Tech Pulse

High-Performance Scrolling on iOS Using Layout Models

Jason Howlin, iOS Engineer, Mobile Shared Tech

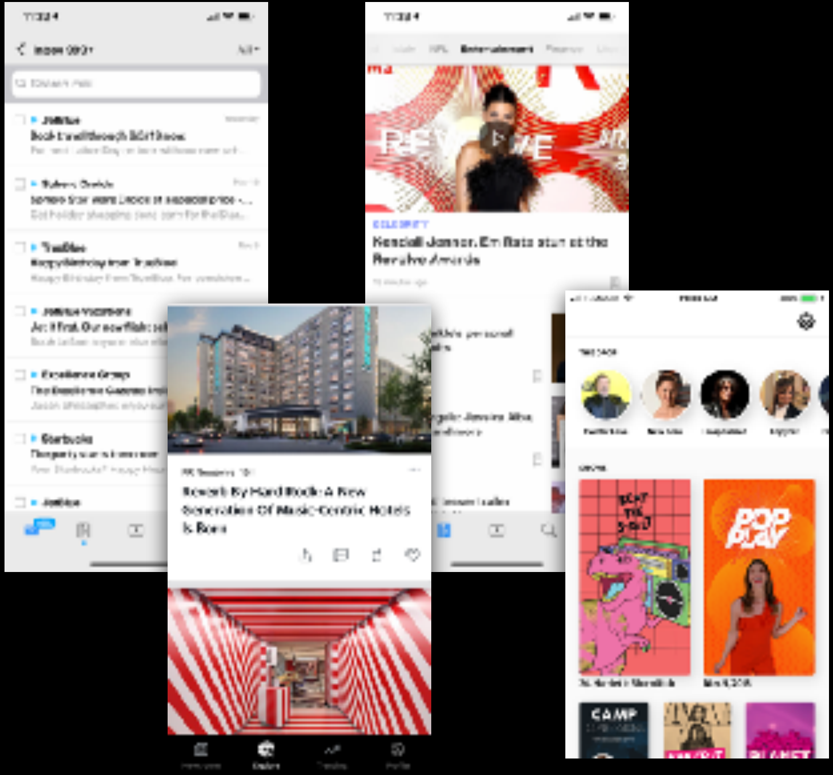
Shamal Nikam, iOS Engineer, Mobile Shared Tech

Overview

- How to improve table view scrolling problems
- Separate the layout from the drawing of view
- Perform layout work off of the main thread
- Perform layout work once and cache the results

Action!

All the Table Views!

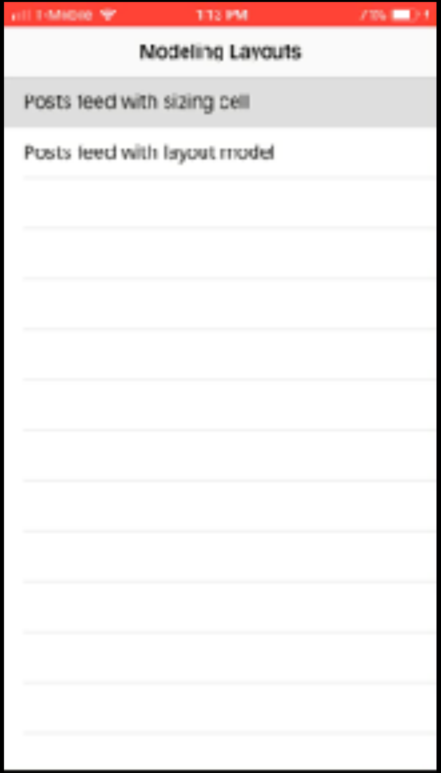


Kendall Jenner, Em Rata stun at the Revolve Awards

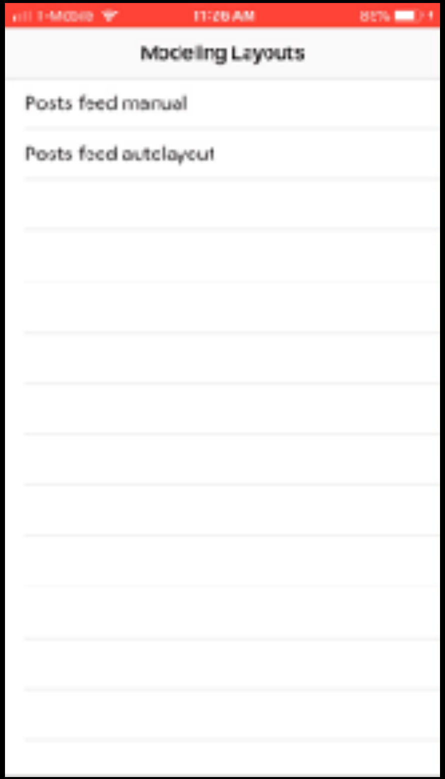
Rework By Hard Rock: A New Generation Of Music-Centric Hotels Is Born

What’s the Problem ?

- Poor scrolling performance of long lists of content with variable height cells
- 55 fps, sometimes dips to 40 fps



What it *should* look like…



Modeling Layouts

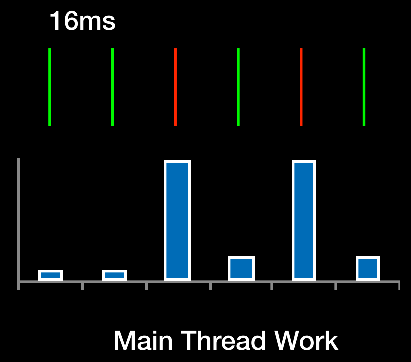
Posts feed manual

Posts feed autolayout

Tech Pulse

Goal: 60 FPS

- Update UI every 16 ms
- When main thread is busy, we cannot update the UI, and drop a frame
- While scrolling dropping a frame will cause a stutter



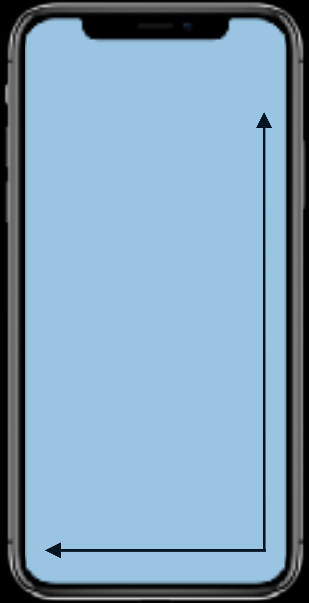
Why is the tableview stuttering ?

