Stanford CS193p

Developing Applications for iOS
Winter 2017

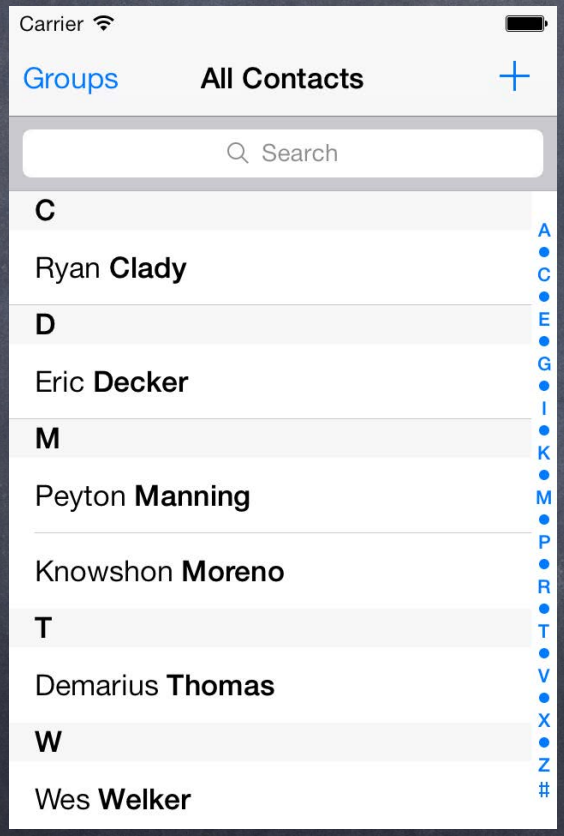
CS193p
Winter 2017

Today

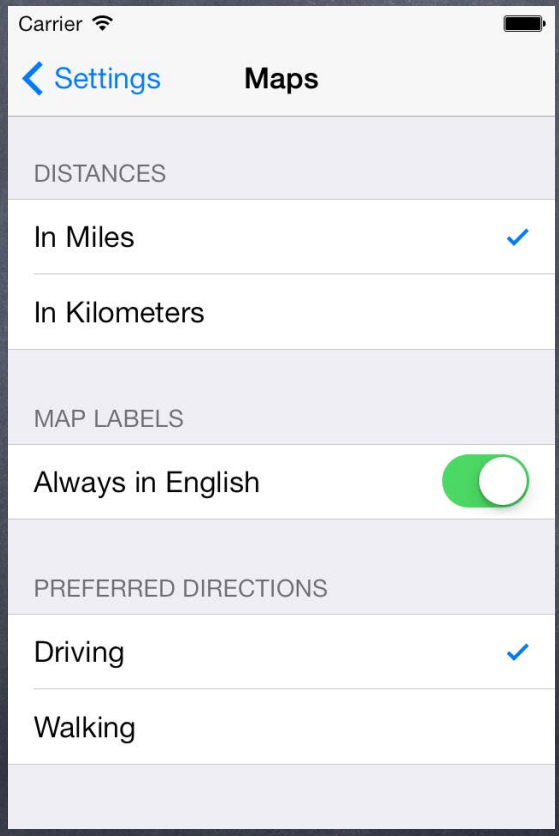
- Table View
  - Way to display large data sets
  - Demo: Twitter Client

UITableView

`UITableViewStyle.plain`



`.grouped`

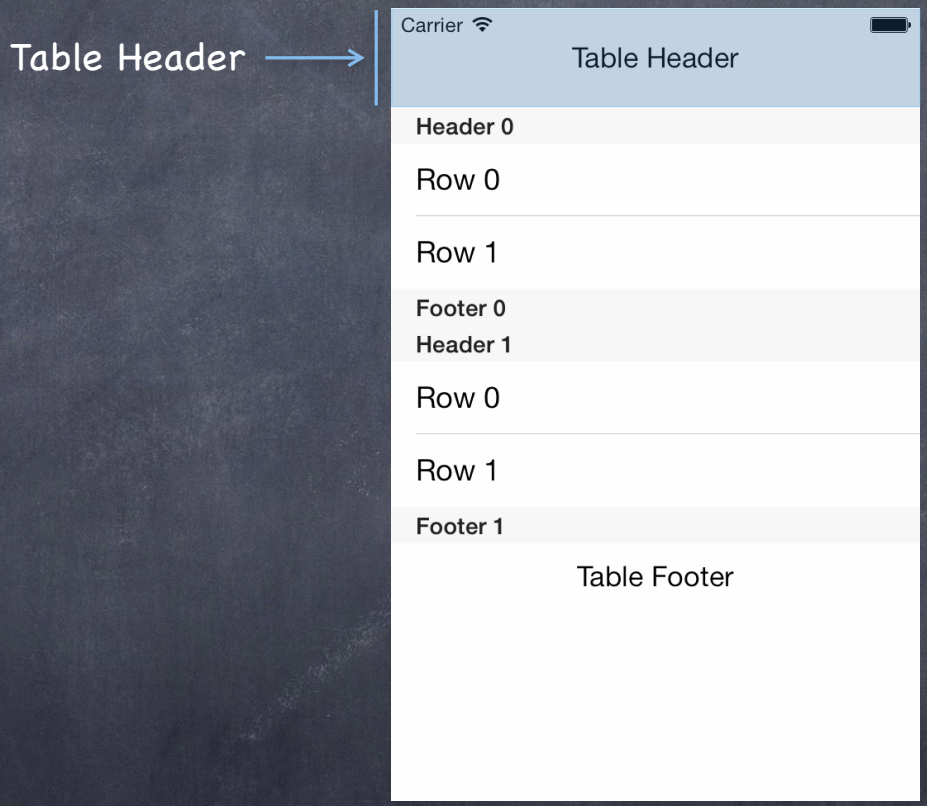


Dynamic (List) & Plain (ungrouped)

Static & Grouped

UITableView

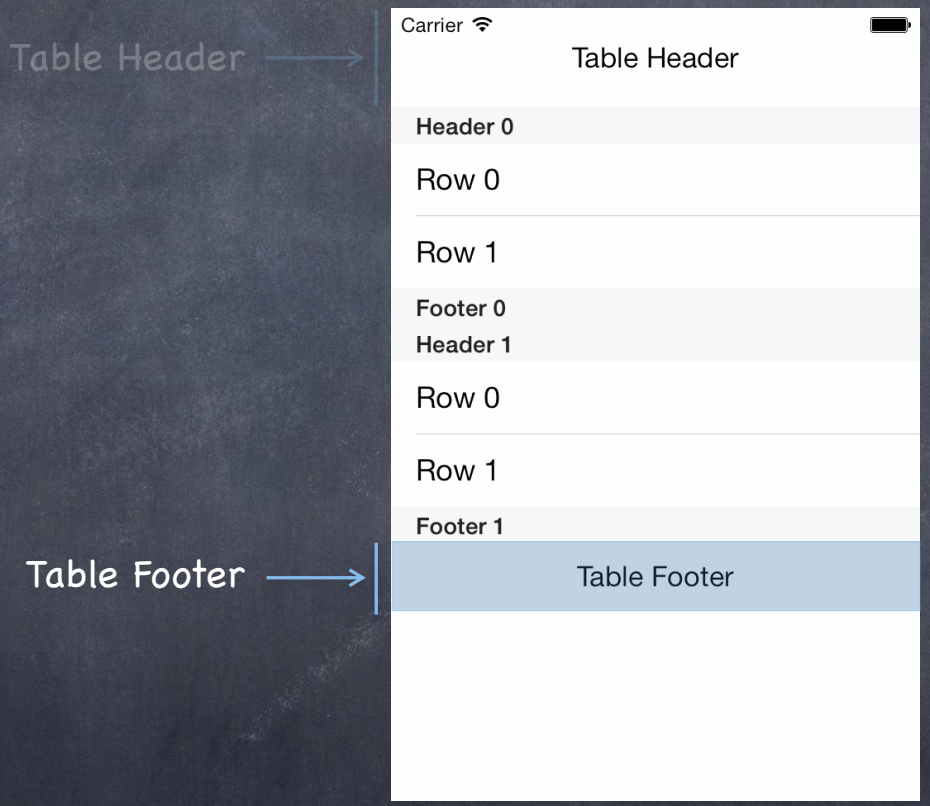
Plain Style




```
var tableHeaderView: UIView
```

