T16

Concurrent Class
10/3/2013 1:30:00 PM

"Automate Mobile App Testing- Or Go Crazy"

Presented by:

Stu Stern
Gorilla Logic

Brought to you by:

340 Corporate Way, Suite 300, Orange Park, FL 32073
888-268-8770 · 904-278-0524 · sqeinfo@sqe.com · www.sqe.com

Stewart Stern

Gorilla Logic, Inc.

Stu Stern is CEO of Gorilla Logic, an enterprise IT consulting firm specializing in web and mobile development, and the creators of several automation tools including FlexMonkey, FoneMonkey, and MonkeyTalk. Prior to founding Gorilla Logic in 2002, Stu was head of Java consulting at Sun Microsystems. Even further back, Stu spent many years building trading systems on Wall Street. Although a long-time senior IT executive, after thirty years in the industry Stu is proud to still be very much a geek

GORILLA
LOGIC

AUTOMATE MOBILE APP TESTING (OR GO INSANE)

STU STERN, CEO, GORILLA LOGIC

WELCOME TO GORILLA LOGIC

WE'RE TWO COMPANIES IN ONE

- Enterprise Services
  - Mobile and Web App Development
  - Front-end and Back-end
  - Project outsourcing and talent acquisition
- Testing Tools
  - MonkeyTalk
  - CloudMonkey
  - Support, training, quick start services
- Founded in 2002 by ex-Sun execs
- 100 employees
- Based in Boulder with offices in New York, Bangalore, and Costa Rica

BROWSERS WERE BAD (FF, IE, SAFARI, …)

BUT MOBILE IS WORSE!

- Android 2.x, 3.x, 4.x
- iOS 4.x, 5.x, 6.x
- Huge range of physical screen resolutions and pixel densities
- Runtime differences among various device brands and models

AN INSANE NUMBER OF TESTING TARGETS!

1. Functional – Does the app work according to spec?
   - Verify on virtual device of each platform (2)
2. OS compatibility – Does the app work on each OS version
   - Verify on virtual device of each version (6 - 12)
3. Display compatibility
   - Verify on virtual device at each resolution (6 - 12)

- Device compatibility
  - Run on each major make and model (dozens – hundreds)

Literally hundreds of possible configurations!

AUTOMATION MIGHT HELP ;)

- MonkeyTalk is free and open source
- Powerful script recording
  - But scripts can also be created or edited with any text editor
- Simple script commands
- Provides complete automation solution
  - Android 2.2+, iOS 3.x+, Mobile Web
  - Full-featured Integrated Development Environment
  - Interactive script creation and execution
  - Command line script runner
  - Continuous Integration friendly

AGENDA
(ASPIRATIONAL)

- MonkeyTalk Platform Overview
- Recording Tests
- Understanding MonkeyTalk Commands
- Running Tests
- Reusing Parameterized Scripts
- Data-Driving
- Test Suites
- Working with Variables
- JavaScript Scripting

MONKEY EVOLUTION

OUT OF MANY, ONE

- Gorilla Logic Open Source Projects
  - 2009 – FlexMonkey/FlexMonkium for Adobe Flex
  - 2010 – FoneMonkey for iOS
  - 2011 – FoneMonkey for Android
- 2012 – MonkeyTalk for iOS, Android, and Mobile Web
  - Unites FoneMonkey for iOS and Android and adds Mobile Web support
  - Affero GNU Open Source License
  - Downloaded 40,000+ times since first released in March, 2012
- 2013 – CloudMonkey
  - Mobile device testing farm

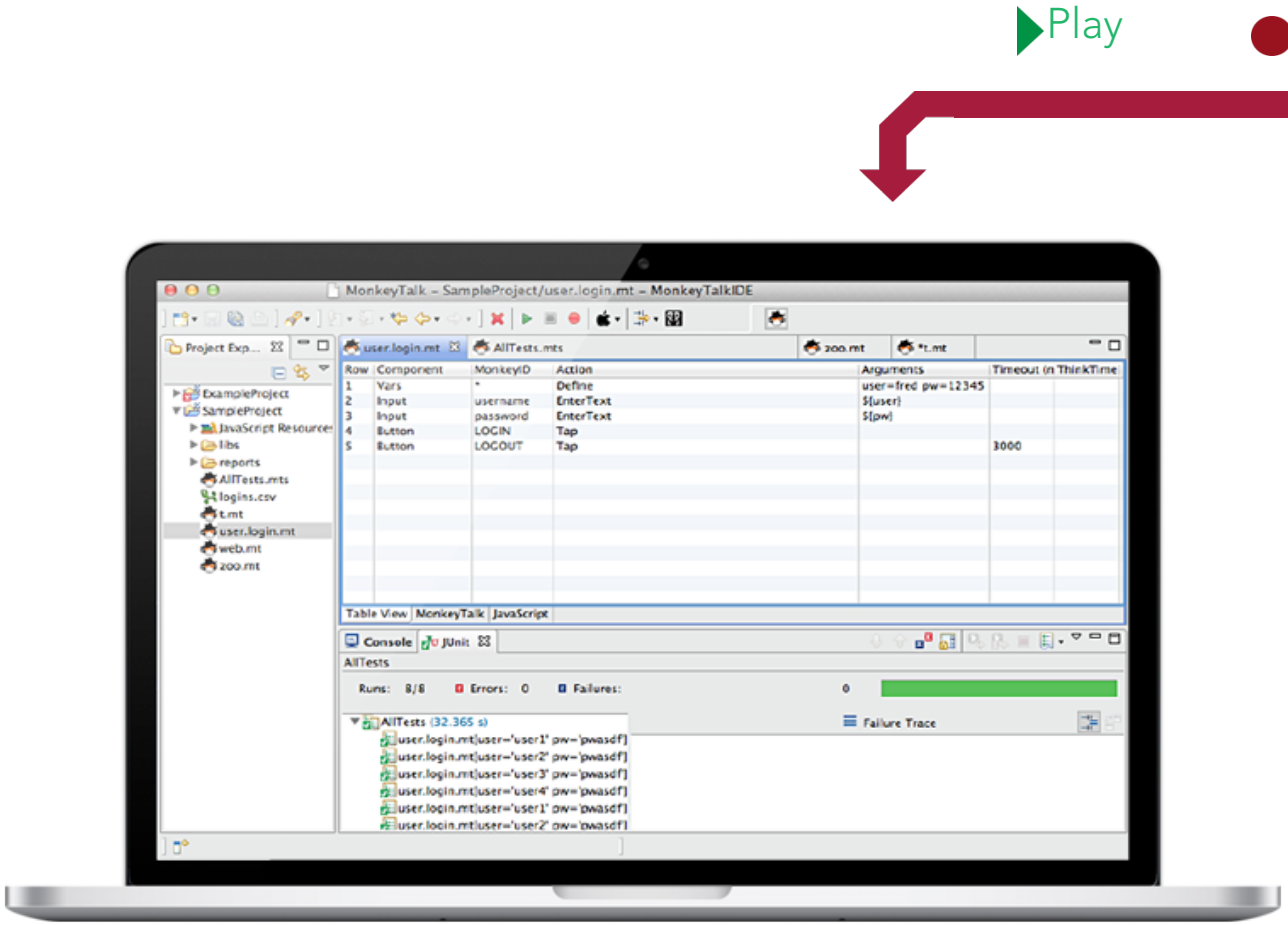
MONKEYTALK 1.0 DESIGN PRECEPTS

POWERFULLY SIMPLE

- High-level operation and gesture recording/scripting
  - Low-level events are too numerous for maintainable scripts
- Low-punctuation scripting
  - MonkeyTalk uses a simple, readable, space-separated command language
- Easy script parameterization and data-driving
  - Built into MonkeyTalk Command Processing
- Cross-Platform
  - iOS, Android, Mobile Web
- "Natural" JavaScript Scripting
  - Generated JavaScript API wraps scripts and commands

CROSS-PLATFORM FUNCTIONAL TESTING

RECORD ON ONE, PLAY BACK ON ANOTHER



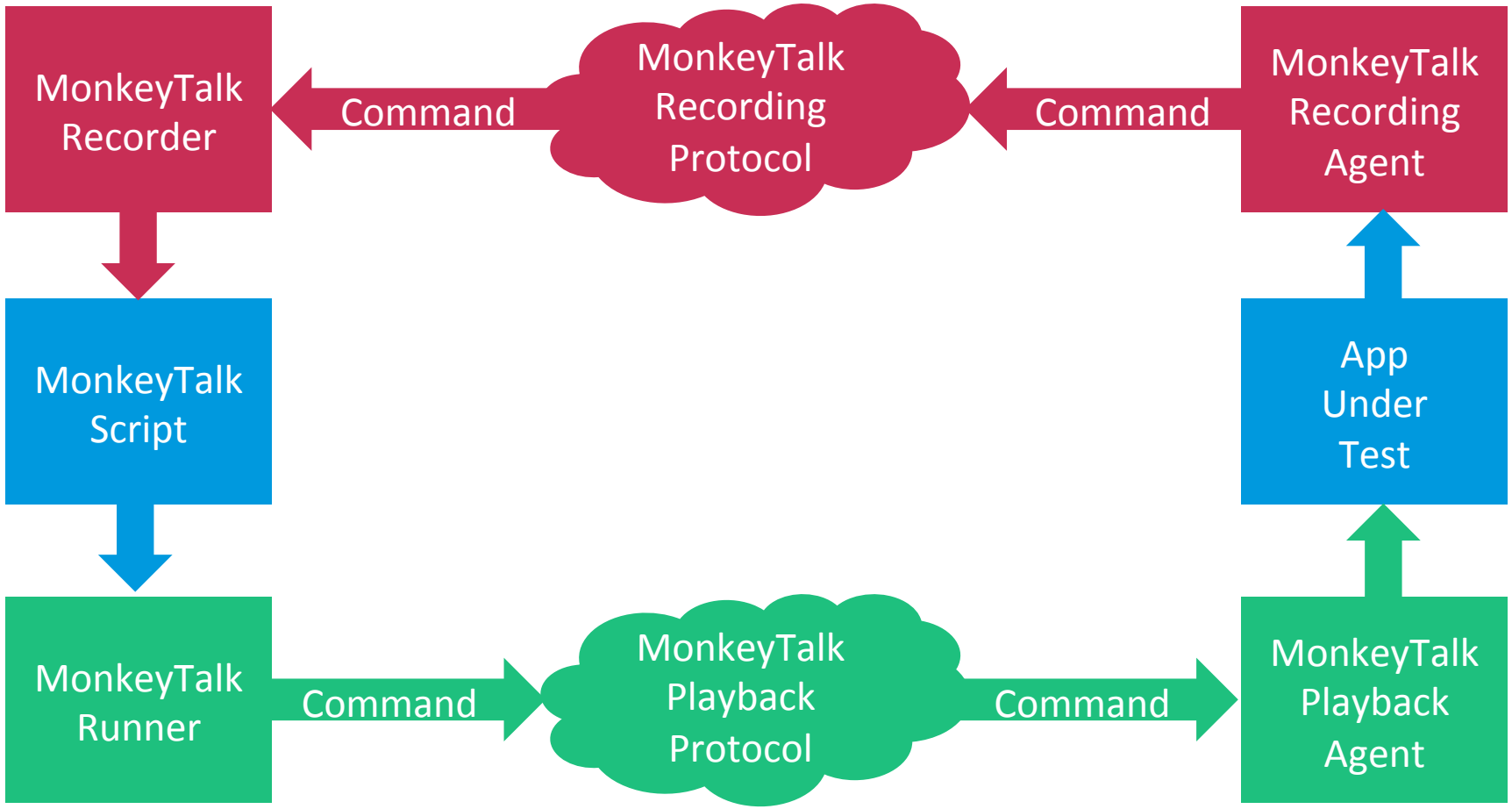
OBJECT- VS IMAGE-BASED TESTING TOOLS

MONKEYTALK IS OBJECT-BASED

	Object	Image
Mechanism	Analyze actual user interface components	Analyze bitmap screen images
Setup	Link agent into app	Run agent on host
Ability to script “outside” app	No	Yes
Events Recorded	Higher Level	Lower Level
Script Readability	More Readable	Less Readable
Brittleness	Less Brittle	More Brittle
Jail-Breaking or physical device harness	Not Required	Required

