App Frameworks #WWDC16

What's New in UICollectionView in iOS 10

Session 219

Steve Breen UIKit Frameworks Engineer
Peter Hajas UIKit Frameworks Engineer

© 2016 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

Overview

Overview

Smooth scrolling

Overview

Smooth scrolling

Improvements to self-sizing cells

Overview

Smooth scrolling

Improvements to self-sizing cells

Interactive reordering

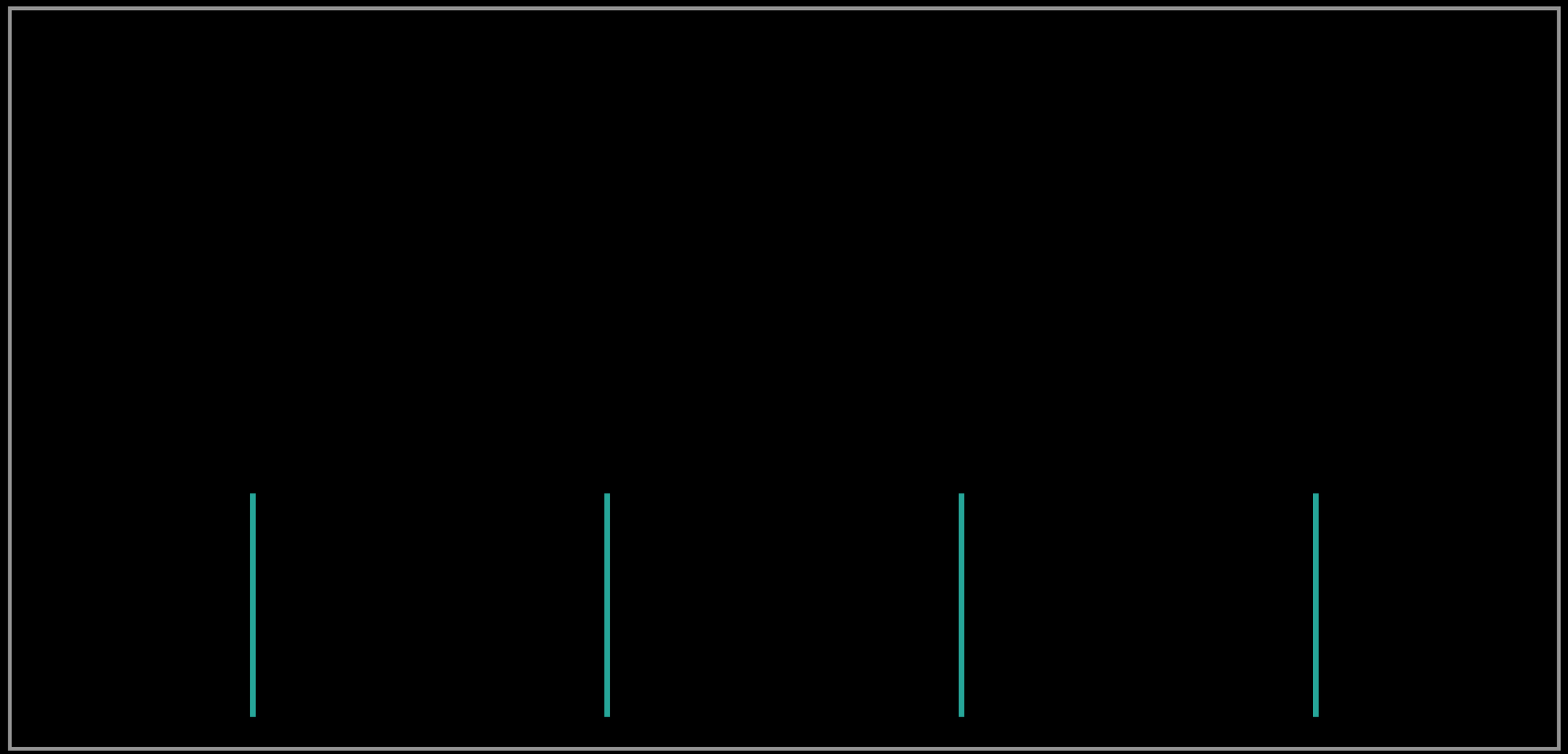
Smooth Scrolling

Combating choppy scrolling in your app

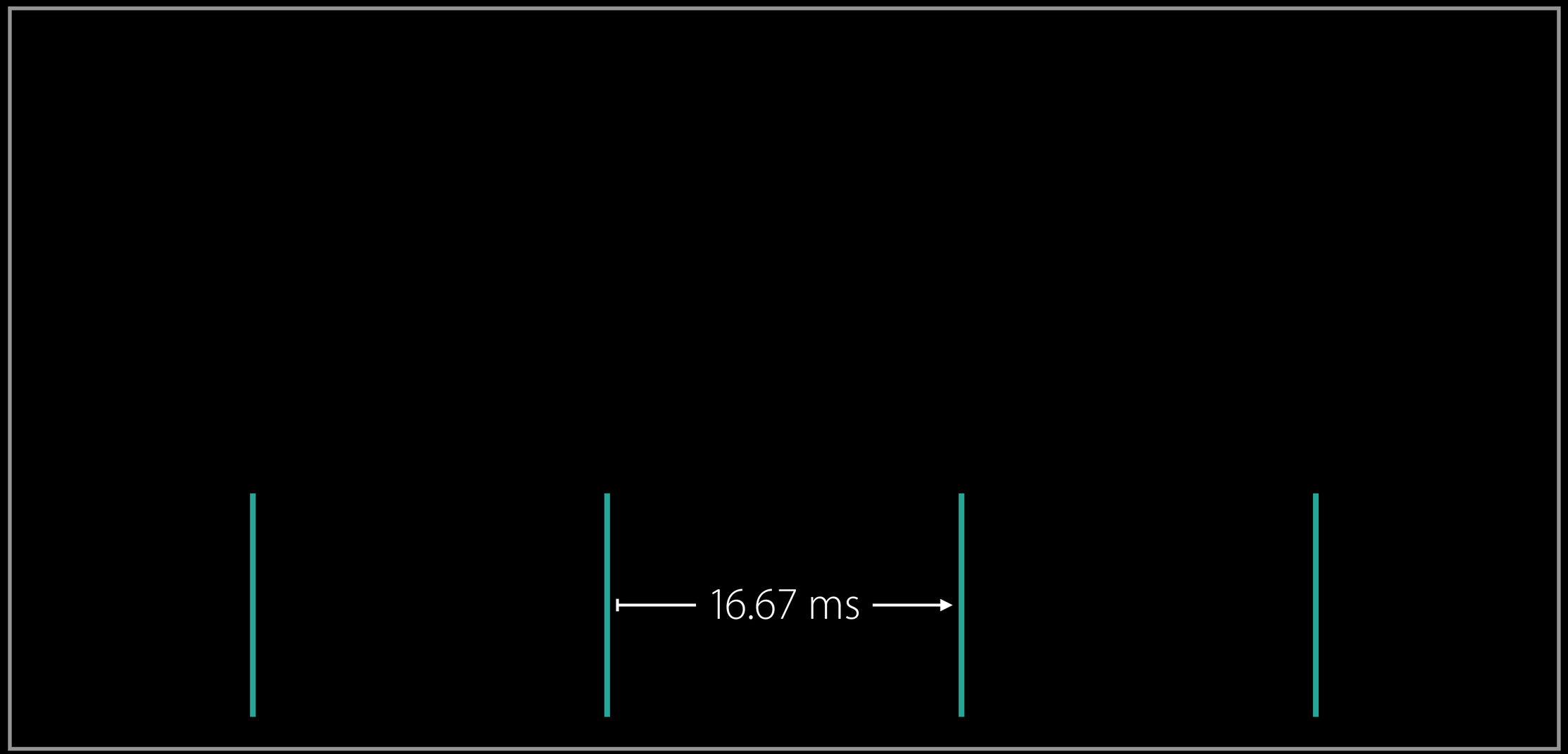
Demo

Scrolling like butter^W chunky peanut butter

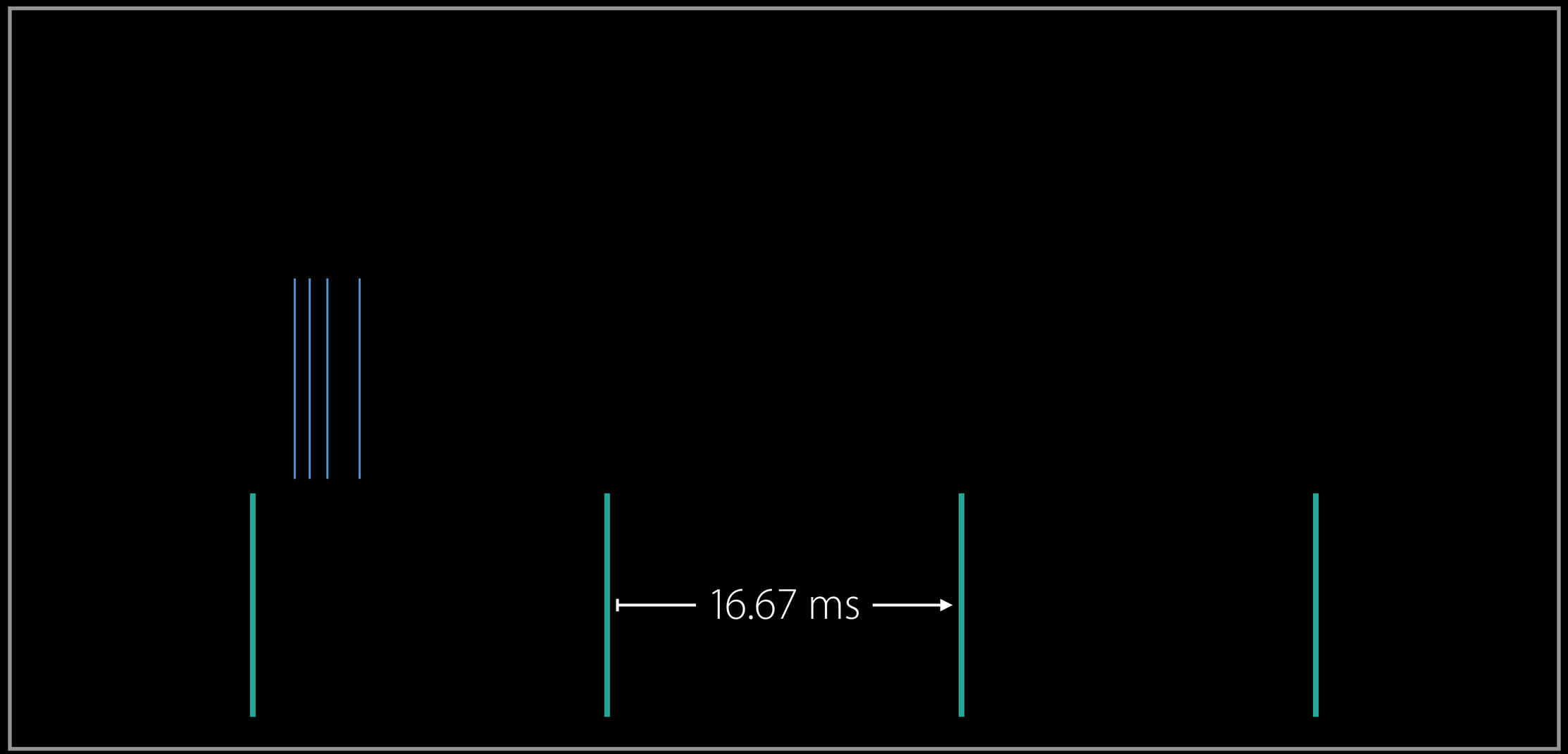
Anatomy of a Dropped Frame



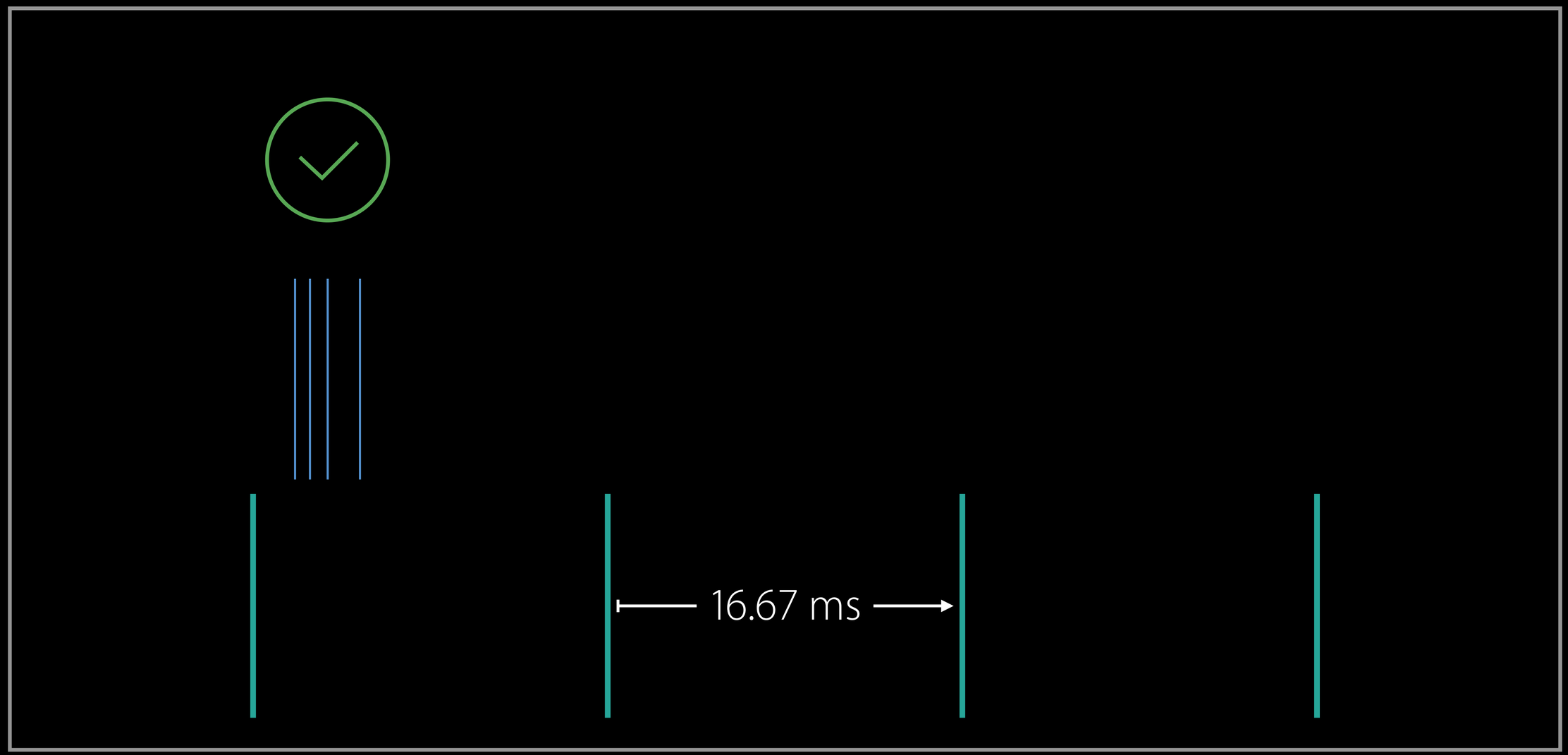
Anatomy of a Dropped Frame



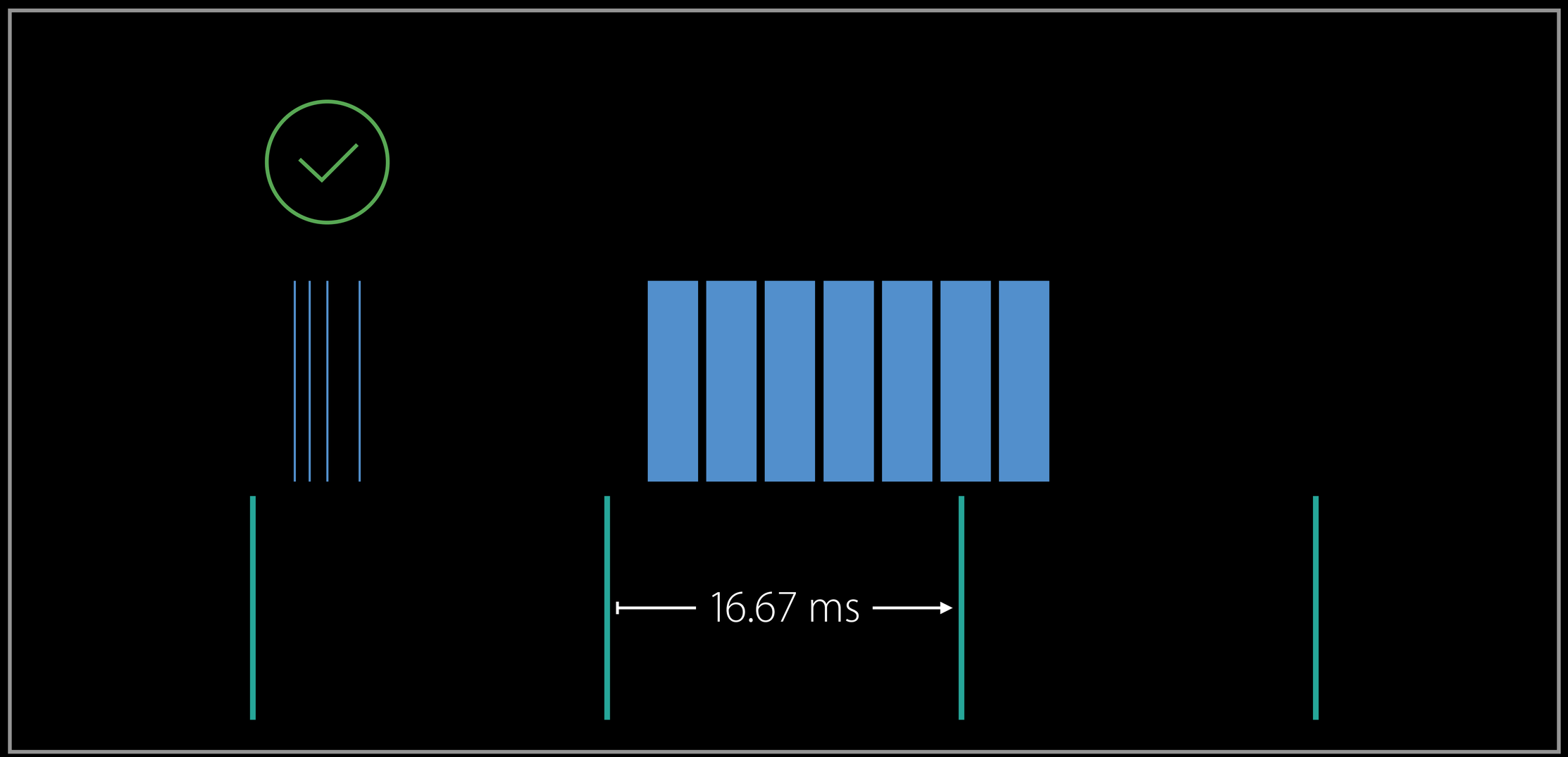
Anatomy of a Dropped Frame



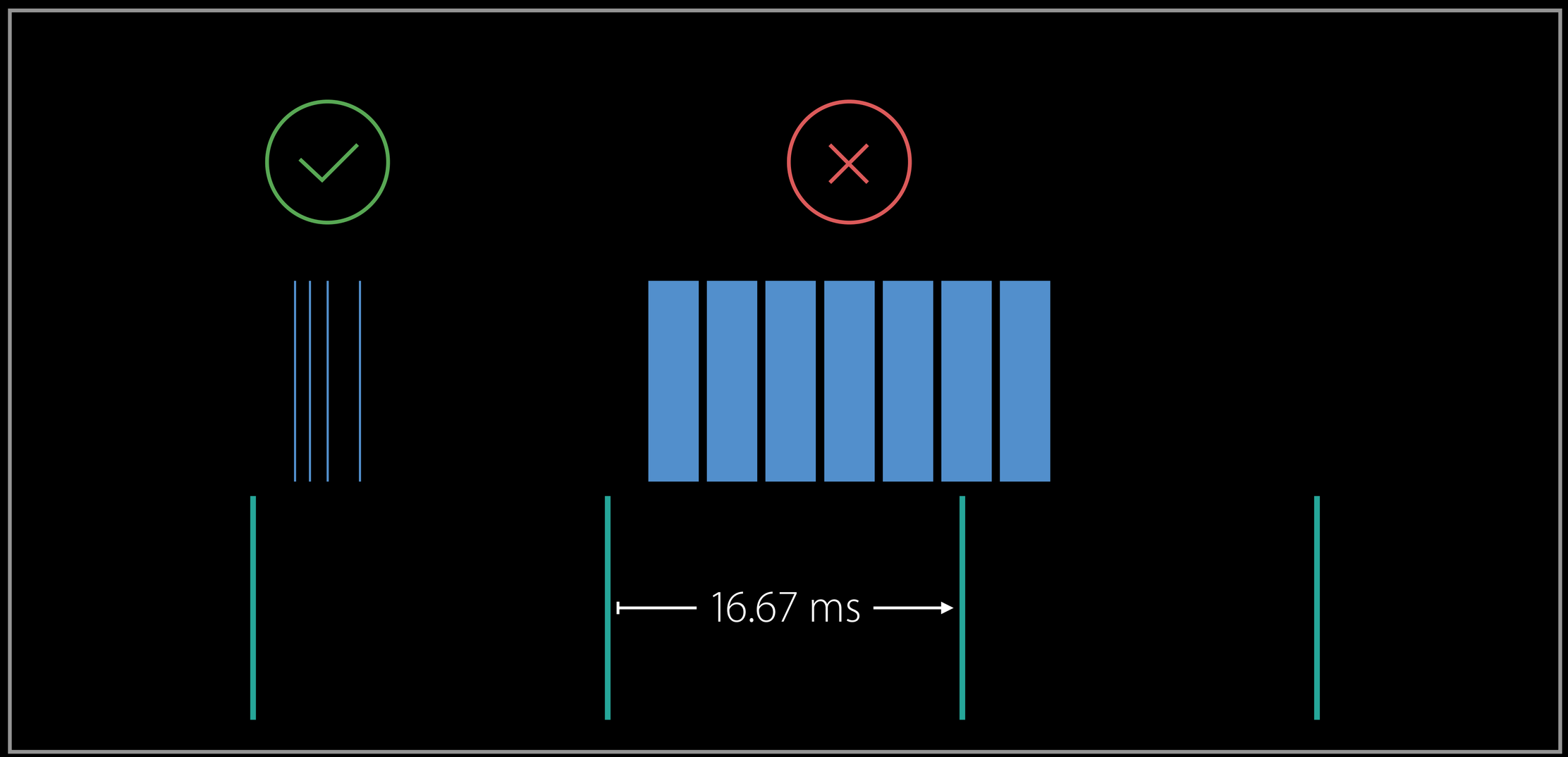
Anatomy of a Dropped Frame



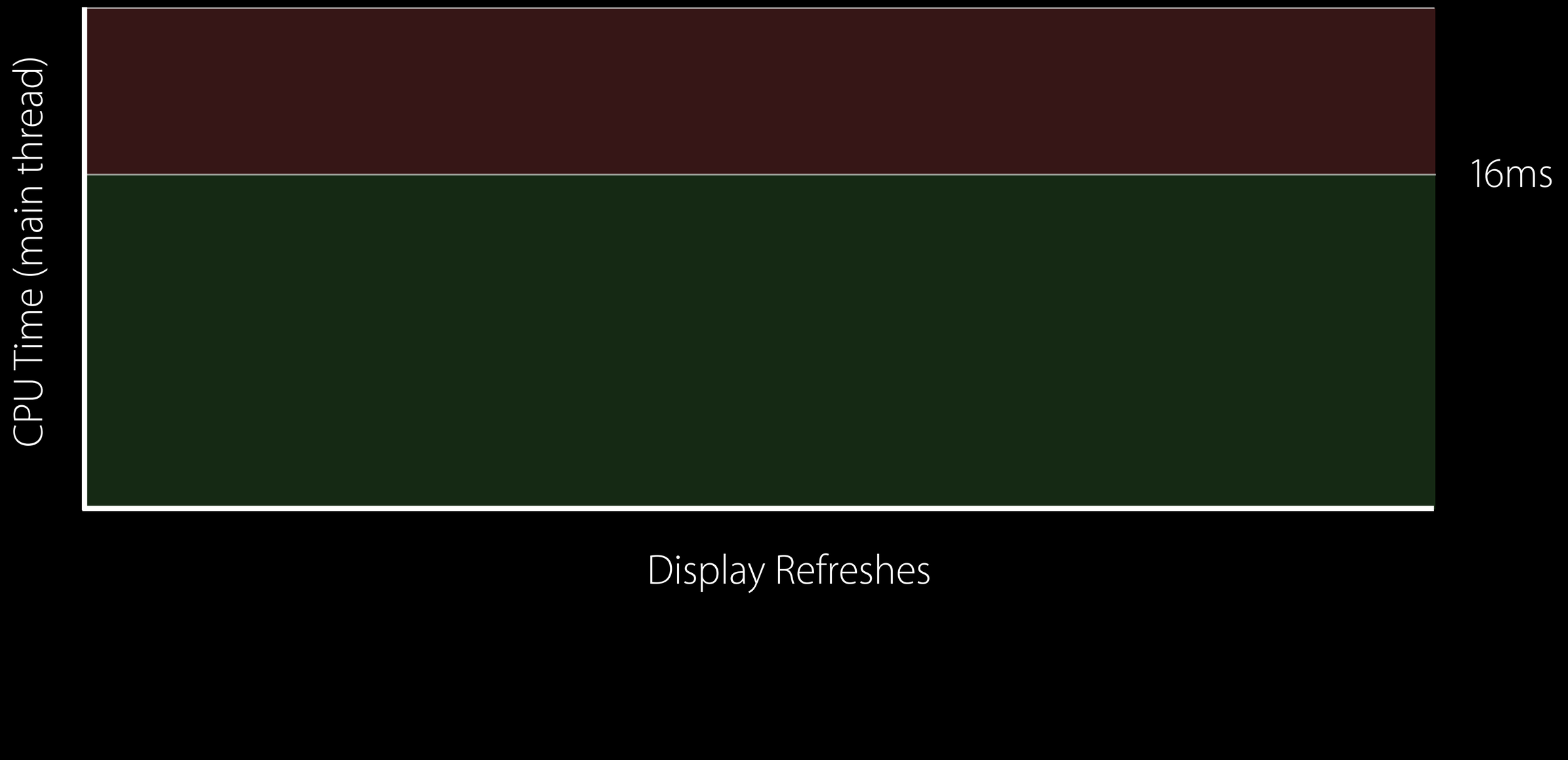
Anatomy of a Dropped Frame

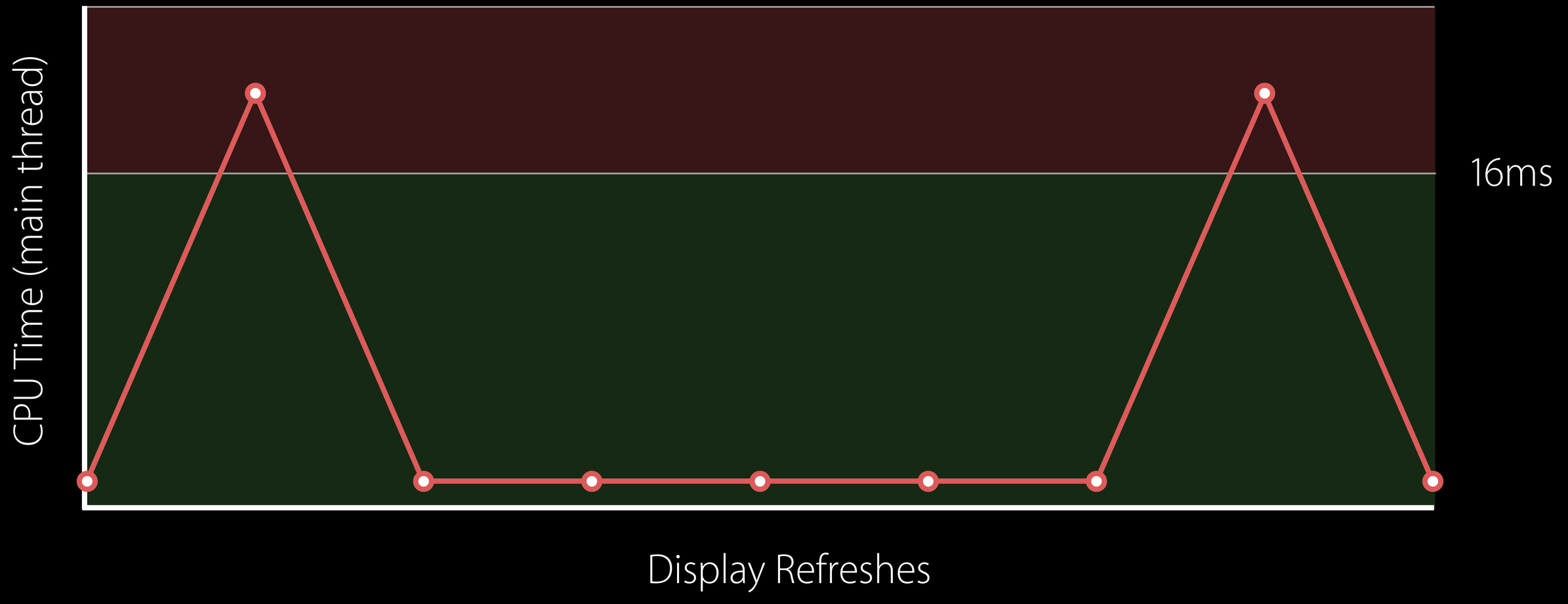


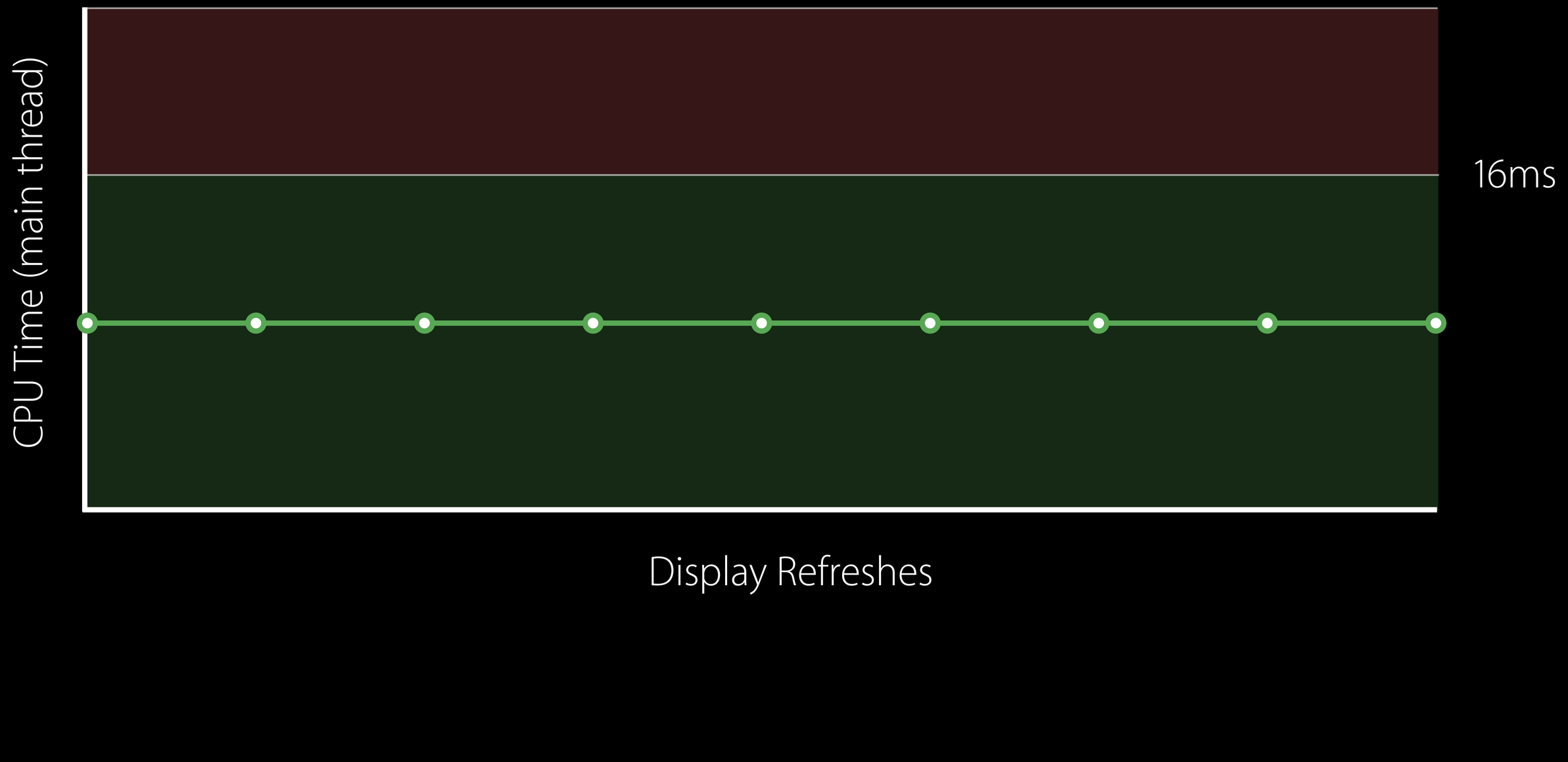
Anatomy of a Dropped Frame



16ms


CPU Time (main thread)
16ms
Display Refreshes


CPU Time (main thread)
16ms
Display Refreshes

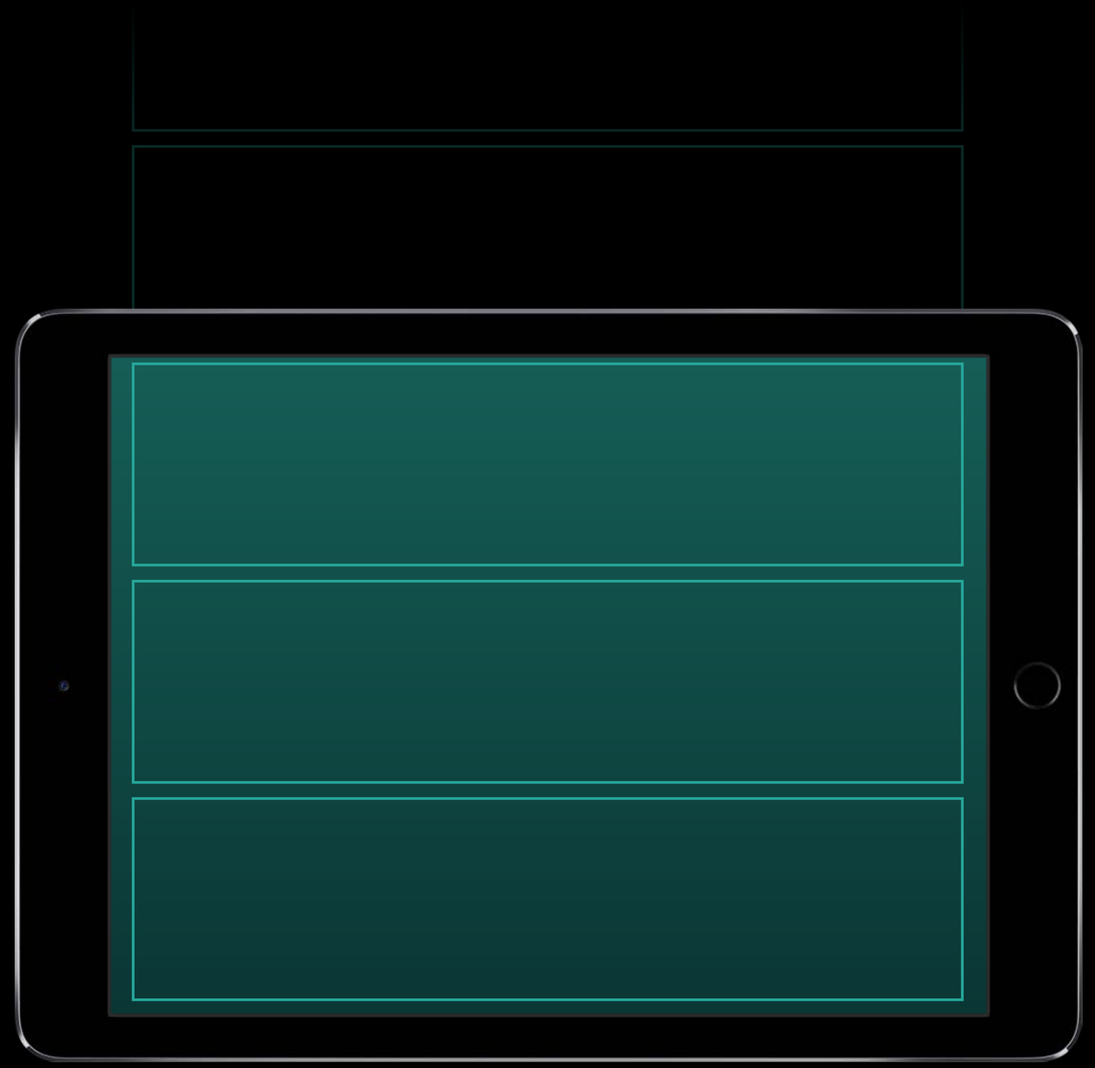

CPU Time (main thread)
16ms
Display Refreshes

Life Cycle of a Cell

iOS 9

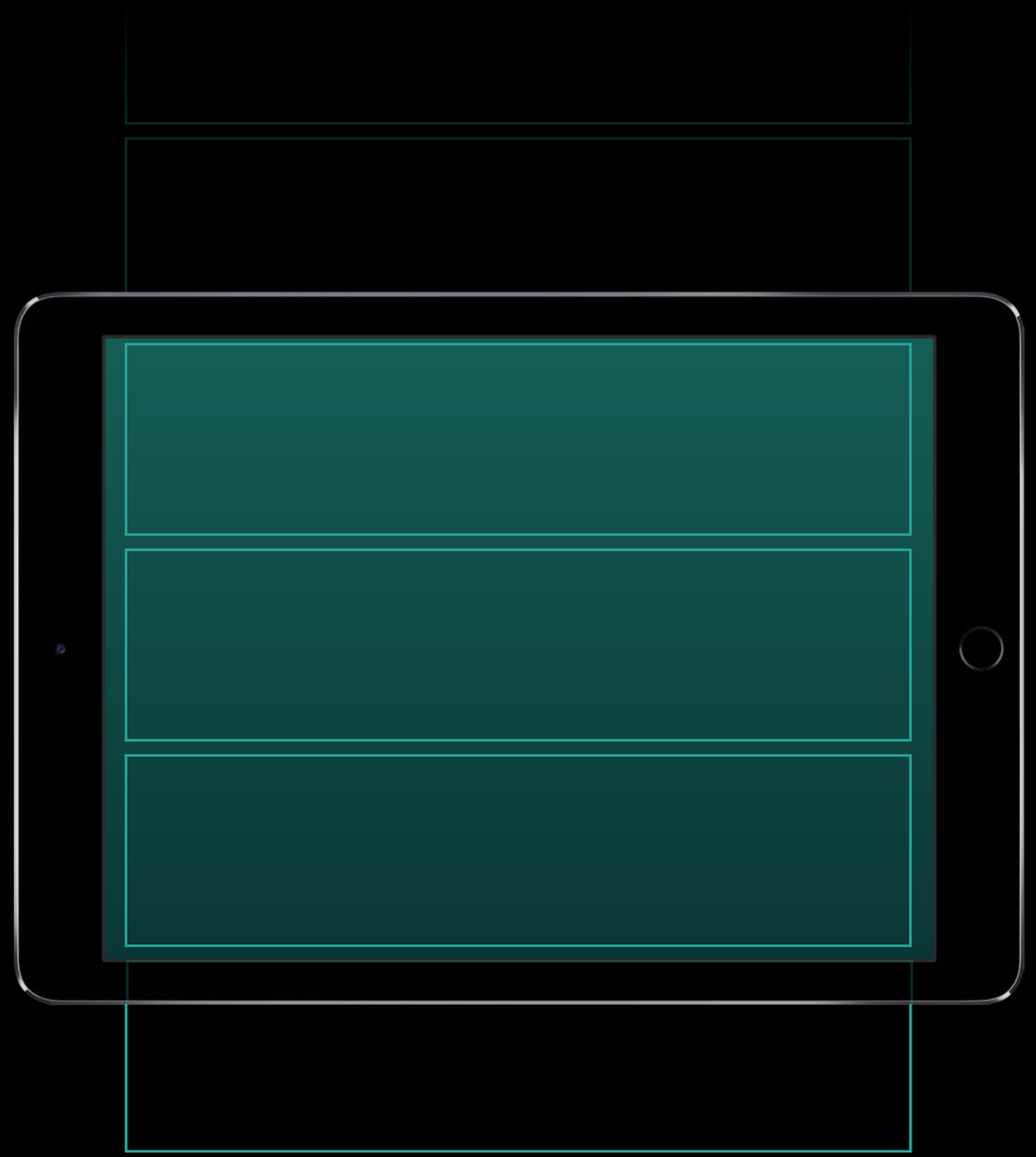
Life Cycle of a Cell

iOS 9



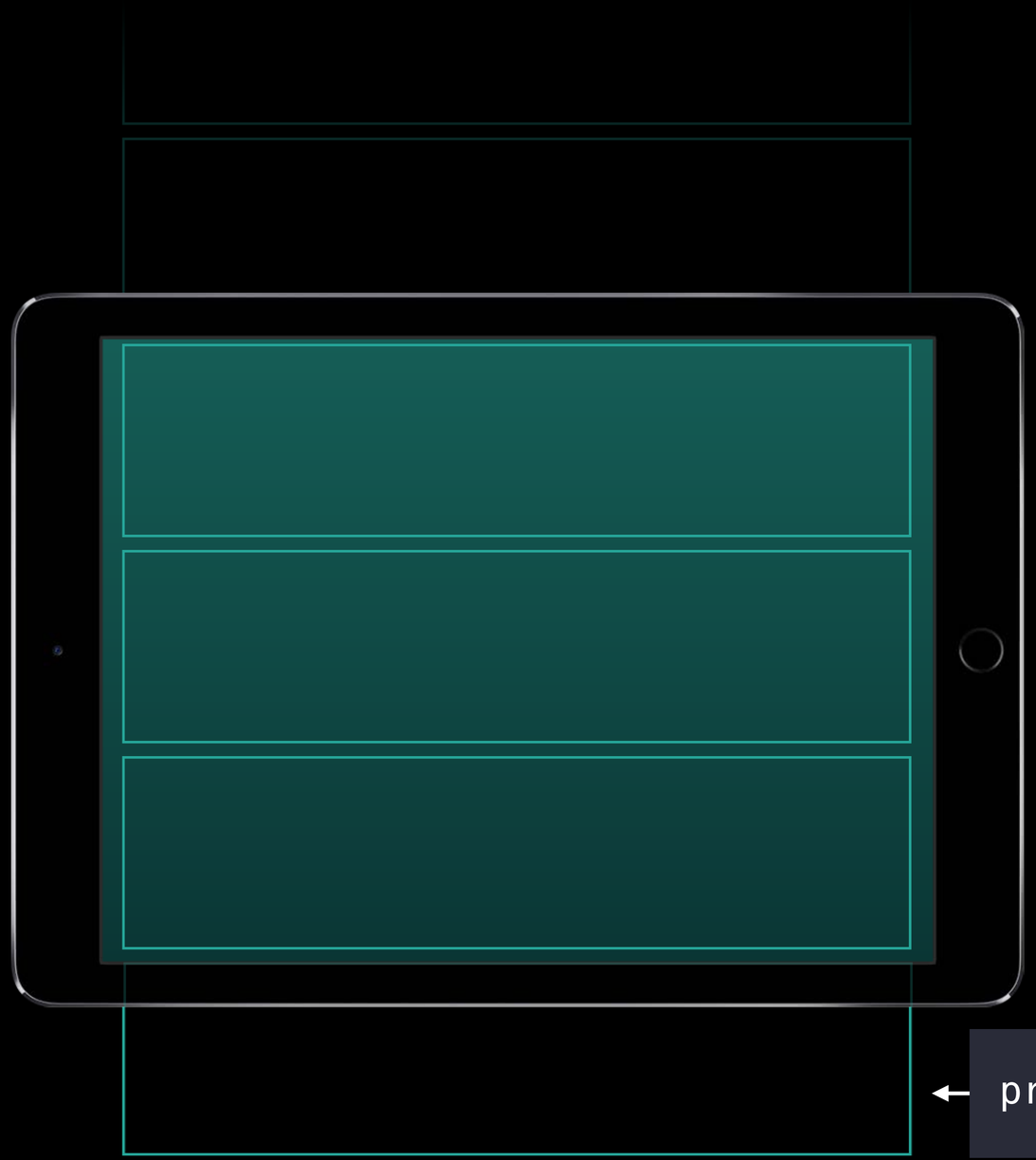
Life Cycle of a Cell

iOS 9



Life Cycle of a Cell

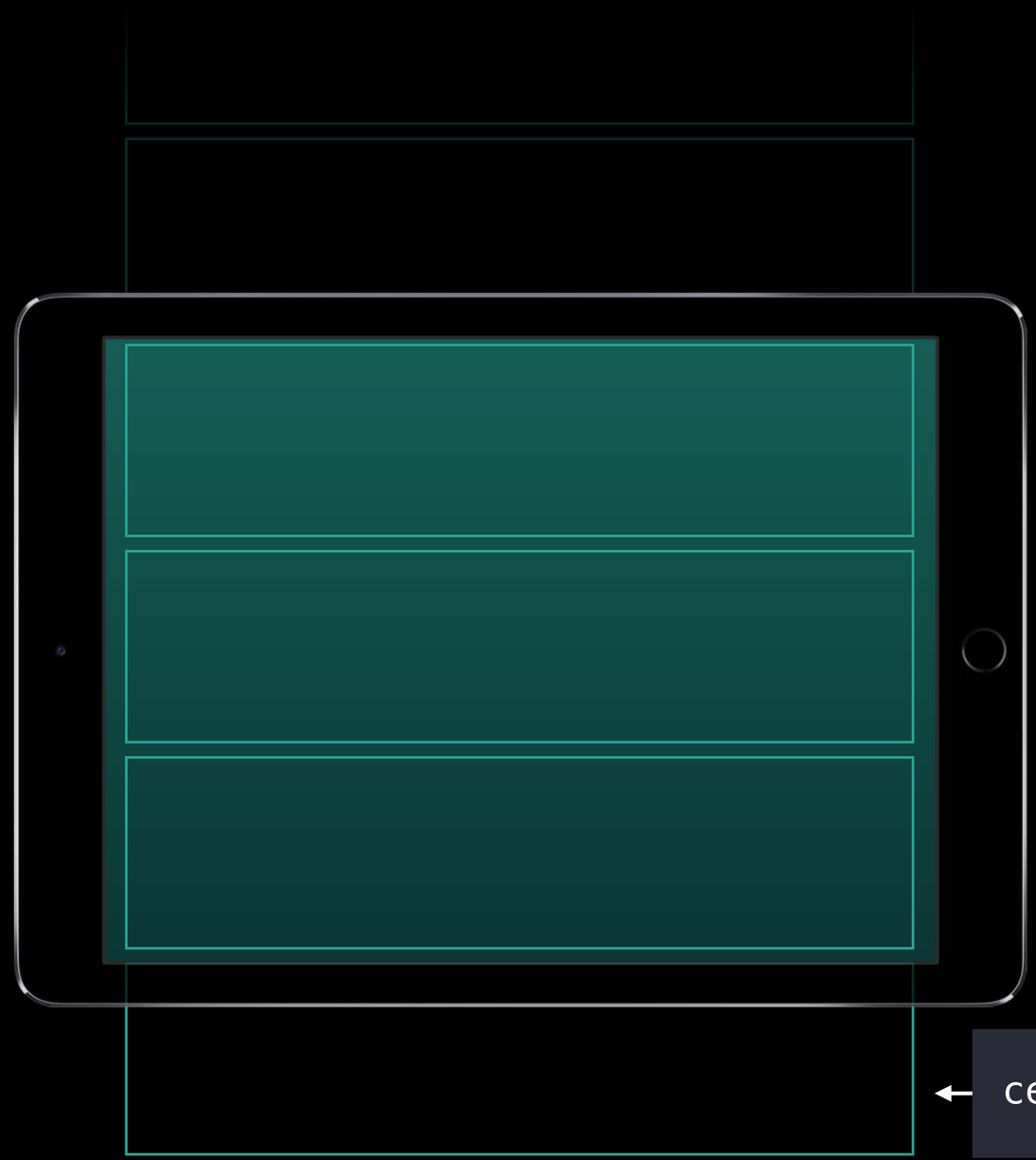
iOS 9



prepareForReuse

Life Cycle of a Cell

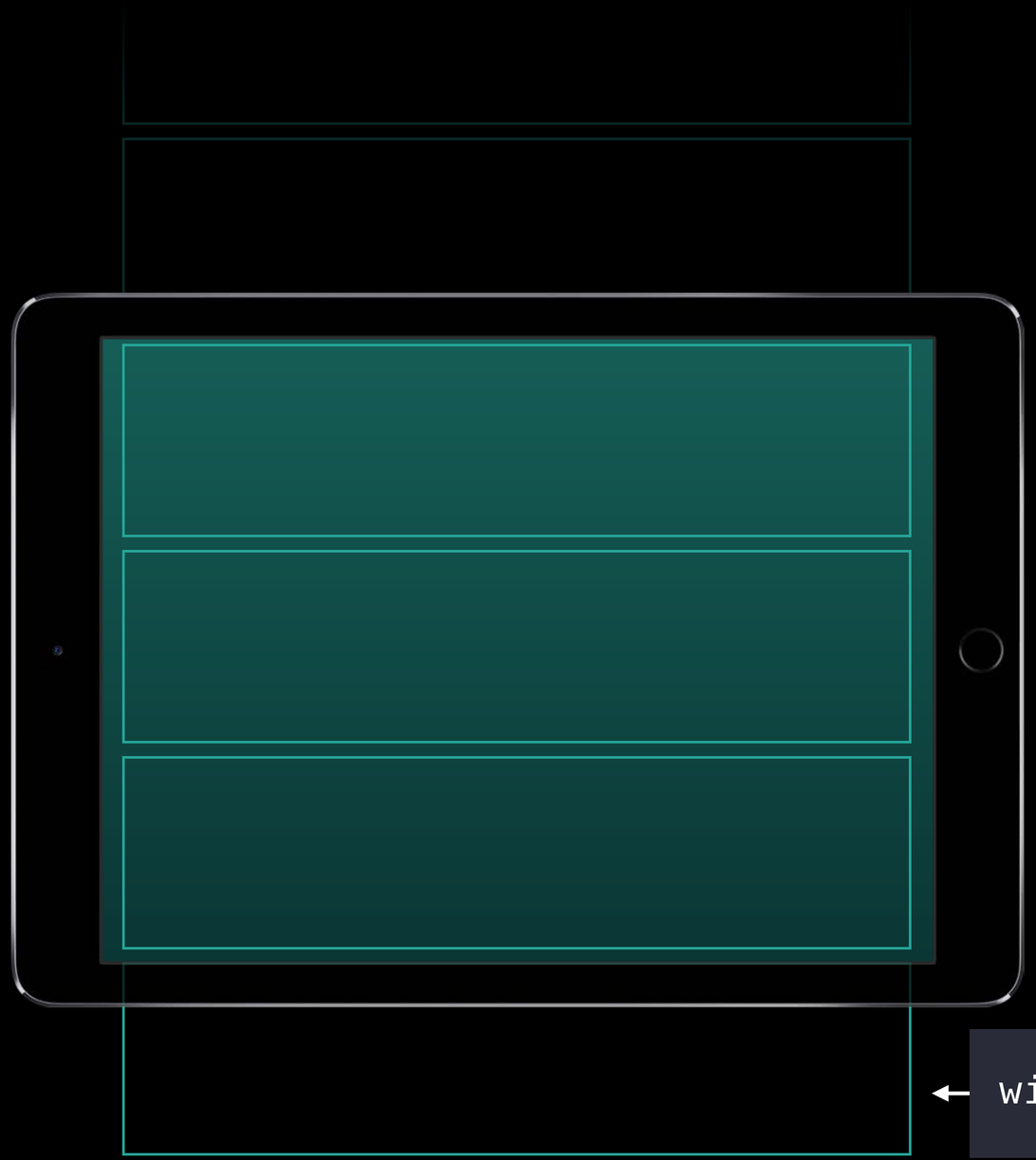
iOS 9



```
cellForItemAtIndexPath
```

Life Cycle of a Cell

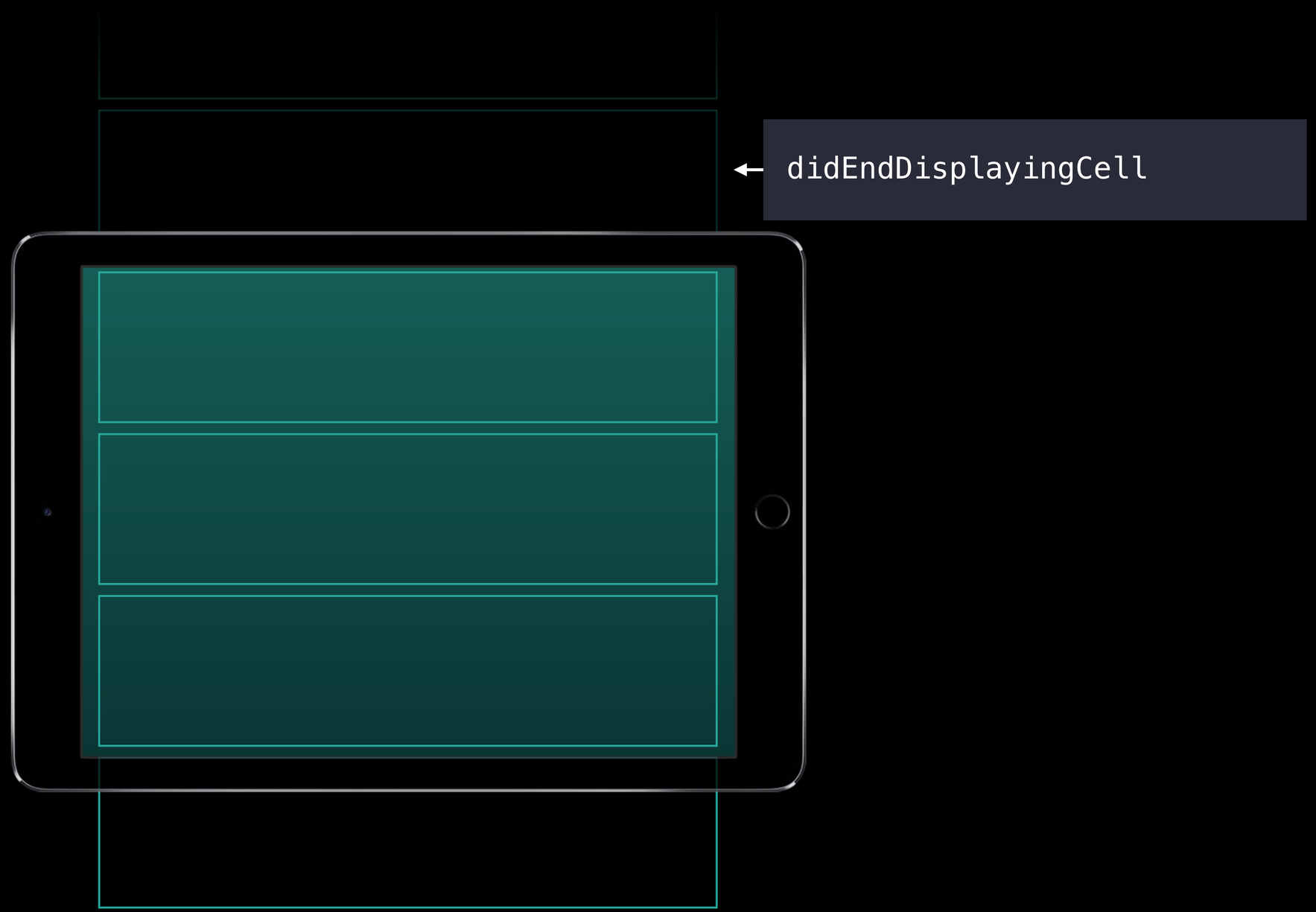
iOS 9



willDisplayCell

Life Cycle of a Cell

iOS 9



Life Cycle of a Cell

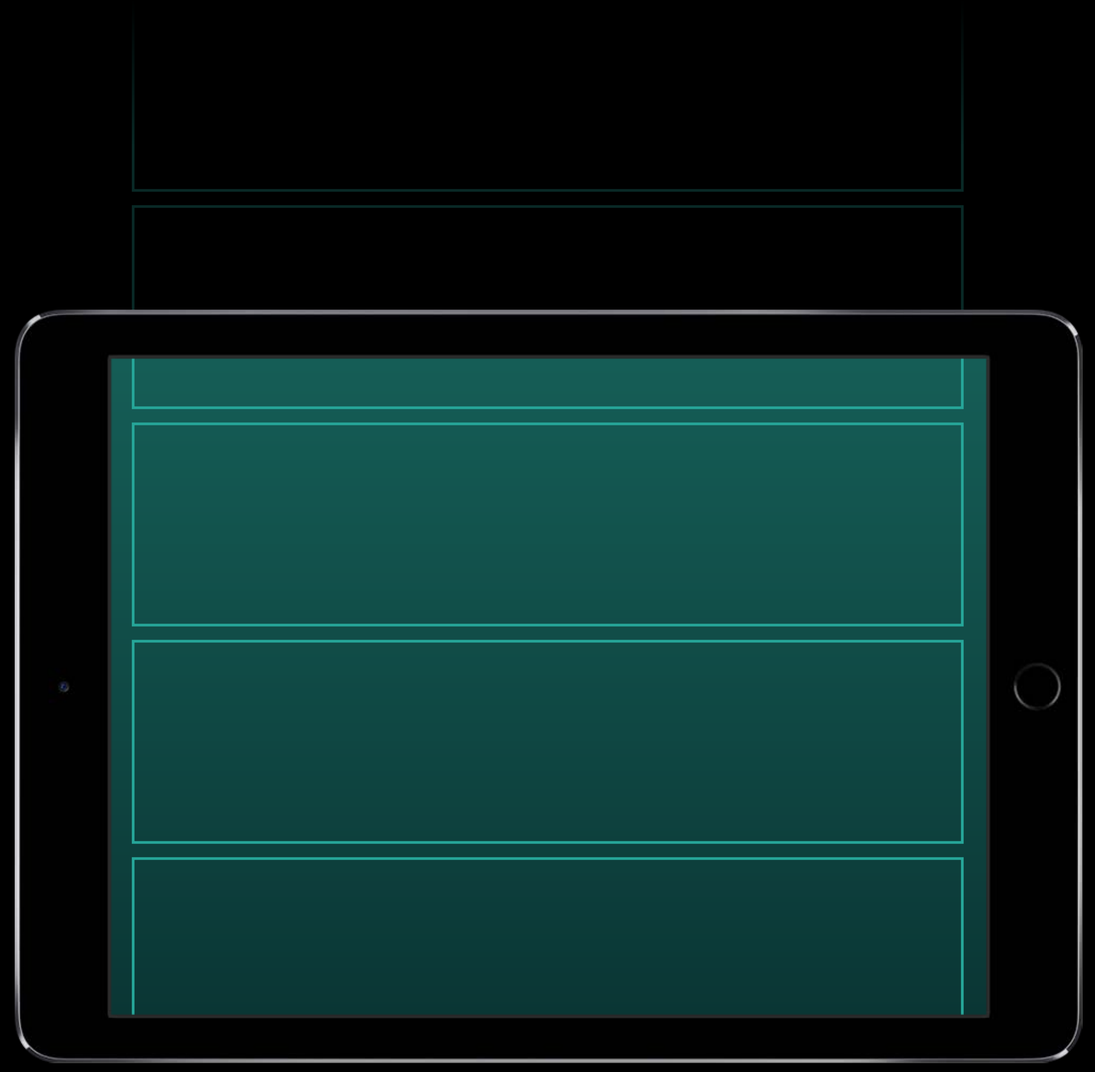
iOS 10

NEW

Life Cycle of a Cell

iOS 10

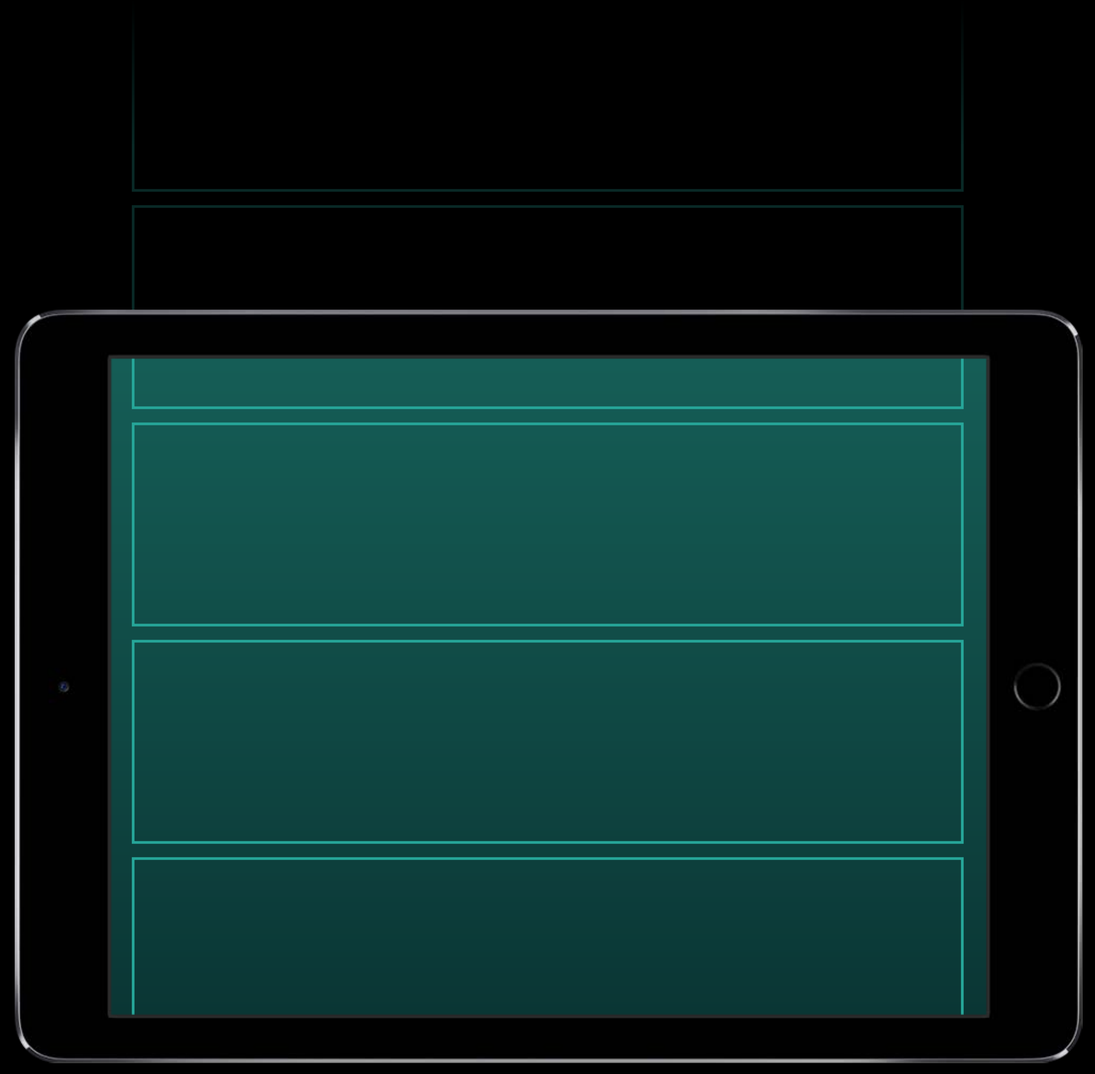