- Displaying large number of items
- Taking too long to calculate heights of cells
- Taking too long to render each item

Tech Pulse

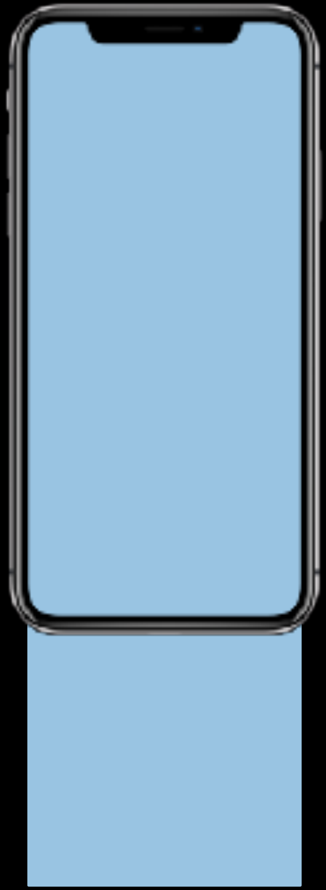
Understanding UIScrollView

- Content size > screen size
- What is the content size?



UITableView

- Content size width constrained to screen width
- What is the content size height?

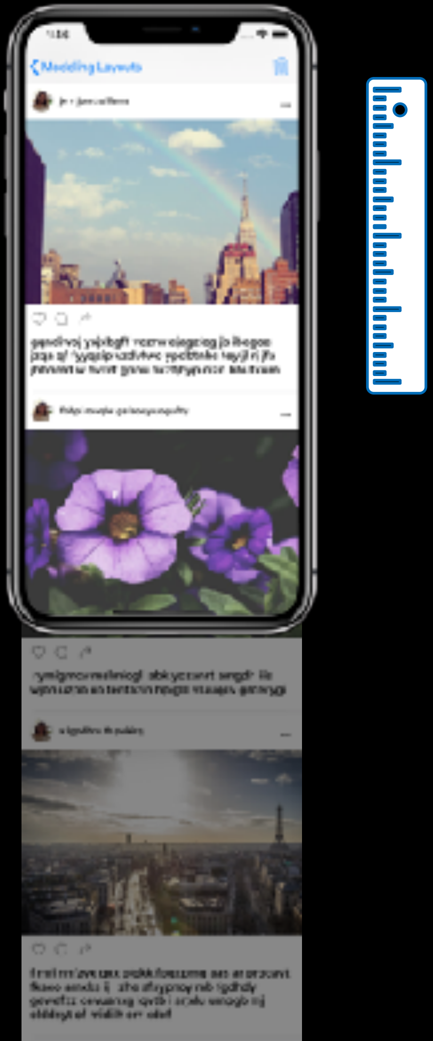


Calculating Height

- Visit each cell to ask for height
- Sum of all cell heights is the content size height

Calculating Height

- Visit each cell to ask for height
- Sum of all cell heights is the content size height
- But how do we get the height ?



Calculating Height: Using a Sizing Cell

```
func heightForRowAtIndexPath…
{
    let post = posts[indexPath.row]

    sizingCell.setPost(post: post)

    let height = sizingCell.sizeThatFits(…).height

   return height

 }
```

Writing a Class Function

```
 class func heightForPostConstrainedToWidth…
{
          var offset = CGPoint(x: 10, y: 10)

          let availableWidth = width - 10 - 10

          let headline = NSAttributedString(string: article.headline)

          let headlineHeight = headline.boundingRect...

          return headlineHeight
}
```

Layout Subviews

```
override func layoutSubviews()
{
    super.layoutSubviews()

    var offset = CGPoint(x: 10, y: 10)

    let headlineWidth = bounds.size.width - 10 - 10

    let headlineHeight = headlineLabel.sizeThatFits(CGSize(width: headlineWidth...)

    headlineLabel.frame = CGRect(x: offset.x, y: offset.y,
                                 width: headlineWidth, height:headlineHeight)

 }
```

Table View Data Source Hot Spots

```
func tableView(_ tableView: UITableView,
    cellForRowAt indexPath: IndexPath) -> UITableViewCell {

}

func tableView(_ tableView: UITableView,
  heightForRowAt indexPath: IndexPath) -> CGFloat {

}
```

Table View Data Source Hot Spots

	Number of Times Method Called (iOS 12)
Height For Row on Initial Load	51
Height For Row Scroll All 300 Items	1200

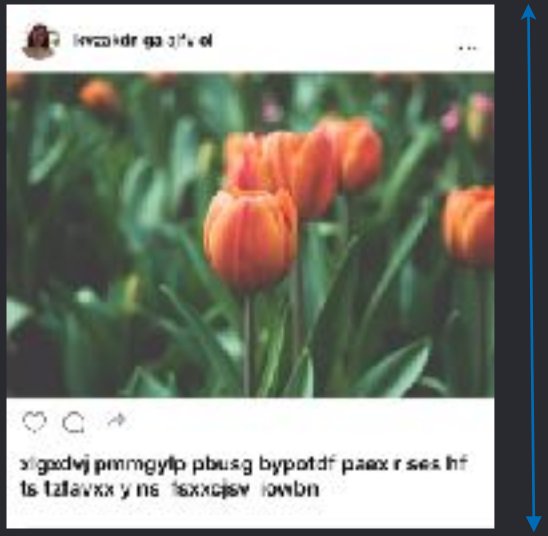
How can we be fast in these hot spots?

