction
- touchesEnded
- UIGestureRecognizer target

Tabs and modal views

- viewWillAppear and viewDidAppear

Setting Performance Goals

목표는 어떻게 세울것인가

Setting Performance Goals

60fps scrolling and animations

애니메이션은 60프레임을 목표로 한다. (60프레임 기기는)

Setting Performance Goals

60fps scrolling and animations

Setting Performance Goals

반응성에 대해서 이야기 하는게 가장 많은듯

Setting Performance Goals

Respond to user actions in 100ms

100 밀리 이하여야 자연스럽다

Setting Performance Goals

Respond to user actions in 100ms

…on older devices!

오래된 기기에서 …

Setting Performance Goals

Respond to user actions in 100ms

…on older devices!

Performance Workflow

퍼포먼스 향상을 위해 코드 수정하고 한다.

Performance Workflow

Don't guess 추측 금물

프로파일링은 툴을 이용한 증거로 수행

Performance Workflow

Don’t guess

Avoid premature optimization 조급하게 고치지 말라

더 복잡해진다

Performance Workflow

Don’t guess

Avoid premature optimization

Make one change at a time

한번에 하나씩 고쳐라
적당에 너무 의존 말고

Performance Workflow

Don’t guess

Avoid premature optimization

Make one change at a time

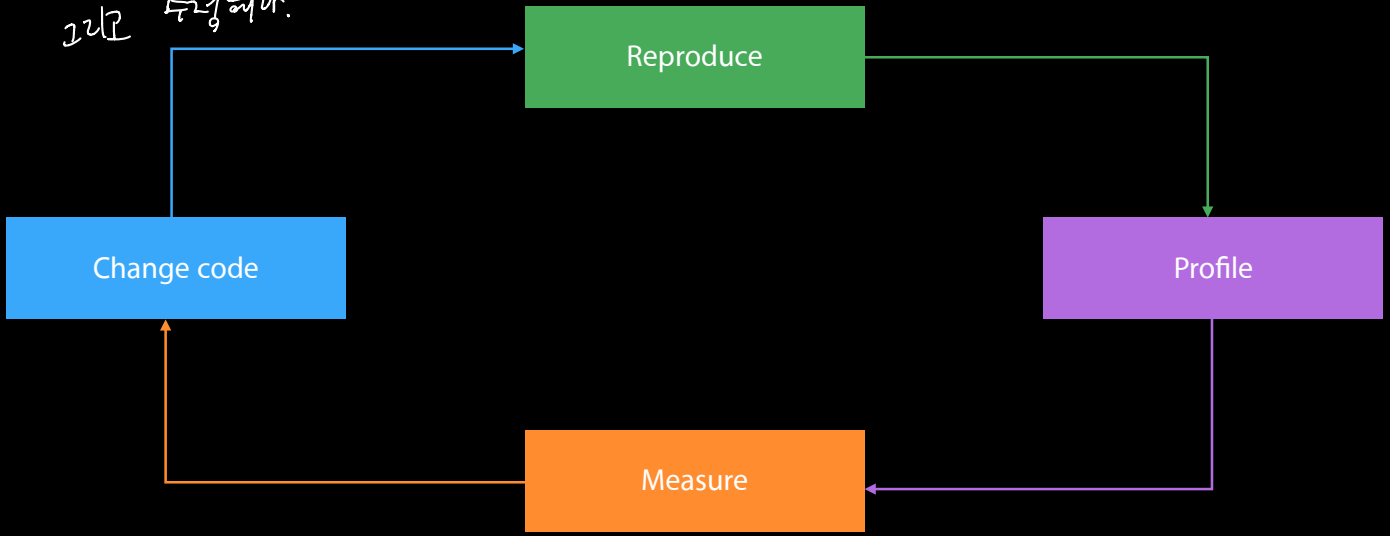
Just like ordinary debugging

마법은 없다

일반적인 디버깅과 같다.

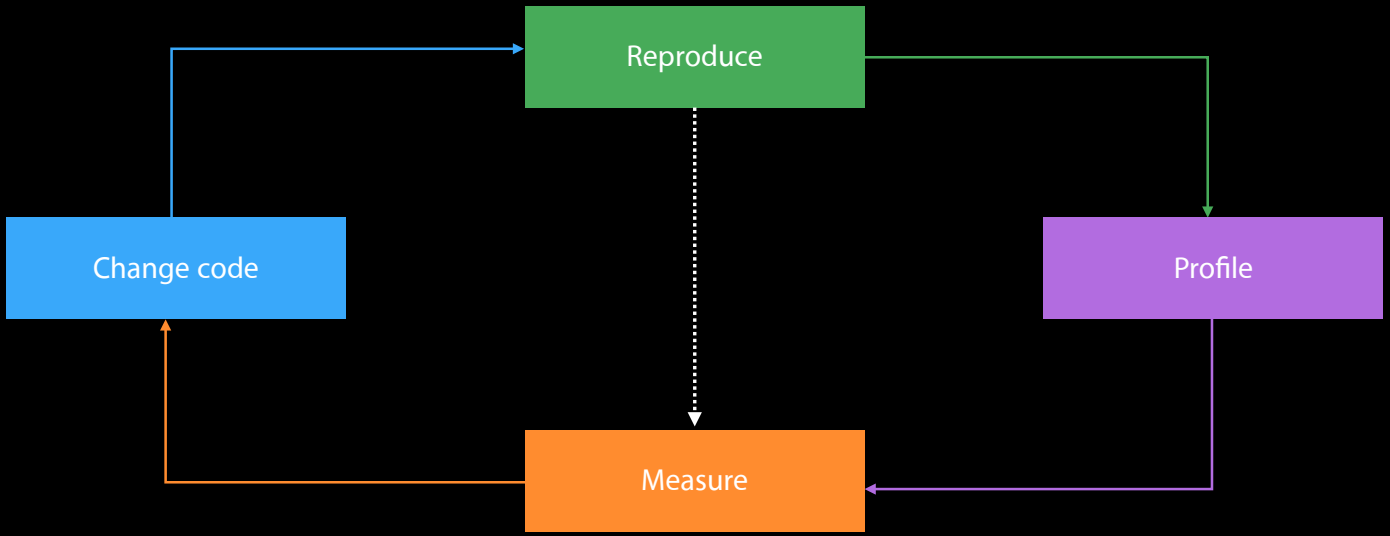
Performance Workflow

가장 먼저 이슈 재현
tool을 사용하고 우선순위 수정
그리고 두려워마.



Performance Workflow

한번의 수정으로 해결이 안되면
복합적인 문제이므로 반복



Profiling vs. Measuring

Profiling: Understanding overall app activity

- Xcode debugger
- Instruments: Time Profiler

Measuring: Instrumenting a specific action

- CFAbsoluteTimeGetCurrent
- Instruments: System Trace

Responsiveness 반응성

Reacting to user input

메인스레드 이야기 없이 컨트롤이 불가능하다

Main Thread Consumes User Input

모든 input을 소모하는곳

Touches and scrolling

Orientation

Multitasking resizes

Main Thread Consumes User Input

Touches and scrolling

Orientation

Multitasking resizes

A responsive main thread makes your app feel great

Main Thread Consumes User Input

Touches and scrolling

Orientation

Multitasking resizes

A responsive main thread makes your app feel great

Busy main thread makes your app appear frozen

Avoid Using the Main Thread for…

피할것

예를들면 (외부 리소스에 의존하거나,
긴 문자열 파싱, 네트워크, 이미지 핸들, ...)