NEW



Life Cycle of a Cell

iOS 10

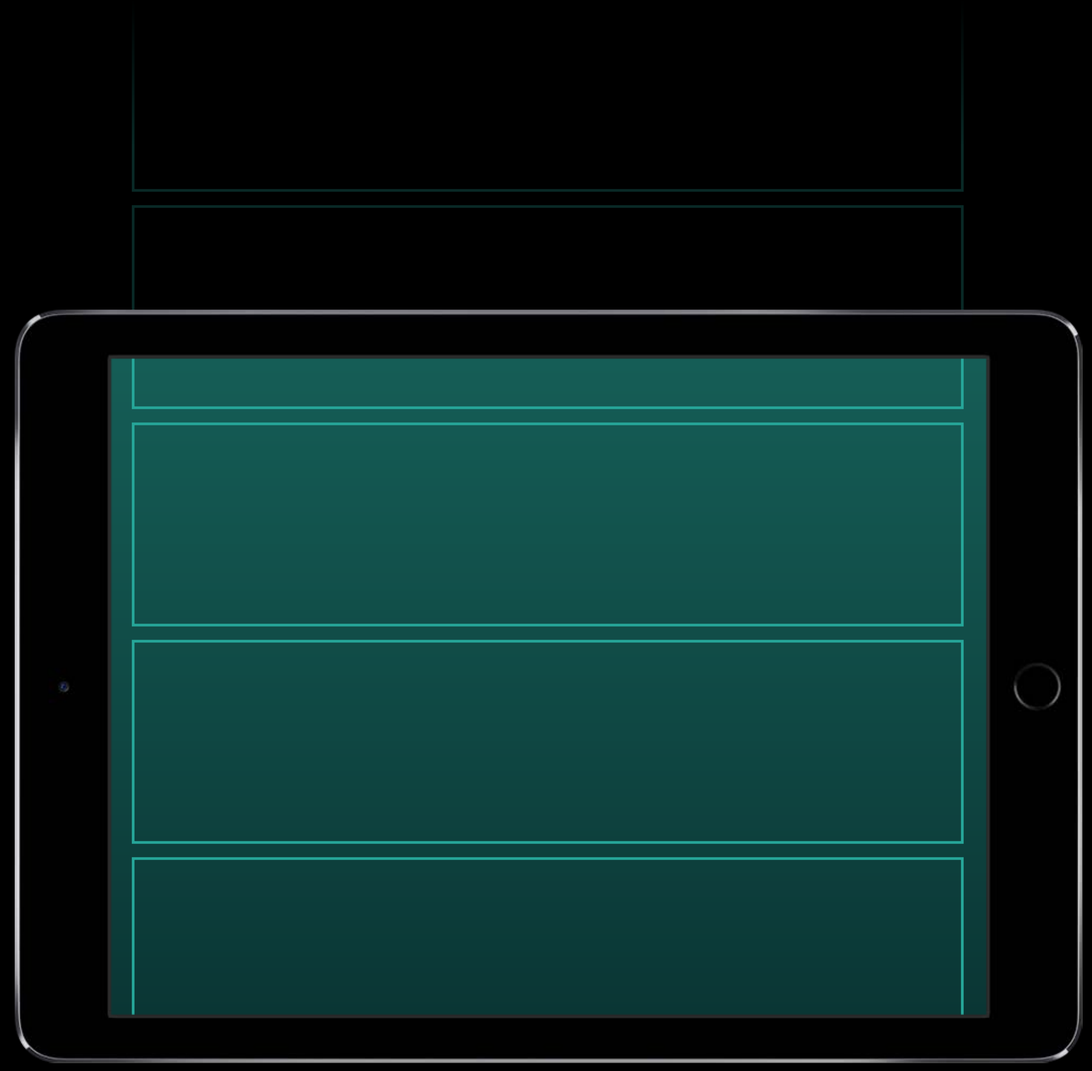
NEW



Life Cycle of a Cell

iOS 10

NEW

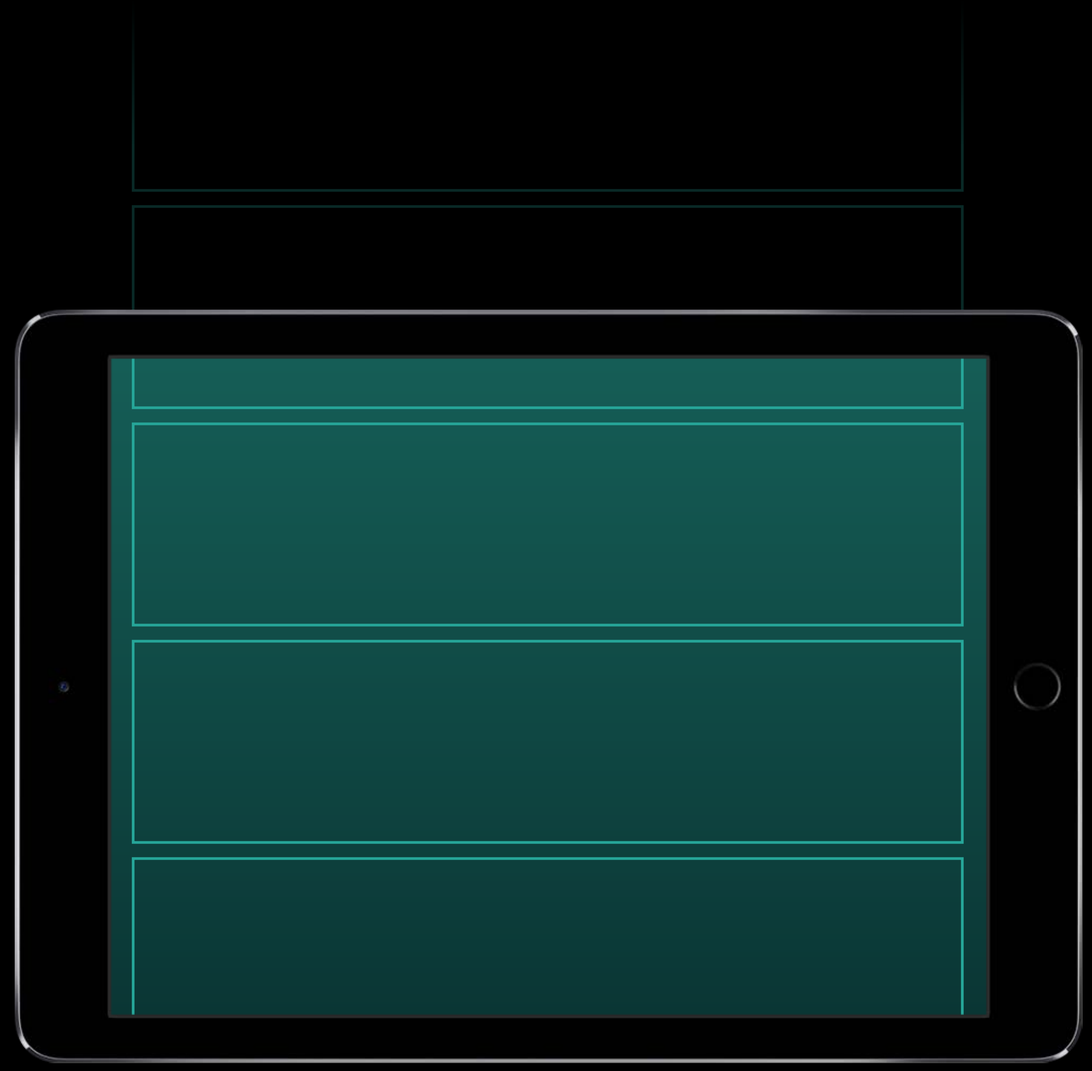


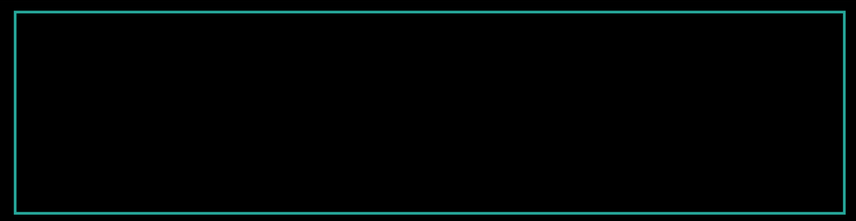
prepareForReuse

Life Cycle of a Cell

iOS 10

NEW



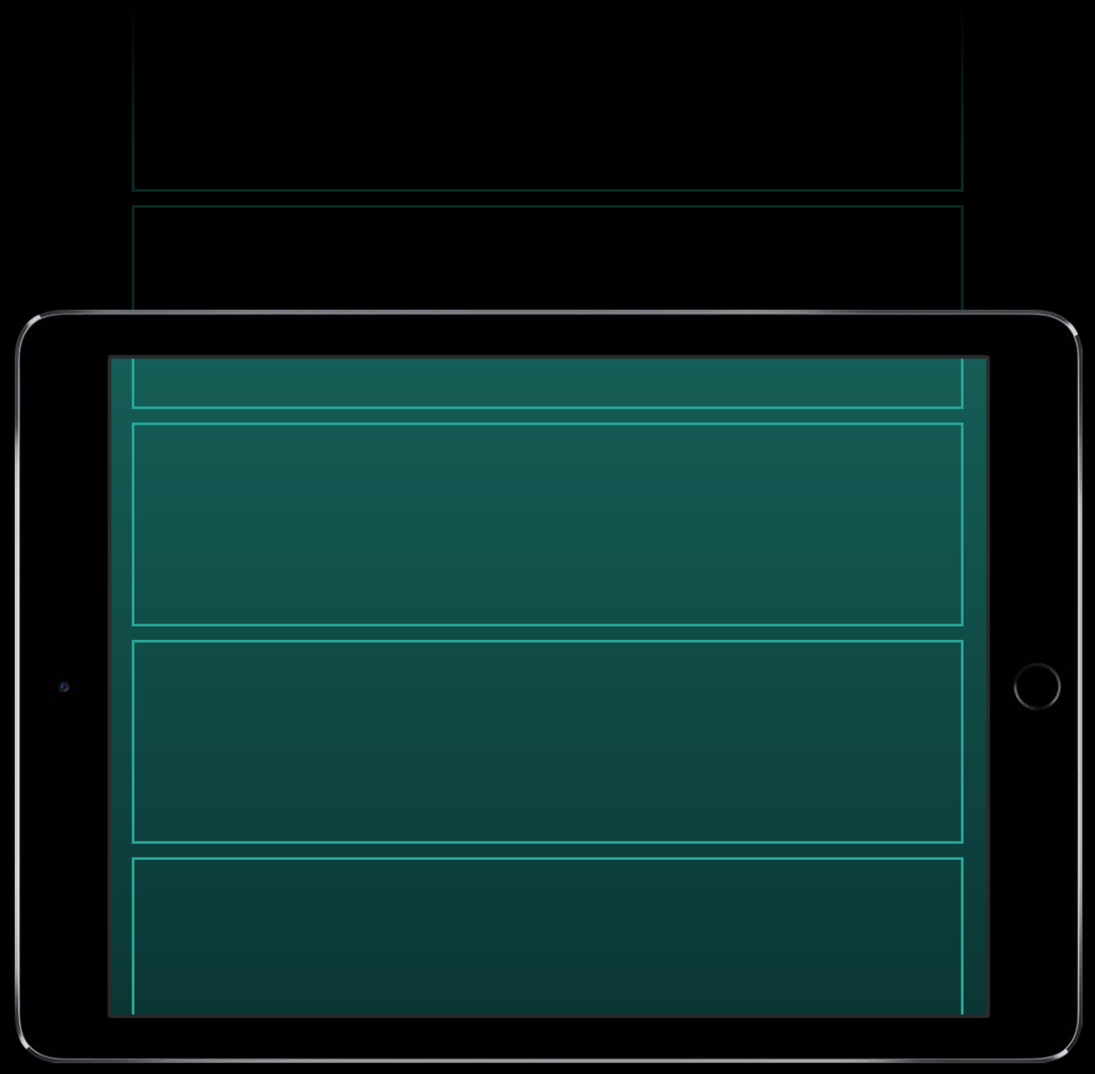


cellForItemAtIndexPath

Life Cycle of a Cell

iOS 10

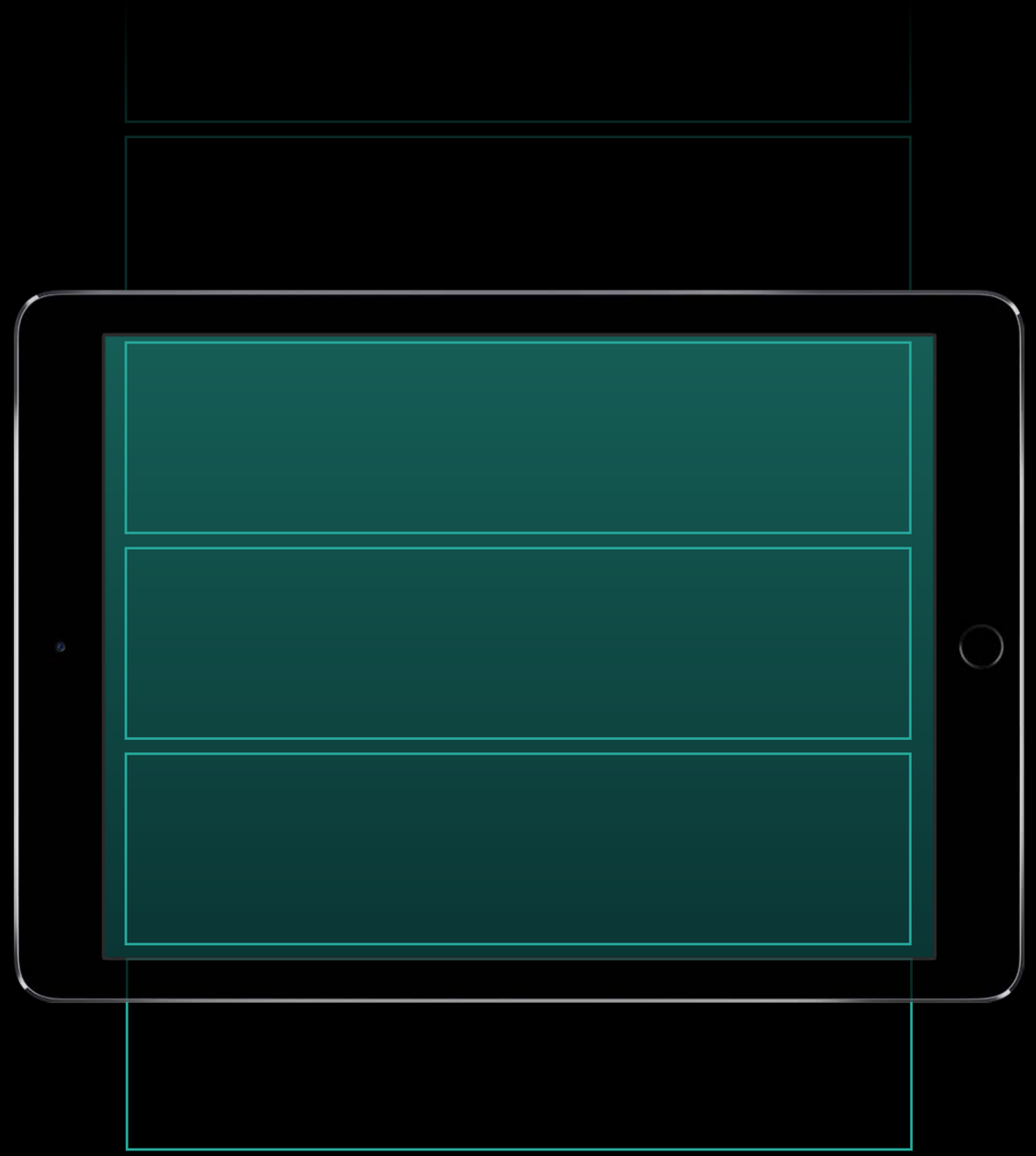
NEW



Life Cycle of a Cell

iOS 10

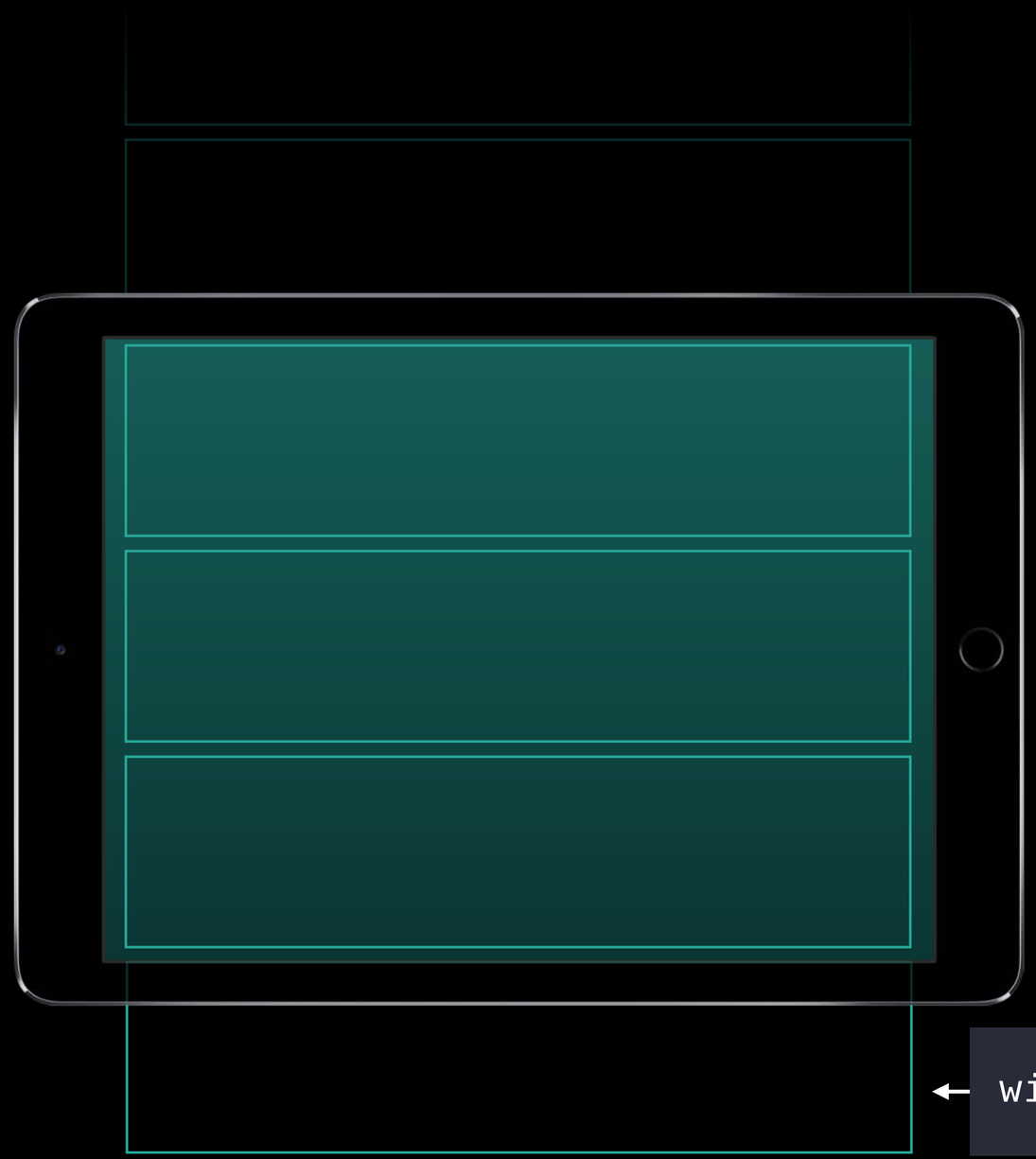
NEW



Life Cycle of a Cell

iOS 10

NEW

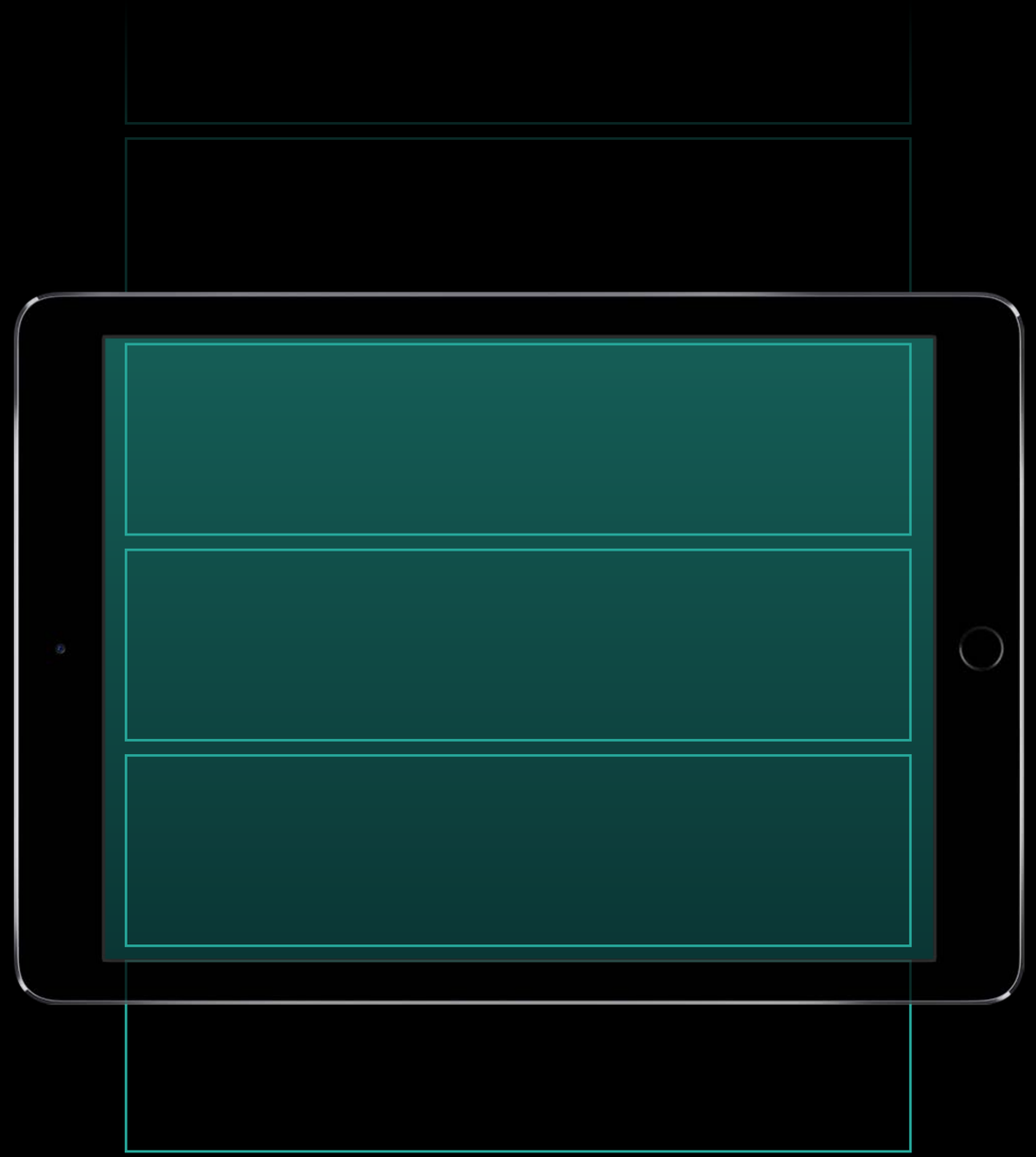


willDisplayCell

Life Cycle of a Cell

iOS 10

NEW



Life Cycle of a Cell

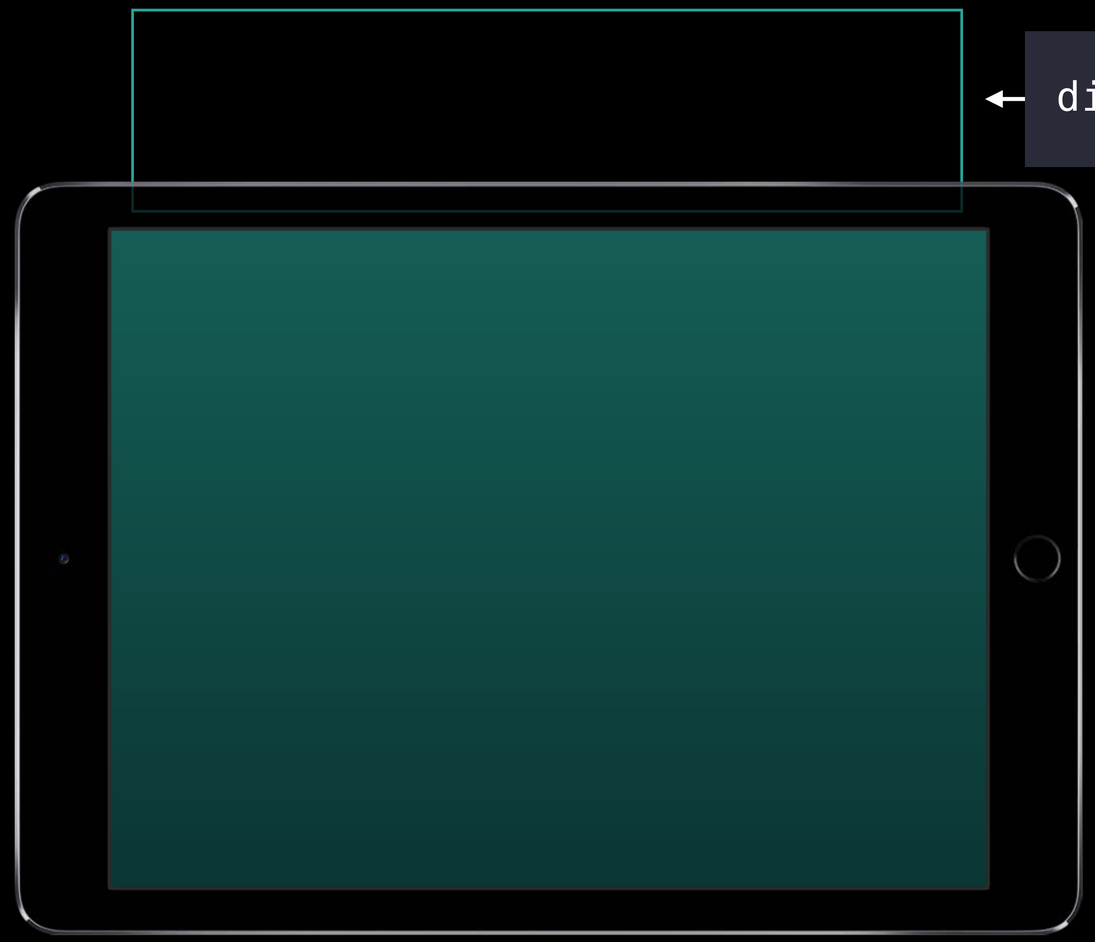
iOS 10

NEW

Life Cycle of a Cell

iOS 10

NEW



```
didEndDisplayingCell
```

Life Cycle of a Cell

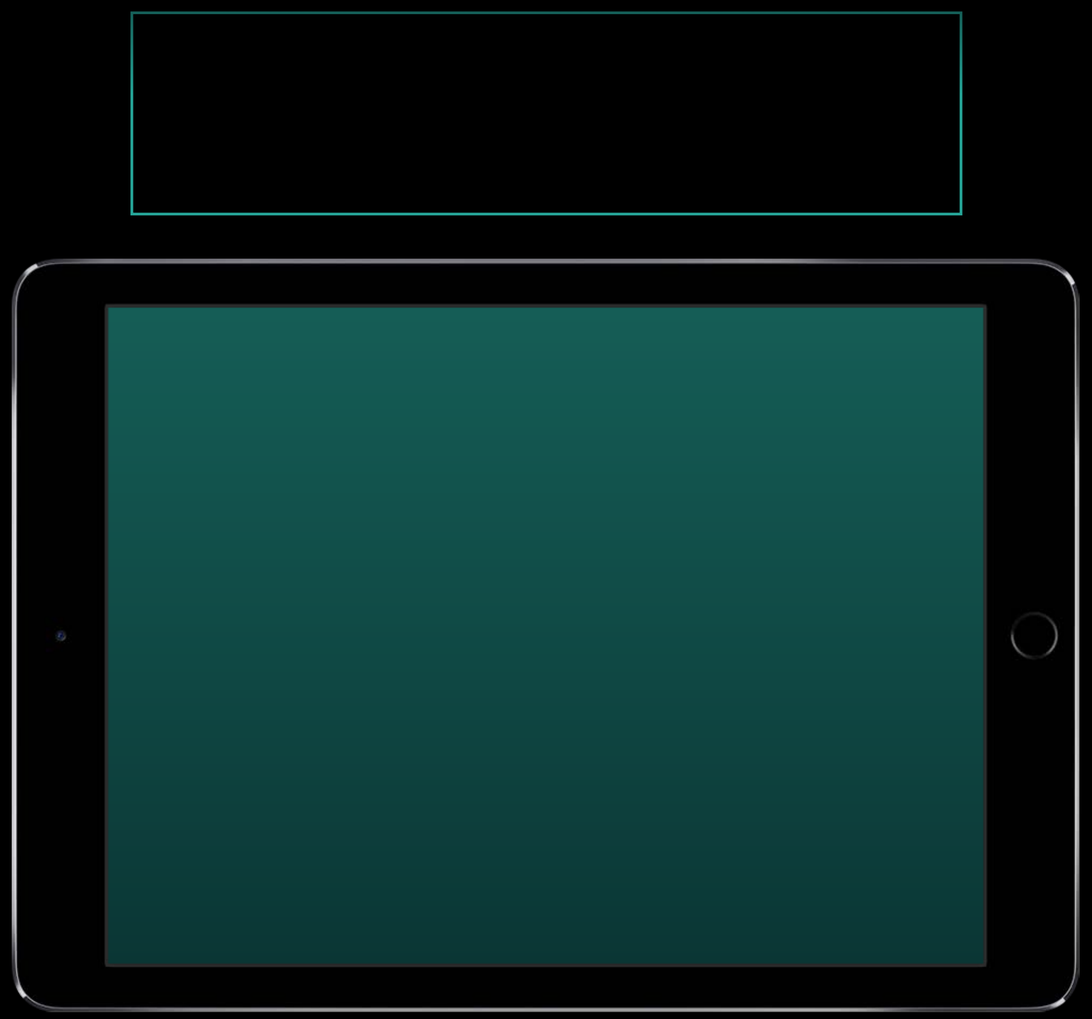
iOS 10

NEW

Life Cycle of a Cell

iOS 10

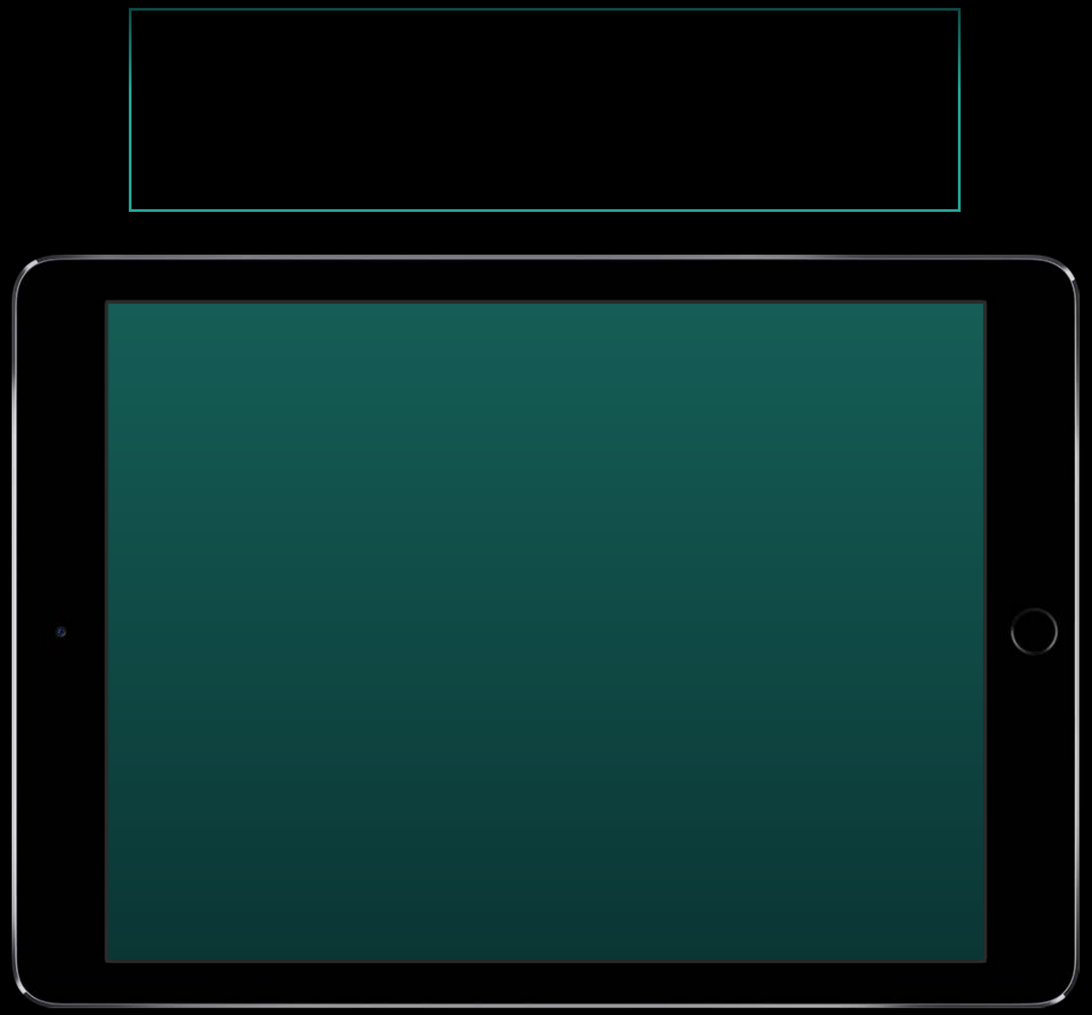
NEW



Life Cycle of a Cell

iOS 10

NEW



Life Cycle of a Cell

iOS 10

NEW

Life Cycle of a Cell

iOS 10

NEW

Life Cycle of a Cell

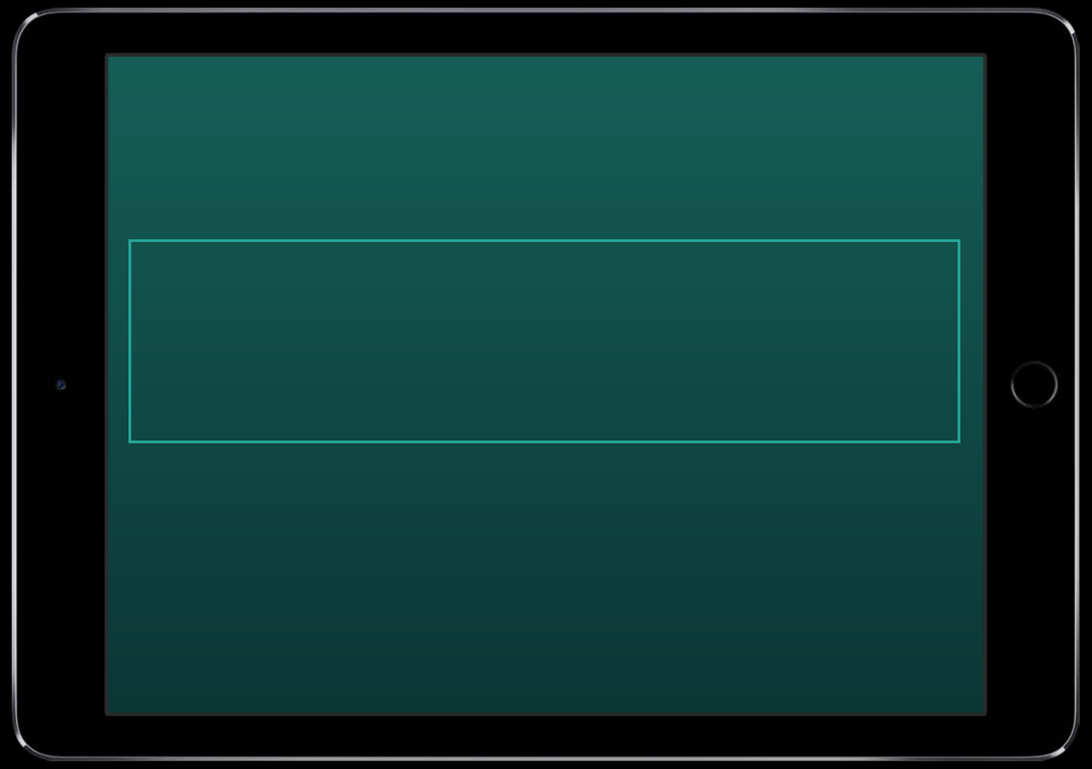
iOS 10

NEW

Life Cycle of a Cell

iOS 10

NEW



Life Cycle of a Cell

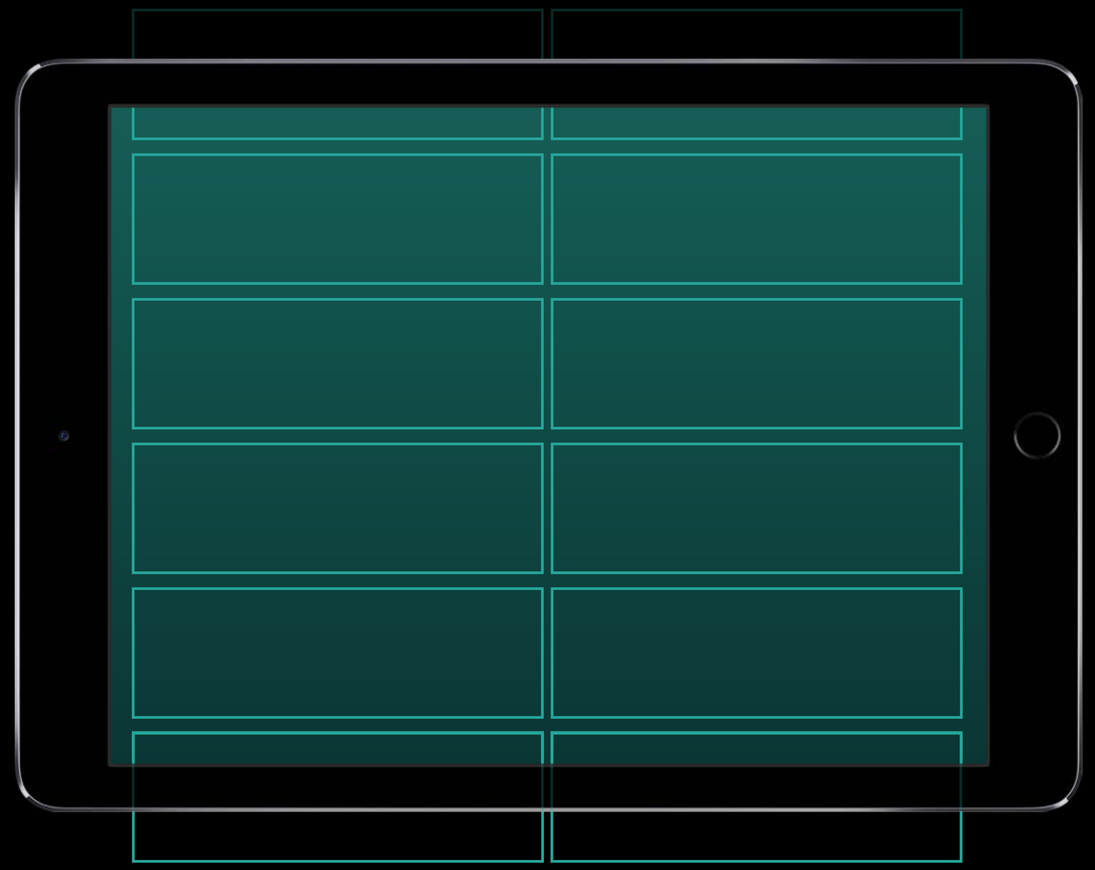
iOS 10

NEW

Life Cycle of a Cell

iOS 10

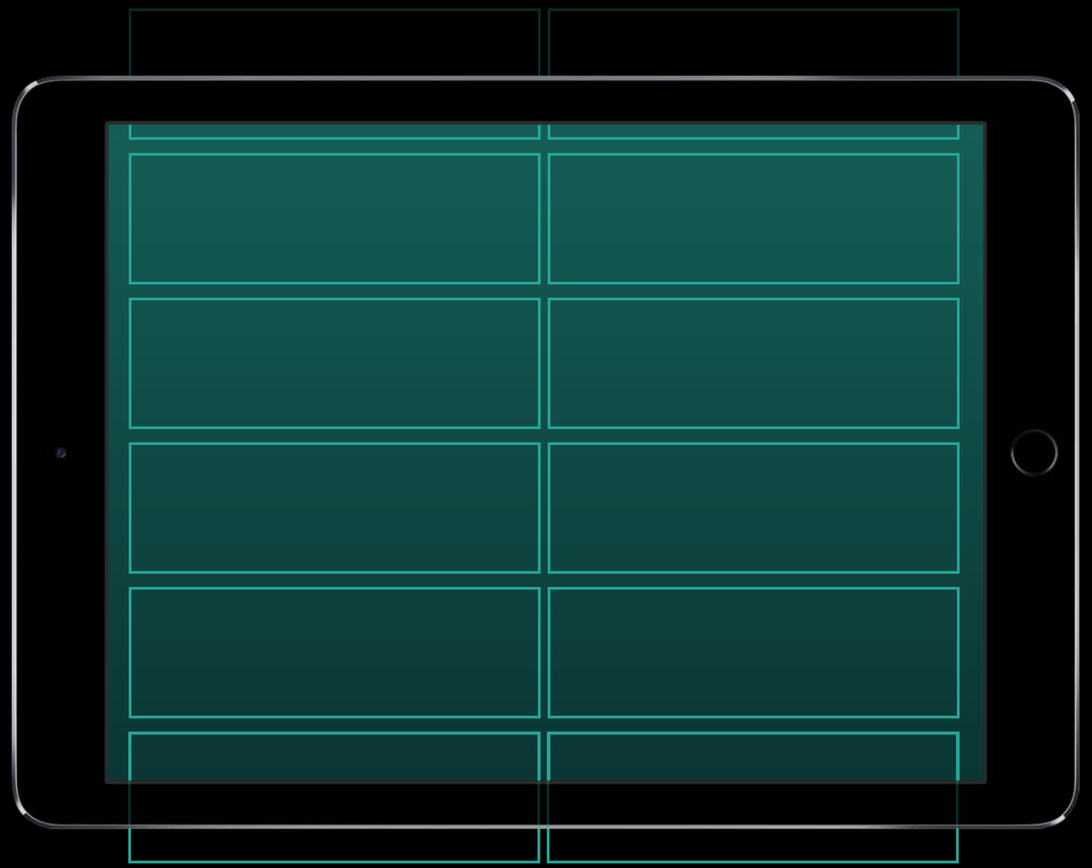
NEW



Life Cycle of a Cell

iOS 10

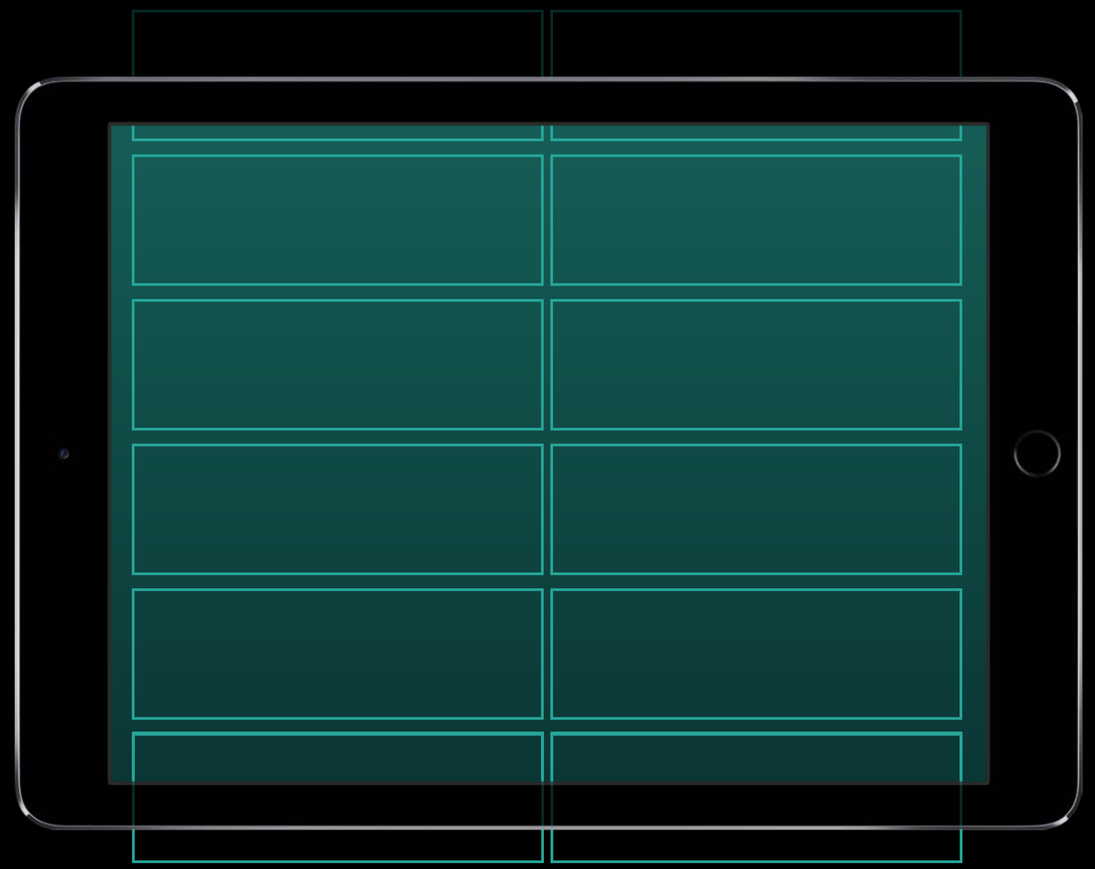
NEW



Life Cycle of a Cell

iOS 10

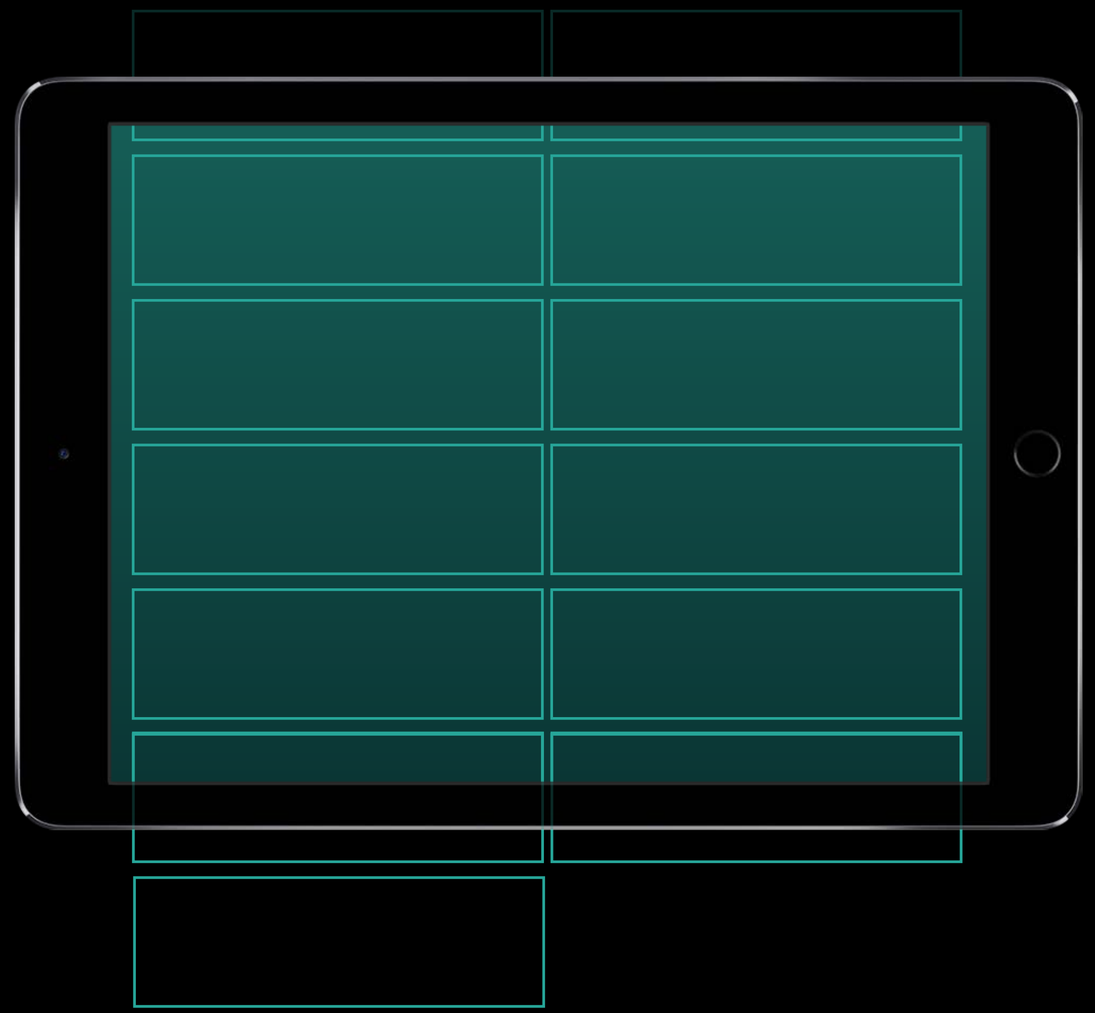
NEW



Life Cycle of a Cell

iOS 10

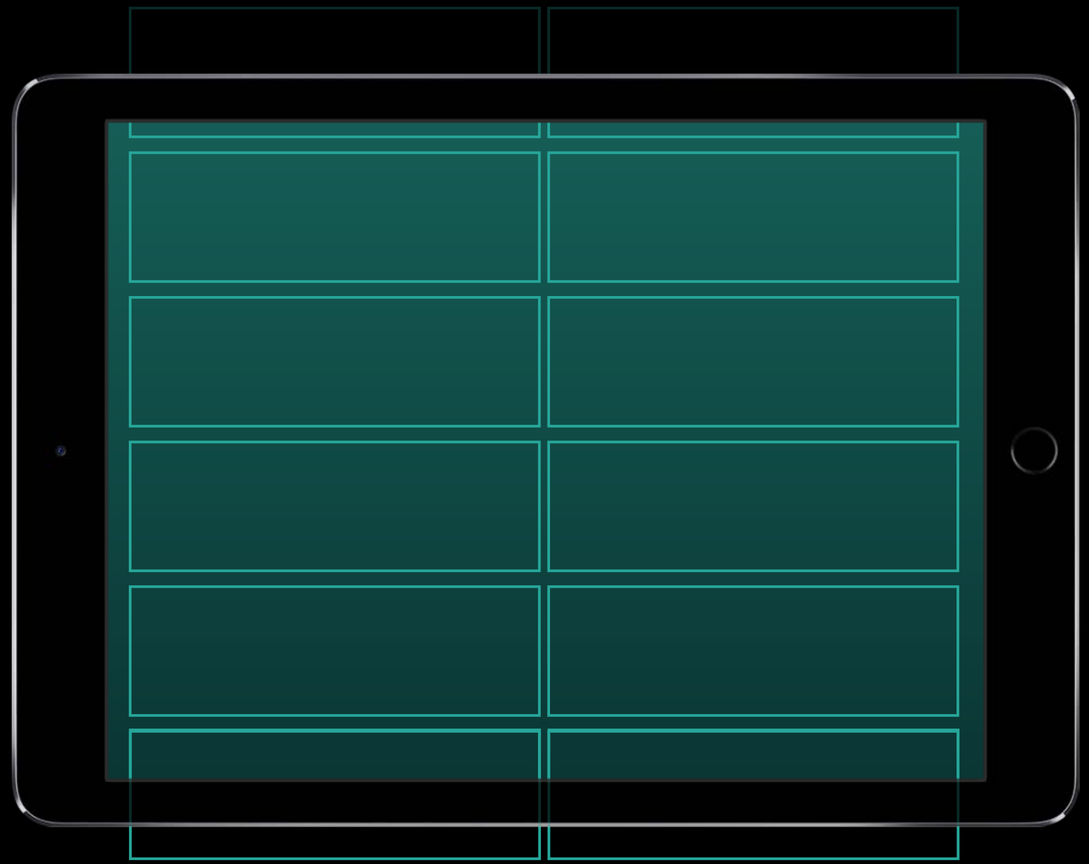
NEW



Life Cycle of a Cell

iOS 10

NEW

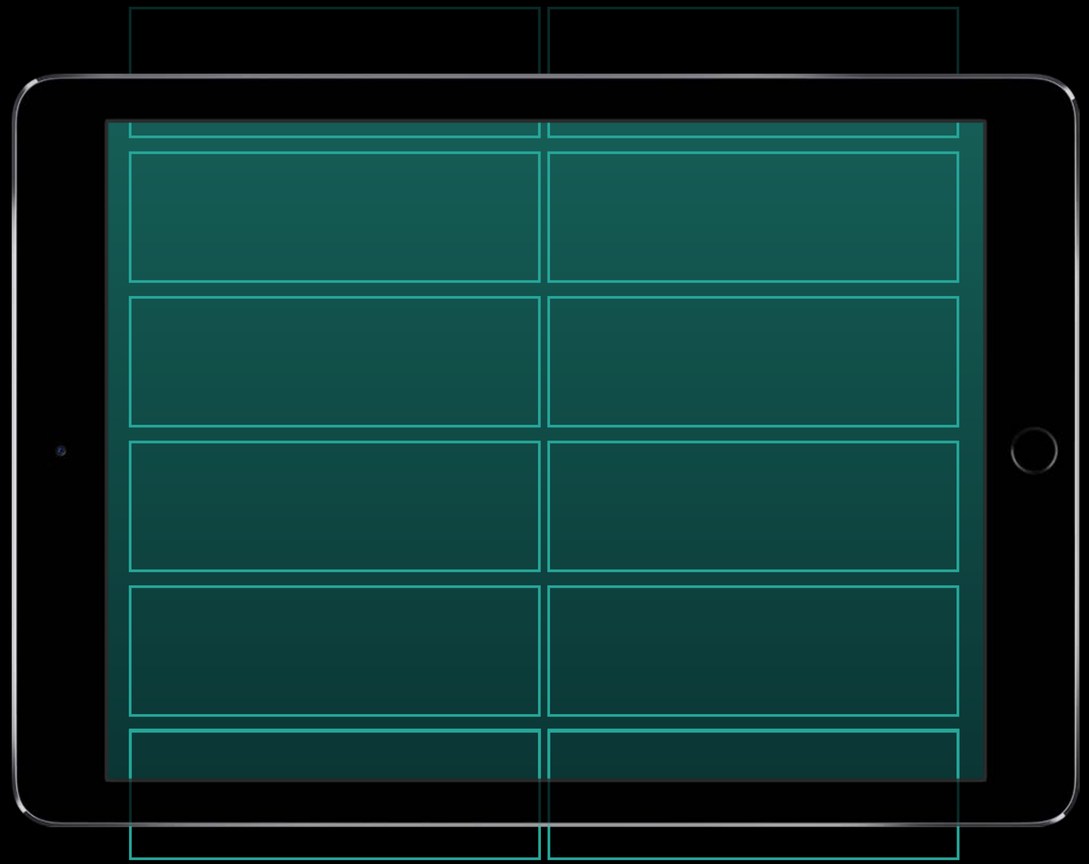


prepareForReuse

Life Cycle of a Cell

iOS 10

NEW

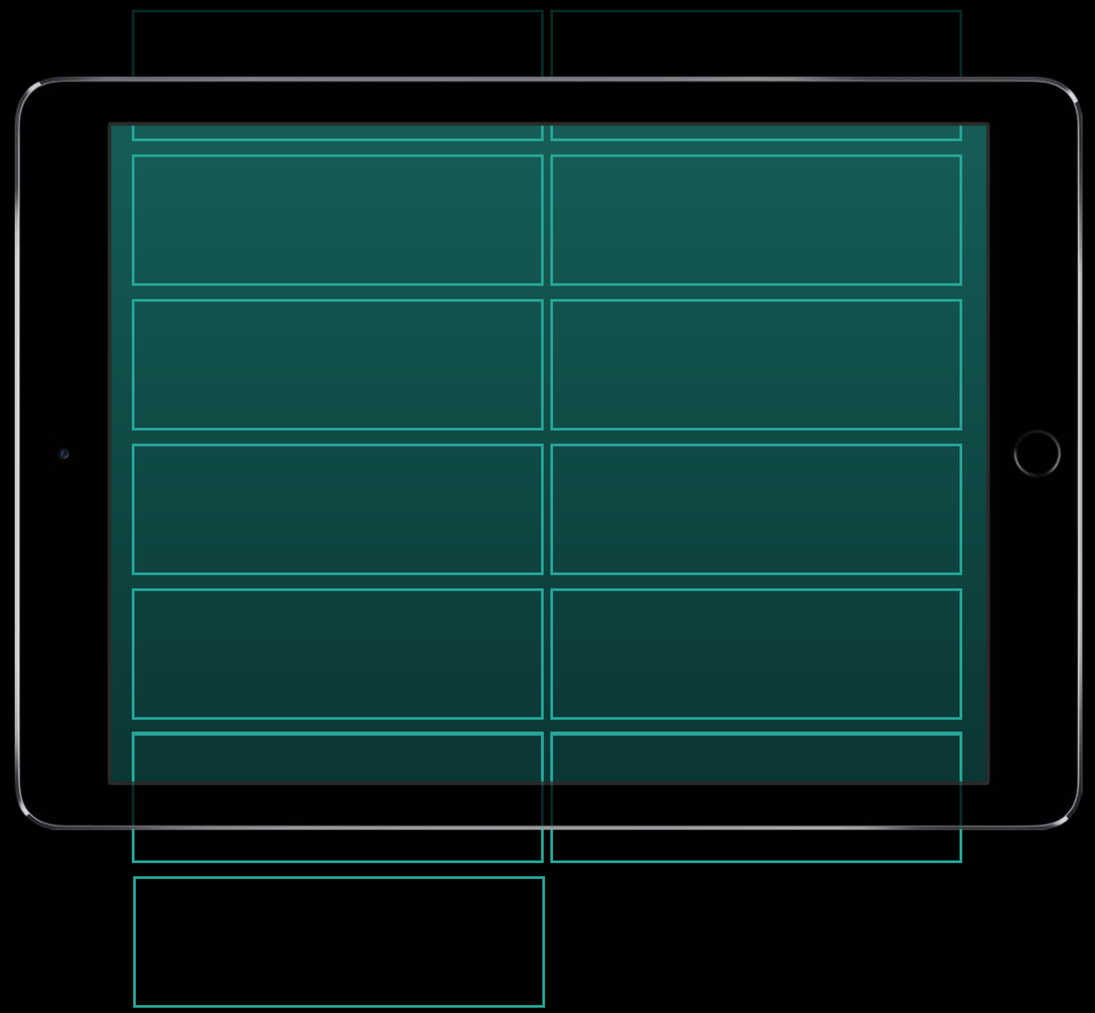


```
cellForItemAtIndexPath
```

Life Cycle of a Cell

iOS 10

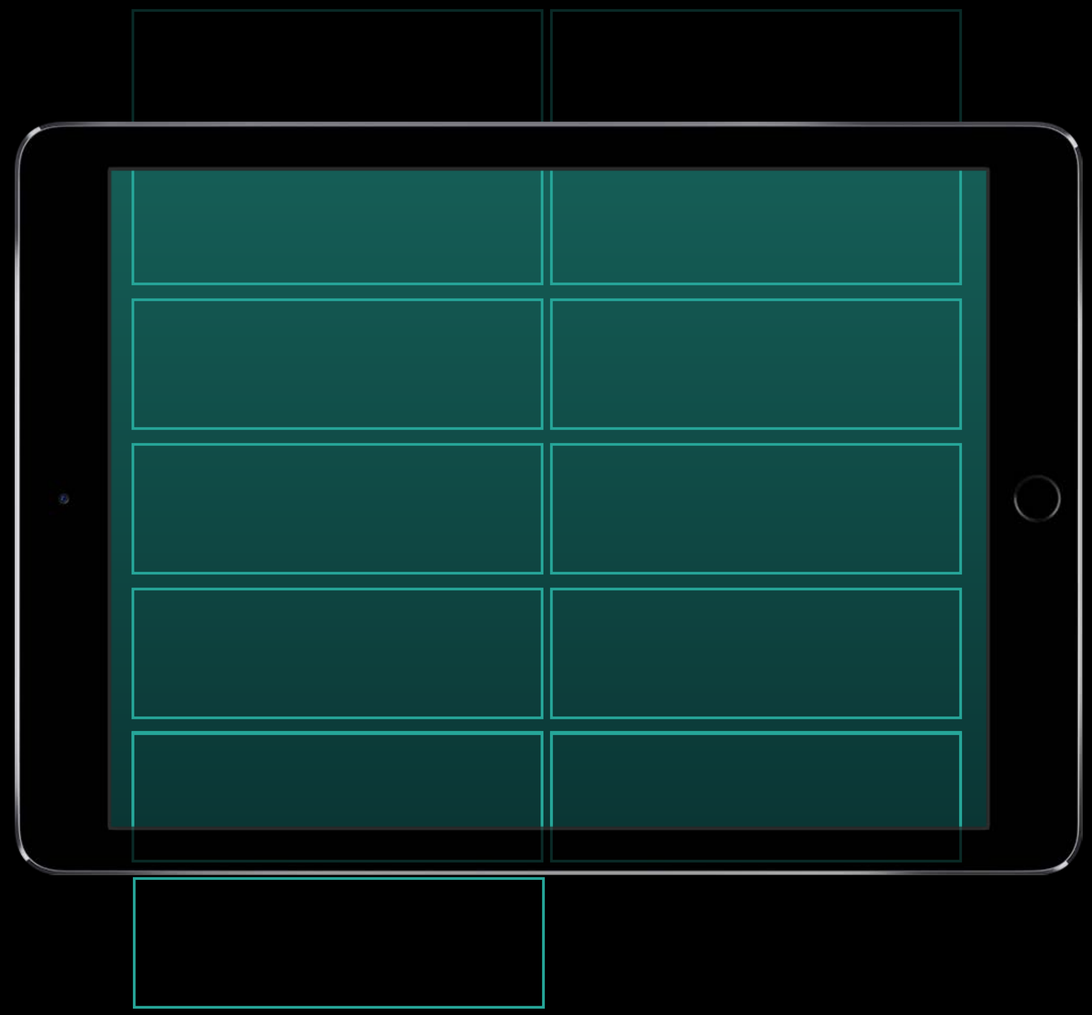
NEW



Life Cycle of a Cell

iOS 10

