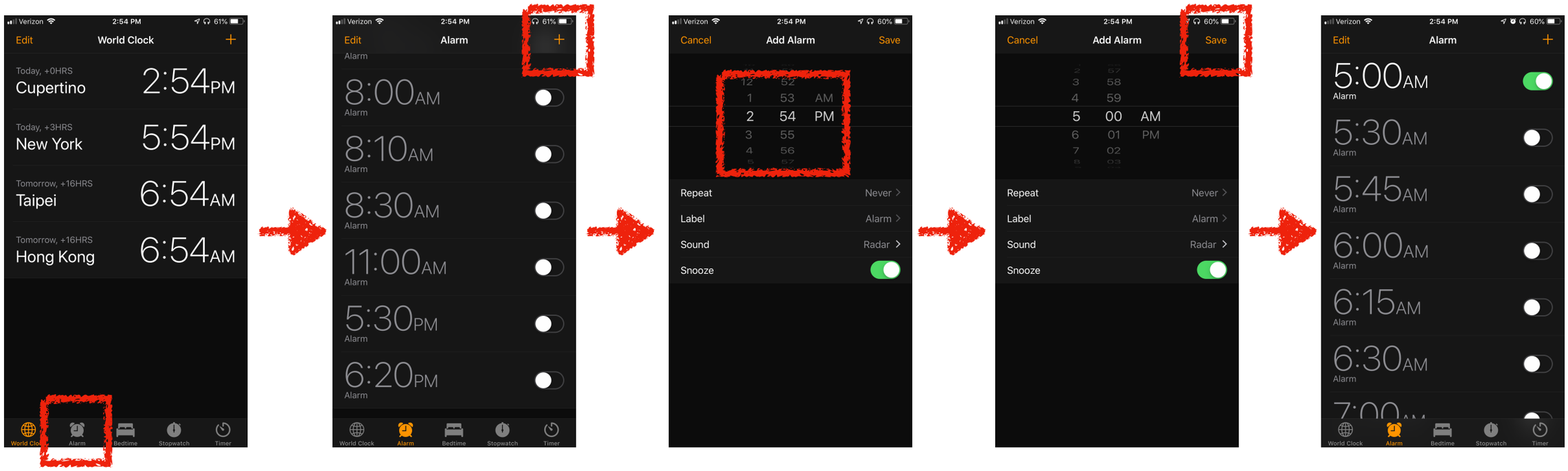
UI Testing App Workflows

Dave Shu
Software Engineer @ ECHO

November 14, 2018 REVB

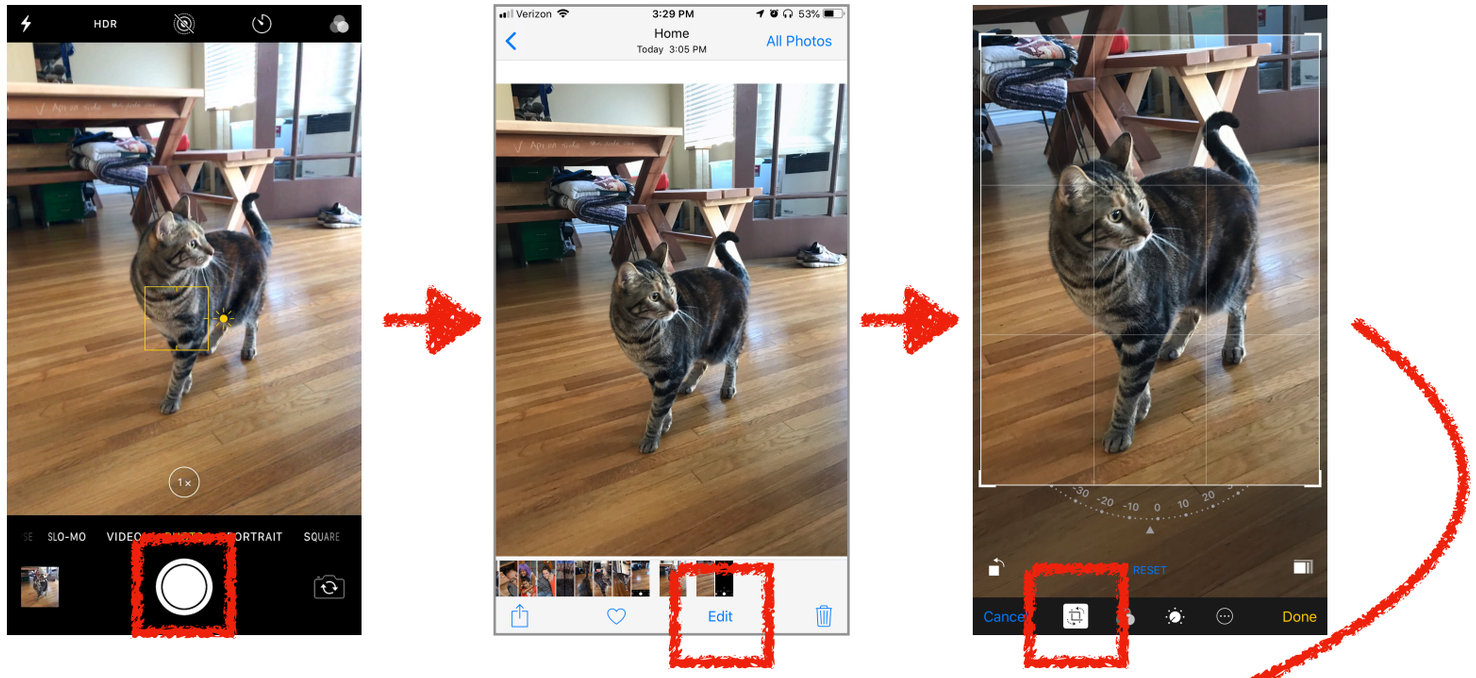
What’s a workflow?