CPU-intensive work CPU 많이쓰는것들.

Tasks that depend on external resources

Avoid Using the Main Thread for…

CPU-intensive work

Tasks that depend on external resources

What's a Blocking Call?

코드를 멈추는

Any code path that ends up making a syscall

What's a Blocking Call?

Any code path that ends up making a syscall

Accessing resources not currently in memory

What's a Blocking Call?

Any code path that ends up making a syscall

Accessing resources not currently in memory

- Disk I/O

What's a Blocking Call?

Any code path that ends up making a syscall

Accessing resources not currently in memory

- Disk I/O
- Network access

What's a Blocking Call?

Any code path that ends up making a syscall

Accessing resources not currently in memory

- Disk I/O
- Network access

Waiting for work to complete on another thread

What's a Blocking Call?

"synchronous" is a synonym for blocking

동기 == blocking 동의어

```
NSURLConnection.sendSynchronousRequest(…
```

What's a Blocking Call?

"synchronous" is a synonym for blocking

```
NSURLConnection.sendSynchronousRequest(…
```

Strategies for Avoiding Blocking Calls

In many cases, there is an existing asynchronous API you can switch to

```
NSURLConnection.sendSynchronousRequest(…
```

Strategies for Avoiding Blocking Calls

In many cases, there is an existing asynchronous API you can switch to

```
NSURLConnection.sendAsynchronousRequest(…
```

비동기를 사용

하지만 구조를 많이 바꿔야 함

Strategies for Avoiding Blocking Calls

In many cases, there is an existing asynchronous API you can switch to

Some restructuring required

```
NSURLConnection.sendAsynchronousRequest(…
```

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Use Grand Central Dispatch (GCD)

다른방법으로 GCD라는 애플의 기술을 사용

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Use Grand Central Dispatch (GCD)

GCD manages a global thread pool

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Use Grand Central Dispatch (GCD)

GCD manages a global thread pool

Express tasks as closures (a.k.a. blocks)

작업을 블록으로

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Use Grand Central Dispatch (GCD)

GCD manages a global thread pool

Express tasks as closures (a.k.a. blocks)

Closures run on an arbitrary thread

클로저는 임의의 스레드에서 실행

Strategies for Avoiding Blocking Calls

In other cases, there isn't an async equivalent

Use Grand Central Dispatch (GCD)

GCD manages a global thread pool

Express tasks as closures (a.k.a. blocks)

Closures run on an arbitrary thread

Ensure operations performed are thread-safe!

작업이 스레드 세이프 한지 확인

Thread Safety

Some objects are restricted to the main thread

메인스레드만 가능한 객체

Thread Safety

Some objects are restricted to the main thread

Some objects, once created, can be used from any thread

어떤 쓰레드던 가능한 객체

Thread Safety

Some objects are restricted to the main thread

Some objects, once created, can be used from any thread

- Protection is not built-in

Thread Safety

Some objects are restricted to the main thread

Some objects, once created, can be used from any thread

- Protection is not built-in
- Implement protection using serial GCD queues

시리얼 큐를 사용해 스레드 세이프 구현 (선택)

Thread Safety

Some objects are restricted to the main thread

Some objects, once created, can be used from any thread

- Protection is not built-in
- Implement protection using serial GCD queues

Read the headers 헤더에 사용법이 있다.

Strategies for Avoiding Blocking Calls

```
@IBAction func showImageTapped(sender: UIButton) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Strategies for Avoiding Blocking Calls

```
@IBAction func showImageTapped(sender: UIButton) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Load file data

Strategies for Avoiding Blocking Calls

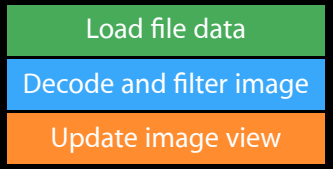
```
@IBAction func showImageTapped(sender: UIButton) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Load file data

Decode and filter image

Strategies for Avoiding Blocking Calls

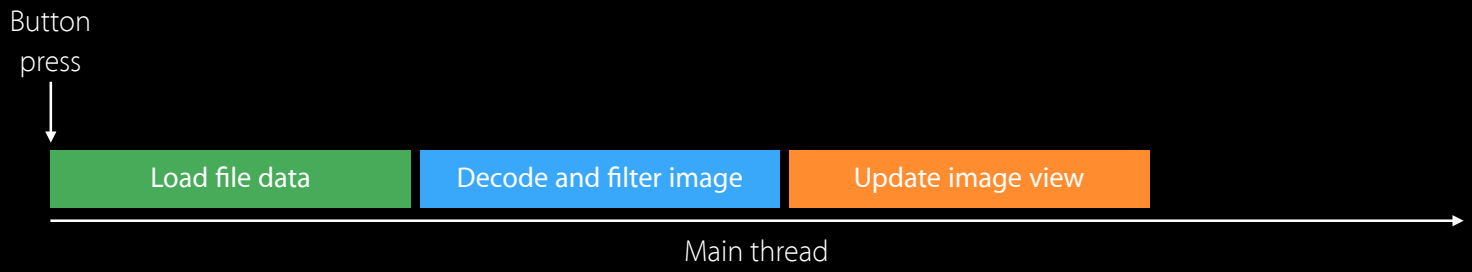
```
@IBAction func showImageTapped(sender: UIButton) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```