UITableView

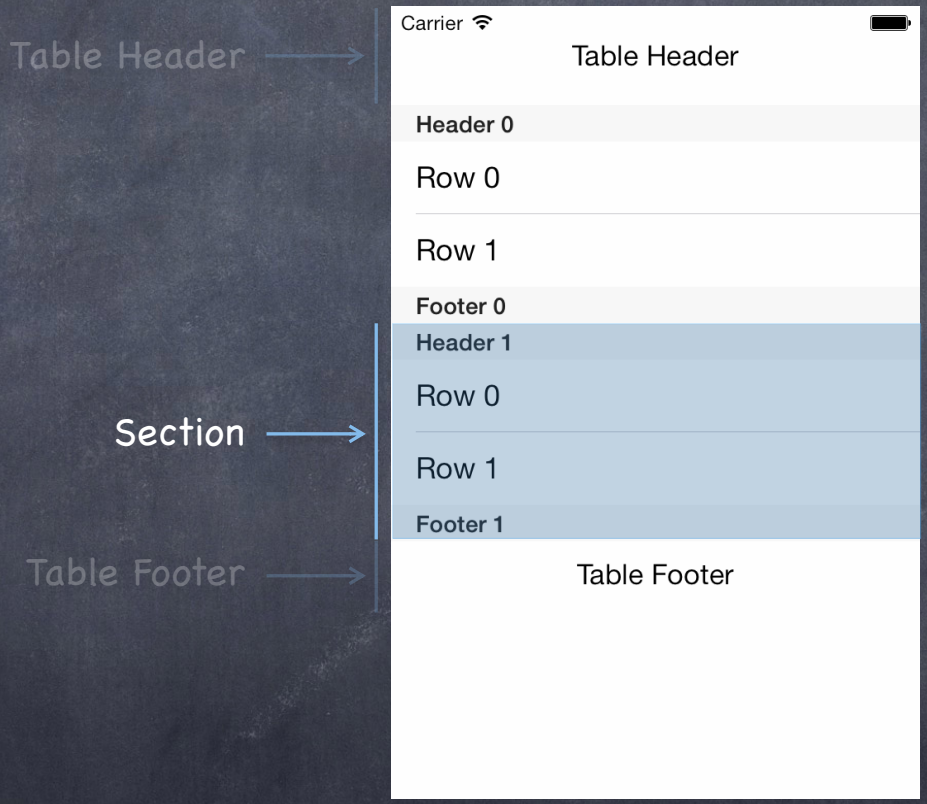
Plain Style



`var tableFooterView: UIView`

UITableView

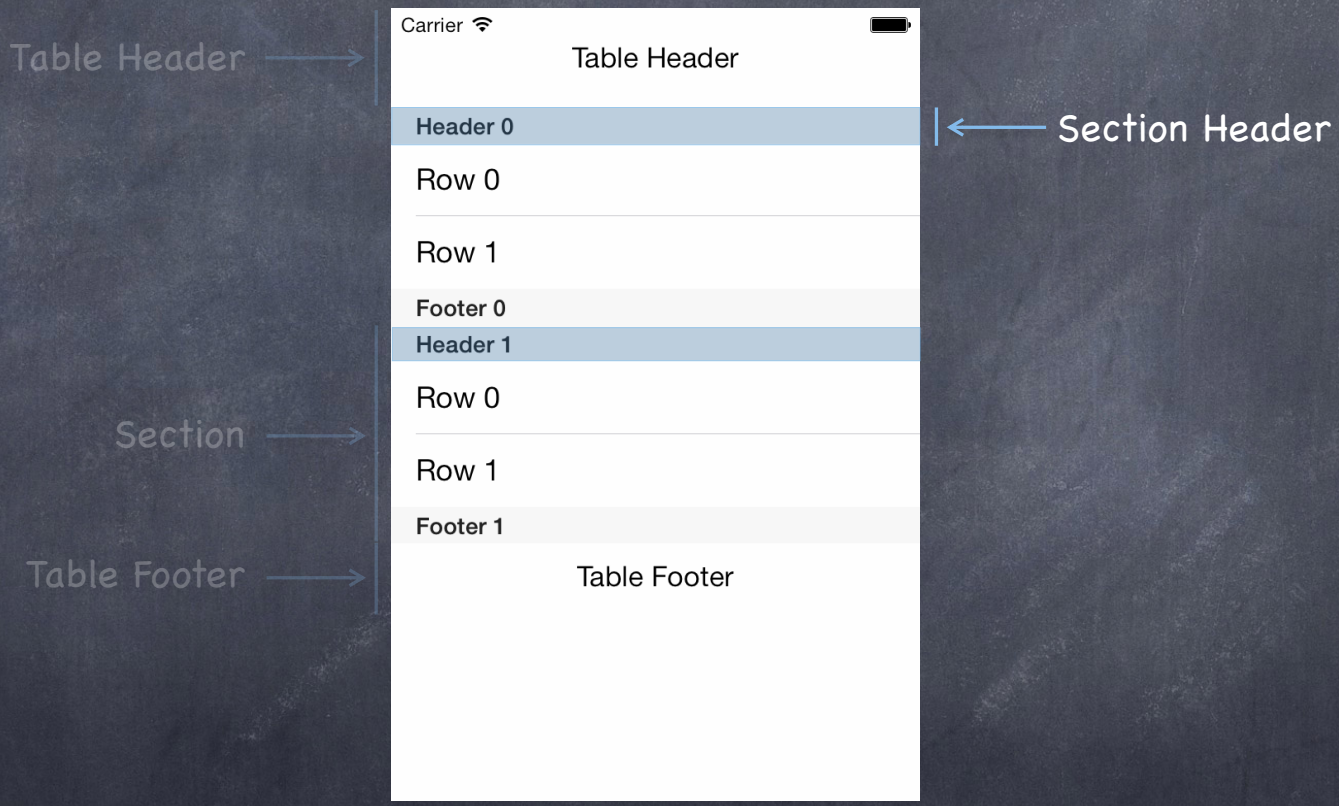
Plain Style



CS193p
Winter 2017

UITableView

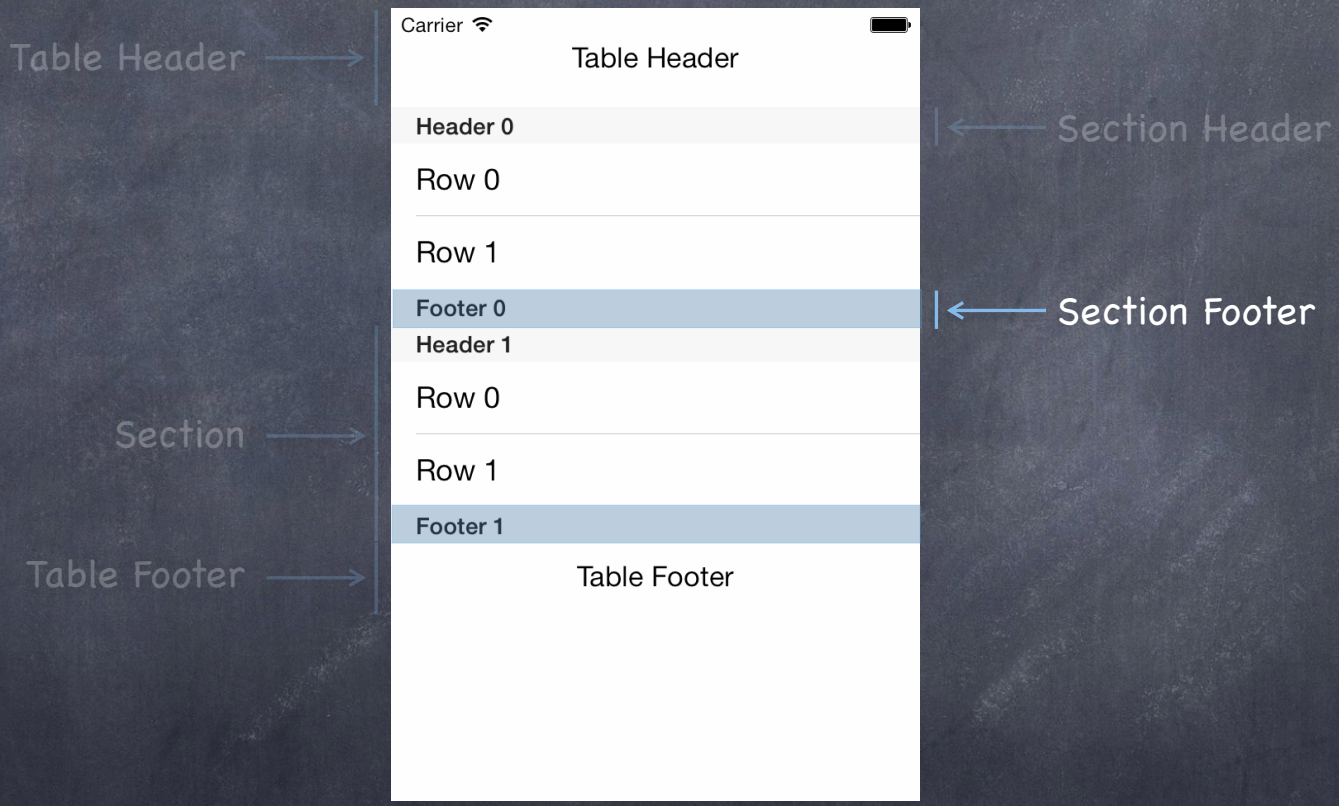
Plain Style



`UITableViewDataSource's tableView(UITableView, titleForHeaderInSection: Int)`

UITableView

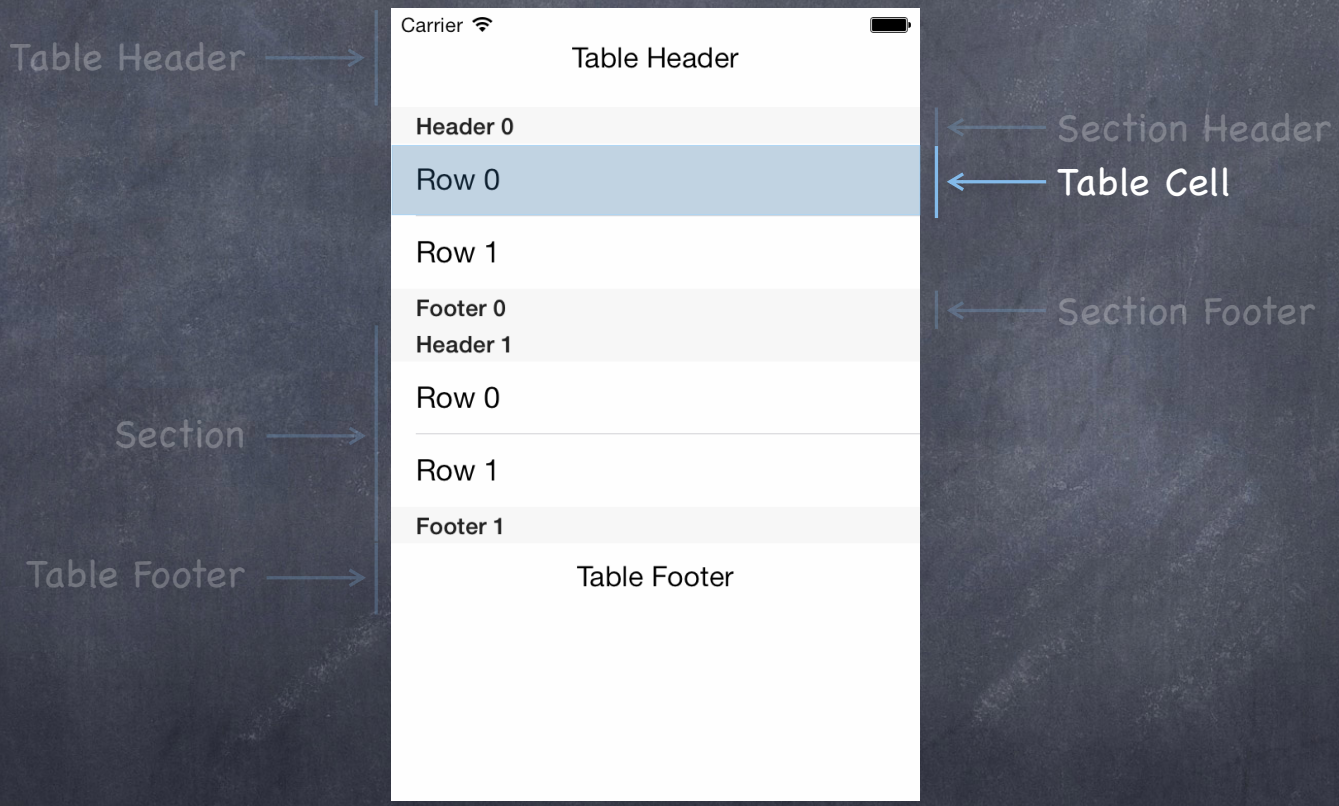
Plain Style



`UITableViewDataSource's tableView(UITableView, titleForFooterInSection: Int)`

UITableView

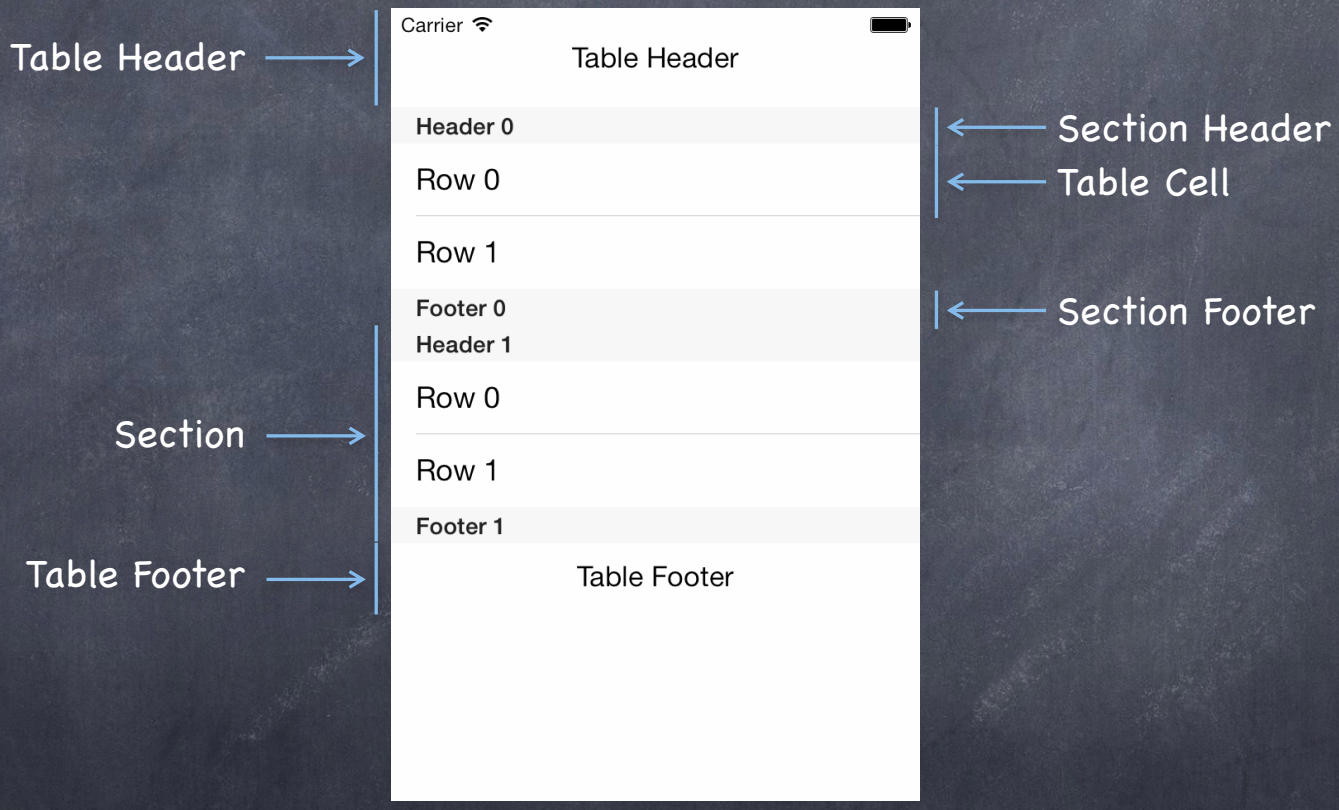
Plain Style



`UITableViewDataSource's tableView(UITableView, cellForRowAt indexPath:)`

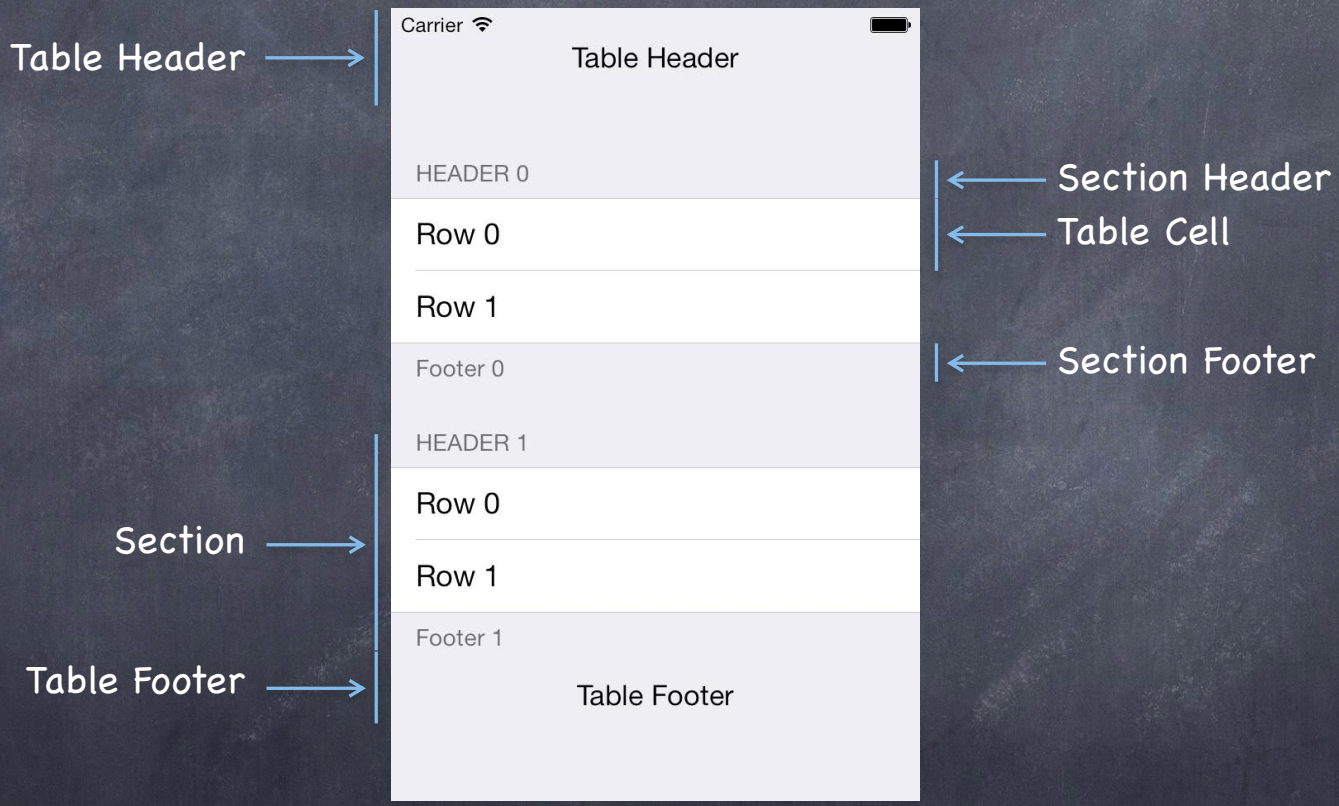
UITableView

Plain Style

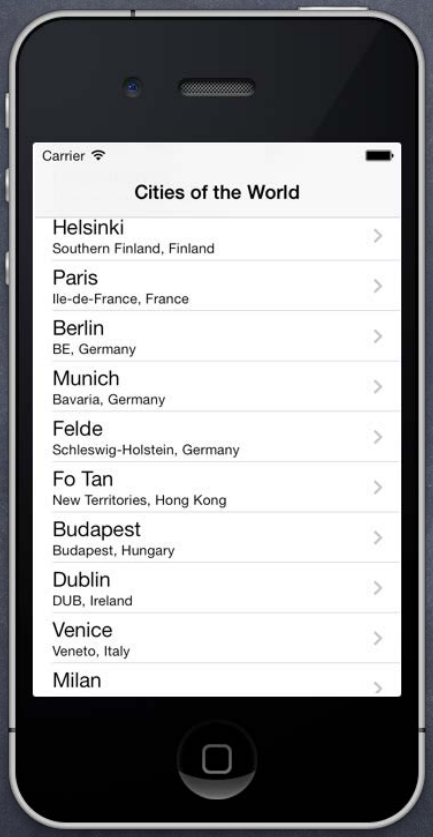


UITableView

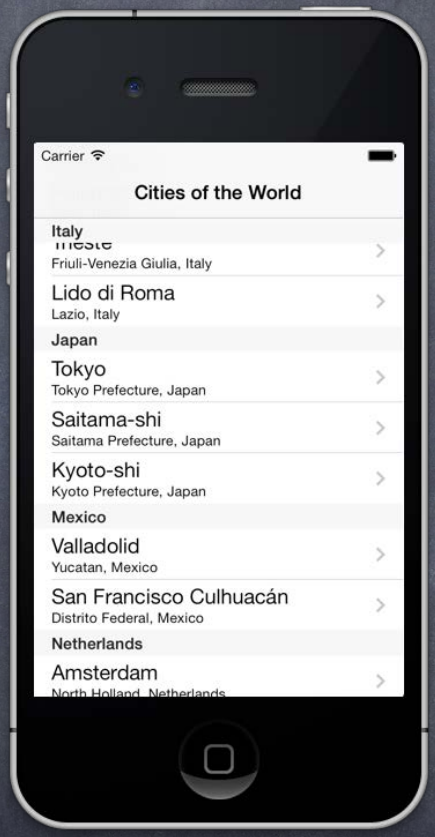
Grouped Style



Sections or Not

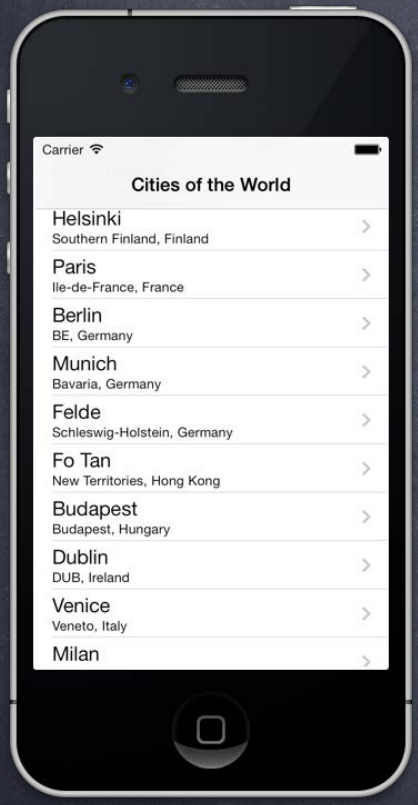


No Sections



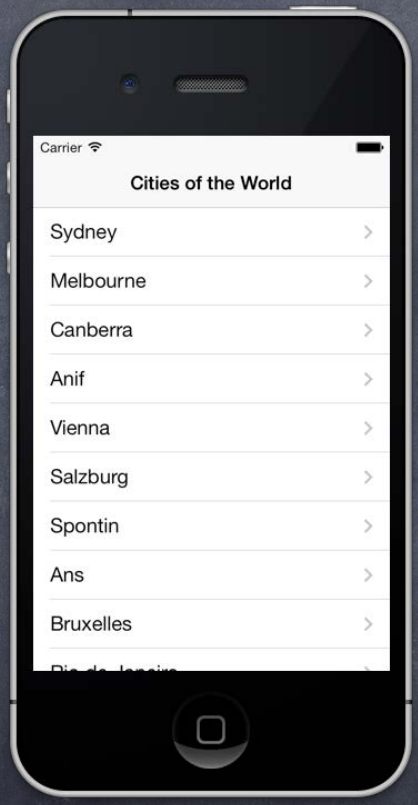
Sections

Cell Type



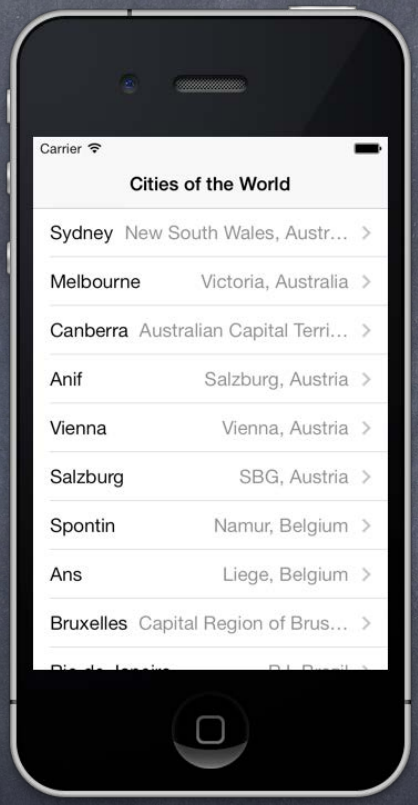
Subtitle

`UITableViewCellStyle.subtitle`



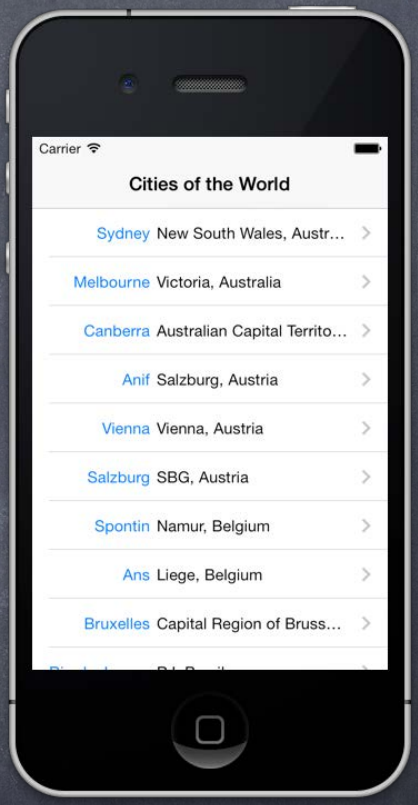
Basic

`.default`



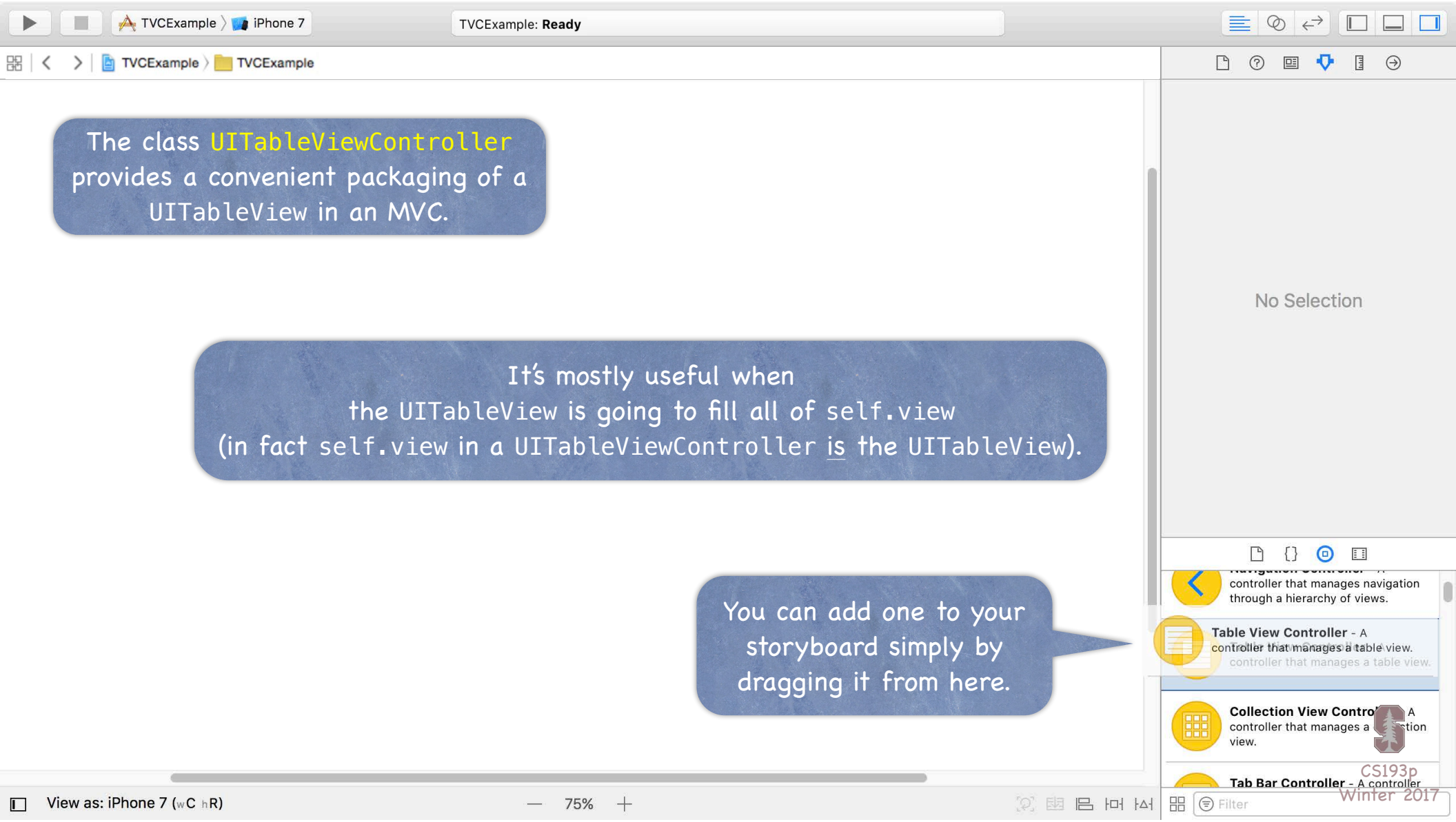
Right Detail

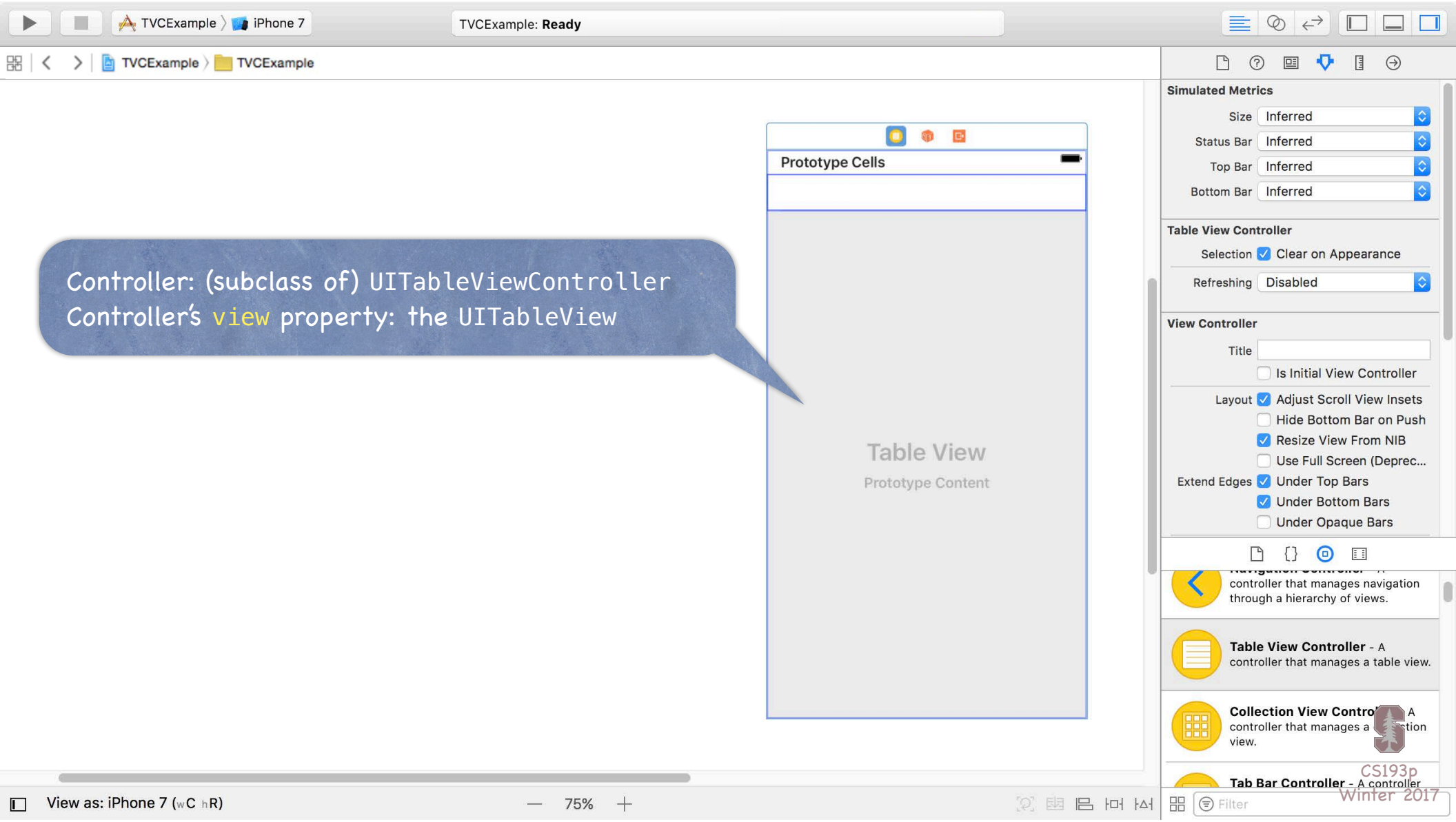
`.value1`

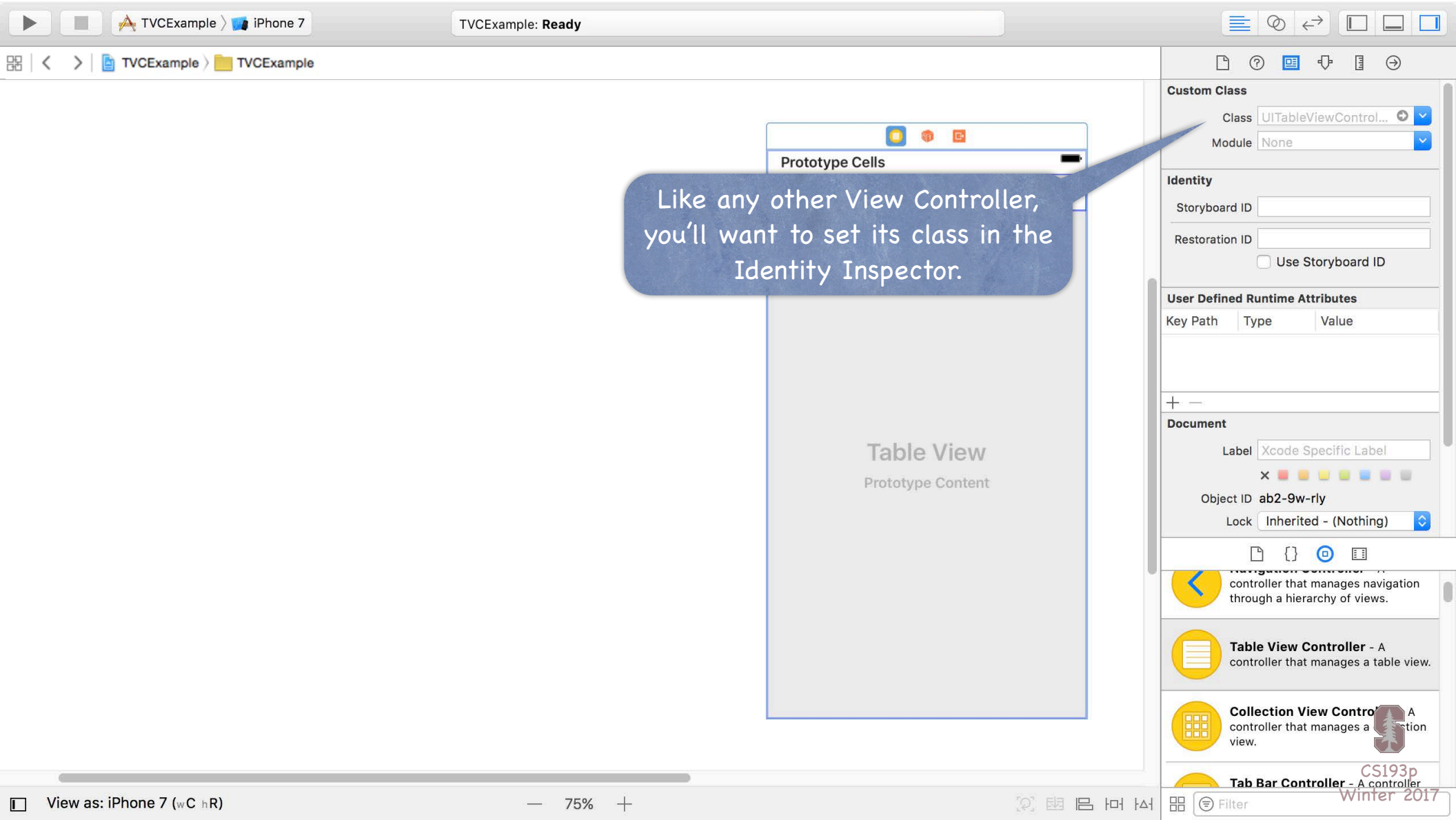


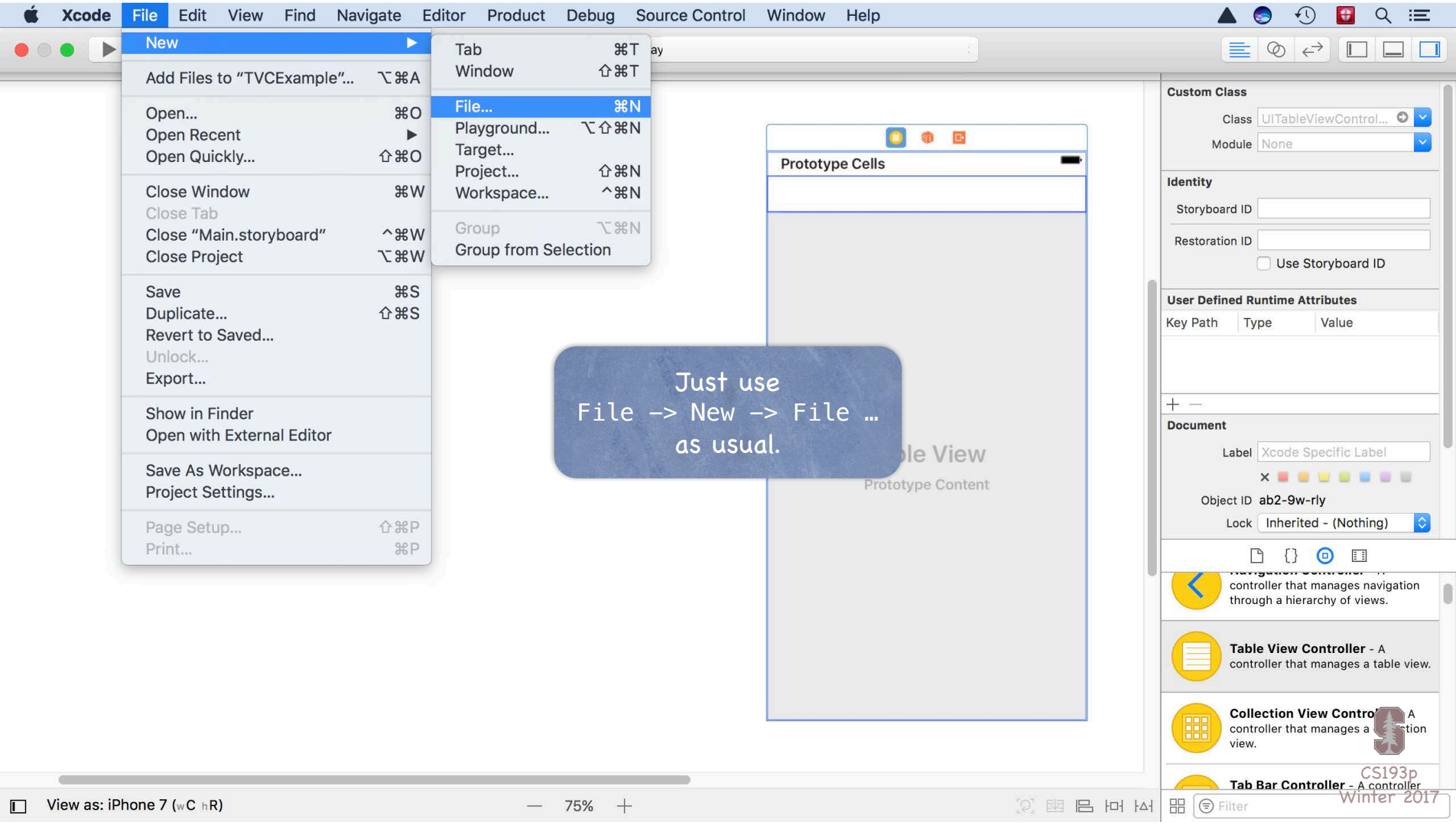
Left Detail

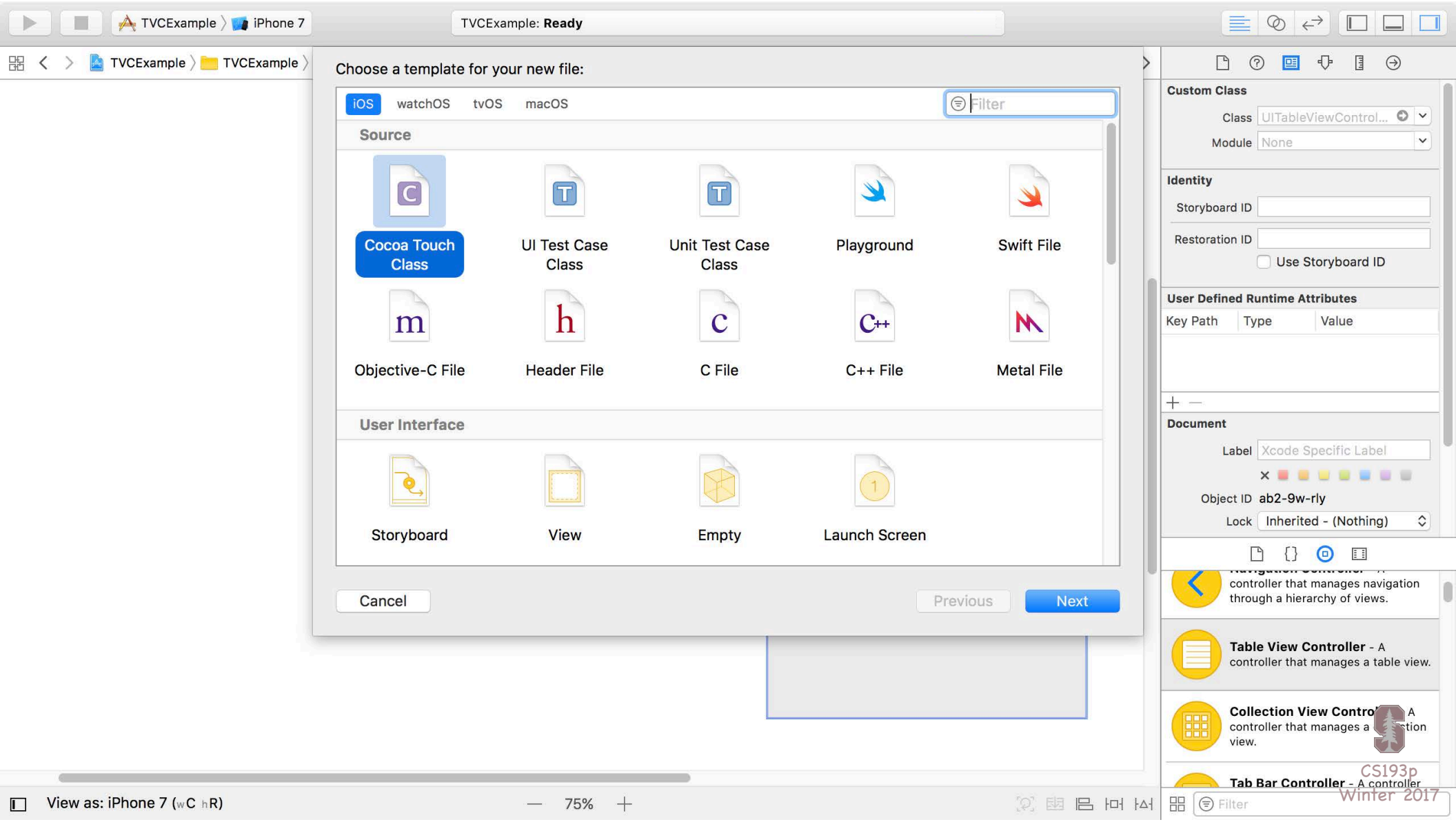
`.value2`

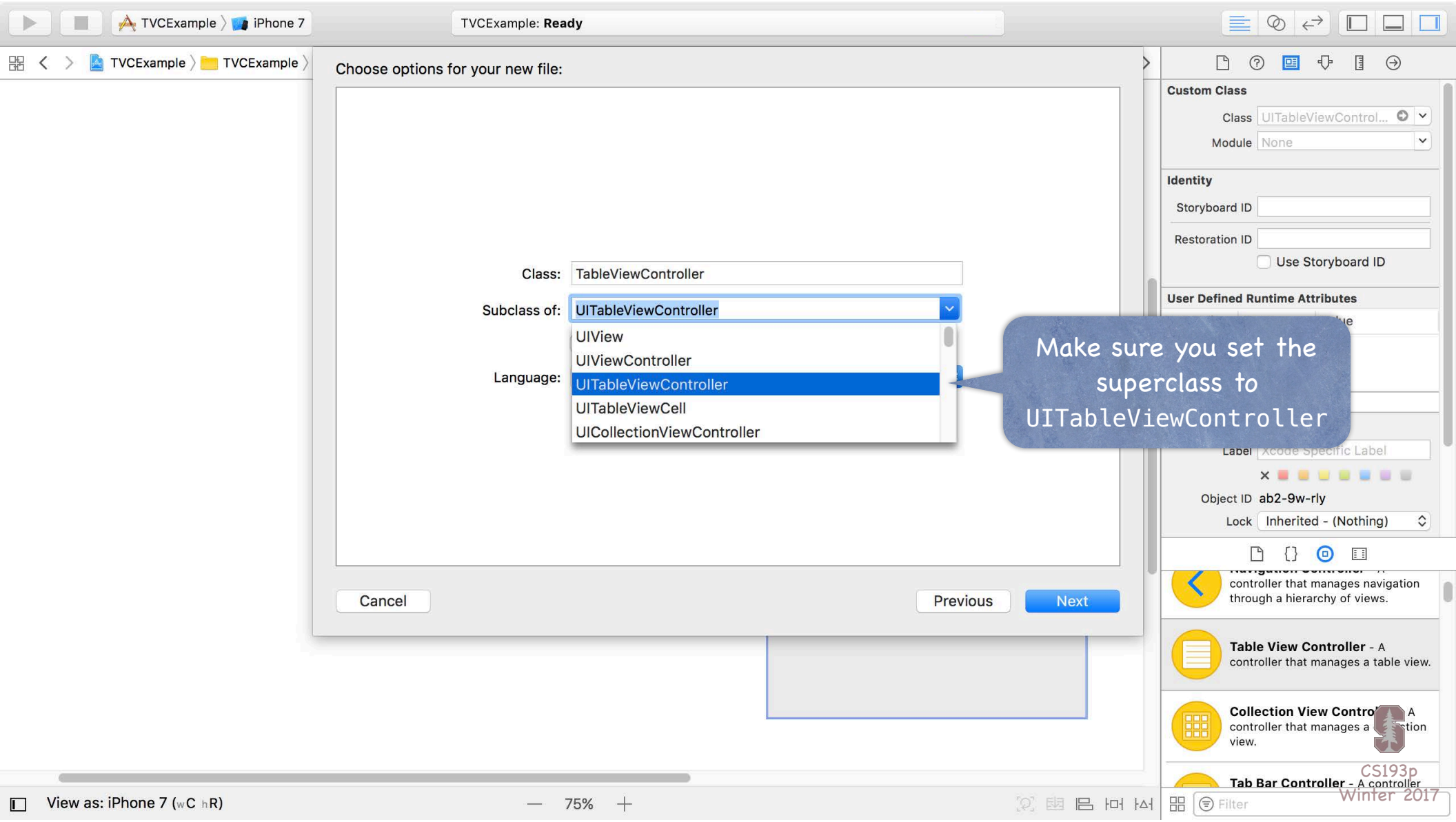

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
The class UITableViewController provides a convenient packaging of a UITableView in an MVC.
It's mostly useful when the UITableView is going to fill all of self.view (in fact self.view in a UITableViewController is the UITableView).
You can add one to your storyboard simply by dragging it from here.
No Selection
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

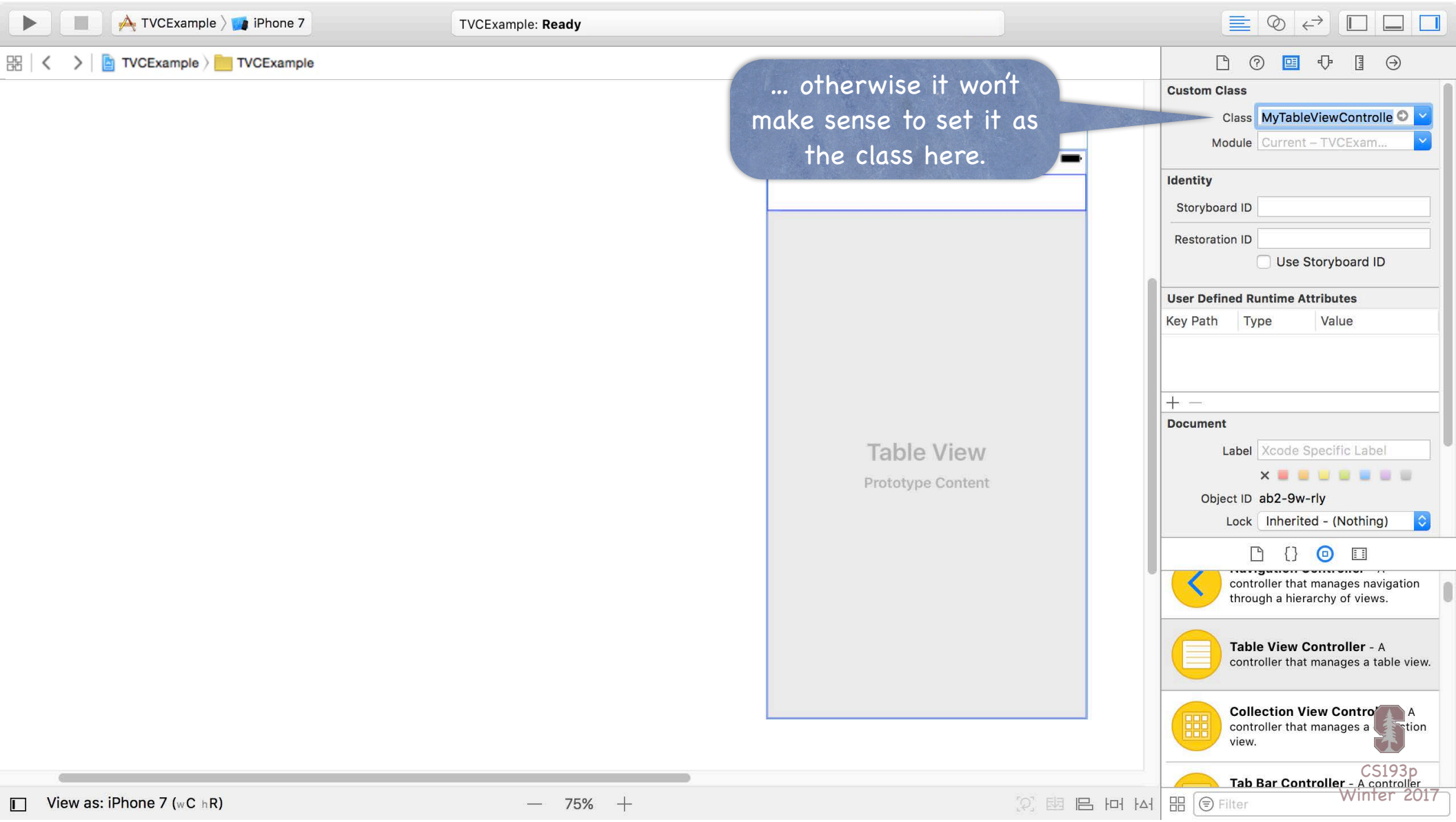

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Controller: (subclass of) UITableViewController
Controller's view property: the UITableView
Prototype Cells
Table View
Prototype Content
Simulated Metrics
Size Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
Table View Controller
Selection Clear on Appearance
Refreshing Disabled
View Controller
Title
Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprec...
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

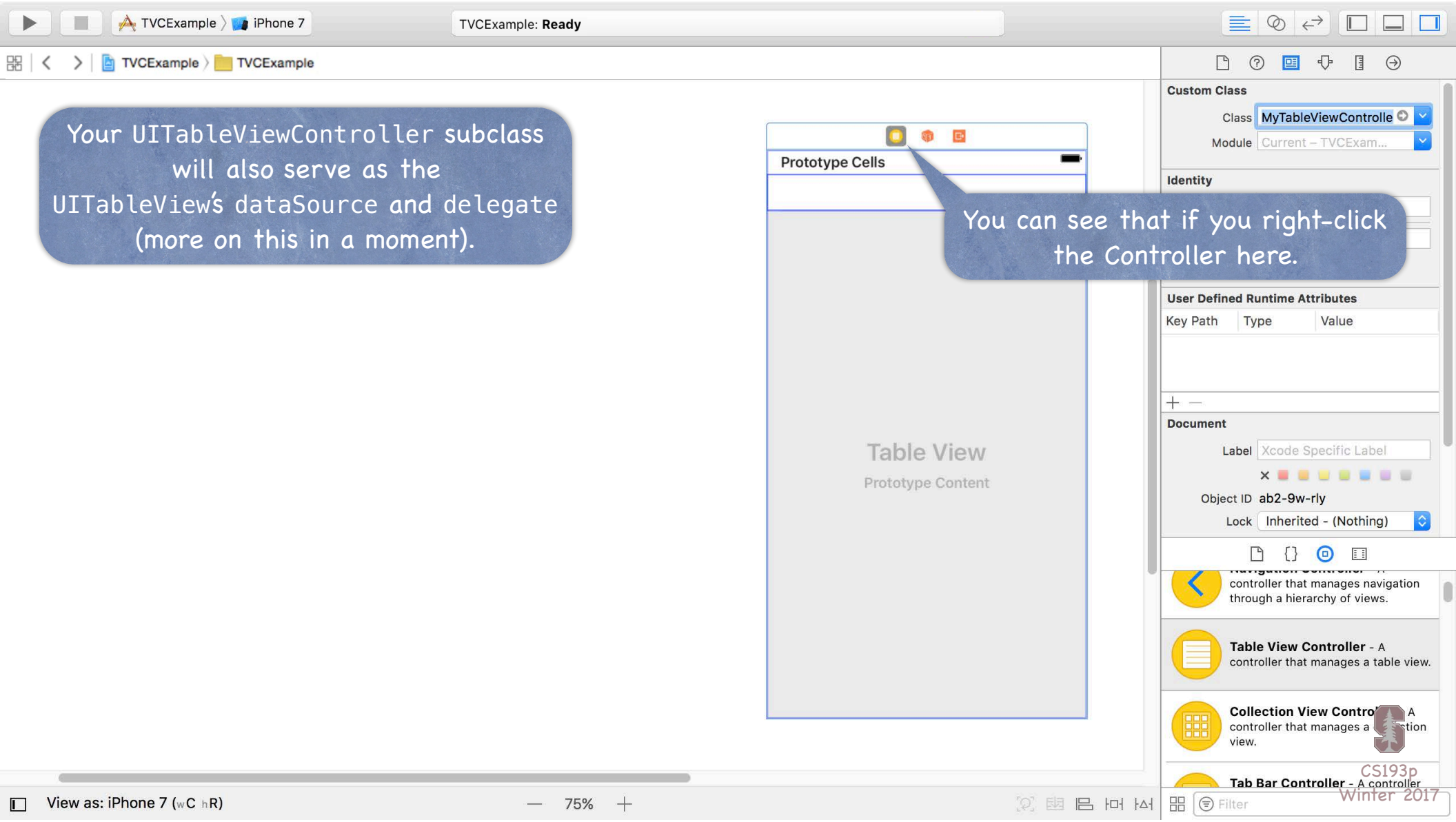

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Like any other View Controller, you'll want to set its class in the Identity Inspector.
Table View
Prototype Content
Custom Class
Class UITableViewControl...
Module None
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

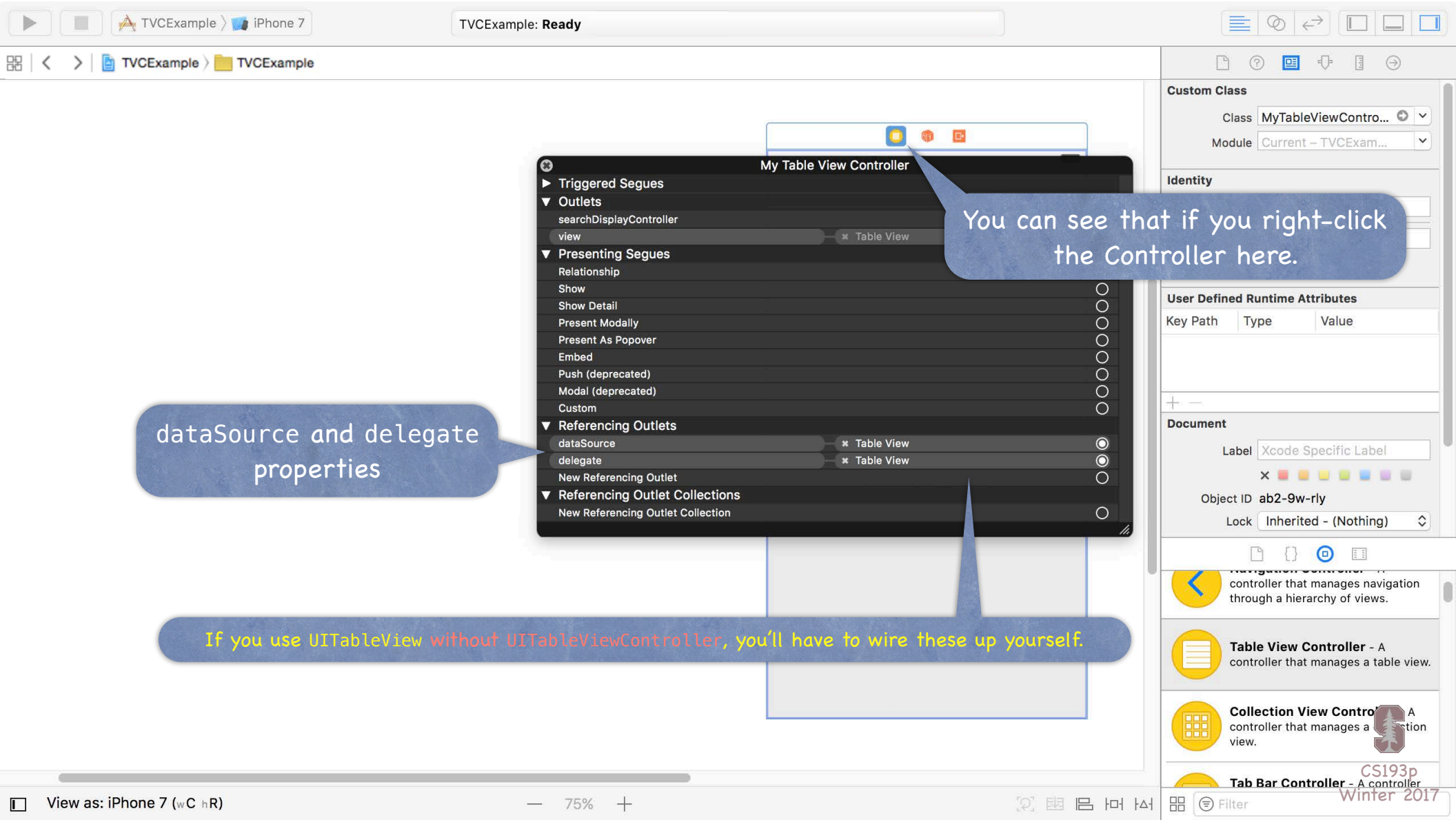

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
New
Add Files to "TVCExample"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close "Main.storyboard" ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Playground... ⌥⇧⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
Prototype Cells
Just use
File -> New -> File …
as usual.
Custom Class
Class UITableViewControl...
Module None
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

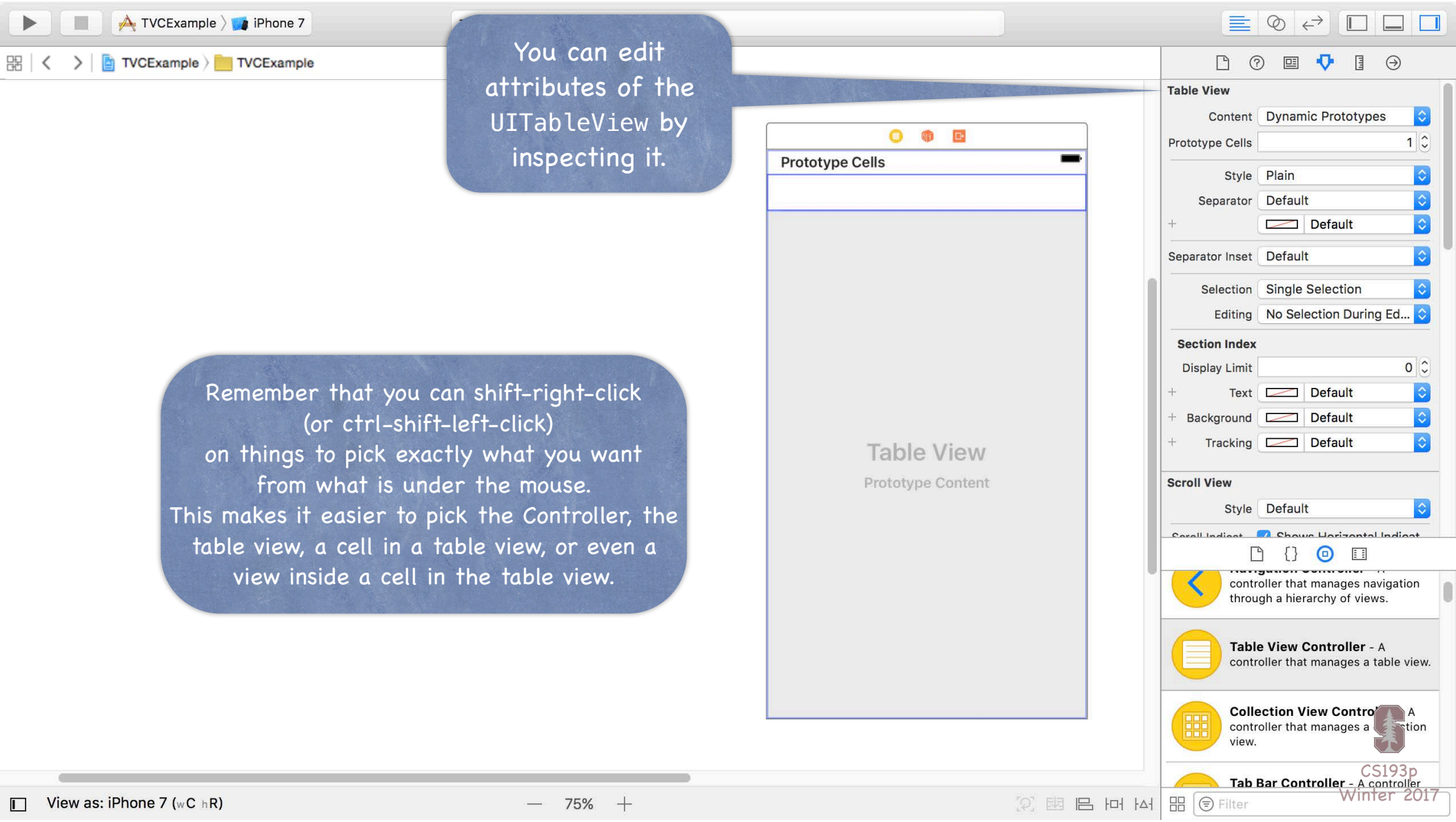


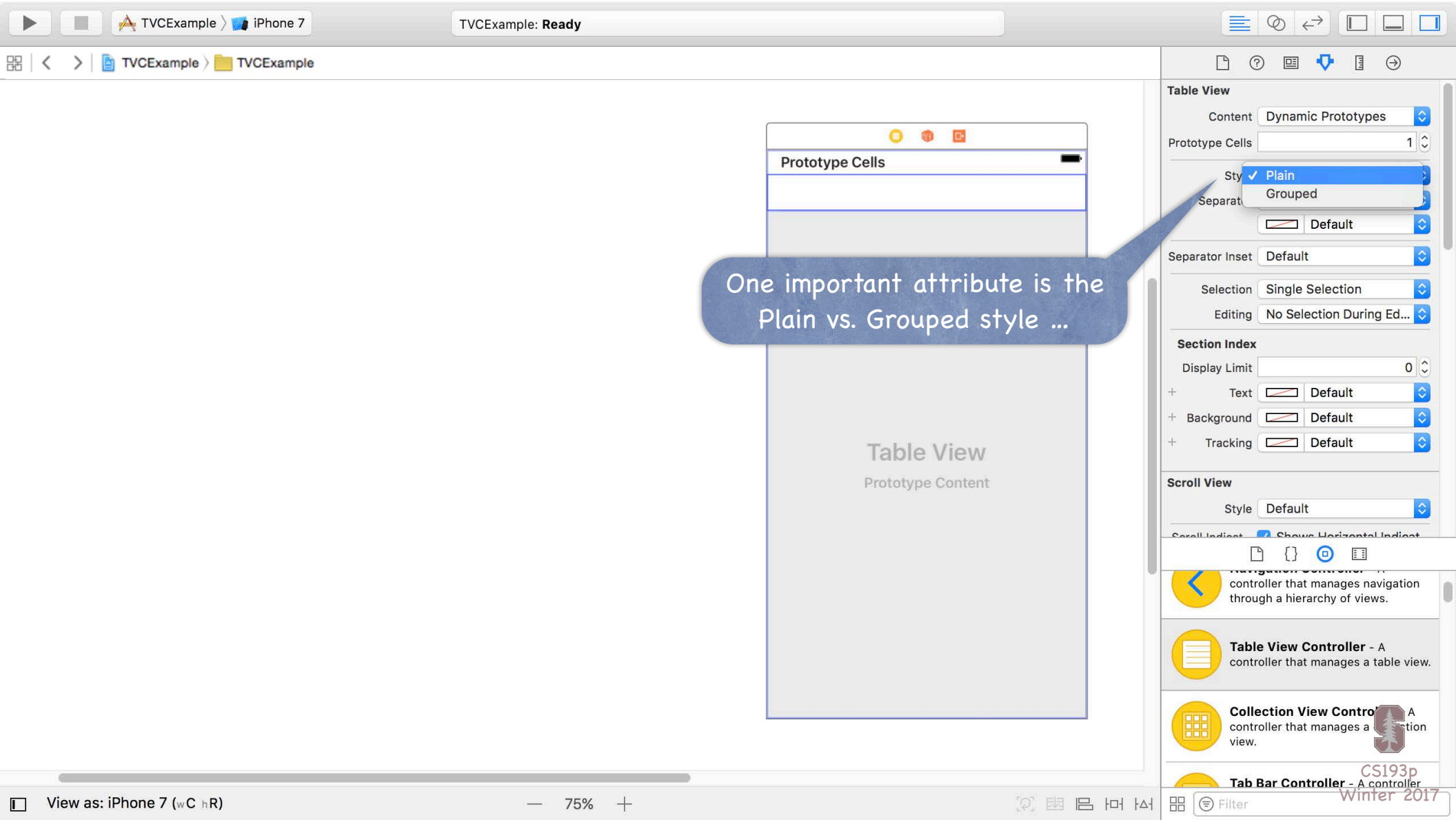

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Choose options for your new file:
Class: TableViewController
Subclass of: UITableViewController
UIView
UIViewController
UITableViewController
UITableViewCell
UICollectionViewController
Language:
Make sure you set the superclass to UITableViewController
Cancel
Previous
Next
Custom Class
Class UITableViewControl...
Module None
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro... A controller that manages a ...tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

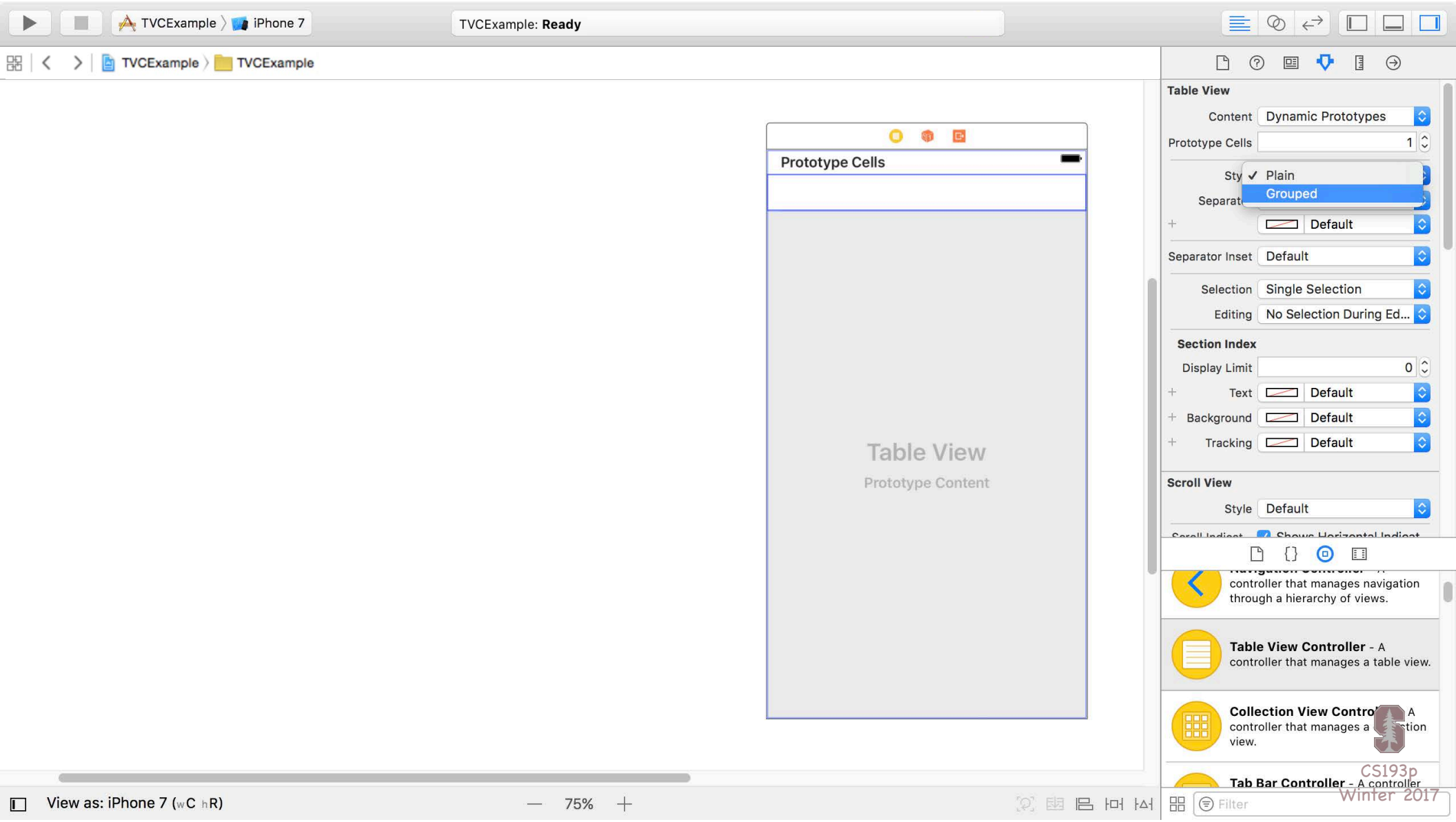

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
... otherwise it won't make sense to set it as the class here.
Table View
Prototype Content
Custom Class
Class MyTableViewControlle
Module Current – TVCExam...
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

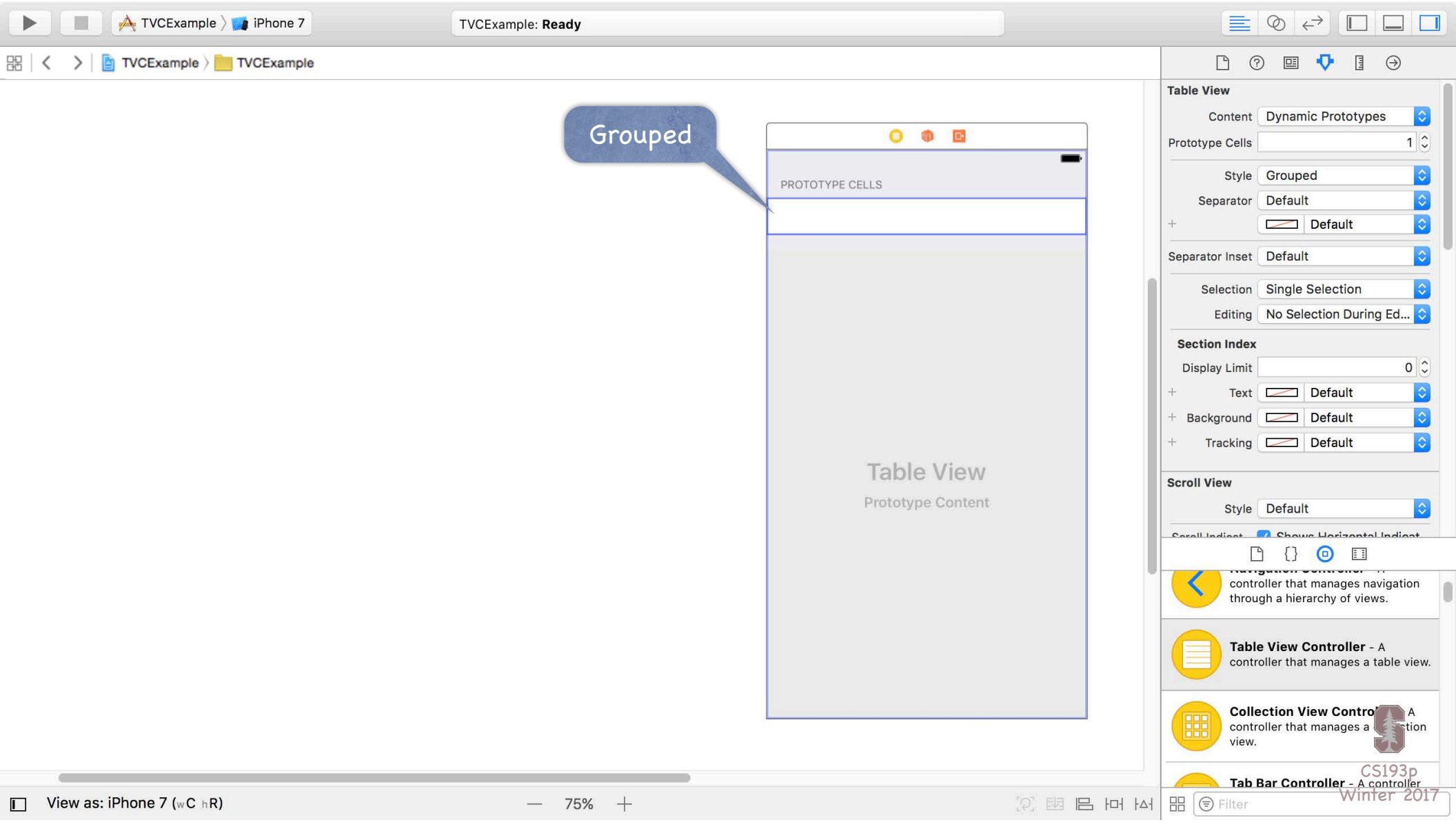

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Your UITableViewController subclass will also serve as the UITableView's dataSource and delegate (more on this in a moment).
Prototype Cells
You can see that if you right-click the Controller here.
Table View
Prototype Content
Custom Class
Class MyTableViewControlle
Module Current – TVCExam...
Identity
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