Layout Model Pattern

- Design pattern for efficient rendering of content in UITableView and UICollectionView
- Model layouts as data independent of UIView objects
- Layout models provide size and position for a view’s subviews, without rendering the view
- Can be calculated off of the main thread

From Data Model to View

```
struct PostDataModel {

    let headline:String
    let imageURL:String
    let userName:String
    let date:Date
    let avatarURL:String

}
```



Height is the function of data model

Creating a View Model

```
struct PostDataModel {
    let headline:String
    let imageURL:String
    let userName:String
    let date:Date
    let avatarURL:String
}
```

```
class PostViewModel {
    let headlineAttrString:NSAttributedString
    let imageURL:URL
    let userNameAttrString:NSAttributedString
    let date:NSAttributedString
    let avatarURL:URL
}
```

PostDataModel

headline - Tech Pulse
imageURL - http://image.png
userName - user123
date - 2018-12-12 14:17:39 +0000
avatarURL - http://avatar.png

View Model

- Contains formatted data.
- In the past this might have been done when setting the data model on the view.

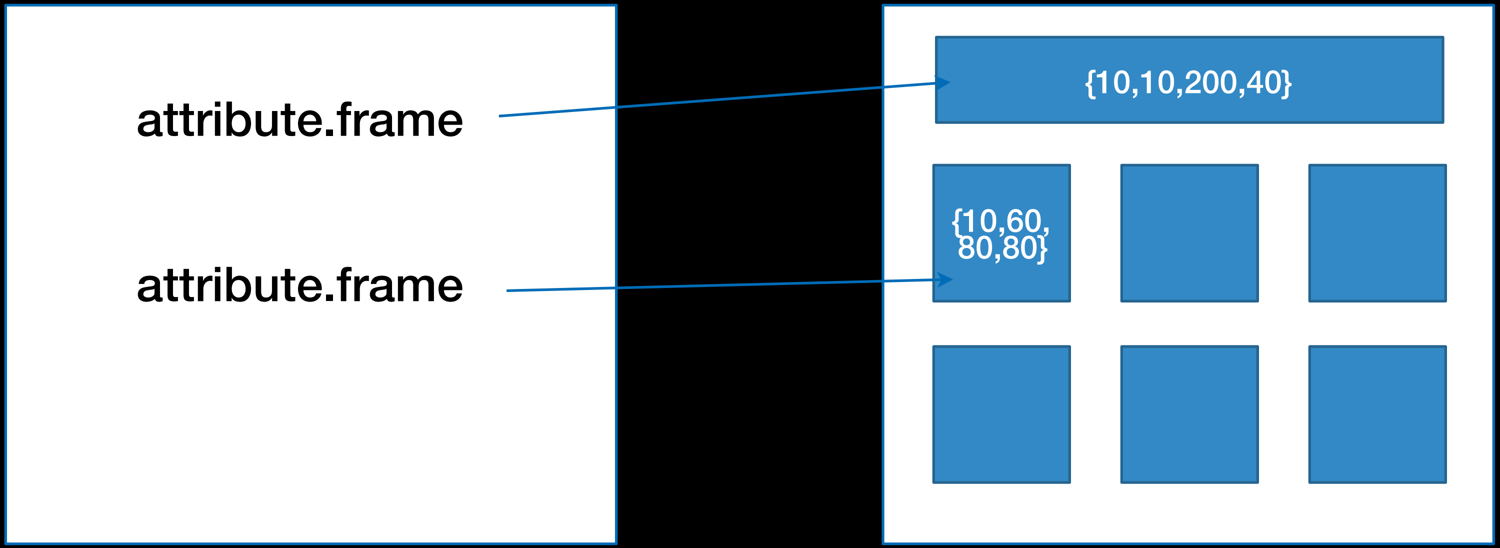
Layout Model

- Frames, frames and frames!

```
struct PostViewLayoutModel {
    var avatarImageFrame:CGRect
    var userNameLabelFrame:CGRect
    var headlineLabelFrame: CGRect
    var dateLabelFrame:CGRect
}
```

Collection View Layout

Collection View



attribute.frame

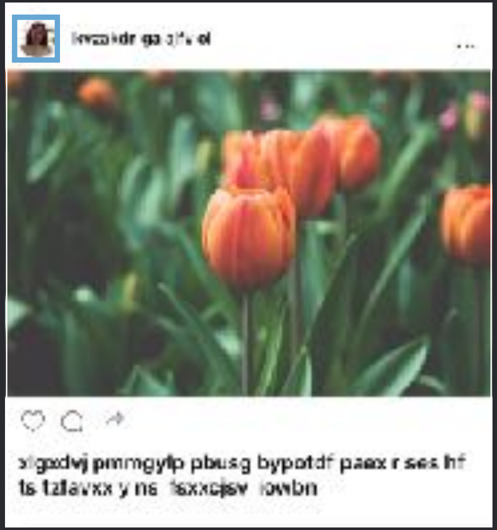
attribute.frame

{10,10,200,40}

{10,60,
80,80}

Building the Layout Model

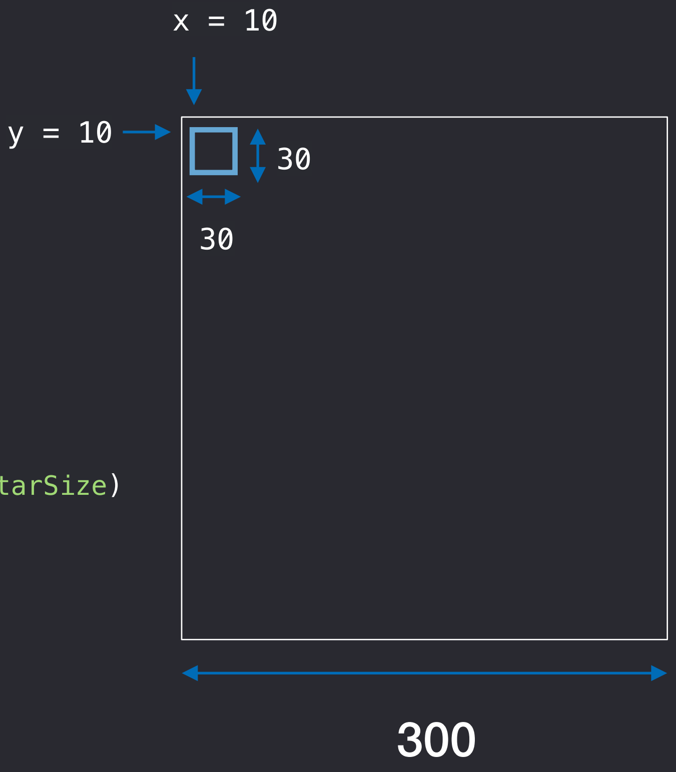
- Need a view model
- Need a width constraint
- Goal is to calculate each subview's frame