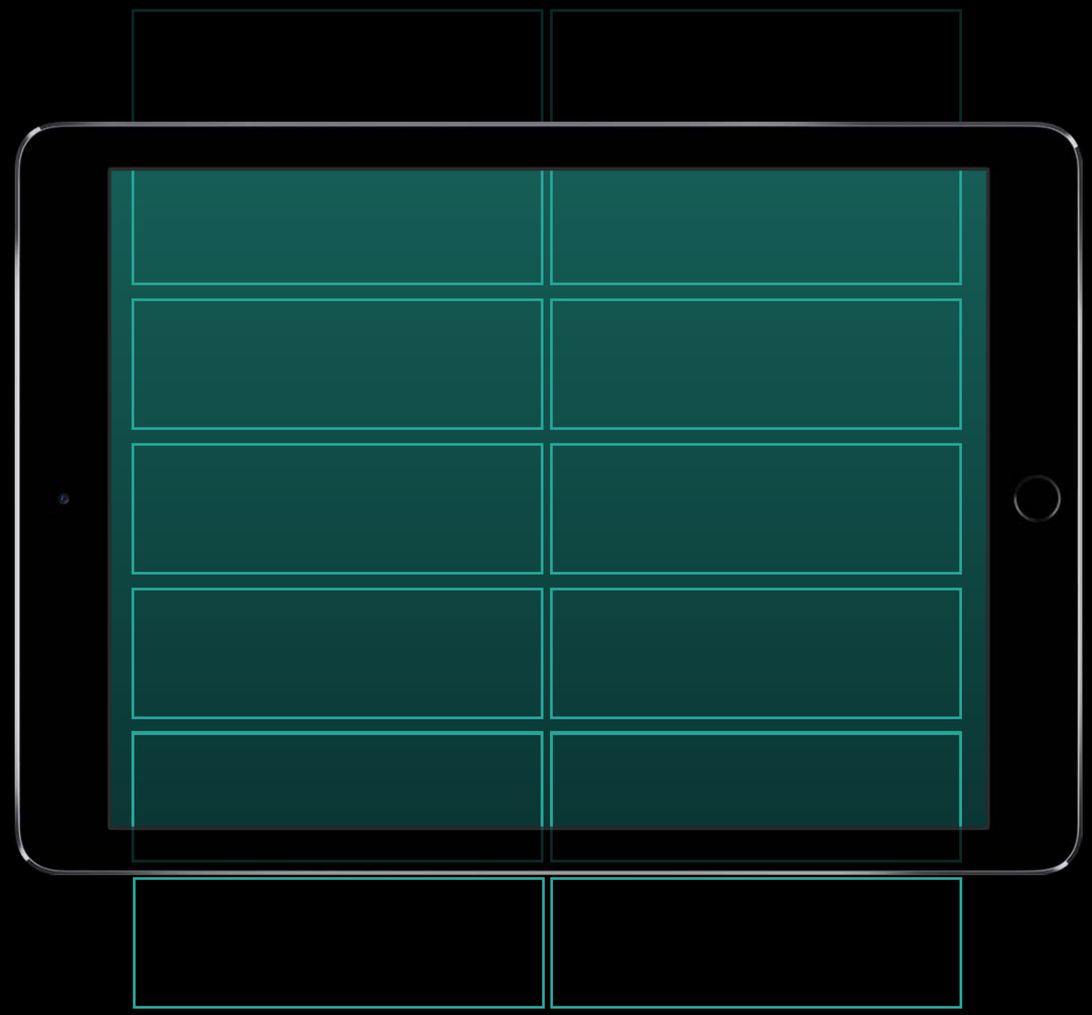
NEW



Life Cycle of a Cell

iOS 10

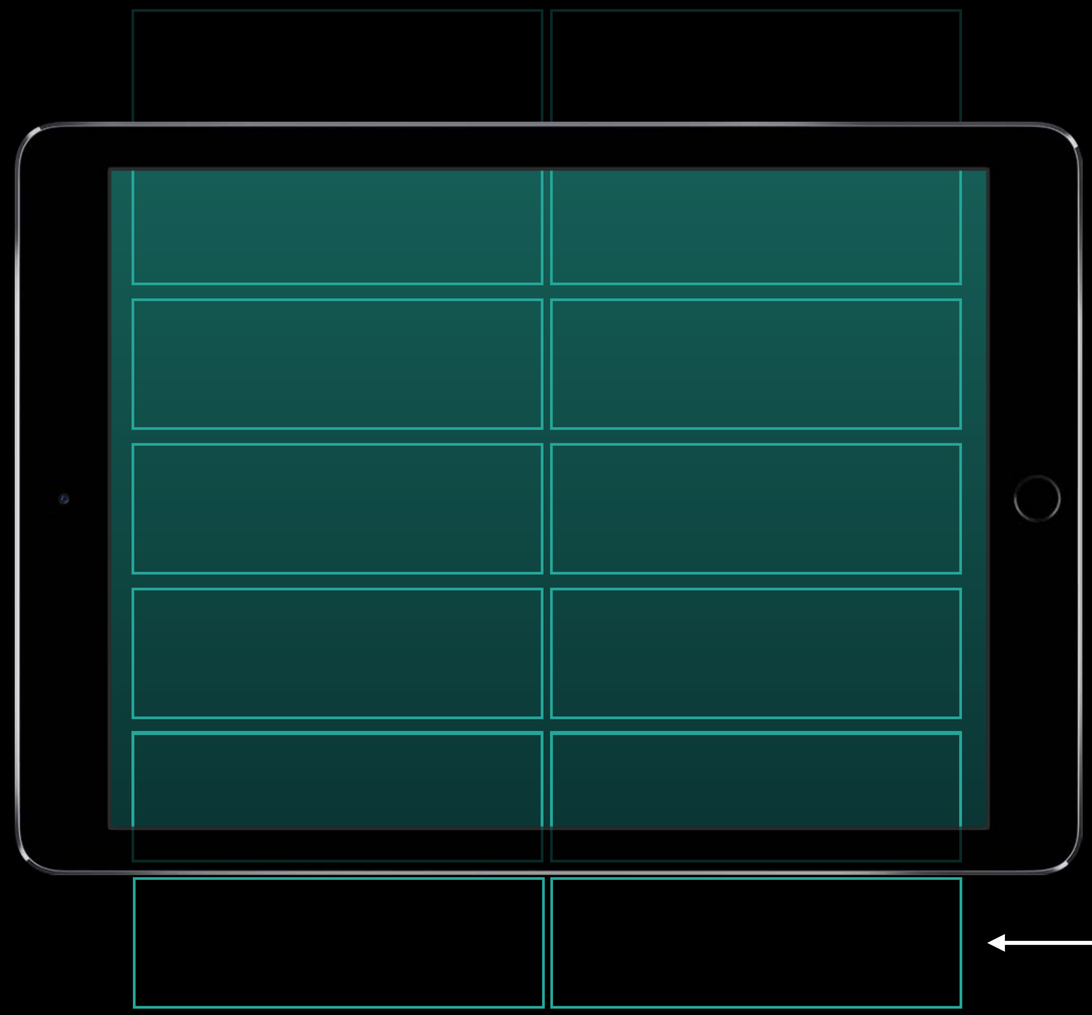
NEW



Life Cycle of a Cell

iOS 10

NEW

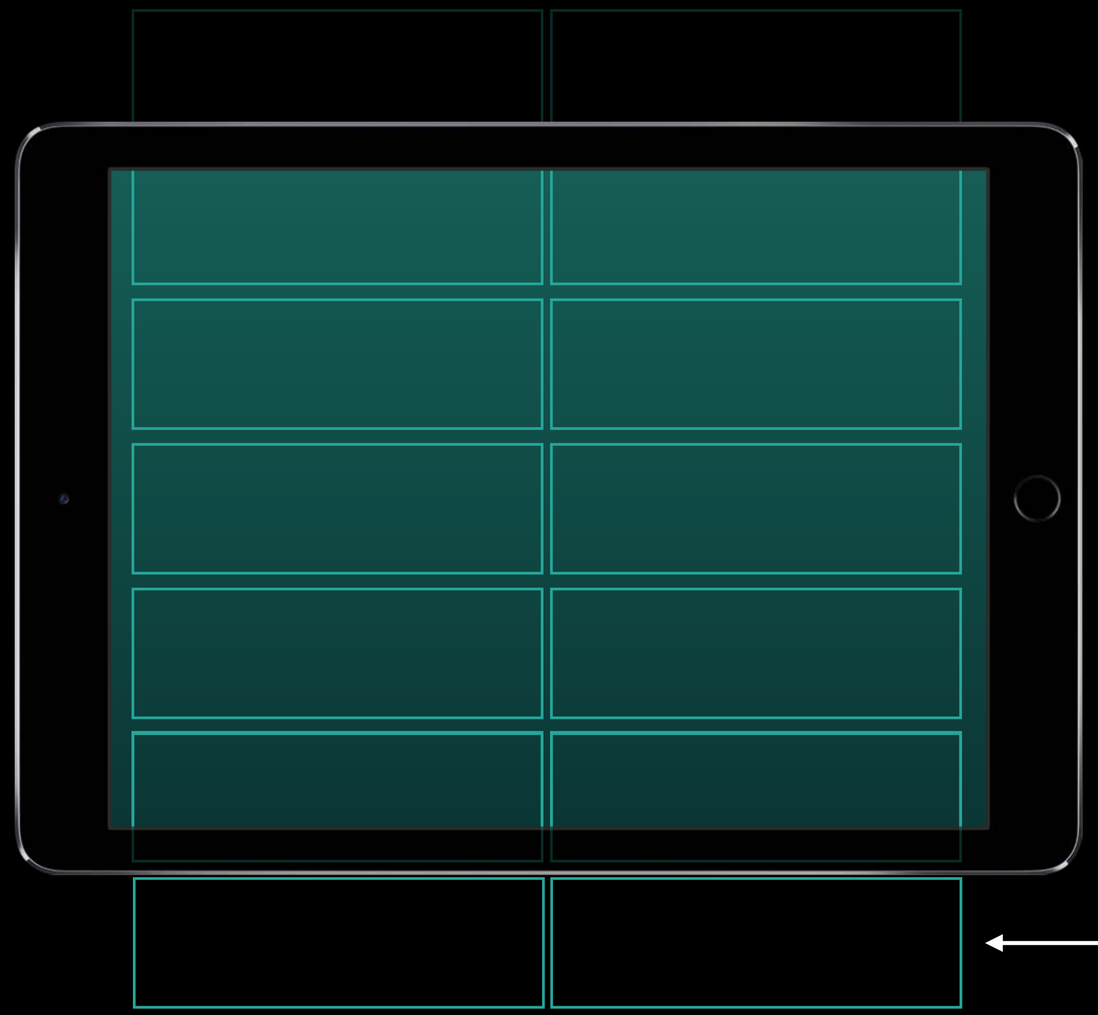


prepareForReuse

Life Cycle of a Cell

iOS 10

NEW

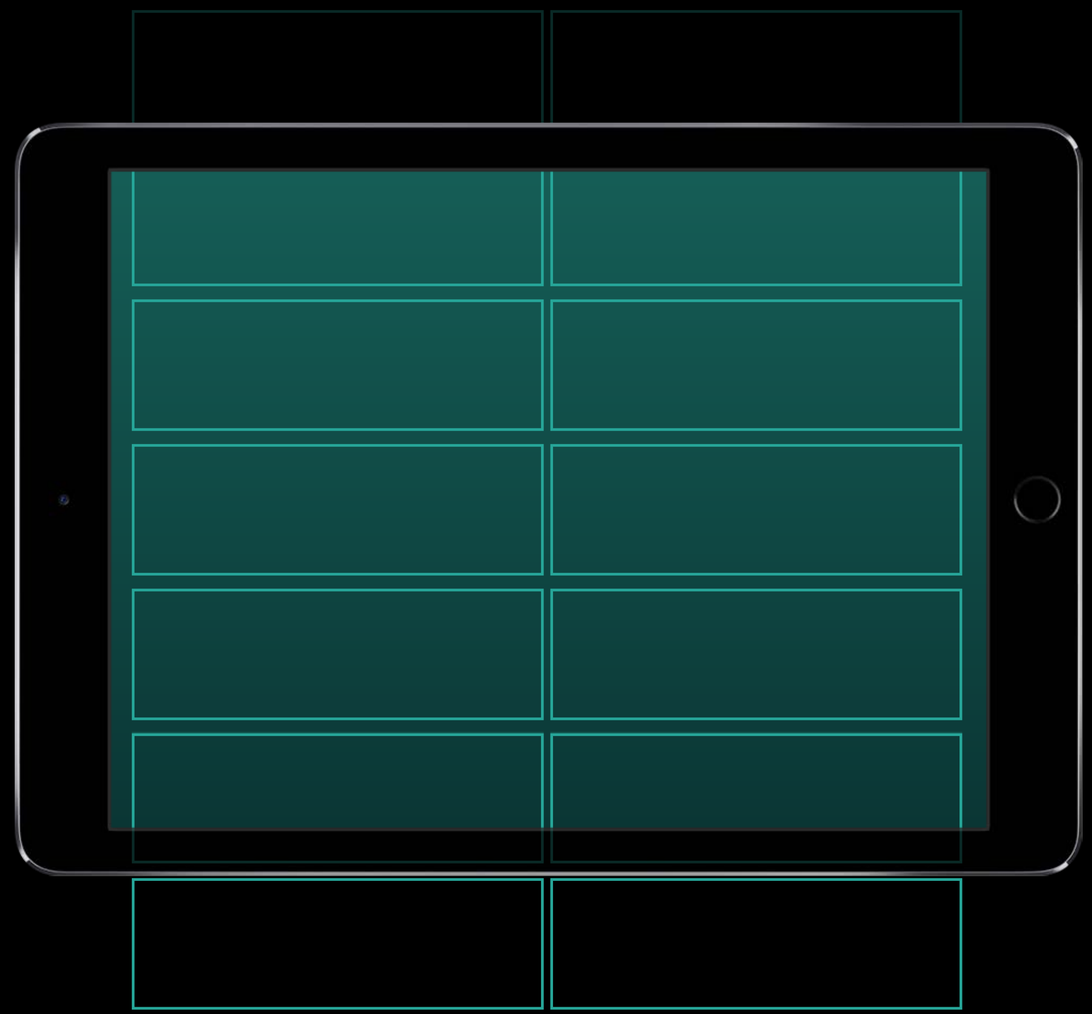


cellForItemAtIndexPath

Life Cycle of a Cell

iOS 10

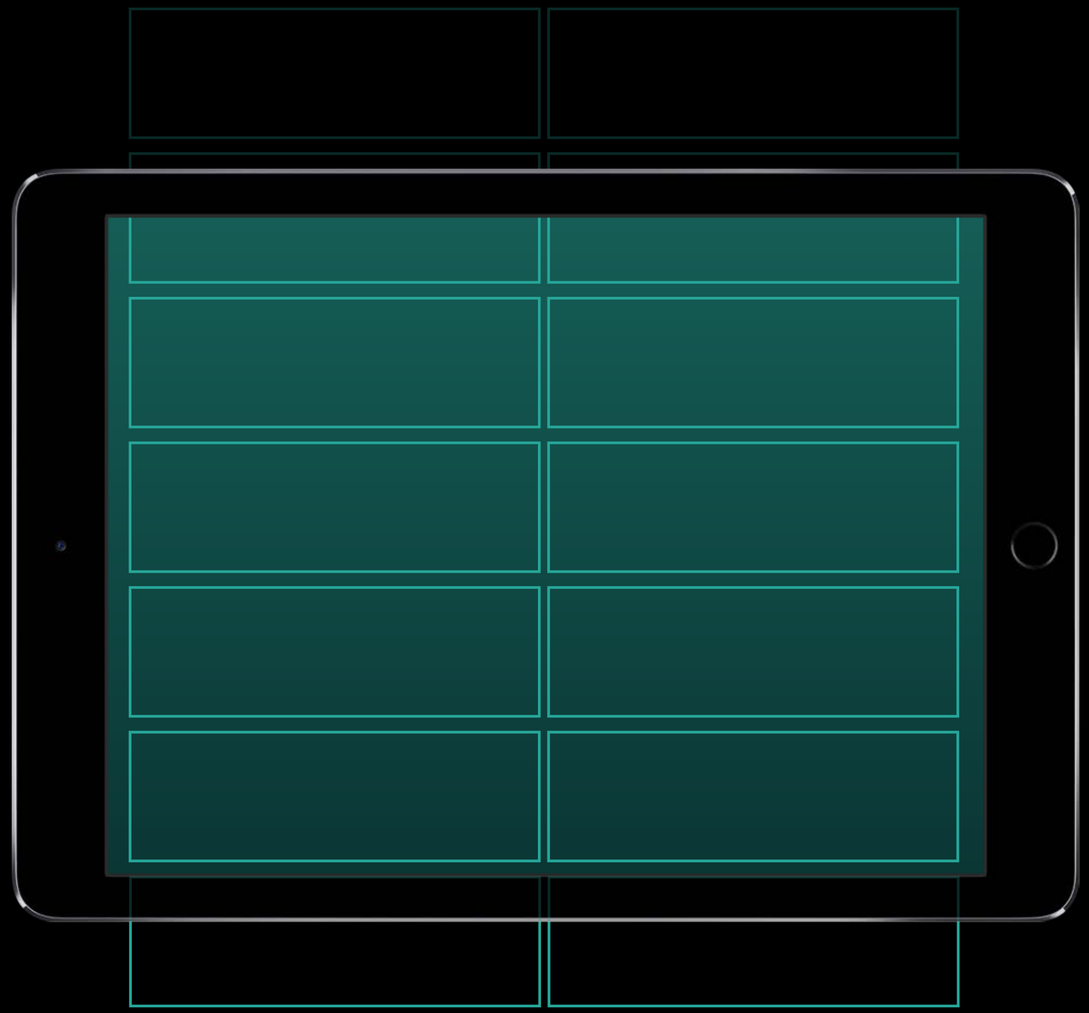
NEW



Life Cycle of a Cell

iOS 10

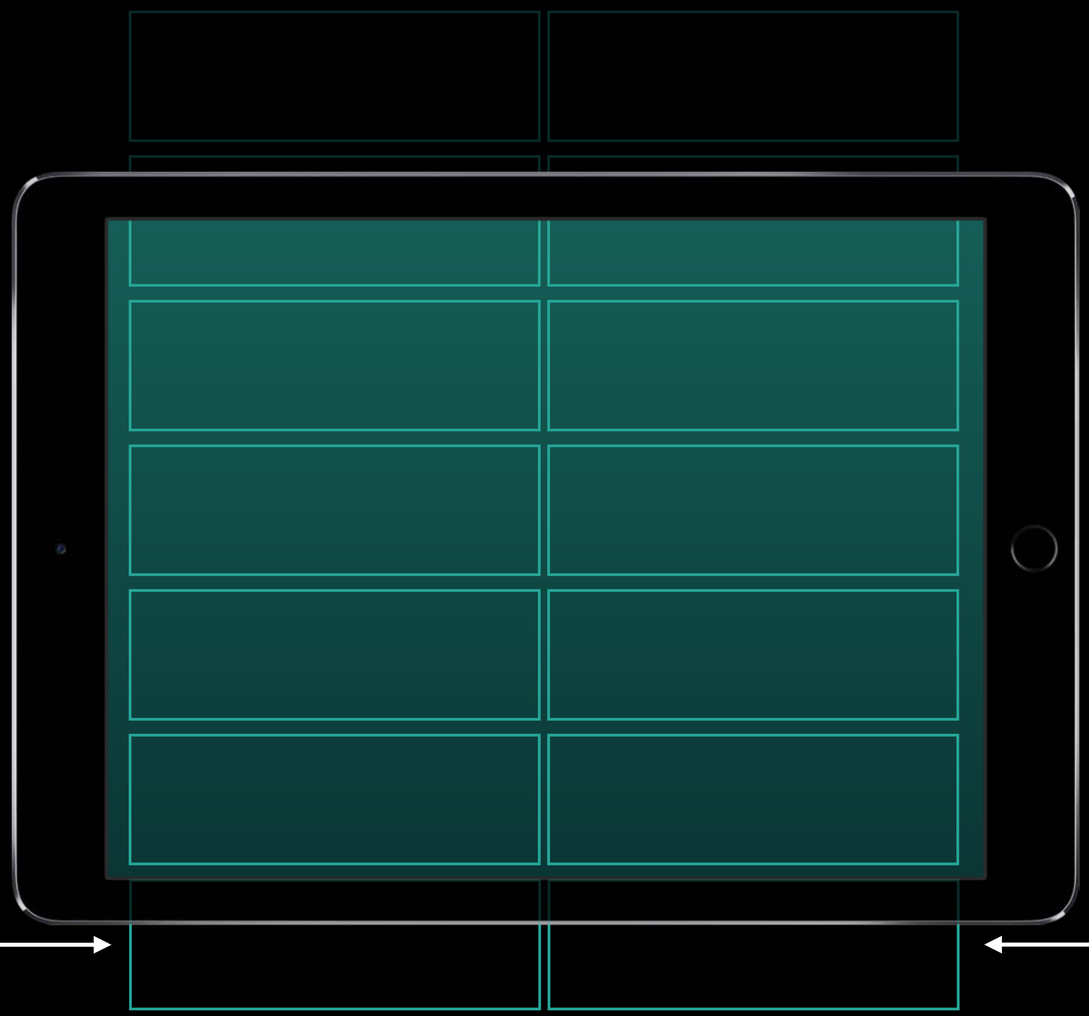
NEW



Life Cycle of a Cell

iOS 10

NEW



`willDisplayCell`

`willDisplayCell`

Demo

Scrolling like butter

Introducing Cell Pre-Fetching

NEW

Introducing Cell Pre-Fetching

NEW

Enabled by default for apps compiled on iOS 10

Introducing Cell Pre-Fetching

NEW

Enabled by default for apps compiled on iOS 10

There is no step one

Introducing Cell Pre-Fetching

NEW

Enabled by default for apps compiled on iOS 10

There is no step one

Opt-out is easy

```
collectionView.isPrefetchingEnabled = false
```

For Great Pre-Fetching Performance

For Great Pre-Fetching Performance

Do heavy lifting in `cellForItemAtIndexPath`

For Great Pre-Fetching Performance

Do heavy lifting in `cellForItemAtIndexPath`

Do minimal work in `willDisplayCell` / `didEndDisplayingCell`

For Great Pre-Fetching Performance

Do heavy lifting in `cellForItemAtIndexPath`

Do minimal work in `willDisplayCell` / `didEndDisplayingCell`

`cellForItemAtIndexPath` prepared cell may never display

What About Expensive Data Models?

What About Expensive Data Models?

Many cells require expensive data access

What About Expensive Data Models?

Many cells require expensive data access

Images, databases, Core Data

UICollectionView Pre-Fetching API

NEW

```
protocol UICollectionViewDataSourcePrefetching {
    func collectionView(_ collectionView: UICollectionView,
        prefetchItemsAt indexPaths: [NSIndexPath])