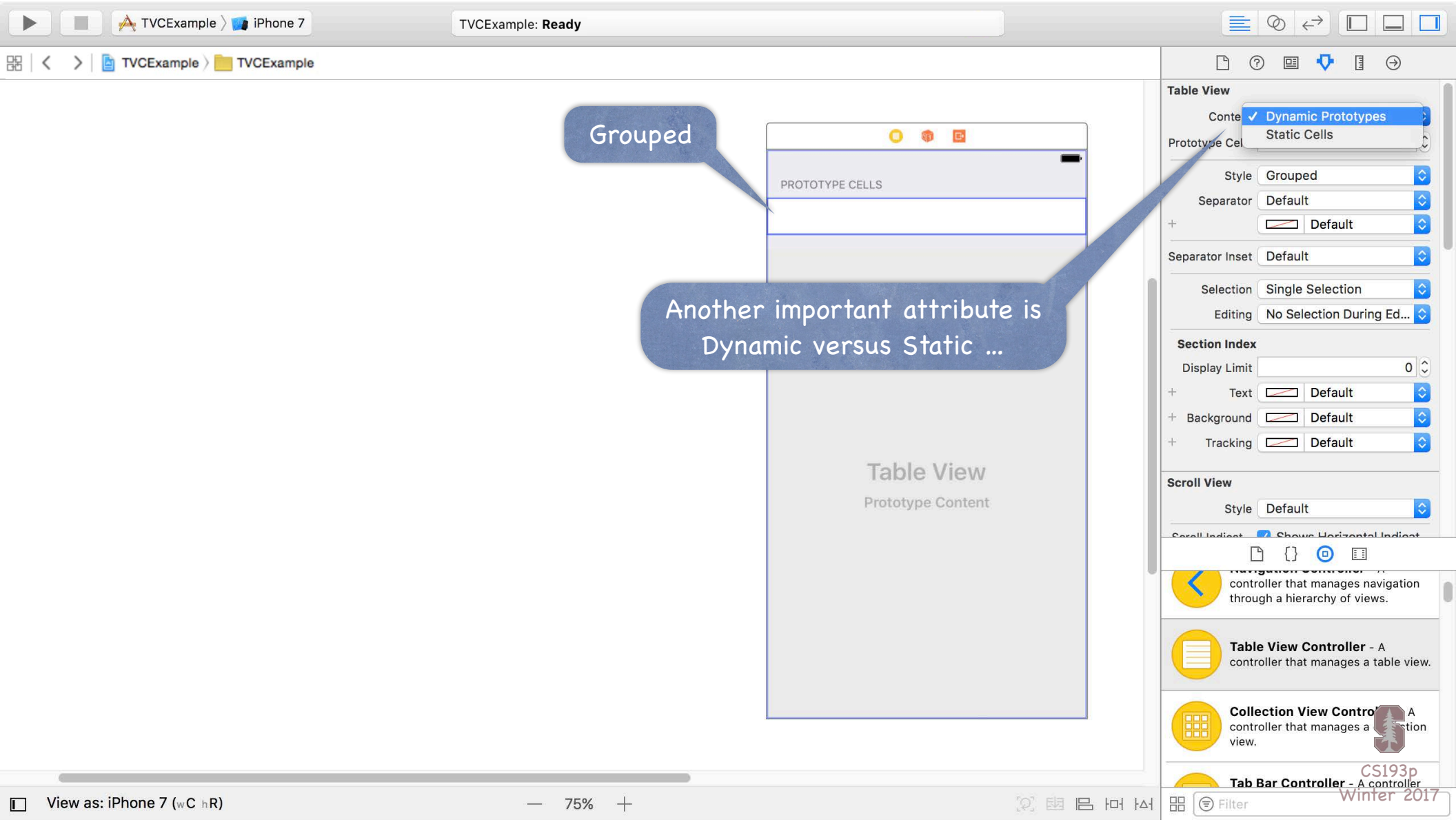

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
My Table View Controller
Triggered Segues
Outlets
searchDisplayController
view
Table View
Presenting Segues
Relationship
Show
Show Detail
Present Modally
Present As Popover
Embed
Push (deprecated)
Modal (deprecated)
Custom
Referencing Outlets
dataSource
Table View
delegate
Table View
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
You can see that if you right-click the Controller here.
dataSource and delegate properties
If you use UITableView without UITableViewController, you'll have to wire these up yourself.
Custom Class
Class MyTableViewContro...
Module Current – TVCExam...
Identity
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID ab2-9w-rly
Lock Inherited - (Nothing)
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro... A controller that manages a ...tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

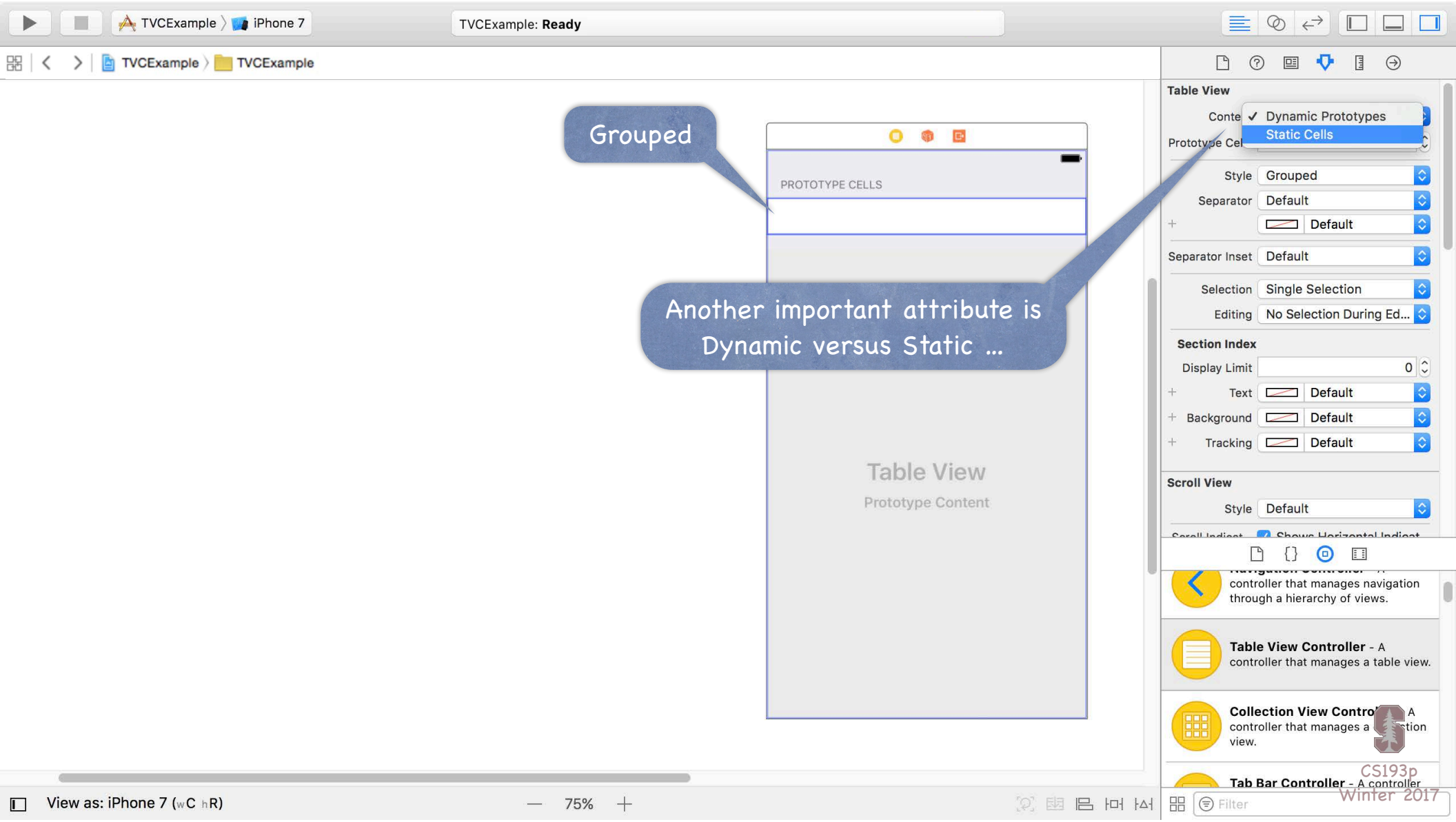


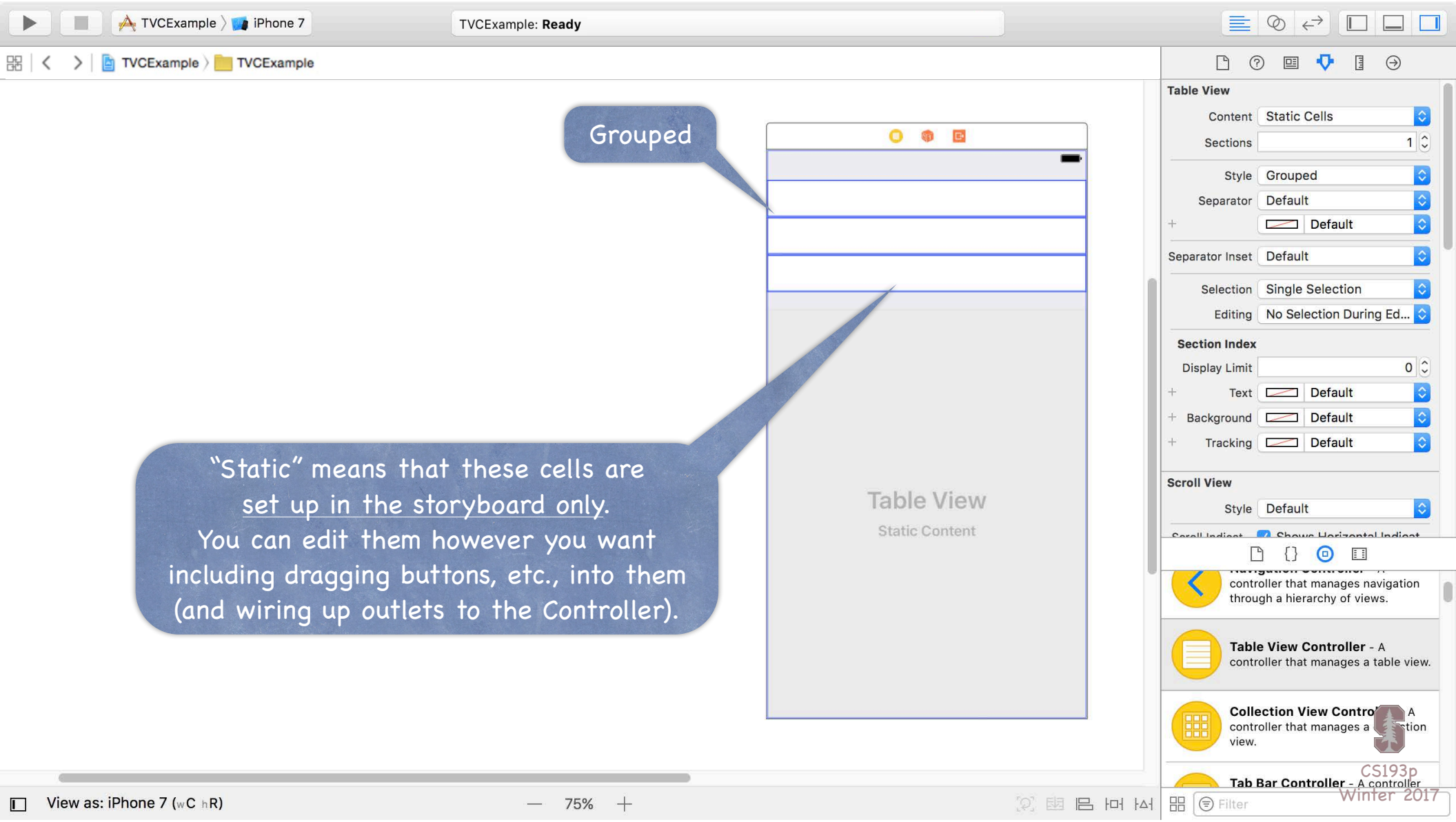


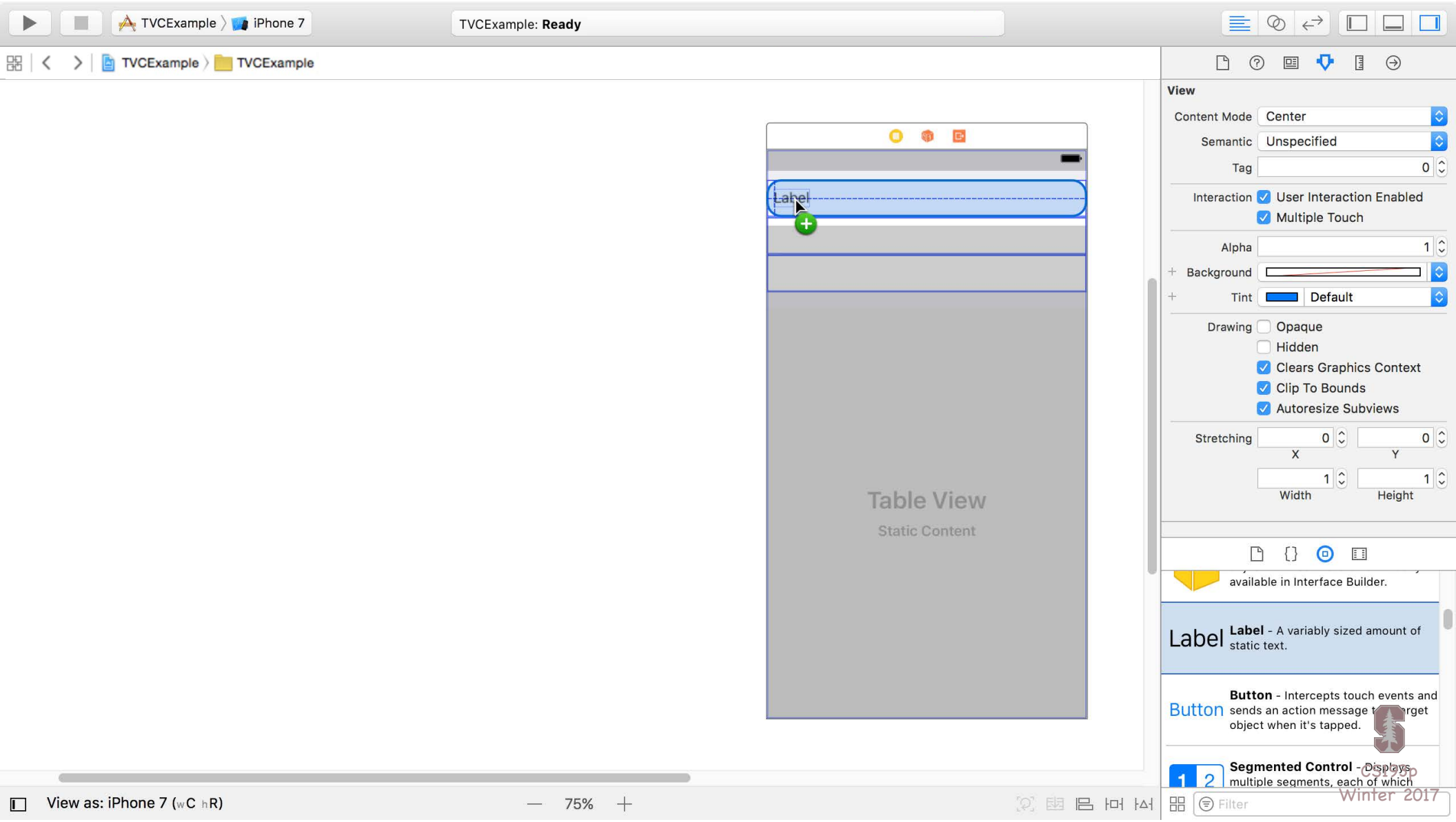

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Table View
Prototype Content
Table View
Content Dynamic Prototypes
Prototype Cells 1
Plain
Grouped
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

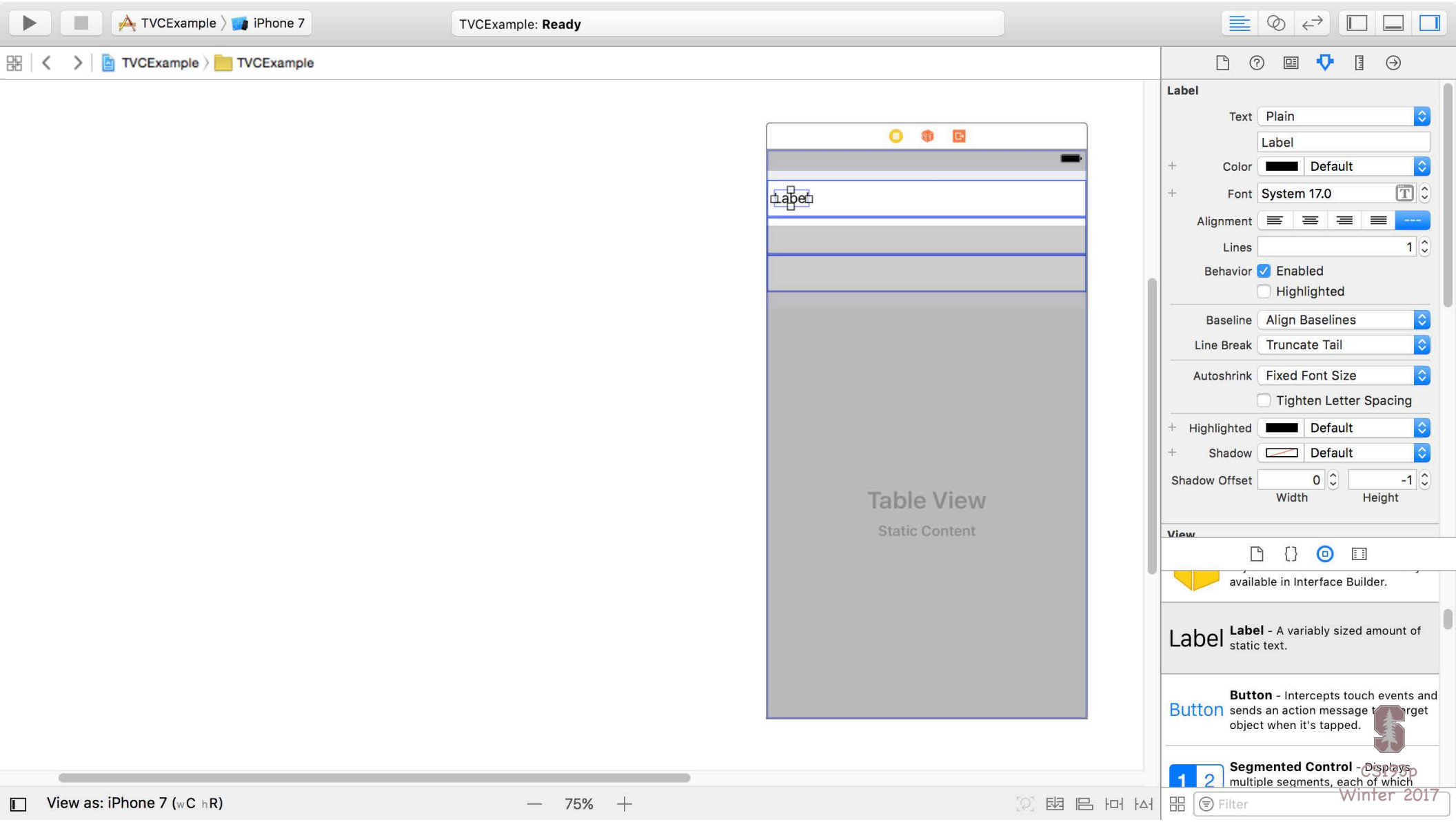

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Grouped
PROTOTYPE CELLS
Table View
Prototype Content
Table View
Content Dynamic Prototypes
Prototype Cells 1
Style Grouped
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

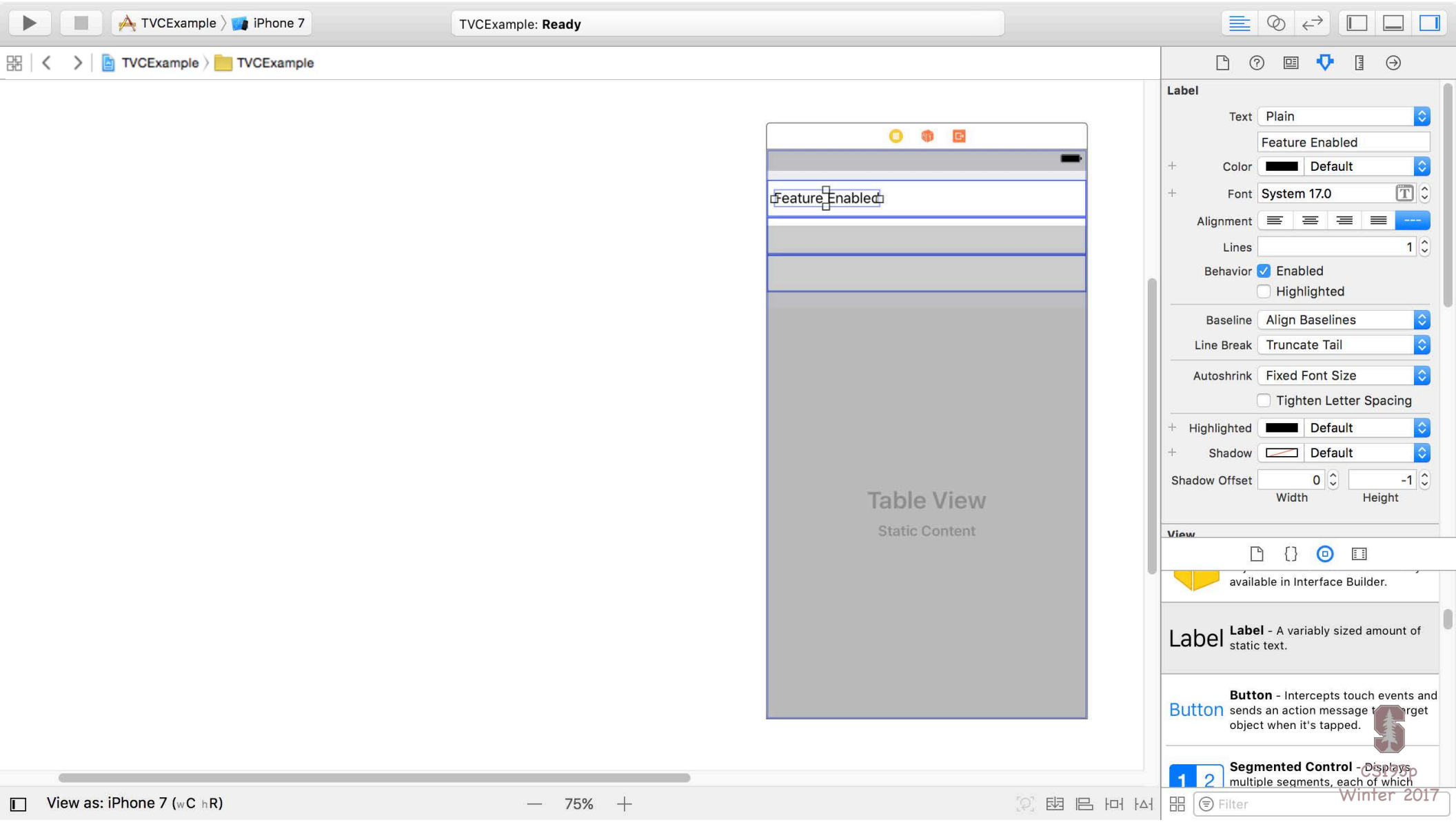


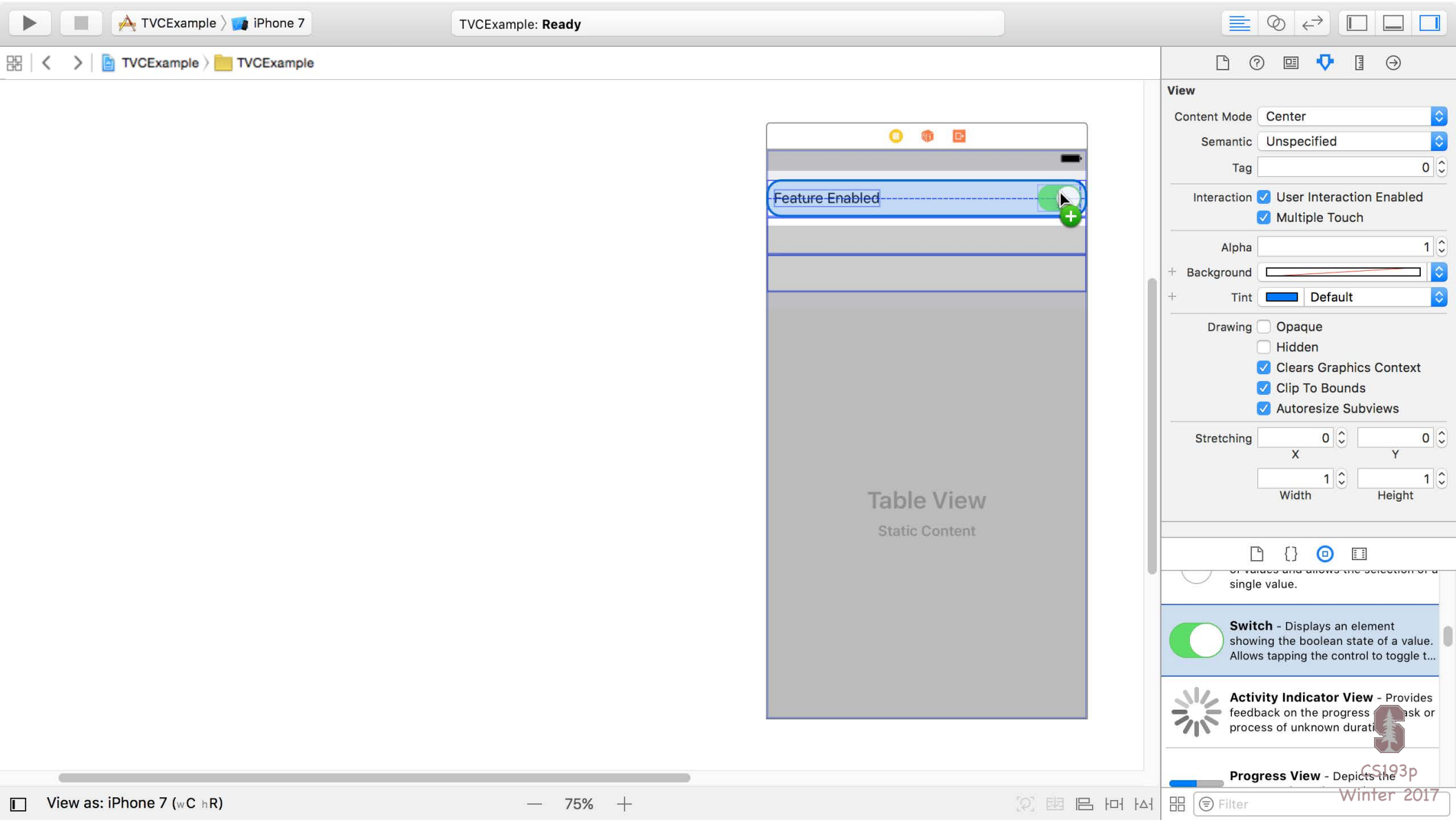


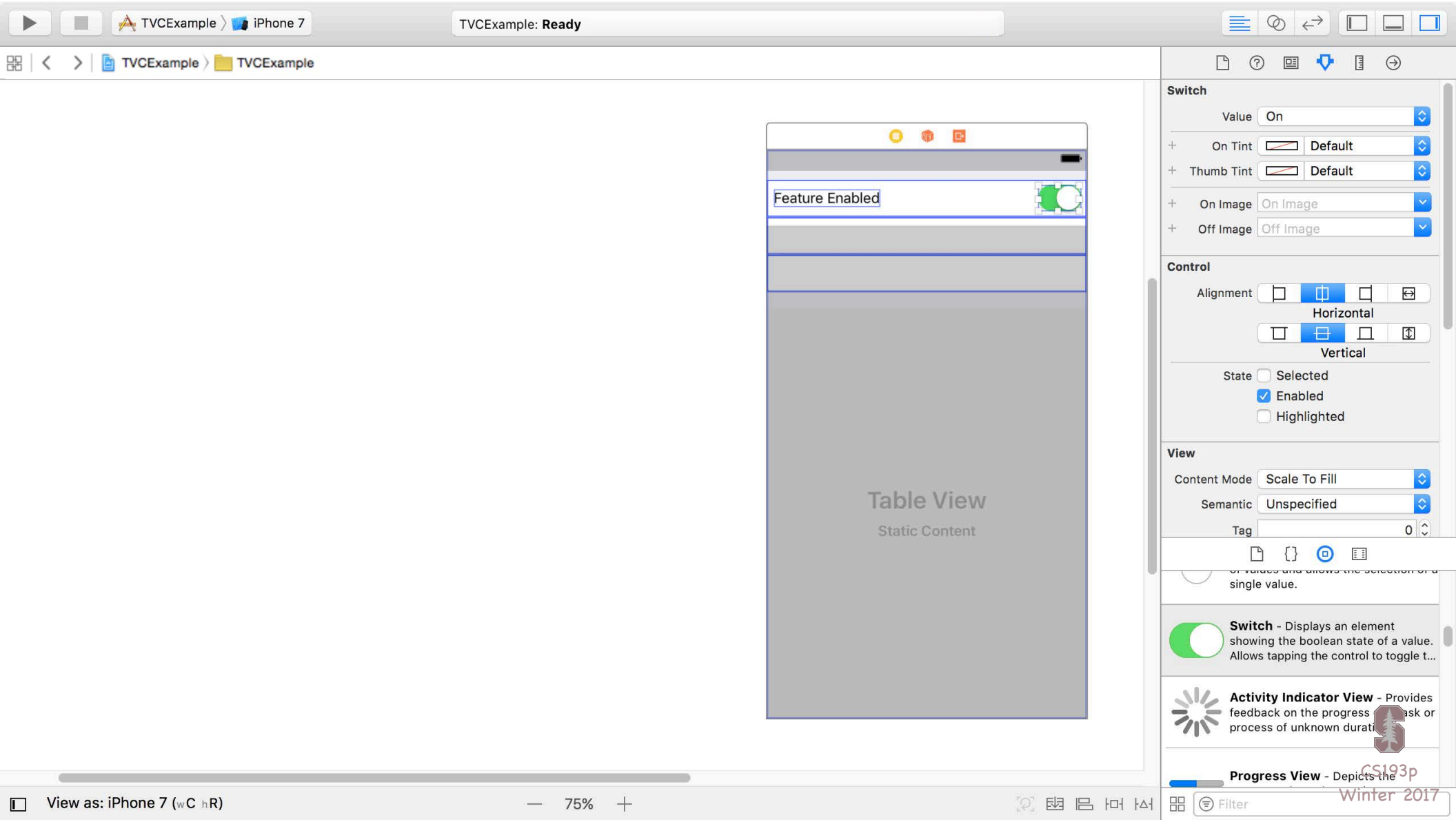

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Grouped
"Static" means that these cells are set up in the storyboard only.
You can edit them however you want including dragging buttons, etc., into them (and wiring up outlets to the Controller).
Table View
Static Content
Table View
Content Static Cells
Sections 1
Style Grouped
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
controller that manages navigation through a hierarchy of views.
Table View Controller - A controller that manages a table view.
Collection View Contro A controller that manages a tion view.
Tab Bar Controller - A controller
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

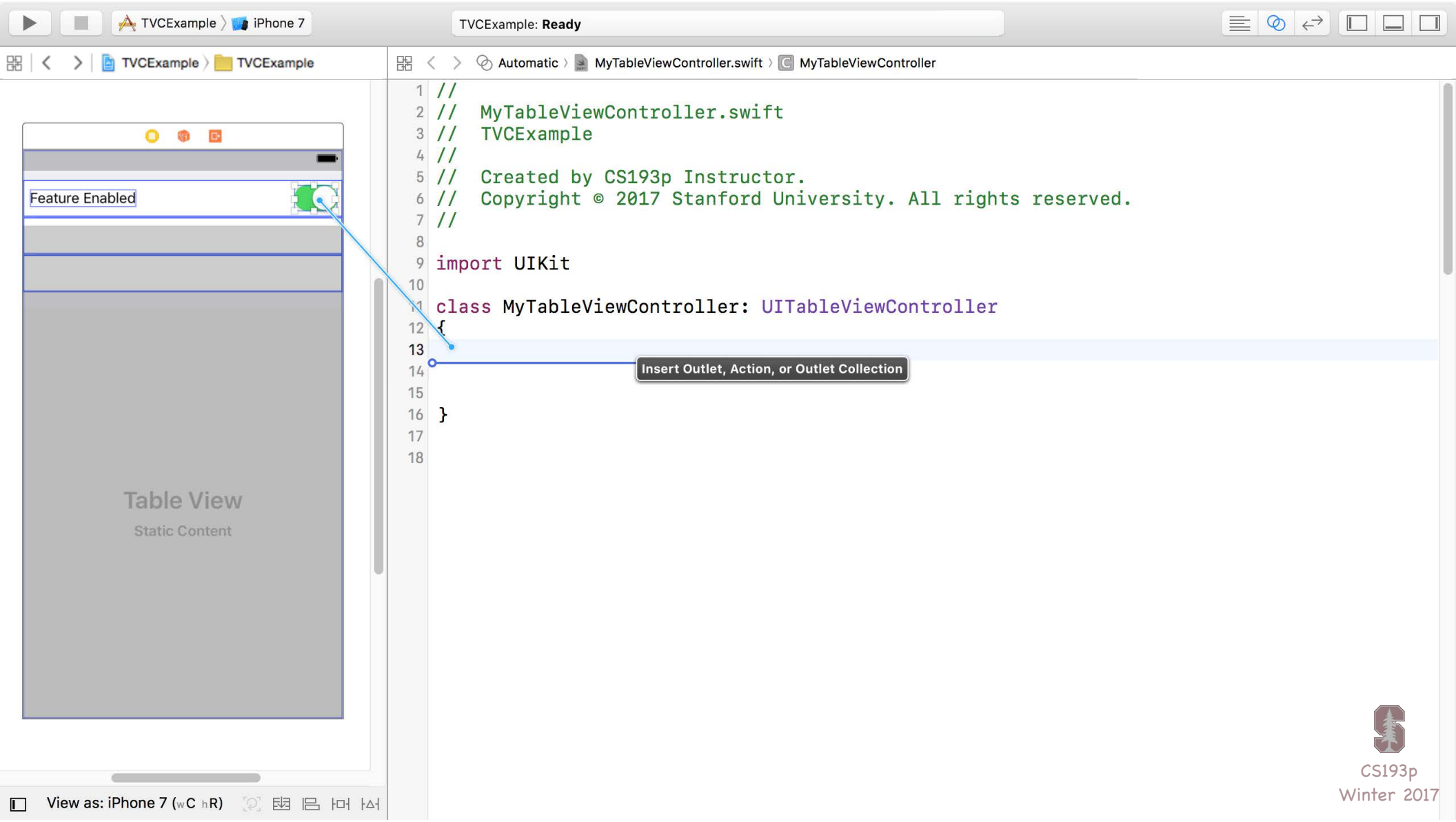

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Label
Table View
Static Content
View
Content Mode Center
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Autoresize Subviews
Stretching 0 0
X Y
1 1
Width Height
available in Interface Builder.
Label Label - A variably sized amount of static text.
Button Button - Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control - Displays multiple segments, each of which
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Label
Table View
Static Content
Label
Text Plain
Label
Color Default
Font System 17.0
Alignment
Lines 1
Behavior Enabled
Highlighted
Baseline Align Baselines
Line Break Truncate Tail
Autoshrink Fixed Font Size
Tighten Letter Spacing
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Width Height
View
available in Interface Builder.
Label - A variably sized amount of static text.
Button - Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control - Displays multiple segments, each of which
View as: iPhone 7 (wC hR)
75%
Filter

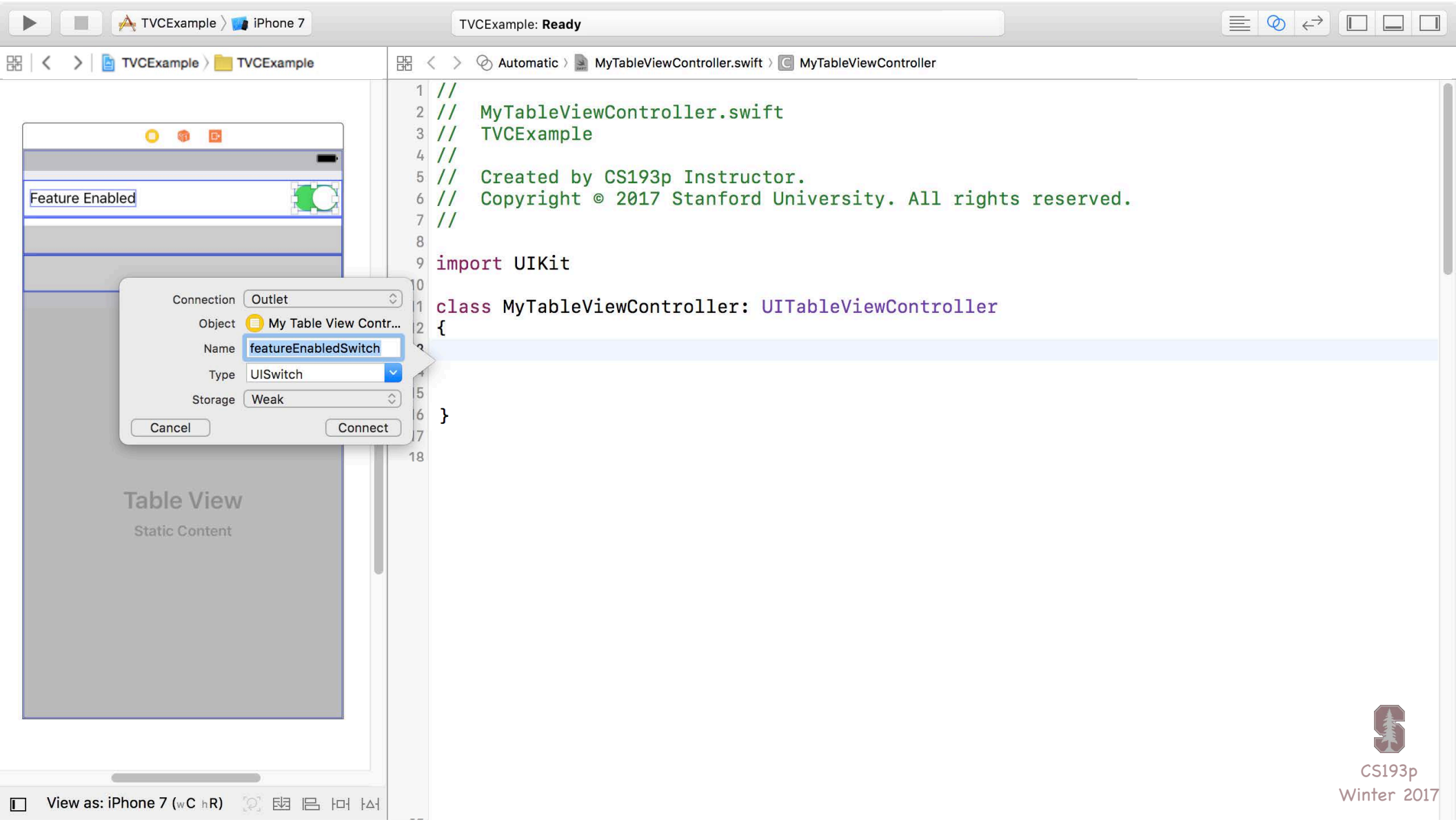

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Feature Enabled
Table View
Static Content
Label
Text
Plain
Feature Enabled
Color
Default
Font
System 17.0
Alignment
Lines
1
Behavior
Enabled
Highlighted
Baseline
Align Baselines
Line Break
Truncate Tail
Autoshrink
Fixed Font Size
Tighten Letter Spacing
Highlighted
Default
Shadow
Default
Shadow Offset
0
Width
-1
Height
available in Interface Builder.
Label
Label - A variably sized amount of static text.
Button
Button - Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control - Displays multiple segments, each of which
View as: iPhone 7 (wC hR)
75%
Filter


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Feature Enabled
Table View
Static Content
View
Content Mode Center
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Autoresize Subviews
Stretching 0 X 0 Y
1 Width 1 Height
single value.
Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle t...
Activity Indicator View - Provides feedback on the progress
process of unknown durati
Progress View - Depicts the
CS193p
Winter 2017
View as: iPhone 7 (wC hR)
75%
Filter

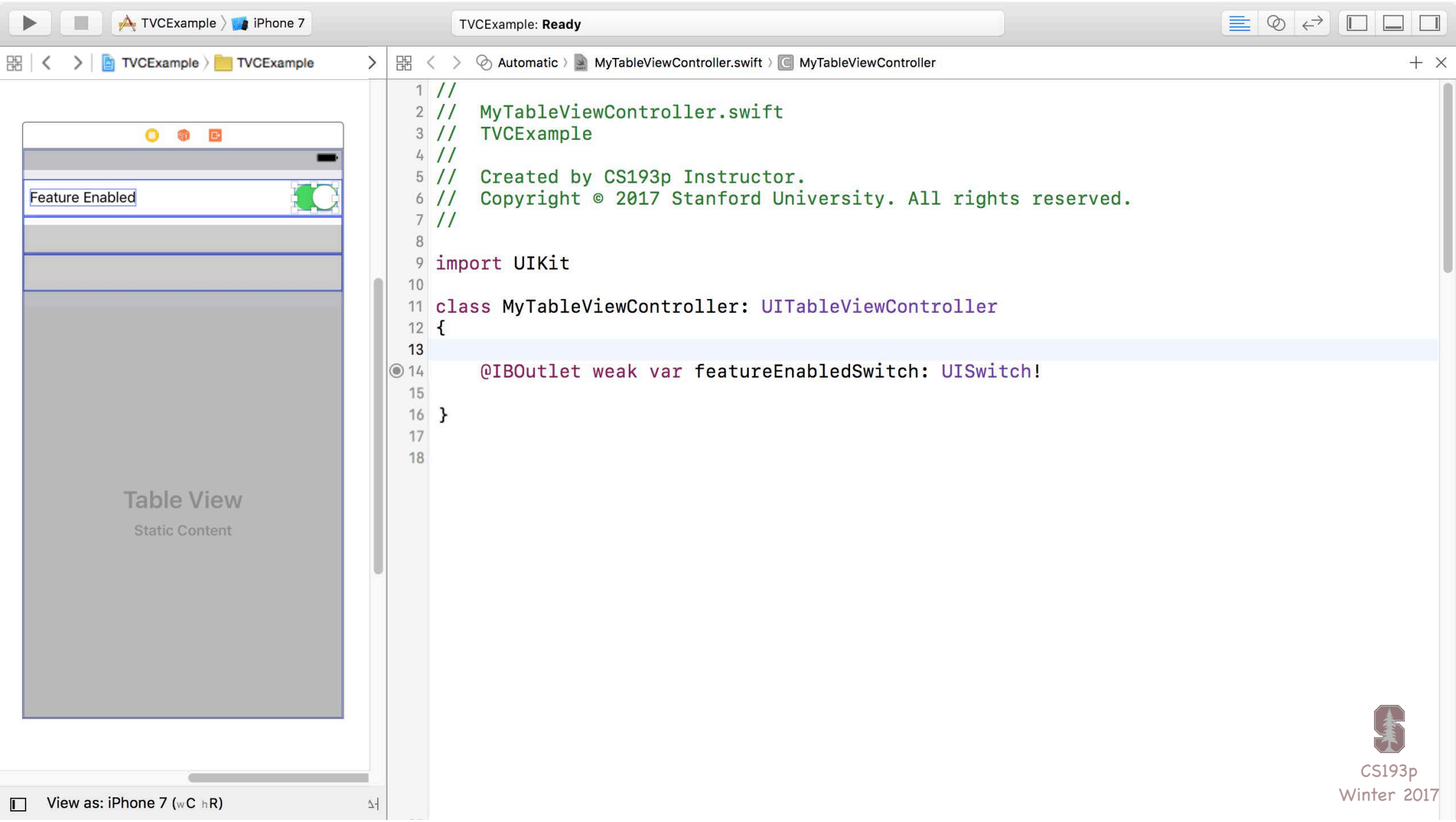

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Feature Enabled
Table View
Static Content
Switch
Value On
On Tint Default
Thumb Tint Default
On Image On Image
Off Image Off Image
Control
Alignment
Horizontal
Vertical
State Selected
Enabled
Highlighted
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
single value.
Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle t...
Activity Indicator View - Provides feedback on the progress
process of unknown durati
Progress View - Depicts the
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