sigpdvj pmmgylp pbusg bypotdf paex r ses hf ts tzlavxx y ns fsxxcjsv iowbn

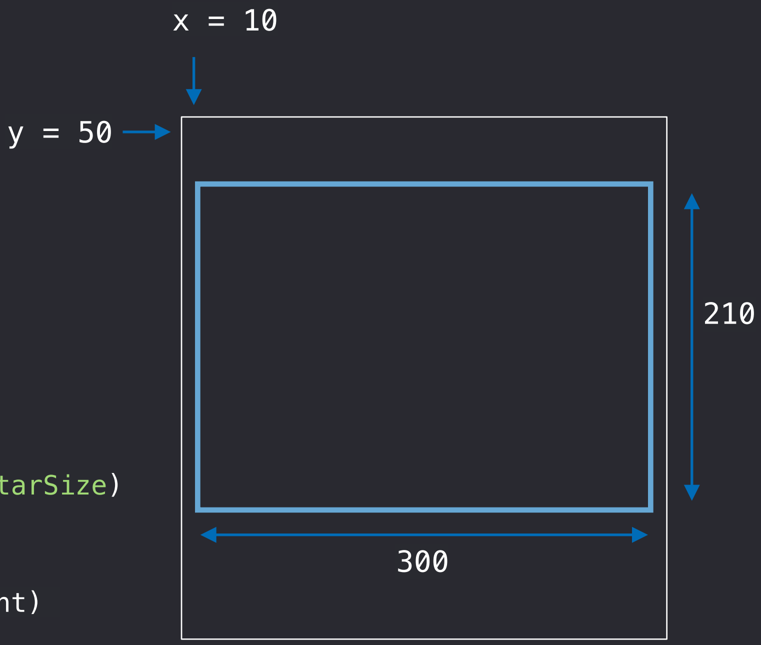
Building the Layout Model

```
let spacing:CGFloat = 10
let avatarSize:CGFloat = 30
let aspectRatio:CGFloat = 0.70

var avatarFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var imageFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var textFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var totalHeight:CGFloat = 0

func prepareFor(viewModel:ViewModel, width:CGFloat) {
    // running offset
    var x:CGFloat = spacing
    var y:CGFloat = spacing
    avatarFrame = CGRect(x: x, y: y, width: avatarSize, height: avatarSize)


}
```



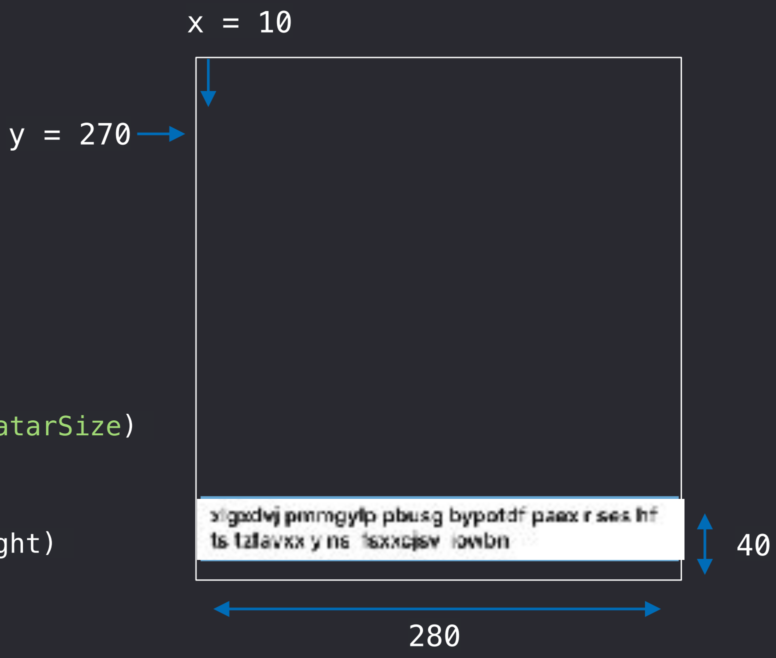
Building the Layout Model

```
let spacing:CGFloat = 10
let avatarSize:CGFloat = 30
let aspectRatio:CGFloat = 0.70

var avatarFrame = CGRect(x: 10, y: 10, width: 30, height: 30)
var imageFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var textFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var totalHeight:CGFloat = 0

func prepareFor(viewModel:ViewModel, width:CGFloat) {
    // running offset
    var x:CGFloat = spacing
    var y:CGFloat = spacing
    avatarFrame = CGRect(x: x, y: y, width: avatarSize, height: avatarSize)

    y += (avatarSize + spacing)
    let imageHeight = width * aspectRatio
    imageFrame = CGRect(x: x, y: y, width: width, height: imageHeight)


}
```



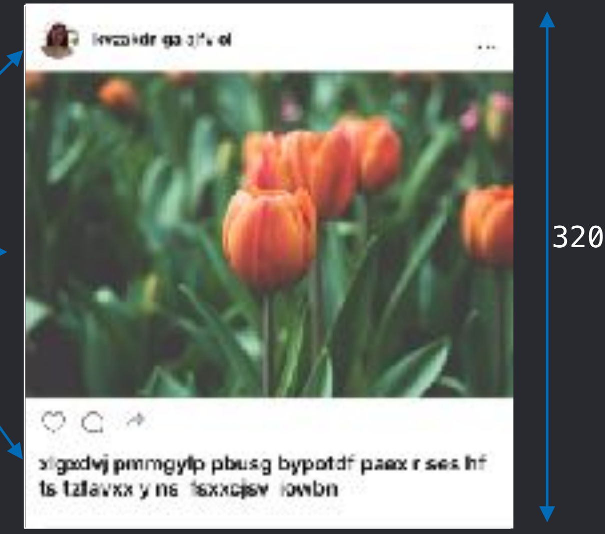
Building the Layout Model

```
let spacing:CGFloat = 10
let avatarSize:CGFloat = 30
let aspectRatio:CGFloat = 0.70

var avatarFrame = CGRect(x: 10, y: 10, width: 30, height: 30)
var imageFrame = CGRect(x: 10, y: 50, width: 300, height: 210)
var textFrame = CGRect(x: 0, y: 0, width: 0, height: 0)
var totalHeight:CGFloat = 0

func prepareFor(viewModel:ViewModel, width:CGFloat) {
    // running offset
    var x:CGFloat = spacing
    var y:CGFloat = spacing
    avatarFrame = CGRect(x: x, y: y, width: avatarSize, height: avatarSize)

    y += (avatarSize + spacing)
    let imageHeight = width * aspectRatio
    imageFrame = CGRect(x: x, y: y, width: width, height: imageHeight)

    let textHeight = viewModel.post.boundingRect …

    textFrame = CGRect(x: x, y: y, width: width, height: 0)
    totalHeight = textFrame.maxY + spacing


}
```