코드 변환 적용

Grand Central Dispatch

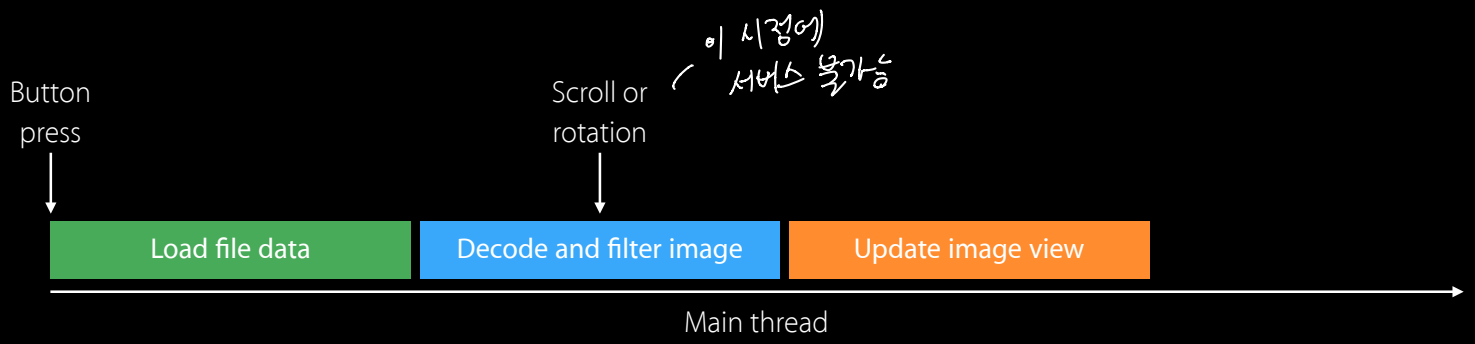
Current implementation



서버 없이 동시에

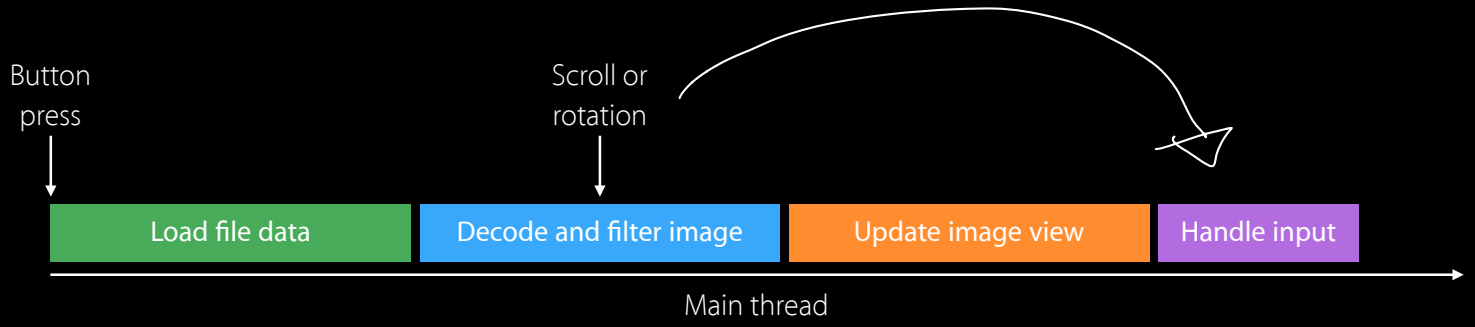
Grand Central Dispatch

Current implementation



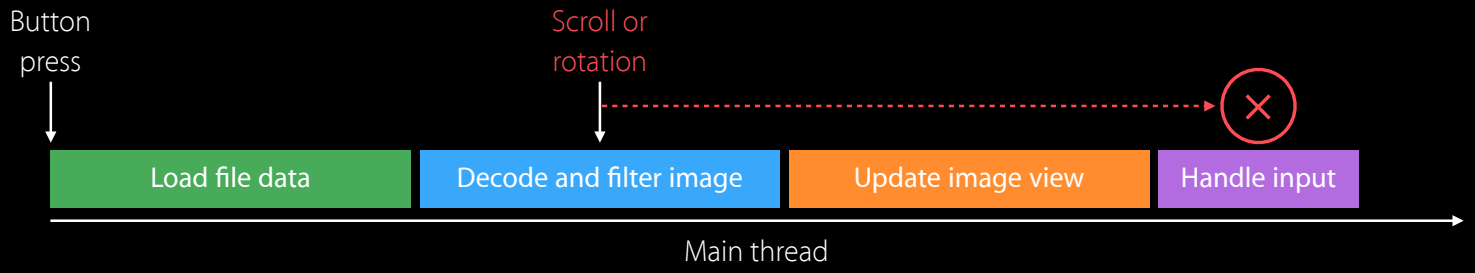
Grand Central Dispatch

Current implementation



Grand Central Dispatch

User input delayed



GCD 사용

Strategies for Avoiding Blocking Calls

GCD 사용법. 가장 먼저 어떤 큐를 얻는지 결정

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Strategies for Avoiding Blocking Calls

글로벌큐 사용

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Strategies for Avoiding Blocking Calls

Quality of Service (QoS)

QoS3 우선순위 3번

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

Strategies for Avoiding Blocking Calls

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

비동기 동작 완료 코드

Strategies for Avoiding Blocking Calls

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)
    self.imageView.image = myImage
}
```

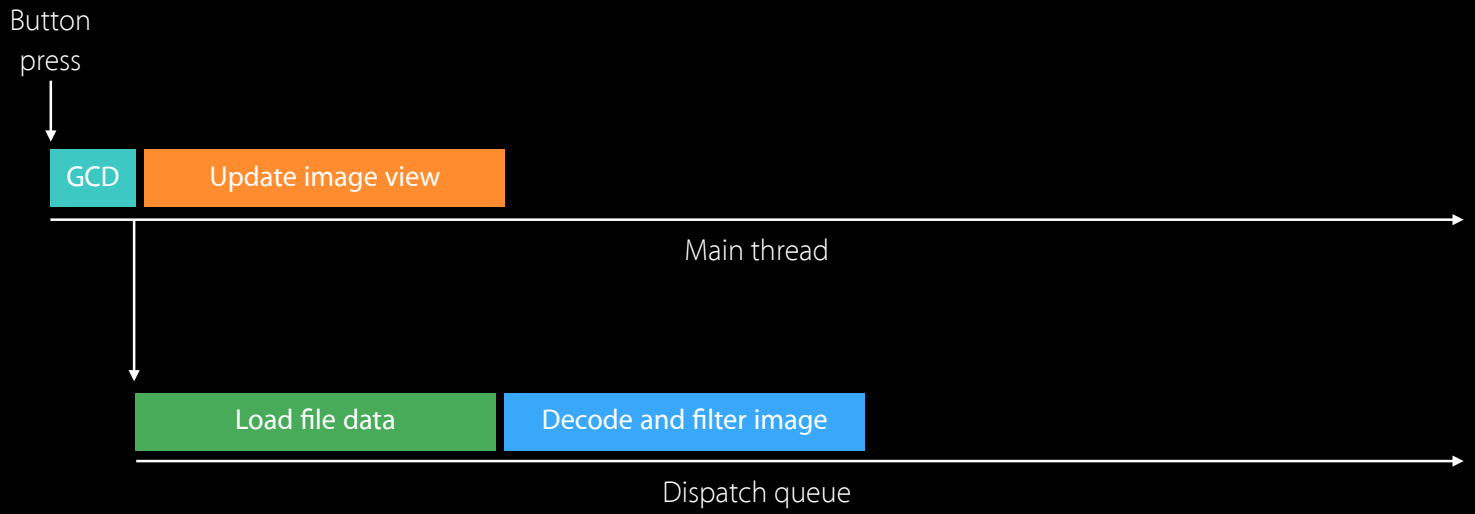
Strategies for Avoiding Blocking Calls

```
var myImage: UIImage? = nil
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    myImage = self.watermarkedImageFromData(myData)
}

self.imageView.image = myImage
```

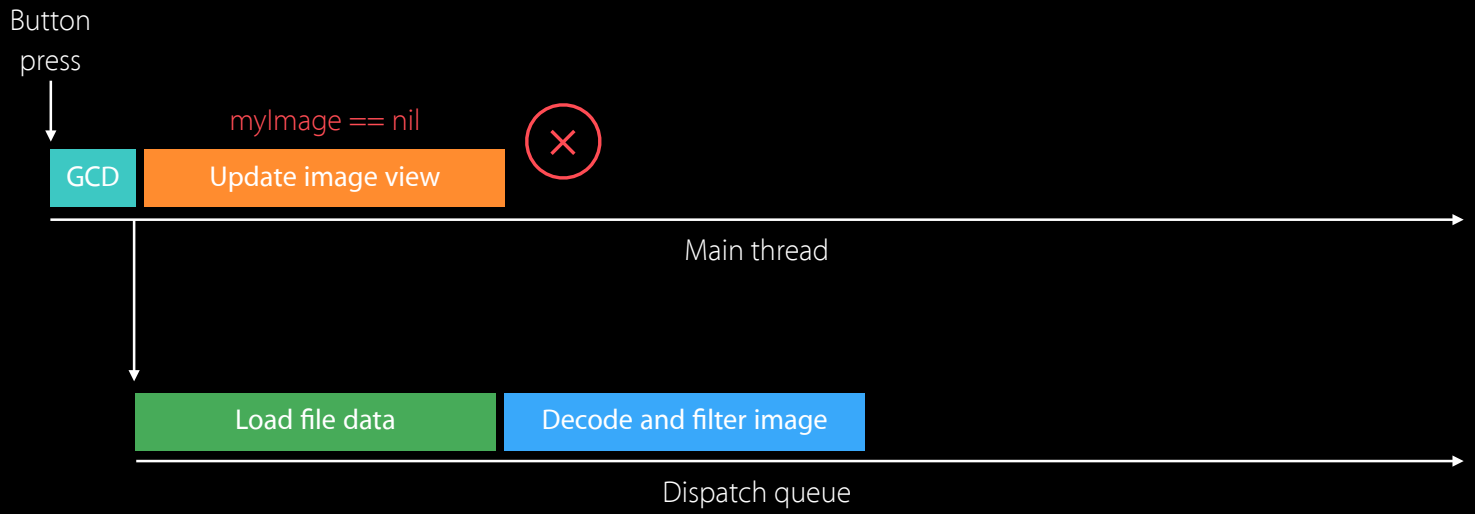
Strategies for Avoiding Blocking Calls