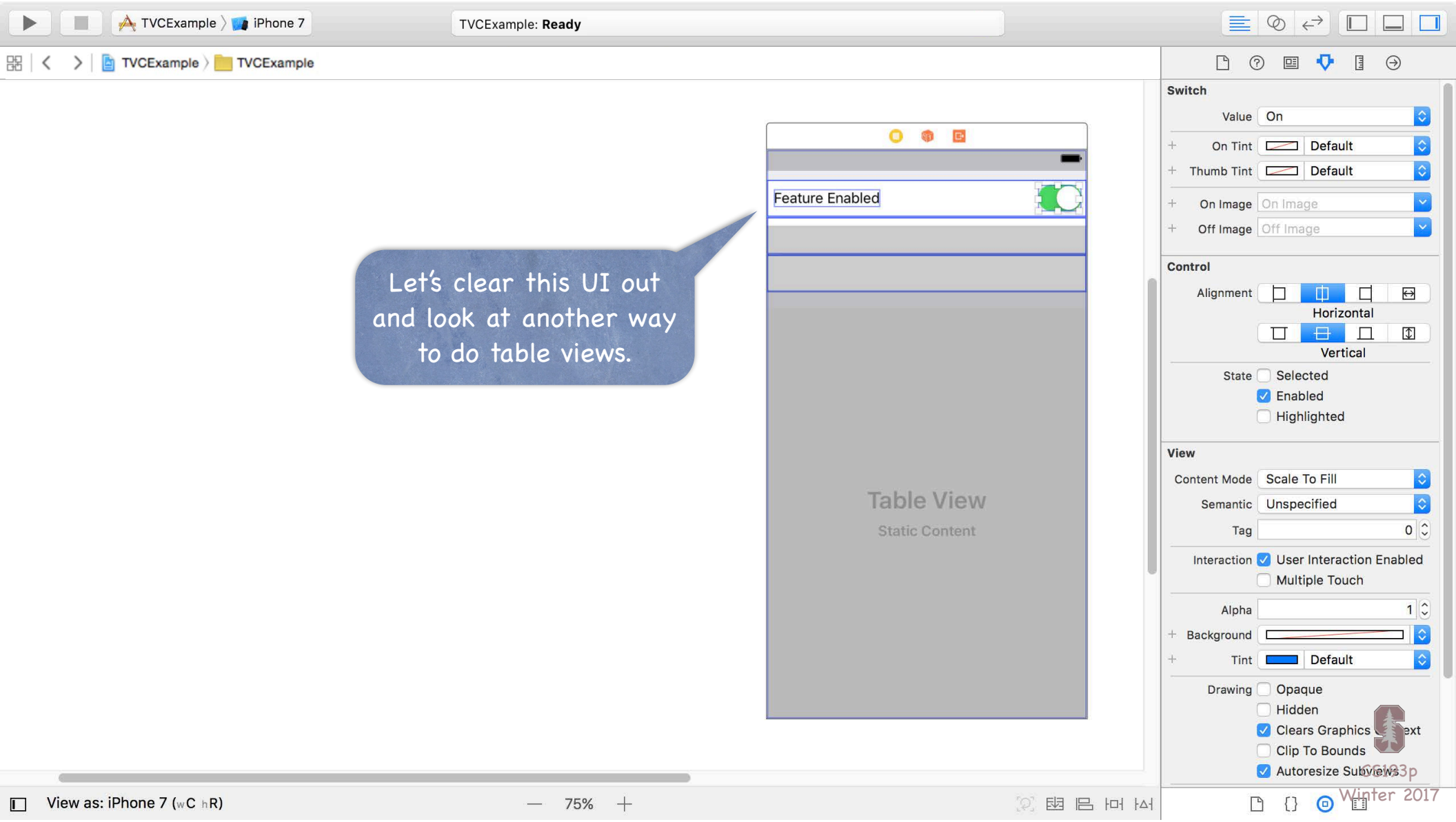


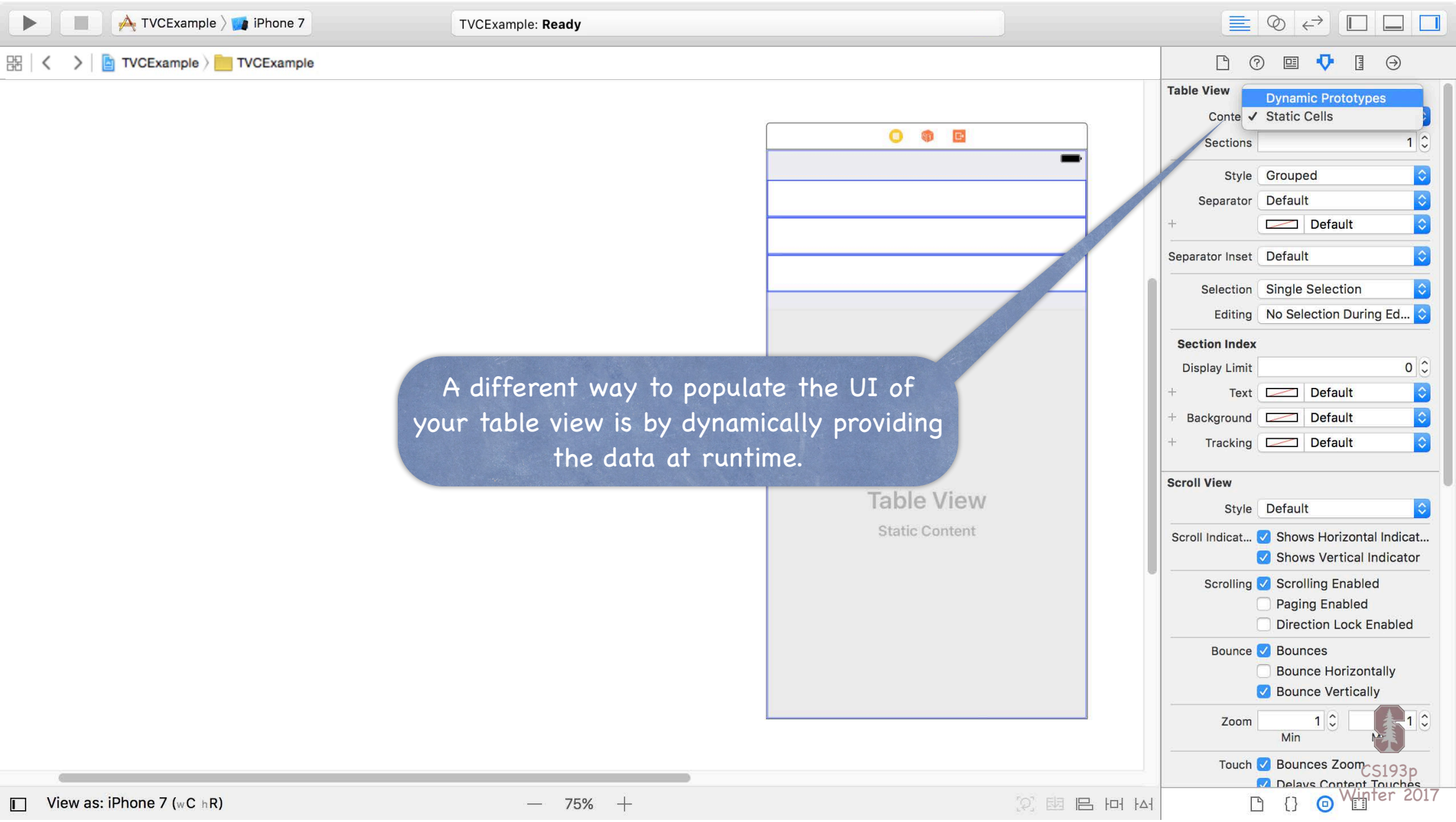
CS193p
Winter 2017

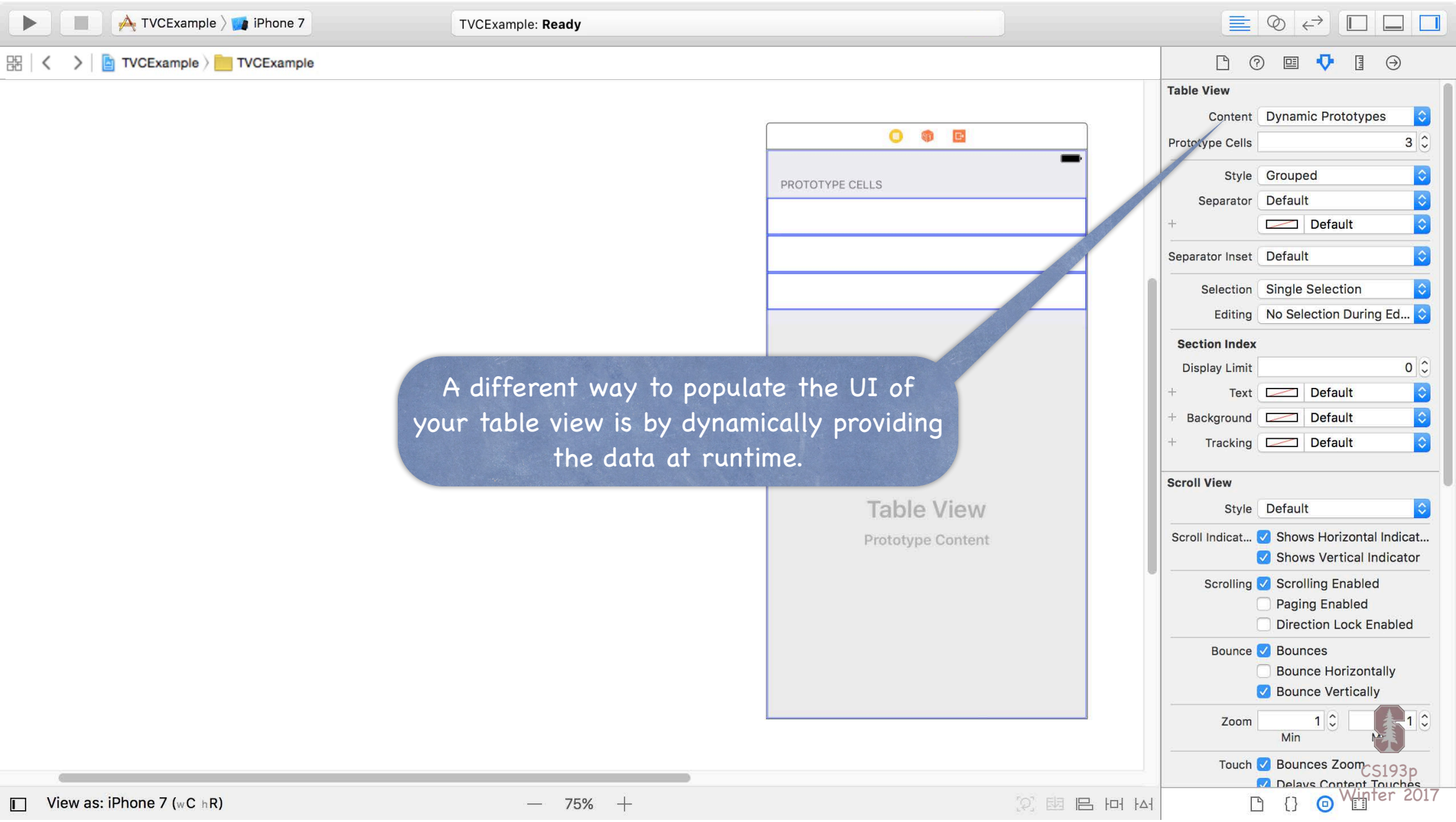


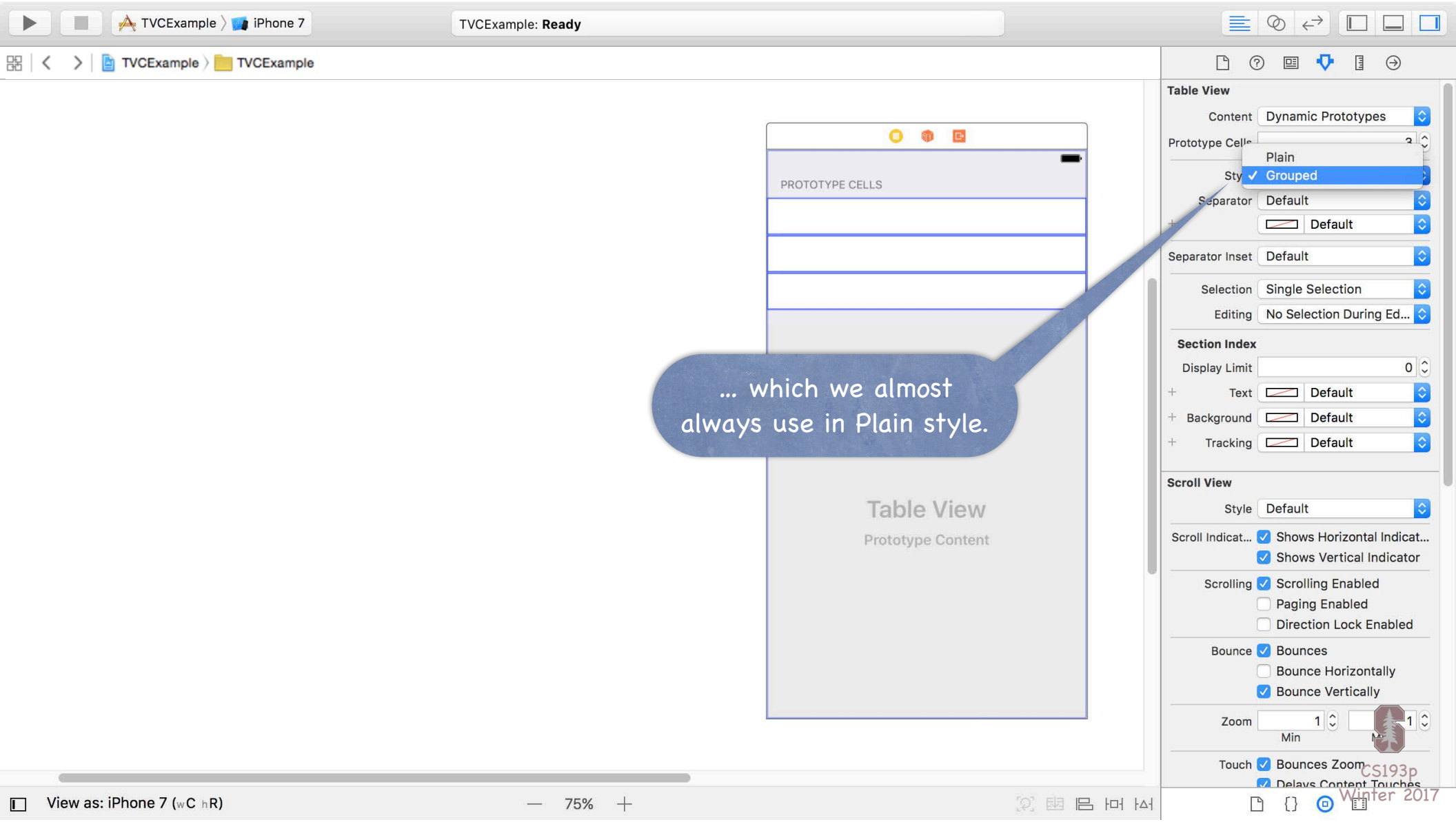
CS193p
Winter 2017

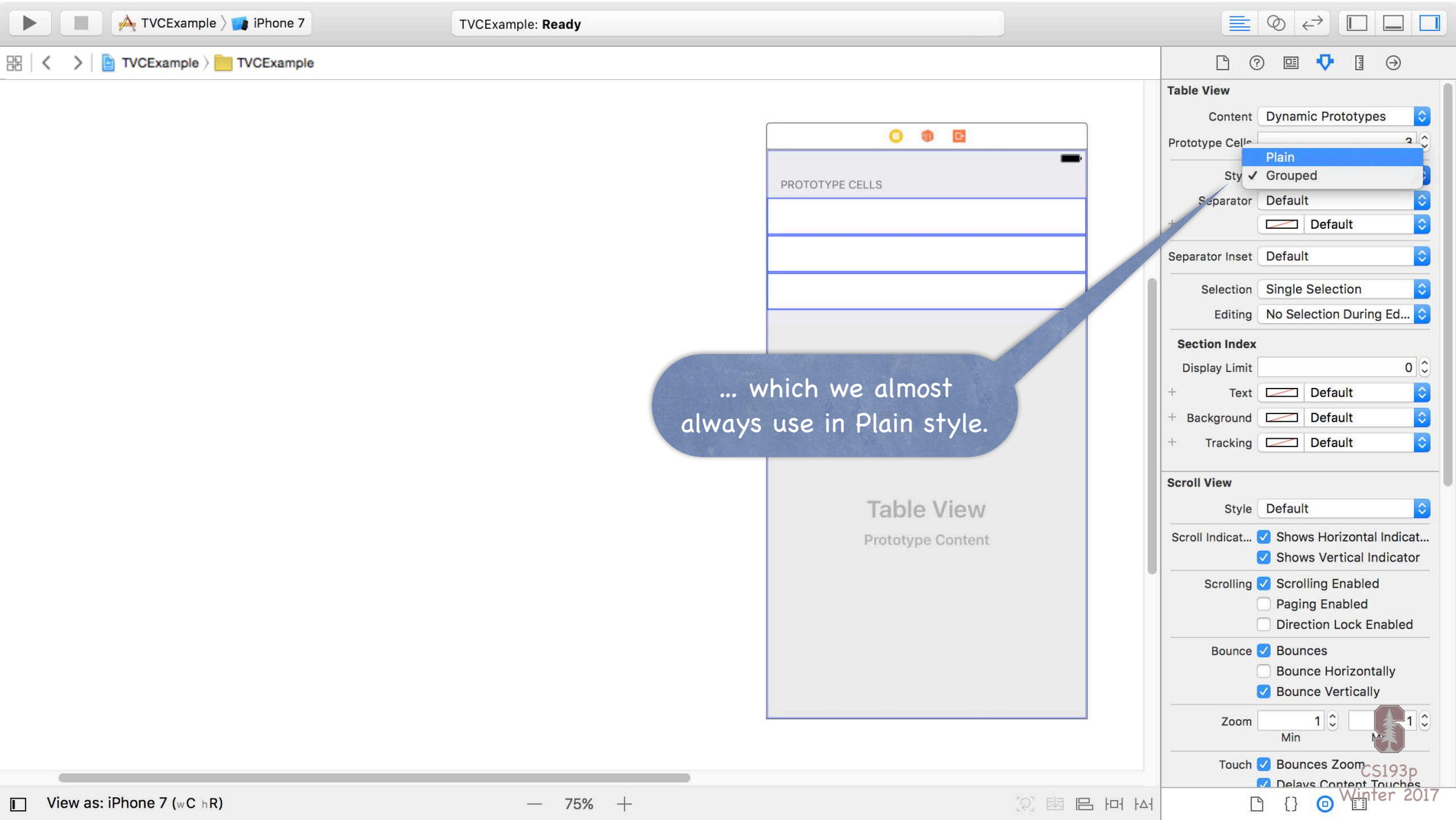

TVCExample
iPhone 7
TVCExample: Ready
TVCExample
TVCExample
Automatic
MyTableViewController.swift
MyTableViewController
Feature Enabled
Table View
Static Content
View as: iPhone 7 (wC hR)
1 //
2 // MyTableViewController.swift
3 // TVCExample
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14   @IBOutlet weak var featureEnabledSwitch: UISwitch!
15
16 }
17
18
CS193p
Winter 2017

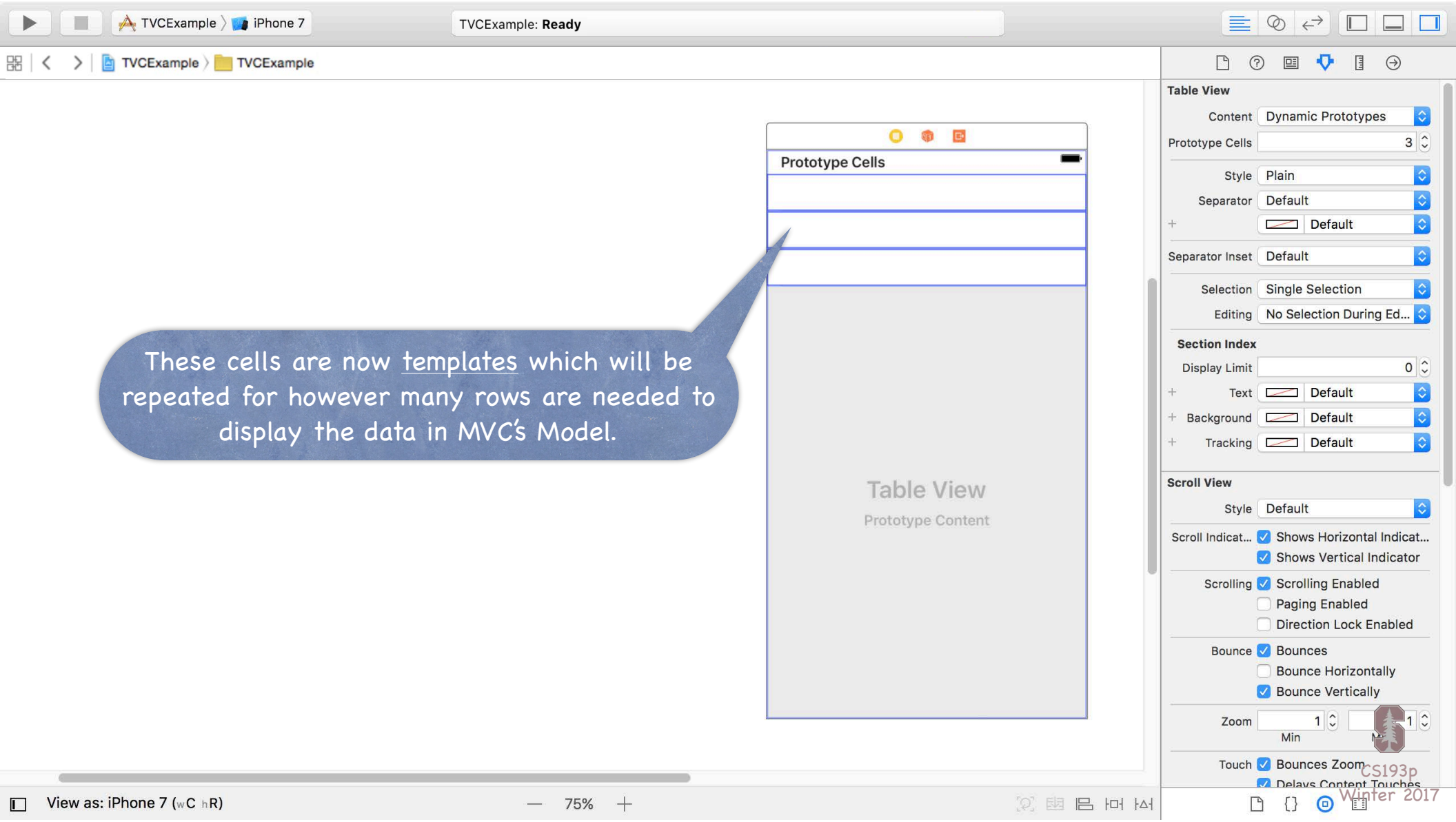

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Let's clear this UI out and look at another way to do table views.
Feature Enabled
Table View
Static Content
Switch
Value On
On Tint Default
Thumb Tint Default
On Image On Image
Off Image Off Image
Control
Alignment
Horizontal
Vertical
State Selected
Enabled
Highlighted
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

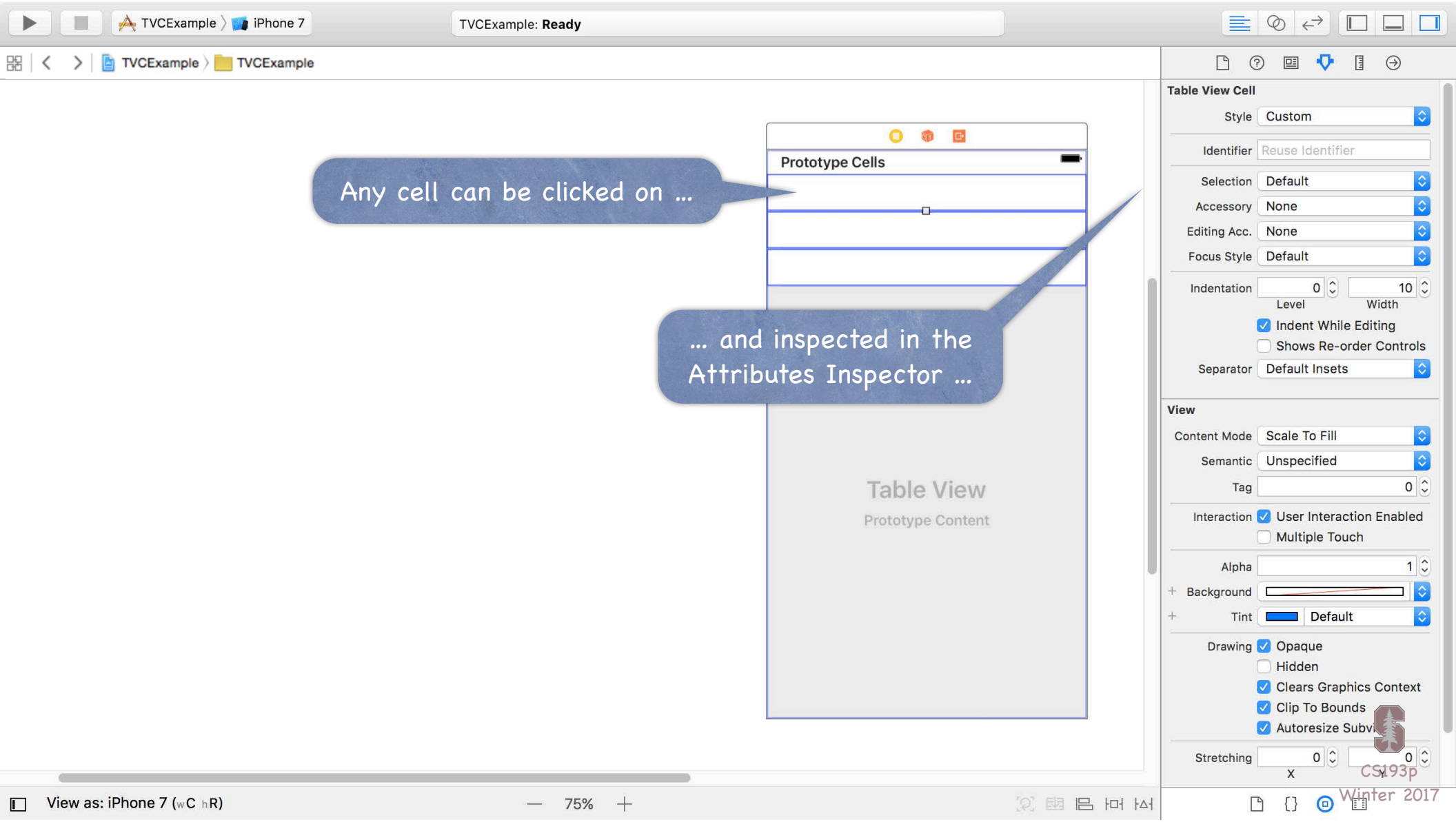

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Table View
Dynamic Prototypes
Static Cells
Sections
Style
Grouped
Separator
Default
Separator Inset
Selection
Single Selection
Editing
No Selection During Ed...
Section Index
Display Limit
Text
Background
Tracking
Scroll View
Scroll Indicat...
Shows Horizontal Indicat...
Shows Vertical Indicator
Scrolling
Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce
Bounces
Bounce Horizontally
Bounce Vertically
Zoom
Min
Touch
Bounces Zoom
A different way to populate the UI of your table view is by dynamically providing the data at runtime.
Table View
Static Content
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

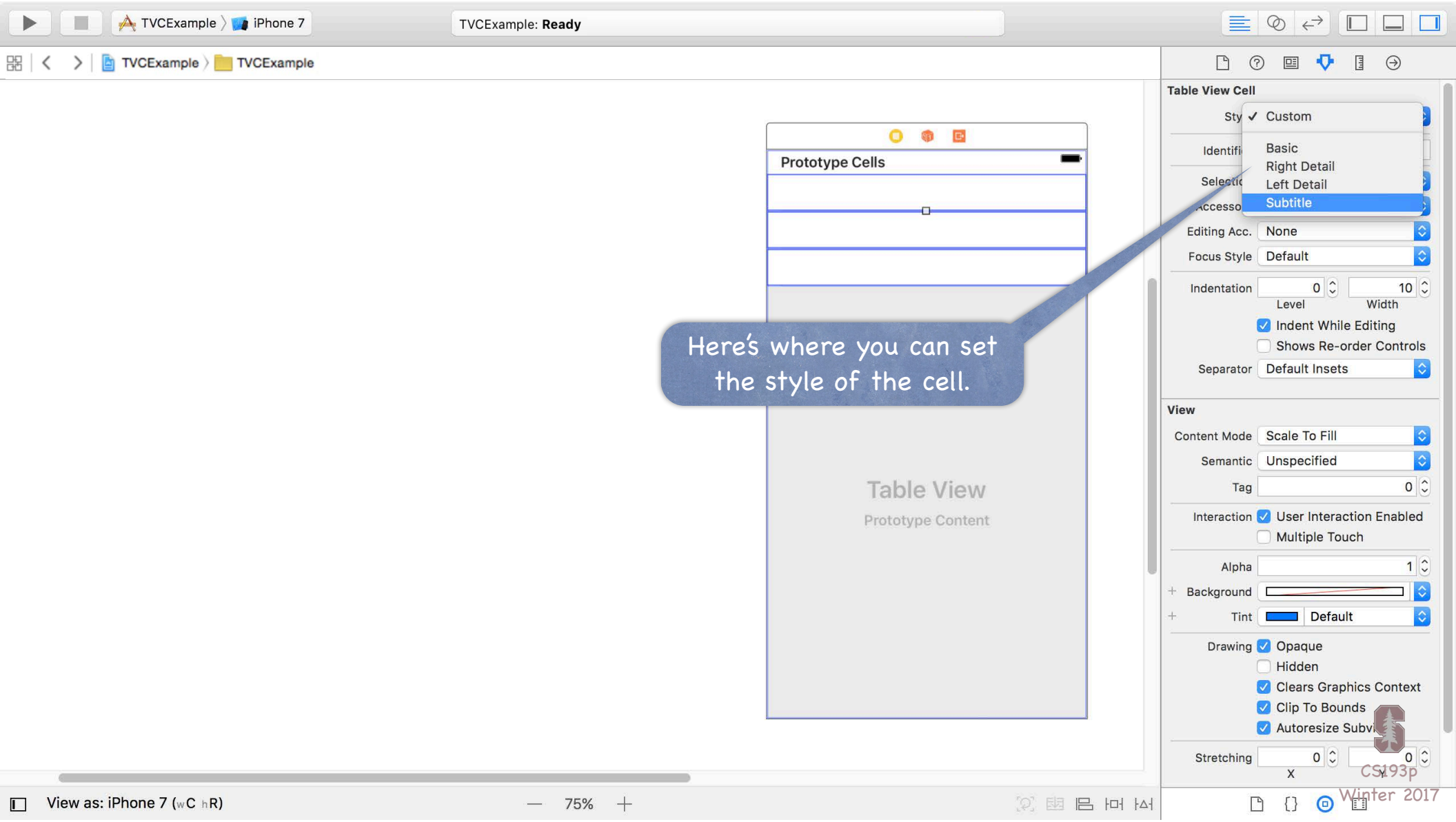

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
PROTOTYPE CELLS
Table View
Prototype Content
A different way to populate the UI of your table view is by dynamically providing the data at runtime.
Table View
Content Dynamic Prototypes
Prototype Cells 3
Style Grouped
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
Scroll Indicat... Shows Horizontal Indicat...
Shows Vertical Indicator
Scrolling Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce Bounces
Bounce Horizontally
Bounce Vertically
Zoom 1 1
Min
Touch Bounces Zoom
Delays Content Touches
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

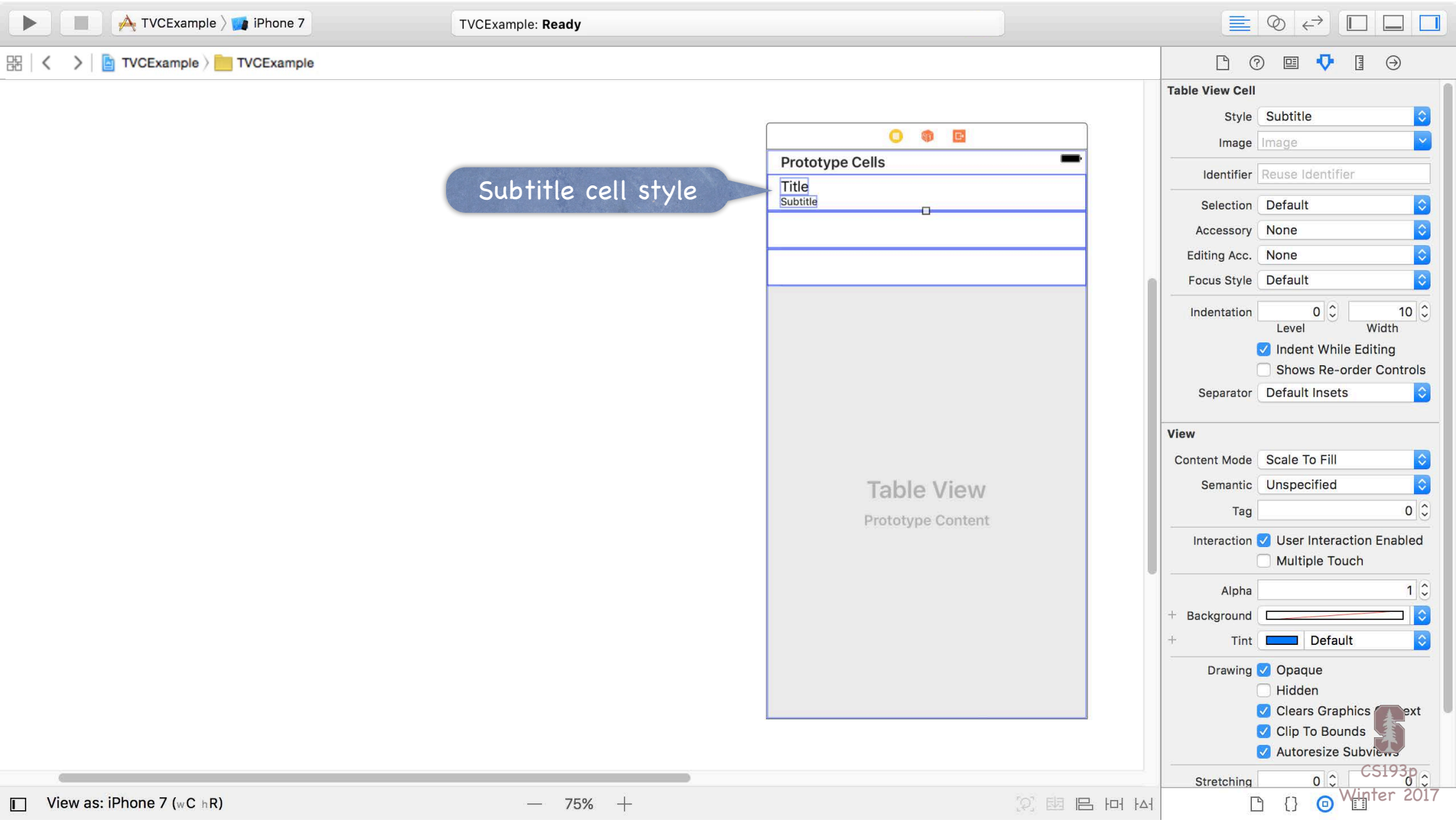

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
PROTOTYPE CELLS
Table View
Prototype Content
... which we almost always use in Plain style.
Table View
Content Dynamic Prototypes
Prototype Cells
Plain
Grouped
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
Scroll Indicat... Shows Horizontal Indicat...
Shows Vertical Indicator
Scrolling Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce Bounces
Bounce Horizontally
Bounce Vertically
Zoom 1 1
Min
Touch Bounces Zoom
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

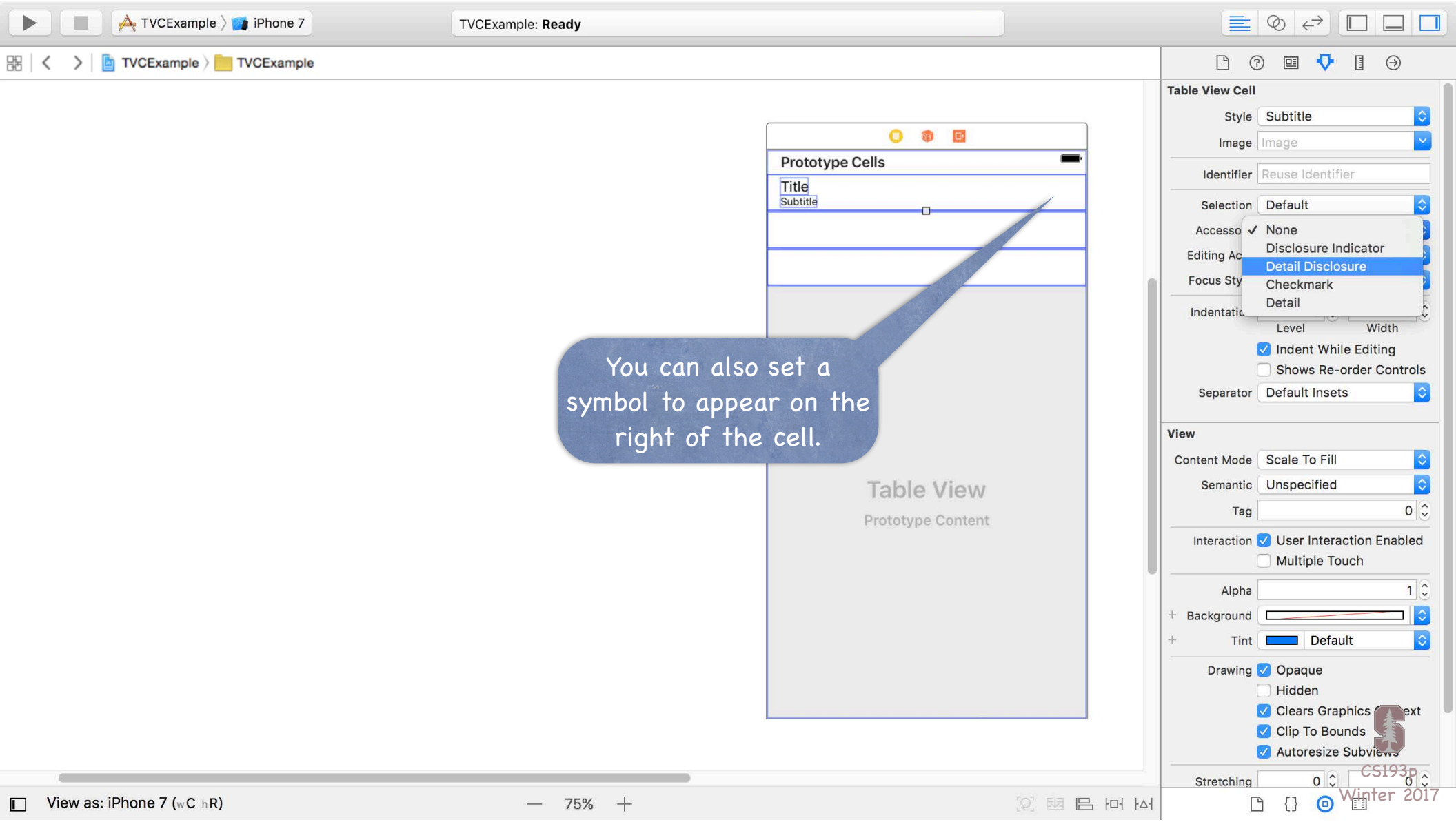

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
PROTOTYPE CELLS
Table View
Prototype Content
… which we almost always use in Plain style.
Table View
Content Dynamic Prototypes
Prototype Cells 3
Plain
Grouped
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
Scroll Indicat... Shows Horizontal Indicat...
Shows Vertical Indicator
Scrolling Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce Bounces
Bounce Horizontally
Bounce Vertically
Zoom 1 1
Min
Touch Bounces Zoom
Delays Content Touches
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

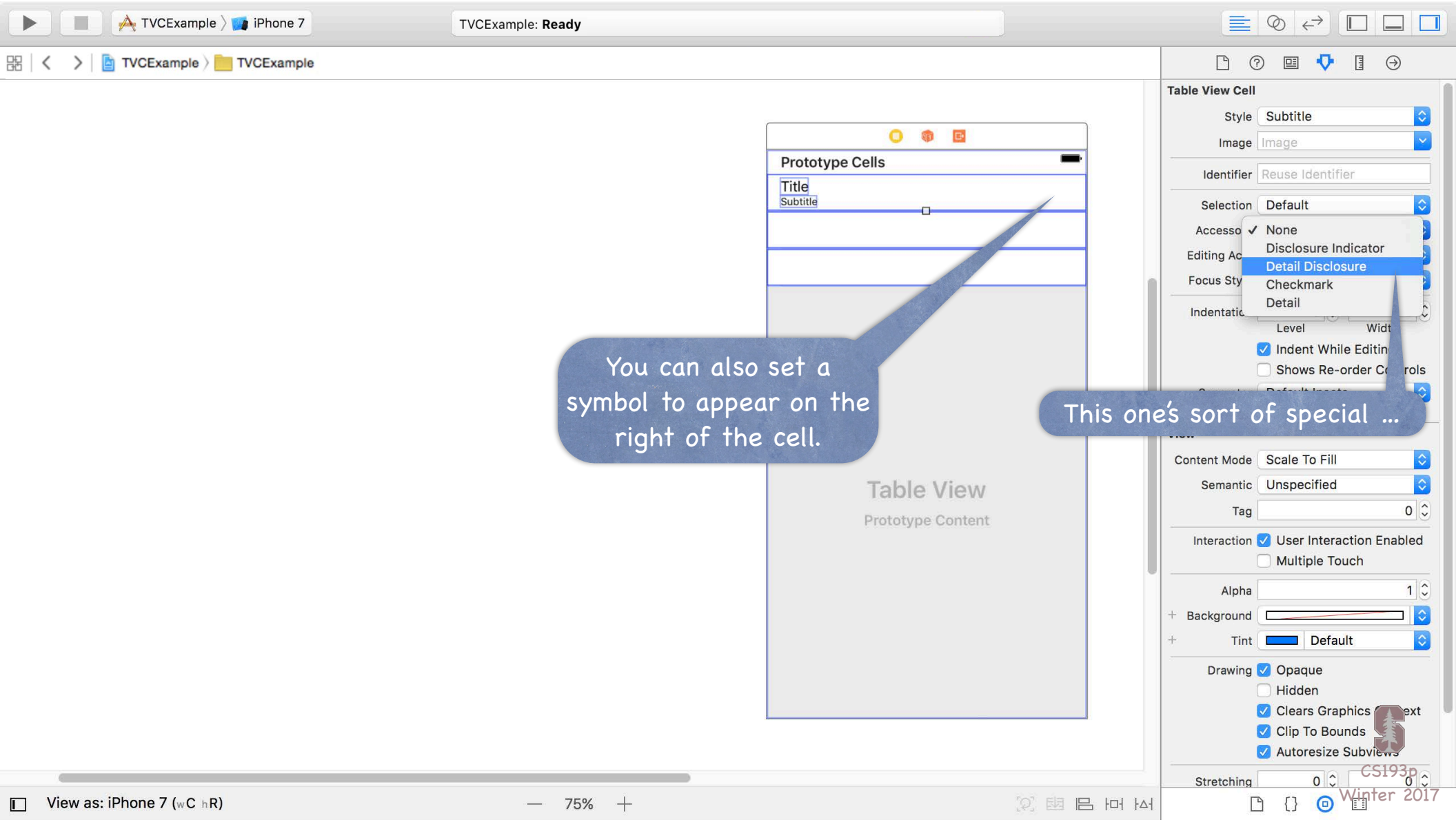

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Table View
Prototype Content
These cells are now templates which will be repeated for however many rows are needed to display the data in MVC's Model.
Table View
Content Dynamic Prototypes
Prototype Cells 3
Style Plain
Separator Default
Default
Separator Inset Default
Selection Single Selection
Editing No Selection During Ed...
Section Index
Display Limit 0
Text Default
Background Default
Tracking Default
Scroll View
Style Default
Scroll Indicat... Shows Horizontal Indicat...
Shows Vertical Indicator
Scrolling Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce Bounces
Bounce Horizontally
Bounce Vertically
Zoom 1 1
Min
Touch Bounces Zoom
Delays Content Touches
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