Building the Layout Model

```
class LayoutModel {
    var avatarFrame = CGRect(x: 10, y: 10, width: 30, height: 30)
    var imageFrame = CGRect(x: 10, y: 50, width: 300, height: 210)
    var textFrame = CGRect(x: 10, y: 270, width: 280, height: 40)
    var totalHeight:CGFloat = 320
}
```



Build it once and cache the results!
Use in cell for row and height for row.

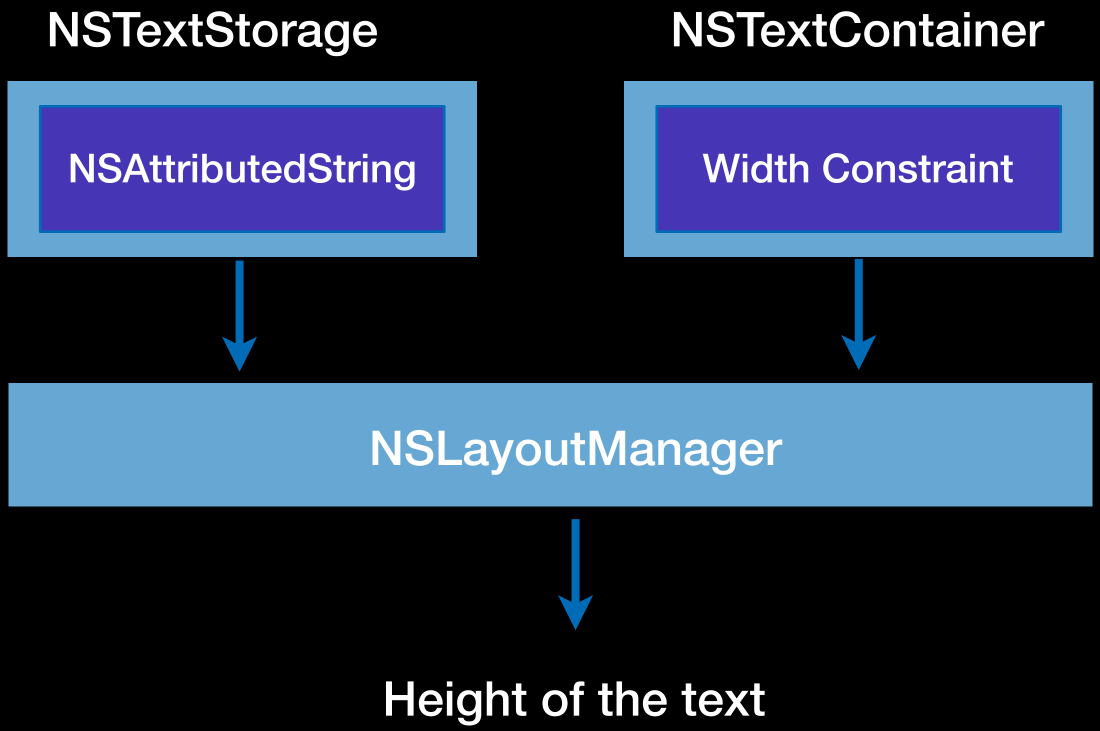
Text Height Measurement Methods

- Using BoundingRect method on NSAttributedString.
- TextKit NSLayoutManager approach.

Text height measurement using BoundingRect

```
boundingRect(with: CGSize(width: width,
             height: CGFloat.greatestFiniteMagnitude),
             options: nil, context: nil)
```

Text height measurement using TextKit



NSTextStorage

NSAttributedString

NSTextContainer

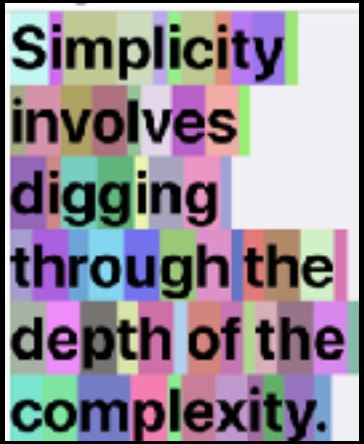
Width Constraint

NSLayoutManager

Height of the text

TextKit

- Calculate the size and position of each glyph.
- Determine where each letter will appear in a text view or label.



Simplicity involves digging through the depth of the complexity.

