COCOA

NEW PROJECT WIZARD

We will always use the new project wizard with the following settings:

- iOS / Application
- Single View Application
- Language: Swift
- Devices: Universal

PROJECT FILES I

You can find the following files in your newly created project:

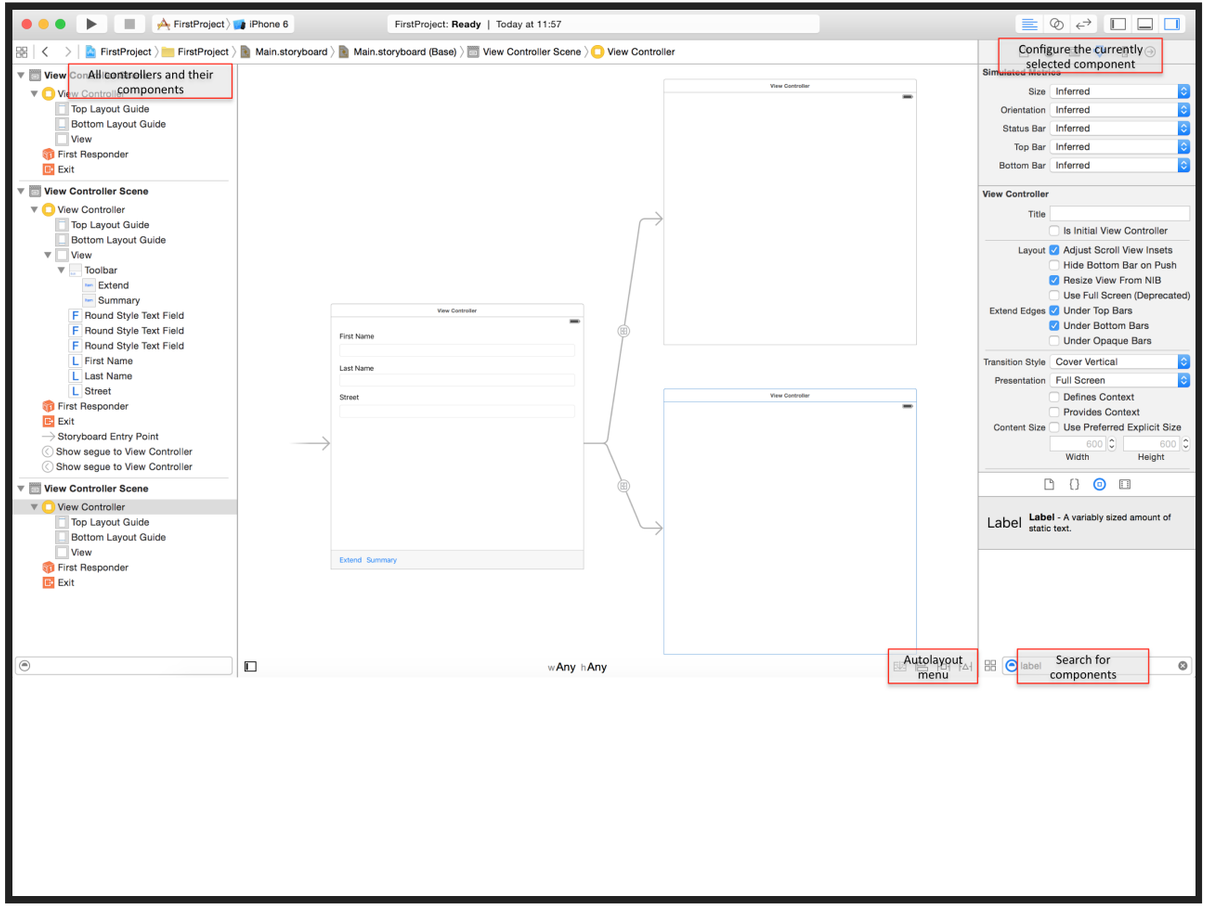
- AppDelegate.swift: The AppDelegate class provides methods for the lifetime of your app. You can use this for example to do something when the user switches your app to the background.
- ViewController.swift: The class for your initial view controller.
- Main.storyboard: The storyboard containing all view controllers, currently only one.
- LaunchScreen.storyboard: The storyboard for the launch screen.