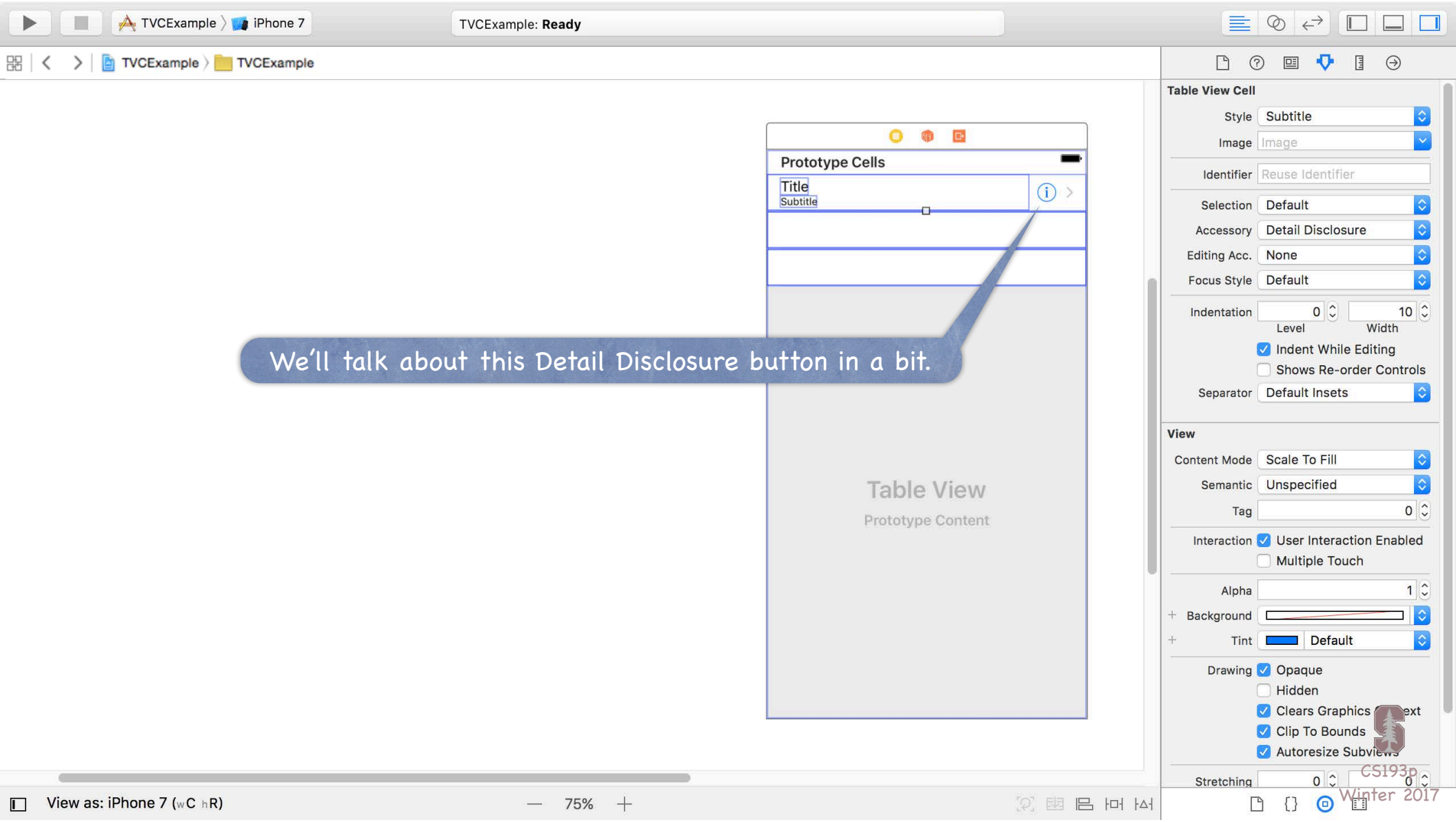

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Any cell can be clicked on …
… and inspected in the Attributes Inspector …
Prototype Cells
Table View
Prototype Content
Table View Cell
Style Custom
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Stretching 0 X 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

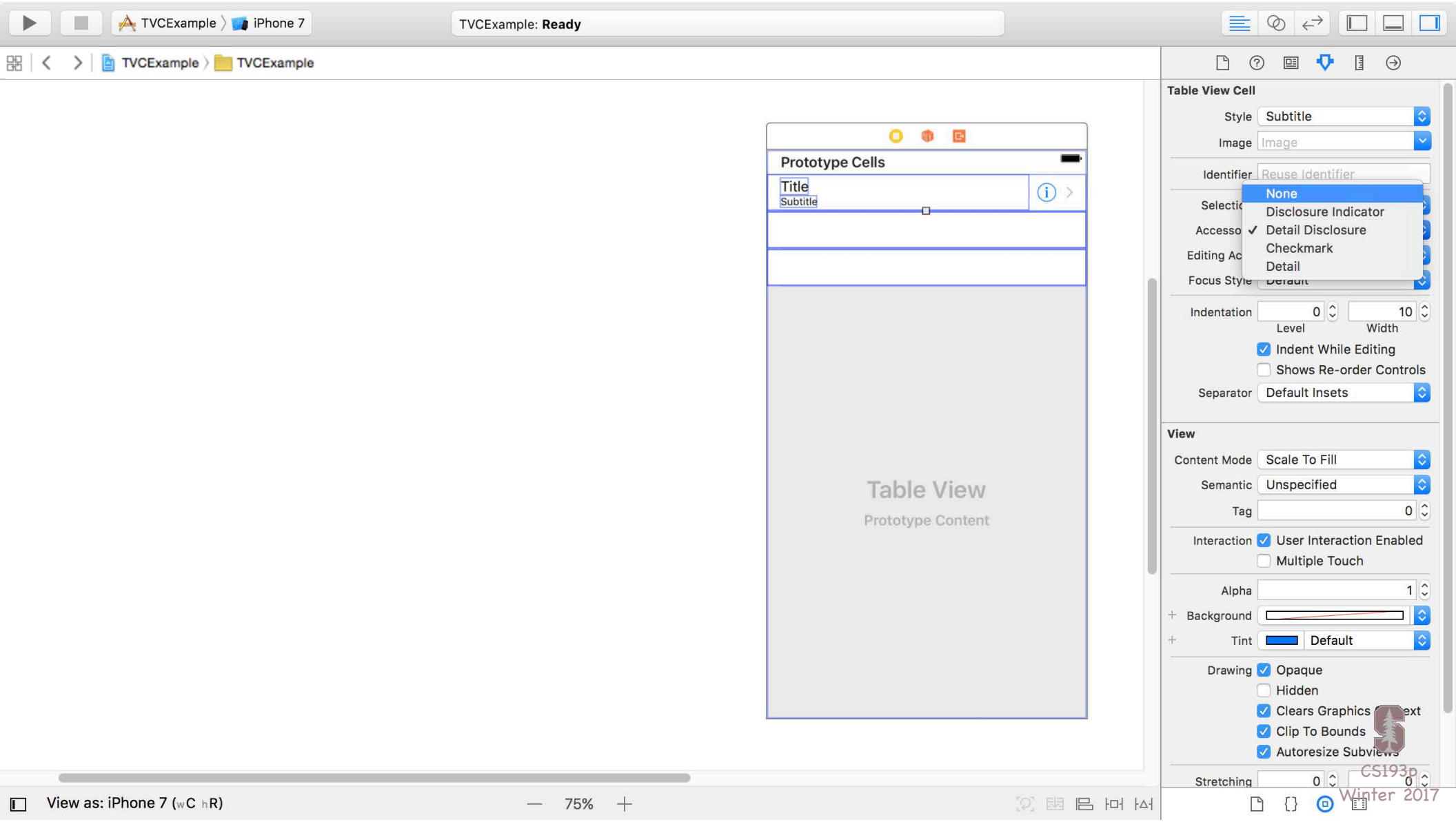

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Here's where you can set the style of the cell.
Table View
Prototype Content
Table View Cell
Sty
Custom
Basic
Right Detail
Left Detail
Subtitle
Editing Acc.
None
Focus Style
Default
Indentation
0
Level
10
Width
Indent While Editing
Shows Re-order Controls
Separator
Default Insets
View
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Stretching
0
X
0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

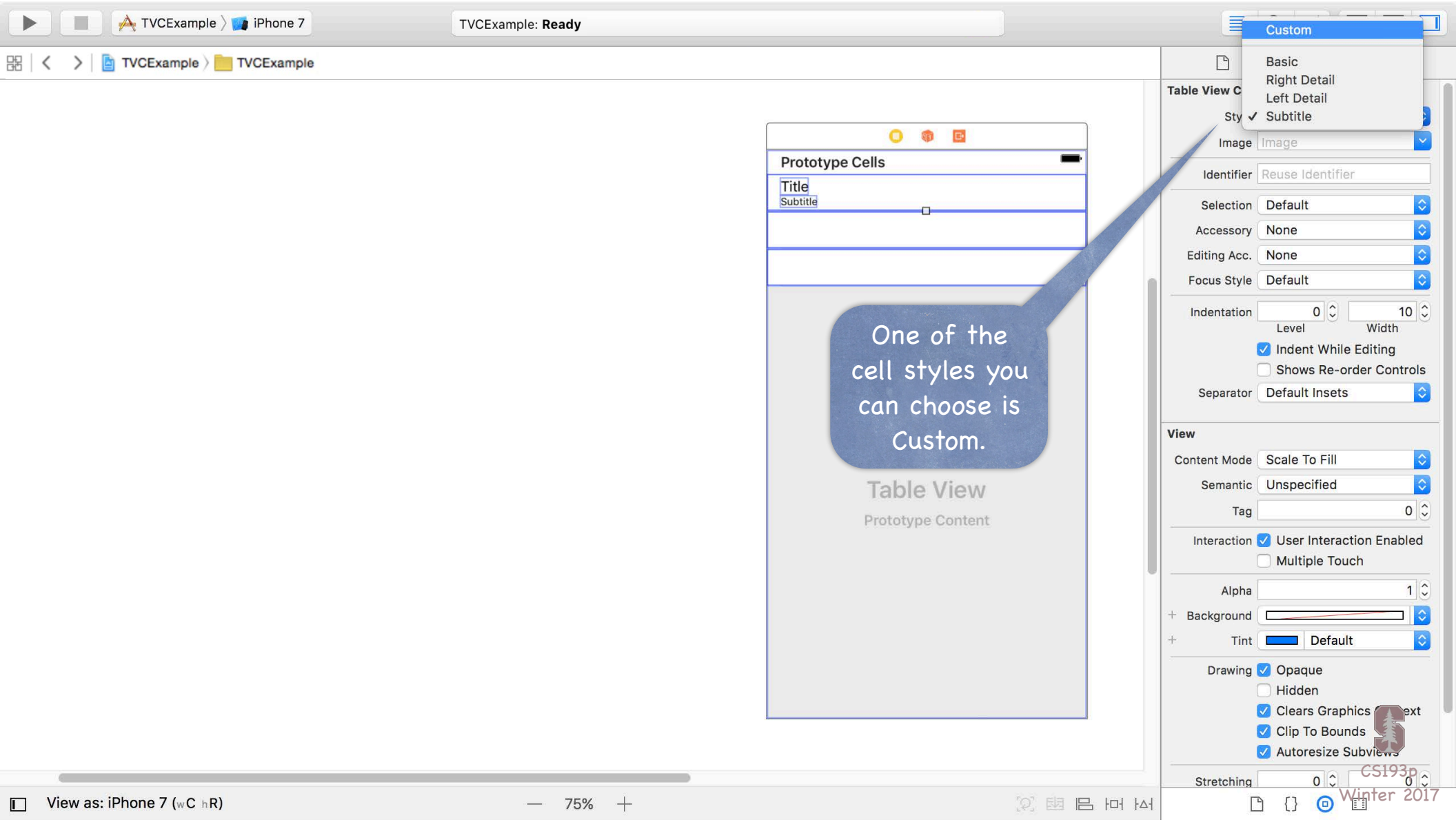

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Subtitle cell style
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Table View Cell
Style Subtitle
Image Image
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

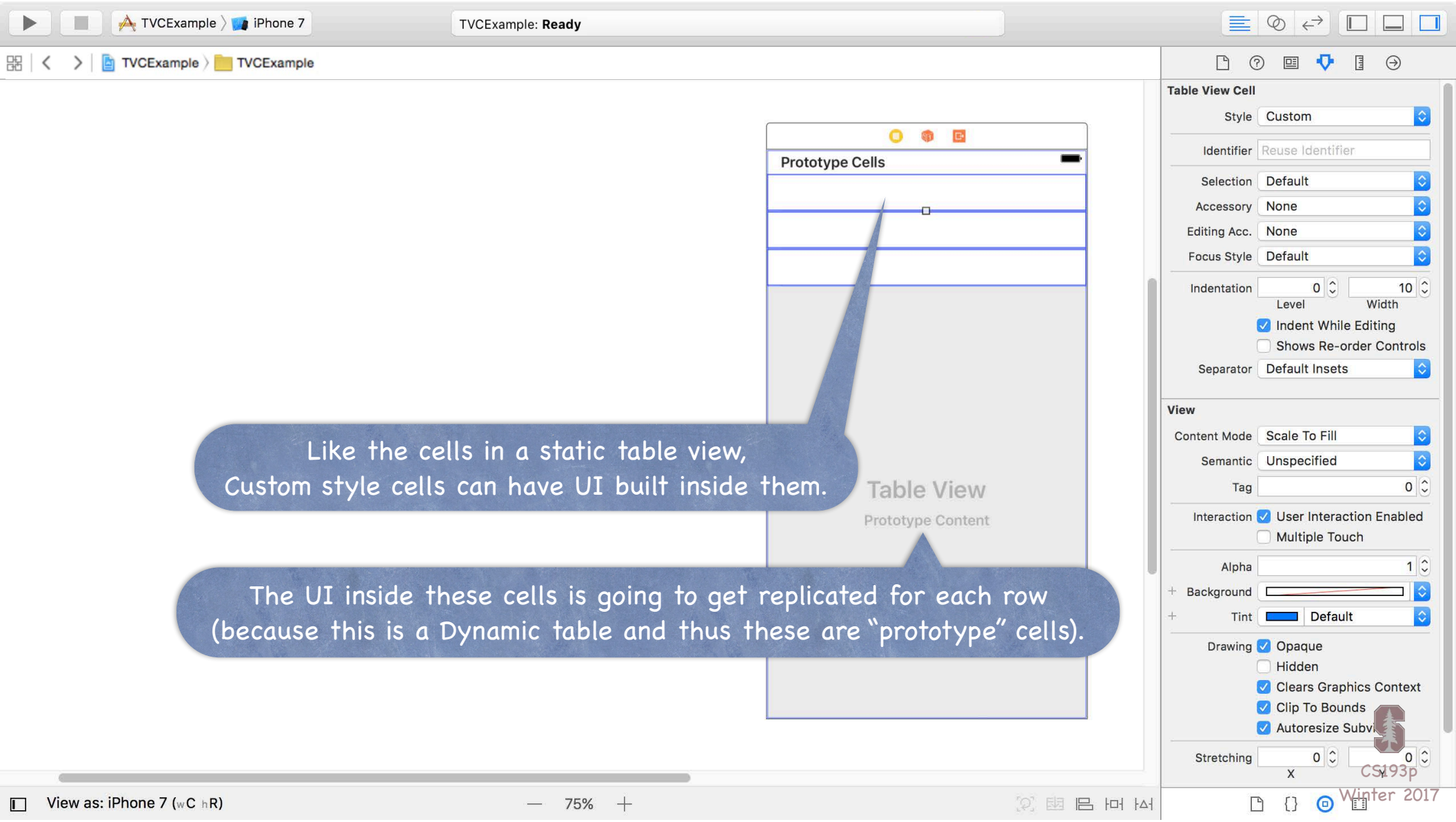

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Subtitle
You can also set a symbol to appear on the right of the cell.
Table View
Prototype Content
Table View Cell
Style Subtitle
Image Image
Identifier Reuse Identifier
Selection Default
None
Disclosure Indicator
Detail Disclosure
Checkmark
Detail
Level
Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

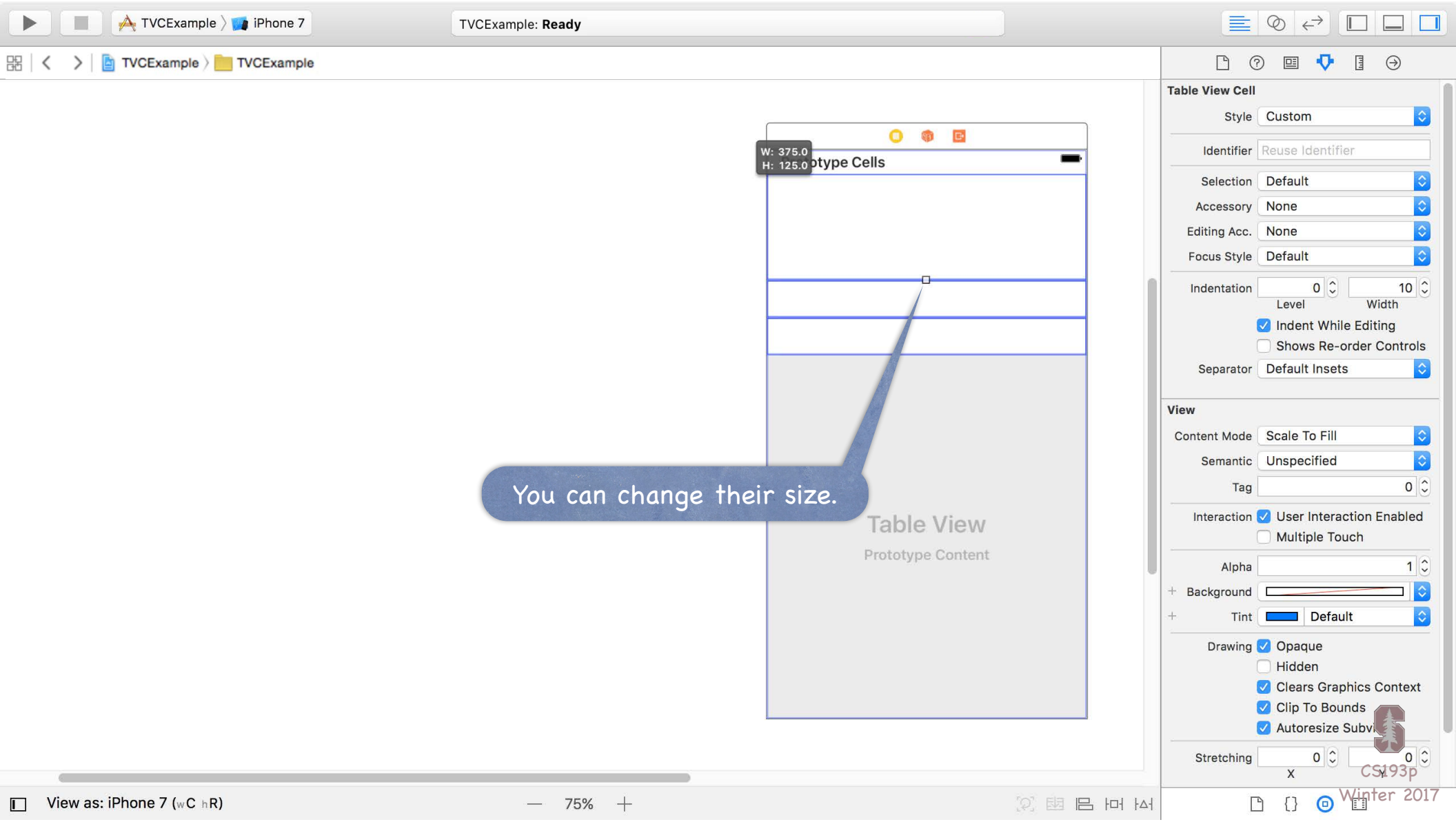

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Subtitle
Table View
Prototype Content
You can also set a symbol to appear on the right of the cell.
This one's sort of special ...
Table View Cell
Style
Subtitle
Image
Image
Identifier
Reuse Identifier
Selection
Default
None
Disclosure Indicator
Detail Disclosure
Checkmark
Detail
Indent While Editin
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Tint
Default
Drawing
Opaque
Hidden
Clip To Bounds
Autoresize Subviews
Stretching
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

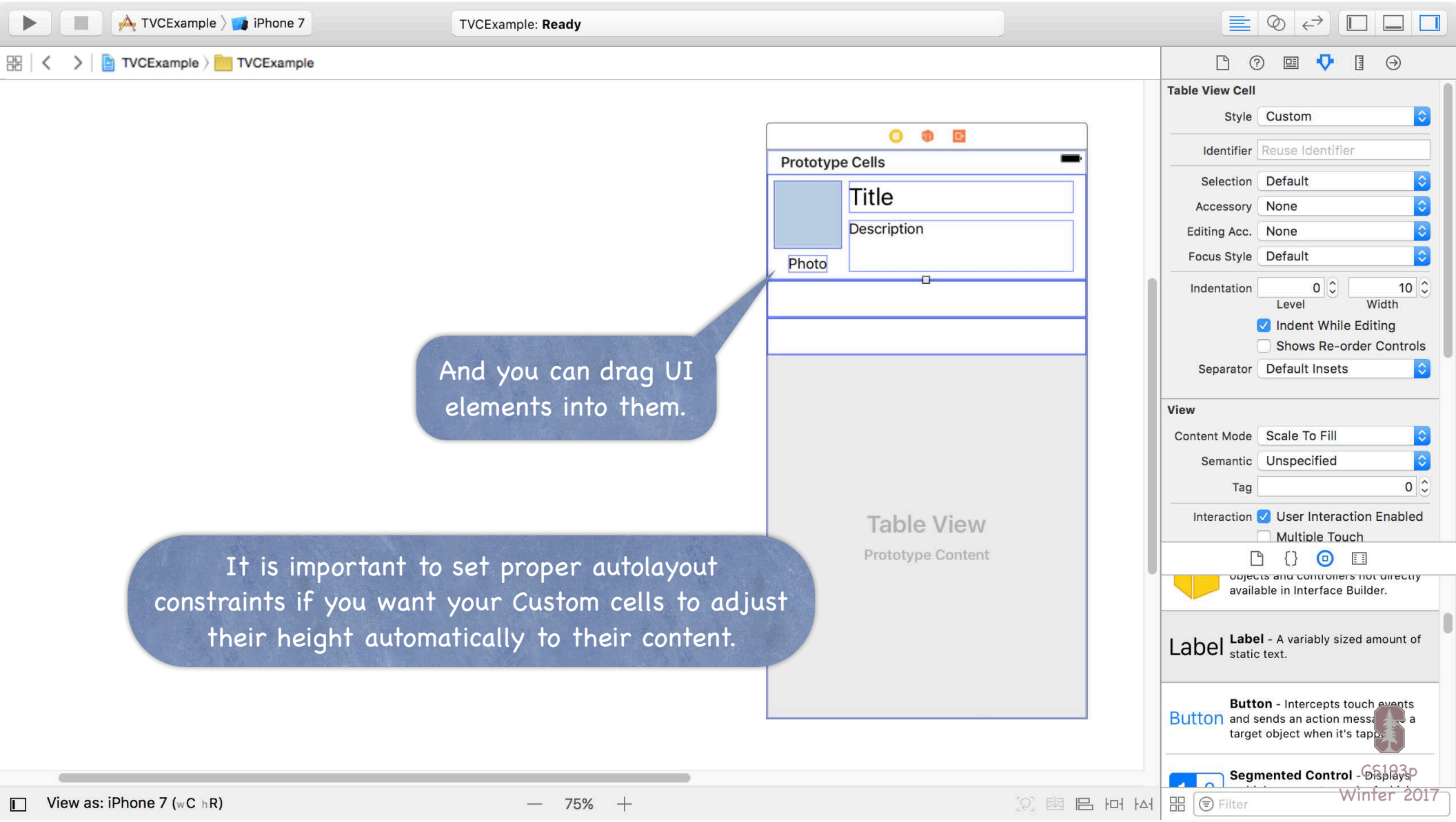

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Subtitle
We'll talk about this Detail Disclosure button in a bit.
Table View
Prototype Content
Table View Cell
Style Subtitle
Image Image
Identifier Reuse Identifier
Selection Default
Accessory Detail Disclosure
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

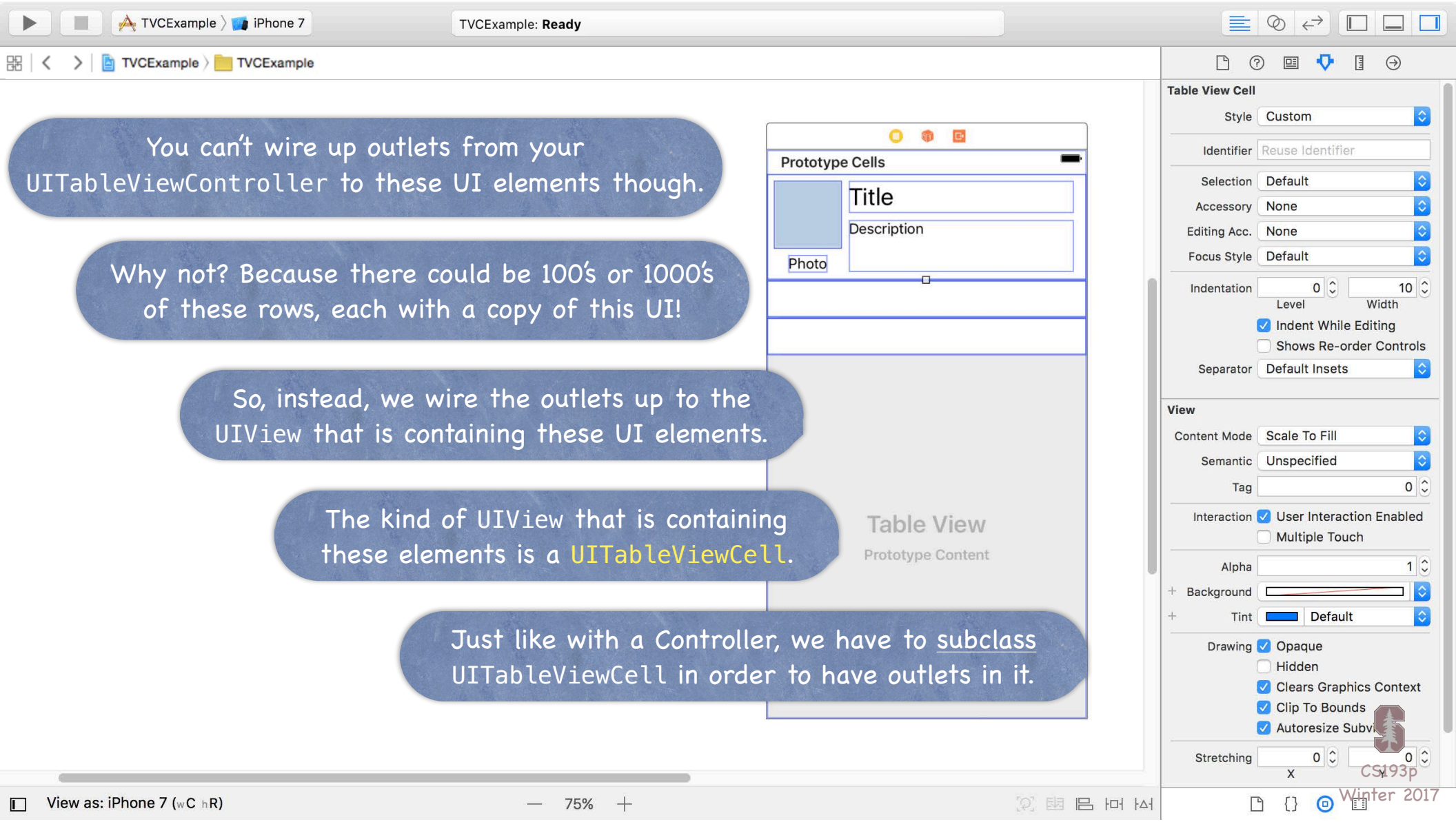

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Table View Cell
Style
Subtitle
Image
Image
Identifier
Reuse Identifier
None
Disclosure Indicator
Detail Disclosure
Checkmark
Detail
Indentation
0
Level
10
Width
Indent While Editing
Shows Re-order Controls
Separator
Default Insets
View
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching
0
0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

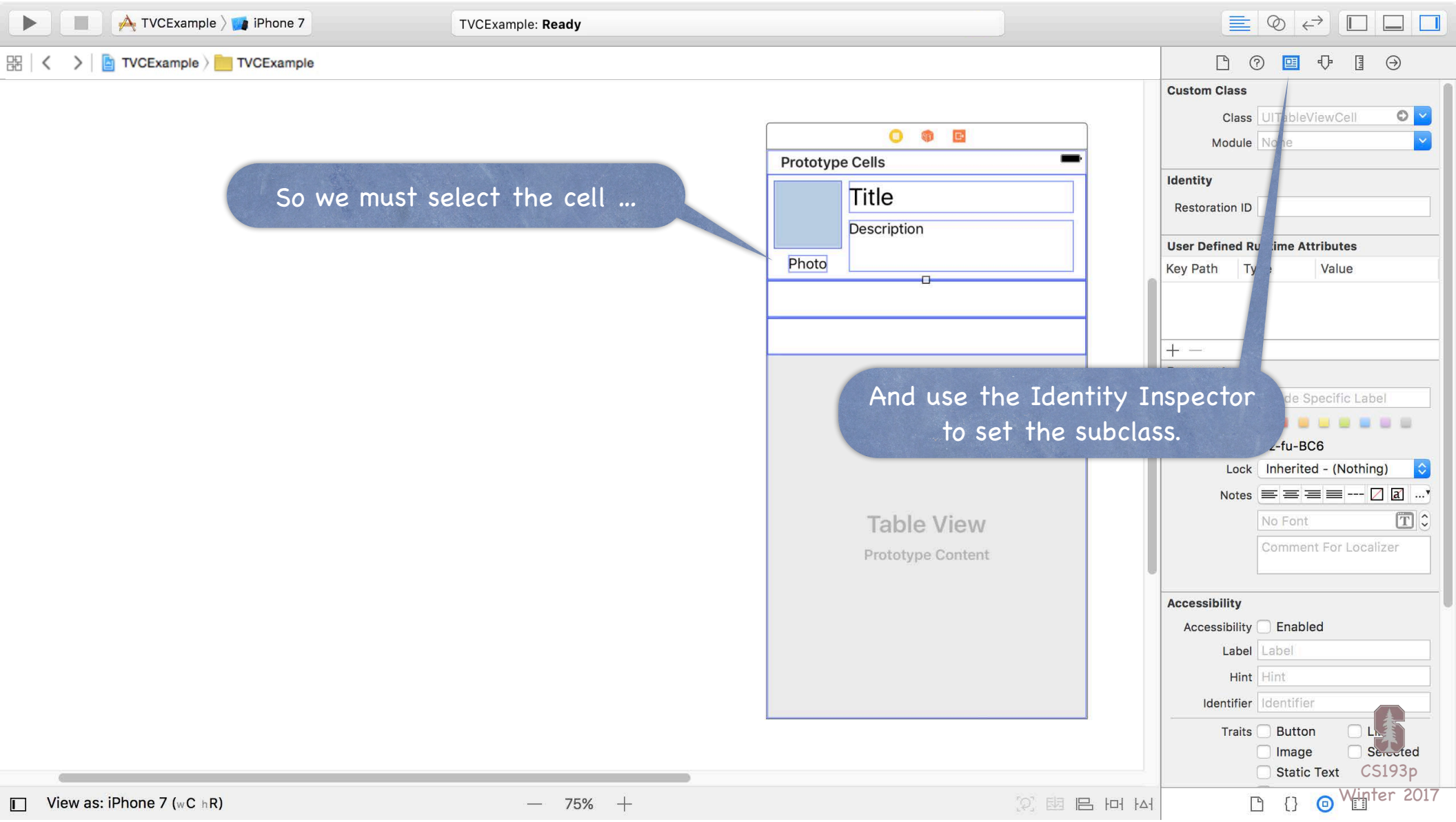

TVCExample
iPhone 7
TVCExample: Ready
TVCExample
TVCExample
Custom
Basic
Right Detail
Left Detail
Subtitle
Table View C
Image
Identifier
Reuse Identifier
Selection
Default
Accessory
None
Editing Acc.
None
Focus Style
Default
Indentation
Level
Width
Indent While Editing
Shows Re-order Controls
Separator
Default Insets
View
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching
Prototype Cells
Title
Subtitle
One of the cell styles you can choose is Custom.
Table View
Prototype Content
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

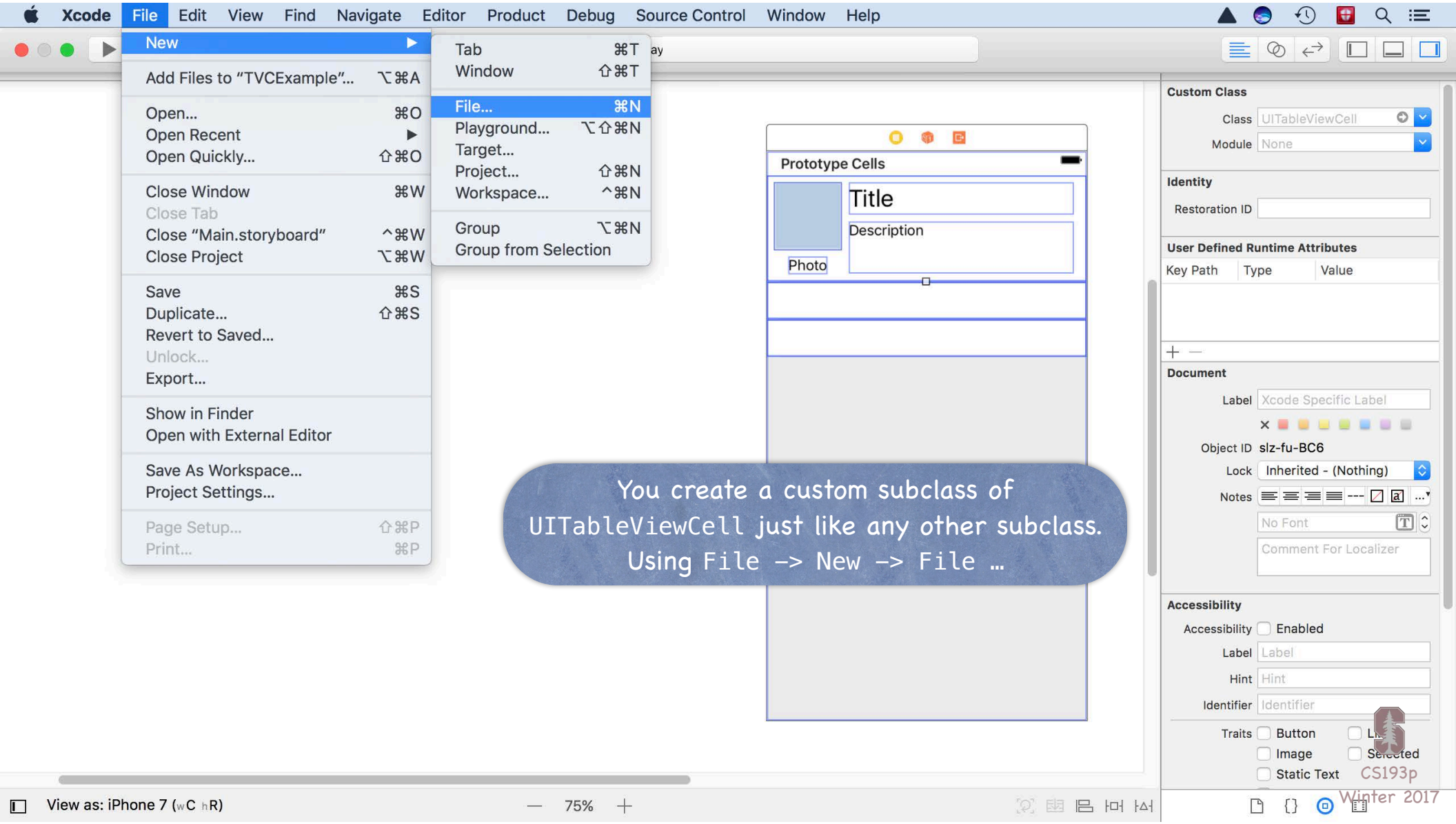

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Table View
Prototype Content
Like the cells in a static table view,
Custom style cells can have UI built inside them.
The UI inside these cells is going to get replicated for each row
(because this is a Dynamic table and thus these are "prototype" cells).
Table View Cell
Style Custom
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Autoresize Subv
Stretching 0 X 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

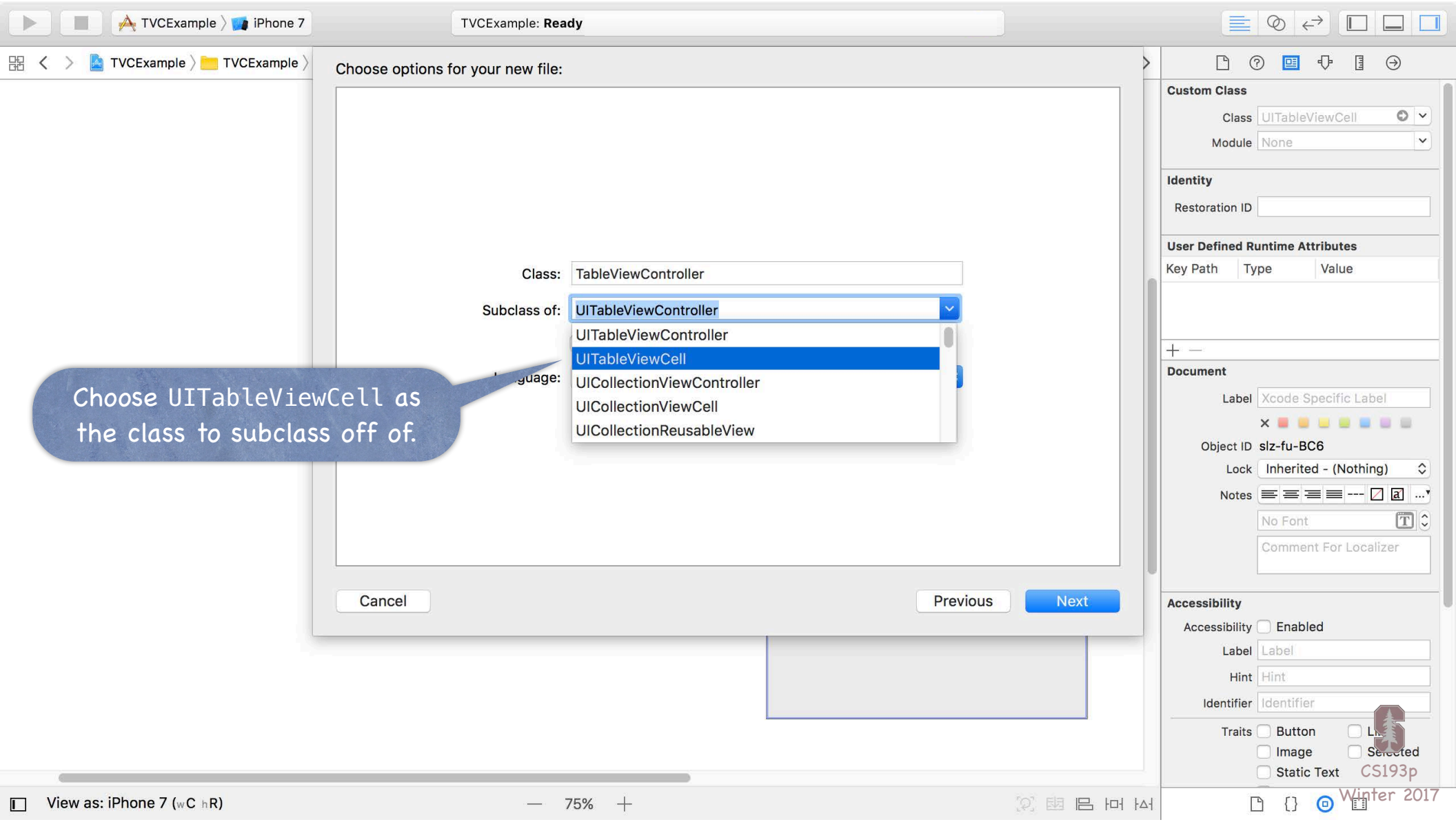

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
W: 375.0
H: 125.0
You can change their size.
Table View
Prototype Content
Table View Cell
Style Custom
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Stretching 0 X 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