Create an alarm:

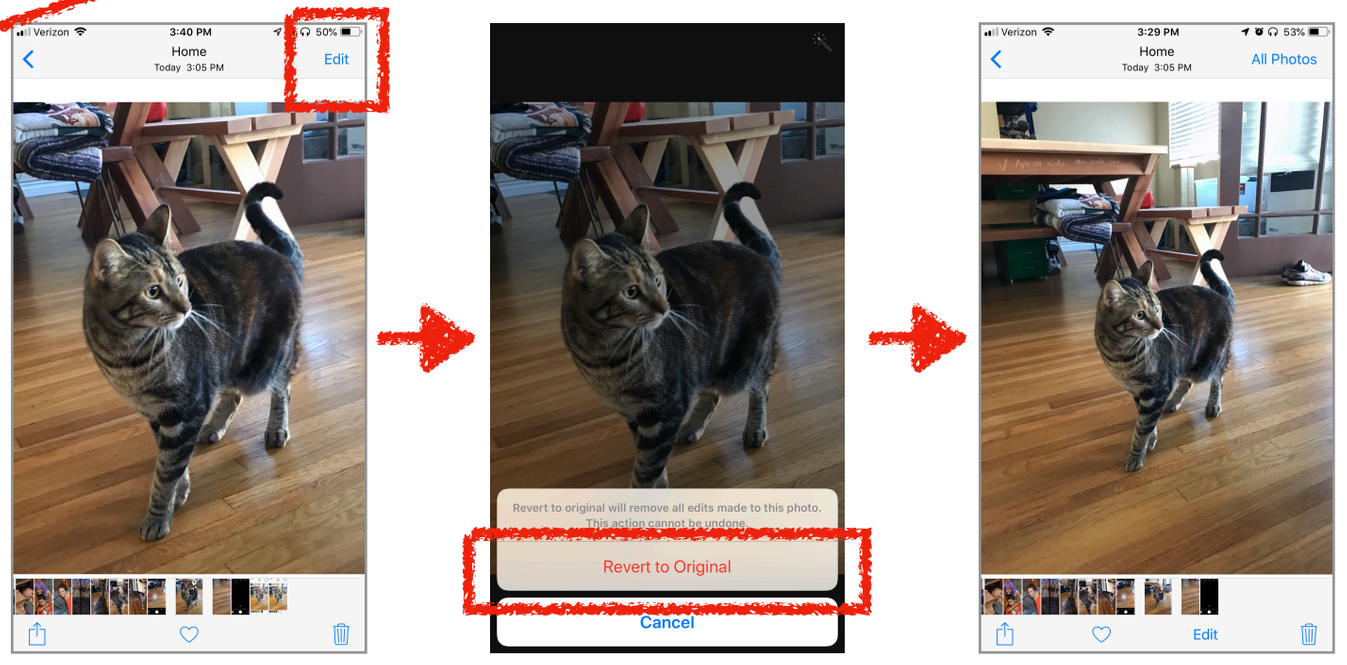
1. Tap Alarm
2. Tap Add
3. Adjust time
4. Tap Save

A more complicated example...



