### What is Nightwatch?

Nightwatch.js is an automated testing framework for web applications and websites, written in [Node.js](http://nodejs.org/) and using the [Selenium WebDriver API](http://code.google.com/p/selenium/wiki/JsonWireProtocol).

It is a complete browser automation (End-to-End) solution which aims to simplify the process of setting up **Continous Integration**and writing of automated tests.

The current list of features can be found on Nightwatch website:

Simple but powerful syntax which enables you to write tests very quickly, using only JavaScript and CSS selectors. No need to initialize other objects and classes, you only need to write the test specs.

Built-in command-line test runner which enables you to run the tests either altogether, by group or single.

Manages the Selenium server automatically; can be disabled if Selenium runs on another machine.

Continuous Integration support: JUnit XML reporting is built-in so you can integrate your tests in your build process with systems such as Hudson or Teamcity.

Use CSS selectors or Xpath to locate and verify elements on the page or execute commands.

Easy to extend if you need to implement your own commands specific to your application.

Overview of Selenium

Selenium is a very popular and comprehensive set of tools for browser automation, initially written for Java but now with support for most programming languages.

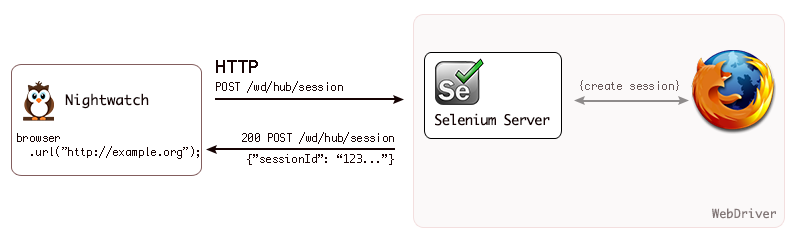
Selenium's main projects are:

* [Selenium IDE](http://docs.seleniumhq.org/projects/ide/)
* [Selenium Remote Control](http://docs.seleniumhq.org/projects/remote-control/)
* [Selenium WebDriver](http://docs.seleniumhq.org/projects/webdriver/)
* [Selenium Grid](http://docs.seleniumhq.org/projects/grid/)

Nightwatch uses the Selenium WebDriver, specifically the [WebDriver Wire Protocol](http://code.google.com/p/selenium/wiki/JsonWireProtocol) to perform the browser automation related tasks.

### Theory of Operation

Nightwatch works by sending HTTP requests to the Selenium server with the right parameters and interpreting the response. The restful API protocol is defined by the [Selenium JsonWireProtocol](http://code.google.com/p/selenium/wiki/JsonWireProtocol#/session). See below for an example workflow for browser initialization.



Most of the times, Nightwatch needs to send at least 2 requests to the Selenium server in order to perform a command or assetion, the first one being the request to locate an element given a CSS selector (or Xpath expression) and the next to perform the actual command/assertion on the given element.

## Installation

### Install Node.js

From [nodejs.org](http://nodejs.org/):

*"Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."*

There are installation packages and instructions for most major Operating systems on its website [nodejs.org](http://nodejs.org/). Remember to install also the **npm** tool, which is the node package manager and is distributed with the Node.js installer.

### Install Nightwatch

To install the latest version using the npm command line tool, run the following:

$ npm install nightwatch

To make nightwatch runner available globally in your system, add the **-g** option after npm install and sudo if you're on Mac OS X / Linux or other Unix based OS.

### Running the Selenium Server

The Selenium WebDriver server is simply a Java servlet which runs separately on the machine with the browser you want to test.

#### Download Selenium

Download the latest version of the selenium-server-standalone-{VERSION}.jar file from the [Selenium downloads page](http://selenium-release.storage.googleapis.com/index.html) and place it on the computer with the browser you want to test. In most cases this will be on your local machine and typically inside your project's source folder.

A good practice is to create a separate subfolder (e.g. bin) and place it there as you might have to download other driver binaries if you want to test multiple browsers.

#### Running Selenium Automatically

If the server is on the same machine where Nightwatch is running, it can be started/stopped directly by the [Nightwatch Test Runner](http://nightwatchjs.org/guide" \l "test-runner).

#### Running Selenium Manually

To run the selenium server manually, from the directory with the jar run the following:

$ java -jar selenium-server-standalone-{VERSION}.jar

More info about running the Selenium server can be found here:  
<http://code.google.com/p/selenium/wiki/RemoteWebDriverServer>

Configuration

The test runner expects a configuration file to be passed, using by default a nightwatch.json file from the current directory, if present. Let's create one in the project's root folder.

### Writing Tests

Using the preferred CSS selector model to locate elements on a page, Nightwatch makes it very easy to write automated End-to-End tests.

Create a separate folder for tests in your project, e.g.: tests. Each file inside it will be loaded as a test by the Nightwatch test runner. A basic test will look like this:

module.exports = {

'Demo test Google' : function (browser) {

browser

.url('http://www.google.com')

.waitForElementVisible('body', 1000)

.setValue('input[type=text]', 'nightwatch')

.waitForElementVisible('button[name=btnG]', 1000)

.click('button[name=btnG]')

.pause(1000)

.assert.containsText('#main', 'Night Watch')

.end();

}

};

Remember **always** to call the .end() method when you want to close your test, in order for the selenium session to be properly closed.

Running Tests

Test Runner

Nightwatch includes a command-line test runner which makes it easy to run tests and generate useful output.Example usage:

$ ./nightwatch --test tests/demotest.js

If you have installed nightwatch with-g (global) option you can skip this.

To use the test runner in your project simply create a new file called nightwatch and add the following:

For Linux and MacOSX:

#!/usr/bin/env node

require('nightwatch/bin/runner.js');

Then set the permissions:

$ chmod a+x nightwatch

For Windows:

Name the file nightwatch.js and add the following line:

require('nightwatch/bin/runner.js');

Then run as follows:

C:\workspace\project> node nightwatch.js

The test will do the following:

1. open up the given url and wait for body element to be visible
2. first client only:
   * wait for #localVideo element to become connected
   * wait for remote video element to become connected
   * after the remote video is connected, waits 1000ms
   * wait for the remote video element to be removed - i.e. the other peer has left the room
3. second client only - no assertions performed, only pauses for 10 seconds and then exits

### Using Grunt

[Grunt](http://gruntjs.com/) is a popular JavaScript task runner. Starting with version 0.6 Nightwatch is bundled with an easy to use Grunt task which can be used in your existing Grunt-based build configuration for running the tests.

#### Usage

First, load the Nightwatch grunt task at the top in your Gruntfile.js.

#### Task Configuration and Targets

The Nightwatch task will have one more more targets which can be used in various ways, one way being to map them to environments. Available settings are:

* options - the only available option so far is cwd - current working directory
* argv - command-line arguments that would normally be passed to the Nightwatch runner (e.g.: env);
* settings - the test settings specified to a single Nightwatch environment.