MONKEYTALK OPEN ARCHITECTURE

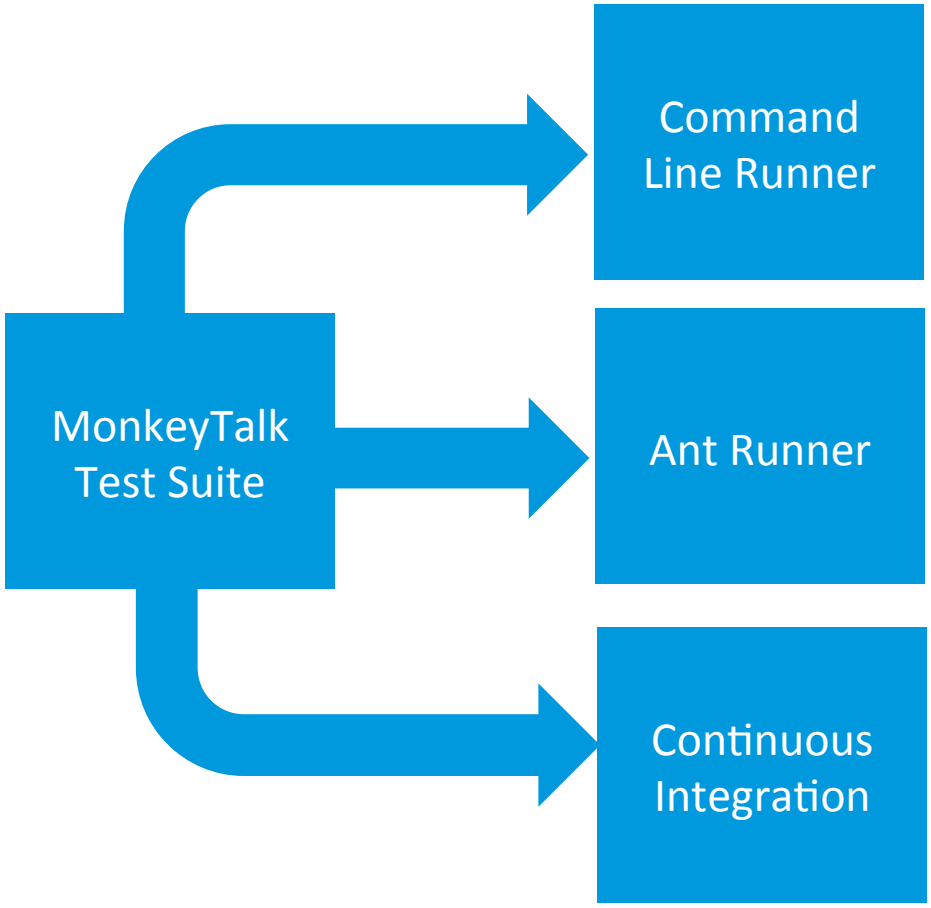
SWAPPABLE COMPONENT FRAMEWORK



AUTOMATING AUTOMATED TESTING

RUN INTERACTIVELY OR UNATTENDED

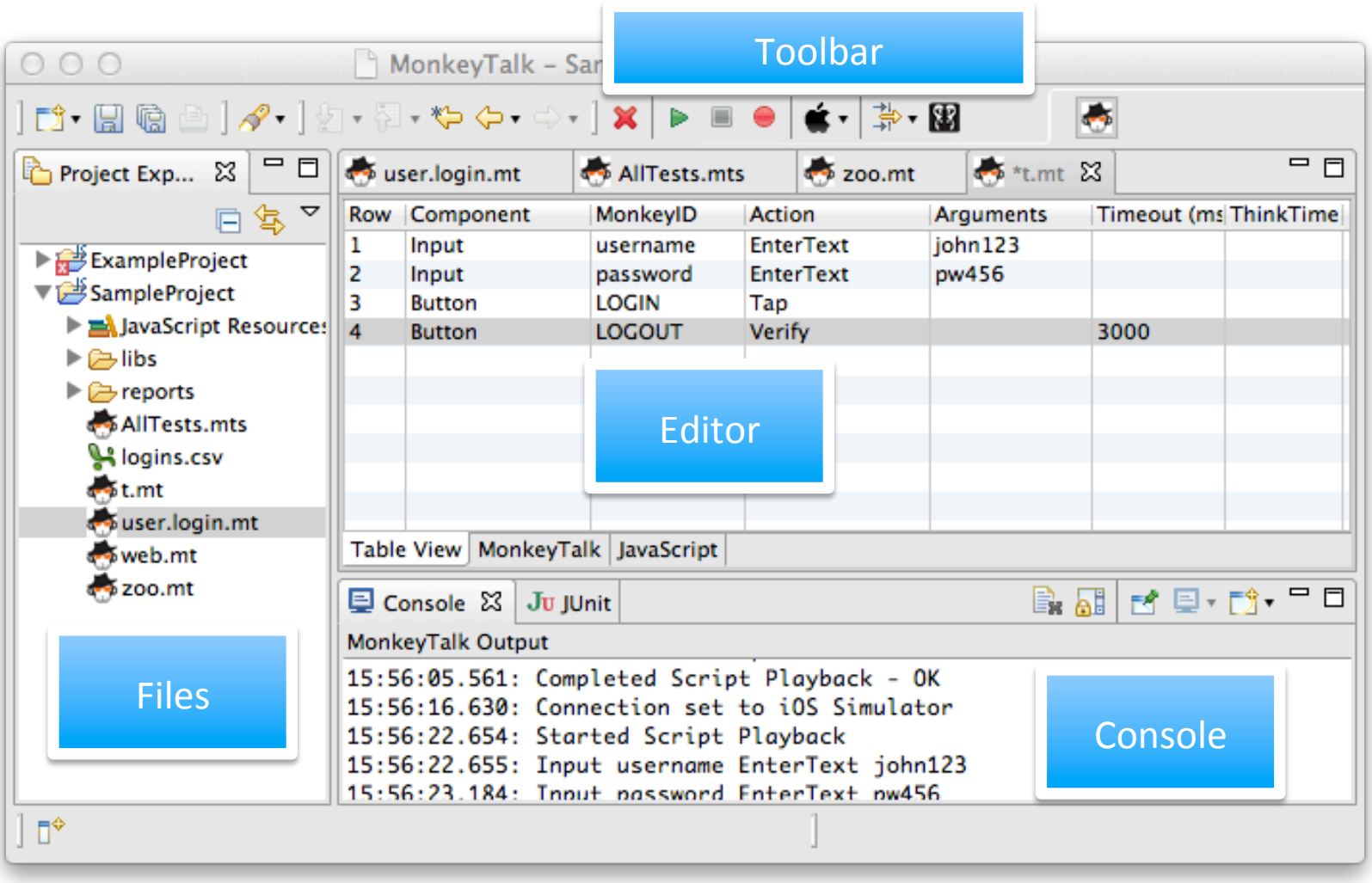
- Various methods for running unattended MonkeyTalk tests