Revert an image

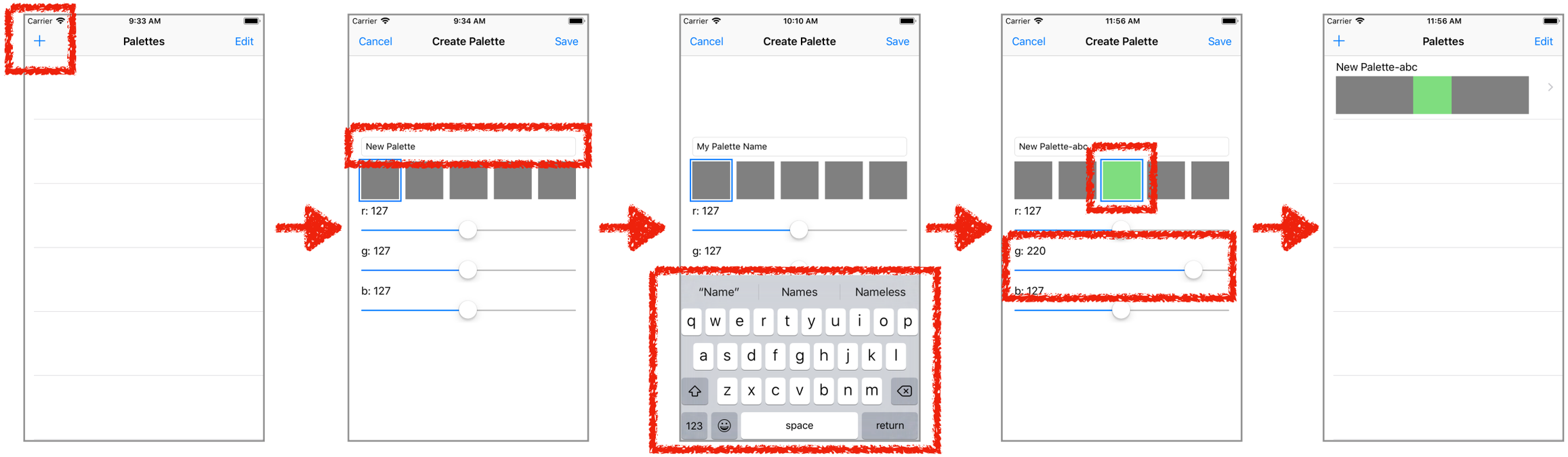
1. Take picture
2. Edit
3. Crop
4. Edit
5. Revert



Example app:
Palette Creator

- Create 5-color palettes
- View my palettes
- View a palette in detail
- Edit a palette

Create Palette Workflow



Create palette:

