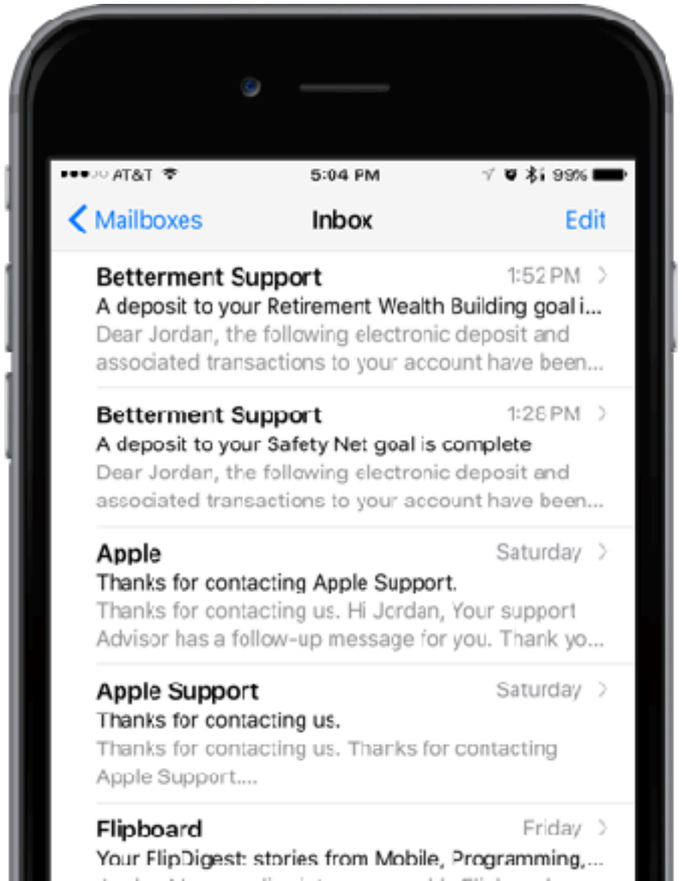
Peek and Pop

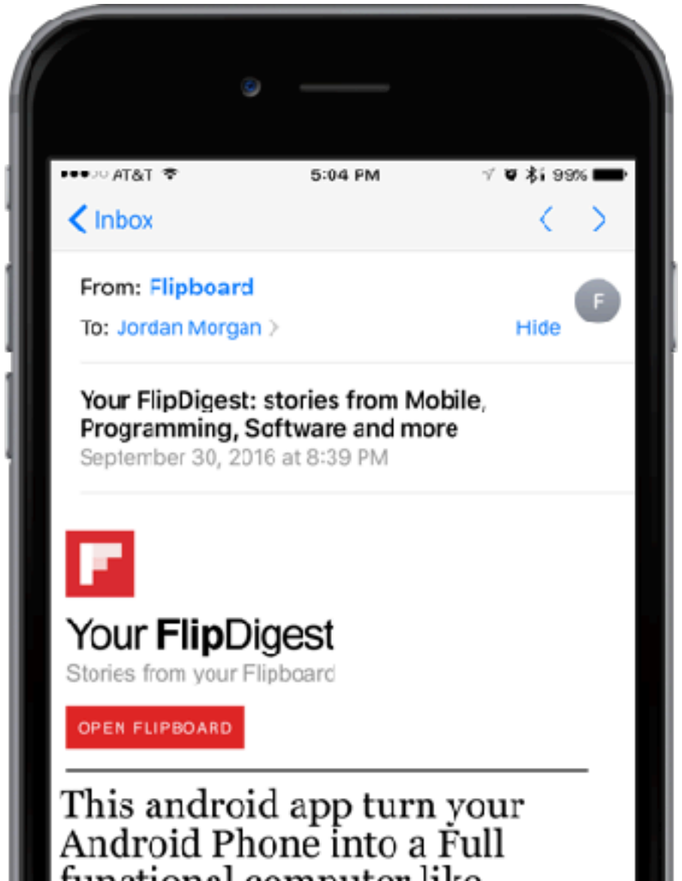
Peek and Pop in Action

When to Use Peek and Pop

Parent

Child





```
// Check for force touch feature, and add force touch/previewing capability.
if traitCollection.forceTouchCapability == .available
{
    registerForPreviewing(with: self, sourceView: tableView)
}
```

Registering for Previewing

Tells iOS this view controller will:

- **Preview content**
- **Provide a source view**
- **Provide a delegate to mediate**

```
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController?
{
        // Grab a data model from the CGPoint
        // Set data source
        // Just return nil.
        // Create a view controller and return it
}
```

UIViewControllerPreviewingDelegate

Implementing previewingContext:viewControllerForLocation

```swift
/// Present the view controller for the "Pop" action.
func previewingContext(_ previewingContext: UIViewControllerPreviewing,
commit viewControllerToCommit: UIViewController)
{
    // Reuse the "Peek" view controller for presentation.
    show(viewControllerToCommit, sender: self)
}
```

UIViewControllerPreviewingDelegate

Implementing previewingContext:commitViewController

Demo

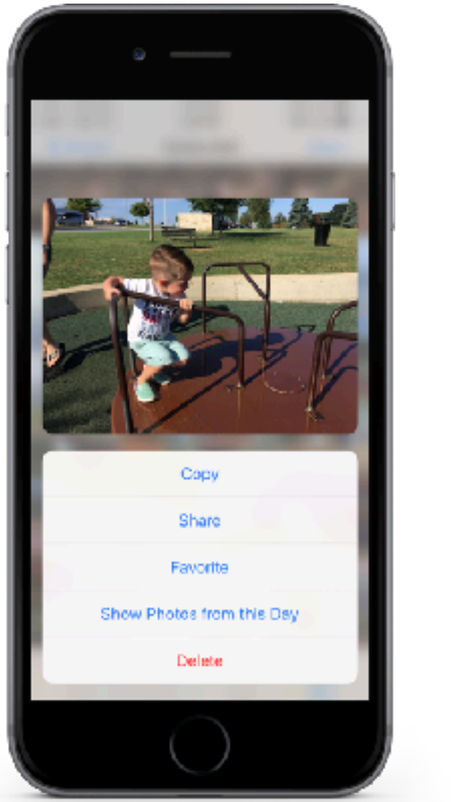
- Register for previewing
- Handle 3D Touch being toggled

Demo

- Prepare a preview controller
- Present a previewed controller

Best Practices: Peek and Pop

Best Practices



- Tappable content should be peekable
- Return fast from delegate methods
- Query UITraitCollection changes
- Set previewing context sourceRect

Summary

Peek and Pop boosts previewing and navigation

Works well in parent/child UX scenarios

Registering view controllers for Peek and Pop

UIViewControllerPreviewingDelegate

Best Practices

- Tappable content should be peekable
- Return quick from delegate methods
- Check for trait collection changes
- Set a sourceRect on preview context