- Scheduled
- Build-Time
- Commit Time

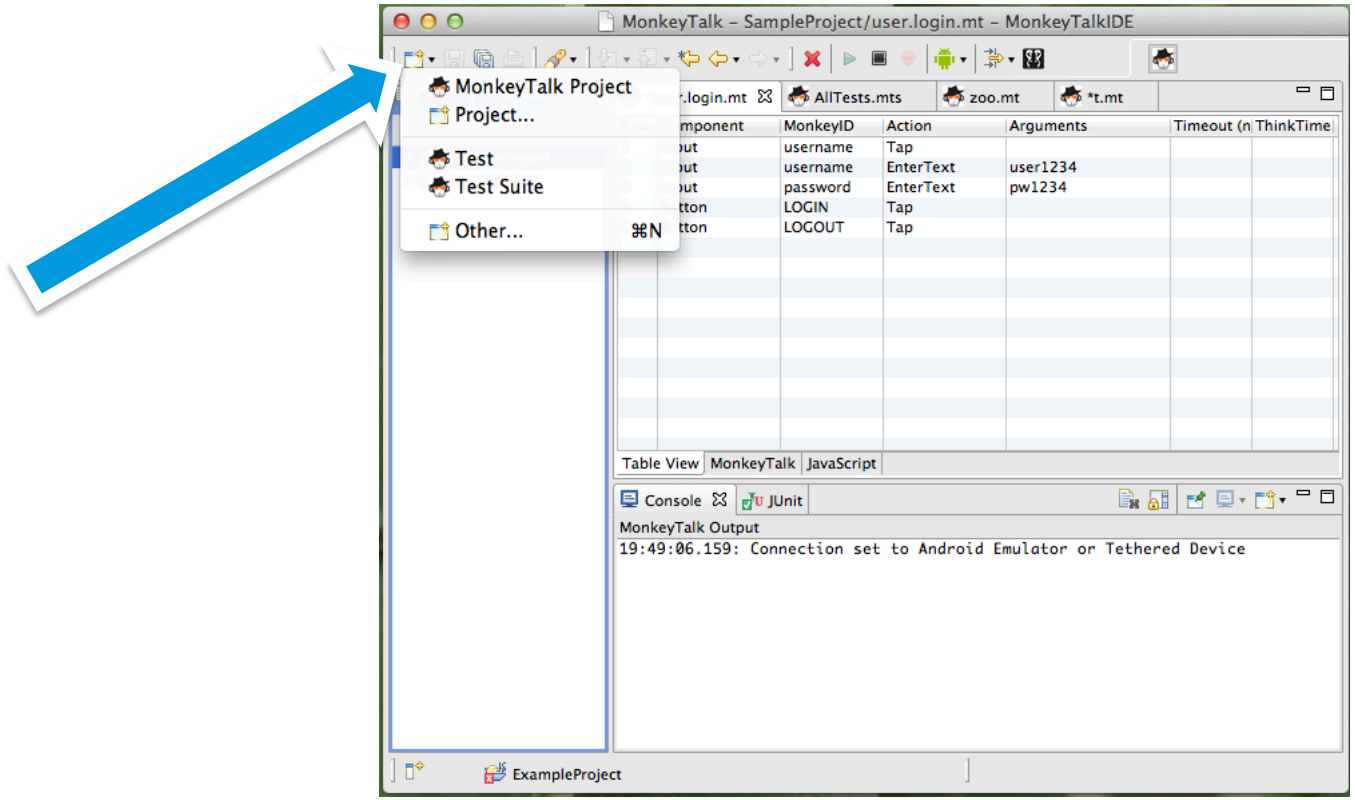
MONKEYTALK IDE

FUNCTIONAL TEST SUITE CREATION AND MANAGEMENT WORKBENCH



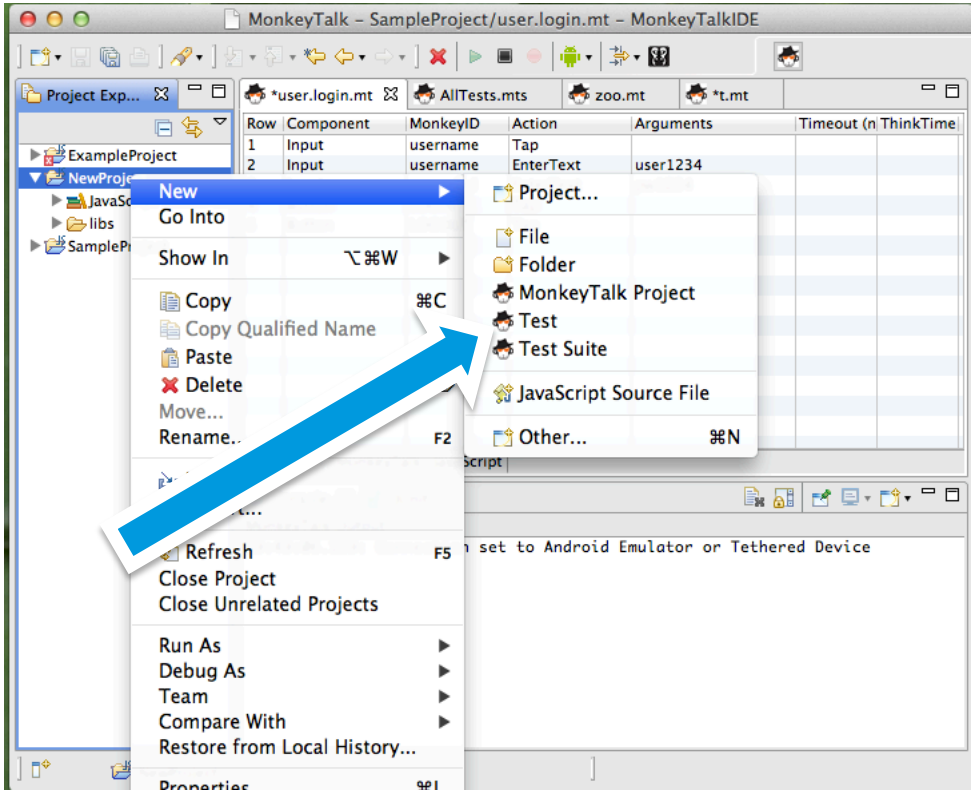
CREATE A MONKEYTALK PROJECT FOLDER

CONTAINS ALL YOUR SUITES, SCRIPTS, AND TEST DATA



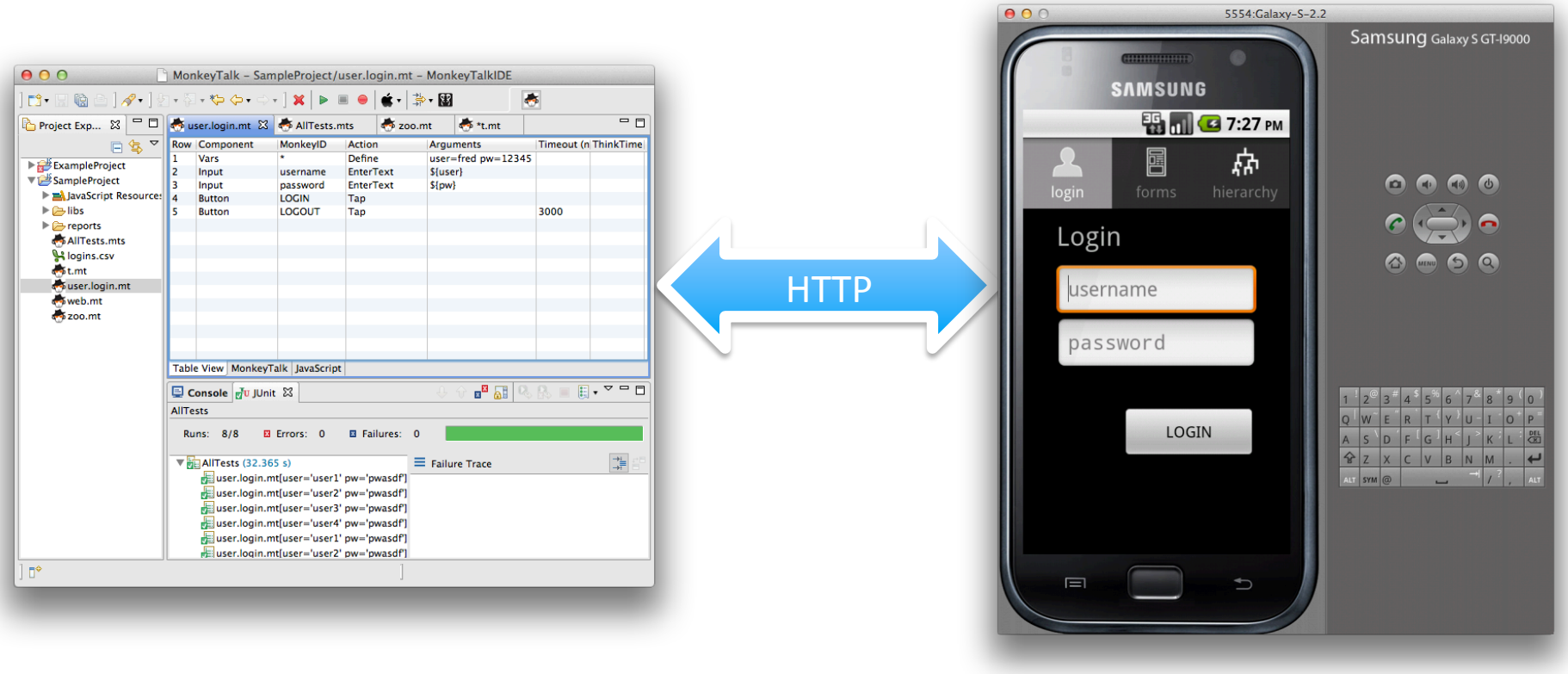
CREATE A NEW, EMPTY TEST SCRIPT

RIGHT-CLICK ON PROJECT



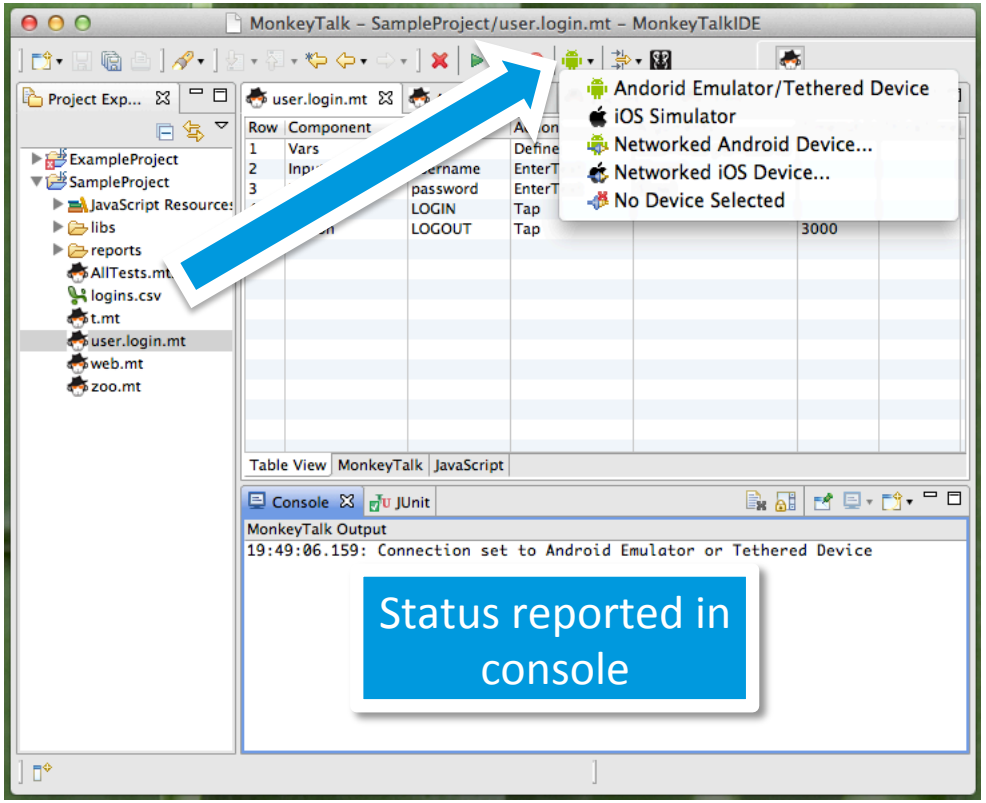
IDE AND APP COMMUNICATE OVER HTTP

VIA USB TETHER OR WI-FI CONNECTION

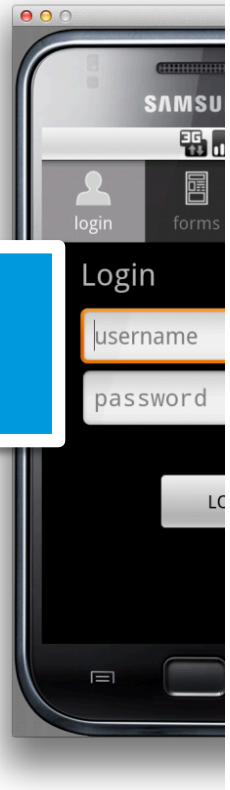


CONNECTION SELECTOR

SELECTS DEVICE AND CONNECTION TYPE

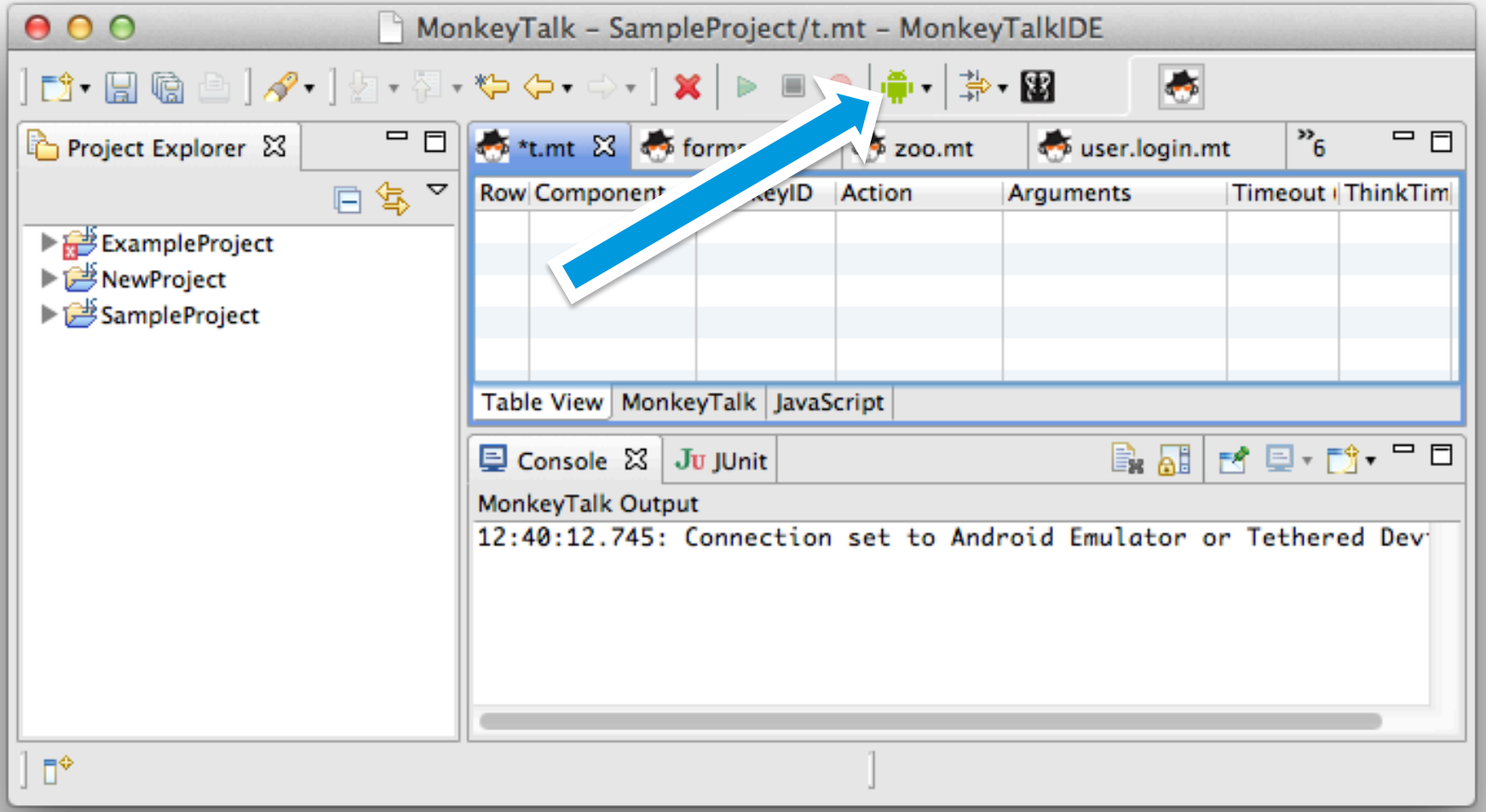


App must already be running



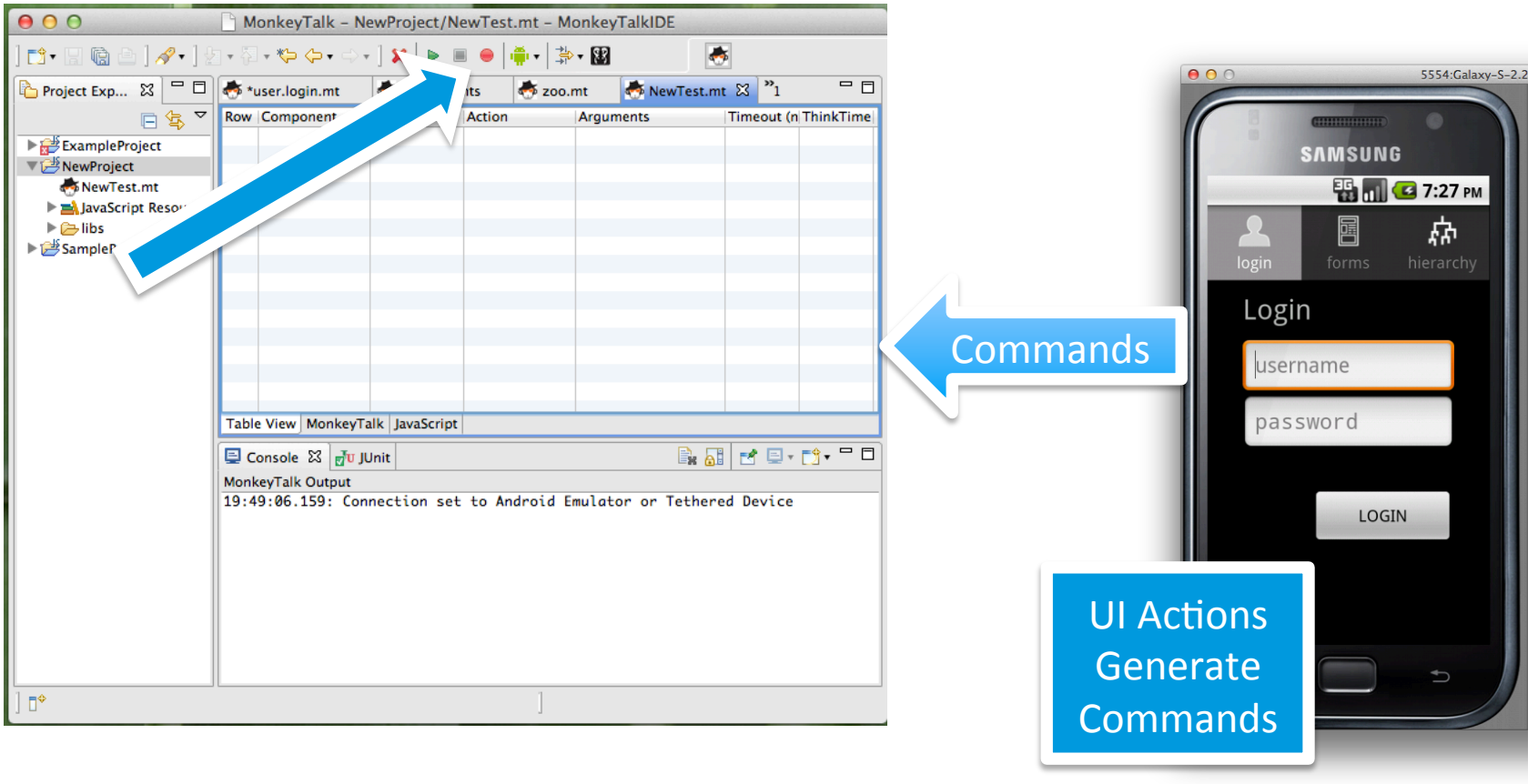
CONNECTION BUTTON