1. Tap Add
2. Tap text field
3. Type name
4. Tap color button #2
5. Adjust slider
6. Tap Save
7. Verify palette cell

Demo

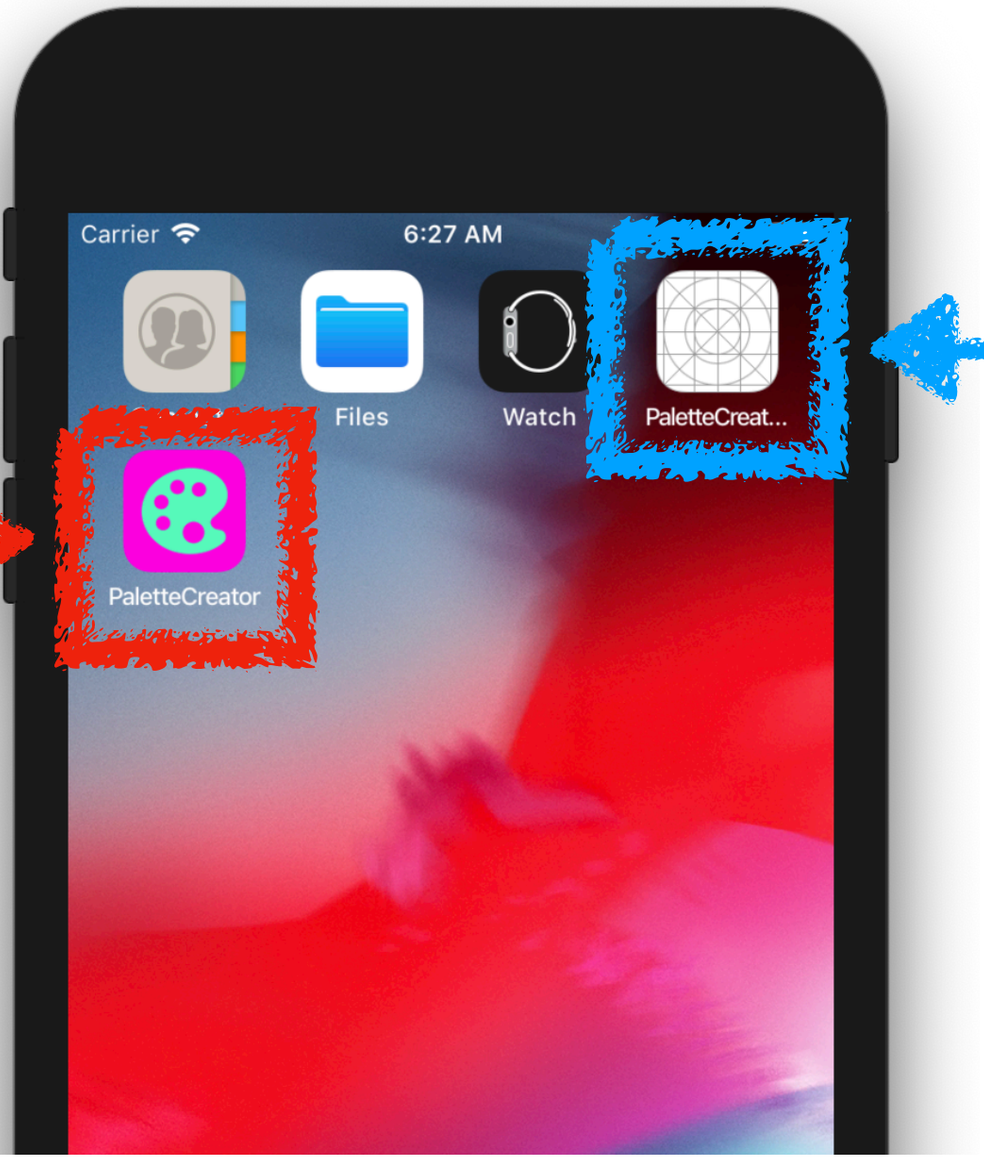
What did you see?

1. UI Test runner app
2. Launching the tested app
3. Interacting with UI (tapping, adjusting sliders, text field entry)
4. Not visible: test is verifying UI

Concepts

1. UI Test runner app
2. XCUIApplication
3. XCUIElement
4. XCAssert

UI Test runner app



Tested app

Test code

“Sees” the tested app like a real user
=>
No access to tested app file system or internal components

