How to implement didReceiveMemoryWarning in Swift?

Asked 9 years, 3 months ago Modified 7 years, 9 months ago Viewed 18k times

Part of Mobile Development Collective

35

Whenever I create a new View Controller subclass, Xcode automatically adds the method

```
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated
}
```

Usually I just delete it or ignore it. This is what all the tutorials I have seen do, too. But I assume that since Xcode gives it to me every time, it should be somewhat important, right? What should I be doing here? I assume that disposing of resources means setting them to nil, but what exactly are "resources that can be recreated"?

I have seen these questions:

- How to implement didReceiveMemoryWarning?
- UIViewController's didReceiveMemoryWarning in ARC environment
- iPhone Development - Simulate Memory Warning

But they are all pre-Swift. Although I don't know much about Objective-C, I have heard that the memory management is different. How does that affect what I should do in `didReceiveMemoryWarning`?

Other notes:

- I am fuzzily aware of Automatic Reference Counting and lazy instantiation
- The documentation on `didReceiveMemoryWarning` that I found was rather brief.

MD ios swift didreceivememorywarning

Share
Improve this question
Follow

edited May 23, 2017 at 10:31
Community Bot
1 • 1

asked Sep 4, 2015 at 2:07
Suragch
510k • 335 • 1.4k • 1.4k

2 Answers

Sorted by: Highest score (default)

45

Swift

Swift uses ARC just like Objective-C does (source to Apple Docs). The same kind of rules apply for freeing memory, remove all of the references to an object and it will be deallocated.

How to free memory

> I assume that disposing of resources means setting them to nil, but what exactly are "resources that can be recreated"?

"resources that can be recreated" really depends on your application.

Examples

Say you are a social media app that deals with a lot of pictures. You want a snappy user interface so you cache the next 20 pictures in memory to make scrolling fast. Those images are always saved on the local file system.

- Images can take up a lot of memory
- You don't need those images in memory. If the app is low on memory, taking an extra second to load the image from a file is fine.
- You could entirely dump the image cache when you receive that memory warning.
- This will free up memory that the system needs

You are creating an amazing game that has a number of different levels. Loading a level into your fancy game engine takes a while so if the user has enough memory you can load level 3 while they are playing level 2.

- Levels take up a lot of memory
- You don't NEED the next level in memory. They are nice to have but not essential.
- `LevelCache.sharedCache().nextLevel = nil` frees up all that memory

What shouldn't you deallocate

Never deallocate the stuff that is on screen. I've seen some answers to related questions deallocate the view of the UIViewController. If you remove everything from the screen you might at well crash (in my opinion).

Examples

If the user has a document open that they are editing, DON'T deallocate it. Users will get exceptional angry at you if your app deletes their work without ever being saved. (In fact, you should probably have some emergency save mechanism for when that happens)

If your user is writing a post for your fabulous social media app, don't let their work go to waste. Save it and try to restore it when they open the app again. Although it's a lot of work to setup I love apps that do this.

Note

Most modern devices rarely run out of memory. The system does a pretty good job of killing apps in the background to free up memory for the app running in the foreground. You have probably seen an app "open" in the App Switcher yet when you tapped on the app it opened to its initial state. The OS killed the app in the background to free up memory. See State Restoration for info on how to avoid this problem.

If your app is getting consistent memory warnings when you aren't doing a huge amount of processing make sure that you aren't leaking memory first. Detecting memory leaks is outside the scope of this answer. Docs and a tutorial.

Share

Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 • 1

answered Sep 4, 2015 at 2:37

Kevin

17.5k • 7 • 53 • 68

1 In general, you shouldn't need to dispose of that many resources. I updated the answer to give more detail. – Kevin Sep 4, 2015 at 3:37

3 I just came back and read your answer again. It was very helpful at the time and the refresher was helpful again. I see you made an edit to help other people who are coming to this Q&A. Another edit that I think would be helpful for people is a code example. It doesn't need to be long, but it should include the `didReceiveMemoryWarning()` function. It always helps me to see specific examples in context. – Suragch Mar 16, 2017 at 23:44

When `didReceiveMemoryWarning` is called, it means your app is using too much memory (compare with memory of device), and you should `release any additional`

5

`memory used by your view controller` to reduce the memory of your app. If the memory app gets over the memory of device, iOS will kill your app immediately. "resources that can be recreated" means somethings you can recreate it again at somewhere, you don't need them now (don't need to put them in the memory). And you can release it when get `didReceiveMemoryWarning`.

Here is another detail topic: [ios app maximum memory budget]

Share

Improve this answer

Follow

edited May 23, 2017 at 12:10

Community Bot

1 • 1

answered Sep 4, 2015 at 2:16

tuledev

10.3k • 4 • 31 • 50

1 Your link was useful for knowing the memory threshold on different devices. – Suragch Sep 4, 2015 at 2:42