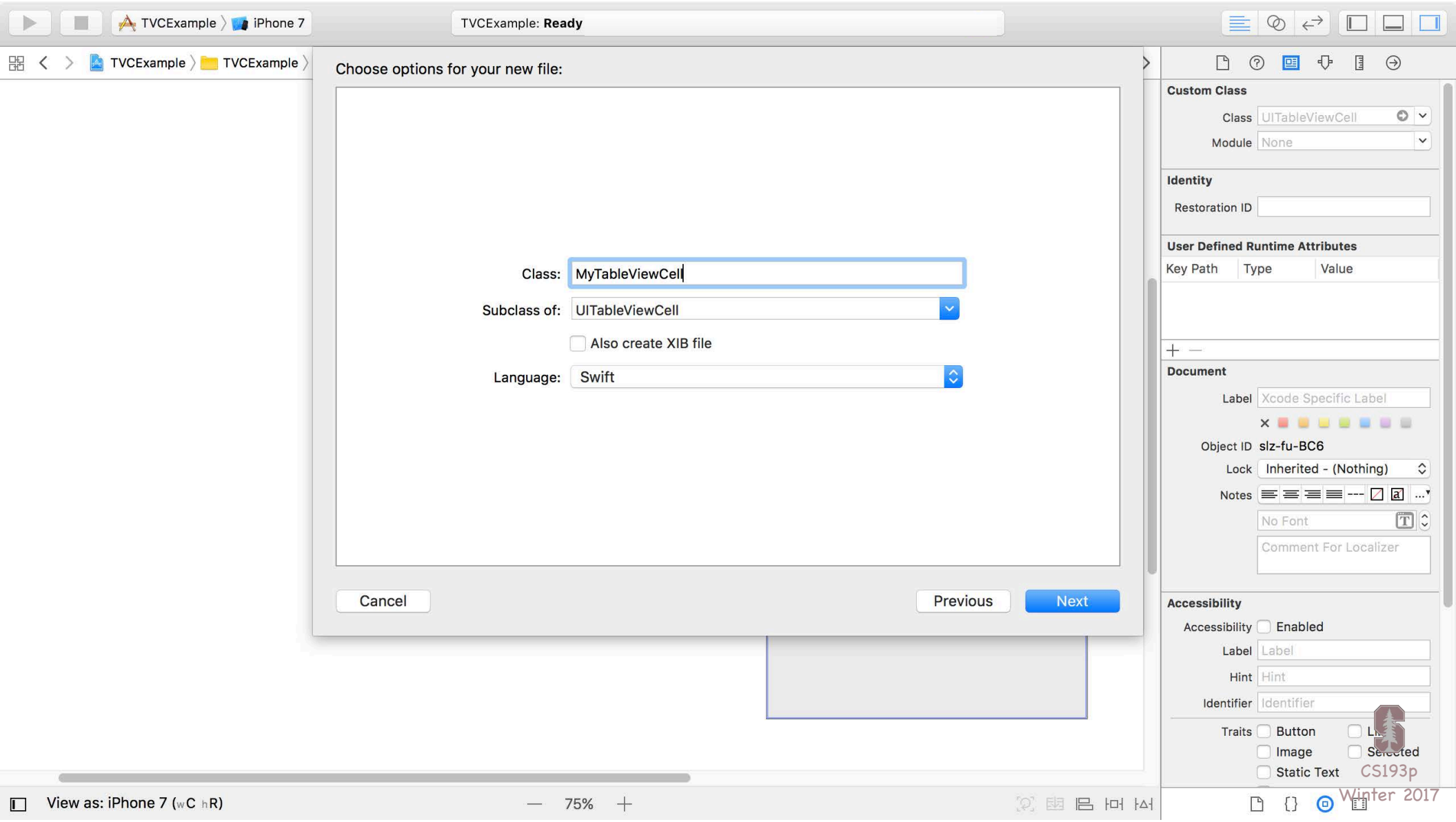

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
And you can drag UI elements into them.
It is important to set proper autolayout constraints if you want your Custom cells to adjust their height automatically to their content.
Table View Cell
Style Custom
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Label Label - A variably sized amount of static text.
Button Button - Intercepts touch events and sends an action message a target object when it's tapp
Segmented Control - Displays
View as: iPhone 7 (wC hR)
75%
Filter
CS193p
Winter 2017

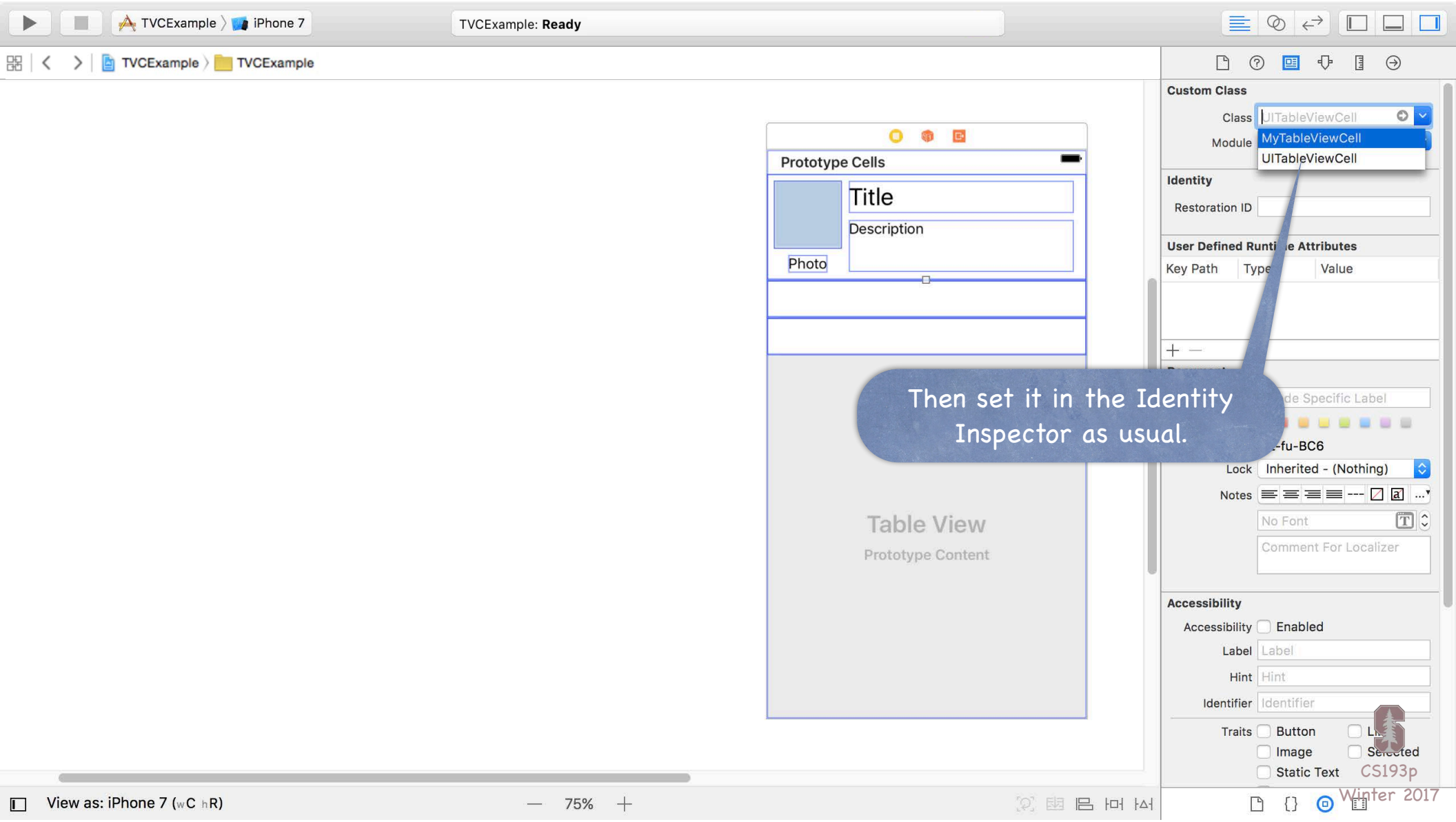

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
You can't wire up outlets from your UITableViewController to these UI elements though.
Why not? Because there could be 100's or 1000's of these rows, each with a copy of this UI!
So, instead, we wire the outlets up to the UIView that is containing these UI elements.
The kind of UIView that is containing these elements is a UITableViewCell.
Just like with a Controller, we have to subclass UITableViewCell in order to have outlets in it.
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
Table View Cell
Style Custom
Identifier Reuse Identifier
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Stretching 0 X 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

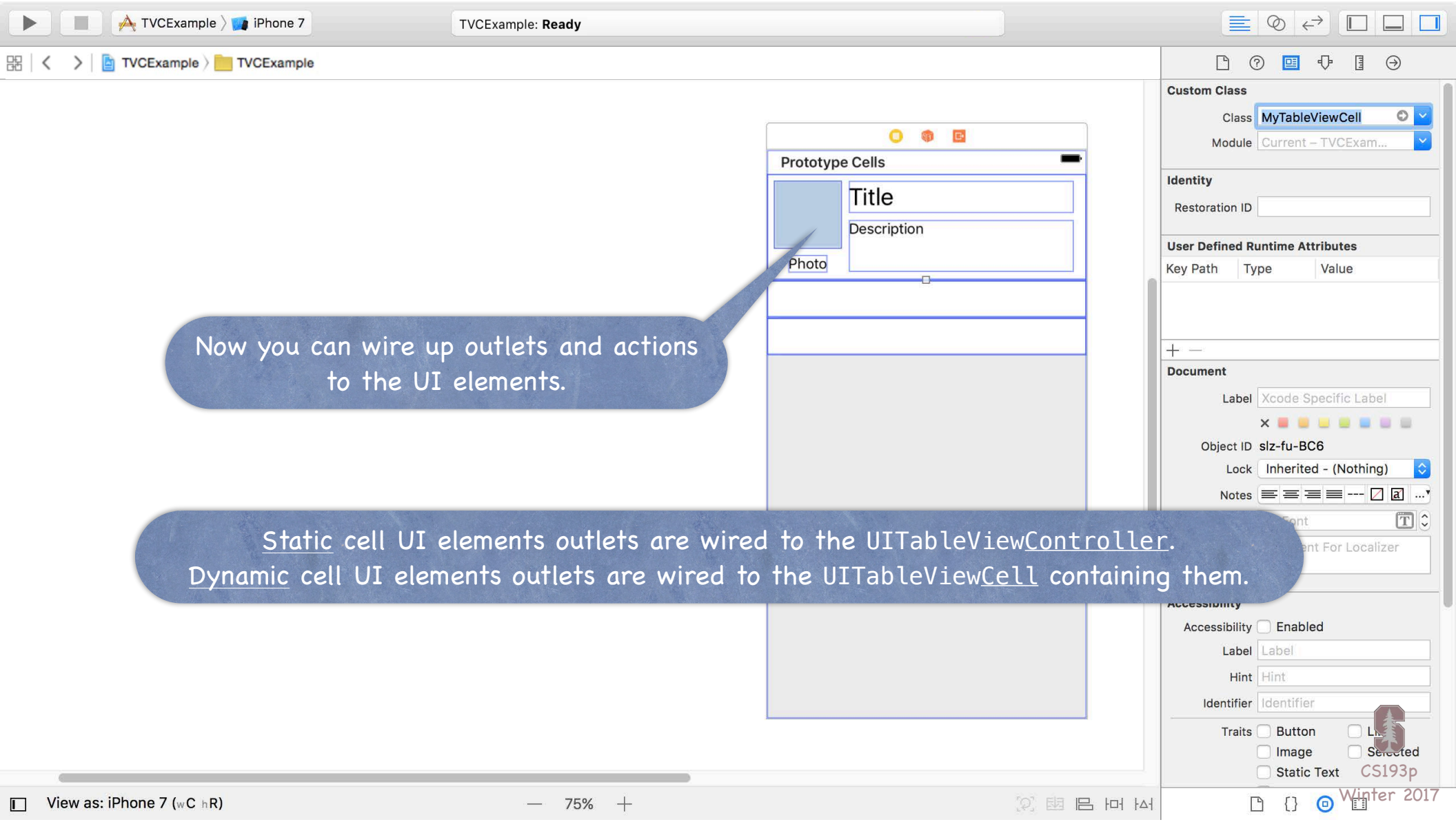

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
So we must select the cell ...
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
And use the Identity Inspector to set the subclass.
Custom Class
Class
Module
Identity
Restoration ID
User Defined Runtime Attributes
Key Path
Value
Lock
Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility
Enabled
Label
Hint
Identifier
Traits
Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

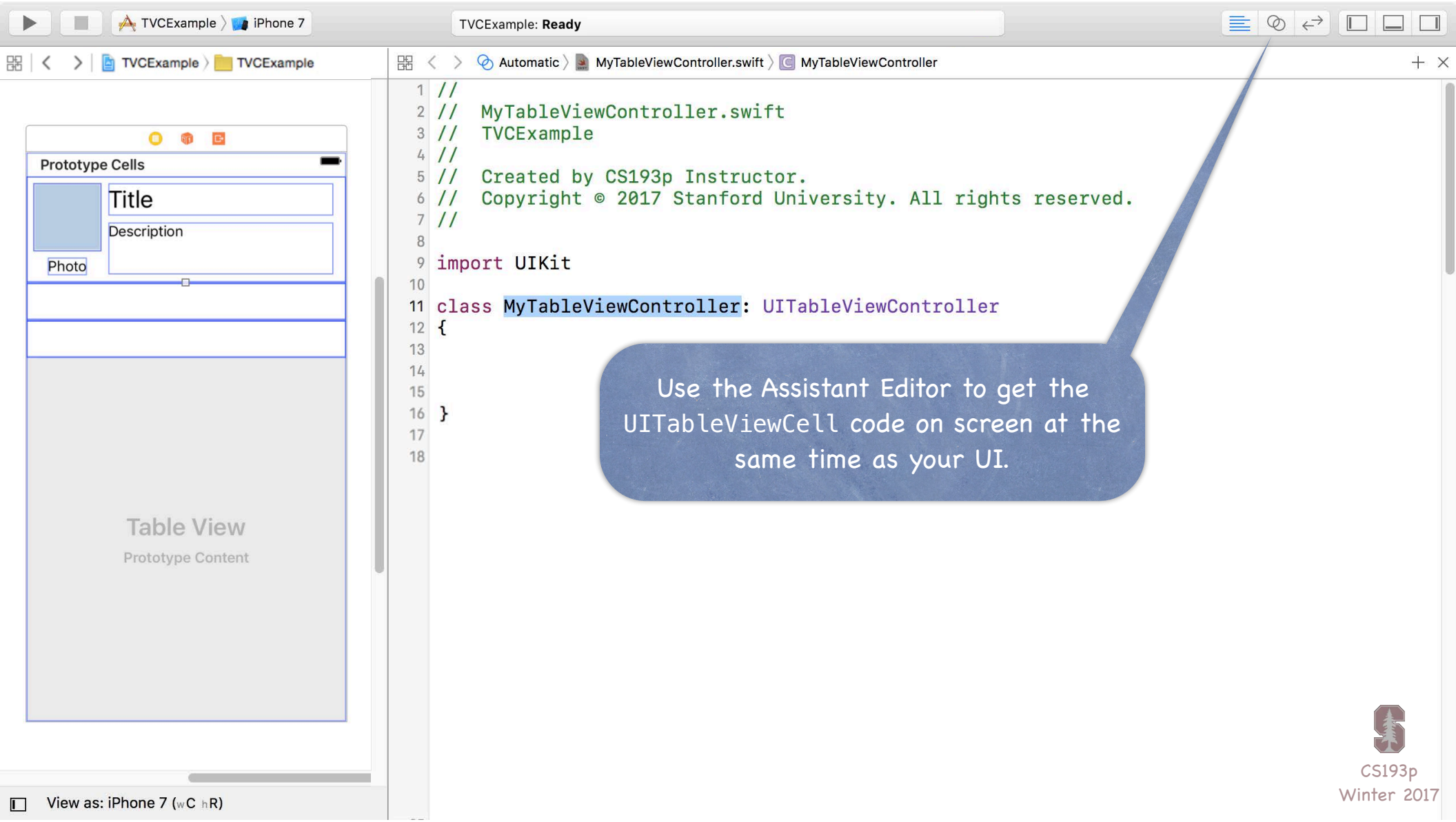

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
New
Add Files to "TVCExample"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close "Main.storyboard" ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Playground... ⌥⇧⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
Prototype Cells
Title
Description
Photo
You create a custom subclass of UITableViewCell just like any other subclass. Using File -> New -> File ...
Custom Class
Class UITableViewCell
Module None
Identity
Restoration ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID slz-fu-BC6
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button Image Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

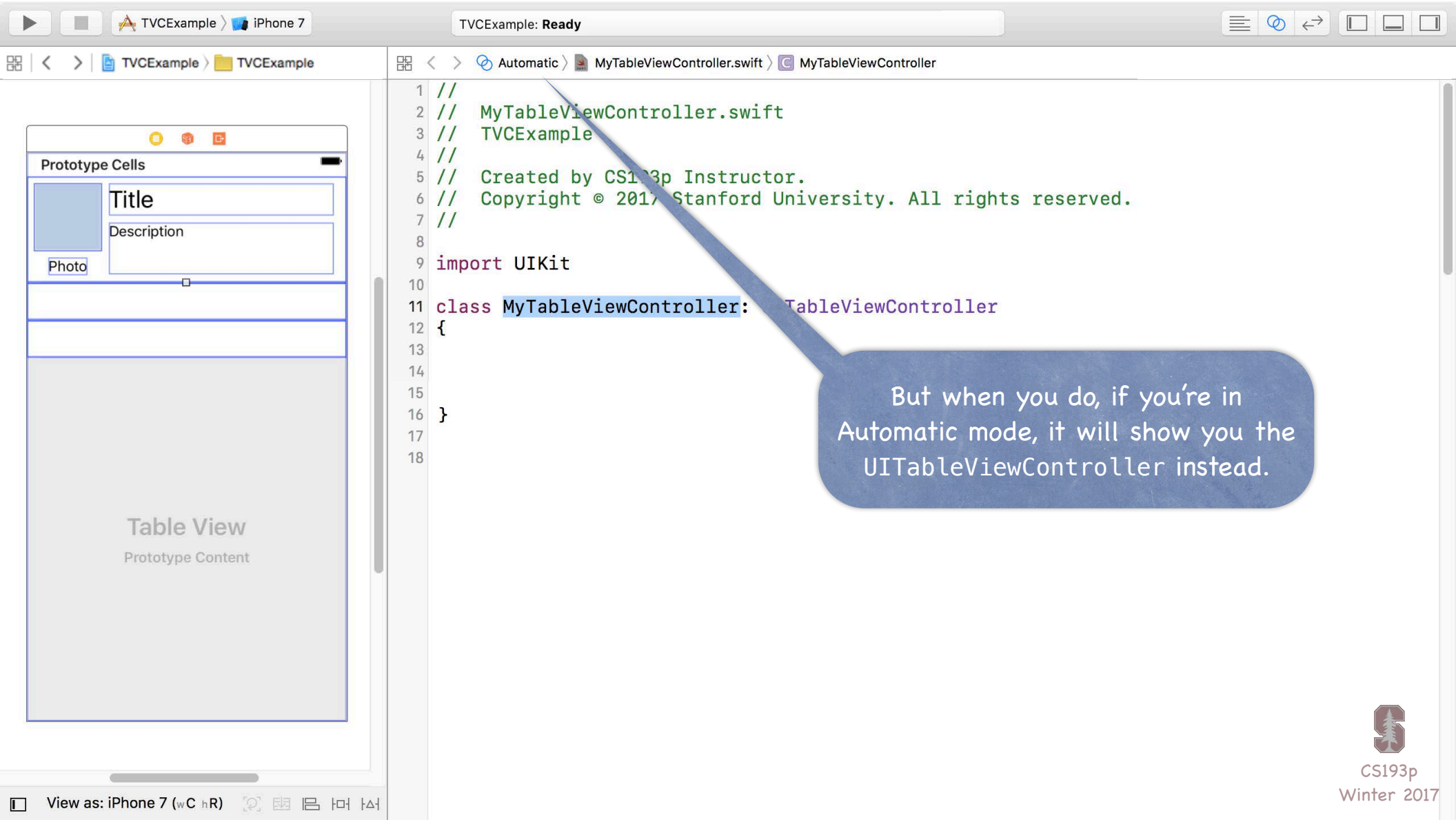

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Choose options for your new file:
Class: TableViewController
Subclass of: UITableViewController
UITableViewController
UITableViewCell
UICollectionViewController
UICollectionViewCell
UICollectionReusableView
Choose UITableViewCell as the class to subclass off of.
Cancel
Previous
Next
Custom Class
Class UITableViewCell
Module None
Identity
Restoration ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID slz-fu-BC6
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

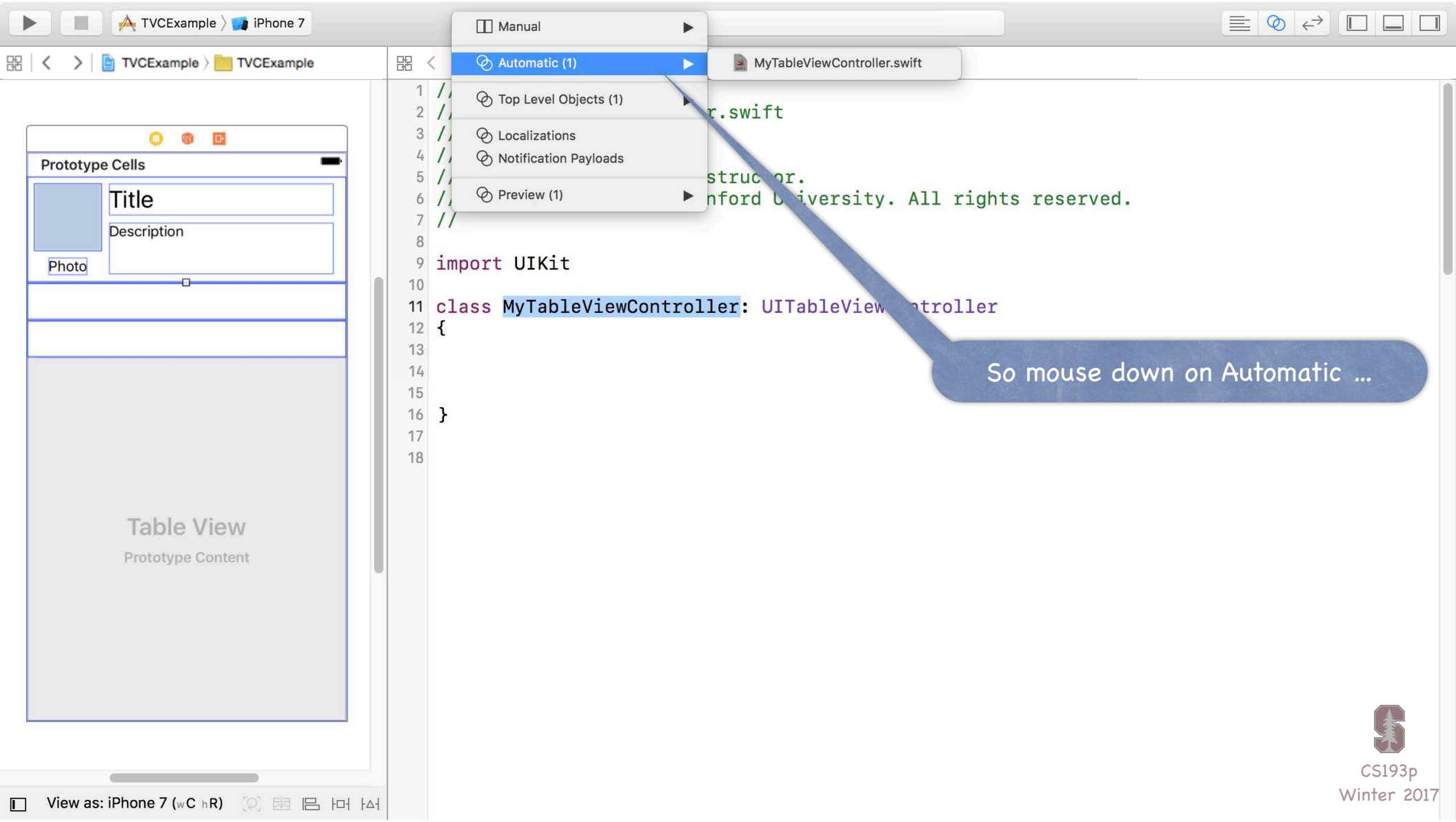

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Choose options for your new file:
Class: MyTableViewCell
Subclass of: UITableViewCell
Also create XIB file
Language: Swift
Cancel
Previous
Next
Custom Class
Class UITableViewCell
Module None
Identity
Restoration ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID slz-fu-BC6
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

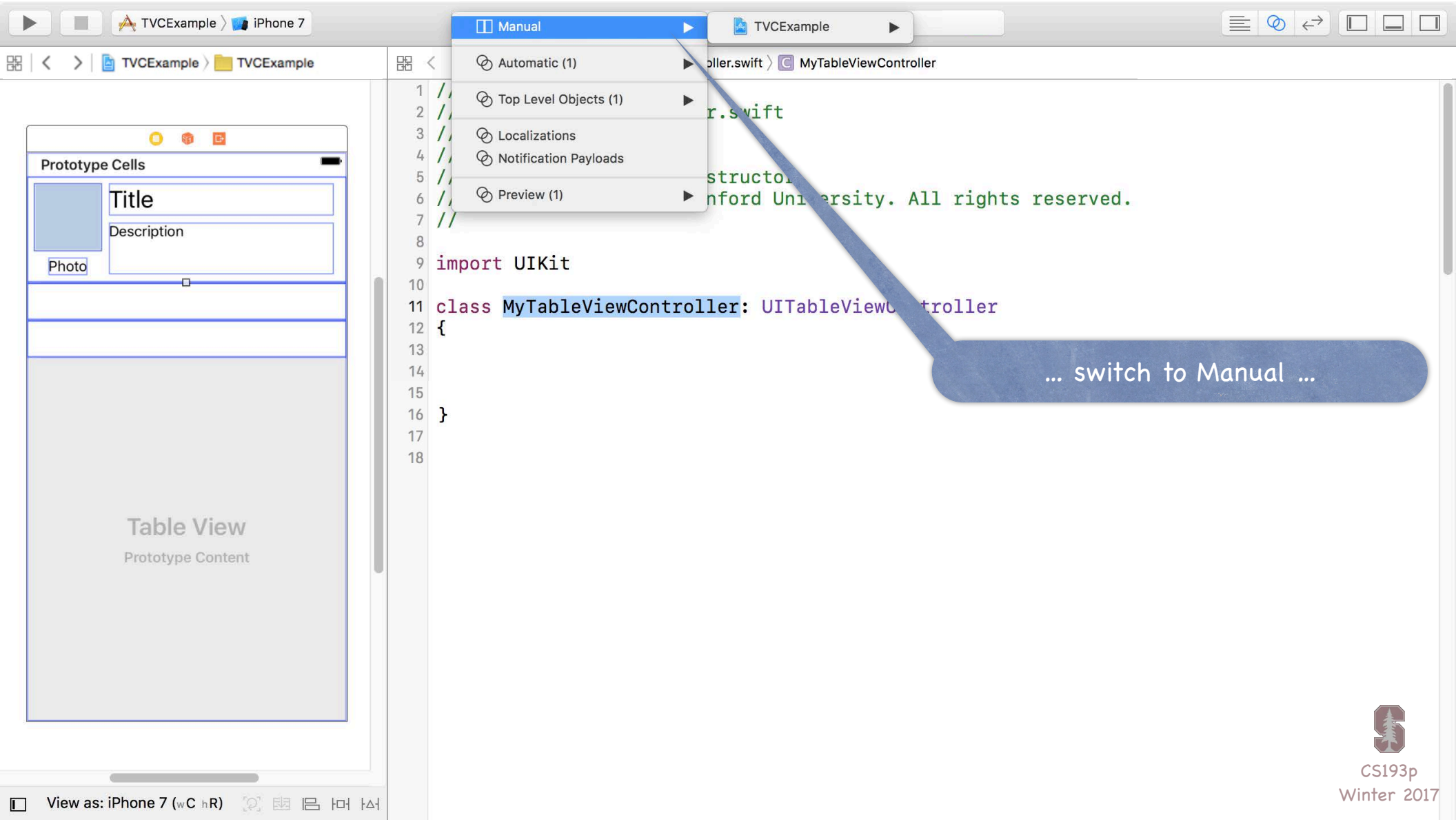

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
Then set it in the Identity Inspector as usual.
Custom Class
Class
UITableViewCell
Module
MyTableViewCell
UITableViewCell
Identity
Restoration ID
User Defined Runtime Attributes
Key Path
Type
Value
Lock
Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility
Enabled
Label
Hint
Identifier
Traits
Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

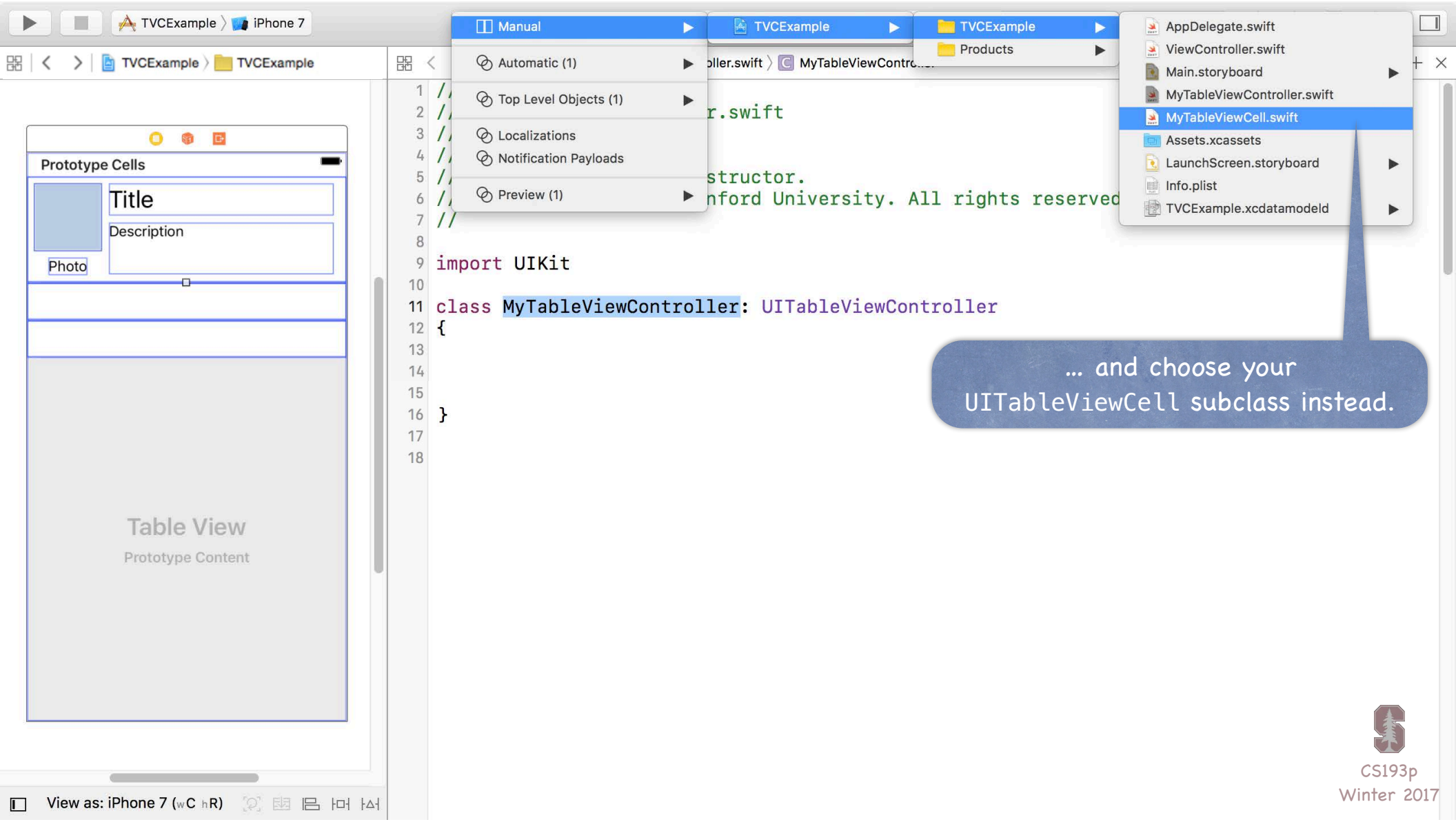

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Prototype Cells
Title
Description
Photo
Now you can wire up outlets and actions to the UI elements.
Static cell UI elements outlets are wired to the UITableViewController.
Dynamic cell UI elements outlets are wired to the UITableViewCell containing them.
Custom Class
Class MyTableViewCell
Module Current – TVCExam...
Identity
Restoration ID
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID slz-fu-BC6
Lock Inherited - (Nothing)
Notes
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Automatic 〉 MyTableViewController.swift 〉 MyTableViewController
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
View as: iPhone 7 (wC hR)
1 //
2 // MyTableViewController.swift
3 // TVCExample
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15
16 }
17
18
Use the Assistant Editor to get the UITableViewCell code on screen at the same time as your UI.

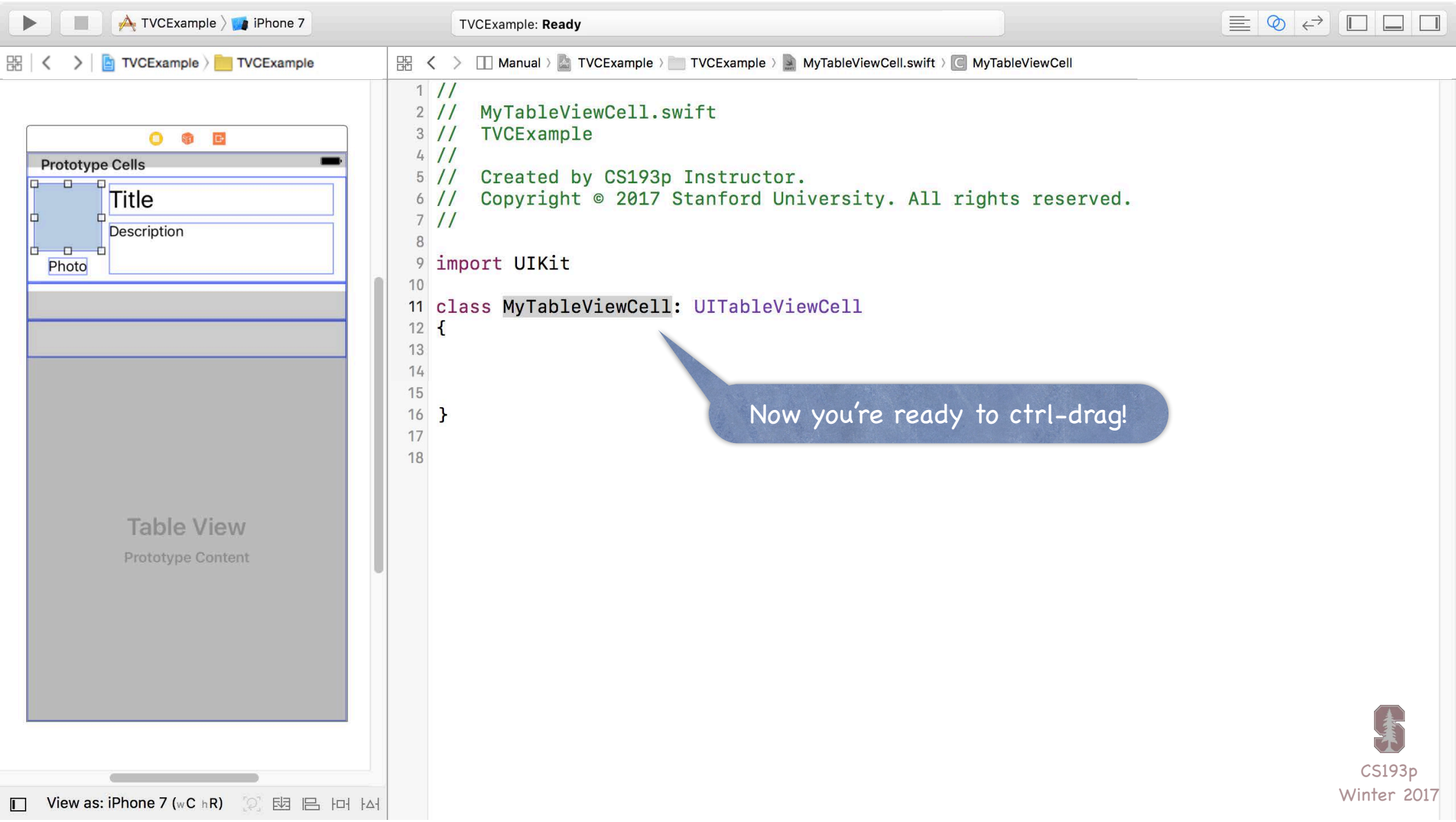

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Automatic 〉 MyTableViewController.swift 〉 MyTableViewController
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
View as: iPhone 7 (wC hR)
1 //
2 // MyTableViewController.swift
3 // TVCExample
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewController: TableViewController
12 {
13
14
15
16 }
17
18
But when you do, if you're in Automatic mode, it will show you the UITableViewController instead.

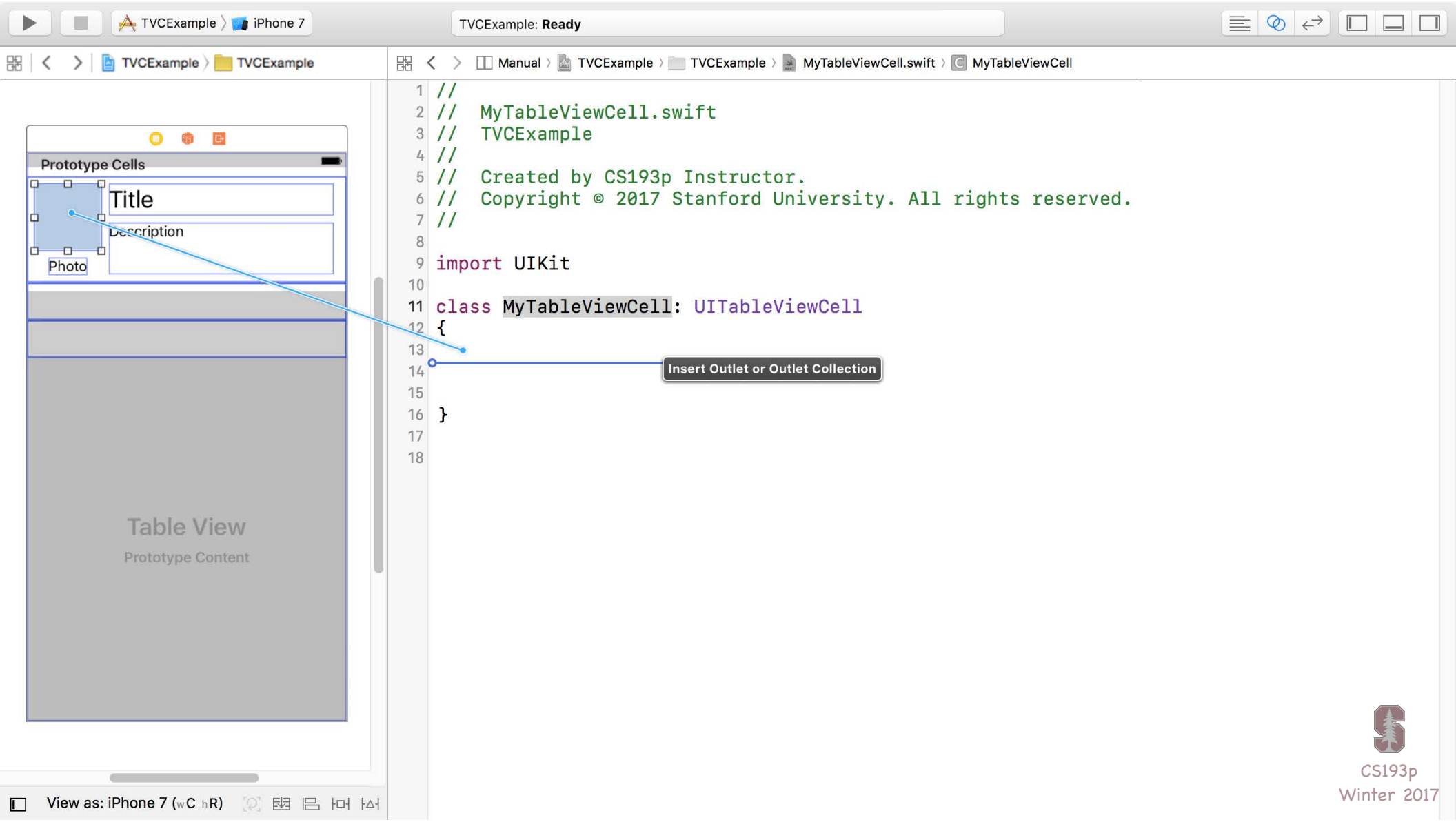

TVCExample
iPhone 7
TVCExample
TVCExample
Manual
Automatic (1)
Top Level Objects (1)
Localizations
Notification Payloads
Preview (1)
MyTableViewController.swift
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
View as: iPhone 7 (wC hR)
1 //
2 //
3 //
4 //
5 //
6 //
7 //
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15
16 }
17
18
So mouse down on Automatic ...


TVCExample
iPhone 7
Manual
Automatic (1)
Top Level Objects (1)
Localizations
Notification Payloads
Preview (1)
TVCExample
TVCExample
TVCExample
MyTableViewController
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
View as: iPhone 7 (wC hR)
1 //
2 //
3 //
4 //
5 //
6 //
7 //
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15
16 }
17
18
... switch to Manual ...