PROJECT FILES II

Assets.xcassets: An empty folder where you can store your resources (icons, bitmaps, ...).
Info.plist: A key/value storage for your app configuration.
xxxTests and xxxUITests: Folders containing test classes for your model and UI.
Products: A folder where your output (typically your compiled app and tests) is stored.

STORYBOARD

- Can be used to graphically design your GUI.
- You can add controllers and views as well as transitions between controllers.
- You can have multiple Storyboards, each project comes with two, a default one (Main.storyboard) and one for the launch screen (LaunchScreen.storyboard).
- By default, the storyboard is shown in a graphical editor (interface builder). However, the data is stored as XML (you can open the XML by using the "Open As / Source Code" menu entry.


All controllers and their components
Configure the currently selected component
Autolayout menu
Search for components
wAny hAny

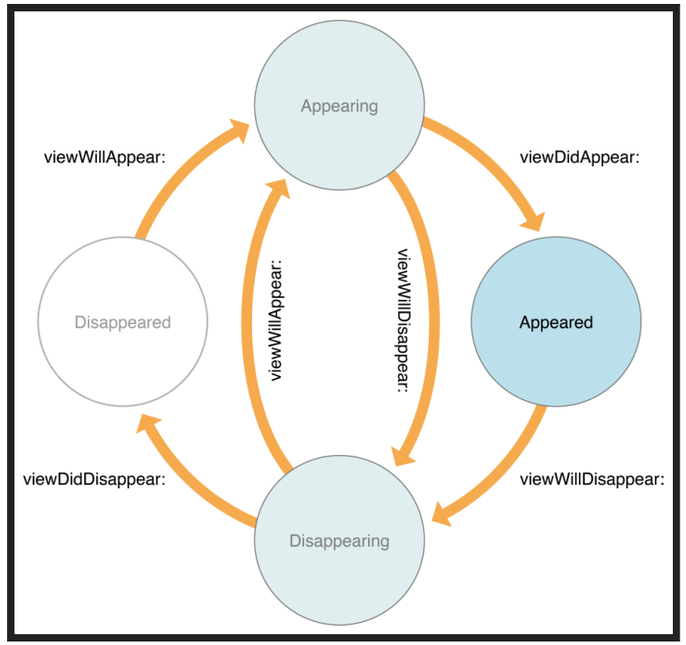
DISCUSS

What are the advantages/disadvantages when using a storyboard

COCOA UIVIEWCONTROLLER

- Each page you see in the app is based on one or more UIViewControllers.
- The UIViewController connects the view and model (MVC pattern).
- You typically inherit from UIViewController and create your own logic.
- You typically override the viewDidLoad method to initalize all components. You update components on the viewWillAppear method.
- When you switch between pages, iOS might reuse your UIViewController.