XCUIApplication, Part 1

- Represents your tested app

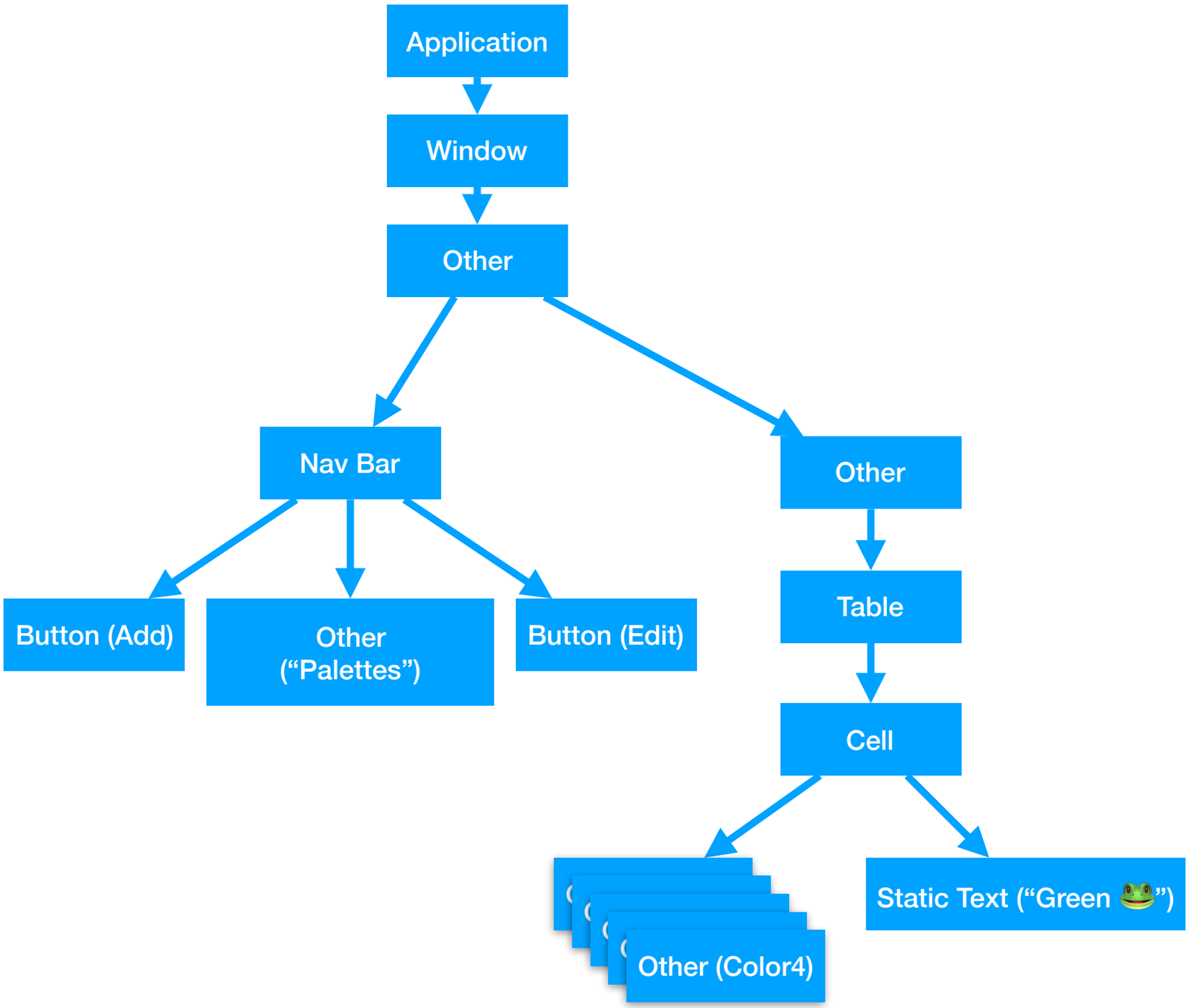
Instantiate	`let app = XCUIApplication()`
Launch	`app.launch()`
Terminate	`app.terminate()`