    optional func collectionView(_ collectionView: UICollectionView,
        cancelPrefetchingForItemsAt indexPaths: [NSIndexPath])
}
class UICollectionView : UIScrollView {
    weak var prefetchDataSource: UICollectionViewDataSourcePrefetching?
    var isPrefetchingEnabled: Bool
}
```

UICollectionView Pre-Fetching API

NEW

```
protocol UICollectionViewDataSourcePrefetching {
    func collectionView(_ collectionView: UICollectionView,
        prefetchItemsAt indexPaths: [NSIndexPath])
    optional func collectionView(_ collectionView: UICollectionView,
        cancelPrefetchingForItemsAt indexPaths: [NSIndexPath])
}
class UICollectionView : UIScrollView {
    weak var prefetchDataSource: UICollectionViewDataSourcePrefetching?
    var isPrefetchingEnabled: Bool
}
```

UICollectionView Pre-Fetching API

NEW

```
protocol UICollectionViewDataSourcePrefetching {
    func collectionView(_ collectionView: UICollectionView,
        prefetchItemsAt indexPaths: [NSIndexPath])
    optional func collectionView(_ collectionView: UICollectionView,
        cancelPrefetchingForItemsAt indexPaths: [NSIndexPath])
}
class UICollectionView : UIScrollView {
    weak var prefetchDataSource: UICollectionViewDataSourcePrefetching?
    var isPrefetchingEnabled: Bool
}
```

UICollectionView Pre-Fetching API

NEW

Works with your existing asynchronous solution

```
protocol UICollectionViewDataSourcePrefetching {
    func collectionView(_ collectionView: UICollectionView,
        prefetchItemsAt indexPaths: [NSIndexPath])
    optional func collectionView(_ collectionView: UICollectionView,
        cancelPrefetchingForItemsAt indexPaths: [NSIndexPath])
}
class UICollectionView : UIScrollView {
    weak var prefetchDataSource: UICollectionViewDataSourcePrefetching?
    var isPrefetchingEnabled: Bool
}
```

Demo

Putting it all together—with science!

Pre-Fetching API Tips