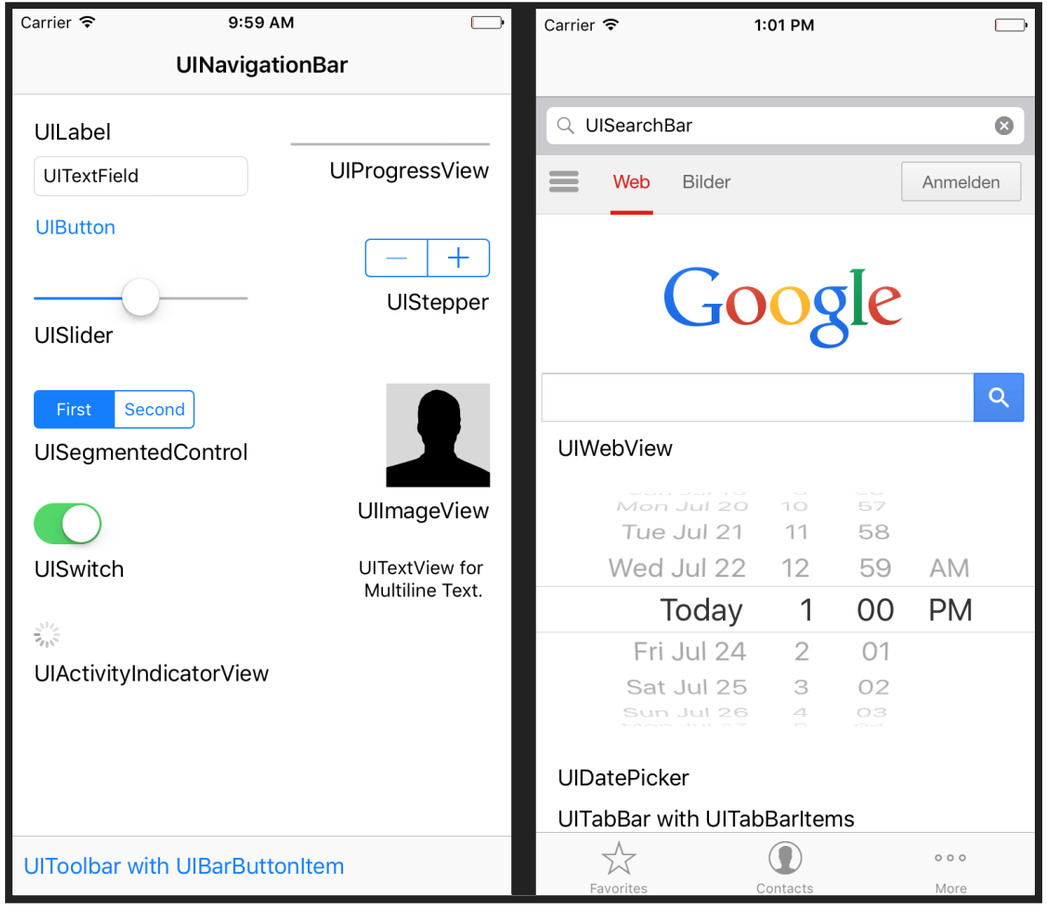
COCOA UIVIEWCONTROLLER STATES



Reproduced from the official Apple Documentation on UIViewControllers.

COCOA COMPONENTS I

- Cocoa provides a number of components.
- Some of them are shown on the next slide.
- There are others like UITableView, UIStackView, UIPageNavigation.
- Use a storyboard to create them in your UIViewController.


Carrier
9:59 AM
UINavigationBar
UILabel
UITextField
UIButton
UISlider
First
Second
UISegmentedControl
UISwitch
UIActivityIndicatorView
UIProgressView
UIStepper
UIImageView
UITextView for Multiline Text.
UIToolbar with UIBarButtonItem
Carrier
1:01 PM
UISearchBar
Web
Bilder
Anmelden
Google
UIWebView
Tue Jul 21 11 58
Wed Jul 22 12 59 AM
Today 1 00 PM
Fri Jul 24 2 01
Sat Jul 25 3 02
UIDatePicker
UITabBar with UITabBarItems
Favorites
Contacts
More

EXERCISE

Create a new project with the mentioned settings and add two textfields "first name" and "last name" as well as a button.

COCOA COMPONENTS: UIVIEW

UIView is

- the base component, where all other components inherit from.
- a simple rectangular space with a size and position (origin (0,0) is the upper left corner).
- a receiver for events (ie. a touch event).
- a possible parent for subviews.

