INSTALLATION
Download the automation repository then run the following commands

```
// Run the install file to add Soda to your .bash_profile
$ cd path/to/Soda/
$ ./bin/install
```

BUILDING PROJECTS
You can use Soda to quickly build project binaries using the supported frameworks.

```
$ sodab [framework specific arguments] –f [framework]

// XCode & Instruments
$ sodab [app name] [path to workspace] [sdk] [build folder] -f instruments

XCode 6.x
$ sodab MyApp ~/path/to/workspace.xcworkspace iphonesimulator8.4 ~/path/to/build/to -f instruments

XCode 7.x
$ sodab MyApp ~/path/to/workspace.xcworkspace iphonesimulator9.0 ~/path/to/build/to -f instruments

// Android
// @todo Not currently supported
```

RUNNING TESTS
Soda is a **directory** based testing framework and organizes tests based upon folder hierarchy (see the DirectoryStructure.pdf file for more information). Therefore, in order to run Soda from the command line, you must either *cd* to your project folder or set the testing path flag (-t) to the path to your scripts directory.

There are two methods of running tests: using the interactive REPL (read, evaluate, print, loop) mode (e.g. *soda*) or the command line run mode (e.g. *sodar*).

The REPL allows you to develop tests by enabling “interactive mode.” Interactive mode allows you to pause, step through, and repeat test evaluations much like a debugger.

Run mode (sodar) runs tests, modules, or suites straight through without interruption.

To start the interactive test REPL (read, evaluate, print, loop) module…

```
$ soda [simulator, device name or id, or browser] [path to app] –f [framework]
```

Examples

```
// iOS
$ soda iPhone\ 6\ Plus\ 8.4 path/to/my/app -f instruments -t path/to/scripts
$ soda “iPad Air 8.4” path/to/my/app -f instruments -t path/to/scripts

// Android
// * Requires simulators already setup through ADM
$ soda Nexus_5_API_23 path/to/my/apk -f automator -t path/to/scripts

// Web
soda  http://starting.url -f selenium -x web -t path/to/scripts
```

To start a simple test run, without the interactive REPL enabled

```
$ sodar [simulator or device name or id] [path to app] [type] [name]… –f [framework]
```

Examples

```
$ sodar iPhone\ 6\ Plus\ 8.4 path/to/my/app -f instruments -t path/to/scripts --suite=[suite] --
module=[module] --test=[test]

$ sodar Nexus_5_API_23 path/to/apk -f automator -t path/to/scripts --suite=[suite] --
module=[module] --test=[test]

$ sodar  http://starting.url -f selenium -x web -t path/to/scripts --suite=[suite] --
module=[module] --test=[test]
```

USING THE INTERACTIVE REPL

The Soda REPL is a JavaScript *eval* loop. You can change various Soda settings and run tests directly from the REPL

To run tests using the REPL

```
Soda:PID:# > :r [type] [args]
```

Examples

```
Soda:PID:# > :r test my_test my_module my_suite my_platform
Soda:PID:# > :r module my_module my_suite my_platform
Soda:PID:# > :r suite my_suite my_platform
```

You can set defaults for each component (test, module, suite, platform) by executing

```
Soda:PID:# > :x myplatform
Soda:PID:# > :s mysuite
Soda:PID:# > :m mymodule
```

Then you can execute tests, modules, or suites without specifying *every* argument

Examples

```
Soda:PID:# > :r test my_test
Soda:PID:# > :r module my_module
Soda:PID:# > :r suite my_suite
Soda:PID:# > :r test
Soda:PID:# > :r module
... etc.
```

OTHER USEFUL REPL COMMANDS

```
// Prints the DOM tree for currently active screen
> :print

// Sets the variable `e` to the elements object so you can call
// e.withName(...), e.withSelector(...), etc.
> :e

// Prints and array of all buttons with the id “button:0”
> e.withId(“button:0”);


// Prints and array of all buttons with the selector “.userName”
> e.withSelector(“.userName”);

// Starts a framework
:fstart [framework name] [args]

// Stops the current framework
:fstop

// Restarts the current framework
:frestart

// Quits the program
:q

// Re-loads the suites folder
:load
```

AVAILABLE COMMAND LINE OPTIONS

```
-e  // Sets the testing environment, and the _env_ variable
-p  // Port for the VisualDOM
-x  // The testing platform (e.g. iphone, ipad, android, androidtab)
-d  // Log debug
-v  // Log verbose testing (shows the action JSON as it’s being evaluated)
-c  // Log colors
-t  // Testing path (path to suites)
-f  // Framework to launch
-s  // Sets the default suite
-m  // Sets the default module
-z  // Dev mode (Test/module/suite results won’t be logged)
```

OTHER USEFUL COMMANDS

```
// Switch between XCode versions
sudo xcode-select --switch /path/to/XCode/you/want/to/use
```