Pre-Fetching API Tips

Use GCD or NSOperationQueue

Pre-Fetching API Tips

Use GCD or NSOperationQueue

Pre-Fetching is an adaptive technology

Pre-Fetching API Tips

Use GCD or NSOperationQueue

Pre-Fetching is an adaptive technology

Use `cancelPrefetching` API to adapt to shifting user focus

UITableView Pre-Fetching API

NEW

```
protocol UITableViewDataSourcePrefetching {
   func tableView(_ tableView: UITableView, prefetchRowsAt indexPaths: [NSIndexPath])
   optional func tableView(_ tableView: UITableView, cancelPrefetchingForRowsAt indexPaths:
[NSIndexPath])
}
class UITableView : UIScrollView {
   weak var prefetchDataSource: UITableViewDataSourcePrefetching?
}
```

UITableView Pre-Fetching API

NEW

```
protocol UITableViewDataSourcePrefetching {
    func tableView(_ tableView: UITableView, prefetchRowsAt indexPaths: [NSIndexPath])
    optional func tableView(_ tableView: UITableView, cancelPrefetchingForRowsAt indexPaths:
[NSIndexPath])
}
class UITableView : UIScrollView {
    weak var prefetchDataSource: UITableViewDataSourcePrefetching?
}
```

UITableView Pre-Fetching API

NEW

```
protocol UITableViewDataSourcePrefetching {
    func tableView(_ tableView: UITableView, prefetchRowsAt indexPaths: [NSIndexPath])
    optional func tableView(_ tableView: UITableView, cancelPrefetchingForRowsAt indexPaths:
[NSIndexPath])
}
class UITableView : UIScrollView {
    weak var prefetchDataSource: UITableViewDataSourcePrefetching?
}
```

UITableView Pre-Fetching API

NEW

Works with your existing asynchronous solution

```
protocol UITableViewDataSourcePrefetching {
   func tableView(_ tableView: UITableView, prefetchRowsAt indexPaths: [NSIndexPath])
   optional func tableView(_ tableView: UITableView, cancelPrefetchingForRowsAt indexPaths:
[NSIndexPath])
}
class UITableView : UIScrollView {
   weak var prefetchDataSource: UITableViewDataSourcePrefetching?