INITIATES CONNECTION TO APP UNDER TEST

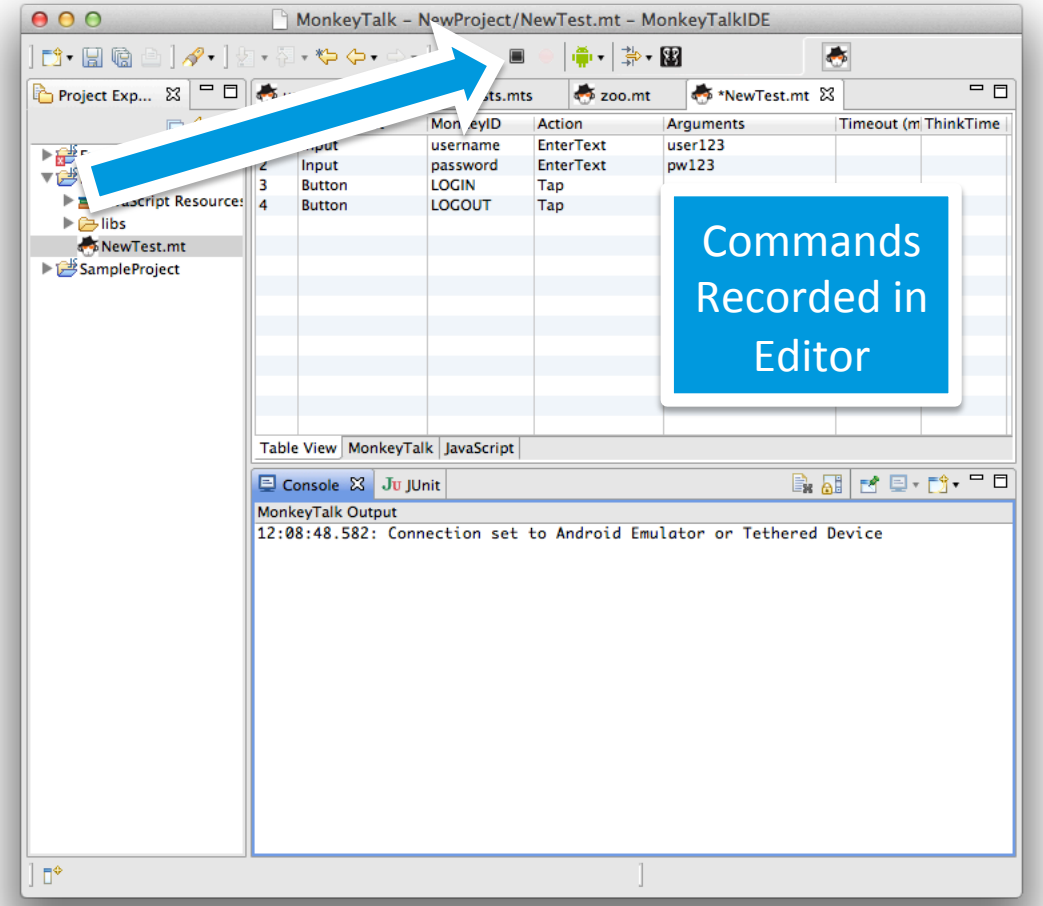


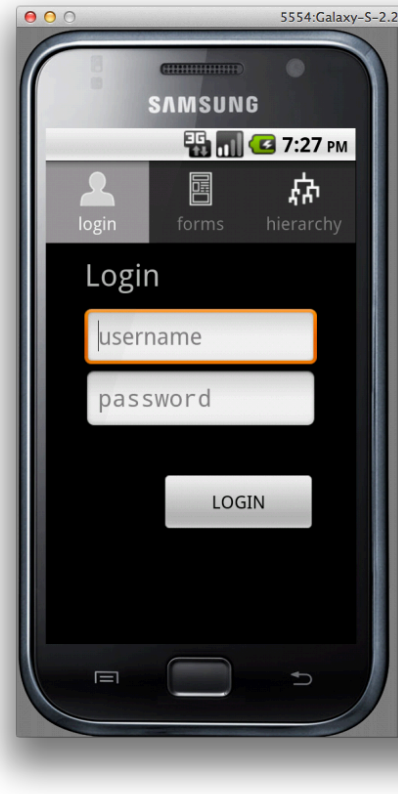
RECORD BUTTON

STARTS CAPTURING OF COMMANDS FROM APP UNDER TEST



STOP BUTTON
STOPS RECORDING





SIMPLE MONKEYTALK COMMAND FORMAT

EVERY COMMAND CONSISTS OF SAME COLUMNS

```
ComponentType MonkeyId Action Arguments…
```

Row	Component	MonkeyID	Action	Arguments	Timeout (m	ThinkTime
1	Input	username	EnterText	user123		
2	Input	password	EnterText	pw123		
3	Button	LOGIN	Tap			
4	Button	LOGOUT	Tap			

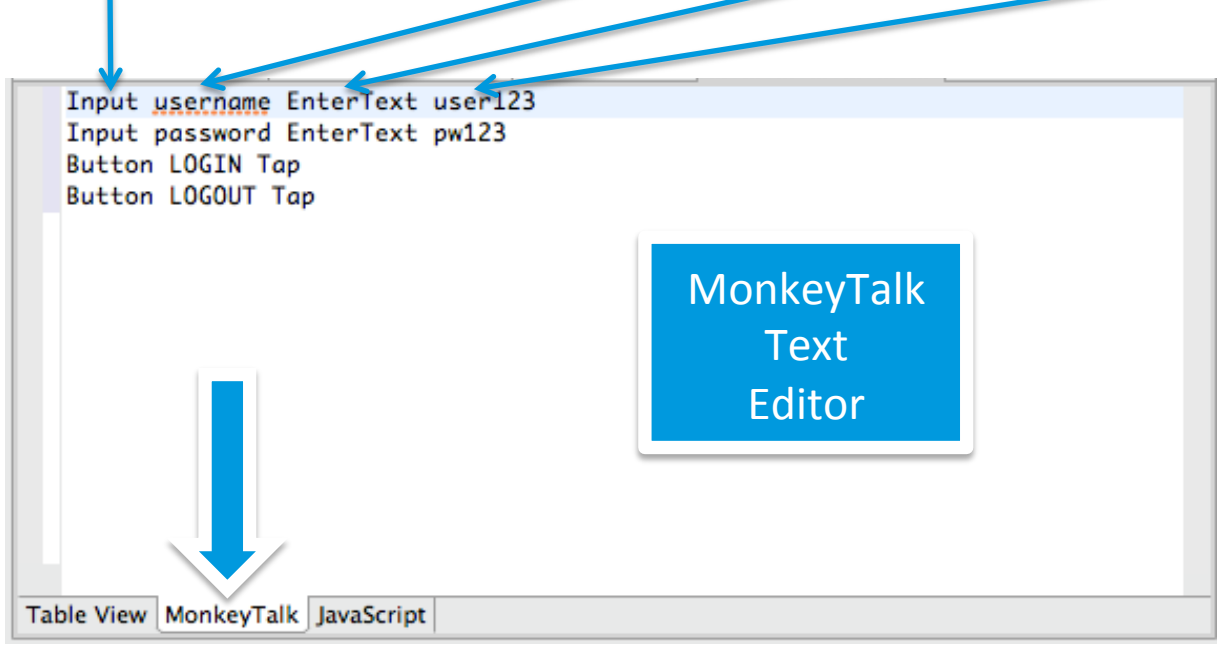
Table View | MonkeyTalk | JavaScript

Table View Editor

CHOOSE YOUR EDITOR

EDIT WITH TABULAR OR TEXT EDITOR

```
# Comments begin with a hash-sign
ComponentType MonkeyId Action Arguments…
```