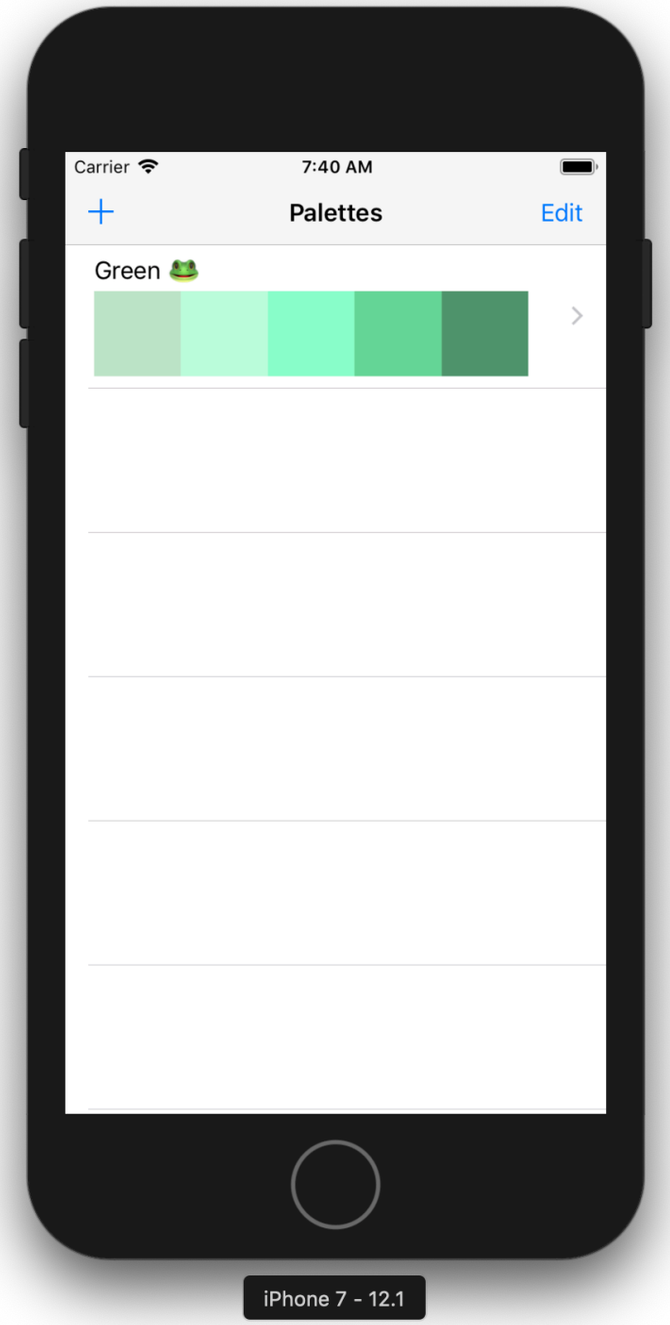
XCUIElement

- Represents a UI element in your tested app

Check if it exists	`app.otherElements["color-view-1"].exists`
Wait for existence	`app.navigationBars[“edit-palette-bar”].waitForExistence(timeout: 1)`
Tap	`app.buttons["color-button-1"].tap()`
Adjust slider position	`app.sliders["red-slider"].adjust(toNormalizedSliderPosition: 0.5)`
Type text	`app.textFields["palette-entry-name-text-field"].typeText("MyName")`
Read labels	`app.staticTexts["red-label"].label`
Query descendants	`app.tables["palette-table"].cells`

Element Hierarchy





XCUIApplication, Part 2

- Is itself an XCUIElement => Your starting point for finding UI elements

Instantiate	`let app = XCUIApplication()`
Launch	`app.launch()`
Terminate	`app.terminate()`
Query descendants	`app.buttons["color-button-1"]`

How do you name UI elements?

- UIAccessibility
  - Assistive technology

- Best practice:
  - Set **accessibilityIdentifier** on UI elements
- Example:
  - `addButton.accessibilityIdentifier = "palette-table-add-button"`

XCAssert

- Verify your expectations

- Also used for Unit Tests

- Examples:
  - `XCTAssert(app.buttons["color-button-1"].exists)`
  - `XCTAssertEqual(app.staticTexts["red-label"].label, "r: 155")`

Sliders

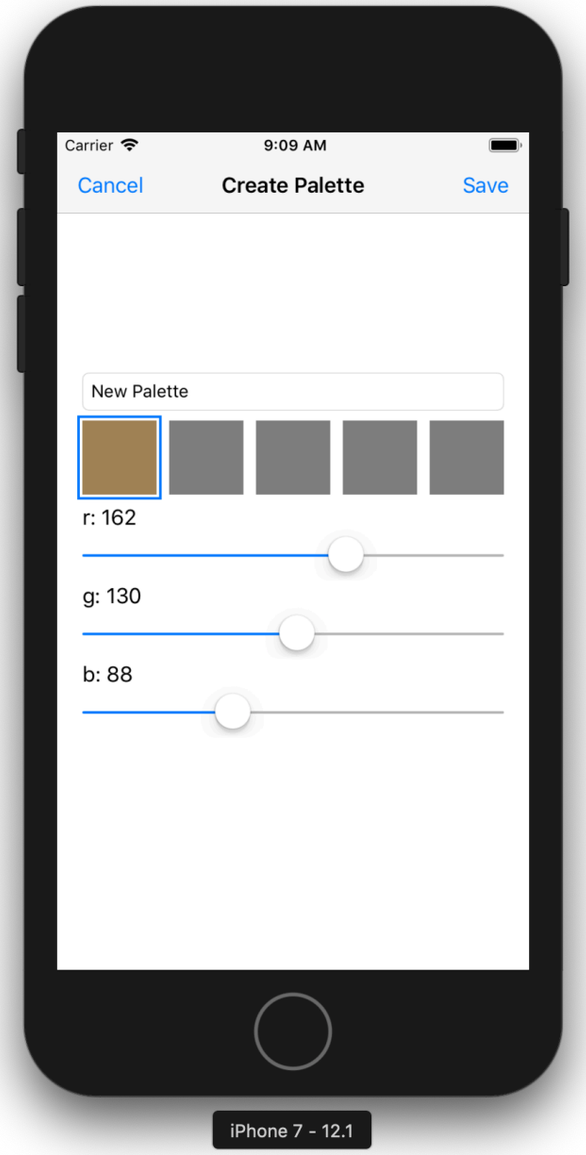
- Adjusting sliders => **Best effort only!**