}
```

Improvements to Self-Sizing Cells

Self-Sizing Cells API Overview

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

- `layout.estimatedItemSize = CGSize(width:50,height:50)`

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

- `layout.estimatedItemSize = CGSize(width:50,height:50)`

Three options for specifying actual cell size

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

- `layout.estimatedItemSize = CGSize(width:50,height:50)`

Three options for specifying actual cell size

- Auto Layout

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

- `layout.estimatedItemSize = CGSize(width:50,height:50)`

Three options for specifying actual cell size

- Auto Layout
- Override `sizeThatFits()`

Self-Sizing Cells API Overview

Full support in UICollectionViewFlowLayout

- `layout.estimatedItemSize = CGSize(width:50,height:50)`

Three options for specifying actual cell size

- Auto Layout
- Override `sizeThatFits()`
- Override `preferredLayoutAttributesFittingAttributes()`

Picking a Good estimatedItemSize

Picking a Good estimatedItemSize

Hard to guess

Picking a Good estimatedItemSize

Hard to guess

Can this be computed from actual sizes?

Automatic Self-Sizing Cells Estimates

NEW

Automatic Self-Sizing Cells Estimates

NEW

```
layout.estimatedItemSize = UICollectionViewFlowLayoutAutomaticSize
```

Automatic Self-Sizing Cells Estimates

NEW

```
layout.estimatedItemSize = UICollectionViewFlowLayoutAutomaticSize
```

Collection View will do the math for you

Demo

UICollectionViewFlowLayoutAutomaticSize

Interactive Reordering

Demo

Interactive Reordering

API Overview

```
class UICollectionView : UIScrollView {
    func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
    func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
    func endInteractiveMovement()
    func cancelInteractiveMovement()