```
var myImage: UIImage? = nil
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    myImage = self.watermarkedImageFromData(myData)
}

self.imageView.image = myImage
```

Grand Central Dispatch



Grand Central Dispatch



Strategies for Avoiding Blocking Calls

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)

    dispatch_async(dispatch_get_main_queue()) {
        self.imageView.image = myImage
    }
}
```

Strategies for Avoiding Blocking Calls

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)

    dispatch_async(dispatch_get_main_queue()) {
        self.imageView.image = myImage
    }
}
```

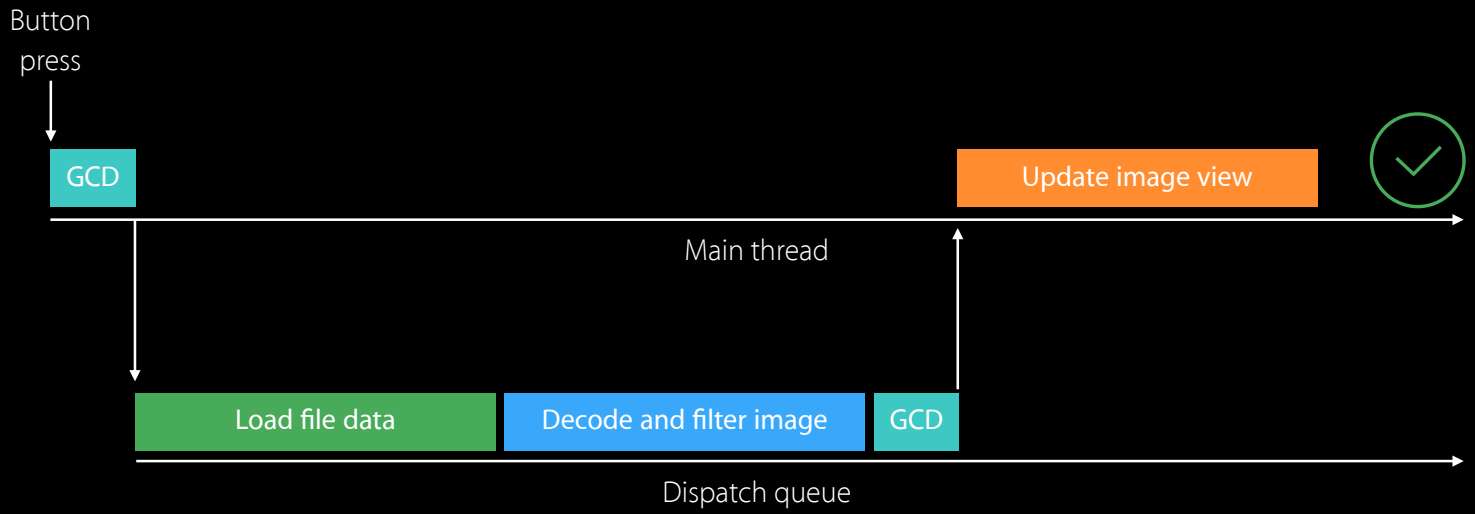
메인큐 : 메인스레드 의미

Strategies for Avoiding Blocking Calls

```
dispatch_async(dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)) {
    let myData = NSData(contentsOfFile: self.path)!
    let myImage = self.watermarkedImageFromData(myData)