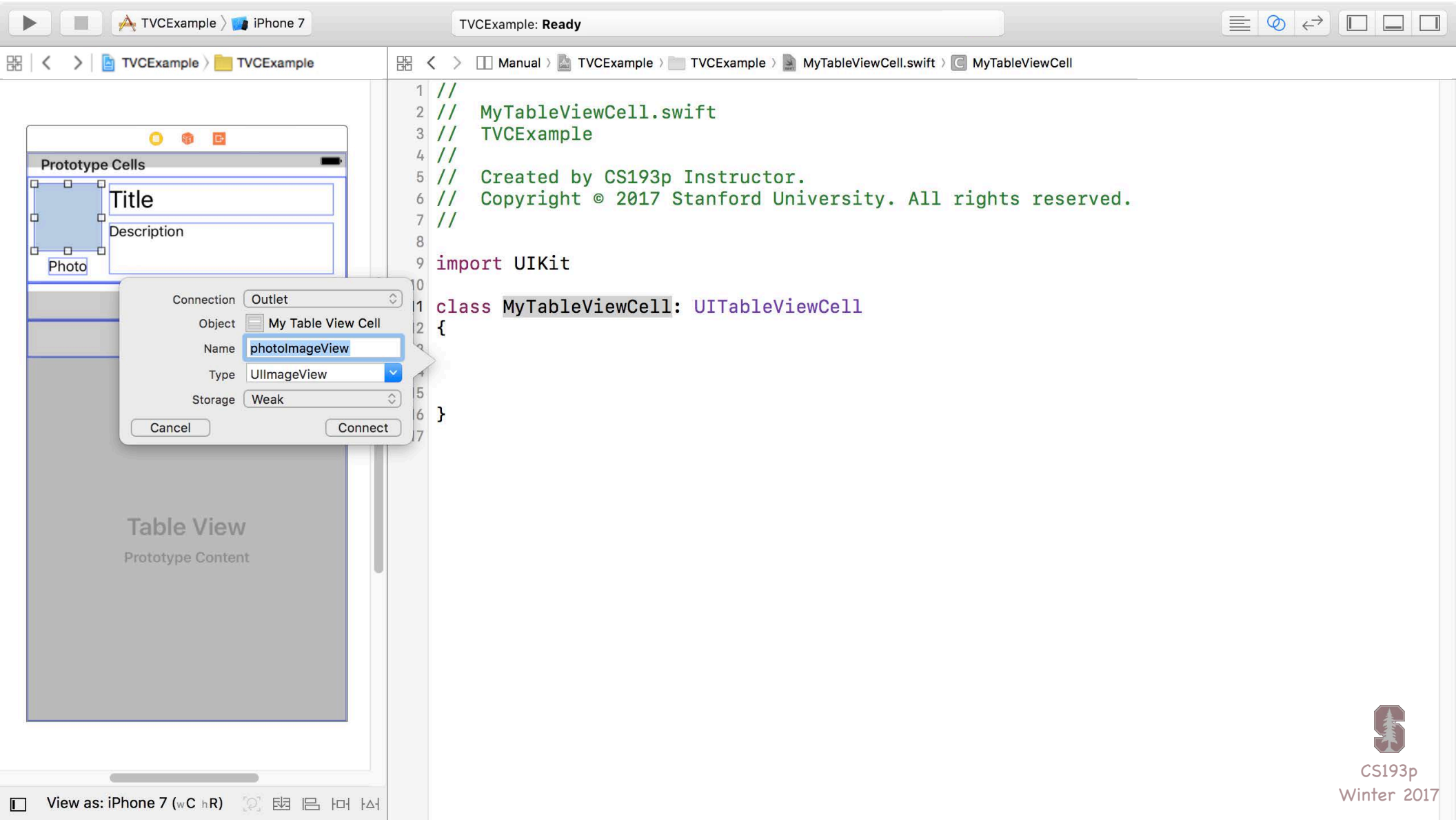


CS193p
Winter 2017

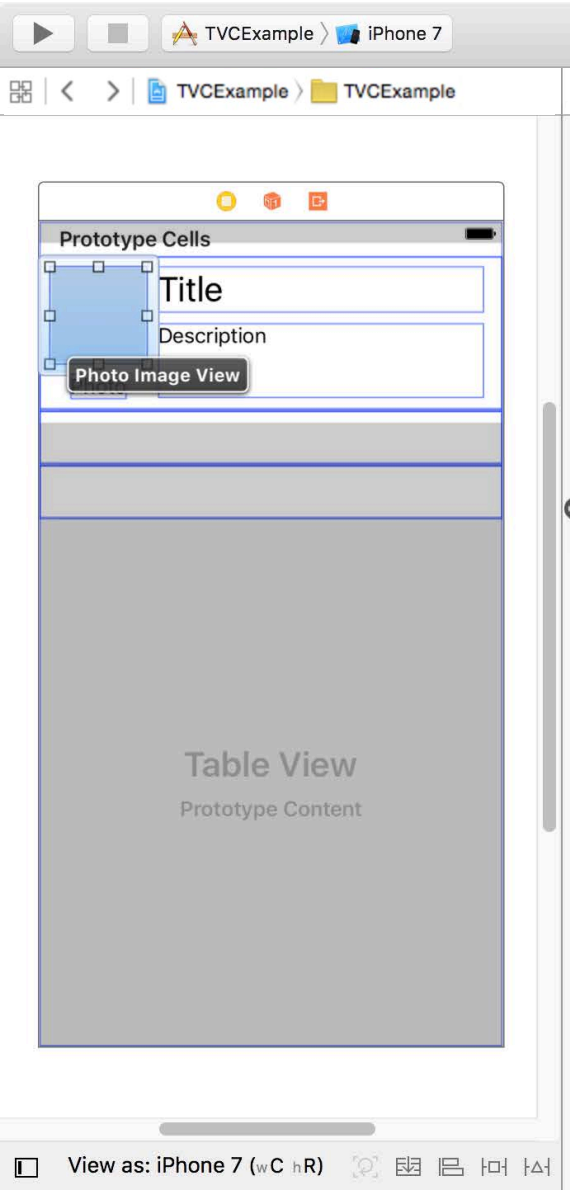

TVCExample
iPhone 7
TVCExample: Ready
TVCExample
TVCExample
Manual
TVCExample
TVCExample
MyTableViewCell.swift
MyTableViewCell
Prototype Cells
Title
Description
Photo
Table View
Prototype Content
View as: iPhone 7 (wC hR)
//
// MyTableViewCell.swift
// TVCExample
//
// Created by CS193p Instructor.
// Copyright © 2017 Stanford University. All rights reserved.
//
import UIKit
class MyTableViewCell: UITableViewCell
{
}
Now you're ready to ctrl-drag!

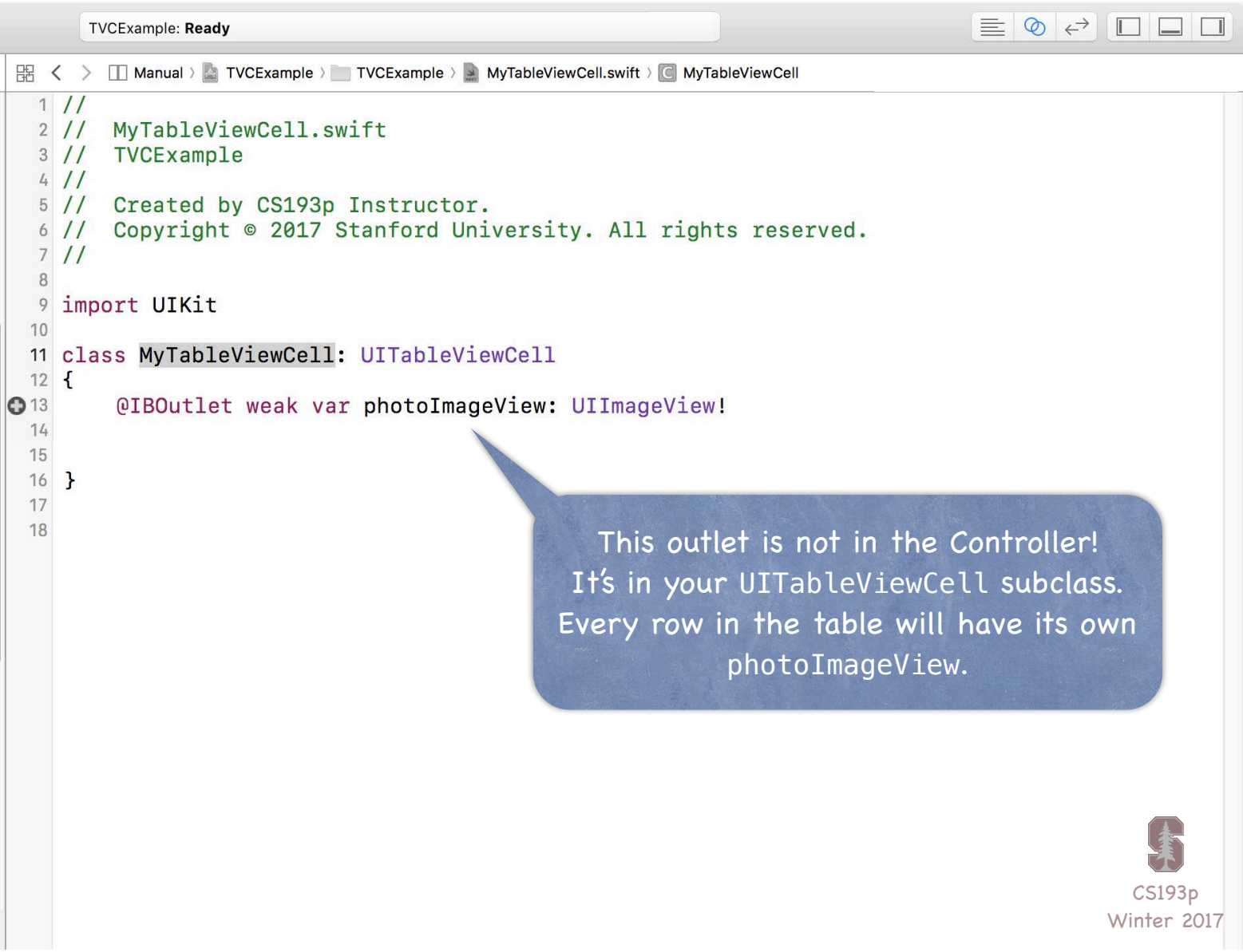


CS193p
Winter 2017

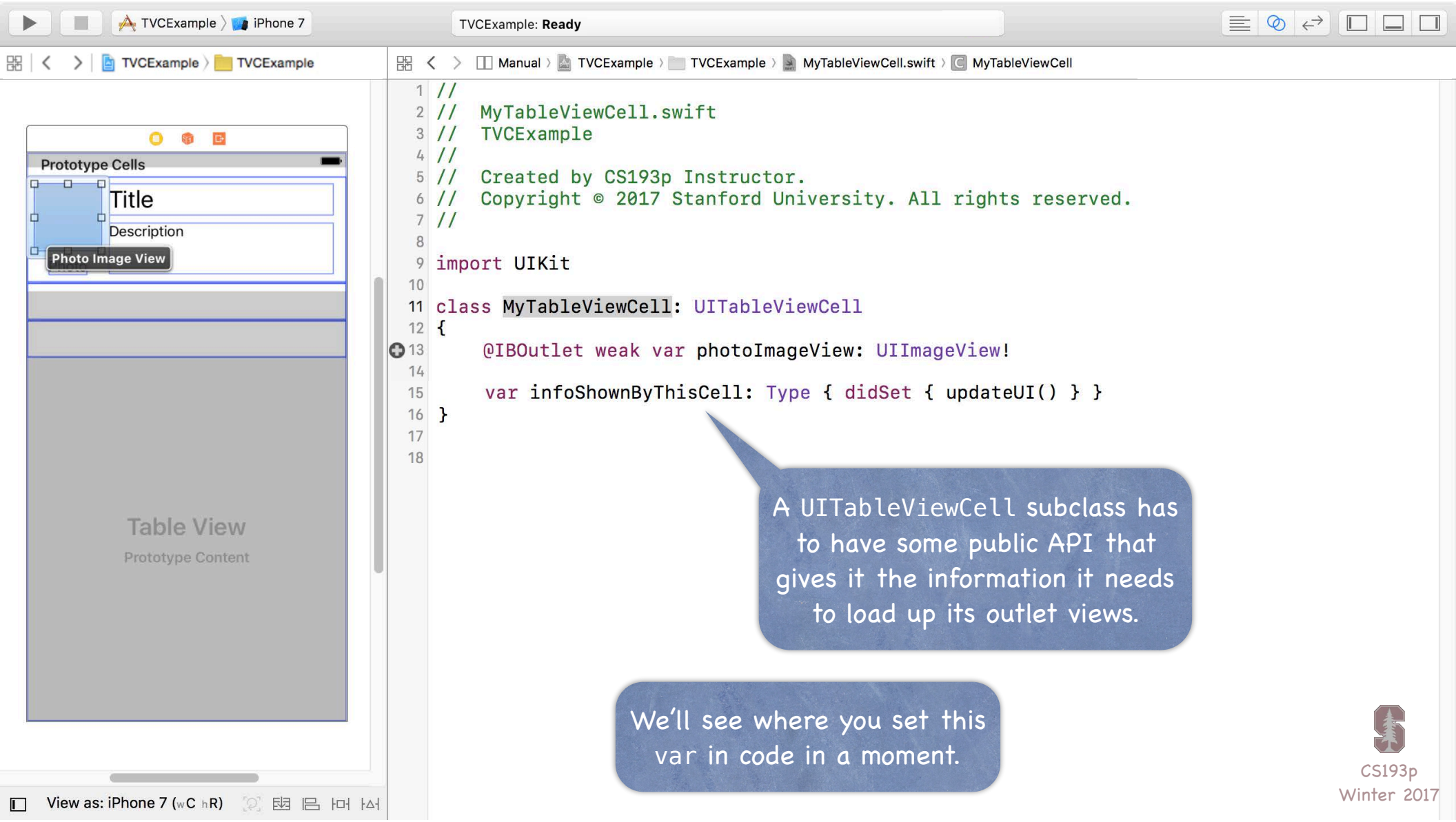


CS193p
Winter 2017





This outlet is not in the Controller!
It's in your UITableViewCell subclass.
Every row in the table will have its own photoImageView.


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Manual 〉 TVCExample 〉 TVCExample 〉 MyTableViewCell.swift 〉 MyTableViewCell
Prototype Cells
Title
Description
Photo Image View
Table View
Prototype Content
View as: iPhone 7 (wC hR)
1 //
2 // MyTableViewCell.swift
3 // TVCExample
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewCell: UITableViewCell
12 {
13 @IBOutlet weak var photoImageView: UIImageView!
14
15 var infoShownByThisCell: Type { didSet { updateUI() } }
16 }
17
18
A UITableViewCell subclass has to have some public API that gives it the information it needs to load up its outlet views.
We'll see where you set this var in code in a moment.

UITableView Protocols

- How to connect all this stuff up in code?

  Connections to code are made using the UITableView's dataSource and delegate

  The delegate is used to control how the table is displayed (it's look and feel)

  The dataSource provides the data that is displayed inside the cells

  UITableViewController automatically sets itself as the UITableView's delegate & dataSource

  Your UITableViewController subclass will also have a property pointing to the UITableView …

  ```
  var tableView: UITableView // self.view in UITableViewController
  ```

- When do we need to implement the dataSource?

  Whenever the data in the table is dynamic (i.e. not static cells)

  There are three important methods in this protocol …

  How many sections in the table?

  How many rows in each section?

  Give me a view to use to draw each cell at a given row in a given section.

  Let's cover the last one first (since the first two are very straightforward) …

Customizing Each Row

- Providing a UIView to draw each row …
  - It has to be a UITableViewCell (which is a subclass of UIView) or subclass thereof
  - Don't worry, if you have 10,000 rows, only the visible ones will have a UITableViewCell
  - But this means that UITableViewCells are reused as rows appear and disappear
  - This has ramifications for multithreaded situations, so be careful in that scenario

  The UITableView will ask its UITableViewDataSource for the UITableViewCell for a row …

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



}
```

IndexPath is just a container to pass you the section and row in question.

Customizing Each Row

- Providing a UIView to draw each row …
  - It has to be a UITableViewCell (which is a subclass of UIView) or subclass thereof
  - Don't worry, if you have 10,000 rows, only the visible ones will have a UITableViewCell
  - But this means that UITableViewCells are reused as rows appear and disappear
  - This has ramifications for multithreaded situations, so be careful in that scenario

  The UITableView will ask its UITableViewDataSource for the UITableViewCell for a row …

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    // myInternalDataStructure is conceptual here: it doesn't have to be an Array of Arrays


}
```

Customizing Each Row

- Providing a UIView to draw each row …

  It has to be a UITableViewCell (which is a subclass of UIView) or subclass thereof

  Don't worry, if you have 10,000 rows, only the visible ones will have a UITableViewCell

  But this means that UITableViewCells are reused as rows appear and disappear

  This has ramifications for multithreaded situations, so be careful in that scenario

  The UITableView will ask its UITableViewDataSource for the UITableViewCell for a row …

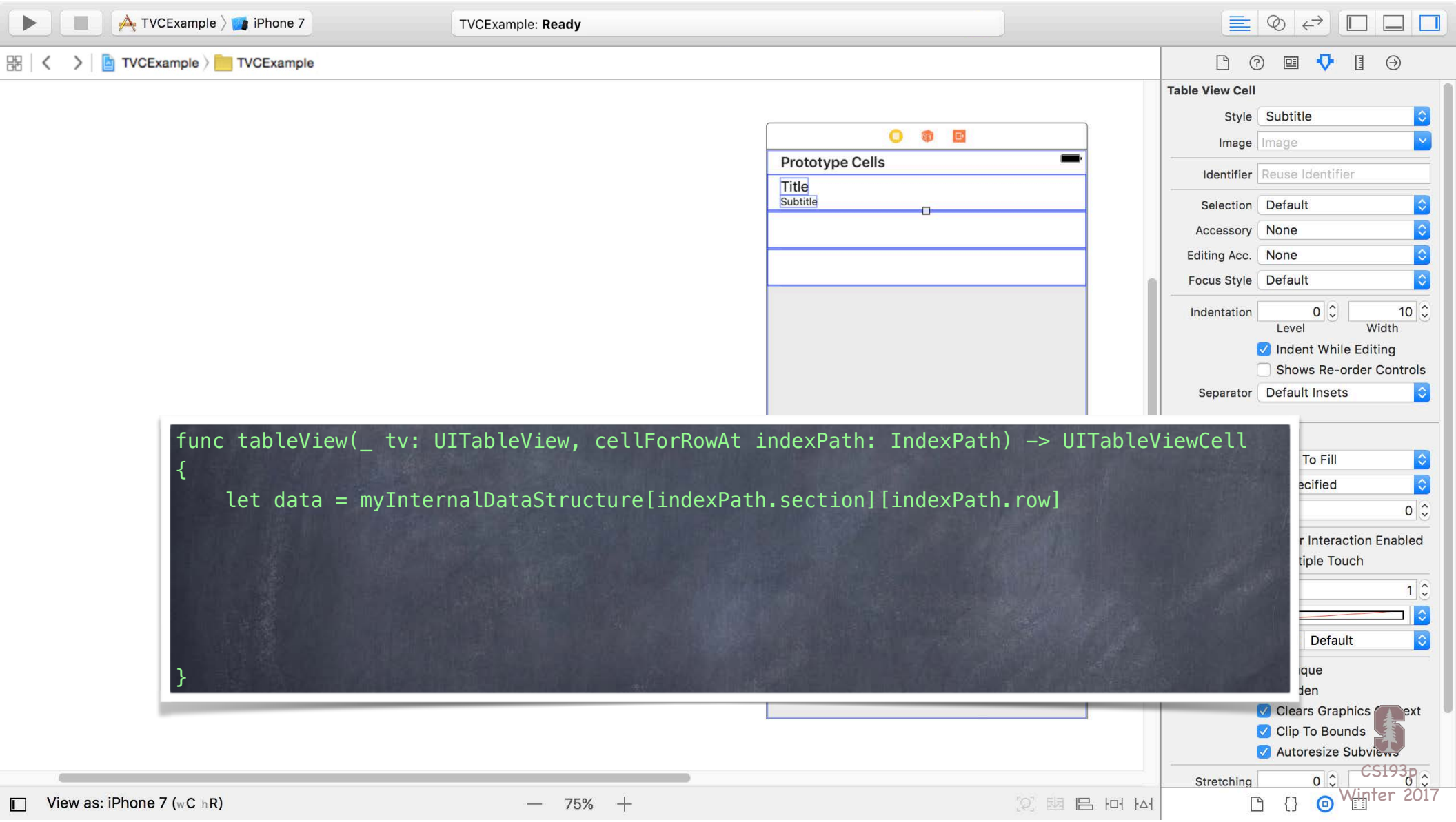
```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]

    let cell = . . . // create a UITableViewCell and load it up with data

    return cell
}
```

Customizing Each Row

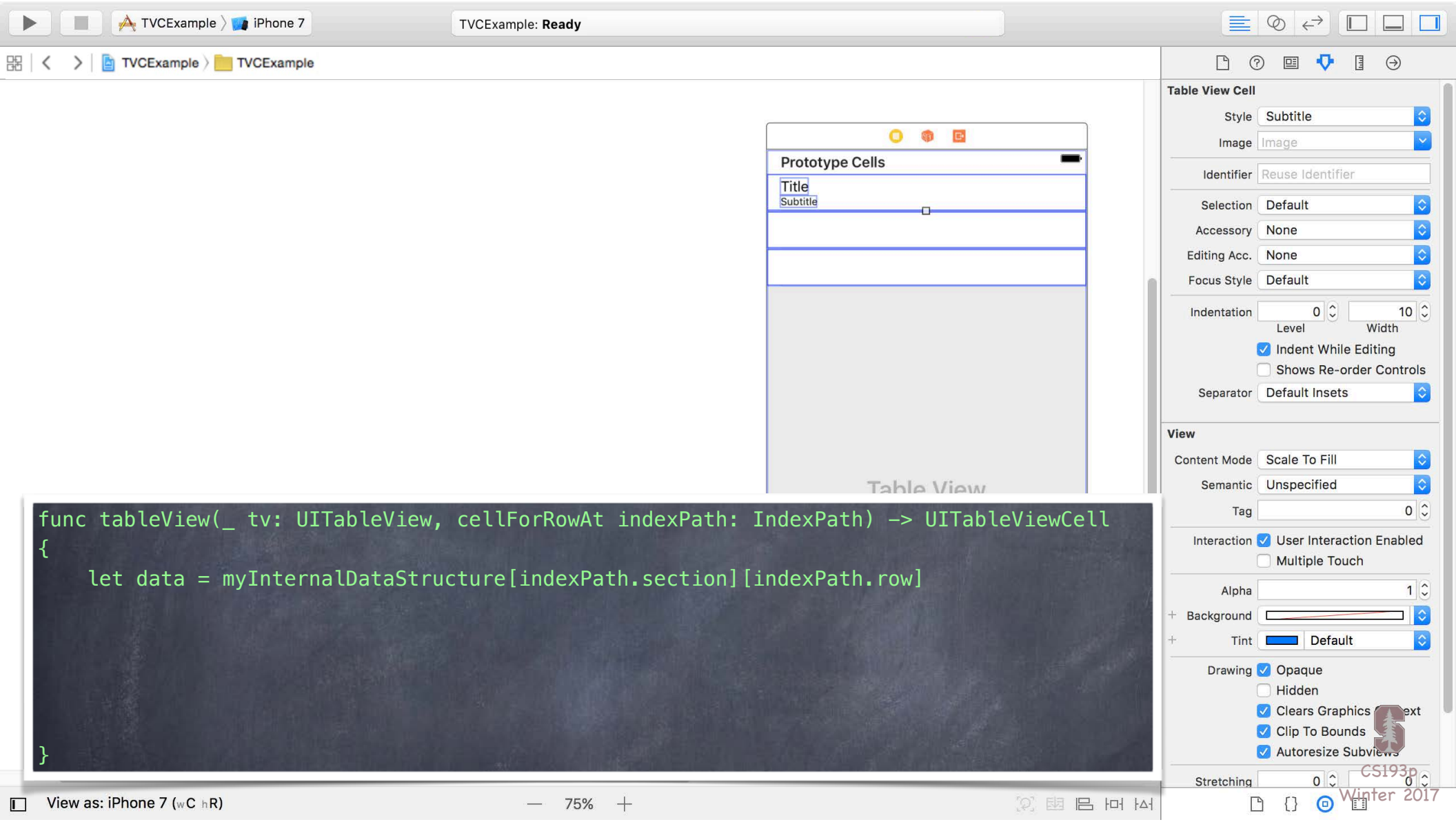
- Providing a UIView to draw each row …
  - It has to be a UITableViewCell (which is a subclass of UIView) or subclass thereof
  - Don't worry, if you have 10,000 rows, only the visible ones will have a UITableViewCell
  - But this means that UITableViewCells are reused as rows appear and disappear
  - This has ramifications for multithreaded situations, so be careful in that scenario

  The UITableView will ask its UITableViewDataSource for the UITableViewCell for a row …

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]

    let cell = . . . // create a UITableViewCell and load it up with data

    return cell
}
}
```

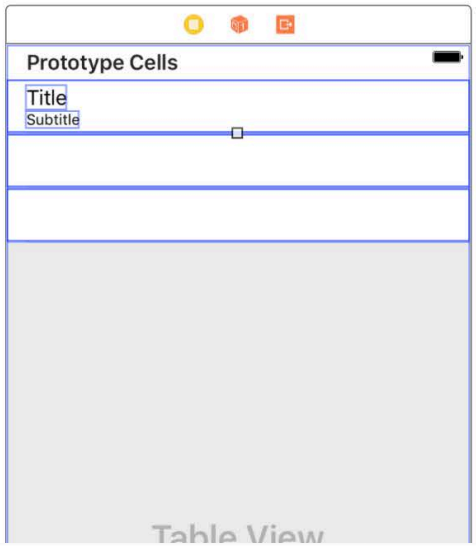



```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]



}
```




```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]



}
```

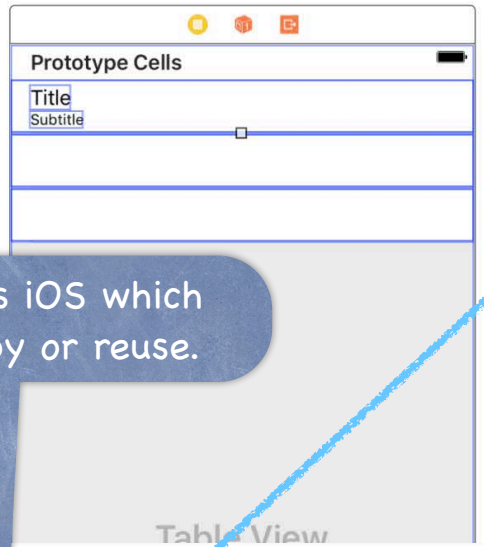
This method gets a UITableViewCell for us either by reusing one that has gone off screen or by making a copy of one of our prototypes in the storyboard.



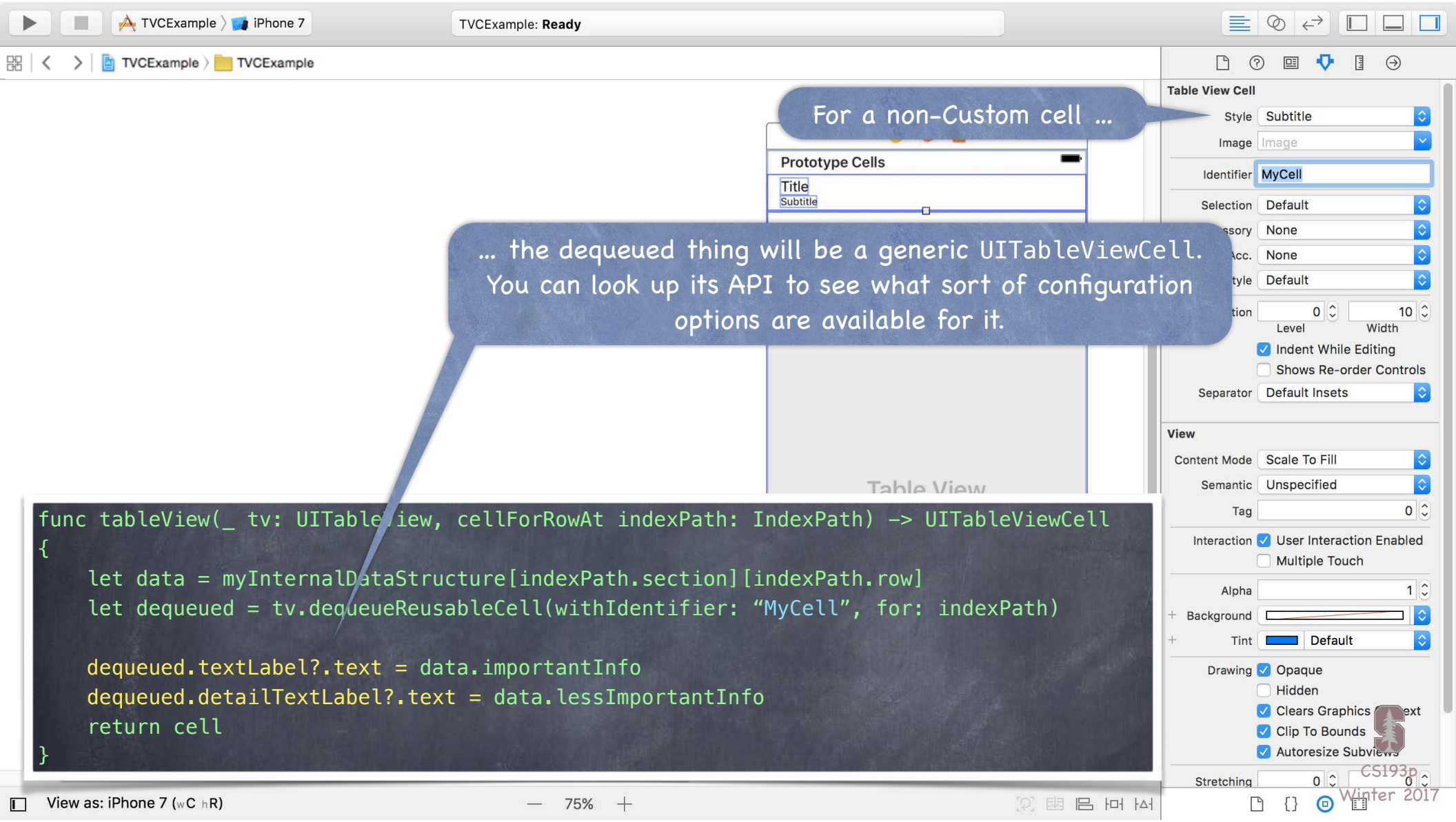

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCell", for: indexPath)



}
```

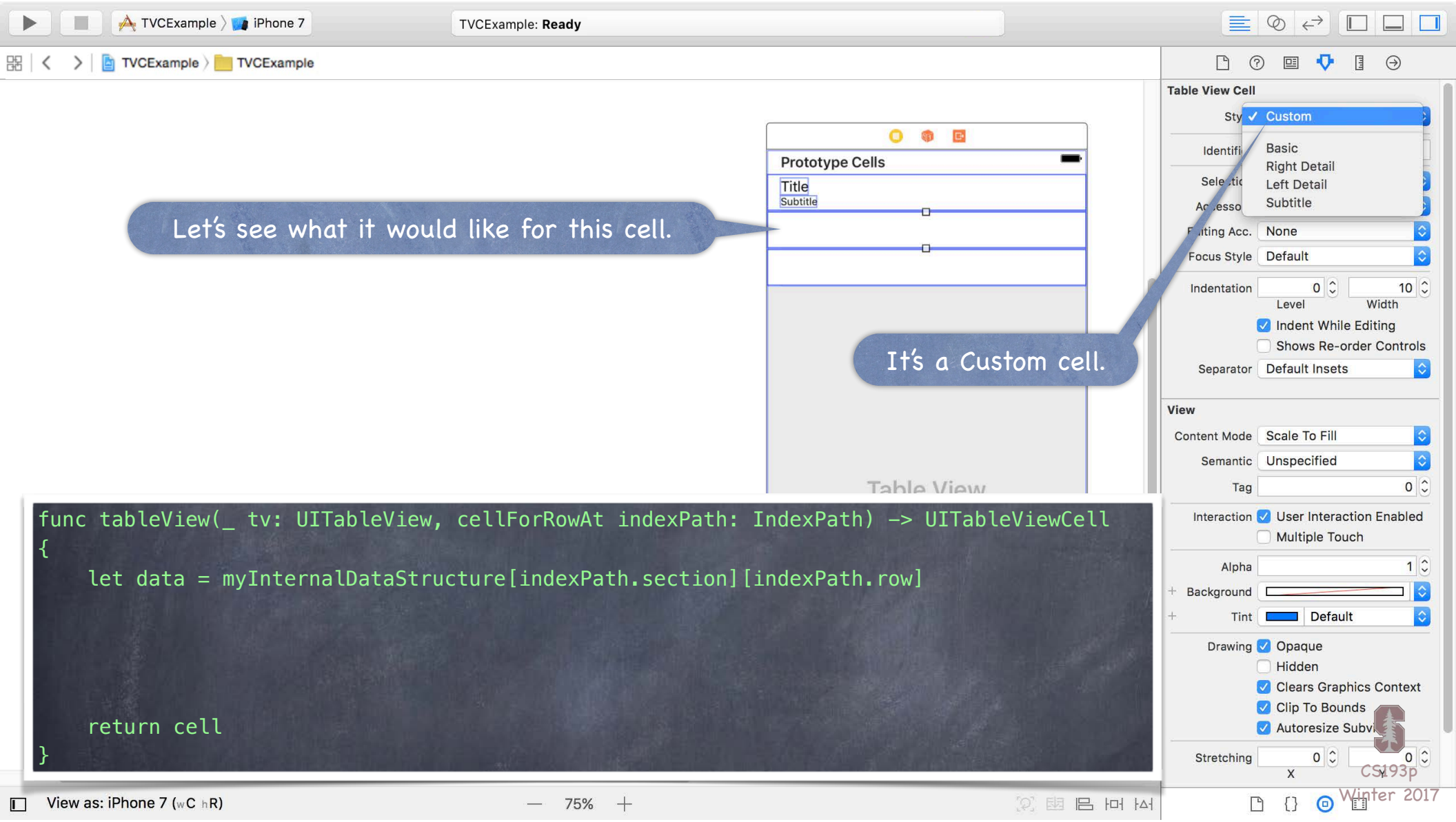
This method gets a UITableViewCell for us either by reusing one that has gone off screen or by making a copy of one of our prototypes in the storyboard.

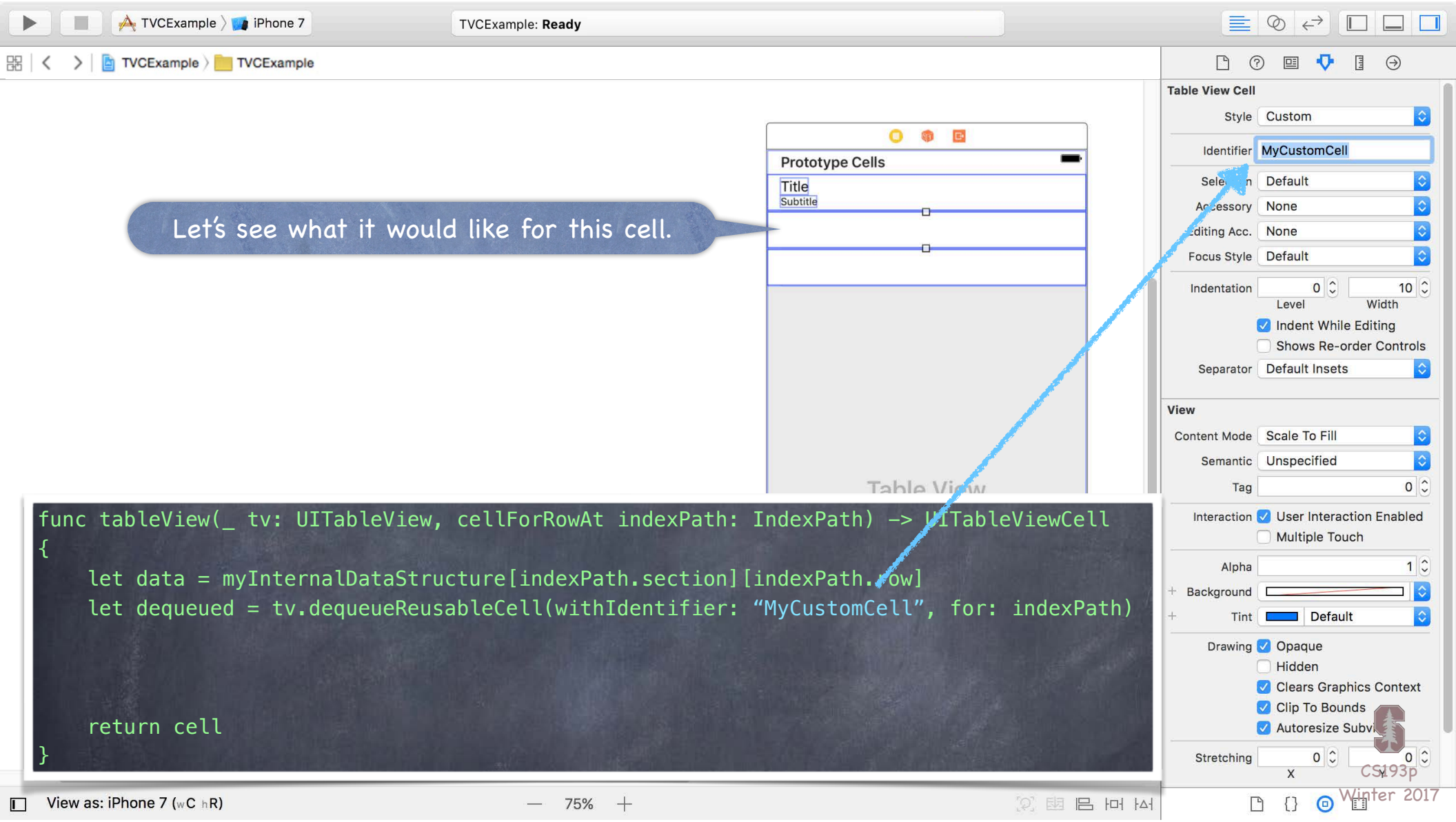
This String tells iOS which prototype to copy or reuse.

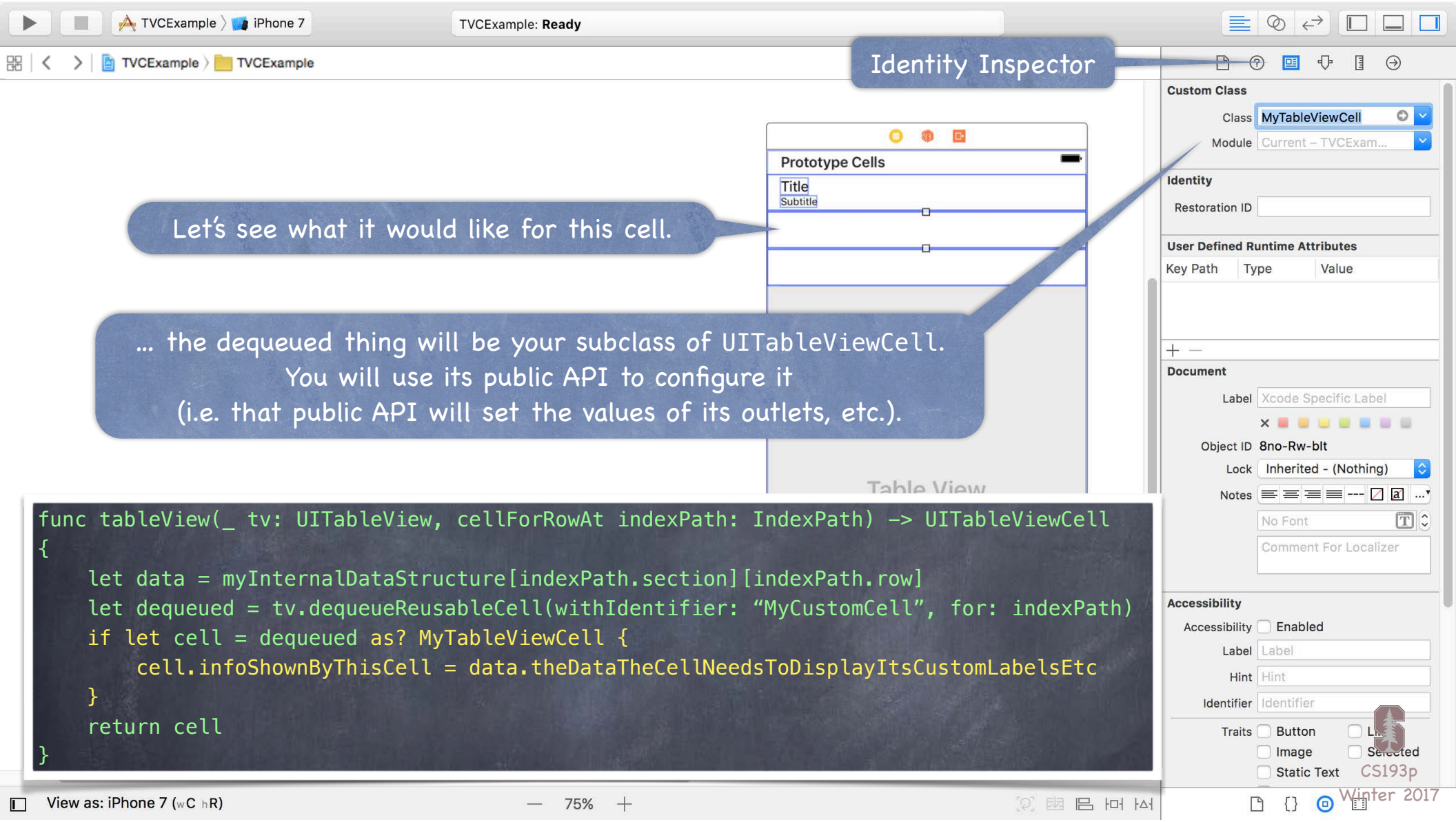


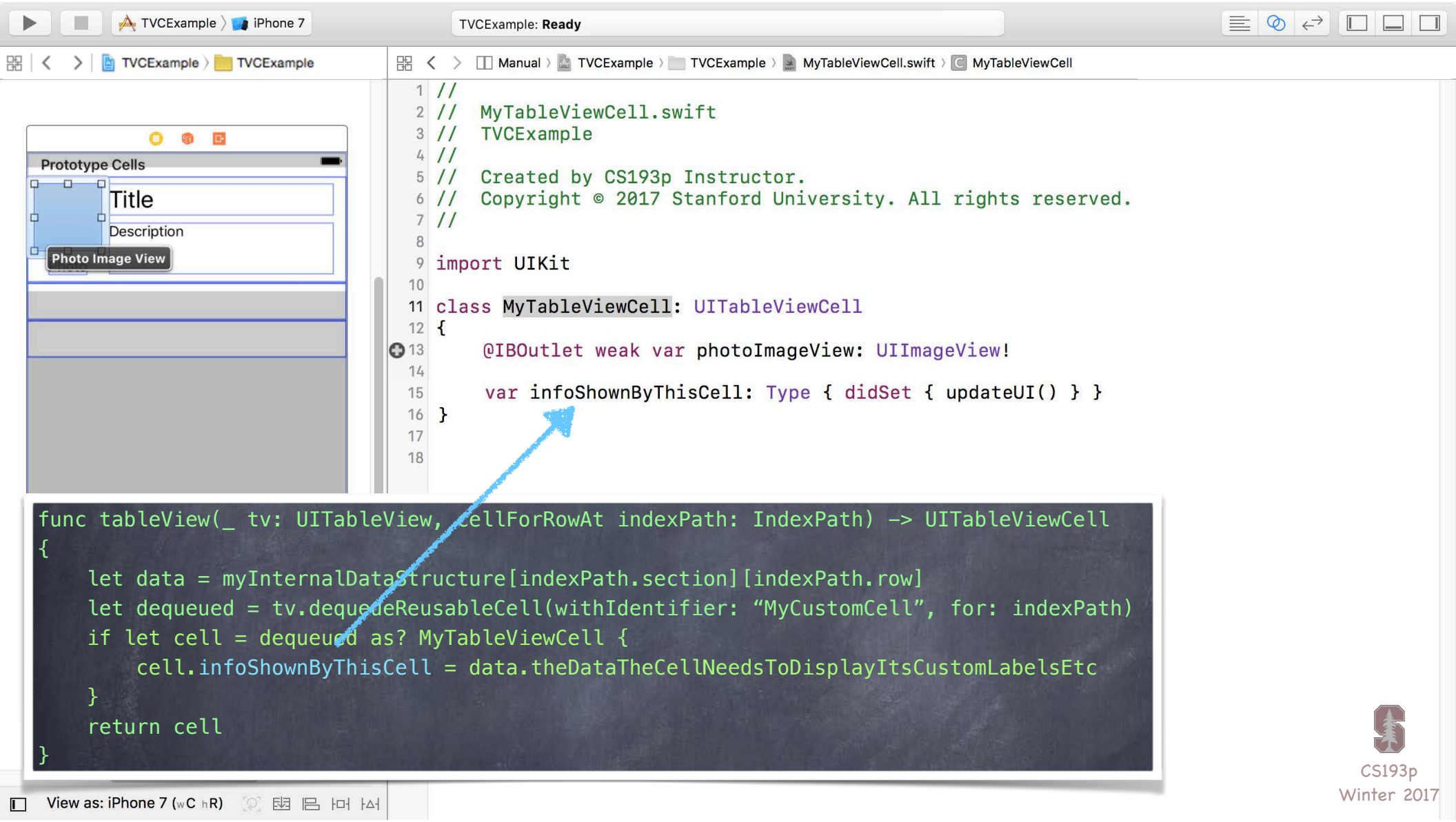

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCell", for: indexPath)