}
```

API Overview

```
class UICollectionView : UIScrollView {
    func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
    func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
    func endInteractiveMovement()
    func cancelInteractiveMovement()
}
```

API Overview

```
class UICollectionView : UIScrollView {
    func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
    func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
    func endInteractiveMovement()
    func cancelInteractiveMovement()
}
```

API Overview

```
class UICollectionView : UIScrollView {
    func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
    func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
    func endInteractiveMovement()
    func cancelInteractiveMovement()
}
```

API Overview

```
class UICollectionView : UIScrollView {
    func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
    func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
    func endInteractiveMovement()
    func cancelInteractiveMovement()
}
```

API Overview

```
class UICollectionView : UIScrollView {
   func beginInteractiveMovementForItem(at indexPath: NSIndexPath) -> Bool
   func updateInteractiveMovementTargetPosition(_ targetPosition: CGPoint)
   func endInteractiveMovement()
   func cancelInteractiveMovement()
}
```

```
class UICollectionViewController : UIViewController {
   var installsStandardGestureForInteractiveMovement: Bool
}
```

Paging Support

Paging Support

```
collectionView.isPagingEnabled = true
```

Demo

Interactive reordering—with paging support

NEW

Demo

Interactive reordering—with paging support

JORDAN

UIRefreshControl!

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIScrollView too!

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIScrollView too!

```
let refreshControl = UIRefreshControl()