    dispatch_async(dispatch_get_main_queue()) {
        self.imageView.image = myImage
    }
}
```

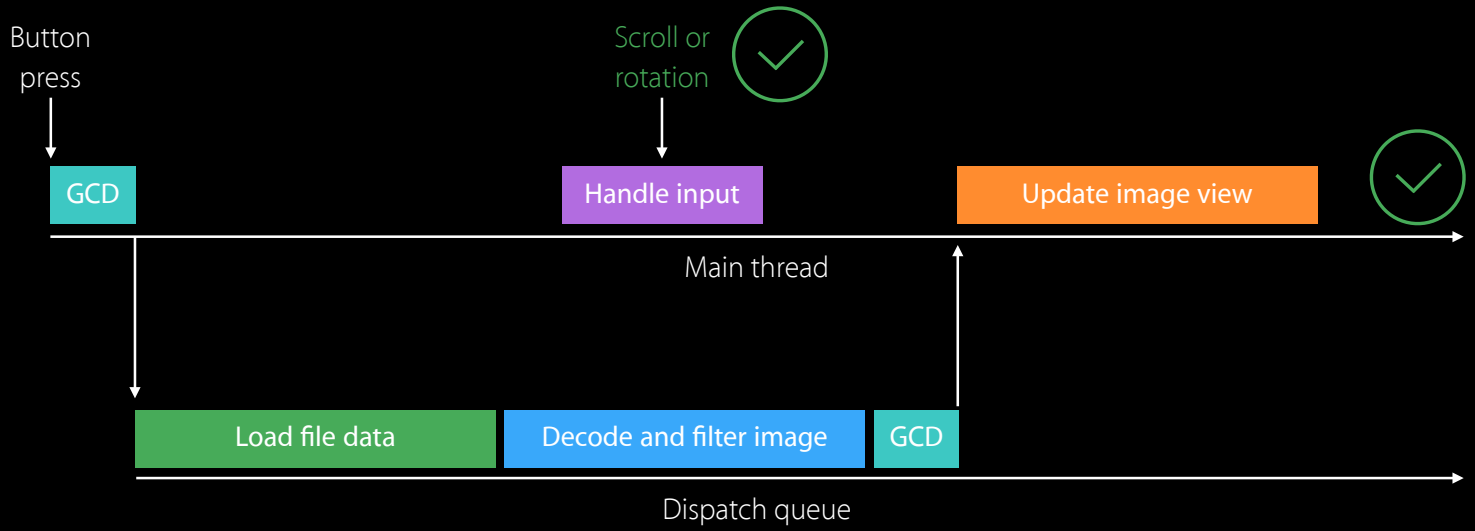
Grand Central Dispatch

Timely and thread-safe object access



Grand Central Dispatch

Timely handling of user input



Common Blocking Calls

동기 코드들

Common Blocking Calls

Networking: NSURLConnection and friends

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session 백그라운드 세션도 알아둘것

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

- contentsOfFile:

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

- contentsOfFile:
- contentsOfURL:

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

- contentsOfFile:
- contentsOfURL:

Core Data

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

- contentsOfFile:
- contentsOfURL:

Core Data

- Move some Core Data work to different concurrency modes

코어데이터는 동시성 모드가 있다.

bulk 처리?

Common Blocking Calls

Networking: NSURLConnection and friends

- Use asynchronous API
- NSURLSession background session

Foundation initializers

- contentsOfFile:
- contentsOfURL:

Core Data

- Move some Core Data work to different concurrency modes

What's New in Core Data	Mission	Thursday 2:30PM

Strategies for Avoiding Blocking Calls

Switch to asynchronous API

Strategies for Avoiding Blocking Calls

Switch to asynchronous API

Use GCD

Strategies for Avoiding Blocking Calls

Switch to asynchronous API

Use GCD

Memory

Memory

Multitasking requires memory tuning

멀티 태스킹은 메모리 튜닝 필요

Memory

Multitasking requires memory tuning

watchOS considerations Watch는 리소스 고려 필요

Memory

Multitasking requires memory tuning

watchOS considerations

Older hardware

Memory

Multitasking requires memory tuning

watchOS considerations

Older hardware

Extensions

iOS Memory System

Never enough to go around

iOS Memory System

Never enough to go around

Suspended apps are not persisted

iOS Memory System

Never enough to go around

Suspended apps are not persisted

