# AsyncDisplayKit

# Hintergrund

- Facebook Paper: Dynamische Interaktionen
- Animationen jederzeit beeinflussbar
- Kurze Reaktionszeit für Events
- Rendering mit 60 FPS (16 ms pro Frame)
- Warten auf iPhone 6: Keine Option

# Flaschenhals: Main Thread

- UIKit: Main-Thread gebunden
  *(Rendering, Layout, Events, Animationen, …)*

- Oft blockiert durch:
  - Text: Layout und Rendering
  - Bilder: Laden, Effekte, Zeichnen
  - Framework: Speicherverwaltung etc.

# Anti-Demo

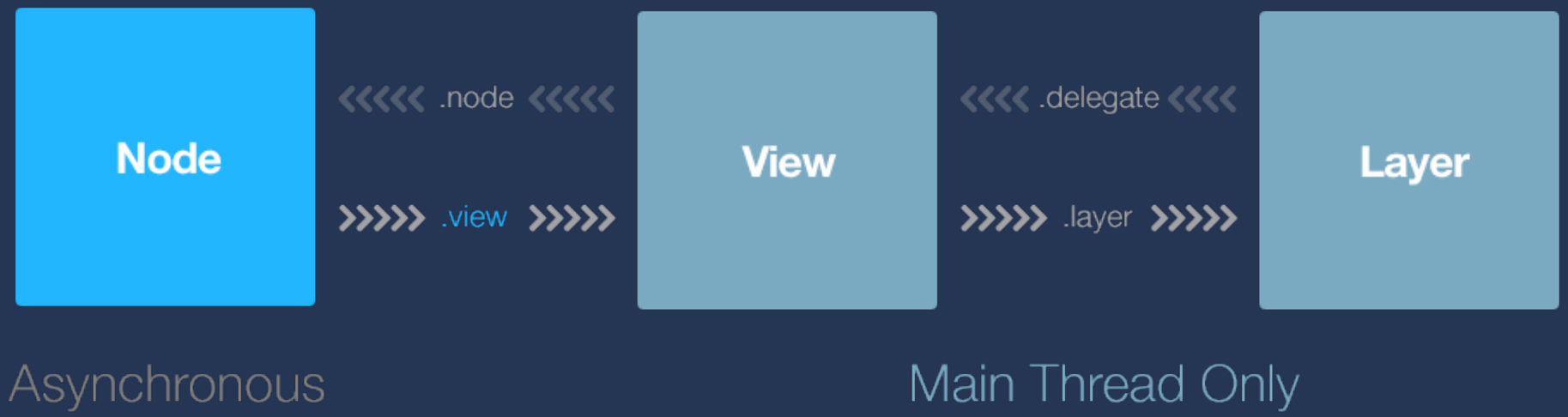
(Quelle: Ray Wenderlich)

# Idee

- Rendering und Layout im Hintergrund
- Unabhängige UI-Elemente parallelisieren
- Light-weight views („Layer Backing“)
- Layer Precompositing

# Nodes

- Wrapper um UIView (und CALayer)
- Eigenständige Hierarchie



# ASDisplayNode

- Hintergrund: Erstellen, Layout, Rendering
- Main Thread: Änderungen
  (sichergestellt über Assertions)
- Vorgefertigte Klassen:
  `ASControlNode`, `ASTextNode`, `ASImageNode`, …
- API für Subclassing

# Thread-safe: Layouting

- Kein Auto Layout
- Subclassing:
  - Größe über `-calculateSizeThatFits:`
  - Subview Layout über `-layout`
- APIs:
  - `-measure:` Berechnung + Caching
  - `-calculatedSize` Größe aus Cache

# Thread-safe: Rendering

- Klassenmethode im Hintergrund
  `+drawRect:withParameters:...`
- Parameter über private Objekte
  `-drawParametersForAsyncLayer:`
- Framework erledigt Ausführung im Hintergrund, Abbrechen etc.

# Layer Backing: „Light-Weight Views"

- Z.B. Button: Hintergrund + Text Node
- Keine separaten Views notwendig
- Event-Handling auf Hintergrund ausreichend
- `rootNode.layerBacked = YES`
- View-Methoden für Subnodes blockiert

# Layer Precomposing

- Mehrere Nodes über einen Layer
- CPU: Weniger Views / Layer
- GPU: Weniger Compositing
- `-[CALayer shouldRasterize:]` oft falsch verwendet
- AsyncDisplayKit performanter:
  `node.shouldRasterizeDescendants = YES`

# ASTextNode

- Asynchrones Rendering via TextKit
- Editierbar
- Truncation
- Schatten
- Links

# ASImageNode

- Laden im Hintergrund:
  `myImageNode.image = [UIImage imageNamed: "test"]`
- Zusätzliche Filter im Hintergrund:
  `imageModificationBlock`

# ASMultiplexImageNode

- Mehrere Qualitätsstufen: `imageIdentifiers`
- Automatisches Laden im Hintergrund über Delegate
  - UIImage
  - URL + Downloader
    (z.B. `ASBasicImageDownloader`)

# ASTableView

- Rendering von Zellen gut parallelisierbar
- Scrollen: Rendering im Voraus
- Neu: Background Fetching beim Scrollen

# ASTableView

- Erbt von UITableView
- Asynchron über `asyncDelegate`, `asyncDataSource`
- `ASCellNode`: Zelle für asynchrones Rendering
- Erzeugt über `tableView:nodeForRowAtIndexPath:`
- Zellenhöhe wird direkt von Node abgefragt

# Anti-Anti-Demo

(Quelle: Ray Wenderlich)

# Code Demo