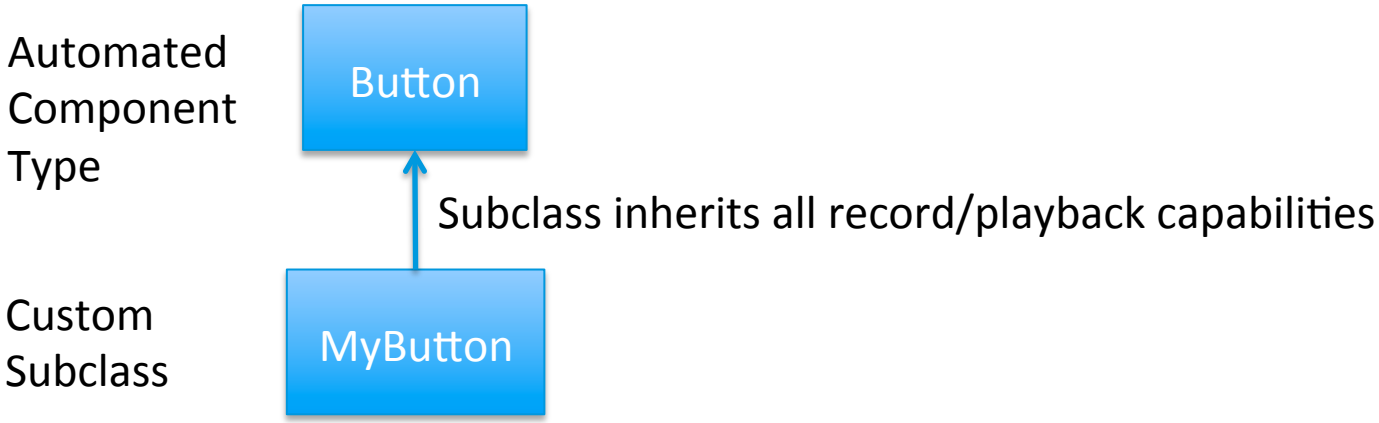


MONKEYTALK IS OBJECT-ORIENTED

COMPONENT TYPE HIERARCHY AND ACTION INHERITANCE

```
ComponentType MonkeyId Action Arguments…
```

Class Instance Method



LOGICAL COMPONENT TYPES

MAPPED TO EACH SDK FOR CROSS-PLATFORM PLAYBACK

MonkeyTalk	iOS	Android	Web
View	UIView	View	Any Tag
Input	UITextField	EditText	Input, type(text)
Button	UIButton	Button	Button, Input(submit)
ItemSelector	UIPicker	Spinner	Select
CheckBox	UISwitch	CheckBox	Input(checkbox)
Table	UITableView	ListView	Table
TabBar	UITabBar	TabHost	
ButtonSelector	UISegmentedControl	RadioGroup	Input(radiobutton)
TextArea	UITextView	EditText	TextArea
Menu	TabBar	Menu	
Label	UILabel	TextView	Label, Div
Slider	UISlider	SeekBar	
HtmlTag			Any Tag

Component types form an inheritance hierarchy

MONKEY ID'S
DISTINGUISH BETWEEN COMPONENTS OF THE SAME TYPE

- MonkeyTalk identifies a component by its textual label, or other identifying property value
  - For example, the label value on a button

- MonkeyId's can be set explicitly by a developer via a component's accessibility label
  - iOS: UIView.accessibilityLabel
  - Android: View.contentDescription
    - Android components can also be identified by android:id
  - Web: id, name, title, or styleClass

COMPONENT TYPE + MONKEYID

COMBINATION IDENTIFIES UNIQUE COMPONENT

```
ComponentType MonkeyId Action Arguments…
```

```
# Tap Some Buttons
Button OK Tap
Button Cancel Tap
# Select the row with value “Belgium”
Table Countries Select Belgium
```

- At playback, monkeyId must match some component’s text label, or the accessibility label, or other componentType-specific id prop
- If multiple components match, the upper-left-most one will be returned
- If no component matches, the command will fail
- MonkeyId’s with embedded blanks are enclosed in quotes

```
Button “Save All” Tap
```

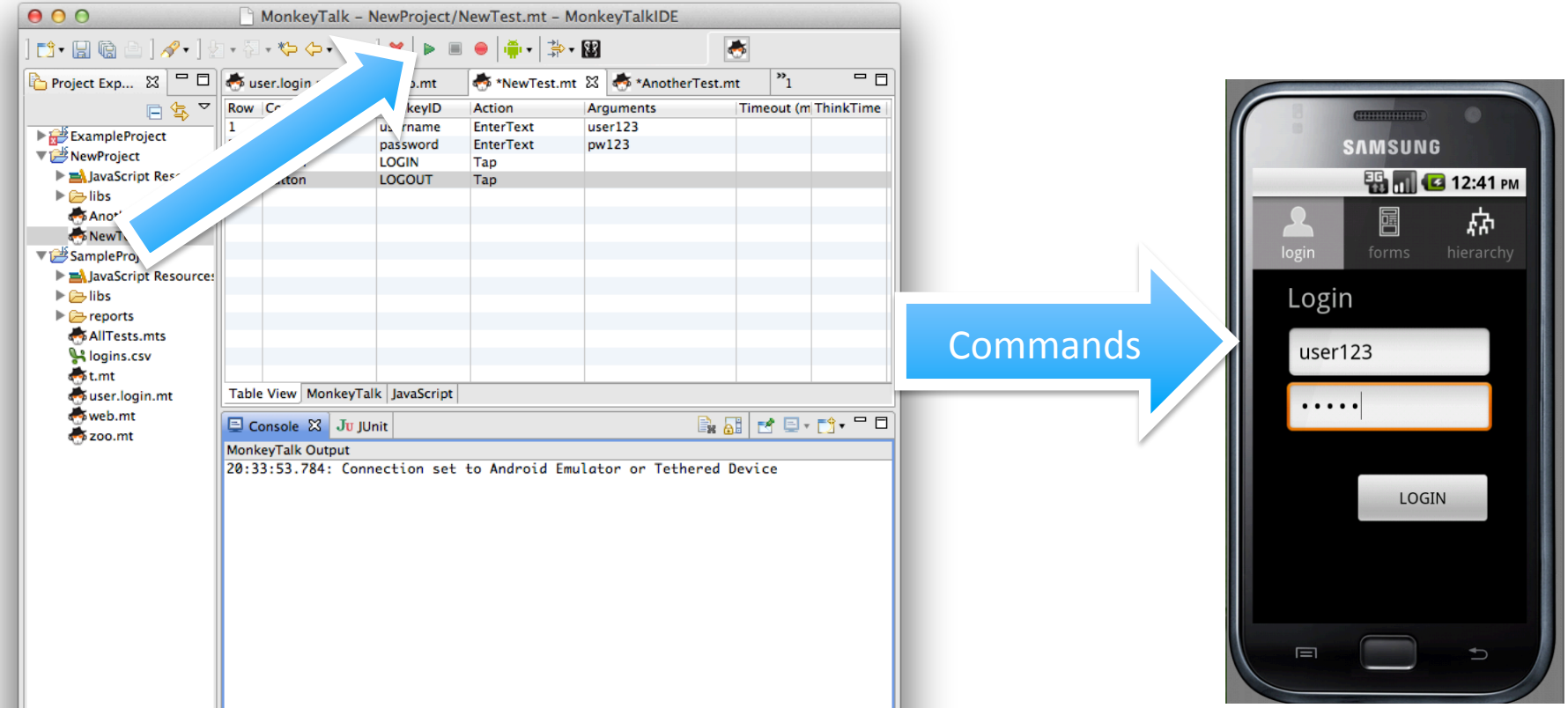
INDEXED MONKEYID'S
IDENTIFY COMPONENTS BY POSITION

- Indexes are specified as #N, beginning with 1 (not zero), eg, #1, #2, #3…
- #1 is the upper-leftmost component of the specified type
- A MonkeyId of asterisk (*) matches the first component of the specified type (same as #1)
- You can also index the monkeyID value with an index value in parentheses

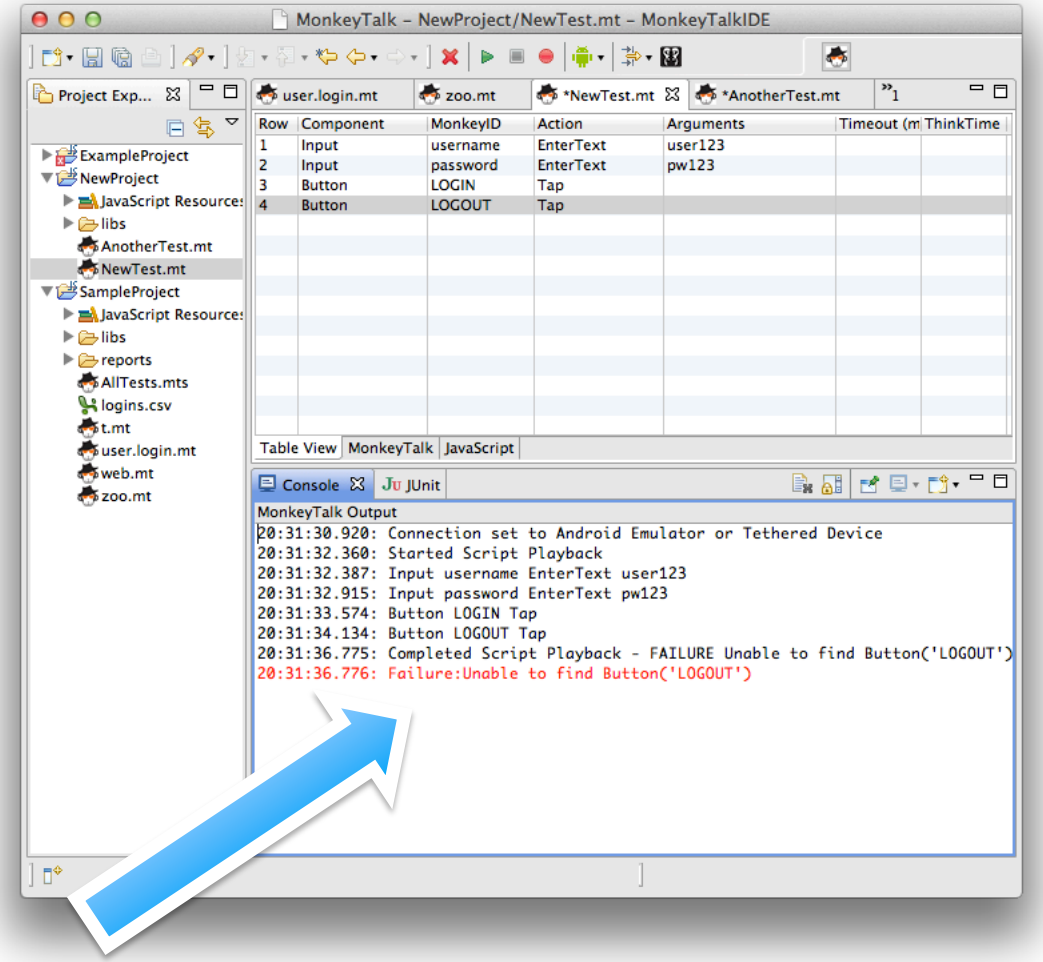
```
# First Input Field
Input #1 EnterText Fred
Input #2 EnterText Mertz
# Second Button
Button #2 Tap
# Only one TabBar
TabBar * Select Places
Table * Select Madagascar
# Tap the second OK button
Button OK(2) Tap
```