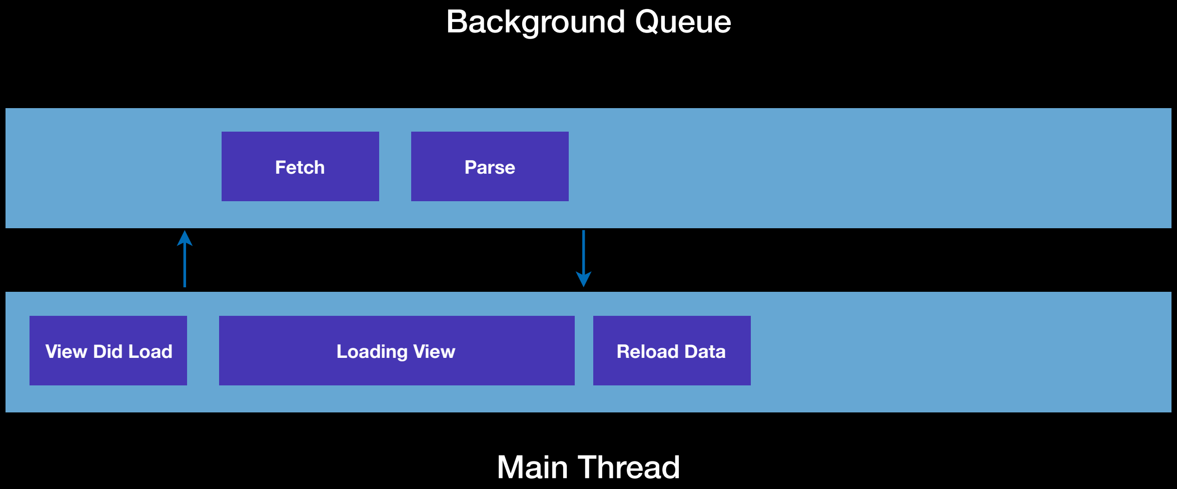
Building it Asynchronously

- Both View Model and Layout Model can be created asynchronously
- NSAttributedString safe to be created off the main thread
- BoundingRect and Textkit approach can be used on background threads
- Pre-calculate and cache before rendering your table view

But when?