COCOA COMPONENTS IN CODE

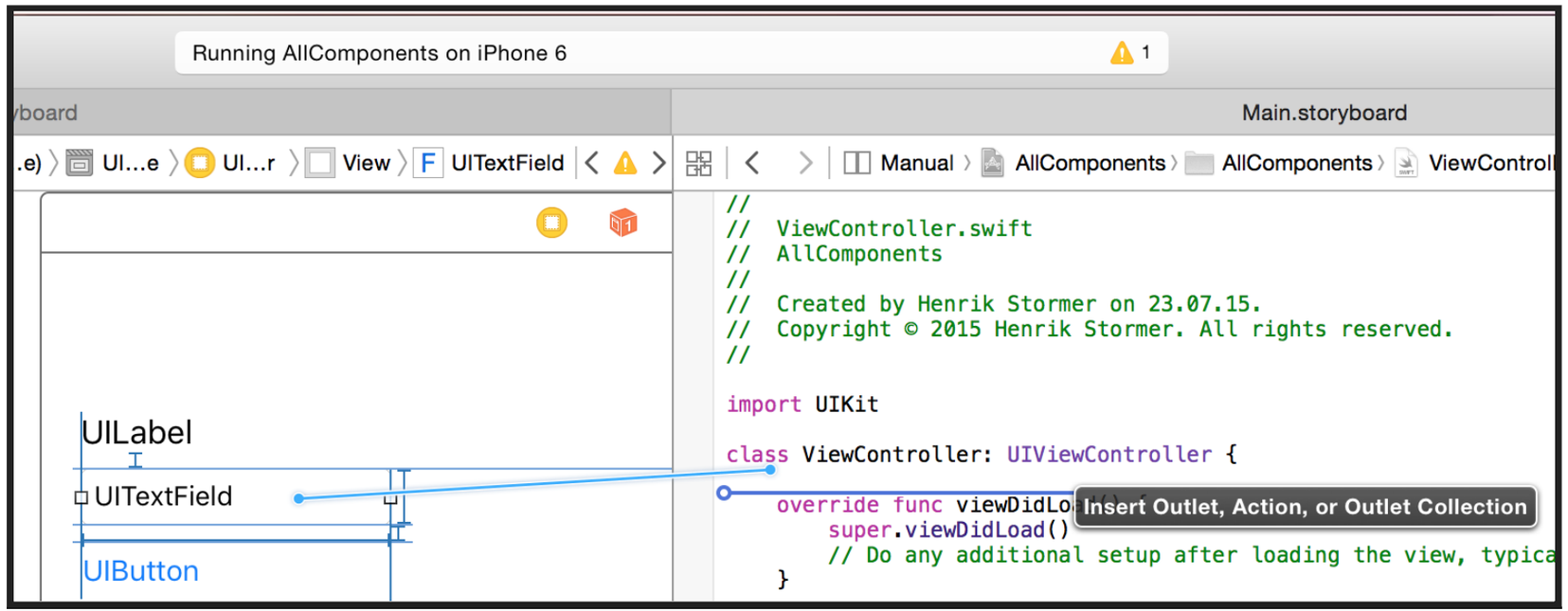
```
//all components can also be created in code
//you can, for example, create a UITextView in your viewDidLoad method
override func viewDidLoad() {
        super.viewDidLoad()

        //define the position and size of the label (x, y, width and height)
        //we will see later that this is not such a good idea
        let field : UILabel = UILabel(frame: CGRectMake(1, 30, 200, 30))

        //set a text
        field.text = "This is my label"

        //and, most important, add the label to the parent view
        self.view.addSubview(field);
}
```

CONFIGURE COMPONENTS

```
//you can configure a component either in code or in your storyboard
let label : UILabel = UILabel(frame: CGRectMake(1, 30, 200, 30))

//set a background color for the label
label.backgroundColor = UIColor.redColor()

//there are different line break modes available
label.lineBreakMode = NSLineBreakMode.ByTruncatingMiddle

//set a long text
label.text = "This is a longer text that does not fit in our label space"
```

OUTLET I

To use a component in code that you defined in your storyboard, create a reference (so called outlet) from the storyboard to your controller.



OUTLET II

You can create an outlet (a reference), but also an action. An action is a method that is called when en event occurs, for example when a button is clicked or a value changed.

EXERCISE

Define an action for the button's click event on your project. Use Swift's print method to output a message when the button has been clicked. Test your code in the simulator.

SCREEN SIZE I

```
//all views have three properties
view.bounds //the space of the view
view.frame  //the space of the view in relation to the parent view
view.center //the center point in relation to the parent view
```

SCREEN SIZE II

Device	Points	Pixels	Notes
Non Retina iPhone	320x480	320x480	-
iPhone 4 / 4s	320x480	640x960	-
iPhone 5 / 5s	320x568	640x1136	-
iPhone 6	375x667	750x1334	-
iPhone 6 Plus	414x736	1080x1920	(rendered at 3x, downsampled)

AUTOLAYOUT

AUTOLAYOUT CONCEPT

- With autolayout, the developer can define a layout for all GUI elements.
- Autolayout allows to set minimum and maximum width and height as well as distances between elements that adapt automatically to different screen sizes.
- An iOS app should always use autolayout to ensure that the GUI looks nice on all devices.
- Autolayout works by defining constraints. A constraint defines a rule between two GUI elements.
- Autolayout constraints need to be fully defined. Missing constraints will result in an error when starting the app.