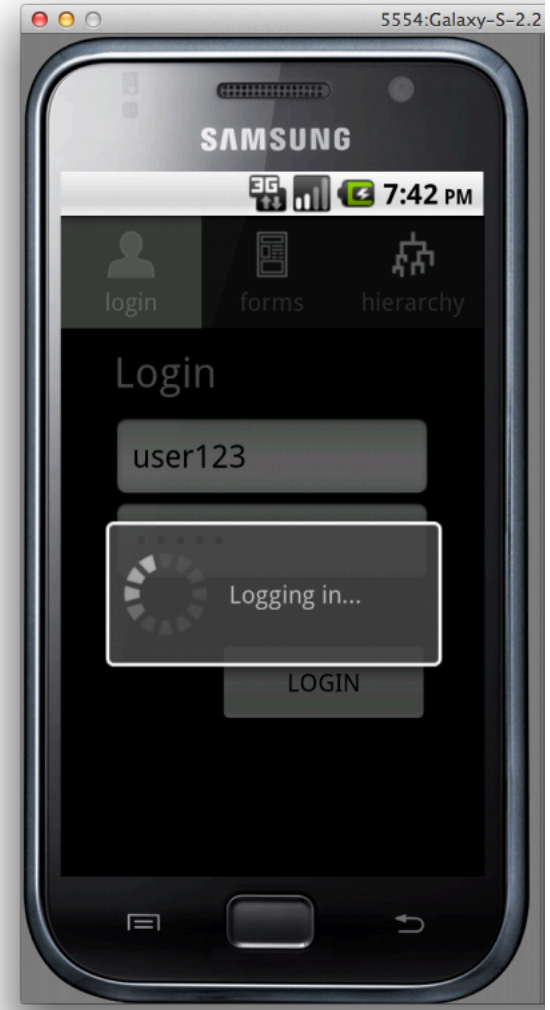
PLAYBACK

IDE SENDS COMMANDS TO APP UNDER TEST



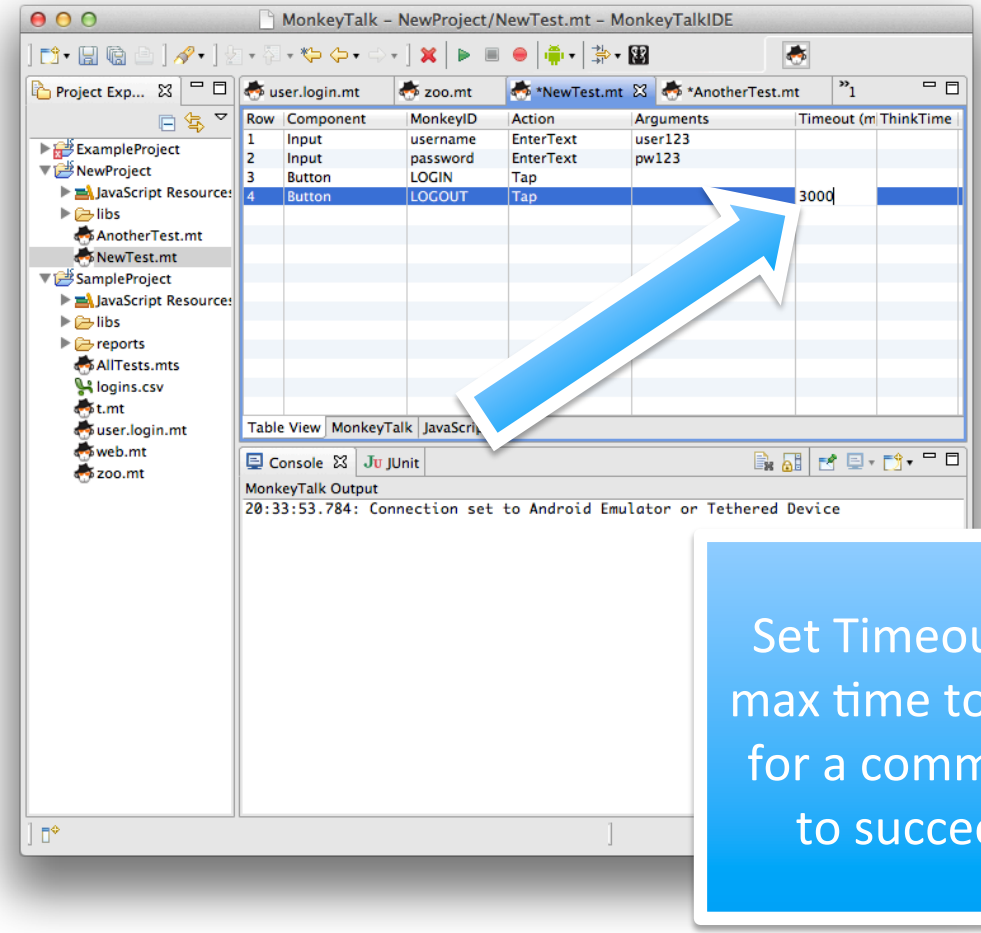
PLAYBACK RESULTS LOG
DISPLAYED IN CONSOLE VIEW





COMMANDS RETRIED BY DEFAULT

UNTIL SUCCESS OR TIMEOUT



Set Timeout to max time to wait for a command to succeed

- **Timeout** – Maximum time to retry the command before failing
  - Defaults to 2000 ms
- **ThinkTime** – How long to pause before executing the command
  - Defaults to 500 ms

VERIFY COMMANDS
COMPARE EXPECTED AND ACTUAL VALUES

```
ComponentType MonkeyId Verify expectedValue [propertyName] [failMsg]
```

- Several different Verify commands
  - Verify – Tests for a perfect match
  - VerifyWildcard – Tests against a wildcard expression
  - VerifyRegex – Tests against a regular expression
  - VerifyNot, VerifyNotWildcard and VerifyNotRegex test for a non-matching value
- The default propertyName is "value"

```
Slider Speed Verify 123 value
Label Greeting VerifyWildcard "Hello *"
Input ssn VerifyRegex \d{3]-\d{2}-\d{4} value "Invalid SSN"
```

LOGICAL COMPONENT PROPERTIES

VALID ACROSS PLATFORMS

- **size** – number of items in a list or other item selector
- **min**, **max** – mininun value for a slider or other numeric selectors

```
Table Countries Verify 196 size
Slider Speed Verify 200 max “Speed exceeds maximum”
```

- Platform-specific properties can be specified by prefixing the property name with a dot

```
# Test if the label’s text property equals “Banana”
Label Name Verify Banana .text
```

‘ITEMS’ ARRAY PROPERTY

REFERENCES TABLE ITEMS

- Specify the index (starting with 1) of the item as list reference in parentheses: *(n)*
- Optional second dimension denotes the column: *(row, col)*
- On iOS, second dimension can also refer to the table *section*

```
# See if the fourth item is “American Samoa”
Table * Verify “American Samoa” items(4)
# Test the first item of the second section
Table * Verify Blue items(1,2)
```

‘GET’ ACTION
RETRIEVES COMPONENT PROPERTY VALUES

```
ComponenType MonkeyId Get varName propName
```

- Works like an assignment statement (varName = propName)
- The default property is “value”

```
// Get label value currently displayed
Label lastname Get name
// Enter name into input field
Input query EnterText ${name}

// Get the first table item
Table countries Get country items[1]
// Enter into input field
Input * EnterText ${country}
```

ACTIONS CAN BE HIGH OR LOW LEVEL

UI EVENTS, GESTURES, OR OPERATIONS

Low-level UI Event Recording

TouchDown 5 5
TouchMove 5 6
TouchMove 5 7
TouchMove 5 8
TouchUp 5 9

Gesture Recording

Swipe Up

High-Level Operation Recording

Select "Chocolate Cake"

COMMON ACTIONS

MANY ACTIONS ARE COMMON ACROSS COMPONENT TYPES

- **Tap** – Tap on a component
- **EnterText** – Enter text into a field
- **Select** – Select an item from a list **by value**
- **SelectIndex** – Select an item from a list **by position**
- **On, Off** – Turn a switch on or off

```
Button OK Tap
Input “First Name” EnterText Fred
RadioButtons Gender Select Male
Table Countries SelectIndex 2
Toggle Enabled On
```

Actions are not case-sensitive (eg, tap, Tap, TAP are equivalent)

```
Button OK tap
Input “First Name” entertext Fred
```

POSITIONAL ARGUMENTS

SOME COMMANDS REQUIRE ONE OR MORE ARGUMENTS

- Arguments are space separated
  - Embedded blanks are quoted
- Some arguments are optional
- Use asterisk (*) as placeholder for default value
- For example, specifying "enter" for EnterText's second argument triggers an Enter key tap after text entry

```
Input name EnterText "Fred Mertz" enter
# Omit the second arg to enter text without Enter key
TextArea address EnterText "125 Main St"
```

- Timeout and ThinkTime are specified after any arguments, using the **%timeout** and **%thinktime** "command modifiers"

COMMAND MODIFIERS
THINKTIME AND TIMEOUT

- Specified with %name=value format
- Follow arguments (if any) on any command
- Two built-in modifiers
  - Thinktime
  - Timeout

```
# Retry until true or until 4 second timeout
Label message VerifyWildcard “Welcome *” %timeout=4000
#Pause two seconds before executing command
Button OK Tap %thinktime=2000
```

REUSING SCRIPTS

SCRIPTS CAN BE CALLED FROM OTHER SCRIPTS

```
Script ScriptName Run
```

- Scripts can call other scripts by filename
- MonkeyTalk looks for called scripts in the project's root folder
- MonkeyTalk scripts have an .mt file extension

```
# Call the login script
Script login.mt Run
# Do other stuff after login
Button OK Tap
```

PARAMETERIZING SCRIPTS

DEFINE SUBSTITUTION VARIABLES WITH THE 'VARS' ACTION

- Parameterize a script so you can run it with different values
  - For example, login and password values in a login script
- Scripts can pass positional arguments to called scripts
- Use the **Vars * Define** command to declare variable names, positions, and default values

```
Vars * Define var1=defaultValue1 var2=defaultValue2…

# Example
Vars * Define user=fred password=1234
```

WORKING WITH PARAMETERS

PASS VALUES TO CALLED SCRIPTS

- Substitution variables use $-curly format: ${varname}

```
# login.mt
Vars * Define user=fred password=pw123
Input username EnterText ${user}
Input password EnterText ${password}
Button OK Tap
Label message Verify “Welcome ${user}!
```

- Call ther Script

```
# Call with substitution values for user and password
Script login.mt Run ethel secret234
# Call with default values (fred pw123)
Script login.mt Run
```

