Stage 3.4 - Pros & Cons

Scene 1: On-Set

Singletons are not the answer to every problem. Like any tool, they can be misapplied and overused.
Some developers are critical of singletons for various reasons.
We'll examine these critiques and discuss ways to address them.

The criticisms, for the most part, fall into two categories:
<KEYNOTE>

1. **Singletons hinder unit testing**
2. **Singletons create hidden dependencies**

</KEYNOTE>

1. A singleton can hinder unit-testing when objects and their methods become dependent upon it to the point of being untestable without a fully functional singleton class. Unit testing relies on the ability to test a method, object or module in relative isolation from the rest of the codebase. Integration (or coupling) with objects which hampers the ability to isolate a test target is problematic. Singletons with complex interfaces and significant internal state can make unit testing more difficult.
2. Since the singleton reference is readily available throughout an application through a class method class, it can be overused. And, since the reference is not explicitly passed, dependency on the singleton can become difficult to track.

To avoid these complications, when considering the singleton pattern a developer should:
<KEYNOTE>

1. **Make certain you class should be a singleton**
2. **Consider unit testing when designing the singleton architecture**
3. **Use *dependency injection* when possible**

</KEYNOTE>

1. A class should be a singleton if there can, and should be only one. If there is a question as to whether uniqueness is appropriate, it probably isn't.
2. Unit testing should always be a consideration, but singletons should be given special attention. A simple interface that can be simulated as a *test-harness* is recommended. The singleton should provide an interface that hides complexity and allows for the implementation to be swapped out for testing. State stored in the singleton should also be minimized.
3. Dependency injection is the explicit passing of a reference to expose dependency: Though any class can access the singleton reference through a class method, this

should be done sparingly. When possible pass the singleton as an init method parameter when objects are created.

Questions:

1. True or False: A singleton should never be allowed to do network communication. (ans. false)
2. NSUserDefaults stores data in:
    a. Core Data
    b. iCloud
    c. the file system (correct answer)
3. True or False. It is acceptable to have two coexisting singleton instances. (ans false)
4. Deferring allocation and initialization until an object is needed is called? (ans. “lazy instantiation”)
5. To ensure that a +sharedInstance method works correctly with possible subclasses, the call to alloc should be ________. (ans. [[self class] alloc])