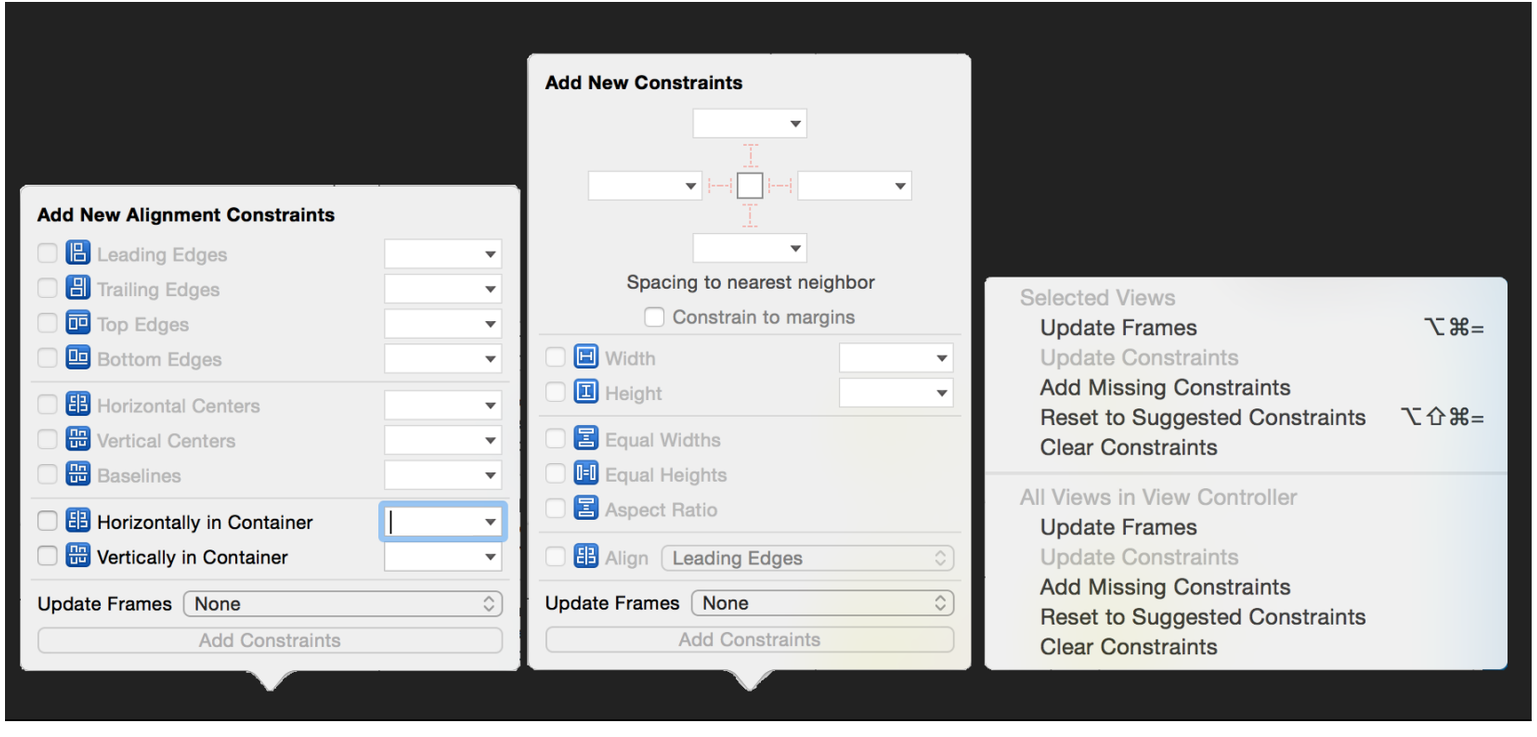
CONSTRAINTS

The following constraints can be defined in autolayout:

- Leading, trailing, top and bottom edges.
- Center horizontally or vertically
- Width and height, also equal width and height

CONSTRAINTS IN STORYBOARD I

Use the storyboard to define constraints using the three autolayout menus.