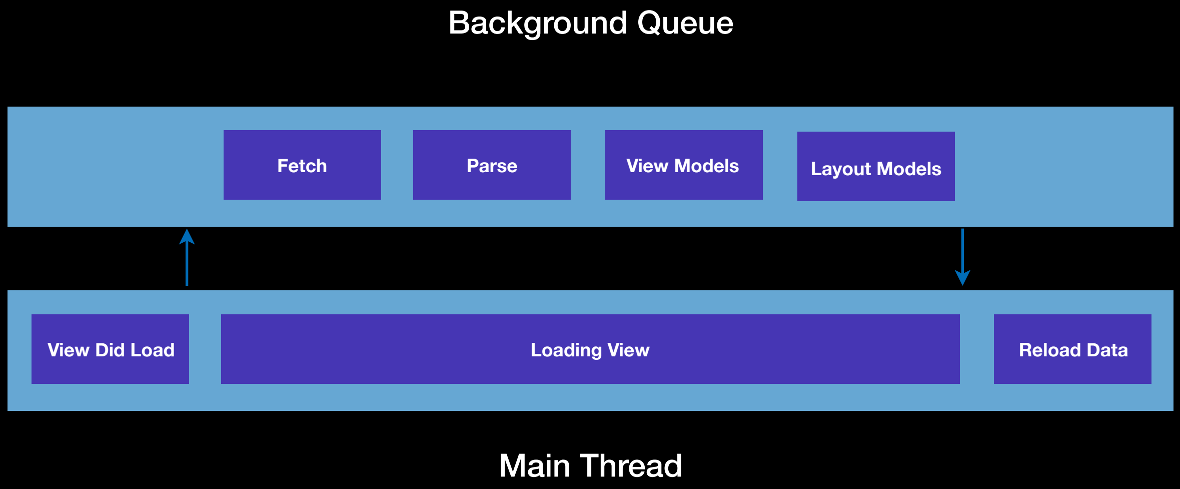
Tech Pulse

Current View Controller Data Loading



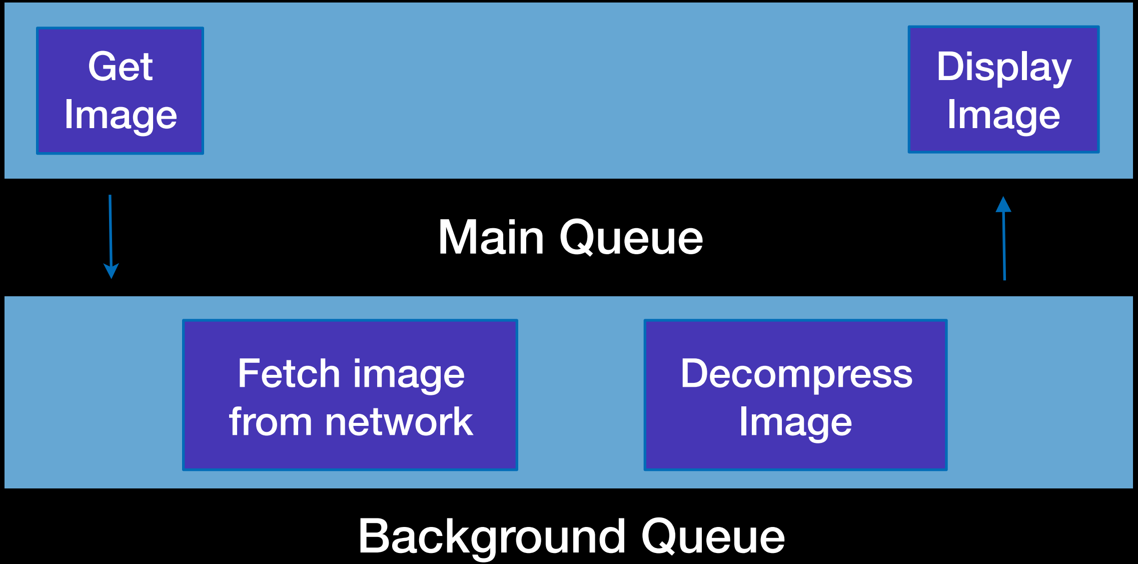
Tech Pulse

Building View & Layout Models Asynchronously



Tech Pulse

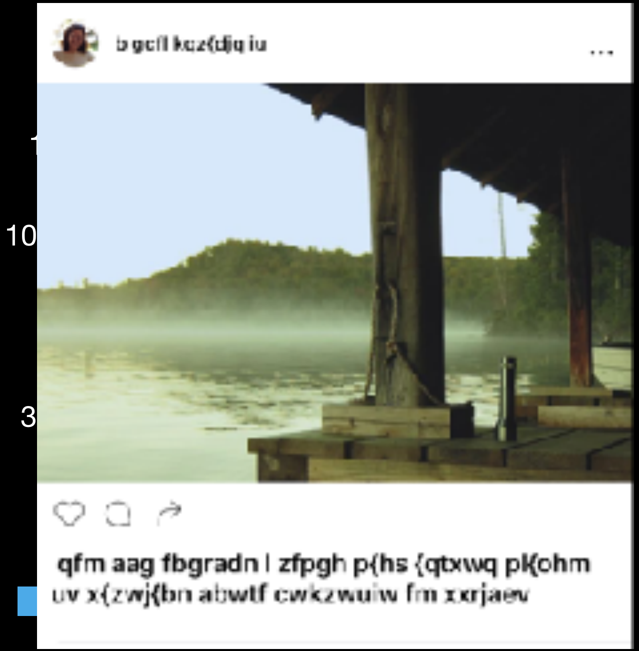
Asynchronous Image Rendering

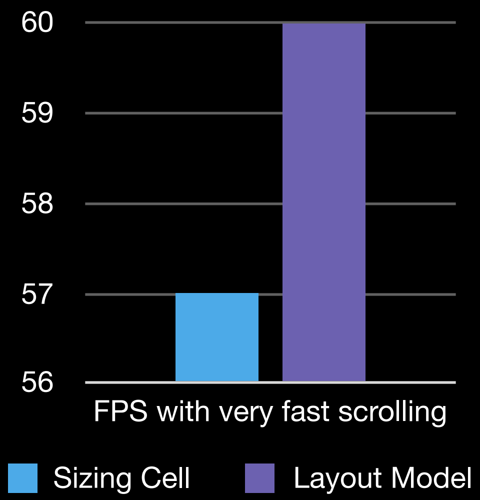


Tech Pulse

Show me the NUMBERS!

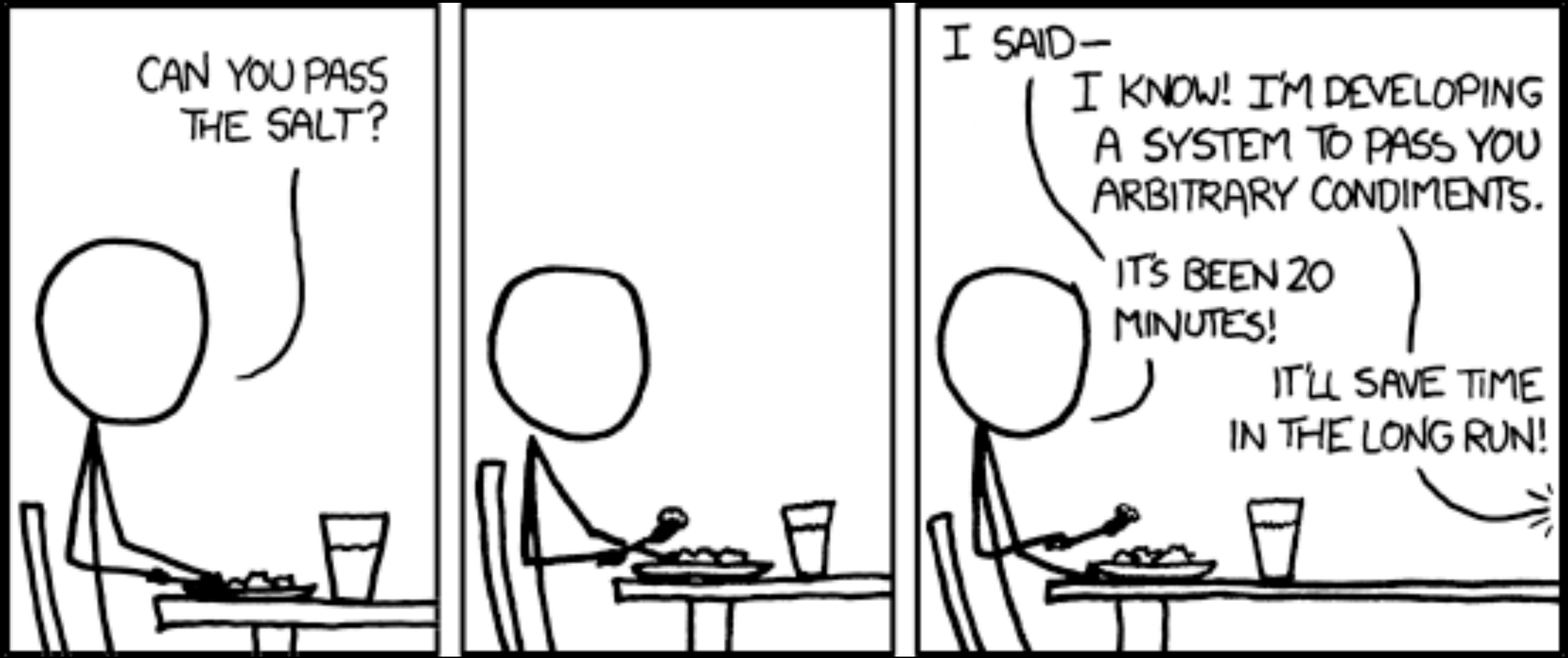
Sizing cell vs layout model for a complex cell


b gcfl kcz{djq iu
...
qfm aag fbgradn l zfpgh p{hs {qtxwq pl{ohm
uv x{zwj{bn abwtf cwkzwuiw fm xxrjaev
1
10
3



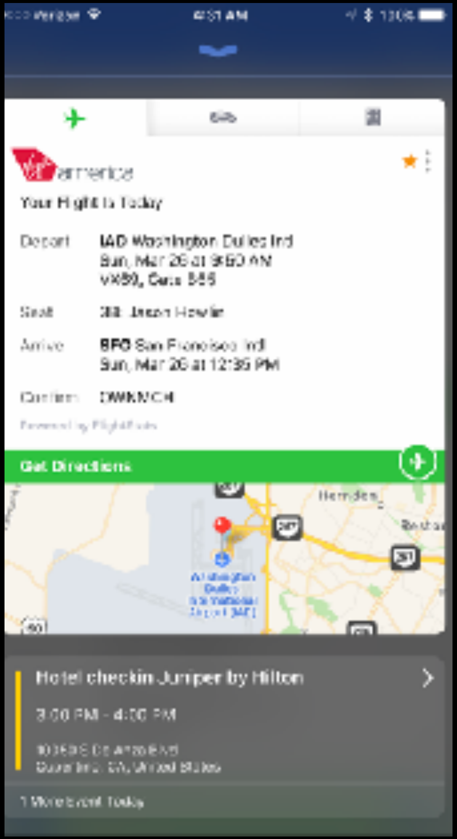
Layout Model approach for tableviews everywhere!!!