They are evicted without storing

iOS Memory System

Never enough to go around

Suspended apps are not persisted

They are evicted without storing

백그라운드 앱은 언제 꺼질지 보장 못함

Memory

Memory is time

Reclaiming memory takes time

Memory

Memory is time

Reclaiming memory takes time

Sudden high-memory demand impacts responsiveness

Memory

Memory is time

Reclaiming memory takes time 메모리 확보는 시간이걸림

Sudden high-memory demand impacts responsiveness 갑자기 큰 메모리 사용은 반응성에 영향

Preserves state in the background

런칭대신 resume이 바쁨

Rationalize Your App's Memory Footprint

Rationalize Your App's Memory Footprint

Resources

Rationalize Your App's Memory Footprint

Resources

- Strings

Rationalize Your App's Memory Footprint

Resources

- Strings
- Images

Rationalize Your App's Memory Footprint

Resources

- Strings
- Images
- Core Data managed objects

Rationalize Your App's Memory Footprint

Resources

- Strings
- Images
- Core Data managed objects

Create a mental model of accessed resources

Rationalize Your App's Memory Footprint

Resources

- Strings
- Images
- Core Data managed objects

Create a mental model of accessed resources

Check your work using Xcode debugger

Rationalize Your App's Memory Footprint

Resources

- Strings
- Images
- Core Data managed objects

Create a mental model of accessed resources

Check your work using Xcode debugger

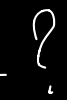
Instruments: Allocations and Leaks

Rationalize Your App's Memory Footprint

Resources

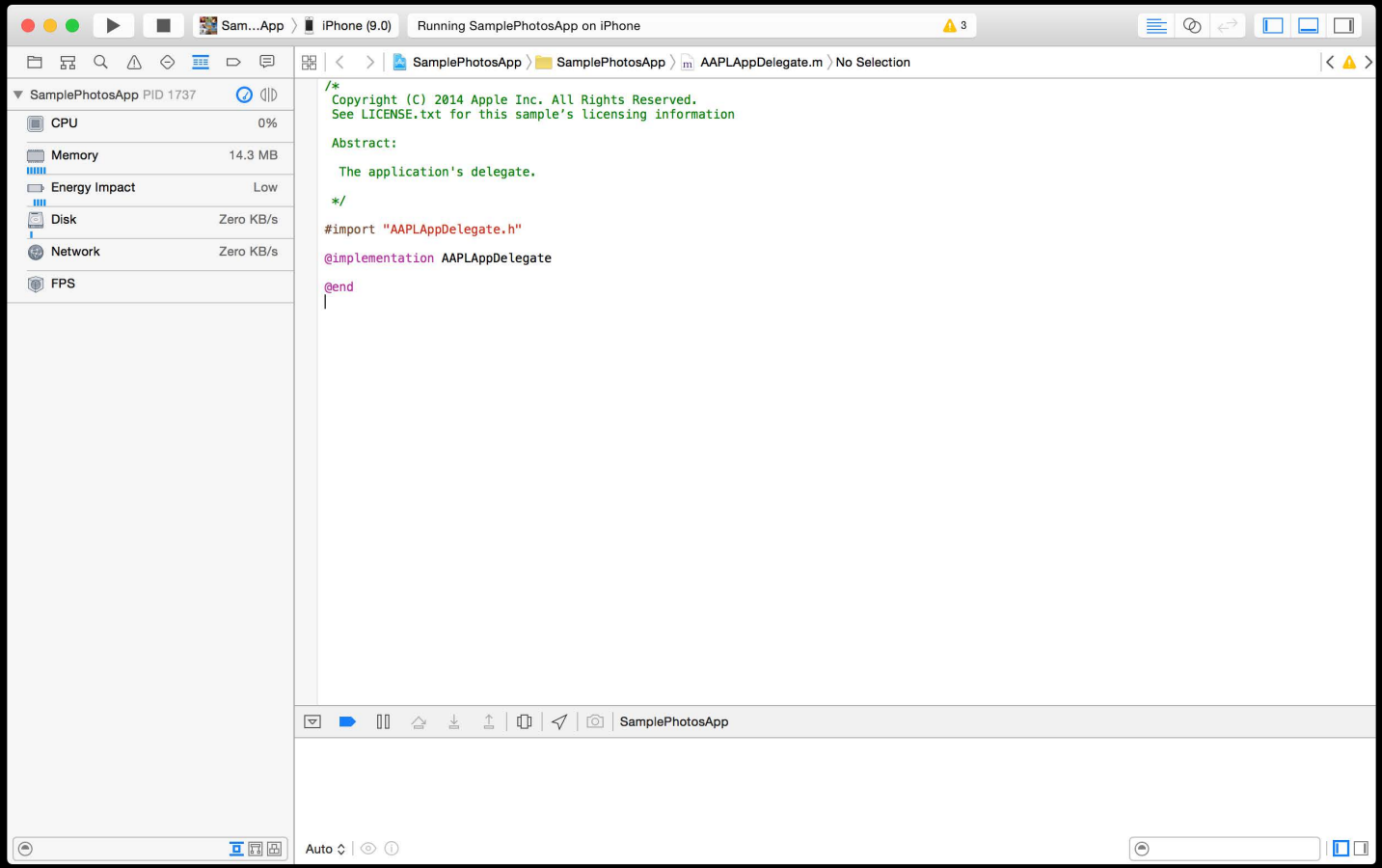
- Strings
- Images
- Core Data managed objects

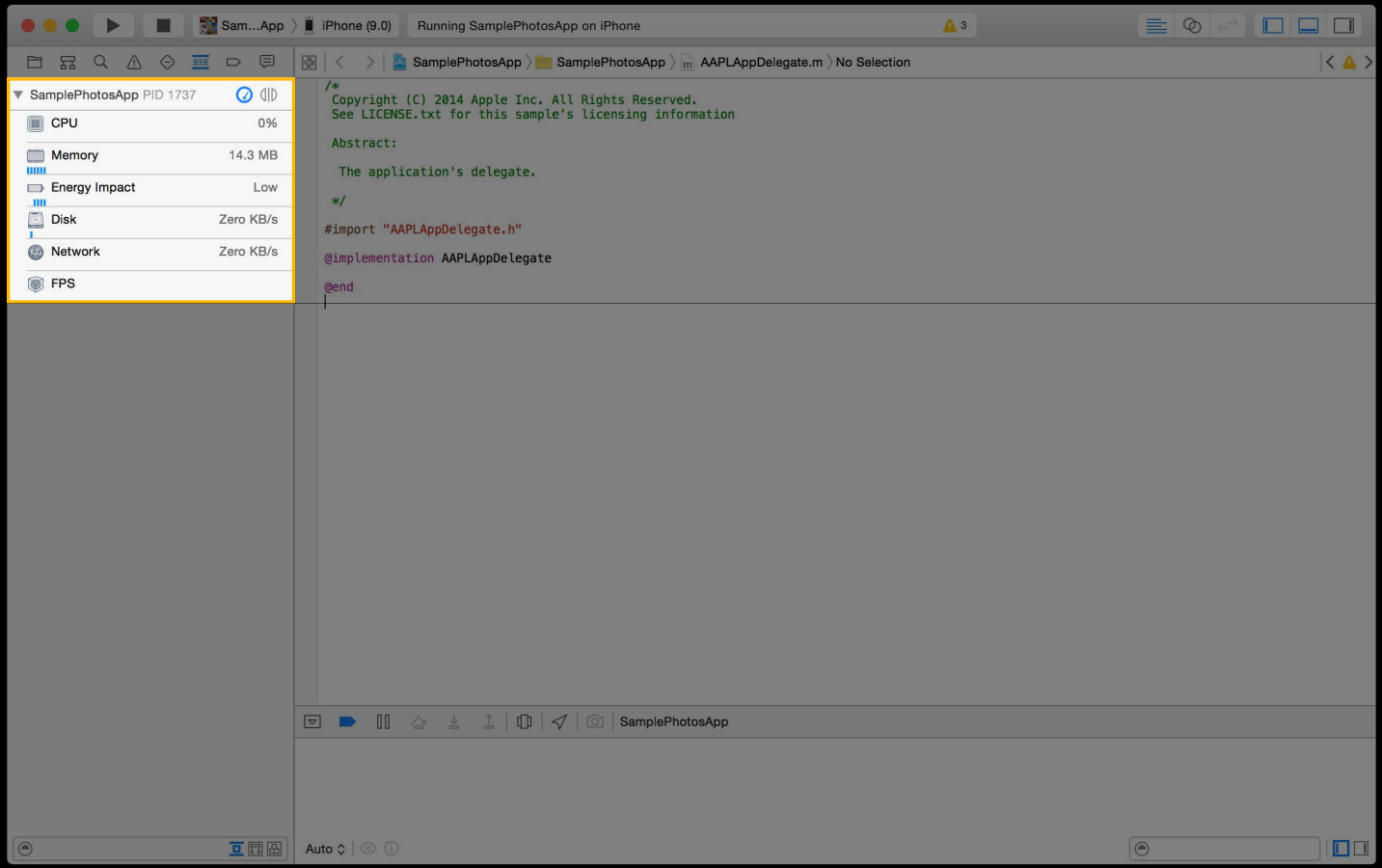
Create a mental model of accessed resources

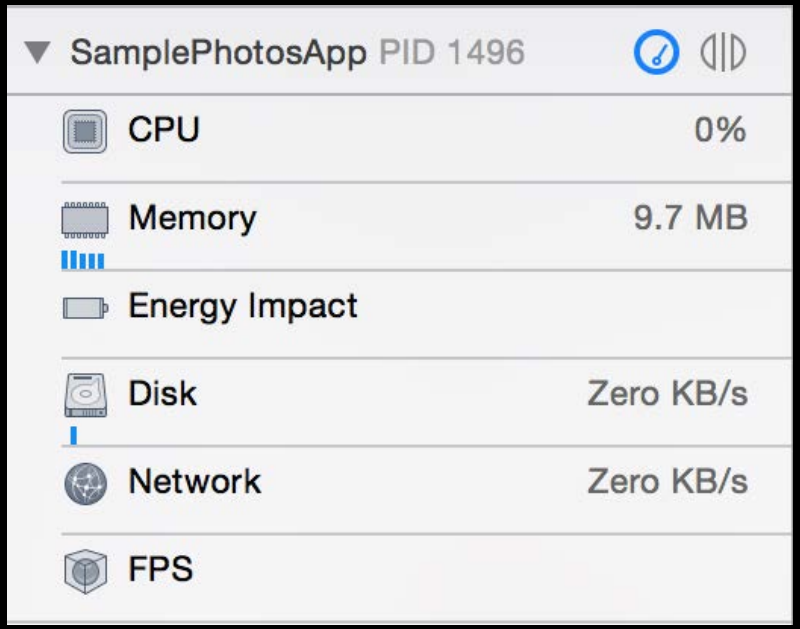


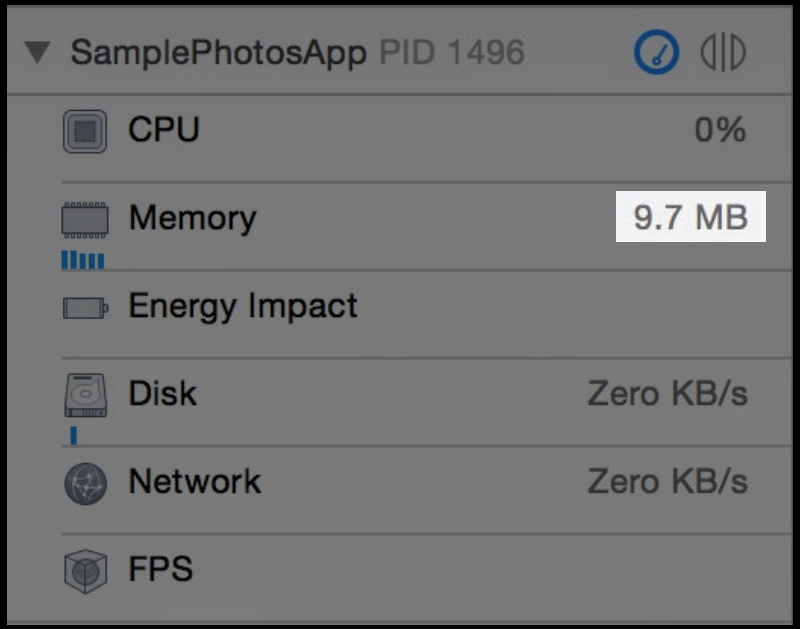
Check your work using Xcode debugger

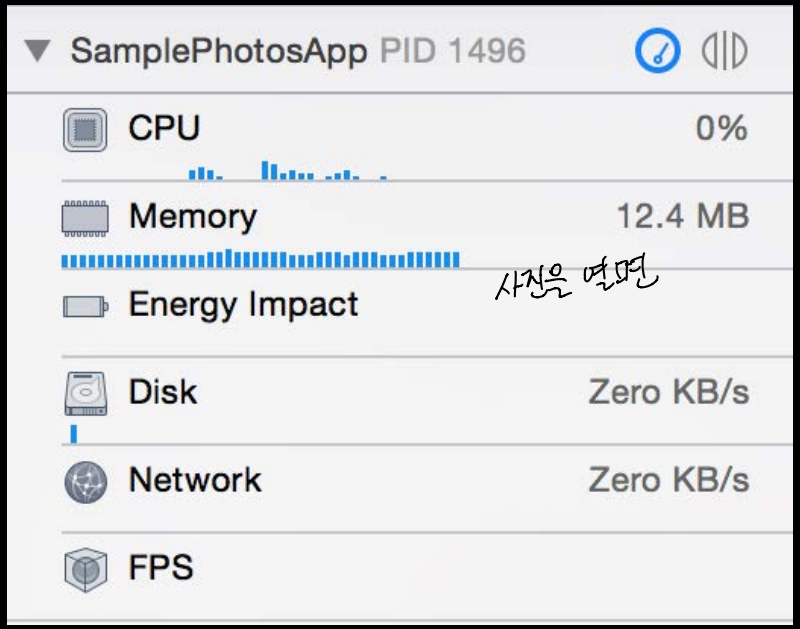
Instruments: Allocations and Leaks

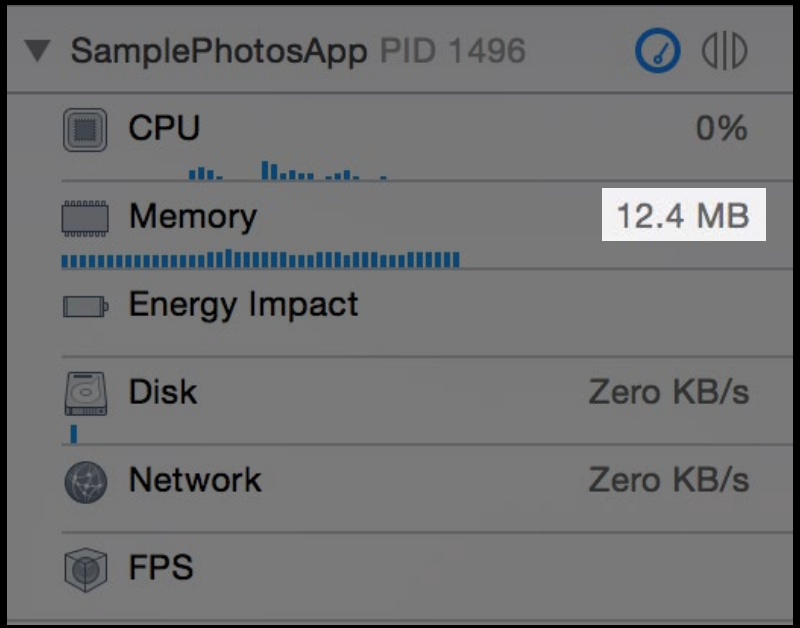

Sam…App
iPhone (9.0)
Running SamplePhotosApp on iPhone
3