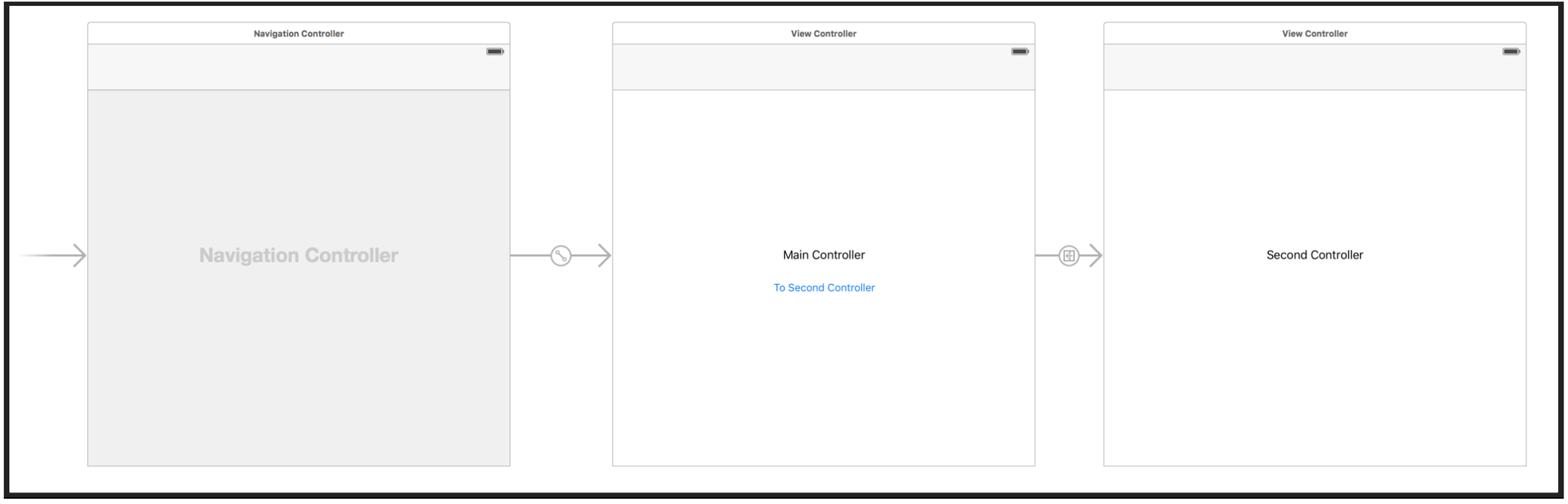
EXERCISE

Define autolayout rules for your project and test them in different simulators (iPhone 5, iPhone 6 plus, iPad...).