- What can you do?

1. Adjust slider to value (“original value”)

2. Read the value (“actual value”)

3. Base your expectations on the value you read



Concepts Review

1. **UI Test runner app** - your test code that runs in a separate app

2. **XCUIApplication** - represents your tested app

3. **XCUIElement** - represents a UI element in your tested app

4. **XCAssert** - verify your expectations

Palette Creator app

Follow along!

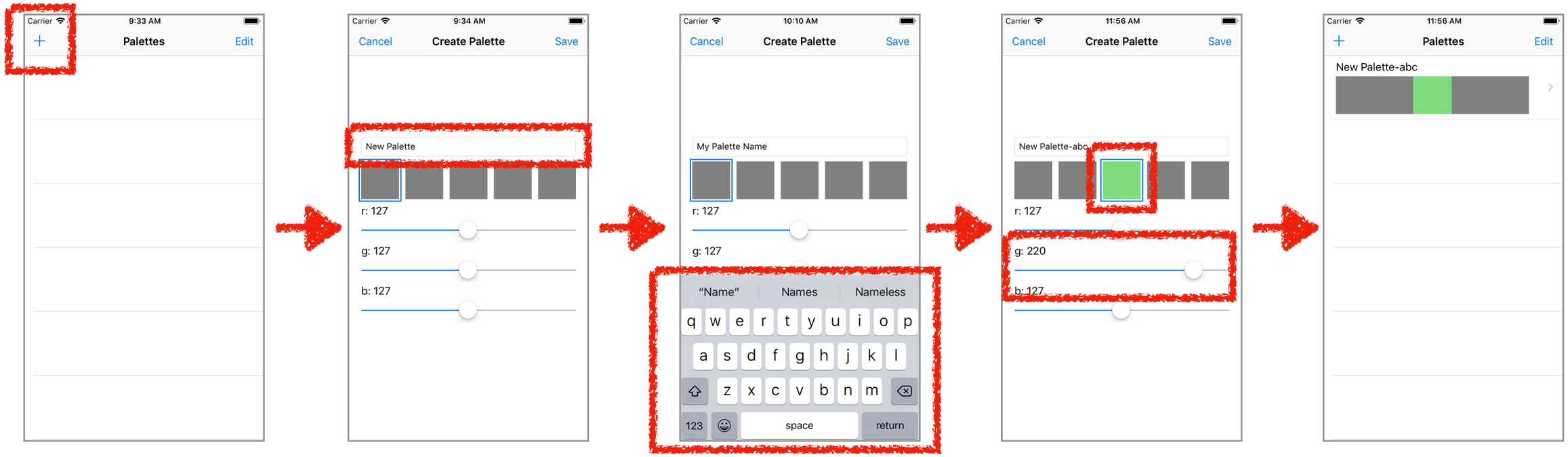
`http://bit.ly/echo-pal-start`

WiFi name: eSUB
Password: 4875esubwireless

Demo: Add UI Tests to PaletteCreator

1. Create a UI Test target
2. UI Test recording
3. Set accessibilityIdentifiers on UI elements
4. Write Create Palette test

Demo: Create Palette Workflow



Create palette:

1. Tap Add
2. Tap text field
3. Type name
4. Tap color button #2
5. Adjust slider
6. Tap Save
7. Verify palette cell

Demo

Review

1. Create a UI Test target
2. Use recording as a starting point
3. Set accessibilityIdentifiers on UI elements
4. Write tests

Questions?