SamplePhotosApp PID 1737
CPU 0%
Memory 14.3 MB
Energy Impact Low
Disk Zero KB/s
Network Zero KB/s
FPS
SamplePhotosApp
SamplePhotosApp
AAPLAppDelegate.m
No Selection
/*
Copyright (C) 2014 Apple Inc. All Rights Reserved.
See LICENSE.txt for this sample's licensing information
Abstract:
The application's delegate.
*/
#import "AAPLAppDelegate.h"
@implementation AAPLAppDelegate
@end
SamplePhotosApp
Auto

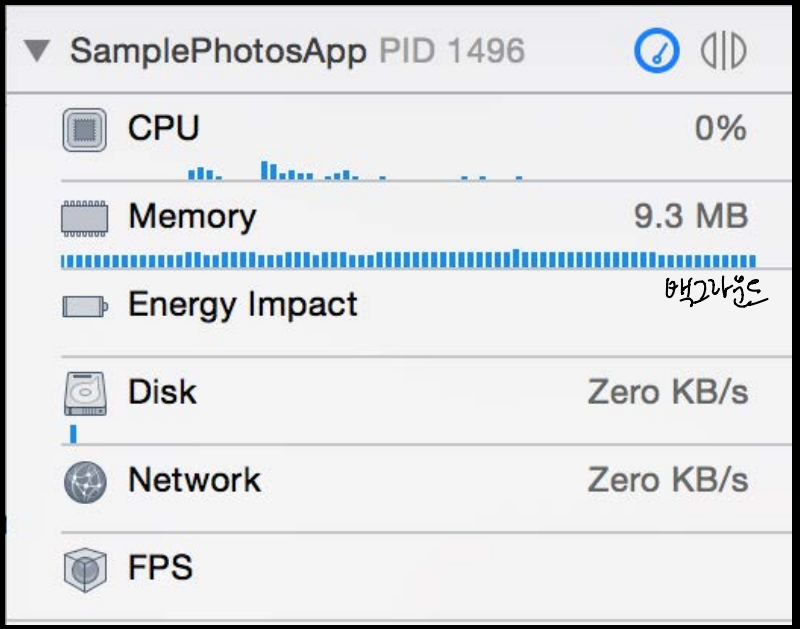

Sam...App
iPhone (9.0)
Running SamplePhotosApp on iPhone
3
SamplePhotosApp PID 1737
CPU 0%
Memory 14.3 MB
Energy Impact Low
Disk Zero KB/s
Network Zero KB/s
FPS
SamplePhotosApp
SamplePhotosApp
AAPLAppDelegate.m
No Selection
/*
Copyright (C) 2014 Apple Inc. All Rights Reserved.
See LICENSE.txt for this sample's licensing information
Abstract:
The application's delegate.
*/
#import "AAPLAppDelegate.h"
@implementation AAPLAppDelegate
@end
SamplePhotosApp
Auto

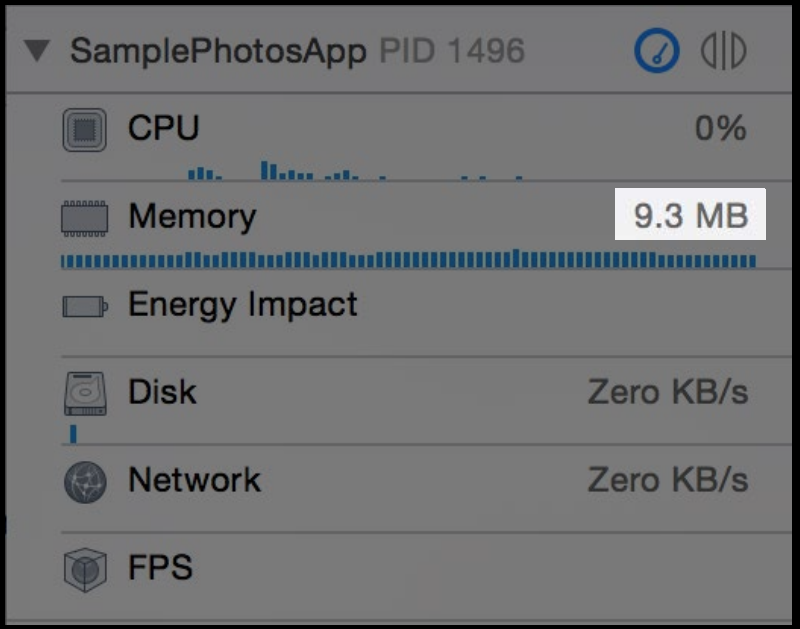

SamplePhotosApp PID 1496
CPU
0%
Memory
9.7 MB
Energy Impact
Disk
Zero KB/s
Network
Zero KB/s
FPS


SamplePhotosApp PID 1496
CPU
0%
Memory
9.7 MB
Energy Impact
Disk
Zero KB/s
Network
Zero KB/s
FPS


SamplePhotosApp PID 1496
CPU
0%
Memory
12.4 MB
Energy Impact
사진은 연못
Disk
Zero KB/s
Network
Zero KB/s
FPS


SamplePhotosApp PID 1496
CPU
0%
Memory
12.4 MB
Energy Impact
Disk
Zero KB/s
Network
Zero KB/s
FPS


SamplePhotosApp PID 1496
CPU
0%
Memory
9.3 MB
Energy Impact
백그라운드
Disk
Zero KB/s
Network
Zero KB/s
FPS


SamplePhotosApp PID 1496
CPU
0%
Memory
9.3 MB
Energy Impact
Disk
Zero KB/s
Network
Zero KB/s
FPS

Application Lifecycle

Use NSCache

Application Lifecycle

Use NSCache

Listen for notifications

Application Lifecycle

Use NSCache

Listen for notifications

- UIApplicationDidEnterBackgroundNotification

Application Lifecycle

Use NSCache

Listen for notifications

- UIApplicationDidEnterBackgroundNotification
- UIApplicationDidReceiveMemoryWarningNotification

Application Lifecycle

Responding to changes