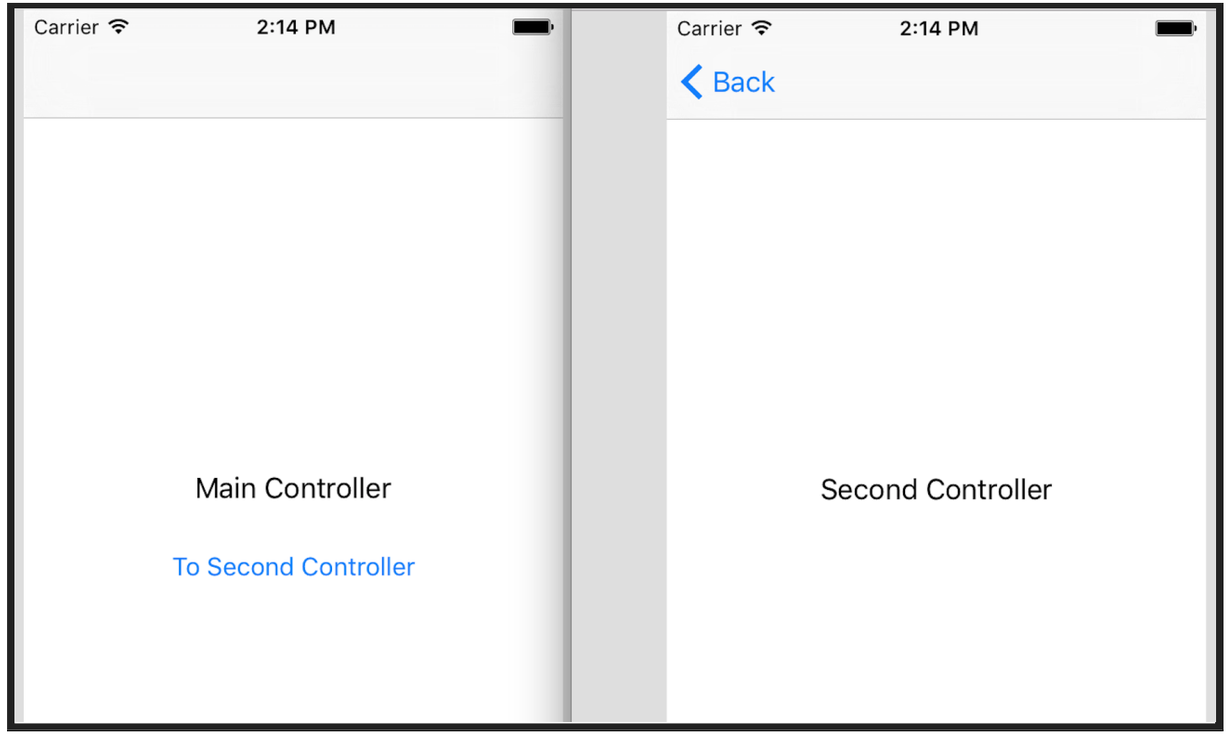
NAVIGATE BETWEEN UIVIEWCONTROLLERS I

- You can define the naviagation in your Storyboard.
- If you want to navigate, the usage of UINavigationController is recommended as you will get an automatic "back" button

STORYBOARD DEFINITION



RUNTIME: NOTE THE BACK BUTTON



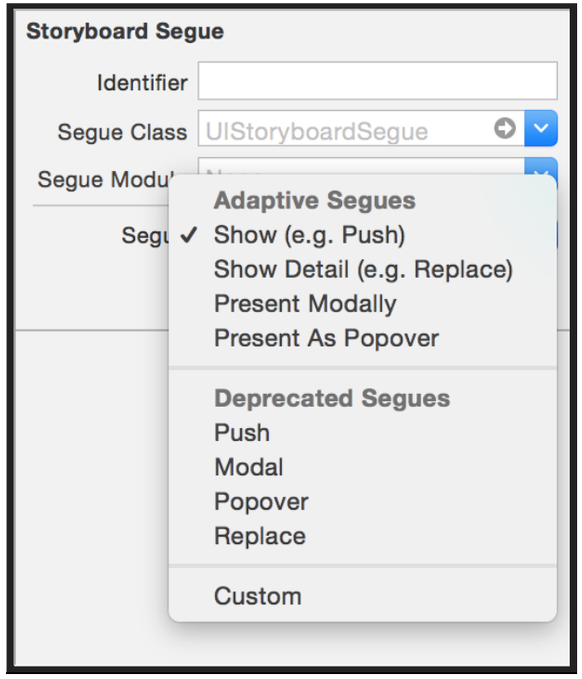
NAVIGATE BETWEEN UIVIEWCONTROLLERS II

- You can also navigate to a different UIViewController programatically.
- When navigating between controllers, you use a so called segue.
- A segue is an object that can store data that you want to transfer between controllers.

SEGUE ACTIONS

There are different ways to show the new UIViewController:

- Show: Standard, use this one by default.
- Show Detail: For UISplitViewControllers.
- Present Modally: Use it when the controller should be finished or canceled.
- Popover presentation: Will present the new UIViewController as a popover on an iPad.
- Custom: Define your own style.



PREPAREFORSEGUE

```
//send data when using the segue by overriding the
//prepareForSegue method in your source controller
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        //this method is called for all segues. We need to check
        //the identifier to get the right one
        if (segue.identifier == "detailSegue") {
                //now we know our destination controller and can upcast
                let destinationController =
                        segue.destinationViewController as! DetailViewController;

                //set whatever you like here
                destinationController.value = "Hello World";
        }
}
```

DISMISS CONTROLLER

```
//you can close an open view controller by calling
//dismissViewController:

 self.dismissViewControllerAnimated(true, completion: nil);

//your app will then show the previous one
```

DEBUGGER

- Xcode provides a simple but powerful debugger.
- You set a breakpoint on a line by clicking on the left border.
- When the program halts, you can examine the variables as well as step through your code.

EXERCISE

Test the debugger by using your project and setting breakpoints.