}
```


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
For a non-Custom cell …
Prototype Cells
Title
Subtitle
… the dequeued thing will be a generic UITableViewCell. You can look up its API to see what sort of configuration options are available for it.
Table View
Table View Cell
Style Subtitle
Image Image
Identifier MyCell
Selection Default
None
None
Default
Level
Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag
Interaction User Interaction Enabled
Multiple Touch
Alpha
Background
Tint Default
Drawing Opaque
Hidden
Clips To Bounds
Autoresize Subviews
Stretching
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
let data = myInternalDataStructure[indexPath.section][indexPath.row]
let dequeued = tv.dequeueReusableCell(withIdentifier: “MyCell”, for: indexPath)
dequeued.textLabel?.text = data.importantInfo
dequeued.detailTextLabel?.text = data.lessImportantInfo
return cell
}
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Let's see what it would like for this cell.
Prototype Cells
Title
Subtitle
Table View
It's a Custom cell.
Table View Cell
Custom
Basic
Right Detail
Left Detail
Subtitle
None
Default
Indentation
Level
Width
Indent While Editing
Shows Re-order Controls
Separator
Default Insets
View
Content Mode
Scale To Fill
Semantic
Unspecified
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Stretching
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
let data = myInternalDataStructure[indexPath.section][indexPath.row]
return cell
}
View as: iPhone 7
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Let's see what it would like for this cell.
Prototype Cells
Title
Subtitle
Table View
Table View Cell
Style Custom
Identifier MyCustomCell
Selection Default
Accessory None
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip To Bounds
Autoresize Subv
Stretching 0 X 0
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
let data = myInternalDataStructure[indexPath.section][indexPath.row]
let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCustomCell", for: indexPath)
return cell
}
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Identity Inspector
Prototype Cells
Title
Subtitle
Let's see what it would like for this cell.
... the dequeued thing will be your subclass of UITableViewCell.
You will use its public API to configure it
(i.e. that public API will set the values of its outlets, etc.).
Table View
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
let data = myInternalDataStructure[indexPath.section][indexPath.row]
let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCustomCell", for: indexPath)
if let cell = dequeued as? MyTableViewCell {
cell.infoShownByThisCell = data.theDataTheCellNeedsToDisplayItsCustomLabelsEtc
}
return cell
}
Custom Class
Class MyTableViewCell
Module Current – TVCExam...
Identity
Restoration ID
User Defined Runtime Attributes
Key Path
Type
Value
Document
Label Xcode Specific Label
Object ID 8no-Rw-blt
Lock Inherited - (Nothing)
Notes
No Font
Comment For Localizer
Accessibility
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button
Image
Static Text
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Manual 〉 TVCExample 〉 TVCExample 〉 MyTableViewCell.swift 〉 MyTableViewCell
Prototype Cells
Title
Description
Photo Image View
1 //
2 // MyTableViewCell.swift
3 // TVCExample
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewCell: UITableViewCell
12 {
13 @IBOutlet weak var photoImageView: UIImageView!
14
15 var infoShownByThisCell: Type { didSet { updateUI() } }
16 }
17
18
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
let data = myInternalDataStructure[indexPath.section][indexPath.row]
let dequeued = tv.dequeueReusableCell(withIdentifier: “MyCustomCell”, for: indexPath)
if let cell = dequeued as? MyTableViewCell {
cell.infoShownByThisCell = data.theDataTheCellNeedsToDisplayItsCustomLabelsEtc
}
return cell
}
View as: iPhone 7 (wC hR)

UITableViewDataSource

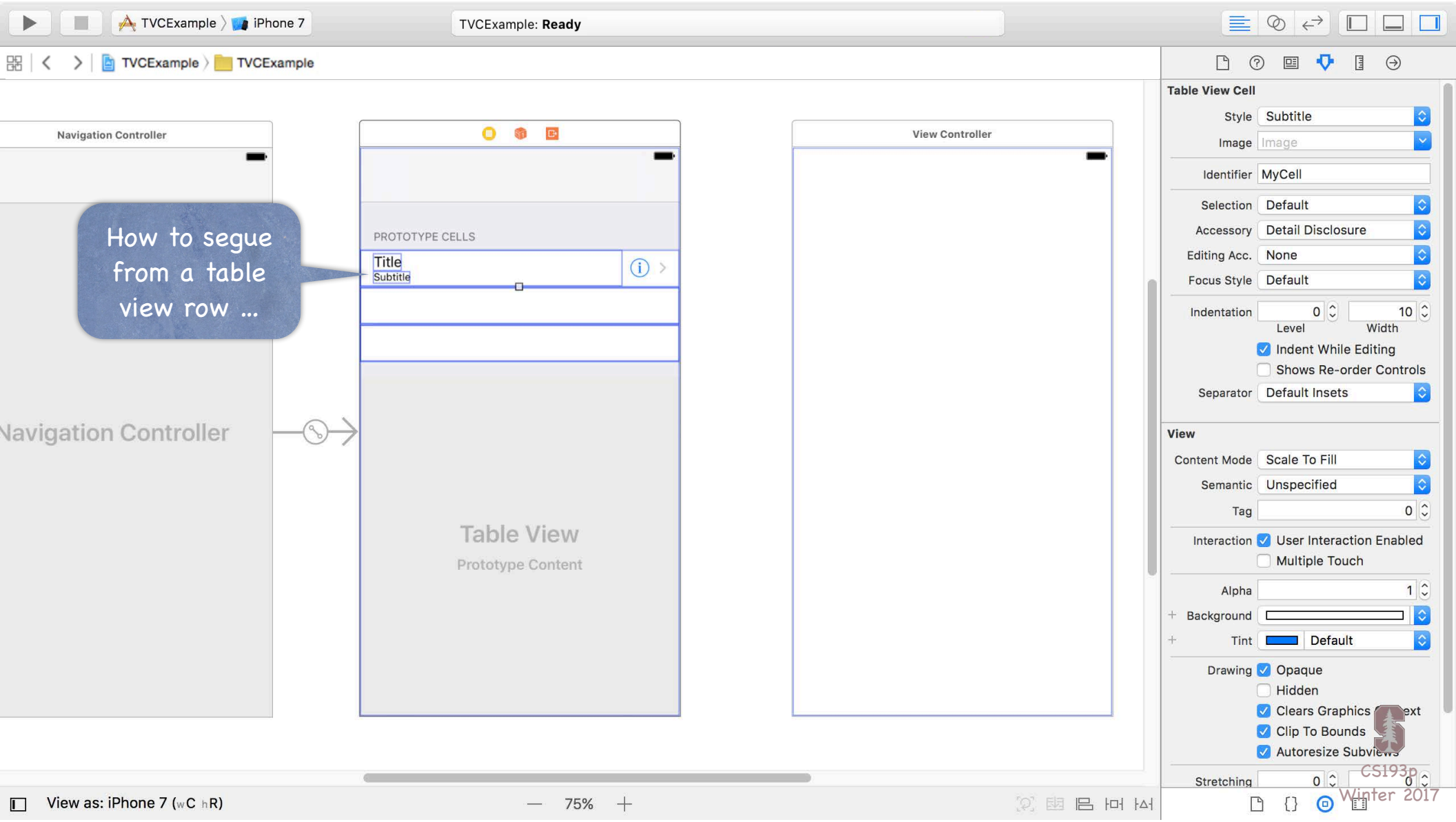
- How does a dynamic table know how many rows there are?
  - And how many sections, too, of course?
  - Via these UITableViewDataSource protocol methods …

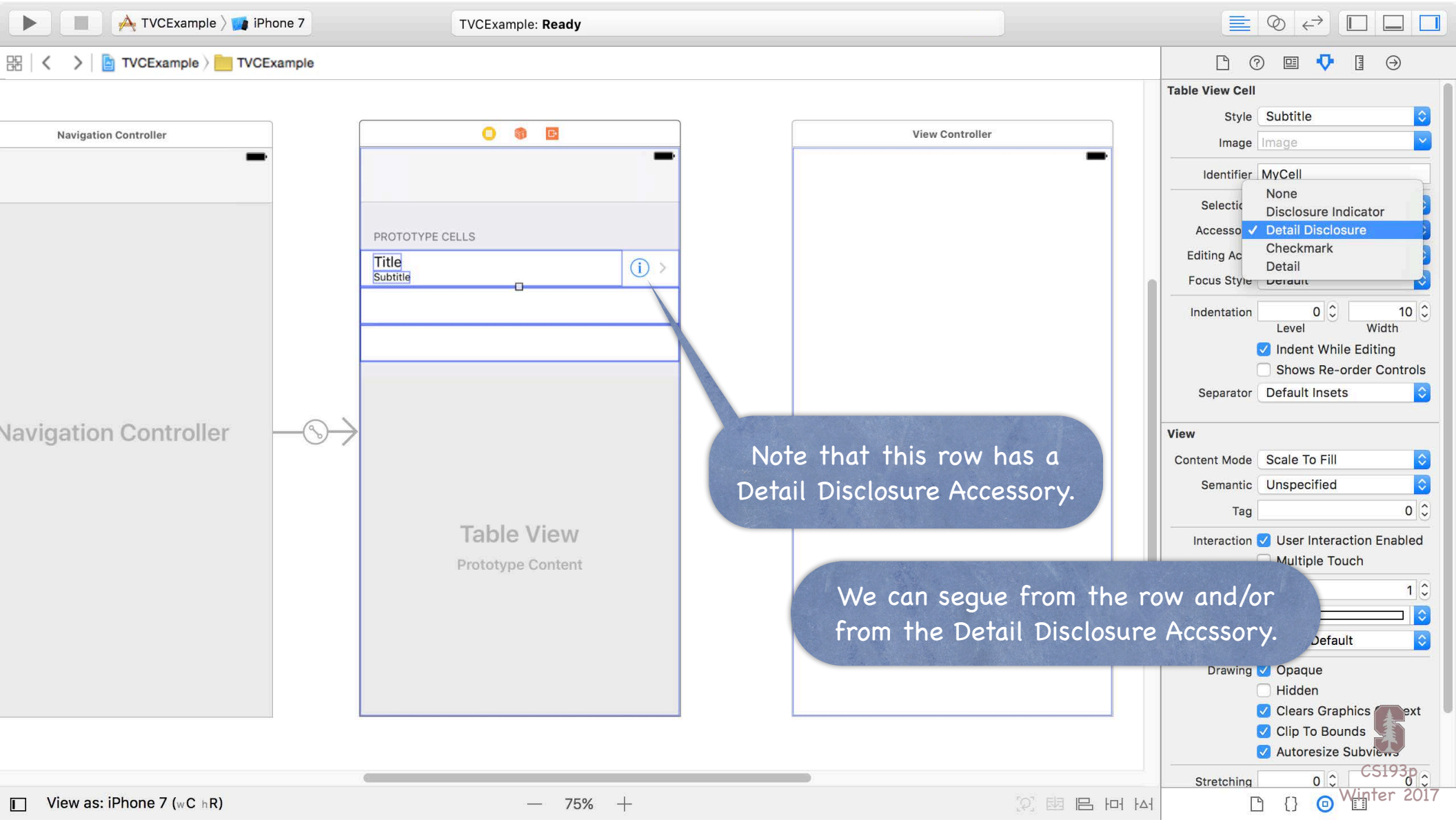
```
func numberOfSections(in tv: UITableView) -> Int
func tableView(_ tv: UITableView, numberOfRowsInSection: Int) -> Int
```

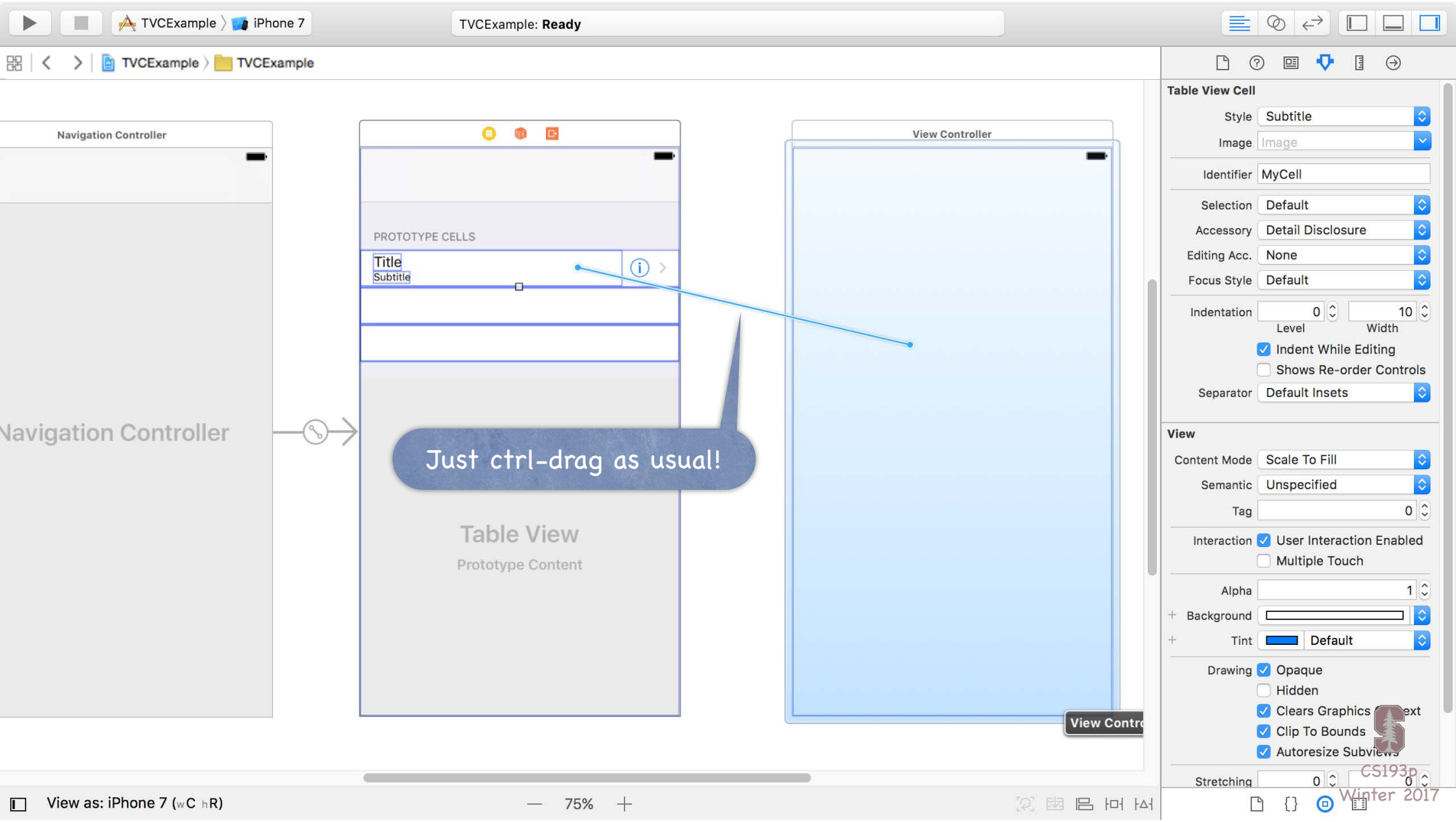
- Number of sections is 1 by default
  - In other words, if you don't implement numberOfSectionsInTableView, it will be 1
- No default for numberOfRowsInSection
  - This is a required method in this protocol (as is cellForRowAt)
- What about a static table?
  - Do not implement these dataSource methods for a static table
  - UITableViewController will take care of that for you
  - You edit the data directly in the storyboard

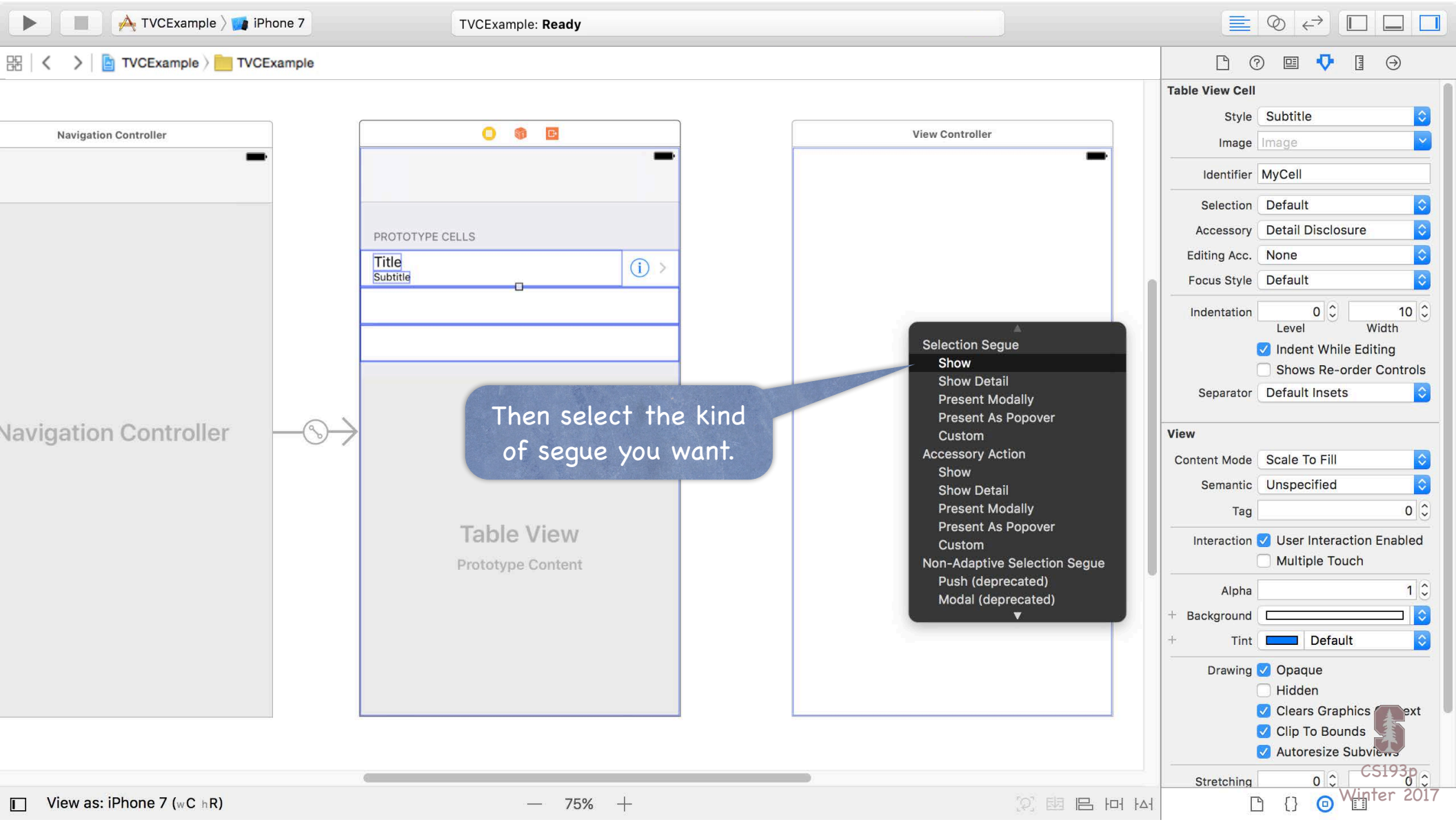
UITableViewDataSource

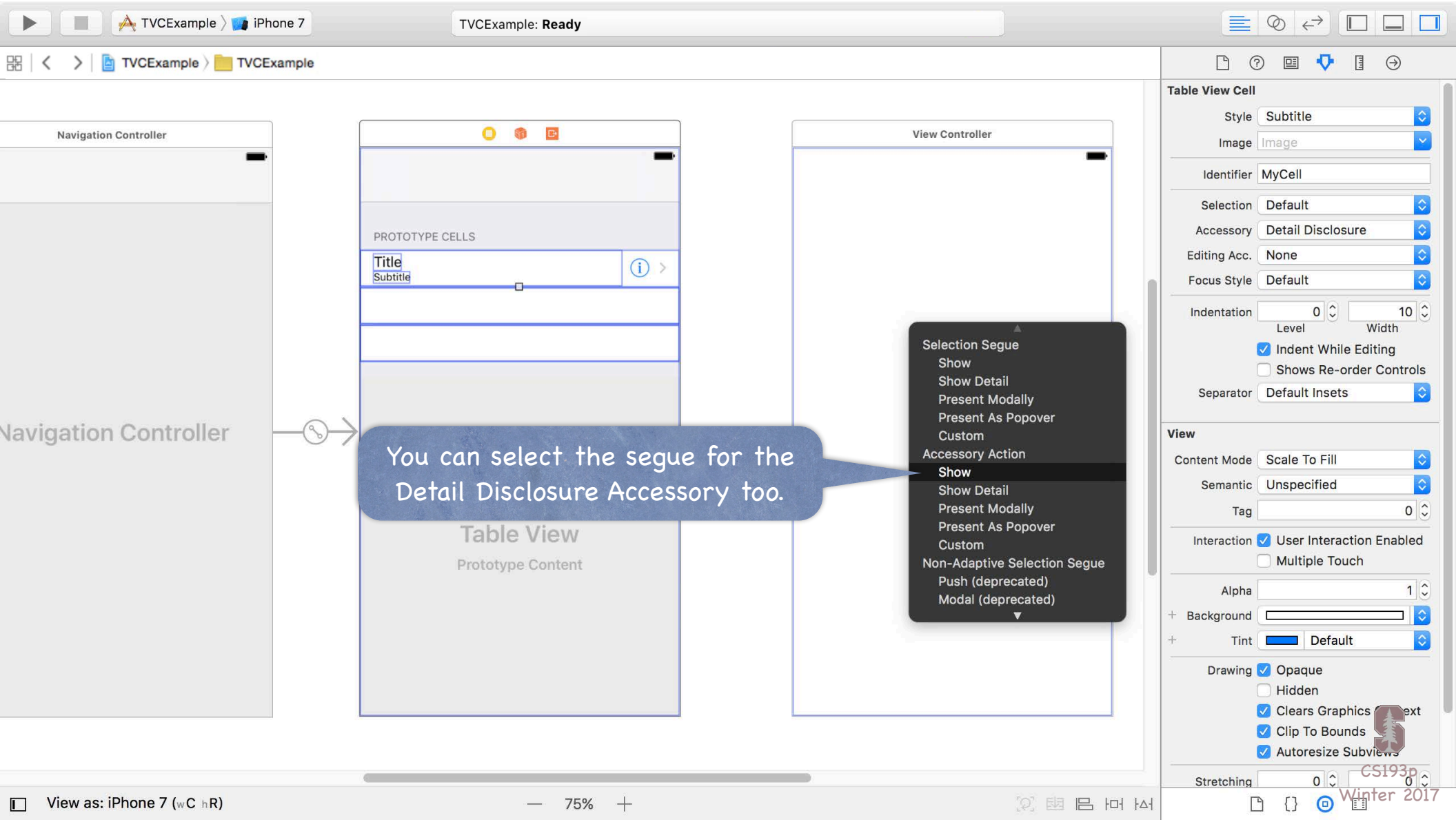
- Summary
  - Loading your table view with data is simple …
    1. set the table view's `dataSource` to your Controller (automatic with `UITableViewController`)
    2. implement `numberOfSections` and `numberOfRowsInSection`
    3. implement `cellForRowAt` to return loaded-up `UITableViewCells`
- Section titles are also considered part of the table's "data"
  - So you return this information via `UITableViewDataSource` methods …
  - `func tableView(UITableView, titleFor{Header,Footer}InSection: Int) -> String`
  - If a `String` is not sufficient, the `UITableView`'s `delegate` can provide a `UIView`
- There are a number of other methods in this `protocol`
  - But we're not going to cover them in lecture
  - They are mostly about dealing with editing the table by deleting/moving/inserting rows
  - That's because when rows are deleted, inserted or moved, it would likely modify the Model (and we're talking about the `UITableViewDataSource` protocol here)

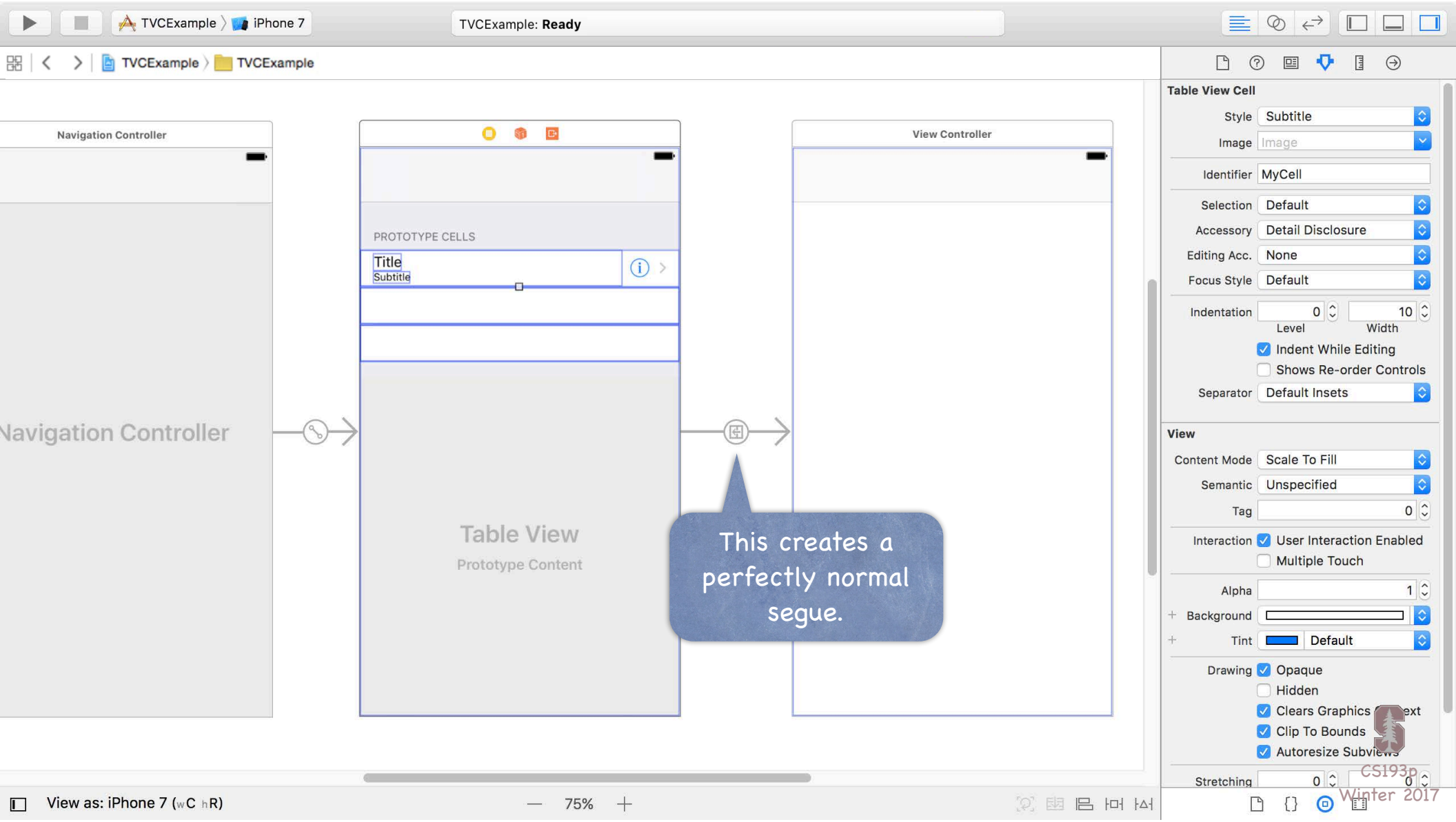


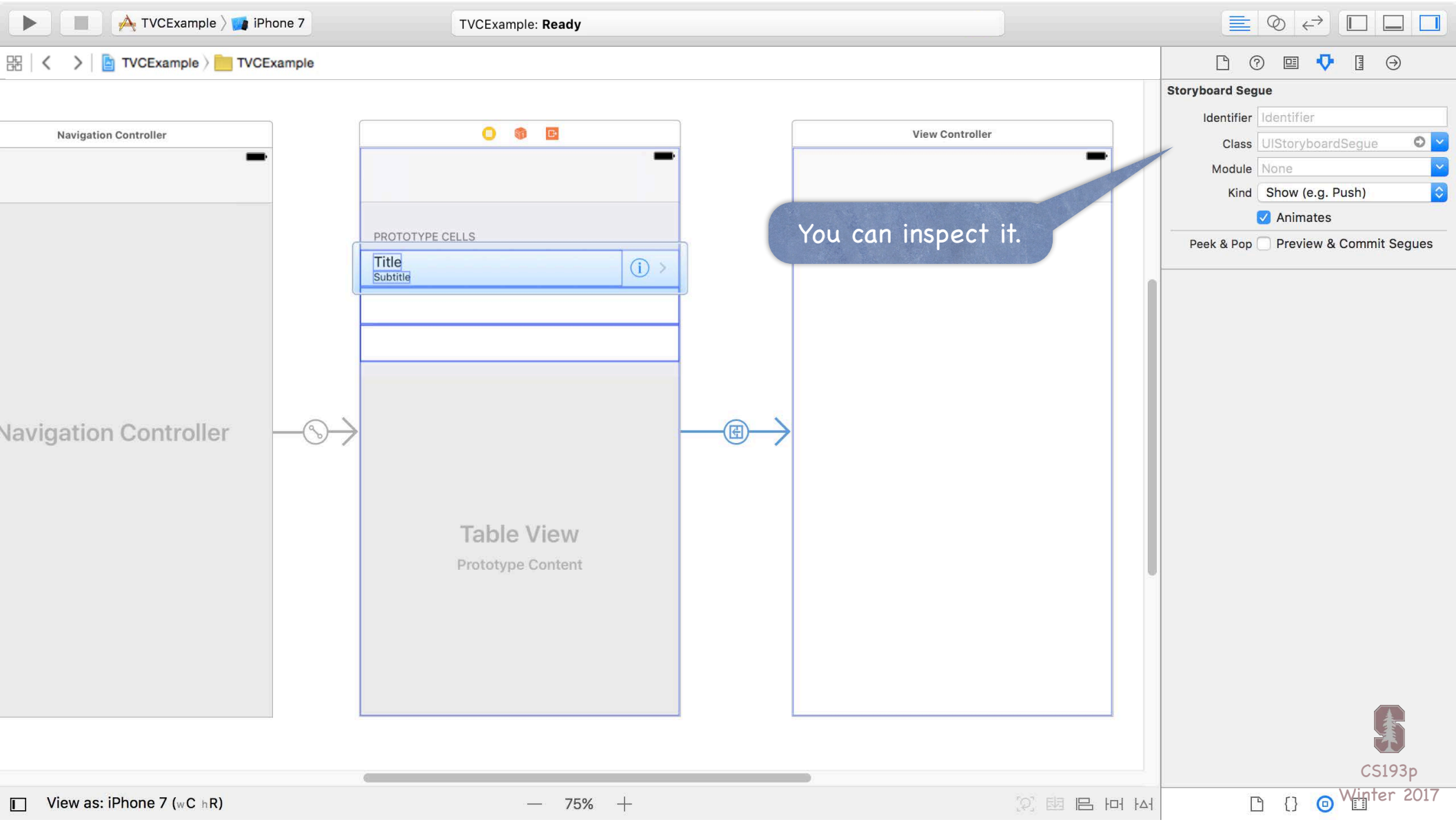



TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Navigation Controller
Navigation Controller
PROTOTYPE CELLS
Title
Subtitle
Just ctrl-drag as usual!
Table View
Prototype Content
View Controller
View Contro
Table View Cell
Style Subtitle
Image Image
Identifier MyCell
Selection Default
Accessory Detail Disclosure
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017




TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Navigation Controller
Navigation Controller
PROTOTYPE CELLS
Title
Subtitle
Table View
Prototype Content
View Controller
You can select the segue for the Detail Disclosure Accessory too.
Selection Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
Accessory Action
Show
Show Detail
Present Modally
Present As Popover
Custom
Non-Adaptive Selection Segue
Push (deprecated)
Modal (deprecated)
Table View Cell
Style Subtitle
Image Image
Identifier MyCell
Selection Default
Accessory Detail Disclosure
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Navigation Controller
Navigation Controller
PROTOTYPE CELLS
Title
Subtitle
Table View
Prototype Content
View Controller
This creates a perfectly normal segue.
Table View Cell
Style Subtitle
Image Image
Identifier MyCell
Selection Default
Accessory Detail Disclosure
Editing Acc. None
Focus Style Default
Indentation 0 Level 10 Width
Indent While Editing
Shows Re-order Controls
Separator Default Insets
View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque
Hidden
Clears Graphics
Clip To Bounds
Autoresize Subviews
Stretching 0 0
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017


TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Navigation Controller
PROTOTYPE CELLS
Title
Subtitle
View Controller
You can inspect it.
Navigation Controller
Table View
Prototype Content
Storyboard Segue
Identifier
Identifier
Class
UIStoryboardSegue
Module
None
Kind
Show (e.g. Push)
Animates
Peek & Pop
Preview & Commit Segues
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

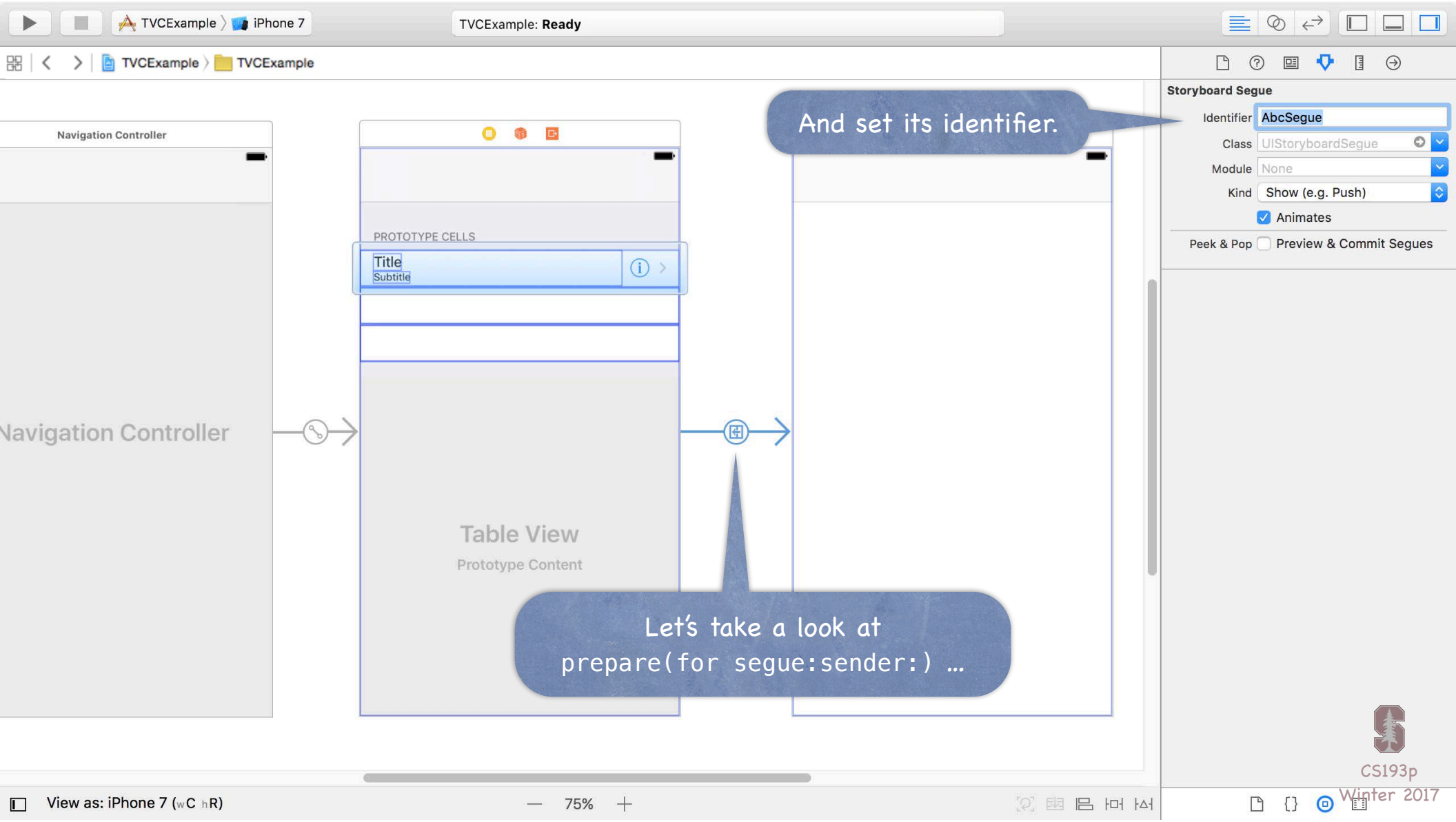

TVCExample 〉 iPhone 7
TVCExample: Ready
TVCExample 〉 TVCExample
Navigation Controller
Navigation Controller
PROTOTYPE CELLS
Title
Subtitle
Table View
Prototype Content
And set its identifier.
Let's take a look at
prepare(for segue:sender:) …
Storyboard Segue
Identifier AbcSegue
Class UIStoryboardSegue
Module None
Kind Show (e.g. Push)
Animates
Peek & Pop Preview & Commit Segues
View as: iPhone 7 (wC hR)
75%
CS193p
Winter 2017

Table View Segues

- Preparing to segue from a row in a table view

  The sender argument to prepareForSegue is the UITableViewCell of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:




        default: break
        }
    }
}
```

  You can see now why sender is Any

  Sometimes it's a UIButton, sometimes it's a UITableViewCell

Table View Segues

- Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:
            if let cell = sender as? MyTableViewCell {



            }
        default: break
        }
    }
}
```

So you will need to cast sender with as? to turn it into a UITableViewCell

If you have a custom UITableViewCell subclass, you can cast it to that if it matters

Table View Segues

- Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:
            if let cell = sender as? MyTableViewCell,
               let indexPath = tableView.indexPath(for: cell) {


            }
        default: break
        }
    }
}
```

indexPath(for cell:) does not accept Any. It has to be a UITableViewCell of some sort.

Usually we will need the IndexPath of the UITableViewCell

Because we use that to index into our internal data structures

Table View Segues

- Preparing to segue from a row in a table view

  The `sender` argument to `prepareForSegue` is the `UITableViewCell` of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:
            if let cell = sender as? MyTableViewCell,
               let indexPath = tableView.indexPath(for: cell),
               let seguedToMVC = segue.destination as? MyVC {

            }
        default: break
        }
    }
}
```

  Now we just get our destination MVC as the proper class as usual …

Table View Segues

- Preparing to segue from a row in a table view

  The `sender` argument to `prepareForSegue` is the `UITableViewCell` of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:
            if let cell = sender as? MyTableViewCell,
               let indexPath = tableView.indexPath(for: cell),
               let seguedToMVC = segue.destination as? MyVC {
                   seguedToMVC.publicAPI = data[indexPath.section][indexPath.row]
            }
        default: break
        }
    }
}
```

  and then get data from our internal data structure using the `IndexPath`'s `section` and `row`

Table View Segues

- Preparing to segue from a row in a table view

  The sender argument to prepareForSegue is the UITableViewCell of that row …

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case “XyzSegue”: // handle XyzSegue here
        case “AbcSegue”:
            if let cell = sender as? MyTableViewCell,
               let indexPath = tableView.indexPath(for: cell),
               let seguedToMVC = segue.destination as? MyVC {
                   seguedToMVC.publicAPI = data[indexPath.section][indexPath.row]
            }
        default: break
        }
    }
}
```

  and then get data from our internal data structure using the IndexPath's section and row and use that information to prepare the segued-to API using its public API

UITableViewDelegate

- So far we've only talked about the UITableView's dataSource
  - But UITableView has another protocol-driven delegate called its delegate
- The delegate controls how the UITableView is displayed
  - Not the data it displays (that's the dataSource's job), how it is displayed
- Common for dataSource and delegate to be the same object
  - Usually the Controller of the MVC containing the UITableView
  - Again, this is set up automatically for you if you use UITableViewController
- The delegate also lets you observe what the table view is doing
  - Especially responding to when the user selects a row
  - Usually you will just segue when this happens, but if you want to track it directly …

UITableView "Target/Action"

- UITableViewDelegate method sent when row is selected

  This is sort of like "table view target/action" (only needed if you're not segueing, of course)

  Example: if the master in a split view wants to update the detail without segueing to a new one

```
func tableView(UITableView, didSelectRowAt indexPath: IndexPath) {
    // go do something based on information about my Model
    // corresponding to indexPath.row in indexPath.section
    // maybe directly update the Detail if I'm the Master in a split view?
}
```

- Delegate method sent when Detail Disclosure button is touched 

  `func tableView(UITableView, accessoryButtonTappedForRowWith indexPath: IndexPath)`

  Again, you can just segue from that Detail Disclosure button if you prefer

UITableViewDelegate

- Lots and lots of other `delegate` methods
  - `will`/`did` methods for both selecting and deselecting rows
  - Providing `UIView` objects to draw section headers and footers
  - Handling editing rows (moving them around with touch gestures)
  - `willBegin`/`didEnd` notifications for editing (i.e. deleting, inserting, moving rows)
  - Copying/pasting rows

UITableView

- What if your Model changes?

  `func reloadData()`

  Causes the `UITableView` to call `numberOfSectionsInTableView` and `numberOfRowsInSection` all over again and then `cellForRowAt` on each visible row

  Relatively heavyweight, but if your entire data structure changes, that's what you need

  If only part of your Model changes, there are lighter-weight reloaders, for example …

  `func reloadRows(at indexPaths: [IndexPath], with animation: UITableViewRowAnimation)`

UITableView

- Controlling the height of rows
  - Row height can be fixed (`UITableView`'s `var rowHeight: CGFloat`)
  - Or it can be determined using autolayout (`rowHeight = UITableViewAutomaticDimension`)
  - If you do automatic, help the table view out by setting `estimatedRowHeight` to something
  - The `UITableView`'s `delegate` can also control row heights …
  - `func tableView(UITableView, {estimated}heightForRowAt indexPath: IndexPath) -> CGFloat`
  - Beware: the non-estimated version of this could get called A LOT if you have a big table

UITableView

- There are dozens of other methods in `UITableView` itself
  - Setting headers and footers for the entire table.
  - Controlling the look (separator style and color, default row height, etc.).
  - Getting cell information (cell for index path, index path for cell, visible cells, etc.).
  - Scrolling to a row (`UITableView` is a subclass of `UIScrollView`).
  - Selection management (allows multiple selection, getting the selected row, etc.).
  - Moving, inserting and deleting rows, etc.
  - As always, part of learning the material in this course is studying the documentation