```
init() {
  NSNotificationCenter.defaultCenter()
    .addObserverForName(UIApplicationDidReceiveMemoryWarningNotification,
       object: self,
       queue: NSOperationQueue.mainQueue())
             { [unowned self] (NSNotification notification) -> Void in
                 self.purgeCaches() // custom cache purging behavior
             }
}

deinit {
  NSNotificationCenter.defaultCenter().removeObserver(self)
}
```

Application Lifecycle

Responding to changes

```
init() {
  NSNotificationCenter.defaultCenter()
    .addObserverForName(UIApplicationDidReceiveMemoryWarningNotification,
       object: self,
       queue: NSOperationQueue.mainQueue())
            { [unowned self] (NSNotification notification) -> Void in
                self.purgeCaches() // custom cache purging behavior
            }
}

deinit {
  NSNotificationCenter.defaultCenter().removeObserver(self)
}
```

Application Lifecycle

Responding to changes

```
init() {
  NSNotificationCenter.defaultCenter()
    .addObserverForName(UIApplicationDidReceiveMemoryWarningNotification,
       object: self,
       queue: NSOperationQueue.mainQueue())
             { [unowned self] (NSNotification notification) -> Void in
                 self.purgeCaches() // custom cache purging behavior
             }
}

deinit {
  NSNotificationCenter.defaultCenter().removeObserver(self)
}
```

필수?

Memory Strategies

Covered in detail

Memory Strategies

Covered in detail

Optimizing Your App for Multitasking on iPad in iOS 9	Presidio	Wednesday 3:30PM

Memory Strategies

Covered in detail

Resource types and access patterns

Memory Strategies

Covered in detail

Resource types and access patterns

Responding to system memory state while running

Optimizing Your App for Multitasking on iPad in iOS 9 Presidio Wednesday 3:30PM

New Platform

Native code on watchOS

NEW

New Platform

Native code on watchOS

NEW

Designing for Apple Watch Presidio Wednesday 4:30PM

New Platform

Native code on watchOS

Reuse what makes sense

Designing for Apple Watch Presidio Wednesday 4:30PM

New Platform

Native code on watchOS

Reuse what makes sense

- Your existing code

New Platform

Native code on watchOS

NEW

Reuse what makes sense

- Your existing code
- Familiar APIs and frameworks

Designing for Apple Watch Presidio Wednesday 4:30PM

New Platform

Native code on watchOS

NEW

Reuse what makes sense

- Your existing code
- Familiar APIs and frameworks

Implement new mechanisms

Designing for Apple Watch Presidio Wednesday 4:30PM

watchOS

Quick and simple

Short, simple interactions

watchOS

Quick and simple

Short, simple interactions

Recent and relevant data in Apps, Notifications, Glances

watchOS

Quick and simple

Short, simple interactions

Recent and relevant data in Apps, Notifications, Glances

Launch time is critical

watchOS Performance Strategies

Minimize network traffic and processing

Implementing new server logic

watchOS Performance Strategies

Minimize network traffic and processing

Implementing new server logic

- Send appropriately sized and formatted responses

watchOS Performance Strategies

Minimize network traffic and processing

Implementing new server logic

- Send appropriately sized and formatted responses
- Remove unused keys from JSON or XML blobs

JSON, XML 안쓰는 키 지우기

watchOS Performance Strategies

Minimize network traffic and processing

Implementing new server logic

- Send appropriately sized and formatted responses
- Remove unused keys from JSON or XML blobs
- Send appropriately sized images

watchOS Performance Strategies

Minimize network traffic and processing

Implementing new server logic

- Send appropriately sized and formatted responses
- Remove unused keys from JSON or XML blobs
- Send appropriately sized images
- Send an appropriate number of records (one screen)

watchOS Performance Strategies

Show fresh, relevant information

Keep app context updated

watchOS Performance Strategies

Show fresh, relevant information

Keep app context updated

- Bidirectional shared state

watchOS Performance Strategies

Show fresh, relevant information

Keep app context updated

- Bidirectional shared state
- `WCSession.defaultSession().updateApplicationContext(…)`

watchOS Performance Strategies

Show fresh, relevant information

Keep app context updated

- Bidirectional shared state
- `WCSession.defaultSession().updateApplicationContext(…)`
- Benefit from Background App Refresh

watchOS Performance Strategies

Minimize network traffic and processing

Relying on existing server logic

watchOS Performance Strategies

Minimize network traffic and processing

Relying on existing server logic

- Implement a lightweight service on iPhone

watchOS Performance Strategies

Minimize network traffic and processing

Relying on existing server logic

- Implement a lightweight service on iPhone
- `WCSession.defaultSession().sendMessage(…)`

watchOS Performance Strategies

Minimize network traffic and processing

Relying on existing server logic

- Implement a lightweight service on iPhone
- `WCSession.defaultSession().sendMessage(…)`
- Parse and pare down server responses on iPhone

watchOS Performance Strategies

Minimize network traffic and processing

Relying on existing server logic

- Implement a lightweight service on iPhone
- `WCSession.defaultSession().sendMessage(…)`
- Parse and pare down server responses on iPhone
- Reply over WCSession with minimal working set

Summary

효율적인 앱은 사용할때 기분이 좋다.

Performance is a feature

Efficient apps feel great, build trust, and save power

Learn about Apple technologies and choose the best ones for your app

Keep your main thread ready for user input

Understand when and why your app uses memory

On watchOS, fetch and process a minimal set of information

More Information

Documentation
Performance Overview
Instruments User Guide
Concurrency Programming Guide
Threading Programming Guide

http://developer.apple.com/library

Technical Support
Apple Developer Forums
Developer Technical Support
http://developer.apple.com/forums

General Inquiries
Curt Rothert, App Frameworks Evangelist
rothert@apple.com

Related Sessions

Optimizing Your App for Multitasking on iPad in iOS 9	Presidio	Wednesday 3:30PM
Designing for Apple Watch	Presidio	Wednesday 4:30PM
What's New in Core Data	Mission	Thursday 2:30PM
Profiling in Depth	Mission	Thursday 3:30PM
Building Responsive and Efficient Apps with GCD	Nob Hill	Friday 10:00AM
iOS App Performance: Memory		WWDC12
Advanced Graphics and Animations for iOS Apps		WWDC14
Improving Your App with Instruments		WWDC14

Related Labs

Power and Performance Lab	Frameworks Lab C	Friday 12:00PM


WWDC15