DATA-DRIVING A SCRIPT

RUNNING A SCRIPT FOR EACH DATA ROW FROM A FILE

- Use the **RunWith** action to pass a CSV (comma-separated-values) file to a script
- First line of CSV must contain parameter names corresponding to the columns of data stored in the file
- Script is called once for each data line in the file

```
Script login.mt RunWith logins.csv
```

logins.csv

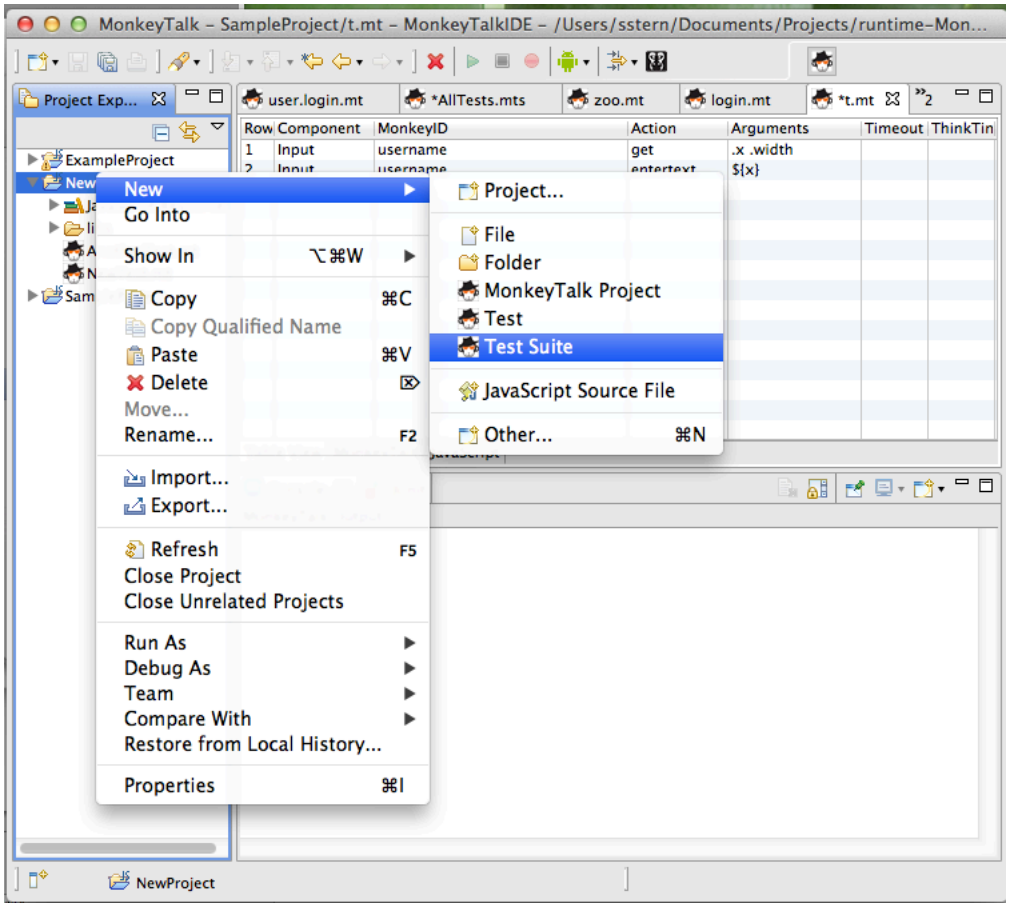
```
user, password
fred, pw123
Johnr, secrets
sally, m0nday
```

```
# login.mt
Vars * Define user=fred password=pw123
Input username EnterText ${user}
Input password EnterText ${password}
Button OK Tap
Label message Verify “Welcome ${user}!
```

TEST SUITES

RUNS MULTIPLE SCRIPTS WITH AGGREGATE RECORDING

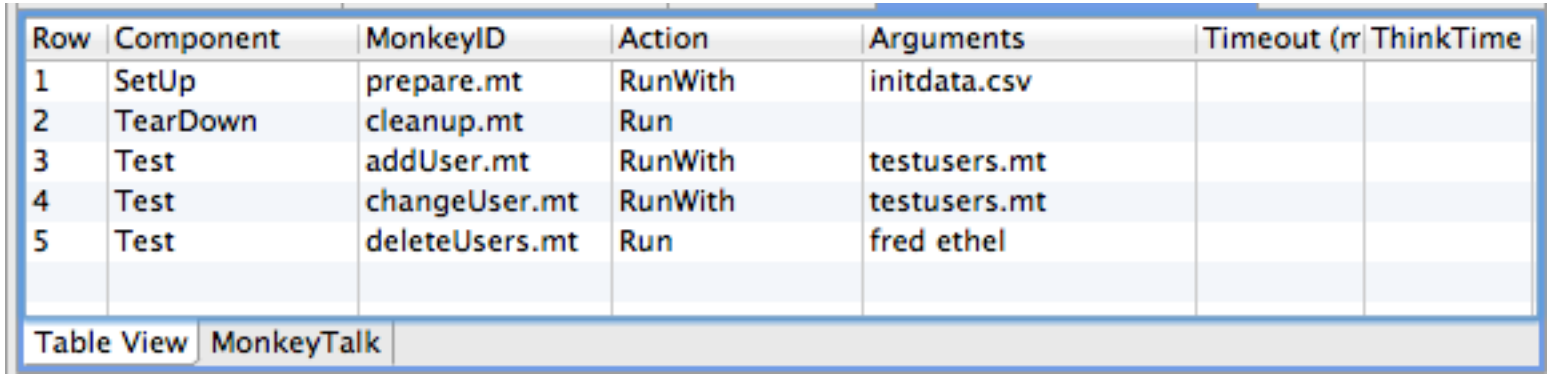
- Right-click on a project to create a test suite



SUITES CALL SCRIPTS OR OTHER SUITES

SUITES MAKE IT EASY TO RUN COLLECTIONS OF TESTS

- Suite files have an .mts extenstion
- Four types of commands permitted
  - Test – runs a script as a test
  - Suite – runs another suite
  - SetUp – a script to run before each test in the suite
  - TearDown – a script to run after each test in a suite
- Both **Run** (with args) and **RunWith** (with csv) supported



Row	Component	MonkeyID	Action	Arguments	Timeout (r	ThinkTime
1	SetUp	prepare.mt	RunWith	initdata.csv		
2	TearDown	cleanup.mt	Run			
3	Test	addUser.mt	RunWith	testusers.mt		
4	Test	changeUser.mt	RunWith	testusers.mt		
5	Test	deleteUsers.mt	Run	fred ethel		

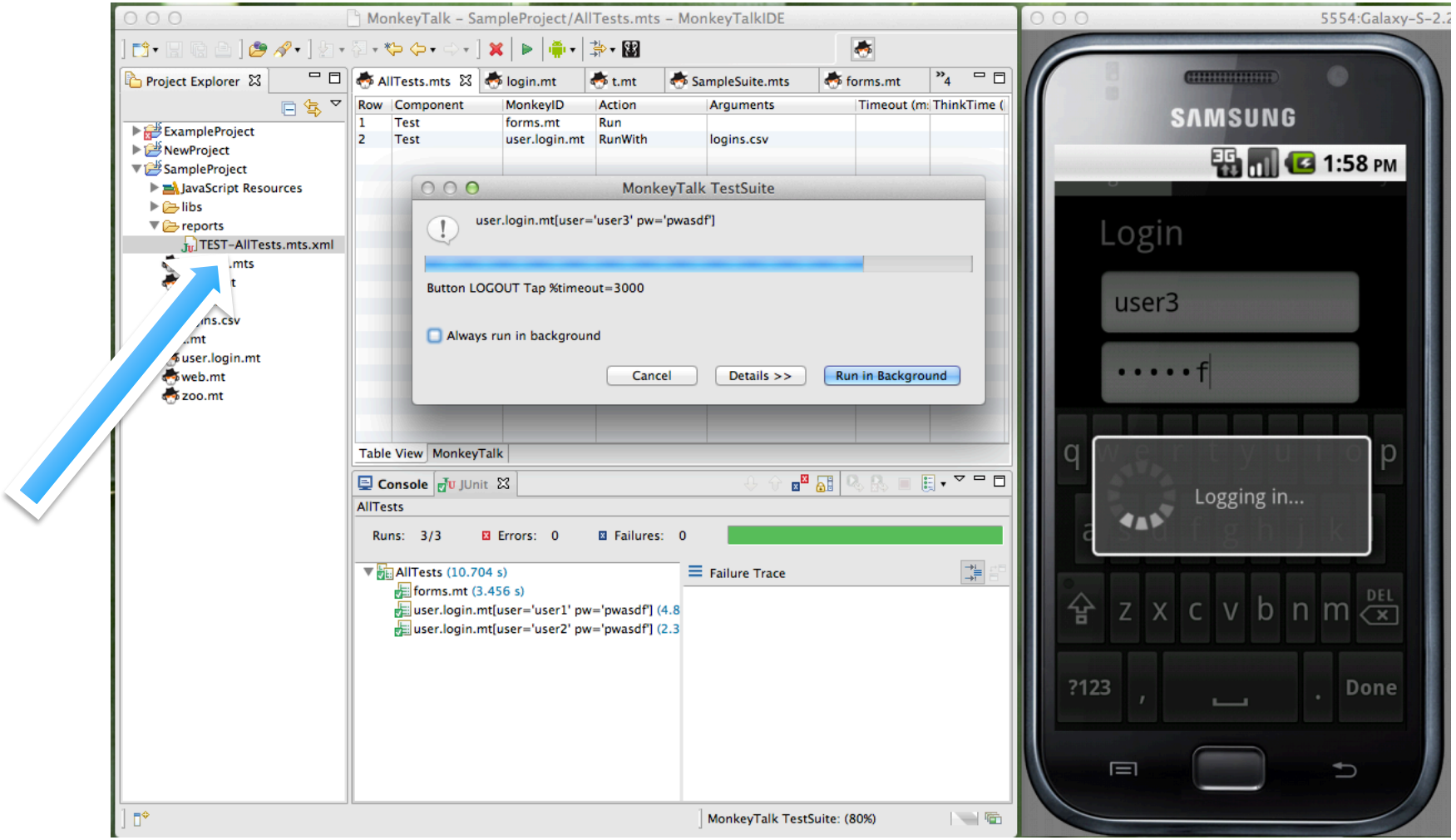
Table View | MonkeyTalk

- Suites can call other suites

```
Suite CoreTests.mts Run
Suite MoreTests.mts Run
```

SUITE RESULT REPORTING

DISPLAYED IN IDE AND SAVED IN REPORTS FOLDER

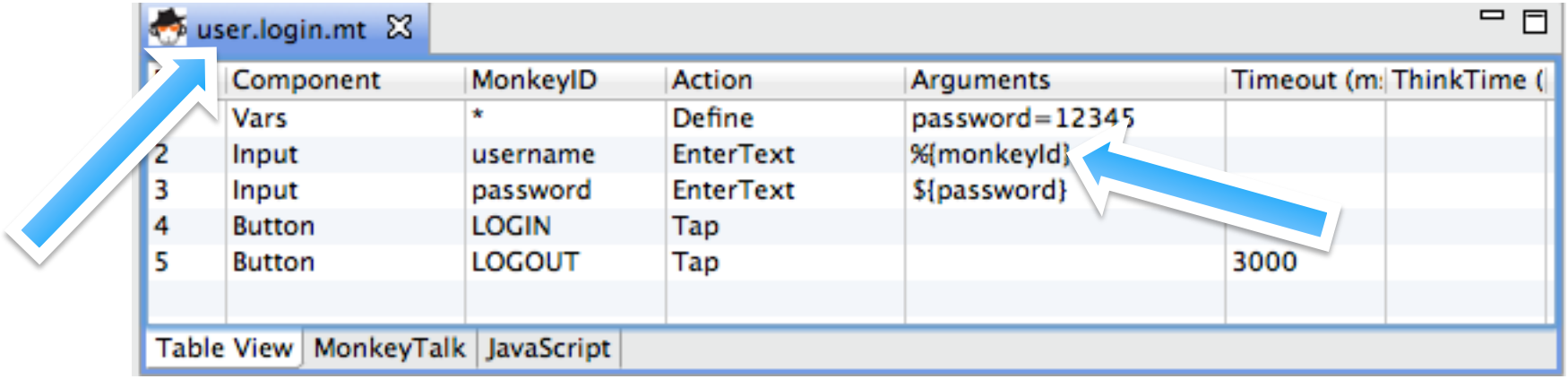


CUSTOM COMMANDS

MONKEYTALK CAN BE EXTENDED WITH NEW COMMANDS

```
CustomCmdName monkeyIdArg CustomActionName args…
```