refreshControl.addTarget(self, action: #selector(refreshControlDidFire(_:)),
for: .valueChanged)

collectionView.refreshControl = refreshControl
```

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIScrollView too!

```
let refreshControl = UIRefreshControl()

refreshControl.addTarget(self, action: #selector(refreshControlDidFire(_:)),
for: .valueChanged)

collectionView.refreshControl = refreshControl
```

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIScrollView too!

```
let refreshControl = UIRefreshControl()

refreshControl.addTarget(self, action: #selector(refreshControlDidFire(_:)),
for: .valueChanged)

collectionView.refreshControl = refreshControl
```

UIRefreshControl!

UIRefreshControl now directly supported in UICollectionView!

UITableView too!

UIScrollView too!

```
let refreshControl = UIRefreshControl()

refreshControl.addTarget(self, action: #selector(refreshControlDidFire(_:)),
for: .valueChanged)

collectionView.refreshControl = refreshControl
```

Summary

Summary

UICollectionView cell pre-fetching

Summary

UICollectionView cell pre-fetching

UICollectionView and UITableView prefetchDataSource API

Summary

UICollectionView cell pre-fetching

UICollectionView and UITableView prefetchDataSource API

Improvements to self-sizing cells

Summary

UICollectionView cell pre-fetching

UICollectionView and UITableView prefetchDataSource API

Improvements to self-sizing cells

Interactive reordering

More Information

https://developer.apple.com/wwdc16/219

Related Sessions

Advances in UIKit Animations and Transitions	Pacific Heights	Wednesday 10:00AM
What's New in Core Data	Pacific Heights	Friday 10:00 AM
Concurrent Programming with GCD in Swift 3	Pacific Heights	Friday 4:00 PM

Labs

UIKit and UIKit Animations Lab	Frameworks Lab C	Thursday 1:00 PM
Cocoa Touch and 3D Touch Lab	Frameworks Lab C	Friday 10:30 AM

WWDC16