Registering for Key-Value Observing

In order to receive key-value observing notifications for a property, three things are required:

- The observed class must be key-value observing compliant for the property that you wish to observe.
- You must register the observing object with the observed object, using the method `addObserver:forKeyPath:options:context:`.
- The observing class must implement `observeValueForKeyPath:ofObject:change:context:`.

Important: Not all classes are KVO-compliant for all properties. You can ensure your own classes are KVO-compliant by following the steps described in KVO Compliance. Typically properties in Apple-supplied frameworks are only KVO-compliant if they are documented as such.

Registering as an Observer

In order to be notified of changes to a property, an observing object must first register with the object to be observed by sending it an `addObserver:forKeyPath:options:context:` message, passing the observer object and the key path of the property to be observed. The options parameter specifies the information that is provided to the observer when a change notification is sent. Using the option `NSKeyValueObservingOptionOld` specifies that the original object value is provided to the observer as an entry in the change dictionary. Specifying the `NSKeyValueObservingOptionNew` option provides the new value as an entry in the change dictionary. To receive both values, you would bitwise `OR` the option constants.

The example in Listing 1 demonstrates registering an inspector object for the property `openingBalance`.

Listing 1 Registering the inspector as an observer of the openingBalance property

```
- (void)registerAsObserver {
    /*
     Register 'inspector' to receive change notifications for the "openingBalance" property of
     the 'account' object and specify that both the old and new values of "openingBalance"
     should be provided in the observe… method.
     */
    [account addObserver:inspector
             forKeyPath:@"openingBalance"
                 options:(NSKeyValueObservingOptionNew |
                            NSKeyValueObservingOptionOld)
                    context:NULL];
}
```

When you register an object as an observer, you can also provide a context pointer. The context pointer is provided to the observer when `observeValueForKeyPath:ofObject:change:context:` is invoked. The context pointer can be a C pointer or an object reference. The context pointer can be used as a unique identifier to determine the change that is being observed, or to provide some other data to the observer.

Note: The key-value observing `addObserver:forKeyPath:options:context:` method does not maintain strong references to the observing object, the observed objects, or the context. You should ensure that you maintain strong references to the observing, and observed, objects, and the context as necessary.

Receiving Notification of a Change

When the value of an observed property of an object changes, the observer receives an `observeValueForKeyPath:ofObject:change:context:` message. All observers must implement this method.

The observer is provided the object and key path that triggered the observer notification, a dictionary containing details about the change, and the context pointer that was provided when the observer was registered.

The change dictionary entry `NSKeyValueChangeKindKey` provides information about the type of change that occurred. If the value of the observed object has changed, the `NSKeyValueChangeKindKey` entry returns `NSKeyValueChangeSetting`. Depending on the options specified when the observer was registered, the `NSKeyValueChangeOldKey` and `NSKeyValueChangeNewKey` entries in the change dictionary contain the values of the property before, and after, the change. If the property is an object, the value is provided directly. If the property is a scalar or a C structure, the value is wrapped in an `NSValue` object (as with key-value coding).

If the observed property is a to-many relationship, the `NSKeyValueChangeKindKey` entry also indicates whether objects in the relationship were inserted, removed, or replaced by returning `NSKeyValueChangeInsertion`, `NSKeyValueChangeRemoval`, or `NSKeyValueChangeReplacement`, respectively.

The change dictionary entry for `NSKeyValueChangeIndexesKey` is an `NSIndexSet` object specifying the indexes in the relationship that changed. If `NSKeyValueObservingOptionNew` or `NSKeyValueObservingOptionOld` are specified as options when the observer is registered, the `NSKeyValueChangeOldKey` and `NSKeyValueChangeNewKey` entries in the change dictionary are arrays containing the values of the related objects before, and after, the change.

The example in Listing 2 shows the `observeValueForKeyPath:ofObject:change:context:` implementation for an inspector that reflects the old and new values of the property `openingBalance`, as registered in Listing 1.

Listing 2 Implementation of observeValueForKeyPath:ofObject:change:context:

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context {

    if ([keyPath isEqual:@"openingBalance"]) {
        [openingBalanceInspectorField setObjectValue:
            [change objectForKey:NSKeyValueChangeNewKey]];
    }
    /*
     Be sure to call the superclass's implementation *if it implements it*.
     NSObject does not implement the method.
     */
    [super observeValueForKeyPath:keyPath
```

```
                                  ofObject:object
                                    change:change
                                   context:context];
}
```

Removing an Object as an Observer

You remove a key-value observer by sending the observed object a `removeObserver:forKeyPath:` message, specifying the observing object and the key path. The example in Listing 3 removes the inspector as an observer of `openingBalance`.

Listing 3 Removing the inspector as an observer of openingBalance

```
- (void)unregisterForChangeNotification {
    [observedObject removeObserver:inspector forKeyPath:@"openingBalance"];
}
```

If the context is an object, you must keep a strong reference to it until removing the observer. After receiving a `removeObserver:forKeyPath:` message, the observing object will no longer receive any `observeValueForKeyPath:ofObject:change:context:` messages for the specified key path and object.