- Define a custom command by saving a script with a 2-part name of the form **CmdName.ActionName**.mt
- Optionally use the %{monkeyId} built-in substituion variable to access monkeyId argument value



USING CUSTOM COMMANDS

CALL USING COMMAND OR RUN SYNTAX

- Command Syntax

```
User ethel Login secret123
```

- Use Script Syntax for data-driving (monkeyId set to script name)

```
Script user.login.mt RunWith logins.csv
```

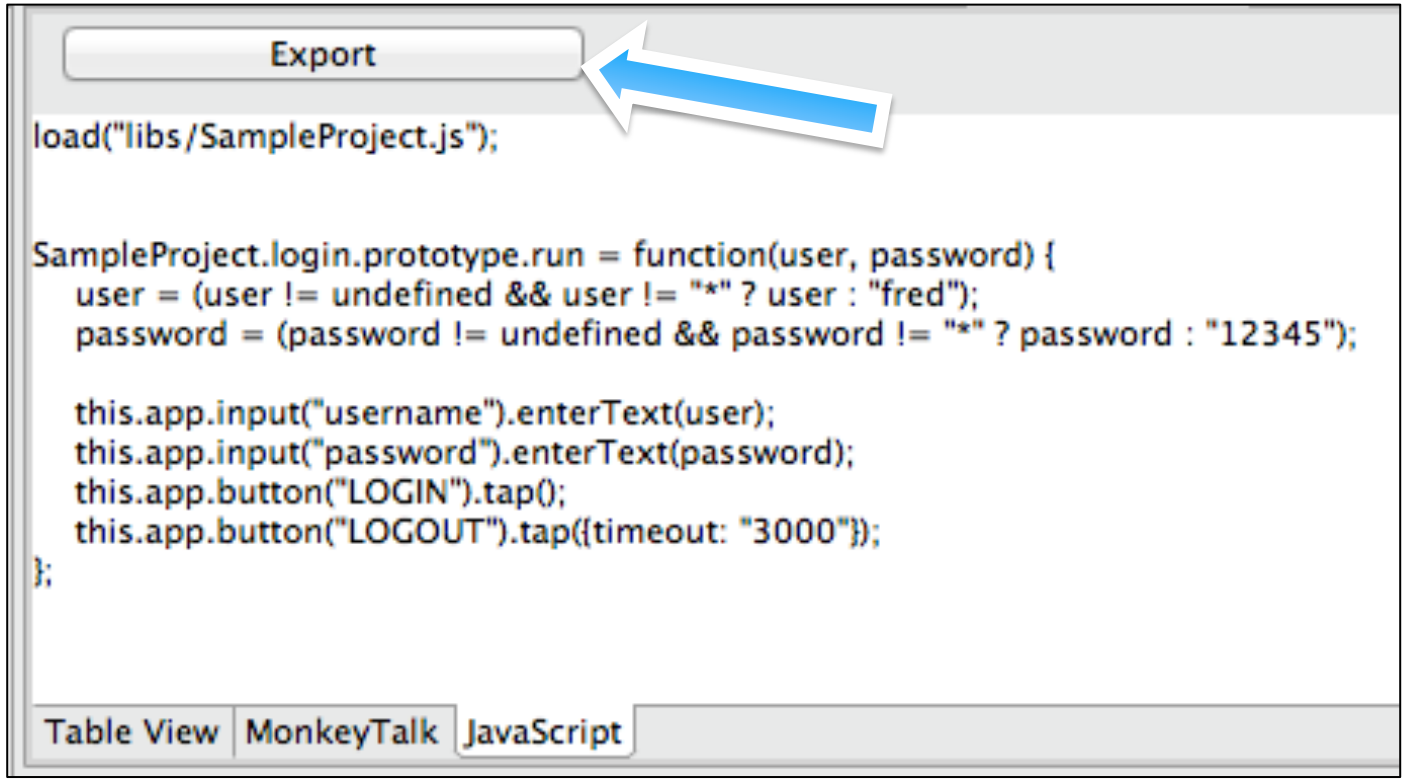
- Call Custom Commands From Suites by Invoking as Tests

```
Test user.login.mt Run ethel secret123
```

GENERATING JAVASCRIPT

USE MONKEYTALK JAVASCRIPT API FOR COMPLEX SCRIPT LOGIC

- The JavaScript Tab displays a JavaScript function generated from the MonkeyTalk Script
- Export button Opens JavaScript Editor



EDIT AND SAVING A SCRIPT

CLICKING THE EXPORT BUTTON OPENS JAVASCRIPT EDITOR

- Edit to add any required additional logic

CALLING JAVASCRIPT FROM MONKEYTALK

CALL JAVASCRIPT LIKE ANY OTHER MONKEYTALK SCRIPT

- Call .js Script Files in Same Way as .mt Scripts

Row	Component	MonkeyID	Action	Arguments	Timeout (ms)	ThinkTime (ms
1	Script	login.js	Run	ethel secret123		

- JavaScript Can Also Be Data-Driven

```
Script login.js RunWith logins.csv
```

JAVASCRIPT API
MIRRORS MONKEYTALK COMMAND SYNTAX

```
# Button OK Tap
this.app.button("OK").tap()

# Input firstname EnterText fred
this.app.input("firstname").enterText("fred");

# Label message Verify "Hello, fred"
this.app.label("message").verify("Hello, fred");

# Script login.mt Run ethel secret123
this.app.script("login.mt").run("ethel", "secret123");
# Script login.mt RunWith logins.csv
this.app.script("login.mt").runWith("logins.csv");

# Custom commands too
# User ethel Login secret123
this.app.user("ethel").login("secret123")
```

CUSTOM COMMANDS IN JAVASCRIPT

JAVASCRIPT CAN BE CALLED AS CUSTOM COMMAND

- Give JavaScript file a two-part, Command.Action.js name.

- Call user.login.js

```
User ethel Login secret123
```

OVERRIDING AN .MT SCRIPT

JAVASCRIPT OVERRIDES MT FILE WITH SAME NAME

- If no file extension specified, MonkeyTalk looks first for a .js file and if none is found, searches for an .mt file of the same name

```
# Assume both login.js and login.mt files exist

# Run user.login.js
Script login Run ethel secret123

# Run user.login.mt
Script login.mt Run ethel secret123

# Run user.login.js (assuming it exists, otherwise run
# user.login.mt)
User ethel Login secret123
```

HYBRID NATIVE/WEBVIEW APP TESTING

SCRIPT HTML COMPONENTS SAME AS NATIVE ONES

- MonkeyTalk commands also work with embedded web views
- Component Types mapped to HTML tags

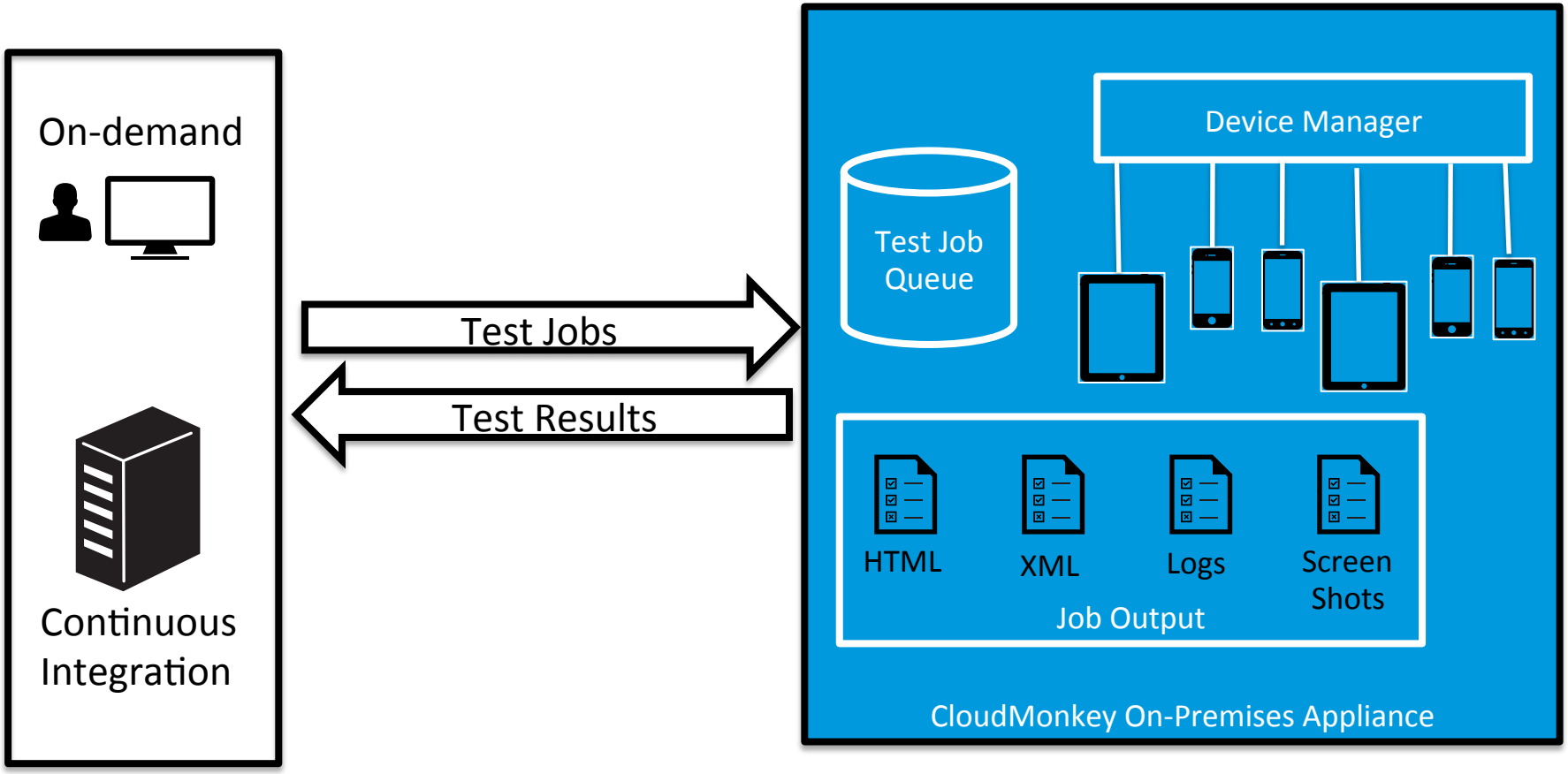
```
# Select native tab
TabBar * Select Web
# Tap HTML button
Button “Click Me” Tap
# Select HTML table cell
Table * SelectIndex 3 3
```

MOBILE WEB BROWSER APP TESTING

THE MONKEYTALK 'MTBROWSER'

- Use MTBrowser for mobile web app testing
  - Mobile Browsers linked with MonkeyTalk agents
- Supports Browser component actions
  - Browser * Open URL
  - Browser * Back
  - Browser * Forward

CLOUDMONKEY
"MOBILE DEVICE FARM IN A BOX"



"Device Farm in a Box"

GORILLA LOGIC

MONKEY TALK

STU STERN
STU.STERN@GORILLALOGIC.COM