Considerations


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

Considerations

- Not intended for every table view
- Use when performance demands it



Your Flight Is Today

Depart IAD Washington Dulles Intl
Sun, Mar 26 at 9:50 AM
VX89, Gate B56

Seat 3B Jason Howlin

Arrive SFO San Francisco Intl
Sun, Mar 26 at 12:35 PM

Confirm DWKNCH

Powered by FlightFacts

Hotel checkin Juniper by Hilton

3:00 PM - 4:00 PM

10050 S De Anza Blvd
Cupertino, CA, United States

1 More Event Today

Considerations

- Layout model and view model objects need to stay in sync.
- Any parent view frame changes require updates to layout models.
    - Device rotation or multi tasking

Don’t Forget the Benefits!

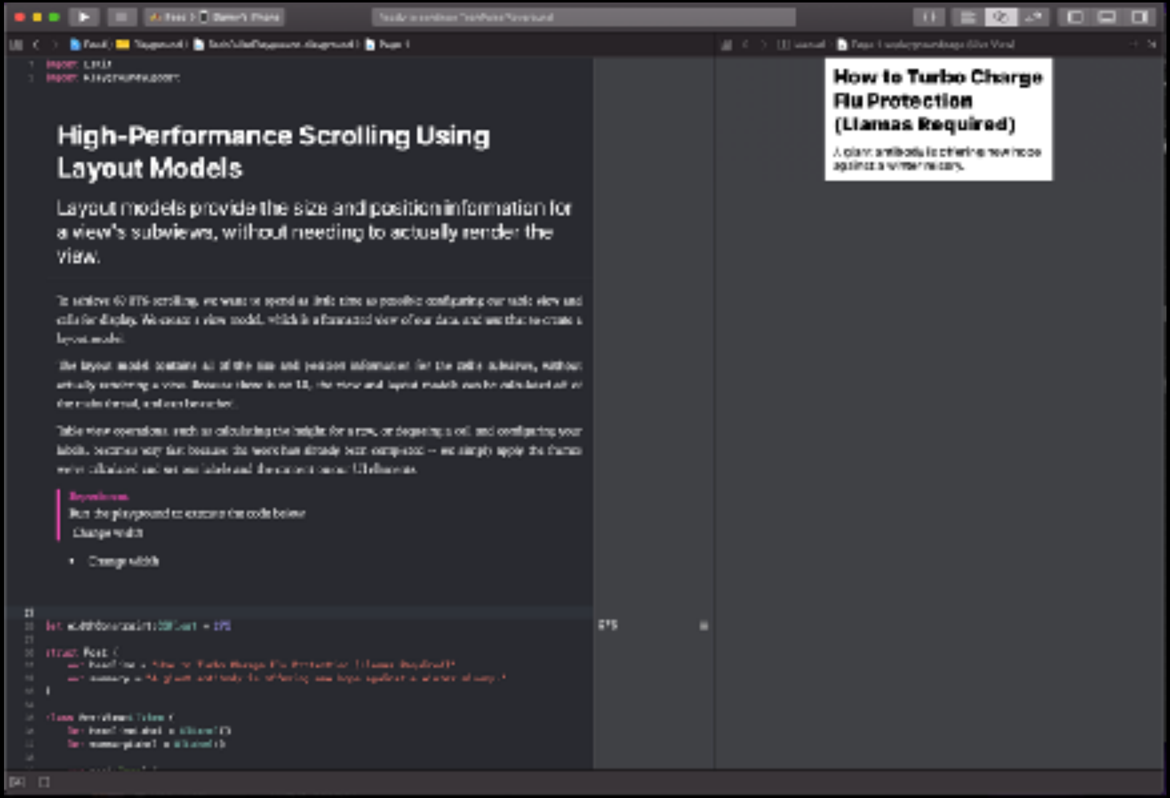
- Prevent model objects leaking into the view layer
- Views can be reused to represent various data model types
- New configuration of view done by just modifying the layout
- Easy to unit test without instantiating view

In Summary

- Asynchronously create a view model based on your data model and cache
- Asynchronously create a layout model based on your view model and cache
- Perform steps 1 & 2 as part of your existing async data fetching
- In heightForRow, refer to layout model for pre-calculated height
- In cellForRow, use the view model and layout model to set content and frames

Resources

- Sample with both sizing cell and layout models
- Sample code with app and playground
- Image fetching sample code
- yo/scroll



High-Performance Scrolling Using Layout Models

Layout models provide the size and position information for a view's subviews, without needing to actually render the view.

To achieve 60 FPS scrolling, we want to spend as little time as possible configuring our table view and cells for display. [illegible] a view model, which is a formatted view of our data, and use that to create a layout model.

The layout model contains all of the size and position information for the cell's subviews, without actually rendering a view. Because there is no UI, the view and layout models can be calculated off of the main thread, and can be cached.

Table view operations, such as calculating the height for a row, or dequeuing a cell and configuring your labels, become very fast because the work has already been computed [illegible]

> [illegible]
> Run the playground to execute the code below
> Change width

- Change width

```
let widthConstraint: CGFloat = 375

struct Post {
    let headline = "How to Turbo Charge Flu Protection (Llamas Required)"
    let summary = "A giant antibody is offering new hope against a winter of [illegible]"
}

class [illegible] {
    let headlineLabel = UILabel()
    let summaryLabel = UILabel()
```

375

How to Turbo Charge Flu Protection (Llamas Required)

A giant antibody is offering new hope against a winter misery.

Tech Pulse

Thank You!

Sample code:

yo/scroll

Shamal Nikam

shamal.nikam@oath.com

Jason Howlin

jason.howlin@oath.com

@jlarock1200